FFT GYRO G4.0
USER MANUAL

Initial Setup and Installation

Series 250, 450, 600, 800, 1000, 1200, 1500 and 2000

Version	Date	Comments	Modified by
1.0	20-may-2018	Creation of the document and information update.	Abraham Villanueva
2.0	15-march-2019	Board update and feedback to user.	Abraham Villanueva
3.0	10-april-2019	Changes in the format and operation description.	Abraham Villanueva
4.0	15-april-2019	Correction of information of data rate.	Abraham Villanueva
5.0	11-sep.-2019	Addition of general use of the FFT GYRO system.	Abraham Villanueva
6.0	23-aug-2021	Overall spelling and grammar checkup.	Brianda Valle
6.1	24-sep-2021	Translation and addition of information about the operation of the FFT GYRO.	Brianda Valle
6.2	05-nov-2021	Overall checkup of texts for better understanding.	Brianda Valle
7.0	05-feb-2025	Update to G4.0.	Luis Molina

Contents

Thank you for trusting us by choosing the FFT GYRO G4.0 as a tool to help you implement your development with drones. This manual will guide, you step-by-step on how to use this system to test Unmanned Aerial Vehicles (UAVs).

1 Introduction

Drone innovation requires flight tests during implementation, but the leap from development to implementation can be quite complex, since UAVs have fast and unstable dynamics. Even with state-of-the-art equipment, flight tests are not fail-proof and mid-flight errors likely end in an incident where a drone crashes, is lost, or otherwise malfunctions.

The name FFT GYRO stands for *First Flight Tester Gyroscope*, which is a platform where diverse types of multi-rotors can be supported to perform the first flight tests in a secure and efficient manner. The system provides a great understanding of the aircraft dynamics, which is especially useful when developing and implementing new control laws in vertical take-off and landing aerial vehicles (VTOL).

Its safe structure allows the system to be installed in an open or closed space and interact with people in real time. For example, in a classroom or laboratory for educational and research purposes, allowing students and researchers to have a closer look at the UAV's dynamics and control, performing experiments that resemble real-life drone flights, without the risk of injury or collision.

By using our system, you:

- Protect your team and staff members from accidents.
- Avoid damage of expensive equipment due to collisions.
- Avoid hiring professionals to use the system. Your own staff can use it.
- Accelerate the test phase by analyzing fewer variables (divide and conquer).
- Stop figuring out how new developments are going to be tested.

Having an FFT GYRO G4.0 empowers your team to take that leap from investigation to experimentation, performing projects easier and faster with our patented technology.

Figure 1. First Flight Tester Gyroscope (FFT GYRO G4.0).

2 Components of the FFT GYRO G4.0

The FFT GYRO G4.0 has a similar structure to a gyroscope with three degrees of freedom (3 DOF). The main components of the FFT GYRO G4.0 are identified in Section 2.2 and described in Section 2.3.

2.1 Why the G4.0?

We are always working to improve our products based on user feedback. Some users pointed out that the yaw movement, being the weakest on a drone, had too much inertia. To address this, we simplified the design in the FFT GYRO G4.0 to reduce yaw inertia by changing the order of rotation to Yaw first, then Roll, and finally Pitch (YRP).

Overall, the G4.0 has less inertia, which makes it better for testing drone maneuvers.

2.2 Component Nomenclature

The main components of the FFT GYRO G4.0 are labeled below in Figure 2 and listed in Table 1.

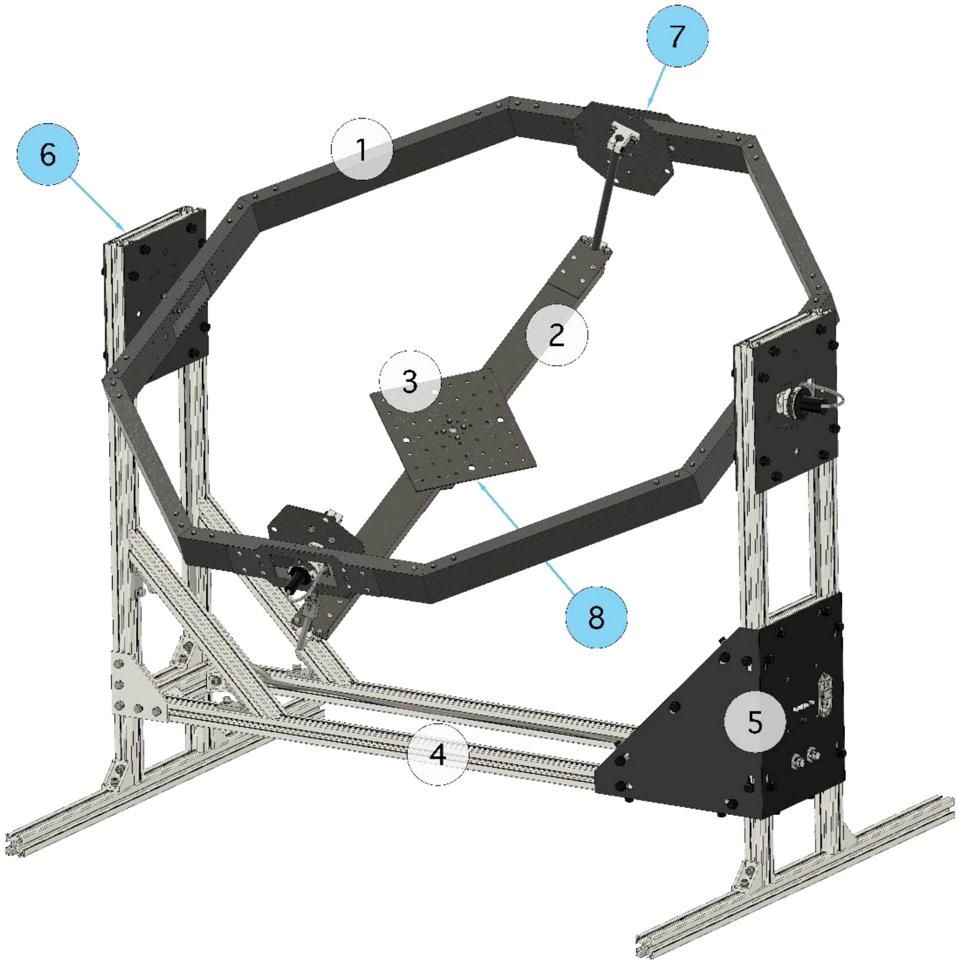


Figure 2. Main components of the FFT GYRO G4.0 system.

NUMBER	COMPONENT
1	Pitch gimbal
2	Roll gimbal: Sliding structure
3	Yaw gimbal: Multi-rotor base
4	Base structure
5	Electronic board: FFT Gyroboard
6	Angular sensor “Pitch”
7	Angular sensor “Roll”
8	Angular sensor “Yaw”

Table 1. Main components of the FFT GYRO G4.0 system.

2.3 Component description

In this section, the functions of the main components will be described to better explain the overall function and operation of the FFT GYRO system. The nomenclature used to describe each component will be the one presented in Table 1.

2.3.1 Base structure

The base structure (**4**) is designed to provide structural support to the entire system. This supporting assembly is composed of aluminum plates and profiles; this grants the system desired properties such as corrosion resistance and a light but strong structure.

The structure delimits the space in which the multi-rotor will move, thus bringing safety to the user and the equipment. In case of more safety requirements, a protection fence[1] can be attached to the aluminum profiles of the structure.

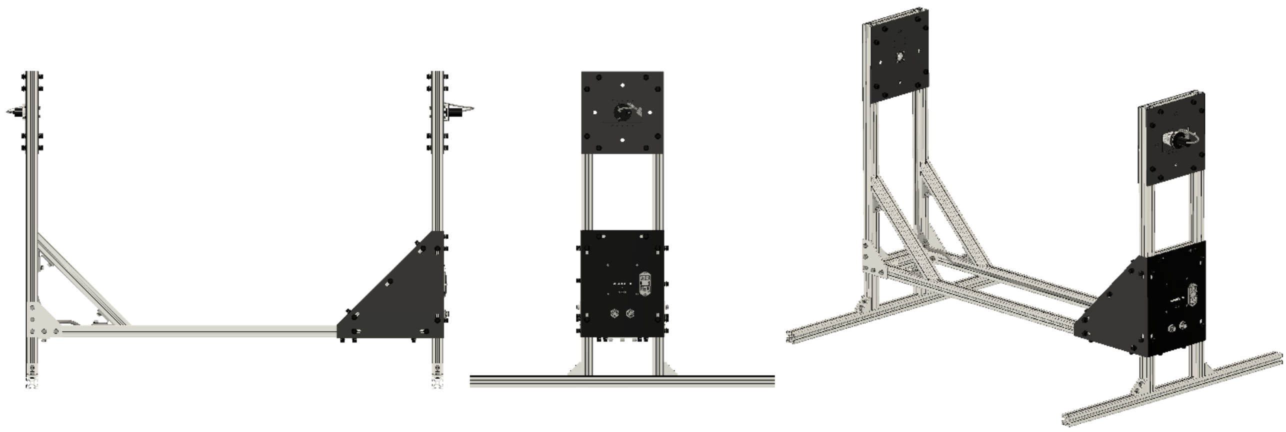


Figure 3. Base structure of the FFT GYRO 4.0 system.

2.3.2 Angular position sensors

The platform has three angular sensors with the purpose of measuring the three degrees of freedom of the system: Yaw (**8**), Roll (**7**) and Pitch (**6**).

The angular sensors (or encoders) are of magnetic type; they do not add friction to the system since there are no parts in contact. The encoders are also of absolute type and multi-spin; the system can measure with high precision (12 to 16 bits) while the multi-rotor spins freely in any direction.

These components are of reduced size and weight to ensure that the extra inertia that the system adds to the multi-rotor is minimal.

Figure 4. Angular sensor or encoder.

The encoders can be replaced by servomotors[2] to generate external forces to the multi-rotor, simulating disturbances like wind or turbulence. The motors cannot be installed simultaneously with the encoders; they are interchangeable with each other. Either there are three encoders or three motors, there is no intermediate configuration.

When the encoders are installed, the system has extremely low friction or "free movement". This configuration is exceptionally good for evaluating the dynamics of the aircraft, calibrating controllers, or simulating as if the drone is "flying freely".

When the motors are installed, the system does not have free movement, as the motors have a gear train that adds a considerable load to the system. This configuration is ideal for evaluating the robustness of the controllers in the presence of external forces, which are simulated by the motors themselves.

The sensors and actuators are connected through an internal wiring system to the FFT Gyroboard card, which has special firmware and drivers to communicate to a PC via USB.

2.3.3 Pitch gimbal

The octagonal structure is designed to maximize the structural strength of the system while maintaining the less weight possible. These components are built with aluminum and carbon fiber machined parts.

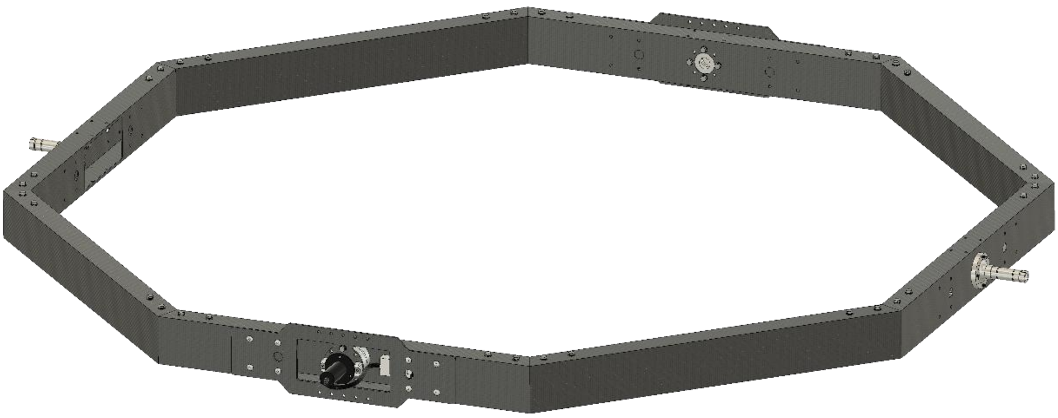


Figure 5. Pitch structure (1).

[1]Protection fence sold separately.
[2]Motors are sold separately.

The Pitch gimbal (**1**) is attached to the base frame (**4**) by two low-friction rotary joints. In one of these junctions the sensor (**6**) is located to measure the Pitch angle, while at the opposite end it has a slip-ring that allows the wired communication of the sensors to the FFT Gyroboard.

The Pitch gimbal (**G2**) is clamped to **G1** in the same way that **G1** is clamped to **F1**, but this junction uses the **S2** sensor to measure the Pitch angle.

2.3.4 Roll gimbal

The roll gimbal is made of either aluminum or carbon fiber and is attached to two sliding tubes connected to the circular structure by rotary joints. The sliding structure helps adjust the height of the platform where the multi-rotor is held to align the rotation center of the multi-rotor with the rotation center of the FFT GYRO system. In one of these joints, the sensor (**6**) is located, to measure the Roll angle, while at the opposite end a slip-ring allows wired communication.

Figure 6. Roll Gimbal (2).

2.3.5 Yaw Gimbal

The multi-rotor base (**2**) is designed to support several types of multi-rotors, both in configuration and size.

2.3.5.1 Yaw Gimbal in 250-800 series.

The 250-800 series include three different plates for your preference. Holes in the plates enable to install grips to mount a drone.

Figure 7. Yaw Gimbal—series 250-800 (3): Custom, Quad, and Hexa plates (from left to right).

2.3.5.2 Yaw Gimbal in 1000-1500 and 2000 series.

Sizes between 1000 and 2000 support larger and heavier drones, this requires a stronger frame. The Yaw gimbal is built with carbon fiber tubes and plates.

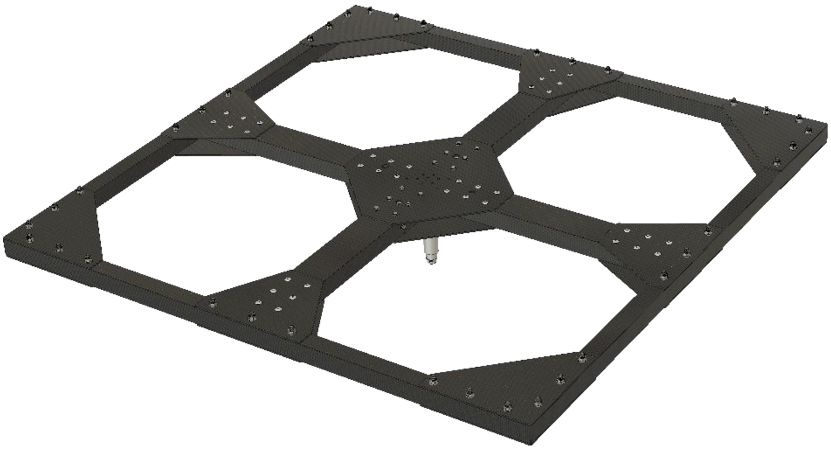


Figure 8. Yaw Gimbal—series 1000-1500 and 2000.

2.3.6 FFT Gyroboard

The FFT GYRO G4.0 system is composed of three angular position sensors (replaceable by servomotors) connected to the data acquisition board via USB as a communication interface with the PC. This data acquisition board is the FFT Gyroboard.

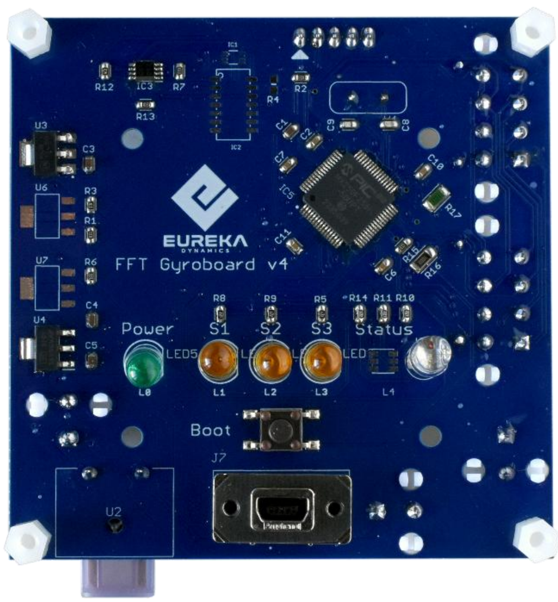


Figure 9. FFT Gyroboard (5).

The system can be used with an open-source, multi-platform and very simple interface called "FFT GYRO Test Tool" that can be downloaded [here](#).

MATLAB/Simulink programs are also included in the FFT GYRO system, for reading and writing of the encoders and motors. On section 6, the communication protocol is explained in detail for developers that wish to implement their own applications.

2.4 Dimensions and main parameters

In this section, the main dimensions and parameters of the FFT GYRO G4.0 system are presented. Dimensions are expressed using the International System of Units (SI).

The FFT GYRO's frame is composed of aluminum or carbon fiber. The material selection allows a rigid structure, safe enough to support the drone, and light enough to minimize the inertia effects on the performance of the UAV.

To accommodate different drone sizes and types, the FFT GYRO G4.0 comes in multiple sizes. Each size is designed to support drones within the same category (similar size and weight). Based on standard commercial drone size-to-weight ratios, the available series include: 250, 450, 600, 800, 1000, 1200, 1500, and 2000. Below are the key dimensions and parameters for each series.

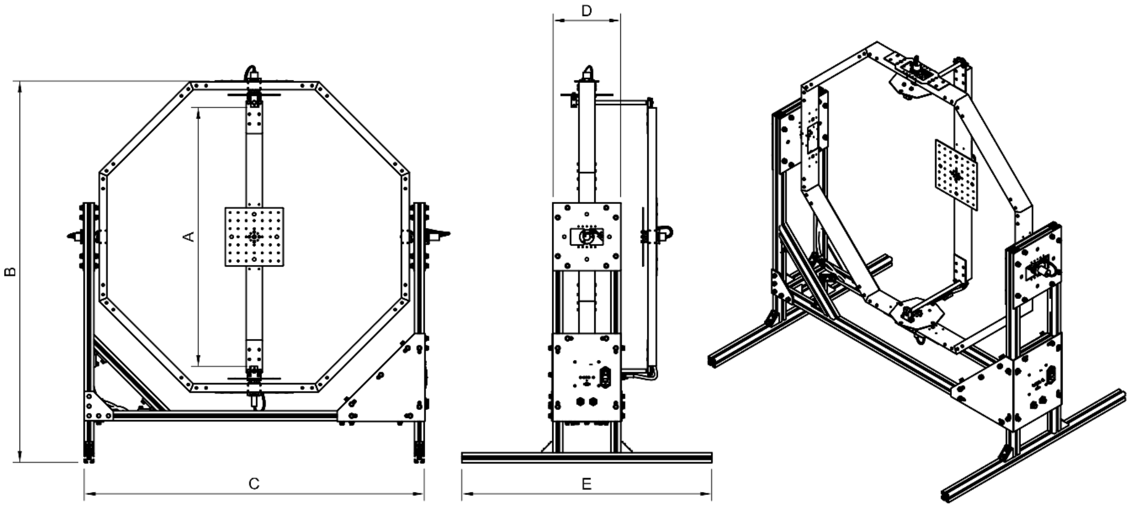


Figure 10. Principal dimensions of the FFT GYRO G4.0.

The principal mechanical parameters are shown in the following tables:

250-800 SERIES	FFT GYRO 250	FFT GYRO 450	FFT GYRO 600	FFT GYRO 800
A	502 mm	802 mm	1002 mm	1252 mm
B	883 mm	1183 mm	1383 mm	1633 mm
C	754 mm	1054 mm	1254 mm	1504 mm
D	210 mm	210 mm	210 mm	210 mm
E	625 mm	775 mm	875 mm	1000 mm
Compatible multi-rotor sizes	100 mm to 250 mm rotor-to-rotor distance or up to 450 mm drone's outer diameter with unfolded propellers.	250 mm to 450 mm rotor-to-rotor distance or up to 750 mm drone's outer diameter with unfolded propellers.	450 mm to 600 mm rotor-to-rotor distance or up to 1000 mm drone's outer diameter with unfolded propellers.	600 mm to 800 mm rotor-to-rotor distance or up to 1200 mm drone's outer diameter with unfolded propellers.
Roll, pitch, and yaw range	*360° (multi-rotations).*	*360° (multi-rotations).*	*360° (multi-rotations).*	*360° (multi-rotations).*

Table 2. Main parameters of the FFT GYRO G4.0 —series 250-800.

1000-1500 SERIES	FFT GYRO 1000	FFT GYRO 1200	FFT GYRO 1500
A	1501 mm	1801 mm	2201 mm
B	1823 mm	2123 mm	2523 mm
C	1803 mm	2103 mm	2503 mm
D	210 mm	210 mm	210 mm
E	1655 mm	1955 mm	2355 mm
Compatible multi-rotor sizes	800 mm to 1000 mm rotor-to-rotor distance or up to 1500 mm drone's outer diameter with unfolded propellers	1000 mm to 1200 mm rotor-to-rotor distance or up to 1800 mm drone's outer diameter with unfolded propellers	1200 mm to 1500 mm rotor-to-rotor distance or up to 2050 mm drone's outer diameter with unfolded propellers
Roll, pitch, and yaw range	360° (multi-rotations).	360° (multi-rotations).	360° (multi-rotations).

Table 3. Main parameters of the FFT GYRO G4.0 —series 1000-1500.

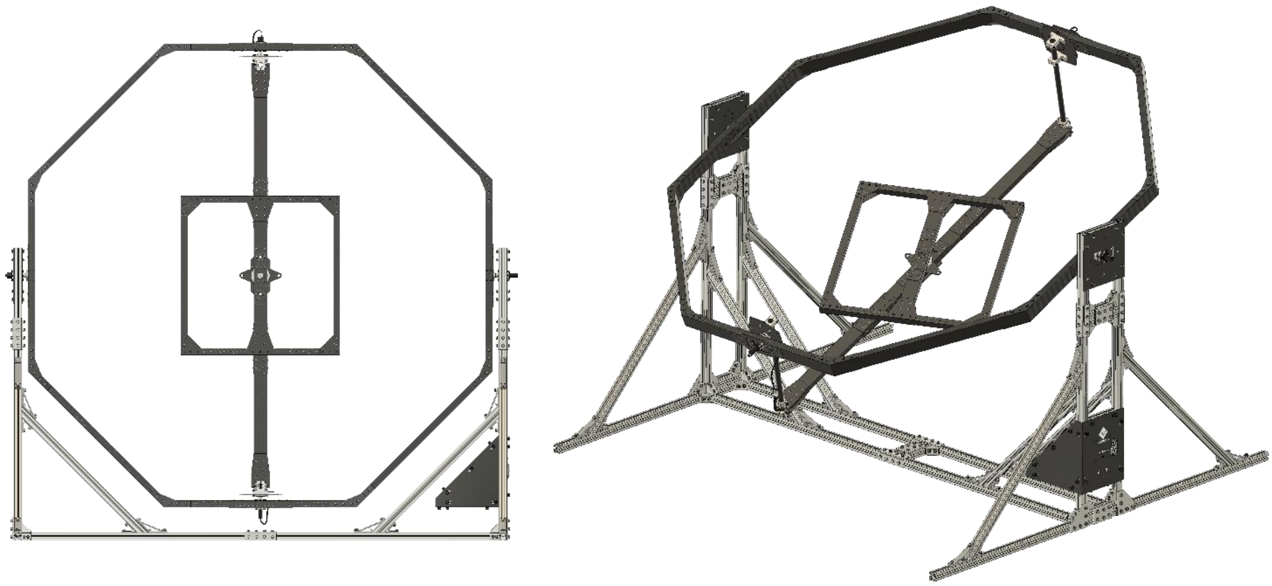


Figure 11. FFT GYRO G4.0—series 1000-1500.

2000 SERIES	FFT GYRO 2000
A	2801 mm
B	3186 mm
C	3179 mm
D	230 mm
E	2977 mm
Compatible multi-rotor sizes	1500 mm to 2000 mm rotor-to-rotor distance or up to 2800 mm drone's outer diameter with unfolded propellers
Roll, pitch, and yaw range	360° (multi-rotations).

Table 4. Main parameters of the FFT GYRO G4.0 —series 2000.

	STANDARD	PRO	PRO MOTORS 2.5	PRO MOTORS 6.0
Encoder Resolution	-	12 or 16 bits resolution (0.087° or 0.0055°)		
Power Capacity Specifications	-	12 VDC, 10A max current, 120W max.		
Communication Protocols	-	USB 2.0, serial port, acquisition rate of 100 Hz.		

Table 5. Electrical specifications of the FFT GYRO G4.0.

3 System installation

3.1 Unpacking and assembly

The system is shipped unassembled. The package content and the steps to fully assemble the FFT GYRO G4.0 are linked below, or request them to us at info@eurekadynamics.com.

- FFT GYRO G4.0 —series 250-800 Assembly Manual —or watch the video.
- FFT GYRO G4.0 —series 1000-1500 Assembly Manual —or watch the video.
- FFT GYRO G4.0 —series 2000 Assembly Manual —or watch the video.

3.2 General considerations

It is recommended to install the FFT GYRO system in a space with good ventilation. It is also recommended to place it on a table with stability and free space on all four sides, so that air can flow easily.

It is recommended to have an electrical contact (100-250VAC, 50-60 HZ) close to where the system is going to be installed, as well as a place where to place a PC and its peripherals.

⚠ It is necessary to place a regulation and overload protection device to protect the integrity of the electronic system of the FFT GYRO G4.0.

⚠ When using the system with a multi-rotor, the introduction of any object into the internal area of the system is not allowed, for the safety of everyone and the device.

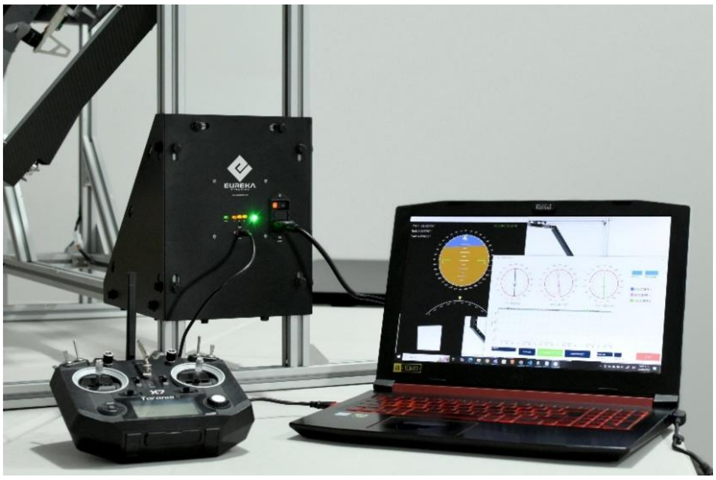


Figure 12. Example of the FFT GYRO G4.0 on a table, connected to a PC.

3.3 Component verification

Once the system is installed, evaluate the mobility of the three degrees of freedom by unlocking the gimbals and manually moving each one without being connected to a power supply. This is just to verify the free rotation of G1, G2 and G3. Each gimbal must rotate smoothly in both ways and should not have any physical obstruction that prevents free rotation (test multi-roll movements in both directions).

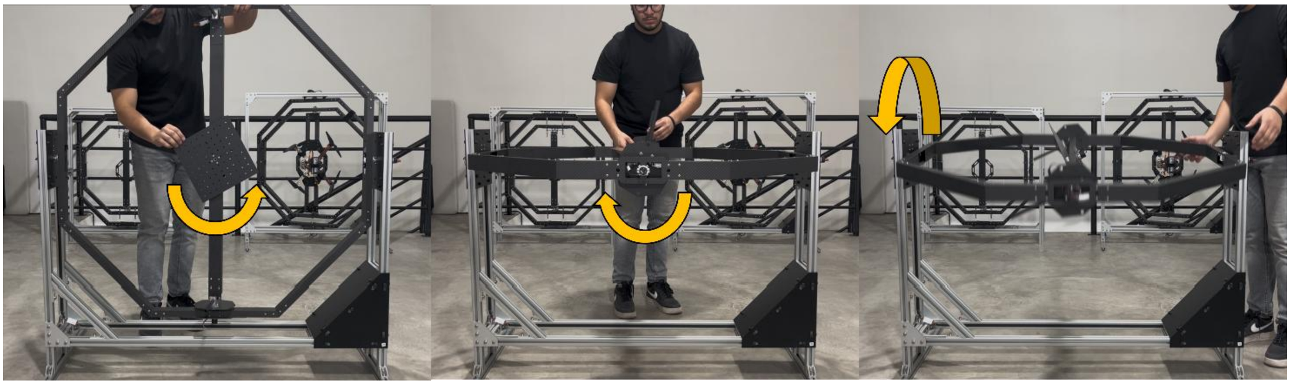


Figure 13. Movement verification of Yaw, Roll and Pitch movements (from left to right).

If the three degrees of freedom can move freely, then the system is suitable for use. Otherwise, if any of the gimbal structures presents an obstruction to movement, please contact us at info@eurekadynamics.com.

4 Software settings

In this section, the requirements, considerations, and instructions will be described to install and use the programs that communicate with the FFT GYRO system.

4.1 General settings

First, it should be mentioned that the communication of the FFT Gyroboard electronic card with the PC is through the serial (USB) port. The firmware driver is a generic USB type, compatible with most computer equipment.

Eureka Dynamics has developed a graphic interphase to interact with the FFT GYRO, which is called “FFT GYRO Test Tool”. This tool is compatible with Windows, OS, and Linux. The processing requirements are minimum since the communication protocol is quite simple.

Requirements for the FFT GYRO Test tool:

- One of the following operational systems: Windows 7 and above, MacOS X 10.8.5 and above, or Ubuntu Linux 12.04 and above.
- 2 GB of RAM as minimum.
- 1 GB of storage available.
- 1 USB 2.0 port.

Once you have all these requirements, you can start with the system connection.

Connect the mini-USB cable end to the FFT Gyroboard, and the USB end in an available port in your PC.

Figure 14. Mini-USB cable connected to the FFT GYRO’s front panel.

In the PC (Windows) you can verify if the device has been recognized by accessing the Device Manager and look for the COM Ports section and identifying which COM has been assigned to the FFT Gyroboard, as an example, we will assume that the system has been identified in the COM 9.

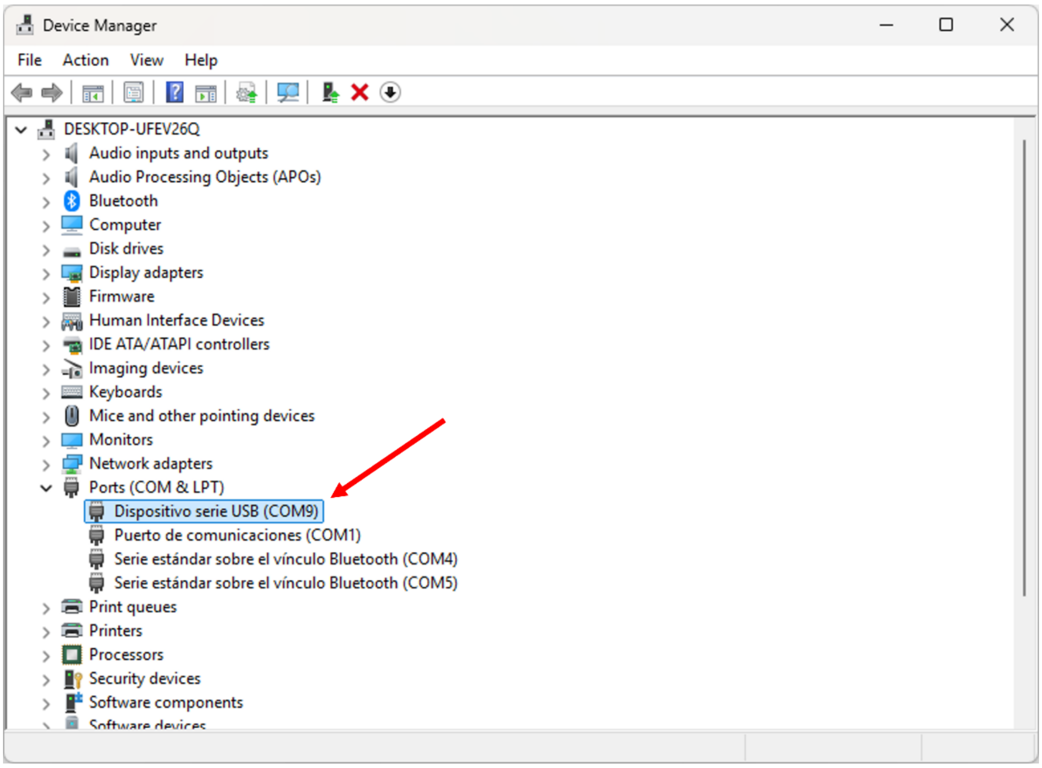


Figure 15. Recognized COM port in Device Manager (Windows).

4.2 FFT GYRO Test tool

The FFT GYRO Test Tool was developed to connect to the FFT GYRO and help users understand how a particular UAV performs. It offers a variety of commands and charts to control and visualize the incoming data.

There are two ways of using the FFT GYRO Test Tool, as a user or as a developer.

4.2.1 Requirements to run the executable file

Exe files were compiled for different operating systems such as windows 32 and 64 bytes, and Linux with different kernel architectures. Please verify the distribution and architecture of your computer, and stick to the recommendations made here to ensure a proper operation. If any issue comes up with an exe file, we suggest you to open the program as a developer with Processing, compile the code, and make the exe file using the option File/Export Application.

To run the FFT GYRO Test Tool as a developer is necessary to install Processing 3.0 version or higher, and before running the code, it is also mandatory to install the graphical library available in Tools/Add Tool/Library. This will be explained later.

4.2.2 Requirements to run FFT GYRO Test tool source files

In all cases, requirements to install Processing are:

- **Windows:** Windows 7 or above.
- **Ubuntu/Linux:** Compatible with Linux and it is recommended to use Ubuntu version 14.04 or above.

4.2.3 Quick Installation

1. Download the source code files here. Follow the steps in figure 16.
2. Decompress the file and access the folder, named "fft-gyro-master".
3. Once inside the folder, go to the following direction (for this example, we will assume that the designated operating system is Windows 11): fft-gyro-master\FFTGYROTestTool\windows-amd64
4. Locate the exe file called "FFTGYROTestTool", select it, and press enter.
5. As the file opens, a menu will display two options. Click option *V4*, then click *CONTINUE*, as in figure 17.
6. Three windows will be displayed. Locate the Main Control Window, it should be totally blank, as in figure 18.
7. Next, plug the FFT GYRO (using the included USB cable) to the computer, push the *REFRESH* button to update the ports available on the list. On that list, select the port corresponding to the FFT GYRO, and click *CONNECT*. If everything is ok, you will see the *CONNECT* button turn green, and the label will display *CONNECTED*. Now you will be receiving data from the FFT GYRO.

For more information about this, please go to section 6.2.

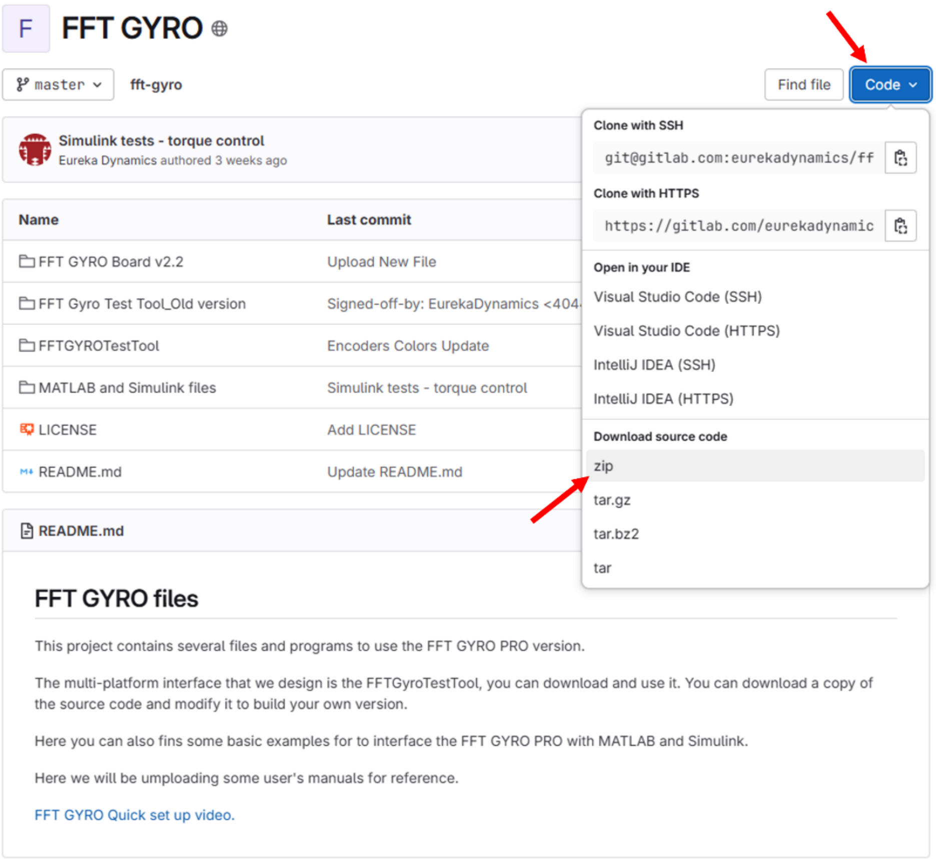


Figure 16. Steps to download the Zip file: Go to the link in GitLab, click on “Code”, then click on “zip”.

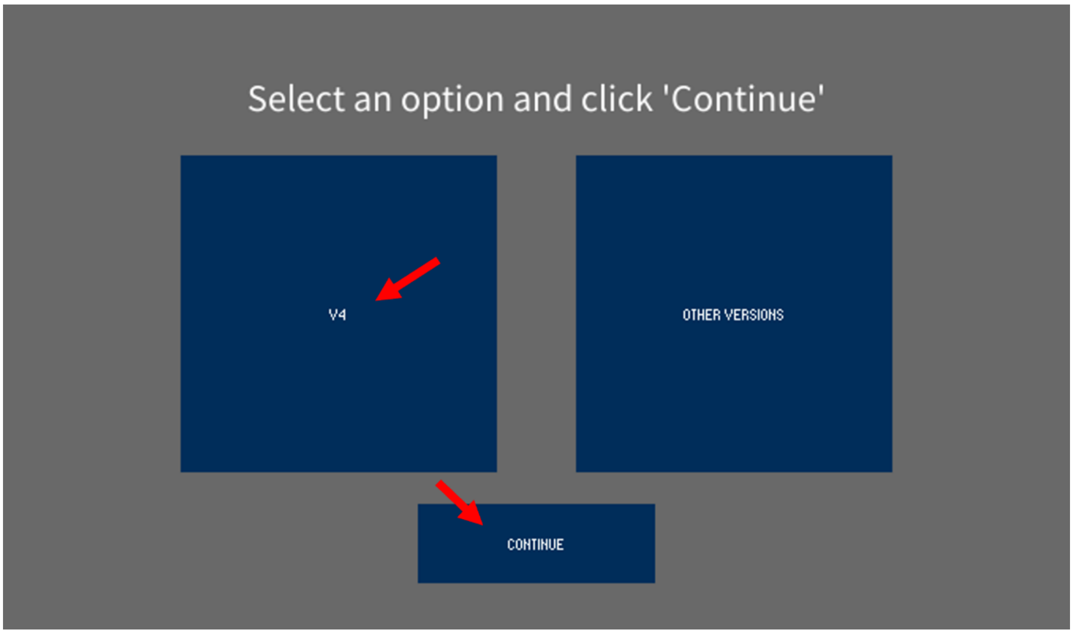


Figure 17. As the exe file opens, click “V4”, then click “CONTINUE”.

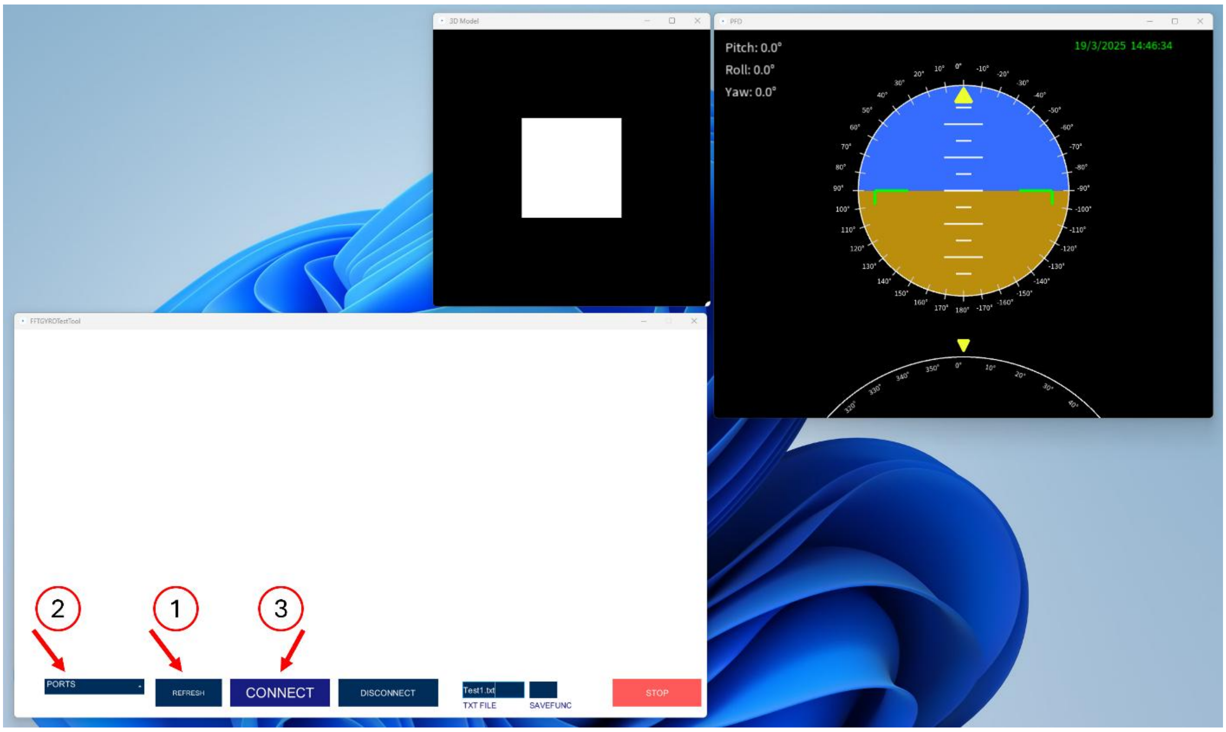


Figure 18. Three windows are displayed. Steps to connect the FFT GYRO: Click "REFRESH", select the port, click on "CONNECT".

4.3 MATLAB

In this section we review the requirements to run MATLAB samples and Processing code. In both cases (samples and codes) they were executed and tested in Windows 64 bits, but they should run without issues in other operating systems like MacOS or Ubuntu, as long as MATLAB and Processing are compatible.

4.3.1 Requirements for Matlab

Scripts have also been developed in MATLAB and Simulink for reading/writing in the FFT Gyroboard. To run Matlab Samples (Scripts and Simulink sample) you will need Matlab 2017 or further versions (licenses are sold separately). Codes were tested using 2017 on Windows 10 64 bits.

MATLAB/Simulink Requirements:

(Necessary)

- MATLAB 2017 and above.
- SIMULINK 2017 and above.
- Instrument Control Toolbox.

(Recommended)

- Signal Processing Toolbox
- Data Acquisition Toolbox

As for the PC requirements for MATLAB/Simulink, the configuration is not strict. It is sufficient if the PC can run Matlab. All you need is a free USB port.

For more information about this, please go to section 6.7.

4.4 Simulink

1. Open MATLAB 2017 or further.
2. Go to the following direction **fft-gyro-master\MATLAB and Simulink files**, and you will see 2 files, ***SimulinkSampleEncoder.slx*** and ***SimulinkSampleMotor.slx***.
    a. ***SimulinkSampleEncoder***: Before the start of the simulation make sure to select the correct port in Serial configuration block and Serial Receive block. Push the Run button located at the top of the Simulink Environment.
    b. ***SimulinkSampleMotor***: In the same way as in the ***SimulinkSampleEncoder*** file, you must be sure to set the proper configuration of the Serial Configuration Block. Once you are sure the serial configuration is correct, in this sample you can found all blocks you can use whether you want to set velocity, set position or read the PORT to receive the information. Finally push the button Run to start the simulation.

5 Operation of the FFT GYRO

In this section, the basic operation of the FFT GYRO will be described. This section is also explained in the following video.

5.1 Turning-on the FFT GYRO

Start by plugging and securing the power supply, then turn on the power switch. Now the system will be set to connection standby, which means it is waiting for the USB cable to be connected to the FFT GYRO and the computer.

When the FFT GYRO is successfully connected to the computer, the green and yellow LEDs will be lit, indicating that the system is energized and ready to operate with the software.

Figure 19. Turning-on the FFT GYRO.

5.2 Communication with PC

After connecting the system to a power supply and to the computer, the FFT GYRO Test Tool can be opened. Follow the steps in 4.2.3 to start the FFT GYRO Test Tool. Once the steps are completed, you can start reading encoders right away. You may manually move each gimbal to see the angular readings in the display.

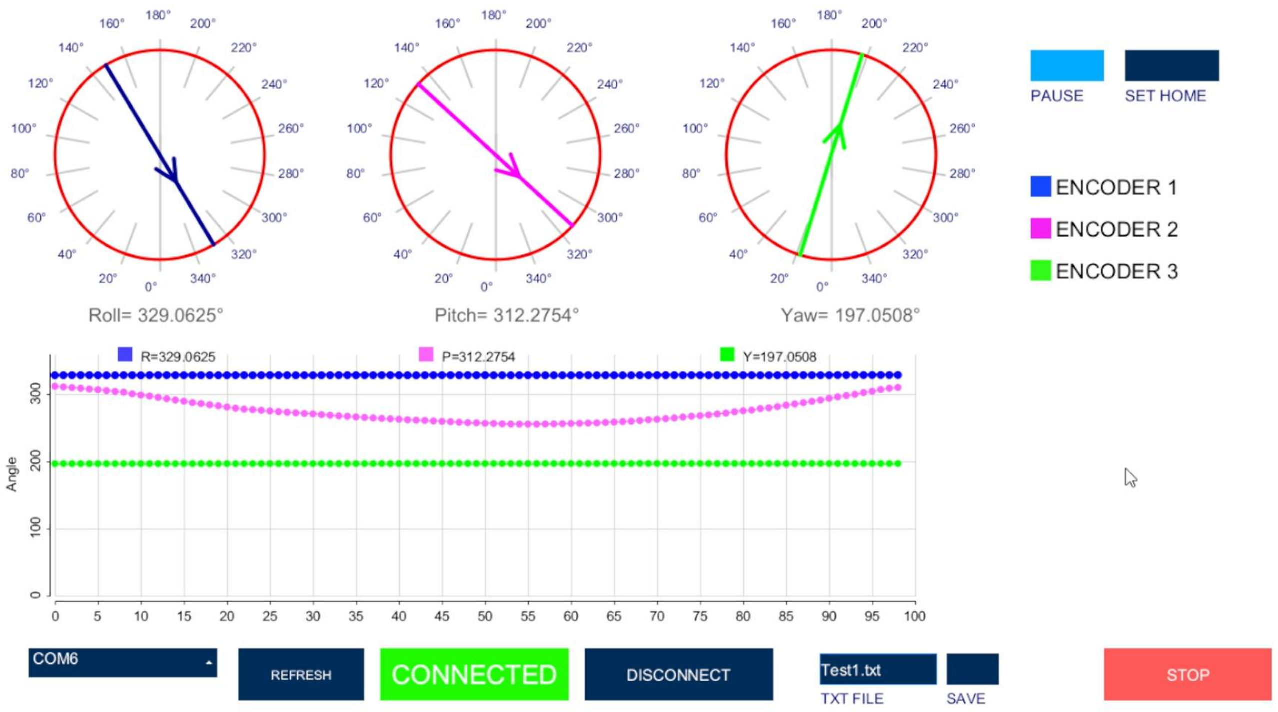


Figure 20. FFT GYRO Test Tool displaying encoder readings.

5.3 Multi-rotor installation in FFT GYRO base.

Start by installing the included grip standoffs to the YAW plate using a 3mm Allen key. Once secured, position the drone to align each rotor arm with a grip, and use the included Velcro straps to firmly fasten in place.

To learn how to balance a drone in the FFT GYRO G4.0 look at the following video.

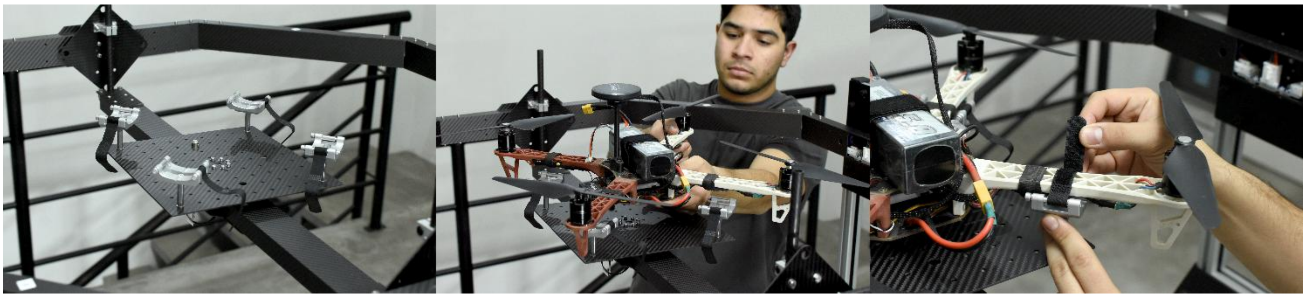


Figure 21. Steps to Mount a Drone: Install grips to the plate, place the drone, and secure straps (from left to right).

5.4 Sensor calibration to relative zero.

In the FFT GYRO Test Tool screen display, set home position for encoders. Encoder readings usually don't match onboard sensors. To compensate for the mismatch, turn on the multirotor in stabilized mode, wait until stabilization and then click the *Set Home* button.

5.5 Data acquisition and sensor reading

For encoder readings in MATLAB open the *SampleReadEncoder.m* file and match the communication ports. Then proceed to run the script.

For encoder readings in Simulink open the *SimulinkSampleEncoder.slx* file, configure serial configuration block to match the actual port and click run.

5.6 Motor Installation Steps

First, align the gimbals at home position (0° labels should match) and lock rotation with locking pins. Uninstall encoders by disconnecting the encoder cable carefully. Use a 2mm Allen key to remove the encoder mount. Use a 1.5mm key to unscrew the set screw, and finally, remove the axis magnet. Repeat the same process with every encoder.

Install the motor, every tag must match. Once the motor is installed, adjust the motor disk set screw using a 1.5mm key, the set screw must be very tight. Finally, connect the motor cable.

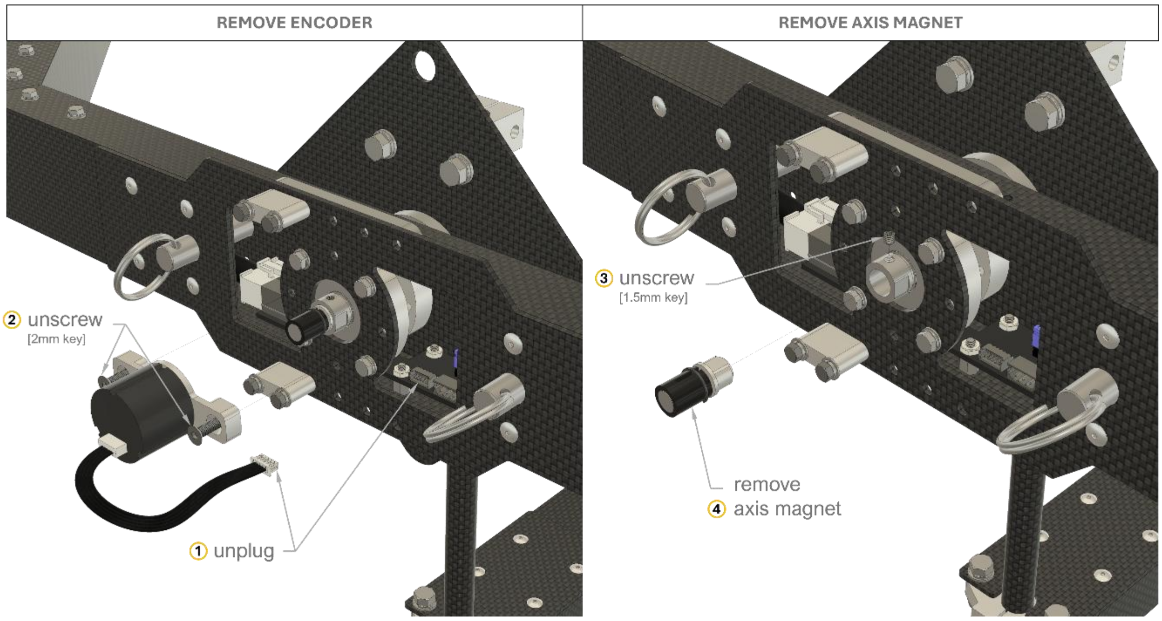


Figure 22. Prepare for motor installation: Steps to remove the encoder.

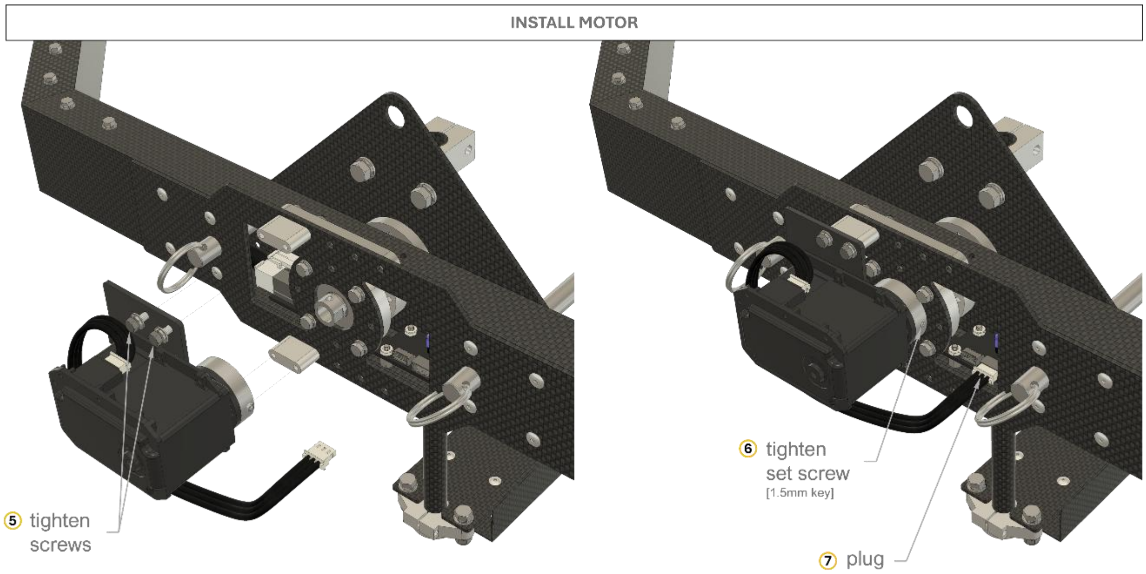


Figure 23. Motor installation steps.

5.7 Motors with the FFT GYRO Test Tool

Click refresh, select port, and click connect. Please be careful when motors are installed. We recommend limiting the torque to 30% in the first trials. Click the Set Torque Limit button to confirm. Click STOP to turn off the motors. You can use the spacebar as a safety stop button.

6 Support – FFT Gyroboard v2.2

6.1 Technical Specifications

FFT Gyroboard v2.2
✓ Sensors: Absolute Magnetic Encoders: ▪ 5 volts ▪ 12-bit resolution Digital motor capable of reading applied torque (12 V Operation) ▪ SSI Interface ▪ 10-bit resolution to measure torque
✓ 40 MHz clock system.
✓ Updatable firmware.
✓ Protocol USB 2.0 (USB Mini B).
✓ Auto select mode. The system automatically selects the mode of operation when three sensors of the same kind are connected.
✓ 12-15 DC volts operation with external energy source.
✓ PCB size: 9.8 x 5.1 cm.
✓ Fuse protection of 2 Amp.
✓ Variable sensor sampling, from 10 milliseconds to 2.5 seconds (Default: 10 ms, steps of 10 ms)

Table 5. Technical specifications of the FFT GYRO Gyroboard v2.2.

6.1.1 Schematic diagram of the system

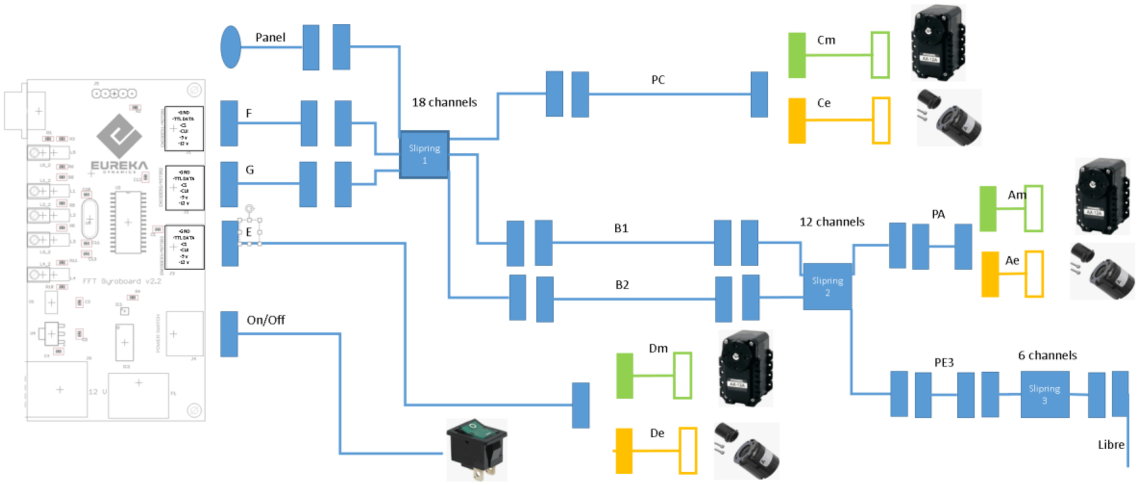


Figure 22. Schematic diagram of the FFT GYRO Gyroboard v2.2 system.

6.1.2 Description of the LEDs of the FFT Gyroboard v2.2

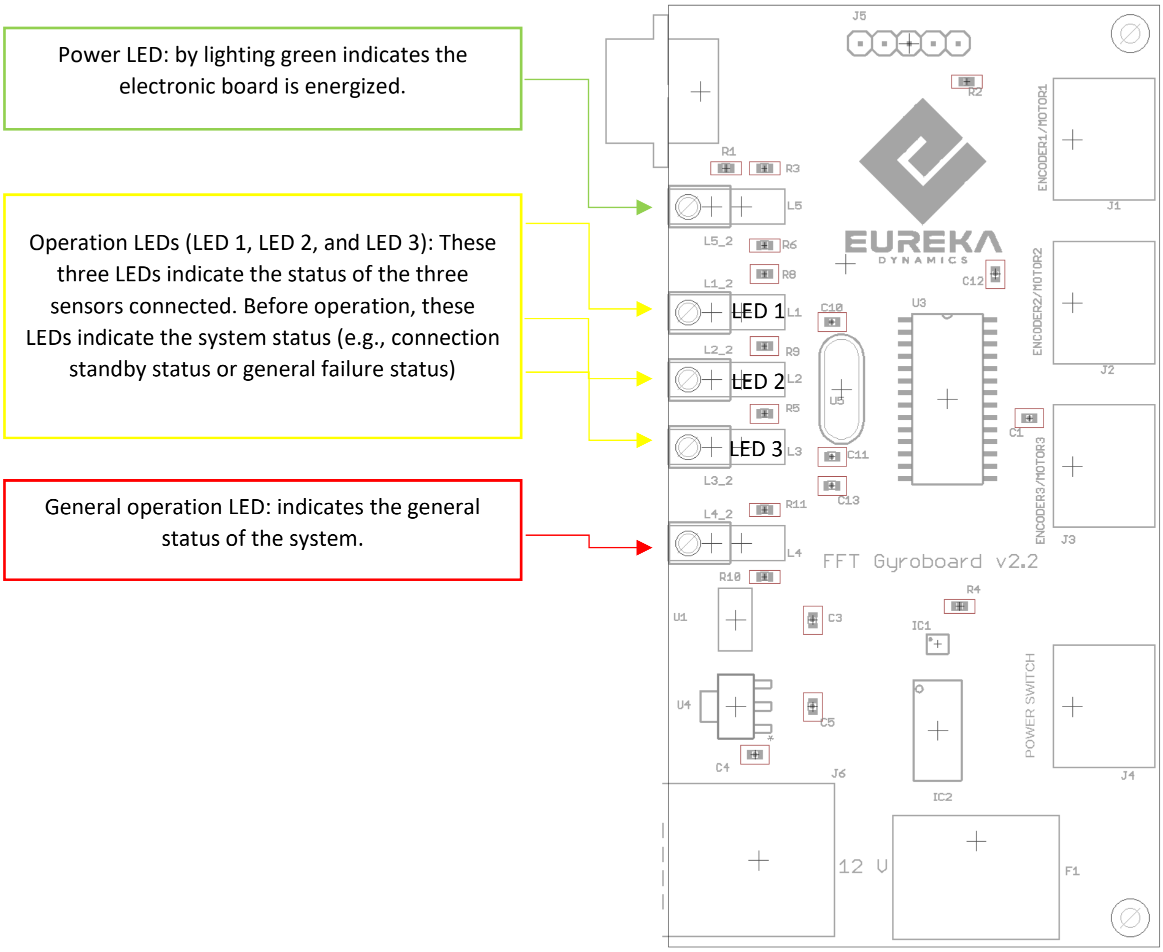
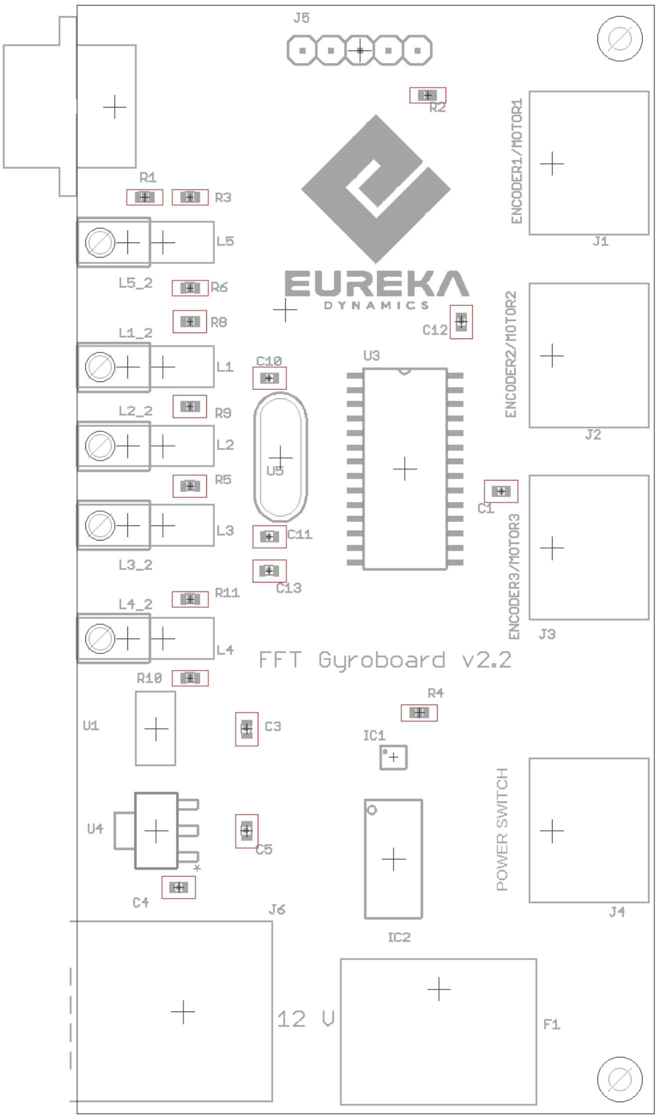


Figure 23. Schematic diagram of the FFT GYRO Gyroboard v2.2 system with LED description.

6.2 Installation process

1. Make sure the Fuse is placed.
2. When the fuse is placed, the external energy source can be connected, and the switch turned on.
3. To start using the system it is necessary to connect the FFT Gyroboard v2.2 board to a PC with 2.0 or above USB ports and a Windows 7 or above OS.
4. Once the system is connected, the Windows OS will detect it and make the necessary updates to make the system recognizable and functional. After Windows finishes, the system will appear as a recognized device: FFT Gyro System. It will also show the port assigned by Windows (e.g., COM 8, COM 23, etc.). The user needs to recognize the port assigned (COM), this can be checked in the device manager section. This is important to start using the system.
5. The port assigned is necessary to open the serial communication in MATLAB, Simulink or another program or IDE.



6.3 Setup

6.3.1 Encoder Mode

1. Make sure the on-off switch is *off*. The system must be in *off* mode.
2. Connect the external energy source (12 VCD) in the barrel Jack port and secure it with its thread.
3. Connect the encoders with all their respective connectors (Connector A, Connector C, Connector D) to the system.
4. When all the encoders are connected, the system can be turned *on*, positioning the switch in *on* mode.
5. Once energized, the energy LED will light up (green led).
6. At this point, the system is ready to start the data transmission showing the LEDs in the connection standby state. Now, the user can connect the USB connector to a PC. To know everything is OK with the system, the following images show the LED sequence of each mode.

6.3.1.1 Normal Operation: Encoder Mode

Energy LED *on*. Must always be lit to indicate the system is energized.

Yellow LEDs *on* (LED 1, LED 2, LED 3). Indicate good operation of the sensors. If one of them is off, one or two sensors have failure (e.g., If LED 1 is off, it means that the sensor 1 has failure).

General operation LED *on* (green).

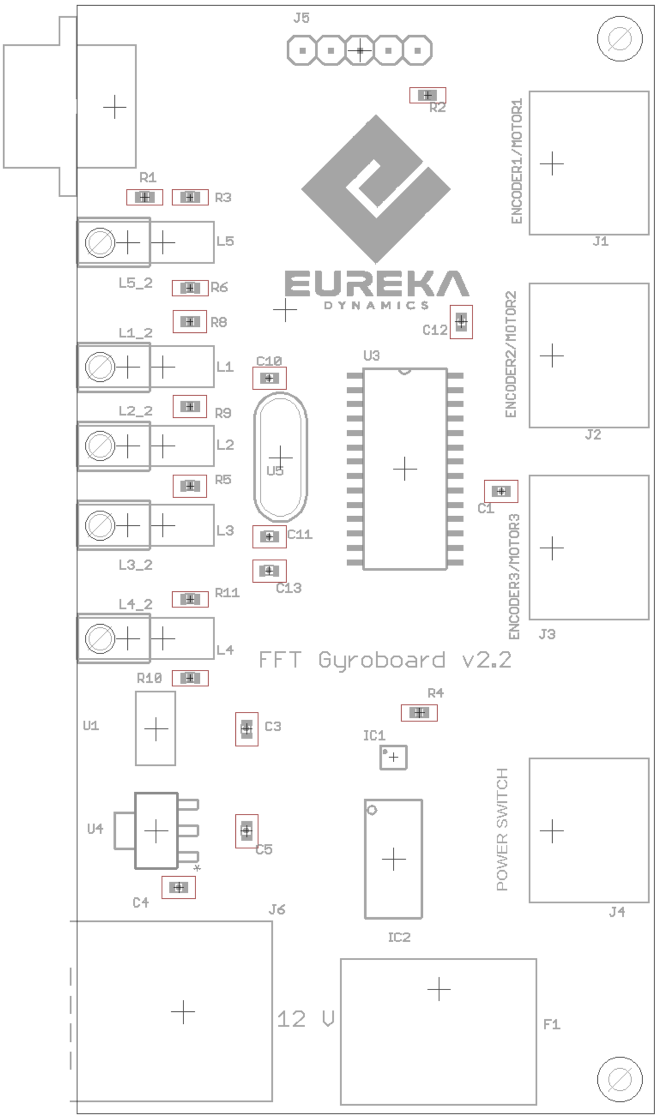
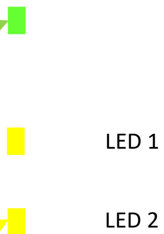


Figure 23. Schematic diagram of the FFT GYRO Gyroboard v2.2 with LED description in normal operation encoder mode.

6.3.1.2 Normal Operation: Encoder Mode (Failure in one or two encoders)

Energy LED *on*. Must always be lit to indicate the system is energized.



One or two yellow LEDs *off*. It indicates a problem in the corresponding sensor. If the LED 1 is *off*, the sensor 1 is having failure; if the LED 2 is *off*, the sensor 2 is having failure, or finally if the LED 3 is *off*, the sensor 3 is having failure.

LED 3

General operation LED *on* (green). Indicates that the operation will continue although 1 or 2 sensors have failure.

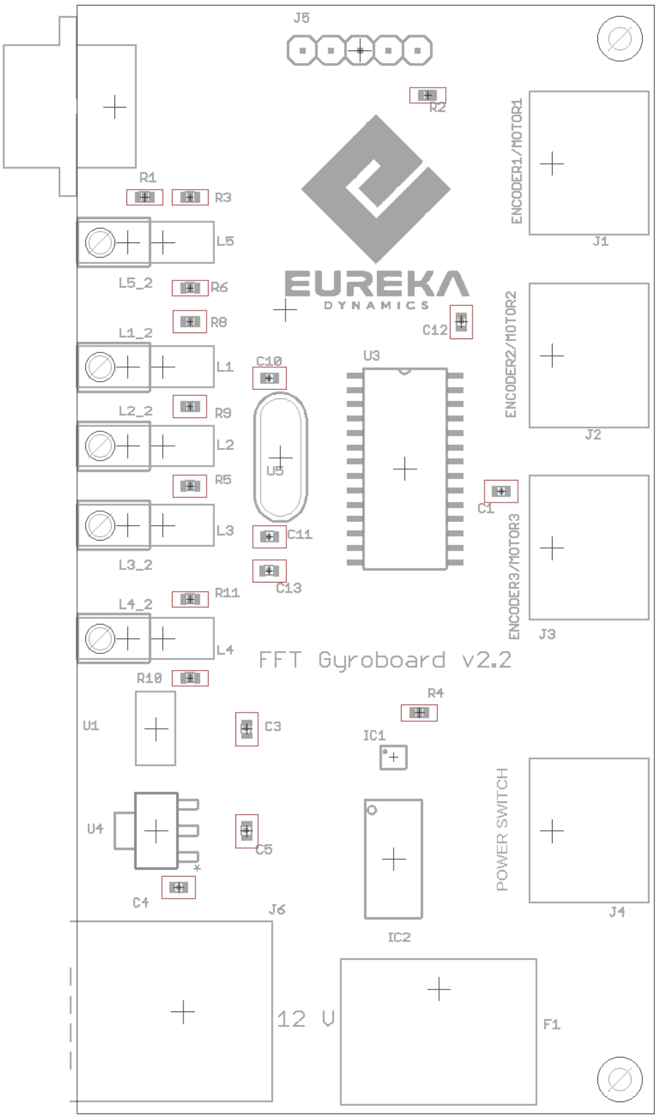


Figure 24. Schematic diagram of the FFT GYRO Gyroboard v2.2 with LED description with failure two encoders.

in one or

6.3.2 Motor Mode

1. Make sure the switch is *off*.
2. Connect the external energy (12 VCD) in the barrel Jack port secure it with its thread. source and
3. Connect the motors with all respective connectors (Connector A, Connector C, Connector D) to the system. their
4. When all the sensors are connected, the system can be turned on, positioning the switch in *on*.
5. Once energized, the LEDs will light up and the energy LED (green) should be *on*.
6. At this point, the system is ready to start the data transmission showing the LEDs in connection standby status. At this moment the user can connect the USB connector to a PC. To verify everything is OK with the system, the following images show the LED sequence of each mode.

6.3.2.1 Normal Operation: Motor Mode

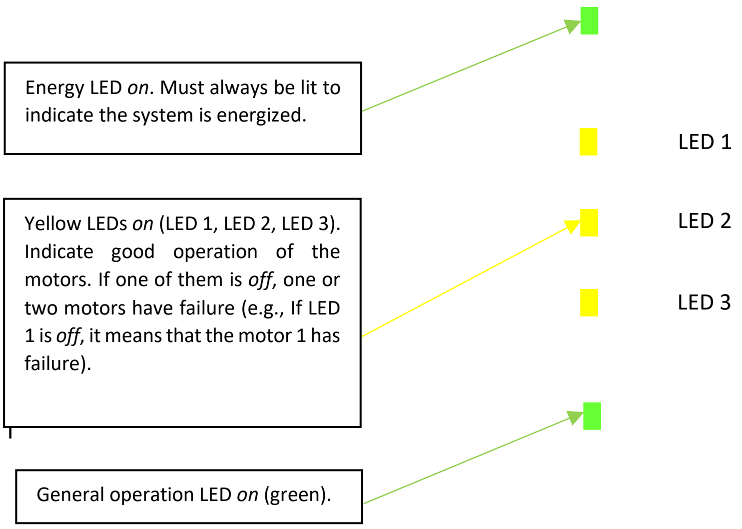


Figure 25. Schematic diagram of the FFT GYRO Gyroboard v2.2 with LED description in normal operation motor mode.

6.3.2.2 Normal Operation: Motor Mode (Failure in one or two motors)

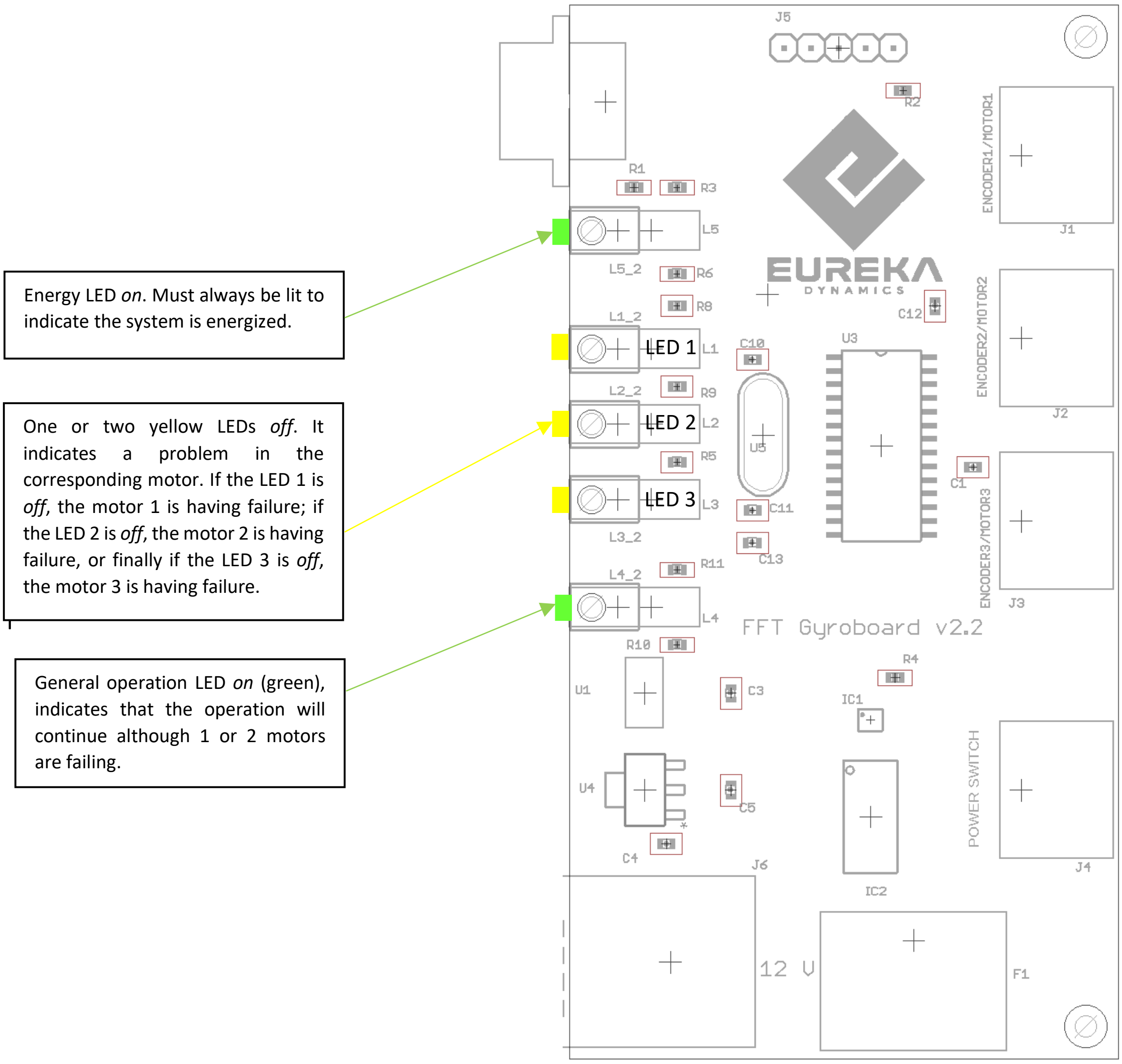


Figure 26. Schematic diagram of the FFT GYRO Gyroboard v2.2 with LED description with failure in one or two motors.

In addition, there are two more configuration modes that are very useful to check the correct operation of the system.

6.3.3 Connection Standby Mode

Standby mode indicates that the system is waiting for connection with the PC.

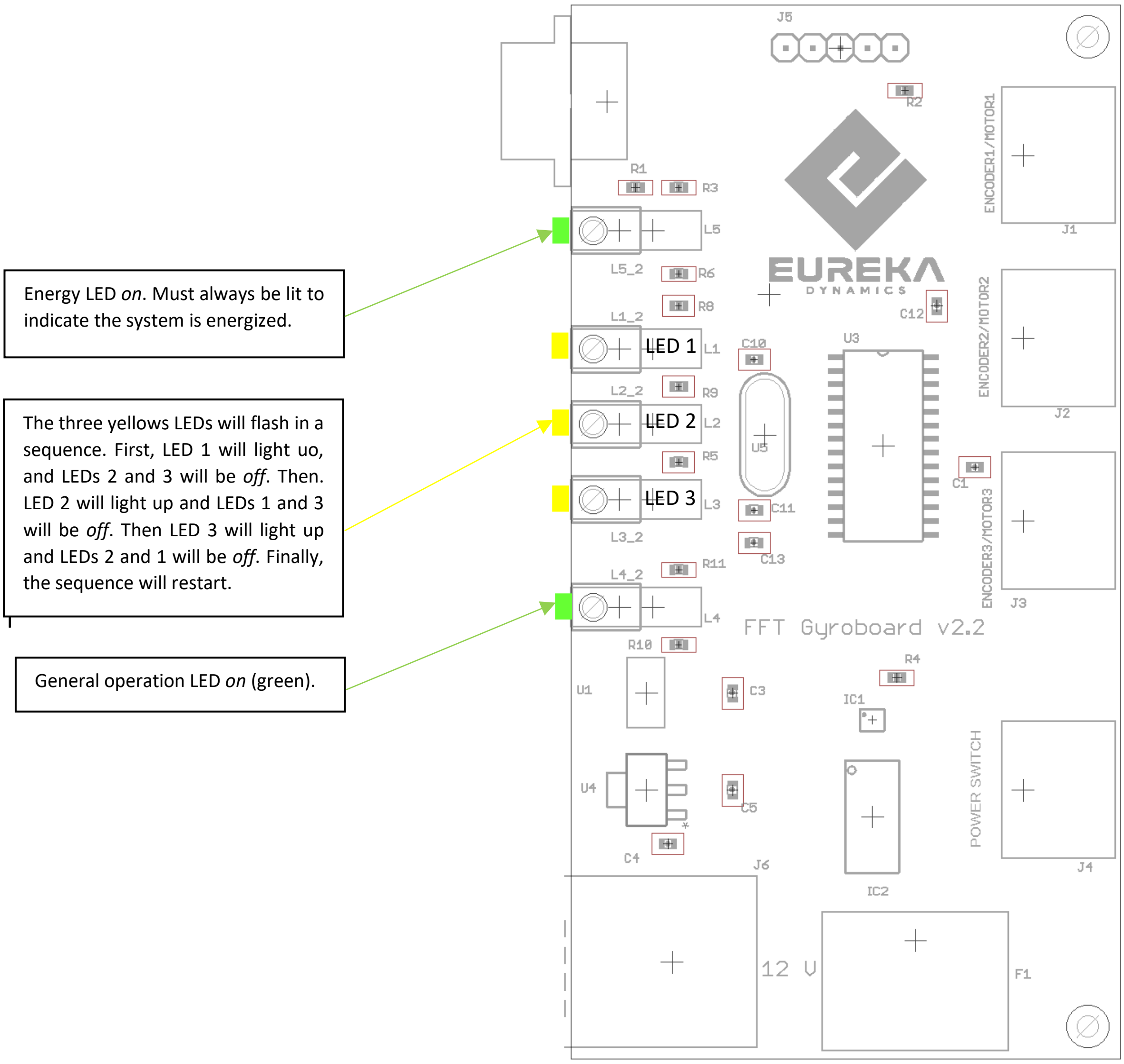


Figure 27. Schematic diagram of the FFT GYRO Gyroboard v2.2 with LED description in connection standby mode.

6.3.4 General Failure Mode

The system enters this mode when it does not detect sensors or motors. The problem could be that the sensors or motors are not connected properly or a failure with the connectors.

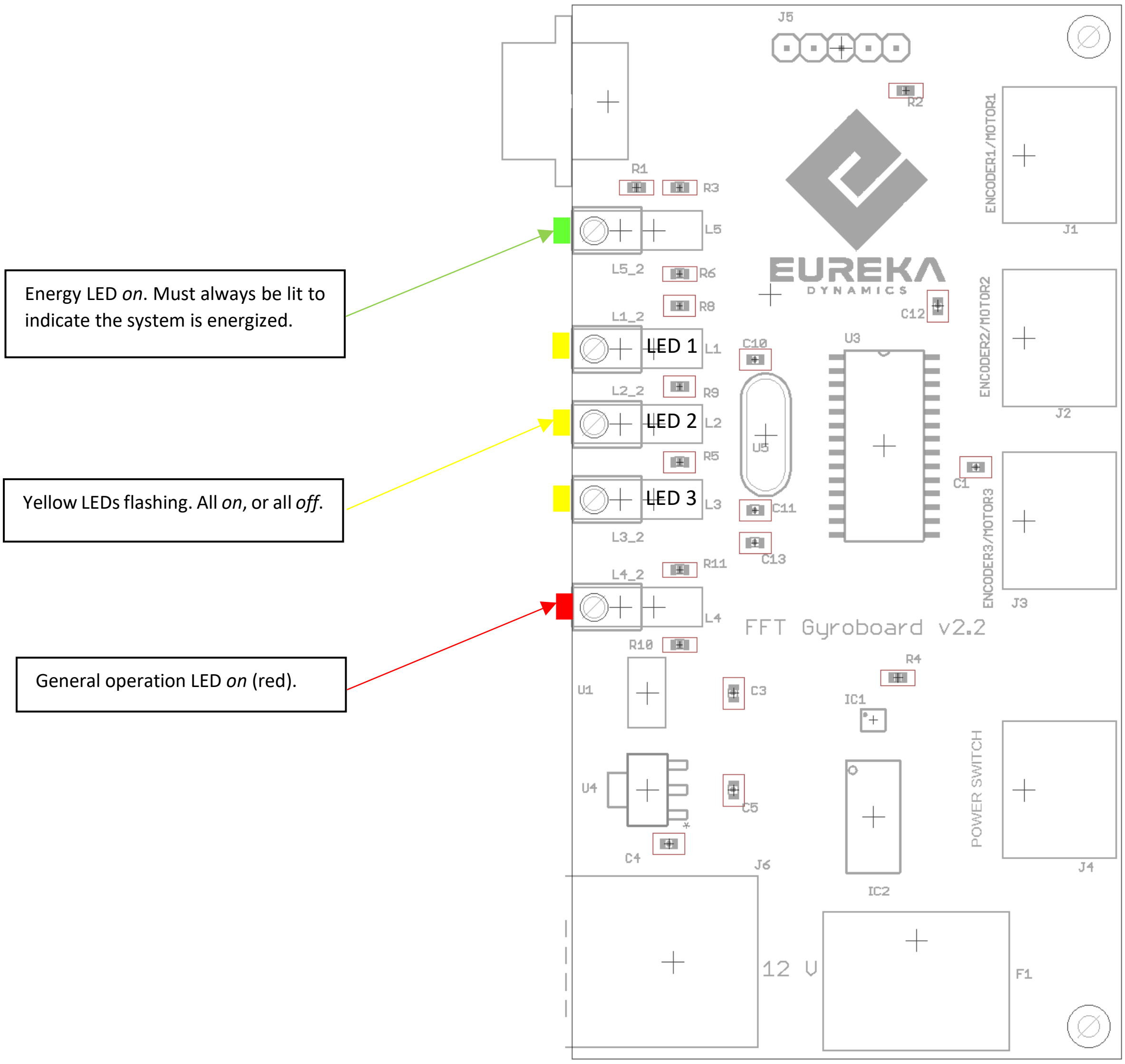


Figure 28. Schematic diagram of the FFT GYRO Gyroboard v2.2 with LED description in general failure mode.

6.3.5 Serial connection

Once the system set up is configured as mentioned above, the user needs to know the COM port to start reading or writing through the serial port with the next parameters.

Baud rate	**9600**
Data bits	8 bits
Parity	None
Stop Bits	1 bit

Table 6. Serial parameters to start the serial communication with the FFT Gyroboard v2.2.

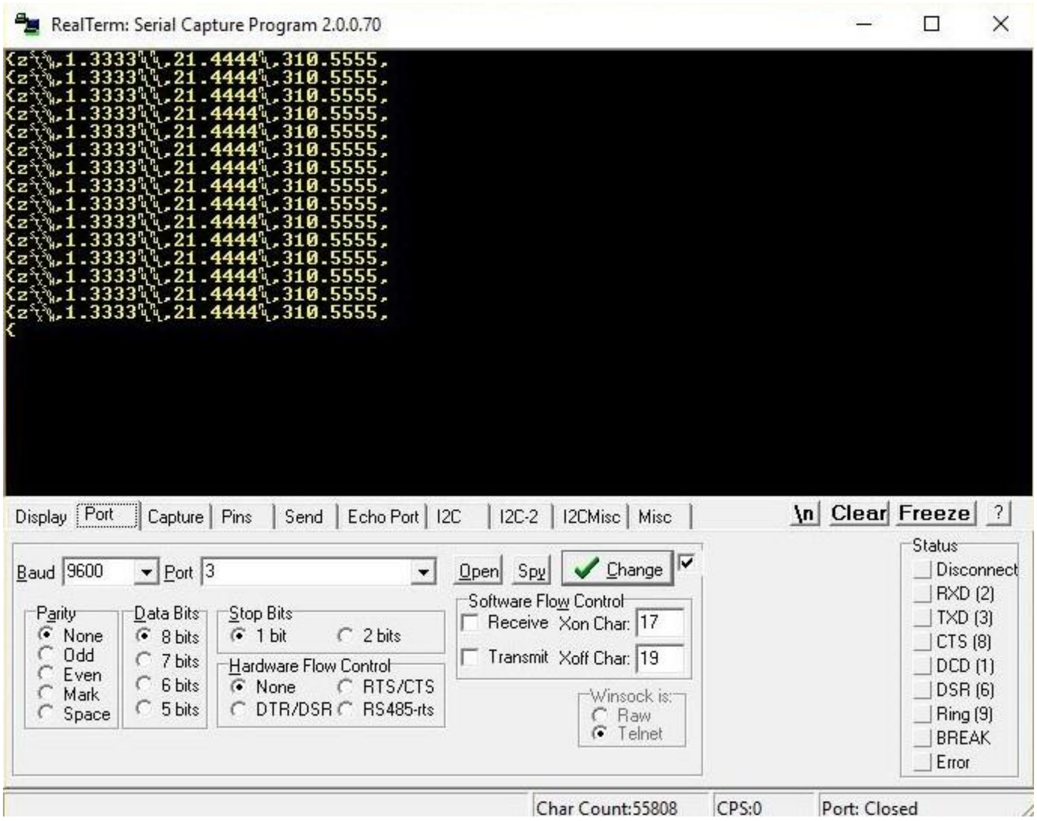


Figure 29. Data reception in Encoder Mode using the serial interface RealTerm.

6.4 MATLAB: Simulink

6.4.1 Encoder Mode

Once the user connects the encoders, the system will automatically detect them, and set the encoder operation mode. This mode allows to write only the data rate frequency and to read data about the encoders position. Using Simulink, you first need to place a **Serial Configuration Block** and a **Receive Serial Block** to open the port and get data through the USB.

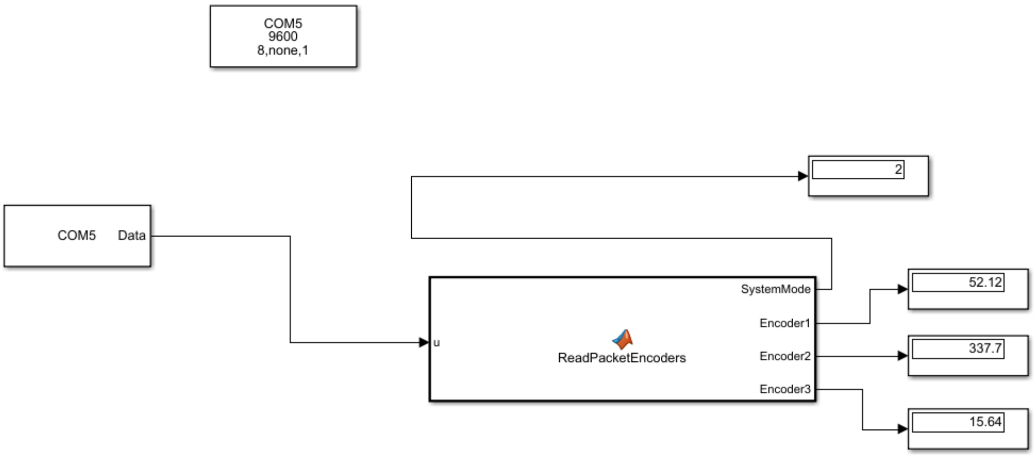


Figure 30. ReadPacketEncoders Block Example with Simulink.

6.4.2 Serial Configuration

The **Serial configuration** and **Serial Receive Block** parameters are shown in the next pictures. It's important to consider the size of the data package the system is sending, therefore, you must configurate the serial receive block with data size [32 1].

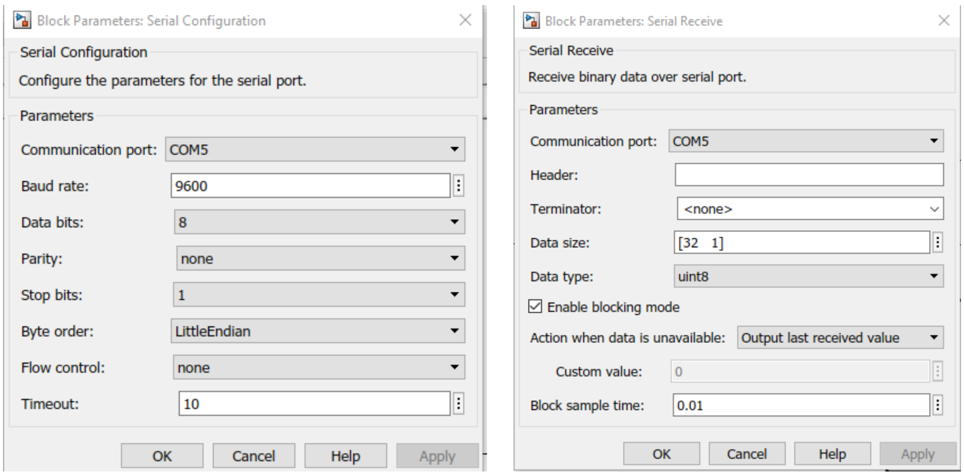


Figure 31. From left to right: Serial Configuration Parameters to open COM PORT and Serial Receive Parameters to get data through COM PORT, respectively.

6.4.3 Motor Mode

In this mode, users can receive data through the COM port, configurate the data rate the packet receives and have some other variables to write since the motors can be commanded to set their velocity, position, torque, and other options. For more information about the motors please refer to the writing packet in the next section. To start communication with the system users must set the **Serial Configuration Block**, and configurate the values as same as the section Serial Configuration.

6.4.3.1 setMotorsConfiguration Block

Once the Serial Configuration Block is placed and configurated, you can use this block to send a configuration packet to the motors, and configurate the motors mode (Joint or Wheel) and also the LEDs state (On and Off).

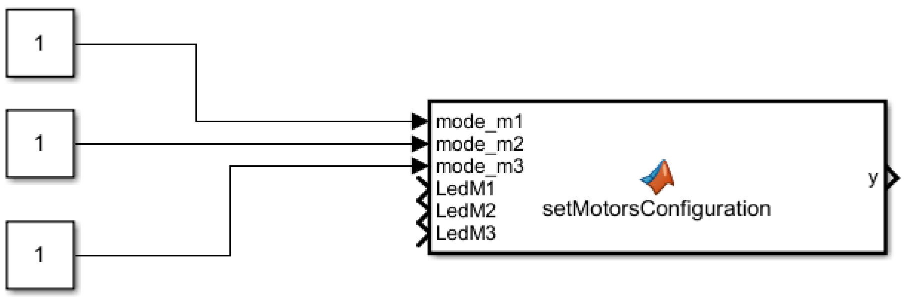


Figure 32. setMotorsConfiguration Block.

6.4.3.2 setMotorsVelocity Block

This block sends a write packet 2 to the motors. With this packet user can configurate velocity, angle limits (CW and CCW) and positions motors. This block in particular, only configurates the velocity of the motors, and has the option to write each one individually setting (or not) the setVelM variable. All the motors have their setVelM variable to indicate if the user would like to set them a value (setVelM = 1) or not(setVelM=0).

Variable	Range		Description
VelM	Modo Joint	0-1023	Velocity the motor can take depending on the mode, for more information about this variable in Wheel mode or Joint mode, please refer to the manufacturer official manual.
	Modo Wheel	0-2047	
setVM	0-1		Set velocity value to the motor (OFF/ON)

Table 7. Range and description of VelM and setVM variables.

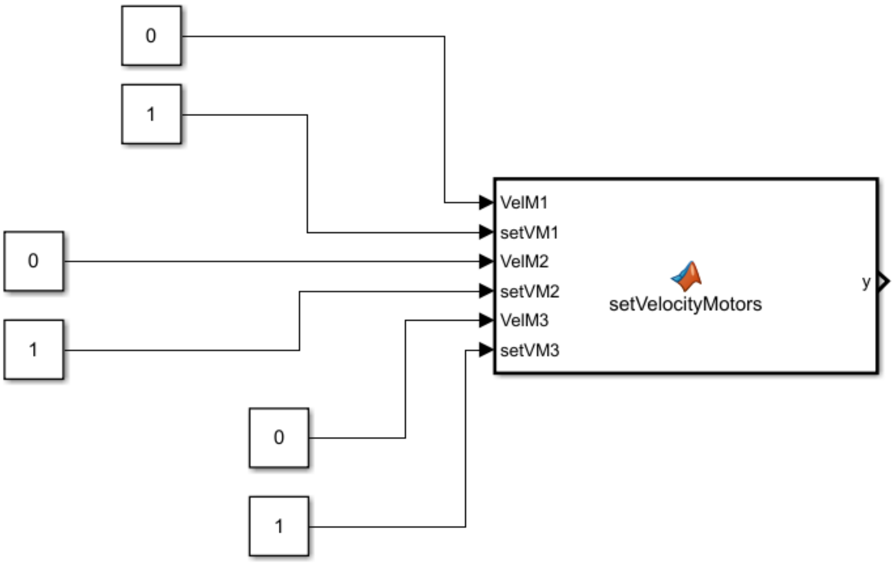


Figure 33. setMotorsVelocity Block.

6.4.3.3 setMotorsTorqueLimit Block

This block will send a write packet 1 to the motors. In this packet the user can configurate torque limit, max torque and torque enable variables of each motor. In the next table are shown the variables and the range of values of each one. This block has 15 input variables (5 for each motor).

Variable	Range	Description
TorqueLimit	0 – 1023	Torque limit value
setTL	0 -1	Set motor torque limit (OFF/ON)
MaxTorque	0-1023	Maximum torque value. When the motor is turned on, torque limit is reset to this value.
setMT	0-1	Set motor maximum torque (OFF/ON)
setTorqueEnable	0-1	Motor Torque Enable (ON/OFF)

Table 8. Range and description of the motor variables.

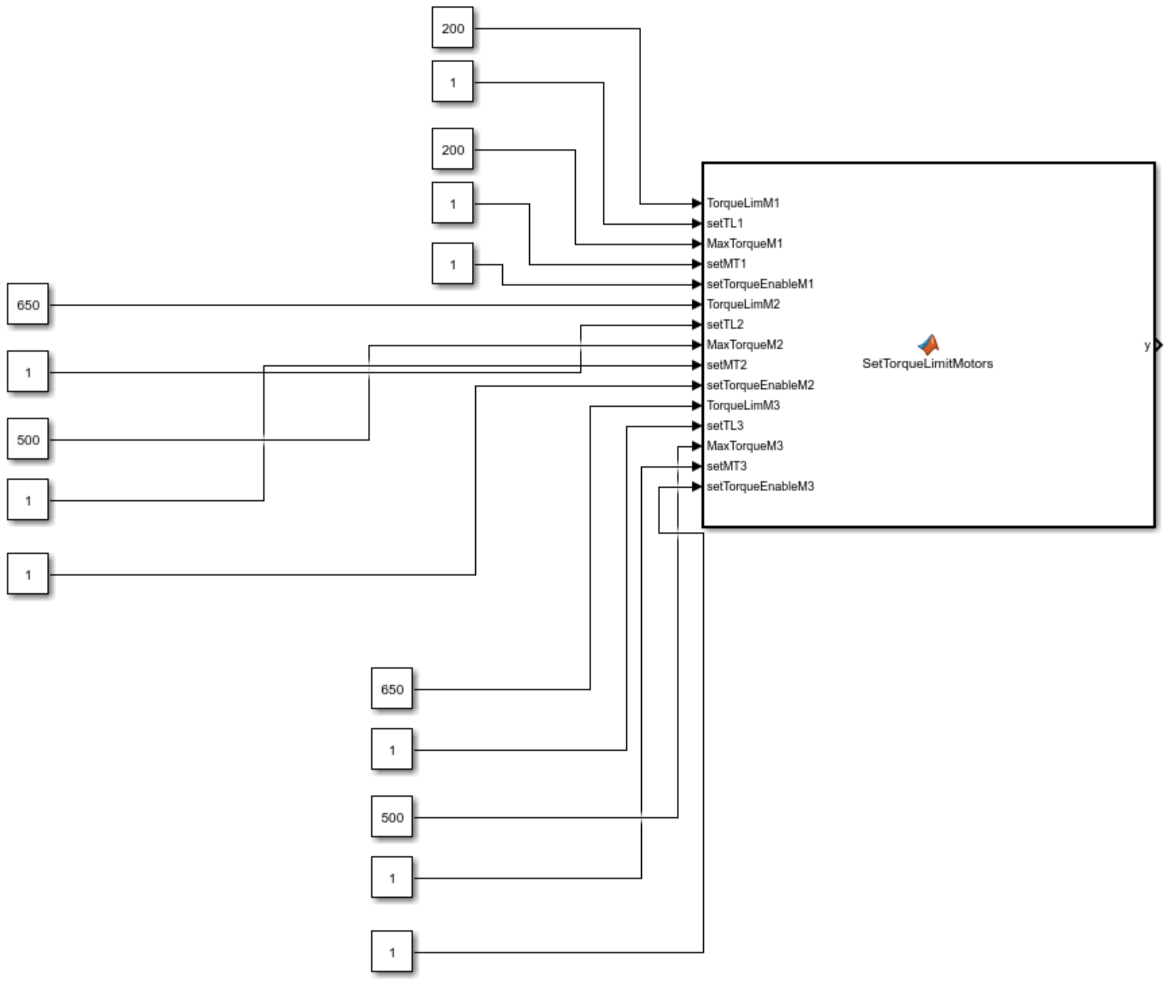


Figure 34. setMotorsTorqueLimit Block.

6.4.3.4 setMotorsPosition Block

This block will send a write packet 2. With this packet user can configurate velocity, angle limits (CW and CCW) and positions motors. This block has 21 variables, 7 for each motor, to configurate individually. In the next table is shown the variables, the range of this variables and a short description.

Variable	Range	Description
PosM	1-1023	Motor Goal Positon
SetPosM	0-1	Binary value tha indicates if the user wants to update the motor goal position (OFF/ON)
MovVel	0-1023	Motor moving speed value to reach the goal value.
setVel	0-1	Binary value indicating if the user wants to update the motor moving speed (OFF/ON)
CW_AngleLimit	1-1023	This is the minium angle the goal position can take. This value helps to restrain motion and also the set Wheel mode and Joint Mode. When both CW and CCW are 0 the motor Will automatically enter in Wheel mode.
CCW_AngleLimit	1-1023	This is the máximum angle the goal position can take. This value helps to restrain motion and also the set Wheel mode and Joint Mode. When both CW and CCW are 0 the motor Will automatically enter in Wheel mode.
setAngleLimit	1,2,3	Binary value to indicate if the user wants to update CW and CCW Angles Limit Values. 1: CW is updated 2: CCW is updated 3: both (CW and CCW) are updated.

Table 9. Velocity, angle and positions range and description for motors mode.

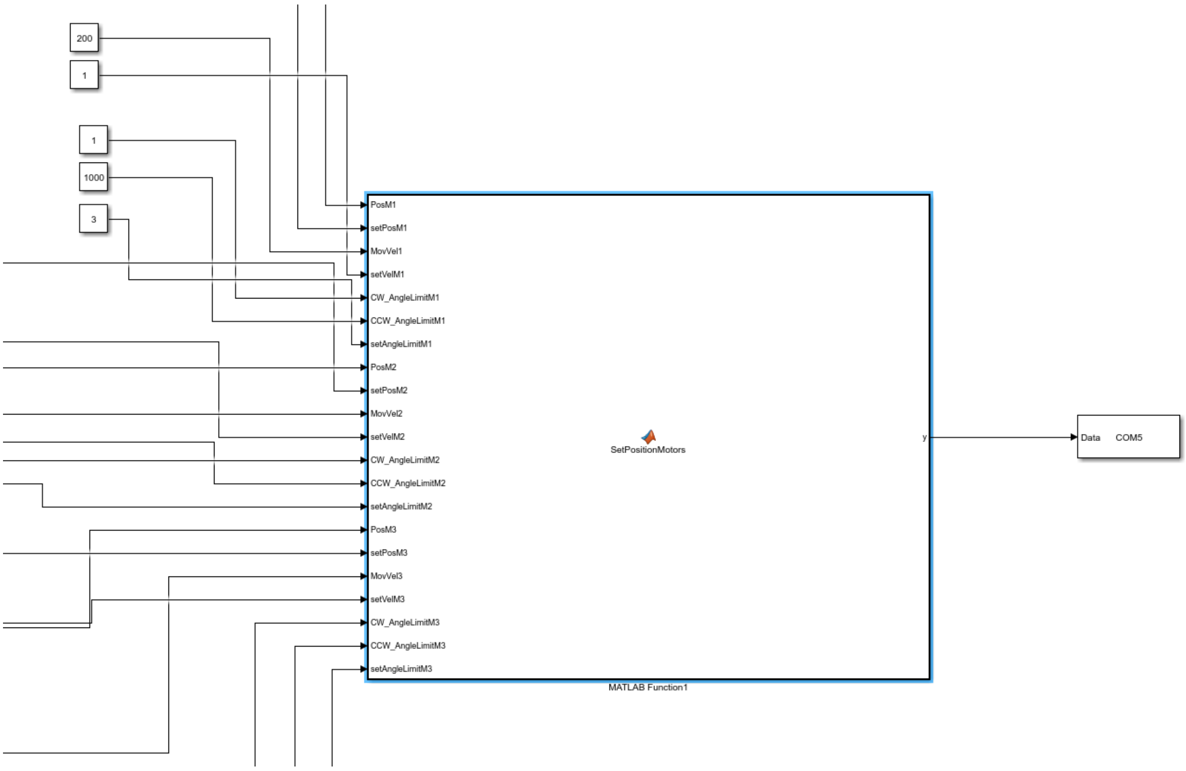


Figure 35. setMotorsPosition Block.

6.4.3.5 ReadMotorsPacket Block

This block receives the motor packet and will allow you to know the status of the motors and more specific information about them. For more specific information about this packet and its content please refer to the section “Read Packets/Motor Read Packet” in which we describe it with more detail .

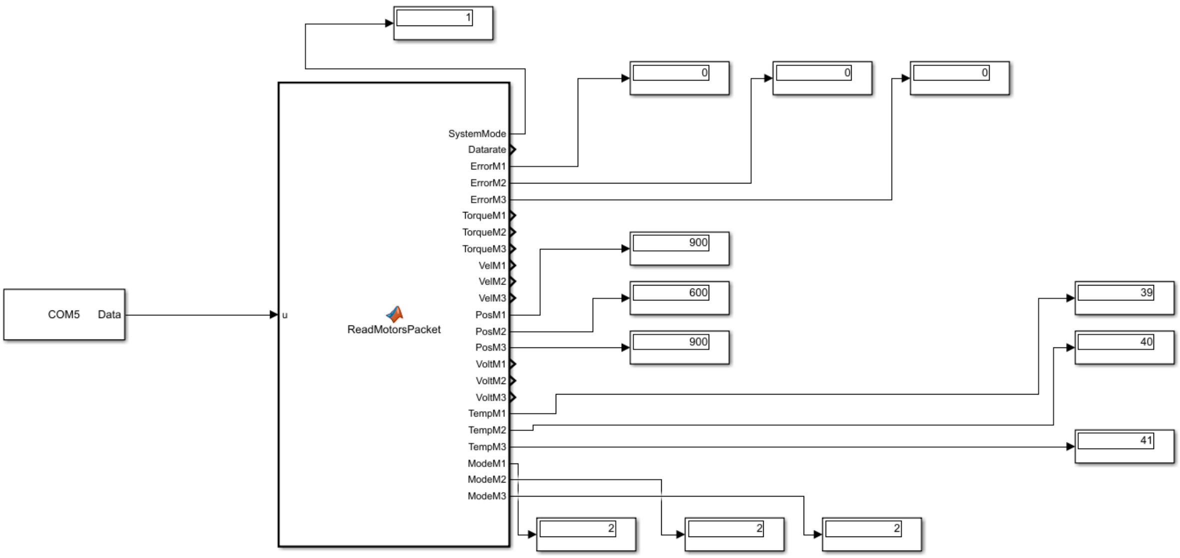


Figure 36. ReadPacketMotors Block Example with Simulink.

6.5 MATLAB: Scripts

6.5.1 Configuration Packet

With this function, users can configurate the motor's mode and the LED's state.

```
y = setMotorsConfiguration(puerto_serial,mode_m1,mode_m2,mode_m3,LedM1,LedM2,LedM3)
```

Input	Description
puerto_serial	Serial Port
mode_m1 **mode_m2** **mode_m3**	0: No change 1: Wheel Mode 2: Joint Mode
LedM1 **LedM2** **LedM3**	0: Led OFF 1: Led ON

Table 10. Input and description to configurate the motor's mode and LED state.

6.5.2 Read Packet

To get the FFT Gyro system data (in both encoder mode and motor mode) using MATLAB scripts, the user can use the next method:

```
[Data1, Data2, Data3] = getDataFromGyroboard (puerto_serial,Nbytes,SystemMode)
```

6.5.3 INPUT

Input	Description
puerto_serial	It is the port opened for the serial connection with the FFT Gyroboard v2.2.
Nbytes	It is a constant that indicates the number of bytes to read through the serial port (FFTGyroboardv2.2 will always send packets with 32 bytes).
SystemMode	It is the mode of the system. String variable that could be 'Motor' or 'Encoder'.

Table 11. Input and description to get the system data in encoder and motor mode.

6.5.4 OUTPUT

Motor Mode	*Encoder Mode*
Data1 (For Motor 1), Data2 (For Motor 2), Data3 (For Motor 3): **Double type matrix, where six values are stored corresponding to:** **Error Motor: Indicates if there is any error with the motor.**	**Data1, Data2 y Data3:** It corresponds to the values of encoders angle.

Torque Motor: Indicates the torque applied to the corresponding motor in an instant of time. **Velocity Motor: Indicates the velocity of the corresponding motor.** **Position Motor: Indicates the position of the corresponding motor.** **Voltage Motor: Indicates the voltage of the corresponding motor.** **Temperature Motor: Indicates the temperature of the corresponding motor.**	**Data1, Data2 y Data3:** It corresponds to the values of encoders angle.

Table 12. Output in motor mode and encoder mode.

6.5.5 Write Packets

There are 2 different types of write packets the user can use to modify the parameters of each motor individually. The first one, can be used to modify torque parameters as shown next.

```
Arr = sendPacket1ToGyroboard(puerto_serial,TorqueLimM1,setTL1,MaxTorqueM1,setMT1,setTorqueEnableM1, TorqueLimM2,setTL2,MaxTorqueM2,setMT2,setTorqueEnableM2, TorqueLimM3,setTL3,MaxTorqueM3,setMT3,setTorqueEnableM3)
```

Input	Description
puerto_serial	Serial Port
TorqueLimM1 **TorqueLimM2** **TorqueLimM3**	Configurate the value of Torque Limit of each motor.
setTL1 **setTL2** **setTL3**	0: Not change motor torque limit 1: set motor torque limit
MaxTorqueM1 **MaxTorqueM2** **MaxTorqueM3**	Configurate the value of Max Torque of each motor.
setMT1 **setMT2** **setMT3**	0: Not change motor max torque 1: set motor max torque
setTorqueEnableM1 **setTorqueEnableM2** **setTorqueEnableM3**	1: Set Torque Enable. Torque is Off by default; user can enable torque using this value. 0: Set Torque Off.

Table 13. Write packet 1 input and description.

Using the Write Packet 2, users can modify variables as position, velocity and limit angle.

```
Arr = sendPacket2ToGyroboard(puerto_serial,
PosM1,setPosM1,MovVel1,setVelM1,CW_AngleLimitM1,CCW_AngleLimitM1,setAngleLimitM1,
PosM2,setPosM2,MovVel2,setVelM2,CW_AngleLimitM2,CCW_AngleLimitM2,setAngleLimitM2,
PosM3,setPosM3,MovVel3,setVelM3,CW_AngleLimitM3,CCW_AngleLimitM3,setAngleLimitM3)
```

Input	Description
`puerto_serial`	Serial Port

`PosM1`
`PosM2`
`PosM3`
	Goal position value of each motor

`setPosM1`
`setPosM2`
`setPosM3`
	0: Does not change motor goal position
	1: Update motor goal position

`MovVel1`
`MovVel2`
`MovVel3`
	Motor's velocity value of each motor

`setVelM1`
`setVelM2`
`setVelM3`
	0: Does not change motor velocity
	1: Update motor velocity

`CW_AngleLimitM1`
`CW_AngleLimitM2`
`CW_AngleLimitM3`
	CW angle limit value of each motor

`CCW_AngleLimitM1`
`CCW_AngleLimitM2`
`CCW_AngleLimitM3`
	CCW angle limit value of each motor

`setAngleLimitM1`
`setAngleLimitM2`
`setAngleLimitM3`
	0: Does not change angle limits
	1: set CW angle limit
	2: set CCW angle limit
	3: set CW and CCW angle limit

Table 14. Write packet 2 input and description.

6.6 Processing: FFT GYRO Test Tool

The FFT GYRO Test Tool was developed to connect with the FFT GYRO System, and help the user test the system, using a variety of commands and charts to control and visualize the incoming data. The FFT GYRO Test Tool is a graphical user interface (GUI) and has been created using Processing 3 and Java and has three windows in which different information about the system is displayed.

The first window is the *Main Control* window and works as a command center and is different when motors are connected or when encoders are connected. The Main Control Window will help the user to connect through USB to the FFT GYRO System, send and receive information (in Motor and Encoder Mode) and display the data received.

The second window is *The Simple Primary Flight Display Emulator* and uses the position information from encoders and motors to translate them to the interface and emulate the flight display of an airplane.

The third and last window is the *3D Model* and takes the position values from motors and encoders and sends them to the model to indicate the orientation of the figure.

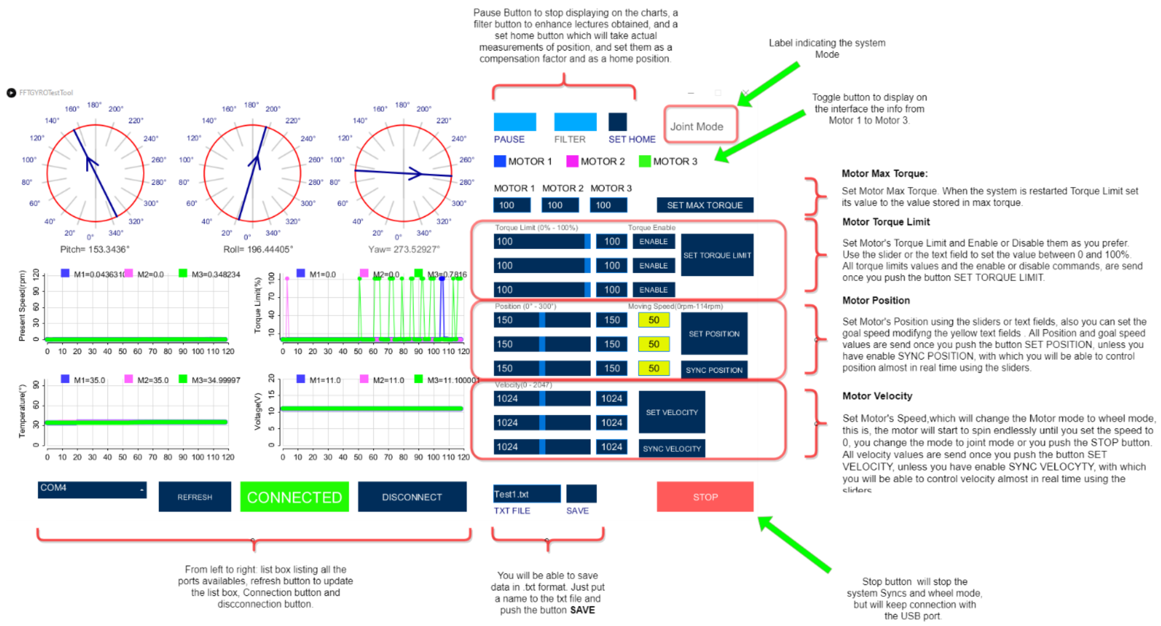


Figure 37. Main Control Window connected to the FFT Gyro System using motors.

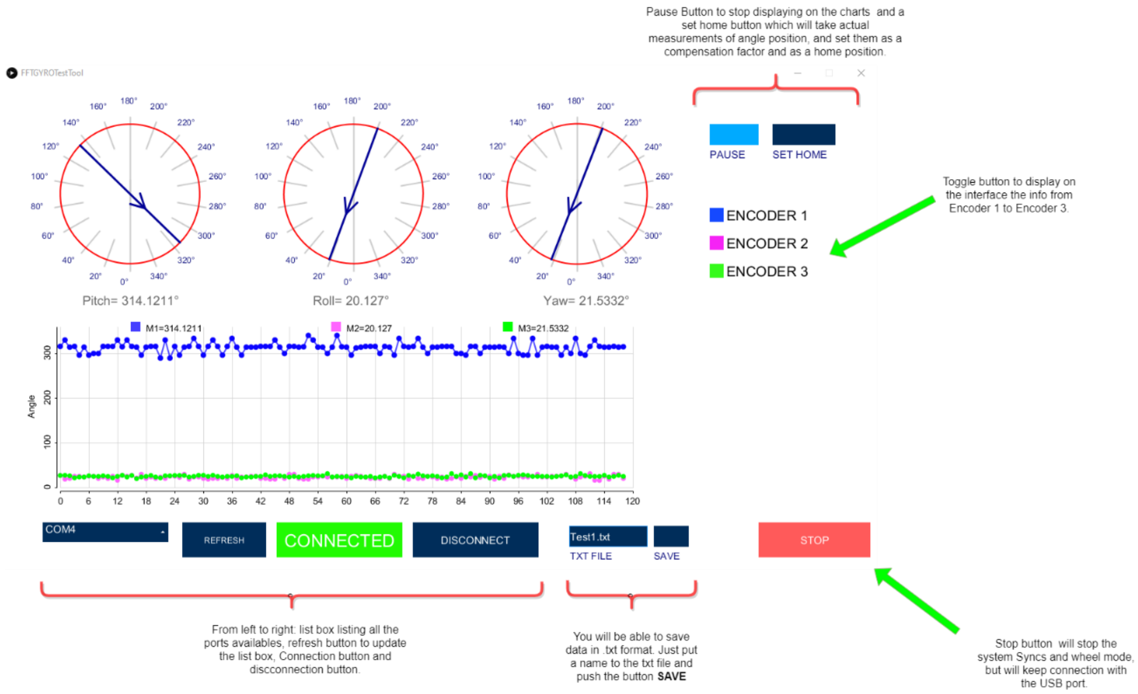


Figure 38. Main Control Window connected to the FFT Gyro System using encoders.

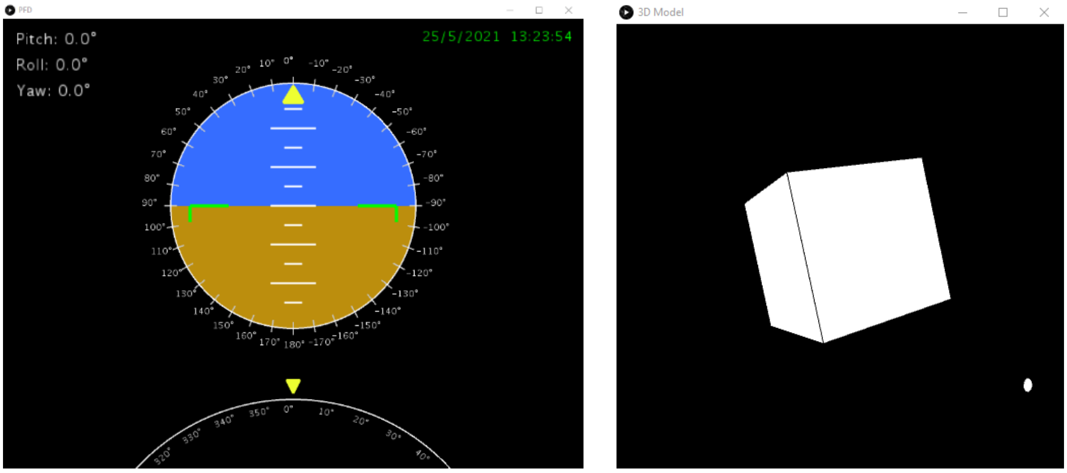


Figure 39. From left to right: Simple Primary Flight Display Emulator Illustration and 3D Model.

6.7 Quick Start Using MATLAB and Processing

6.7.1 Requirements

In this section we present some useful information about the requirements to run Matlab samples and Processing code. In both cases (samples and codes) were executed and tested in Windows 64 bits, but should run without issues in other operating systems like MacOS or Ubuntu as long as Matlab and Processing are compatible.

6.7.2 Requirements to run executable file

Exe files were compiled for different operating systems such as windows 32 and 64 bytes ,Mac Os X and Linux with different kernel architectures, check up what distribution and architecture have your computer and stick to the recommendations made here to ensure a proper operation. If any issue comes up with an exe file, we suggest you to open the program with processing, compile the code and make the exe file using the opcion File/Export Application.

6.7.3 Requirements for Matlab

To run Matlab Samples (Scripts and Simulink sample)you will need Matlab 2015 or further versions. Codes were tested using 2015 and 2017 on Windows 10 64 bits.

6.7.4 Requirements for Processing

To run the FFT GYRO Test Tool is necessary to install Processing 3.0 version or higher, and before running the code, it is also mandatory to install the graphic library available in Tools/Add Tool/Library. Once you are ready push play button located on top left of the Processing IDE and enjoy. Some requirements to install processing are:
Windows: Windows 7 onwards.

MacOS: It’s strongly recommended to install processing in MacOS X 10.8 and higher.

Ubuntu Linux: Processing is compatible with Linux, and it is recommended to use Ubuntu version 14.04 and higher. Most of the problems presented in this operating system are related with the JRE, which is recommended to be version 8.

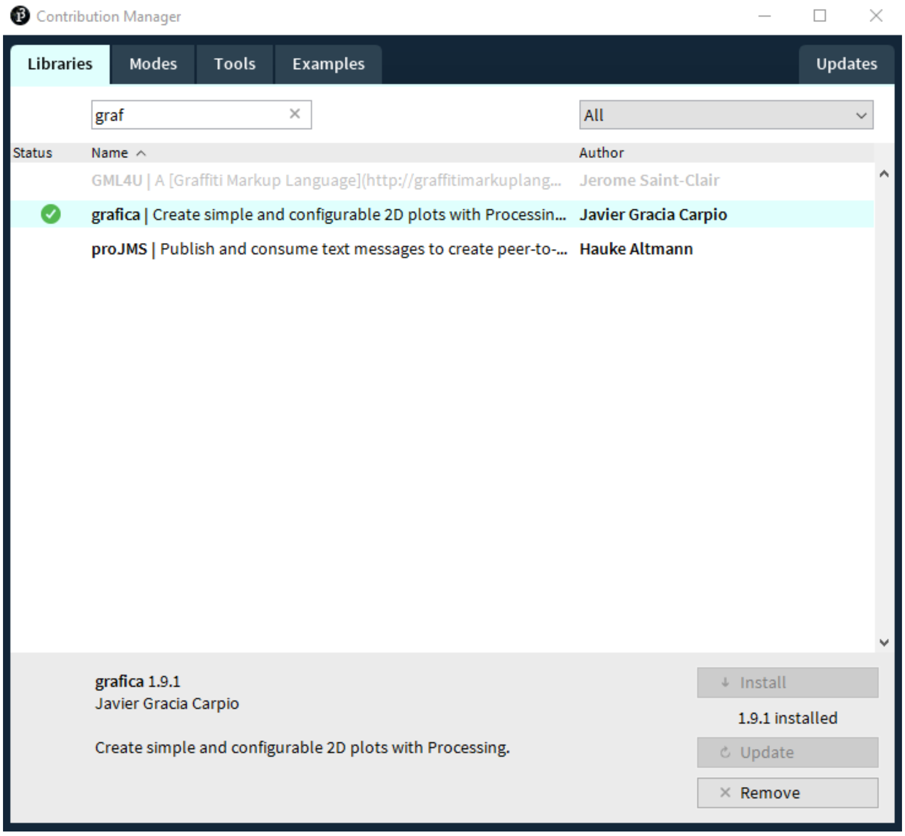


Figure 40. Contribution manager window in Processing.

6.8 Run FFT GYRO Test Tool

1. Download the WinRar or Zip file which contains the exe file for your favorite operating system.
2. Unzip the file and access the folder just created named, for instance in windows 64, “application.windows64”.
3. Once inside the folder, locate the exe file, select it, and press enter.

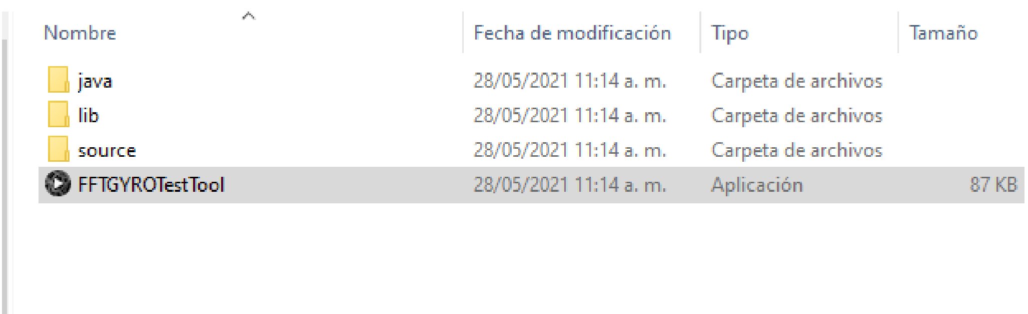


Figure 41. All files contained in the applicaition.windows64 folder.

4. You will see how 3 windows will be displayed. Locate the Main Control Window, and make sure that is totally in blank like the next image.

Figure 42. FFT GYRO Test Tool without being connected.

5. Next, plug the FFT GYRO System USB to the computer, push *REFRESH* button to update the ports available on the list, and on that list select the port corresponding to the system, and then click connect. If everything is ok, you will see the button connect turn to green and the label will display CONNECTED, for now on you can start receiving data from the system.

6.8.1 Run Sample Matlab Script

1. Open Matlab 2015 or higher.
2. Locate inside the folder Sample Matlab Script, and you will see the functions and scripts necessary to get ready and start getting data from the System. There are 3 sample files called, ***SamplePosition, SampleVelocity***, which will help you to test motors, and ***SampleReadEncoder*** which will help you to test encoder.

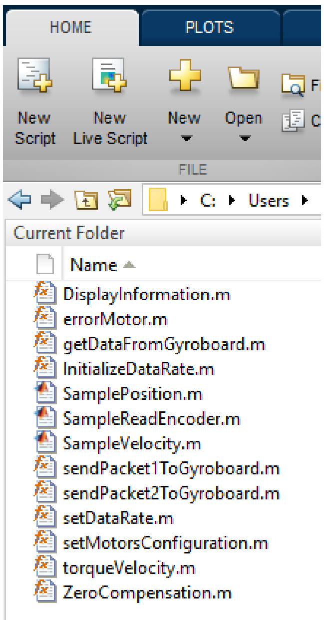


Figure 43. Files contained in the folder SampleMatlabScript.

6.8.2 Run Sample MATLAB Simulink

1. Open MATLAB 2015 or further.
2. Locate inside the folder **ModeloSimulink**, and you will see 2 files, ***SimulinkSampleEncoder.slx*** and ***SimulinkSampleMotor.slx***.

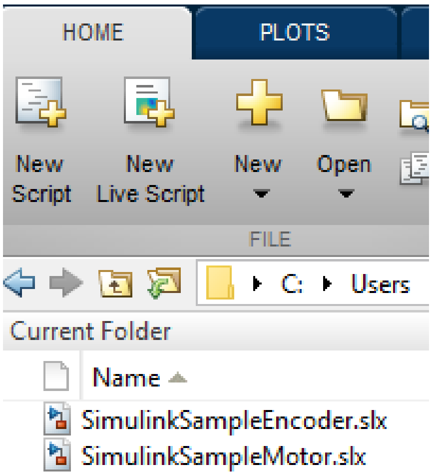


Figure 44. Files contained in the folder ModeloSimulink.

3. ***SimulinkSampleEncoder***: Before the start of the simulation make sure to select the correct port in Serial configuration block and Serial Receive block. Follow the instructions above in the section Matlab:Simulink to make sure to select the proper configuration port. Push the Run button locate on the top of the Simulink Environment.

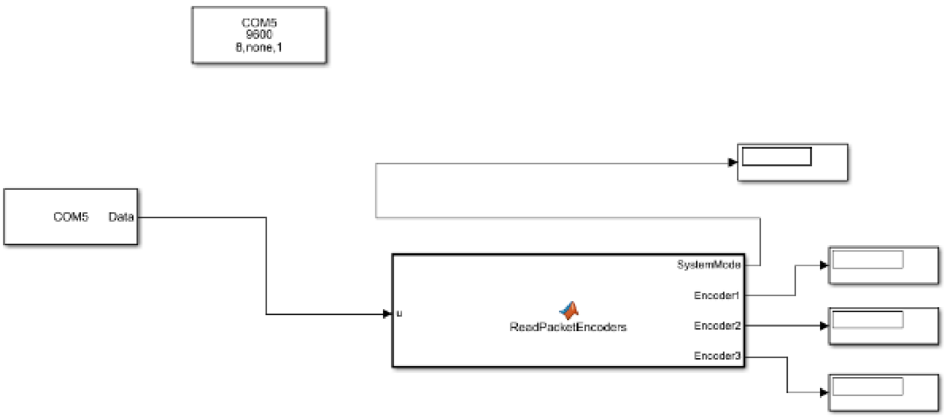


Figure 45. Simulink Sample Encoder.

4. ***SimulinkSampleMotor***: In the same way as in the **SimulinkSampleEncoder** file, you must be sure to set the proper configuration of the Serial Configuration Block. You can find all this information in the section Matkab: Simulink in the subsection Serial configuration above. Once you are sure the serial configuration is correct, in this sample you can found all blocks you can use whether you want to set velocity, set position or read the PORT to receive the information. Finally push the button Run to start the simulation.

6.8.3 Run FFT GYRO Test Tool Using Processing

1. Open Processing.
2. Clic Play button located on top left from the screen. Automatically 3 windows will display, the Main Control Window, the Simple Primary Flight Display Simulation and the 3D Model.
3. Depending on what is connected to the system (Encoders or Motors) you will see different displays and controls.

6.9 Read packets Structure

6.9.1 Encoder Read Packet

The packet that sent the system when it is in Encoder mode consists in 32 bytes which are described in the next table:

Byte position	Byte (Hexadecimal)	Description
0	0x7A	Constant byte that means the start of the packet.
1	Mode system	Current mode of the system: 1 = Motor mode 2 = Encoder mode
2	Data rate	The current data rate of the system. It is the value that multiply the default value. If the value is 0 (zero) the data rate is the default (10 ms). Example: Data rate = 100 Sample velocity = (10 ms) x 100 = 1000 ms = 1 second
3	‘,’ (0x2C)	Constant byte that allows to locate the values in the packet.
4	Encoder 1 position[0]	Byte 0 of the position’s value of encoder 1 Example: ‘3’
5	Encoder 1 position [1]	Byte 1 of the position’s value of encoder 1 Example: ‘1’
6	Encoder 1 position [2]	Byte 2 of the position’s value of encoder 1 Example: ‘0’
7	Encoder 1 position [3]	Byte 3 of the position’s value of encoder 1 Example: ‘.’

8	Encoder 1 position [4]	Byte 4 of the position's value of encoder 1 Example: '1'
9	Encoder 1 position [5]	Byte 5 of the position's value of encoder 1 Example: '2'
10	Encoder 1 position [6]	Byte 6 of the position's value of encoder 1 Example: '0'
11	Encoder 1 position [7]	Byte 7 of the position's value of encoder 1 Example: '9'
12	',' (0x2C)	Constant byte that allows to locate the values in the packet.
13	Encoder 2 position [0]	Byte 0 of the position's value of encoder 2 Example: '3'
14	Encoder 2 position [1]	Byte 1 of the position's value of encoder 2 Example: '1'
15	Encoder 2 position [2]	Byte 2 of the position's value of encoder 2 Example: '0'
16	Encoder 2 position [3]	Byte 3 of the position's value of encoder 2 Example: '.'
17	Encoder 2 position [4]	Byte 4 of the position's value of encoder 2 Example: '1'
18	Encoder 2 position [5]	Byte 5 of the position's value of encoder 2 Example: '2'
19	Encoder 2 position [6]	Byte 6 of the position's value of encoder 2 Example: '0'
20	Encoder 2 position [7]	Byte 7 of the position's value of encoder 2 Example: '9'
21	',' (0x2C)	Constant byte that allows to locate the values in the packet.
22	Encoder 3 position[0]	Byte 0 of the position's value of encoder 3 Example: '3'
23	Encoder 3 position[1]	Byte 1 of the position's value of encoder 3 Example: '1'
24	Encoder 3 position [2]	Byte 2 of the position's value of encoder 3 Example: '0'
25	Encoder 3 position [3]	Byte 3 of the position's value of encoder 3 Example: '.'
26	Encoder 3 position [4]	Byte 4 of the position's value of encoder 3 Example: '1'
27	Encoder 3 position [5]	Byte 5 of the position's value of encoder 3 Example: '2'
28	Encoder 3 position [6]	Byte 6 of the position's value of encoder 3 Example: '0'

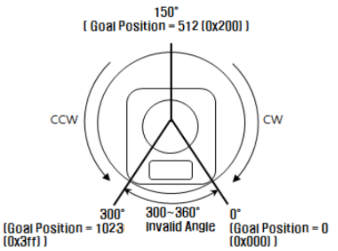
29	Encoder 3 position [7]	Byte 7 of the position's value of encoder 3 Example: '9'
30	',' (0x2C)	Constant byte that allows to locate the values in the packet.
31	0x7B	Constant byte that means the end of the packet.

Table 15. Encoder read packet.

6.9.2 Motor Read Packet

The packet sent by the FFT Gyro System in Motor Mode is always a packet of 32 bytes.

<table>
<tr><th>Byte position</th><th>Byte (Hexadecimal)</th><th>Description</th></tr>
<tr><td>0</td><td>0x7A</td><td>Constant byte that means the start of the packet.</td></tr>
<tr><td>1</td><td>Mode system</td><td>Current mode of the system:
1 = Motor mode
2 = Encoder Mode</td></tr>
<tr><td>2</td><td>Data rate</td><td>The current data rate of the system. It is the value that multiply the default value. If the value is 0 (zero) the data rate is the default (10 ms).
Example:
Data rate = 10
Sample velocity = (100 ms) x 10 = 1000 ms = 1 second</td></tr>
<tr><td>3</td><td>Error packet Motor 1</td><td>Error byte thrown off by the dynamixel motor A-12X number 1. The byte can be processed with the description of the next table:
ERROR
It displays the error status occurred during the operation of Dynamixel.
The meaning of each bit is described in the below table.
<table>
<tr><th>Bit</th><th>Name</th><th>Contents</th></tr>
<tr><td>Bit 7</td><td>0</td><td>-</td></tr>
<tr><td>Bit 6</td><td>Instruction Error</td><td>In case of sending an undefined instruction or delivering the action command without the reg_write command, it is set as 1.</td></tr>
<tr><td>Bit 5</td><td>Overload Error</td><td>When the current load cannot be controlled by the set Torque, it is set as 1.</td></tr>
<tr><td>Bit 4</td><td>Checksum Error</td><td>When the Checksum of the transmitted Instruction Packet is incorrect, it is set as 1.</td></tr>
<tr><td>Bit 3</td><td>Range Error</td><td>When a command is out of the range for use, it is set as 1.</td></tr>
<tr><td>Bit 2</td><td>Overheating Error</td><td>When internal temperature of Dynamixel is out of the range of operating temperature set in the Control table, it is set as 1.</td></tr>
<tr><td>Bit 1</td><td>Angle Limit Error</td><td>When Goal Position is written out of the range from CW Angle Limit to CCW Angle Limit , it is set as 1.</td></tr>
<tr><td>Bit 0</td><td>Input Voltage Error</td><td>When the applied voltage is out of the range of operating voltage set in the Control table, it is as 1.</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Error packet Motor 2</td><td>Error byte thrown off by the dynamixel motor A-12X number 2. Processed with the same table as the byte "Error packet Motor 1"</td></tr>
<tr><td>5</td><td>Error packet Motor 3</td><td>Error byte thrown off by the dynamixel motor A-12X number 3.
Processed with the same table as the byte "Error packet Motor 1"</td></tr>
<tr><td>6</td><td>Torque Motor 1 [0]</td><td>Byte 0 of torque generated by the motor 1.
The range contemplated for the 2 bytes is 0~2047 and the step and unit is 0.1 %. If the value is between 0 and 1023 it means that the charge is in CCW direction (opposite direction on a traditional clock). If the value is between 1024 and 2047 it means that the charge is in CW direction (direction as a traditional clock).The tenth bit only selects the direction, for this, 1024 is equal to 0 (no charge)
Example:
If the value of this two registers is 512, it means that the charge detected is opposite clock direction and 50 % of the maximum possible torque generated.
This applies to all the motors (1, 2 and 3)
<table>
<tr><td>BIT</td><td>15~11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Value</td><td>0</td><td>Load Direction</td><td colspan="10">Data (Load Ratio)</td></tr>
</table>
Load Direction = 0 : CCW Load, Load Direction = 1: CW Load</td></tr>
<tr><td>7</td><td>Torque Motor 1 [1]</td><td>Byte 1 of torque generated by the motor 1.</td></tr>
<tr><td>8</td><td>Torque Motor 2 [0]</td><td>Byte 0 of torque generated by the motor 2.</td></tr>
<tr><td>9</td><td>Torque Motor 2 [1]</td><td>Byte 1 of torque generated by the motor 2.</td></tr>
</table>

<table>
<tr><td>10</td><td>Torque Motor 3 [0]</td><td>Byte 0 of torque generated by the motor 3.</td></tr>
<tr><td>11</td><td>Torque Motor 3 [1]</td><td>Byte 1 of torque generated by the motor 3.</td></tr>
<tr><td>12</td><td>Velocity Motor 1 [0]</td><td>Byte 0 of the velocity detected in the motor 1.
The range contemplated for the 2 bytes is 0~2047 and the step and unit is 0.1 %. If the value is between 0 and 1023 it means that the velocity is in CCW direction (opposite direction on a traditional clock). If the value is between 1024 and 2047 it means that the velocity is in CW direction (direction as a traditional clock).The tenth bit only selects the direction, for this, 1024 is equal to 0 (no velocity)
Example:
If the value of this two registers is 512, it means that the velocity detected is opposite clock direction and 50 % of the maximum possible velocity generated.
This applies to all the motors (1, 2 and 3)
<table>
<tr><td>BIT</td><td>15~11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Value</td><td>0</td><td>Load Direction</td><td colspan="10">Data (Load Ratio)</td></tr>
</table>
Load Direction = 0 : CCW Load, Load Direction = 1: CW Load</td></tr>
<tr><td>13</td><td>Velocity Motor 1 [1]</td><td>Byte 1 of the velocity detected on motor 1.</td></tr>
<tr><td>14</td><td>Velocity Motor 2 [0]</td><td>Byte 0 of the velocity detected on motor 2.</td></tr>
<tr><td>15</td><td>Velocity Motor 2 [1]</td><td>Byte 1 of the velocity detected on motor 2.</td></tr>
<tr><td>16</td><td>Velocity Motor 3 [0]</td><td>Byte 0 of the velocity detected on motor 3.</td></tr>
<tr><td>17</td><td>Velocity Motor 3 [1]</td><td>Byte 1 of the velocity detected on motor 3.</td></tr>
<tr><td>18</td><td>Position Motor 1 [0]</td><td>Byte 0 of the position detected on motor 1.
Present Position
It is the current position value of Dynamixel.
The range of the value is 0~1023 (0x3FF), and the unit is 0.29 degree.


<The picture above is based on the front of relevant model>
Caution: If it is set to Wheel Mode, the value cannot be used to measure the moving distance and the rotation frequency.
Applies on the positions of all the motors (1, 2 and 3)</td></tr>
<tr><td>19</td><td>Position Motor 1 [1]</td><td>Byte 1 of the position detected on motor 1.</td></tr>
<tr><td>20</td><td>Position Motor 2 [0]</td><td>Byte 0 of the position detected on motor 2.</td></tr>
<tr><td>21</td><td>Position Motor 2 [1]</td><td>Byte 1 of the position detected on motor 2.</td></tr>
<tr><td>22</td><td>Position Motor 3 [0]</td><td>Byte 0 of the position detected on motor 3.</td></tr>
<tr><td>23</td><td>Position Motor 3 [1]</td><td>Byte 1 of the position detected on motor 3.</td></tr>
<tr><td>24</td><td>Voltage Motor 1</td><td>Voltage detected on the motor 1. The read value is 10 times more than the actual voltage.
Example: when the motor is energized with 10 volts, the value read will be 100 (0x64). The same applies to all motors (1, 2 and 3).</td></tr>
<tr><td>25</td><td>Voltage Motor 2</td><td>Voltage detected on the motor 1. The read value is 10 times more than the actual voltage.
Example: when the motor is energized with 10 volts, the value read will be 100 (0x64). The same applies to all motors (1, 2 and 3).</td></tr>
<tr><td>26</td><td>Voltage Motor 3</td><td>Voltage detected on the motor 1. The read value is 10 times more than the actual voltage.
Example: when the motor is energized with 10 volts, the value read will be 100 (0x64). The same applies to all motors (1, 2 and 3).</td></tr>
<tr><td>27</td><td>Temperature Motor 1</td><td>It is the internal temperature of the motor 1. The byte read is the current temperature in Celsius.
Example: if the byte read is 85 (0x55), the temperature of the motor would be 85 Celsius. The same applies to all the motors (1, 2 and 3).</td></tr>
<tr><td>28</td><td>Temperature Motor 2</td><td>It is the internal temperature of the motor 1. The byte read is the current temperature in Celsius.
Example: if the byte read is 85 (0x55), the temperature of the motor would be 85 Celsius. The same applies to all the motors (1, 2 and 3).</td></tr>
</table>

<table>
<tr><td>29</td><td>Temperature Motor 3</td><td>It is the internal temperature of the motor 1. The byte read is the current temperature in Celsius.
Example: if the byte read is 85 (0x55), the temperature of the motor would be 85 Celsius. The same applies to all the motors (1, 2 and 3).</td></tr>
<tr><td>30</td><td>Modemotors</td><td>This byte indicates the mode of each motor. The modes supported by the motors are Joint mode and Wheel mode, whit this byte you can read it using the following structure and the next truth table.

<table>
<tr><td colspan="8">Modemotors byte</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>X</td><td>X</td><td>M1_MSB</td><td>M1_LSB</td><td>M2_MSB</td><td>M2_LSB</td><td>M3_MSB</td><td>M3_LSB</td></tr>
</table>

<table>
<tr><td>Mx_MSB</td><td>Mx_LSB</td><td>Meaning</td></tr>
<tr><td>0</td><td>1</td><td>Mx on Mode Wheel</td></tr>
<tr><td>1</td><td>0</td><td>Mx on Mode Joint</td></tr>
<tr><td>1</td><td>1</td><td>MX on unknown mode</td></tr>
</table>
</td></tr>
<tr><td>31</td><td>0x7B</td><td>Constant byte that means the end of the packet.</td></tr>
</table>

Table 16. Motor read packet.

6.10 Write packets Structure

The write packet is only available in the Motor Mode using the Dynamixel motors AX-12A with protocol version 1.0. To know more about the motor implemented in the FFT Gyro System the online manual can be found on https://emanual.robotis.com/docs/en/dxl/ax/ax-12a.

6.10.1 Write packet 1

This packet have 32 bytes that you can use to configure some properties and options related to the torque of motors and data rate of the system. The packet that the FFT Gyro System can read from a computer through the serial port is 32 bytes long and is described in the next tables.

<table>
<tr><th>Byte position</th><th>Byte (Hexadecimal)</th><th>Available Options</th><th>Description</th></tr>
<tr><td>0</td><td>0x7A</td><td>0x7A</td><td>Constant start byte</td></tr>
<tr><td>1</td><td>0x01</td><td>0x01</td><td>Write packet number 1</td></tr>
<tr><td>2</td><td>Config data rate</td><td>1 ascii (‘1’): if the user wants to change the data rate. The system will take the value of the next byte to calculate de new data rate.
0 ascii (‘0’): if the user does not want to change the data rate of the system.

Example: Change data rate (‘1’)
<table>
<tr><td colspan="8">Config data rate</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
</table>
</td><td>Flag that indicates a change in the data rate of the system.</td></tr>
<tr><td>3</td><td>Data rate</td><td>Example: The user wants a 0.1 seconds data rate then the value of this byte needs to have 10 (0x0A), then the data rate will be:
Data rate = 10 x (10 ms) = 100 ms =0. 1 second</td><td>The value that will multiply the default value (10 ms), in order to achieve the data rate wanted.</td></tr>
</table>

<table>
<tr><td>4</td><td>Torque Enable M1 M2 M3</td><td>
<table>
<tr><td colspan="8">Torque Enable M1 M2 M3 (8 bits)</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Z</td><td>Y</td><td>X</td></tr>
</table>
Truth table:
<table>
<tr><td>Z</td><td>Y</td><td>X</td><td>Meaning</td></tr>
<tr><td>NC</td><td>NC</td><td>1</td><td>Enable torque on motor 1</td></tr>
<tr><td>NC</td><td>1</td><td>NC</td><td>Enable torque on motor 2</td></tr>
<tr><td>1</td><td>NC</td><td>NC</td><td>Enable torque on motor 3</td></tr>
</table>
Example: Enable torque in Motor 1 and Motor 2.
<table>
<tr><td colspan="8">Torque Enable M1 M2 M3 (8 bits)</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table>
</td><td>Flag that indicates the system, the user wants to enable the torque on one or more motors.</td></tr>
<tr><td>5</td><td>Set torque limit M1 M2 M3</td><td>
<table>
<tr><td colspan="8">Set torque limit M1 M2 M3 (8 bits)</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Z</td><td>Y</td><td>X</td></tr>
</table>
Truth table:
<table>
<tr><td>Z</td><td>Y</td><td>X</td><td>Meaning</td></tr>
<tr><td>NC</td><td>NC</td><td>1</td><td>Set torque limit on motor 1</td></tr>
<tr><td>NC</td><td>1</td><td>NC</td><td>Set torque limit on motor 2</td></tr>
<tr><td>1</td><td>NC</td><td>NC</td><td>Set torque limit on motor 3</td></tr>
</table>
Example: Enable a change in torque limit in Motor 1 and Motor 2.
<table>
<tr><td colspan="8">Set torque limit M1 M2 M3 (8 bits)</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table>
</td><td>Flag that indicates the system, the user wants to change the torque limit of the motors.</td></tr>
<tr><td>6</td><td>Torque Limit M1 [0]</td><td rowspan="2">These two bytes contain the value of the torque used by motor 1, from 0 to 100 %. The value range can be from 0 to 1023 with 0.1 % steps.

Example: Use the limit torque of 100 % (1023 o 0x3ff) on Motor 1
<table>
<tr><td colspan="8">Torque Limit M1 [1]</td><td colspan="8">Torque Limit M1 [0]</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>
</td><td>Byte 0 of the torque limit value for motor 1. This value is wrote on the RAM of the motor, for this the value is lost when the power is turned off. The default value is 1023</td></tr>
<tr><td>7</td><td>Torque Limit M1 [1]</td><td>Byte 1 of the torque limit value for motor 1. This value is wrote on the RAM of the motor, for this the value is lost when the power is turned off. The default value is 1023</td></tr>
<tr><td>8</td><td>Torque Limit Motor 2 [0]</td><td>These two bytes contain the value of the torque used by motor 2, from 0 to 100 %. The value range can be from 0 to 1023 with 0.1 % steps.</td><td>Byte 0 of the torque limit value for motor 2. This value is wrote on the RAM of the motor, for this the</td></tr>
</table>

<table>
<tr><td></td><td></td><td rowspan="2">The same example of bytes 6 and 7 applied to this motor.</td><td>value is lost when the power is turned off. The default value is 1023</td></tr>
<tr><td>9</td><td>Torque Limit Motor 2 [1]</td><td>Byte 1 of the torque limit value for motor 2. This value is wrote on the RAM of the motor, for this the value is lost when the power is turned off. The default value is 1023</td></tr>
<tr><td>10</td><td>Torque Limit Motor 3 [0]</td><td rowspan="2">These two bytes contain the value of the torque used by motor 3, from 0 to 100 %. The value range can be from 0 to 1023 with 0.1 % steps.

The same example of bytes 6 and 7 applied to this motor.</td><td>Byte 0 of the torque limit value for motor 3. This value is wrote on the RAM of the motor, for this the value is lost when the power is turned off. The default value is 1023</td></tr>
<tr><td>11</td><td>Torque Limit Motor 3 [1]</td><td>Byte 1 of the torque limit value for motor 3. This value is wrote on the RAM of the motor, for this the value is lost when the power is turned off. The default value is 1023</td></tr>
<tr><td>12</td><td>Set max torque M1 M2 M3</td><td>
<table>
<tr><td colspan="8">Set max torque M1 M2 M3 (8 bits)</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Z</td><td>Y</td><td>X</td></tr>
</table>

Truth table:
<table>
<tr><td>Z</td><td>Y</td><td>X</td><td>Meaning</td></tr>
<tr><td>NC</td><td>NC</td><td>1</td><td>Set Max torque on motor 1</td></tr>
<tr><td>NC</td><td>1</td><td>NC</td><td>Set Max torque on motor 2</td></tr>
<tr><td>1</td><td>NC</td><td>NC</td><td>Set Max torque on motor 3</td></tr>
</table>

Example: Set Ma x torque on Motor 1, Motor 2 and Motor 3
<table>
<tr><td colspan="8">Set max torque M1 M2 M3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
</table>
</td><td>Flag that indicates the system, the user wants to change the Max torque force used by the motors.</td></tr>
<tr><td>13</td><td>Max Torque M1 [0]</td><td>These two bytes contain the value of the maximum torque force used by the motor 1, from 0 to 100 %. The value range can be from 0 to 1023 with 0.1 % steps. .

Example: Maximum torque force used 50 % (512 o 0x200) on Motor 1
<table>
<tr><td colspan="8">Max Torque M1 [1]</td><td colspan="8">Max Torque M1 [0]</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>
</td><td>Byte 0 of the maximum torque value applied by motor 1. This value is wrote on the EEPROM of the motor, for this the value persist even when the motor Is</td></tr>
</table>

		0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	turned off. The default value is 1023.
14	Max Torque M1 [1]		Byte 1 of the maximum torque value applied by motor 1. This value is wrote on the EEPROM of the motor, for this the value persist even when the motor Is turned off. The default value is 1023.
15	Max Torque M2 [0]	These two bytes contain the value of the maximum torque force used by motor 2, from 0 to 100 %. The value range can be from 0 to 1023 with 0.1 % steps. . The example on bytes 13 and 14 apply for this register.	Byte 0 of the maximum torque value applied by motor 2. This value is wrote on the EEPROM of the motor, for this the value persist even when the motor Is turned off. The default value is 1023.
16	Max Torque M2 [1]		Byte 1 of the maximum torque value applied by motor 2. This value is wrote on the EEPROM of the motor, for this the value persist even when the motor Is turned off. The default value is 1023.
17	Max Torque M3 [0]	These two bytes contain the value of the maximum torque force used by motor 3, from 0 to 100 %. The value range can be from 0 to 1023 with 0.1 % steps. . The example on bytes 13 and 14 apply for this register.	Byte 0 of the maximum torque value applied by motor 3. This value is wrote on the EEPROM of the motor, for this the value persist even when the motor Is turned off. The default value is 1023.
18	Max Torque M3 [1]		Byte 1 of the maximum torque value applied by motor 3. This value is wrote on the EEPROM of the motor, for this the value persist even when the motor Is turned off. The default value is 1023.
19	0x00	-	Not used

20	0x00	-	Not used
21	0x00	-	Not used
22	0x00	-	Not used
23	0x00	-	Not used
24	0x00	-	Not used
25	0x00	-	Not used
26	0x00	-	Not used
27	0x00	-	Not used
28	0x00	-	Not used
29	0x00	-	Not used
30	0x00	-	Not used
31	0x7B	0x7B	Constant byte that means the end of the packet.

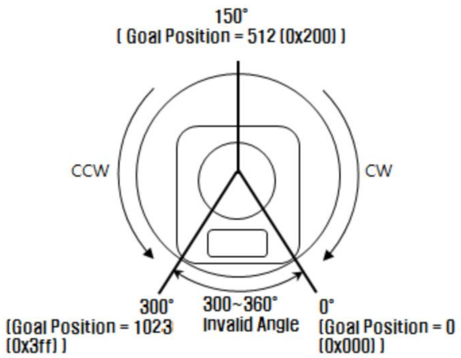
Table 17. Write packet 1 structure.

6.10.2 Write packet 2

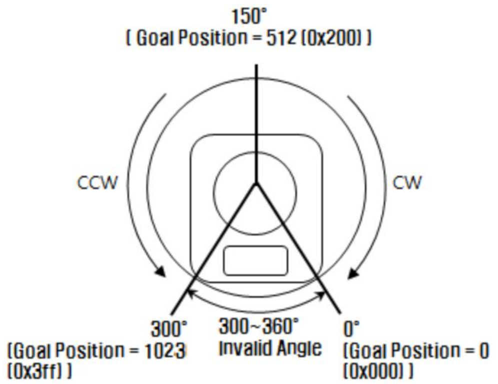
This packet is 32 bytes long and can be used to configure some options related to the movement of the motors, as angle limit for the CW and CCW movement, goal position in Joint mode, velocity to reach the goal position, and velocity for free spin in Wheel mode. The next table show the description of the bytes.

<table>
<tr><th>Byte position</th><th>Byte (Hexadecimal)</th><th>Available Options</th><th>Description</th></tr>
<tr><td>0</td><td>0x7A</td><td>0x7A</td><td>Constant start byte</td></tr>
<tr><td>1</td><td>0x02</td><td>0x02</td><td>Write packet number 2</td></tr>
<tr><td>2</td><td>Set angle limit M1 M2 M3</td><td>Angle limit CW is the inferior limit and Angle limit CCW is the upper limit. The range value are from 0 to 1023 and is used in Joint mode to limit the angle operation. The default value is 0 in all the motors because this value force the motor to stay in Wheel mode and have a free movement in the system. It is not recommended that the user modify this values because the FFT Gyro system is not designed for restrained movement.

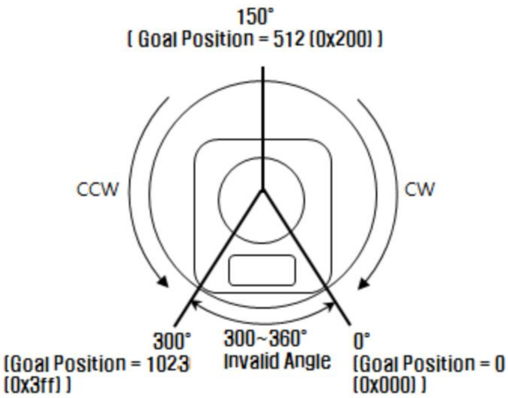
Image from the online manual for AX-12A



Byte description
<table>
<tr><td colspan="8">Set angle limit M1 M2 M3 (8 bits)</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>0</td><td>0</td><td>Z</td><td>Z</td><td>Y</td><td>Y</td><td>X</td><td>X</td></tr>
</table></td><td>This let the user indicate the system he wants to change the CW angle limit or CCW angle limit in the motors, following the structure of this byte.</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Truth table for Motor 1
<table>
<tr><td>X</td><td>X</td><td>Meaning</td></tr>
<tr><td>0</td><td>1</td><td>Set angle limits CW</td></tr>
<tr><td>1</td><td>0</td><td>Set angle limits CCW</td></tr>
<tr><td>1</td><td>1</td><td>Set angle limits CW and CCW</td></tr>
</table>
Truth table for Motor 2
<table>
<tr><td>Y</td><td>Y</td><td>Meaning</td></tr>
<tr><td>0</td><td>1</td><td>Set angle limits CW</td></tr>
<tr><td>1</td><td>0</td><td>Set angle limits CCW</td></tr>
<tr><td>1</td><td>1</td><td>Set angle limits CW and CCW</td></tr>
</table>
Truth table for Motor 3
<table>
<tr><td>Z</td><td>Z</td><td>Meaning</td></tr>
<tr><td>0</td><td>1</td><td>Set angle limits CW</td></tr>
<tr><td>1</td><td>0</td><td>Set angle limits CCW</td></tr>
<tr><td>1</td><td>1</td><td>Set angle limits CW and CCW</td></tr>
</table>
Example: Change angle limit CW on motor 1, angle limit CW and CCW on motor 2.
<table>
<tr><td colspan="8">Set angle limit M1 M2 M3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
</table>
</td><td></td></tr>
<tr><td>3</td><td>Minimum CW angle limit M1 [0]</td><td>Value range for this register is from 0 (0x000) to 1023 (0x3ff) with steps of 0.29 degrees. (0 to 300 degrees).</td><td>Byte 0 of the inferior limit of movement (CW) register of motor 1.</td></tr>
<tr><td>4</td><td>Minimum CW angle limit M1 [1]</td><td>Example: Minimum angle CW of 0.29 grades on Motor 1
<table>
<tr><td colspan="8">Minimum CW angle limit M1 [1]</td><td colspan="8">Minimum CW angle limit M1 [0]</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
</table>


</td><td>Byte 1 of the inferior limit of movement (CW) register of motor 1.</td></tr>
<tr><td>5</td><td>Minimum CW angle limit M2 [0]</td><td>Value range for this register is from 0 (0x000) to 1023 (0x3ff) with steps of 0.29 degrees. (0 to 300 degrees).</td><td>Byte 0 of the inferior limit of movement (CW) register of motor 2.</td></tr>
<tr><td>6</td><td>Minimum CW angle limit M2 [1]</td><td>The example on bytes 3 and 4 apply for this register.</td><td>Byte 1 of the inferior limit of movement (CW) register of motor 2.</td></tr>
<tr><td>7</td><td>Minimum CW angle limit M3 [0]</td><td>Value range for this register is from 0 (0x000) to 1023 (0x3ff) with steps of 0.29 degrees. (0 to 300 degrees).</td><td>Byte 0 of the inferior limit of movement (CW) register of motor 3.</td></tr>
<tr><td>8</td><td>Minimum CW angle limit M3 [1]</td><td>The example on bytes 3 and 4 apply for this register.</td><td>Byte 1 of the inferior limit of movement</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>(CW) register of motor 3.</td></tr>
<tr><td>9</td><td>Maximum CCW angle limit M1 [0]</td><td rowspan="2">Value range for this register is from 0 (0x000) to 1023 (0x3ff) with steps of 0.29 degrees. (0 to 300 degrees).


Example: Change Maximum angle limit (CCW) to 1000 on Motor 1
<table>
<tr><td colspan="8">Maximum CCW angle limit M1 [1]</td><td colspan="8">Maximum CCW angle limit M1 [0]</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
</table></td><td>Byte 0 of the upper limit of angle movement (CCW) register of motor 1.</td></tr>
<tr><td>10</td><td>Maximum CCW angle limit M1 [1]</td><td>Byte 1 of the upper limit of angle movement (CCW) register of motor 1.</td></tr>
<tr><td>11</td><td>Maximum CCW angle limit M2 [0]</td><td>Value range for this register is from 0 (0x000) to 1023 (0x3ff) with steps of 0.29 degrees. (0 to 300 degrees).</td><td>Byte 0 of the upper limit of angle movement (CCW) register of motor 2.</td></tr>
<tr><td>12</td><td>Maximum CCW angle limit M2 [1]</td><td>The example on bytes 9 and 10 apply for this register.</td><td>Byte 1 of the upper limit of angle movement (CCW) register of motor 2.</td></tr>
<tr><td>13</td><td>Maximum CCW angle limit M3 [0]</td><td>Value range for this register is from 0 (0x000) to 1023 (0x3ff) with steps of 0.29 degrees. (0 to 300 degrees).</td><td>Byte 0 of the upper limit of angle movement (CCW) register of motor 3.</td></tr>
<tr><td>14</td><td>Maximum CCW angle limit M3 [1]</td><td>The example on bytes 9 and 10 apply for this register.</td><td>Byte 1 of the upper limit of angle movement (CCW) register of motor 3.</td></tr>
<tr><td>15</td><td>Set moving speed M1 M2 M3</td><td>
<table>
<tr><td colspan="8">Set moving speed M1 M2 M3 (8 bits)</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Z</td><td>Y</td><td>X</td></tr>
</table>
Truth table:
<table>
<tr><td>Z</td><td>Y</td><td>X</td><td>Significado</td></tr>
<tr><td>NC</td><td>NC</td><td>1</td><td>Set moving speed on motor 1</td></tr>
<tr><td>NC</td><td>1</td><td>NC</td><td>Set moving speed on motor 2</td></tr>
<tr><td>1</td><td>NC</td><td>NC</td><td>Set moving speed on motor 3</td></tr>
</table>
Example: Set moving speed on Motor 1 and Motor 3
<table>
<tr><td colspan="8">Set moving speed M1 M2 M3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
</table></td><td>Flag that indicates a change in the moving speed of the motors.</td></tr>
<tr><td>16</td><td>Moving speed M1 [0]</td><td>This bytes contain the value of the moving speed register, and the range is:</td><td>Byte 0 of the moving speed value for motor 1.</td></tr>
</table>

<table>
<tr><td>17</td><td>Moving speed M1 [1]</td><td>Joint mode:
0 (0x00) to 1023 (0x3FF), with 0.111 rpm step units. For example if the value 1023 represent 114 rpm

Wheel mode:
0 (0x00) to 2047 (0x7FF), from 0 to 1023 is CCW movement direction, and 0 is stop. The range from 1024 to 2047 is CW movement direction, and 1024 is stop.

Example: Moving speed of 1023 (CCW movement) on Motor 1
<table>
<tr><td colspan="8">Moving speed M1 [1]</td><td colspan="8">Moving speed M1 [0]</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table></td><td>Byte 1 of the moving speed value for motor 1.</td></tr>
<tr><td>18</td><td>Moving speed M2 [0]</td><td rowspan="2">This bytes contain the value of the moving speed register, and the range is:

Joint mode:
0 (0x00) to 1023 (0x3FF), with 0.111 rpm step units. For example if the value 1023 represent 114 rpm

Wheel mode:
0 (0x00) to 2047 (0x7FF), from 0 to 1023 is CCW movement direction, and 0 is stop. The range from 1024 to 2047 is CW movement direction, and 1024 is stop.

The example for bytes 16 and 17 apply to this register.</td><td>Byte 0 of the moving speed value for motor 2.</td></tr>
<tr><td>19</td><td>Moving speed M2 [1]</td><td>Byte 1 of the moving speed value for motor 2.</td></tr>
<tr><td>20</td><td>Moving speed M3 [0]</td><td rowspan="2">This bytes contain the value of the moving speed register, and the range is:

Joint mode:
0 (0x00) to 1023 (0x3FF), with 0.111 rpm step units. For example if the value 1023 represent 114 rpm

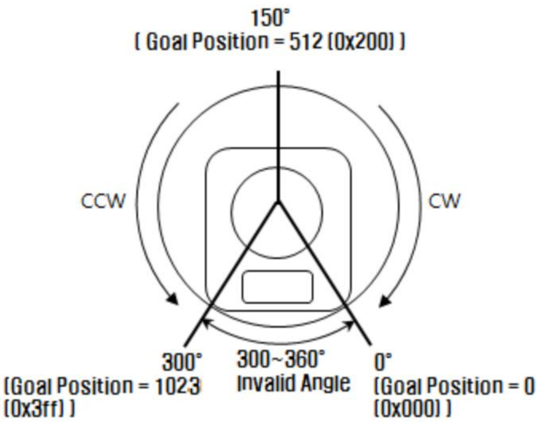
Wheel mode:
0 (0x00) to 2047 (0x7FF), from 0 to 1023 is CCW movement direction, and 0 is stop. The range from 1024 to 2047 is CW movement direction, and 1024 is stop.

The example for bytes 16 and 17 apply to this register.</td><td>Byte 0 of the moving speed value for motor 3.</td></tr>
<tr><td>21</td><td>Moving speed M3 [1]</td><td>Byte 1 of the moving speed value for motor 3.</td></tr>
<tr><td>22</td><td>Set position M1 M2 M3</td><td>
<table>
<tr><td colspan="8">Set position M1 M2 M3 (8 bits)</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Z</td><td>Y</td><td>X</td></tr>
</table>

<table>
<tr><td>Z</td><td>Y</td><td>X</td><td>Meaning</td></tr>
<tr><td>NC</td><td>NC</td><td>1</td><td>Set position on motor 1</td></tr>
<tr><td>NC</td><td>1</td><td>NC</td><td>Set position on motor 2</td></tr>
<tr><td>1</td><td>NC</td><td>NC</td><td>Set position on motor 3</td></tr>
</table>

Example: Set position on Motor 2 and Motor 3
<table>
<tr><td colspan="8">Set position M1 M2 M3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
</table></td><td>Flag that indicates a change in the position of the motors.</td></tr>
<tr><td>23</td><td>Position M1 [0]</td><td>This bytes contain the position value for register of the motor 1. The range goes from 0 to 1023 with steps of 0.29 degrees and is</td><td>Byte 0 of the position value for motor 1.</td></tr>
</table>

<table>
<tr><td>24</td><td>Position M1 [1]</td><td>limited by the CW and CCW angle limits. The default range is from 0 to 1023 and the changes of the position is only available in Joint mode.



Example: Change position to 512 (150 degrees) on Motor 1
<table>
<tr><td colspan="8">Position M1 [1]</td><td colspan="8">Position M1 [0]</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table></td><td>Byte 1 of the position value for motor 1.</td></tr>
<tr><td>25</td><td>Position M2 [0]</td><td rowspan="2">This bytes contain the position value for register of the motor 1. The range goes from 0 to 1023 with steps of 0.29 degrees and is limited by the CW and CCW angle limits. The default range is from 0 to 1023 and is only available in Joint mode.

The example of bytes 23 and 24 apply to this register.</td><td>Byte 0 of the position value for motor 2.</td></tr>
<tr><td>26</td><td>Position M2 [1]</td><td>Byte 1 of the position value for motor 2.</td></tr>
<tr><td>27</td><td>Position M3 [0]</td><td rowspan="2">This bytes contain the position value for register of the motor 1. The range goes from 0 to 1023 with steps of 0.29 degrees and is limited by the CW and CCW angle limits. The default range is from 0 to 1023 and is only available in Joint mode.

The example of bytes 23 and 24 apply to this register.</td><td>Byte 0 of the position value for motor 3.</td></tr>
<tr><td>28</td><td>Position M3 [1]</td><td>Byte 1 of the position value for motor 3.</td></tr>
<tr><td>29</td><td>0x00</td><td>-</td><td>Not used</td></tr>
<tr><td>30</td><td>0x00</td><td>-</td><td>Not used</td></tr>
<tr><td>31</td><td>0x7B</td><td>0x7B</td><td>Final byte</td></tr>
</table>

6.10.3 Write configuration packet

This packet is used when the user wants to configure the mode of the motors, turn on or turn off the motor leds or change the length of the packet sent by the FFT Gyro system to the computer.

<table>
<tr><th>Posición del byte</th><th>Byte (Hexadecimal)</th><th>Opciones</th><th>Descripción</th></tr>
<tr><td>0</td><td>0x7A</td><td>0x7A</td><td>Constant start byte</td></tr>
<tr><td>1</td><td>0x03</td><td>0x03</td><td>Write configuration packet</td></tr>
<tr><td>2</td><td>Set mode M1</td><td>1 ascii (‘1’): if the user wants to change the mode of the motor 1.
0 ascii (‘0’): if the user does not want to change the mode of the motor 1

Example: Change mode on motor 1
<table><tr><td>Set mode M1</td></tr></table></td><td>This flag tells the system that the user wants to change the mode of the motor 1.</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table></td><td></td></tr>
<tr><td>3</td><td>Set mode M2</td><td>1 ascii (‘1’): if the user wants to change the mode of the motor 1.
0 ascii (‘0’): if the user does not want to change the mode of the motor 1.

The example on byte 2 apply to this byte.</td><td>This flag tells the system that the user wants to change the mode of the motor 2.</td></tr>
<tr><td>4</td><td>Set mode M3</td><td>1 ascii (‘1’): if the user wants to change the mode of the motor 1.
0 ascii (‘0’): if the user does not want to change the mode of the motor 1.

The example on byte 2 apply to this byte.</td><td>This flag tells the system that the user wants to change the mode of the motor 3.</td></tr>
<tr><td>5</td><td>Mode M1</td><td>1 ascii (‘1’): Wheel mode
2 ascii (‘2’): Joint mode

Example: Select Joint mode for motor 1<table><tr><td colspan="8">Mode M1</td></tr><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table></td><td>This flag indicate which mode is selected for motor 1(‘1’ (0x31) for Wheel mode, and ‘2’(0x32) for Joint mode)</td></tr>
<tr><td>6</td><td>Mode M2</td><td>1 ascii (‘1’): Wheel mode
2 ascii (‘2’): Joint mode

The example on byte 5 apply to this byte.</td><td>This flag indicate which mode is selected for motor 1(‘1’ (0x31) for Wheel mode, and ‘2’(0x32) for Joint mode)</td></tr>
<tr><td>7</td><td>Mode M3</td><td>1 ascii (‘1’): Wheel mode
2 ascii (‘2’): Joint mode

The example on byte 5 apply to this byte.</td><td>This flag indicate which mode is selected for motor 1(‘1’ (0x31) for Wheel mode, and ‘2’(0x32) for Joint mode)</td></tr>
<tr><td>8</td><td>Set id M1</td><td>-</td><td>Not used</td></tr>
<tr><td>9</td><td>Set id M2</td><td>-</td><td>Not used</td></tr>
<tr><td>10</td><td>Set id M3</td><td>-</td><td>Not used</td></tr>
<tr><td>11</td><td>Id M1</td><td>-</td><td>Not used</td></tr>
<tr><td>12</td><td>Id M2</td><td>-</td><td>Not used</td></tr>
<tr><td>13</td><td>Id M3</td><td>-</td><td>Not used</td></tr>
<tr><td>14</td><td>Turnonoff led M1</td><td>1 ascii (‘1’):On
0 ascii (‘0’):Off

Example: Turn off of motor 1<table><tr><td colspan="8">Turnonoff led M1</td></tr><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table></td><td>Turn on or turn off the lead of motor 1.</td></tr>
<tr><td>15</td><td>Turnonoff led M2</td><td>1 ascii (‘1’):On
0 ascii (‘0’):Off

The example on byte 14 apply to this byte.</td><td>Turn on or turn off the led of motor 2.</td></tr>
<tr><td>16</td><td>Turnonoff led M3</td><td>1 ascii (‘1’):On
0 ascii (‘0’):Off

The example on byte 14 apply to this byte.</td><td>Turn on or turn off the led of motor 3.</td></tr>
<tr><td>17</td><td>0x00</td><td>-</td><td>Not used</td></tr>
<tr><td>18</td><td>0x00</td><td>-</td><td>Not used</td></tr>
</table>

19	0x00	-	Not used
20	0x00	-	Not used
21	0x00	-	Not used
22	0x00	-	Not used
23	0x00	-	Not used
24	0x00	-	Not used
25	0x00	-	Not used
26	0x00	-	Not used
27	0x00	-	Not used
28	0x00	-	Not used
29	0x00	-	Not used
30	Data mode packet	0x01: Simple 0x02: Extended Example: Enable the extended packet. Data mode packet 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0	This byte set the packet format sent by the FFT Gyro system to the computer: 1) Simple: 32 byte packet with the most important information for the system. 2) Extended: 64 byte packet with more information that can be useful for the user. Beware: Enabling this format can affect the latency of the information since is the length of the packet is twice (64 bytes) compared with the original format (simple).
31	0x7B	0x7B	Final constant byte.

Table 18. Write packet 2 structure.

7 Conclusion

The leap from development to implementation can be quite complex. Since UAVs have very fast and unstable dynamics, implementation comes with accidents or failures that risk the staff and equipment, and accidents cost money. The FFT GYRO removes those technical barriers that arise when implementing with real drones. This way you can focus your time, creativity, and energy on your development and progress.

With our system you can do all this and more:

- Protect your team and staff members from accidents.
- Avoid damage of expensive equipment due to collisions or malfunctions
- Accelerate the test phase by analyzing fewer variables (divide and conquer).
- Test and validate your project; innovate, experiment, and develop your research; tune, optimize and calibrate your solutions; and exhibit or display your drones in movement.
- Provide unique experience for customers to see real drones in movement and live.
- Exhibit, show and sell your drones with our patented technology.

In conclusion, having a multi-purpose tool like the FFT GYRO is essentially useful for a company that works with drones or UAVs. It comes out as a solution to a wide variety of problems in the UAV and VTOL field.

Thank you

END