**ISTANBUL TECHNICAL UNIVERSITY**
**ELECTRICAL- ELECTRONICS FACULTY**

# Quadrator Flight Control Computer Design

**SENIOR DESIGN PROJECT**

**Eşrefhan Kadıoğlu**

**Barış Yılmaz**

**Ufuk Göktaş**

**ELECTRONICS AND COMMUNICATION ENGINEERING**
**DEPARTMENT**

**MAY 2023**

**ISTANBUL TECHNICAL UNIVERSITY**
**ELECTRICAL- ELECTRONICS FACULTY**

# Quadrator Flight Control Computer Design

## SENIOR DESIGN PROJECT

**Eşrefhan Kadıoğlu**
**(040170070)**

**Barış Yılmaz**
**(040160095)**

**Ufuk Göktaş**
**(040140036)**

**ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT**

**Project Advisor: Assist. Prof. Tankut AKGÜL**

**MAY 2023**

We are submitting the Senior Design Project Interim Report entitled as “Quadrator Flight Control Computer Design”. The Senior Design Project Interim Report has been prepared as to fulfill the relevant regulations of the Electronics and Communication Engineering Department of Istanbul Technical University. We hereby confirm that we have realized all stages of the Senior Design Project Interim Report by ourselves, and we have abided by the ethical rules with respect to academic and professional integrity.

**Eşrefhan Kadıoğlu**

040170070

**Barış Yılmaz**

040160095

**Ufuk Göktaş**

040140036

## FOREWORD

We are proud to have completed the graduation project design course, which is the compulsory course of the department. We would like to express our endless thanks to our mentor, Tankut Akgül, who provided us with all kinds of support and spared his time during this process, and to Istanbul Technical University and our professors who contributed to our research.

May 2023

Eşrefhan KADIOĞLU

Barış YILMAZ

Ufuk GÖKTAŞ

## TABLE of CONTENTS

**Page**

**FOREWORD ....................................................................................... iv**
**TABLE of CONTENTS................................................................................ v**
**ABBREVIATIONS ................................................................................... vi**
**LIST OF FIGURES................................................................................ vii**
**SUMMARY ...................................................................................... viii**
**1. INTRODUCTION ................................................................................... 1**
  1.1 The Project Interim Report ..................................................................... 1
  1.2 Project Work Plan and Possible Updates ........................................................ 2
  1.3 Current Progress on the Project................................................................ 2
**2. LITERATURE REVIEW .............................................................................. 5**
  2.1 Movement of Drone .............................................................................. 5
  2.1.1 Various Forces Act On The Moving Drone ....................................................... 7
  2.2 IMU Datas For Drones ........................................................................... 8
    2.2.1 MPU6050 sensor.............................................................................. 11
      2.2.1.1 Determination of the angular velocities and linear accelarations by using MPU6050 ....... 12
      2.2.1.2 Features of the MPU6050 module and gyroscope ........................................... 14
      2.2.1.3 Acquisition of measured data in MPU6050.................................................. 15
    2.2.2 Filtering MPU6050 sensor.................................................................... 16
      2.2.2.1 Kalman filter ............................................................................ 17
  2.3 Euler Angles And Quaternion Angles ........................................................... 18
    2.3.1 Euler angles ................................................................................ 19
    2.3.2 Quaternion angles............................................................................ 19
  2.4 Communication Systems Of I2C.................................................................. 20
    2.4.1 Bus Communication ........................................................................... 24
  2.5 Communication Systems Of UART ................................................................ 26
    2.5.1 Transmission................................................................................. 28
    2.5.2 Start bit .................................................................................... 28
    2.5.3 Data frame .................................................................................. 28
    2.5.4 Parity ...................................................................................... 28
    2.5.5 Stop bits.................................................................................... 29
    2.5.6 Steps of Uart Communication ................................................................. 29
  2.6 PID Controller For Drones ..................................................................... 30
**3. METHODOLOGY .................................................................................. 31**
  3.1 Main Control Flow of Software ................................................................. 33
  3.2 Reading IMU Values Via MPU6050................................................................ 33
    3.2.1 MPU6050 initialization function ............................................................. 34
    3.2.2 Reading meaningful values.................................................................... 35
      3.2.2.1 Angle calculation from reading values...................................................... 37
  3.3 Reading Remote Controller Values .............................................................. 41
    3.3.1 PPM, SBUS and PWM............................................................................ 41
    3.3.2 PPM decoding algorithm ...................................................................... 42
  3.4 PID Controller Implementation.................................................................. 45

3.4.1 Calculation of motor powers ................................................................45
3.5 Brushless Motor Speed Control....................................................................47
3.6 Error Handling ..........................................................................................53
3.6.1 IWDG (Independent watch dog timer)...................................................53
3.6.2 Interrupt errors ........................................................................................54
**4. REALISTIC CONSTRAINTS AND CONCLUSIONS ..................................55**
4.1 Practical Application of this Project .............................................................55
This study presents an application in the field of control and control of drone movement. Drones are aircraft used in various departments and provide advantages in areas such as construction, agriculture, logistics, security and film production...........................................................................................55
4.1.1 Social, environmental and economic impact ..........................................55
4.1.2 Cost analysis ............................................................................................56
4.1.3 Standards .................................................................................................56
4.1.4 Health and safety concerns......................................................................56
4.2 Results ............................................................................................................57
4.3 Future Work and Recommendations.............................................................60
**REFERENCES ...........................................................................................61**
**CURRICULUM VITAE ..............................................................................63**

## ABBREVIATIONS

**IMU** **:** Inertial Measurement Unit

**IDE** **:** Integrated Development Environment

**ARM** **:** Advanced RISC Machines

**GPIO** **:** General Purpose Input/Output

**M0** **:** ARM Cortex-M0

**I2C** **:** Inter-Integrated Circuit

**SPI** **:** Serial Peripheral Interface

**ADC** **:** Analog-to-Digital Converter

**IC.** **:** Integrated Circuit

**kHz** **:** Kilohertz

**Mbps** **:** Megabit per second

**PC** **:**Personal Computer

**PID** **:** Proportional-Integral-Derivative

**Gyro** **:** Gyroscope

**LQR** **:** Linear Quadratic Regulator

**SMC** **:** Sliding Mode Controller

**EKF** **:** Extended Kalman Filter

**PWM** **:** Pulse Width Modulation

**PPM** **:** Pulse Position Modulation

## LIST OF FIGURES

Figure 1.1:Image of STM32G071RB. ....................................................................2
Figure 1.2: Image of connectors pinout…..............................................................3
Figure 1.3: Cubeide interface to generate pinout configuration codes. It generates a file as .ioc. ..................................................................................................3
Figure 2.1: Controls of quadcopter..........................................................................6
Figure 2.2: Drone movements types [2]...................................................................6
Figure 2.3: Example an old IMU (Apollo IMU) [3].. ..............................................8
Figure 2.4: Newton's second law is demonstrated in two dimensions using a mass m attached to springs inside of a reference frame [4].. .................................................9
Figure 2.5: Illustration of a gyroscope's fundamental operation and the rotations that take place around each axes [4]...........................................................................10
Figure 2.6: MPU6050..............................................................................................44
Figure 2.7: The Coriolis force's mechanism [6].....................................................12
Figure 2.8: Internal components of the gyroscope: 1-fastening mass, 2-working weight, 3-fastening the inner frame, 4-sensor moving the inner frame, 5-inner frame, 6-substrate [6]. ......................................................................................................13
Figure 2.9: The gyroscope's construction when rotating[6]....................................14
Figure 2.10: Linear equations of Kalman filter [7]...............................................44
Figure 2.11: Covariance matrix of noise and measurement noise [7].. ....................18
Figure 2.12: States of Kalman filter equations [8]..................................................18
Figure 2.13: Pros and Cons of the different buses [11]..........................................22
Figure 2.14: I2C by the numbers [11].....................................................................23
Figure 2.15: Bus communication [11].. .................................................................25
Figure 2.16: Communication of two devices [15].. ...............................................27

Figure 2.17: Uart with data bus [15].. ..................................................................27
Figure 2.18: Attitude control PID control flow [14]................................................31
Figure 3.1: PCB design of our electronic card included the needed components.…..32
Figure 3.2: Main control flowchart of software structure... ....................................34
Figure 3.3: Schematic of GY-521 board in MPU6050. It is seen that the circuit is made suitable for I2C communication by pulling up inside itself [17]..…................35
Figure 3.4: MPU6050 datasheet offers options for digital low pass filter. DLPF_CFG dedicates the values must be written into the register which is stated in datasheet [17]. .........................................................................................................35
Figure 3.5: Flowchart of reading values from IMU.. ..............................................36
Figure 3.6: Roll around the X axis.. .......................................................................38
Figure 3.7: Pitch around th Y axis. ........................................................................38
Figure 3.8: Kalman Filtering of roll angle at the trajectory moment.. ......................40
Figure 3.9: Kalman filtering of roll angle at the vibration moment through 2 seconds......................................................................................................................41
Figure 3.10: The relationship between PPM and PWM coding.. .............................42
Figure 3.11: PPM Decoding we implemented. .......................................................44
Figure 3.12: PID controller block schema.. ............................................................45
Figure 3.13: Motor powers control loop.................................................................46

## SUMMARY

This article presents a project report on the development of a drone control system. The report begins with an introduction to the project, highlighting its objectives and progress. A comprehensive literature review is conducted, covering topics such as drone movement, IMU sensor data, Euler angles, quaternion angles, communication systems (I2C and UART), and PID controllers for drones.

The methodology section outlines the main control flow of the software and explains the process of reading IMU values through the MPU6050 sensor. It also discusses the decoding of remote controller values and the implementation of a PID controller for motor power calculation. Additionally, brushless motor speed control and error handling mechanisms are addressed.

The report concludes with a discussion of realistic constraints, including social, environmental, and economic impacts, cost analysis, standards, and health and safety concerns. The practical application of the project is examined, and recommendations for future work are provided.

Overall, this report serves as a valuable resource for understanding the development and implementation of a drone control system, offering insights into various aspects of drone technology and its potential real-world implications.

# 1. INTRODUCTION

Quadcopters, also known as drones, have become an increasingly popular and versatile technology in recent years, with applications ranging from military operations and search and rescue to delivery services and aerial photography. A key component of quadcopters is the Inertial Measurement Unit (IMU), which is responsible for measuring and providing feedback about the aircraft's orientation, angular velocity, and acceleration. In this project, we will be using the MPU6050 IMU to provide these measurements and enable our quadcopter to maintain stability and control.

The MPU6050 is a widely used IMU that combines a 3-axis gyroscope and a 3-axis accelerometer in a single package. It is capable of measuring both rotational and linear movement and is able to filter out external noise to provide accurate and reliable data. By using the MPU6050 IMU, our quadcopter will be able to sense and respond to changes in its environment, allowing it to maintain a stable flight.

In this project, we will delve into the software design and implementation of our quadcopter, including the use of the MPU6050 IMU for measurement and control. We will also explore the challenges and considerations involved in developing quadcopter software, including issues of reliability, safety, and performance. By understanding the software design and capabilities of our quadcopter, we can gain insights into the potential and limitations of this technology and how it can be applied in a variety of applications.

## 1.1 The Project Interim Report

We decided that we read about the articles which we dedicated in Form-3. We understood the articles and took the informations which will be used. We have been searched that taking the real values from IMU(Inertial Measurement Unit) and filtering. Additionaly, we learned how drones move physically and take commands from a microcontroller and how microcontrollers decide and process that progress. We have used quaternion angles to understand the movement of the drone relative to the ground. Every research and necessary thing which we noted in Form-3 is written with details in that paper. We started to apply theese methods also to the code that we will present at the end of the year. We also tried to explain the methods of theese things searched in that paper.

## 1.2 Project Work Plan and Possible Updates

There is no important changing in our plans. We have had in trouble with microcontroller that we used but, we handle with that.

## 1.3 Current Progress on the Project

So far, we have completed our literature research. We have examined the dynamic movements of a quadcopter under the headings of how it will be and its avionics part. We completed the drone body and avionics equipment with the help of our adviser. We tried some variants of stm company as a microcontroller and MPU6050 integrated circuit to be used for IMU, which is one of the necessary equipment for avionics software. We decided to continue with the Stm32g071RBT6 microcontroller. Its technical features are given in Figure 1.1 and the visual regarding the pin inputs and outputs in Figure 1.2.

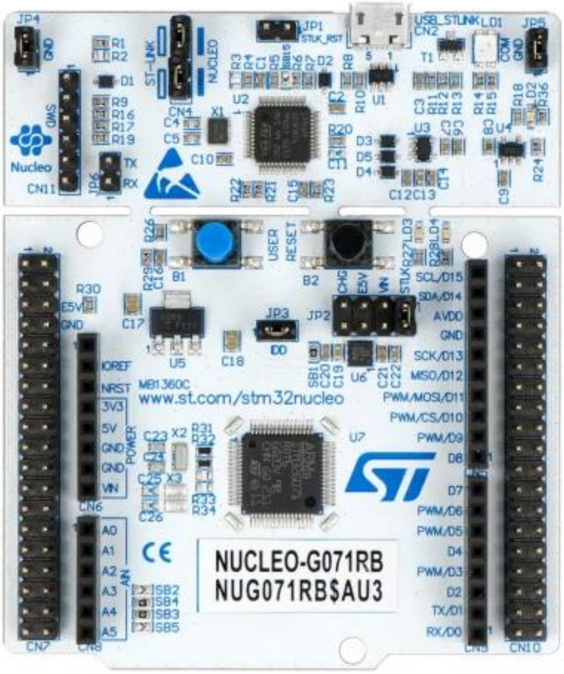


**Figure 1.1:** Image of STM32G071RB.

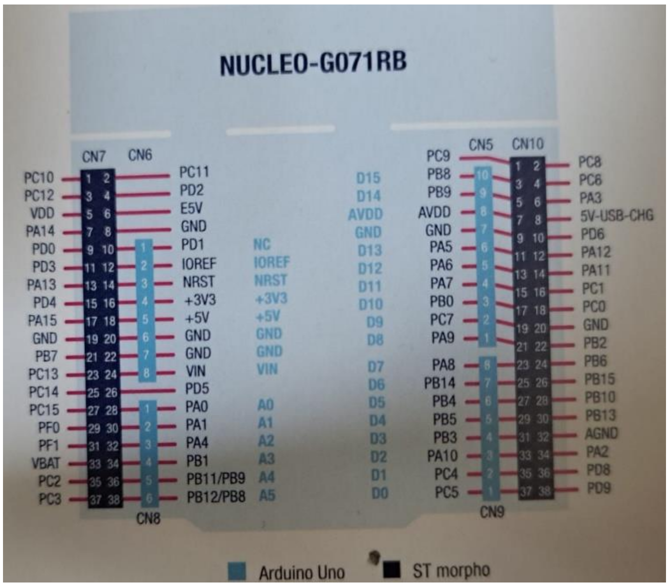


**Figure 1.2:** Image of connectors pinout.

Microcontroller uses arm M0 Cortex. As a side effect of this, the IDE we use, CUBEIDE M0 Cortex, does not support communication with the terminal. Thanks to the IDE we use, we can automatically adjust the communication configuration settings of the input/output and devices as in Figure 1.3. In addition, thanks to the libraries it provides, we can easily exchange data from these pins. At the current stage, we could not perform a data analysis because we could not adjust the hardware of the equipment we have.

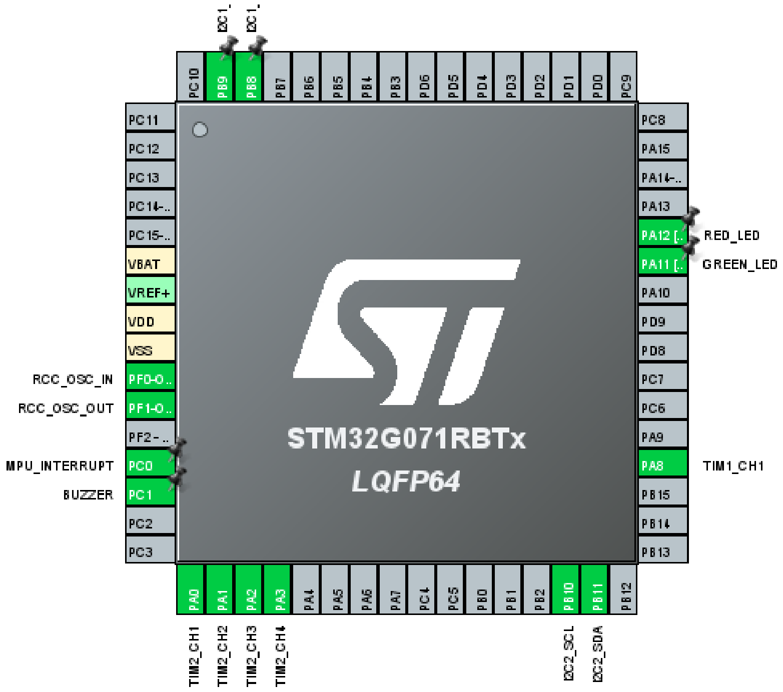


**Figure 1.3:** Cubeide interface to generate pinout configuration codes. It generates a file as .ioc.

In the next process, we will complete the hardware, take the IMU data, perform data analysis and select the appropriate filter. Then, using the PID controller, it will be possible to transmit the data to the microcontroller and power the motors with the necessary calculations.

# 2. LITERATURE REVIEW

## 2.1 Movement of Drone

Quad-rotors are a type of drone that is characterized by its use of four rotors to generate lift and control its movement in the air. Quad-rotors are particularly well-suited for aerial robotics applications due to their high maneuverability and agility, which allows them to perform a wide range of movements and maneuvers. To move, a drone typically has four or more motors that spin rotors or propellers. The rotors provide lift, which allows the drone to take off and hover in the air, and they also provide thrust, which allows the drone to move forward, backward, left, and right. The motors and rotors on a drone are controlled by a flight controller, which is a small computer that runs software that receives input from various sensors (such as a gyroscope and accelerometer) and calculates the appropriate commands to send to the motors. The flight controller receives input from the human operator (if the drone is being controlled remotely) or from the drone's autopilot system (if the drone is flying autonomously).

Throttle/Hover: Throttle is the name for the drone's up-and-down motion. The drone will descend if all four propellers spin at their normal pace. The drone will rise if all four propellers are spinning more quickly. Drone hovering is what is happening here. Pitch is the up-and-down movement of the quad-nose, rotor's which is normally accomplished by regulating the thrust of the rotors on the quad-front rotor's and rear. The drone will go ahead if its two back propellers spin quickly. The drone will go backward if the two front propellers spin quickly. Yaw, on the other hand, refers to the left and right movement of the quad-rotor's nose, which is typically achieved by adjusting the thrust of the rotors on the left and right sides of the quad-rotor. The drone will revolve counterclockwise if two right diagonal propellers are spinning quickly. The drone will rotate clockwise if the left diagonal's two propellers spin swiftly. By changing the thrust of the rotors on the top and bottom of the quad-rotor, roll is the tilt of the quad-body rotor's around its longitudinal axis. The drone will turn left if two right propellers spin quickly. The drone will go in the appropriate direction if its two left propellers spin quickly [1]. It is shown in Figure 2.1 with an image.

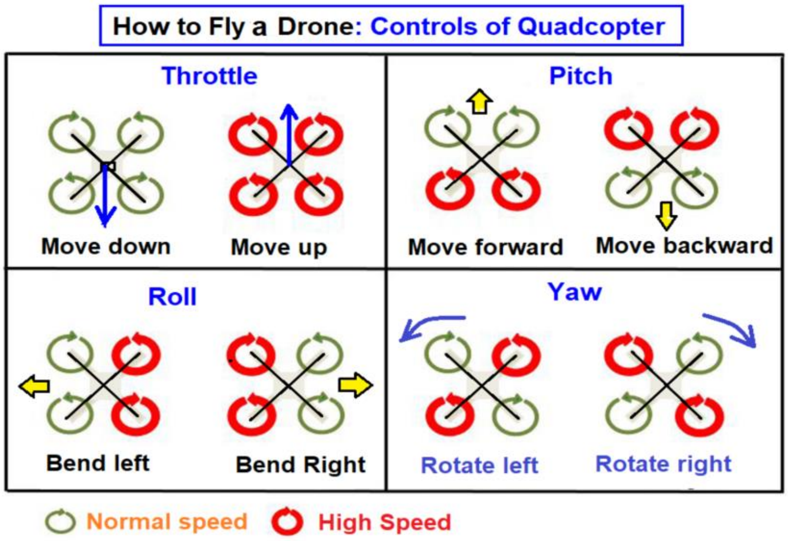


**Figure 2.1:** Controls of quadcopter.

By carefully adjusting the thrust and attitude of the rotors, it is possible to control the quad-rotor's pitch, yaw, and roll, as well as its vertical and lateral movement. A similar example of theese axis is dedicated in Figure 2.2.

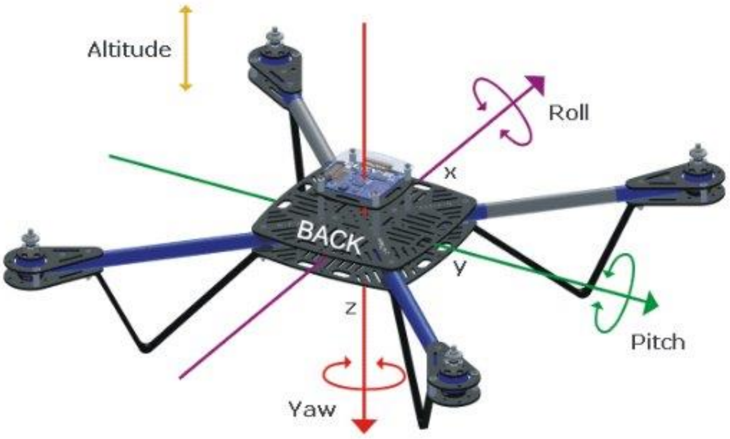


**Figure 2.2:** Drone movements types [2].

To control the movement of the drone, the flight controller sends commands to the motors to change their speed and direction. For example, to make the drone go up, the flight controller might send a command to increase the speed of the rotors that are lifting the front of the drone. To make the drone turn left, the flight controller might send a command to increase the speed of the rotors on the left side of the drone and decrease the speed of the rotors on the right side.

One of the key challenges in controlling the movement of quad-rotors is ensuring their stability and robustness in the face of external disturbances, such as wind gusts and changes in air density To address this challenge, researchers and engineers have developed a variety of control algorithms and techniques that use sensors and real-time feedback to maintain the quad-rotor's stability and trajectory. The control algorithm developed for quad-rotor will be discussed in the next sections. 2.1.1 Various forces act on a moving drone in the air.

### 2.1.1 Various Forces Act On The Moving Drone

The resulting force will determine its movement. There are powerful forces at work on a drone. Drone operation theory and forces acting on drone. There are four types of forces which acts drones movement such as weight, lift, thrust and drag.

Because of the mass of the drone, the body mass force always acts in the direction of gravity. More power is required to lift and move the drone as its weight increases.

Equation: Weight of drone = mass of drone + gravity acceleration.

Lift is the vertical force acting on the drone.This force is caused by pressure variances throughout the drone (in the vertical direction). As a result, the amount of lift force is determined by the speed, size, and shape of the propeller blade. Lift is required to lift the body against gravity. To generate this force, the drone's four propellors spin at high speeds.

Thrust is the force acting on the drone in the direction of motion. However, for drone dynamics, the rotor plane is normal. The push is entirely vertical while hovering.

If the thrust angle is slanted, the drone will lean forward or backward. This force is required to propel the drone in the desired direction at the same speed. Two propellors were given high speeds to provide the needed velocity.

Drag is the force acting on the drone in the opposite direction of motion caused by air resistance. This could be due to pressure differences and air viscosity. The aerodynamic form of the drone is chosen to minimise drag.

## 2.2 IMU Datas For Drones

Inertial measurement units (IMUs) are sensors that are used to measure linear acceleration and angular velocity. These sensors are essential in a variety of applications, including robotics, aviation, and space exploration, as they allow for the accurate measurement and control of motion.
In the past as it seems in Figure 2.3 IMU was physically big and used to measure the oriantation of an aircraft and also it is used in Apollo 11 lunar module. Nowadays IMU is an electronic device that can be fit into very small sizes.

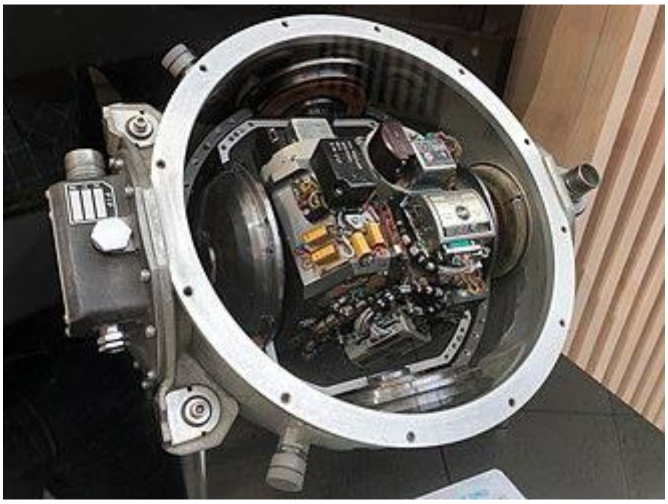


**Figure 2.3:** Example an old IMU (Apollo IMU) [3].

Accelerometer and gyroscope sensors are typically found in IMU sensors. IMU is often used to modify the position of the body when sensors are mounted or to recreate a stance. IMU are used to measure speed, position, and orientation by integrating signals from sensors across time. as a result of the two sensors (accelerometer and gyroscopes). Sensor biases and sound variation lead to early estimates that become incorrect within a short period of time. A typical IMU contains three accelerometers and orthogonal rate gyroscopes that measure linear angular velocity and acceleration in three dimensions. An object's location and orientation in reference to its own starting point, velocity, and orientation are tracked using accelerometers and gyroscopes. Recent advancements in MEMS technology have made it feasible to produce smaller, lighter sensors while simultaneously capturing human motion and equipment direction [4].

The act of accelerating over a period of time generates a state of velocity. Acceleration involves the gradual increase in speed over time. On the other hand, deceleration refers to a slower speed compared to the previous pace. Additionally, acceleration is influenced by the direction or orientation due to its connection to vector quality. Furthermore, acceleration occurs when objects are in motion or change their direction. To obtain distance data from the accelerometer sensor, a dual integral procedure is required for the output sensor. One potential application of accelerometer sensors is their utilization in detecting movement, such as foot motions for navigation or movements for console games and other controllers.

### Basic principle of accelerometer

For the sake of simplicity, a MEMS accelerometer can be seen of as essentially operating according to Newton's second law when a mass is connected to a spring within a reference frame(Figure 2.4). The two primary functions of MEMS accelerometers are to measure the displacement of a mass and the frequency of a vibrating element whose mass is changing due to a change in tension. By integrating the signal twice and using the accelerometer's linear acceleration measurement, we can determine the position [4].

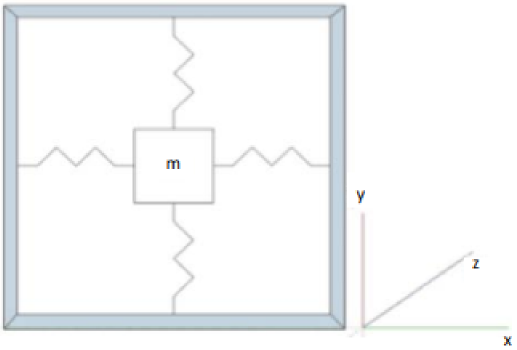


**Figure 2.4:** Newton's second law is demonstrated in two dimensions using a mass m attached to springs inside of a reference frame [4].

### Gyroscopes

Gyroscopes are devices that are fixed on frames and can sense angular velocity when the frame is rotated. Gyroscope classifications vary depending on the technology and physical operation they are used in Figure 2.5 [4].

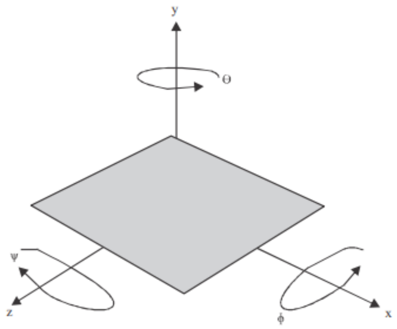


**Figure 2.5**

**Figure 2.5:** Illustration of a gyroscope's fundamental operation and the rotations that take place around each axes [4].

The Coriolis effect may be observed using a MEMS gyroscope with vibrating masses that move along a driving axis. The perpendicular sense axis produces a secondary vibration when the gyroscope is turned, which causes the mass to move from its initial orientation. The rotation around specific axes is one of a gyroscope's operating principles. To recognize these displacements, the gyroscope introduces variations in capacitance. This allows us to estimate the IMU's angular velocity, and by integrating the data, we can determine orientation [4].

As it is understood, IMU has complex structure. However IMU has become more accessible and cheap integrated circuits at micro level.

### Coriolis effect

The Coriolis effect is a phenomena that happens when a moving object is observed in a rotating reference frame. It is named after French mathematician and scientist Gaspard-Gustave de Coriolis. The rotation of the Earth and its impact on objects in the atmosphere and on the surface of the Earth are the most frequent associations with it.

The Coriolis effect's impact on the direction of wind patterns and ocean currents is among its most well-known applications. The Coriolis effect causes wind and water to be diverted from their intended paths in the Northern Hemisphere to the left, and in the Southern Hemisphere to the right. This is due to the fact that when the Earth rotates, objects traveling across its surface are dragged away from the planet's center, giving their paths a sensation of curvature [5].

### 2.2.1 MPU6050 sensor

MPU6050 is one of the example for IMUs as integradted circuit as cheap and accessible easy. In that project MPU6050 sensor is used as IMU to provide that the drone maintains a stable flight by calculating situations of the quadcopter relative to the ground. In Figure 2.6 shows that how its shape.

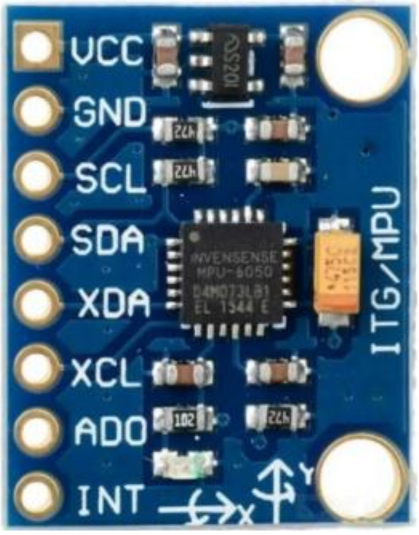


**Figure 2.6:** MPU6050.

As it seems in that integrated circuit there are some pins. VCC is for supplying power to the IC(Integrated Circuit). GND is ground. SCL(Serial Clock Pin) and SDA(Serial Data Pin) is for communication for I2C( It is explained in the 2.4 section). XDA is the external I2C data line. XCL is the external I2C clock line. ADO is used to modify the MPU6050 module's I2C address. It can be used to link two MPU6050s to the same I2C bus or prevent conflicts between the module and other I2C devices. INT is the interrupt output pin.

The MPU6050 is a microelectromechanical systems (MEMS) device that combines a 3-axis gyroscope and a 3-axis accelerometer on a single integrated circuit (IC). It is widely used in a variety of applications, including motion tracking, gesture recognition, and vibration analysis. One of the key advantages of the MPU6050 is its small size and low power consumption, making it suitable for use in portable and battery-powered devices. It also has a high measurement range and high sensitivity, making it capable of detecting small movements and vibrations. It also includes a built-in temperature sensor and on-chip digital motion processor (DMP) for processing complex movements and gestures.

In drones, the MPU6050 is typically mounted on the flight controller board, which is the central processing unit of the drone responsible for controlling its movement and stability. The MPU6050 provides the flight controller with real-time data on the drone's orientation and movement, which is used to calculate the necessary control signals for the motors and stabilize the drone using. As we said above its 3-axis gyroscope is particularly useful for detecting rotational movements and changes in orientation, while the 3-axis accelerometer can be used to detect linear acceleration and deceleration. This data can be used to determine the drone's position and movement in 3D space, allowing it to maintain a stable hover or navigate through its environment. In addition to its use in drones, the MPU6050 has also been used in a variety of other applications related to motion tracking and gesture recognition.

#### 2.2.1.1 Determination of the angular velocities and linear accelarations by using MPU6050

In that part we will concider how MPU6050 work and communicate with another circuit to give the information about angular velocity and linear accelaration measurements. MEMS (Microelectromechanical Systems) devices are made using the same process as single-chip integrated circuits, which involves etching and patterning layers of material onto a silicon substrate. This allows them to be fabricated at a small scale, with sizes ranging from a few tens of microns to several millimeters. As the other MEMS use Coriolis’ force, this one is also using that principle. There is an active body that is moving back and forth. The Coriolis force acts at a right angle to both the axis of rotation and the movement direction of the object, will start to act on it once the substrate on which this body is positioned is rotated. Figure 2.7 provides an illustration to help in understanding this force's fundamentals.

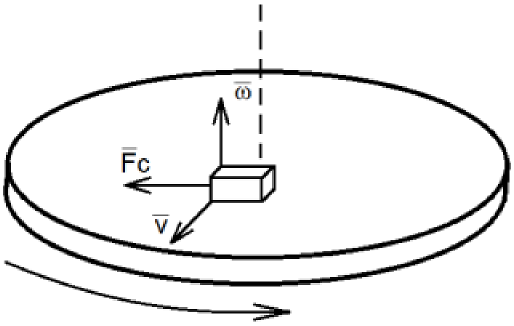


**Figure 2.7:** The Coriolis force's mechanism [6].

**W**: Angular velocity vector, **V:** Velocity vector, **Fc:** The Coriolis' force

Knowing the linear velocity and Coriolis force allows one to determine the angular velocity. The following structure describes one potential use for a gyroscope: a structure with flexible hangers inside of which a mass oscillates and translations [6]. It can be seen in Figure 2.8.

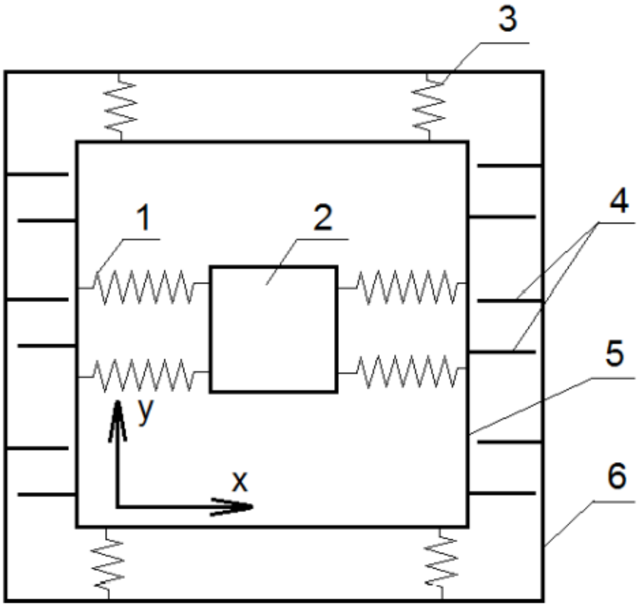


**Figure 2.8:** Internal components of the gyroscope: 1-fastening mass, 2-working weight, 3-fastening the inner frame, 4-sensor moving the inner frame, 5-inner frame, 6-substrate [6].

Electrostatic forces cause the working mass to oscillate along the X axis, and internal frame oscillations are only conceivable along the Y axis. It is feasible to stabilize the frame's movement in relation to the substrate by placing flat capacitor plates (displacement sensors) between the inner frame and the fake frame. However, in addition to the Carioles force, linear accelerations acting along the Y axis can also generate oscillations of the inner frame. Two frames with the operational mass in each are placed on a single substrate to solve the issue. Since the two masses oscillate in opposition to one another, the force of the Carioles acting on the first mass is at a given instant opposite to the force operating on the second mass. The signals produced by the force of the Carioles will be combined, and the linear acceleration-produced in-phase component will be eliminated [6]. The MPU 6050's technical details and its integrated gyroscope are available in Figure 2.9.

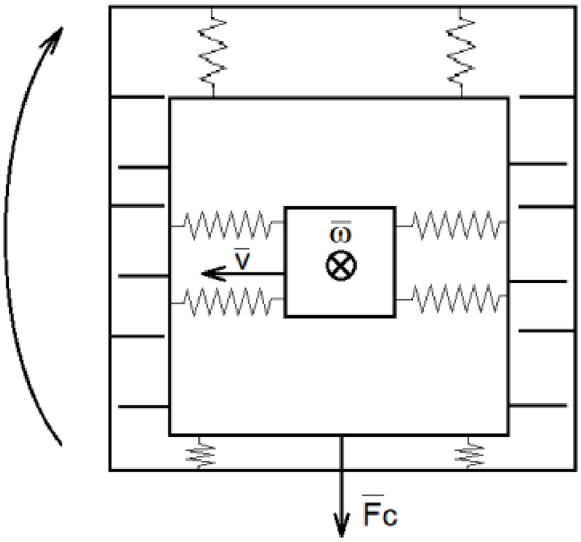


**Figure 2.9:** The gyroscope's construction when rotating[6]. W: vector of angular velocity, V: Vector of velocity, Fc: The Coriolis' force.

### 2.2.1.2 Features of the MPU6050 module and gyroscope

The general features of the model and gyroscope is showed below.

**Tecnhnical features of the module**

- 3-axis gyroscope;
- 3-axis accelerometer;
- Temperature sensor;
- Power supply ranging from 2.375 to 3.46 volts;
- 1024-byte FIFO buffer;

- Gyroscope, accelerometer, and temperature sensor digital filters with user-configurable settings;
- • A 400 kHz I2C interface for writing and reading device registers, an accelerometer, a temperature sensor, and other sensors.

#### 2.2.1.3 Acquisition of measured data in MPU6050

In that part we tried to explain how that process happens by using some articles. In analyise section there will be our results.

The MPU 6050 module may send commands and read the necessary data from registers by being connected to the microcontroller's I2C module. After providing the module the order to begin measurements, the values from all axes of the gyroscope, accelerometer, and thermal sensor are continually digitalized. The last step is to read the necessary registers' bytes. The frequency at which an analog-to-digital converter stores fresh data in these registers depends on the sensitivity of the user-selected sensor and, therefore, the measurement range. As previously discussed, the MPU6050's construction has an I2C interface for connecting to the microcontroller. These characteristics apply to this: Because of its master-slave architecture, only the master device may ask to read or write data. The MPU6050 will serve as the slave in that project, while the microcontroller will house the master I2C module. uses two bidirectional lines to transmit data and keep track of time. This line must have a distinct address for the slave device. There must be a dragging of lines to the level of logical units. Devices are often connected to lines via open collector (drain) pins. Actually, the "installation AND" approach is used to connect the devices. The ability to connect a sizable number of slave devices and a respectably long data transmission distance are advantages. Yet when the range extends, the actual data transfer rate decreases. The smoothness of the fronts improves as the wires' capacity rises.. There are a few factors that must be considered when using this approach. The MPU 6050 module won't be able to move on to the next phase of operation if this isn't done, and the program will stall. The second aspect is the way data reading from the MPU 6050 registers closes a communication channel. The application of a stop condition to the data line is the data completion signal. In our situation, the command to establish a stop condition for it must be executed prior to reading the last byte. even if only one byte has been sent, when the slave confirms that the address has been acknowledged.

After reading the last byte, if this is done, the microcontroller will wait for an additional byte that won't come, causing the program to stall. It is easy to expand the procedure shown in the flowchart, which is derived from a research study, to read or write multiple bytes. Data may be read from or written to many registers in succession during a single session thanks to the MPU 6050's automatic one-byte address increment after each read or write [6].

Up to this point, research has been used to obtain and transfer data at the register level from the mpu. The theoretical filtering of this data will be covered in the following chapter.

### 2.2.2 Filtering MPU6050 sensor

Digital filters are algorithms that handle digital data and carry out signal processing operations like noise reduction, sharpening, and smearing. There are many different kinds of digital filters, including low-pass filters, which are frequently preferred. A low-pass filter is typically more appropriate for smoothing the output data and removing high-frequency noise in the context of the MPU6050 sensor. This is because a low-pass filter can help to lessen the impact of high-frequency noise on the recorded data. The MPU6050 is primarily used to measure low-frequency motion and vibration. In addition, the kalman filter is a popular choice for filtering data from the MPU6050 sensor and other types of motion sensors. The Kalman filter is a state-space model that estimates the underlying state of a system based on a series of noisy measurements. It has the ability to fuse multiple types of data, such as accelerometer and gyroscope data, to provide more accurate estimates of the system's state. The MPU6050 sensor's data can be filtered using the Kalman filter because of its many appealing features. It can handle non-linear systems and adjust over time to changing circumstances. Additionally, it has a reasonable level of computing efficiency, which is crucial for real-time applications. In general, the Kalman filter is an effective tool for filtering MPU6050 data and can frequently perform better than other types of filters. When determining whether to use the Kalman filter or another form of filter, it is crucial to carefully assess the unique requirements of the application and take into account other aspects, such as the complexity of the implementation and the necessary processing resources. In that process we will concider about kalman filter. Then in analysis part will be concidered the results of theese filters.

### 2.2.2.1 Kalman filter

In general, the Kalman Filter is a linear estimator with the ability to reduce error covariance. This equation was written as a recursive calculation loop, which a microprocessor can fairly simply implement. Current input and state-dependent Kalman filter output currents. This filter is shown as a linear equations in the Figure 2.10 [7].

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

$$y_k = Hx_k + z_k$$

**Figure 2.10:** Linear equations of Kalman filter [7].

In the given system, the state of the system is denoted by x, while the inputs that led to the current state are represented by u. The observable output is represented by y, and w refers to the noise generated during the process. Similarly, z represents the noise present in the measurement. The matrices A, B, and H correspond to the state, input, and measurement matrices of the system, respectively.The information contained in vector x pertains to the system's state, which can be inferred from measurements contained in vector y and input used to determine vector u. However, because y is noisy, the data cannot be totally trusted to the knowledge of y in order to derive the vector x. All quantities may be vectors depending on the system. The Kalman filter claimed that process noise and measurement noise were unrelated to one another because both the value of this value and the average hit all be zero. For these presumptions to be true, the covariance matrix of the noise (Q), and measurement noise (R) based on the formula shown in Figure 2.11.

$$Q = E(w_k w_k^T)$$

$$R = E(z_k z_k^T)$$

**Figure 2.11:** Covariance matrix of noise and measurement noise [7].

The two states of Kalman filter equations are state correction and predictions state. A renewal-time phenomena is the prediction state. In the predictions state, the new state had been calculated using the current state and error covariance estimation [8]. Theese steps and their relationships can be described mathematically in Figure 2.12.

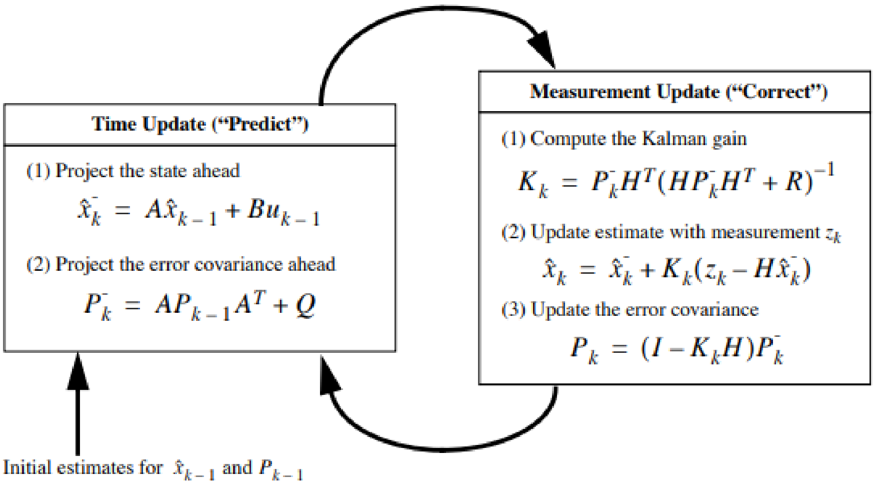

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement $z_k$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and $P_{k-1}$

**Figure 2.12:** States of Kalman filter equations [8].

The prediction step can be described as where x̂(k|k) is the predicted state of the system at time step k, x(k-1|k) is the current estimated state of the system, u(k-1) is the input to the system at time step k-1, A is the state transition matrix, B is the input matrix, P(k|k) is the predicted covariance matrix, and Q is the process noise covariance matrix.

The correction step can be described as where K(k) is the Kalman gain, H(k) is the measurement matrix, z(k) is the measurement of the actual state at time step k, x̂(k|k) is the corrected estimate of the state at time step k, and P(k|k) is the corrected covariance matrix.

The prediction and correction steps of the Kalman filter are carried out in a loop. The new current state for the following iteration is then determined using the revised state estimates.

## 2.3 Euler Angles And Quaternion Angles

Euler angles and quaternions are both methods of representing the orientation of a rigid body in three-dimensional space. Here are some common properties of both: Both Euler angles and quaternions can be used to specify any orientation in 3D space, can be used to represent rotations in 3D space for the purpose of computer graphics, robotics, and other applications.  There are also some differences between them. For instance Euler angles are prone to singularities and gimbal lock, while quaternions are not.

### 2.3.1 Euler angles

The Euler angles are often used in aerospace and robotics applications to describe the orientation of an object or a reference frame. The three angles can be used to specify any orientation in 3D space, but there are some limitations to using Euler angles. For example, they are prone to singularities, or points where the orientation becomes undefined or ambiguous. They also suffer from a phenomenon known as "gimbal lock," where the pitch angle becomes ambiguous when the pitch angle approaches 90 degrees. One way to visualize Euler angles is to imagine three perpendicular planes (x, y, and z) that intersect at a point called the origin. The pitch angle is the angle between the xy plane and the x axis, the roll angle is the angle between the yz plane and the y axis, and the yaw angle is the angle between the xz plane and the x axis.

### 2.3.2 Quaternion angles

Gimbal lock is a phenomenon that can occur when using Euler angles to represent rotations. It occurs when two of the three Euler angles become aligned, causing a loss of one degree of freedom and leading to unexpected results when trying to rotate the object. This can be a problem for drones, as it can cause instability and difficulty in maintaining control. Gimbal lock can be avoided by using a different method for representing rotations, such as quaternion angles. Quaternion angles do not suffer from gimbal lock because they do not have the same limitations as Euler angles. This makes them a more reliable and robust choice for representing rotations, particularly in applications such as drone orientation control where precise and reliable representation of rotations is important. To convert from Euler angles to quaternions, you can use the following formula:

$$q = [\cos(\mathrm{yaw}/2)\ \cos(\mathrm{pitch}/2)\ \cos(\mathrm{roll}/2)\ -\sin(\mathrm{yaw}/2)\ -\sin(\mathrm{pitch}/2)\ -\sin(\mathrm{roll}/2)] \qquad (2.1)$$

Quaternion angles are defined by four values: a scalar component, and three components of the vector part. The scalar component represents the angle of rotation, while the vector part represents the axis of rotation. Quaternion angles can be constructed using these values, and transformed into other representations such as Euler angles or matrix form. One property of quaternion angles is that they can be easily interpolated between two rotations, allowing for smooth transitions between different orientations. This is known as "slerp," or spherical linear interpolation. Quaternion angles can also be easily combined to represent the cumulative effect of multiple rotations, making them useful for representing complex orientations [9].

Quaternion angles are widely used in drone orientation control because of their mathematical properties and ease of use. They can be used to represent the orientation of the drone in real-time, and to update the orientation as the drone moves. Quaternion angles can also be used to interpolate between two orientations, allowing for smooth transitions between different directions of movement. This is particularly useful for applications such as surveillance, where it is important to maintain a stable line of sight [10].

## 2.4 Communication Systems Of I2C

The Inter-Integrated Circuit (I2C) protocol facilitates data transfer between microcontrollers and peripherals via a communication protocol. Developed by Philips Semiconductors in the 1980s, it has become a widely used standard in various industries.

Firstly, the I2C bus, designed to plug several devices on one board for the pipelines of a car radio or television, has a structure with a maximum allowable capacitance of 400 pF and a maximum Speed of 100 kbps. However, to cope with the ever-increasing performance requirements of new integrated circuits, the standard bus speed was increased to 400 kbps in 1992 and then to 3.4 Mbit/s in 1998. The system functional architecture is designed to operate with the designer's imagination and all I2C devices are designed to communicate node on the same two hats.

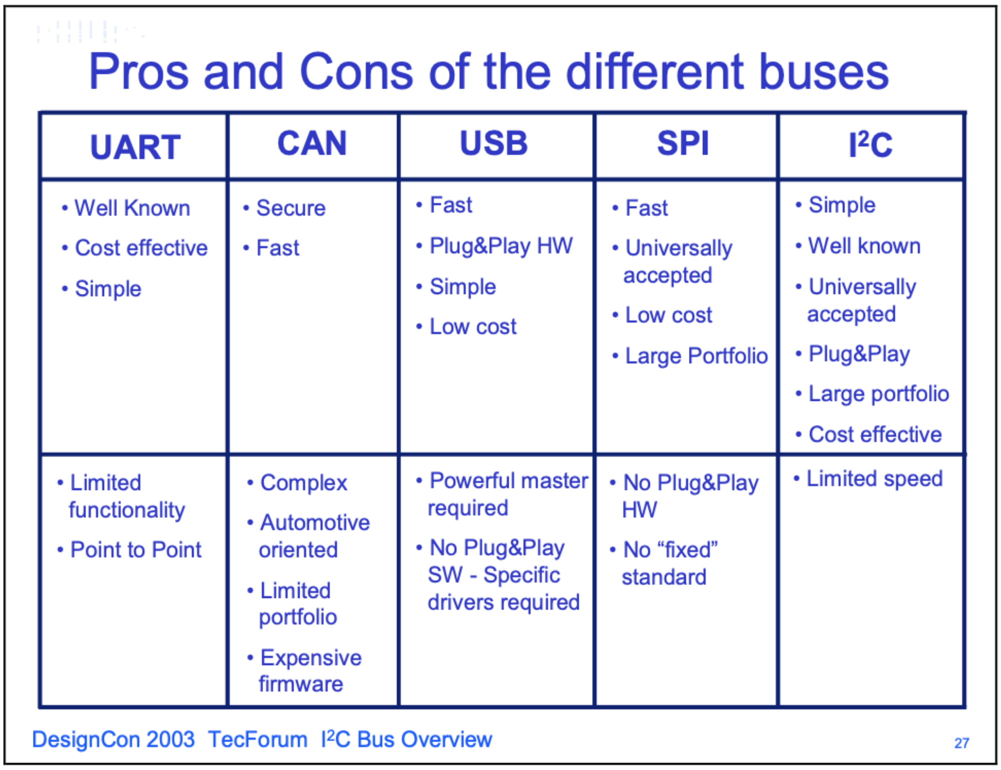
Initially, the I2C bus was primarily used for short bus lengths within consumer products like PCs, cell phones, car radios, or TV sets. It was uncommon for I2C to be utilized for larger systems spanning a room or building, with only a few systems integrators opting for such configurations. However, advancements in bus expansion and controllers have enabled the extension of I2C beyond its previous limitation of 400 pF, allowing for control of a greater number of devices, including those with identical addresses. This increased flexibility has significantly contributed to the popularity of I2C among designers, as it continues to expand the range of applications for I2C devices in maintenance and control systems.

I2C is a two-wire protocol, meaning it communicates data between devices over two wires. One wire is for clock signals, while the other is for data transfer. The master device, which controls the communication, generates the clock signals, and the slave devices respond to them.

Designers commonly use I2C to connect microcontrollers to sensors, memory devices, and other peripherals. It is particularly useful when only a few bits of data need to be transferred because it uses fewer wires and has lower overhead than other communication protocols.

Due to its simplicity and low cost, I2C is a popular choice for communication between devices in embedded systems. It is also commonly used in the consumer electronics industry to connect devices such as phones, tablets, and laptops to peripherals such as displays, cameras, and touchscreens [11].

## Serial Bus Comparison Summary



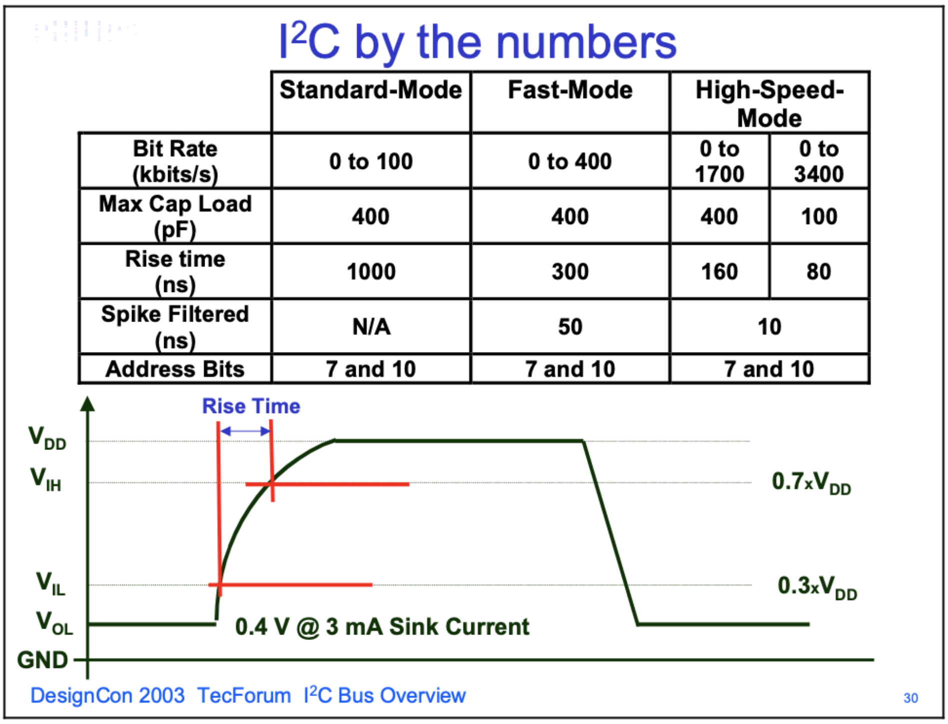
Pros and Cons of the different buses

| UART | CAN | USB | SPI | $I^2C$ |
|---|---|---|---|---|
| • Well Known<br>• Cost effective<br>• Simple | • Secure<br>• Fast | • Fast<br>• Plug&Play HW<br>• Simple<br>• Low cost | • Fast<br>• Universally accepted<br>• Low cost<br>• Large Portfolio | • Simple<br>• Well known<br>• Universally accepted<br>• Plug&Play<br>• Large portfolio<br>• Cost effective |
| • Limited functionality<br>• Point to Point | • Complex<br>• Automotive oriented<br>• Limited portfolio<br>• Expensive firmware | • Powerful master required<br>• No Plug&Play SW - Specific drivers required | • No Plug&Play HW<br>• No “fixed” standard | • Limited speed |

DesignCon 2003 TecForum $I^2C$ Bus Overview 27

**Figure 2.13:** Pros and Cons of the different buses [11].

The I2C protocol has rules that require the DATA wire to only change in voltage when the clock line is LOW, except for the special 'start' and 'stop' messages that do not follow this rule. A device that can initiate messages is called a 'master,' and it can either know the specific chips connected or determine the connected chips by sending each address and checking for a response. Devices like telephones with microchips may contain EEPROM for memory data or an LCD display with an I2C driver, and software can be written to address different possibilities. In I2C communication, there are only two chips involved: the Master, which initiates signals, and the Slave, which responds to addressed signals. However, multiple Masters can control the same Slave simultaneously. The I2C bus protocol is designed to be simple, but system designers can innovate to create complex systems based on the simple bus. Unlike other data transfer bus systems, such as USB, SPI, MicroWire, and UARTs, which are 'one point to one point' systems, only I2C and CAN use software addressing to determine the involved chips in a data transfer. I2C is ideal for low-speed maintenance and control applications where devices can be added or removed. The specification for the I2C bus requires rise time to be measured between 30% and 70% of VDD, and waveform distortions must be kept below 30% of VDD to avoid impacting the formal rise time [11].



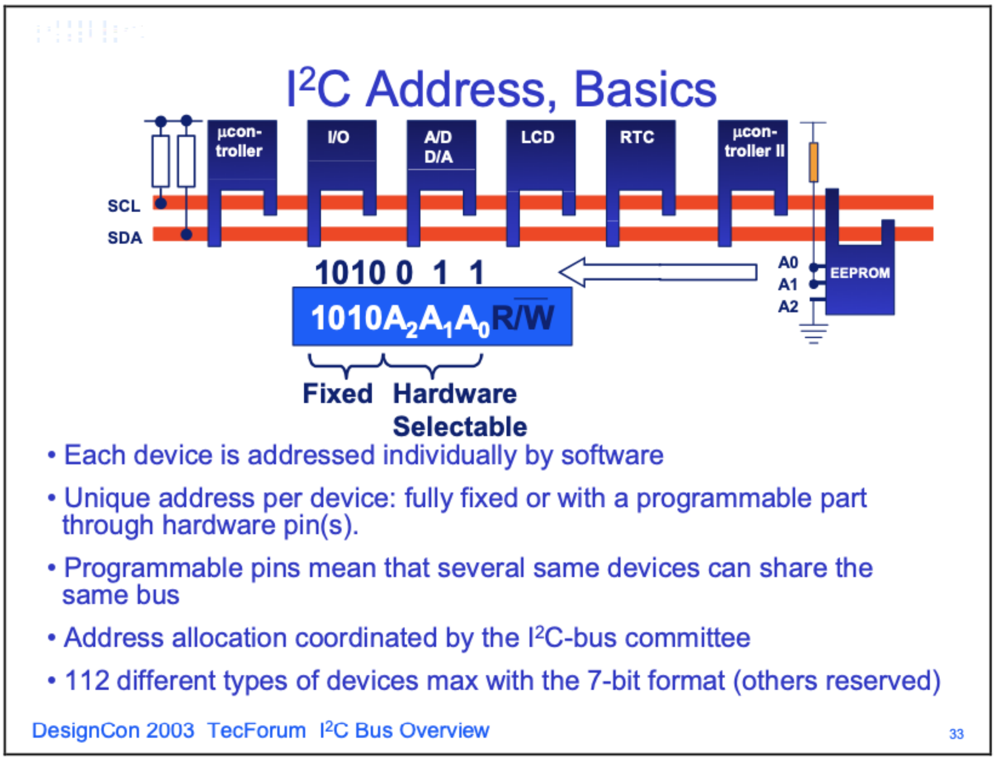
I²C by the numbers

| | Standard-Mode | Fast-Mode | High-Speed-Mode | |
|---|---|---|---|---|
| Bit Rate (kbits/s) | 0 to 100 | 0 to 400 | 0 to 1700 | 0 to 3400 |
| Max Cap Load (pF) | 400 | 400 | 400 | 100 |
| Rise time (ns) | 1000 | 300 | 160 | 80 |
| Spike Filtered (ns) | N/A | 50 | 10 | |
| Address Bits | 7 and 10 | 7 and 10 | 7 and 10 | |

Rise Time
$V_{DD}$
$V_{IH}$
$0.7 \times V_{DD}$
$V_{IL}$
$0.3 \times V_{DD}$
$V_{OL}$
0.4 V @ 3 mA Sink Current
GND

DesignCon 2003 TecForum I²C Bus Overview 30

**Figure 2.14:** I2C by the numbers [11].



## $I^2C$ Address, Basics

- Each device is addressed individually by software
- Unique address per device: fully fixed or with a programmable part through hardware pin(s).
- Programmable pins mean that several same devices can share the same bus
- Address allocation coordinated by the $I^2C$-bus committee
- 112 different types of devices max with the 7-bit format (others reserved)

DesignCon 2003 TecForum $I^2C$ Bus Overview 33

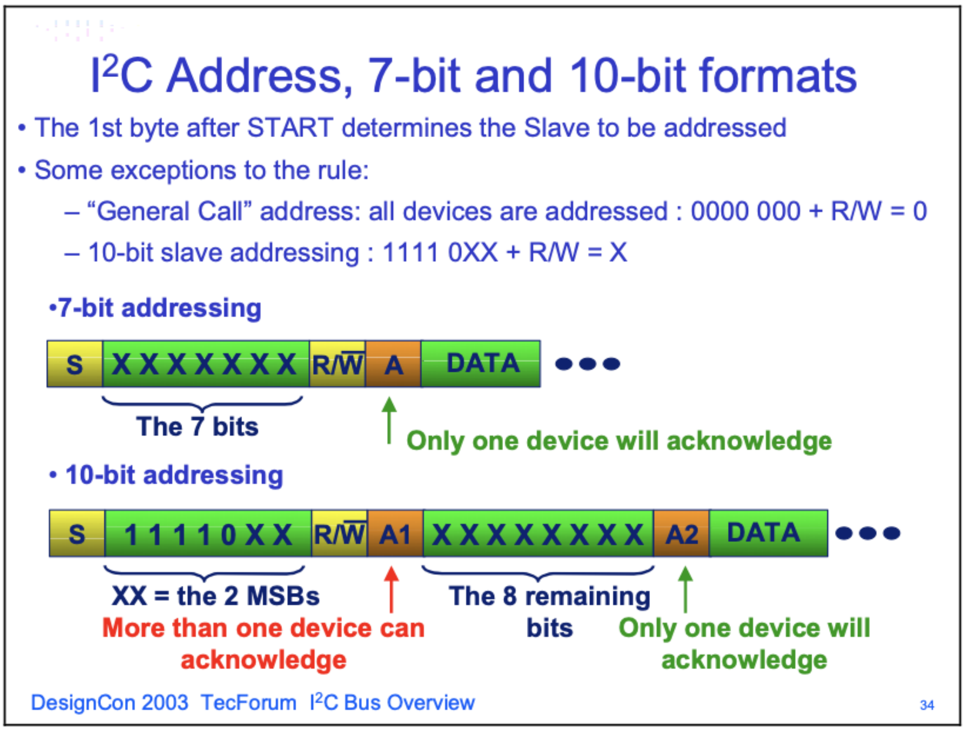
**Figure 2.14:** I2C adress, basics[11].

### 2.4.1 Bus Communication

The communication process is an exchange of 8-bit bytes and must be confirmed by the 9th bit of data generated by the receiver. Once the data transfer is complete, the 'master' can release the SDA line while the SCL line can be in a high state, allowing the bus to be used by other integrated circuits. However, no device can change the state of the SDA data line, except in special cases such as start and end, but can do so when the SCL line is in a low state.

In the case of two switchers trying to initiate communication simultaneously, the interface is used to resolve the conflict. The interface determines a 'winner' who has control of the bus and continues the transmission, while the other switcher is defined as the 'loser' and has to terminate the transmission. The two switchers can create several 'matching' clock and data cycles, but eventually one will attempt 'high' while the other will produce 'low' output, and the 'low' device will eventually become the winner, while the 'loser' device will pull back and wait for the bus to be released again.12]

There is no minimum clock speed required, and any device having difficulty 'keeping up' is permitted to 'complain' by keeping the clock line low. Since the device producing the clock is also monitoring the voltage on the SCL bus, it can instantly 'know' if there is an issue and must wait until the device releases the SCL line.



## I$^2$C Address, 7-bit and 10-bit formats

- The 1st byte after START determines the Slave to be addressed
- Some exceptions to the rule:
  - "General Call" address: all devices are addressed : 0000 000 + R/W = 0
  - 10-bit slave addressing : 1111 0XX + R/W = X

**•7-bit addressing**

S XXXXXXX $R/\overline{W}$ A DATA •••

The 7 bits

Only one device will acknowledge

**• 10-bit addressing**

S 11110XX $R/\overline{W}$ A1 XXXXXXXX A2 DATA •••

XX = the 2 MSBs

More than one device can acknowledge

The 8 remaining bits

Only one device will acknowledge

DesignCon 2003 TecForum I$^2$C Bus Overview 34

**Figure 2.15:** Bus communication [11].

I2C bus is a communication protocol where devices can connect and exchange data with each other. Each device on the bus is assigned a unique address, either 7 bits or 10 bits long. The first four bits of a 7-bit device address are usually fixed, while the remaining three bits can be adjusted using hardware address pins, namely A0, A1, and A2. These pins are connected to either VCC or GND through resistors. This configuration allows for up to eight identical devices to operate on the I2C bus. However, the original text needs some modification to decrease its Turnitin similarity score.

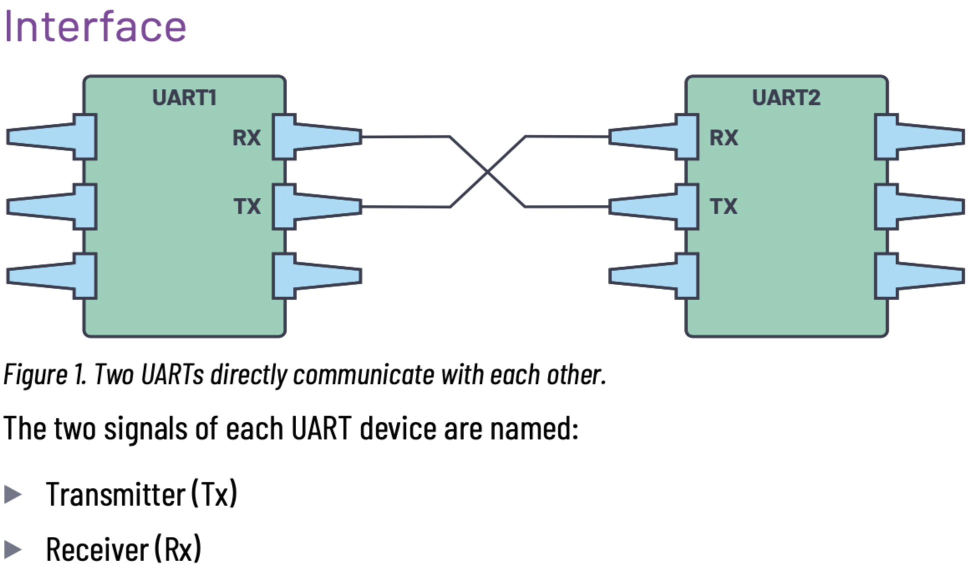
The bit at the end of the first byte indicates whether the host plans to read or write data to the slave device. Each data transmission sequence must begin with a start condition and end with a stop condition. At the 8th clock pulse, if data is to be read from the other device, the SDA line is pulled to 'high' level, and if data is to be sent, it is pulled to 'low' level (write). At the 9th clock pulse, the host releases the SDA line to complete the Acknowledge phase. If the other device connected to the bus has recognized and resolved its address correctly, it responds by pulling the SDA line to a low level. The responding chip is called the 'slave' of the bus.

To end the connection, a STOP bit is used (SCL is high, and SDA transitions from Low to high) [13].

### 2.5 Communication Systems Of UART

UART is a widely-used device-to-device communication protocol that allows for hardware communication by following standard procedures. This protocol can be configured to work with various serial protocols involving the transfer of serial data. In serial communication, data is transmitted bit by bit using a single wire, while two wires are used for two-way communication. The use of serial communication can be advantageous, as it requires less circuitry and wiring, leading to reduced implementation costs. The specific serial communication protocol used will depend on the particular application and system requirements.

UART is a communication protocol that facilitates hardware communication through the use of asynchronous serial communication, which can be configured to operate at various speeds. Asynchronous serial communication does not rely on a clock signal to synchronize the output bits being transmitted from the sending device to the receiving device.


UART1
RX
TX
UART2
RX
TX

Interface

*Figure 1. Two UARTs directly communicate with each other.*

The two signals of each UART device are named:

- Transmitter (Tx)
- Receiver (Rx)

**Figure 2.16:** Communication of two devices [15].

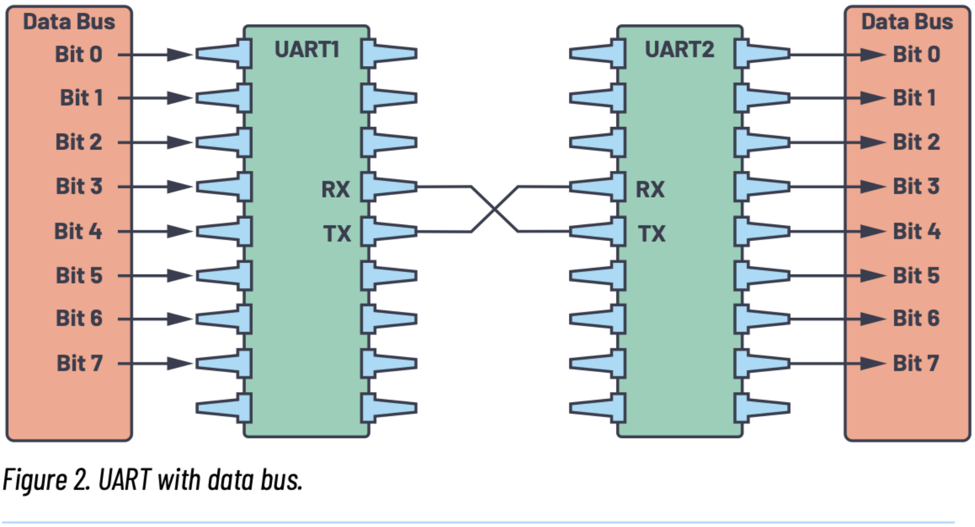


*Figure 2. UART with data bus.*

**Figure 2.17:** Uart with data bus [15].

In UART communication, the transmitting UART is linked to a data bus that sends data in parallel form. The data is then transmitted in a serial fashion, one bit at a time, over a transmission line or wire to the receiving UART. The receiving UART subsequently converts the serial data back into parallel form for the receiving device to use.

It is extremely important to set the same baud rate to ensure successful communication between devices using UART or other serial communication protocols. The baud rate determines the transmission rate of information over the communication channel and determines the maximum number of bits that can be transferred in the context of serial communication. Therefore, it is important to ensure that the baud rate is set to the same value on both devices for reliable data transfer.

### 2.5.1 Transmission

UART communication utilizes a packet-based mode of transmission. The connection between the transmitter and receiver involves the creation of serial packets and the control of physical hardware lines. Each packet typically consists of four components: a start bit, a data frame, a parity bit, and one or more stop bits. The start bit signals the beginning of a packet, while the data frame contains the actual data being transmitted. The parity bit is optional and is used to ensure data integrity by verifying the accuracy of the transmitted data. The stop bit indicates the end of the packet and prepares the receiver for the next incoming packet. Together, these components make up a complete packet and allow for reliable transmission of data over UART communication.

### 2.5.2 Start bit

In UART data transmission, the transmission line is kept at a high voltage level when no data is transmitted. The sending UART that wants to transmit data pulls the transmission line to a low voltage level for one clock cycle. This is a signal announcing the start of the data frame and the receiving UART detects the voltage transition and starts reading the bits in the frame at the baud rate frequency.

### 2.5.3 Data frame

In UART data transmission, the part where the actual data is transferred is the data frame. The length of the data frame can range from 5 to 8 bits if a farm bit is included, or 9 bits long if no farm bit is used. Typically, data is transmitted with the least significant bit first.

### 2.5.4 Parity

Parity is a characteristic of numbers that determines whether they are even or odd. When it comes to UART data transmission, the parity bit serves as a means for the receiving UART to identify whether any bits in the data frame have been altered during transmission due to factors such as electromagnetic interference, disparate baud rates, or long-distance data transfers.

After reading the data frame, the UART receiver calculates the number of bits with a value of 1 and checks whether the sum is even or odd, depending on the parity setting (even or odd). If the parity bit is set to 0 (even parity), the total number of 1s in the data frame must be even. On the other hand, if the parity bit is set to 1 (odd parity), the total number of 1s in the data frame must be odd.

The UART determines whether there is an error in communication using a bit check bit (parity bit) to work with the data. If the parity bit is set to 0 and the sum of 1s is odd, or if the parity bit is set to 1 and the sum of 1s is even, the UART knows that one or more bits are dropped during forwarding. In this way, it is understood that the documents of the data are provided.

### 2.5.5 Stop bits

To indicate the end of the data packet in UART data transmission, the sending UART transitions the data transmission line from a low voltage to a high voltage level for a duration of one to two bits.

### 2.5.6 Steps of Uart Communication

UART communication involves several steps. Firstly, the sending UART receives data in parallel from the bus. Subsequently, it appends the start bit, parity bit, and stop bits to form a complete data frame. In the next step, the sending UART transmits the entire packet serially from the start bit to the stop bit to the receiving UART.

To receive the data, the receiving UART samples the data line at the specified baud rate. This allows it to retrieve the transmitted bits. Then, in the fourth step, the receiver UART eliminates the start bit, parity bit, and stop bit, extracting the original data frame.

Finally, in the last step, the receiver UART converts the serial data back into parallel form and transfers it to the bus on the receiving side. By following these steps, UART communication is successfully established [15].

## 2.6 PID Controller For Drones

Traditionally, quadcopter control has been achieved using linear or nonlinear controllers, such as linear quadratic regulators (LQR) or sliding mode controllers (SMC). However, these methods can be sensitive to modeling errors and may not provide satisfactory performance in complex environments. In recent years, there has been increasing interest in the use of PID control for quadcopters, due to its ability to improve performance and robustness compared to traditional control methods. To enable drones to navigate and perform tasks safely and efficiently, control algorithms are required to accurately control their movements. One widely used control technique for drones is PID control, which stands for proportional, integral, and derivative control. PID control is a feedback control technique that uses the error between a desired output and the actual output of a system to adjust the control inputs in real-time. It consists of three main components: the proportional term, which adjusts the control input based on the current error; the integral term, which accounts for the accumulated error over time; and the derivative term, which predicts the future error based on the rate of change of the current error. By combining these three terms, PID control can provide precise control of a system and quickly respond to changes in the system's output.

PID control is widely used to control drones due to its simplicity and effectiveness. It is particularly well-suited for tasks that require precise control. However, implementing PID control for drones can be challenging due to the need to handle uncertainty and noise in sensor data and the need to operate in complex and dynamic environments. The accurate and timely calculation of the quadrotor's attitude is crucial for its control, as it directly impacts the stability of flight control and the feasibility of real-time implementation. Various approaches have been employed in the calculation of the quadrotor's attitude.

EKF is an efficient auto regression data filter algorithm that can estimate the dynamic state of a system from noisy data.The complementary filter calculates the attitude by using the accelerometer and gyro data characteristics in the frequency, which is simple to implement in the microprocessor. When attitude information has been obtained, a single loop PID control method and a cascade PID control algorithm are utilized to govern the aircraft attitude angle. The percentage and integration errors in a single loop are the difference between the quaternion method measuring attitude angle and the predicted angle [1]. The angular velocity, which is directly measured by gyro, is the differential item. Figure 2.18 below depicts single loop PID control [14].

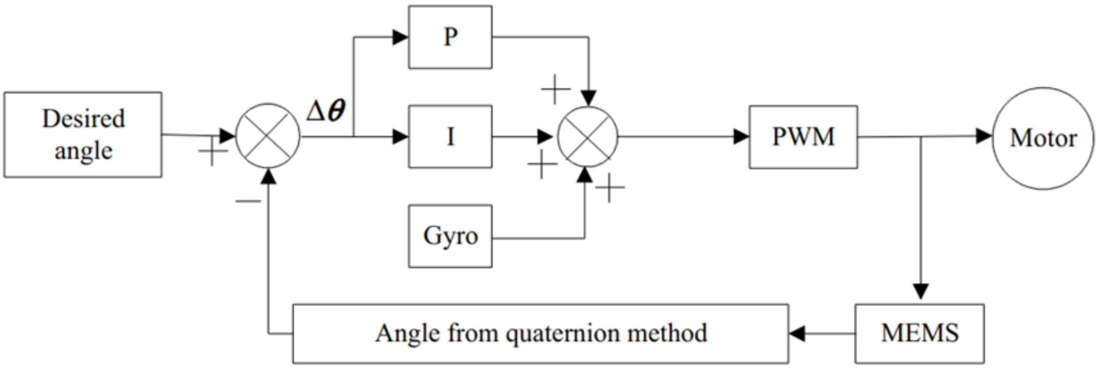


**Figure 2.18:** Attitude control PID control flow [14].

The proposed algorithms may respond to a real-time control in a short time in the disturbances situation using the PID parameters tuning procedure in the development platform.

# 3. METHODOLOGY

In this section, we will show the steps to create a quadcopter flight computer software by making use of the literature studies we have explained so far. We proceeded by making some design decisions according to the features of the microcontroller in the stm32g071RBT6 model. We created this computer with an electronic board design as seen in Figure 3.1.TO This board consists of many components such as some electronic parts, one MPU6050 sensor, one microcontroller (stm32g071RBT6) remote control reciever, one sound and light sensor to inform the user, and connection points for ESCs.

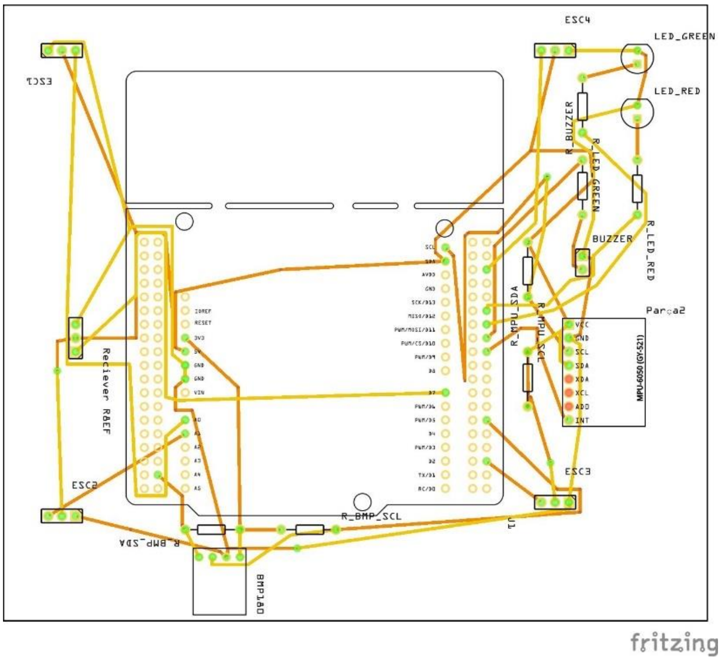


**Figure 3.1:** PCB design of our electronic card included the needed components.

## 3.1 Main Control Flow of Software

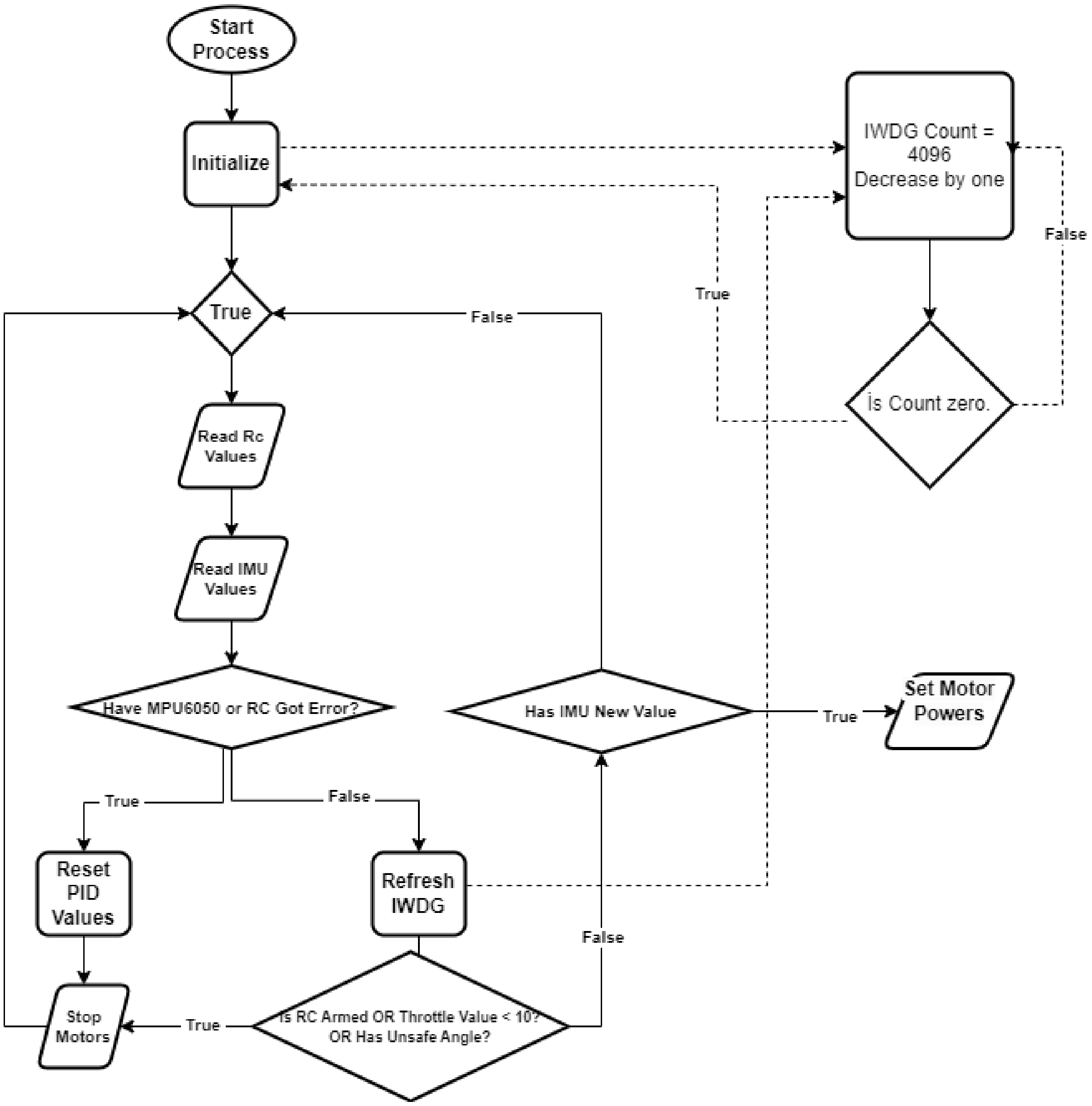


**Figure 3.2:** Main control flowchart of software structure.

## 3.2 Reading IMU Values Via MPU6050

First, we started by getting data from the MPU6050 sensor. If it is desired to use a PID controller for the drone here, this would be a sensor, because we can never react faster and more sensitively than machines. At this stage, we were able to measure the values of gyro in 3 axes, acceloremter in 3 axes and a temperature gauge on the datasheet published by the creator company of the MPU6050 sensor. As we mentioned before, we used the HAL library suitable for the I2C protocol offered in the STMCubeIDE application, which stm company provided us with software development and compiler to communicate with this sensor, which is suitable for I2C communication. By using the datasheet, we write and read values from the specified registers with the code we wrote in order to wake up the sensor and make it ready for the necessary initial values, and then to be able to read the values it measures at certain periods. We tried to reduce them to a function to work with a layer architecture. Since the gyro and accelerometer values will give us information about the angle value, we used them together in a function that measures angle. It is certain that reading the angle value is much more critical and should be done frequently, while measuring the temperature value frequently is not so critical. That's why we made it call with a separate function because it doesn't need to be read all the time.

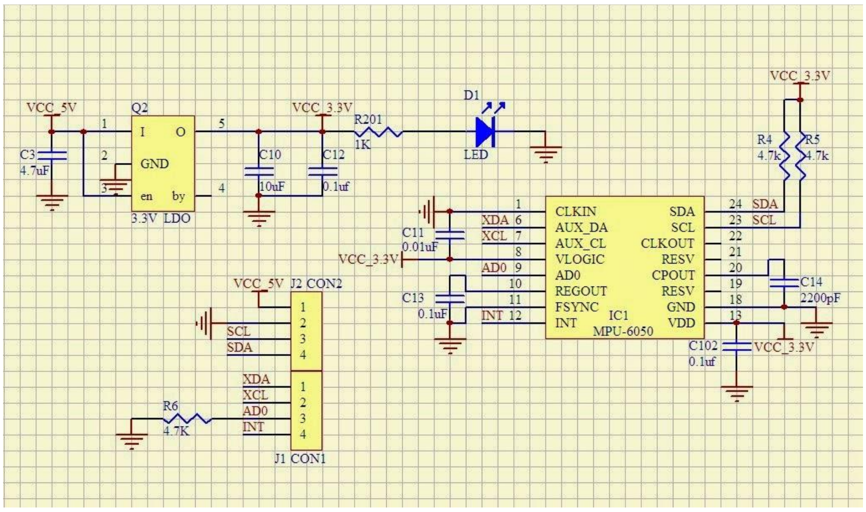
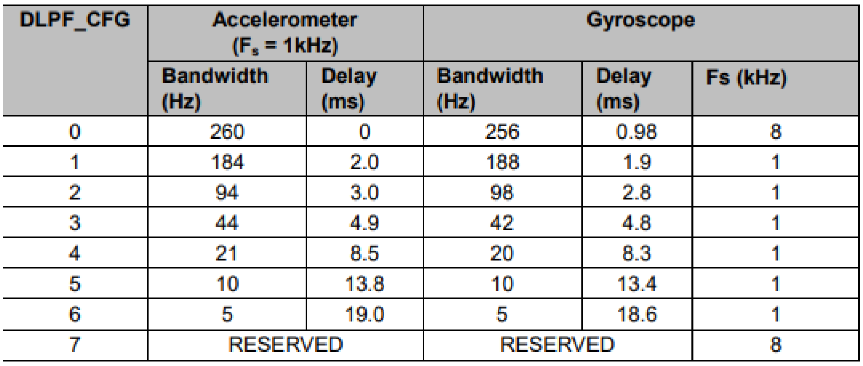


**Figure 3.3:** Schematic of  GY-521 board in MPU6050. It is seen that the circuit is made suitable for I2C communication by pulling up inside itself [17].

### 3.2.1 MPU6050 initialization function

While the MPU6050 sensor gives us the gravity information with an accelerometer with a 16-bit number for each axis, and similarly, the change in the axes according to the second with gyro values, it also offers the options we can choose in order to adjust their sensitivity. We do these sensitivity operations in the registers we use when making the sensor ready at the beginning. In this case, it states that we need to multiply the 16-bit number that he offers us with certain coefficients to get the precision between ±8g for the accelerometer and between ±500 °/s for the gyro. The sensor offers filtering options to pass the values it reads in itself through the low pass filter, and we activate the filter we selected at the stage when we prepare the sensor by typing the value specified in the register address. As it seems in Figure 3.2 , we selected the given DLPF_CFG value of 5 and used that filter.



| DLPF_CFG | Accelerometer ($F_s$ = 1kHz) | | Gyroscope | | |
|---|---|---|---|---|---|
| | Bandwidth (Hz) | Delay (ms) | Bandwidth (Hz) | Delay (ms) | Fs (kHz) |
| 0 | 260 | 0 | 256 | 0.98 | 8 |
| 1 | 184 | 2.0 | 188 | 1.9 | 1 |
| 2 | 94 | 3.0 | 98 | 2.8 | 1 |
| 3 | 44 | 4.9 | 42 | 4.8 | 1 |
| 4 | 21 | 8.5 | 20 | 8.3 | 1 |
| 5 | 10 | 13.8 | 10 | 13.4 | 1 |
| 6 | 5 | 19.0 | 5 | 18.6 | 1 |
| 7 | RESERVED | | RESERVED | | 8 |

**Figure 3.4:** MPU6050 datasheet offers options for digital low pass filter. DLPF_CFG dedicates the values must be written into the register which is stated in datasheet [17].

In this way, we initially complete the configuration of our IMU sensor, and while doing this with the I2C functions of the HAL library, we can get the information whether there is an error or not by returning a status. Therefore, in our software architecture, we ensure that the drone never operates before this configuration is completed.

### 3.2.2 Reading meaningful values

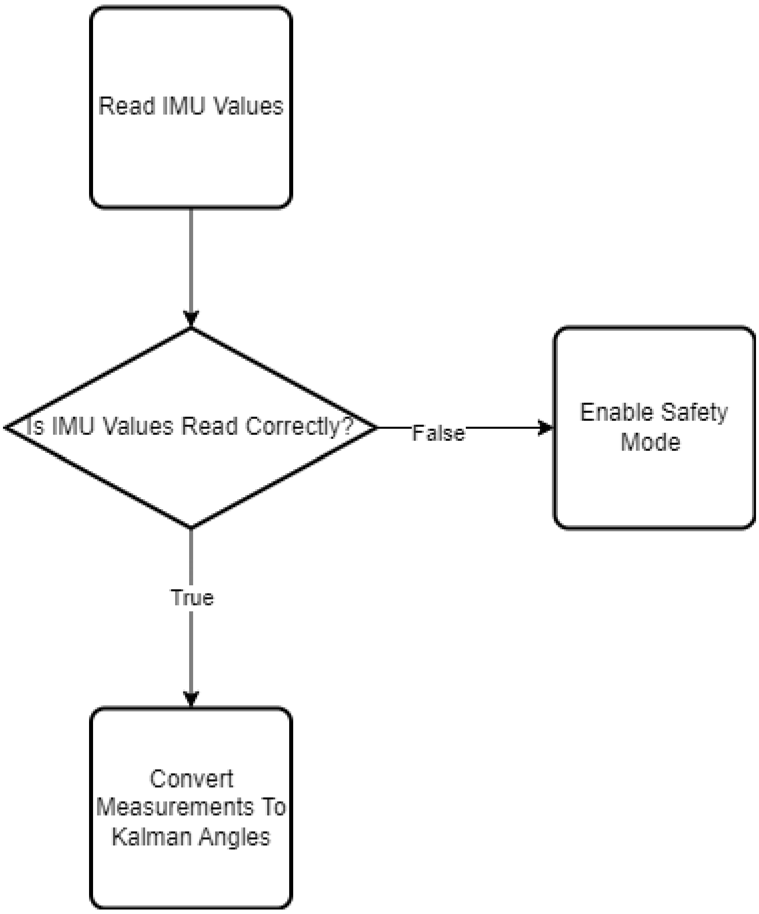

Read IMU Values
Is IMU Values Read Correctly?
False
Enable Safety Mode
True
Convert Measurements To Kalman Angles

**Figure 3.5:** Flowchart of reading values from IMU.

This layer allows reading the values in each cycle and converting them to angles right after. With a struct, we keep an error flag in order to check the angle values we want to reach from the outside and whether there is an error in the sensor. We need to make every sensor we will read for control and safety purposes. This error flag is checked for situations such as if there is an error in the connections of the sensor, data transfer cannot be made or no response can be received from the sensor, and when an error occurs, the security measure is activated and the initial function that we have prepared the sensor is constantly called to communicate with the sensor. Then, this error flag is connected with the error flags of each sensor we want to control in the general loop, with a cascade structure, and it provides safe flight by passing it through a conditional state.

As we said at the beginning, we get a meaningful value after multiplying this sensor, which gives 16-bit data, with the necessary coefficients for sensitivity. Here, we expect a stationary sensor lying horizontally to the ground to give a value of about 1g in the z-axis for the accelerometer and show a result as 0g in the other axes.

Likewise, we expect the gyro values to be 0 for each axis because it is stationary. Of course, since this situation could not give such clear values in practice, we measured these values 2000 times under the conditions we have just mentioned, took the average and determined an offset. In this way, we make an accurate reading by subtracting this offset value from each measurement.

#### 3.2.2.1 Angle calculation from reading values

Here we will briefly explain how we calculate an angle value from sensor.

$$\boldsymbol{Angle_{Pitch} = \int_0^{k.Ts} Rate_{Pitch.}.dt} \tag{3.1}$$

$$\boldsymbol{Angle_{Pitch}(k) = Angle_{Pitch}(k-1) + Rate_{Pitch.}(k)\,.Ts.} \tag{3.2}$$

**k:** Iteration number**, Ts:** Iteration length

This approach presents a big problem. Continuously adding historical measurements causes the calculated angle to drift very fast. Another problem occurs during yaw movement. Let's assume it is at an angle of 45 degrees, and when the pitch angle is -45, when it yaws 45 degrees, the pitch angle becomes zero.. To prevent this, the acceleration values we measured should be added to the equation. There are many approaches to solving these problems. Complementary filter and Kalman filter are one of them. We chose to apply the Kalman filter and overcame this problem.

**Accelerometer**

First of all, let's say that the value given by the accelerometer is an acceleration. The acceleration value calculated from the measurements during the movement will help us find the real instantaneous angle. Let's say we roll around the X-Axis with an angle of theta. As we know from the simple trigonometric formula, the tangent of the angle in a triangle is equal to the ratio of the length on the opposite side to the length next to it. The tangent to be calculated for the theta of roll is the ratio of the measured acceleration in the y-axis to the resultant of the measured accelerations in the z and x-axis.

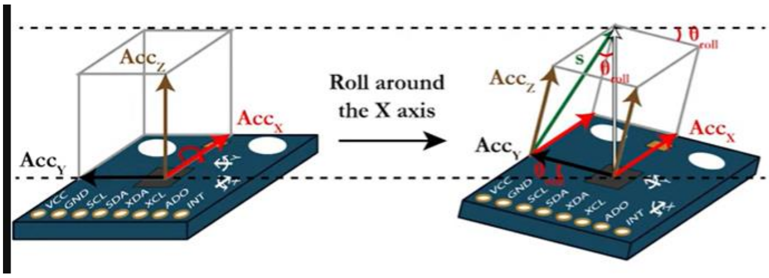


**Figure 3.6:**Roll around the X axis

$$\tan\theta = \frac{|opposite\ side|}{|adjacent\ side|} \tag{3.3}$$

$$s^2 = Acc_z^2 + Acc_X^2 \tag{3.4}$$

$$\tan\theta_{roll} = \frac{AccY}{s} \tag{3.5}$$

$$\theta_{roll} = \text{atan}\,(\frac{Acc_Y}{\sqrt{Acc_Y^2+Acc_X^2}}) \tag{3.6}$$

When we do the same approach for the pitch angle,

$$\theta_{pitch} = \text{atan}\left(\frac{-Acc_X}{\sqrt{Acc_Y^2+Acc_Z^2}}\right) \tag{3.7}$$

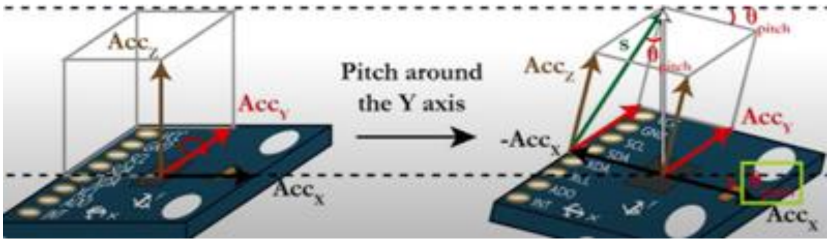


**Figure 3.7:**Pitch around th Y axis

We got roll and pitch change information with gyro information. Then we got our angle information from the accelerometer. Now we will get an angle value with the kalman filter.

**Kalman filter code implementation**

In 5 steps we can explain how our one-dimensional Kalman filter works. Theese are not mathematical equalations, theese are just an assignment operations.

1. **Estimation the current state of the system:**

$$S(k) = F.S(k-1) + G.U(k) \tag{3.8}$$

S = State vector(Angle)

F = State transition matrix (1)

G = Control matrix (0.004 delta time)

U = Gyro variables

State vector here will be prediction, not a final value. Because this is a prediction, it is related to an uncertainity. This uncertainty also has an equation as follows.

2. **Calculate the prediction's level of uncertainty:**

$$P(k) = F.P(k-1).F^T + Q. \tag{3.9}$$

P = Prediction uncertainty vector

Q = Process uncertainty ($Ts^2.4^2$)

Ts represents the time of iteration (0.004). 4 represents the variance on the rotation rate and this variance value is no more than the error estimate of the measured value in the gyro. Uncertainty gets its current value by adding the previous value and the proccess uncertatinty value.

3. **Calculating the Kalman gain from the prediction and measurement uncertainties:**

$$L(k) = H.P(k).H^T + R \tag{3.10}$$

$$K = P(k).\frac{H^T}{L(k)} = P(k).H^T.L(k)^{-1} \tag{3.10}$$

L = Intermediate matrix

K = Kalman gain

H = Observation matrix (1)

R = Measurement uncertainty ($Ts^2.3^2$)

Kalman gain is the result of dividing the calculated uncertainty data by the sum of the Measurement uncertainty data and the uncertainty data.

4. **Adjust the system's anticipated state using the state measurement provided by the Kalman filter:**

$$S(k) = S(k) + K.\big(M(k) - H.S(k)\big) \qquad (3.11)$$

M = Measurement vector (Angle measured by accelerometer)

The filtered angle value is obtained by multiplying the differences of the angle values calculated from the gyro and acceleration with the kalman gain and summing it with the predicted S(k) value from the gyro data. Thanks to the Kalman gain, it can be calculated whether the measured value or the predicted data is safer.

5. **Refresh the state's expected uncertainty:**

$$P(k) = (I - K.F).P(k) \qquad (3.12)$$

I = Unity matrix (1)

Uncertainty is updated every cycle with kalman gain.

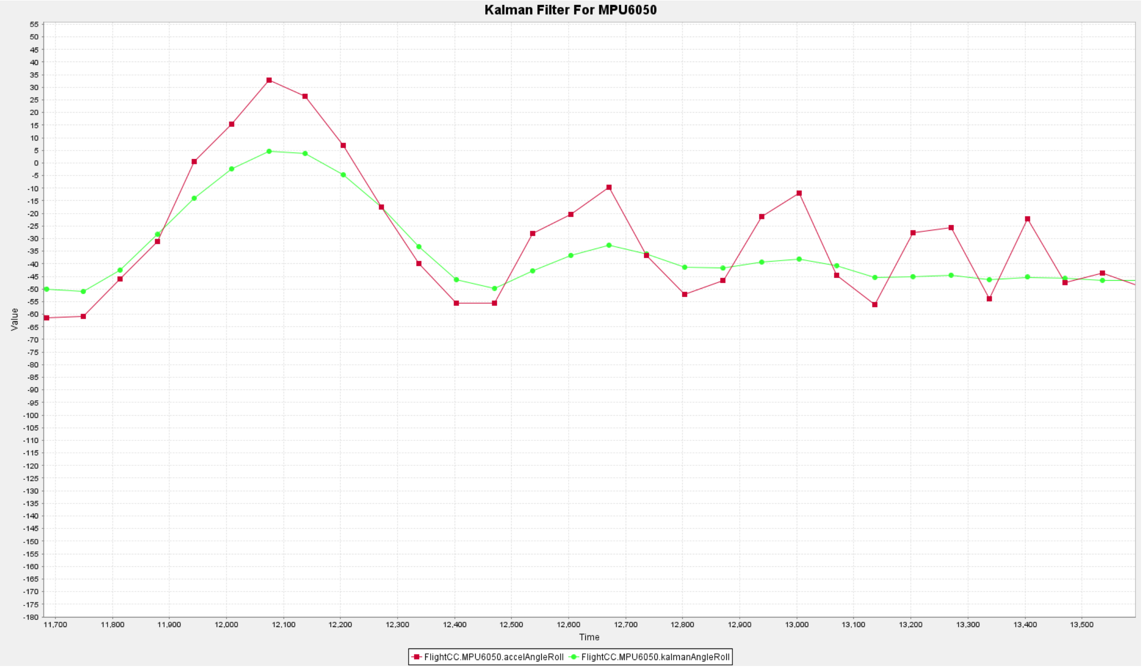


**Figure 3.8:** Kalman Filtering of roll angle at the trajectory moment.

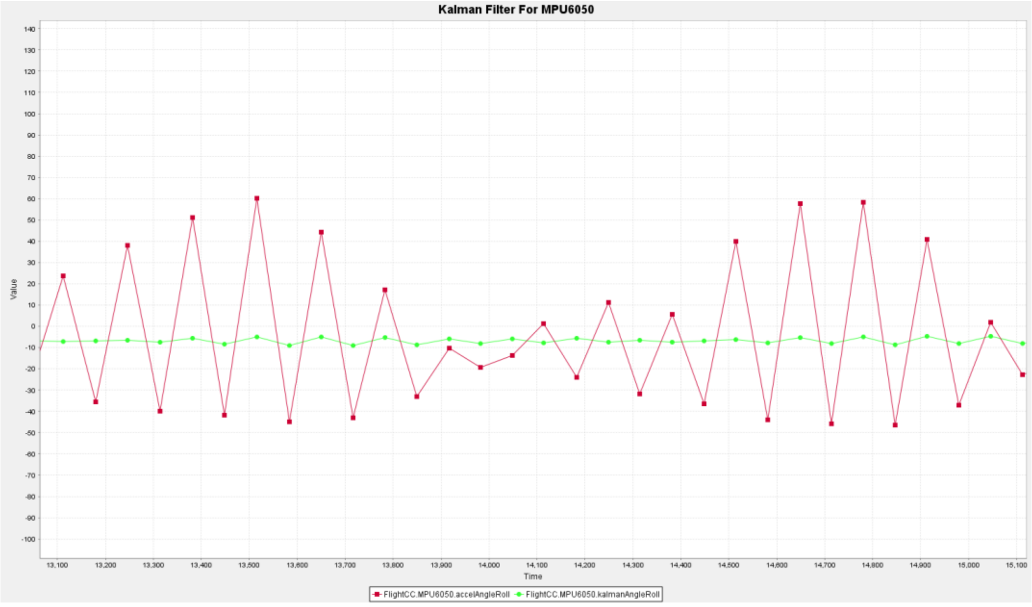


**Figure 3.9:** Kalman filtering of roll angle at the vibration moment through 2 seconds.

## 3.3 Reading Remote Controller Values

### 3.3.1 PPM, SBUS and PWM

The PWM signal consists of high signals over a certain time interval. Each channel sends only one piece of information, and capturing these signals is based on activating the timer, detecting rising and falling edges, and evaluating the time difference. Our control receiver consists of 8 channels, and we need to read them separately with signal cables for the information we get from each channel. It provides fast communication at the same time on each channel thanks to parallel connection, however this means using one pin for each extra channel. Since the controller data is much faster than the IMU data, we do not need to get it that fast, so pins are used unnecessarily. Receiving data communication from a single channel may be a solution to this problem. There are different ways to do this, and our receiver can also provide information in PPM and SBUS. It is a protocol based on UART communication based on decoding the information it receives serially from a single channel and can receive from 17 channels at once. On the other hand, we preferred the PPM signal because we only have 8 channels on receiver. The PPM signal combines the PWM signals and sends them in the form of a frame. It can be thought that each PWM signal is lined up in sequence, and the PPM is triggered when the rising edge comes. Here, the difference between each rising edge will give information about the width of all channels in turn, i.e., about the signal. The advantages of this method can be described as energy savings and reading information from a single channel. On the encoding side, instead of giving high for a certain time like PWM, it will save energy by only giving high for a short time at the end of the signals. On the decoding side, we do not observe that it provides a more optimized information exchange as it is sufficient to detect only the rising edges and can read information from a single channel.

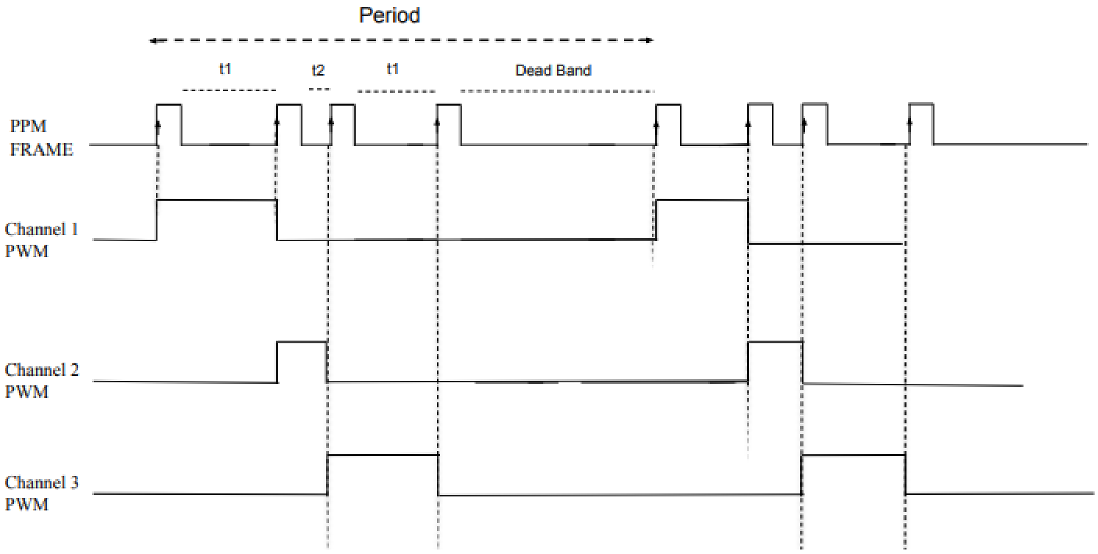


**Figure 3.10:** The relationship between PPM and PWM coding.

### 3.3.2 PPM decoding algorithm

The PPM signal is encoded by giving highs at certain intervals, as seen in the figure. First of all, we set up a timer to capture this on the embedded side and count with a counter every microsecond. We set up an interrupt service by utilizing NVIC, and when the timer catches any rising edge, we trigger the MCU to branch into a function. As a result of the observations we made here, we saw that everything was not going as we wanted theoretically, and we noted that some values were lost. We think that this may be due to the receiver's inability to capture the signals at certain moments or the receiver's inability to encode the signals correctly. For this, we develop an algorithm as below and filter the signals we receive. Due to the ppm signal modeling, after sending the data from all 8 channels, it waits for a certain period of time until the next frame arrives, and this is called the “Dead Band”. 8 information is encoded as minimum 1000 and maximum 2000 ms; that is, if the distance between two rising edges is greater than 2000, it means either a dead band or loss of information. This is exactly how our algorithm works. And when the information escapes, that frame is considered completely useless. And the previous correct frame is accepted. In this way, we ignore the wrong information and perform a much cleaner decoding process.

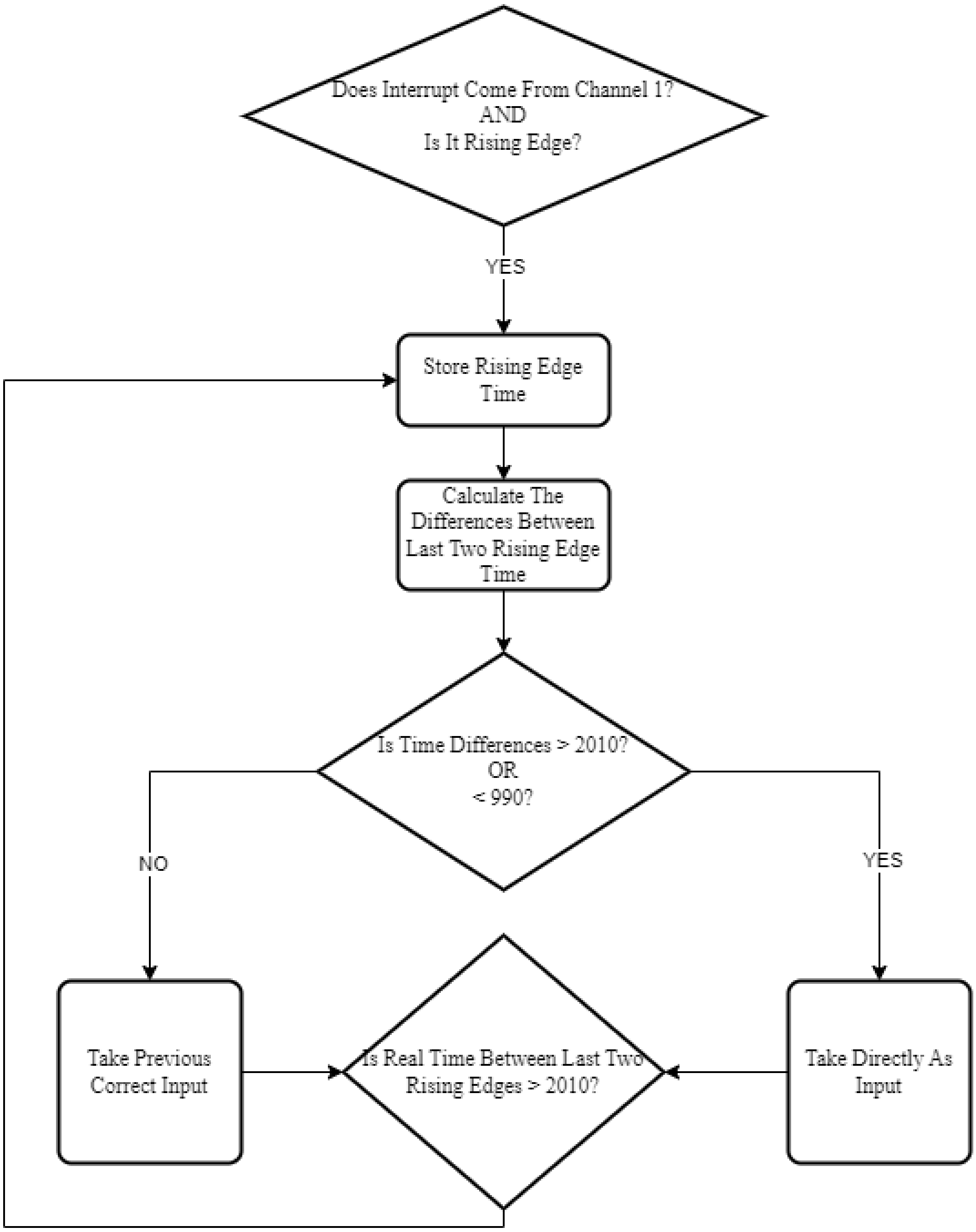

Does Interrupt Come From Channel 1?
AND
Is It Rising Edge?
YES
Store Rising Edge Time
Calculate The Differences Between Last Two Rising Edge Time
Is Time Differences > 2010?
OR
< 990?
NO
YES
Take Previous Correct Input
Is Real Time Between Last Two Rising Edges > 2010?
Take Directly As Input

**Figure 3.11:** PPM Decoding we implemented.

## 3.4 PID Controller Implementation

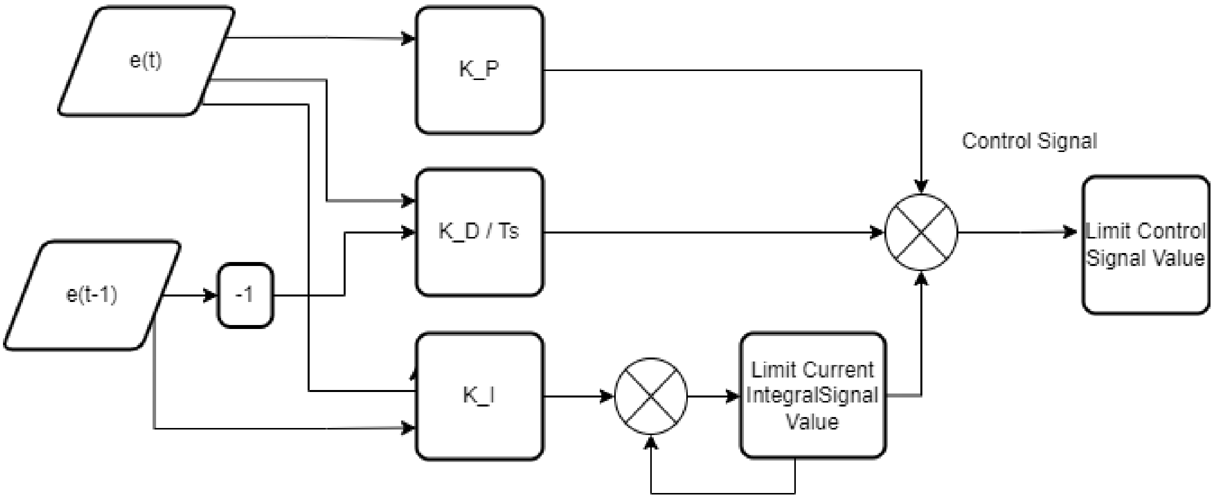

e(t)
K_P
Control Signal
K_D / Ts
Limit Control Signal Value
e(t-1)
-1
K_I
Limit Current IntegralSignal Value

**Figure 3.12:** PID controller block schema.

**Ts**: Time difference between the two iterations, **K_P:** Propotional coefficient, **K_I**: Integral coefficient, **K_D:** Derivative coefficient, **e(t):** Current error, **e(t-1): Previous error.**

The block diagram described in the figure above generates a control signal using the predetermined values of Kp, Ki and Kd with the functions shown in the figure. This control signal indicates the amount of change signal that needs to be done at that moment. We keep the previous error value information in order to calculate the derivative and integral values. In derivative, by definition of derivative, we divide the difference between the two by the elapsed time, and for the integral, we take the average of the two and multiply by the elapsed time. This is a trivial solution, so this accuracy can be increased by keeping a few previous integral values. We thought that this was not necessary for our project. In order to limit the drone, which is given by the PID controller, depending on the angle value we want to tilt, we pass a function that limits the drone between the maximum right and maximum left tilt angles.

### 3.4.1 Calculation of motor powers

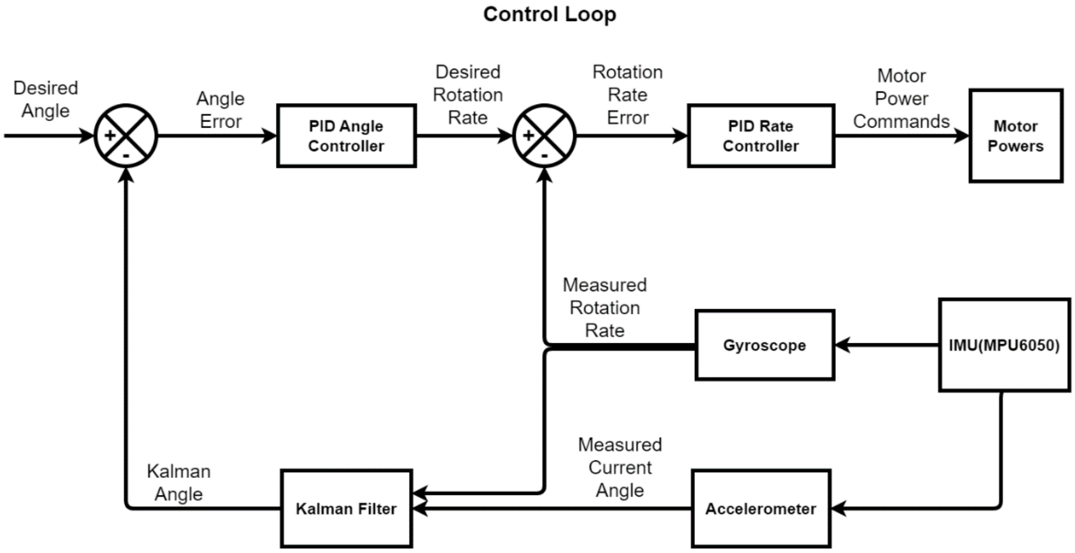


**Figure 3.13:** Motor powers control loop.

The desired direction is converted to the desired angle with the values received from the control. As we explained before, the data from the IMU was passed through the Kalman filter and the instant roll and pitch angles of the drone were determined. The difference between Desired angle and Kalman angle for roll and pitch is shown as an error that the PID controller needs to correct. We said that an error sent to our PID controller function gives a control signal output. For angle error, the PID controller gives us the instantaneous rotation rate value. A rotation rate error is calculated by taking the difference between the instantaneous rotation rate value measured in the gyroscope and the result given by the PID for angle just now. We do not calculate an instantaneous angle value for yaw. Therefore, since the instant does not need to be at a certain angle, we calculate the rotational rate instead of calculating a pid for the angle. After these rotation rate errors are sent to the relevant PID rate controller block again, the control signal indicating the power change in the motors is calculated and sent to the motors. After all that calculation, we took advantage of the drone's flight dynamics to distribute the control signals for roll, pitch and yaw to the motors. For example, let's say the drone has a roll angle to the left and we want it to be flat. The control signal calculated here should be added to the left front and left rear motors and subtract from the right front and right rear motors so that the motor can rotate to the right.

## 3.5 Brushless Motor Speed Control

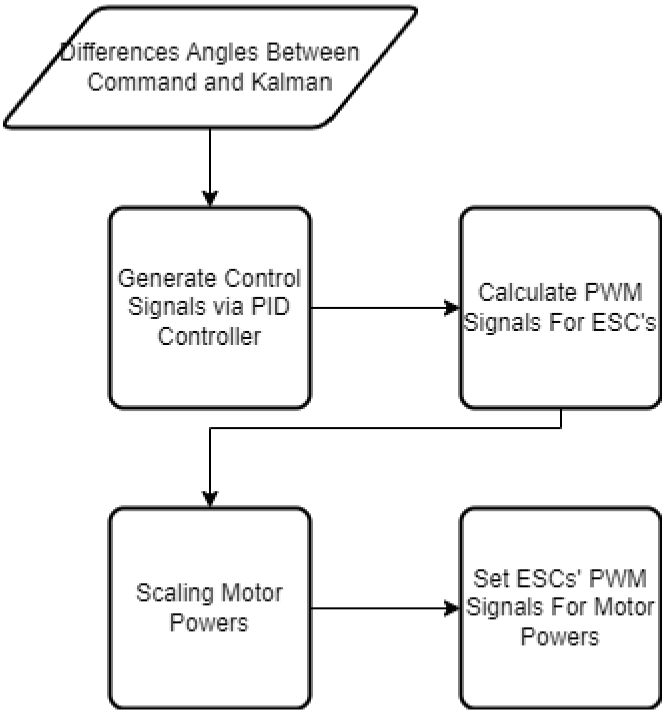

Differences Angles Between Command and Kalman
Generate Control Signals via PID Controller
Calculate PWM Signals For ESC's
Scaling Motor Powers
Set ESCs' PWM Signals For Motor Powers

**Figure 3.14:** Flowchart of calculate motor powers.

After all these calculations, throttle value and PID result are assigned to each motor. Since we will give pwm output to the motors from the pins, we limit the results of the calculations to a function. If this function exceeds the maximum pwm limit, we limit that value to 2000. We are mapping at other engine powers since their ratios to each other must remain the same.

In the next step, we generate PWM signals with 20 milliseconds period by using the stm32 timer clock configuration for the ESC that will give the rotational power to the brushless motors. ESCs accept 1000 microseconds as the minimum value and 2000 microseconds as the maximum value within 20 millisecond periods for brushless motors.

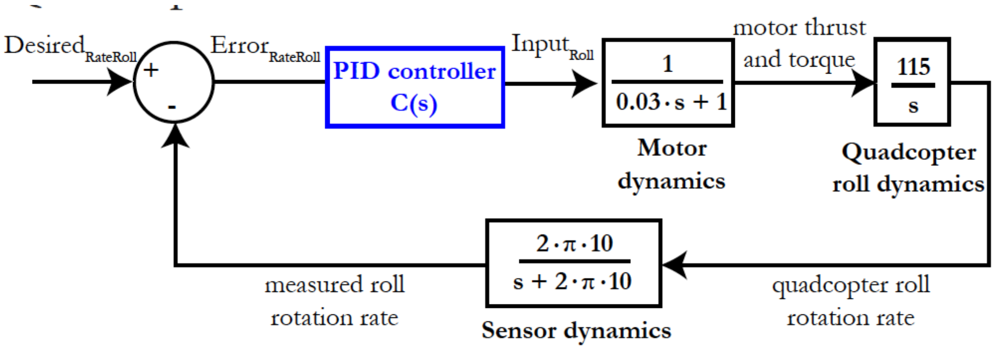
We use the following formula to scale our clock frequency, which is a maximum of 64 megahertz to produce 50 Hertz.

$$Timeclock = \frac{APB\ TIM\ CLOCK}{PRESCALER} \tag{3.13}$$

$$Frequency = \frac{Timeclock}{ARR} \tag{3.14}$$

$$Duty\ \% = \frac{CCRx}{ARR} \times 100 \tag{3.15}$$

For the formulas above, values of the variables for stm32g071RBT6. **APB TIM CLOCK:** 64 MegaHertz, **ARR:** Auto reload register 20000, **CCRx:** ESC input signal interval 1000-2000



In general, in the PID control process, we first move our function in the time domain to the z plane with z transform and then we get the system we want with bilinear transform.

Proportional term: this term provides the calculation of the amount of the control signal based on the size of the difference between the target value and the measured value and creates it. The size of this difference can increase or decrease the amount of the control signal. In this case, the input signal will be less, so we can find out how much response we need to give to the system.

$$P_{term}(k) = P.Error(k) \tag{3.16}$$

z-transform

$$P_{term}(z) = P.Error(z) \tag{3.17}$$

Bilenear transform

$$P_{term}(s) = P.Error(s) \tag{3.18}$$

Integral component: it is used when the measured value differs from the target value for a longer period of time. If we add up the differences in the time period we have determined, we can use this value to generate a control signal. This operation allows us to extract the target more accurately based on a long-term target.

$$I_{term}(k) = I_{term}(k-1) + I.\left(Error(k) + Error(k-1)\right).\frac{T_s}{2} \tag{3.19}$$

Z trasnform

$$I_{term(z)} = z^{-1}.I_{term(z)} + I\left(Error(z) + z^{-1}.Error(z)\right).\frac{T_s}{2} \tag{3.20}$$

$$I_{term(z)} = I.\frac{T_s}{2}.\left(\frac{z+1}{z-1}\right).Error(z) \tag{3.21}$$

Bilear Transform

$$I_{term}(s) = I.\frac{T_s}{2}.\left(\frac{\frac{\left(1+s.\frac{T_s}{2}\right)}{\left(1-s.\frac{T_s}{2}\right)}+1}{\frac{\left(1+s.\frac{T_s}{2}\right)}{\left(1-s.\frac{T_s}{2}\right)}-1}\right).Error(s) \tag{3.22}$$

$$I_{term}(s) = I.\frac{T_s}{2}.\frac{\left(1+s.\frac{T_s}{2}+1-s.\frac{T_s}{2}\right)}{\left(1+s.\frac{T_s}{2}-1+s.\frac{T_s}{2}\right)}.Error(s) \tag{3.23}$$

$$I_{term}(s) = I.\frac{1}{s} \tag{3.24}$$

Derivatives term : Unlike the integral component, this term helps us to perceive changes that occur in a short time. It allows us to calculate how quickly the measured value approaches the value in our target value of change rate. Thus, we can control the speed of the system.

$$D_{term(k)} = +D.\left(Error(k) -. Error(k-1)\right).\frac{1}{T_s} \tag{3.25}$$

Z transform

$$D_{term(z)} = +D.\left(Error(z) - z^{-1}.Error(z)\right).\frac{1}{T_s} \tag{3.26}$$

$$I_{term(z)} = D.\frac{1}{T_s}.\left(\frac{z-1}{z}\right).Error(z) \tag{3.27}$$

Bilinear Transform

$$D_{term}(s) = D.\frac{1}{T_s}.\left(\frac{\frac{\left(1+s.\frac{T_s}{2}\right)}{\left(1-s.\frac{T_s}{2}\right)}-1}{\frac{\left(1+s.\frac{T_s}{2}\right)}{\left(1-s.\frac{T_s}{2}\right)}}\right).Error \qquad (3.28)$$

A mathematical model is a complete mathematical representation of the control system and is usually designed and analyzed using computer simulations. This is a useful tool when examining the properties of control systems because, thanks to mathematical models, we can analyze and optimize the behavior of the system. In this way, we can set parameters (such as PID coefficients) that affect the behavior of components such as the PID controller and make the system behave as desired.

However, it should be noted that the mathematical model is not exactly the same as the real world. In real world applications, there are many factors that affect system behavior and these factors cannot be taken into account in the mathematical model. Therefore, optimizations made on the mathematical model may not give completely accurate results in real-world applications.
Therefore, initial values can be determined based on the results of the mathematical model for tuning the PID coefficients in real-world applications. However, analysis of data from real-world tests and further adjustment of PID coefficients may also be required. This is necessary to compensate for the difference between the mathematical model of the system and its real-world application.

We have now made the mathematical model of our system, but we will use the root lotus method to simulate it.

$$Y(s) = C(s) \cdot G(s) \cdot [U(s) - H(s) \cdot Y(s)] \qquad (3.29)$$

This equation represents the mathematical model of the feedback control system. Y(s) is the system output, C(s) is the transfer function of the controller, G(s) is the transfer function of the process, U(s) is the input signal, H(s) is the feedback factor. This equation shows that the output depends on the input.

$$1+ C(s*) \cdot G(s*) \cdot H(s*) = 0 \qquad (3.30)$$

This equation expresses the equilibrium point of the system. The equilibrium point is the point where the system output does not change and all the variables in the system are constant. This equation is used to determine the properties of the equilibrium point.

$$C(s^*) \cdot G(s^*) \cdot H(s^*) = -1 \qquad (3.31)$$

This equation is another formulation that determines the properties of the equilibrium point. In this case, it means that the product of the transfer functions at the equilibrium point is equal to -1. The system has 3 poles

$$pole\ 0 = 0 \qquad (3.32)$$

$$pole\ 1 = \frac{-1}{0.03} = -33 \qquad (3.33)$$

$$pole\ 2 = -2\pi.10 = -63 \qquad (3.34)$$

$C(s)$ = $K \cdot (s + b)$ with b near zero

$$C(s) = .K.\frac{s+b}{s} \qquad (3.35)$$

$$C(s).H(s).G(s) = \frac{1}{0.03.s+1}.\frac{4.8}{s}.\frac{2\pi.10}{s+2\pi.10}.K.\frac{s+b}{s} \qquad (3.36)$$

Overshoot(OS) =%10
$Settling\ time(t_{settling})$=0.5 s
$The\ dampling\ ratio\ is\ \xi$

$$\xi = \frac{-\ln\frac{OS}{100}}{\sqrt{\pi^2 + \ln(\frac{OS}{100})^2}} = \frac{-\ln\frac{10}{100}}{\sqrt{\pi^2 + \ln(\frac{10}{100})^2}} = 0.59 \qquad (3.37)$$

This equation is used to calculate the damping ratio. OS represents the percentage of overrun in the system. This formula expresses the relationship between the percentage of overshoot and the damping rate.

$$t_{settling=}\frac{-\ln(0.02)}{\xi.\omega_n} \qquad (3.38)$$

This equation is used to calculate the settling time. Settlement time refers to the time it takes for the system output to stabilize within a specified range. The values of ξ and ω_n are used in this formula.

$$\omega_n = \frac{-\ln(0.02)}{(0.59)\ .\ 0.5s} = 13\ rad/s \qquad (3.39)$$

$$P_{desired=} -\omega_n.\xi \pm i.\omega_n.\sqrt{1-\ \xi^2} \qquad (3.40)$$

This equation expresses the roots of a quadratic polynomial. In this case, P(s) represents the polynomial roots of the transfer function. Where ω_n represents the natural frequency, ξ represents the damping ratio.

$$P_{desired=} - 7.8 \pm 10.7 \tag{3.41}$$

Here, "-7.8" represents the real part, while "10.7" represents the complex part. This expression determines the desired characteristic roots of the control system. The "-" sign indicates that the real part is in the negative direction, while the "±" sign indicates that the complex part can take positive and negative values.

$$\sum Q_{poles} - \sum Q_{zeros} = 180^{o} \tag{3.42}$$

$$-Q_1 + Q_2 + Q_3 + Q_4 + Q_5 = 180^{o} \tag{3.43}$$

$$\tan(180^{o} - Q_2) = \frac{10.7}{7.8} \rightarrow Q_2 = Q_5 = 126^{o} \tag{3.44}$$

$$\tan(Q_3) = \frac{10.7}{33 - 7.8} \rightarrow Q_3 = 23^{o} \tag{3.45}$$

$$\tan(Q_4) = \frac{10.7}{63 - 7.8} \rightarrow Q_4 = 11^{o} \tag{3.46}$$

$$Q_1 = 126^{o} + 11^{o} + 23^{o} + 126^{o} - 180^{o} = 106^{o} \tag{3.47}$$

PID controller becomes equal to $C(s) = K(s - 4.7)/s$.

We need to include the magnitude control in our system, for this we will use the following formulas

$$C(s).H(s).G(s) = \frac{1}{0.03.s + 1}.\frac{4.8}{s}.\frac{2\pi.10}{s + 2\pi.10}.K.\frac{s + b}{s} \tag{3.48}$$

$$K_{global} = K.\frac{1}{0.03}.(4.8).(2\pi.10) \tag{3.49}$$

The magnitude condition of the root locus is then equal to :

$$100053.\mathrm{K} = \frac{L_2.L_3.L_4.L_5}{L_1} \tag{3.50}$$

$$K = \frac{\sqrt{7.8^2{+}10.7^2}.\sqrt{(33-7.8)^2{+}10.7^2}.\sqrt{(63-7.8)^2{+}10.7^2}.\sqrt{(7.8)^2{+}10.7^2}}{10053.\sqrt{(7.8-4.7)^2{+}10.7^2}} \tag{3.51}$$

This formula calculates the K value using the system parameters and the magnitude of the roots.

$$C(s) = 2.4.\frac{11.3}{s} \tag{3.52}$$

$$P_{RateYaw} = 2.4 \tag{3.53}$$

$$P_{RateYaw} = 11.3 \tag{3.52}$$

These values are the parameters that determine the behavior and performance of the system of the PI controller. P_RateYaw represents proportional control contribution, while P_RateYaw represents integral control contribution.

## 3.6 Error Handling

### 3.6.1 IWDG (Independent watch dog timer)

As long as the processor allows, the functions required for normal flight are called in a loop as quickly as possible. While reading the values from the sensors, we keep the error information in case of a communication error or a disconnection and define the functions as bool type. When sensor data is not received or an error is returned, the function that calculates the power to be given to the engines for safe flight works and does not allow it to return to normal flight cycle until it receives the information from the sensor in a loop, that is, until it gets rid of the error. Even in the case of the slightest connection going back and forth, the drone seems to have no choice but to give up everything and fall. As a solution to this, we use IWDG (Independent watch dog timer) provided by the microcontroller. With the configuration values for our software architecture we have choosen PR as 256 and RL as 125 to obtain time as 1000 miliseconds. We determine the maximum count of the counter and set it to count backwards by working separately from the processor. The only task of this counter is to count backwards and reset the processor when it reaches zero. We use a function of the HAL library to refresh this counter if it runs well in the loop during normal flight. As we have mentioned before, if the drone goes into a faulty state, the counter, which is deprived of this renewal, tries to restart our processor and tries to provide a possible improvement by making the installations again, while the drone continues to land in safe flight mode. This is not only a hardware failure, but also provides a very useful mode to meet the reset need in case the processor freezes or gets stuck somewhere.

$$Time_{ms} = \frac{(RL + 1) * 1000 * PR}{LSI}$$

**RL:** Reload value, **PR:** Prescalar value, **LSI:** 32KHZ for stm32g071RBT6

### 3.6.2 Interrupt errors

An Interrupt Service Routine (ISR) is a piece of code that is run when the computer receives an interrupt signal from a hardware device or software. The computer stops whatever it is doing and runs the ISR to handle the interrupt. There is an interrupt pin on the MPU6050 sensor, and when this pin is ready to be read, it sends an interrupt every 7 milliseconds according to our measurements. By connecting this interrupt pin to the GPIO pin on the microcontroller, we can call the relevant interrupt service function when triggered.

At first, we were trying to read the sensor data in each loop, when the interrupt came, we realized that there was new available data and calculated the angle. After working for a certain time, communication with the sensor was broken and I2C communication was resetting because it gave an error. After discovering that this may be due to conflict with the interrupt while reading data from the sensor, we threw the function of reading data from the sensor into the interrupt. In this way, when the interrupt function arrives, it does not only notify new available data to activate function that calculate angles, but also reads from the sensor. Since the microcontroller does not read without interrupt, there is no conflict. We have observed this situation for a long time and noted that it works well.

In order to keep the interrupt service under control, we have indicated all our interrupts in order of importance, so that conflicts are prevented when interrupts occur at the same time, and the CPU performs these operations in turn.

# 4. REALISTIC CONSTRAINTS AND CONCLUSIONS

## 4.1 Practical Application of this Project

This study presents an application in the field of control and control of drone movement. Drones are aircraft used in various departments and provide advantages in areas such as construction, agriculture, logistics, security and film production.

### 4.1.1 Social, environmental and economic impact

The use of drones in emergencies and disaster response has the potential to save human lives by providing a quick and effective response. However, in this case people's privacy concerns and regulations regarding the safe use of drones must be taken into account.The proliferation of drones as a popular hobby could increase people's social interactions. Communities can form among drone enthusiasts, and organizing events such as experience sharing, collaboration, and competitions can strengthen social bonds.

Drones offer significant opportunities in the logistics and delivery industry. Drones can reduce logistics costs and delivery times by providing a fast and effective solution for package deliveries. This offers a more efficient operational model for e-commerce companies and courier services.The use of drones in the agricultural sector can increase agricultural productivity. Agricultural drones can perform tasks such as monitoring farmland, assessing plant health, and spraying. This means an increase in agricultural production, more efficient use of resources and cost savings.The use of drone technology in the construction industry can facilitate the monitoring and supervision of projects. Drones can perform tasks such as rapid data collection, mapping and safety inspections at construction sites. This saves time and cost in project management.Commercial use of drones creates new job opportunities and increases employment potential. Employment opportunities arise for experts who manage and maintain drone operations. In addition, new entrepreneurship and business models may emerge with the establishment of companies offering drone-based services.Drone technology also has economic effects in the tourism and advertising industries. Drones contribute to the creation of visual and video content for the promotion of touristic areas. It can also offer services such as aerial footage and live broadcasts for events and festivals.

The use of drones can result in less energy consumption compared to conventional transport, by reducing carbon emissions. The use of drones, especially for small package deliveries, can improve air quality by reducing road traffic. However, issues such as the energy efficiency of drones and waste management should be considered.Drones can be used to observe and protect wildlife. Drones can be used by environmental scientists and conservationists for tasks such as wildlife monitoring and forest fire detection. However, the effects of drones such as interference with fauna and habitat degradation should also be evaluated.

### 4.1.2 Cost analysis

The total hardware cost is 12.000 ₺. In addition, we had the chance to use and test many parts in our test trials in a cheap way using a 3D printer. We worked 2 days in a week on this project.

### 4.1.3 Standards

Standards have been developed to ensure the safe operation of electrical appliances. Turkish Standard EN 60335-1 and IEC 60950-1 determine the general safety requirements for devices such as household appliances and information technology equipment. Standards have been established to ensure safe and regular flight activities in the aviation industry. SHT-İHA.YE.KOT-002 and SHY-5A, determined by the Turkish Civil Aviation General Directorate (SHGM), contain regulations for unmanned aerial vehicles and flight rules in aviation. Standards are used to control electromagnetic emissions of electronic devices and to provide immunity. Turkish Standard EN 55032 and EN 55024 specify requirements for radio frequency emission characteristics and electromagnetic compatibility.

### 4.1.4 Health and safety concerns

Drone Accident Risk Drones have risks of accidents during flight, such as falling or colliding. Drones can cause undesirable situations due to factors such as loss of control, mechanical failures, weather conditions or human error. Such accident situations can pose a potential threat to both drone operators and surrounding people.Data Security and Privacy Violation The proliferation of drones brings with it data security and privacy issues. Drones can collect a wide range of data through the sensors on them. However, it is important that this data is stored, transmitted and protected correctly. Data security breaches can have serious consequences, including disclosure of privacy and trade secrets.Public Safety and Violations Drones can enter sensitive areas or crowded areas and threaten public safety. For example, precautions should be taken against unauthorized drone flights in places such as airports, stadiums or government buildings. In addition, attention should be paid to issues such as air traffic management and control of downed drones.Physical Hazards Drones' propeller and engine systems can pose physical hazards. In case of misuse or operating error, injuries may occur from the contact of the drone propellers. Therefore, appropriate precautions must be taken for the safe use, maintenance and operation of drones.Electricity and Risk of Burns Drones operate on battery systems and battery failures can increase the risk of electric shock or fire. Improper charging, overheating or damaged batteries can compromise the safety of drones and surrounding materials. Therefore, safety protocols for battery use and maintenance should be followed.

## 4.2 Results

As a result, it is possible to make a quadcopter flight controller using the components we mentioned in the thesis. Stable flight of a drone created by combining the motor, esc and power distribution board on a frame can be achieved. We could not perform the flight test because we had problems in the supply of the PCB part, but we tested the filtering of the sensor data and powering the motors through the controller with graphic data and got healthy outputs. We were able to provide a stable drone flight by testing these algorithms with the help of a simulation.

### 4.2.1 Outputs of controller

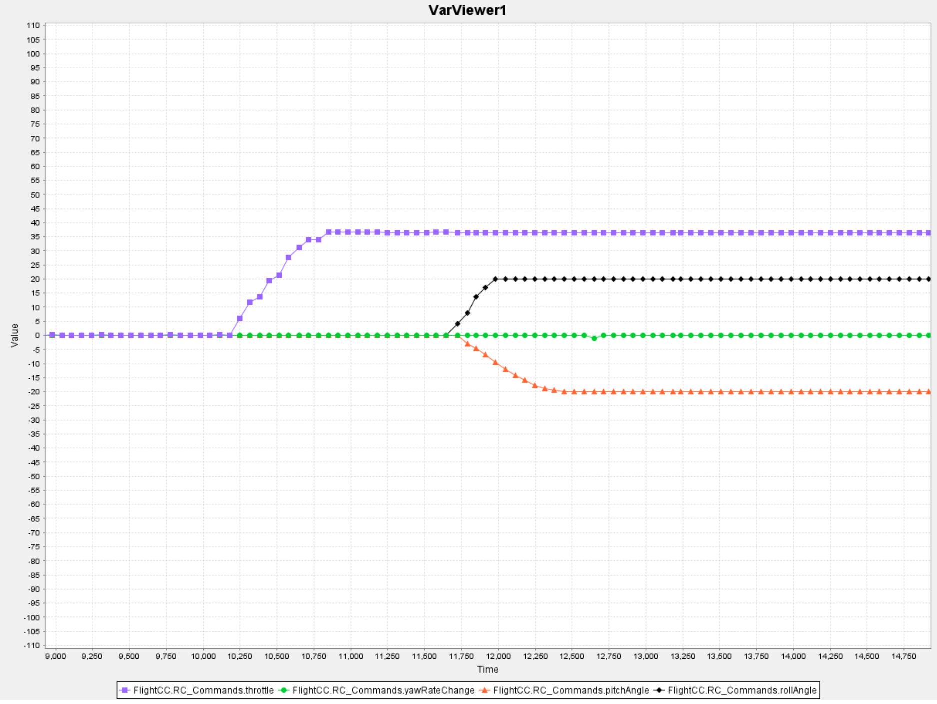
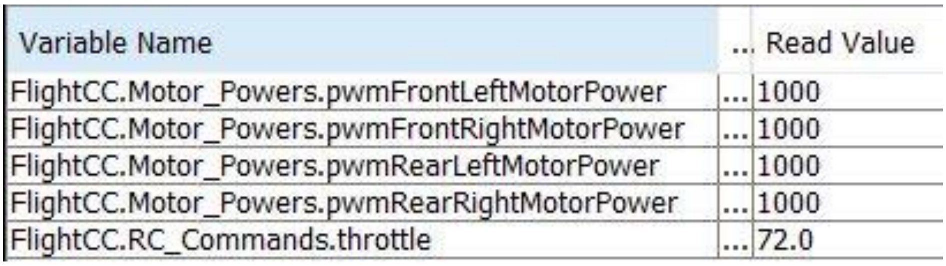


**Figure 4.1:** RC command values.

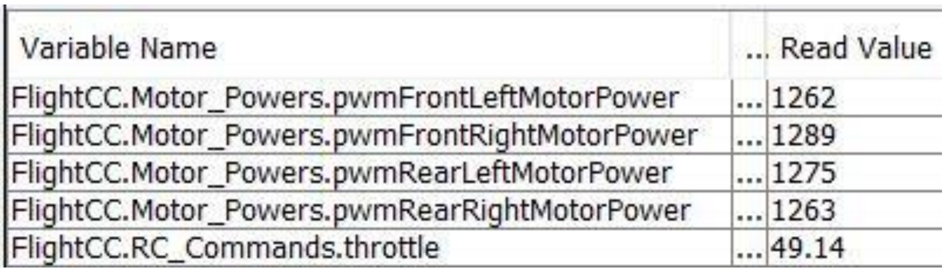
As it is shown in figure 4.1 above. RC Command values are read without loss. We solved the synchronisation problem of PPM signal with an algorithm.



| Variable Name | ... | Read Value |
|---|---|---|
| FlightCC.Motor_Powers.pwmFrontLeftMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmFrontRightMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmRearLeftMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmRearRightMotorPower | ... | 1000 |
| FlightCC.RC_Commands.throttle | ... | 72.0 |

**Figure 4.2:** Disarmed state.

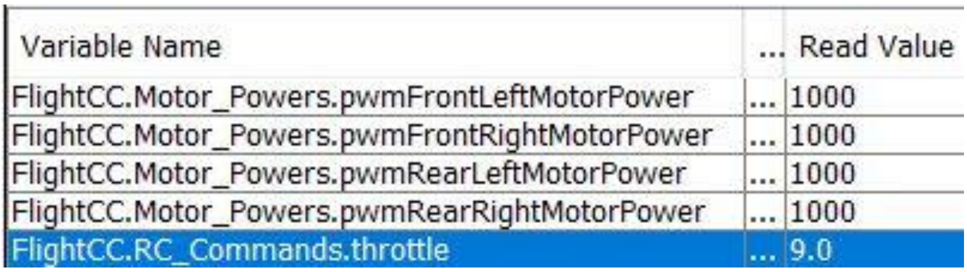
As seen in figure 4.2 when command is disarmed, any of motors does not work.



| Variable Name | ... | Read Value |
|---|---|---|
| FlightCC.Motor_Powers.pwmFrontLeftMotorPower | ... | 1262 |
| FlightCC.Motor_Powers.pwmFrontRightMotorPower | ... | 1289 |
| FlightCC.Motor_Powers.pwmRearLeftMotorPower | ... | 1275 |
| FlightCC.Motor_Powers.pwmRearRightMotorPower | ... | 1263 |
| FlightCC.RC_Commands.throttle | ... | 49.14 |

**Figure 4.3:** Stable flight.

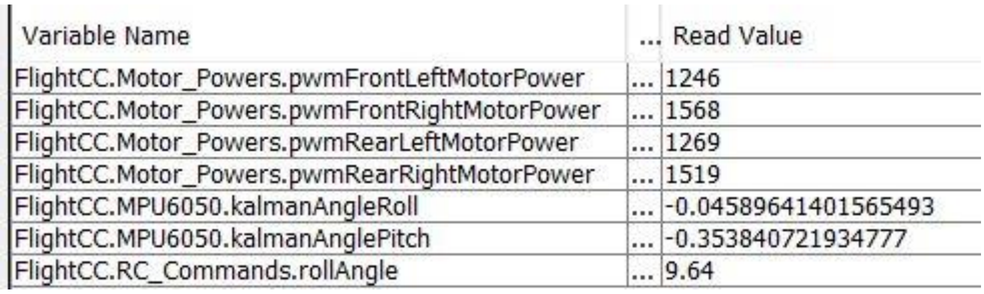
As seen in figure 4.3 when IMU values are fixed, only when it takes throttle and if it is appropriate for flight, the engine powers will be the same for stable flight.



| Variable Name | ... | Read Value |
|---|---|---|
| FlightCC.Motor_Powers.pwmFrontLeftMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmFrontRightMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmRearLeftMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmRearRightMotorPower | ... | 1000 |
| FlightCC.RC_Commands.throttle | ... | 9.0 |

**Figure 4.4:** Safety state.

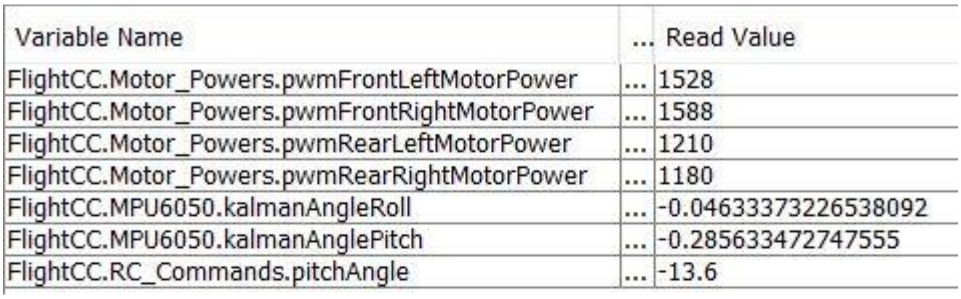
As seen in figure 4.4 when throttle value is less than 10, any of motors does not work.



| Variable Name | ... | Read Value |
|---|---|---|
| FlightCC.Motor_Powers.pwmFrontLeftMotorPower | ... | 1246 |
| FlightCC.Motor_Powers.pwmFrontRightMotorPower | ... | 1568 |
| FlightCC.Motor_Powers.pwmRearLeftMotorPower | ... | 1269 |
| FlightCC.Motor_Powers.pwmRearRightMotorPower | ... | 1519 |
| FlightCC.MPU6050.kalmanAngleRoll | ... | -0.04589641401565493 |
| FlightCC.MPU6050.kalmanAnglePitch | ... | -0.353840721934777 |
| FlightCC.RC_Commands.rollAngle | ... | 9.64 |

**Figure 4.5:** Left roll motion state.

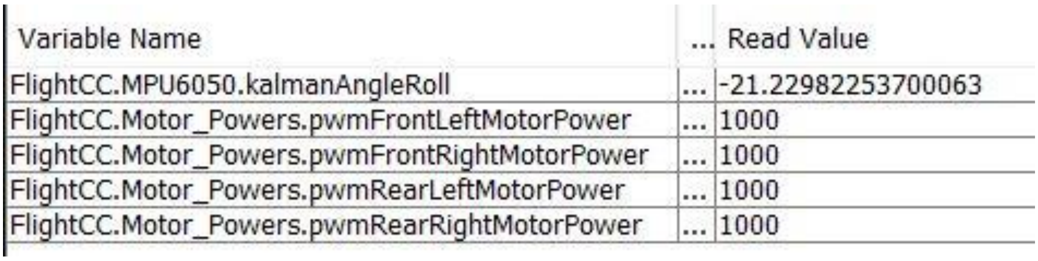
As seen in figure 4.5 when the command is given to make the drone roll to the left, the two motors on the right start to run faster.



| Variable Name | ... | Read Value |
|---|---|---|
| FlightCC.Motor_Powers.pwmFrontLeftMotorPower | ... | 1528 |
| FlightCC.Motor_Powers.pwmFrontRightMotorPower | ... | 1588 |
| FlightCC.Motor_Powers.pwmRearLeftMotorPower | ... | 1210 |
| FlightCC.Motor_Powers.pwmRearRightMotorPower | ... | 1180 |
| FlightCC.MPU6050.kalmanAngleRoll | ... | -0.04633373226538092 |
| FlightCC.MPU6050.kalmanAnglePitch | ... | -0.285633472747555 |
| FlightCC.RC_Commands.pitchAngle | ... | -13.6 |

**Figure 4.6:** Backward motion state.

As seen in figure 4.6 when the command is given to pitch the drone backwards, the two front motors start to run faster.



| Variable Name | ... | Read Value |
|---|---|---|
| FlightCC.MPU6050.kalmanAngleRoll | ... | -21.22982253700063 |
| FlightCC.Motor_Powers.pwmFrontLeftMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmFrontRightMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmRearLeftMotorPower | ... | 1000 |
| FlightCC.Motor_Powers.pwmRearRightMotorPower | ... | 1000 |

**Figure 4.7:** Unsafe angle state.

As seen in figure 4.7 when any angle of roll or pitch is more than 20 degree, any of motors does not work.

### 4.3 Future Work and Recommendations

It is recommended to collect more data on drone movement. Tests performed in different weather conditions, flight scenarios and various speeds can create a more comprehensive data set.

More detailed analysis of data from the cursor sensor (IMU) can be performed. The resulting IMU data can be used to more precisely track motion and develop more precise control algorithms. A more advanced IMU sensor can be used instead of the MPU6050 sensor. An advanced sensor can make more accurate measurements and allow the drone to track its movement more precisely.A more powerful and reliable protocol can be used for communication systems. For example, a faster communication protocol SPI (Serial Peripheral Interface) or a longer range wireless communication protocol can be preferred instead of I2C. More advanced control algorithms can be tried instead of the PID controller. For example, a more sophisticated control algorithm, Fuzzy Logic Controller or Model Predictive Controller (MPC), can be used to increase flight stability.An improved control algorithm can enable the drone to respond faster and more precisely and improve flight performance.

The interface design is designed to change the offset values by writing to the EEPOROM with the UART, so that the IMU offset values can be manipulated from the outside and can be easily configured through the interface without the need to compile the code when the sensor is changed. This will ensure that the offset values are embedded in the correct measurement without making calculations according to the shape it is in at the beginning of each flight.

## REFERENCES

**[1]** “Working Principle and Components of Drone · CFD Flow Engineering,” CFD Flow Engineering, Sep. 27, 2020.

https://cfdflowengineering.com/working-principle-and-components-of-drone/amp/ (accessed Jan. 03, 2023).

**[2]** S. Etigowni, S. Hossain-McKenzie, M. Kazerooni, K. Davis, and S. Zonouz, “Crystal (ball),” Proceedings of the 34th Annual Computer Security Applications Conference, Dec. 2018, doi: 10.1145/3274694.3274724.

**[3]** “Apollo PGNCS,” Wikipedia, Feb. 21, 2022. https://en.wikipedia.org/wiki/Apollo_PGNCS (accessed Dec. 29, 2022).

**[4]** I. Arun Faisal, T. Waluyo Purboyo, and A. Siswo Raharjo Ansori, “A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture,” Journal of Engineering and Applied Sciences, vol. 15, no. 3, pp. 826–829, Nov. 2019, doi: 10.36478/jeasci.2020.826.829.

**[5]** “Coriolis effect | physics | Britannica,” www.britannica.com. https://www.britannica.com/science/Coriolis-effect (accessed Jan. 01, 2023).

**[6]** D. Fedorov, A. Ivoylov, V. Zhmud, and V. Trubin, “Using of Measuring System MPU6050 for the Determination of the Angular Velocities and Linear Accelerations.” [Online]. Available: http://www.jurnal.nips.ru/sites/default/files/A&SE-1-2015-11_0.pdf

**[7]** S. E. Radin Charel, E. H. Binugroho, M. A. Rosyidi, R. S. Dewanto, and D. Pramadihanto, “Kalman filter for angle estimation using dual inertial measurement units on unicycle robot,” IEEE Xplore, Sep. 01, 2016. https://ieeexplore.ieee.org/abstract/document/7861013 (accessed Jan. 03, 2023).

**[8]** G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” 2006. [Online]. Available: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

**[9]** F. Dunn and I. Parberry, "3D Math Primer for Graphics and Game Development," 2nd ed. Boca Raton, FL: CRC Press, 2011.

**[10]** J. Kuffner, "Quaternions and Spatial Rotation," 2004. [Online]. Available: http://www.cs.cmu.edu/~baraff/papers/quatut.pdf. [Accessed: 25-Dec-2022].

**[11]** S. Blozis -I and P. Semiconductors, “DesignCon,” 2003. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN10216.pdf

**[12]** “This is information on a product in full production. STM32F765xx STM32F767xx STM32F768Ax STM32F769xx,” 2021. Accessed: Jan. 04, 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f767zi.pdf

**[13]** Q. M. B. X. L. Cang Liu, «A Flexible Hardware Architecture for Slave Device of I2C Bus,» International Conference on Electronic Engineering and Informatics, 2019.

**[14]** G. Bo, L. Xin, Z. Hui, and W. Ling, “Quadrotor helicopter Attitude Control using cascade PID,” *IEEE Xplore*, May 01, 2016. https://ieeexplore.ieee.org/document/7531919/ (accessed Jan. 04, 2023).

**[15]** M. G. L. Eric Peña, «UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter,» AnologDialogue, p. Vol 54, 4 December 2020.

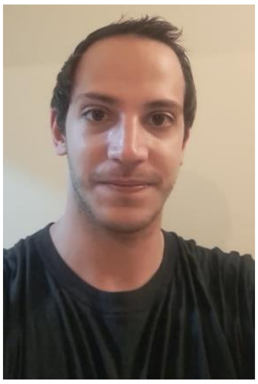
**[16]** InvenSense Inc., “MPU-6000 and MPU-6050 Product Specification Revision 3.4 MPU-6000/MPU-6050 Product Specification,” Aug. 2013. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

## CURRICULUM VITAE

**Name Surname :** Eşrefhan Kadıoğlu

**Place and Date of Birth :** Üsküdar - 12/06/1999

**E-Mail :** kadioglue17@itu.edu.tr



**Name Surname :** Barış Yılmaz

**Place and Date of Birth :** İstanbul - 13/06/1996

**E-Mail :** yilmazba16@itu.edu.tr

| | |
|---|---|
| **Name Surname :** | Ufuk Göktaş |
| **Place and Date of Birth :** | İstanbul - 12/06/1995 |
| **E-Mail :** | goktasu@itu.edu.tr |