Palestine Polytechnic University

College Of Information Technology and Computer Engineering

Department of Computer Engineering

PC Remote Control

Team Members

Ahmad Hashim Titi

Amane Sharawneh

Supervisors

Eng. Islam Amar

Dr. Zain Salah

2022 - 2023

PC Remote Control

By:

Ahmad Hashim Titi

Amane Sharawneh

Supervised By:

Eng. Islam Amar

Dr. Zain Salah

Contents

List of Figures

List of Tables

Abstract:

This project aims to design an infrared remote control that controls the PC, it was found as an alternative for Keyboard and Mouse in **<u>some</u>** cases. With the remote, you have full control over the PC, you can control the mouse and move it in all directions, and you can open any application, increase or decrease the volume, or even shut the computer down using the remote. We will assign a task for each button and explain it in later sections of the document. The project requires an Infrared remote, infrared receiver, Arduino Nano, ESP32, and a gyroscope.

It's planned that the Infrared receiver will be connected to the Computer directly using a USB cable, and the Air Mouse (ESP32 & Gyroscope MPU6050) will be connected to the PC through a Bluetooth adapter. Note that the system will be designed for **LG** Infrared remotes. Any other brands can be programmed to control the PC, but in our system we're planning to program an **LG** remote.

Chapter 1: Introduction

1.1 Overview

In our project, we're planning to make an IR Remote Controller that can control computers. You will be able to control the cursor using hand gestures (By moving the gyroscope in all directions). And do actions on the PC by pressing different remote buttons.

For example, some buttons will work for turning up/down the volume, some buttons will start the browser, some buttons will open up the win menu, etc.

1.2 Motivation

With this IR Remote you will be able to control the PC screen easily without difficulties, if you were laying on the couch a distance away from your PC, you'll still be able to control it however you want. Plus that you won't need a surface to run the mouse on.

The product will give you the freedom to control your PC like you're controlling a TV.

1.3 Project Objectives

The project aims to:

1. Replace the traditional mouse and keyboard in some cases.
2. Save desk space.
3. Reduce the clutter caused by wired mouse and keyboard.
4. Improving productivity and introducing multitasking.

1.4 Problem Statement

This product gives you the ability to control the screen by a remote control when you’re not able to use the mouse and keyboard. If you’re lying on the couch or standing in the middle of the room, you don’t have to find a surface to sweep the mouse on it. But you have a remote control that you can hold with one hand and control the screen with it.

Chapter 2: Background

2.1 Introduction

To control the PC using a remote, you will need a transmitter, which comes with the remote control, and a receiver which has to be connected to the computer, this requires Arduino components. In this chapter I will present a background of the project, and some of the formal products of other developers.

2.2 Theoretical Background

1. Arduino:

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices enabling users to create interactive electronic projects. [1]

2. Infrared Technology:

IR wireless is the use of wireless technology in devices or systems that convey data through infrared (IR) radiation. Infrared is electromagnetic energy at a wavelength or wavelengths somewhat longer than those of red light. [2]

2.1 Infrared Remote Control:

The dominant remote-control technology in home-theater applications is infrared (IR). Infrared light is also known as plain-old "heat." The basic premise at work in an IR remote control is the use of light to carry signals between a remote control and the device it's directing. Infrared light is in the invisible portion of the electromagnetic spectrum. An IR remote control (the transmitter) sends out pulses of infrared light that represent specific binary codes. These binary codes correspond to commands, such as Power On/Off and Volume Up. The IR receiver in the TV, stereo or other device decodes the pulses of light into the binary data (ones and zeroes) that the device's microprocessor can understand. The microprocessor then carries out the corresponding command. [3]

4. Programming Language Used:

The programming language used by the Arduino NANO is the C++. The Arduino NANO IDE has a well-defined function for each task that is easy to remember, for the remote control, we used a C++ code to program the receiver to receive signals from the remote and decode these signal, an additional Python coded application is used to program functions for each button.

For the Air Mouse, C++ was used in Arduino IDE environment as will, by using different useful libraries.

5. Air Mouse Technology:

An air mouse is a computer mouse that controls the cursor using motion-sensing technology and does not require a desk. You can control the cursor by waving the mouse in the air as if you were pointing to where you want the cursor to go. For example, a person doing a presentation may use an air mouse to control a cursor and the presentation while standing.
In addition to controlling the mouse pointer while standing, an air mouse often has additional buttons to help control the presentation slides. One button may be used to go back to the previous slide, and another button to go forward to the next slide.

Our Remote should be able to do the tasks mentioned above with ease. [4]

2.3 Conceptual Design

This section describes the concept of the project. Simplified, input and output.

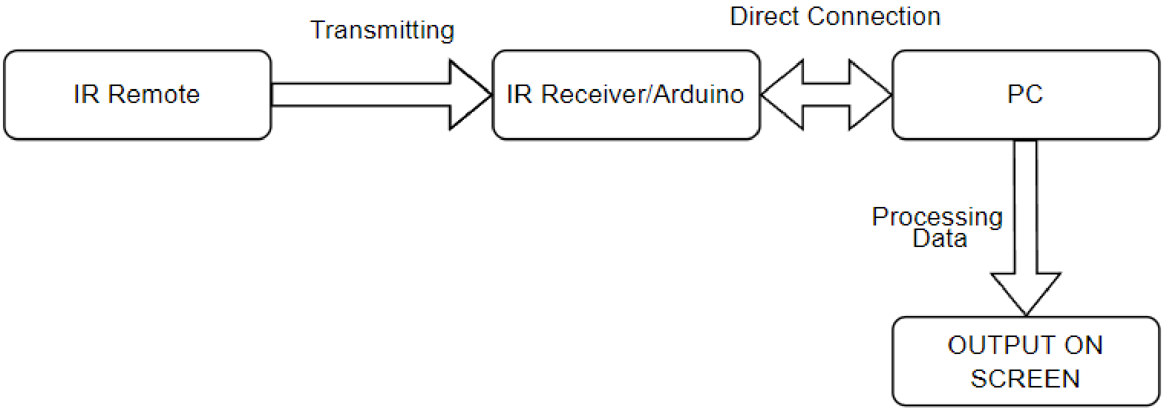


Figure 2.1: PC Remote Conceptual Design

1. Remote: LG Infrared Remote that sends data to the receiver.
2. Receiver: IR receiver connected to the PC, attached to Arduino components.
3. The PC will receive the signals from the remote and perform actions on the screen.

This section describes the concept of the air mouse. Simplified.

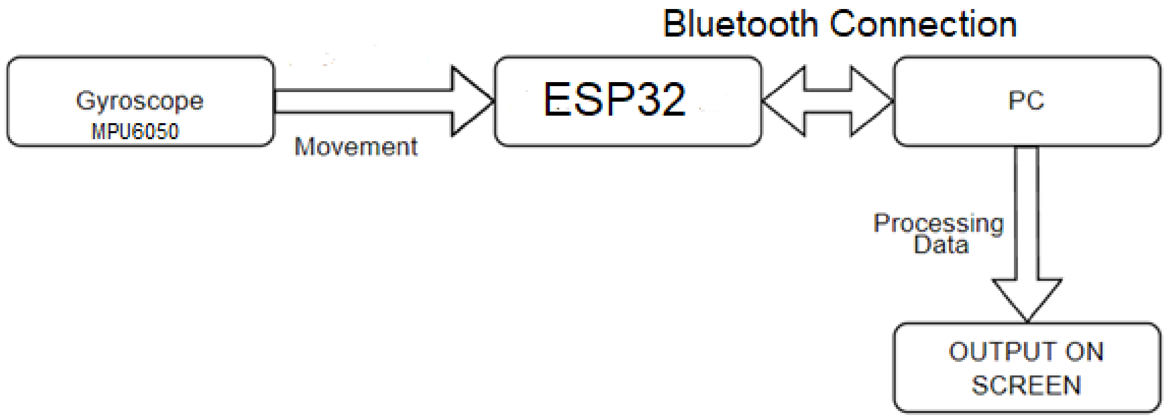


Figure 2.2: Air Mouse Conceptual Design

Gyroscope: Accelerometer sensor, used to control the mouse movement. Connected to an ESP32. When you move the gyroscope, you will be able to control the mouse movement on the screen.

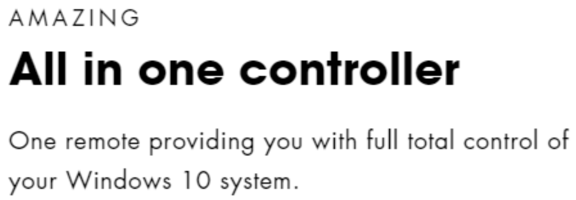
2.4 Literature Review

Some of the previous projects that inspired me.

1- In 2009 a similar project to this one was applied by 2 Computer Engineering students at Al-Najah University. **Adham Al-Dwiek** & **Ibrahim Al-Adham** both worked on a PC remote control project under the supervision of **Dr. Luai Malhis**. They designed the hardware circuit using a *PIC18F6420 microcontroller* and programmed the circuit using PIC C compiler. In the software part, they used *C#* language to develop their product's software user interface on Windows. With the ability to adapt to any infrared remote control, which means that you can use any remote with frequency 38 kHz. [6]

2- W10 GYRO Smart Remote

W10 GYRO is the world's 1st and only 6-axis gyro air mouse designed specifically for the Windows 10 system.



W10 GYRO Smart Remote

All shortcut buttons and hotkeys are fully optimized for Windows 10. Utilizing 2.4GHz wireless technology with a USB receiver, W10 GYRO can be operated without any need for a manual driver installation. Simply plug in the bundled USB receiver to your Windows 10 based PC and you're all set and ready to enjoy the amazing control experience from the comfort of your own sofa. W10 GYRO is equipped with backlit LED which means you don't have to worry about not seeing the keys while using it in the dark. The operating distance of W10 GYRO is up to 10 meters from your computer. In addition, W10 GYRO comes with TV remote learning features on the front side, allowing it to learn up to 34 keys from your TV IR remote! [7]

W10 GYRO Smart Remote

3- Warren Parsons' DIY USB Receiver.

Warren Parsons is a Canadian programmer who worked in 2021 on the same idea of programming an IR remote that can control the PC. Using HTPC + WinLIRC, he made his own USB IR Receiver for his PC that helped him control anything on his PC using any 38 kHz remote.

The items that Warren used in his project are very similar to the items I am using in mine. An Arduino Nano (using an ATmega328P w/ USB) and a SM0038 IR Receiver. [8]

4- WeChip Air Mouse

2.4GHz wireless keyboard and mouse combo, 6-Axis inertia sensors and infrared remote control. Air mouse with keyboard. [9]

WeChip 2 in 1 Air Mouse & Keyboard

Chapter 3 System Design:

In this chapter we're discussing the system needed components and the system's design.

3.1 System Design

Figure 3.1 below shows how the hardware components are planned to be designed:

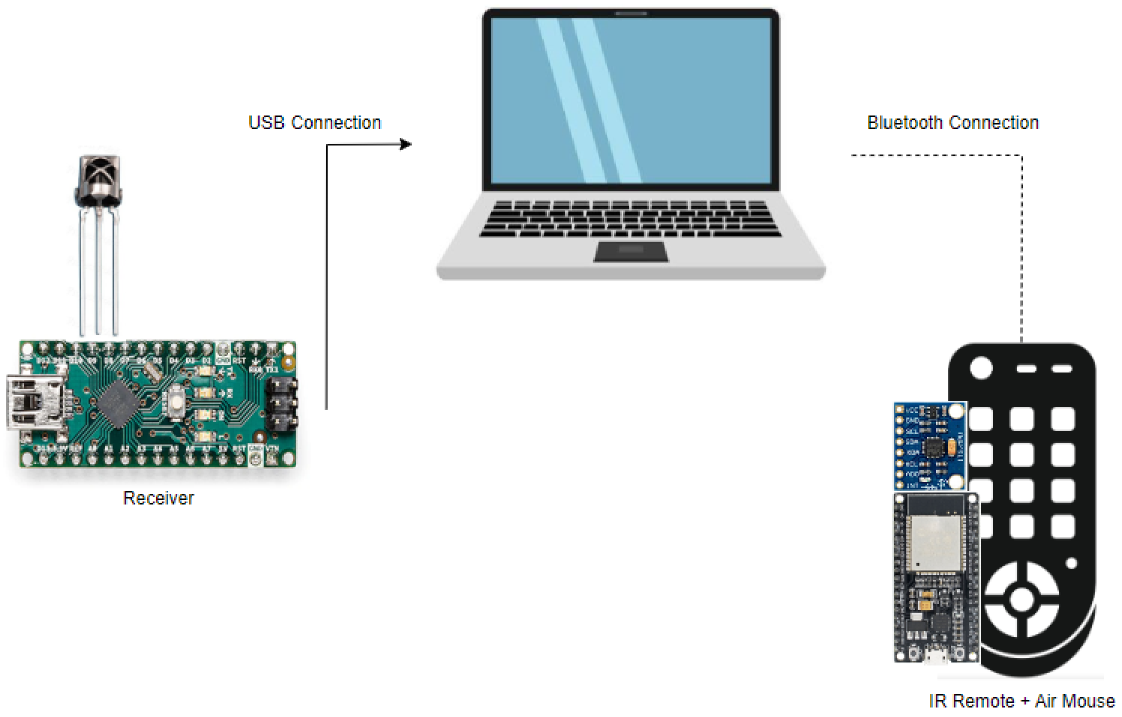
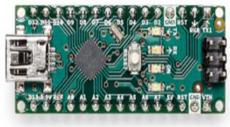
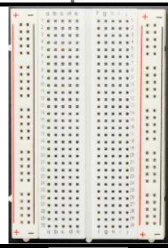
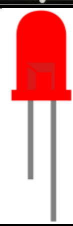


Figure 3.1: System Design

The figure shows two parts, part 1 shows an IR TSOP sensor connected to an Arduino Nano which is responsible of receiving data from the LG infrared remote, this part is connected to the PC through USB. Part 2 shows a gyro sensor connected to an esp32 which is responsible of controlling the mouse movement, part 2 will be paired to a Bluetooth adapter connected to the PC.

3.2 System Hardware Components.

Table 3.1: System Hardware Components

Component	No. of pieces	Image	Component	No. of pieces	Image
Arduino Nano	1		LG IR remote	**1**	
MPU-6050	1		USB Cable	**2**	
TSOP VS1838b universal Infrared receiver	1		ESP32	1	
Bluetooth 4.2 USB adapter	1		Breadboard Mini	**2**	
5V lithium battery	**1**		LED	1	

3.2.1 Arduino Nano [10]

The **Arduino Nano** is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

Figure 3.2: Arduino Nano

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27)

The power source is automatically selected to the highest voltage source.

The ATmega328 has 32 KB, (also with 2 KB used for the bootloader. The ATmega328 has 2 KB of SRAM and 1 KB of EEPROM

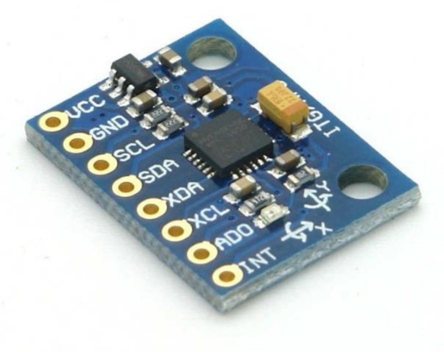
The Table below shows the full specifications of Arduino Nano

Table 3.2: Arduino Nano Specifications

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

3.2.2 MPU6050 Module

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. Also, it has additional feature of on-chip Temperature sensor. It has I2C bus interface to communicate with the microcontrollers.



MPU6050 Sensor

It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer, Pressure sensor etc.

If 3-axis Magnetometer is connected to auxiliary I2C bus, then MPU6050 can provide complete 9-axis Motion Fusion output. [11][17][18][21]

MPU6050 inside sensors.

- When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a MEM inside MPU6050.

- The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate.

- This voltage is digitized using 16-bit ADC to sample each axis.

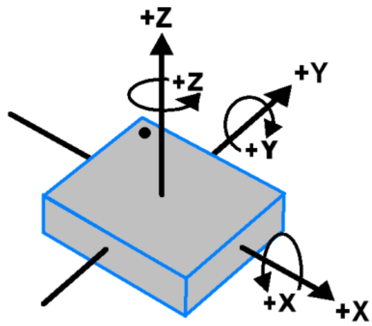


Figure 3.3

MPU-6050 Orientation & Polarity of Rotation

- The full-scale range of output are +/- 250, +/- 500, +/- 1000, +/- 2000.

- It measures the angular velocity along each axis in degree per second unit.

3-Axis Accelerometer

The MPU6050 consist 3-axis Accelerometer with Micro Electro Mechanical (MEMs) technology. It used to detect angle of tilt or inclination along the X, Y and Z axes as shown in below figure.

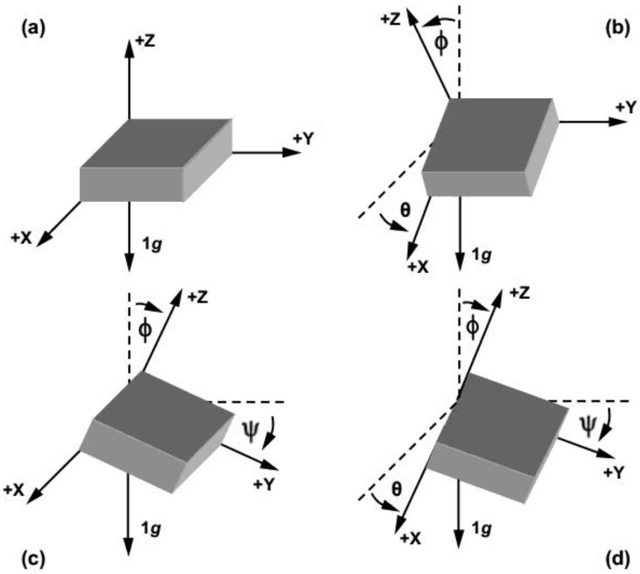


Figure 3.4: 3-Axis Gyro Accelerometer movement

- Acceleration along the axes deflects the movable mass.

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

- 16-bit ADC is used to get digitized output.

- The full-scale range of acceleration are +/- 2g, +/- 4g, +/- 8g, +/- 16g.

- It measured in g (gravity force) unit.

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

DMP (Digital Motion Processor)

The embedded Digital Motion Processor (DMP) is used to compute motion processing algorithms. It takes data from gyroscope, accelerometer and additional 3rd party sensor such as magnetometer and processes the data. It provides motion data like roll, pitch, yaw angles, landscape and portrait sense etc. It minimizes the processes of host in computing motion data. The resulting data can be read from DMP registers.

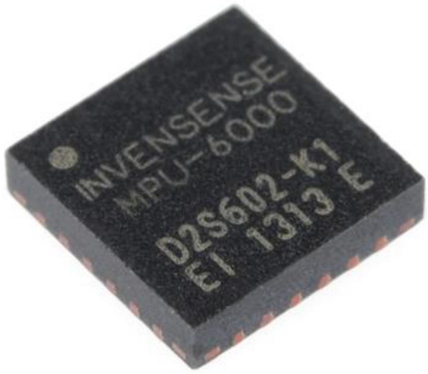


Figure 3.5: Digital Motion Processor

MPU-6050 Module

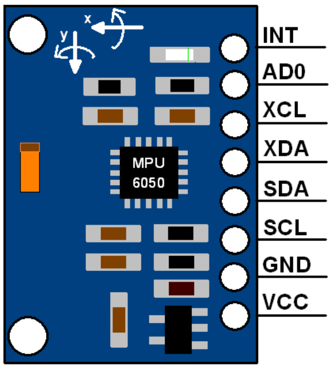


Figure 3.6: MPU-6050 module

The MPU-6050 module has 8 pins:

INT: Interrupt digital output pin.

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

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

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

SCL: Serial Clock pin. Connect this pin to microcontrollers SCL pin.

SDA: Serial Data pin. Connect this pin to microcontrollers SDA pin.

GND: Ground pin. Connect this pin to ground connection.

VCC: Power supply pin. Connect this pin to +5V DC supply.

MPU-6050 module has Slave address (When AD0 = 0, i.e. it is not connected to Vcc) as,

Slave Write address(SLA+W): 0xD0

Slave Read address(SLA+R): 0xD1

3.2.3 TSOP Infrared Receiver VS1838B

Figure 3.7: TSOP infrared receiver vs1838B

VS1838 includes high-speed high-sensitivity PIN photodiode and a low-power, high-gain Preamplifier IC, using epoxy plastic package design, the product has passed REACH and SGS certified as environmentally friendly products, as in the infrared remote control system receiver Uses. There is a commonly used IR receiver, you can use it with the Infrared Remote Control to build your remote control project. It is easy to use and low cost. It mates well with embedded electronics and can be used with common IR remotes. [12]

Features:

Table 3.2: TSOP vs1838B features

Working Voltage	2.7V to 5.5V
Reception Distance	10-15 M
Reception Angle	±35 Degree
Low Level Voltage	0.4V
High Level Voltage	4.5V
4.5V	Alloy
Carrier frequency	38KHz

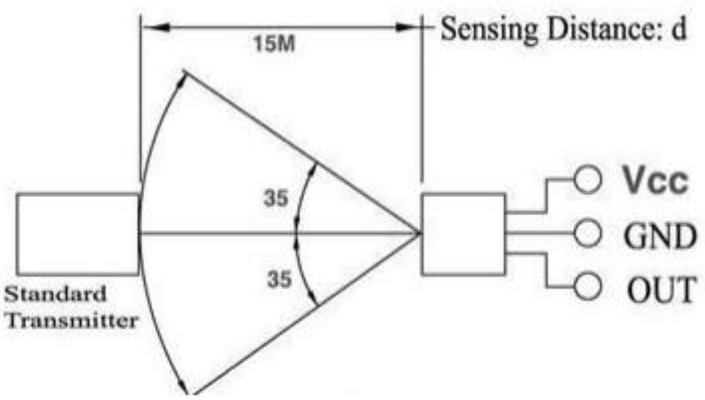


Figure 3.8: TSOP 1838B sensor sensing distance & sensing degree

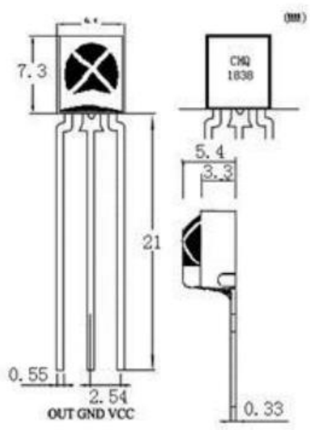


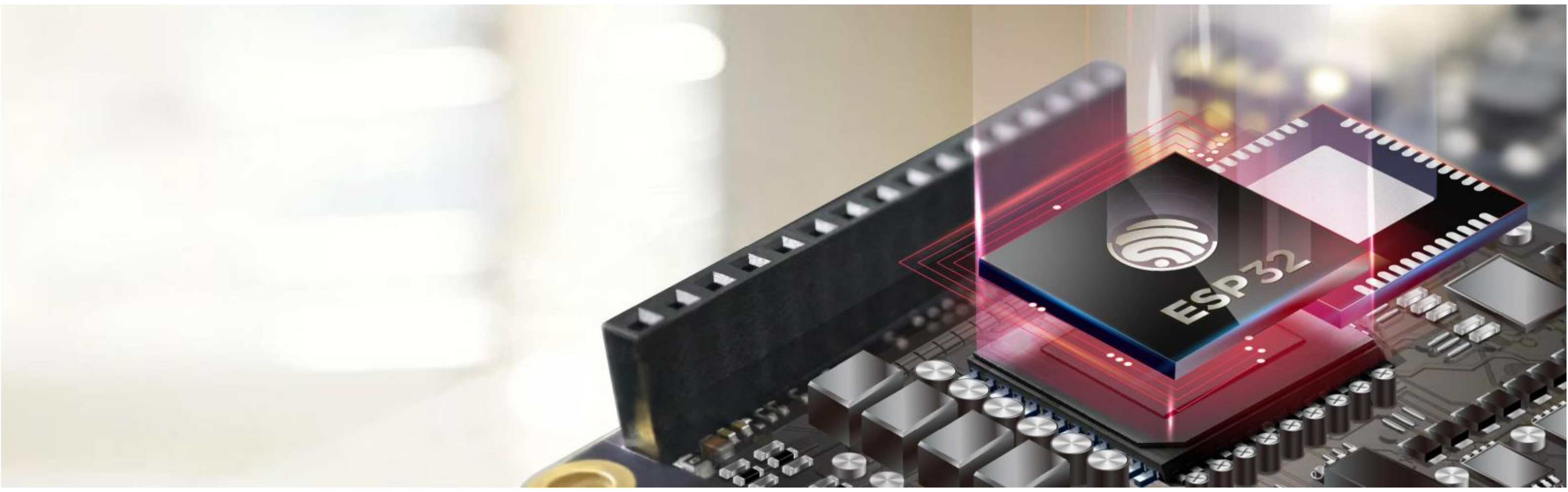
Figure 3.9: TSOP 1838B sensor pinout

Table 3.3: Recommended Conditions of Use for TSOP 1838B sensor

Recommended Conditions of Use

Project	Symbol	Min	Typ	Mnx	Unit
Operating Voltage	Vcc	2.7	-----	5.5	V
Input Frequency	FM		38		kHz
Operating Temperature	Topr	-20			

3.2.4 ESP Wroom32 [13]



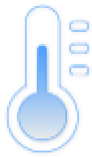
ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor and a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, and low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

Programming

- Arduino IDE with the ESP32 Arduino Core
- Espruino – JavaScript SDK and firmware closely emulating Node.js
- MicroPython (and CircuitPython) – lean implementation of Python 3 for microcontrollers
- Lua Network/IoT toolkit for ESP32-Wrover
- Mongoose OS – an operating system for connected products on microcontrollers; programmable with JavaScript or C. A recommended platform by Espressif Systems, AWS IoT, and Google Cloud IoT
- mruby for the ESP32
- NodeMCU – Lua-based firmware
- PlatformIO
- Visual Studio Code with the officially supported Espressif Integrated Development Framework (ESP-IDF) Extension
- Zerynth – Python for IoT and microcontrollers, including the ESP32

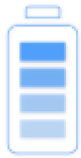
ESP32 specifications:

Robust Design



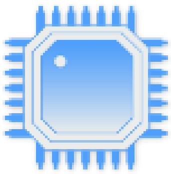
ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions.

Ultra-Low Power Consumption



Engineered for mobile devices, wearable electronics and IoT applications, ESP32 achieves ultra-low power consumption with a combination of several types of proprietary software. ESP32 also includes state-of-the-art features, such as fine-grained clock gating, various power modes and dynamic power scaling.

High Level of Integration



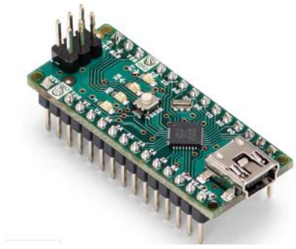
ESP32 is highly-integrated with in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 adds priceless functionality and versatility to your applications with minimal Printed Circuit Board (PCB) requirements.

Hybrid Wi-Fi & Bluetooth Chip

ESP32 can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.

3.2.5 ESP32, Arduino UNO & Arduino Nano comparison.

Table 3.4: Table of Comparison

#	Features	ESP32 [22] 	Arduino UNO [23] 	Arduino Nano[24] 
1	**Microcontroller**	Tensilica Xtensa LX6 microprocessor @ 160 or 240 MHz	ATmega168	ATmega328
2	**Voltage Usage**	2.2 – 3.6 V	5 V	5V
3	**Total Pins**	25	20	30 (6 PWM output)
4	**Clock Speed**	240 MHz	16 MHz	16 MHz
5	**SRAM**	250 Kbytes	2Kbytes	2Kbytes
6	**Flash Memory**	16 Mbytes	32 Kbytes	32Kbytes (2KB used by bootloader)
7	**Wi-Fi**	2.4 GHz	✗	✗
8	**Bluetooth**	✓	✗	✗

3.3 System software components

The best languages to match the way project components are used to create them and to get the best accuracy, outcomes, and performance for the project were sought after in the software section. We decided to use **C++** in Arduino environment as the language for programming the Air Mouse functions. We also used **C++** in Arduino environment for decoding the remote signals to get their serial codes. And the **Python** language for programming the application for the remote functions.

Choosing the best programming language

In order for the system to work as a whole, it was necessary to take into account the languages through which we wanted to connect the system's components. After conducting research, learning the appropriate language, and comparing it to other languages, it was found that the C++ language, was the best suitable language for programming the system's Air Mouse to get the best performance. And the Python programming language for programming the Remote functions due to the libraries Python has that helped us a lot. Let’s briefly talk about these languages.

C++ Programming Language: [19]

C++ is an object-oriented programming language that can identify both classes and objects. It is a flexible programming language with many potential applications. It can make games, browsers, and operating systems, among other things. It offers a variety of programming paradigms, including functional, procedural, object-oriented, etc. C++ is consequently robust and flexible. C++ is an old but still functional language. It is frequently used to create highly skilled gaming software and powerful applications.

Python Programming Language: [20]

Python is a powerful programming language that has automatic dynamic typing, the capability to dynamically bind different operations. Beginner programmers frequently use Python due to its straightforward syntax, well-organized packages, and plugins. Python's design philosophy makes extensive use of whitespace, which makes its code simpler to read.   Its object-oriented programming methodology ensures that programmers will receive assistance in writing clear, logical code for both complicated and straightforward applications.

The main reason we used Python for programming the remote functions is the high capabilities with GUI programming and the availability of useful libraries.

3.4 Differences between C++ & Java

Table 3.5: Table of Comparison between C++ & Python

C++	**Python**
Compiled Programming language	Interpreted Programming Language
Operator overload is supported.	Operator overload is supported.
Has a small number of library patrons	It includes a sizable library collection that makes it possible to use it for applications in data science, AI, and other fields.
The programming language C++ compilers quickly.	When an interpreter is used, execution is delayed.
Platform dependent	Platform independent
Syntax rules are strictly followed.	It isn’t necessary to use semicolon ‘;’

3.5 Libraries Used

Libraries play the main role in system software, they come with great benefits to ready-to-use codes, and with libraries you can save so much time and effort. In our system software we used the following libraries:

1. **ESP32 BLE Mouse** for Air Mouse using Gyroscope MPU6050 in Arduino. This library doesn't come with Arduino and has to be manually installed, the github link to this library will be provided in the references.

2. **IRremote** for decoding remote signals in Arduino.

3. **EspSoftwareSerial** for programming ESP32 on Arduino environment in Arduino.

4. **Adafruit_BusIO** for gyroscope sensor in Arduino.

5. **Pyserial** for programming remote functions in Python.

6. **Pyautogui** for programming remote functions in Python.

3.6 System Components Detailed Diagrams & Pinouts

3.6.1 Arduino Nano Pinout [14]

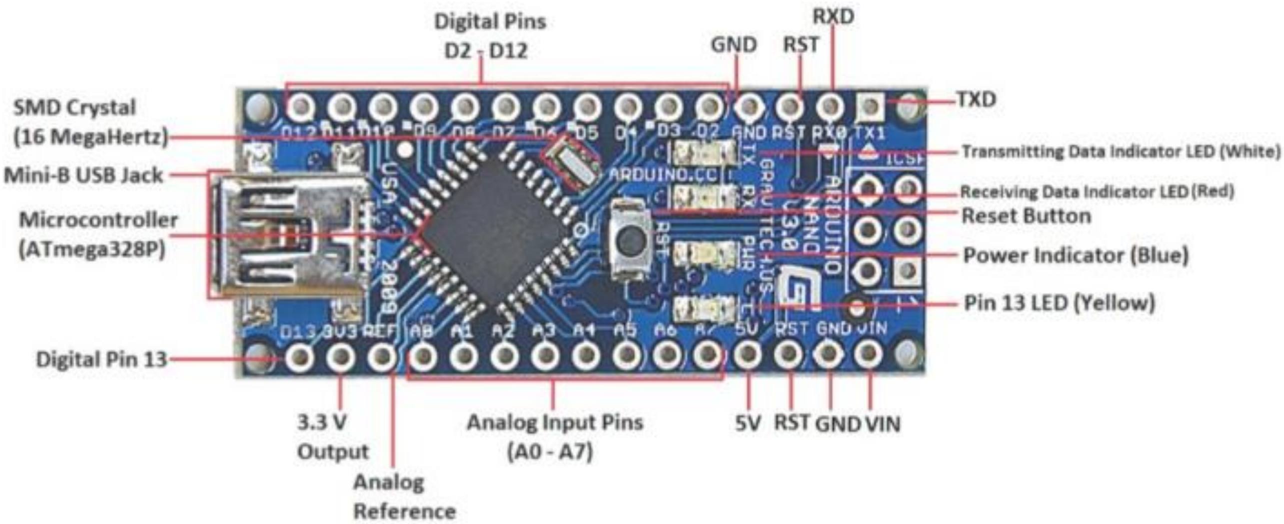


Figure 3.10: Arduino Nano Pinout

Arduino has 14 digital PINs for digital devices and 8 analog PINs for analog devices. The Table 3.1 below shows each Pin and information about it

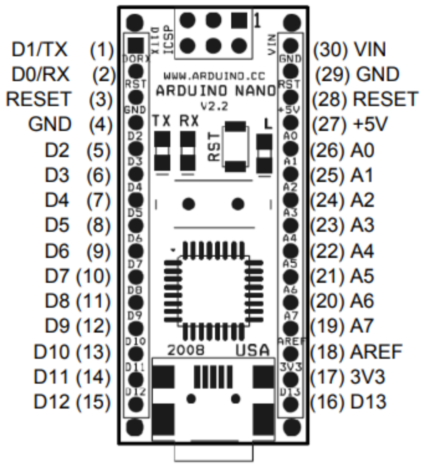


Figure 3.11: Arduino Nano Pinout Numbered

Table 3.5: Arduino Nano Pins and Description

Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

3.6.4 ESP32 Pinout [15]

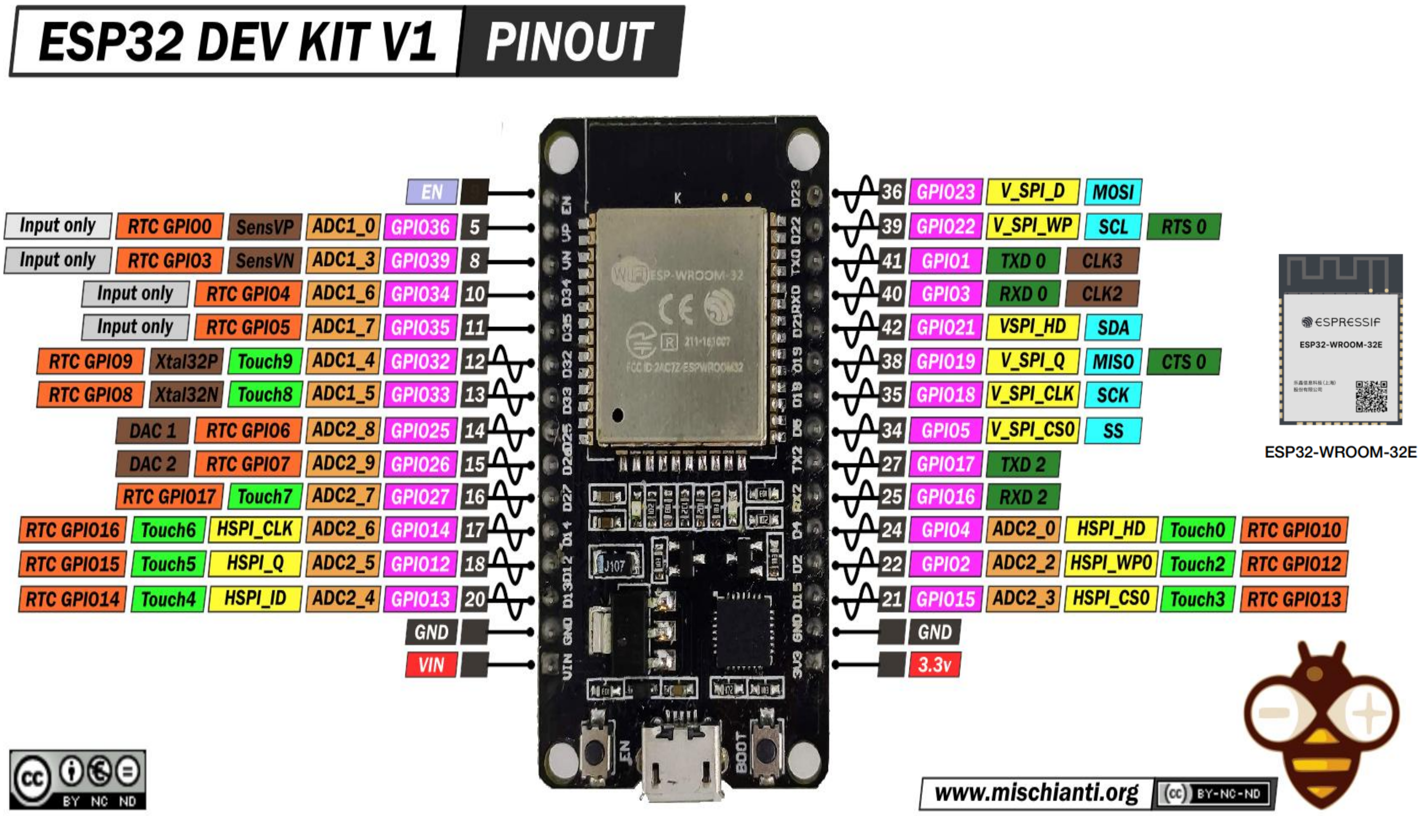


Figure 3.12: ESP WROOM32 Pinout

3.6.5 MPU-6050 Pinout [16]

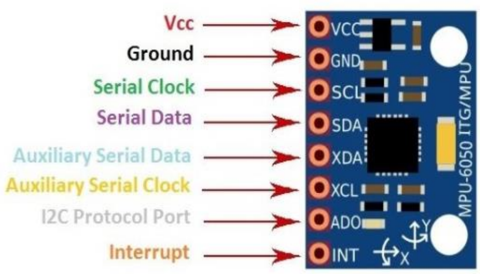


Figure 3.13: MPU-6050 pinout

3.6.6 Air Mouse Circuit Design

The following sketch shows the system design for the air mouse and connected to the PC via USB or Bluetooth.

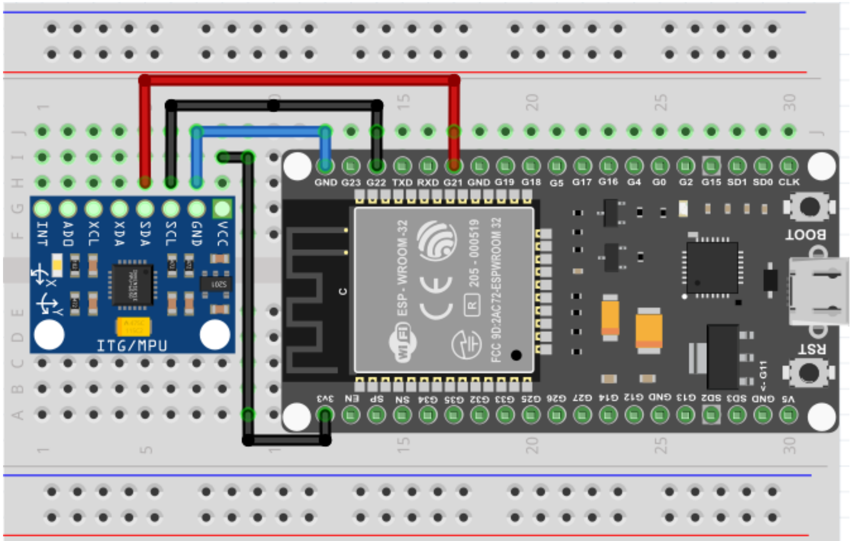


Figure 3.14: Air Mouse Circuit Design

3.6.2 IR Receiver Circuit Design:

Note: The circuit shows Arduino Micro. They're not different, Micro and Nano perform same operations.

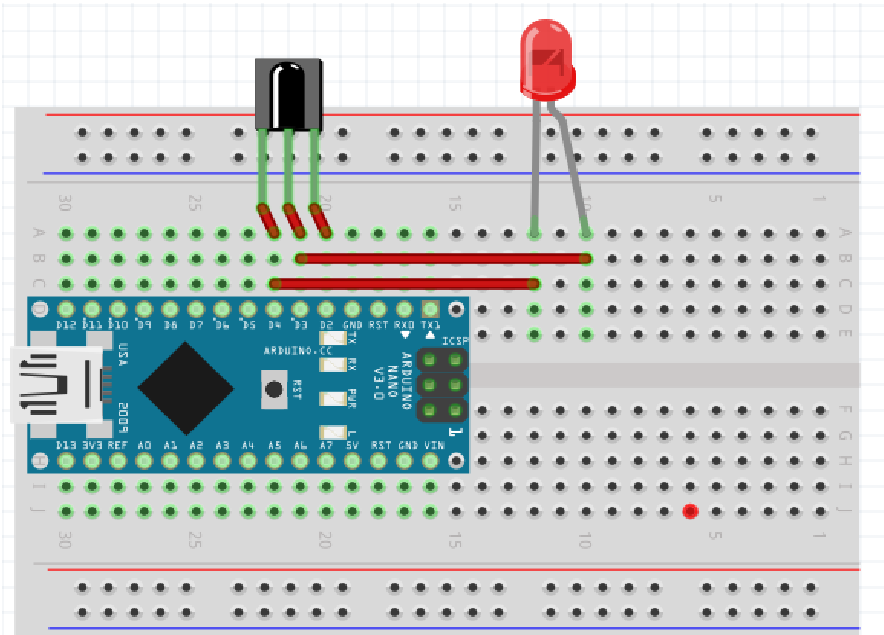


Figure 3.15: IR Receiver Circuit Design

3.6.7 Air Mouse Schematic Diagram

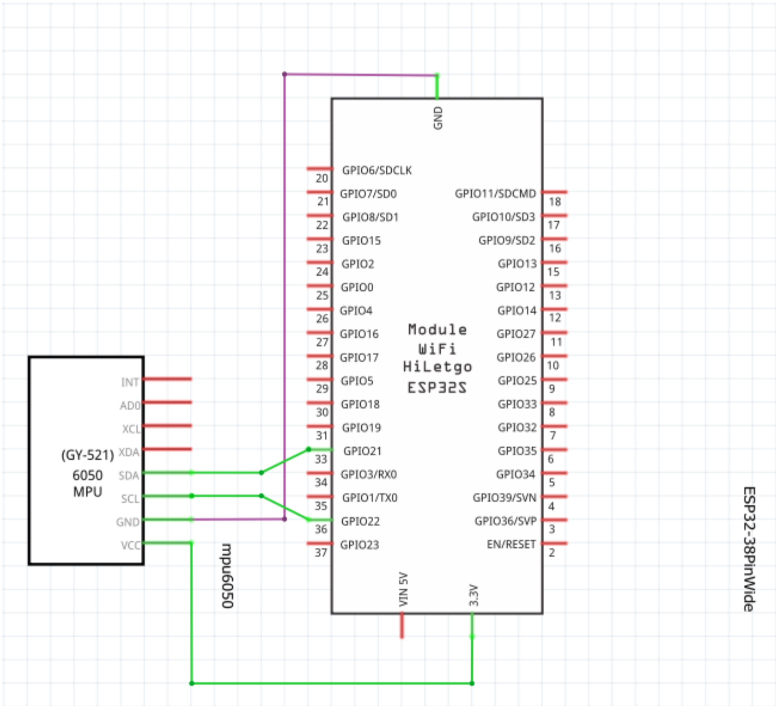


Figure 3.16: Air Mouse Schematic Diagram

Circuit Connections:

SDA is connected to GPIO21

SCL is connected to GPIO22

GND is connected to GND

VCC is connected to the 3.3V

3.6.3 Receiver Schematic Diagram

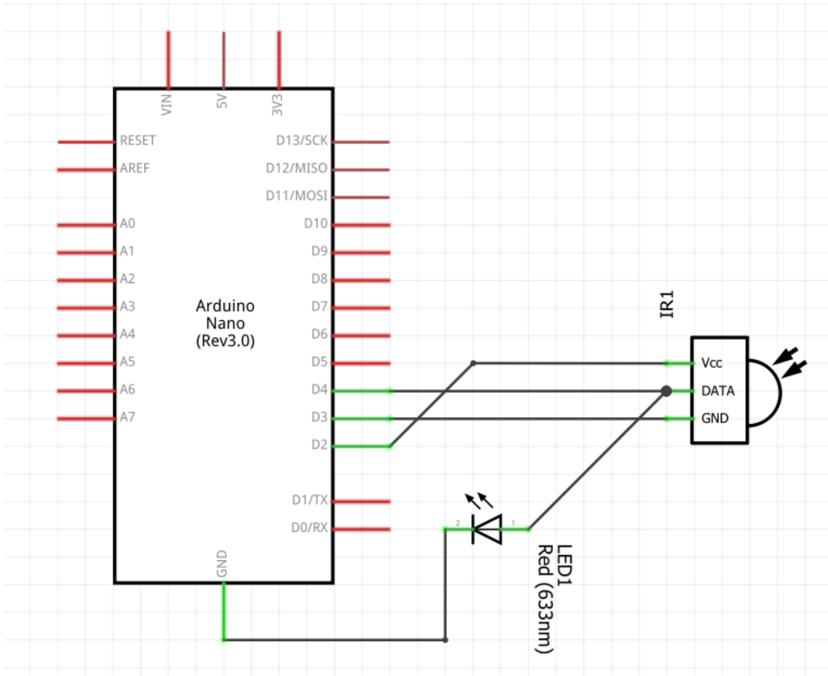


Figure 3.17: IR Receiver Schematic Diagram

Circuit Connections:

VCC is connected to D2

GND is connected to D3

OUTPUT is connected to D4

Led Cathode is connected to the Output

Led Anode is connected to the GND

3.7 Summary:

At the end of Chapter 3, we were able to describe the project's operation and the procedures. A decision on the parts to use and how to connect them was also made after viewing sources of each hardware components and checking libraries on different environments.

The initial setting of the system, we need a PC using Windows OS. The Arduino Nano should be connected to the PC via USB port, the Arduino Nano has an infrared receiver attached to it to receive signals from the remote and perform actions.

For Air Mouse, the sensor will be attached to the ESP32 and sticked to the remote so you can air control the mouse corsair as you move your hand in all directions.

Remote needed: Any spare LG remote will work, as long as the frequency is 38 kHz.

Chapter 4: System Implementation

4.1 Overview

We present the system implementation in this chapter. It is done by designing and implementing the receiver and sender as we mentioned earlier in chapter 3. We will describe the logic of the receiver and the logic of the sender.

4.2 The Receiver

In this section we will describe the receiver logic, connection, functionality, and how it works.

4.2.1 Receiver Functionality

The main functionality of the receiver is to catch the remote signal and send it to the computer to execute it, these operations also have to be in real time and synced between the remote press and action execution on the computer.

4.2.2 Receiver Logic (TSOP sensor)

1. The receiver has to connect to the computer and run it to start.

2. The user press a button on the remote control

3. The receiver will receive the signal from the remote, such that the remote sends the signal and the VS1838b universal IR receiver, which is connected to the Arduino, will receive this signal.

4. The Arduino will read the signal from the VS1838b universal IR receiver and converts this signal into an executable signal (action) according to the type of this signal.

5. Transfers the decoded signal to the computer (target device).

Figure 4.1 summarize the receiver logic steps in order, and figure 4.2 shows the hardware components of the receiver and how they’re connected with each other.

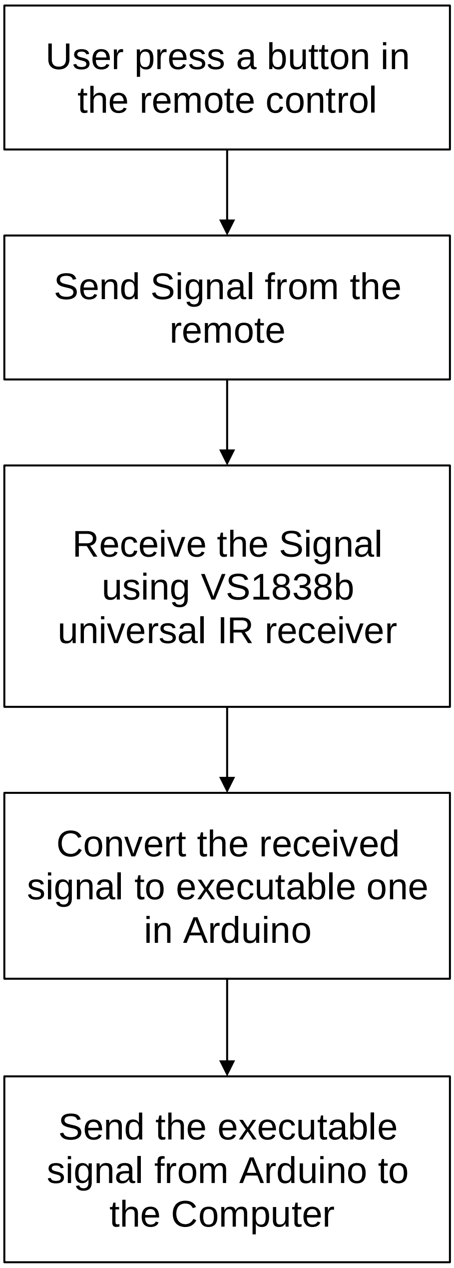


Figure 4.1: Receiver Logic

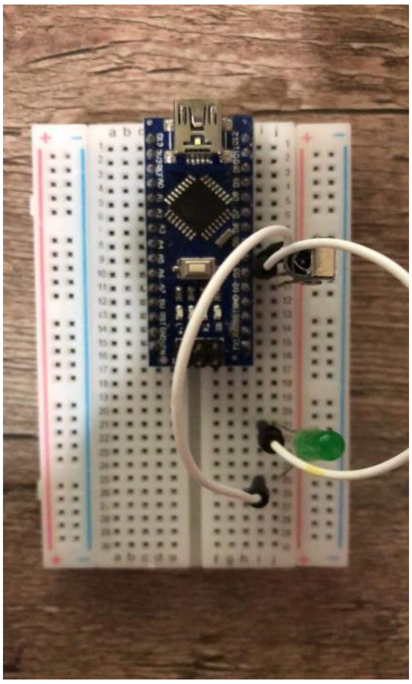


Figure 4.2: The Hardware components of the Receiver

4.3 The Sender

In this section we will describe the sender logic, we have 2 sending components, an IR remote and an MPU 6050 sensor (Air Mouse).

4.3.1 The Air Mouse Functionality.

The main functionality of the mouse is to control the movement of the corsair on the computer, this operation have to be in real time and synced between the mouse movement and action execution on the computer.

4.3.2 The Air Mouse Logic

1. The Mouse has to be connected to the computer via the Bluetooth and run it to start.

2. The use freely moves the mouse.

3. The MPU6050 sensor module detects the user movement and sends it to the ESP32 through I2C protocol to process the movement.

4. The ESP32 receives the movement from the MPU6050 and converts this movement into an executable signal.

5. The ESP32 will transfer the decoded signal to the computer via Bluetooth.

Figure 4.3 summarizes the (Air Mouse) logic steps in order, and figure 4.4 shows the hardware component of the Air Mouse and how connected with each other

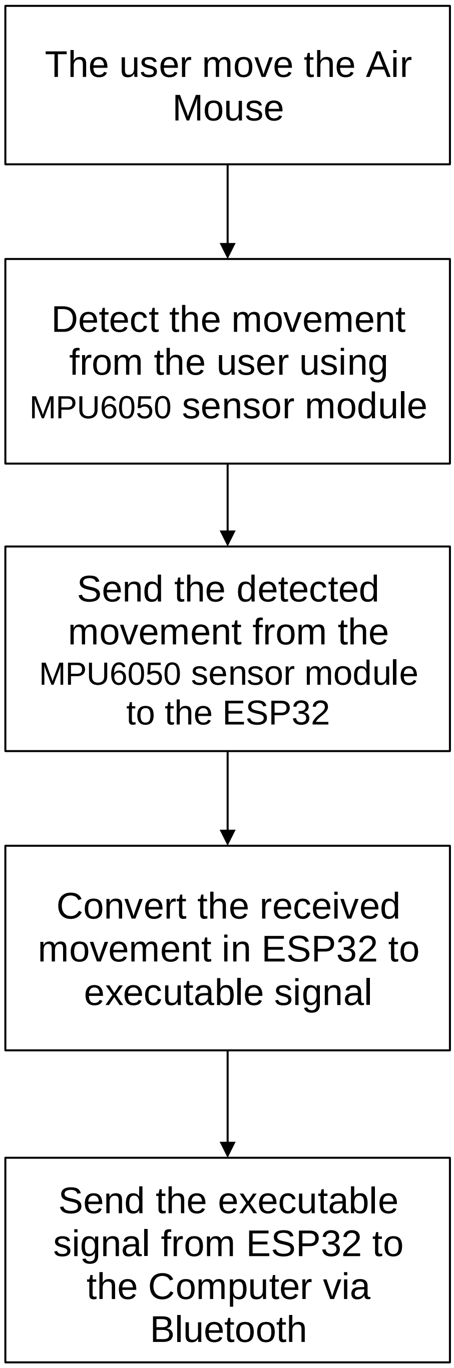


Figure 4.3: Air Mouse Logic

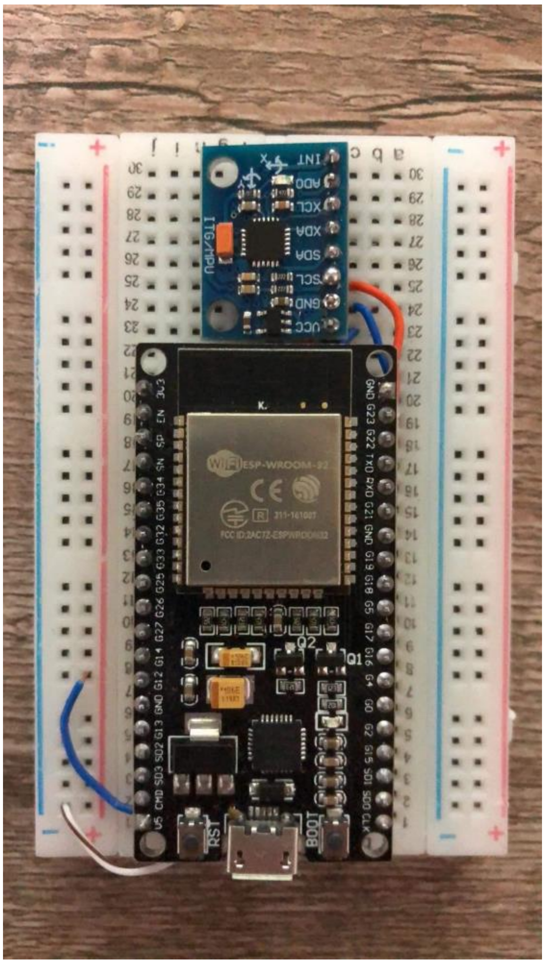


Figure 4.4: Air Mouse Hardware Components

4.4 System Testing

The system testing used to make sure that the system is achieved the target of it successfully or not, so in this section we represent the test result for the sender and receiver

4.4.1 Receiver Testing

We run the Arduino board and the computer to be ready for testing and then start testing the buttons to collect the serial bytes of each button.
The Table 4.1 shows each button and its serial bytes in Hexadecimal. No buttons duplicate the same serial

Table 4.1: The Remote Buttons and their Serials

#	Button	Serial
1	POWER	20DF10EF
2	HOME	20DF7E81
3	DISP	20DFCA35
4	RETURN	20DF14EB
5	EXIT	20DFDA25
6	MUTE	20DF906F
7	SOURCE	20DFD02F
8	CH +	20DF00FF
9	CH -	20DF907F
10	VOL +	20DF40BF
11	VOL -	20DFC03F
12	OK	20DF22DD
13	1	20DF8877
14	2	20DF48B7
15	3	20DFC837
16	4	20DF28D7

#	Button	Serial
17	5	20DFA857
18	6	20DF6897
19	7	20DFE817
20	8	20DF18E7
21	9	20DF9867
22	0	20DF08F7
23	YouTube	20DFD52A
24	Amazon	20DF3AC5
25	SLEEP	20DF708F
26	MENU	20DFC23D
27	-/--	20DF7887
28	Yellow Button	20DFC639
29	Arrow Right	20DF609F
30	Arrow Left	20DFE01F
31	Arrow Up	20DF02FD
32	Arrow Down	20DF827D
33	Amazon	20DF3AC5

After testing the receiver and collecting each button's serial, we can program these buttons to do certain actions using a python code.

Each button will be assigned with certain action to perform, table 4.2 will show the testing and the performed action of each button.

Table 4.2: The Remote Buttons and their Actions

#	Case	expected output	Obtained Output	status (pass/fail)
1	Press **POWER** button	Shutdown the computer	Shutdown the computer	pass
2	Press **HOME** button	Show the desktop	Show the desktop	pass
3	Press **DISP** button	The same behavior of click the right mouse button	The same behavior of click the right mouse button	pass
4	Press **RETURN** button	The same behavior of click the backspace button	The same behavior of click the backspace button	pass
5	Press **EXIT** button	The same behavior of click the alt + f4	The same behavior of click the alt + f4	pass
6	Press **MUTE** button	Mute the computer volume	Mute the computer volume	pass
7	Press **SOURCE** button	Do nothing	Do nothing	pass
8	Press **CH +** button	Page Up	Page Up	pass
9	Press **CH -** button	Page Down	Page down	pass
10	Press **VOL +** button	Turn up the computer volume	Turn up the computer volume	pass
11	Press **VOL -** button	Turn down the computer volume	Turn down the computer volume	pass
12	Press **OK** button	The same behavior of clicking the left mouse button	The same behavior of clicking the left mouse button	pass
13	Press **1** button	Ctrl + A	Ctrl + A	Pass
14	Press **2** button	Ctrl + C	Ctrl + C	Pass

#	Case	expected output	Obtained Output	status (pass/fail)
	Press **3** button	Ctrl + V	Ctrl + V	pass
	Press **4** button	Ctrl + X	Ctrl + X	pass
17	Press **5** button	Ctrl + Z	Ctrl + Z	pass
18	Press **6** button	Delete	Delete	pass
19	Press **7** button	Caps Lock	Caps Lock	pass
20	Press **8** button	Win + R	Win + R	pass
21	Press **9** button	Win + S	Win + S	pass
22	Press **NETFLIX** button	Open browser and visit Netflix.com	Open browser and visit Netflix.com	pass
23	Press **YouTube** button	Open browser and visit YouTube.com	Open browser and visit YouTube.com	pass
24	Press **Amazon** button	Open browser and visit Shahid.net	Open browser and visit Shahid.net	pass
25	Press **MENU** button	Open Windows Menu	Open Windows Menu	pass
26	-/--	The same behavior of click the spacebar button	The same behavior of click the spacebar button	pass
26	Press **Yellow** button	Open Chrome.exe	Open Chrome.exe	pass
27	Press **Right Arrow** button	Right Button	Right Button	pass
28	Press **Left Arrow** button	Left Button	Left Button	pass
29	Press **Up Arrow** button	Up Button	Up Button	pass
30	Press **Down Arrow** button	Down Button	Down Button	pass
31	Press **0** button	Enter	Enter	pass

4.4.2 Air Mouse Testing

We run the Arduino board and the computer to be ready for testing, then connect the ESP32 with the computer via Bluetooth. Start test the mouse movement by moving the Air Mouse and see what happened in the computer mouse. The result was seen as the following in table 4.3 below.

Table 4.3: The Mouse test Results

#	Case	expected output	Obtained Output	status (pass/fail)
1	Move the sender to the north	The computer mouse move to the north	The computer mouse move to the north	pass
2	Move the sender to the south	The computer mouse move to the south	The computer mouse move to the south	pass
3	Move the sender to the west	The computer mouse move to the west	The computer mouse move to the west	pass
4	Move the sender to the east	The computer mouse move to the east	The computer mouse move to the east	pass
5	Move the sender to the northeast	The computer mouse move to the northeast	The computer mouse move to the northeast	pass
6	Move the sender to the southeast	The computer mouse move to the southeast	The computer mouse move to the southeast	pass
7	Move the sender to the southwest	The computer mouse move to the southwest	The computer mouse move to the southwest	pass
8	Move the sender to the northwest	The computer mouse move to the northwest	The computer mouse move to the northwest	pass

4.5 Challenges and Issues.

This section represent the main challenges that we faced in this system and how we solved them. The main issues are the following:

1- Turning the PC on with the remote was a challenge we face because the receiver won't work unless:

a. The PC is turned on to power it.

b. The Python application (System Software) has to be running as long as you're using the Remote. If the Software was turned off, you can't control your PC.

2- Receiver changes behavior after turning off the PC/Laptop for more than a day, and it starts giving some random signals even without any key pressed on remote, after doing some research I reached out that it might be one of these reasons:

Possibility 1: remote sensor, may be TSOP1838 or so, is not working properly.
Possibility 2: if sensor is ok, there might be loose contacts
Possibility 3: there are other IR emitters active in the room

There're some ways to fix this issue:
1. Disconnect and reconnect the Arduino to check whether bytes are received properly.
2. Disconnect the Arduino, shut down the PC and turn it back on, reconnect the Arduino and use it to check whether bytes are received properly.
3. Make sure there're no remotes of other devices being used in the room.
4. Make sure there're no IR emitters working in the room, such as LED emitters.
5. If none of the steps above worked, disconnect the Arduino and change the position of the TSOP sensor on the breadboard.

Chapter 5: Conclusion and Future Upgrades

5.1 Future Upgrades:

1. Wireless Keyboard (possible future expansion):

A wireless keyboard is a computer keyboard that allows the user to communicate with computers, tablets, or laptops with the help of radio frequency such as Wi-Fi and Bluetooth or with infrared (IR) technology.
It is common for wireless keyboards available these days to be accompanied by a wireless mouse.
Wireless keyboards based on infrared technology use light waves to transmit signals to other infrared-enabled devices. But, in case of radio frequency technology, a wireless keyboard communicates using signals which range from 27 MHz to up to 2.4 GHz.
Most wireless keyboards today work on 2.4 GHz radio frequency. Bluetooth is another technology that is being widely used by wireless keyboards. These devices connect and communicate to their parent device via the Bluetooth protocol. A wireless keyboard can be connected using RF technology with the help of two parts, a transmitter and a receiver. The radio transmitter is inside the wireless keyboard. The radio receiver plugs into a keyboard port or USB port. Once the receiver and transmitter are plugged in, the computer recognizes the keyboard and mouse as if they were connected via a cable. [5]

In our project, we weren't able to apply the wireless keyboard technology to the remote due to the lack of resources. But it's always possible to add this technology to the remote in the future.

PC remote control with wireless Keyboard

5.2 Conclusion:

What we know is a drop, what we don't know is an ocean.

The idea of this project came from what I noticed to be a struggle that people have with controlling their PCs or Laptops when they want to connect them to a big TV screen to watch a football game or watch a movie, I myself struggle from this issue because I have a TV screen connected to my PC through an HDMI cable, and it's attached on the wall, I use it for watching YouTube, Football games or Netflix movies usually when it's bed time. Every time I want to change the movie or skip a video on YouTube I have to get up, do it and come back to bed. This can be annoying when it happens repeatedly, this life scenario stormed the idea of a TV IR remote controlling the PC.

I introduced this idea to my team member, and I thought to myself a lot, whether we will be able to apply it or not. And here we are today, the project met its goals successfully. We managed to build a PC Infrared Remote Control with Air Mouse that solved this problem.

With the great help of Arduino environment, we were able to use the minimal hardware requirements to build this project, we didn't struggle with finding and assembling codes due to the enormous libraries that it offers for all types of projects.

References:

[1] https://www.arduino.cc

[2] https://www.techtarget.com

[3] https://www.techtarget.com

[4] https://www.computerhope.com/jargon/a/air-mouse.htm

[5] Wikipedia

[6] https://eng-old.najah.edu/ar/graduation-projects/2760

[7] https://www.pepper-jobs.com/products/w10-gyro-smart-remote

[8] https://github.com/AdvancedNewbie/IRNanoLIRC

[9] http://www.wechipbox.com

[10] https://store.arduino.cc/products/arduino-nano

[11] https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module

[12] https://www.makerfabs.com/infrared-receiver-vs1838b.html

[13] https://www.espressif.com/en/products/socs/esp32

[14] https://docs.arduino.cc/hardware/nano

[15] https://www.mischianti.org/2021/07/17/esp32-devkitc-v4-high-resolution-pinout-and-specs/

[16] https://circuits-diy.com/accelerometer-and-gyroscope-sensor-module-mpu6050

[17] https://www.electronicwings.com/avr-atmega/mpu6050-gyroscope-accelerometer-temperature-interface-with-atmega16

[18] https://www.electronicwings.com/arduino/mpu6050-interfacing-with-arduino-uno

[19] https://en.wikipedia.org/wiki/C%2B%2B

[20] https://en.wikipedia.org/wiki/Python_(programming_language)

[21] https://www.electronicwings.com/esp32/mpu6050-gyroscope-interfacing-with-esp32

[22] https://www.electronicshub.org/getting-started-with-esp32/

[23] https://docs.arduino.cc/hardware/uno-rev3

[24] https://store.arduino.cc/products/arduino-nano & https://components101.com/microcontrollers/arduino-nano

Appendices

A. Air Mouse:

```
#include <BleConnectionStatus.h>
#include <BleMouse.h>
#include <Wire.h>
#include <SPI.h>
#include <SoftwareSerial.h>


uint8_t data[6];
int16_t gyroX, gyroZ;

int Sensitivity = 600;
int delayi = 20;

BleMouse bleMouse;

uint32_t timer;
uint8_t i2cData[14];

const uint8_t IMUAddress = 0x68;
const uint16_t I2C_TIMEOUT = 1000;

uint8_t i2cWrite(uint8_t registerAddress, uint8_t* data, uint8_t
length, bool sendStop) {
  Wire.beginTransmission(IMUAddress);
  Wire.write(registerAddress);
  Wire.write(data, length);
  return Wire.endTransmission(sendStop); // Returns 0 on success
}

uint8_t i2cWrite2(uint8_t registerAddress, uint8_t data, bool sendStop)
{
  return i2cWrite(registerAddress, &data, 1, sendStop); // Returns 0 on
success
}

uint8_t i2cRead(uint8_t registerAddress, uint8_t* data, uint8_t nbytes)
{
  uint32_t timeOutTimer;
  Wire.beginTransmission(IMUAddress);
  Wire.write(registerAddress);
  if(Wire.endTransmission(false))
    return 1;
  Wire.requestFrom(IMUAddress, nbytes,(uint8_t)true);
  for(uint8_t i = 0; i < nbytes; i++) {
```

```
    if(Wire.available())
      data[i] = Wire.read();
    else {
      timeOutTimer = micros();
      while(((micros() - timeOutTimer) < I2C_TIMEOUT) &&
!Wire.available());

      if(Wire.available())
        data[i] = Wire.read();
      else
        return 2;
    }
  }
  return 0;
}

void setup() {
  Wire.begin();

  i2cData[0] = 7;
  i2cData[1] = 0x00;
  i2cData[3] = 0x00;

  while(i2cWrite(0x19, i2cData, 4, false));
  while(i2cWrite2(0x6B, 0x01, true));
  while(i2cRead(0x75, i2cData, 1));
  delay(100);
  while(i2cRead(0x3B,i2cData,6));
 void ICACHE_RAM_ATTR ISRoutine ();
  timer = micros();
  Serial.begin(115200);
  bleMouse.begin();
  delay(100);
}

void loop() {
  while(i2cRead(0x3B,i2cData,14));

  gyroX = ((i2cData[8] << 8) | i2cData[9]);
  gyroZ = ((i2cData[12] << 8) | i2cData[13]);

  gyroX = gyroX / Sensitivity / 1.1  * -1;
  gyroZ = gyroZ / Sensitivity  * -1;

  if(bleMouse.isConnected()){
    Serial.print(gyroX);
    Serial.print("   ");
    Serial.print(gyroZ);
    Serial.print("\r\n");
    bleMouse.move(gyroZ, -gyroX);
  }
  delay(delayi);
}
```

B. Arduino Remote Receive Demo

```
/*
 * IRremote: IRrecvDemo - demonstrates receiving IR codes with IRrecv
 * An IR detector/demodulator must be connected to the input RECV_PIN.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 */

#include <IRremote.h>

int RECV_PIN = 4; //DEFAULT RECEIVER PIN, IF WE WANT WE CAN CHANGE


IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  pinMode(3,OUTPUT); digitalWrite(3,LOW);// GND
  pinMode(2,OUTPUT); digitalWrite(2,HIGH);//VCC
  Serial.begin(115200);

  Serial.println("Enabling IRin");
  irrecv.enableIRIn(); // Start the receiver
  Serial.println("Enabled IRin");
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
  delay(10);//DELAY FOR STABILITY
}
```

C. Remote Python Application

```
# PC Remote Control
import time
import threading
import tkinter
from tkinter import ttk
from tkinter import *
import serial
import win32api
import pyautogui
import os
import subprocess
import serial.tools.list_ports
ports=serial.tools.list_ports.comports()
print ("list of COM ports: \n")
for port, desc,hwid in sorted(ports):
    print("{}: {} ".format(port, desc))

# code assembled and edited by Ahmad Titi

serial_data = ''
filter_data = ''
update_period = 5
serial_object = None
gui = Tk()
gui.title("PC Remote Control")
gui.configure(background="blue")

def connect():
    global serial_object
    port = port_entry.get()
    baud = 115200 # baud_entry.get()
    try:
        serial_object = serial.Serial('COM'+ str(port), baud)
    except ValueError:
        print ("Enter Baud and Port")
        return
    t1 = threading.Thread(target = get_data)
    t1.daemon = True
    t1.start()

def get_data():
    """This function serves the purpose of collecting data from the
serial object and storing the filtered data into a global variable.
    The function has been put into a thread since the serial event is
a blocking function. """
```

```
    global serial_object
    global filter_data

    while(1):
        try:
            serial_data = serial_object.readline()
            refined=str(serial_data.decode('ascii'))

            serial_data=refined
            text.insert(END, serial_data)
            if '20DF40BF' in serial_data:
                pyautogui.press('volumeup',10)
            elif '20DFC03F' in serial_data:
                pyautogui.press('volumedown',10)
            elif '20DF906F' in serial_data:
                pyautogui.press('volumemute')
            elif '20DF00FF' in serial_data:
                pyautogui.press('pageup')
            elif '20DF807F' in serial_data:
                pyautogui.press('pagedown')
            elif '180BD9FF' in serial_data:
                pyautogui.press('win')
            elif '20DFCA35' in serial_data:
                pyautogui.rightClick()
            elif '20DF22DD' in serial_data:
                pyautogui.click()
            elif '20DF02FD' in serial_data:
                pyautogui.press('up')
            elif '20DF827D' in serial_data:
                pyautogui.press('down')
            elif '20DFE01F' in serial_data:
                pyautogui.press('left')
            elif '20DF609F' in serial_data:
                pyautogui.press('right')
            elif '20DF7887' in serial_data:
                pyautogui.press('space')
            elif '20DF14EB' in serial_data:
                pyautogui.press('backspace')
            elif '20DF8877' in serial_data:
                pyautogui.hotkey('ctrl', 'a')
            elif '20DF48B7' in serial_data:
                pyautogui.hotkey('ctrl', 'c')
            elif '20DFC837' in serial_data:
                pyautogui.hotkey('ctrl', 'v')
            elif '20DF28D7' in serial_data:
                pyautogui.hotkey('ctrl', 'x')
            elif '20DFA857' in serial_data:
                pyautogui.hotkey('ctrl', 'z')
```

```
            elif '20DF6897' in serial_data:
                pyautogui.press('delete')
            elif '20DFE817' in serial_data:
               pyautogui.press('capslock')
            elif '20DF18E7' in serial_data:
              pyautogui.hotkey('win', 'r')
            elif '20DF9867' in serial_data:
              pyautogui.hotkey('win', 's')
            elif '20DF08F7' in serial_data:
                pyautogui.press('enter')
            elif '20DF7E81' in serial_data:
                pyautogui.hotkey('win','d')
            elif '20DFDA25' in serial_data:
                pyautogui.hotkey('alt','F4')
            elif '20DFC23D' in serial_data:
                pyautogui.press('win')
            elif '20DF10EF' in serial_data:
                subprocess.call(["shutdown", "/s"])
            elif '20DF6A95' in serial_data:
                win32api.ShellExecute(0, 'open', 'C:\Program
Files\Google\Chrome\Application\\Netflix.html', '', '', 1)
            elif '20DF3AC5' in serial_data:
                win32api.ShellExecute(0, 'open', 'C:\Program
Files\Google\Chrome\Application\Shahid.html', '', '', 1)
            elif '20DFD52A' in serial_data:
                win32api.ShellExecute(0, 'open', 'C:\Program
Files\Google\Chrome\Application\YouTube.html', '', '', 1)
            elif '20DFC639' in serial_data:
                win32api.ShellExecute(0, 'open', 'C:\Program
Files\Google\Chrome\Application\chrome.exe', '', '', 1)


        except TypeError:
            pass


def update_gui():
    """" This function is an update function which is also threaded.
The function assimilates the data and applies it to its corresponding
progress bar. The text box is also updated every couple of seconds.

    A simple auto refresh function .after() could have been used, this
has been avoid purposely due to various performance issues. """
    global filter_data
    global update_period
    global serial_object

    text.place(x = 12, y = 170)
    new = time.time()

    while(1):
        if time.time() - new >= update_period:
            text.delete(0.0, END)
            new = time.time()

```

```python
def send():
    """This function is for sending data from the computer to the host
controller.
       The value entered in the entry box is pushed to the UART. The
data can be of any format, since
        the data is always converted into ASCII, the receiving device
has to convert the data into the required f
        format."""
    send_data = data_entry.get()
    if not send_data:
        print ("Sent Nothing")
    serial_object.write((send_data))

def disconnect():
    """
    This function is for disconnecting and quitting the application.

    Sometimes the application throws a couple of errors while it is
being shut down, the fix isn't out yet
    but will be pushed to the repo once done.

    simple GUI.quit() calls. """

    try:
        serial_object.close()
    except AttributeError:
        print ("Closed without Using it -_-")
    gui.destroy()
    gui.quit()
######################################################################
if __name__ == "__main__":

    """
    The main loop consists of all the GUI objects and its placement.

    The Main loop handles all the widget placements.

    """
##    global serial_data
    #frames
    frame_1 = Frame(height = 285, width = 480, bd = 3, relief =
'groove').place(x = 7, y = 5)
    frame_2 = Frame(height = 150, width = 480, bd = 3, relief =
'groove').place(x = 7, y = 300)
    text = Text(width = 58, height = 7)#17
#threads
    t2 = threading.Thread(target = update_gui)
    t2.daemon = True
    t2.start()
```

```
# labels
    heading=Label(text="PC REMOTE CONTROL",font="Times 25 bold italic
").place(x=12, y=10)
    heading10=Label(text="Ahmad H. Titi",font="Times 15 bold
").place(x=120, y=50)
    #baud   = Label(text = "Baud").place(x = 100, y = 348)
    port   = Label(text = "Port").place(x = 200, y = 348)
    received=Label(text="Received Serial data:",font="Times 15
").place(x=12, y=140)
    contact = Label(text = "Designed by \n
ahmadtheotherside@gmail.com
",font="Tahoma 9 bold ").place(x =8 , y = 450)
# data input
    #baud_entry = Entry(width = 7)
    #baud_entry.place(x = 100, y = 365)

    port_entry = Entry(width = 7)
    port_entry.place(x = 200, y = 365)
#commands
    connect = Button(text = "Connect", command = connect).place(x =
15, y = 360)
    disconnect = Button(text = "Exit", command =
disconnect,width=17).place(x =300, y = 360)
#mainloops
    gui.geometry('500x500')
    gui.mainloop()
```