Smart 3D Pen for Engineering Drawing

Prepared by:

Kareem Salah Metwaly Mahmoud	1501038
Omar Nasr Eldien Mohamad Mohamad	1500933
Mohamad Khaled Awaad Metwaly	1501199
Fatma Abd El moniem Mohamed madian	1501003
Shimaa Mahmoud Mohamed Ali	1500723
Sara Khaled sayed abo elnour	1500639

Supervised by Doctor:

Dr Watheq El-kharashi

Dr Ali El-moursy

Table Of Content

Table of contents

Table of contents

List of Figures

List of figures

1 ABSTRACT

This project is to design a smart 3-D pen that will assist the Engineering drawing with a handy,smart and easy to use tool.The touhes and movements of engineer in the free space will outomatically be converted to a three dimensional drwaing on the Computer Aided Design (CAD) software application. The engineer will be able to realize his design and import it to the computer system witout the need to learnthe sophisticated design and drawing tools such as 3D studio or Autocad.

Chapter 1: Introduction

Part 1: Software

1.1 Introduction For Software

Our project needs a software to draw in 3D with the ability to communicate with outer micro controller and this Software must communicate fast to get a real time Result.

There are many software tool can be used like python and processing. We used in our project processing.

1.1.1 Processing

Processing is an open-source graphical library and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology.

Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations. It also provides a graphical user interface for simplifying the compilation and execution stage.

The Processing language and IDE have been the precursor to other projects including Arduino, Wiring and p5.js.

1.1.1.1 Features

Processing includes a *sketchbook*, a minimal alternative to an integrated development environment (IDE) for organizing projects.

Every Processing sketch is actually a subclass of the PApplet Java class (formerly a subclass of Java's built-in Applet) which implements most of the Processing language's features.

When programming in Processing, all additional classes defined will be treated as inner classes when the code is translated into pure Java before compiling. This means that the use of static variables and methods in classes is prohibited unless Processing is explicitly told to code in pure Java mode.

Processing also allows for users to create their own classes within the PApplet sketch. This allows for complex data types that can include any number of arguments and avoids the limitations of solely using standard data types such as: int (integer), char (character), float (real number), and color (RGB, RGBA, hex).

And the most important features are:

1.1.1.1.1 Flexibility

Like a software utility belt, Processing consists of many tools that work together in different combinations. As a result, it can be used for quick hacks or for in-depth research. Because a Processing program can be as short as one line or as long as thousands, there's room for growth and variation. More than 100 libraries extend Processing even further into domains including sound, computer vision, and digital fabrication.

1.1.1.1.2 Giants

People have been making pictures with computers since the 1960s, and there's much to be learned from this history. In life, we all stand on the shoulders of giants, and the titans for Processing include thinkers from design, computer graphics, art, architecture, statistics, and the spaces between. Have a look at Ivan Sutherland's Sketchpad (1963), Alan Kay's Dynabook (1968), and the many artists featured in Ruth Leavitt's Artist and Computer 1 (Harmony Books, 1976). The ACM SIGGRAPH archives provide a fascinating glimpse into the history of graphics and software.

1.1.1.1.3 Family Tree

Like human languages, programming languages belong to families of related languages. Processing is a dialect of a programming language called Java; the language syntax is almost identical, but Processing adds custom features related to graphics and interaction (Figure 1.1.1.2.3). The graphic elements of Processing are related to PostScript (a foundation of PDF) and OpenGL (a 3D graphics specification).

Because of these shared features, learning Processing is an entry-level step to programming in other languages and using different software tools.

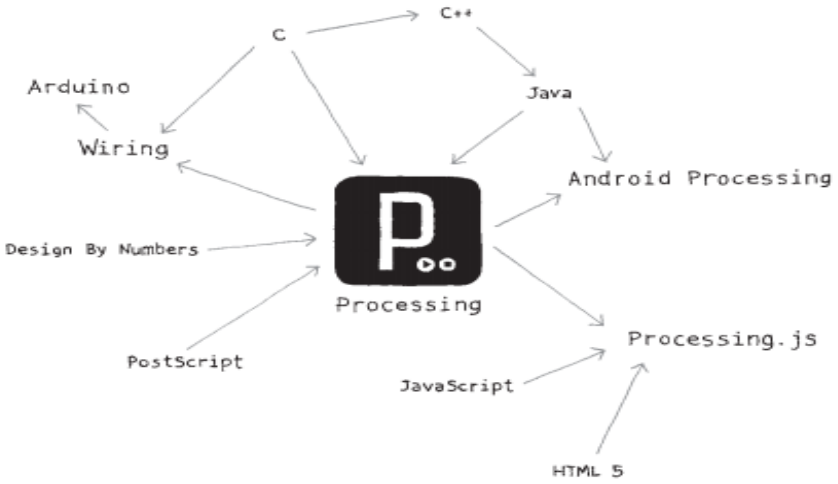


Fig 1.1.1: Family Tree Of Processing.

Part 2:Hardware

1.2 Introduction For Hardware

This project main objective is 3D drawing in real time on a CAD like files.
It helps the engineers and facilitates their work with the 3D drawing .Earlier it takes a lot of time to use 3D software drawing to create their vision of their work.so we tried to create a tool that not only helps engineers but also to reduce the time needed to complete their work.
This project is yet to be complete but it is a start in the field with many flaws that need to be enhanced with a lot of work in the future.

This project is divided into two parts Hardware and Software. The hardware part is responsible for sensing, operating on data and then sending it to the software. This mainly done using IMU and sensor fusion technique

The hardware used in this project is MetaMotionR – MMR. This board is a mini MCU (Microcontroller Unit) which also contain IMU (**Inertial measurement unit**) sensors and internal Bluetooth module. The board includes internally a sensor fusion algorithm that is applied on the output of the IMU sensors

1.2.1 IMU

An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes and sometimes magnetometers. IMUs are typically used to maneuver aircraft (an attitude and heading reference system), including unmanned aerial vehicles (UAVs), among many others, and spacecraft, including satellites and landers. Recent developments allow to produce IMU-enabled GPS devices. An IMU allows a GPS receiver to work when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interference is present.

1.2.1.1 Accelerometer

Accelerometers are devices that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared (m/s^2) or in G-forces (g). A single G-force for us here on planet Earth is equivalent to 9.8

m/s^2, but this does vary slightly with elevation (and will be a different value on different planets due to variations in gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications. Accelerometers are electromechanical devices that sense either static or dynamic forces of acceleration. Static forces include gravity, while dynamic forces can include vibrations and movement. Accelerometers can measure acceleration on one, two, or three axes. 3-axis units are becoming more common as the cost of development for them decreases.

1.2.1.2 Gyroscope

A **gyroscope** is a device used for measuring or maintaining orientation and angular velocity. It is a spinning wheel or disc in which the axis of rotation (spin axis) is free to assume any orientation by itself. When rotating, the orientation of this axis is unaffected by tilting or rotation of the mounting, according to the conservation of angular momentum.

Gyroscopes based on other operating principles also exist, such as the microchip-packaged MEMS gyroscopes found in electronic devices, solid-state ring lasers, fiber optic gyroscopes, and the extremely sensitive quantum gyroscope.

Gyroscope sensors are also called as Angular Rate Sensor or Angular Velocity Sensors. These sensors are installed in the applications where the orientation of the object is difficult to sense by humans. Measured in degrees per second, angular velocity is the change in the rotational angle of the object per unit of time.

1.2.1.3 Magnetometer

A **magnetometer** is a device that measures magnetism—the direction, strength, or relative change of a magnetic field at a location. The measurement of the magnetization of a magnetic material (like a ferromagnet) is an example. A compass is one such device, one that measures the direction of an ambient magnetic field, in this case, the Earth's magnetic field.

A magnetometer is a scientific instrument used to measure the strength and direction of the magnetic field in the vicinity of the instrument. Magnetism varies from place to place because of differences in Earth’s magnetic field caused by the differing nature of rocks and the interaction between charged particles from the sun and the magnetosphere of a planet

Chapter 2: Related Work

Part 1: Software

2.1 Software Related Work

2.1.1 Libraries

Processing supports more than (100) libraries as I mentioned before But we used in our project some of these libraries such as:

2.1.1.1 Peasy

PeasyCam provides a dead-simple mouse-driven camera for Processing, where a mouse left-drag will rotate the camera around the object, a right drag will zoom in. A double-click restores the camera to its original position. The shift key constraints rotation and panning to one axis or the other.

The PeasyCam is positioned on a sphere whose radius is given distance from the look-at point. Rotations are around axes that pass through the looked-at point.

Two ways to initialize:

- PeasyCam(PApplet parent, double lookAt_X, double lookAt_Y, double lookAt_Z, double distance);
- PeasyCam(PApplet parent, double distance); // look at 0,0,0

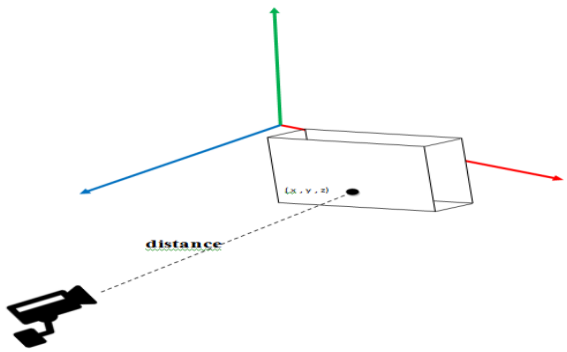


Fig 2.1.1: Initialize of Peasy

2.1.1.2 OSC P5

Open Sound Control (OSC) is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology and has been used in many application areas and oscP5 is an OSC implementation for programming environment processing.

2.1.1.2.1 Features

- **Automatic event detection:** oscP5 locates functions inside your sketch and will link incoming OSC messages to matching functions automatically oscP5Plug. Incoming OSC messages can easily be captured within your sketch by implementing the oscEvent function oscP5message.
- **Network protocols:** oscP5 supports TCP, UDP and Multicast. Networking operations are handled by package netP5 which can also be used on its own for various networking operations which do not necessarily require OSC as protocol.

2.1.1.3 Nervous system OBJ Export

This is a library to export meshes from Processing as OBJ or X3D files. It can export color meshes with triangle and quad shaped faces as an OBJ or X3D with a PNG texture map. It can also export meshes with faces with an arbitrary number of sides (without color support). The OBJExport library is used the same way the PDF library is used.

OBJExport works with beginRecord(), beginRaw(), and createGraphics(). The OBJ format and the exporter work with arbitrarily sided polygon, not just triangles. This allows for the versatility to export quad or hex based meshes not just triangles. If you require triangles but are drawing with polygons instead,

You can use beginRaw () to get the tessellated output of Processing rather than the original polygons.
The library also supports color mesh export creating a .png texture and a .mtl material file to go along with the obj. Color mode is not enabled by default. Use setColor (true) on the OBJExport object before beginDraw () to turn on color export.

In our project we use it for saving the user's drawing as an OBJ file so the user can display his/her drawing as a 3D model.

2.1.1.4 G4P (GUI 4 Processing)

The G4P library provides an extensive collection of 2D GUI controls for your sketch and comes with many example sketches to showcase these controls.

In our project we used it to make some operations to be easy by using some Buttons such as (Save - Delete - Undo - Exit) and box to display coordinates of points.

2.1.2 Render techniques

By default, everything is drawn to the primary display window. Sometimes, however, there is an advantage in drawing to another graphics surface. All of the drawing features available in the display window can be applied to an off screen drawing surface and then drawn back into the display window as an image or texture. This technique makes it easier to imagine a program as a stack of layers similar to the technique used in photo editing and vector drawing software. Similarly, drawing surfaces in Processing can be moved around, drawn using blending effects and transparency, and drawn in different orders to change how the layers combine. Before the discussion moves to multiple drawing surfaces, this chapter starts with a discussion of the different renderers used by Processing.

Processing has three primary renderers: the default renderer, P3D, and P2D Switching to P2D or P3D is advisable given one of the following scenarios:

1. **You are drawing in 3D**
   In three-dimensional space, a third axis (the z-axis) refers to the depth of any given point. How far in front or behind the window does a pixel live? Now, we

all know there are no actual pixels floating in the air in front of or behind your screen! What we're talking about here is how to use the theoretical z-axis to create the illusion of three-dimensional space in your Processing window. P3D is required for this.

2. **You are drawing in 2D**
   Use a particular graphic effect not available in default renderer: Some graphics functions are only available in P3D such as textures and lighting.

In our project we use P3D mode (because all drawing- in three-dimensional space) .

2.1.3 Built-in Functions

Processing has among of built-in functions but in the following we discuss the functions that we used in our project.

2.1.3.1 setup()

The setup() function is run once, when the program starts. It's used to define initial environment properties such as screen size and to load media such as images and fonts as the program starts. There can only be one setup () function for each program and it shouldn't be called again after its initial execution.If the sketch is a different dimension than the default, the size() function or fullScreen() function must be the first line in setup ().

2.1.3.2 draw()

The draw() function continuously executes the lines of code contained inside its block until the program is stopped or noLoop() is called. draw() is called automatically and should never be called explicitly. All Processing programs update the screen at the end of draw(), never earlier.

2.1.3.3 background()

The background() function sets the color used for the background of the Processing window.

2.1.3.4 stroke()

Sets the color used to draw lines and borders around shapes. This color is either specified in terms of the RGB or HSB color depending on the current colorMode(). The default color space is RGB, with each value in the range from 0 to 255.

2.1.3.5 fill()

Sets the color used to fill shapes. This color is either specified in terms of the RGB or HSB color depending on the current colorMode(). The default color space is RGB, with each value in the range from 0 to 255.

2.1.3.6 line()

Draws a line (a direct path between two points) to the screen. The version with six parameters allows the line to be placed anywhere within XYZ space. Drawing this shape in 3D with the z parameter requires the P3D parameter in combination with size().

2.1.3.7 translate()

Specifies an amount to displace objects within the display window. The x parameter specifies left/right translation, the y parameter specifies up/down translation, and the z parameter specifies translations toward/away from the screen. Using this function with the z parameter requires using P3D as a parameter in combination with size().

2.1.3.8 rotate()

Rotates the amount specified by the angle parameter. Angles must be specified in radians (values from 0 to TWO_PI), or they can be converted from degrees to radians with the radians() function.

2.1.3.8.1 rotateX()

Rotates around the x-axis the amount specified by the angle parameter. Angles should be specified in radians (values from 0 to TWO_PI) or converted from degrees to radians with the radians() function. Coordinates are always rotated around their relative position to the origin.

2.1.3.8.2 rotateY()

Rotates around the Y-axis the amount specified by the angle parameter. Angles should be specified in radians (values from 0 to TWO_PI) or converted from degrees to radians with the radians() function. Coordinates are always rotated around their relative position to the origin.

2.1.3.8.3 rotateZ()

Rotates around the Z-axis the amount specified by the angle parameter. Angles should be specified in radians (values from 0 to TWO_PI) or converted from degrees to radians with the radians() function. Coordinates are always rotated around their relative position to the origin.

2.1.3.9 BeginShape() & endShape()

Using the beginShape() and endShape() functions allow creating more complex forms. beginShape() begins recording vertices for a shape and endShape() stops recording. After calling the beginShape() function, a series of vertex() commands must follow. To stop drawing the shape, call endShape().

2.1.3.10 vertex()

All shapes are constructed by connecting a series of vertices. vertex() is used to specify the vertex coordinates for points, lines, triangles, quads, and polygons. It is used exclusively within the beginShape() and endShape() functions.All shapes are constructed by connecting a series of vertices. vertex() is used to specify the vertex coordinates for points, lines, triangles, quads, and polygons. It is used exclusively within the beginShape() and endShape() functions. Drawing a vertex in 3D using the z parameter requires the P3D parameter in combination with size.

2.1.3.11 text()

Draws text to the screen. Displays the information specified in the first parameter on the screen in the position specified by the additional parameters.

Part 2: Harware

2.2 Hardware Related Work

2.2.1 MetaMotionR board (MMR)

The board includes:

- an ARM MCU,
- Bluetooth Low Energy,
- memory,
- an LED,
- a push button,
- a coin cell or rechargeable battery.
- IMU sensors

2.2.1.1 Circuit and Sensor Details

Accelerometer / Gyroscope
The 6-axis accelerometer and gyroscope sensor BMI160 attached via the SPI bus, with pin numbers detailed in the module to SoC mapping table.
Magnetometer
The 3-axis magnetometer BMM150 is attached via the SPI bus, with pin numbers detailed in the module to SoC mapping table.

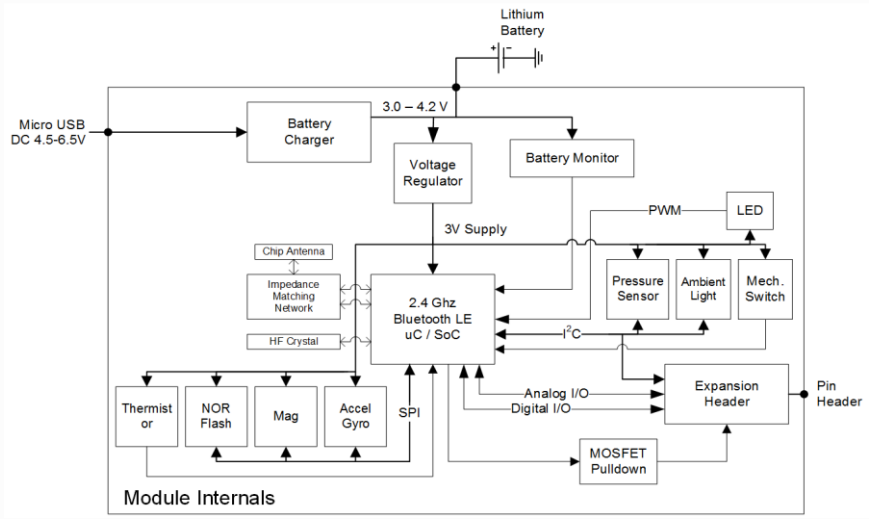


Fig 2.2.1 Block Diagram

Pin Assignments

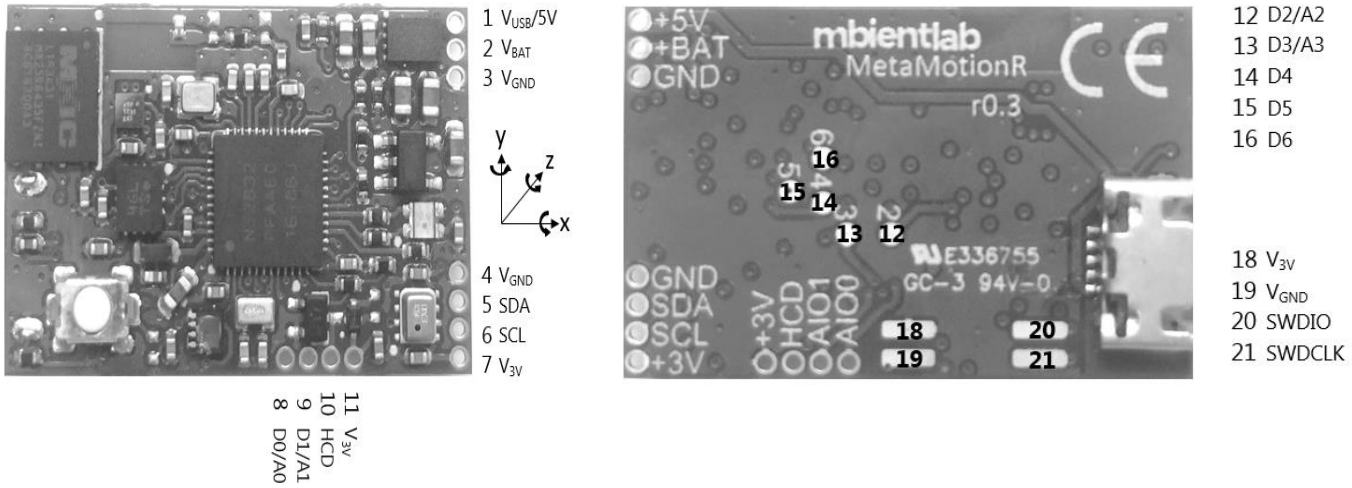


Fig 2.2.2 a) and b) pin assignments

Pin	Pin Name	Function	Description
Power Supply			
1	V_{USB}	Power	Positive supply alternative to micro USB port.
2	V_{BAT}	Power	Positive battery connection.
3	V_{GND}	Power	Ground connection intended for battery.
Digital Peripheral Bus			
4	V_{GND}	Power	Supplementary ground for peripheral expansion.
5	SDA	Digital I/O	I^2C serial data.
6	SCL	Digital I/O	I^2C serial clock.
7	V_{3V}	Power	Regulated 3V output for peripheral expansion.

Fig 2.2.3 pin functions

Module Pin	nF52832 Pin	Function	Description
GPIO			
D0/A0	P0.02	GPIO	General purpose I/O and analog input.
D1/A1	P0.05	GPIO	General purpose I/O and analog input.
D2/A2	P0.31	GPIO	General purpose I/O and analog input.
D3/A3	P0.04	GPIO	General purpose I/O and analog input.
D4	P0.15	GPIO	General purpose I/O.
D5	P0.17	GPIO	General purpose I/O.
D6	P0.21	GPIO	General purpose I/O.
Internal Module Pins			
	P0.06	Switch	Micro Push Button. Active Low.
	P0.16	5V Present	Active high indicator of an attached power source.
	P0.30	Battery Voltage	Analog voltage at V_{BAT}*(10.0/14.02)
	P0.20	Battery Charging	Open drain, active low indicator that the charger is active.
	P0.07	Thermistor En	Active High Enable for Thermistor
	P0.03	Themistor Voltage	Voltage output from Thermistor Voltage Divider
	P0.25	Acc/Gyro Interrupt	INT1 Pin of BMI160.
	P0.11	Acc/Gyro Interrupt	INT2 Pin of BMI160.
	P0.28	Acc/Gyro SCK	SCK Pin of BMI160, BMM150, W25Q64FV.
	P0.26	Acc/Gyro MISO	MISO Pin of BMI160, BMM150, W25Q64FV.
	P0.27	Acc/Gyro MOSI	MOSI Pin of BMI160, BMM150, W25Q64FV.
	P0.29	Acc/Gyro nCS	nCS Pin of BMI160.
	P0.23	Mag Interrupt	INT Pin of BMM150.
	P0.22	Mag Data Ready	DRDY Pin of BMM150.
	P0.24	Mag nCS	nCS Pin of BMM150
	P0.09	NOR Flash nCS	nCS Pin of W25Q64FV.
	P0.08	NOR Flash nWP	nWP Pin of W25Q64FV.
	P0.13	LED Red nEn	Current Sink for Red LED Channel.
	P0.14	LED Green nEn	Current Sink for Green LED Channel.

Fig 2.2.4 Module to SoC Pin Mapping

Spec	Description	Min.	Typ.	Max.	Units
V_{USB}	USB charging voltage.	-0.3		+7.0	V
V_{BAT}	Lithium battery voltage.	-0.3		+6.0	V
V_{GND}	Ground voltage			0	V
V_{IO}	I/O Pin Voltage	-0.3		+3.3	V
T_{MAX}	Storage Temperature	-40		125	°C

Fig 2.2.5 Absolute Maximum Ratings

Spec	Description	Min.	Typ.	Max.	Units
V_{USB}	USB charging voltage.	4.0	5.0	6.0	V
V_{BAT}	Lithium battery voltage.	3.0	3.7	4.2	V
T_A	Operating temperature.	-40	25	85	°C
I_{IDLE}	Idle current consumption.		20	50	uA

Fig 2.2.6 Operating Conditions

Spec	Description	Min.	Typ.	Max.	Units
V_{IH}	Input high voltage.	2.1		3.0	V
V_{IL}	Input low voltage.	0		0.9	V
V_{OH}	Output high voltage.	2.6		3.0	V
V_{OL}	Output low voltage.	0		0.4	V
R_{PU}	Pull-up resistance.	11	13	16	kΩ
R_{PD}	Pull-down resistance.	11	13	16	kΩ

Fig 2.2.7 General Purpose I/O (GPIO) Specifications

Spec	Description	Min.	Typ.	Max.	Units
	Measurement range.	±2		±16	g
	Resolution.	2048		16384	counts/g
f_{DATA}	Data sample frequency.	0.78		1600	Hz
$I_{12.5}$	Low data rate current (3.125 Hz).		5		uA
I_{100}	Mid data rate current (100 Hz).		24		uA
I_{1600}	High data rate current (1600 Hz).		180	300	uA
$I_{STANDBY}$	Standby current.		3	10	uA

Fig 2.2.8 Accelerometer (ACCEL) Specifications

Spec	Description	Min.	Typ.	Max.	Units
	Measurement range.	±125		±2000	°/s
	Resolution.	16		262	counts/°
f_{DATA}	Data sample frequency.	25		3200	Hz
I_{GYRO}	Gyro active current. All Data Rates.		850	900	uA
$I_{STANDBY}$	Standby current. Included in Accel Standby Current.				

Fig 2.2.9 Gyroscope (GYRO) Specifications

Spec	Description	Min.	Typ.	Max.	Units
	Measurement range.	±1200	±1300		uT
	Heading Accuracy.			±2.5	°
f_{DATA}	Data rate.		25	300	Hz
$I_{STANDBY}$	Standby current.		1	3	uA

Fig 2.2.10 Magnetometer (MAG) Specifications

Spec	Description	Min.	Typ.	Max.	Units
f_{NDOF}	NDoF operating frequency.	-1%	100	+1%	Hz
$f_{IMUPLUS}$	IMUPlus operating frequency.	-1%	100	+1%	Hz
$f_{COMPASS}$	Compass operating frequency.	-1%	25	+1%	Hz
f_{M4G}	M4G operating frequency.	-1%	50	+1%	Hz

Fig 2.2.11 Sensor Fusion (SF) Specifications

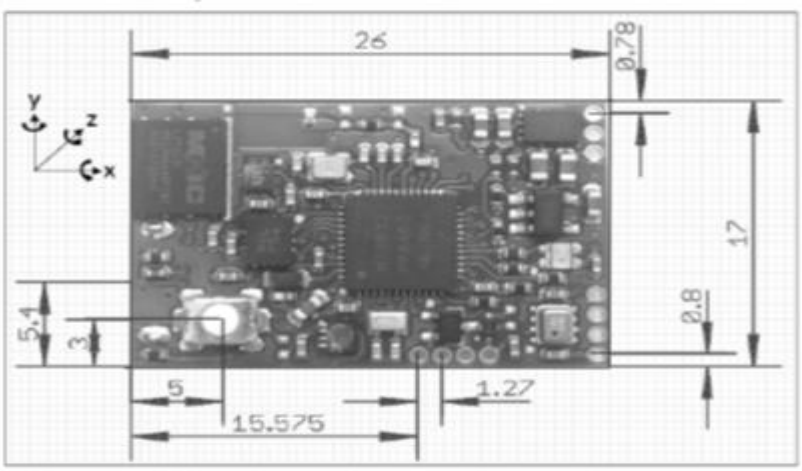


Fig 2.2.12 Mechanical Specifications

2.2.2 IMU

An IMU is a specific type of sensor that measures angular rate, force and sometimes magnetic field. IMUs are composed of a 3-axis accelerometer and a 3-axis gyroscope, which would be considered a 6-axis IMU. They can also include an additional 3-axis magnetometer, which would be considered a 9-axis IMU. Technically, the term "IMU" refers to just the sensor, but IMUs are often paired with sensor fusion software which combines data from multiple sensors to provide measures of orientation and heading. In common usage, the term "IMU" may be used to refer to the combination of the sensor and sensor fusion software. IMUs are used to measure acceleration, angular velocity and magnetic fields, and, when combined with sensor fusion software, they can be used to determine motion, orientation and heading. They are found in many applications across consumer electronics and the industrial sector.

Where do you find IMUs?

Common applications for IMUs include determining direction in a GPS system, tracking motion in consumer electronics such as cell phones and video game remotes, or following a user's head movements in AR (augmented reality) and VR (virtual reality) systems. This motion and orientation information also apply to maintaining a drone's balance, improving the heading of your robot vacuum cleaner, and other IoT and connected home devices.

In industrial use, you may use IMUs to align and measure positioning of equipment like antennas. IMUs are also used to help maneuver aircraft, with or without a

manned pilot. In the consumer airspace, some in-flight entertainment systems use IMUs in their remotes to add accessibility in addition to touch. A similar approach can be seen with LG Smart TV remotes which allow users to control the TV's user interface with an intuitive point-and-click approach instead of directional buttons.
Future applications will likely see tighter integration and fusion of the IMU with technologies like GPS (Global Positioning System), RF (Radio Frequency), and LiDAR (Light Detection And Ranging), which will enable accurate localization of people, vehicles and equipment (including autonomous ones), both indoors and outdoors.

How does an IMU work?

An IMU provides 2 to 6 DOF (Degrees of Freedom), which refers to the number of different ways that an object is able to move throughout 3D space. The maximum possible is 6 DOF, which would include 3 degrees of translation (flat) movement across a straight plane/along each axis (front/back, right/left, up/down) and 3 degrees of rotational movement across the x, y and z axes/about each axis.

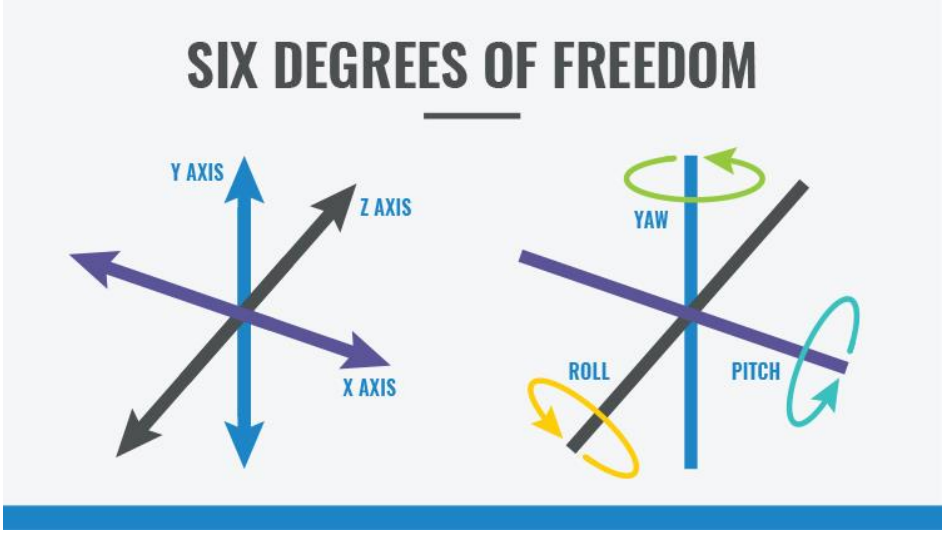


Fig 2.2.13 degree of freedom

The raw data collected from an IMU gives some idea of the world around it, but that information can also be processed for additional insight. Sensor fusion is the (mathematical) art of combining the data from each sensor in an IMU to create a more complete picture of the device's orientation and heading. For instance, while looking at gyroscope information for rotational motion, you can incorporate an accelerometers sense of gravity to create a reference frame. You can additionally add information about the Earth's magnetic field to align the whole sensor to the Earth's frame.

3 Main Types of Motion Sensor Devices

- **Accelerometer:** The most used type of motion sensor is the accelerometer. It measures acceleration (change of velocity) across a single axis, like when you step on the gas in your car or drop your phone. Accelerometers measure linear acceleration in a direction. An accelerometer can also be used to measure gravity as a downward force. Integrating acceleration once reveals an estimate for velocity and integrating again gives you an estimate for position. Due to the double integration and the state of today's technology, an accelerometer is not a recommended method of distance estimation.

- **Gyroscope:** While accelerometers can measure linear acceleration, they cannot measure twisting or rotational movement. Gyroscopes, however, measure angular velocity about three axes: pitch (x axis), roll (y axis) and yaw (z axis). When integrated with sensor fusion software, a gyro can be used to determine an object's orientation within 3D space. While a gyroscope has no initial frame of reference (like gravity), you can combine its data with data from an accelerometer to measure angular position. For an in-depth look at the different types of gyroscopes.

- **Magnetometer:** A magnetometer, as the name suggests, measures magnetic fields. It can detect fluctuations in Earth's magnetic field, by measuring the air's magnetic flux density at the sensor's point in space. Through those fluctuations, it finds the vector towards Earth's magnetic North. This can be fused in conjunction with accelerometer and gyroscope data to determine absolute heading. As you have seen, IMUs are used to measure acceleration, angular velocity and magnetic fields, and, when combined with sensor fusion software, they can be used to determine motion, orientation and heading. They are found in many applications across consumer electronics and the industrial sector.

2.2.2.1 Accelerometer

An accelerometer is a device that measures the vibration, or acceleration of motion of a structure. The force caused by vibration or a change in motion (acceleration) causes the mass to "squeeze" the piezoelectric material which produces an electrical charge that is proportional to the force exerted upon it. Since the charge is proportional to the force, and the mass is a constant, then the charge is also proportional to the acceleration.

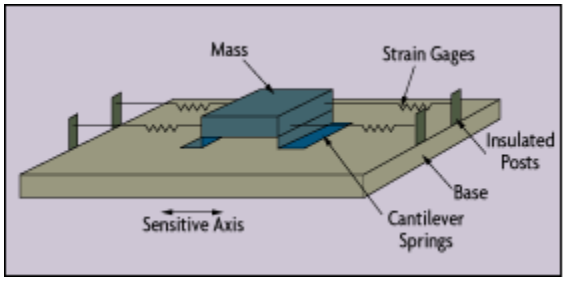


Fig 2.2.14 Accelerometer

2.2.2.1.1 Accelerometer Types

There are two types of piezoelectric accelerometers (vibration sensors):

The first type is a "high impedance" charge output accelerometer. In this type of accelerometer, the piezoelectric crystal produces an electrical charge which is connected directly to the measurement instruments. The charge output requires special accommodations and instrumentation most found in research facilities. This type of accelerometer is also used in high temperature applications (>120C) where low impedance models cannot be used.

The second type of accelerometer is a low impedance output accelerometer. A low impedance accelerometer has a charge accelerometer as its front end but has a tiny built-in micro-circuit and FET transistor that converts that charge into a low impedance voltage that can easily interface with standard instrumentation. This type of accelerometer is commonly used in industry. An accelerometer power supply like the ACC-PS1, provides the proper power to the microcircuit 18 to 24 V @ 2 mA constant current and removes the DC bias level, they typically produces a zero based output signal up to +/- 5V depending upon the mV/g rating of the

accelerometer. All OMEGA(R) accelerometers are this low impedance type.
Accelerometer Specifications

Dynamic Range is the +/- maximum amplitude that the accelerometer can measure before distorting or clipping the output signal. Typically specified in g's.

Frequency Response is determined by the mass, the piezoelectric properties of the crystal, and the resonance frequency of the case. It is the frequency range where the output of the accelerometer is within a specified deviation, typically +/- 5%. g 1g is the acceleration due to the earth's gravity which is 32.2 ft/sec2, 386 in/sec2 or 9.8 m/sec2.
Sensitivity is the output voltage produced by a certain force measured in g's. Accelerometers typically fall into two categories - producing either 10 mV/g or 100 mV/g. The frequency of the AC output voltage will match the frequency of the vibrations. The output level will be proportional to the amplitude of the vibrations. Low output accelerometers are used to measure high vibrational levels while high output accelerometers are used to measure low level vibrations.

2.2.2.1.2 Other Considerations when choosing an Accelerometer

The mass of the accelerometers should be significantly smaller than the mass of the system to be monitored. The accelerometer dynamic range should be broader than the expected vibration amplitude range of the sample. The frequency range of the accelerometer should fit the expected frequency range. The Sensitivity of the accelerometer should produce an electrical output compatible with existing instrumentation. Use a low sensitivity accelerometer to measure high amplitude vibrations and conversely use a high sensitivity accelerometer to measure low amplitude vibrations.

2.2.2.2 Gyroscope

A gyroscope is defined as a device which uses the earth’s gravity to help determine orientation. It is termed as gyro-sensors when it comes to smartphones. They measure the rate of rotation around the device’s x, y and z axes.

2.2.2.2.1 Six Degrees of Freedom

six degrees of freedom, or 6DOF. That is, the freedom of movement experienced by a user in 3D space. 6DOF is commonly represented by 3 translations – up/down, right/left, forward/backward, and 3 rotations – pitch, yaw, and roll.
The freedom of movement that can be achieved in 3D space is important for virtual reality, enabling greater ease in tracking the movement of users and levels of interactivity within the environment.
A gyroscope is defined as a device which uses the earth's gravity to help determine orientation. It is termed as gyro-sensors when it comes to smartphones.They measure the rate of rotation around the device's x, y and z axes.Gyro sensors, also known as angular rate sensors or angular velocity sensors, are devices that sense angular velocity.

2.2.2.3 Magnetometer

A magnetometer is a device that measures magnetism—the direction, strength, or relative change of a magnetic field at a particular location. The measurement of the magnetization of a magnetic material (like a ferromagnet) is an example. A compass is one such device, one that measures the direction of an ambient magnetic field, in this case, the Earth's magnetic field.

Magnetometers are widely used for measuring the Earth's magnetic field, and in geophysical surveys, to detect magnetic anomalies of various types. In an aircraft's attitude and heading reference system, they are commonly used as a heading reference. Magnetometers are also used in the military to detect submarines. Consequently, some countries, such as the United States, Canada and Australia, classify the more sensitive magnetometers as military technology, and control their distribution.

Chapter 3: Design

Part 1: Software

3.1 Software Design

3.1.1 Software Application flow

In the following section we will illustrate the design of the application and how it responds to certain events

The sketch window contain the following

- 3D grid to draw on
- Pointer viewed as a red sphere indicating the pen position on the sketch
- Camera to enable rotation and zooming on the sketch (grid)
- Rectangle to write the pen current position
- Menu bar that can be accessed using the GUI mode, which include the following functions
    - Save, to save the sketch as a object file
    - Undo, to erase the last point sent from the data points list
    - Delete, to empty the list and remove the drawn objects
    - Exit, to close the processing sketch

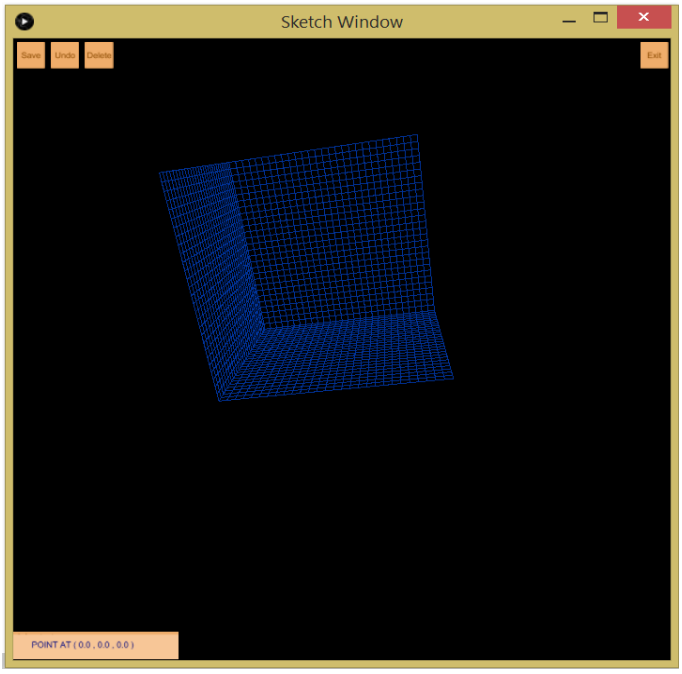


Fig 3.0: Sketch Window

3.1.2 Data retrieval

Both the hardware and software parties communicate using OSC library, where the pen sends the data information on a port while the processing application keeps listening to the port for new information.

The data is sent as a string while every information is followed by a space to indicate the end of the information.

The data follows this format (draw flag – erase flag – X_coordinate -- Y_coordinate – Z_coordinate – GUI_mode)

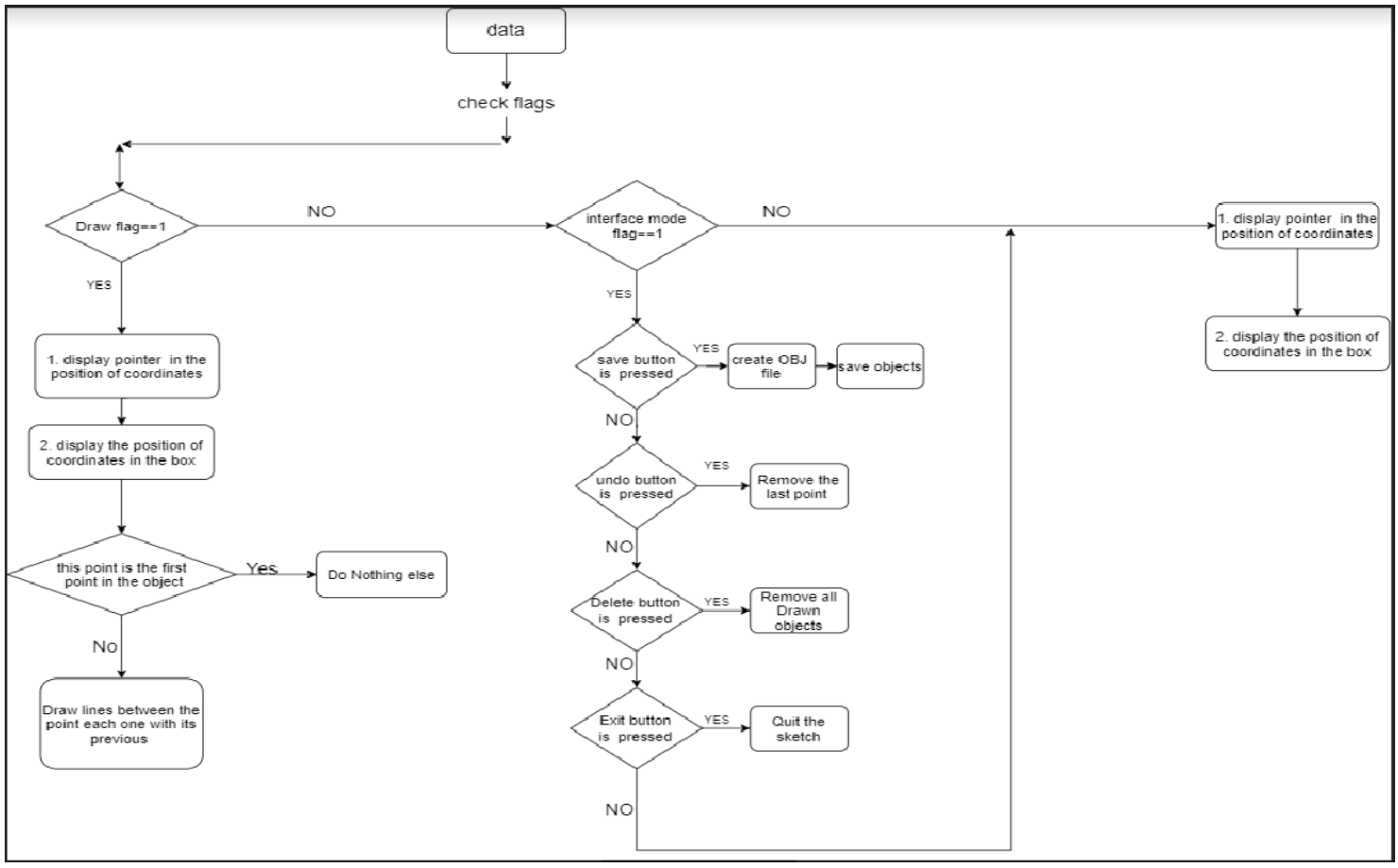


Fig 3.1: Flow Chart of Code Scenario

3.1.2.1 Scenarios of operations

Every line is split and the pen position in space is extracted (i.e x, y, z) and according to the information given we have different Scenarios.

3.1.2.1.1 Drawing mode

- In this situation the draw flag is set while other flags are reset.
- The separate object start point is defined with a set of the draw flag following a reset.
- Connect every point with the previous one by a line.

3.1.2.1.2 Erasing mode

- In this situation the erase flag is set while other flags are reset
- The eraser works by over writing with the background color
- Every point extracted from the pen position is displayed as a black sphere

3.1.2.1.3 GUI mode

- This mode can be accessed using:
    1. Keyboard (asynchronous event):
        - s:- for save.
        - q:- for exiting the program.
        - backspace:- for undo (deleting the last point sent).
        - d:- for clearing the list and the sketch.
    2. Mouse / pen position:
        - For it to be accessed gui flag is set while other flags are reset.
        - Using the mouse / pen position to check it is over the required operation to execute it.

3.1.2.1.3.1 Save function

This operation is done on two steps:

1. Create 2D array where the index indicates the object id and what are the points included in this object.
   - This is done by iterating over the data points list and insert the points coordinates as one element in the array as long the draw flat is set, once it is reset we move to the second element on the 2d array.
2. Create an object file and record the different shapes in it.
   - This is done by iterating over the 2d array and for each element we record the object points in a while loop as a separate.

Part 2: Hardware

3.2 Hardware Design

3.2.1 Intro to Hardware Design

As mentioned before the main idea is to get data from the IMU sensors and use it to draw the positions. But the data could have a lot of errors and mistakes generated from both sensing errors and human errors.

In our design we tried to eliminate the error as much as possible. So, our process for data is that we get the outputs of IMU and combine the data to get us the linear acceleration which can be integrated twice to get the real position. There are many errors and many algorithms used to fix these errors and this part we will examine all the algorithms used in this project so we can get out position near to reality while sending them to our software to be drawn in real time measurement.

So again, our main target is walkthrough from the point of getting data from board to the point of eliminating the errors and sending clean positions to software.

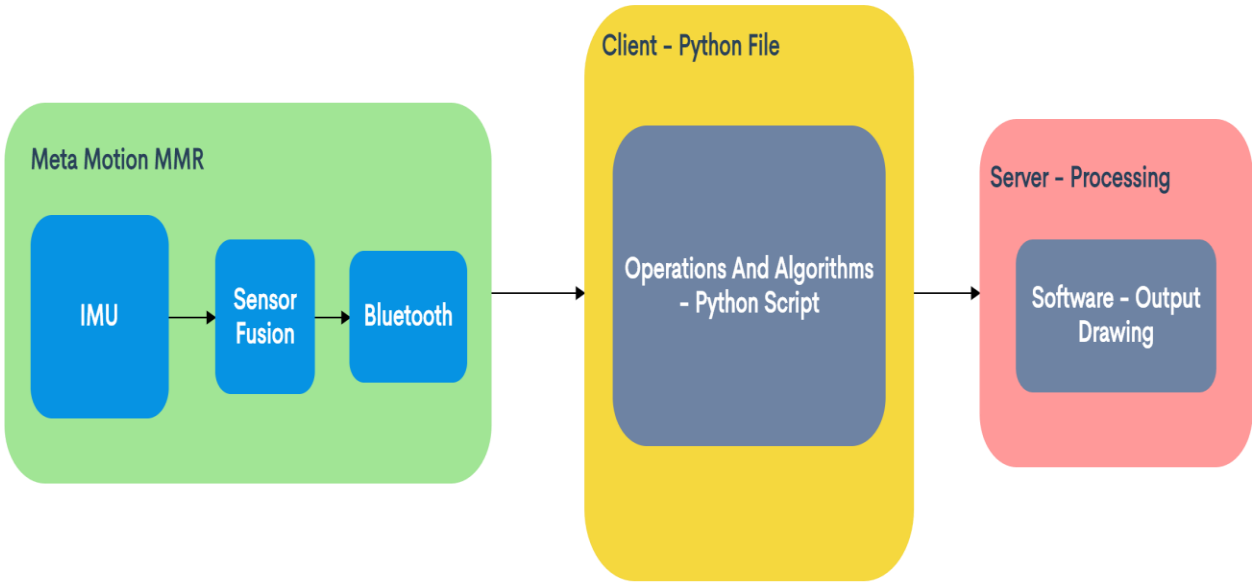


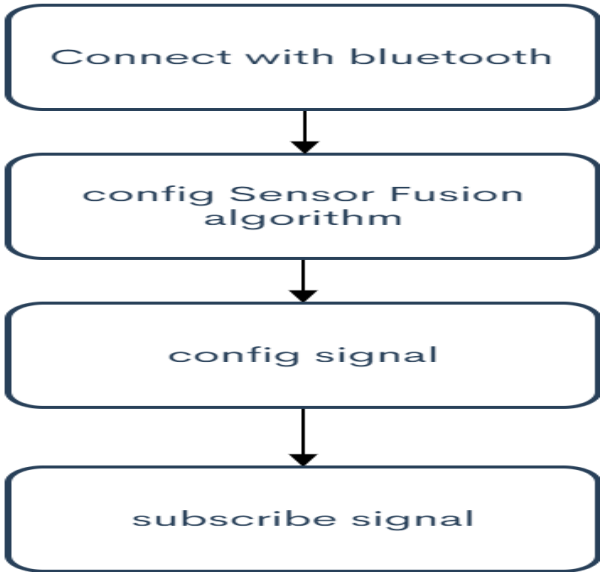
Fig 3.2.1: Complete Design

3.2.2 Design Flow

3.2.2.1 Initialize the board

We are using the board MMR that generate the data and also connect to the board Bluetooth. The IMU sensor inside the board generates three types of reading which are acceleration and orientation of the board and magnetic field change.
But first we need to calibrate each sensor in order to get less error in the reading. While calibration doesn't terminate the error completely but it's a step needed for better accuracy.
MMR board has internal APIs recommended for calibration which we used in our project.

The **problem** with the data is that the acceleration can be affected by the gravity so it accumulates the errors to give different result.
The **solution** here is sensor fusion where we combine the data of the different sensors of IMU to get better readings without errors. The main idea of how this sensor fusion is implemented is by Kalman filter which is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces linear acceleration variable that tend to be more accurate than those based on a single measurement alone.
This is done internally as our board has APIs for sensor fusion algorithm which only needs to be configured.



As figure 3.2.2 shows we connect between our computer and the board via Bluetooth and then we configure our board to sensor fusion algorithm and configure the range of each of the IMU sensor acceleration and gyroscope then we get the data as linear acceleration put it into a signal and subscribe that signal where the data from the board can be read every 0.01 second.

Fig 3.2.2: Initiating the Board with Bluetooth connection

3.2.2.2 Start Streaming Data

Now after we configure the board, we start the streaming and the user can start the movement.
For starting the user has to use the draw button which allows the connection between the software(processing) and the hardware (python code).
The reading comes each 0.01 where we get linear acceleration which needs to be integrated twice for the data.

3.2.2.3 Neglection Window

When a no movement condition is present, minor errors in acceleration could be interpreted as a constant velocity due to the fact that samples not equal to zero are being summed; the ideal case for a no movement condition is all the samples to be zero. That constant velocity indicates a continuous movement condition and therefore an unstable position.
Even with the previous calibration some data can be erroneous, so a window of Neglection between valid data and invalid data for the no movement condition must be implemented.

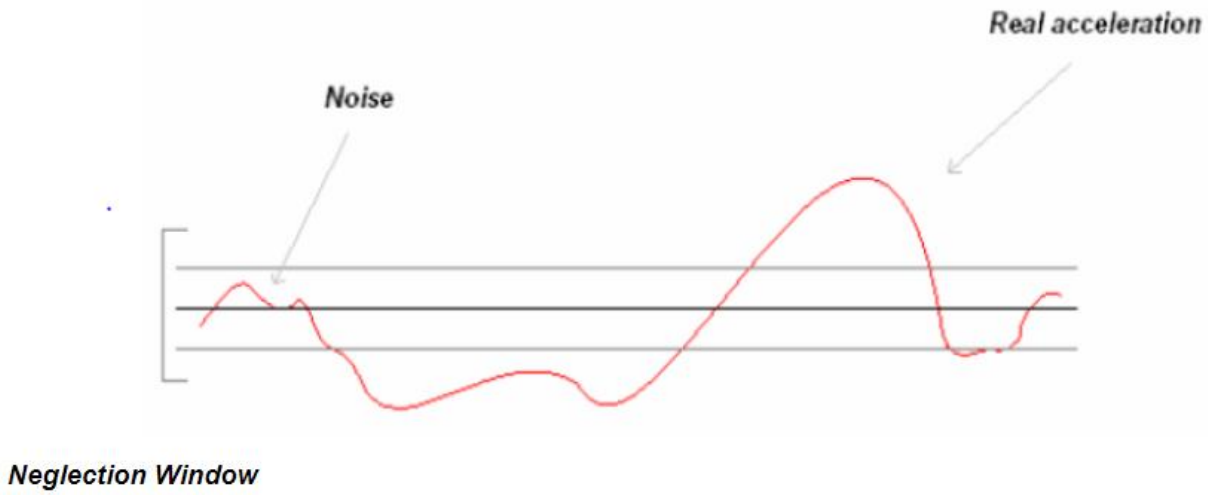


Fig 3.2.3: Reducing mechanical noise by using Neglection window

This Neglection window is implemented by defining a range of readings where the data is known to have an error so it needs to be zeroed.
This is shown clearly on figure 3.2.4

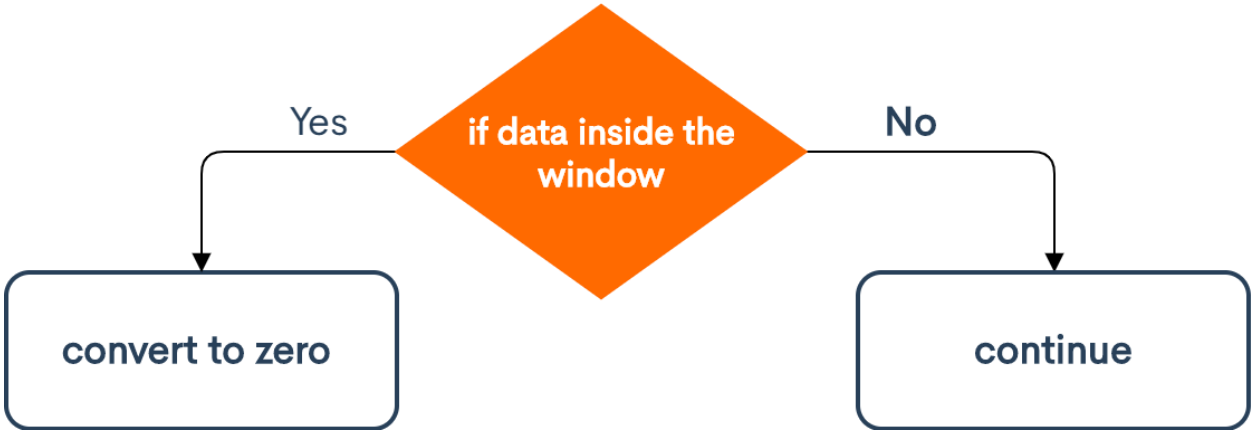


Fig 3.2.4: Neglection Window Implementation

3.2.2.4 Suppressing Human Error

The sensitivity of the board can be very large where any minor deflection in movement can cause a drift in the drawing and because of this random in reading which is not part of the original movement , we need an algorithm to remove this randomness by detecting the true path of the movement and ignore all the deflection happens.

To remove this error we first check which direction the user moved to and suppress all the other directions change.

For example if the user is moving in a straight line in the +ve X direction his hand can't be steadily sharp and all sorts of vibration can happen, so we suppress all the movement in Y and Z directions so no error will be shown.

That improves the accuracy of Engineering drawing with precise movement directions

Following figure 3.2.5 which shows how this is done in our project.

```
List sumX,sumY,sumZ = [0,0,0,0,0,0,0,0,0,0]
Function
Which_to_be_suppressed(velX,velY,VelZ):
        avgX=Calculate_average_velocity(sumX)
        avgY=Calculate_average_velocity(sumY)
        avgZ=Calculate_average_velocity(sumZ)
        if avgX!=0
                If absolute(avgZ/avgX)>limit
                        Suppress X
                If absolute(avgY/avgX)>limit
                        Suppress X
        if avgY!=0
                If absolute(avgX/avgY)>limit
                        Suppress Y
                If absolute(avgZ/avgY)>limit
                        Suppress Y
        if avgZ!=0
                If absolute(avgY/avgZ)>limit
                        Suppress Z
                If absolute(avgX/avgZ)>limit
                        Suppress Z
        sumX=sumX[1:10]
        sumX.append(velX)
        sumY=sumY[1:10]
        sumY.append(velY)
        sumZ=sumZ[1:10]
        sumZ.append(velZ)
```

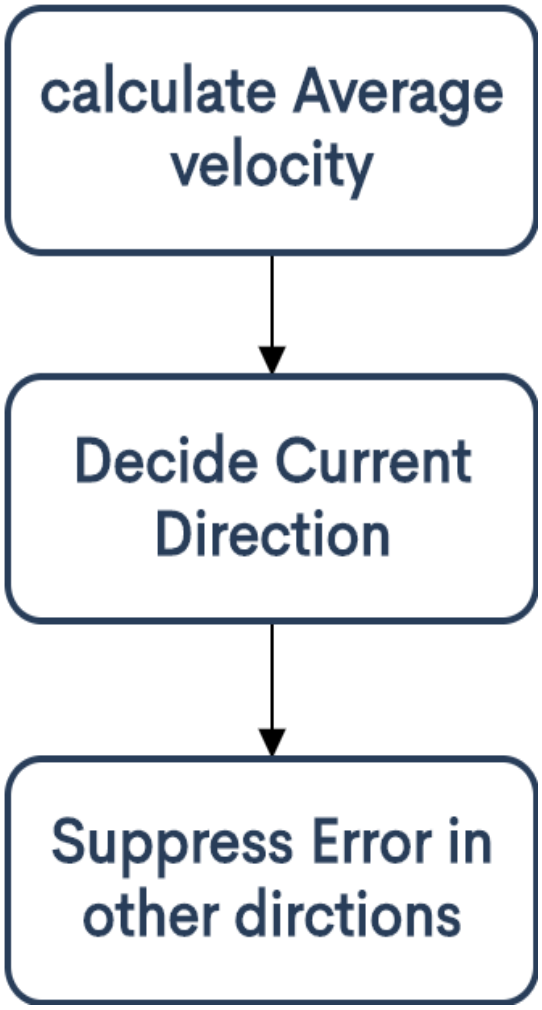


Fig 3.2.5: suppress human error in movement

3.2.2.5 Neglect Sign Change

Some reading might have sign change when a -ve value or +ve value of acceleration occur between two +ve or -ve values respectively. This error can lead in false change in direction while the user still moves in the opposite direction. The cumulative error changes the position where if we want to move 50 cm in reality we find that we only moved for 30cm in the software or maybe worse that the error can completely change the final position. So, we try to minimize this error to get precise position as in real time and space.

This is how the data looks in figure 3.2.6

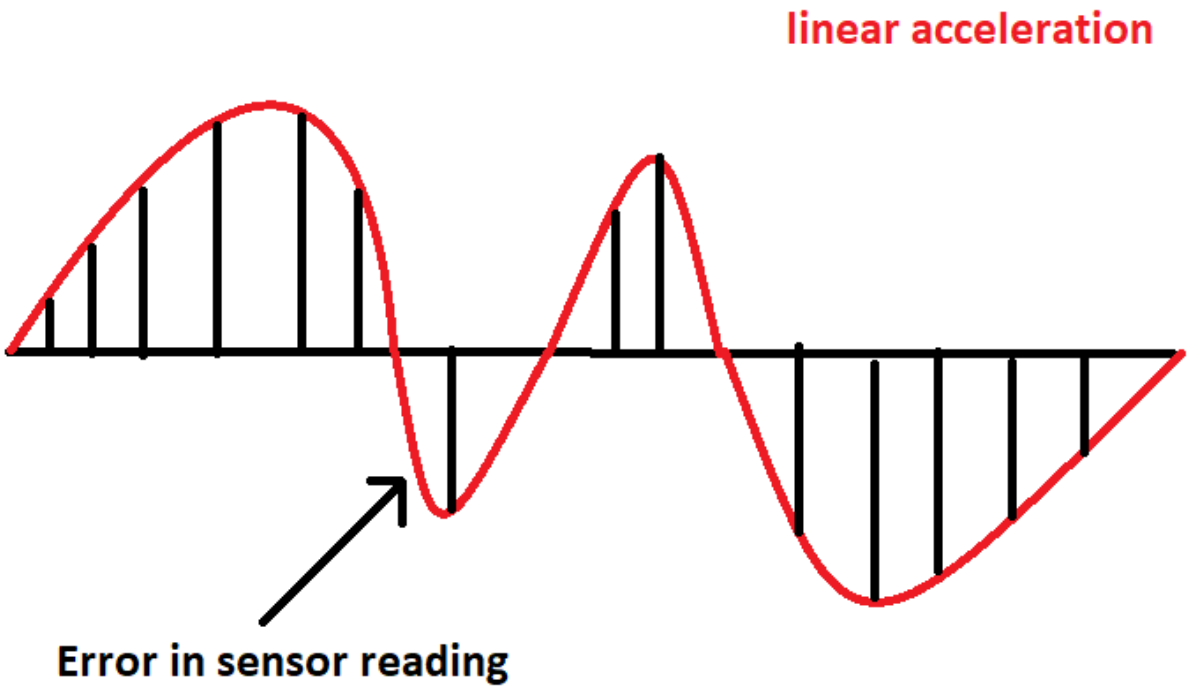


Fig 3.2.6: change in sign Error

The fix we used here is that we ignored the value where sign changes and we check for the next value if the sign like the previous value then probably the questionable value wasn't an error se we put it back in calculations but if the sign was the same as the original direction of reading then we ignore the middle value as shown in figure 3.2.7.

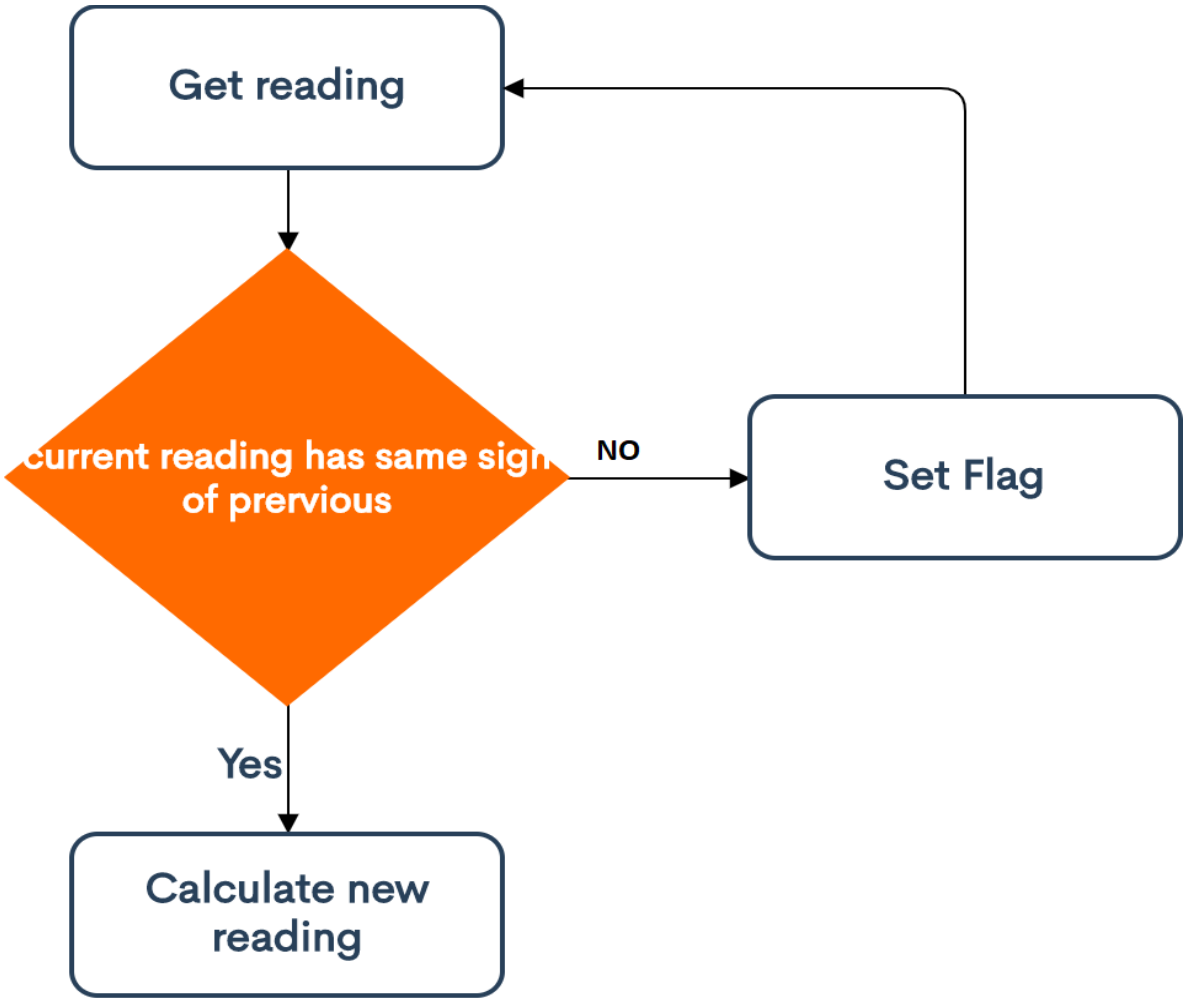


Fig 3.2.7: Neglecting sign change values

3.2.2.6 Calculate Positions

This is the main goal of the design so we need to do integration for each sample to get us the positions.

Our design uses the trapezoidal integration as its the most efficient method to get the position from a continuous linear acceleration data.

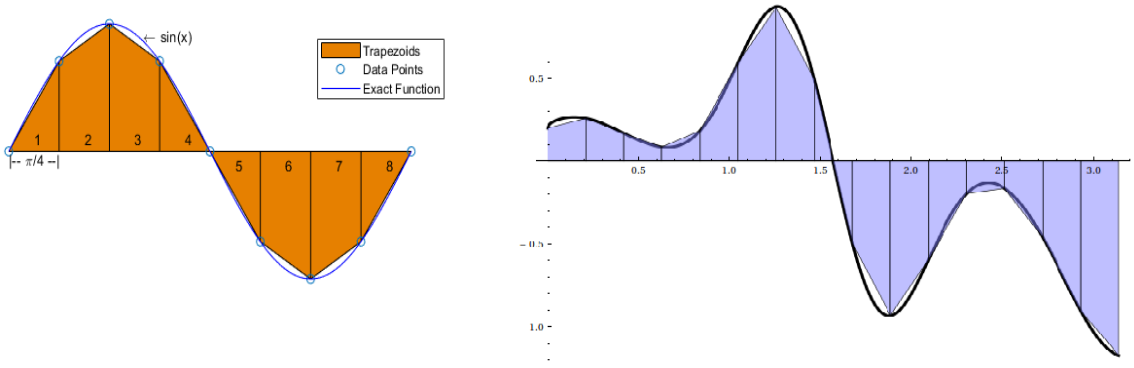


Fig 3.2.8: example of trapezoidal integration

The Rule we used are shown in figure 3.2.9:

$$Vel_n = Vel_o + (Acc_n + Acc_o) * dt/2$$

$$Pos_n = Pos_o + (Vel_n + Vel_o) * dt/2$$

Fig 3.2.9: Equations of numerical trapezoidal integral

3.2.2.7 Check for Transition End

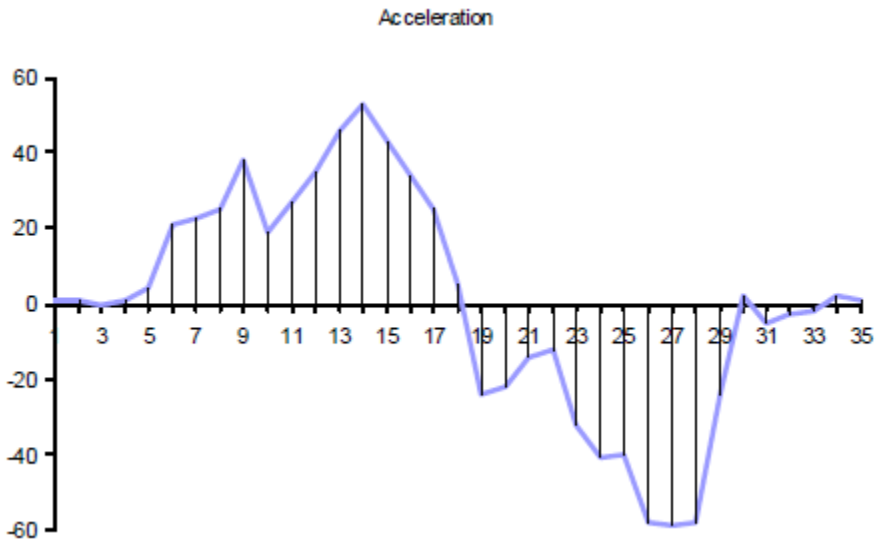


Fig 3.2.10: Typical Accelerometer Output Signal As The Result of Dragging an Object in a Single Axis

Looking at graph in figure 3.2.10 above, there is an initial acceleration or deceleration until a maximum velocity is reached. Then that acceleration flips the opposite way until it reaches zero again. At this point a stable displacement and a new position are reached.

In a real-world scenario where the area below the positive side of the curve is not the same as the area above the negative side, the integration result would never reach a zero velocity and therefore would be a sloped positioning (never stable).

Because of this, it is crucial to force the velocity down to zero. This is achieved by constantly reading the acceleration and comparing it with zero. If this condition exists during a certain number of samples, velocity is simply returned to zero as shown in figure 3.2.11.

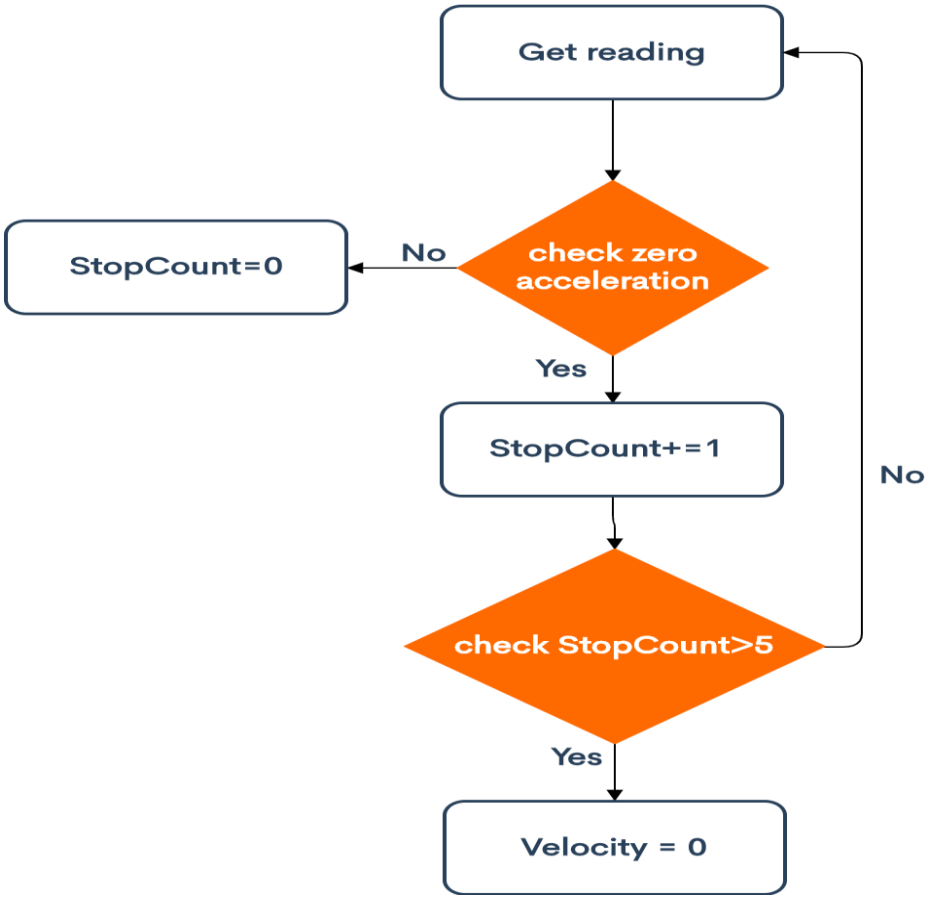


Fig 3.2.11: check for transition end

3.2.2.8 Send Data

This is where the communication occurs after all the error removal algorithms between the python acts like a client sending the final position output to the server (software) to be drawn.

This happens continuously every sample. But when does it end ?

We used a button with drawing such that when we are pressing the button we signal to start the drawing until the drawing ends.

Important notice is that we need to stop for half a second between every transition so that all the acceleration and velocity reaches zero and the algorithm works correctly on the next algorithm.

3.2.2.9 Reset data

If the user needs to start a new drawing, he needs to reset all the data in the board by using the Reset button before the new drawing.

3.2.2.10 Terminate streaming

At the end of the drawing the user must release the button and use a special key ’e’ to exit from the streaming.

Figure 3.2.12 explains all events created in our project.

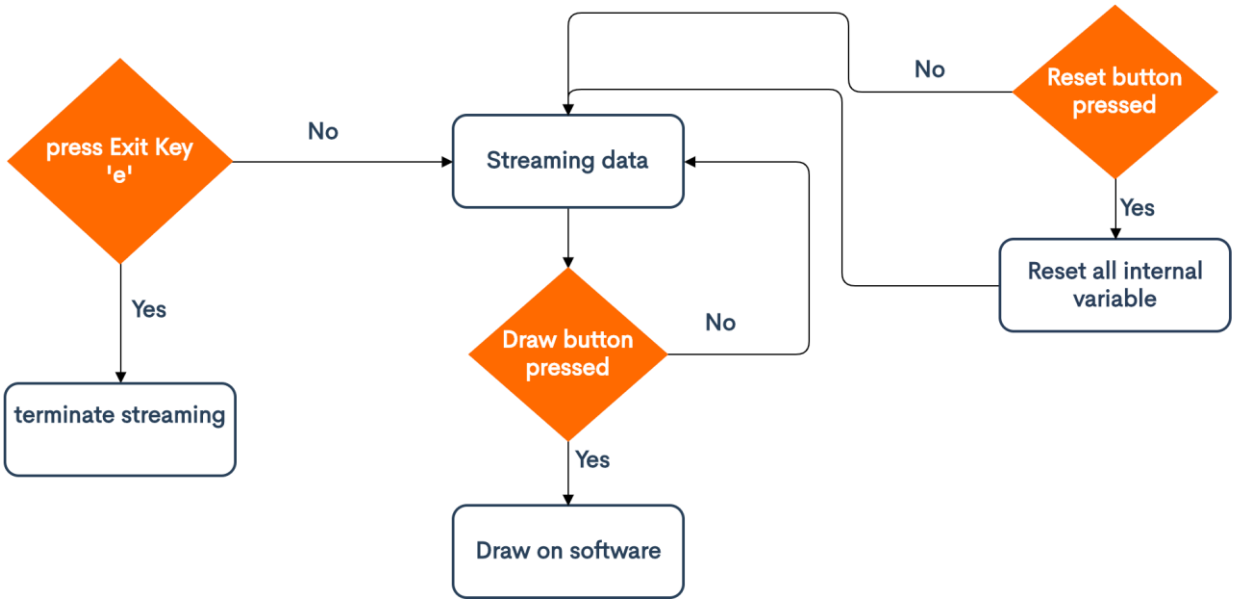


Fig 3.2.12: Hardware Event handling

3.2.2.11 Summary of the chapter hardware design for this project

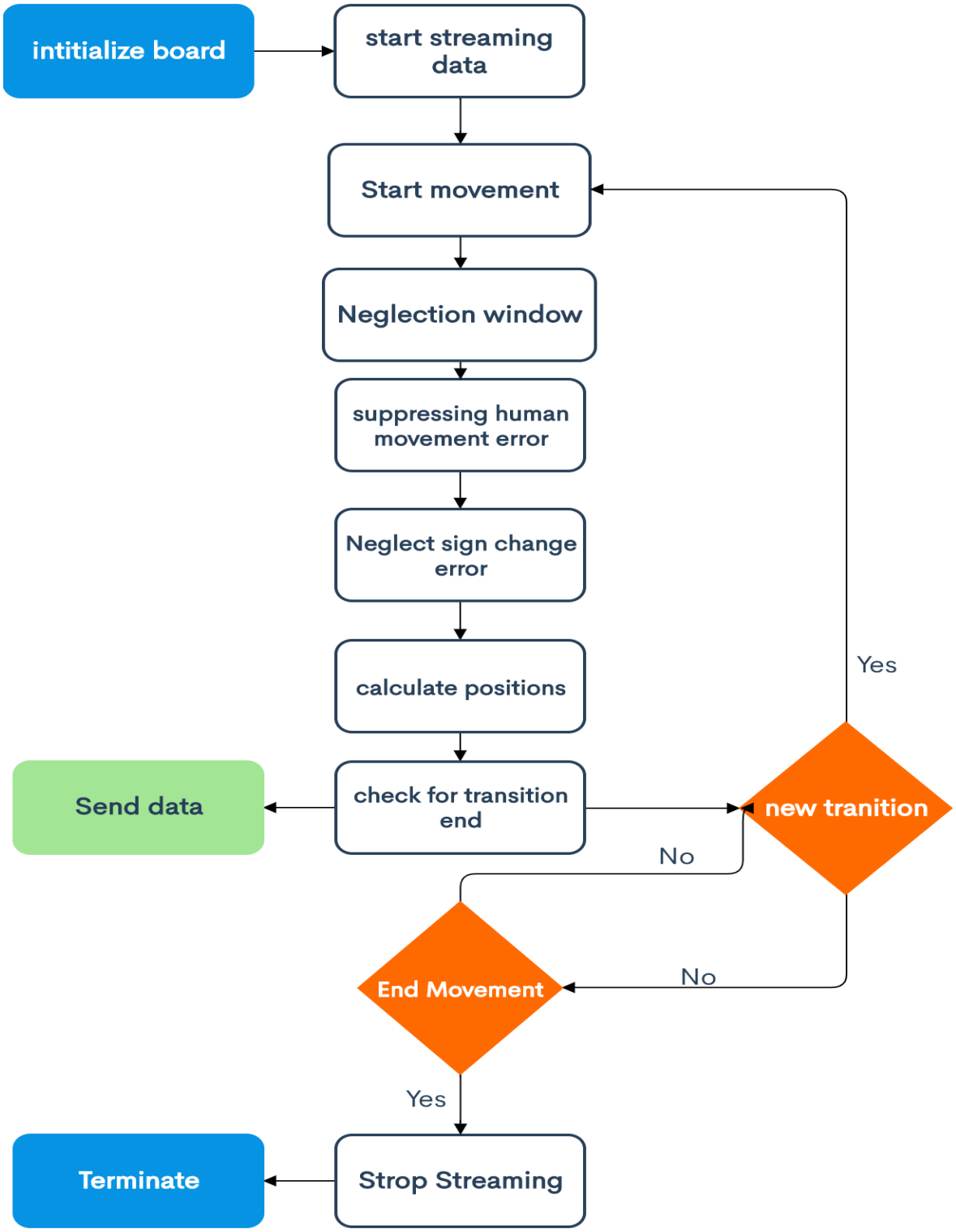


Fig 3.2.13: Summary of hardware design operations and algorithms

Chapter 4: EXPERIMENT

Part 1: Software

4.1 Software Experiments

4.1.1 Software Selection

At the beginning of our project we searched for software tools and we found two software tools python and processing. Each of them has some advantages and disadvantages in using, we used in our project the most suitable one which is processing.

4.1.1.1 Processing

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts.

4.1.1.1.1 Advantages of processing

- It has a well-documented reference, tutorials list and it's a free software.
- It can draw in 2D or 3D coordinates.
- Can be integrated with external microcontrollers.
- Can be used for real time drawing.
- Can be directly in the computer vision Area, with motion detection with using a camera.
- It has a platform for Q&A.
- Coding through language APIs , similar to C language.

4.1.1.1.2 Disadvantage of processing

- It's not widely used yet and not many projects have been made with it so it's hard to find sample codes that will assist our project if needed.
- Might take a while for the learning phase with it.

4.1.1.2 Python

There are Several libraries in python used in drawing and plotting, the most important three are NumPy, Matplotlib and Plotly.

4.1.1.2.1 Libraries

4.1.1.2.1.1 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object.
- sophisticated (broadcasting) functions.
- tools for integrating C/C++ and Fortran code.
- useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

4.1.1.2.1.2 Matplotlib

Matplotlib is a Python 2D and 3D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

4.1.1.2.1.3 Plotly.py

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts Plotly.py is free and open source. Most useful one is Matplotlib library.

4.1.1.2.2 Advantages of python

- Easy to learn and use.
- Each library has its own documentation.
- Many online Tutorials and Open source projects.
- Can be used for 3D Real time tracking.
- Can be integrated with outer microcontrollers.

4.1.1.2.3 Disadvantage of python

- Might be limited through defined functions so harder to add features.
- Hard to manipulate/edit drawing after finishing.

4.1.2 Hardware and software communication

During the project timeline the communication methods have changed according to the following.

4.1.2.1 Text file communication

- The first approach was for the hardware to write data points in a text file in real time while the processing sketch read the data line by line
- Each line is formatted in a way both parties agrees on where information are split by space to indicate the end of line segment
- Line is written according to the following (draw flag erase flag x position y position z position)
- The processing load the lines and begin splitting to extract the information
- Simulation process
  - While testing the software application we added the points manually to a text file and loaded to the processing sketch

4.1.2.2 Serial communication

- The second approach was to send the data directly to the processing sketch with the same format agreed on
- Simulation process
  - While testing the software application we used arduino as the hardware to test the serial communication library provided by the processing IDE.
  - points manually was sent loaded to the processing sketch

- short delay was added between every two points sent to allow for the to be loaded properly

4.1.2.3 OSC library (Open sound control)

- similar to the first approach but using a (processing sketch or python script).
- Simulation process :
  - While testing the software application we communicated two processing projects as the sender and the receiver.
  - The receiver and the sender communicated using the same port number.
  - The sender send the data using the same format line by line to the port while the receiver keeps listening to the port checking if new data was added.
  - When reading the new data it insert it to the points list to by split and run according to the required operation.

4.1.3 Erase function

4.1.3.1 Hardware side

When wishing to erase or delete lines on the sketch, the user holding the pen must do the following:

1. Press the erase button to set the flag.
2. Move the pen in the 3d space to map the pen position to the sketch, pointing over the lines that wished to be erased

4.1.3.2 Software side

After checking if the eraser flag is set, there are two approaches to perform the operation

First approach

- Receive the points wished to erased
- Check their occurrence in the list containing the points to be displayed
- Remove these points from the list
- Pros

- It doesn't affect the rest of the sketch points

- Cons
  - Needs more information to be passed (due to the infinity loop by the draw function, the next iteration will connect the wrong lines together)
    - every iteration we connect each point with the previous one
    - if we wished to delete horizontal line, we need to erase point 3 from the list

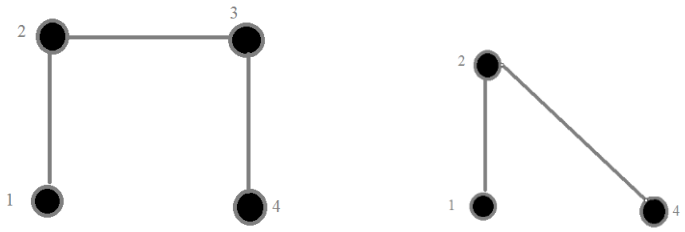


Fig 4.1.1: ERASE

Second approach (Used)

- Receive the points wished to erased
- Over right the points with a sphere (or any 3d shape) having the same background color
- Pros
  - Doesn't affect the original shape
  - When saving the sketch as a 3d object the eraser shape used won't be viewed as we only check the points that has the draw flag set
- Cons
  - It can be seen when rotating the sketch in the 3d space as a shadow over the grid lines
  - When saving the sketch as a 3d object the eraser used can be viewed

4.1.4 Save

The save function is provided to the user to allow him to save the sketch as a 3d design and view it using any application that support 3d models, these application can be later used for view the model, manipulate it or printing the model as 3d design.

4.1.4.1 ToxicLibs library collection

Toxiclibs is an independent, open source library collection for computational design tasks with Java & Processing, which support different packages to maximize the coding flexibility.

Toxi.geom is one of the packages included in the collection which provide (2d/3d vector maths, Mesh container, OBJ and STL exporters, ….etc)

- Pros
  - Able to support massive files, with optional STL colour support.
- Cons
  - The 3d object must be constructed as a mesh.
  - In the simulation process we were unable to display the points as a mesh.

4.1.4.2 ObjectExport (Chosen approach)

As previously discussed, it is a library to export 3d objects or mesh from the processing as a STL or OBJ file, it has two modes of operation.

- Export predefined 3d objects supported by the processing language.
- Export user defined 3d objects (shapes) using begin-end shape functions supported by the processing language.
- Pros
  - Ability to export and 3d (closed) shape in the processing sketch.
- Cons

- Initially it considered every user defined 3d (closed) shape as only one and connected them together.
- Needed more computation to separate the shapes.

An Example of saving separated plates at Figure 4.1.2:A (output of processing sketch)and Figure 4.1.2:B.(output of object file).

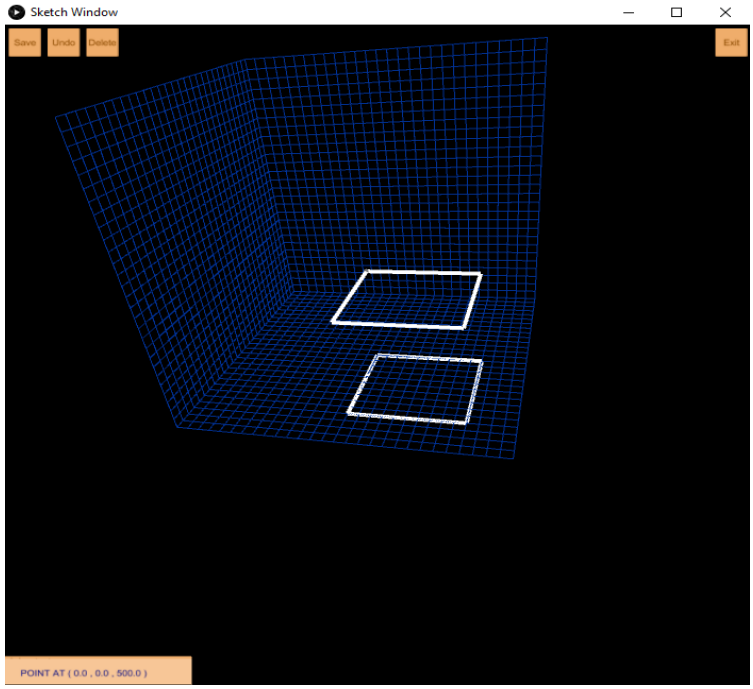


Fig 4.1.2: A.

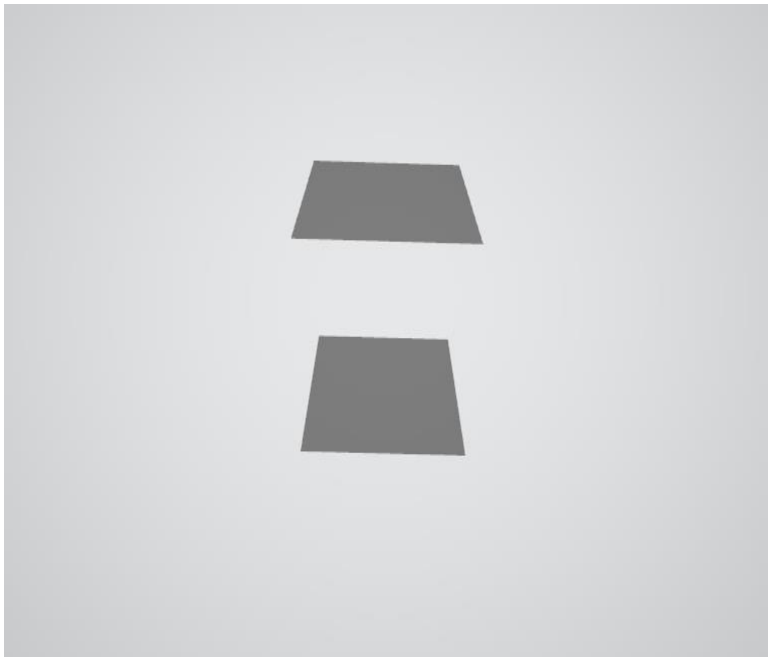


Fig 4.1.2:B

Part 2: Hardware

4.2 Hardware Experiments

4.2.1 Intro to Hardware Experiments

In this chapter we are going to discuss the experiments that have been done to build the whole system to receive acceleration data from the metamotionR board using the bluetooth and then accept it in python script and do operations on it as described on chapter 3 then send data to the software to be drawn .

One of the hardest developments is to develop an application related to hardware due to errors generated from the sensors specially those related to measure the motion as motion of humans is random and doesn't have similar patterns for the errors so its be hard to handle it ,another reason of hardness that application is a real time application so each received sample should be processed and send it to software within the sampling period only so that the next sample shouldn't be delayed which was one of largest problem encountered us.

In the remaining part we will discuss the flow of design in terms of successful and unsuccessful Experiments

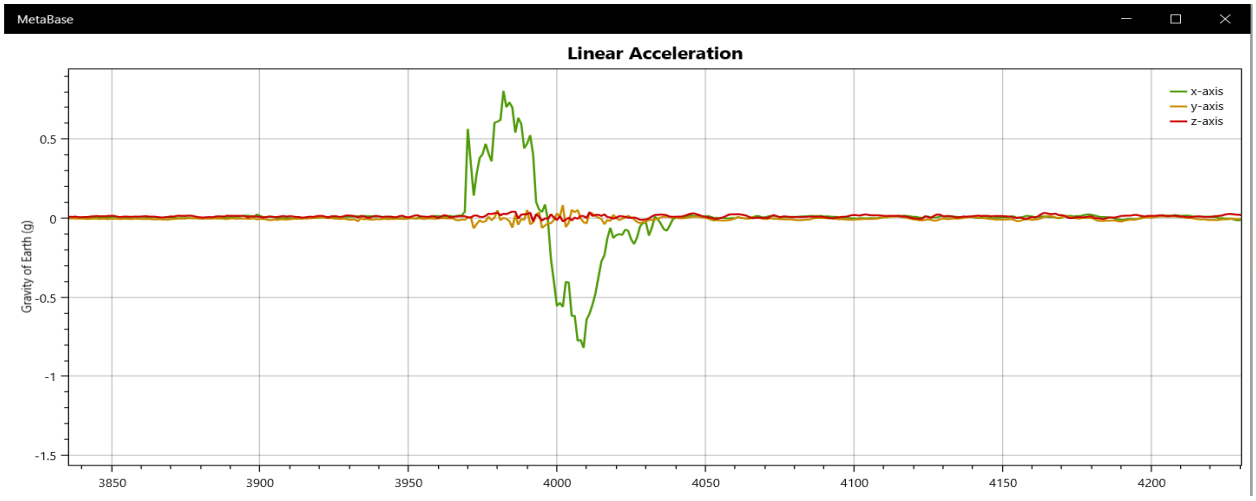


Fig 4.2.1: Graph of Acceleration for moving in positive X-axis of board

4.2.2 Experiments

4.2.2.1 Experiment 1

In the first when board arrived we just run the streaming mode in the board using the “metabase” application which is a software developed by the company which manufactured the board and we saw the graphs of output linear acceleration.

Fig 4.2.1 in previous page show an example for output graph for moving the board in positive X-axis direction.

4.2.2.2 Experiment 2

After testing the board from the ‘Metabase’ app, we tried to program our own code in python to test the board APIs and to print the output linear acceleration data on the console.

First we connected the computer Bluetooth to the board Bluetooth by using the board’s MAC address, then we requested to subscribe our signal output for streaming method and we received the values in the data handler and printed it on the console.

Results were that the output is variant to each single shaking and it has large sensitivity so it might make no sense with large chunks of data.

Also we tested for the time between each sample and we found that it’s not really constant but its close to 0.01 sec. As shown in Figure 4.2.2 although direction of movement on positive X-axis sometimes acceleration magnitude on Y-axis direction be large which makes an error (Human movement error).

Note: values in units of g

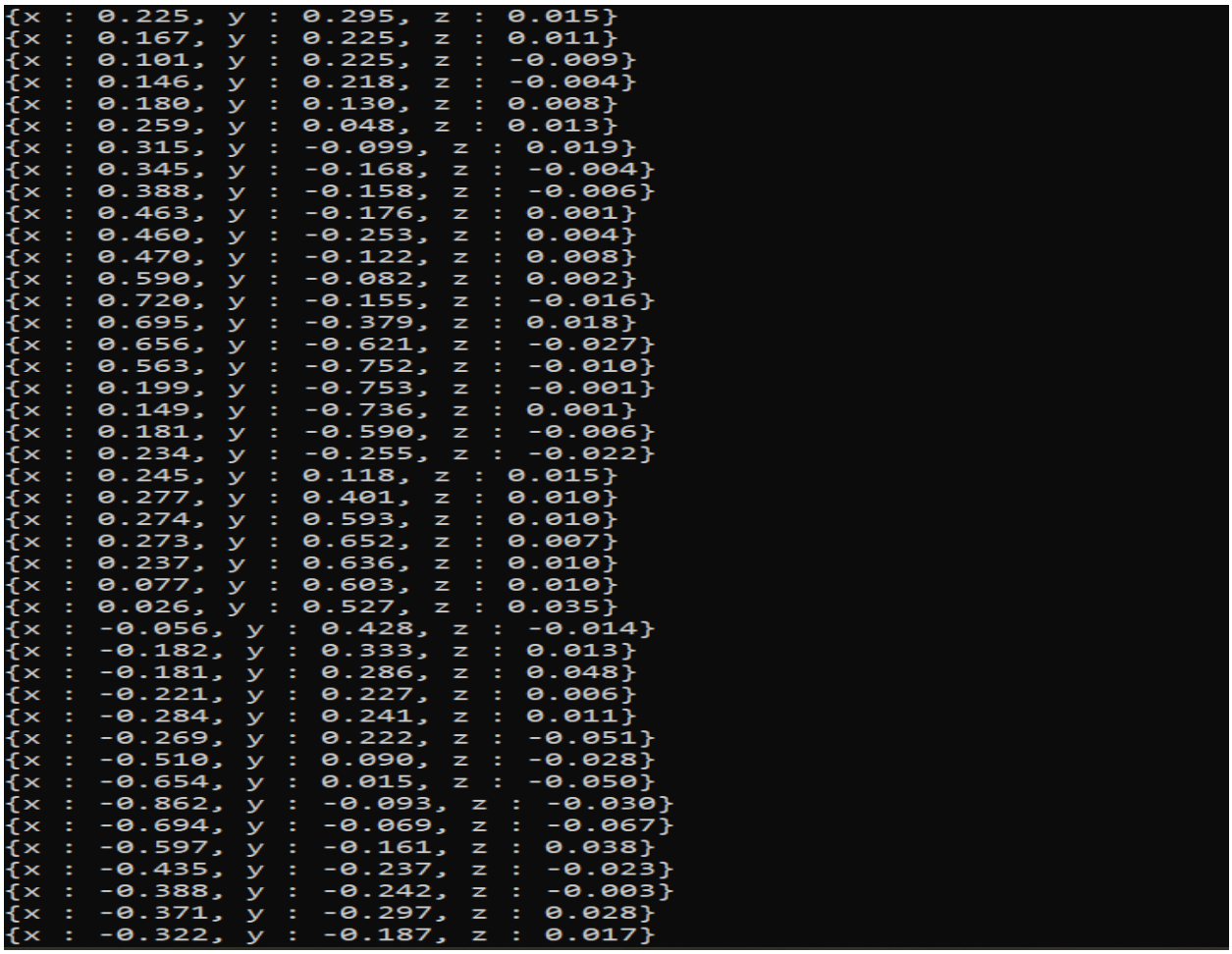


Fig 4.2.2: Acceleration data for moving in positive X-axis of board

4.2.2.3 Experiment 3

We tried to stabilize the movement and tested for the reading again carefully and found that there are some errors that need to be solved.

For example the start and endpoint for motion the acceleration never returns to zero.

4.2.2.4 Experiment 4

This time we tried to send more than one data and subscribe them in one signal.

The purpose behind this experiment is to send both acceleration and button value in one signal.

But later on we didn't use this method as it requires that the two signals must have the sampling rate so we subscribed each data on its own signal and made a callback function for everyone.

4.2.2.5 Experiment 5

In this experiment we started to calculate the position using ordinary rectangular numerical integration method to calculate the velocity then integrate again to calculate the position and we found it's better to compare the calculated positions to real position and find out how accurate our reading is.

We found that the calculated positions deviated from the real position greatly. This confirmed the wrong raw reading came from the board which needs further algorithms to remove the errors.

4.2.2.6 Experiment 6

In this experiment we changed the integration algorithm and used the trapezoidal rule instead of ordinary rectangular .

We noticed that calculation became much better than experiment 5 but still quite away from real position. So it's need further algorithm to handle the errors

4.2.2.7 Experiment 7

We noticed that while the board is in a stationary position in the human hand, there's a small acceleration value which accumulates a drift in the calculated position over the time so we put the neglection window for each acceleration component which suppresses it.

4.2.2.8 Experiment 8

When we made a transition with the board in real and ended it we noticed that position component in direction was the transition on it still changes this is because the sensor isn't ideal so the area under the curve for acceleration won't be zero so the velocity at the end of transition won't be zero which is an error so we developed the transition end check which suppress the velocity when the transition ends.

4.2.2.9 Experiment 9

We tried to make a filter on data to smooth it by simply averaging on it but it made a trade off with the neglection window and evaluated some non zero samples to zero so we removed it.

4.2.2.10 Experiment 10

After examining the data received we noticed that there was a sample came with different sign with its surrounding samples so we developed the neglect sign change algorithm which neglects the sample with the different sign.

4.2.2.11 Experiment 11

In this experiment we added the draw button by using Apis for gpio pins of the board and configure it while pressing we draw the data sent to our software that done by making internal resistance in PULL UP mode and with the FALLING edge we set the IsPressed flag and reset it with the RISING edge.

4.2.2.12 Experiment 12

In this experiment we added the reset button by using Apis for gpio pins on the board and configure it active LOW on pressing and internal resistance in PULL UP mode and with the FALLING edge we reset all variables.

4.2.2.13 Experiment 13

In this experiment we developed the suppressing human error algorithm which check if the human move the board in a direction of the 3 direction ten multiples of

another direction, we suggest that human want to move only on that direction so we suppress the acceleration component with low magnitude will be suppressed We noticed that outputs didn't meet out expected output as we find that using acceleration to decide was wrong.

4.2.2.14 Experiment 14

In this experiment we developed the suppressing human error algorithm but we used the velocity to decide the ten multiple and which to be suppressed instead of using the acceleration.
Finally it works fine.

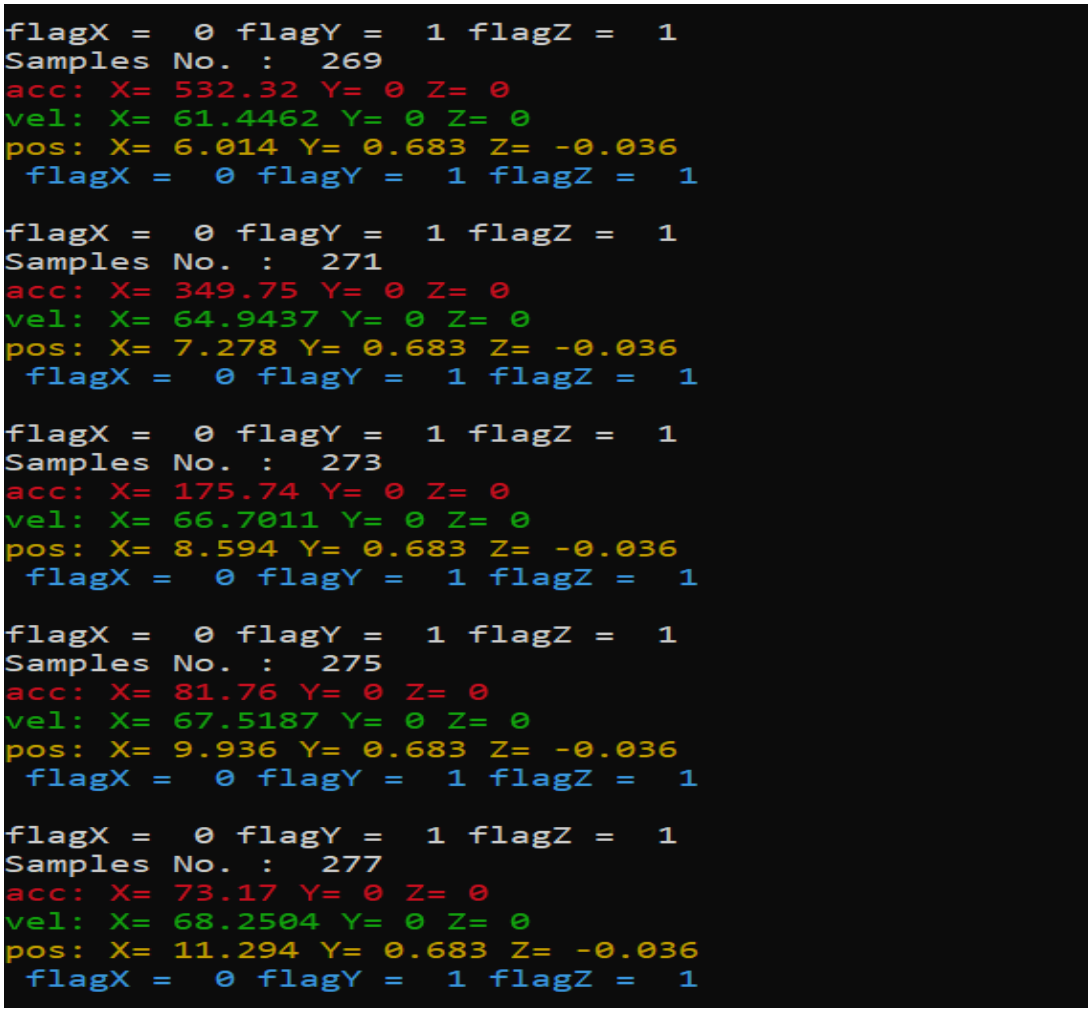


Fig 4.2.3: Example for final output of moving on positive X-axis

Chapter 5:

Limitations, Conclusions and Future work

5.1 Limitations

We have some limitations on our project like limitations on moving the board and orientation of it and other limitations that we discuss in detail in the next part of chapter.

1- Limitation on the orientation
The board should be in the same orientation at the beginning of the drawing and hold it in this orientation while whole of the drawing.
It should be in horizontal orientation and any deflection away from this orientation it will cause an error in calculations.

2- Limitation on the moving of the board
The board should be moved with quite high acceleration, not very low as the neglection window will neglect it and not very high so that the board can't capture many samples for this high acceleration in a short time which causes error in calculations.

Another limitation on the moving of the board is that we must stop between each transition around second and start the next transition as mentioned on the design chapter.
NOTE:the algorithm optimized for vertical and horizontal lines only.

3- Limitation on the drawn shapes
In our project we can draw the shapes that consist of vertical and horizontal lines, it's not preferred to draw circular shapes as it would be shown distorted.

5.2 Conclusions

Finally, our project has a relatively high accuracy to for engineering drawing on the shapes the consist of vertical and horizontal lines with very good straight lines but sloped lines won't be straight and have some tortuous segments, so it can be very useful for the freshmen to practice to draw simple isometrics with studying Engineering drawing course like the isometric shown in fig 5.1.

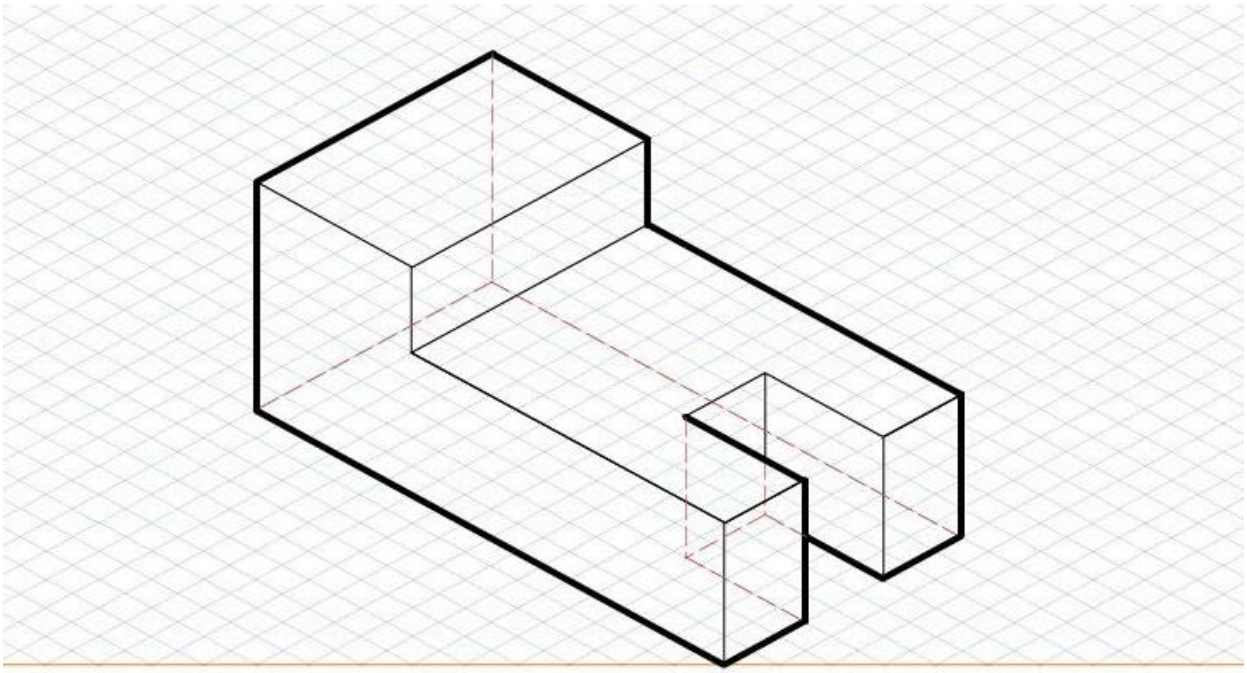


Fig 5.1: Simple Isometric consist of horizontal and vertical lines

5.3 Future Work

5.3.1 Hardware

1- Make the board and buttons on a pcb and make a casing like a pen using 3d printing and put the pcb inside it which is the final product of the project.
2- Make the board move in any orientation not only the same orientation at the beginning.
3- optimize that board can move in a sloped direction not only on vertical and horizontal only.
4- make gloves that can be worn on the other hand and control the software remotely like rotating the drawn shape, zoom in/out etc.

5.3.2 Software

5.3.2.1 SYSTEM ARCHITECTURE

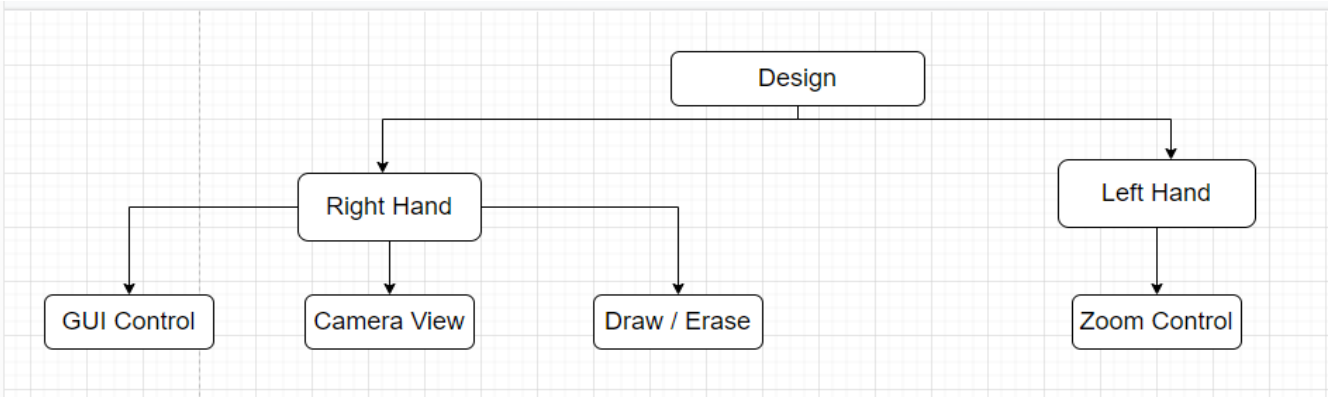


Figure 5.1.1: Design of Future Work

- Right Hand :
    1) GUI control
        - The gui consist of (save – undo – delete – exit)
        - This mode is entered by setting a gui_flag.
        - The GUI buttons are selected by moving the pen over the desired button.
    2) Camera view
        - This mode is entered by setting a camera_flag.

- This mode is controlled by sending the pen vector view to the software to change the camera parameters.

3) Draw / erase
   - Using the current algorithm but the erase functionality needs more enhancement.

- Left Hand :

  - The scale is controlled using a flex sensor.
  - This is for controlling the zooming scale of the camera.

References and Urls

- https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/
- https://en.wikipedia.org/wiki/Inertial_measurement_unit
- https://learn.sparkfun.com/tutorials/accelerometer-basics/all
- https://en.wikipedia.org/wiki/Gyroscope
- https://en.wikipedia.org/wiki/Magnetometer
- https://mbientlab.com/documentation
- https://processing.org/
- https://n-e-r-v-o-u-s.com/tools/obj/
- http://www.lagers.org.uk/g4p/
- http://www.sojamo.de/libraries/oscP5/
- Implementing positioning algorithms using accelerometers using kurt seifert and Oscar Camacho paper
- An Accurate Indoor Positioning Algorithm Using Sensors and Crowdsourced Landmarks Beakcheol Jang , Hyunjung Kim and Jong Wook Kim
- Motion tracking in field sports using GPS and IMU by M. Roobeek
- P. zhan Chen, J. Li, M. Luo, and N. hua Zhu, “Real-Time HumanMotion Capture Driven by a Wireless Sensor Network„” International Journal of Computer Games Technology, vol. vol. 2015, no. Article ID 695874, 2015