GESTURE CONTROLLED MOUSE

A Minor Project work submitted in partial fulfillment of the requirement for the award of the degree of

BACHELO*R* OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

by

M SUNNY BABU	**22215A0425**
S ANKITHA	**21211A04N0**
B THARUN	**21211A0421**

Under the esteemed guidance of
Dr. JAYSHREE DAS
M.Tech, Ph.D.
Associate professor ECE Department

Mr. Pandurang Mirajkar
Assistant professor ECE Department

B.V.Raju Institute of Technology

UGC- AUTONOMOUS
Department of Electronics and Communication Engineering
Vishnupur, Narsapur, Medak.(Dt)
(Affiliated to JNTU, Hyderabad)

2022-2023

B.V.Raju Institute of Technology

Vishnupur, Narsapur, Medak.(Dt) Pin:502313
(Affiliated to JNTU, Hyderabad)
Ph: 08458-222000, 222001 Fax: 08458-222002

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the Minor Project work entitled **GESTURE CONTROLLED MOUSE** is being submitted by Mr. **B THARUN, S ANKITHA, M SUNNY BABU** in partial fulfillment of the requirement for the awardof the degree of **B.Tech. in Electronics & Communication Engineering**, by Jawaharlal Nehru Technological University Hyderabad is a record of bonafide work carried out by them under my guidance and supervision from **2022** to 2023. The results presented in this project have been verified and are found to be satisfactory.

INTERNAL GUIDE
Dr.Jayshree Das, M.Tech, Ph.D
Associate Professor

HEAD OF THE DEPARTMENT
Dr.Sanjeev Reddy
B.E, M.Tech,PhD.
Professor & HOD, Dept. of ECE

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

We take opportunity to express our indebt gratitude to the persons who contributed for our work,for being our inspiration and guide which led to the successful completion of the project.

We are grateful towards our College Management and our beloved Principal Dr. SANJAY DUBEY, M.Tech, Ph.D for providing us the necessary infrastructure and facilities that ensured smooth and satisfactory execution of the project.

We would like to express out profound gratitude to our Head of the department Dr. SANJEEV REDDY, M.Tech, PhD, Professor & HOD, Dept. of ECE, for his encouragement inspiration and close monitoring and guidance he gave us during the execution of the project.

We express our sincere thanks to our guide Dr. Jayshree Das, M.Tech, Ph.D Associate Professorand Mr. Pandurang Mirajkar, Assistant Professor, Dept. of ECE for her/his valuable suggestion and motivation in successful completion of project.

We also wish to express our thanks to all the faculty members and laboratory staff members whowere helpful directly and indirectly for the completion of the project.

B THARUN (21211A0421)
S ANKITHA (21211A04N0)
M SUNNY BABU (22215A0425)

ABSTRACT

This work presents the design and implementation of an innovative gesture-controlled mouse system tailored to address the unique needs of individuals with hand impairments. The traditional interaction between humans and computers heavily relies on manual dexterity, which poses significant challenges for those with limited hand mobility. To bridge this gap, a novel approach leveraging gesture technology is proposed. In this work, a gesture-based device is created that mimics mouse actions and control a windows PC. This device can be used by persons with hand impairments who cannot use a conventional mouse to control the PC. The wearable gesture could communicate with the PC which can in turn communicate with apps and avail many services such as a communication software or accessing the internet for healthcare, food delivery etc. A Dexter is used as HID and connect it to the 3 axis Gyroscope sensor for thereadings of different axis values and moving the mouse accordingly to these axis and hand orientations. Also, a touch-sensors used which are responsible for the mouseclick actions i.e., left click and right click.

DECLARATION

We here by declared that the project work entitled " **GESTURE CONTROLLED MOUSE**" submitted to the JNTU Hyderabad and Department of Electronics and Communication Engineering, BVRIT Narsapur, is a record of an original work done by Mr. B THARUN bearing roll numbers 21211A0421 during the period 2022 to 2023 under the guidance of Dr.Jayshree Das, Associate Professor, Dept. of ECE and Mr. Pandurang Mirajkar Assistant Professor, Dept. of ECE and this project work is submitted in the partial fulfilment of the requirements for the award of the degree "Bachelor of Technology" in "Electronics & Communication Engineering".

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree.

1) Tharun (21211A0421)

2) Sunny Babu(22215A0425)

3) Ankitha (21211A04N0)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In the era of evolving technology, human-computer interaction has made significant progress. Traditional input devices, such as computer keyboards and mice, have been the primary means of interacting with personal computers for decades. But as tech enthusiasts, researchers and innovators seek more intuitive and immersive ways to interact with computers, gesture-controlled interfaces are emerging as a promising frontier.

The Gesture-Controlled Mouse for PC project represents a compelling exploration into the world of gesture-based interactions that aims to redefine the way we navigate and interact with digital content on PCs. This project uses the power of computer vision and motion sensor technology to create a seamless and intuitive interface that allows users to control their computers through natural hand and body movements.

This report details the conceptualization, design, implementation, and evaluation of a gesture-controlled mouse project for computers. We examine the motivation behind this project, the technology stack employed, the various components that make up the system, the challenges faced during development, and the potential applications and impact of this innovative interactive paradigm.

By the end of this report, readers will have a comprehensive understanding of the motion-controlled mouse project for personal computers and its importance in human-computer interaction. Additionally, this project has the potential to change the way we interact with computers, making interactions more intuitive, inclusive, and accessible to users of all backgrounds and abilities.

1.2 MOTIVATION

In the realm of human-computer interaction, the conventional means of using a computer mouse relies heavily on fine motor skills and manual dexterity, making it challenging for individuals with hand impairments to access digital interfaces seamlessly. To address this issue and promote inclusivity, a groundbreaking project aimed at developing a Gesture- Controlled Mouse for PC, specifically tailored for individuals facing hand impairments. This work harnesses the potential of the Dexter Board, a versatile microcontroller platform, alongwith a 3-axis gyroscope and touch sensors, to enable intuitive and hands-free control of the computer cursor.

1.3 OBJECTIVE

The objective of this project is

i) To provide an alternative interaction method that enhances accessibility and usability for users with limited hand mobility. By leveraging gesture recognition technology, the system will translate hand gestures and movements into corresponding cursor actions on the computer screen, eliminating the need for traditional manual manipulation of a physical mouse.

1.4 ADVANTAGES

1) **Intuitive Interaction:** Gesture control provides a more natural and intuitive way to interact with computers, eliminating the need for physical input devices like a mouse or keyboard. Users can simply use hand and body movements to navigate and manipulate digital content.

2) **Enhanced Accessibility:** This project has the potential to significantly enhance computer accessibility for individuals with physical disabilities. People with mobility challenges can benefit from gesture control as it doesn't require fine motor skills or physical contact with input devices.

3) **Reduced Physical Strain:** Traditional input devices like a mouse and keyboard can lead to repetitive strain injuries over time. Gesture control reduces the physical strain on users, potentially lowering the risk of musculoskeletal issues.

4) **Customizability:** Gesture control interfaces can be highly customizable, allowing users to define their own gestures or mappings for specific applications, making it adaptable to individual preferences and needs.

5) **Fun and Engaging:** Interacting with a computer using gestures can be a fun and engaging experience, making computing more enjoyable for users of all ages.

6) **Research and Development:** Gesture-controlled interfaces represent a cutting-edge area of research and development, offering opportunities for innovation and advancing the field of human-computer interaction.

1.5 CONCLUSION

This Chapter explained the introduction about the project Gesture Controlled Mouse for PC

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

This chapter refers literature of last decade showing the research work carried out in the gesture control. The summary of the various conference and journal article are given.

2.2 LITERATURE SURVEY

The authors proposed the Vision-based dynamic gesture recognition which is used to control the mouse of a PC. After obtaining the imagethrough the camera, the original image is binarized, eroded, smoothed and filled with holes, then the circumscribed rectangles and the centroid points of the gesture image are tracked, then the movement direction of the gesture is determined according to themotion track. Finally, through the gesture controlmovement of the mouse pointer to move and simulatethe left-click operation[1].

The authors proposed a method to control the movement of a mouse pointer using simple hand gestures and a webcam is proposed. A real-time tracking algorithm is implemented based on adaptive skin detection and motion analysis. Using the history of motion, the trajectory of the movement of the hand is drawn and then used to identify a gesture. A region of interest algorithm is proposed, in order to scale the motion when the user is located far away from the point of capture. The motion of the mouse pointer is scaled accordingly[2].

The authors proposed that Gesture-controlled laptops and computers have recently gained a lot of traction. Leap motion is the name for this technique. Waving our hand in front of our computer/laptop allows us to manage certain of its functionalities. Over slides and overheads, computer-based presentations have significant advantages. Audio, video, and even interactive programs can be used to improve presentations. Unfortunately, employing these techniques is more complicated than using slides or overheads. The speaker must operate various devices with unfamiliar controls (e.g., keyboard, mouse, VCR remote control). In the dark, these devices are difficult to see, and manipulating them causes the presentation to be disrupted. Hand gestures are the most natural and effortless manner of communicating. The camera's output will be displayed on the monitor. The concept is to use a simple camera instead of a classic or standard mouse to control mouse cursor functions. The Virtual Mouse provides an infrastructure between the user and the system using only a camera. It allows users to interface with machines without the use of mechanical or physical devices, and even control mouse functionalities. This study presents a method for controlling the cursor's position without the need of any electronic equipment. While actions such as clicking and dragging things will be carried out using various hand gestures. As an input device, the suggested system will just require a webcam. The suggested system will require the use of OpenCV and Python as well as other tools. The camera's output will be presented on the system's screen so that the user can further calibrate it[3].

The authors proposed that the basic goal of Human Computer Interaction System is to improve the interaction between users and computers by making the computer more receptive to user needs. Human Computer Interaction with a personal computer is not just limited to keyboard and mouse interaction today. Interaction between humans comes from different sensory modes like gesture, speech, facial and body expressions. The paper presents a literature survey conducted which provides an insight into the different methods that can be adopted and implemented to achieve hand

gesture recognition. It also helps in understanding the advantages and disadvantages associated with the various techniques. Further, in this paper, we have proposed a cost-effective gesture recognition system that translates the detected gestures into actions. The system has two interfaces, the distance sensor interface and the accelerometer interface. The distance sensor interface translates the gestures for actions such as shuffling between different applications, volume control, scrolling, keyboard short-cuts and so on; while the accelerometer interface realizes the typing part of the system[4].

The authors proposed that computers have become an essential part of our lives. Many advances have occurred in these machines over past decade in terms of new faster processors, energy efficiency, HD monitors etc. but the basic input devices such as mouse and keyboard remained the same. This project aims at giving a better mode of human computer interaction by allowing computers to respond to human hand gestures directly. The device aims at using the same interface that a mouse uses for interaction with a computer and thus it can be used in any computer without much modification to its software environment[5].

The authors proposed that hand Gesture Recognition plays a key role in human-computer interactions. As we can see that there are so many new Technological advancements happening such as biometric authentication which we can see frequently in our smart phones, similarly hand gesture recognition is a modern way of human-computer interaction i.e., we can control our system by showing our hands in front of webcam and hand gesture recognition can be useful for all kinds of people. Based upon this idea this paper is presented. This paper provides a detailed explanation to the algorithms and methodologies for the color detection and virtual mouse[6].

2.3 CONCLUSION

This chapter explained the work carried in the Gesture Controlled Mouse for PC using various methods.

CHAPTER 3
ANALYSIS AND DESIGN

3.1 INTRODUCTION

This chapter explains about the components, design and analysis of Gesture Controlled Mouse for PC.

3.2 REQUIRED COMPONENTS

3.2.1 DEXTER BOARD

The Dexter Board is designed using the microcontroller STM32L452RCT6 (LQFP64 package) from STMicroelectronics. It is a versatile microcontroller, which can be used in a variety of ways. For the early Build Club experiments, designed for Engineering and Science students, irrespective of disciplines, Dexter 'Wrapper' functions have been predefined. These wrapper functions use the STM32 CUBE IDE (Integrated Development Environment). The IDE comes with certain default settings. Although it is possible to change these default settings, for the initial Build Club experiments we recommend that only default settings be used.

Fig 1.1.1- Dexter board

The Wrapper functions are as follows:

1) **GPIO - General Purpose Input / Output** To program GPIO pins. Can be used either for Digital signals or Analog signals. They can be thus used for driving LEDs, Motors or receive from sensor

Table 1.1.1- Functional call and description dexter

Type	Function call	Description
Digital IO	Digital_transmit (Pin, State);	To turn ON/OFF Digital Output pins
	Digital_transmit_pwm (Pin, Duty cycle);	To transmit to an output using PWM
	Digital_receive (Pin);	To receive Digital Input pins
Analog IO	Analog_transmit (Pin, State);	To turn ON/OFF Analog Output pins
	Analog_receive (Pin);	To receive Analog Input pins

OUTLINE OF THE DEXTER BOARD

Fig 1.1.2 – Front view of dexter

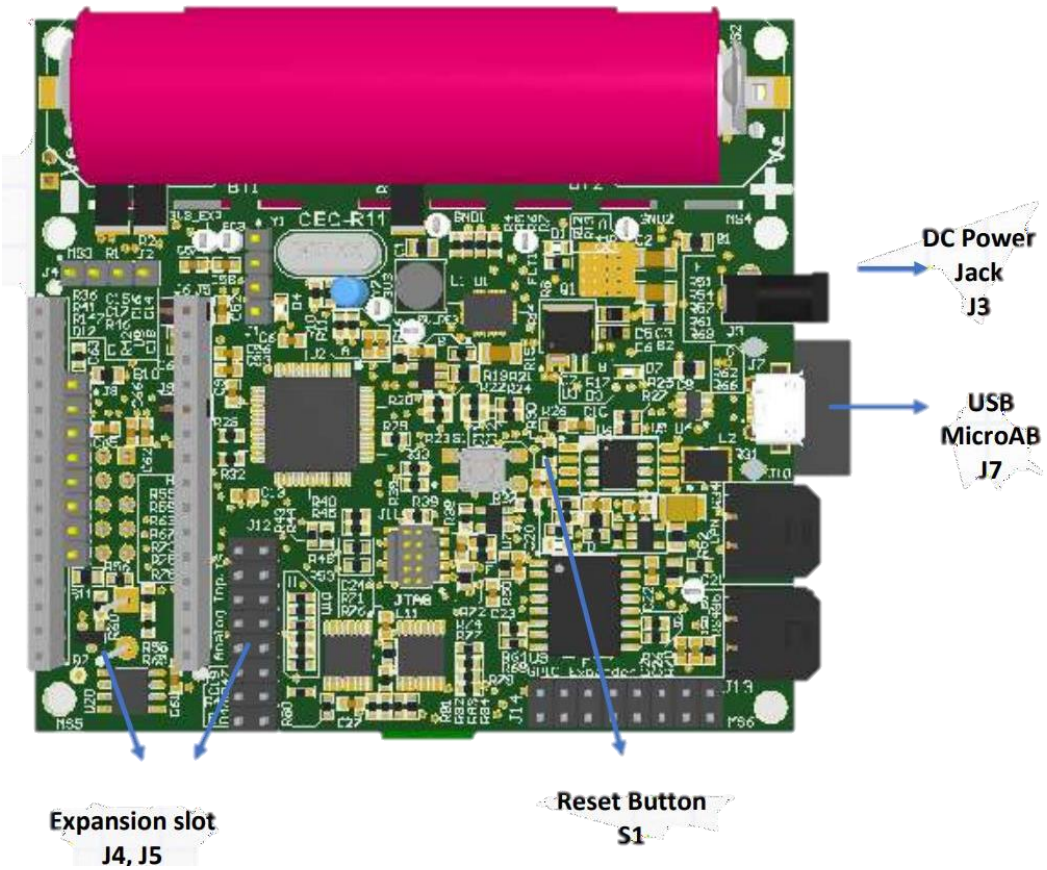
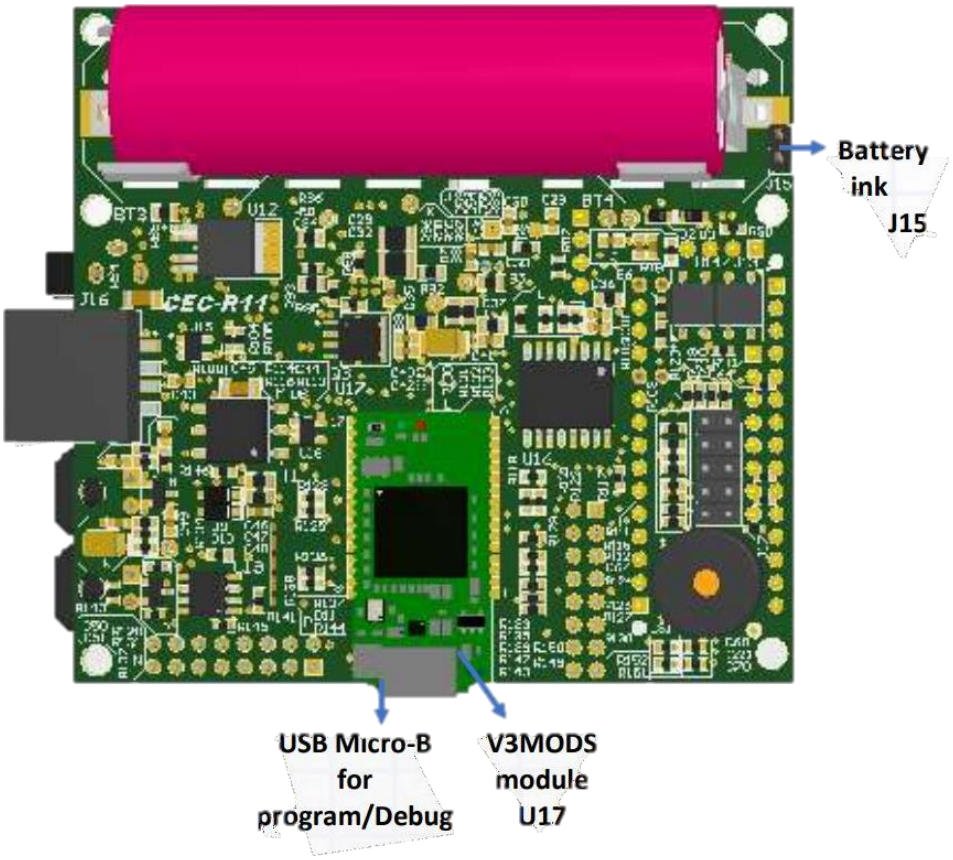


Fig 1.1.3 – Back view of dexter

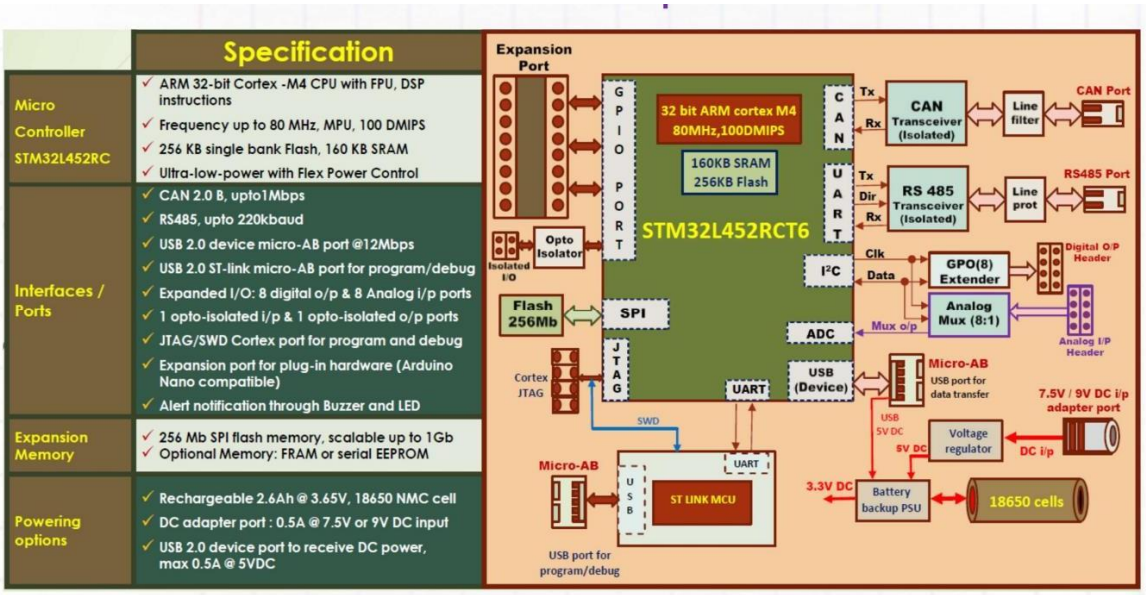


The outline of the dexter board is as follows

1. It consist of reset button
2. Dc power jack
3. USB micro AB for power supply
4. GPIO pins for inputs and outputs
5. Battery holders
6. Expansion slots
7. USB micro-b for program / debug
8. V3MODS module U17

BLOCK DIAGRAM

Fig 1.1.4- Block diagram of the dexter board



Communication Interfaces

Bidirectional serial ports used in Build Club experiments in an Asynchronous (data bytes transmitted between start and stop bits without a clock signal) as well as Half Duplex (can either only Transmit or only Receive at a given time) mode. The Baud Rate of the channel is configurable.

Table 1.1.2 – Communication interfaces of dexter board

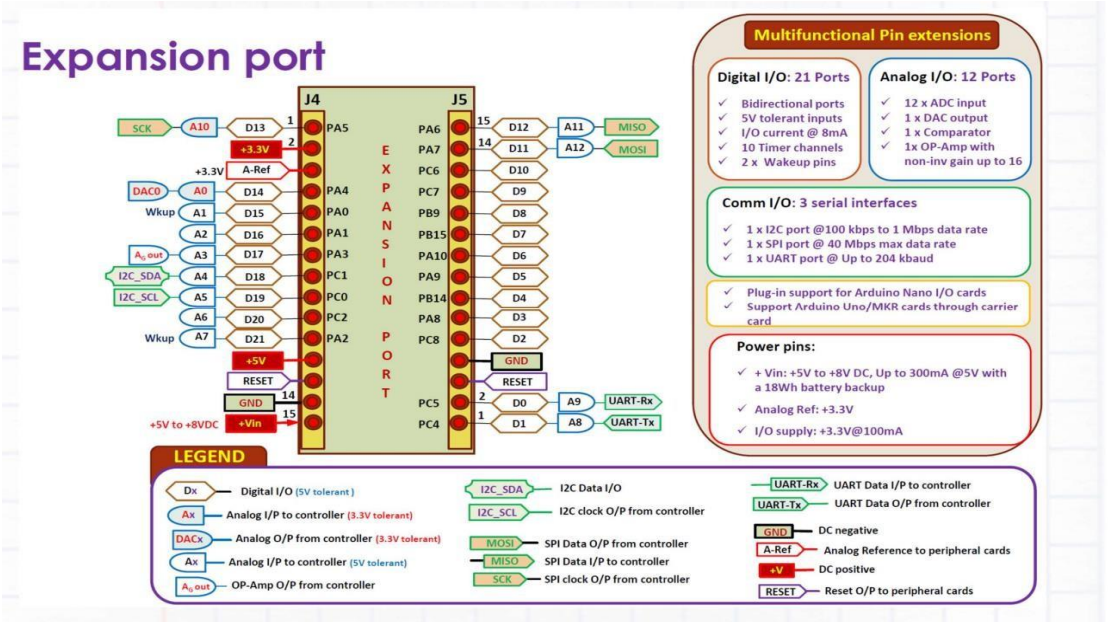
Channel	Function call	Description
UART1	Uart1_init (baud rate)	To initialize UART1 with Baud Rate
	Uart1_receive ()	To receive data via UART1 using PB6 (TX), PB7 (RX)
	Uart1_receive_IT ()	To receive data via UART1 using PB6 (TX), PB7 (RX) through Interrupt mechanism
	Uart1_transmit ()	To transmit data via UART1 using PB6 (TX), PB7 (RX)
UART2	Uart2_init (baud rate)	To initialize UART2 with Baud rate
	Uart2_receive ()	To receive data via UART2 using PA2 (TX), PA3 (RX)
	Uart2_receive_IT ()	To receive data via UART2 using PA2 (TX), PA3 (RX) through Interrupt mechanism
	Uart2_transmit()	To transmit data via UART2 using PA2 (TX), PA3 (RX)
UART3	Uart3_init (baud rate)	To initialize UART3 with Baud rate
	Uart3_receive()	To receive data via UART3 using PC4 (TX), PC5 (RX)
	Uart3_receive_IT ()	To receive data via UART3 using PC4 (TX), PC5 (RX) through Interrupt mechanism
	Uart3_transmit()	To transmit data via UART3 using PC4 (TX), PC5 (RX)

EXPANSION PORT OF THE DEXTER

The expansion ports of dexter board includes the gpio pins and their behaviours and special purposes

GPIO pins are also called as general-purpose input output. These pins can be used as either input as well as output pins. There are some special function pins which serve special functions and works like serial communication, uart protocol, reset pins.

Fig 1.1.5 – Expansion port of dexter



MEMORY DETAILS OF DEXTER

Table 1.1.3 – Memory segmentation of dexter

U16 : Flash Memory – SPI Interface

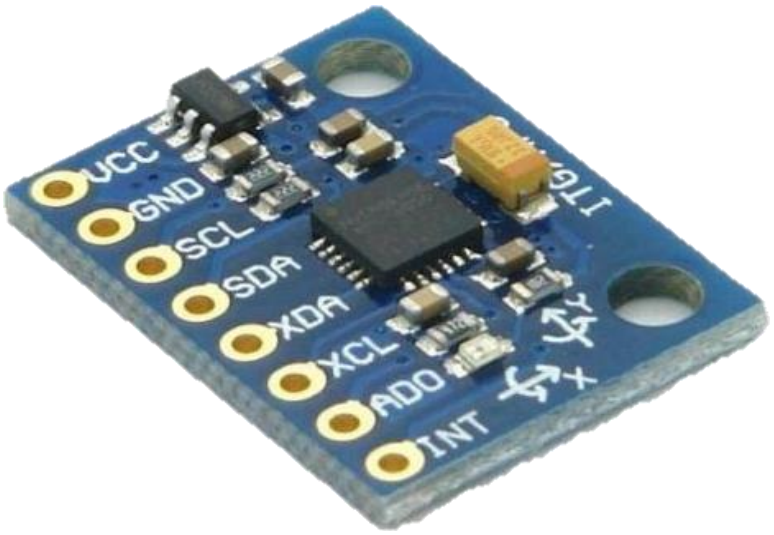
Pin #	Controller Pin mapping	Flash Memory Pin Name	Signal Name	Function
3	NRST	$\overline{\text{RESET}}$	FM_RST	Memory Reset
7	PA15	$\overline{\text{CS}}$	FM_CS	Chip Select
8	PC11	DO(IO1)	FM_MISO	Data I/P to Controller
9	PB5	$\overline{\text{WP(IO2)}}$	FM_WP	Write Protect
15	PC12	DI(IO0)	FM_MOSI	Data O/P from Controller
16	PC10	CLK	FM_SCK	Clock Input

U20 : FRAM Memory – I2C1 Interface

Pin #	Controller Pin mapping	FRAM Memory Pin Name	Signal Name	Function
5	PB7	SDA	uC_I2C1_SDA	Serial Data I/O
6	PB6	SCL	uC_I2C1_SCL	Serial Clock

3.2.2 MPU6050

Fig 1.2.1 – Mpu6050



MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3axiGyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package.
It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer, Pressure sensor etc. If 3-axis Magnetometer is connected to auxiliary I2C bus, then MPU6050can provide complete 9-axis Motion Fusion output.

3-AXIS GYROSCOPE

The MPU6050 consist of 3-axis Gyroscope with Micro Electro Mechanical System(MEMS) technology. It is used to detect rotational velocity along the X, Y, Z axes as shown in below figure.

Acceleration along the axes deflects the movable mass

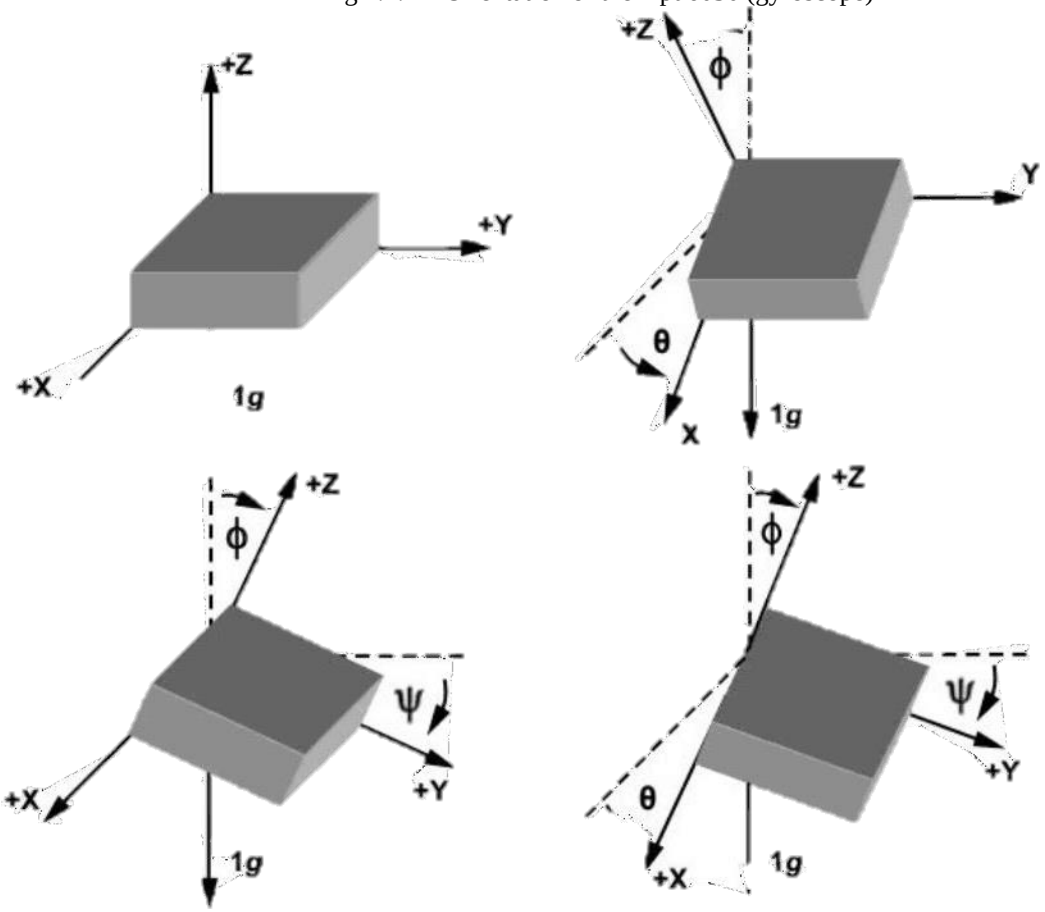
This displacement of moving plate (mass) unbalances the differential capacitor which results in sensor output. Output amplitude is proportional to acceleration.

16-bit ADC is used to get digitized output.

The full-scale range of acceleration are +/- 2g, +/- 4g, +/- 8g, +/- 16g.It measured in g (gravity force) unit.

When device is placed on flat surface it will measure 0g on X and Y axis and +1g on Z axis.

Fig 1.2.2—Orientation of the mpu6050 (gyroscope)



PINOUT OF MPU6050

1. INT: Interrupt digital output pin.

2. AD0: I2C Slave Address LSB pin. This is 0th bit in 7-bit slave address of device. If connected to VCC then it is read as logic one and slave address changes.

3. XCL: Auxiliary Serial Clock pin. This pin is used to connect other I2C interface enabled sensors SCL pin to MPU-6050.

4. XDA: Auxiliary Serial Data pin. This pin is used to connect other I2C interface enabled sensors SDA pin to MPU-6050.

5. SCL: Serial Clock pin. This pin to connected to microcontrollers SCL pin.

6. SDA: Serial Data pin. This pin is connected to microcontrollers SDA pin.

7. GND: Ground pin. This requires a ground connection.

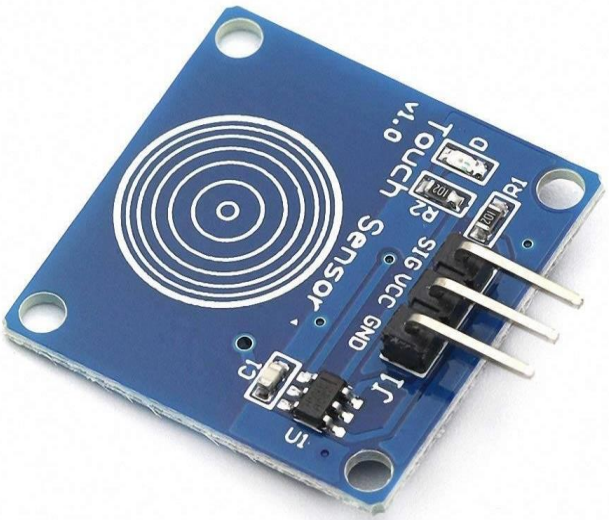
8. VCC: Power supply pin. This requires a +5V DC supply.

3.2.3 TOUCH SENSOR

The human body has five sense elements which are used to interact with our surroundings. Machines also need some sensing elements to interact with there surroundings. To make this possible sensor was invented. The invention of the first manmade sensor, thermostat, dates back to 1883. In 1940s infrared sensors were introduced. Today we have sensors that can sense motion, light, humidity, temperature, smoke, etc…Analog and digital both types of sensors are available today. Sensors have brought a revolutionary change in the size and costof various control systems. One of such sensor which can detect touch is the Touch sensor.

Touch Sensors are the electronic sensors that can detect touch. They operate as a switch when touched. These sensors are used in lamps, touch screens of the mobile, etc… Touch sensors offer an intuitive user interface.

Fig 1.3.1-Touch Sensor



Touch sensors are also known as Tactile sensors. These are simple to design, low cost and are produced in large scale. With the advance in technology, these sensors are rapidly replacing the mechanical switches. Based on their functions there are two types of touch sensors- Capacitive sensor and Resistive sensor. Capacitive sensors work by measuring capacitance and are seen in portable devices. These are durable, robust and attractive with low cost.
Resistive sensors don't depend on any electrical properties for operation. These sensors work by measuring the pressure applied to their surface.

WORKING PRINCIPLE:

Touch sensors work similar to a switch. When they are subjected to touch, pressure or force they getactivated and acts as a closed switch. When the pressure or contact is removed they act as an open switch.

Capacitive touch sensor contains two parallel conductors with an insulator between them. These conductors plates act as a capacitor with a capacitance value C0. When these conductor plates come in contact with our fingers, our finger acts as a conductive object.Due to this, there will be an uncertain increase in the capacitance.

A capacitance measuring circuit continuously measures the capacitance C0 of the sensor. When thiscircuit detects a change in capacitance it generates a signal.

The resistive touch sensors calculate the pressure applied on the surface to sense the touch. Thesesensors contain two conductive films coated with indium tin oxide, which is a good conductor of electricity, separated by a very small distance.

Across the surface of the films, a constant voltage is applied. When pressure is applied to the top film,it touches the bottom film. This generates a voltage drop which is detected by a controller circuit and signal is generated thereby detecting the touch.

ADVANTAGES

More convenient to use without moving parts

Provide increased reliability

Touch screens are also lighter than glass displays, making them less fragile.

3.2.4 BREAD BOARD

The term breadboard can be derived from 2 terms namely bread and board initially this was used to cut the bread into pieces further it was called a bread and it was used in electronic projects and electronic devices in the year 1970 a breadboard is also known as soulder less bored because the component used on the breadboard does not need any soldering to connect to the board so it can be reused.

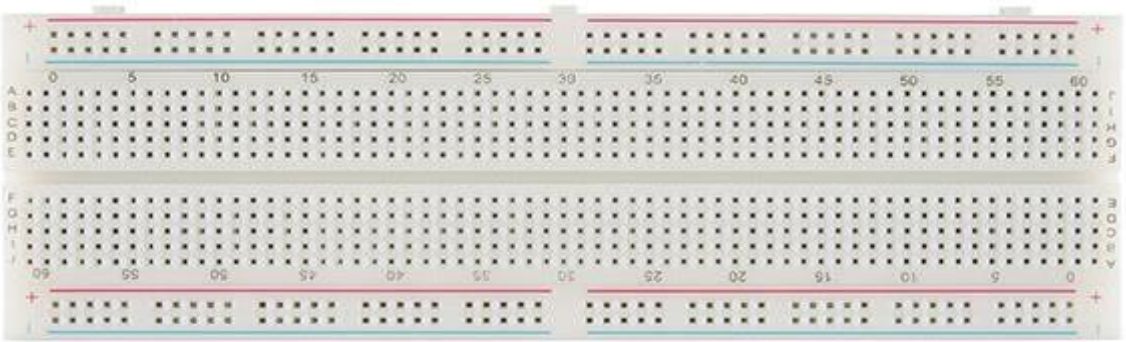


Fig 1.4.1 – Bread board

The arrangement of different components on the breadboard can be done by inserting the terminals into the breadboard so it is frequently known as black board definition is the plastic boat in rectangular shape that includes a lot of small components to build an electronic circuit is known as a breadboard the connection on the breadboard is not permanent but they can be connected without sold in the component.

If you make any mistake in the components you can place or remove the components effortlessly for beginners of Electronics this device is very useful to make mini projects if it is a simple so cute that the desert to analyse, then a breadboard gives a quick solution

The material used to make the breadboard is white plastic at present most of the breadboard are solderless type so we can directly plug in the component directly and connect them through the exterior power supply the different kinds of breadboards are accessible according to the specific point holes for instance 400 point type, and 830 point type

TYPES

Breadboards are classified into two types like a solderless breadboard and shoulderble breadboard

SOLDER LESS BREADBOARD

This is the most commonly used breadboard for prototyping as well as testing electronic circuits without soldring the components they said available different shapes sizes as well as ratings the circuit on these boards or not permanent so you can second Test the functionality

of a circuit before confirming it design on to the PCB. This breadboard rose and columns with holes that allow the lids of components and wire gauges.

The Terminal of the component does not place into the whole of the breadboard then a connecting wire can be solder to the lead of the component that will insert in the breadboard hole

SOLDERABLE BREADBOARD

This types of breadboards of a permanent set of 5 electronics circuits this kind of breadboards give a stronger setup it includes wholes for electronic components including copper tracking. These components can be solder using soldering iron for soldering the component to the breadboard so that an electrical connection can be formed to the copper tracing.

But designing a circuit needed for soldering separately in between this components to make a lane to permit the flow of current. Available in different

SPECIFICATIONS

- Distribution strips are 2
- Wire size is 21 to 26 AWG wire
- tie points are 200
- With standing voltage is 1000 volt AC

- Tie points within IC are 630
- Insulation resistance is DC 500V or 500Mohm
- Dimension is 6.5* 4.4* 0.3 inch
- Rating is 5 Amps
- Abs plastic through colour legend
- Abs heat distortion temperature is 183 F
- hole or pins style is 2.5mm

ADVANTAGES

- It is used to make a temporary prototype for the electronic projects
- This is reusable because it doesn't need any soldering
- These both are less weight because the material used to make this board is a lightweight plastic material
- Testing can be done very easily
- The arrangement of these components can be done very simply into the holes on the vote to make the design of a circuit
- It is economical and simple to use
- It does not use any difficult part
- Drilling is known necessary to connect the components because the holes on the breadboard are embedded already
- Modifying can be done very quickly
- We can add or remove the components on the breadboard
- These both are available in different sizes and shapes
- Can be adjusted very easily

DISADVANTAGES

- These bords are not used for higher current applications
- For low frequency applications low frequency boats or not used
- For making simple circuits it needs more physical space
- The number of connections on the breadboard can make the circuit my C because of the several wires
- The connections on the board can be disturbed once the components are connected or removed.
- Reliable connections are less
- Signaling is limited

SAFETY TIPS FOR BREADBOARD

- It is very important to connect a circuit systematically & neatly on a breadboard so that one can correct it & get it running simply & rapidly. It also assists once someone else requires knowing and inspecting the circuit. The following tips are very useful for

breadboard.

- Use the top & bottom bus rails always for connecting power supply instead of using a direct power supply
- When the jumper wires are coded with colour then it will help in reducing the confusion while designing a circuit.
- For instance, green colour wires are used for GND connections, red colour wire for +Ve power whereas black colour one is for -Ve power connections.
- Jumper wires should be connected lay flat on the board so that the board does not turn cluttered. Connect the jumper wires in the region of the ICs but not on the packages so that IC can be changed easily when required. Cutting the components leads can lead to insert very closely to the board.
- Be careful when connecting components . It is significant to be particularly careful while placing ICs into the holes of the board.
- Power supply terminals should not connect otherwise it may cause a short circuit.
- Once the board is connected to the power supply, do not leave it alone Do not stroke the IC elements with uncovered hands once the circuit supplies through it because they are sensitive components, so there is a chance to get damage.
- Once the power supply is given to the board, do not connect or remove components
- It is necessary to monitor exact polarity once certain components are connected to the circuit, otherwise, that may break down the dielectric within the component
- If water or liquid dropped onto the board, then right away remove it from the power supply
- Maintain your surroundings clean and in sequence

APPLICATIONS

1) The main application of a breadboard is to form simple electrical connections among different components so that you can check your circuit before soldering it to the board
2) These boards allow different components to be simply placed or removed or the term prototyping instantly comes to mind permanently.
3) If a designer designs a simple circuit or module then they need to check, so this board offers a fast & cheap solution

3.2.5 JUMPER WIRES

A jump wire (also known as jumper wire, or jumper) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them-simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering. Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

Fig 1.5.1 Jumper wires

Jumper wires at the end of a multi-coloured ribbon cable are used to connect the pin header at the left side of a blue USB2Serial board to a white breadboard below. Another jumper cable ending in a USB micro male connector mate to the right side of the USB2Serial board.

Red and black tinned jump wires can be seen on the breadboard. There are different types of jumper wires. Some have the same type of electrical connector at both ends, while others have different connectors. Some common connectors are:

1. Solid tips - are used to connect on/with a breadboard or female header connector. The arrangement of the elements and ease of insertion on a breadboard allows increasing the mounting density of both components and jump wires without fear of short-circuits. The jump wires vary in size and colour to distinguish the different working signals.

2. Crocodile clips - are used, among other applications, to temporarily bridge sensors. Buttons and other elements of prototypes with components or equipment that have arbitrary connectors, wires, screw terminals, etc.

3. Banana connectors - are commonly used on test equipment for DC and low- frequency AC signals.

4. Registered jack (RJnn) is commonly used in telephone (RJII) and computer networking (RJ45). Which is used to connect antennas and other components to network cabling. Jumpers are also used in base stations to connect antennas to radio units. Usually, the most bendable jumper cable diameter is 1/2"RCA connectors - are often used for audio, low resolution composite video signals, or other low-frequency applications requiring a shielded cable.

5. RF connectors are used to carry radio frequency signals between circuits, test equipment, and antennas. RF jumper cables Jumper cables is a smaller and more bendable corrugated cable.

6. Male connectors - A connector type with pins instead of holes. These connectors are inserted into a female connector. Good examples of male connectors are power plugs and coaxial cables. In the example picture, the power cord connector on the left-side with holes is a female connector, and on the right-side with pins that connects to the wall outlet is a male connector.

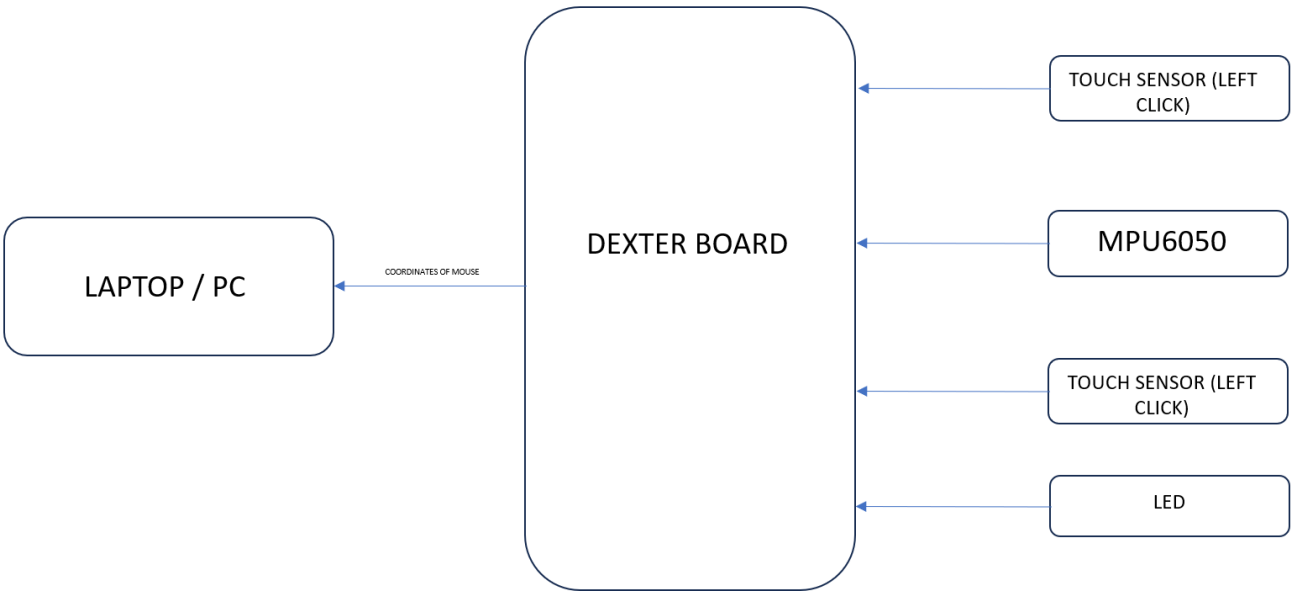
7. **Female connector** - A female connector is a connector attached to a wire, cable, or piece of hardware, having one or more recessed holes with electrical terminals inside, and constructed in such a way that a plug with exposed conductors (male connector) can be inserted snugly into it to ensure a reliable physical and electrical connection. A female connector is also known as a jack, outlet, or receptacle. This type of connector can be recognized by the fact that, when it is disconnected therefore are not likely to make accidental contact with external objects or conductors. The most common female connector is a two- or three-prong electrical outlet, also known as a wall outlet. Other often-encountered examples include telephone jacks, the jacks for headsets, the chassis connectors for coaxial cable, and some D-shell connectors for computer serial and parallel ports.

ADVANTAGES

- Low cost.
- Perfect for Prototyping and testing small circuits. Perfect for all who are starting with Electronics.
- You can change the circuit at any time

3.3 BLOCK DIAGRAM

Fig 3.3.1 Block diagram of gesture controlled mouse for PC



This is the block diagram of a project which consists of the main blocks

- Dexture board, mpu 6050, 2 touch sensors.
- These block are very important for transferring the mouse coordinates to the laptop or PC
- There are two blocks touch sensors which function the electric and the right click of the mouse

Dexter board

The Dexter Board is designed using the microcontroller STM32L452RCT6 (LQFP64 package) from STMicroelectronics. It is a versatile microcontroller, which can be used in a variety of ways.

MPU6050

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer Pressure sensor etc. If 3-axis Magnetometer is connected to auxiliary I2C bus, then MPU6050 can provide complete 9-axis Motion Fusion output.

TOUCH SENSOR

This blocks are to mimic the mouse in real time

Here in this project we are reading the mouse actions with gestures using a 6 Axis gyroscope, and send this data or coordinates to the laptop or pc's

LED

led is used for the indication of the calibration of the mouse to the mpu6050 . This block is important as it represents that the calibration is done or not.

3.4 CONCLUSION

This chapter has described about each component and block diagram of Gesture Controlled Mouse for PC

CHAPTER-4
IMPLEMENTATION

4.1 INTRODUCTION

This chapter explains about the implementation of Gesture Controlled Mouse for PC.

4.2 IMPLEMENTATION DETAILS

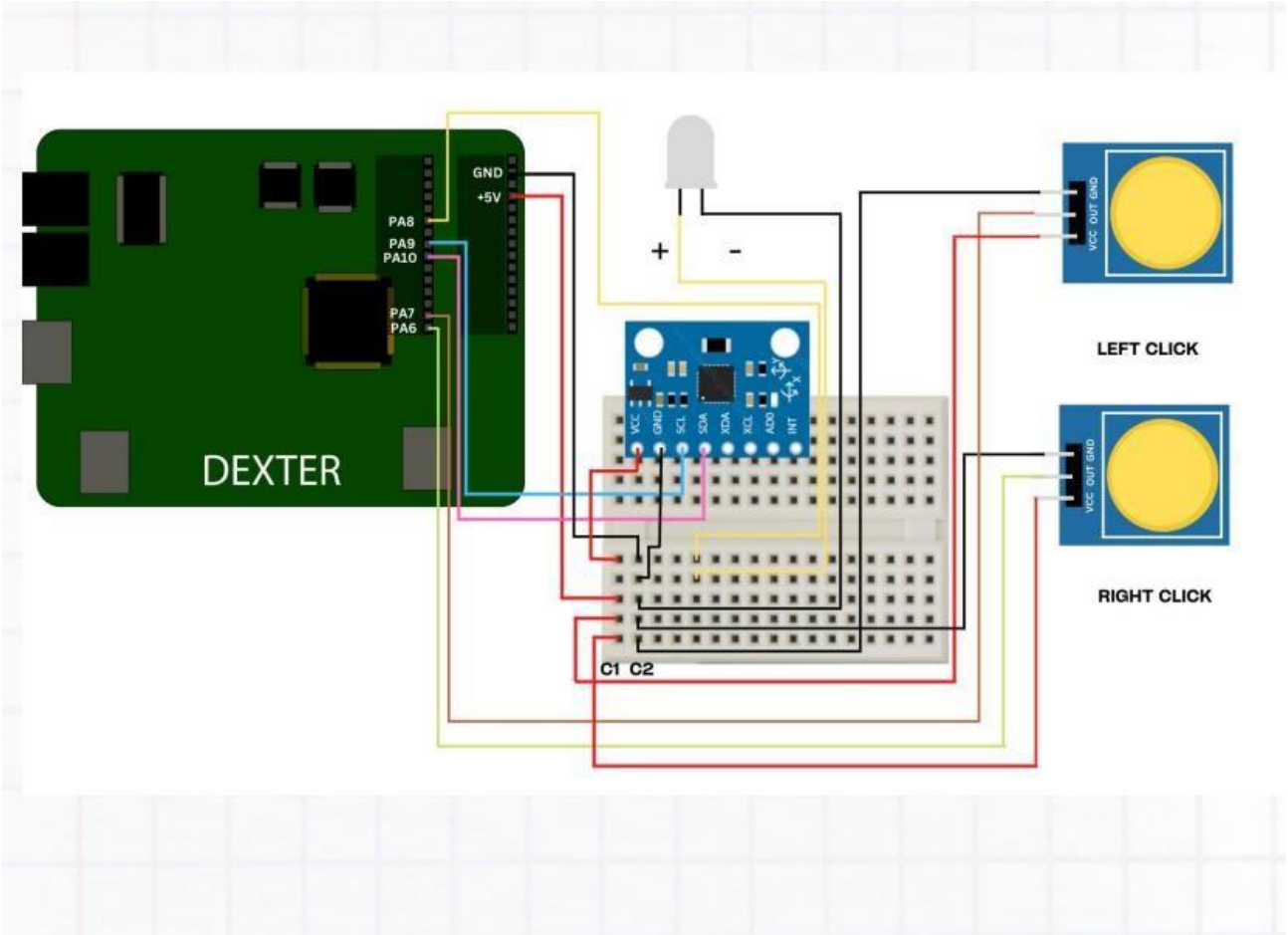
Table 4.1.1- Implementation details

Sr no	Component Name	Quantity
01	Dexter board	1
02	Mpu6050 Sensor	1
03	Touch Sensor	2
04	Breadboard	1
05	Jumper wires	1 bundle
06	USB type B cable	2
07	Wrist support band	1

These are the implementation details of the project:
After the complete design of prototype, the user wears the wrist support band to his hand on which the mpu6050 and touch sensors with breadboard are sticked to the wrist hand.
This complete set is connected to the dexter board and the dexter board is connected to the PC.

4.3 SCHEMATIC DIAGRAM

Fig 4.1.1 Schematic diagram of Gesture Controlled Mouse for PC



CONSTRUCTION

- Circuit diagram of we have many blocks
- The main microcontroller which is dexter which helps to communicate between the input and the output including the sensors to send the data
- In which the microcontroller has 30 pins to read or send different data’s
- it has 3 UART channels and 1 CAN protocol and SPI interface and 1 ADC
- Here we use GPIO opens for receiving the data from MPU 6050

4.4 WORKING

This project works in a very simple manner. The dexter board is the main microcontrollerwhich connect is the input and the output and there is one gyroscope sensor with accelerometer which is called MPU 6050 and two touch sensors with one red LED

The voltage that is required for all the components is taken from the microcontroller on the breadboard and all the circuitry are built on the breadboard itself.

When the device is powered on the red LED terms on indicating that the calibration is under process.

After while the LED automatically turns of indicating the calibration is complete between gyroscope and the device Mouse.

You can also notice that

If you want use left or right click of the mouse you can use the touch sensor which mimics the left and right click of the mouse whenever you touch the left button touch sensor the leftclick is clicked on the device as well as with the right touch sensor

After completion of calibration, we notice that the mouse most according to the moment of gyroscope.

1. Sensing Elements:

Accelerometer: The accelerometer measures acceleration along three perpendicular axes: X, Y, and Z. It detects both static (gravity) and dynamic (movement) acceleration.

Gyroscope: The gyroscope measures angular velocity around the same three axes. It senses rotational movements.

2. Internal ADC:

The analog outputs of the accelerometer and gyroscope sensors are fed into internal analog-to-digital converters (ADCs) within the MPU6050.

The ADCs convert the analog voltage levels from the sensors into digital values that can be processed by the internal digital circuitry.

3. Digital Signal Processing:

The MPU6050 contains digital signal processing circuitry to further process the digital sensor data.

It applies filtering, noise reduction, and calibration to ensure accurate measurements.The processed data is then made available for reading via the I2C interface

4.I2C Communication:

The MPU6050 communicates with external devices, such as microcontrollers, using the I2C (Inter-Integrated Circuit) communication protocol.

I2C requires two lines: SDA (Serial Data) and SCL (Serial Clock).

The microcontroller acts as the master device, while the MPU6050 is the slave device.

5.Data Transmission:

The MPU6050 provides several registers accessible via I2C, where sensor data, configuration settings, and status information are stored.

The microcontroller sends commands to read specific registers to retrieve data.

Data from the accelerometer and gyroscope, as well as temperature data, can be read fromthese registers.

6.Motion Fusion Algorithms:

To obtain more accurate motion and orientation data, the MPU6050's sensor outputs can be combined using sensor fusion algorithms.

Common algorithms like Kalman filters or complementary filters are often used to combine accelerometer and gyroscope data, compensating for the strengths and weaknesses of each sensor.

7.Calibration:

Calibration is essential to eliminate bias, scale factor, and other errors inherent in sensor readings.

Some level of calibration is usually required to obtain accurate measurements

4.5 CONCLUSION

This chapter explained about the implementation details and working of Gesture Controlled Mouse for PC

CHAPTER-5
RESULTS

5.1 IMPLEMENTATION RESULTS

Fig 5.1.1 Hardware implementation of Gesture controlled mouse for PC

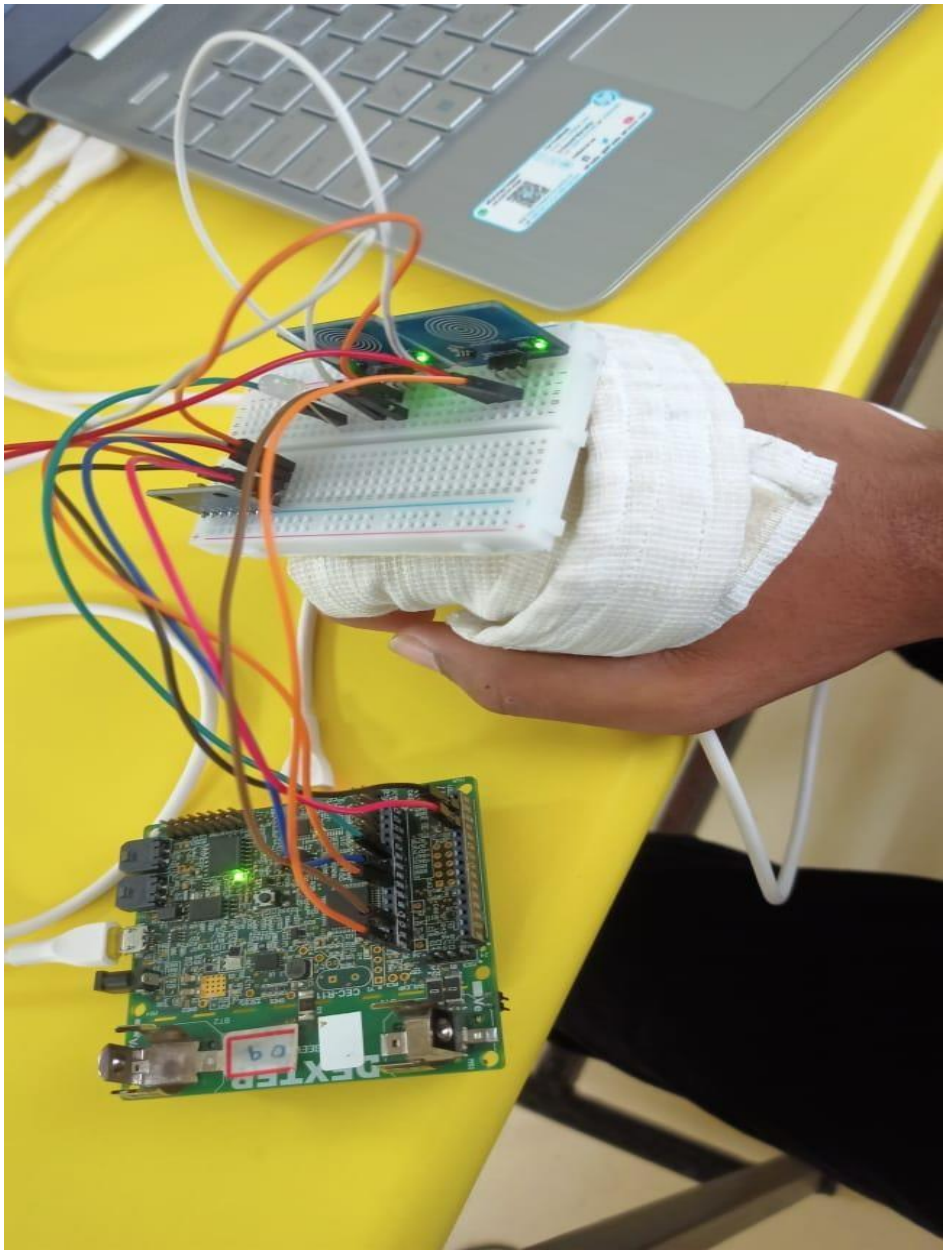


Fig 5.1.2 left click

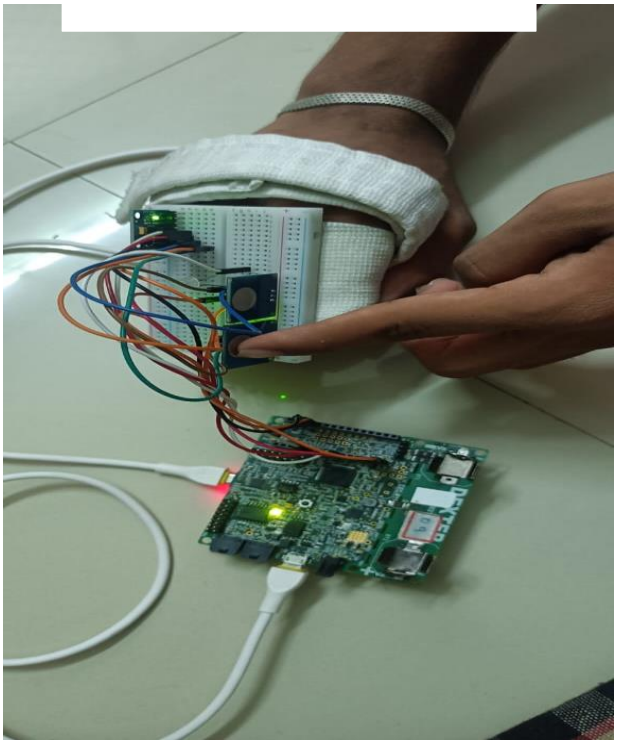


Fig 5.1.3 right click

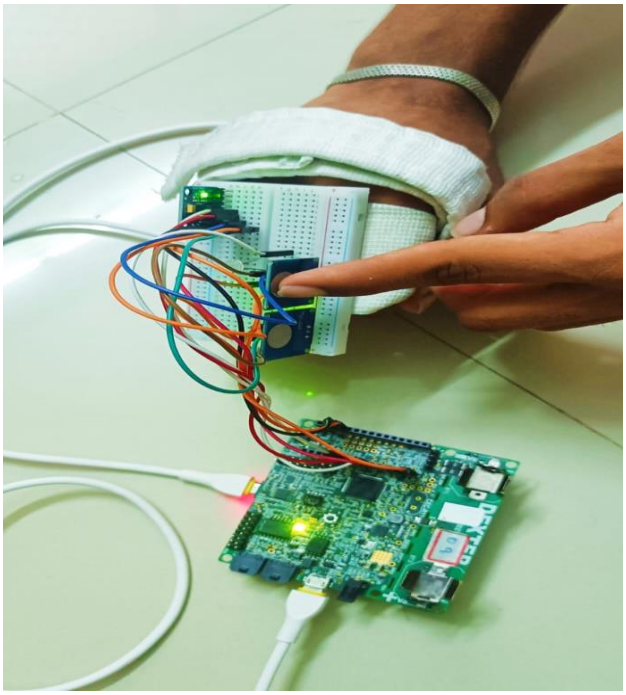


Fig 5.1.4 moving forward

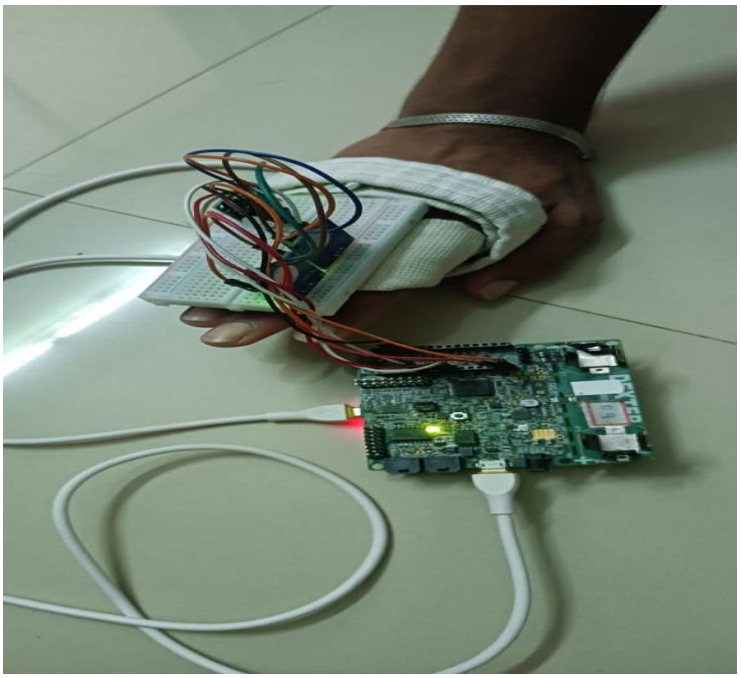


Fig 5.1.5 moving backward

Fig 5.1.6 moving left

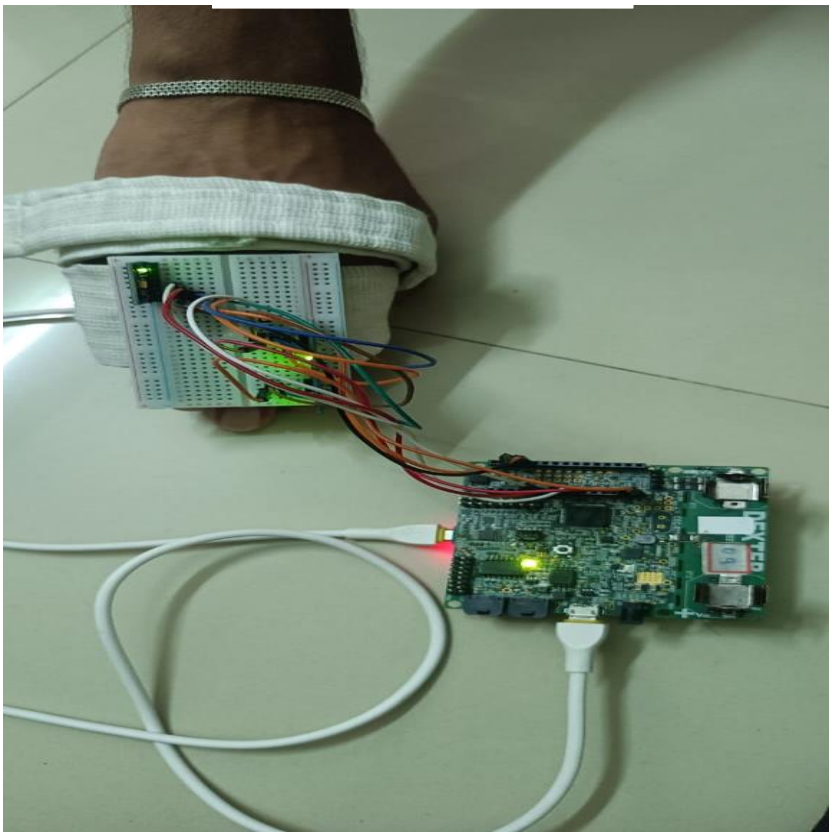
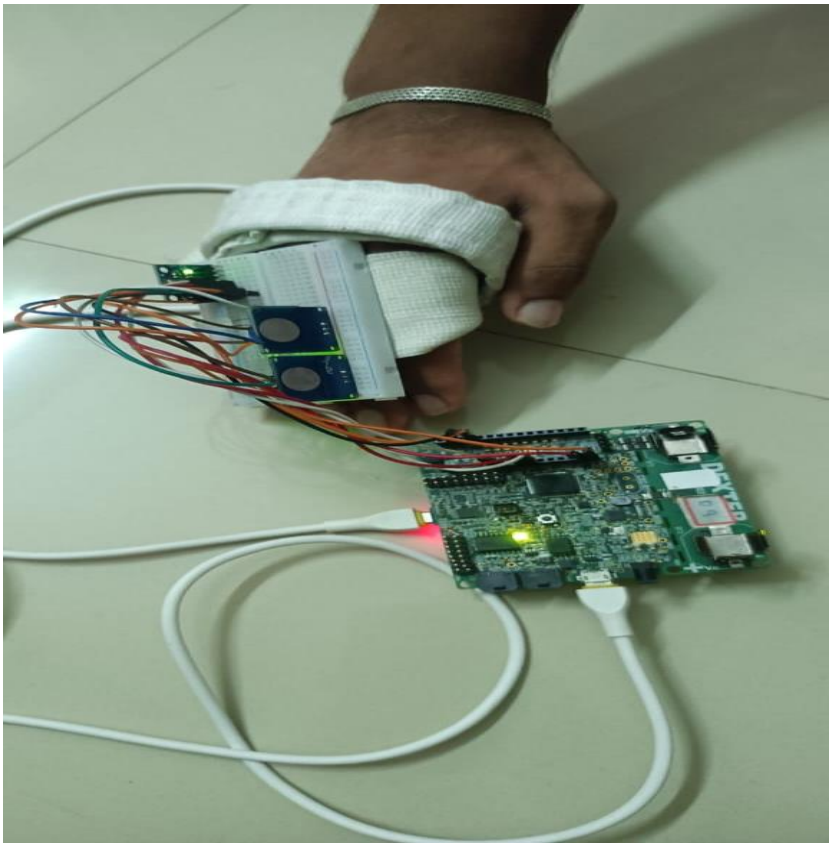


Fig 5.1.7 moving right

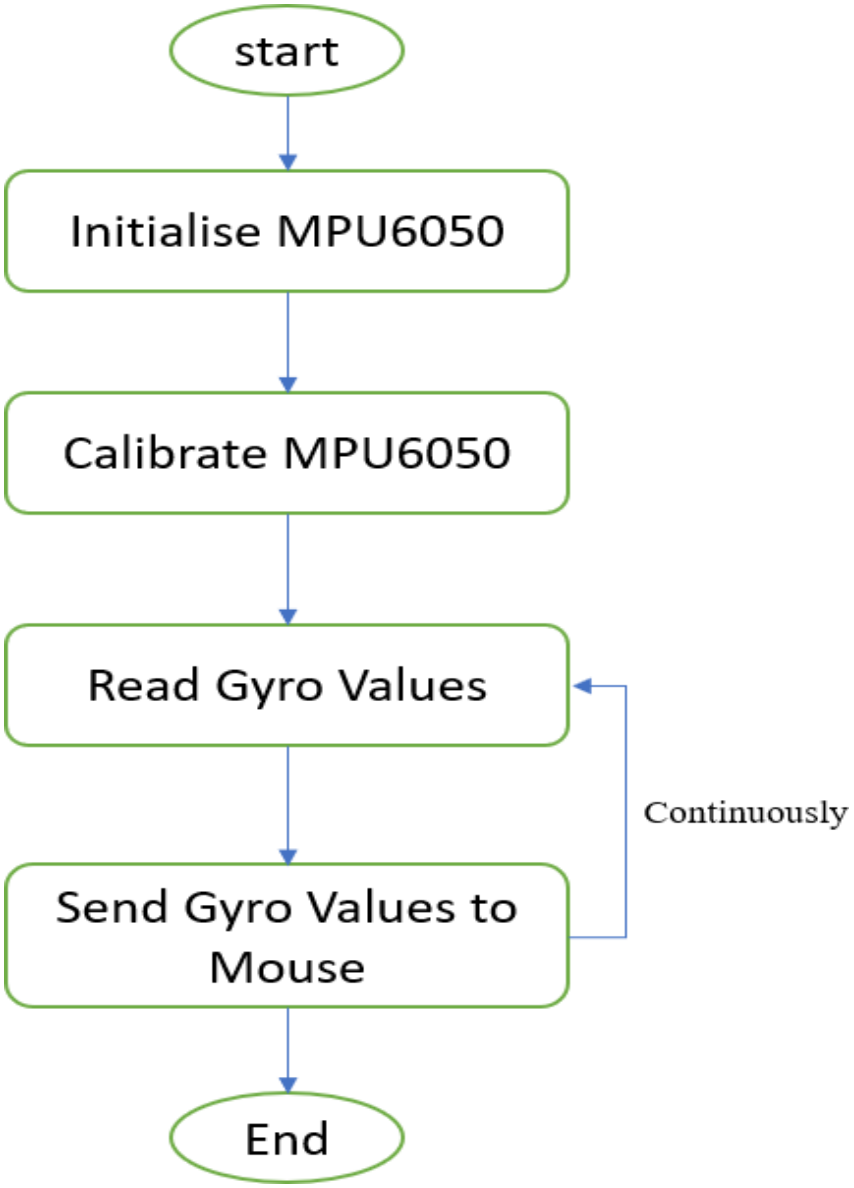


5.2 METHODOLOGY

- Here we used a dexter board
- Which is connected to the laptop or pc
- And used a hand wearable belt for the hand which consists of the gyroscope to send the details of the different axis's
- And they are sent to the pc or laptop mimicking the work of a mouse

5.3 FLOW CHART

Fig 5.3.1 Flow chart



We implemented the project which sends the mouse operation using a hand with dexter board and mpu6050 sensor. Which is very much useful for a disabled person and willing to use a pc or a laptop who has less hand mobility.

5.4 CONCLUSION

This chapter showed the implementation results of gesture controlled mouse for PC.

CHAPTER-6
CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

Developed a Gesture-Controlled Mouse for PC, tailored specifically for individuals with hand impairments, represents a significant stride towards enhancing inclusivity and accessibility in the realm of human-computer interaction.

In a rapidly evolving technological landscape, our Gesture-Controlled Mouse project stands as a testament to the power of innovation, collaboration, and empathy.

By bridging the gap between advanced sensor technologies and the unique needs of our target users, we have paved the way for a more inclusive digital future.

Ultimately, our project contributes to a world where technology adapts to human diversity, ensuring that no one is left behind in the digital age.

6.2 FUTURE SCOPE

The project can be further improved by making it more reliable by implementing machine leaning algorithms and automation. Using more faster boards like raspberry pi and using artificial intelligence and machine learning algorithms, making it easier to use.

Predicting the gesture using AIML and converting the data and making the decisions.

REFERENCES

1. Renuka Annachhatre, Miti Tamakuwala, Prathamesh Shinde, Abhisha Jain, Pradnya V. Kulkarni, "Virtual Mouse Using Hand Gesture Recognition - A Systematic Literature Review", *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, pp.1-6, 2022.

2. Tomonori Shohata, Yuki Takeda, Noboru Nakamichi, Keita Watanabe, ToshiyaYamada, "Spotlight Type Pointing Environment for Librarian", 2021 IEEE 10thGlobal Conference on Consumer Electronics (GCCE), pp.703-706, 2021.

3. Aditi Khandagale, Nidhi Thakkar, Swarali Patil, Vaibhavi Jadhav, Charusheela Nehete, "Jarvis - AI Based Virtual Mouse", *2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*, pp.1-5, 2023.

4. Subhash Chand Agrawal, Rajesh Kumar Tripathi, Neeraj Bhardwaj, Prashun Parashar, "Virtual Drawing: An Air Paint Application", *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, pp.971-975, 2023.

5. Shravani Belgamwar, Sahil Agrawal, "An Arduino Based Gesture Control System for Human-Computer Interface", *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp.1-3, 2018.

6. Akula Kumar Raja, Chidakash Sugandhi, Gorantla Nymish, Nama Sai Havish, Manazhy Rashmi, "Convolutional Neural Network Based Virtual Mouse", *2023 4th International Conference for Emerging Technology (INCET)*, pp.1-7, 2023.

APPENDIX

```
#include "main.h"
#include "usb_device.h"

#include "usbd_hid.h"

ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim1;

UART_HandleTypeDef huart3;
void SystemClock_Config(void);
void PeriphCommonClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_I2C1_Init(void);
static void MX_TIM1_Init(void);
static void MX_USART3_UART_Init(void);
static void MPU6050_Init(void);
int16_t min_zval = 128;
int16_t max_zval = -128;
int16_t min_yval = 128;
int16_t max_yval = -128;
int16_t newzval = 0;
int16_t newyval = 0;
uint8_t L_button_flag = 0;
uint8_t R_button_flag = 0;
int16_t Gyro_X_RAW = 0;
int16_t Gyro_Y_RAW = 0;
int16_t Gyro_Z_RAW = 0;
int sensitivity = 1;
float Gx, Gy, Gz;
#define MPU6050_ADDR 0xD0
#define SMPLRT_DIV_REG 0x19
#define GYRO_CONFIG_REG 0x1B
#define ACCEL_CONFIG_REG 0x1C
#define ACCEL_XOUT_H_REG 0x3B
#define TEMP_OUT_H_REG 0x41
#define GYRO_XOUT_H_REG 0x43
#define PWR_MGMT_1_REG 0x6B
#define WHO_AM_I_REG 0x75
extern USBD_HandleTypeDef hUsbDeviceFS;
typedef struct {
        uint8_t button;
```

```
		int8_t mouse_z;
		int8_t mouse_y;
		int8_t wheel;

} mouseHID;
mouseHID mousehid = { 0, 0, 0, 0 };
uint8_t Rx_Data[6];
int16_t x, y, z;
void MPU6050_Init(void) {
		uint8_t check;
		uint8_t Data;
		HAL_I2C_Mem_Read (&hi2c1, MPU6050_ADDR, WHO_AM_I_REG,
1,&check, 1, 1000);

		if (check == 104) // 0x68 will be returned by the sensor if everything goes well
			{
		 Data = 0;
		 HAL_I2C_Mem_Write (&hi2c1, MPU6050_ADDR,
PWR_MGMT_1_REG,1, &Data, 1,1000);
		 Data = 0x07;
		 HAL_I2C_Mem_Write (&hi2c1, MPU6050_ADDR, SMPLRT_DIV_REG,
1,&Data, 1,1000);
		 Data = 0x00;
		 HAL_I2C_Mem_Write (&hi2c1, MPU6050_ADDR,
ACCEL_CONFIG_REG,1, &Data, 1,1000);
		 Data = 0x00;
		 HAL_I2C_Mem_Write (&hi2c1, MPU6050_ADDR,
GYRO_CONFIG_REG,1, &Data, 1,1000);
		}

}
void MPU6050_Read_Gyro(void) {
		uint8_t Rec_Data[6];
		 HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR, GYRO_XOUT_H_REG,
1, Rec_Data, 6,1000);
		Gyro_X_RAW = (int16_t) (Rec_Data[0] << 8 | Rec_Data[1]);
		Gyro_Y_RAW = (int16_t) (Rec_Data[2] << 8 | Rec_Data[3]);
		Gyro_Z_RAW = (int16_t) (Rec_Data[4] << 8 | Rec_Data[5]);
		Gx = Gyro_X_RAW / 131.0;
		Gy = Gyro_Y_RAW / 131.0;
		Gz = Gyro_Z_RAW / 131.0;
}


void MPU6050_CALIB (void) {

		for (int i = 0; i < 50; i++) {

		 MPU6050_Read_Gyro ();
```

```
                min_zval = MIN (min_zval, Gz);
                max_zval = MAX(max_zval, Gz);
                min_yval = MIN(min_yval, Gy);
                max_yval = MAX(max_yval, Gy);
                HAL_Delay(100);
                }
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
}

void MPU6050_SenttoMouse() {

                if (Gz < min_zval)
                newzval = Gz - min_zval;
                if (Gz > max_zval)
                newzval = Gz - max_zval;
                if (Gy < min_yval)
                newyval = Gy - min_yval;
                if (Gy > max_yval)
                newyval = Gy - max_yval;

                if ((newzval > 10) || (newzval < -10)) {
                mousehid.mouse_y = newzval / sensitivity;
                }

                else
                mousehid.mouse_y = 0;

                if ((newyval > 10) || (newyval < -10)) {
                mousehid.mouse_z = newyval / sensitivity;
                }

                else
                mousehid.mouse_z = 0;

                if (L_button_flag == 1) {
                mousehid.button = 2;  // left click =1, right click =2
                USBD_HID_SendReport(&hUsbDeviceFS, &mousehid, sizeof(mousehid));
                HAL_Delay(50);
                mousehid.button = 0;  // left click =1, right click =2
                USBD_HID_SendReport(&hUsbDeviceFS, &mousehid, sizeof(mousehid));
                L_button_flag = 0;
                }
                if (R_button_flag == 1) {
                mousehid.button = 1;  // left click =1, right click =2
                USBD_HID_SendReport(&hUsbDeviceFS, &mousehid, sizeof(mousehid));
                HAL_Delay(50);
                mousehid.button = 0;  // left click =1, right click =2
                USBD_HID_SendReport(&hUsbDeviceFS, &mousehid, sizeof(mousehid));
                R_button_flag = 0;
```

```
                }

                USBD_HID_Se
                ndReport(&hUs
                bDeviceFS,
                &mousehid,
                sizeof(mousehi
                d));
                HAL_Delay(10);
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
                if (GPIO_Pin == GPIO_PIN_7) {
                 L_button_flag = 1;
                }
                if (GPIO_Pin == GPIO_PIN_6) {
                 R_button_flag = 1;
                }
}
int main(void) {
                HAL_Init();
                SystemClock_Config();
                PeriphCommonClock_Config();
                MX_GPIO_Init();
                MX_DMA_Init();
                MX_ADC1_Init();
                MX_I2C1_Init();
                MX_USB_DEVICE_Init();
                MX_TIM1_Init();
                MX_USART3_UART_Init();
                MPU6050_Init();
                MPU6050_CALIB();
app();
}
void app()
{
                while(1){
                 MPU6050_Read_Gyro();
                 MPU6050_SenttoMouse();
                }
}
```

```
void SystemClock_Config(void) {
		RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
		RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
		if
(HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1)
			!= HAL_OK) {
		 Error_Handler();
		}
		RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
		RCC_OscInitStruct.MSIState = RCC_MSI_ON;
		RCC_OscInitStruct.MSICalibrationValue = 0;
		RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
		RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
		RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
		RCC_OscInitStruct.PLL.PLLM = 1;
		RCC_OscInitStruct.PLL.PLLN = 36;
		RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
		RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
		RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
		if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
		 Error_Handler();
		}
		RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK |
RCC_CLOCKTYPE_SYSCLK
			| RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
		RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
		RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
		RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
		RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

		if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) !=
HAL_OK) {
		 Error_Handler();
		}
}
```

```
void PeriphCommonClock_Config(void) {
                RCC_PeriphCLKInitTypeDef PeriphClkInit = { 0 };
                PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USB |
RCC_PERIPHCLK_ADC;
                PeriphClkInit.AdcClockSelection = RCC_ADCCLKSOURCE_PLLSAI1;
                PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLLSAI1;
                PeriphClkInit.PLLSAI1.PLLSAI1Source = RCC_PLLSOURCE_MSI;
                PeriphClkInit.PLLSAI1.PLLSAI1M = 1;
                PeriphClkInit.PLLSAI1.PLLSAI1N = 24;
                PeriphClkInit.PLLSAI1.PLLSAI1P = RCC_PLLP_DIV7;
                PeriphClkInit.PLLSAI1.PLLSAI1Q = RCC_PLLQ_DIV2;
                PeriphClkInit.PLLSAI1.PLLSAI1R = RCC_PLLR_DIV2;
                PeriphClkInit.PLLSAI1.PLLSAI1ClockOut = RCC_PLLSAI1_48M2CLK
                        | RCC_PLLSAI1_ADC1CLK;
                if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) {
                 Error_Handler();
                }
}
static void MX_ADC1_Init(void) {
        ADC_ChannelConfTypeDef sConfig = { 0 };
         hadc1.Instance = ADC1;
        hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
        hadc1.Init.Resolution = ADC_RESOLUTION_12B;
        hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
         hadc1.Init.ScanConvMode = ADC_SCAN_ENABLE;
         hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
        hadc1.Init.LowPowerAutoWait = DISABLE;
        hadc1.Init.ContinuousConvMode = DISABLE;
        hadc1.Init.NbrOfConversion = 3;
        hadc1.Init.DiscontinuousConvMode = DISABLE;
        hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIG_T1_TRGO;
                hadc1.Init.ExternalTrigConvEdge =
ADC_EXTERNALTRIGCONVEDGE_RISING;
                hadc1.Init.DMAContinuousRequests = DISABLE;
                hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
                hadc1.Init.OversamplingMode = DISABLE;
                if (HAL_ADC_Init(&hadc1) != HAL_OK) {
                 Error_Handler();
                }
                sConfig.Channel = ADC_CHANNEL_1;
                sConfig.Rank = ADC_REGULAR_RANK_1;
                sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
                sConfig.SingleDiff = ADC_SINGLE_ENDED;
                sConfig.OffsetNumber = ADC_OFFSET_NONE;
                sConfig.Offset = 0;
                if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
                 Error_Handler();
                }
                sConfig.Rank = ADC_REGULAR_RANK_2;
                if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
```

```
                Error_Handler();
                }
                sConfig.Rank = ADC_REGULAR_RANK_3;
                if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
                Error_Handler();
                }

}
static void MX_I2C1_Init(void) {
                hi2c1.Instance = I2C1;
                hi2c1.Init.Timing = 0x10808DD3;
                hi2c1.Init.OwnAddress1 = 0;
                hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
                hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
                hi2c1.Init.OwnAddress2 = 0;
                hi2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
                hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
                hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
                if (HAL_I2C_Init(&hi2c1) != HAL_OK) {
                Error_Handler();
                }
                if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1,
I2C_ANALOGFILTER_ENABLE)
                        != HAL_OK) {
                Error_Handler();
                }
                if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK) {
                Error_Handler();
                }

        static void MX_TIM1_Init(void) {
        TIM_ClockConfigTypeDef sClockSourceConfig = { 0 };
        TIM_MasterConfigTypeDef sMasterConfig = { 0 };
         TIM_OC_InitTypeDef sConfigOC = { 0 };
        TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = { 0 };
        htim1.Instance = TIM1;
        htim1.Init.Prescaler = 3999;
        htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
        htim1.Init.Period = 999;
        htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
         htim1.Init.RepetitionCounter = 0; htim1.Init.AutoReloadPreload =
        TIM_AUTORELOAD_PRELOAD_DISABLE;
        if (HAL_TIM_Base_Init(&htim1) != HAL_OK) { Error_Handler();
        }
        sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
         if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) !=HAL_OK) {
        Error_Handler();
        }
        if (HAL_TIM_PWM_Init(&htim1) != HAL_OK) { Error_Handler();
        }
```

```
sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
sMasterConfig.MasterSlaveMode =TIM_MASTERSLAVEMODE_ENABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
!= HAL_OK) {
Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET; if
(HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC,
TIM_CHANNEL_1)
!= HAL_OK) {
Error_Handler();
}
sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
sBreakDeadTimeConfig.DeadTime = 0; sBreakDeadTimeConfig.BreakState =
TIM_BREAK_DISABLE; sBreakDeadTimeConfig.BreakPolarity =
TIM_BREAKPOLARITY_HIGH;
sBreakDeadTimeConfig.BreakFilter = 0;
sBreakDeadTimeConfig.Break2State = TIM_BREAK2_DISABLE;
sBreakDeadTimeConfig.Break2Polarity = TIM_BREAK2POLARITY_HIGH;
sBreakDeadTimeConfig.Break2Filter = 0;
sBreakDeadTimeConfig.AutomaticOutput =
TIM_AUTOMATICOUTPUT_DISABLE;
if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig)
!= HAL_OK) {
Error_Handler()
}
}
static void MX_USART3_UART_Init(void) {
huart3.Instance = USART3; huart3.Init.BaudRate = 9600;
huart3.Init.WordLength = UART_WORDLENGTH_8B;
huart3.Init.StopBits = UART_STOPBITS_1;
huart3.Init.Parity = UART_PARITY_NONE;
huart3.Init.Mode = UART_MODE_TX_RX;
huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart3.Init.OverSampling = UART_OVERSAMPLING_16;
huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT; if
(HAL_UART_Init(&huart3) != HAL_OK) {
Error_Handler();
}
}
```

```
        static void MX_DMA_Init(void) {
        HAL_RCC_DMA1_CLK_ENABLE();
        HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
        HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);

        }
        static void MX_GPIO_Init(void) {
        GPIO_InitTypeDef GPIO_InitStruct = { 0 };
        HAL_RCC_GPIOH_CLK_ENABLE();
        HAL_RCC_GPIOC_CLK_ENABLE();
        HAL_RCC_GPIOA_CLK_ENABLE(); HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8,
        GPIO_PIN_SET);
        GPIO_InitStruct.Pin = GPIO_PIN_6;
        GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING; GPIO_InitStruct.Pull =
        GPIO_PULLDOWN;
        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
        GPIO_InitStruct.Pin = GPIO_PIN_7;
        GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING; GPIO_InitStruct.Pull =
        GPIO_PULLDOWN;
        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); GPIO_InitStruct.Pin = GPIO_PIN_8;
        GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; GPIO_InitStruct.Pull =
        GPIO_NOPULL
                GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
                HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
                HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
                HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);

}
void Error_Handler(void) {
                __disable_irq();
                while (1) {
                }
}

#ifdef  USE_FULL_ASSERT
void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif /* USE_FULL_ASSERT */


#endif /* USE_FULL_ASSERT */
```