

A Project Report on
DEVELOPMENT OF WIRELESS AIR MOUSE

in the partial fulfilment of the requirement for the award of the degree of

Bachelor of Engineering
in
Electronics and Communication Engineering

Submitted by

K. Aditya	(160116735143)
M. Deekshith	(160116735148)
M. Sai Vikas	(160116735169)

Under the Supervision of

G. Mallikharjuna Rao

Assistant Professor

Dept. of ECE



Department of Electronics and Communication Engineering

Chaitanya Bharathi Institute of Technology

Hyderabad – 500075

Year 2019-20

A Project Report on
DEVELOPMENT OF WIRELESS AIR MOUSE

in the partial fulfilment of the requirement for the award of the degree of

Bachelor of Engineering
in
Electronics and Communication Engineering

Submitted by

K. Aditya (160116735143)

M. Deekshith (160116735148)

M. Sai Vikas (160116735169)

Under the Supervision of

G. Mallikharjuna Rao

Assistant Professor

Dept. of ECE



Department of Electronics and Communication Engineering

Chaitanya Bharathi Institute of Technology

Hyderabad – 500075

Year 2019-20

Declaration

We do declare that the project work “**Development of Wireless Air Mouse**” submitted in the department of Electronics and Communication Engineering (ECE), Chaitanya Bharathi Institute of Technology, Hyderabad in fulfilment of degree for the award of **Bachelor of Engineering** is a bonafide work done by us, which was carried under the supervision of “**G. Mallikharjuna Rao**”.

Also, we declare that the matter embedded in this report has not been submitted by us in full or partial thereof for the award of any degree/diploma of any other institution or University previously.

Station: Hyderabad

Signature of the candidates

Date:



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(AUTONOMOUS)

HYDERABAD-500075

C E R T I F I C A T E

This is to certify that the Project work entitled “*Development of Wireless Air Mouse*” is a bonafide work carried out by **K. Aditya (160116735143)**, **M. Deekshith (16016735148)**, **M. Sai Vikas (160116735169)** in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Electronics and Communication Engineering, Osmania University, Hyderabad during the academic year 2019-20. The results embodied in this report have not been submitted to any other University or Institution for the award of any diploma or degree.

Supervisor
G. Mallikharjuna Rao
Assistant Professor

Dr. D. Krishna Reddy
Professor and Head
Department of ECE

Acknowledgements

At the very outset of this report, we would like to thank with utmost gratitude and sincerity to all the people who have helped us in this effort. Without their guidance, support and encouragement we wouldn't have made through this project.

We are gratified and beholden to our guide **G. Mallikharjuna Rao**, Assistant professor, Dept. of ECE-CBIT, for his constant guidance and support throughout our work, attending to our needs and helping us in finding solutions to various problems that we encountered.

We sincerely thank our project coordinator **Dr. K. Suman**, Assistant Professor Dept. of ECE-CBIT, for various suggestions which has benefitted our work in various ways.

We express our profound gratitude to **Dr. D. Krishna Reddy**, Head of the dept.ECE-CBIT, for his valuable guidance, enormous support and providing with required facilities in carrying out our project and in preparing this document.

We sincerely thank our principal for his encouragement, guidance and support.

Abstract

With the day to day advancements in technology, the interaction between the human and the digital world is diminishing. Lot of improvement has taken place in the mobile technology from button keypads to touch screens. However, the current PCs still need a pad to operate its mouse and are wired most of the time. The idea is to develop a wireless mouse which works on the hand movement of the user without the need of a pad making an effortless interaction between the human and the computer. The implementation is done using a sensor named gyroscope to sense the hand movements. Gyroscope is a motion sensor which sense changes in rotation in any of the three axes. In here the gyroscope will be at the user's side, attached to the hand to sense the rotation movement and gives the output to a micro-controller to process it. Necessary modifications are done by the microcontroller to these values and are transmitted through a RF module to the PC. At the receiving end a mouse control program which contains functions to control the mouse reads these values and performs the necessary action.

Key words: RF Module, MPU-6050, Arduino Uno, Arduino Leonardo and gyroscope sensor.

	Contents	Page No
Abstract		i
Contents		ii
List of Figures		iv
List of Tables		v
Abbreviations		vi
1 Introduction		
1.1	Introduction	1
1.2	Aim of the thesis	2
1.3	Motivation of the thesis	2
1.4	Literature survey	3
1.5	Technical approach	4
1.6	Application of the thesis	5
1.7	Organization of the thesis	5
2 System Architecture		
2.1	Wired Mouse	6
2.2	Wireless Mouse	7
2.3	Wireless Air Mouse	8
2.3.1	Specifications of Wireless Air Mouse	8
2.4	Hardware Requirements	9
2.4.1	Arduino Uno	9
2.4.2	Arduino Leonardo	12
2.4.3	MPU-6050	14
2.4.4	RF-Module	21
2.4.3	Power supply	27
2.4.6	Switches	27
2.5	Software Requirements	28
2.5.1	Arduino IDE	28
2.5.2	Wire.h Library	29

2.5.3	VirtualWire.h Library	33
2.5.4	Keyboard.h Library	34
2.5.5	Mouse.h Library	35
2.6	Conclusion	37
3	Project Implementation	
3.1	Block diagram	38
3.2	Hardware interfacing	39
3.2.1	Transmitter circuit diagram	39
3.2.2	Receiver circuit diagram	40
3.3	Handheld module	41
3.4	Host connected device	42
3.5	Software Implementation	42
3.5.1	Transmitter Flow chart	43
3.5.2	Receiver flow chart	46
3.6	Conclusion	47
4	Results and Discussions	48
4.1	Mouse movement	48
4.2	Right Click	50
4.3	Left Click	52
4.4	Right Arrow	54
4.5	Left Arrow	56
4.6	Mouse movement on screen	58
5	Conclusion and Future scope	59
5.1	Conclusion	59
5.2	Future Scope	60
	References	61
	Appendix A Code of the thesis	62

List of Figures

Figure. No	Description	Page. No
2.1	Wired mouse	6
2.2	Wireless mouse	7
2.3	Atmega328P	10
2.4	Arduino Uno Pin description	12
2.5	Pin diagram of Arduino Leonardo	12
2.6	Pin diagram of Atmega32u4	13
2.7	MPU-6050 block diagram	15
2.8	Piezoelectric accelerometer	16
2.9	Capacitive accelerometer	16
2.10	Piezoelectric gyroscope structure	17
2.11	MPU-6050	19
2.12	RF Transmitter and RF Receiver	21
2.13	ASK modulation	22
2.14	OOK modulation	22
2.15	FSK modulation	22
2.16	ASK433 RF transmitter and receiver	23
2.17	Timing diagram of Manchester encoding	23
2.18	Timing diagram of decoding	24
2.19	Implementation of ASK	25
2.20	Waveforms of ASK modulation	25
2.21	7805 Voltage regulator	27
2.22	Connections of a switch to ATMEGA328P-PU	28
3.1	Block diagram	38
3.2	Transmitter circuit diagram	39
3.3	Receiver circuit diagram	41
3.4	Transmitter flow chart	45
3.5	Receiver Flow chart	47
4.1	Mouse movement transmitter result	48
4.2	Mouse movement receiver result	49
4.3	Right click transmitter result	50
4.4	Right click receiver result	51
4.5	Left click transmitter result	52
4.6	Left click receiver result	53
4.7	Right arrow transmitter result	54
4.8	Right arrow receiver result	55
4.9	Left arrow transmitter result	56
4.10	Left arrow receiver result	57
4.11	Mouse movement on screen	58

List of Tables

Table. No	Description	Page. No
2.1	Specifications	9
2.2	Hexadecimal codes for the sensors	18
2.3	Truth table of EX-OR	24
2.4	Pin description of RF transmitter	26
2.5	Pin description of RF receiver	26
3.1	Transmitter circuit connections	40
3.2	Receiver circuit connection	41

Abbreviations

S. No	Abbreviation	Description
1	MPU	Motion Processing unit
2	A/D	Analog to digital
3	LCD	Liquid crystal display
4	PC	Personal computer
5	RF	Radio Frequency
6	PCB	Printed circuit board
7	MEMS	Micro-Electro-Mechanical System
8	IC	Integrated circuit
9	I2C	Inter Integrated circuit
10	USB	Universal serial bus
11	TTL	Transistor-Transistor Logic
12	I/O	Input/output
14	PWM	Pulse width modulation
15	IDE	Integrated development environment
16	SRAM	Static Random-access memory
17	UART	Universal asynchronous receiver/transmitter
18	XDA	External data
19	XCL	External clock
20	INT	Interrupt
21	ISM	Industrial, scientific and medical
22	ASK	Amplitude shift keying
23	OOK	On-off keying
24	FSK	Frequency shift keying
25	MOSI	Master output slave input
26	MISO	Master input slave output
27	SCK	Serial clock
28	SDA	Serial data
30	DIP	Dual in-line package
32	CPU	Central processing unit
36	DMP	Digital motion Processor

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

The usage of wireless devices is increasing day by day. Even the conventional mouse which is usually with a wire is replaced with a wireless mouse which control is confined to a limited range from the target device and should be surface mounted. Also during presentations, it will be difficult to manage the computer remotely by standing in front of the big screen. There are few devices which are used to the control the presentation slides on the screen but their usage is limited only to change the slides. For a teacher while giving lecture to the students using an LCD projector, it is much difficult to manage the computer from the podium and it seems like the students getting distracted from the lecture.

In order to overcome these issues, a configuration for the mouse is proposed which need not be surface mounted to control the cursor on the screen of a computer. As the proposed device is small in physical dimensions and light in weight, it easily fits in the hand of the operator. The wrist movement of the operator is enough to control the cursor. The advancement of technology in the field of sensors made it possible to design a humanoid for any application. Efforts are being made to reduce the gap between a human and a machine. Infra-red sensors, ultrasonic sensors, accelerometers, gyro meters, gas sensors etc. can be used by the machine to sense the surroundings and make decision like a human would. A gyroscope can be used to sense motion, more particularly rotation in each direction. This property of gyroscope can be exploited in the field of Human Computer interface by using it as a mouse. The sensor would sense the rotation of hand and the mouse pointer will move. The gyroscope senses the change in rotation, in our case the rotation of the hand. The change in the values of the gyro meter are transmitted to the PC, where in the software applications take control and moves the mouse cursor. Looking for the hardware requirements, Arduino, the name now synonymous with microcontroller and sensors seemed the suitable platform. Its main advantage is being easily programmable and has massive online support. To make the mouse wireless, RF Modules from Texas instruments were used as they are cheap compared to other modules like Bluetooth and Zigbee.

This proposed device is even more effective and acts like the conventional mouse. The functionalities of the conventional mouse like left click, right click, and even the keyboard functions like up arrow, down arrow and lot many functions can be added to the device. However, the proposed device should be a low power device. In addition to the above functionalities, an extra feature can be added to the device, with which an LCD can be turned ON/OFF, and thus there is no need of maintaining a separate device (remote) to control the projector.

1.2 AIM OF THE THESIS

The aim of the project is to design an Air Mouse which does not require a surface and works on the hand motion. To achieve this aim, the following objectives are fulfilled.

1. To develop a hardware interfacing module that works as a wireless air mouse.
2. To add some keyboard functions to air mouse which can be customized by user.
3. To design our module on a printed circuit board (PCB).

1.3 MOTIVATION OF THE THESIS

With the rapid development of technologies, intelligent and smart information products would be necessities of modern. Nowadays the wireless usage has become very popular by using the sensors. Gesture recognition technology identifies the human movements through mathematical representation. This technology has given the solution for the key topic that how to improve the interaction between machine and human. The touch screen technology is mature, but also the people would like to operate in a space freely. The use of gyroscope is to sense the motion of the hand particularly in any given direction; hence the mouse pointer will move. Different values have been fed into the gyroscope sensor with respect to their gestures or actions to be performed.

The proposed model helps to reduce chances of Carpel Tunnel syndrome and help the handicapped. The main driving force behind implementing some special functions was that we used to attend lectures and PPT's in college and the faculty while giving the lecture through projector had to physically go near the laptop to change the slide, that's when the idea stuck us that what if we can perform the same

functions wirelessly saving time and increasing efficiency. The reason for choosing the project is also that it focusses on the evolving side of technology which is Gesture Recognition which we found quite interesting and challenging and because the fabrication of the model is not very costly.

1.4 LITERATURE SURVEY

Optical mouse(wired) have a limited range of control which can be used within the length of the connecting cable which works on a surface. To have user friendly interface between human being and the computer, optical mouse can be replaced with a new device which should work without the requirement of a surface for mounting it and should be a wireless device. These features can be achieved using the suitable sensors detect the gestures of the human hand. Few MEMS (Micro Electro-Mechanical System) sensors like accelerometer sensor or the gyroscope sensors can be used to detect the human gesture based on the movement of the hand. A lot of research is going on to make a mouse which doesn't need any mounting surface.

An accelerometer sensor-based air mouse has been proposed which detects the hand movement based on the acceleration forces, but the accelerometer sensor can detect only the tilting of the device but not resembles the exact hand movement. Further to implement this wirelessly, it was facilitated with Wi-Fi modules, but, with those modules, the design was complex and even the usage of Wi-Fi modules needs an uninterrupted Wi-Fi connectivity, since without the Wi-Fi network and required internet services, the device is useless. Usage of these modules implies that the device depends on the household power supply and in its absence, there will be no Wi-Fi connectivity which leads to the disconnection between the transmitter and receiver circuits. To overcome these challenges, lot many changes should be made, and new sensors should be used. Instead of using the accelerometer sensor which detects the tilting of the device gyroscope sensor can be used which detects the rotation of the device. As the gyroscope can represent a 3-D object, any of the two axes' can be used to define the X and Y co-ordinates of the mouse. Usage of the tactile switches are better than the usage of the third axis of the sensor to perform click operations because there is a possibility of that feature getting performed unknowingly when the device falls. Also, instead of using the Wi-Fi modules they can be replaced with the RF modules which are cheap and give a better range for operation. As the usage of the device doesn't

need any household power supply, it is free from the power cuts. The interconnectivity between the transmitter and the receiver can be implemented using a pair of Bluetooth devices.

In the existing wireless air mouse device, the transmitter uses an Arduino UNO and the receiver circuit uses an Arduino Leonardo boards. Using the entire Arduino UNO board at the transmitter will be difficult during implementation. So, it can be further improvised by using ATMEGA328P-PU microcontroller instead of using the entire board and the advantages with standalone IC over Arduino UNO is it consumes very low power which is useful in improving the life span of the cycle. As the receiver circuit stays at the laptop and doesn't need any movement, modifications of the circuitry can be ignored. Finally, usage of gyroscope sensor, RF modules and ATMEGA328P-PU can make the device even simpler and effective.

1.5 TECHNICAL APPROACH

Firstly, for our project to work we need a sensor that can detect the rotational movement in all directions. For this we have selected MPU-6050 module which consists of accelerometer sensor, gyroscope sensor and temperature sensor. But we use gyroscope sensor as it can detect the rotational movement in all directions. The data that we get from the MPU sensor should be processed. So, for that we have chosen Arduino Uno as our processing unit as it supports the I2C communication. After processing the data, we need to transmit the data to the receiver which can be done in two ways either wired or wireless. We have chosen wireless medium because of its advantages over wired medium.

For the purpose of transmission and reception we need to choose a transmitter and receiver. For this we have various techniques and technologies such as Bluetooth, Zigbee, RF module etc. But we have chosen RF Module as it doesn't need line of sight between the transmitter and receiver. Finally, there should be a processing unit at the receiver also so we choose Arduino Leonardo as it can establish micro USB to USB communication between PC and itself. By using the RF transmitter and receiver, the problem is that we cannot concurrently transmit multiple values as there is only single channel available so, we are processing all the values at a time and concatenating all the values into a string and transmitting through a single channel.

We found a library called Wire.h where it provides the functions for the transmission of data from MPU-6050 to Arduino Uno through I2C interface communication protocol. For wireless transmission between RF transmitter and RF receiver we used a library called VirtualWire.h.

1.6 APPLICATION OF THESIS

- Mouse cursor movement can be achieved remotely.
- Various controls like closing a window, moving the slides on a computer can be achieved.
- We can even copy some data and paste it to another location using wireless air mouse.
- It is portable - wireless air mouse is hand wearable unlike optical mouse.
- It is compact - Optical mouse requires a surface to work upon. Wireless air mouse is compact because it doesn't need any surface for interaction. Hand movement is enough to control the cursor.
- As the device is wireless, the range it can cover is more than that of wired mouse.
- It reduces the risk of Carpel Tunnel syndrome.
- We added some special functions (Left and Right click, Left and Right arrow) which give an added edge to the device as it increases its functionality and practicality.
- Wireless Air mouse attains an easy and efficient way of Human Computer Interaction.

1.7 ORGANIZATION OF THE THESIS

In this thesis, Chapter 2 gives an insight into the architecture of wireless air mouse and the components we used for developing the mouse. Chapter 3 presents the working of the mouse along with its code and flowchart. This chapter also shows the connections in the transmitter and receiver side. Chapter 4 summarizes the report and tells what happens when a button is pressed in mouse with a detailed explanation that will be used to fulfil the objectives of the project with appropriate results. Finally, Chapter 5 concludes the thesis work and with a brief note on future scope of the work.

CHAPTER-2

SYSTEM ARCHITECTURE

2.1 WIRED MOUSE

A wired mouse is a hand-held pointing device that detects two-dimensional motion relative to a surface. This motion is typically translated into the motion of a pointer on a display, which allows a smooth control of the graphical user interface of a computer. To transmit their input, typical cabled mice use a thin electrical cord terminating in a standard connector, such as RS-232C, PS/2, ADB or USB.

The wired mouse has a better response time than the wireless mouse. The wired mouse does not need batteries which means you will not encounter the interruption of the mouse. The most programmable mouse is wired. With programmable buttons, you can set shortcut keys for the game and after a lot of use the flexing can cause the wire to break inside the insulation. Wired mouse being connected to the computer hardware apparently gives it an advantage that translates to better pointer speed, mouse acceleration, control/malfunction speed. the wired USB mouse is cheaper than the wireless USB mouse. They also don't need charging or buying new batteries periodically. In the long run, a one-time investment in a wired mouse beats having to incur extra costs and the inconvenience of owning a wireless mouse.



Fig.2.1: Wired mouse

On the downside, a wired mouse calls first dibs on your USB port. In the process, this takes one port out of commission at any given time. Newer laptops and slim notebooks usually have ever fewer USB ports, to begin with. It can be quite the inconvenience to have to unplug the mouse whenever you need an extra port

2.2 WIRELESS MOUSE

A mouse typically controls the motion of a pointer in two dimensions in a graphical user interface (GUI). The mouse turns movements of the hand backward and forward, left and right into equivalent electronic signals that in turn are used to move the pointer. Cordless mice instead transmit data via infrared radiation (see IrDA) or radio (including Bluetooth), although many such cordless interfaces are themselves connected through the wired serial buses.

You often need to create some space for a mouse pad. Unlike the wireless mouse, you don't have the luxury of placing the mouse pad somewhere out-of-the-way. On top of which, the wire can only stretch so far. All in all, we feel this messes up with the ergonomics of the device. Going wireless with a mouse means there is no cord, no muss, no fuss. Let's face it – the cord is the biggest complaint most people have with a mouse. It takes up space on your desk. It gets tangled. It gets in the way of mouse movement.



Fig.2.2: Wireless Mouse

The use of a wireless mouse vastly improves mouse functionality. Using a wireless mouse allows you to work from where you are most comfortable. You don't have to sit right next to your computer to use your mouse. This makes the use of a wireless mouse ideal for individuals who do use large screens for giving presentations. You can sit anywhere in a room and control your computer screen. A wireless mouse is far easier to pack up and take with you to different work locations than a wired mouse. The cord of a wired mouse is just one more thing that you must pack when you travel. Plus, a wired mouse cord often gets tangled in a computer bag and leads to an unnecessary frustration.

2.3 WIRELESS AIR MOUSE

Wireless Air Mouse is also called gesture-controlled mouse and it works based on hand gesture. In this project a gyroscope is used for measuring the rotation of hand in X and Y direction and moves the cursor according the tilt. Wireless air mouse and conventional wired mouse does the same functionalities, but the way of working differs which make wireless air mouse more flexible than the conventional mouse. The moment of the mouse defines the cursor moment in a conventional mouse whereas in the wireless air mouse the rotation moment of handheld module defines the cursor moment.

In the backend both have the same functionalities, but wireless air mouse provides the advantages such as it doesn't require any surface and as it is wireless the range can be extended. As it is wireless, we can seamlessly integrate several functionalities that the user requires on handheld module of the wireless air mouse.

2.3.1 SPECIFICATIONS OF WIRELESS AIR MOUSE

Wireless Air Mouse uses gyroscope sensor which is integrated in the MPU-6050 sensor. The modulation used during transmission is amplitude shift keying modulation (ASK) and the working frequency is 433MHz. For the transmission purpose we have use RF module and based on user requirement we can add different buttons in our project we are using four buttons one each for right click, left click, right arrow and left arrow. Table below shows the specifications of wireless air mouse.

Table.2.1: Specifications

Features	Specifications
Sensor	MPU - 6050 sensor
Modulation	ASK
Working Frequency	433 MHz
Transmission technology	RF Module
Buttons	4 buttons <ul style="list-style-type: none">• Right click• Left click• Right arrow• Left arrow

2.4 HARDWARE REQUIREMENTS

2.4.1 ARDUINO UNO

2.4.1.1 INTRODUCTION

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts.

The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

2.4.1.2 ATMEGA328P

The ATmega328P provides the following features like 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 1K bytes EEPROM, 2K

bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. It is a DIP (Dual In-line package) IC.



Fig.2.3: Atmega328P

2.4.1.3 PIN DESCRIPTION

GENERAL PIN FUNCTIONS

LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.

VIN: The input voltage to the Arduino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator and can damage the board.

3V3: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Reset: Typically used to add a reset button to shields that block the one on the board.

SPECIAL PIN FUNCTIONS

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labelled A0 through A5; each provides 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the `analogReference()` function.

In addition, some pins have specialized functions:

Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

External interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

PWM (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the `analogWrite()` function.

SPI (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.

TWI (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.

AREF (analog reference): Reference voltage for the analog inputs.

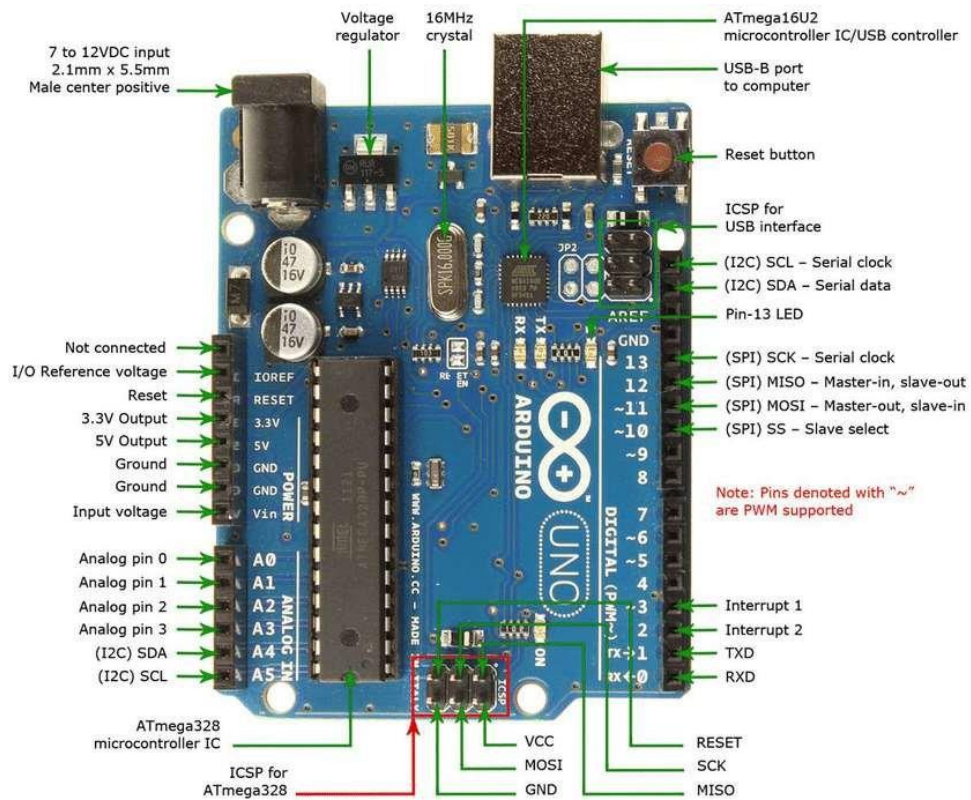


Fig.2.4: Arduino Uno Pin description

2.4.2 ARDUINO LEONARDO

2.4.2.1 INTRODUCTION

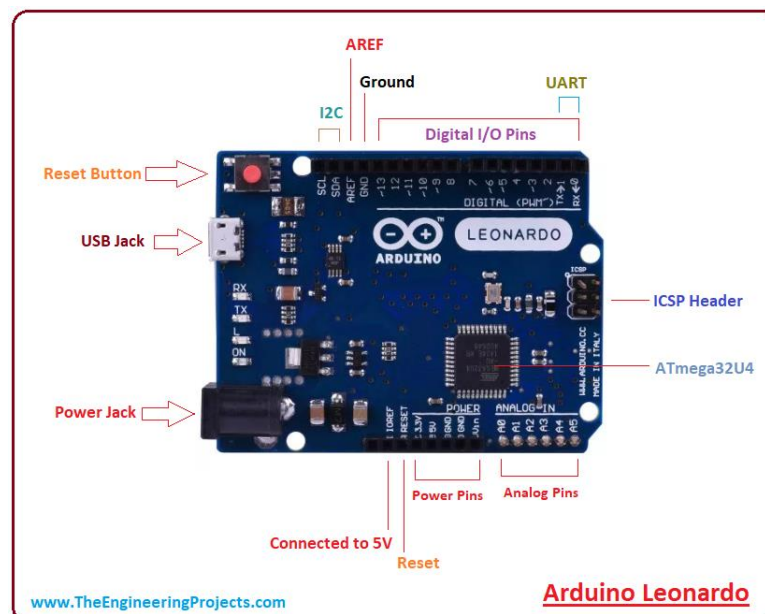


Fig.2.5: Pin diagram of Arduino Leonardo

The Arduino Leonardo is a microcontroller board based on the ATmega32u4. It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Leonardo differs from all preceding boards in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Leonardo to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port.

2.4.2.2 ATMEGA32U4

ATmega32u4 has built in USB communication, and hence there is no need for a secondary processor. This makes the Leonardo to appear as a mouse and keyboard to a computer to which it is connected.

It has 32KB self-programming flash memory, 2.5KB SRAM, 1KB EEPROM, USB 2.0 full-speed/low speed device, 12-channel 10-bit A/D-converter, and JTAG interface for on-chip-debug. The device achieves up to 16 MIPS throughput at 16 MHz 2.7 - 5.5 Volt operation.

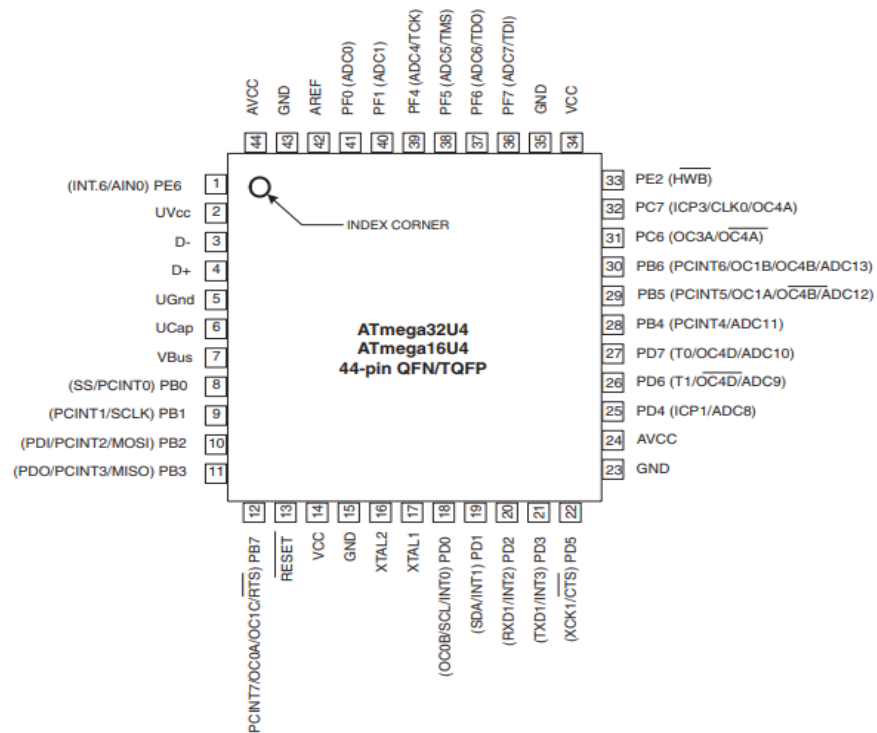


Fig.2.6: Pin diagram of atmega32u4

2.4.2.3 PIN DESCRIPTION

VCC: Supply voltage

GND: Ground

Port B (PB7.....PB0), Port C (PC7.....PC0), Port D (PD7.....PD0), Port E (PE7.....PE0), Port F (PF7.....PF0): 8 bit bi-directional I/O ports with internal pull up resistors.

D-: USB full speed or low speed negative data upstream port.

D+: USB full speed or low speed positive data upstream port.

UGND: USB pads ground.

UVCC: USB pads input power supply.

UCAP: USB pads output power supply and should be connected to an external capacitor.

VBUS: USB VBUS monitor input.

RESET: It is an active low pin. A low level on this pin will reset the chip.

XTAL1: Input to the inverting oscillator amplifier and internal clock operating circuit.

XTAL2: Output from the inverting oscillating amplifier.

AVCC: Input voltage for all A/D converter channels.

AREF: This is the analog reference pin (input) for the A/D Converter.

2.4.3 MPU-6050

2.4.3.1 INTRODUCTION

MPU-6050 is a Motion Processing Unit which is used to detect all types of motion in all directions. It can also include the detection of the acceleration. It consists of 3 sensors integrated within it which are namely 3-axis accelerometer sensor, 3-axis gyroscope sensor and a temperature sensor. All the above sensors can be configured and used either individually or can be used all together.

The following diagram represents the MPU-6050 with all the blocks interconnected such that all the sensors can work accurately.

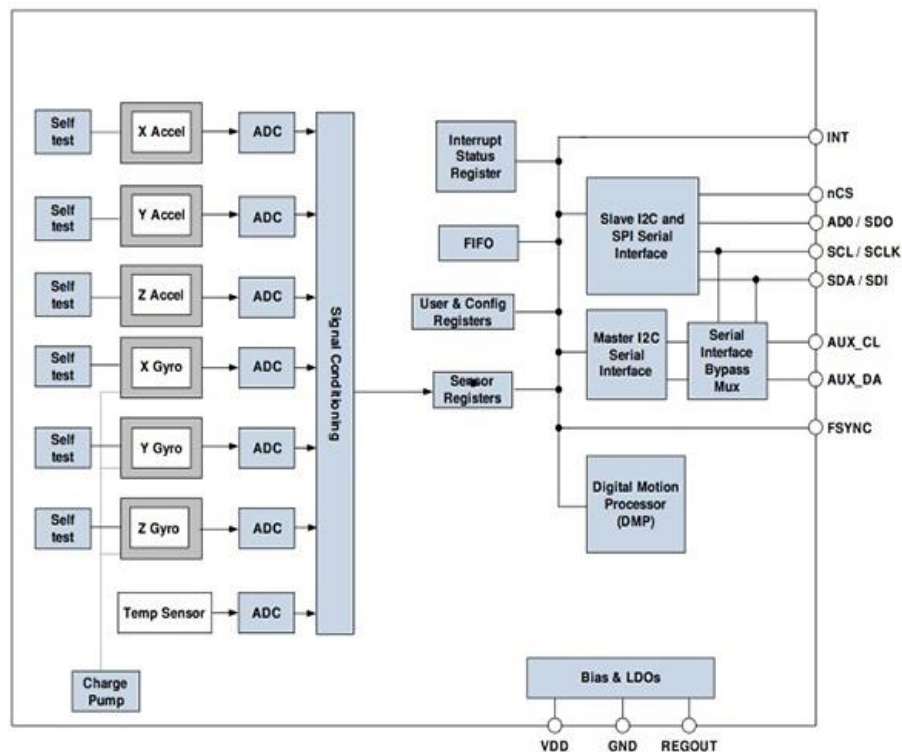


Fig.2.7: MPU-6050 block diagram

2.4.3.2 SELF-TEST

Self-test is generally used to make the debugging process easy and efficient because it lets the user know whether the available device is working as expected. Initially the device is fed with all the possible inputs and the respective results are compared with the reference result values. If the resultant output matches with reference value, then the device is said to be working well.

2.4.3.3 SENSORS

The device consists of sensors which measures the respective values with respect to the movement that was made. As MPU-6050 is a 3-axis accelerometer and gyroscope sensor, its axis information is measured with an individual sensor that it means it for every axis a sensor is used.

2.4.3.4 ACCELEROMETER SENSOR

It can record the tilting movement of the device with respect to the Earth's gravitational force. It is designed to manipulate the range of the device based on the

type of application that we use. All these manipulations can be done using the predefined register values which can be initiated while programming the device. The available user programmable full-scale ranges with this device are $\pm 2g$, $\pm 4g$, $\pm 8g$ or $\pm 16g$.

The accelerometer sensor can be constructed in two ways, one is using the capacitive sensing elements and the other is using the piezoelectric effect. It is arranged in such a way that one of the plates of the capacitor is fixed and the other is movable. So, when some external force is applied to the setup, its movable plate starts to vary its position which leads to change in the distance between two plates which finally changes the capacitance value where the resultant capacitance value defines the amount of tilt made externally. Whereas with piezoelectric materials, a small metal sphere is left free in the closed surface (which is similar to a box) which is made of piezoelectric materials so when the device is tilted it makes the sphere inside move proportionally which touches the surface or surfaces at a time. This causes the material to vibrate and the small currents are generated which is further utilized to measure the amount of acceleration. Individual sensors are used in order to measure acceleration in a particular axis as shown in the block diagram.

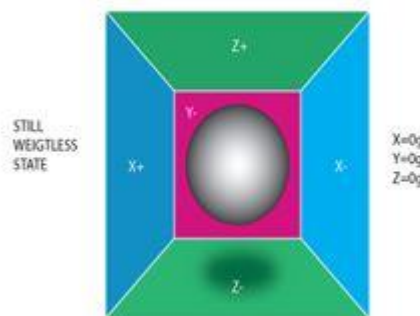


Fig.2.8: Piezoelectric accelerometer

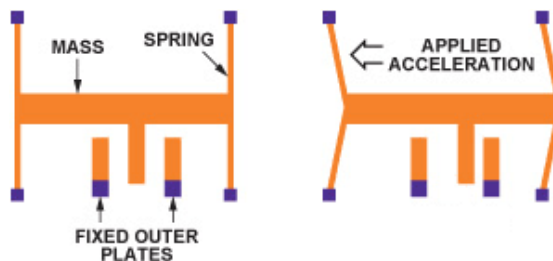


Fig.2.9: Capacitive accelerometer

2.4.3.5 GYROSCOPE SENSOR

Unlike accelerometer it senses when the device is rotated that implies it can able to detect the rotation in particular direction or with respect to a particular axis precisely. The Gyroscope sensor is made based on the principle of piezoelectric effect. When the device is rotated current will be generated. Its construction is like the body structure of an insect. A metal thread like structure held in between two piezoelectric materials where the centre of the thread is fixed with the piezoelectric materials and the remaining part of the thread is left free. When the external force is applied to the device the threads starts to flap to and for which causes the piezoelectric materials vibrate which further generates the respective current values later it is conditioned and amplified. For the gyroscope sensor also the individual axis information is measured using an individual sensor. The full-scale range of the gyroscope sensor is also user programmable which has capable of detecting fast and slow motions whose ranges are ± 250 , ± 500 , ± 1000 , and $\pm 2000^\circ/\text{sec}$. By initiating appropriate register values, the MPU-6050 can detect all available ranges.

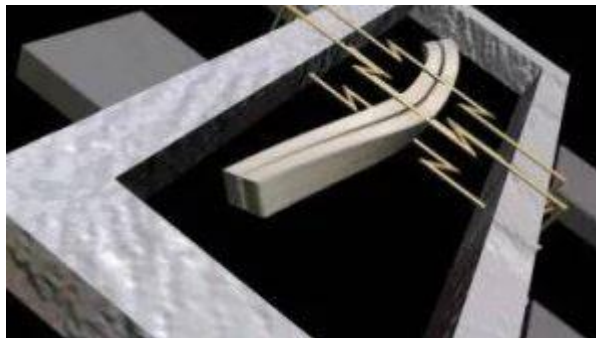


Fig.2.10: Piezoelectric gyroscope structure

2.4.3.6 TEMPERATURE SENSOR

Apart from the accelerometer sensor and gyroscope sensor MPU-6050 also consists of a temperature sensor which gives the temperature value accurately.

All the available sensors can be either enabled or disabled by programming the device using appropriate software and an interface.

2.4.3.7 ANALOG TO DIGITAL CONVERTER

As all the obtained values are available in analog form it is difficult to use in the appropriate applications further. So, there is a need to convert the available analog data into digital data this process can be done using an ADC (Analog to Digital Converter)

unit. All the individual is further connected with the ADC so that the expected data can be achieved precisely.

2.4.3.8 SIGNAL CONDITIONING UNIT

As the available data's signal strength is too small to process the available signal should be modified in order to achieve the required signal strength so the available data is fed to the signal conditioning unit. The signal conditioning unit generally consists of filters and amplifiers once if the data is available it is filtered to get rid of the noise and further it is amplified to maintain the required signal strength. Now the data is ready to be processed and to be stored in the registers.

2.4.3.9 SENSOR REGISTERS

Each sensor has a register in order to store the data temporarily or to use the data while testing its operation by itself. The naming for each register is unique and is assigned in the form of hexadecimal number system. In order to make the user interface easy it was designed with the help of frame structures. So, by manipulating the respective values of the frames the functionalities of the device can be varied or the sensors can be enabled or disabled using an appropriate code.

The below table gives the example view of hexadecimal formats of the various available sensors.

Table.2.2: Hexadecimal codes for the sensors

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

2.4.3.10 DIGITAL MOTION PROCESSOR (DMP)

Getting the individual information alone is not satisfactory it should be further computed, analysed and the features should be able to manipulated as defined by the user all these options can achieved using a processor but a simple processor which is used in computers may not be helpful therefore, a new processor is designed which can perform all the operations related to the motion sensors. The DMP has access to all the registers so that the user can program it and the access the sensors as needed.

2.4.3.11 I²C INTERFACE

I²C stands for Inter IC Communication which is generally used for communicating between the two Integrated circuits. A separate protocol is designed and is available in order maintain the communication in between the ICs. It helpful when a MEMS (Micro Electro-Mechanical System) sensor is used and needed to send its obtained values for further data processing to use those values in any application. Without this I²C protocol it is much difficult for the integrated circuits to communicate with the devices. Especially when it is used in the Arduino IDE it is must to include the corresponding libraries so that these devices can be used in the user designed application. These libraries in the Arduino IDE functions as a protocol in order send or receive the data between the integrated circuits. This interface can also be used as master and slave devices when two similar devices are used, and we need synchronization between the devices.

2.4.3.12 PIN DESCRIPTION



Fig.2.11: MPU-6050

MPU-6050 is an 8-pin device which consists of Vcc, Ground, SCL, SDA, XD0, XD1, AD0, INT. The following diagram shows the breakout board structure of the MPU-6050.

2.4.3.12.1 VCC AND GROUND

The device can be generally operated at a standard voltage of either 3.3V or 5V care should be taken that the voltages should not exceed the rated specifications else there are high chances of the device getting burnt away.

2.4.3.12.2 SCL AND SDA

SCL stands for Serial Clock Line. As we know that every digital device works with the help of a clock so in order to make the sensor work a serial clock pulse should be connected externally which helps in maintaining the synchronization when the devices need to communicate with each other.

SDA stands for Serial Data Access. This helps in sending the sensed values from the transducer to the target device (generally a microcontroller will be used).

For an Arduino UNO the SCL and SDA pins are the 4th and 5th analog pins whereas for the Arduino Leonardo models the pins separately available on the board. In the ATMEGA328P-PU the SDA and SCL pins are 27th and 28th pins respectively.

2.4.3.12.3 XDA, XCL AND AD0

XDA stands for Auxiliary Data Access.

XCL stands for Auxiliary Clock Line.

If single IC is not enough to measure all the required values accurately and if we need another similar sensor which should be in synchronization with the master device then these pins can be used. If no such application exists for the application these pins can be ignored.

2.4.3.12.4 INT

INT stands for interrupt. It used to interrupt the external device which is connected to it. All the interrupts from this device are stored or processed from the internal register which is named as Interrupt register within the MPU-6050.

2.4.4 RF MODULE

2.4.4.1 DEFINITION

Radio Frequency module is a device used for wireless communication. The communication takes place through the radio signals. The frequency range of radio signals is 20 KHz to 300 GHz. The advantage of RF module is that there is no need to maintain line of sight between the transmitter and the receiver.

RF module constitutes RF transmitter and RF receiver.

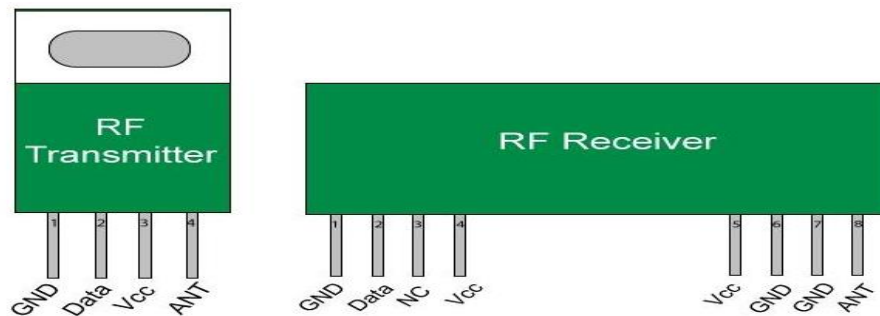


Fig.2.12: RF Transmitter and RF Receiver

2.4.4.1.1 RF TRANSMITTER: Transmitter takes the data from microcontroller and sends it to the receiver wirelessly through modulation. It modulates the radio signals by superimposing the carrier. Modulating the signal increases the strength of the signal so that it can travel to longer distances.

Industrial, Scientific and Medical (ISM) radio bands such as 433.92 MHz, 915 MHz, and 2400 MHz are used as carrier frequencies. For short range devices the unlicensed frequencies such as 315 MHz and 868 MHz can be used.

2.4.4.1.2 RF RECEIVER: The modulated signal is received at the receiving antenna and is demodulated to get the original data.

2.4.4.2 RF SIGNAL MODULATION:

Three types of modulation techniques are generally used in RF communication.

- 1) ASK
- 2) OOK
- 3) FSK

2.4.4.2.1 ASK MODULATION (AMPLITUDE SHIFT KEYING): Variation of amplitude of carrier wave according to the characteristics of the modulating signal.

When the modulating signal is binary 1 then the carrier with higher amplitude is transmitted and when it is binary 0 then the carrier with lower amplitude is transmitted.

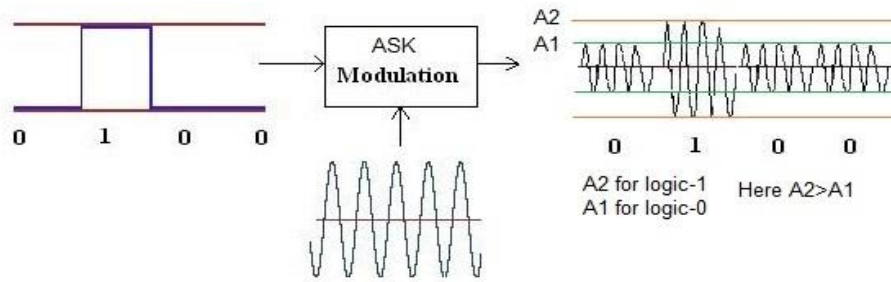


Fig.2.13: ASK modulation

2.4.4.2.2 OOK MODULATION (ON OFF KEYING): It is like ASK but when the modulating signal is binary 0 then no carrier is transmitted.

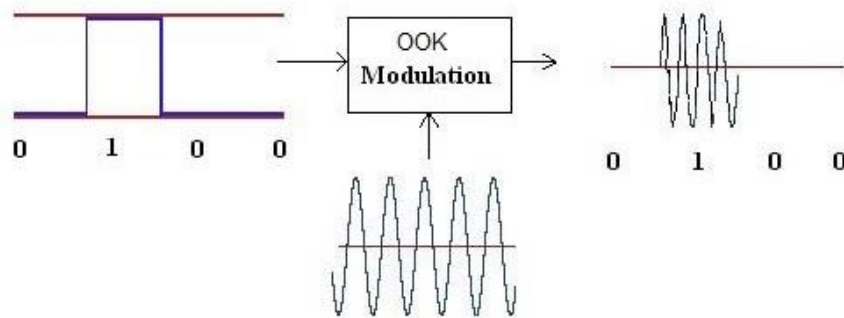


Fig.2.14: OOK modulation

2.4.4.2.3 FSK MODULATION (FREQUENCY SHIFT KEYING): Variation of frequency of carrier wave according to the characteristics of the modulating signal. The carrier wave of higher frequency and lower frequency is transmitted during binary 1 and binary 0 of modulating signal respectively.

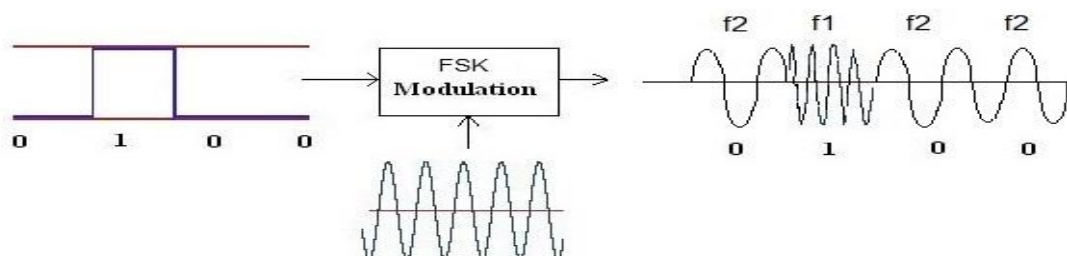


Fig.2.15: FSK modulation

2.4.4.3 ADVANTAGES OF RF OVER IR:

- Radio frequency signals travel longer distances than IR

- RF communication do require line of sight i.e., can be transmitted even when there is an obstacle whereas Infrared signals should have line of sight.
- No interference problem in RF.

2.4.4.4 APPLICATIONS:

- Wireless security systems
- Remote controls
- Automation systems
- Car alarm systems
- Sensor reporting

2.4.4.5 ASK433 RF MODULE:



Fig.2.16: ASK433 RF transmitter and receiver

It uses ASK modulation technique with carrier frequency equal to 433 MHz.

2.4.4.6 MANCHESTER ENCODING:

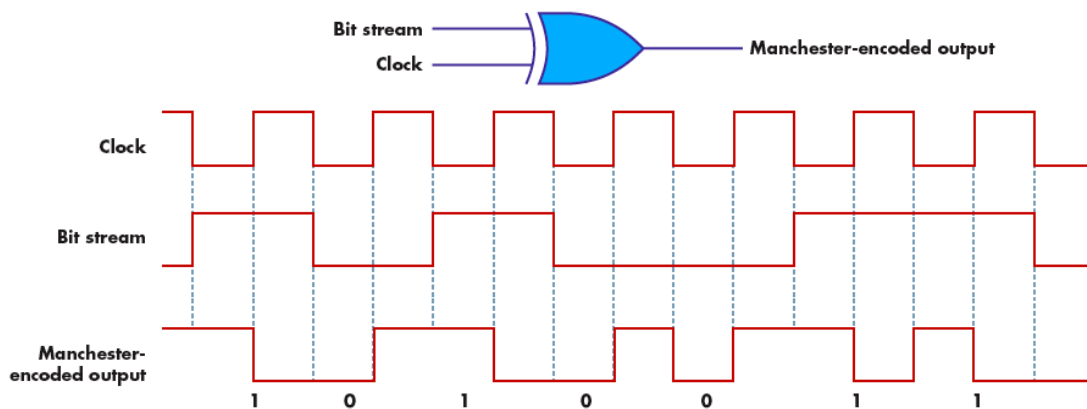


Fig.2.17: Timing diagram of Manchester encoding

Manchester encoding is one of the line encoding techniques which converts the digital data into digital signal.

When the input bit is 1, then it is encoded as a transition from 1 to 0.

When the input bit is 0, then it is encoded as a transition from 0 to 1.

2.4.4.6.1 IMPLEMENTATION:

As shown in the fig, the bit stream and clock are given as inputs to the XOR gate. If two inputs have same value, then the output will be 0 otherwise 1.

Table.2.3: Truth table of EX-OR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

2.4.4.6.2 DECODING:

By performing an Exclusive-OR operation of the Manchester encoded signal with logic 1 at a bit boundary sample points, the bit stream is decoded.

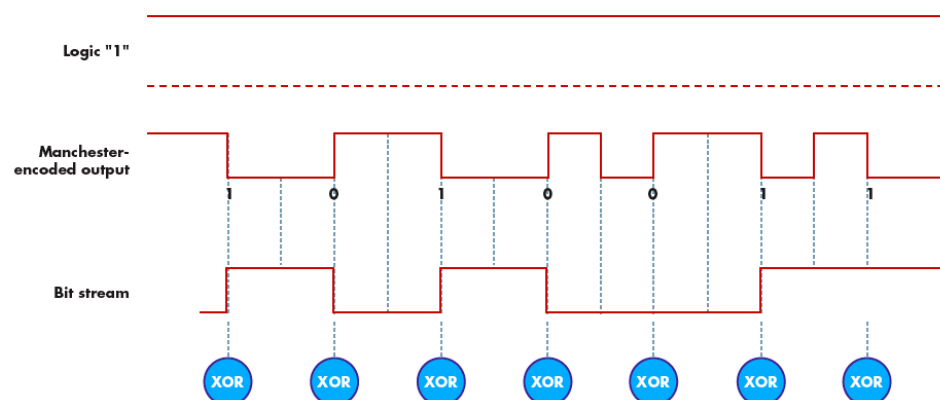


Fig.2.18: Timing diagram of decoding

2.4.4.7 AMPLITUDE SHIFT KEYING:

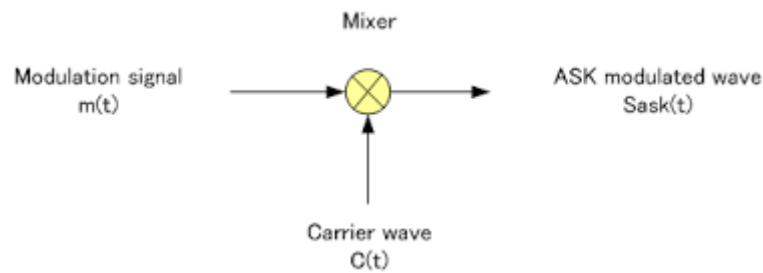


Fig.2.19: Implementation of ASK

When the modulating signal is binary 1, then the carrier wave will be the output.

When the digital signal is binary 0, there is no output

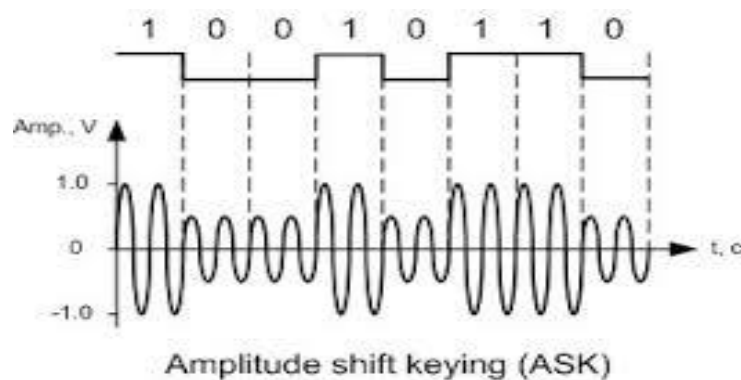


Fig.2.20: Waveforms of ASK modulation

$$S_{ask}(t) = m(t) * C(t)$$

$$\text{When } m(t) = 1, S_{ask}(t) = 1 * C(t) = C(t)$$

$$\text{When } m(t) = 0, S_{ask}(t) = 0 * C(t) = 0$$

2.4.4.8 SPECIFICATIONS:

2.4.4.8.1 RF TRANSMITTER:

- Frequency Range: 433.92 MHz
- Supply voltage: 3-12 V
- Output power: 4-16 dBm

2.4.4.8.2 RF RECEIVER:

- Receiver Frequency: 433.92MHz
- Supply current: 3.5mA
- Low power consumption
- Operating voltage: 5 V

2.4.4.9 PIN DESCRIPTION:

2.4.4.9.1 RF TRANSMITTER:

Rf transmitter present in the Rf module has 4 pins. They are Ground, data, Vcc, ANT. The functions of those pins are shown in the following table.

Table.2.4: Pin description of RF transmitter

Pin No	Function	Name
1	Ground (0V)	Ground
2	Serial data input pin	Data
3	Supply voltage; 5V	Vcc
4	Antenna output pin	ANT

2.4.4.9.2 RF RECEIVER:

Rf Receiver present in the Rf module has 8 pins. They are three Ground pins, one data pin, two Vcc pins and an ANT pin. The functions of those pins are shown in the following table.

Table.2.5: Pin description of RF receiver

Pin No	Function	Name
1	Ground (0V)	Ground
2	Serial data output pin	Data
3	Linear output pin; not connected	NC
4	Supply voltage; 5V	Vcc
5	Supply voltage; 5V	Vcc
6	Ground (0V)	Ground
7	Ground (0V)	Ground
8	Antenna input pin	ANT

2.4.5 POWER SUPPLY

2.4.5.1 INTRODUCTION

As many of the devices need some form of energy to work as expected, there is need to design a proper power supply. A battery of 9V is used for the proposed portable circuitry. As the rated specifications and operating voltage range of the Integrated Circuits is standard with 3.3V and 5V. So, there is a need to regulate the 9V and attenuate it down to 5V. In order to achieve the required voltages, a voltage regulator is needed.

2.4.5.2 VOLTAGE REGULATOR

It is a 3-pin device which takes DC voltage as input and gives the regulated voltage as the output. In order to regulate the voltage from 9V to 5V, a 7805 where 78 is the series number of the voltage regulators and 05 defines the regulated outputs that it gives as output. If the input applied to the device is greater than the value of last two digits on the IC, the output is maintained as constant (say 5V for 7805). If the input value is less than the last two digits on the then the output is same as the input. The available voltage regulators are 7805, 7809, 7812.

The following is a 7805 Voltage regulator where the first pin is for the input, second pin for ground and third is for the output.



Fig.2.21: 7805 Voltage regulator

2.4.6 SWITCHES

A switch is a device which is used to either make or break the contact. It can also be used to manually control the system only when it is necessary. Switches play a vital role in making the user interface easy and comfortable. Switches that are used are push button (tactile switches), Momentary switches (Push to ON/OFF switch).

2.4.6.1 TACTILE SWITCH

It is a 4-pin device where 2 pairs of pins are always shorted. When an external force is applied on the button those are 2 pairs of pins get opened and forms a short circuit with the remaining combination switches. When the external force is released from the button the internal connections get back to their default positions.

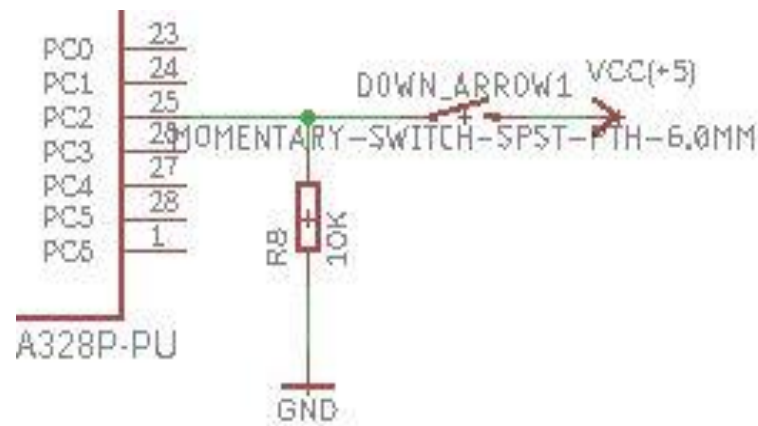


Fig.2.22: Connections of a switch to ATMEGA328P-PU

In order to make a particular pin logically HIGH a switch cannot be used directly because the direct voltage from the source may damage the internal circuitry of the microcontroller. So, resistor should be used which acts as pull down resistor such that when the switch is closed the voltage drop occurs across the resistor that makes the controller's pin HIGH. One of the pins of the switch should be connected to the Vcc and other pin should be connected to one of the pins microcontroller and resistor and the other lead of the resistor should be connected to the ground. This circuit connection can be effectively used to drop the voltage to the microcontroller.

2.5 SOFTWARE REQUIREMENTS

For the development of the project we need Arduino software as we are using Arduino Uno and Arduino Leonardo boards. Also, we require Wire.h, VirtualWire.h, Keyboard.h and Mouse.h libraries for our project.

2.5.1 ARDUINO IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++.

It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub `main()` into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program `avrdude` to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, `avrdude` is used as the uploading tool to flash the user code onto official Arduino boards.

2.5.2 Wire.h LIBRARY

This library allows you to communicate with I2C / TWI devices. On the Arduino boards with the R3 layout (1.0 pinout), the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. The Arduino Due has two I2C / TWI interfaces SDA1 and SCL1 are near to the AREF pin and the additional one is on pins 20 and 21. As a reference the data shows where TWI pins are located on various Arduino boards, in Uno A4 (SDA), A5 (SCL) and in Leonardo 2 (SDA), 3 (SCL).

As of Arduino 1.0, the library inherits from the Stream functions, making it consistent with other read/write libraries. Because of this, `send()` and `receive()` have been replaced with `read()` and `write()`.

To use this library, we should use `#include <Wire.h>` at the beginning of the program

Functions used in Wire.h:

1. Wire.begin()

`Wire.begin(address)`

- Description

Initiate the Wire library and join the I2C bus as a master or slave. This should normally be called only once.

- Parameters

address: the 7-bit slave address (optional); if not specified, join the bus as a master.

- Returns

None

2. Wire.requestFrom()

- Description

Used by the master to request bytes from a slave device. The bytes may then be retrieved with the `available()` and `read()` functions. As of Arduino 1.0.1, `requestFrom()` accepts a boolean argument changing its behaviour⁶ for compatibility with certain I2C devices.

If true, `requestFrom()` sends a stop message after the request, releasing the I2C bus. If false, `requestFrom()` sends a restart message after the request. The bus will not be released, which prevents another master device from requesting between messages. This allows one master device to send multiple requests while in control.

The default value is true.

- Syntax

`Wire.requestFrom(address, quantity)`

`Wire.requestFrom(address, quantity, stop)`

- Parameters

address: the 7-bit address of the device to request bytes from

quantity: the number of bytes to request

stop : boolean. true will send a stop message after the request, releasing the bus.

false will continually send a restart after the request, keeping the connection active.

Returns

byte : the number of bytes returned from the slave device

3. Wire.beginTransmission(address)

- Description

Begin a transmission to the I2C slave device with the given address. Subsequently, queue bytes for transmission with the `write()` function and transmit them by calling `endTransmission()`.

- Parameters

address: the 7-bit address of the device to transmit to

- Returns

None

4. Wire.endTransmission()

- Description

Ends a transmission to a slave device that was begun by beginTransmission() and transmits the bytes that were queued by write(). As of Arduino 1.0.1, endTransmission() accepts a boolean argument changing its behaviour for compatibility with certain I2C devices.

If true, endTransmission() sends a stop message after transmission, releasing the I2C bus.

If false, endTransmission() sends a restart message after transmission. The bus will not be released, which prevents another master device from transmitting between messages. This allows one master device to send multiple transmissions while in control.

The default value is true.

- Syntax

Wire.endTransmission()

Wire.endTransmission(stop)

- Parameters

stop : boolean. true will send a stop message, releasing the bus after transmission.

false will send a restart, keeping the connection active.

- Returns

byte, which indicates the status of the transmission:

0:success

1:data too long to fit in transmit buffer

2:received NACK on transmit of address

3:received NACK on transmit of data

4:other error

5. write()

- Description

Writes data from a slave device in response to a request from a master, or queues bytes for transmission from a master to slave device (in-between calls to beginTransmission() and endTransmission()).

- Syntax

Wire.write(value)

Wire.write(string)

Wire.write(data, length)

- Parameters

value: a value to send as a single byte

string: a string to send as a series of bytes

data: an array of data to send as bytes

length: the number of bytes to transmit

- Returns

byte: write() will return the number of bytes written, though reading that number is optional

6. Wire.available()

- Description

Returns the number of bytes available for retrieval with read(). This should be called on a master device after a call to requestFrom() or on a slave inside the onReceive() handler.

available() inherits from the Stream utility class.

- Parameters

None

- Returns

The number of bytes available for reading.

7. read()

- Description

Reads a byte that was transmitted from a slave device to a master after a call to requestFrom() or was transmitted from a master to a slave. read() inherits from the Stream utility class.

- Syntax

Wire.read()

- Parameters

none

- Returns

The next byte received

2.5.3 VirtualWire.h LIBRARY

VirtualWire is a communications library for Arduino that allows multiple Arduino's to communicate using low-cost RF transmitters and receivers. VirtualWire is an Arduino library that provides features to send short messages, without addressing, retransmit or acknowledgment, a bit like UDP over wireless, using ASK (amplitude shift keying). Supports several inexpensive radio transmitters and receivers. All that is required is transmit data, receive data and (for transmitters, optionally) a PTT transmitter enable.

It is intended to be compatible with the RF Monolithic Virtual Wire protocol, but this has not been tested. Does not use the Arduino UART. Messages are sent with a training preamble, message length and checksum. Messages are sent with 4-to-6 bit encoding for good DC balance, and a CRC checksum for message integrity. Why not just use the Arduino UART connected directly to the transmitter/receiver? As discussed in the RFM documentation, ASK receivers require a burst of training pulses to synchronize the transmitter and receiver, and requires good balance between 0s and 1s in the message stream in order to maintain the DC balance of the message. UARTs do not provide these. They work a bit with ASK wireless, but not as well as this code. To use the VirtualWire library, you must have `#include <VirtualWire.h>`

Functions used in VirtualWire.h:

1. vw_setup

- `extern void vw_setup(uint16_t speed);`
- Initialise the VirtualWire software, to operate at speed bits per second. Call this once in your `setup()` after any `vw_set_*` calls. You must call `vw_rx_start()` after this before you will get any messages.

2. vw_send

- `extern uint8_t vw_send(uint8_t* buf, uint8_t len);`
- Send a message with the given length. Returns almost immediately, and the message will be sent at the right timing by interrupts. Returns true if the message was accepted for transmission. Returns false if the message is too long (`>VW_MAX_PAYLOAD`).

3. vw_wait_tx

- `extern void vw_wait_tx();`

- Block and wait until the transmitter is idle

4. vw_rx_start

- extern void vw_rx_start();
- Start the receiver. You must do this before you can receive any messages. When a message is available (good checksum or not), vw_have_message() will return true.

5. vw_get_message

- extern uint8_t vw_get_message(uint8_t* buf, uint8_t* len);
- If a message is available (good checksum or not), copies up to *len octets to buf. Returns true if there was a message and the checksum was good

6. vw_set_tx_pin

- extern void vw_set_tx_pin(uint8_t pin);
- Set the digital IO pin to use for transmitting data. Defaults to 12.

2.5.4 Keyboard.h LIBRARY

The keyboard functions enable 32u4 or SAMD micro based boards to send keystrokes to an attached computer through their micro's native USB port. The library supports the use of modifier keys. Modifier keys change the behaviour of another key when pressed simultaneously. To use this library, At the starting of the program we must write #include<Keyboard.h>.

Functions used in Keboard.h

1. Keyboard.begin()

- Description

When used with a Leonardo or Due board, Keyboard.begin() starts emulating a keyboard connected to a computer. To end control, use Keyboard.end().

- Syntax

Keyboard.begin()

- Parameters

None

- Returns

Nothing

2. Keyboard.press()

- Description

When called, `Keyboard.press()` functions as if a key were pressed and held on your keyboard. Useful when using modifier keys. To end the key press, use `Keyboard.release()` or `Keyboard.releaseAll()`.

It is necessary to call `Keyboard.begin()` before using `press()`.

- Syntax

`Keyboard.press(key)`

- Parameters

key: the key to press. Allowed data types: char.

- Returns

Number of key presses sent. Data type: size_t.

3. **Keyboard.releaseAll()**

- Description

Let's go of all keys currently pressed. See `Keyboard.press()` for additional information.

- Syntax

`Keyboard.releaseAll()`

- Parameters

None

- Returns

Nothing

2.5.5 **Mouse.h LIBRARY**

The mouse functions enable 32u4 or SAMD micro based boards to control cursor movement on a connected computer through their micro's native USB port. When updating the cursor position, it is always relative to the cursor's previous location. These core libraries allow the 32u4 and SAMD based boards (Leonardo, Esplora, Zero, Due and MKR Family) to appear as a native Mouse and/or Keyboard to a connected computer.

Functions used in Mouse.h:

1. Mouse.begin()

- Description

Begins emulating the mouse connected to a computer. `begin()` must be called before controlling the computer. To end control, use `Mouse.end()`.

- Syntax

Mouse.begin()

- Parameters

None

- Returns

Nothing

2. Mouse.click()

- Description

Sends a momentary click to the computer at the location of the cursor. This is the same as pressing and immediately releasing the mouse button.

Mouse.click() defaults to the left mouse button.

- Syntax

Mouse.click()

Mouse.click(button)

- Parameters

button: which mouse button to press. Allowed data types: char.

MOUSE_LEFT (default)

MOUSE_RIGHT

MOUSE_MIDDLE

- Returns

Nothing

3. Mouse.move()

- Description

Moves the cursor on a connected computer. The motion onscreen is always relative to the cursor's current location. Before using Mouse.move() you must call Mouse.begin()

- Syntax

Mouse.move(xVal, yVal, wheel)

- Parameters

xVal: amount to move along the x-axis. Allowed data types: signed char.

yVal: amount to move along the y-axis. Allowed data types: signed char.

wheel: amount to move scroll wheel. Allowed data types: signed char.

- Returns

Nothing

4. Mouse.press()

- Description

Sends a button press to a connected computer. A press is the equivalent of clicking and continuously holding the mouse button. A press is cancelled with `Mouse.release()`.

Before using `Mouse.press()`, you need to start communication with `Mouse.begin()`. `Mouse.press()` defaults to a left button press.

- Syntax

`Mouse.press()`

`Mouse.press(button)`

- Parameters

button: which mouse button to press. Allowed data types: char.

`MOUSE_LEFT` (default)

`MOUSE_RIGHT`

`MOUSE_MIDDLE`

- Returns

Nothing

2.5 CONCLUSION

This chapter gives an overview of the system architecture and explains in detail about each component used. It also compares the present day wired mouse with our wireless air mouse and mentions its specifications. Looking for the hardware requirements, the gyroscope sensor which is present in the MPU-6050 senses the change in motion, in our case the tilt of the hand. The change in the gyroscope values of the MPU are transmitted to the PC, where in the software applications take control and moves the mouse cursor.

Arduino, the name now synonymous with microcontroller and sensors seemed the suitable platform. Its main advantage is being easily programmable and has massive online support. To control the mouse pointer, we use some mouse and keyboard functions, which take control over the mouse and keyboard. To make the mouse wireless, RF Modules were used as they are cheap compared to other modules like Bluetooth and Zigbee.

CHAPTER-3

PROJECT IMPLEMENTATION

3.1 BLOCK DIAGRAM

The proposed system architecture uses Gyroscope as a sensor to detect the rotational motion which is present in MPU-6050. It is interfaced to a microcontroller, Arduino Uno in this case. The microcontroller does the necessary processing on the sensor data and transmits it through a wireless RF transmitter module. The wireless RF receiver receives the data and feeds it to the computer through one of the interfaces. The gyro meter senses the change in motion, in our case rotation of hand. The change in gyro meter values would be transmitted to the PC. The application running on the PC analyses the received values and perform the necessary action on the mouse pointer. The sensor which gets input from the movement of the gesture and the values would be changed in the gyro meter this would be continuously transmitted through the transmitter and which is processes by the microcontroller and then received by the receiver and given to PC and I will move the mouse pointer.

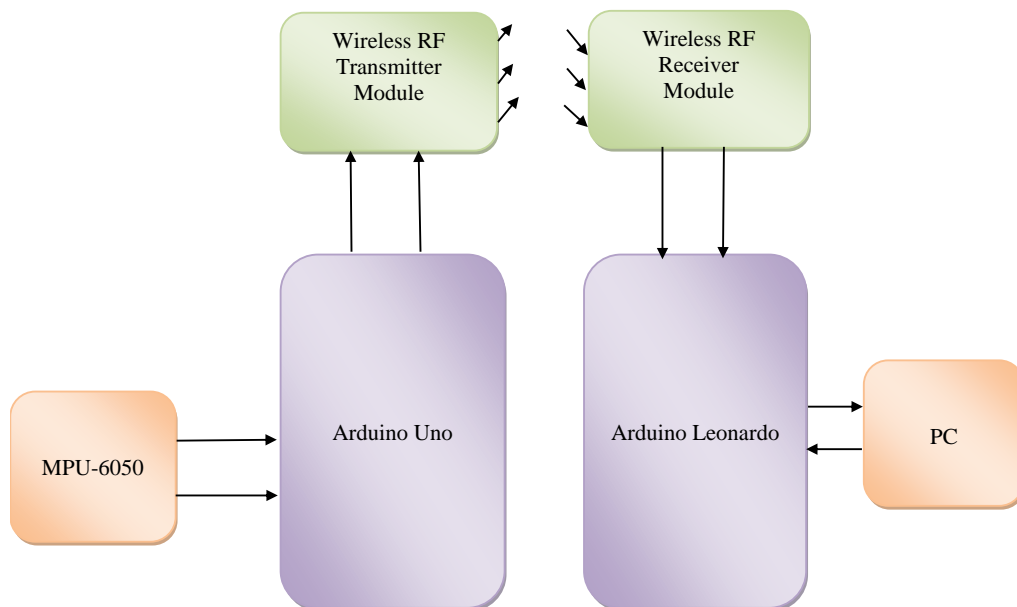


Fig.3.1: Block diagram

3.2 HARDWARE INTERFACING

The wireless air mouse consists of two circuits namely the transmitter and the receiver. These two circuits are interconnected wirelessly using the RF links which uses the digital modulation techniques.

3.1.1 TRANSMITTER CIRCUIT DIAGRAM

MPU-6050 consists of three sensors, but we require only gyroscope sensor. So, we use pins only related to that sensor. Vcc and GND are connected to respective Vcc and GND pins of Arduino Uno and serial clock pin (SCL) and Serial data pin (SDA) are connected to Analog pins (A4) and (A5) respectively. Similarly, Vcc and GND pins of RF transmitter and its data pin is connected to digital pin 2 of Arduino Uno. The four buttons can be connected to desirable digital pins of Arduino and in our project, Right click button is connected to 11th pin of Arduino, left click button is connected to 10th pin of Arduino, Right arrow button is connected to 7th pin of Arduino and left arrow button is connected to 8th pin of Arduino.

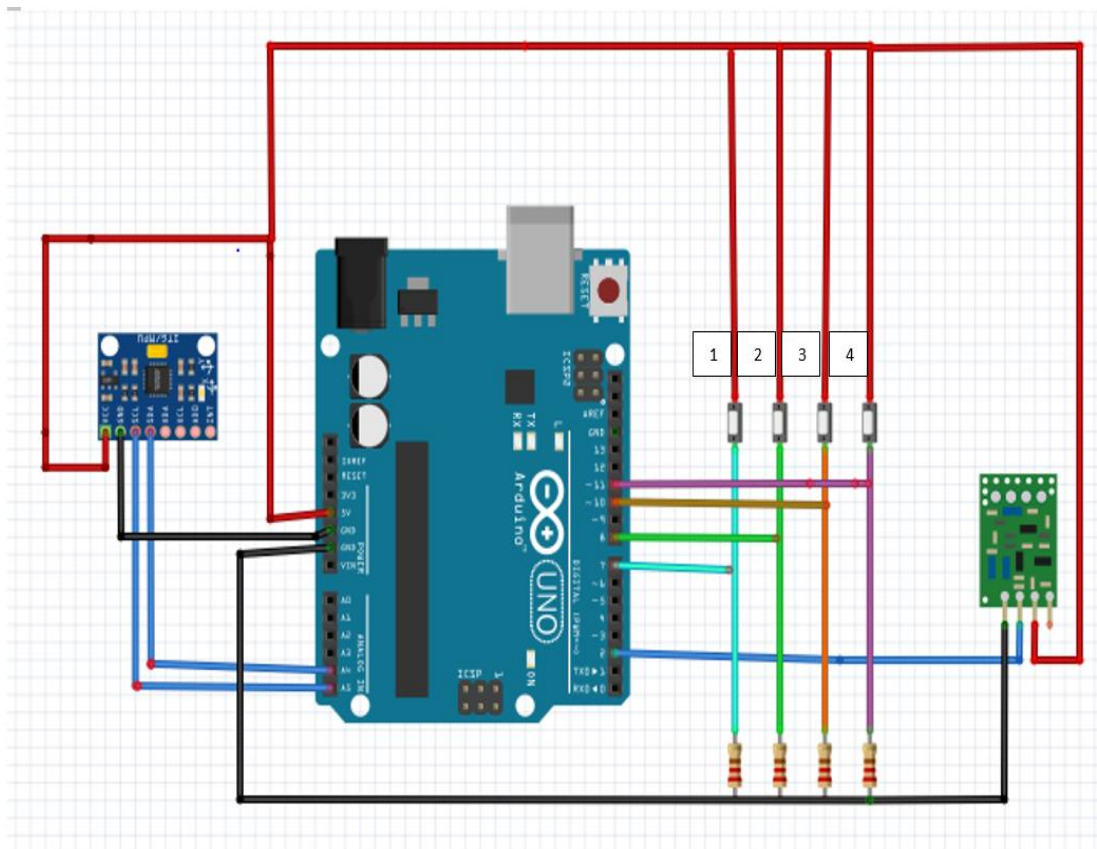


Fig.3.2: Transmitter circuit diagram

We have connected the MPU-6050 module to Arduino Uno following the I2C communication protocol. Arduino pins A4 and A5 are internally assigned as SCL and SDA. the data is transmitted from Arduino to RF transmitter through digital pin2. In the code we assigned each button to a letter, so the button is known just by transmitting the respective letter.

1- Right click

2- Left click

3-Right arrow

4-Left arrow

Table.3.1: Transmitter circuit connections

Name of the pin	Connected to
Gyroscope sensor (Vcc)	5V Pin of Arduino
Gyroscope sensor (GND)	GND Pin of Arduino
Gyroscope sensor (SCL)	A5 Pin of Arduino
Gyroscope sensor (SDA)	A4 Pin of Arduino
RF transmitter (Vcc)	5V Pin of Arduino
RF transmitter (GND)	GND Pin of Arduino
RF transmitter (data)	Digital Pin 2 of Arduino
Right click button	11th Pin of Arduino
Left click button	10th Pin of Arduino
Right arrow button	7th Pin of Arduino
Left arrow button	8 th Pin of Arduino

3.1.2 RECEIVER CIRCUIT DIAGRAM

The receiver circuits comprise of an Arduino Leonardo and a RF receiver module. The receiver is wirelessly connected to the transmitter circuit with a bit rate of 4000 bits per second. The received modulated signal will be demodulated by the RF receiver and decoded. The entire processing of the data is done by the ATMEGA32u4 based Arduino Leonardo.

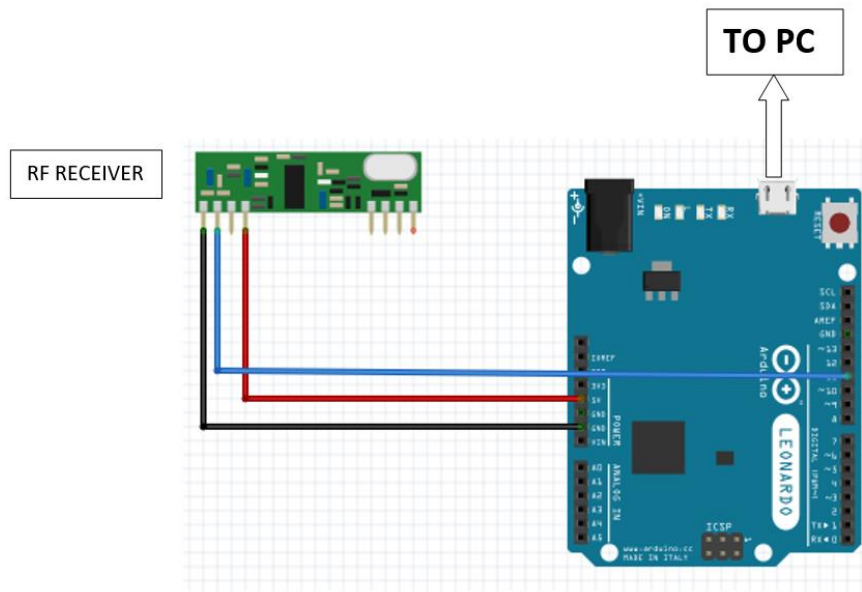


Fig.3.3: Receiver circuit diagram

The data that is received wirelessly through Rf receiver is transmitted to Arduino Leonardo through a digital pin 11. The data is processed and decoded, then it gives commands to the computer to follow certain actions.

Table.3.2: Receiver circuit connection

Name of the pin	Connected to
RF receiver (Vcc)	3.3V / 5v of Leonardo board
RF receiver (GND)	GND of Leonardo board
RF receiver (data)	11 th of Leonardo board

3.3 HANDHELD MODULE

The transmitter circuit consists of 8-bit microcontroller which comes under the family of ATMEGA with an IC number ATMEGA328P-PU. All the functionalities at the transmitter part are controlled with the help of microcontroller. Apart from the microcontroller it consists of a MEMS (Micro Electro-Mechanical System) gyroscope sensor (MPU-6050) and RF transmitter. The MPU-6050 records the hand/wrist movement of the human which resembles a mouse that doesn't have a surface beneath it. The RF transmitter converts the data which is obtained from the gyroscope sensor

and switches into the digital signals with the help of Manchester encoding technique and further modulated using ASK (Amplitude Shift Keying) modulation. It also consists of mouse functionalities like right click, left click, drag and drop and some keyboard functionalities like up arrow and down arrow. All these are implemented using tactile (push button) switches.

The gyroscope sensor detects the rotation of the device with respect to X axis, Y axis and Z axis. As the cursor movement of the mouse depends only on two axes. X axis and Z axis are chosen because these axes exactly resemble the X and Y coordinates of the mouse. The data from the switches and the gyroscope sensor are processed using the ATMEGA328P-PU microcontroller. All the above data is temporarily stored in a buffer. As the RF transmitter module has only one channel to communicate with the receiver circuit, the data is concatenated and made into a string and then transmitted wirelessly.

3.4 HOST CONNECTED DEVICE

The data that gets transmitted through the handheld module should be received continuously as the movement of hand and cursor should not have any latency. We receive the data in terms of bytes. After receiving the whole string, we decode the string and make it into required attributes which tell the movement of mouse in two directions. After decoding the data, we must send the commands to the host device. As Arduino Leonardo has USB 2.0 communication support, we directly send the commands to host based on decoded data.

In brief, the RF receiver receives the data as a string and further it is decoded using Arduino Leonardo. The decoded data is again included in the respective commands to further perform mouse and keyboard functionalities.

3.5 SOFTWARE IMPLEMENTATION

The rotational movement with respect to all directions that is taken from the MPU-6050 is transferred to the processing unit which is Arduino Uno. For transferring MPU data follows the I2C communication protocol. So, the Arduino provides a library called `Wire.h` which provides functions such as `Wire.beginTransmission()`, `Wire.endTransmission()` for communication between MPU and Arduino Uno. The

status of the buttons is also collected concurrently. The data of the mouse movement and the status of the buttons are made into a string and transmitted using a RF transmitter. But if we concatenate all the values and transmit to the receiver there will be an ambiguity at the receiver. For example the mouse movement with respect to x axis is 23 and with respect to y axis is 56 and there is right click at that time, If we concatenate all the values into a string it will look like 2356t. so at the receiver there is ambiguity in decoding as there are three possibilities possible

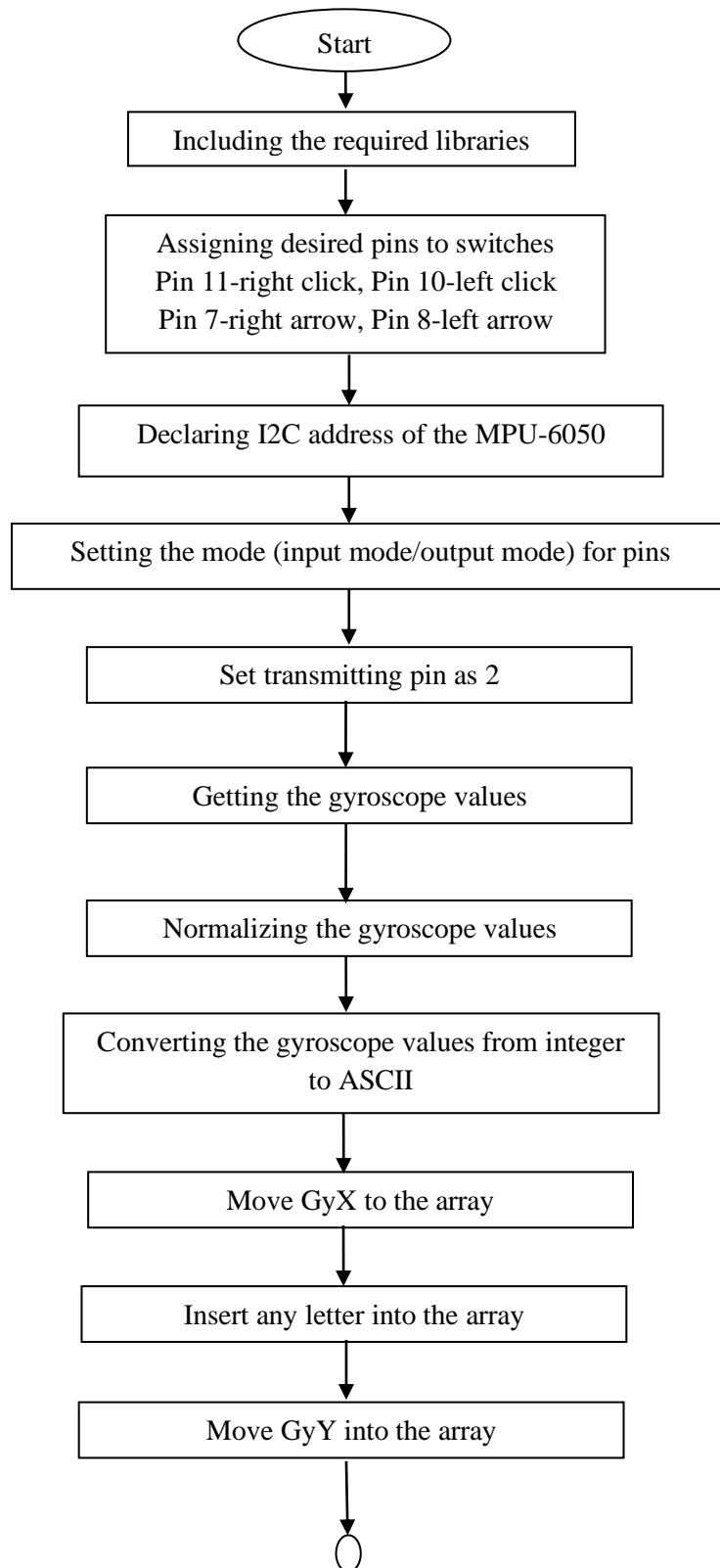
1. movement with respect to x axis =2 and movement with respect to y axis = 356
2. movement with respect to x axis =23 and movement with respect to y axis = 56
3. movement with respect to x axis =235 and movement with respect to y axis = 6

We can overcome the ambiguity by implementing the below process

So, we are impending letter H between x axis movement and y axis movement and, we are impending letter A between y axis movement and button status. So that the receiver can decode the movement of X-axis by parsing the received data from starting unit until we get the character H. We get the movement of y axis by parsing the data between letters H and A. We have a function called `Mouse.move(x,y)` so, by passing the x and y axis movements as arguments to this function we can control the movement of cursor. We also have functions like `Keyboard.press()` to control the keyboard functionalities.

3.5.1 TRANSMITTER CODE FLOW CHART

At the first step we must include the required libraries in the code, assign desired pins to switches and declare the I2C address of MPU. We must set the mode either input or output for all pins and set transmitting pin as 2. After getting the gyroscope values we should normalize them and convert them to Ascii values. In order to differentiate X and Y values we first move X value to an array insert a letter and then move y value to the array. Then check the status of right click button, if the button status is high then concatenate the string with character 'r'. if the right click status is low then check the status of right arrow. If the right arrow button status is high concatenate the sting with character 't'. If the button status is low, then check the status of left arrow. If the left arrow button status is high concatenate the sting with character 'i'. If the button status is low, then check the status of left click. If the left click button status is high concatenate the sting with character 's'. if the button status is low then check and send string through virtual wire.



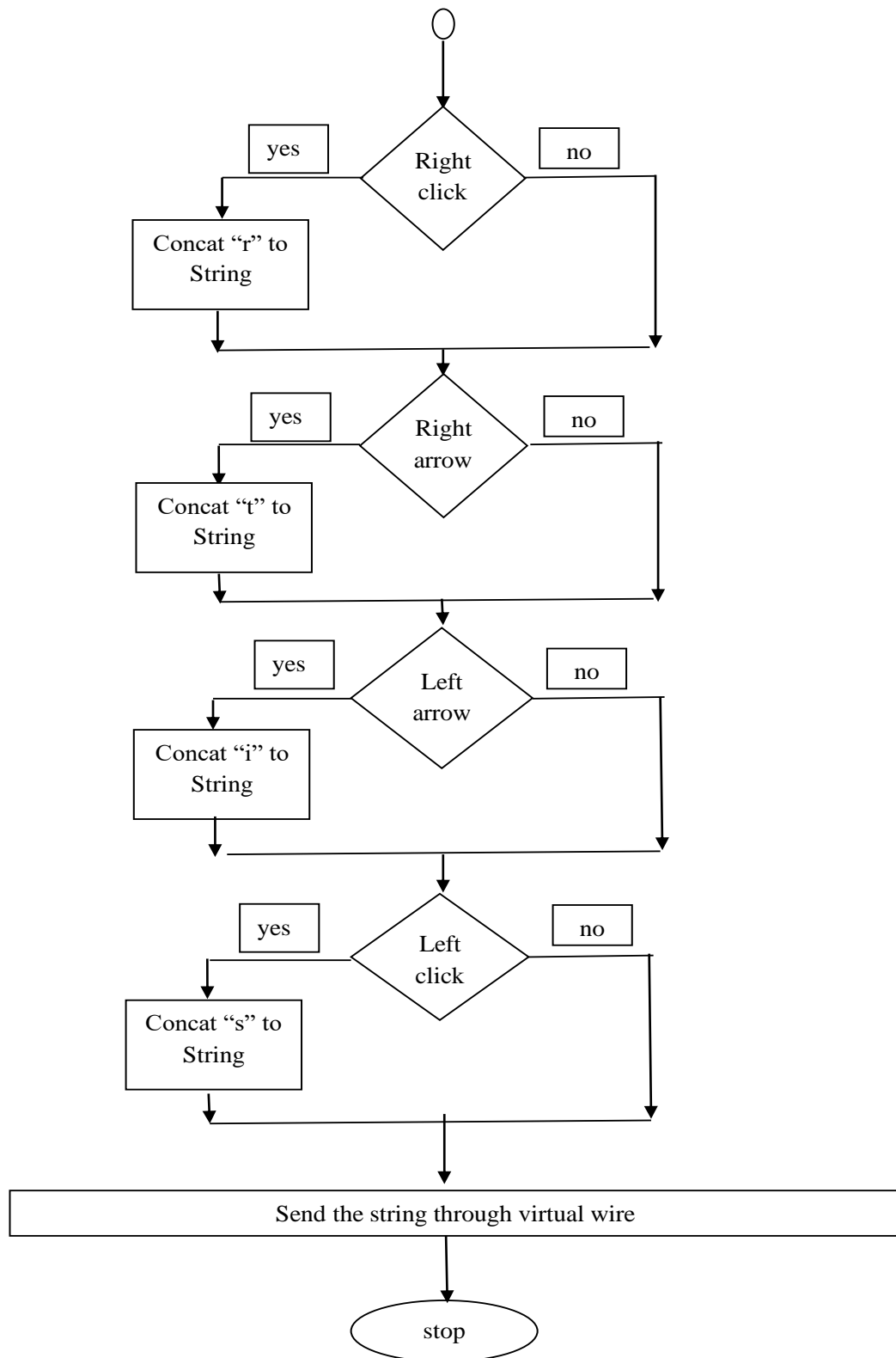
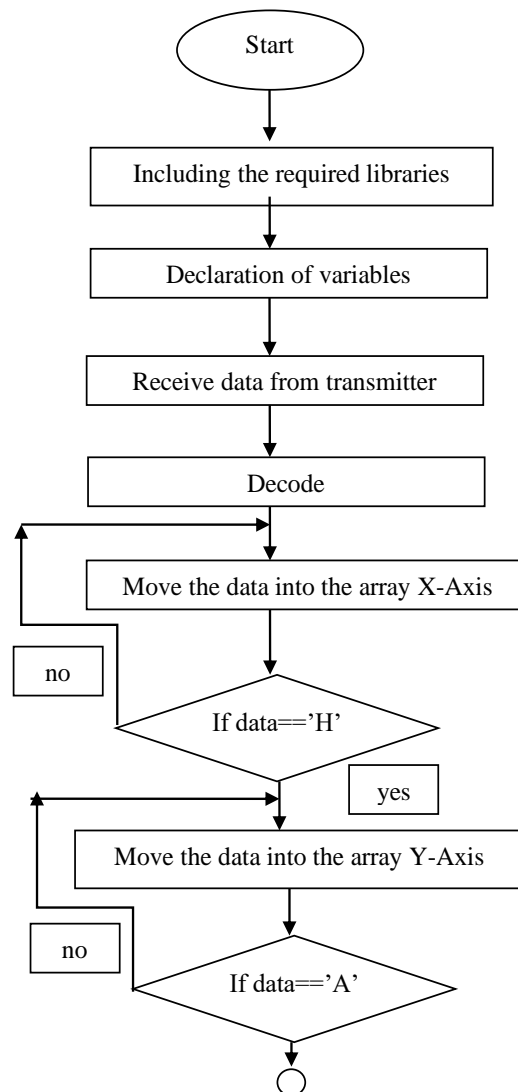


Fig.3.4: Transmitter flow chart

3.5.2 RECEIVER CODE FLOWCHART

We should include the required libraries and declare the variables. Then decode the received data from transmitter. We must move the data to x axis array until we encounter character 'H' and then we must move data to Y axis array until we encounter character 'A'. Then we should store the additional character after A into a variable 'a'. Then the cursor will be moved according to the command `Mouse.move(Xaxis,Yaxis)`. The value present in variable a is compared to the four characters r, s, t and i, then appropriate action will be taken place. If the value present in variable is equal to the past value present in variable a then no change of action take place. We should check the value present in a continuously to perform appropriate action



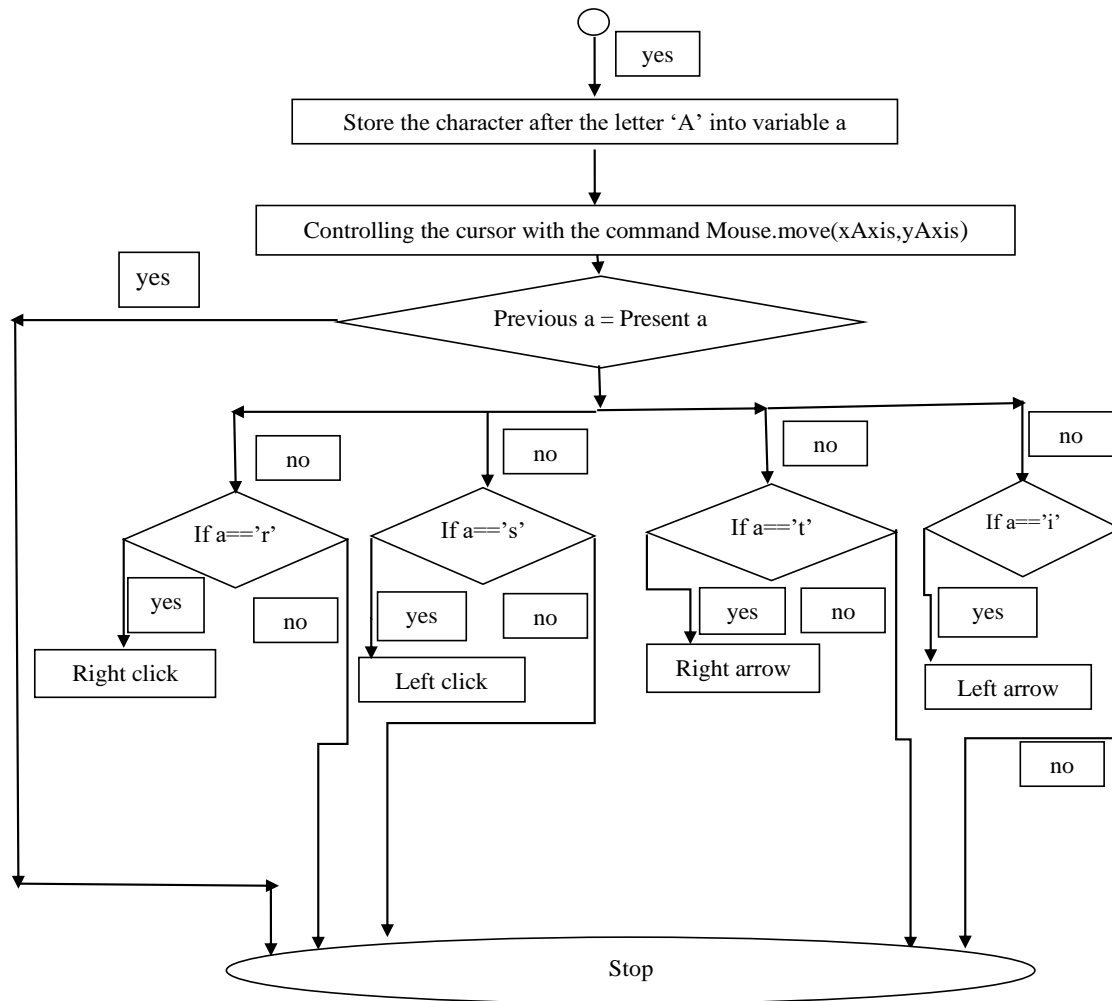


Fig.3.5: Receiver Flow chart

3.6 CONCLUSION

This chapter gives a detailed explanation of the working of the proposed system. The proposed system architecture uses gyroscope as a sensor to detect the motion. Gyroscope present in the MPU-6050 is interfaced to a microcontroller, Arduino Uno in this case. The microcontroller does the necessary processing on the sensor data and transmits it through a wireless transmitter module, RF Transmitter in this case.

The wireless receiver, that is RF receiver receives the data and feeds it to the computer through one of the interfaces. As we used RF technology as the wireless transmission media. We must send the data sequentially. We made the data in the form of string and transmitted it at a rate of 4000 bits per sec. The application running on the PC analyses the received values and perform the necessary action on the mouse pointer. We further added few keyboard functionalities for the flexibility of the user.

CHAPTER 4

RESULTS AND DISCUSSIONS

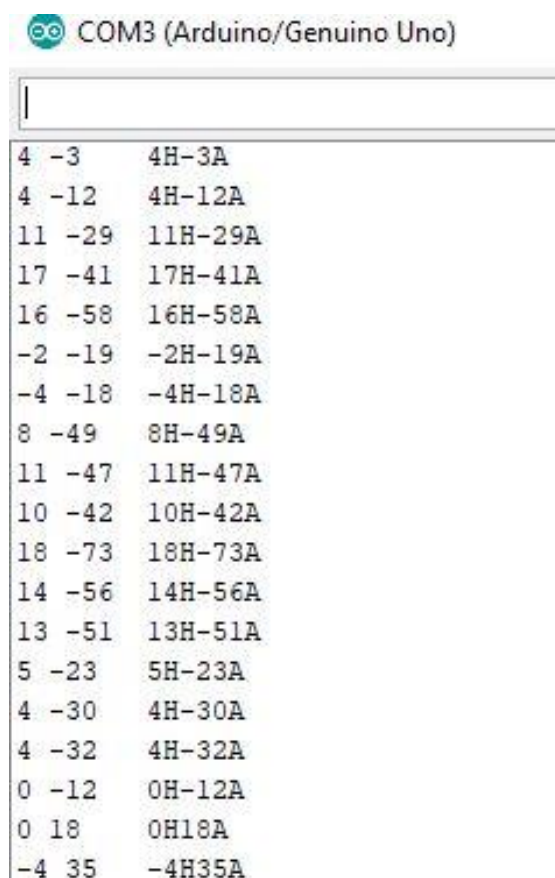
4.1 MOUSE MOVEMENT

The values depending on the mouse movement are encoded in the form of string at transmitter side and received at the receiver and the string is decoded for appropriate movement of cursor. The below two sections represents the transmitter and receiver outputs and their explanation.

4.1.1 TRANSMITTER RESULT

The values obtained due to the movement of handheld module are changed by dividing with a value depending upon our sensitivity requirements.

The first column represents the modified value of the rotational movement along the X axis and the second column represents the modified value of the rotational movement along the Z axis. The values that are modified are encoded in the form of a string and the third column represents this string literal.



4	-3	4H-3A
4	-12	4H-12A
11	-29	11H-29A
17	-41	17H-41A
16	-58	16H-58A
-2	-19	-2H-19A
-4	-18	-4H-18A
8	-49	8H-49A
11	-47	11H-47A
10	-42	10H-42A
18	-73	18H-73A
14	-56	14H-56A
13	-51	13H-51A
5	-23	5H-23A
4	-30	4H-30A
4	-32	4H-32A
0	-12	0H-12A
0	18	0H18A
-4	35	-4H35A

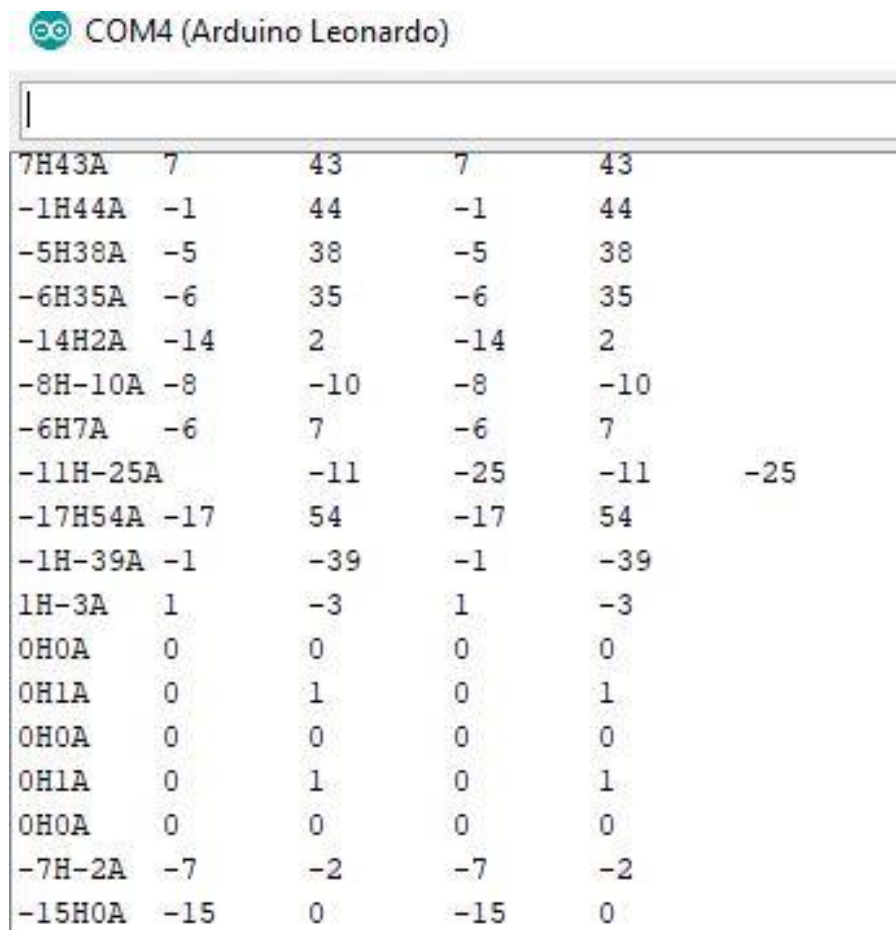
Fig.4.1: Mouse movement transmitter result

These values are printed on the Serial monitor screen of Arduino IDE. Each row represents the position of hand module at that instant of time. This string is transmitted to host device through RF Transmitter.

4.1.2 RECEIVER RESULT

The string will be received from the handheld module through RF receiver. The first column represents this string.

The next two columns represent the substrings decoded from the string whose values depend on the movement of the handheld module. These substrings are then converted into integer literals which are represented by last two columns. The integer values we get represents the movement of handheld module along the two axes.



7H43A	7	43	7	43
-1H44A	-1	44	-1	44
-5H38A	-5	38	-5	38
-6H35A	-6	35	-6	35
-14H2A	-14	2	-14	2
-8H-10A	-8	-10	-8	-10
-6H7A	-6	7	-6	7
-11H-25A		-11	-25	-11 -25
-17H54A	-17	54	-17	54
-1H-39A	-1	-39	-1	-39
1H-3A	1	-3	1	-3
0H0A	0	0	0	0
0H1A	0	1	0	1
0H0A	0	0	0	0
0H1A	0	1	0	1
0H0A	0	0	0	0
-7H-2A	-7	-2	-7	-2
-15H0A	-15	0	-15	0

Fig.4.2: Mouse movement receiver result

These integer values are passed as arguments to a function which is responsible for movement of cursor on the host PC/Laptop.

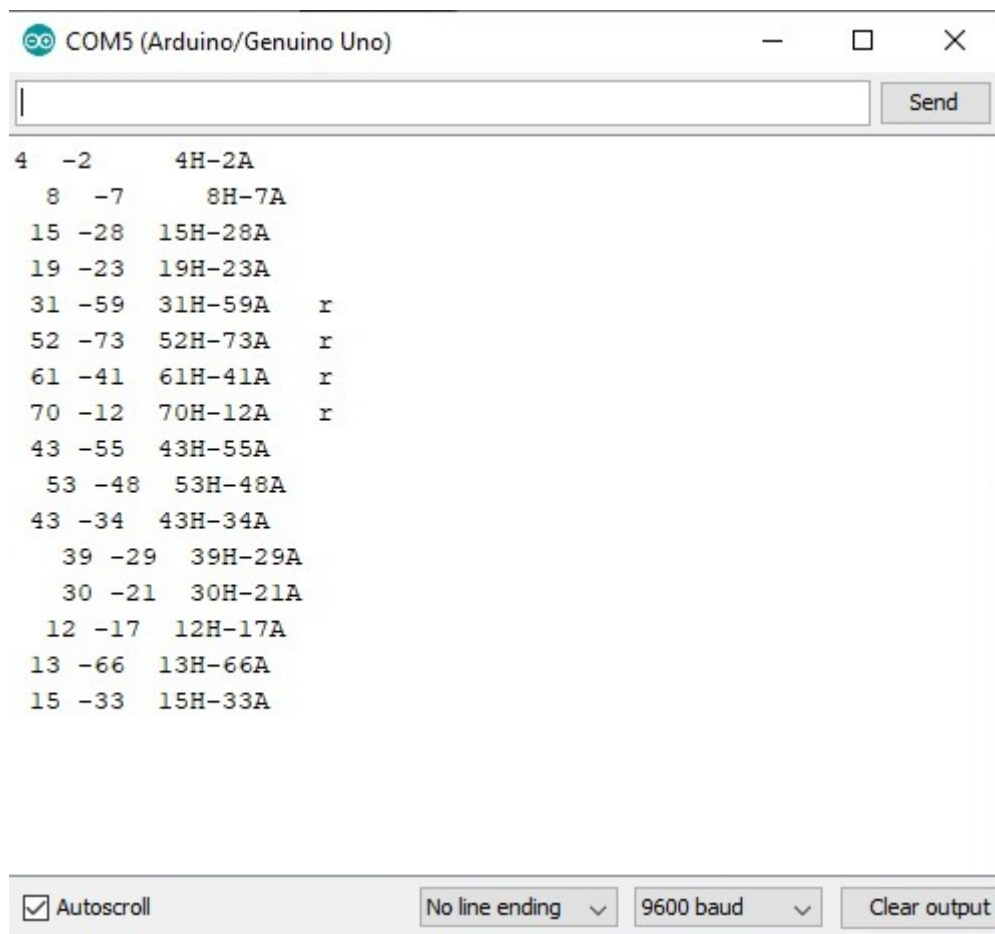
4.2 RIGHT CLICK

The values depending on the mouse movement and status of right click are encoded in the form of string at transmitter side and received at the receiver and the string is decoded for appropriate action and movement of cursor. The below two sections represents the transmitter and receiver outputs and their explanation.

4.2.1 TRANSMITTER RESULT

The values obtained due to the movement of handheld module are changed by dividing with a value depending upon our sensitivity requirements.

The first column represents the modified value of the rotational movement along the X axis and the second column represents the modified value of the rotational movement along the Z axis. The character "r" is assigned for the status of right click. This character is encoded with the values of movement along axes into a string. The third column represents this string literal.



The screenshot shows the Serial Monitor window for COM5 (Arduino/Genuino Uno). The output displays a series of data points in three columns: X-axis modified value, Z-axis modified value, and a string literal. The string literal includes the character 'r' for right click. The data is as follows:

X-axis modified value	Z-axis modified value	String literal
4	-2	4H-2A
8	-7	8H-7A
15	-28	15H-28A
19	-23	19H-23A
31	-59	31H-59A r
52	-73	52H-73A r
61	-41	61H-41A r
70	-12	70H-12A r
43	-55	43H-55A
53	-48	53H-48A
43	-34	43H-34A
39	-29	39H-29A
30	-21	30H-21A
12	-17	12H-17A
13	-66	13H-66A
15	-33	15H-33A

The Serial Monitor window also shows the 'Send' button and settings at the bottom: Autoscroll is checked, No line ending is selected, 9600 baud is selected, and there is a 'Clear output' button.

Fig.4.3: Right click transmitter result

These values are printed on the Serial monitor screen of Arduino IDE. Each row represents the position of hand module at that instant of time. This string is transmitted to host device through RF Transmitter.

4.2.2 RECEIVER RESULT

The string will be received from the handheld module through RF receiver. The first column represents this string.

The next two columns represent the substrings decoded from the string whose values depend on the movement of the handheld module. These substrings are then converted into integer literals which are represented by last two columns. If the left click button is pressed, there will be an additional character along with the string. The integer values we get represents the movement of handheld module along the two axes and the character represents the status of the right click.

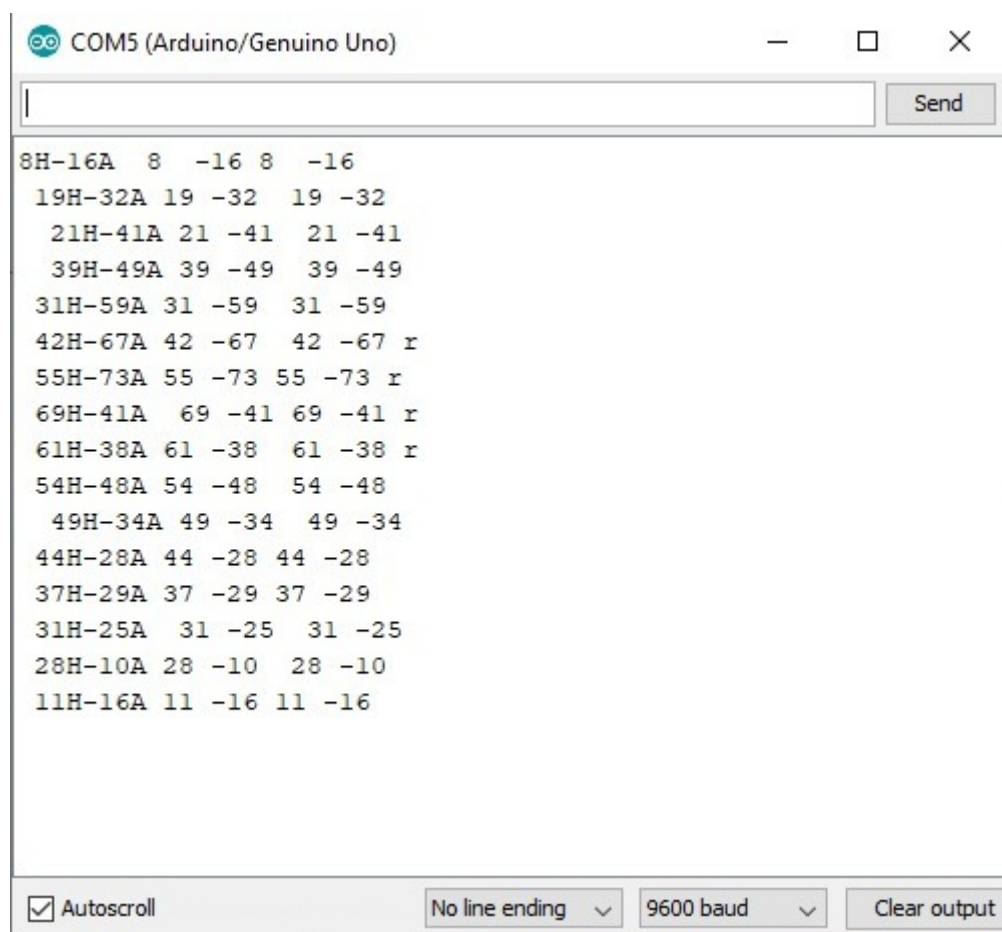


Fig.4.4: Right click receiver result

These values are passed as arguments to a function which is responsible for appropriate action.

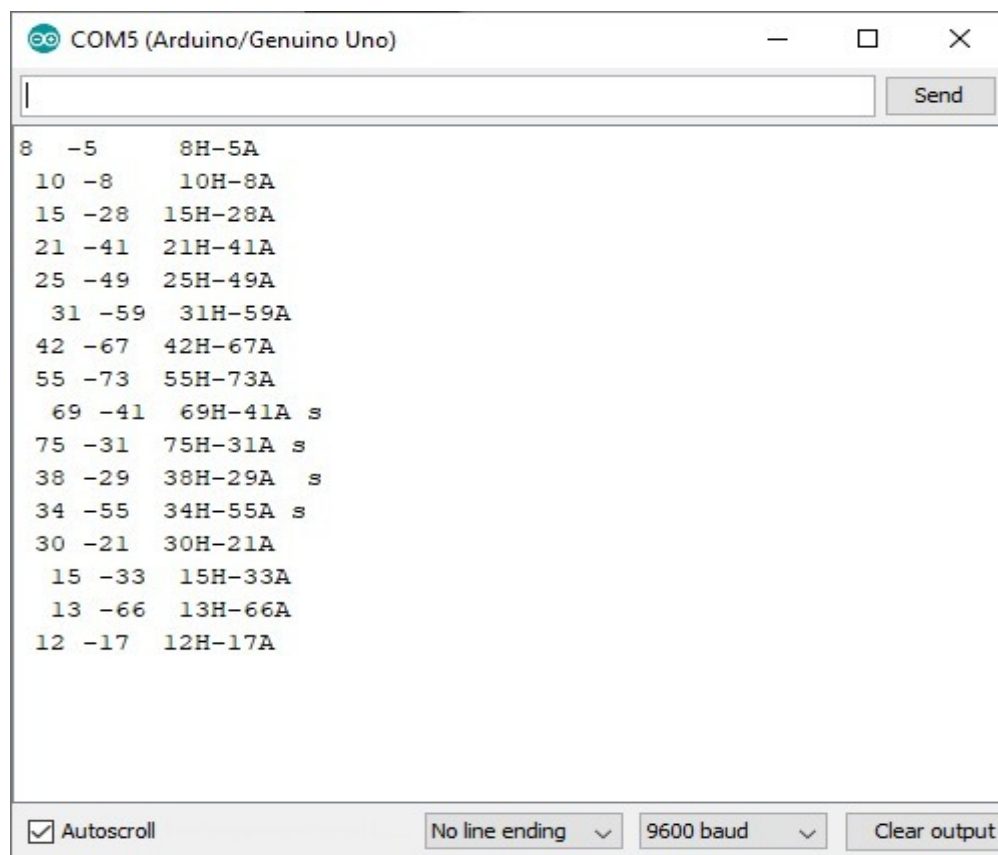
4.3 LEFT CLICK

The values depending on the mouse movement and status of left click are encoded in the form of string at transmitter side and received at the receiver and the string is decoded for appropriate action and movement of cursor. The below two sections represents the transmitter and receiver outputs and their explanation

4.3.1 TRANSMITTER RESULT

The values obtained due to the movement of handheld module are changed by dividing with a value depending upon our sensitivity requirements.

The first column represents the modified value of the rotational movement along the X axis and the second column represents the modified value of the rotational movement along the Z axis. The character "s" is assigned for the status of left click. This character is encoded with the values of movement along axes into a string. The third column represents this string literal.



The screenshot shows the Arduino IDE serial monitor window titled "COM5 (Arduino/Genuino Uno)". The window contains a text area with the following output:

```
8 -5 8H-5A
10 -8 10H-8A
15 -28 15H-28A
21 -41 21H-41A
25 -49 25H-49A
31 -59 31H-59A
42 -67 42H-67A
55 -73 55H-73A
69 -41 69H-41A s
75 -31 75H-31A s
38 -29 38H-29A s
34 -55 34H-55A s
30 -21 30H-21A
15 -33 15H-33A
13 -66 13H-66A
12 -17 12H-17A
```

At the bottom of the window, there are controls: a checked "Autoscroll" checkbox, a "No line ending" dropdown menu, a "9600 baud" dropdown menu, and a "Clear output" button.

Fig.4.5: Left click transmitter result

These values are printed on the Serial monitor screen of Arduino IDE. Each row represents the position of hand module at that instant of time. This string is transmitted to host device through RF Transmitter.

4.3.2 RECEIVER RESULT

The string will be received from the handheld module through RF receiver. The first column represents this string.

The next two columns represent the substrings decoded from the string whose values depend on the movement of the handheld module. These substrings are then converted into integer literals which are represented by last two columns. If the left click button is pressed, there will be an additional character along with the string. The integer values we get represents the movement of handheld module along the two axes and the character represents the status of the left click.

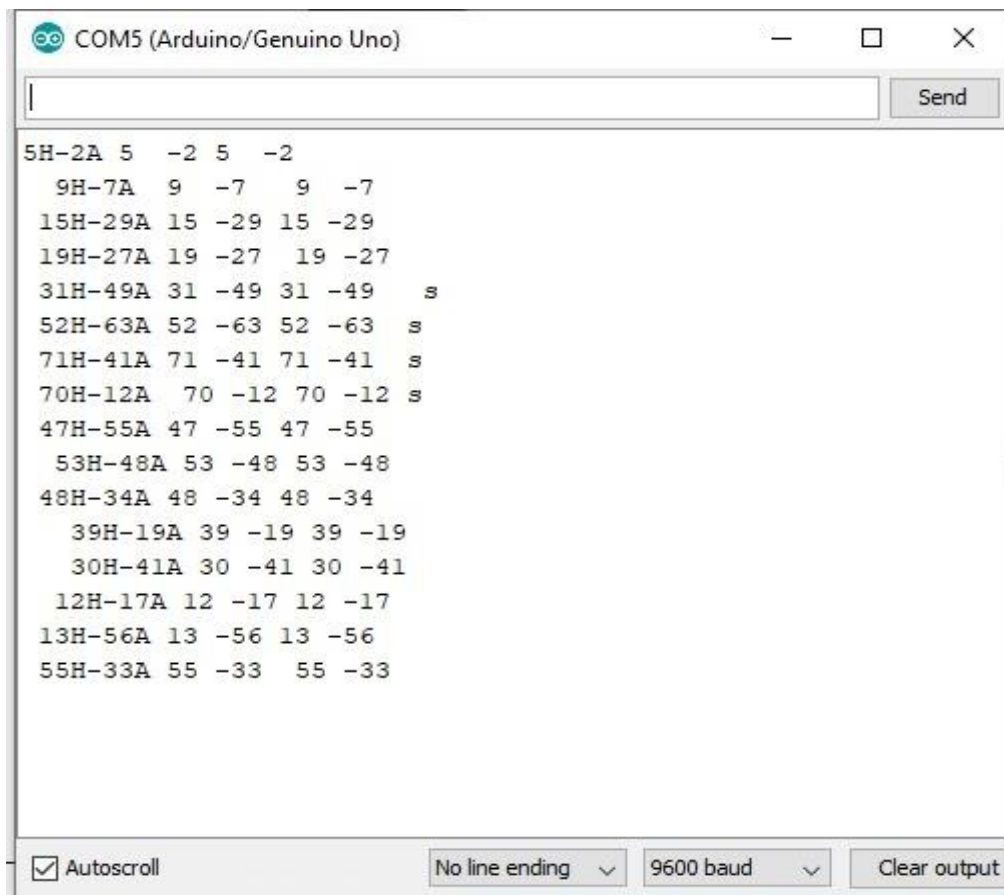


Fig.4.6: Left click receiver result

These values are passed as arguments to a function which is responsible for appropriate action.

4.4 RIGHT ARROW

The values depending on the mouse movement and status of Right arrow are encoded in the form of string at transmitter side and received at the receiver and the string is decoded for appropriate action and movement of cursor. The below two sections represents the transmitter and receiver outputs and their explanation

4.4.1 TRANSMITTER RESULT

The values obtained due to the movement of handheld module are changed by dividing with a value depending upon our sensitivity requirements.

The first column represents the modified value of the rotational movement along the X axis and the second column represents the modified value of the rotational movement along the Z axis. The character "t" is assigned for the status of right arrow. This character is encoded with the values of movement along axes into a string. The third column represents this string literal.

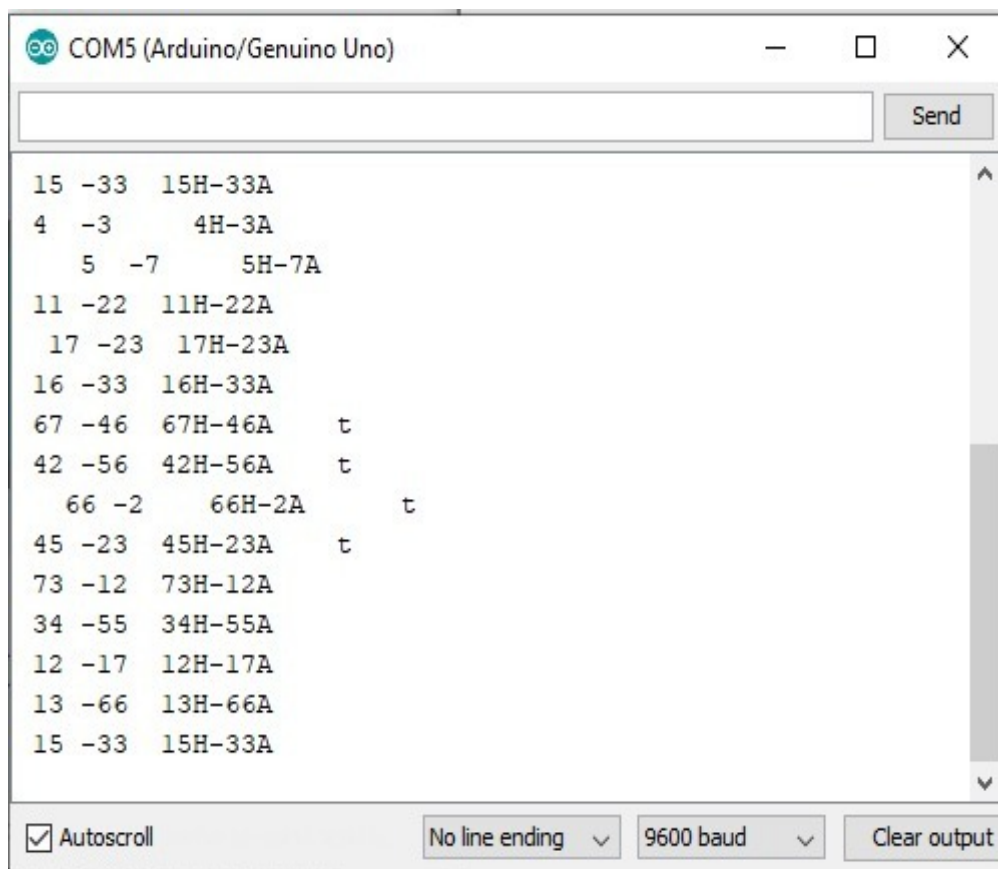


Fig.4.7: Right arrow transmitter result

These values are printed on the Serial monitor screen of Arduino IDE. Each row represents the position of hand module at that instant of time. This string is transmitted to host device through RF Transmitter.

4.4.2 RECEIVER RESULT

The string will be received from the handheld module through RF receiver. The first column represents this string.

The next two columns represent the substrings decoded from the string whose values depend on the movement of the handheld module. These substrings are then converted into integer literals which are represented by last two columns. If the left click button is pressed, there will be an additional character along with the string. The integer values we get represents the movement of handheld module along the two axes and the character represents the status of the right arrow.

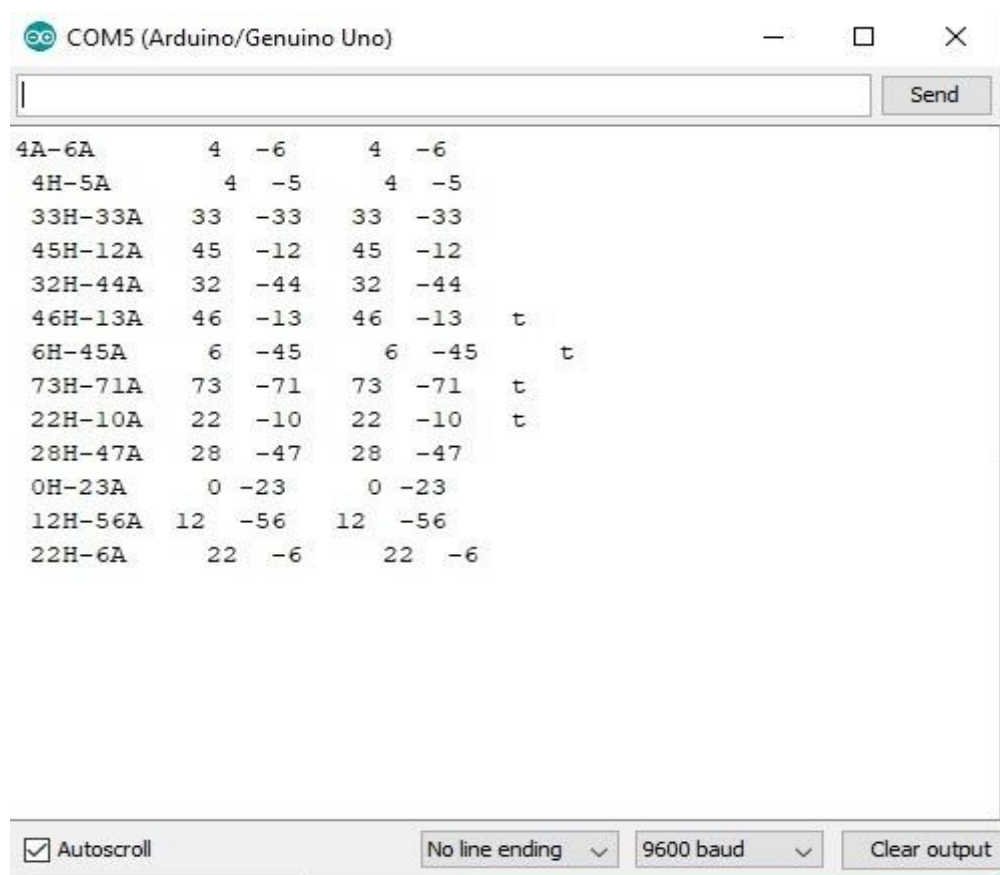


Fig.4.8: Right arrow receiver result

These values are passed as arguments to a function which is responsible for appropriate action.

4.5 LEFT ARROW

The values depending on the mouse movement and status of Right arrow are encoded in the form of string at transmitter side and received at the receiver and the string is decoded for appropriate action and movement of cursor. The below two sections represents the transmitter and receiver outputs and their explanation

4.5.1 TRANSMITTER RESULT

The values obtained due to the movement of handheld module are changed by dividing with a value depending upon our sensitivity requirements.

The first column represents the modified value of the rotational movement along the X axis and the second column represents the modified value of the rotational movement along the Z axis. The character "i" is assigned for the status of left arrow. This character is encoded with the values of movement along axes into a string. The third column represents this string literal.

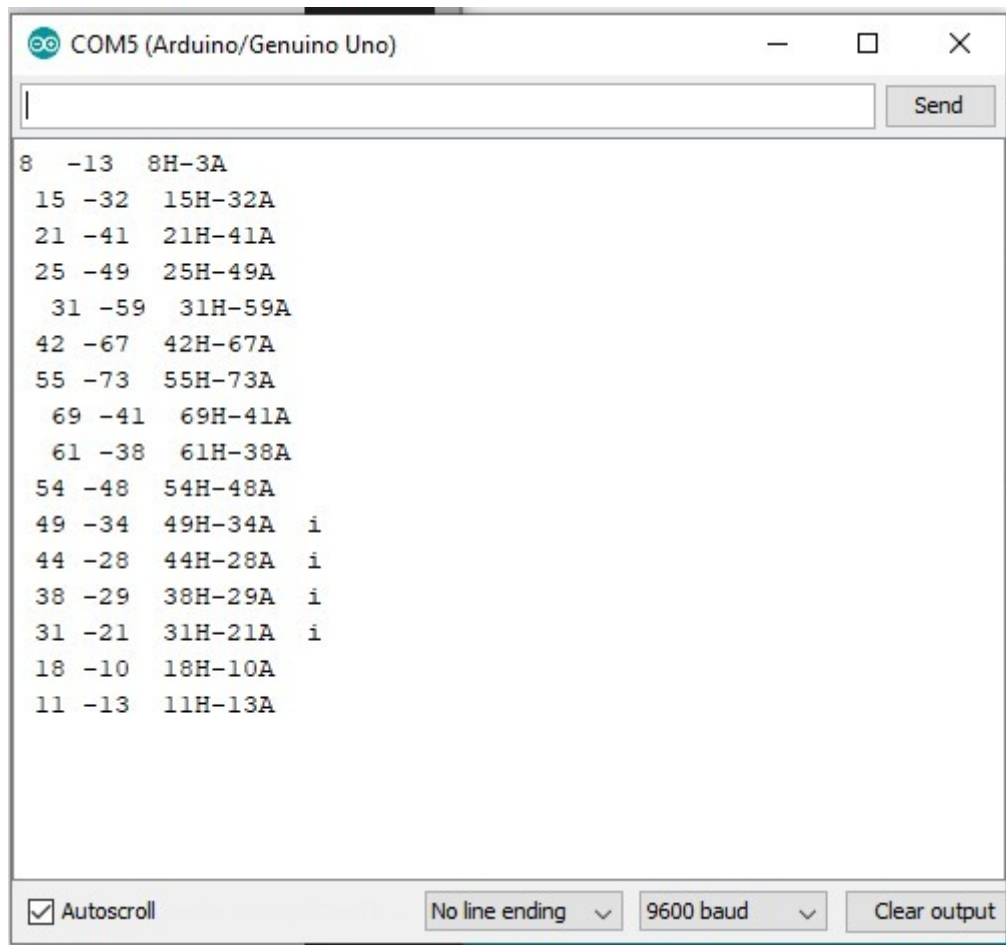


Fig.4.9: Left arrow transmitter result

These values are printed on the Serial monitor screen of Arduino IDE. Each row represents the position of hand module at that instant of time. This string is transmitted to host device through RF Transmitter.

4.5.2 RECEIVER RESULT

The string will be received from the handheld module through RF receiver. The first column represents this string.

The next two columns represent the substrings decoded from the string whose values depend on the movement of the handheld module. These substrings are then converted into integer literals which are represented by last two columns. If the left click button is pressed, there will be an additional character along with the string. The integer values we get represents the movement of handheld module along the two axes and the character represents the status of the left arrow.

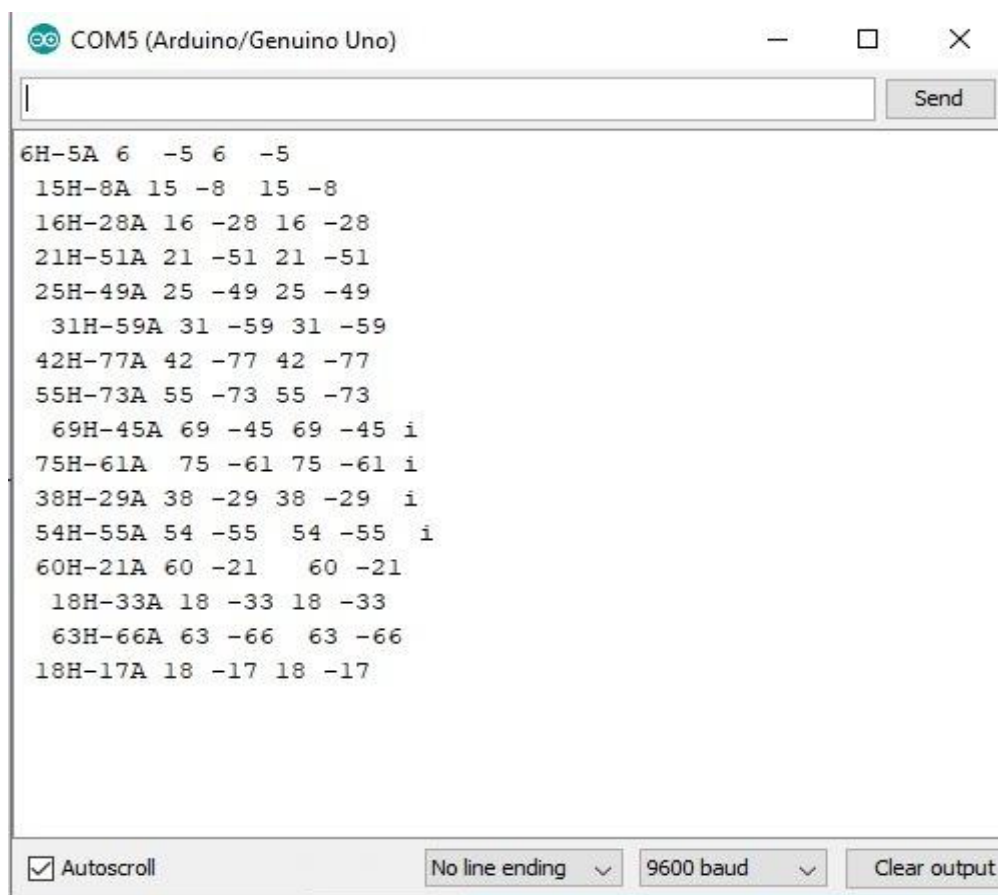


Fig.4.10: Left arrow receiver result

These values are passed as arguments to a function which is responsible for appropriate action.

4.6 MOUSE MOVEMENT ON SCREEN

Below picture is the overlapped version of three pictures where cursor is at different positions at different instants of time and we can interpret the cursor movement from the below picture.

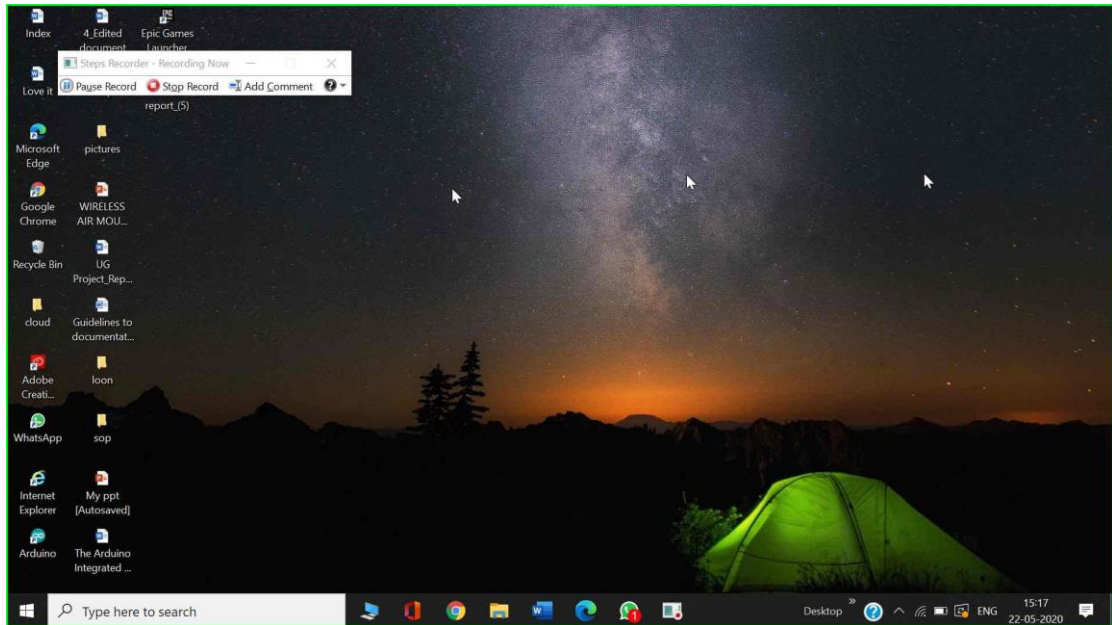


Fig.4.11 Mouse movement on screen

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

Using Wireless air mouse, it will be able to control the cursor movement remotely. It overcomes the difficulties available while using optical mouse. It is compact and don't need any surface for mounting. It can be operated from any corner within a room i.e., it is portable. It facilitates all the functionalities such as left click, right click and long press that are being achieved by a wired mouse. In addition to this, keyboard functions like up arrow and down arrow can also be implemented, which are used during the power point presentation. Making LCD projector ON/OFF also can be implemented. The above functions are implemented with a single handheld device.

The wireless communication is made possible with RF (Radio Frequency) module. Arduino Leonardo is used to support USB functionalities. ATMEGA328P-PU, Gyroscope sensor and RF transmitter are used at the transmitter side. Arduino Leonardo and RF receiver are used at the receiver side. With the help of Printed Circuit Board (PCB), the size and complexity of the circuit is reduced. The circuit can be made even simpler and small using SMD and 3D printing.

5.2 FUTURE SCOPE

In our proposed system the gesture-based approach is carried out. The position-based approach can be tried in the future, where in the mouse cursor just follows the movement of the finger. The proposed system can be applied in digital classrooms, seminar halls, conferences, etc. It can also be used as a writing aid for paralyzed people. This can be made into a compact device which can be carried in the form of touch free anywhere and used for any purpose. This system can be further improved by increasing the processing speed and be implemented. Implementation of high-level gesture language.

Simple low cost, low power inertial sensor-based mouse with wireless capability will provide ease of use. It can be converted to be useful in 3-d gaming application. It also provides health benefits by preventing problem of carpal tunnel syndrome caused due to the use of keyboard and mouse. The above project can be developed in future to move the cursor using the movement of eyes. The EEG waves (brain waves) can also be used for cursor movement.

REFERENCES

- [1] Sohail Ahmed, Mohammed Abdullah Zubair, Irshad Basha Shaik. “Accelerometer based wireless air mouse using Arduino micro-controller board.” Global Conference on Communication Technologies (GCCT), 23-24 April 2015.
- [2] Kesthara V, Niveditha Rati Banakara, Pragati, Priyadarshini G N, Vanishree S V. “Design and Implementation of Air Mouse using Accelerometer Sensor.” International Journal of Advanced Research in Computer and Communication Engineering(IJARCCE), Vol. 7, Issue 4, April 2018.
- [3] Sujata Ahire, Yogesh Borse, Punam Deore. “Wireless gesture-based mouse.” International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Volume 4, Issue 2, February 2015.
- [4] Lluís Ribas-Xirgo and Francisco López-Varquiel. “Accelerometer-Based Computer Mouse for Special Needs People.” Journal of Accessibility and Design for All. Volume 7, Issue 1. (CC) JACCES, 2017.
- [5] Thomas P Rajan, Akhil C S, Ashik Sulaiman, Thomas P J. “Hand Controlled Mouse Pointer: A Typical Wireless Handy Device.” International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering (IJAREEIE), Vol. 7, Issue 4, April 2018.
- [6] R.S.Mathu Bala, R.Gokulapriya, E.Anisha and R.Anupriya. “Gesture Based Wireless Air Mouse using Accelerometer.” Asian Journal of Applied Science and Technology (AJAST), Volume 1, Issue 1, February 2017.

APPENDIX A

CODE OF THE THESIS

TRANSMITTER CODE

```
#include <VirtualWire.h>
#include<Wire.h>

int buttonstate=0;

int Right = 11;
int Left = 10;
int left_arrow = 8;
int right_arrow = 7;
long int i = 0;
char a,b;

const int MPU_addr=0x68;// 12C address of th
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup()
{
  // Initialize the I0 and ISR

  pinMode (Right, INPUT);
  pinMode (Left, INPUT);

  pinMode (left_arrow,INPUT);
  pinMode (right_arrow, INPUT);

  vw_set_tx_pin(2);

  Wire.begin();
  Wire.beginTransmission (MPU_addr);
  Wire.write (0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission (true);
  vw_setup (4000); // Bits per sec
  Serial.begin (9600);
}

void loop()
{
  i=0;
  Wire.beginTransmission (MPU_addr);
  Wire.write (0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission (false);
  Wire.requestFrom (MPU_addr, 14, true); // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read(); // 0x 3B (ACCEL_XOUT_H) : 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) s 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); // 0x3F (RCCEL_ZOUT_M) 0x40 (ACCEL_ZOUT_I)
```

```

Tmp=Wire.read()<<8|Wire.read();// 0x41 (TEMP OUT) 50x42 (TEMP_OUT_L!
GyX=Wire.read()<<8|Wire.read();//0x43 (GYRO XOUT # $ 0x44 (GYRO XOUT L)
GyY=Wire.read()<<8|Wire.read();// 0x45 (GYRO YOUTH) 50x46 (GYRO YOUT L)
GyZ=Wire.read()<<8|Wire.read();// 0x 47 GYRO_20UT_$) 50x45 (GYRO_20UI_L)

GyZ = -GyZ/300;
GyX = GyX/280;
if(GyX ==-1)
    GyX =0;

char cGyZ[4];
    itoa(GyZ,cGyZ,10);

char cGyX[4];
    itoa(GyX,cGyX,10);


char dest[6];
strcpy (dest, cGyZ);
strcat (dest, "H");
strcat (dest, cGyX);
strcat (dest, "A");

if(digitalRead (Right)==HIGH)
{
    strcat (dest, "r");

}
if(digitalRead (Left)==HIGH)
{
    strcat (dest, "s");

}
if(digitalRead(right_arrow) == HIGH)
{
    strcat (dest, "t");

}
if(digitalRead(left_arrow) == HIGH)
{
    strcat(dest, "i");

}


Serial.print (GyZ);
Serial.print (" ");
Serial.print (GyX);
Serial.print ("\t");
Serial.println(dest);
vw_send( (uint8_t *) dest, strlen (dest));
vw_wait_tx ();

}

```

RECEIVER CODE

```
#include<Keyboard.h>
#include<Mouse.h>
#include<VirtualWire.h>
//int 1:
int k = 0;
int n=0;
char a, b,c;
void setup ()
{
  //Serial.begin (9600); // Debugging only
  vw_setup (4000); // Bits per sec
  vw_rx_start (); // Start the receiver PLL running
  Keyboard.begin();
  Mouse.begin();
}
void loop ()
{
  uint8_t buf [VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  char data[VW_MAX_MESSAGE_LEN] = {'0'};
  char X[VW_MAX_MESSAGE_LEN] = {'0'};
  char Y[VW_MAX_MESSAGE_LEN]='0';

  if (vw_get_message (buf, &buflen)) // Non-blocking
  {
    // Message with a good checksum received, dump it.
    for(int i = 0; i < buflen; i++)
    {
      data[i] = (char)buf[i];
    }
    for(int j = 0; j < buflen; j++)
    {
      if(data[j] != 'H'){

        X[j] = data[j];
      }
      else

        break;
    }
    for (k=0;data[k] !='A';k++)
    {
      if (data[k]=='H')
      {
        for (n=k+1;data [n] !='A';n++)
```

```

{
Y[n-k-1]=data [n];
}
break;
}
}

for (int m = 0; m<buflen; m++)
{
if(data[m] == 'A')
{
a=data [m+1];

break;
}
}
int xAxis = 0;
xAxis =atoi (X) ;
int yAxis = 0;
yAxis =atoi (Y);

Serial.print (data);
Serial.print ("\t");

Serial.print (X);
Serial.print ("\t");
Serial.print (Y) ;

Serial.print ("\t");
Serial.print (xAxis);
Serial.print ("\t");
Serial.print (yAxis);
Serial.print ("\t");
Serial.println (a);

Mouse.move (xAxis, yAxis);
if(a!=c)
{
if(a=='r')
{
Mouse.click(MOUSE_RIGHT);
delay (200);
}
if(a=='s')
{
Mouse.click (MOUSE_LEFT);
delay (200);
}
}
}
}

```

```

}
if (a=='h')
{
Mouse.press (MOUSE_LEFT);
delay (200);
}
if (a=='i')
{
Keyboard.press (0xD8) ;
delay (200) ;
Keyboard.releaseAll ();
}
if(a=='t')
{
Keyboard.press (0xD7) ;
delay (200);
Keyboard.releaseAll ();
}
}
c=a;
}
.

```