B. K. BIRLA COLLEGE (AUTONOMOUS), KALYAN

(DEPARTMENT OF COMPUTER SCIENCE)

CERTIFICATE

This is to certify that Mr./Ms. _____

Roll No.____Exam Seat No._____has satisfactorily completed the Practical in **Robotics** *as laid down in the regulation of University of Mumbai for the purpose of M.Sc. Computer Science* **Semester- IV(Practical)** *Examination* **2023-2024**.

Date:___/06/2024

Place: Kalyan

_____

Signature of Examiners

_____

Professor In-charge

Head
Dept. Of Computer Science

Index

Exp no-1A - Study of WokWi Simulator

Aim : To study Wokwi and Tinkercad Simulator to do simulation of IoT hardware

Hardware Requirement

- Intel Processor based Computer with Internet connection
- 8GB RAM
- 250GB Min Harddisk Space

Software Requirement

- Windows
- Wokwi Simulator
- Tinkercad Simulator

Theory

Wokwi is an online Electronics simulator.
You can use it to simulate Arduino, ESP32, and many other popular boards, parts, and sensors.

Features of Wokwi

- Start right now
- No waiting for components or downloading large software.
- In seconds, your browser has everything you need to start coding your next IoT project.
- Mistakes are okay
- You can't destroy the virtual hardware. So don't worry about frying your precious components.
- Unlike real hardware, you can always undo it.
- Easy to get help and feedback
- Sharing a link to your Wokwi project is all you need.

- Gain confidence in your code.
- Separate hardware and software issues.
- Unlimited hardware
- No need to scavenge parts from old projects.
- Use as many parts as you need without worrying about project price and stock.
- Maker-friendly community
- A place for you to share your projects, ask for help and get inspiration.

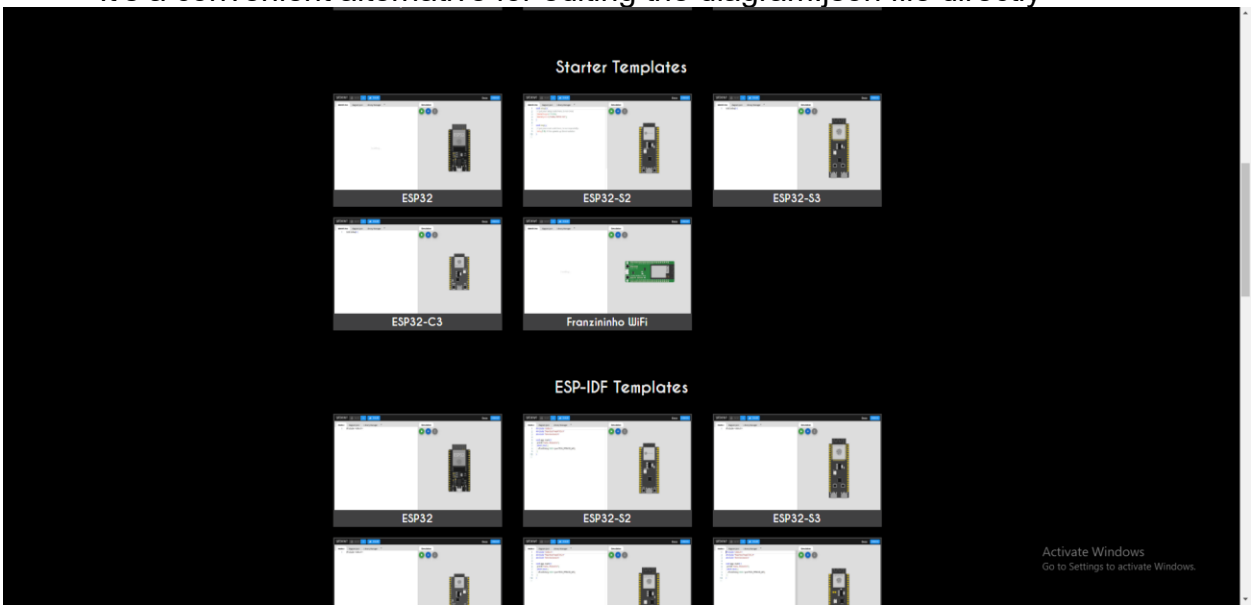
Cost of Wokwi

- Wokwi is free to use.
- Professional and frequent users can join the Club, influence our development roadmap, and get access to advanced features.

Interactive Diagram Editor

The diagram editor provides an interactive way to edit your diagram:

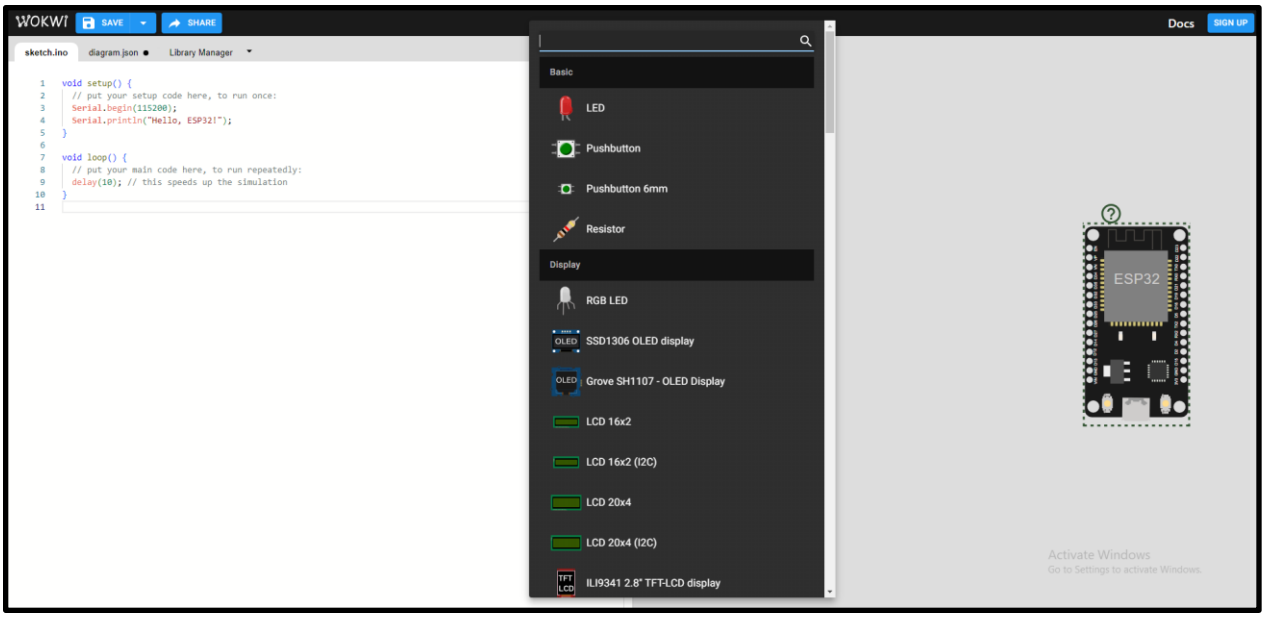
- Add components to the simulation
- Define the connections between them
- It's a convenient alternative for editing the diagram.json file directly



Editing parts

Adding part

- To add a new part, click on the blue "+" button at the top of the diagram editor.
- You'll see a menu with a list of parts you can add.
- Choose a part to add it. The part will be added at position (0, 0), and then you can drag it to the desired position.
- Not all parts are currently available through the menu. For example, MCU boards and micro-controllers such as the Arduino Nano or the ATtiny85 are missing. You can still add these parts by editing diagram.json directly.

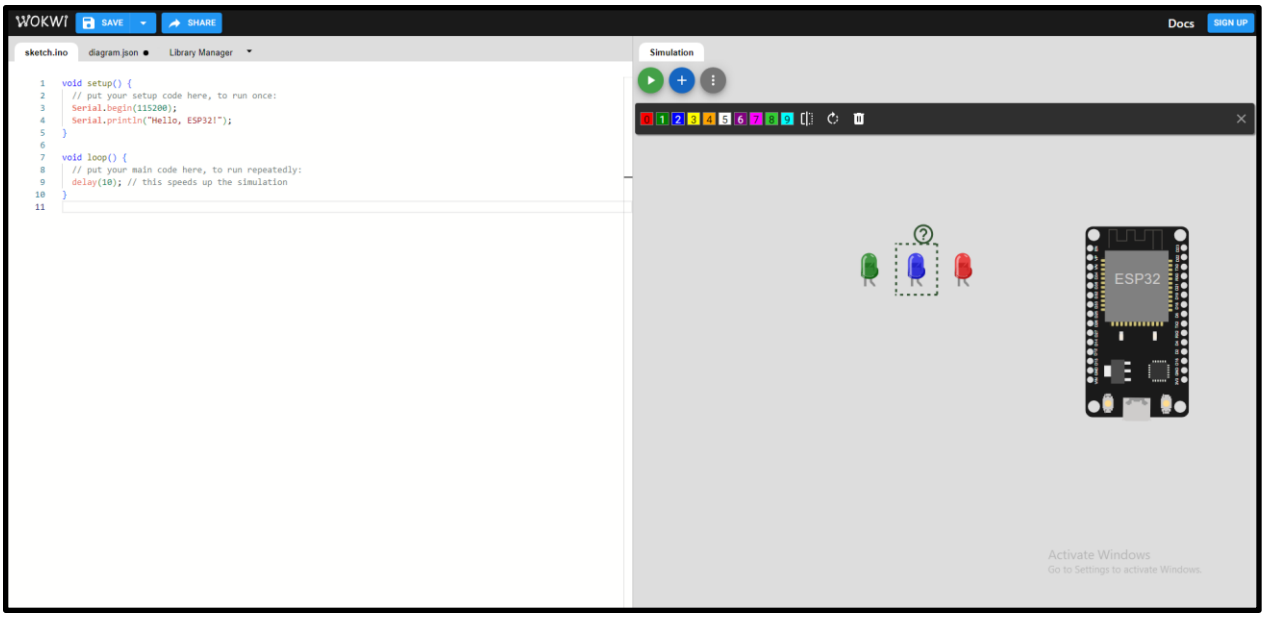


Moving a part

- Move a part by clicking on it and then dragging it with your mouse.
- Rotating a part
- Rotate a part by clicking on it (to select it) and then press "R".
- The part will rotate 90 degrees clockwise. If you need to rotate a part by a different amount (e.g. 45 degrees), you can achieve that by editing diagram.json.

Duplicating a part

- Create a new copy of a part by clicking on it (to select it) and pressing "D".
- You can press "D" several times to create multiple copies of the part.
- Deleting a part
- Delete a part by clicking on it (to select it) and then pressing the Delete button.
- selecting multiple parts
- Select multiple parts by clicking on the parts with the Shift key pressed. You can then move all the parts together, duplicate them (using the "D" key), or delete them using the Delete key.



Copying and pasting parts

- You can copy the selected part(s) using the standard Copy keyboard shortcut (Ctrl+C or ⌘+C).
- If you select multiple parts, all the wires connecting the chosen components are copied.
- The parts you copied are stored in your system clipboard in a JSON format, similar to the diagram.json format.
- To paste the parts you copied, click on the diagram and press the standard Paste keyboard (Ctrl+V or ⌘+V).
- Sometimes, the parts will be pasted outside the currently visible diagram area, so you may have to zoom out to find them. This will be fixed in the future.
- You can use the copy-paste feature between different project and quickly copy several parts (including all the internal connections) at once.

Editing wires

Creating a wire between two parts

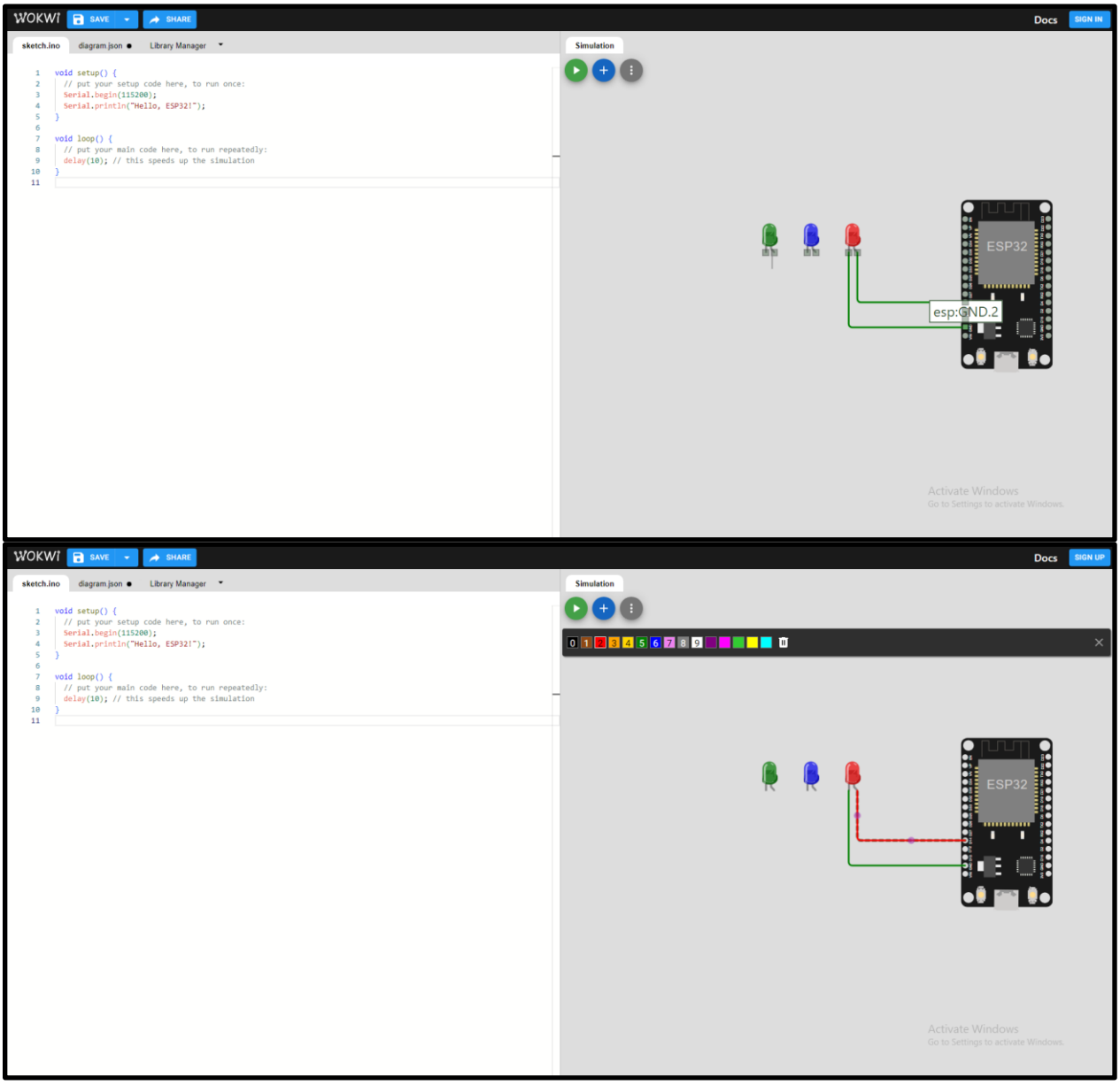
- To create a new wire between two parts, click on one of the pins that you'd like to connect.
- Then click on the second (target) pin. This will create the wire.
- If you want the wire to go in a specific way, you can guide it by clicking where you want it to go after selecting the first pin.
- To cancel a new wire (delete it without selecting a target pin) click the right mouse button or press Escape.
- Changing the color of a wire

Editing wires

Changing the color of a wire

- The function of the pin automatically determines the color of new wires:
- Wires starting from ground pins are black, 5 V pins are red, and other wires are green.

- You can change the color of a wire by clicking on it and then selecting a new color for the wire.
- You can also use the following keyboard shortcuts to set wire colors:
  [0-Black; 1- Brown; 2-Red; 3-Orange; 4- Gold; 5- Green; 6-Blue; 7- Violet; 8- Gray; 9-White; C-Cyan; L- Lime green; M-Magenta; P-Purple; Y-Yellow]



Deleting a wire

- Select a wire by clicking on it, then click the trash icon on it (or press the Delete key).

- You can also delete a wire by double-clicking on it.

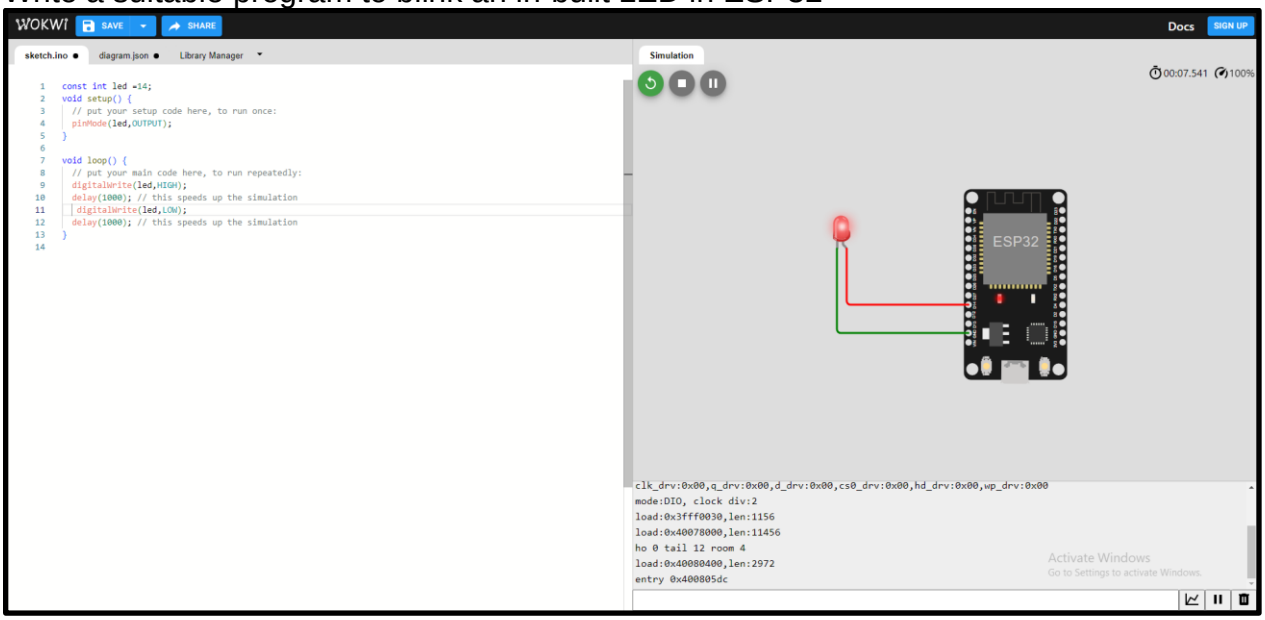
Keyboard shortcuts

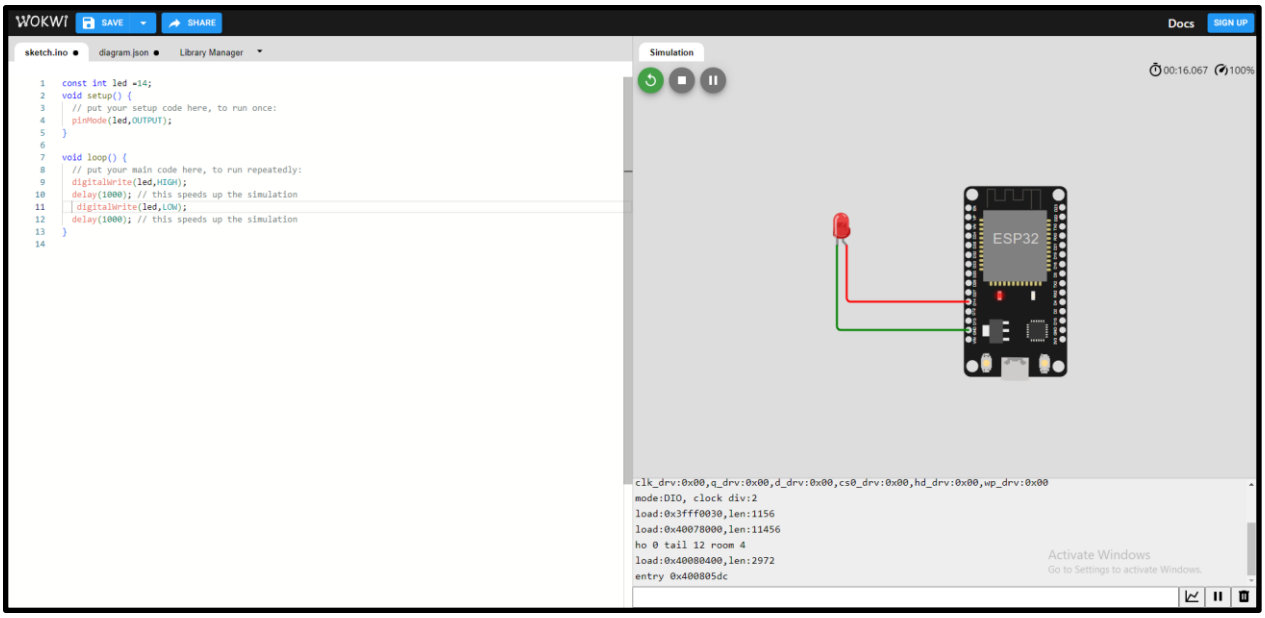
• The following table summarizes the keyboard shortcuts

Key	Control
-	Zoom out
+	Zoom in
F	Fit diagram to window (auto zoom)
D	Duplicate (copy) the selected part
R	Rotate the selected part
Delete	Delete the selected part / wire
?	Open documentation for the selected part

Activity:

Write a suitable program to blink an in-built LED in ESP32

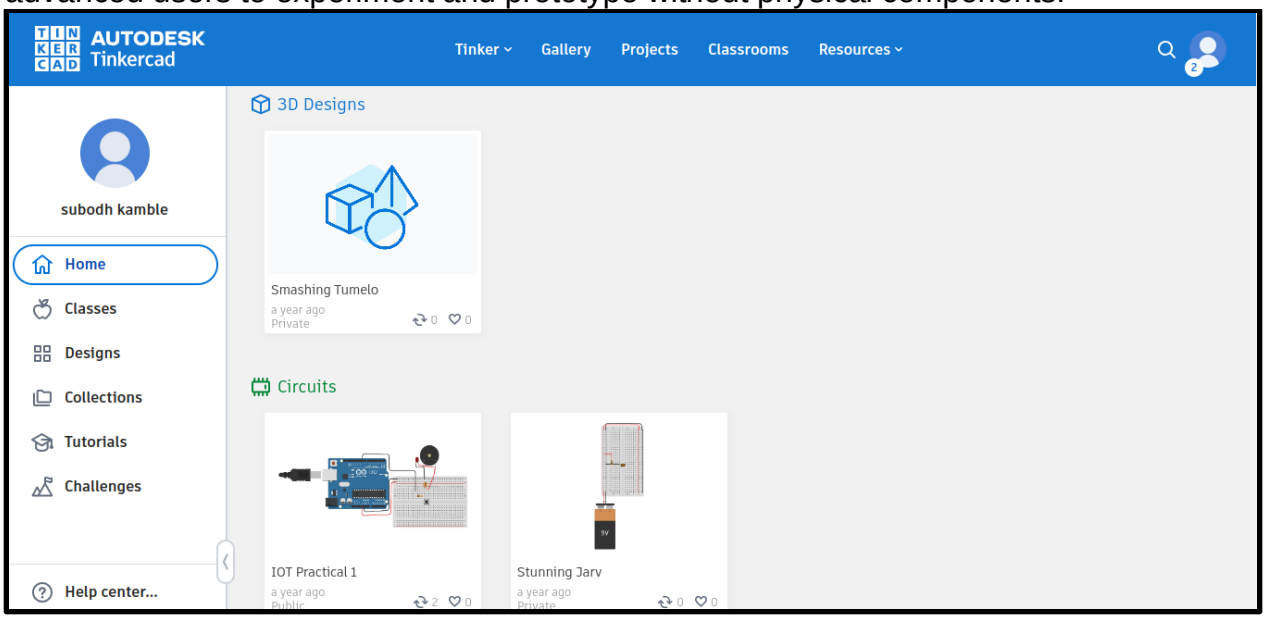


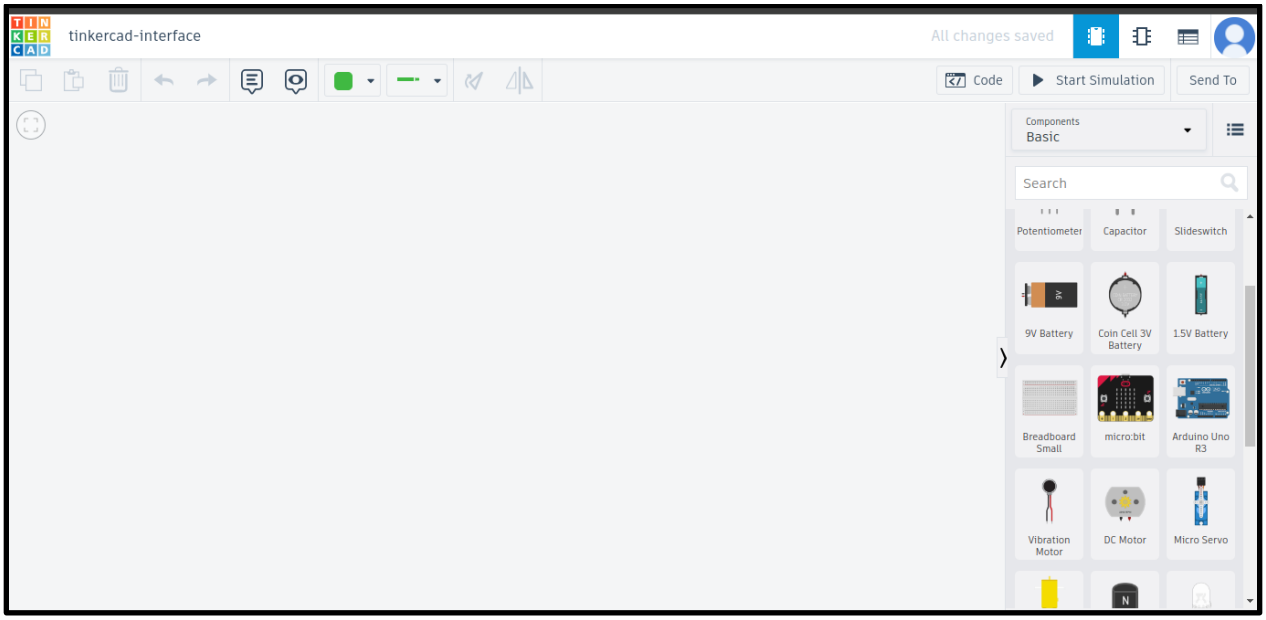

WOKWI
SAVE
SHARE
Docs
SIGN UP
sketch.ino
diagram.json
Library Manager
const int led =14;
void setup() {
// put your setup code here, to run once:
pinMode(led,OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(led,HIGH);
delay(1000); // this speeds up the simulation
digitalWrite(led,LOW);
delay(1000); // this speeds up the simulation
}
Simulation
00:16.067
100%
ESP32
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Activate Windows
Go to Settings to activate Windows.

Exp no-1B - Study of Tinkercad Simulator

Thoery

Tinkercad is another online 3D design and electronics simulator that allows users to create and simulate circuits, including Arduino, micro, and various other components and sensors. It offers an intuitive interface for designing, coding, and testing electronic projects in a virtual environment, making it an excellent tool for both beginners and advanced users to experiment and prototype without physical components.





Features of Tinkercad

- No waiting for components or downloading large software.
- In seconds, your browser has everything you need to start designing and simulating your next electronics project.
- Mistakes are okay
- You can't destroy virtual hardware. So don't worry about damaging your components.
- Unlike real hardware, you can always undo it.
- Easy to get help and feedback
- Sharing a link to your Tinkercad project is all you need.
- Gain confidence in your design.
- Separate hardware and software issues.
- Unlimited hardware
- No need to scavenge parts from old projects.
- Use as many parts as you need without worrying about project price and stock.
- Maker-friendly community
- A place for you to share your projects, ask for help, and get inspiration.

Cost of Tinkercad

- Tinkercad is free to use.
- Advanced users can access additional features and educational resources.
- Interactive Diagram Editor
- The diagram editor provides an interactive way to edit your diagram:
- Add components to the simulation
- Define the connections between them
- It’s a convenient alternative for editing circuit designs directly.

Interactive Diagram Editor

Tinkercad is an interactive diagram editor primarily designed for creating 3D models, circuits, and simulations in a user-friendly environment.

Intuitive Interface: Drag-and-drop components with real-time visual feedback.

Component Diversity: Extensive libraries for 3D models and electronic components.

Simulation Tools: Test circuits virtually before implementation.

Collaboration: Share and work on projects with others in real-time.

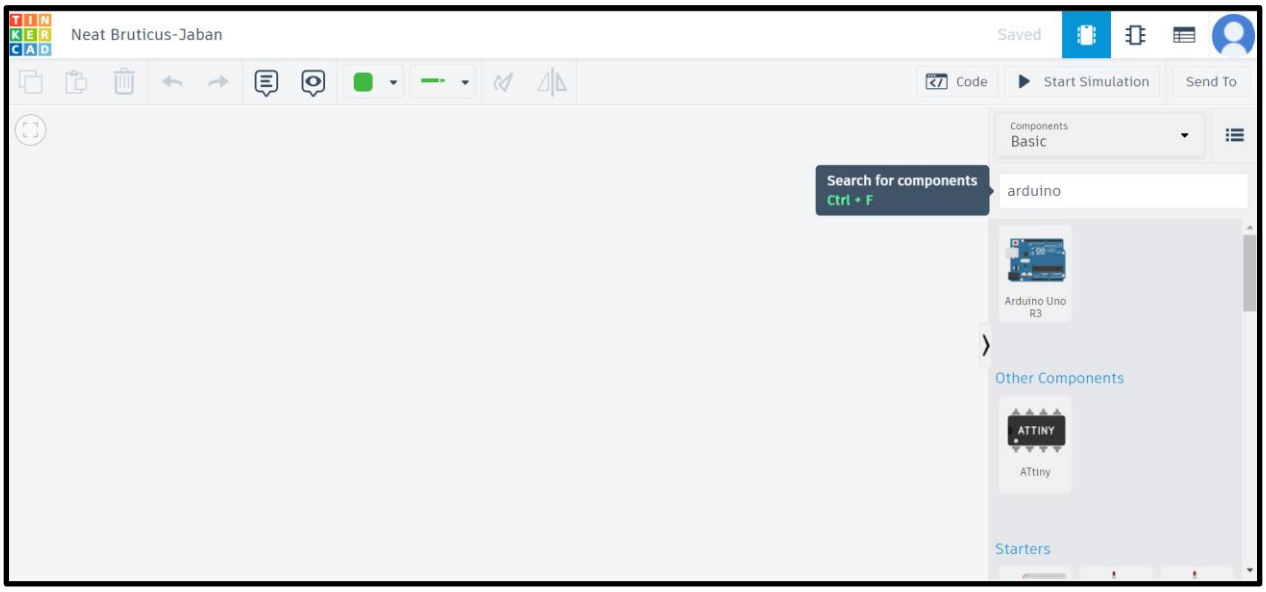
Editing parts

Adding part

Access Components Library: On the right-hand side of the workspace, you'll see a panel titled "Components". This panel contains various categories of electronic components.

Find and Select the Part: Browse through the categories (e.g., Basic, Inputs, Outputs) to locate the part you want to add. You can also use the search bar at the top to find specific components.

Drag and Drop: Click on the desired component, drag it into the workspace, and drop it where you want it to be placed.



Moving a part

Select the Part: Click on the component you want to move. It should become highlighted or show adjustment handles.

Drag the Part: While holding down the mouse button (left-click), drag the part to the desired new location within the circuit workspace.

Release: Once you've positioned the part where you want it, release the mouse button to drop it into place.

Copying and pasting parts

Select the Part: Click on the component or group of components you want to copy. They should become highlighted or show adjustment handles.

Copy the Part: On your keyboard, use the shortcut:

- **Windows/Linux**: Press Ctrl + C.
- **Mac**: Press Cmd + C.

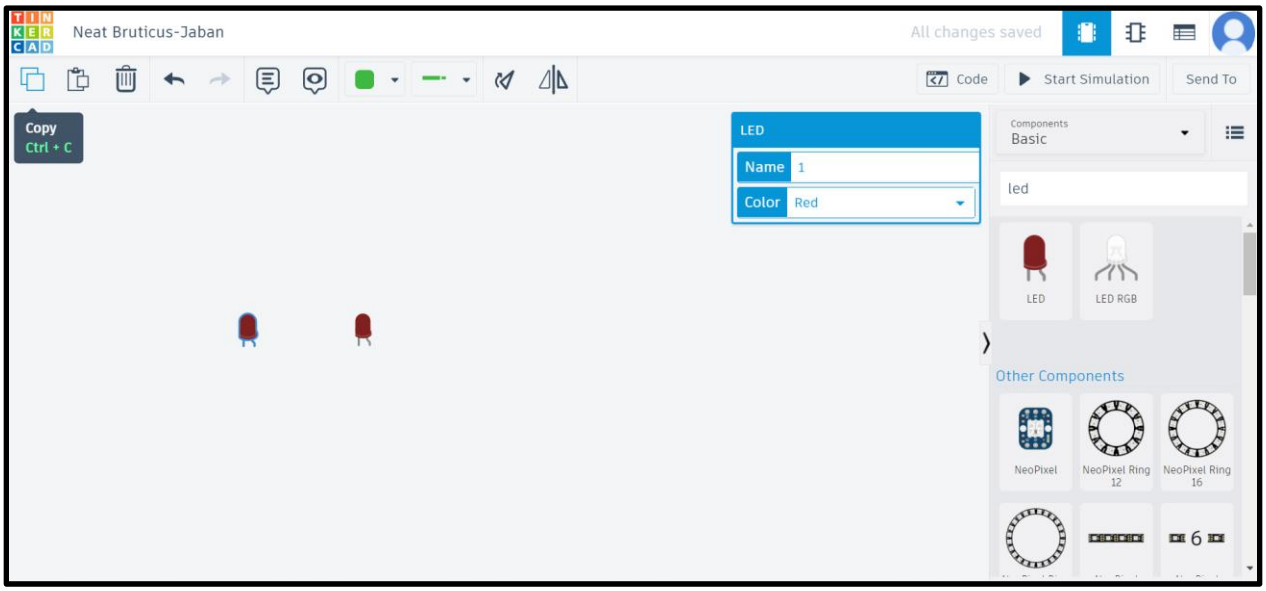
Alternatively, you can right-click on the selected components and choose "Copy" from the context menu.

Paste the Part:

- Move your cursor to the desired location where you want to place the copied parts.
- Use the keyboard shortcut:
    - **Windows/Linux**: Press Ctrl + V.

- **Mac**: Press Cmd + V.
- Alternatively, right-click in the workspace and select "Paste" from the context menu.

Position the Pasted Parts: Drag the pasted parts to the exact position you want within the circuit workspace.

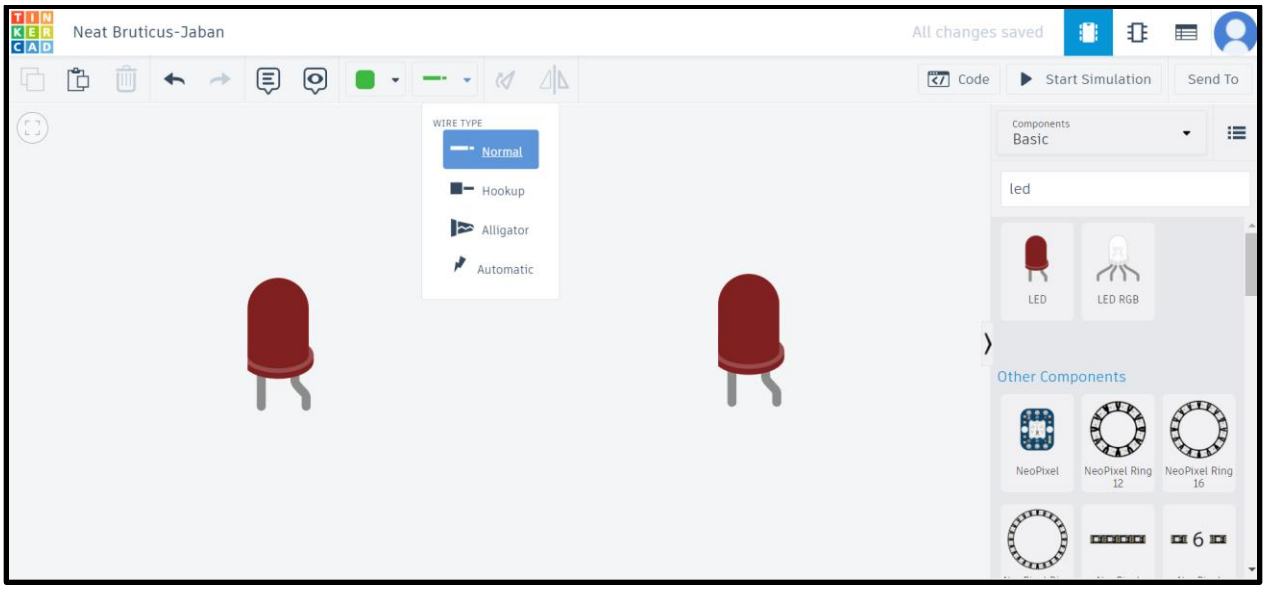


Editing wires
Creating a wire between two parts
Select the Wire Tool: In the Tinkercad Circuits workspace, locate the toolbar on the left-hand side. Click on the "Wire" tool. It typically looks like a straight line or a bent wire icon.
Connect Pins

- Click on the pin (connection point) of the first component from where you want to start the wire. You should see a line extending from the pin.
- Move your cursor to the pin of the second component where you want to end the wire.
- Click again on the pin of the second component to complete the connection.

Adjust and Route the Wire

- If needed, you can click and drag on the wire to adjust its path or route it around other components.
- Tinkercad will automatically adjust the wire's shape to avoid overlapping with other components.



Editing wires

Changing the color of a wire

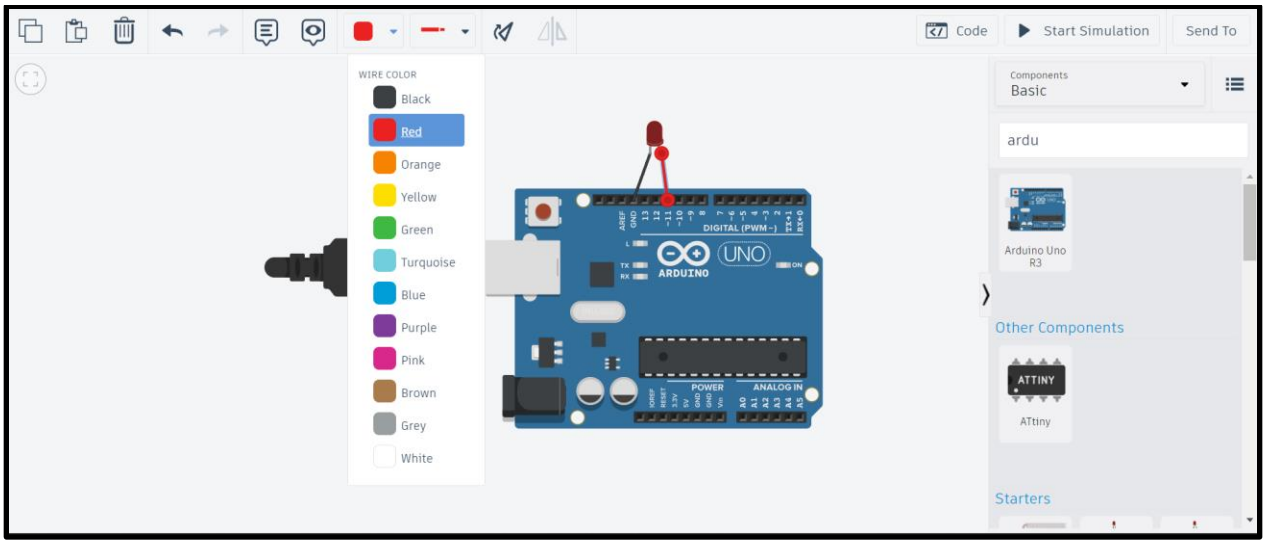
Select the Wire: Click on the wire segment you want to change. This should highlight the wire and show adjustment handles.

Color Selection Bar: Look for a toolbar or ribbon near the top of the workspace. This is where you can access various options including changing the wire color.

Change Wire Color

- In the toolbar or ribbon, locate the option for changing the wire color. It may be represented by a color palette icon or a dropdown menu.
- Click on the icon or dropdown menu to open it.
- Choose a new color from the available options.

Apply Color: The wire color should update automatically in the workspace to reflect your chosen color.



Deleting a wire

Select the Wire: Click on the wire segment that you want to delete. This should highlight the wire and show adjustment handles.

Find the Delete Icon: Look for a toolbar or ribbon near the top of the workspace. In this toolbar, you should see an icon that looks like a trash bin or says "Delete".

Delete the Wire:

- Click on the trash bin icon or the "Delete" option in the toolbar.
- Alternatively, you can press the "Delete" key on your keyboard after selecting the wire.

Keyboard shortcuts

AUTODESK® TINKERCAD®

KEYBOARD SHORTCUTS

Legend: Ctrl + Cmd / Alt + Option

MOVING OBJECT(S)
(Using keyboard)

Move along X/Y axis	← / ↑ / ↓ / →
Move along Z axis	Ctrl + ↓ / ↑
×10 Nudge along X/Y axis	Shift + ← / ↑ / ↓ / →
×10 Nudge along Z axis	Ctrl + Shift + ↓ / ↑

KEYBOARD + MOUSE SHORTCUTS
(Press and hold the keys, then click and drag the mouse)

Duplicate dragged object(s)	Alt + Drag left mouse button
Select multiple object(s)	Shift + Left mouse button
45° rotation	Shift (Hold while rotating)
Scale in one direction	Alt + Hold side handle
Scale in two directions	Alt + Hold corner handle
Uniform scale	Shift + Hold corner handle
Uniform scale in all directions	Alt + Shift + Corner handle
Uniform scale in all directions	Alt + Shift + Top handle

VIEWING DESIGNS
(With the help of a mouse or a mouse pad)

Orbit the view	Right mouse button
Orbit the view	Ctrl + Left mouse button
Pan the view	Shift + Right mouse button
Pan the view	Ctrl + Shift + left button
Zoom the view in or out	Mouse scroll wheel
Zoom-in	+
Zoom-out	-
Fit selected object(s) into view	F

OBJECT SETTINGS

Transparency toggle	T
Turn object(s) into **Holes**	H
Turn object(s) into **Solids**	S
Lock or **Unlock** object(s)	Ctrl + L
Hide object(s)	Ctrl + H
Show all hidden object(s)	Ctrl + Shift + H

TOOLS AND COMMANDS

Copy object(s)	Ctrl + C
Paste object(s)	Ctrl + V
Duplicate object(s) in place.	Ctrl + D
Delete object(s)	Del
Undo action(s)	Ctrl + Z
Redo action(s)	Ctrl + Y
Redo action(s)	Ctrl + Shift + Z
Group object(s)	Ctrl + G
Un-group object(s)	Ctrl + Shift + G
Align object(s)	L
Flip/Mirror objects(s)	M
Select all object(s)	Ctrl + A
Place a **Ruler**	R (Shift toggle midpoint/center)
Place a **Workplane**	W (press Shift to flip direction)
Drop object(s) to workplane	D

Visit www.tinkercad.com/learn for more tips, step-by-step tutorials, and easy projects. Happy Tinkering!

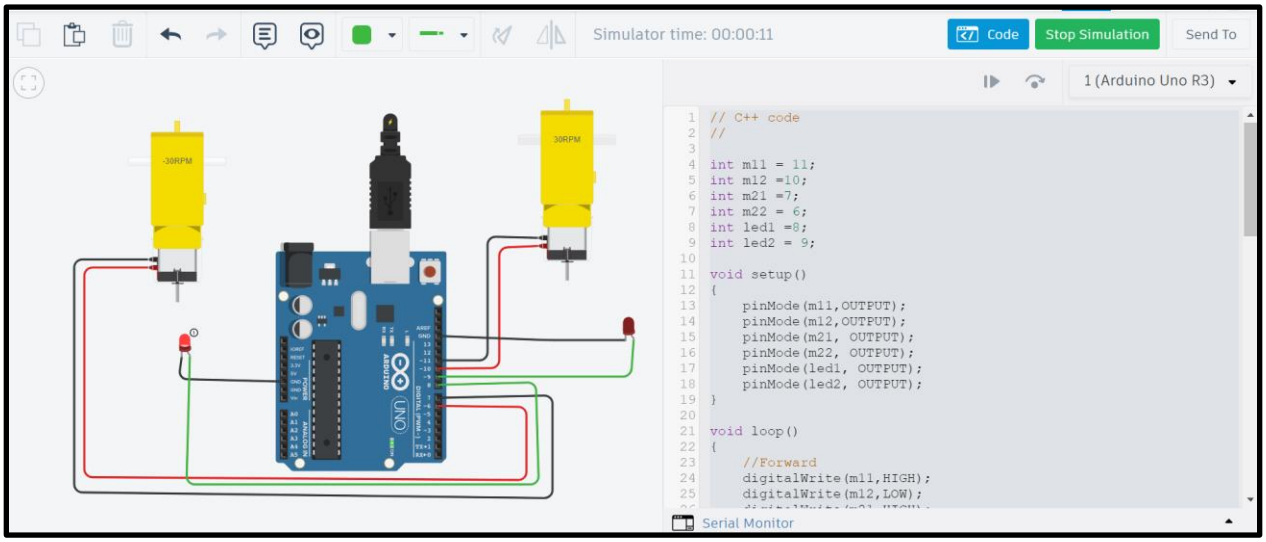
Activity

Write a suitable program to show working of motors in Tinkercad

Code:

```
// C++ code
//
int m11 = 11;
int m12 =10;
int m21 =7;
int m22 = 6;
int led1 =8;
int led2 = 9;

void setup()
{
	pinMode(m11,OUTPUT);
	pinMode(m12,OUTPUT);
	pinMode(m21, OUTPUT);
	pinMode(m22, OUTPUT);
	pinMode(led1, OUTPUT);
	pinMode(led2, OUTPUT);
}

void loop()
{
	//Forward
	digitalWrite(m11,HIGH);
	digitalWrite(m12,LOW);
	digitalWrite(m21,HIGH);
	digitalWrite(m22,LOW);
	digitalWrite(led1,HIGH);
	digitalWrite(led2,HIGH);
	delay(5000);

	//Backward
	digitalWrite(m11,LOW);
	digitalWrite(m12, HIGH);
	digitalWrite(m21,LOW);
	digitalWrite(m22, HIGH);
	digitalWrite(led1, LOW);
	digitalWrite(led2, LOW);
	delay(5000);

	//Left
	digitalWrite(m11,LOW);
	digitalWrite(m12, HIGH);
	digitalWrite(m21,HIGH);
	digitalWrite(m22, LOW);
	digitalWrite(led1, HIGH);
```

```
        digitalWrite(led2, LOW);
        delay(5000);

        //Right
        digitalWrite(m11,HIGH);
        digitalWrite(m12,LOW);
        digitalWrite(m21,LOW);
        digitalWrite(m22,HIGH);
        digitalWrite(led1, LOW);
        digitalWrite(led2, HIGH);
        delay(5000);

        //Stop
        digitalWrite(m11,HIGH);
        digitalWrite(m12,HIGH);
        digitalWrite(m21,HIGH);
        digitalWrite(m22,HIGH);
        digitalWrite(led1,LOW);
        digitalWrite(led2,LOW);
        delay(5000);
}
```



Experiment No – 2 Raspberry Pi

Aim : To Study Rpi Board and its Programming

Raspberry Pi

- The Raspberry Pi is a debit card-sized low-cost computer that connects to a computer Desktop or TV and uses a standard mouse and Keyboard.
- It has a dedicated processor, memory, and a graphics driver, just like a PC. It also comes with its operating system, Raspberry Pi OS, a modified version of Linux.
- Raspberry Pi can browse the internet and stream high-definition video, as well as spreadsheets, word processing, and gaming, just like a desktop computer.
- The Raspberry Pi can communicate with the outside world and has been used in various digital maker projects, including music machines and parent detectors, weather stations, and tweeting birdhouses with infrared cameras.
- People worldwide use Raspberry Pi to learn how to program and understand how computers function.
- Although the Raspberry Pi lacks storage, you can use microSD cards to store whichever operating system you choose (Raspberry Pi, Ubuntu Mate, etc.). Because the Raspberry Pi has Bluetooth, Ethernet, and WiFi connectivity, it may transfer files over the internet. The software and the design of the Raspberry Pi project are not open-source.

Use Cases of Raspberry Pi

Now that you have understood exactly what is Raspberry Pi, you will explore the use cases of Raspberry Pi:

- **Desktop (PC)**
- A simple desktop can be built using a Raspberry Pi, a microSD card, and a power source. An HDMI cable and a suitable display, such as an old monitor, are also required. You'll also need a USB keyboard and mouse.

- **Robotics Controller**
- There are numerous Raspberry Pi robot-controller projects. A specific robotics package for Pi is available, powered by the device battery, and can interface with and control robots.

- **Printing Using a Raspberry Pi**
- With the Raspberry Pi, you can print anything. All you'll need is a Raspberry Pi that's connected to your home network, as well as some print server software. First, this is accomplished by installing the Samba file-sharing program, then CUPS. The Common Unix Printing System (CUPS) includes printer drivers and a management console.

- **Game Servers**
- The basic operating system for the Raspberry Pi has a special version of the Minecraft game pre-installed. Raspberry Pi applications can be used as a game server. It's a fantastic Minecraft game server. A wonderful gaming experience can be created by using many Raspberry Pi.

- **Gaming Machine**
- As a retro gaming machine, the Raspberry Pi is perfect. It is one of the machine's lightest components. The Raspberry Pi Zero, in particular, is a variant that can fit into

small locations and be used for gaming projects. The Raspberry Pi may also be used to restore many popular 16-bit game systems.

Raspberry Pi Applications

The Raspberry Pi model can be used for a variety of purposes such as:

- Media streamer
- Tablet computer
- Home automation
- Internet radio
- Robotics command
- Arcade devices
- cosmic computer
- Projects that use the Raspberry Pi

Types of Raspberry Pi

- Raspberry Pi model B
- Raspberry Pi model A
- Raspberry Pi model B+
- Raspberry Pi Model A+
- Raspberry Pi Zero
- Raspberry Pi 2
- Raspberry Pi 3 Model B
- Raspberry Pi Zero W
- Raspberry Pi 4B
- Raspberry Pi 5

The Raspberry Pi series has grown extensively since its inception, offering a range of models that cater to various needs from basic educational tools to powerful development platforms. Below is an overview of the key Raspberry Pi models including the original B and A models, their enhancements, the compact Pi Zero variants, and the more powerful models up to the 3B.

Raspberry Pi Model B

1. Specifications:

- **Processor:** Broadcom BCM2835, ARM11 700 MHz.
- **Memory:** 256MB SDRAM (later models upgraded to 512MB).
- **Networking:** 10/100 Ethernet.
- **Ports:** 2 USB 2.0 ports, 1 HDMI port, 1 composite video and audio out, 1 SD card slot.
- **GPIO:** 26-pin GPIO header.
- **Power:** 5V/700mA via micro USB connector.

2. Key Features:

- First model launched, setting the foundation for subsequent models.
- Basic computing capabilities suitable for educational purposes.

3. Use Cases:

- Entry-level projects.
- Basic educational and programming tasks.

Raspberry Pi Model A

1. Specifications:

- **Processor:** Broadcom BCM2835, ARM11 700 MHz.
- **Memory:** 256MB SDRAM.
- **Networking:** No Ethernet.
- **Ports:** 1 USB 2.0 port, 1 HDMI port, 1 composite video and audio out, 1 SD card slot.
- **GPIO:** 26-pin GPIO header.
- **Power:** 5V/500mA via micro USB connector.

2. Key Features:

- Lower power consumption and reduced cost compared to Model B.
- Lacks networking capabilities, intended for applications where networking is not required.

3. Use Cases:
- Cost-sensitive projects.
- Battery-powered applications.

Raspberry Pi Model B+

1. Specifications:
- **Processor:** Broadcom BCM2835, ARM11 700 MHz.
- **Memory:** 512MB SDRAM.
- **Networking:** 10/100 Ethernet.
- **Ports:** 4 USB 2.0 ports, 1 HDMI port, 1 composite video and audio out, 1 microSD card slot.
- **GPIO:** 40-pin GPIO header.
- **Power:** 5V/2A via micro USB connector.

2. Key Features:
- Improved power management and increased USB ports.
- More GPIO pins for expanded connectivity.

3. Use Cases:
- Projects requiring multiple USB peripherals.
- More complex hardware interfacing due to additional GPIO pins.

Raspberry Pi Model A+

1. Specifications:
- **Processor:** Broadcom BCM2835, ARM11 700 MHz.
- **Memory:** 256MB or 512MB SDRAM.
- **Networking:** No Ethernet.
- **Ports:** 1 USB 2.0 port, 1 HDMI port, 1 composite video and audio out, 1 microSD card slot.
- **GPIO:** 40-pin GPIO header.
- **Power:** 5V/1A via micro USB connector.

2. Key Features:
- Smaller form factor compared to Model A.
- Lower power consumption with increased GPIO pins.

3. Use Cases:
- Compact and portable projects.
- Applications needing extended GPIO functionality.

Raspberry Pi Zero

1. Specifications:
- **Processor:** Broadcom BCM2835, ARM11 1GHz.
- **Memory:** 512MB LPDDR2 SDRAM.
- **Networking:** None (no Ethernet or Wi-Fi).
- **Ports:** 1 mini HDMI port, 1 micro USB OTG port, 1 micro USB power port, 1 CSI camera connector.
- **GPIO:** 40-pin GPIO header.
- **Storage:** microSD slot.
- **Power:** 5V/500mA via micro USB connector.

2. Key Features:

- Ultra-compact size and minimalistic design.
- Very low cost.

3. Use Cases:

- Ultra-compact and portable projects.
- Embedded systems where space and cost are critical.

Raspberry Pi Zero W

1. Specifications:

- **Processor:** Broadcom BCM2835, ARM11 1GHz.
- **Memory:** 512MB LPDDR2 SDRAM.
- **Networking:** 2.4GHz 802.11n wireless LAN, Bluetooth 4.1, BLE.
- **Ports:** 1 mini HDMI port, 1 micro USB OTG port, 1 micro USB power port, 1 CSI camera connector.
- **GPIO:** 40-pin GPIO header.
- **Storage:** microSD slot.
- **Power:** 5V/500mA via micro USB connector.

2. Key Features:

- Same ultra-compact size as the Pi Zero but with added wireless connectivity.
- Integrated Wi-Fi and Bluetooth.

3. Use Cases:

- IoT projects needing wireless connectivity.
- Compact applications requiring networking without additional peripherals.

Raspberry Pi 2 Model B

1. Specifications:

- **Processor:** Broadcom BCM2836, quad-core ARM Cortex-A7 900 MHz.
- **Memory:** 1GB LPDDR2 SDRAM.
- **Networking:** 10/100 Ethernet.
- **Ports:** 4 USB 2.0 ports, 1 HDMI port, 1 3.5mm audio/composite video jack, 1 microSD card slot.
- **GPIO:** 40-pin GPIO header.
- **Power:** 5V/2A via micro USB connector.

2. Key Features:

- Significant performance boost with quad-core processor.
- Increased memory for better multitasking.

3. Use Cases:

- More demanding projects and applications.
- Better performance for desktop computing tasks.

Raspberry Pi 3 Model B

1. Specifications:

- **Processor:** Broadcom BCM2837, quad-core Cortex-A53 (ARMv8) 64-bit SoC at 1.2GHz.
- **Memory:** 1GB LPDDR2 SDRAM.
- **Networking:** 2.4GHz 802.11n wireless LAN, Bluetooth 4.1, BLE, 10/100 Ethernet.
- **Ports:** 4 USB 2.0 ports, 1 HDMI port, 1 3.5mm audio/composite video jack, 1 microSD card slot.
- **GPIO:** 40-pin GPIO header.

- **Power:** 5V/2.5A via micro USB connector.

2. Key Features:

- Integrated Wi-Fi and Bluetooth for wireless connectivity.
- Significant performance improvement with 64-bit support.

3. Use Cases:

- Wireless networking projects.
- Enhanced IoT applications with built-in connectivity.
- General-purpose computing with better performance.

Raspberry Pi 3B+

1. Specifications:

- **Processor:** Broadcom BCM2837B0, quad-core Cortex-A53 (ARMv8) 64-bit SoC at 1.4 GHz.
- **Memory:** 1GB LPDDR2 SDRAM.
- **Networking:** 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, and Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps).
- **Ports:** 4 USB 2.0 ports, 1 HDMI port, 1 micro USB power port, 1 3.5mm audio/composite video jack, 1 camera interface (CSI), 1 display interface (DSI).
- **GPIO:** 40-pin GPIO header.
- **Storage:** microSD slot.
- **Power:** 5V/2.5A DC via micro USB connector.

2. Key Features:

- Dual-band Wi-Fi and Bluetooth connectivity.
- Improved power management for better thermal performance and lower power consumption.
- Maintains the form factor and GPIO compatibility with previous models.

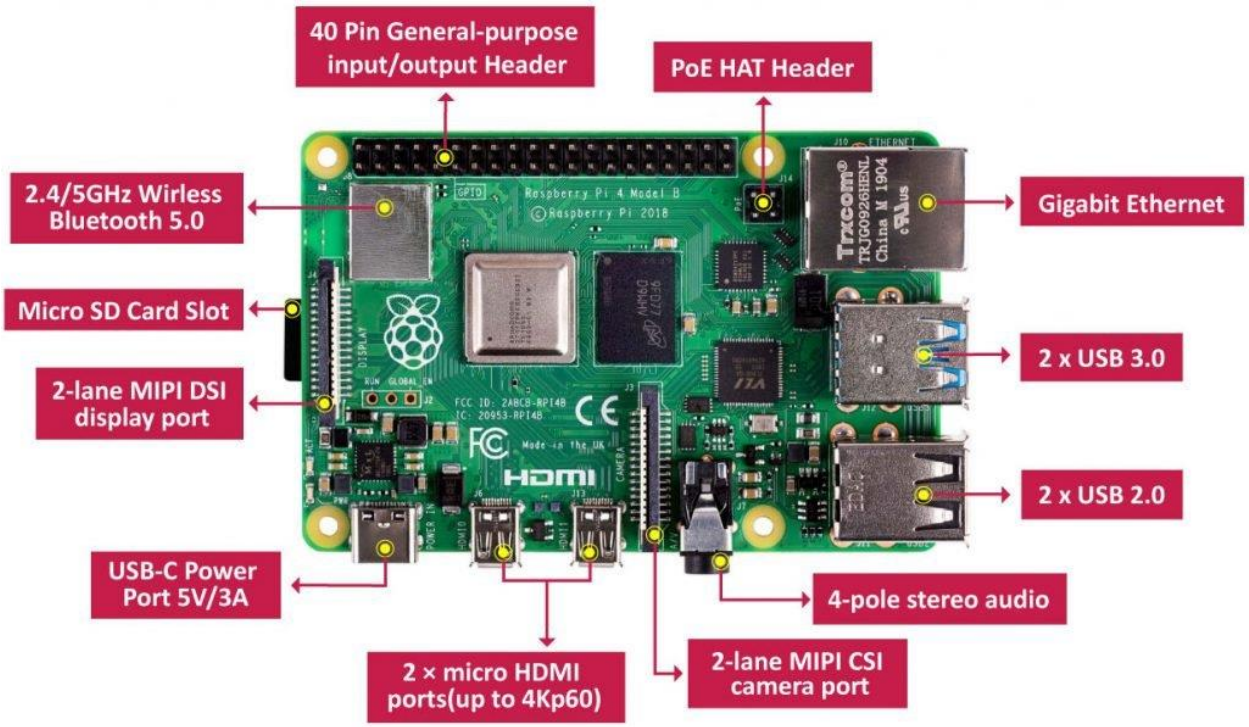
3. Use Cases:

- Basic desktop computing and web browsing.
- Educational projects and DIY electronics.
- Lightweight server applications and IoT projects.

Raspberry Pi 4B

1. Specifications:

- **Processor:** Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC at 1.5 GHz.
- **Memory:** Multiple variants with 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM.
- **Networking:** True Gigabit Ethernet, 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 5.0, BLE.
- **Ports:** 2 USB 3.0 ports, 2 USB 2.0 ports, 2 micro HDMI ports (up to 4Kp60 supported), 1 USB-C power port, 1 3.5mm audio/composite video jack, 1 camera interface (CSI), 1 display interface (DSI).
- **GPIO:** 40-pin GPIO header.
- **Storage:** microSD slot.
- **Power:** 5V/3A DC via USB-C connector.



2. Key Features:

- Significantly improved processing power and RAM options.
- Support for dual 4K displays with dual micro HDMI ports.
- USB 3.0 ports for faster data transfer.
- Enhanced networking with true Gigabit Ethernet.

3. Use Cases:

- High-performance desktop computing.
- Media centers and home theaters with 4K output.
- More complex IoT and robotics projects.
- Edge computing and lightweight server applications.

Raspberry Pi 5B

1. Specifications:

- **Processor:** Broadcom BCM2712, quad-core Cortex-A76 (ARMv8-A) 64-bit SoC at 2.0 GHz.
- **Memory:** Options of 4GB, 8GB, or 16GB LPDDR4X SDRAM.
- **Networking:** Dual 2.5Gb Ethernet ports, 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac/ax wireless LAN, Bluetooth 5.2, BLE.
- **Ports:** 2 USB 3.0 ports, 2 USB 2.0 ports, 2 micro HDMI ports (up to 4Kp60 supported), 1 USB-C power port, 1 3.5mm audio/composite video jack, 1 camera interface (CSI), 1 display interface (DSI), PCIe 2.0 interface.
- **GPIO:** 40-pin GPIO header.
- **Storage:** microSD slot, PCIe M.2 support for NVMe SSDs.
- **Power:** 5V/5A DC via USB-C connector.

2. Key Features:

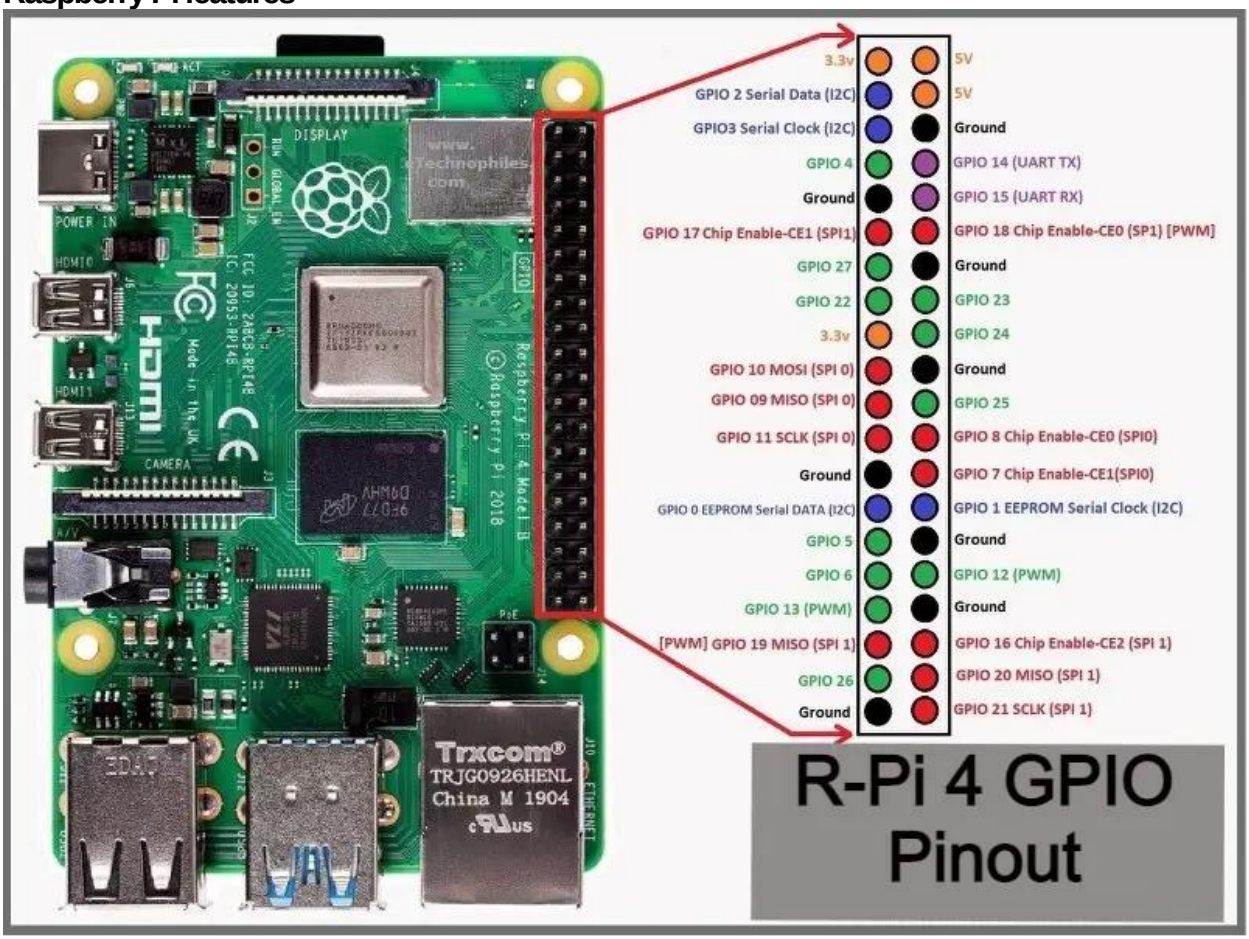
- Major boost in processing power with Cortex-A76 cores.
- Higher memory options, up to 16GB, catering to memory-intensive applications.

- Dual 2.5Gb Ethernet ports for advanced networking capabilities.
- Support for PCIe M.2 NVMe SSDs, allowing for faster storage options.
- Wi-Fi 6 support for enhanced wireless connectivity.

3. Use Cases:

- Advanced desktop replacement with high multitasking capabilities.
- Industrial applications requiring robust performance and connectivity.
- Advanced IoT projects with higher data throughput needs.
- Network-attached storage (NAS) and other server applications.
- AI and machine learning projects that benefit from increased processing power and memory.

Raspberry Pi features



- **Central processing unit.** The brains of the computer, the CPU will receive commands and execute them accordingly.
- **HDMI port.** Without this, there's no way to see what you're doing. An HDMI port is needed to connect to whichever screen you choose.
- **Graphics processing unit.** A GPU is what helps translate calculations into images.
- **Random-access memory.** RAM is important for accessing real-time information. The original Raspberry Pi only had 256MB of RAM. The Raspberry Pi with the largest capacity holds 8GB of RAM.

- **Ethernet port.** An Ethernet port is necessary for connecting to the internet. Access to the internet is paramount for keeping on top of software updates.
- **SD card slot.** Most computers have a built-in harddrive. The Raspberry Pi, however, does not. Therefore, a built in slot for an SD storage card is necessary. The SD card will act as the Raspberry Pi's harddrive.
- **General purpose input/output pins.** GPIO pins are located on one side of the circuit board and look like a small nest of pins. These pins are used to communicate and interact with other circuits.
- **LEDs.** These lights function as status indicators for several functions. There are five different lights:
    - **PWR.** This LED is red and indicates when the power supply is on.
    - **ACT**. This LED is green and indicates when the SD card is in use.
    - **LNK**. This LED is orange and indicates when there is an active internet connection.
    - **100.** This LED is orange and indicates when the internet connection has reached 100Mbps.
    - **FDX.** This LED is orange and indicates when the internet connection is full-duplex, which means it can handle simultaneous data communication between two parties.

Function to Program Hardware

- **Import Rpi.GPIO**
- **Import time.sleep**
- **GPIO.setmode**
- **GPIO.setup**
- **GPIO.output**
- **GPIO.input**

Importing Necessary Libraries
import RPi.GPIO as GPIO:
This statement imports the RPi.GPIO library and allows you to interact with the GPIO pins of the Raspberry Pi using Python. The as GPIO part is an alias to shorten the calls to the functions within the library.

import time:
This imports the time library, which provides various time-related functions such as sleep, which pauses the execution of the program for a specified amount of time.

Example

```
import RPi.GPIO as GPIO
import time
```

Setting Up GPIO Mode
GPIO.setmode(GPIO.BCM):
This sets the pin numbering mode. The GPIO.BCM mode uses the Broadcom SOC channel numbering. Alternatively, you can use GPIO.setmode(GPIO.BOARD) to use the physical pin numbers on the GPIO header.
Example

```
GPIO.setmode(GPIO.BCM)
```

Setting Up GPIO Pins
GPIO.setup(pin_number, GPIO.OUT):
This configures a specific pin as an input or output. The pin_number specifies the GPIO pin to configure, and GPIO.OUT sets the pin as an output. To set a pin as an input, you would use GPIO.IN.
Example

```
led_pin = 18  # Broadcom pin 18
GPIO.setup(led_pin, GPIO.OUT)
```

Contrlling GPIO Pin
GPIO.output(pin_number, GPIO.HIGH):
This sets the specified GPIO pin to a high state (3.3V). GPIO.LOW can be used to set the pin to a low state (0V).

Example

```
GPIO.output(led_pin, GPIO.HIGH)  # Turn on the LED
time.sleep(1)                    # Wait for 1 second
GPIO.output(led_pin, GPIO.LOW)   # Turn off the LED
time.sleep(1)                    # Wait for 1 second
```

Reading GPIO Input
7. GPIO.input(pin_number): This reads the state of the specified GPIO pin. It returns GPIO.HIGH if the pin is high and GPIO.LOW if it is low.

Example

```
button_state = GPIO.input(button_pin)
if button_state == GPIO.LOW:
    print("Button pressed")
else:
    print("Button not pressed")
```

Advantages

- The Raspberry Pi is a small computer that is roughly the size of a credit card.
- The Raspberry Pi is inexpensive.
- Using a group of Raspberry Pi's to function as a server is more efficient than using a regular server.
- The Raspberry Pi is ideal for adaptive technology because it can display visuals and play movies.
- This microcomputer can be used by small businesses on a tight budget to use their product or build new technology that integrates the product.

Disadvantages

- It is not a computer replacement, and the processor is not as fast. Downloading and installing software takes time so that you won't do any intricate multitasking.
- Other operating systems, such as Windows, are incompatible.
- If the extra work is worth it, business owners should think about it.
- This product will not be beneficial for larger businesses that already have large servers, which can perform all of the Raspberry Pi tasks. As a result, it would not be worth it, and putting everything together would take time.

Experiment No - 3

TITLE: OS installation, reading it from network using Wifi & SSH

AIM: OS installation, reading it from network using Wifi & SSH, using SFTP upload file from PC

HARDWARE REQUIREMENT

- SD card
- SD card reader
- Raspberry pi
- Power adapter

SOFTWARE REQUIREMENT

- SD card formatter,
- Rpi imager,
- Filezilla,
- Putty,
- Windows OS
- Raspbian OS

THEORY

FileZilla is a popular FTP (File Transfer Protocol) client used for transferring files between a client (like a computer) and a server. Raspberry Pi, on the other hand, is a small, affordable computer that can be used for various applications, including acting as an FTP server.

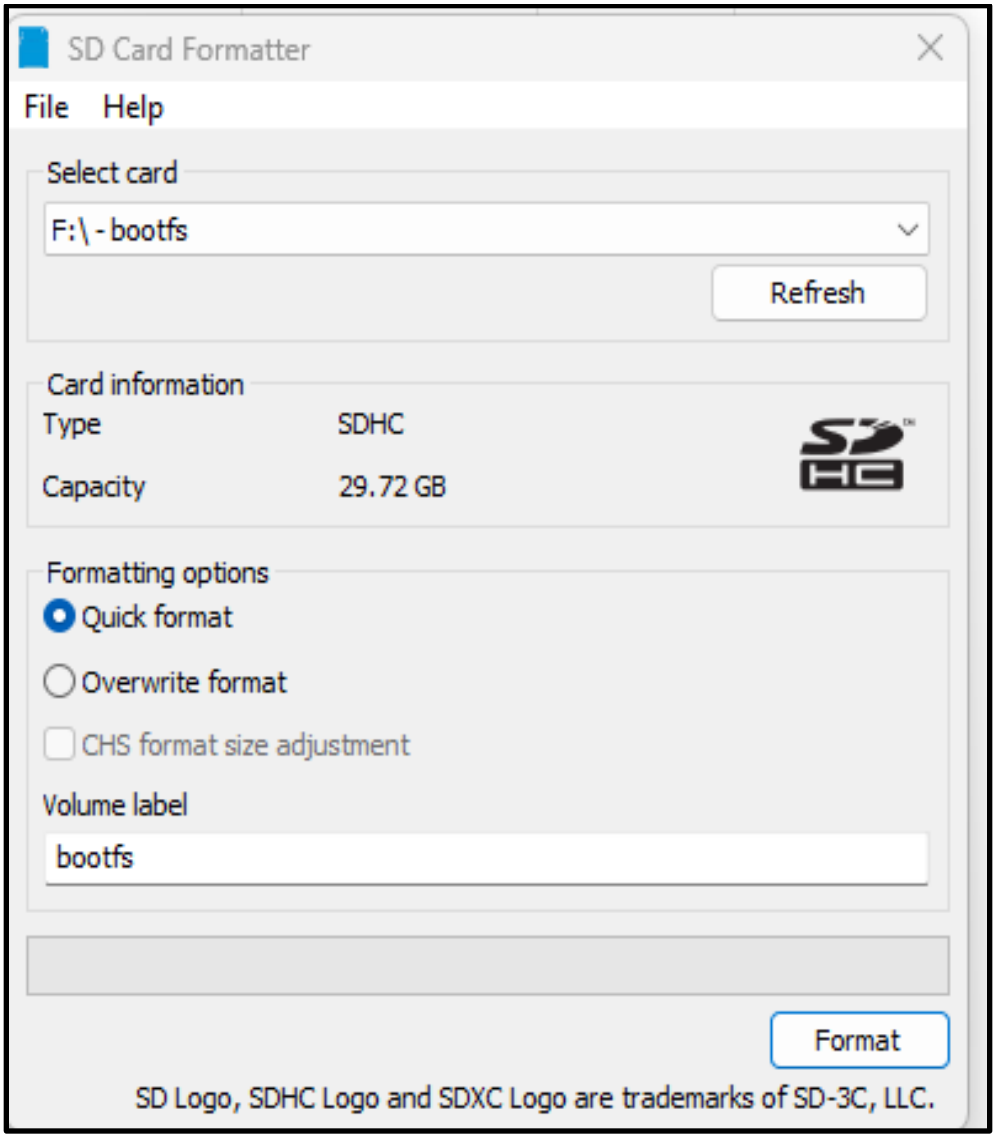
Connecting FileZilla to Raspberry Pi

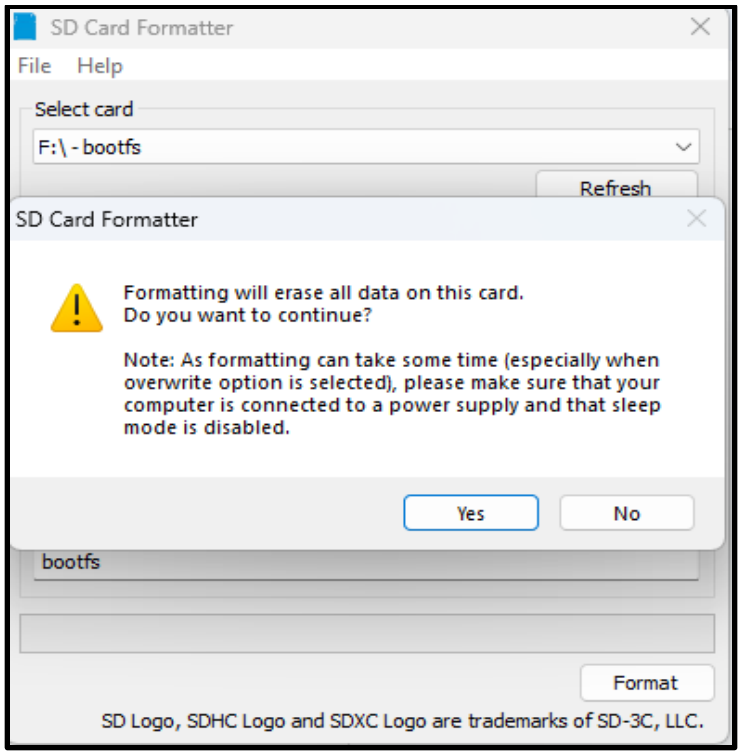
1. **Launch FileZilla:**
    - Open FileZilla on your computer.
2. **Connect to Raspberry Pi:**
    - In the FileZilla interface:
        - Enter the IP address of your Raspberry Pi in the "Host" field.
        - Enter the username and password configured on your Raspberry Pi's FTP server.
        - Set the "Port" to 22 (the default FTP port) unless you've configured a different port on your Raspberry Pi.
        - Click on "Quickconnect" to initiate the connection.
3. **Navigating and Transferring Files:**
    - Once connected, you will see two main panels in FileZilla:
        - The left panel represents files on your local computer.
        - The right panel displays files and directories on your Raspberry Pi.
4. **Upload Files to Raspberry Pi:**
    - Navigate on the left panel to locate the files you want to transfer.
    - Drag and drop files from the left panel (your computer) to the right panel (Raspberry Pi) to upload them.

5. **Download Files from Raspberry Pi (Optional):**
    - You can also download files from your Raspberry Pi to your local computer by dragging them from the right panel to the left panel in FileZilla.

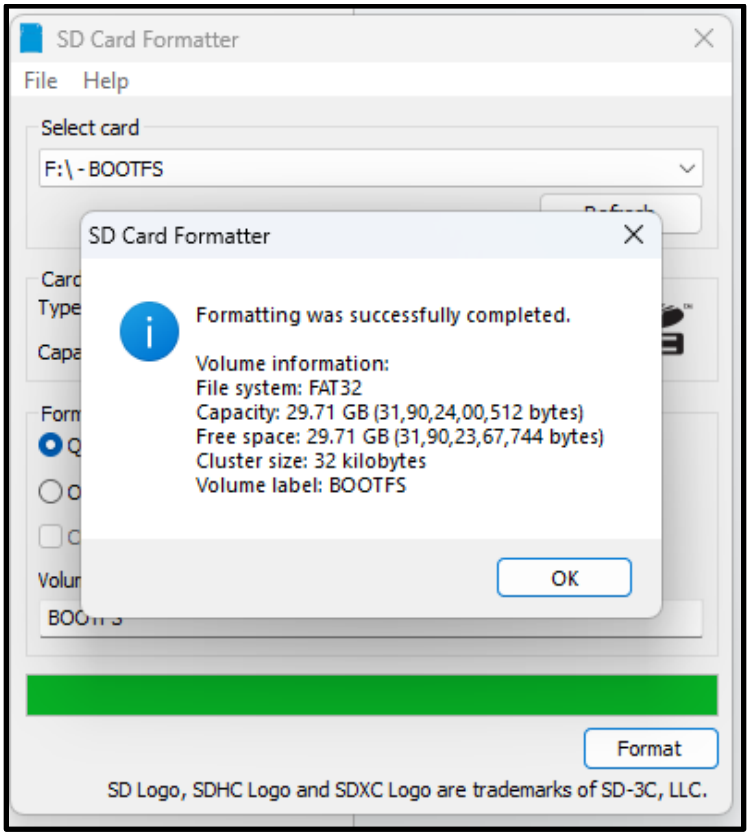
STEPS

1. Connect SD Card to CPU.
2. Go to Explorer open SD card folder.
3. Go to Search and type SD Card Formatter you'll get a popup click on "Format"

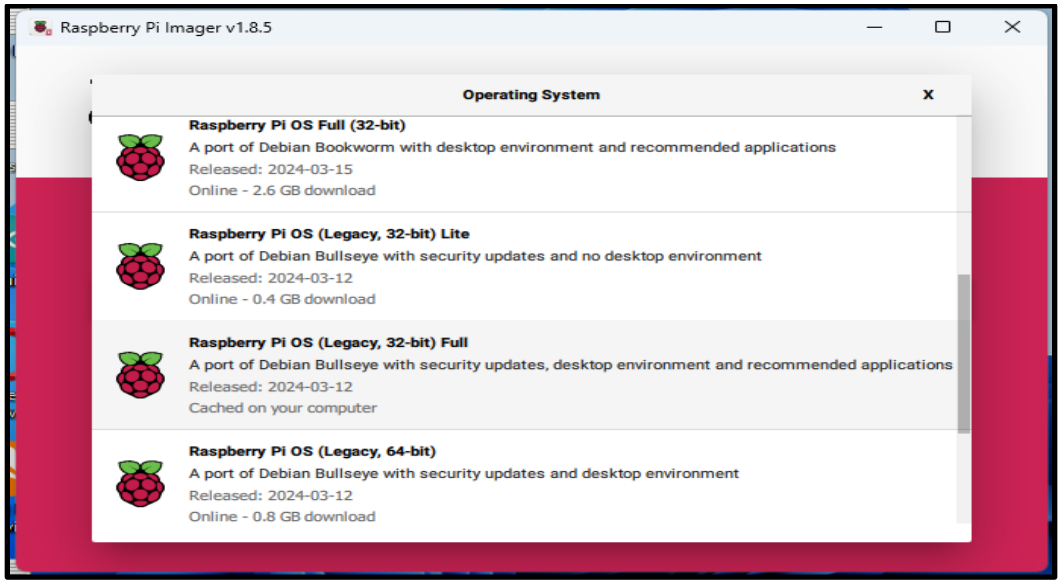




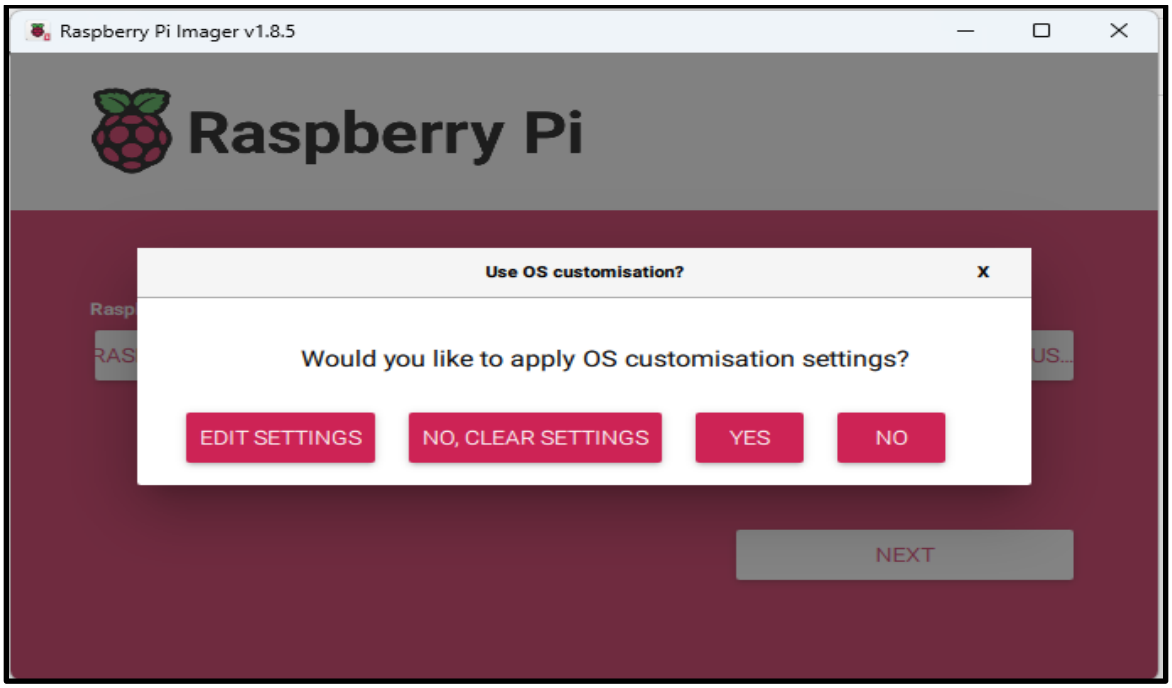
4. Erase the card data. You'll get the successful message at the end.



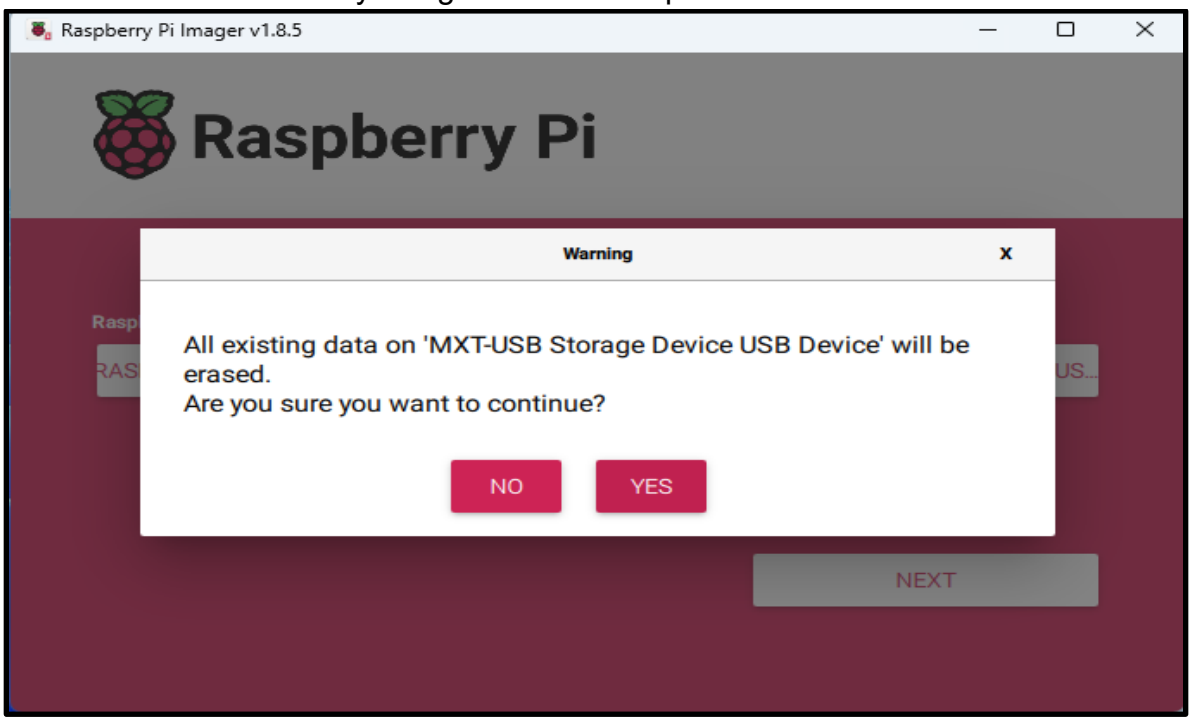
5. Connect Power cable to RPI & Ethernet to Gigabit Ethernet slot.
6. Go to search and start RPi imager.
7. Click on choose device select Raspberry Pi 4.



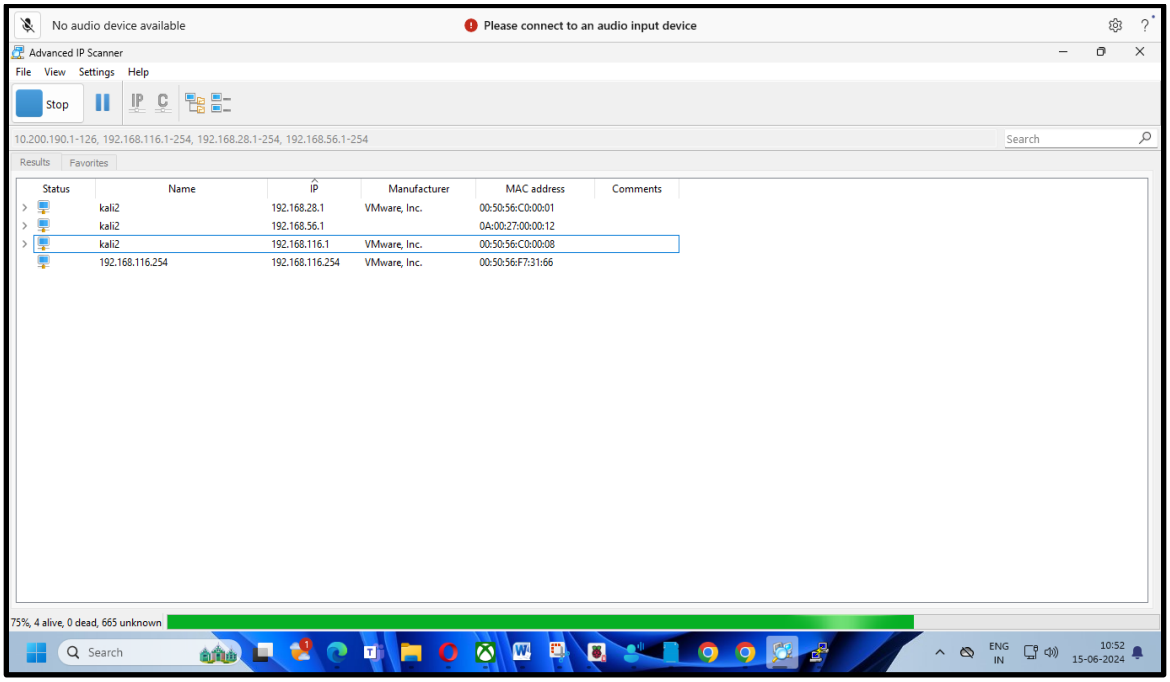
8. Click on Choose OS>Raspberry Pi OS Others>Click on RPi OS (legacy 32-bits) full 2.4Gb .
9. Choose Storage and select the SD card storage>Edit settings>OS customization>General>set Username- pi, password-Raspberry>Click on save>popup message appears-click on Yes



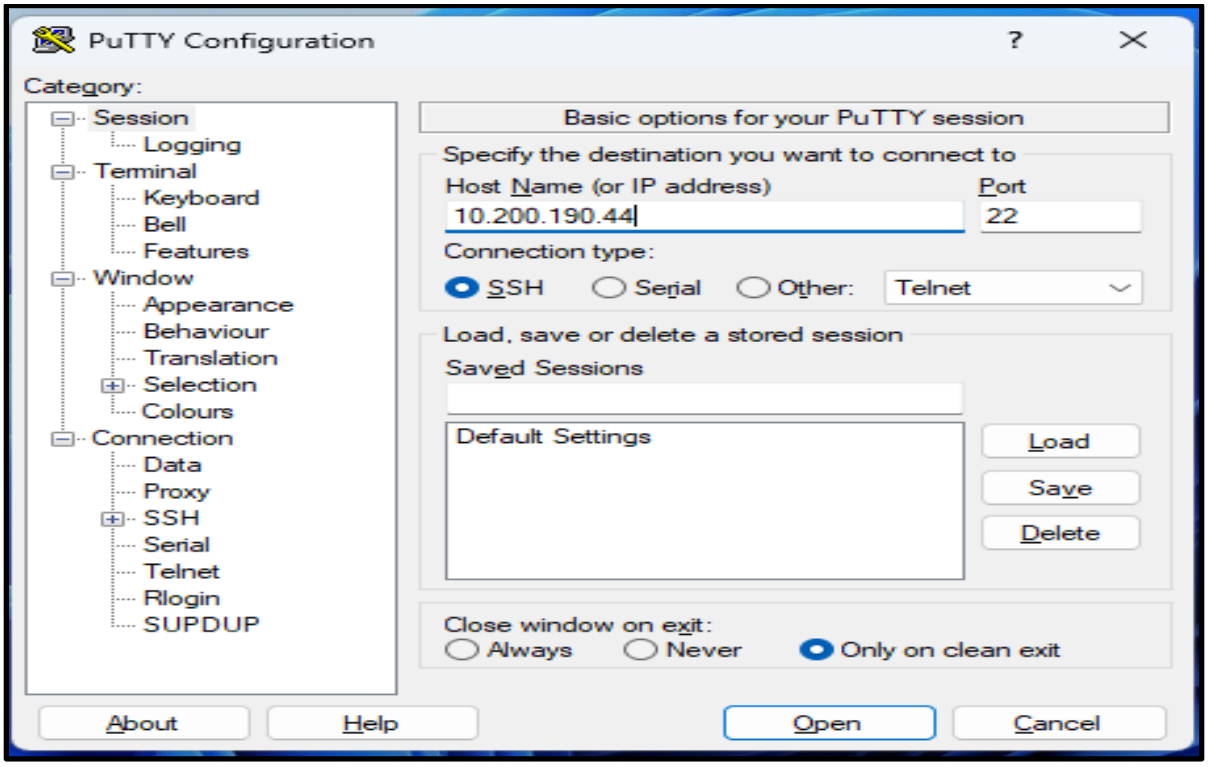
10. After verification is done you'll get a continue option click on it



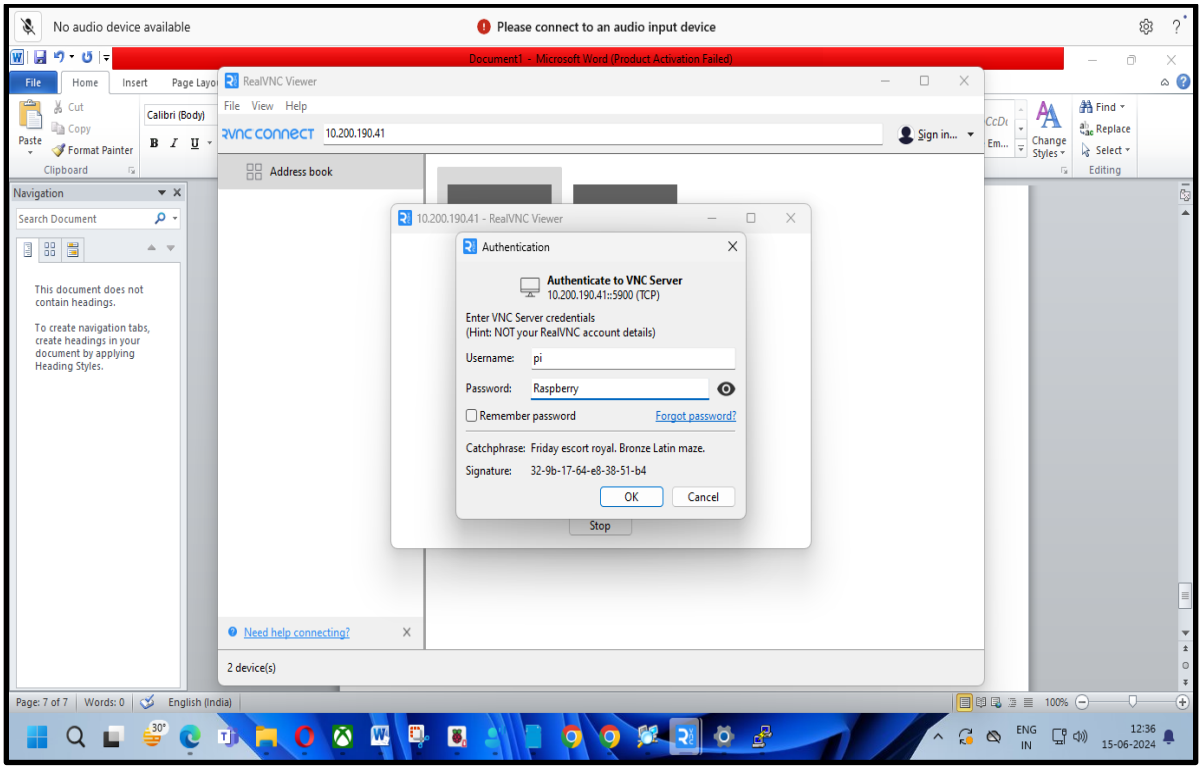
11. Start Advance ip>start scanning and you will get IP address of your Pi



12. Open Putty>Add host IP address>click on Open>putty console will start>

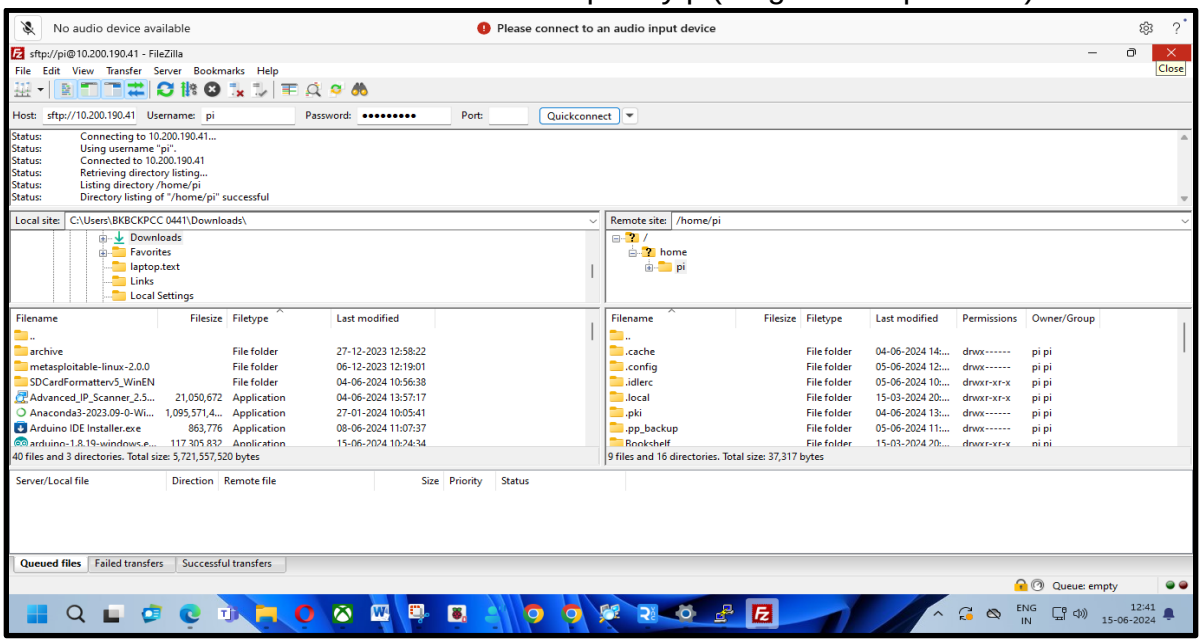


13. Open Real VNC viewer then insert the IP address.

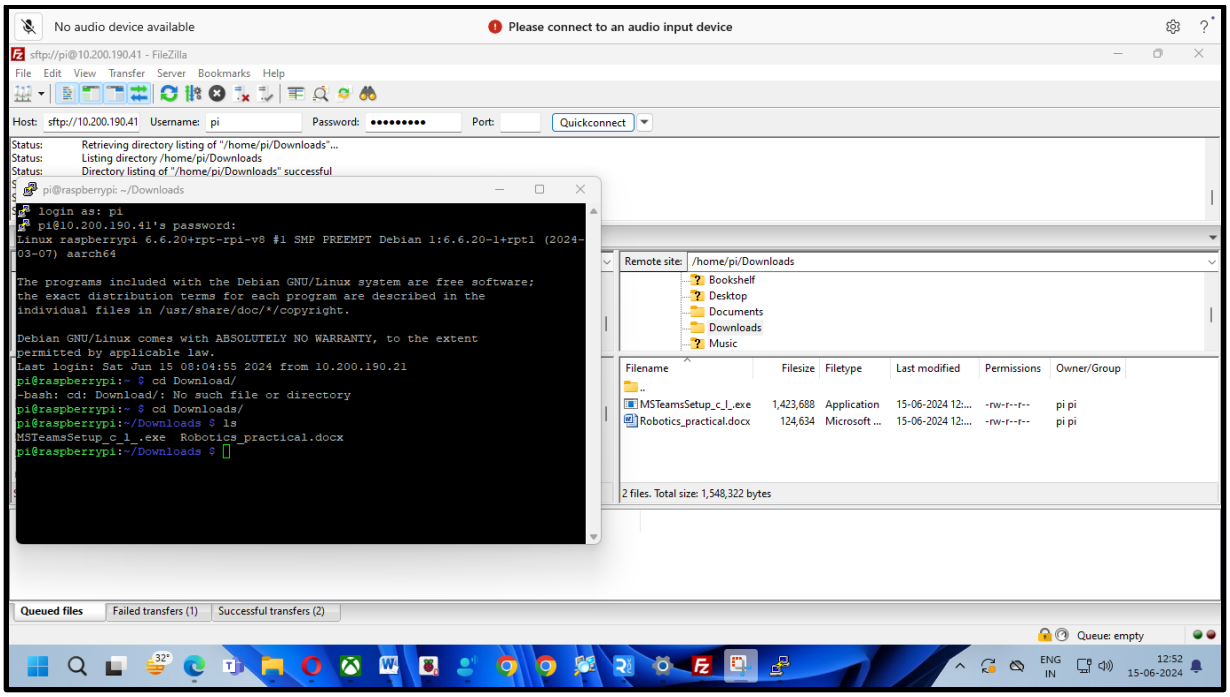


14. Then enter user name (pi)and password as (Raspberry)
15. Download and install Filezilla and open it.

16. Insert host name(ip add), password(Raspberry), port (22)
17. Transfer the file from PC windows to Raspberry pi(drag and drop the file)



18. Then open Putty and confirm the file is successfully transfer or not
19. Type command - cd foldername/ (for change directory)
    - ls (for list of directory)
20. Output will be come.


No audio device available
Please connect to an audio input device
sftp://pi@10.200.190.41 - FileZilla
File Edit View Transfer Server Bookmarks Help
Host: sftp://10.200.190.41 Username: pi Password: ••••••••• Port: Quickconnect
Status: Retrieving directory listing of "/home/pi/Downloads"...
Status: Listing directory /home/pi/Downloads
Status: Directory listing of "/home/pi/Downloads" successful
pi@raspberrypi: ~/Downloads
login as: pi
pi@10.200.190.41's password:
Linux raspberrypi 6.6.20+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.20-1+rpt1 (2024-03-07) aarch64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jun 15 08:04:55 2024 from 10.200.190.21
pi@raspberrypi:~ $ cd Download/
-bash: cd: Download/: No such file or directory
pi@raspberrypi:~ $ cd Downloads/
pi@raspberrypi:~/Downloads $ ls
MSTeamsSetup_c_l_.exe Robotics_practical.docx
pi@raspberrypi:~/Downloads $
Remote site: /home/pi/Downloads
Bookshelf
Desktop
Documents
Downloads
Music
Filename Filesize Filetype Last modified Permissions Owner/Group
MSTeamsSetup_c_l_.exe 1,423,688 Application 15-06-2024 12:... -rw-r--r-- pi pi
Robotics_practical.docx 124,634 Microsoft ... 15-06-2024 12:... -rw-r--r-- pi pi
2 files. Total size: 1,548,322 bytes
Queued files
Failed transfers (1)
Successful transfers (2)
Queue: empty
ENG IN
12:52 15-06-2024

Experiment No 4 : Testing Sensor and Motor

Title: Testing sensor and Motor(Rpi)

Aim: Develop python code for testing sensor and motor(Rpi).

Hardware Requirements: Windows, Raspberry pi, Adapter, PIR Sensor, LDR Sensor, Buzzer, Gas sensor, Motors

Software Requirements: Rpi OS, Python IDLE,

Theory:

Raspberry Pi 4:

The Raspberry Pi 4 is a versatile single-board computer developed by the Raspberry Pi Foundation. It features significant improvements over its predecessors, including:

1. ***Processor*:** It is powered by a Broadcom BCM2711 quad-core Cortex-A72 (ARMv8) 64-bit SoC running at 1.5GHz.
2. ***Memory*:** Options for 2GB, 4GB, or 8GB of LPDDR4-3200 SDRAM, providing improved multitasking capability compared to earlier models.
3. ***Connectivity*:** It includes dual-band 802.11ac Wi-Fi, Bluetooth 5.0, Gigabit Ethernet, and support for multiple USB ports (2x USB 3.0, 2x USB 2.0).
4. ***Video Output*:** Capable of driving two monitors at 4K resolution via HDMI ports, and supports hardware-accelerated video decoding.
5. ***Storage*:** MicroSD card slot for storage, offering flexibility in storage size and type.
6. ***GPIO*:** Maintains the 40-pin GPIO header for interfacing with sensors, LEDs, and other external devices.
7. ***Operating System*:** Supports a variety of operating systems, including Raspberry Pi OS (formerly Raspbian), Ubuntu, and others, making it versatile for different applications.
8. ***Applications*:** Widely used in projects ranging from educational tools to IoT applications, media centers, and even as a basic desktop replacement.

Overall, the Raspberry Pi 4 provides enhanced performance and connectivity options compared to earlier models, making it a powerful tool for both beginners and advanced users in the maker community.

PIR Sensor

A PIR (Passive Infrared) sensor is an electronic device that detects infrared radiation emitted by warm objects within its field of view. It operates on the principle that all objects with a temperature above absolute zero emit heat in the form of radiation. Inside the sensor, a pyroelectric material generates a voltage when exposed to infrared radiation, which changes when an object moves across the sensor's view. This change triggers the sensor to output a signal, typically used for motion detection applications. PIR sensors

are widely employed in security systems, automatic lighting controls, and occupancy sensing due to their low cost, reliability, and ability to detect human presence without emitting any energy themselves.

Fig. PIR Sensor

LDR Sensor

An LDR (Light Dependent Resistor) sensor, also known as a photoresistor, is a passive electronic component that changes its resistance based on the amount of light falling on its surface. The resistance of an LDR decreases as the intensity of incident light increases, and vice versa. This property makes LDRs useful for light sensing applications where they can be used to detect ambient light levels. Commonly used in circuits for automatic lighting controls, street lights, and photography light meters, LDR sensors are simple and cost-effective solutions for measuring and responding to changes in light intensity in various environments.

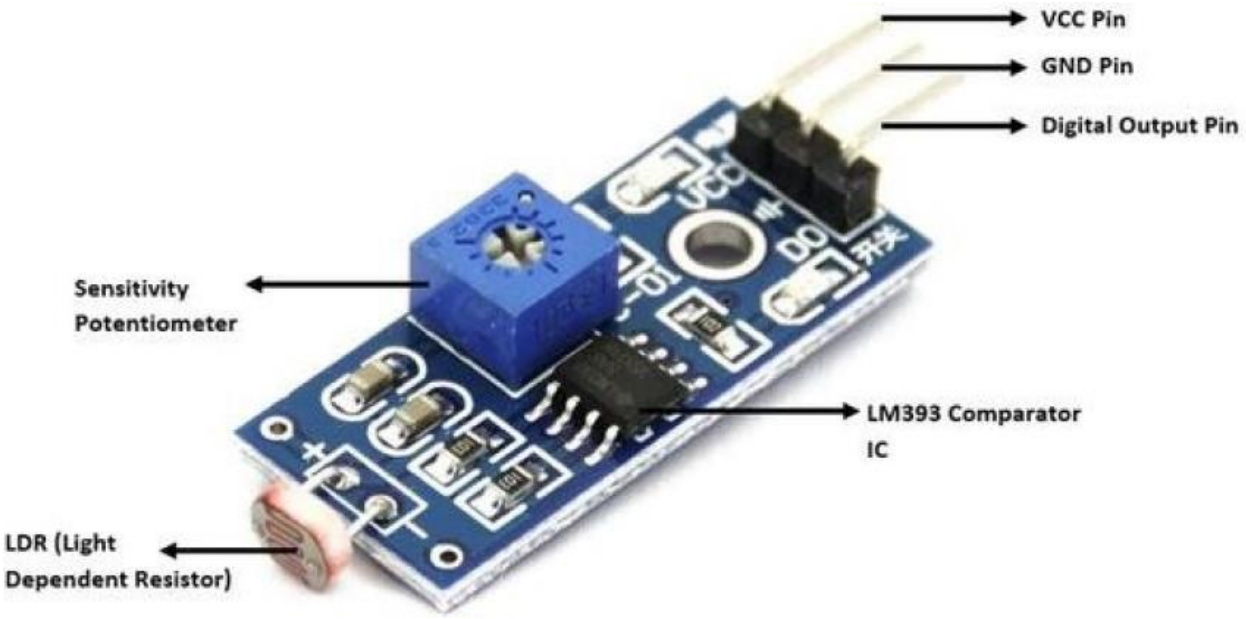


Fig. LDR(Light Sensor)

Buzzer

A buzzer sensor is an electronic device that emits audible sound signals when activated by an electrical signal. Typically composed of a piezoelectric element or an electromagnetic coil, buzzer sensors convert electrical energy into mechanical vibrations that produce sound waves. They are commonly used in alarm systems, electronic devices, and industrial equipment to alert users of specific events or conditions. Buzzer sensors vary in sound frequency, volume, and duration based on their design and intended application, providing versatile auditory feedback in various environments where visual indicators may not be sufficient or practical.

Fig. Buzzer

Gas Sensor

A gas sensor is an electronic device designed to detect and quantify the presence of specific gases in the surrounding environment. These sensors utilize various detection principles such as chemical reactions, conductivity changes, or optical absorption to determine gas concentrations. Common gases monitored include carbon monoxide (CO),

methane (CH4), ammonia (NH3), and volatile organic compounds (VOCs). Gas sensors are critical in industrial settings for ensuring safety by detecting hazardous gas leaks, in environmental monitoring to assess air quality, and in consumer applications such as smart home devices for detecting smoke and carbon monoxide. They provide real-time data to alert users or control systems, enabling timely responses to potential threats or environmental concerns.

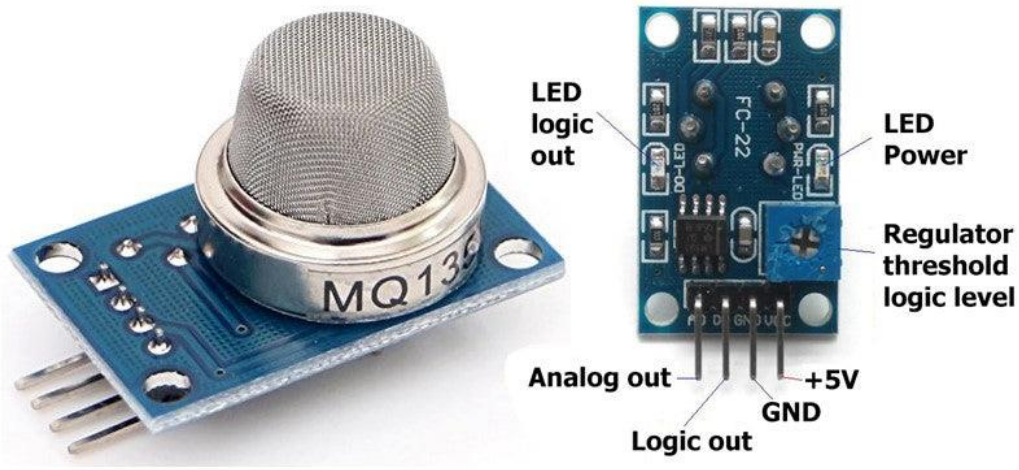


Fig. Gas Sensor

Obstacle Sensor

Infrared Obstacle Sensor Module has a built-in **IR transmitter** and **IR receiver** that sends out IR energy and looks for reflected IR energy to detect the presence of any obstacle in front of the sensor module. The module has an onboard potentiometer that lets users adjust the detection range. The sensor has a very good and stable response even in ambient light or in complete darkness.

The Obstacle Avoidance Sensors usually come in two types — with 3 and 4 pins.

The IR transmitter sends an infrared signal that, in case of a reflecting surface (e.g. white color), bounces off in some directions including that of the IR receiver that captures the signal detecting the object. When the surface is absorbent the IR signal isn't reflected and the object cannot be detected by the sensor. This result would occur even if the object is absent.

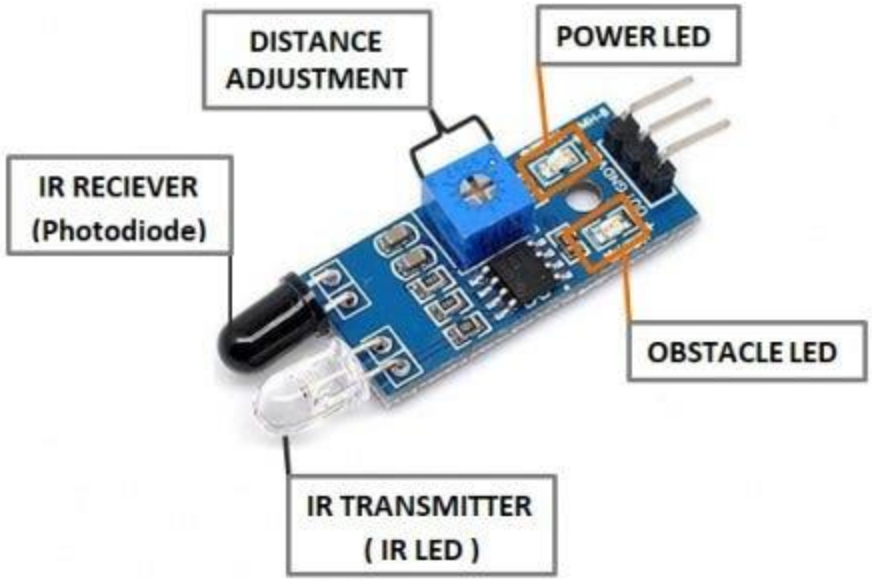
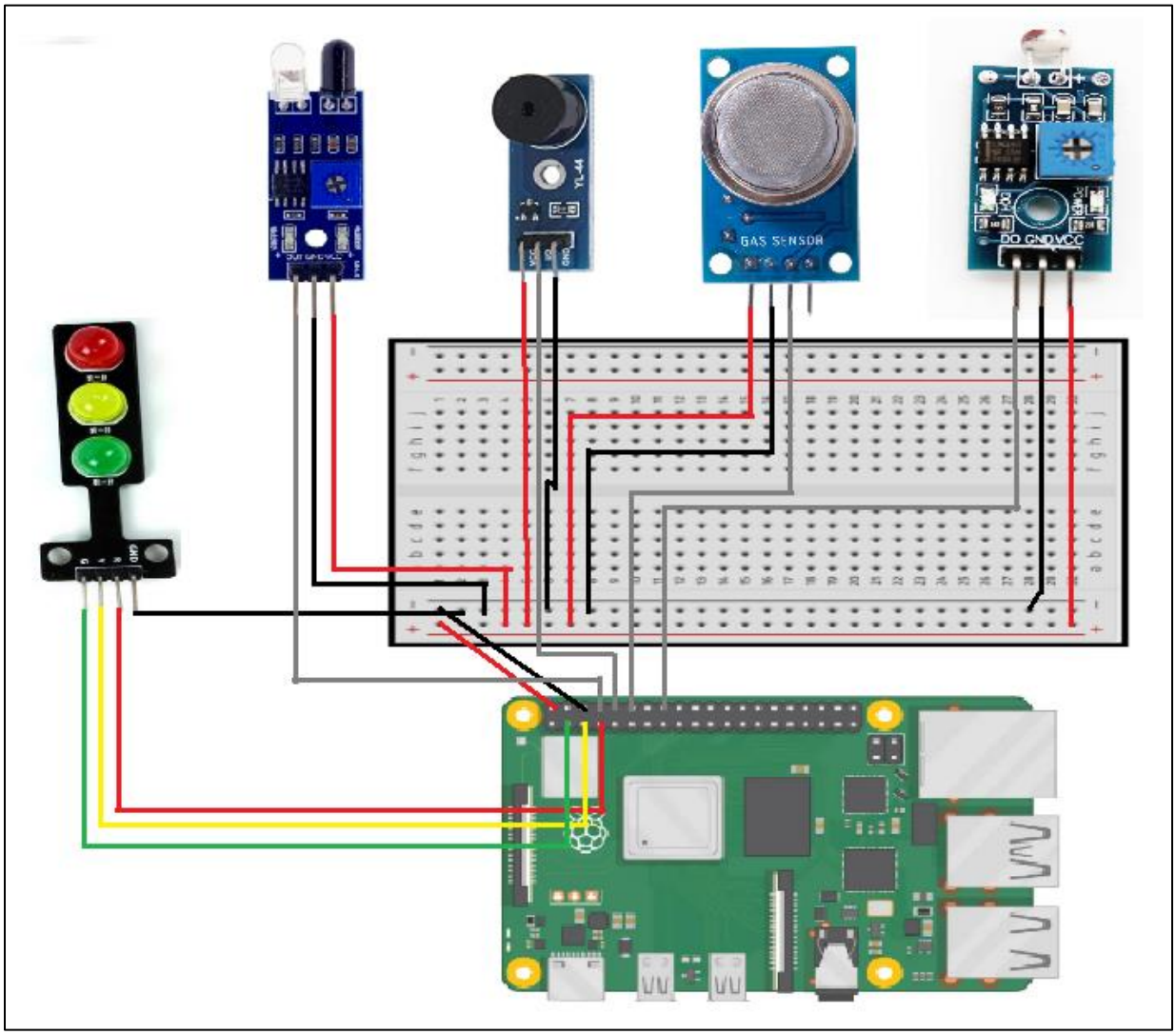
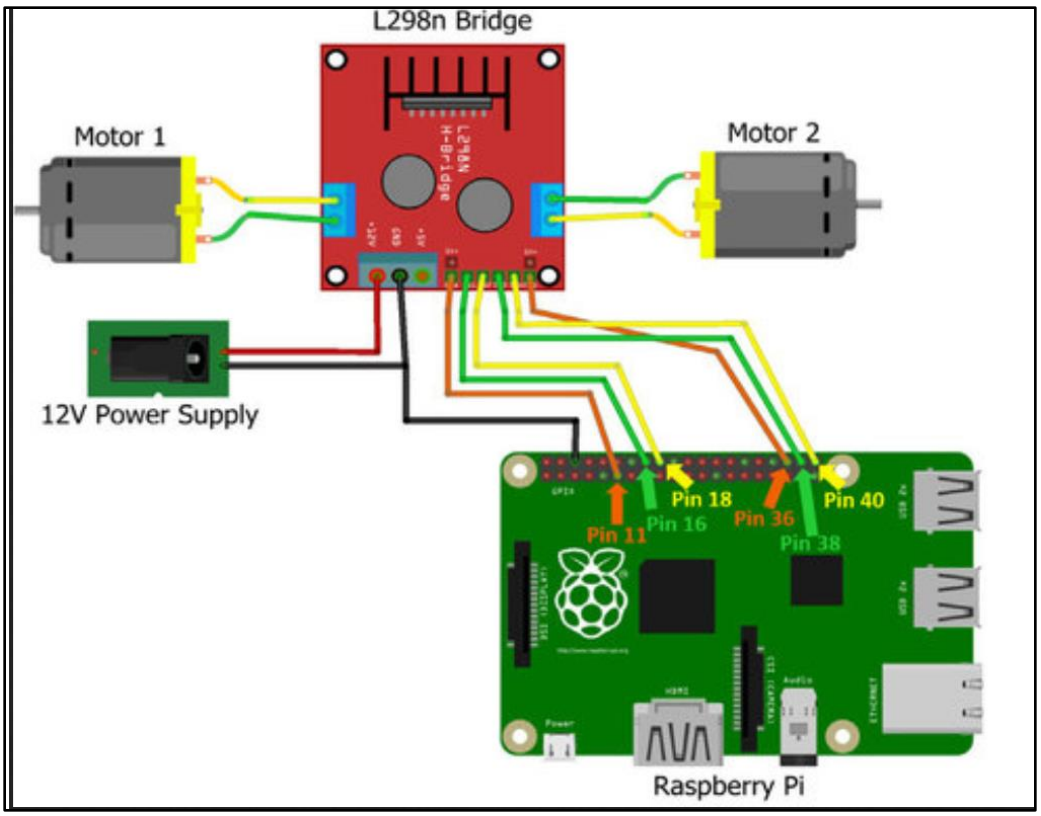


Fig. Obstacle Sensor

The module has 3 pins — Vcc, GND, and the output. We also have 2 LEDs, one is for the power that is it turns ON when it is connected to the power. The other is for obstacle detection.

Circuit Diagram





Python Code for sensors

```
import RPi.GPIO as GPIO
import time

rled = 7
oled = 5
gled = 3
buzzer = 10
light = 16
obstacle = 8
gas = 12

GPIO.setmode(GPIO.BCM)
GPIO.setup(rled , GPIO.OUT)
GPIO.setup(oled , GPIO.OUT)
GPIO.setup(gled , GPIO.OUT)
GPIO.setup(buzzer , GPIO.OUT)
GPIO.setup(light, GPIO.IN)
GPIO.setup(obstacle, GPIO.IN)
GPIO.setup(gas, GPIO.IN)
GPIO.output(buzzer,false)
```

```
try:
    while True:
    lightsense = GPIO.input(light)
    obstaclesense = GPIO.input(obstacle)
    gassense = GPIO.input(gas)
    if lightsense==False:
        GPIO.output(rled,True)
        GPIO.output(gled,True)
        GPIO.output(oled,True)
        print("Light sensed")
    if lightsense==True:
        GPIO.output(rled,False)
        GPIO.output(gled,False)
        GPIO.output(oled,False)
        print("Light not sensed")
    if obstaclesense==True:
        GPIO.output(buzzer,True)
        print("Obstacle sensed")
        time.sleep(0.25)
        GPIO.output(buzzer,False)
        print("Obstacle sensed")
        time.sleep(0.25)
    if obstaclesense==False:
        GPIO.output(buzzer,False)
        print("Obstacle not sensed")
    if gassense==True:
        GPIO.output(buzzer,True)
        print("Gass Sensed")
        time.sleep(0.5);
        GPIO.output(buzzer,False)
        print("Gas Sensed")
        time.sleep(0.5);
    if gassense==False:
        GPIO.output(buzzer,False)
        print("Gass not sensed")

except KeyboardInterrupt:
    GPIO.cleanup()
```

Python Code for Testing Motors:

```
import RPi.GPIO as GPIO
mport time
```

```
# Define GPIO pins
m11 = 16
m12 = 18
m21 = 38
m22 = 40

GPIO.setmode(GPIO.BCM)
GPIO.setup(m11, GPIO.OUT)
GPIO.setup(m12, GPIO.OUT)
GPIO.setup(m21, GPIO.OUT)
GPIO.setup(m22, GPIO.OUT)

try:
    while True:
        GPIO.output(m11, GPIO.HIGH)
        GPIO.output(m12, GPIO.LOW)
        GPIO.output(m21, GPIO.HIGH)
        GPIO.output(m22, GPIO.LOW)
        time.sleep(5)

        GPIO.output(m11, GPIO.LOW)
        GPIO.output(m12, GPIO.HIGH)
        GPIO.output(m21, GPIO.LOW)
        GPIO.output(m22, GPIO.HIGH)
        time.sleep(5)

        GPIO.output(m11, GPIO.HIGH)
        GPIO.output(m12, GPIO.LOW)
        GPIO.output(m21, GPIO.LOW)
        GPIO.output(m22, GPIO.HIGH)
        time.sleep(5)

        GPIO.output(m11, GPIO.LOW)
        GPIO.output(m12, GPIO.HIGH)
        GPIO.output(m21, GPIO.HIGH)
        GPIO.output(m22, GPIO.LOW)
        time.sleep(5)

except KeyboardInterrupt:
    GPIO.cleanup()
```

Experiment No 05

AIM: Write a script to follow predetermined path using ESP32

COMPONENTS:

Hardware Requirements:

1. **ESP32 Development Board:** The ESP32 serves as the brain of the robot, handling communication, sensor data processing, and motor control. ESP32 board has sufficient processing power to handle sensor data and motor control tasks concurrently. It has sufficient GPIO pins for connecting motor drivers, sensors, and other peripherals.

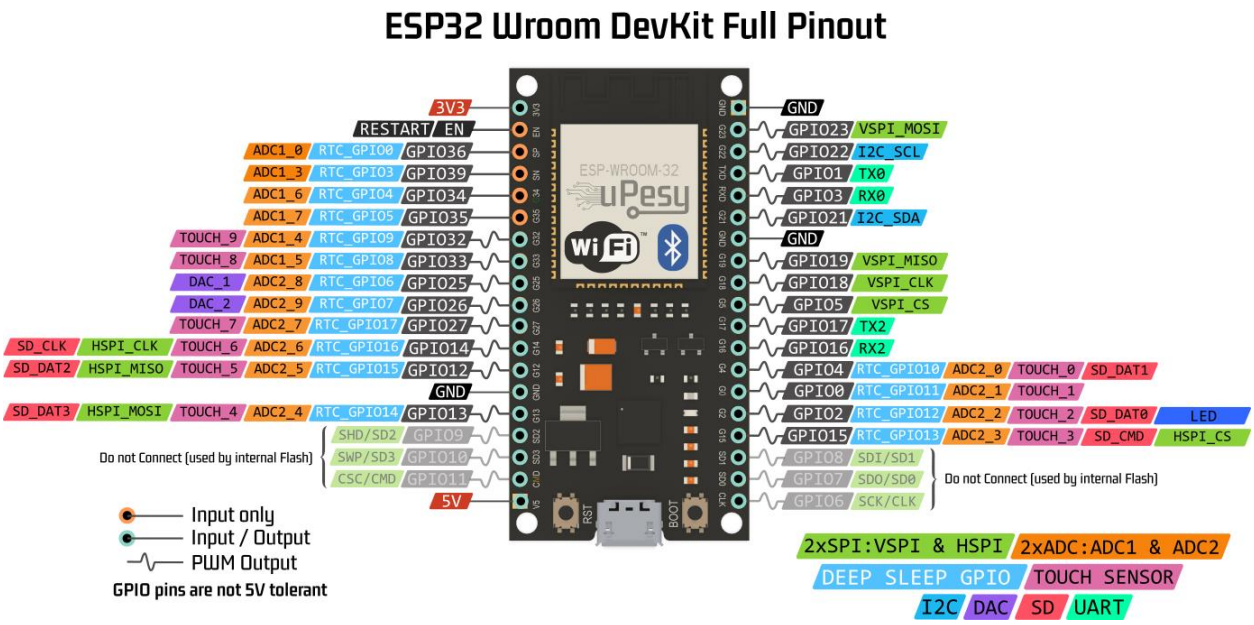


Fig.ESP32 Devkit

2. **L298N Motor Driver:** It control the speed and direction of DC motors, essential for the robot's movement. The L298N is a dual H-bridge motor driver IC commonly used to control DC motors. It can drive two motors simultaneously with bidirectional control (forward, reverse, and braking) using pulse width modulation (PWM) signals. The module also includes built-in diodes for motor protection against back electromotive force (EMF) and can handle peak currents up to 2A per channel (with proper heatsinking).

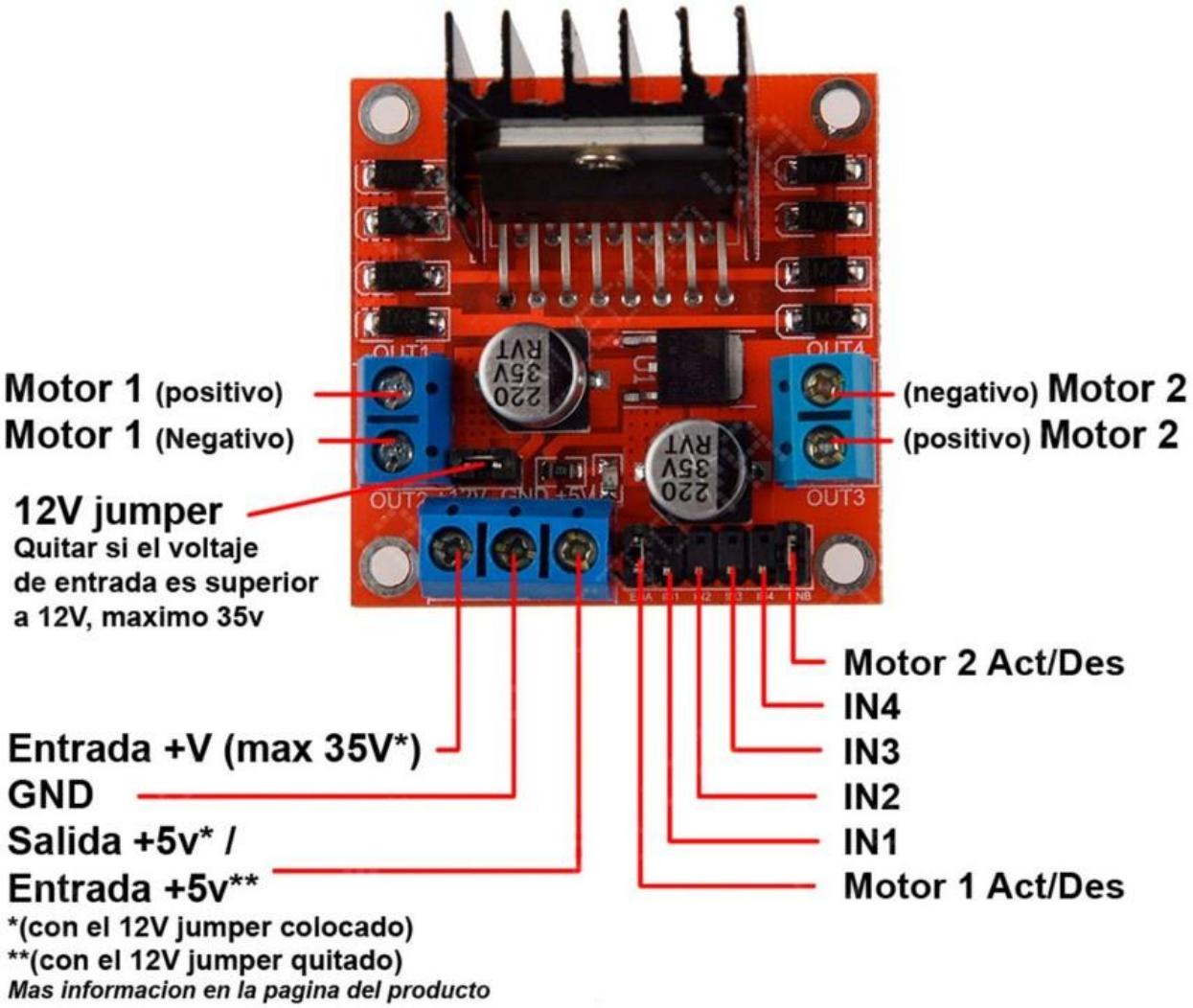


Fig.L298N Motor Driver

3. **Gearmotor:** It is a specialized type of DC motor integrated with a gearbox, designed primarily for hobbyist and DIY projects. The gearbox, often made of plastic or metal gears, serves to reduce the motor's speed while increasing its torque output. This reduction in speed and increase in torque make hobby gear motors particularly suitable for applications where precise control over movement and power efficiency are necessary, such as in robotics, RC vehicles, and small mechanical systems. These motors typically operate on low voltages ranging from 3V to 12V, making them compatible with common power sources used in hobby electronics.

Fig. Gear Motor and Wheel Set

4. **Jumper Wires:** These are short, flexible wires with connectors at each end, typically used in electronics prototyping and bread boarding. They facilitate easy and temporary connections between components on a breadboard or between various points on a circuit board. Jumper wires come in different lengths and colors, aiding in organizing and identifying connections within a circuit. They are essential tools for quickly testing and iterating designs without the need for soldering, enabling rapid prototyping and experimentation in electronics projects.

Fig. Jumper Wires

Software Requirements

1. **Windows:** It serves as the operating system platform for running the Arduino Integrated Development Environment (IDE). It provides a user-friendly interface for writing, compiling, and uploading code to Arduino boards. Windows compatibility ensures seamless integration with Arduino drivers and libraries, facilitating development and debugging processes for various electronics and robotics projects.
2. **Arduino Integrated Development Environment (IDE):** It is a software platform for programming Arduino microcontroller boards. It offers a simplified interface with a text editor for writing code, a compiler to convert code into machine language, and tools for uploading the compiled code to Arduino boards via USB or other connections. The Arduino IDE supports a wide range of Arduino-compatible boards and shields, making it accessible for both beginners and experienced developers in the field of electronics and embedded systems.
3. **ESP32**: It is a versatile microcontroller developed by Espressif Systems, offering powerful features suitable for a wide range of IoT (Internet of Things) applications. It integrates WiFi and Bluetooth connectivity, making it capable of wireless communication and network connectivity. The ESP32 microcontroller includes dual-core processors, low-power modes for energy efficiency, and a rich set of peripherals such as ADCs, DACs, SPI, I2C, UART, and PWM, making it suitable for interfacing with various sensors and actuators.
4. **Programming Language**: Typically uses C/C++ with Arduino libraries and ESP32-specific APIs to control motors, read sensor data, and implement navigation algorithms.

THEORY:

1. **Setting Up the Hardware**
    - **Assemble the Robot**: Mount the motors securely on the chassis, ensuring they are properly aligned for smooth movement. Attach wheels to the motors and assemble the chassis according to the design requirements.
    - **Connect Motors and Motor Driver**: Wire the DC motors to the motor driver, which in turn is connected to the ESP32 board. Ensure correct wiring for power (typically +5V or +12V) and ground connections (GND).
    - **Power Supply**: Provide a suitable power source for the ESP32 board and motors. This could be a battery pack or a regulated power supply depending on the voltage requirements of the motors and other components.
2. **Programming the ESP32 with Arduino IDE**
    - **Install Arduino IDE**: Download and install the Arduino IDE from the official Arduino website if not already installed.
    - **Include Necessary Libraries**: Use libraries such as “esp32” or specific motor driver libraries (if applicable) for controlling the motors.

- **Define Pins**: Define the GPIO pins on the ESP32 board to which the motor driver inputs are connected. This is typically done in the beginning of your Arduino sketch.
- **Define Waypoints or Path**: Declare coordinates or waypoints that define the fixed path the robot should follow. These coordinates represent points on a grid or a map that the robot will move between.

3. **Implement Fixed Path Following Logic**
   - **Sequence of Movements**: Program a sequence of movements (e.g., forward, turn left, turn right) based on predefined waypoints or coordinates. This can be implemented using arrays or variables in your Arduino sketch.
   - **Motor Control**: Use Pulse Width Modulation (PWM) signals to control the speed and direction of the motors through the motor driver. Adjust motor speeds based on the desired path curvature or straight-line movement.
   - **Movement Functions**: Implement functions or procedures to move the robot to specific coordinates or waypoints. These functions calculate the necessary movements (distance and direction) required to reach each waypoint from the current position.

4. **Execution and Testing**
   - **Upload Code to ESP32**: Compile and upload the Arduino sketch to the ESP32 board using the Arduino IDE. Verify that the code compiles without errors and upload it to the ESP32.
   - **Field Testing**: Deploy the robot in a controlled environment to test its ability to follow the fixed path accurately and reliably. Observe its movements and ensure it reaches each waypoint correctly. Make adjustments to the code and waypoints as necessary to improve performance.

CIRCUIT DIAGRAM

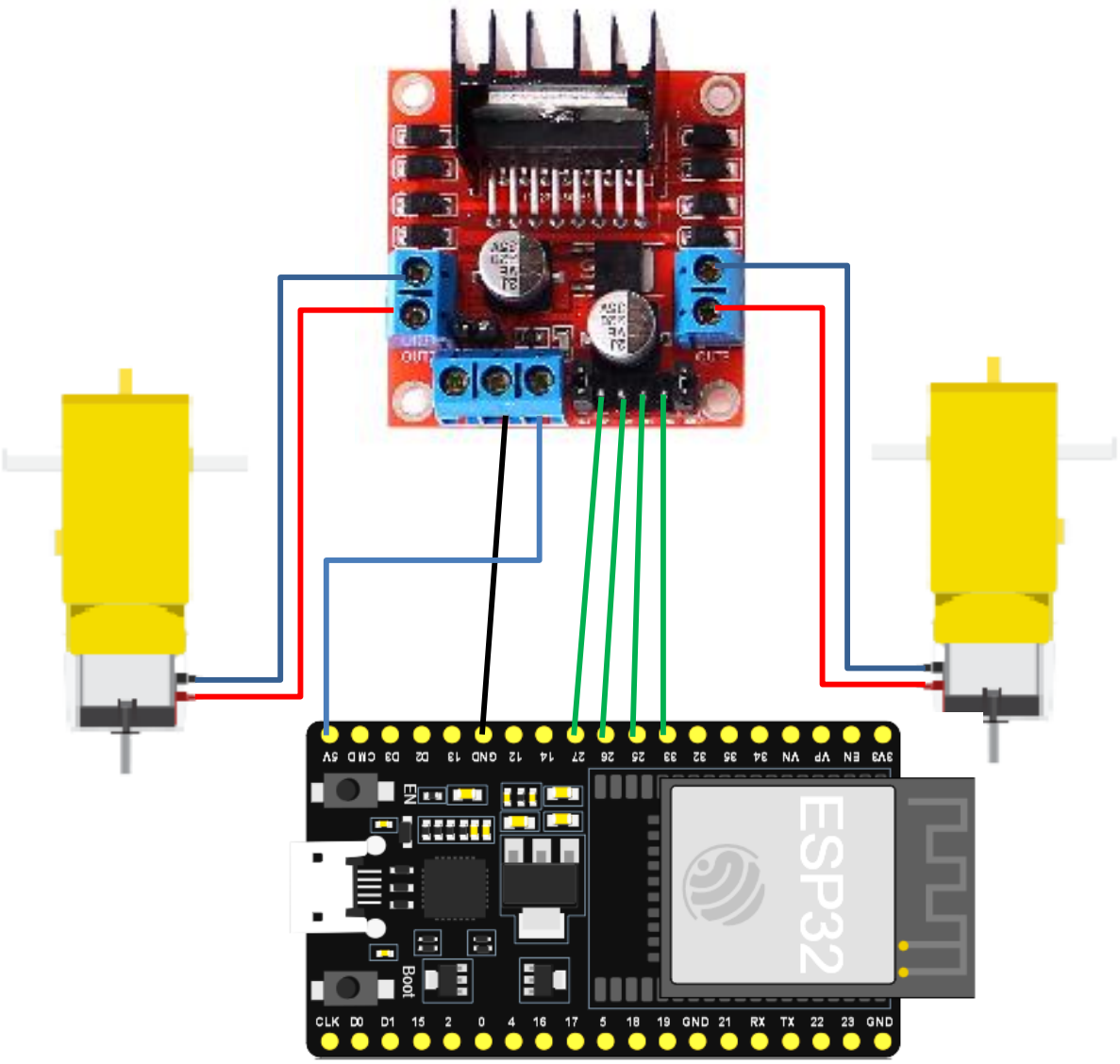


Fig.Pre determined Path Follower

CODE – C Code

```
int m11 = 33;
int m12 = 25;
int m21 = 26;
int m22 = 27;

void setup()
{
  pinMode(m11,OUTPUT);
  pinMode(m12,OUTPUT);
  pinMode(m21,OUTPUT);
```

```
  pinMode(m22,OUTPUT);

}

void loop()
{
 forward();
 delay(3000);

 right();
 delay(500);

 forward();
 delay(3000);

 right();
 delay(500);

 forward();
 delay(3000);

 right();
 delay(500);

 forward();
 delay(3000);

 right();
 delay(500);

 forward();
 delay(3000);

 right();
 delay(500);

 forward();
 delay(3000);

 right();
 delay(500);

 forward();
 delay(3000);
```

```
 motorstop();
 delay(3000);
}

void forward()
{
  digitalWrite(m11,LOW);
  digitalWrite(m12,HIGH);
  digitalWrite(m21,LOW);
  digitalWrite(m22,HIGH);
}

void right()
{
  digitalWrite(m11,LOW);
  digitalWrite(m12,LOW);
  digitalWrite(m21,HIGH);
  digitalWrite(m22,HIGH);
}

void motorstop()
{
  digitalWrite(m11,HIGH);
  digitalWrite(m12,HIGH);
  digitalWrite(m21,HIGH);
  digitalWrite(m22,HIGH);

}
```

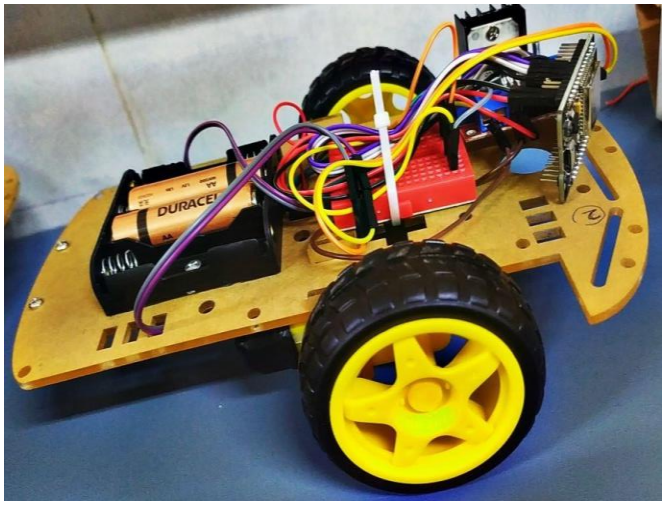
Python Code:

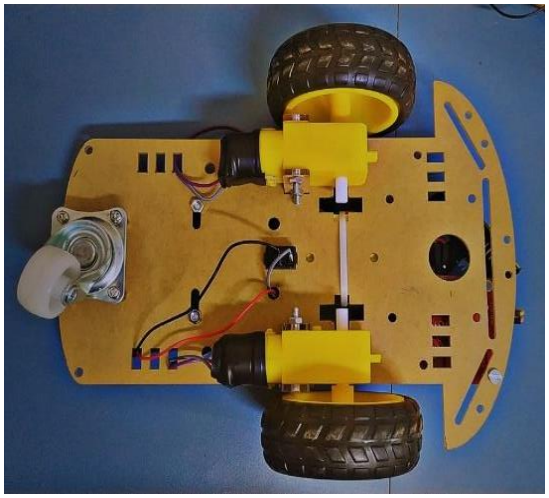
```
import RPi.GPIO as GPIO
importtime

# Define GPIO pins for motor control
m11 = 33
m12 = 25
m21 = 26
m22 = 27

# Setup GPIO mode
GPIO.setmode(GPIO.BCM)
GPIO.setup(m11, GPIO.OUT)
GPIO.setup(m12, GPIO.OUT)
GPIO.setup(m21, GPIO.OUT)
GPIO.setup(m22, GPIO.OUT)
```

```
def forward():
        GPIO.output(m11, GPIO.LOW)
        GPIO.output(m12, GPIO.HIGH)
        GPIO.output(m21, GPIO.LOW)
        GPIO.output(m22, GPIO.HIGH)


def right():
        GPIO.output(m11, GPIO.LOW)
        GPIO.output(m12, GPIO.LOW)
        GPIO.output(m21, GPIO.HIGH)
        GPIO.output(m22, GPIO.HIGH)


def motorstop():
        GPIO.output(m11, GPIO.HIGH)
        GPIO.output(m12, GPIO.HIGH)
        GPIO.output(m21, GPIO.HIGH)
        GPIO.output(m22, GPIO.HIGH)

try:
        while True:
                forward()
                time.sleep(3)
                right()
                time.sleep(0.5)

                forward()
                time.sleep(3)
                right()
                time.sleep(0.5)

                forward()
                time.sleep(3)
                right()
                time.sleep(0.5)

                forward()
                time.sleep(3)
                right()
                time.sleep(0.5)

                forward()
                time.sleep(3)
                right()
```

```
                time.sleep(0.5)


                forward()
                time.sleep(3)
                right()
                time.sleep(0.5)

                forward()
                time.sleep(3)
                motorstop()
                time.sleep(3)


    except KeyboardInterrupt:
            print("Program interrupted, cleaning up GPIO...")
            GPIO.cleanup()
```

OUTPUT





Experiment No 6 : Create Obstacle Avoidance Behavior and test it

Aim: To Create Obstacle Avoidance Behavior and test it. (esp32)

Hardware Requirement:

- ESP32 : A versatile microcontroller with built-in Wi-Fi and Bluetooth.
- IR Sensor : Measures distance to objects.
- Gearmotor Sensor : Controls the motors of the robot.
- Motors and Wheels : To move the robot.
- Power Supply : Battery to power the components.
- Breadboard : More connections for VCC and GND.
- Motor Driver (L298N) : Controls the motors of the robot.

Dekstop :To operate system.

Software Requirement:

- Windows
- Arduino IDE 1.8.19

Theory

Obstacle avoidance is a fundamental behavior for autonomous robots. It enables them to navigate environments without colliding with objects. This behavior is particularly useful for robots performing tasks in dynamic or unknown areas.

Circuit Diagram:

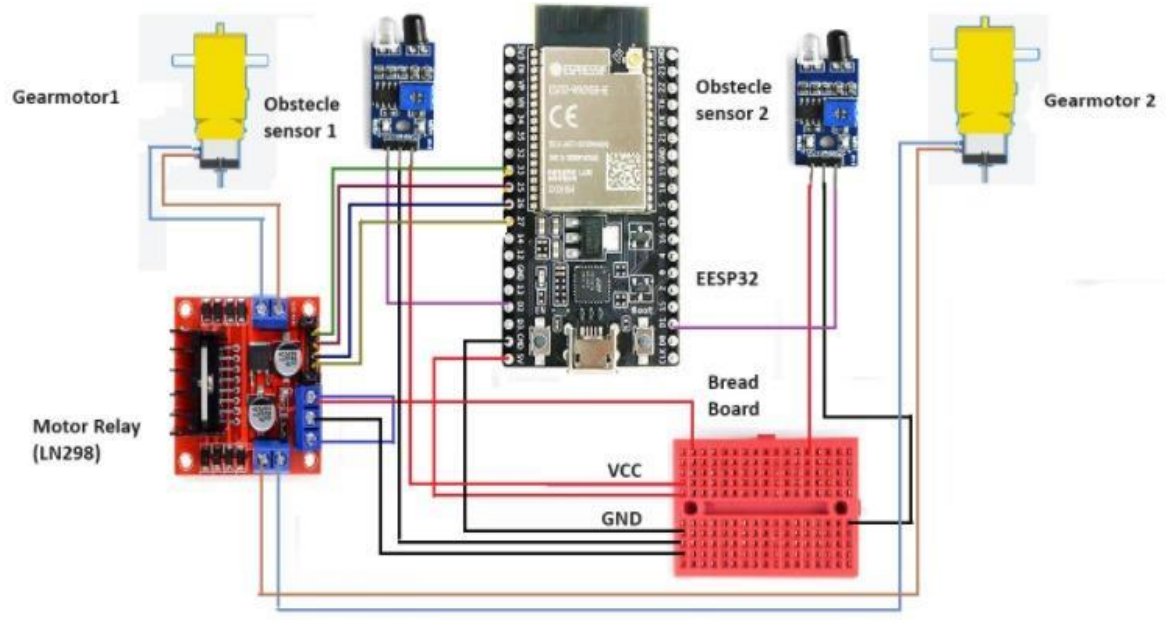


Fig:Obstacle Avoidance Behavior with ESP32

Steps:

Set Up the Hardware :

- Connect the IR Sensor to the ESP32 digital port .
- Connect the Motor Driver to the ESP32 and motors:

- Motor 1: M11 to GPIO port, M12 to GPIO port
- Motor 2: M21 to GPIO port, M22 to GPIO port
- Connect the power supply to the motors and the ESP32.

C- Code

```
int m1Pin11 = 13;
int m1Pin12 = 12;
int m2Pin21 = 14;
int m2Pin22 = 27;
int ir1 = 19;
int ir2 = 15;
int obstaclesense1;
int obstaclesense2;

void setup() {
  pinMode(m1Pin11, OUTPUT);
  pinMode(m1Pin12, OUTPUT);
  pinMode(m2Pin21, OUTPUT);
  pinMode(m2Pin22, OUTPUT);
  pinMode(ir1, INPUT);
  pinMode(ir2, INPUT);
}
void loop()
{
 obstaclesense1 = digitalRead(ir1);
 obstaclesense2 = digitalRead(ir2);
 //LEFT WHEEL
 left();
 //right wheel
 right();
 //straight
 straight();
 //backward
 backward();
}
void straight()
{
  //straight
  if (obstaclesense1 == 0 && obstaclesense2 == 0){
     digitalWrite(m1Pin11,LOW);
     digitalWrite(m1Pin12,HIGH);
     digitalWrite(m2Pin21,LOW);
     digitalWrite(m2Pin22,HIGH);
     delay(250);
  }
```

```
}
  void right()
  {
    if (obstaclesense1 ==0 && obstaclesense2 == 1) {
     digitalWrite(m1Pin11,HIGH);
     digitalWrite(m1Pin12,LOW);
     digitalWrite(m2Pin21,LOW);
     digitalWrite(m2Pin22,HIGH);
     delay(250);
    }
  }
  void left()
  {
    if (obstaclesense1 ==1 && obstaclesense2 == 0) {
     digitalWrite(m1Pin11,LOW);
     digitalWrite(m1Pin12,HIGH);
     digitalWrite(m2Pin21,HIGH);
     digitalWrite(m2Pin22,LOW);
     delay(250);
    }
  }
void backward()
{
  //backward
  if (obstaclesense1 == 1 && obstaclesense2 == 1){
     digitalWrite(m1Pin11,HIGH);
     digitalWrite(m1Pin12,LOW);
     digitalWrite(m2Pin21,HIGH);
     digitalWrite(m2Pin22,LOW);
     delay(250);
  }
}
```

Python Code:

```
import RPi.GPIO as GPIO
import time

# Define GPIO pins
m1Pin11 = 13
m1Pin12 = 12
m2Pin21 = 14
m2Pin22 = 27
ir1 = 19
ir2 = 15

# Setup GPIO mode and pins
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(m1Pin11, GPIO.OUT)
GPIO.setup(m1Pin12, GPIO.OUT)
GPIO.setup(m2Pin21, GPIO.OUT)
GPIO.setup(m2Pin22, GPIO.OUT)
GPIO.setup(ir1, GPIO.IN)
GPIO.setup(ir2, GPIO.IN)

try:
    while True:
        # Read sensor values
        obstaclesense1 = GPIO.input(ir1)
        obstaclesense2 = GPIO.input(ir2)

        # Conditionally control the motors
        if obstaclesense1 == 0 and obstaclesense2 == 0:  # Both sensors clear
            GPIO.output(m1Pin11, GPIO.LOW)
            GPIO.output(m1Pin12, GPIO.HIGH)
            GPIO.output(m2Pin21, GPIO.LOW)
            GPIO.output(m2Pin22, GPIO.HIGH)
            time.sleep(0.25)

        elif obstaclesense1 == 0 and obstaclesense2 == 1:  # Only second
sensor detects obstacle
            GPIO.output(m1Pin11, GPIO.HIGH)
            GPIO.output(m1Pin12, GPIO.LOW)
            GPIO.output(m2Pin21, GPIO.LOW)
            GPIO.output(m2Pin22, GPIO.HIGH)
            time.sleep(0.25)

        elif obstaclesense1 == 1 and obstaclesense2 == 0:  # Only first sensor
detects obstacle
            GPIO.output(m1Pin11, GPIO.LOW)
            GPIO.output(m1Pin12, GPIO.HIGH)
            GPIO.output(m2Pin21, GPIO.HIGH)
            GPIO.output(m2Pin22, GPIO.LOW)
            time.sleep(0.25)

        elif obstaclesense1 == 1 and obstaclesense2 == 1:  # Both sensors
detect obstacle
            GPIO.output(m1Pin11, GPIO.HIGH)
            GPIO.output(m1Pin12, GPIO.LOW)
            GPIO.output(m2Pin21, GPIO.HIGH)
            GPIO.output(m2Pin22, GPIO.LOW)
            time.sleep(0.25)

except KeyboardInterrupt:
    print("Program stopped by user")
finally:
```

```
GPIO.cleanup()
```

Experiment No - 7. Title: Control robot using Bluetooth, voice and Gesture control

Aim: Write a program in Arduino IDE to control a robot using Bluetooth, voice and gesture control in ESP32.

Hardware Requirements:

1. **ESP32 Development Board**:
    - Choose an ESP32 development board with built-in Bluetooth capability.
    - Ensure it has sufficient GPIO pins for motor control and other peripherals.
2. **Hobby Gear Motors**:
    - Select DC hobby gear motors suitable for your robot's size and movement requirements.
    - Ensure they provide adequate torque and speed for your application.
3. **L298N Motor Driver Module**:
    - Use the L298N dual H-bridge motor driver module to control the DC motors.
    - Ensure it can handle the voltage and current requirements of your motors.
4. **Wheels and Chassis**:
    - Choose wheels and a chassis appropriate for your robot design.
    - Ensure the chassis can accommodate the motors, motor driver, and other components.
5. **Power Bank**:
    - Select a power bank with the following specifications:
        - **Output voltage:** Typically 5V through USB, suitable for powering the ESP32 and other 5V components.
        - **Capacity (mAh):** Sufficient to power the robot for extended periods based on current draw calculations.
        - **Output current:** Adequate to supply power to the motors and ESP32 simultaneously without voltage drops or overheating..
6. **Breadboard, Jumpers, and Connectors**:
    - Essential for prototyping and connecting various components together.

Software Requirements:

1. **Arduino IDE**:
    - Install the Arduino IDE on your computer.
    - Ensure it supports ESP32 development by installing the necessary board definitions and libraries.
2. **BluetoothSerial Library**:
    - Include the BluetoothSerial library for ESP32 in your Arduino sketch.
    - This library facilitates Bluetooth communication between the ESP32 and the Sritu app.
3. **Sritu App (or Similar)**:
    - Install the Sritu app on your mobile device (available on Google Play Store for Android).

- Verify it supports Bluetooth communication and offers voice, gesture, and remote control functionalities.

Integration Steps:

1. **Hardware Setup**:
   - Connect hobby gear motors to the outputs of the L298N motor driver module.
   - Power the ESP32 board using the power bank's USB output.
   - Optionally, use a USB to DC jack adapter for powering the L298N motor driver and motors from the power bank.
2. **Software Setup**:
   - Develop or modify your Arduino sketch to:
     - Initialize BluetoothSerial for communication with the Sritu app.
     - Implement logic to interpret commands ('F', 'B', 'L', 'R', 'S') received via Bluetooth for voice, gesture, and remote control modes.
     - Control motor actions (forward, backward, left, right, stop) based on the received commands.
3. **Sritu App Integration**:
   - Pair the Sritu app with the ESP32 Bluetooth module.
   - Use the app's interface for voice commands, gesture-based controls, and manual directional controls.
   - Ensure the app sends Bluetooth commands that the ESP32 interprets correctly to drive the motors accordingly.

Theory:

Components Explanation:

1. **ESP32 Microcontroller:**
   - **Functionality**: The ESP32 is a powerful microcontroller with built-in Wi-Fi and Bluetooth capabilities.
   - **Role in the Project**: It serves as the brain of the robot, controlling motor movements based on commands received via Bluetooth from the Sritu app.
   - **Features**:
     - Multiple GPIO pins for interfacing with sensors, motor drivers, and other peripherals.
     - Built-in Wi-Fi and Bluetooth for wireless communication.
     - ADC (Analog-to-Digital Converter) for reading sensor inputs.
     - PWM (Pulse Width Modulation) outputs for controlling motor speed.

2. **Hobby Gear Motors:**
   - **Functionality**: DC motors with gear reduction for higher torque at lower speeds.
   - **Role in the Project**: Provides locomotion to the robot, enabling movement in different directions.
   - **Features**:

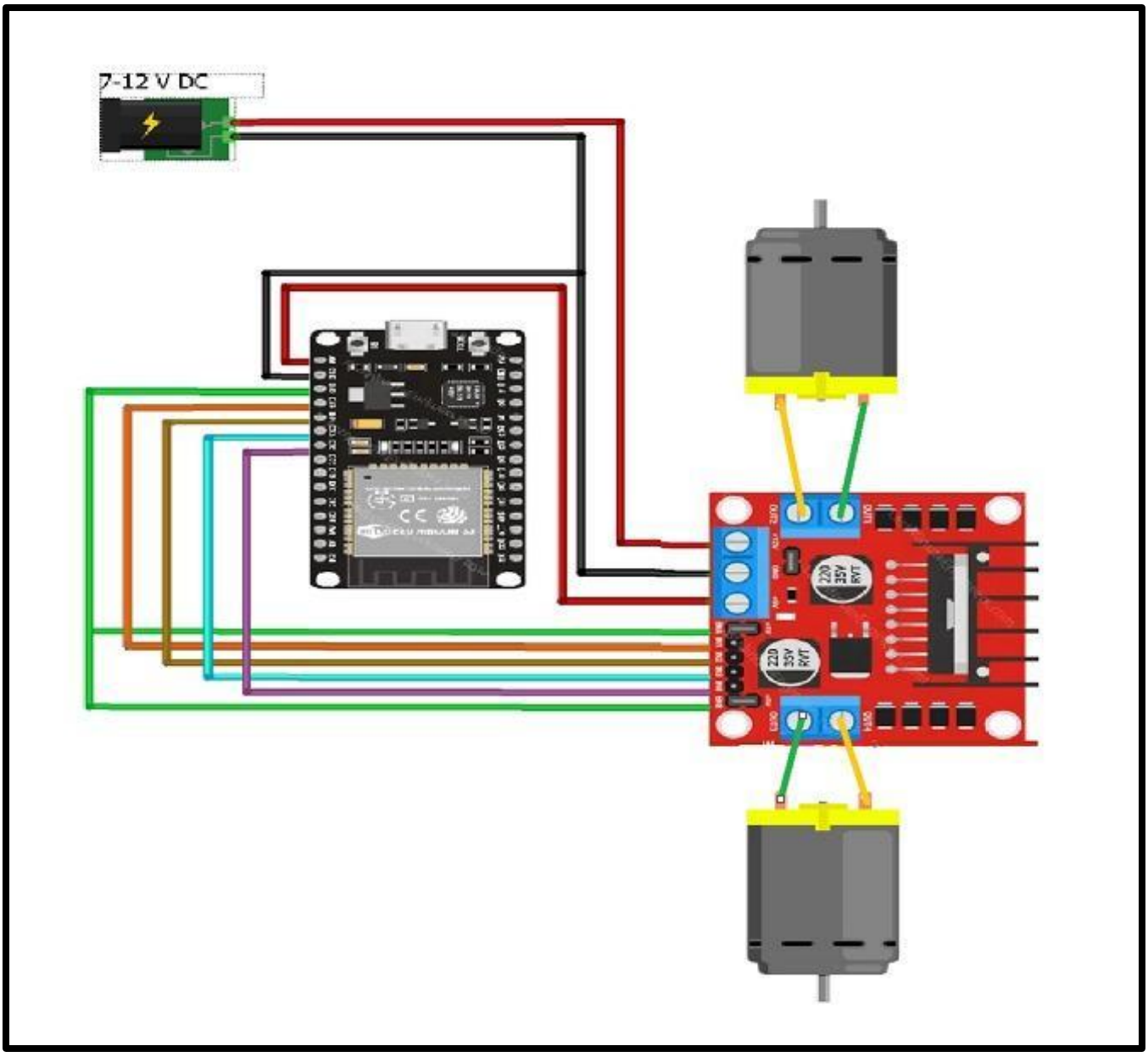
- Typically low RPM (revolutions per minute) with high torque, suitable for robot applications.
- Operate on DC voltage supplied by the motor driver.

3. **L298N Motor Driver Module:**
   - **Functionality**: Dual H-bridge motor driver IC for controlling DC motors.
   - **Role in the Project**: Interfaces between the ESP32 and the hobby gear motors, allowing bidirectional control (forward, backward) and speed regulation.
   - **Features**:
     - Supports up to two DC motors.
     - Provides current up to 2A per motor channel (with proper heat sinking).
     - Logic inputs (IN1, IN2, IN3, IN4) control motor direction and speed through PWM signals from the ESP32.

4. **Power Bank:**
   - **Functionality**: Portable battery pack providing DC power.
   - **Role in the Project**: Supplies power to the ESP32, L298N motor driver, and hobby gear motors.
   - **Features**:
     - Outputs typically 5V through USB ports.
     - Capacity measured in mAh (milliampere-hours) determines runtime.
     - Provides sufficient current to drive both microcontroller and motors simultaneously.

5. **Sritu App (or Similar):**
   - **Functionality**: Mobile application for Android with Bluetooth communication capabilities.
   - **Role in the Project**: Provides a user-friendly interface for remote control of the robot.
   - **Features**:
     - **Voice Control**: Allows users to command the robot verbally using voice recognition.
     - **Gesture Control**: Utilizes sensors (e.g., accelerometer) on the mobile device to interpret gestures for controlling the robot.
     - **Remote Control**: Offers manual directional controls (forward, backward, left, right, stop) through a graphical interface.

Circuit Diagram:


7-12 V DC

C- Code

```
#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) ||
!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run make menuconfig to and enable it
#endif
int m11 = 13;
int m12 =12;
int m21 =14;
int m22 = 27;
BluetoothSerial SerialBT;

void forward(){
        digitalWrite(m11,LOW);
        digitalWrite(m12,HIGH);
        digitalWrite(m21,LOW);
        digitalWrite(m22,HIGH);
        //delay(3000);
}

void stops(){
        //stop
        digitalWrite(m11,LOW);
        digitalWrite(m12,LOW);
        digitalWrite(m21,LOW);
        digitalWrite(m22,LOW);
        // delay(1300);
}

void left(){
        //LEFT
        digitalWrite(m11,LOW);
        digitalWrite(m12,LOW);
        digitalWrite(m21,LOW);
        digitalWrite(m22,HIGH);
        delay(200);
}

void right(){
        //rIGHT
        digitalWrite(m11,LOW);
        digitalWrite(m12,HIGH);
        digitalWrite(m21,LOW);
        digitalWrite(m22,LOW);
        delay(200);
}
void reverse(){
```

```
        digitalWrite(m11,HIGH);
        digitalWrite(m12,LOW);
        digitalWrite(m21,HIGH);
        digitalWrite(m22,LOW);
        delay(300);
}
void setup() {
        pinMode(m11,OUTPUT);
        pinMode(m12,OUTPUT);
        pinMode(m21, OUTPUT);
        pinMode(m22, OUTPUT);
        Serial.begin(115200);
        SerialBT.begin("ESP32test"); //Bluetooth device name
        Serial.println("The device started, now you can pair it with bluetooth!");
        // pinMode(LED_BUILTIN, OUTPUT);
}
char keyvalue;
void loop() {
if (SerialBT.available()) {
        keyvalue = SerialBT.read();
                        // SerialBT.write(keyvalue.toString);
        Serial.println(keyvalue);
if (keyvalue=='F' || keyvalue=='f' || keyvalue=='U')
{
        Serial.println("Foward Movement.....");
        digitalWrite(m11,LOW);
        digitalWrite(m12,HIGH);
        digitalWrite(m21,LOW);
        digitalWrite(m22,HIGH);
}
if(keyvalue=='L' || keyvalue=='l')
{
        Serial.println("Left Movement.....");
        left();
}
if(keyvalue=='R' || keyvalue=='r')
{
        Serial.println("right Movement.....");
        right();
}
if(keyvalue=='B' || keyvalue=='b' || keyvalue=='D')
{
        Serial.println("Backward Movement.....");
        reverse();
}
if(keyvalue=='S' || keyvalue=='s')
{
        Serial.println("Stop Movement.....");
```

```
        stops();
}
}
//delay(20);
}
```

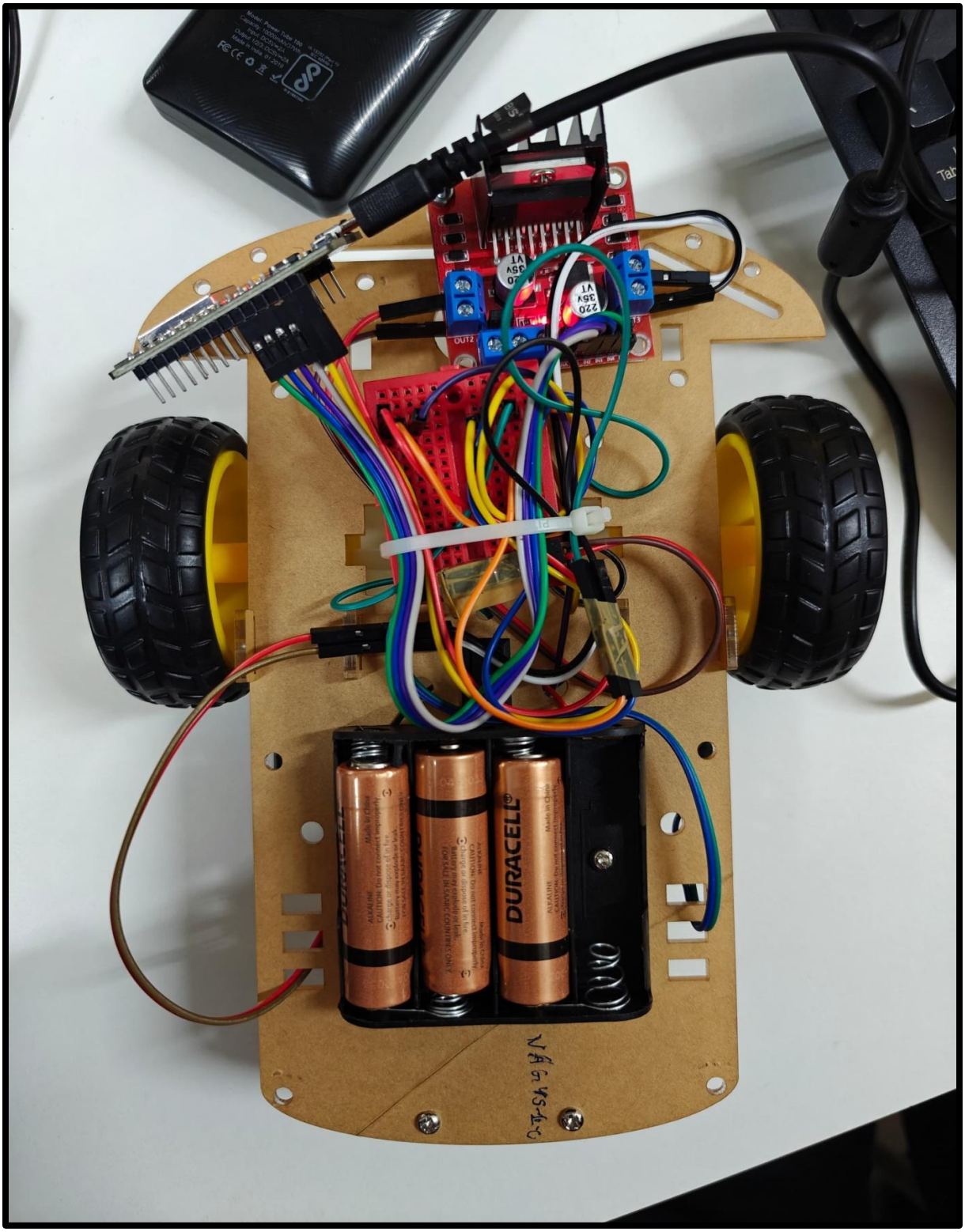
Python Code:

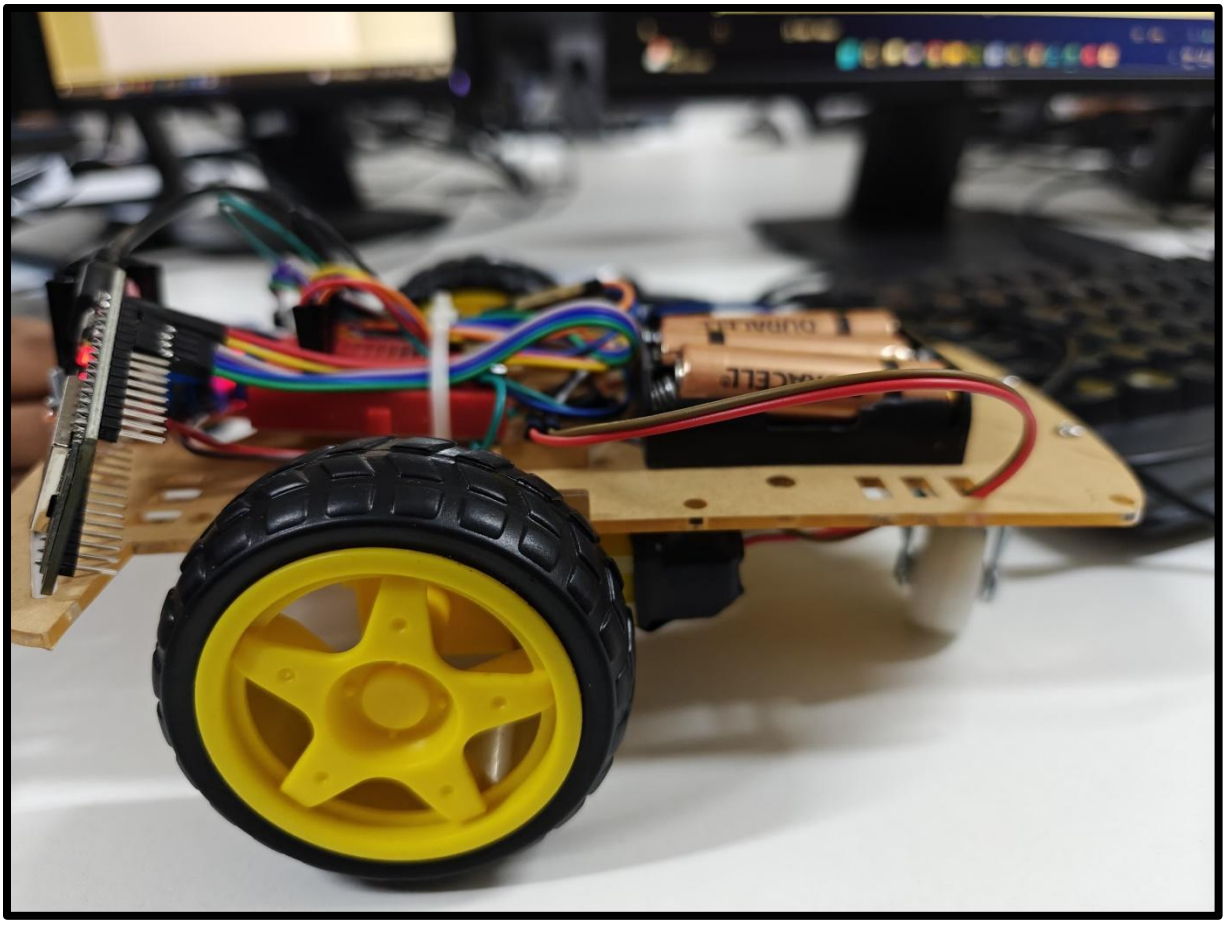
```
        import bluetooth
        import RPi.GPIO as GPIO
        import time


        # Define GPIO pins for motor control
        m11 = 13
        m12 = 12
        m21 = 14
        m22 = 27


        # Setup GPIO mode
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(m11, GPIO.OUT)
        GPIO.setup(m12, GPIO.OUT)
        GPIO.setup(m21, GPIO.OUT)
        GPIO.setup(m22, GPIO.OUT)

        # Bluetooth setup
        server_socket = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
        port = 1
        server_socket.bind(("", port))
        server_socket.listen(1)

        print("Waiting for Bluetooth connection on RFCOMM channel", port)

        client_socket, address = server_socket.accept()
        print("Accepted connection from", address)

        def forward():
                GPIO.output(m11, GPIO.LOW)
                GPIO.output(m12, GPIO.HIGH)
                GPIO.output(m21, GPIO.LOW)
                GPIO.output(m22, GPIO.HIGH)

        def stops():
                GPIO.output(m11, GPIO.LOW)
                GPIO.output(m12, GPIO.LOW)
                GPIO.output(m21, GPIO.LOW)
```

```
        GPIO.output(m22, GPIO.LOW)

def left():
        GPIO.output(m11, GPIO.LOW)
        GPIO.output(m12, GPIO.LOW)
        GPIO.output(m21, GPIO.LOW)
        GPIO.output(m22, GPIO.HIGH)
        time.sleep(0.2)
        stops()


def right():
        GPIO.output(m11, GPIO.LOW)
        GPIO.output(m12, GPIO.HIGH)
        GPIO.output(m21, GPIO.LOW)
        GPIO.output(m22, GPIO.LOW)
        time.sleep(0.2)
        stops()

def reverse():
        GPIO.output(m11, GPIO.HIGH)
        GPIO.output(m12, GPIO.LOW)
        GPIO.output(m21, GPIO.HIGH)
        GPIO.output(m22, GPIO.LOW)
        time.sleep(0.3)
        stops()


try:
        while True:
                data = client_socket.recv(1024).decode().strip()
                print("Received:", data)

                if data == 'F' or data == 'f' or data == 'U':
                        print("Forward Movement...")
                        forward()
                elif data == 'L' or data == 'l':
                        print("Left Movement...")
                        left()
                elif data == 'R' or data == 'r':
                        print("Right Movement...")
                        right()
                elif data == 'B' or data == 'b' or data == 'D':
                        print("Backward Movement...")
                        reverse()
                elif data == 'S' or data == 's':
                        print("Stop Movement...")
```

```
                    stops()

except KeyboardInterrupt:
        print("Keyboard interrupt detected, closing connections...")
        client_socket.close()
        server_socket.close()
        GPIO.cleanup()
```

Output:





Experiment No - 8 : Robot object and develop Line Following Behavior

Aim : Add the sensor to the Robot object and develop line following behavior code.

Hardware Requirement:

1) ESP32
2) Connecting wires
3) IR Sensor
4) Breadboard
5) Motor Driver
6) Hobby Gearmotor

Software Requirement :

1) Arduino IDE
2) Windows System

Theory:

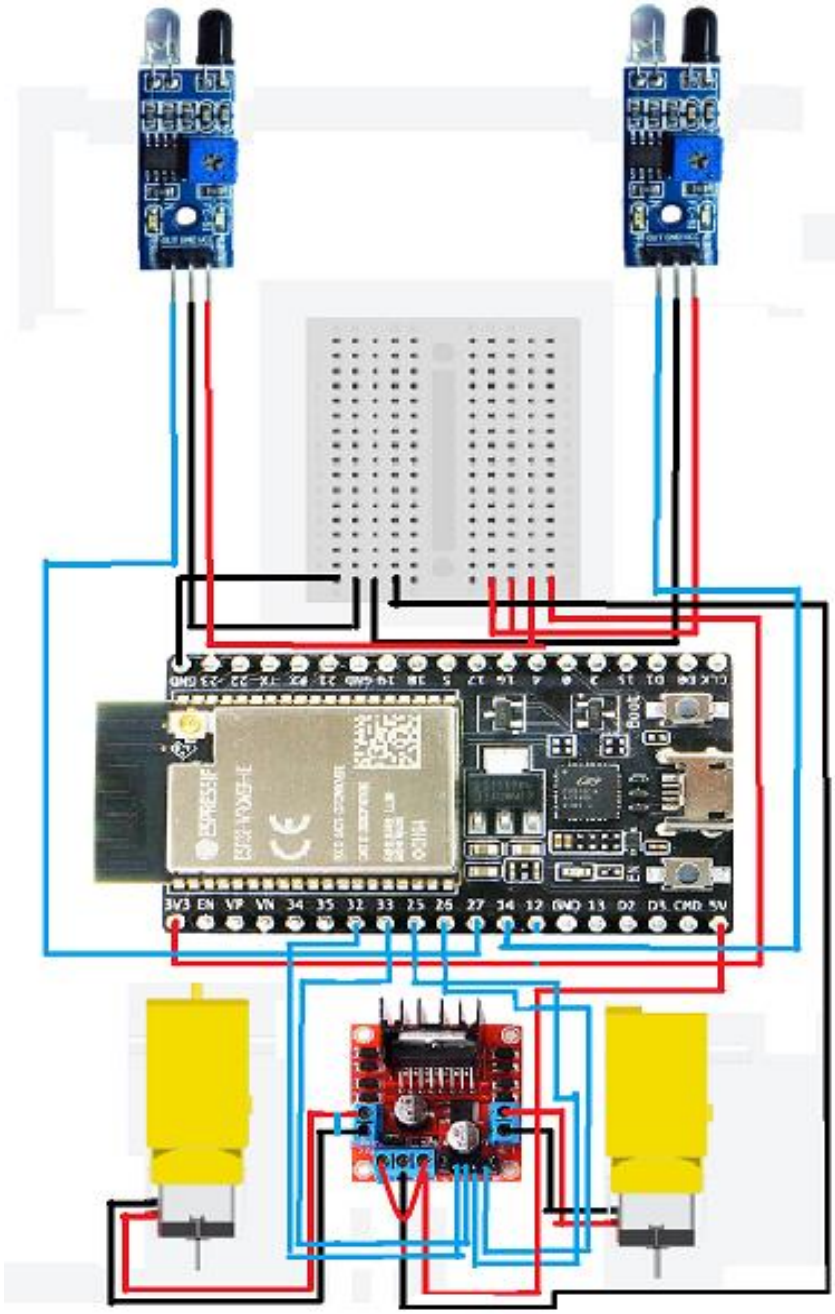
1) **ESP32 -** The ESP32, equipped with its dual-core processor and integrated Wi-Fi and Bluetooth capabilities, serves as the central controller for the robot. By interfacing an IR sensor (like the TCRT5000) with the ESP32's GPIOs and ADC, the robot can detect and follow lines based on reflected infrared light. Programming this behavior involves reading sensor data, applying logic to interpret line presence or absence, and controlling motor movements using PWM signals. This integration showcases the ESP32's versatility in real-time sensor data processing and motor control, essential for autonomous navigation tasks like line following in robotics.

2) **IR Sensor** - An IR (Infrared) sensor, such as the TCRT5000, consists of an infrared LED and a phototransistor. It operates on the principle of reflection: the LED emits infrared light, which bounces off a surface (like a line on the ground) and is detected by the phototransistor. The sensor outputs an analog voltage proportional to the amount of reflected infrared light, which varies depending on whether the sensor is over a light or dark surface (line or background).

3) **Gearmotor** - Hobby gearmotors are compact electromechanical devices that integrate a DC motor with a gearbox. They are designed for low-power applications where torque multiplication and precise control over rotational speed are essential.

4) **Motor Drive**r : A motor driver is an electronic circuit or module that interfaces between a microcontroller or other control signal source and a DC motor. Its primary function is to regulate the speed and direction of the motor by adjusting the voltage and current supplied to it.

Circuit diagram:



C - Code

```
// Define motor pins
int motorPin1 = 32;
 int motorPin2 = 33;
 int motorPin3 = 25;
int motorPin4 = 26;

// Define IR sensor pins
int irSensorPinLeft = 37;
int irSensorPinRight = 14;

void setup() {
  Serial.begin(9600);

  // Motor pins setup
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);

  // IR sensor pins setup
  pinMode(irSensorPinLeft, INPUT);
  pinMode(irSensorPinRight, INPUT);
}

void loop() {

  int sensorValueLeft = digitalRead(irSensorPinLeft);
  int sensorValueRight = digitalRead(irSensorPinRight);


  Serial.print("Left Sensor: ");
  Serial.print(sensorValueLeft);
  Serial.print(" | Right Sensor: ");
  Serial.println(sensorValueRight);

  // Determine line-following behavior based on sensor readings
  if (sensorValueLeft == LOW && sensorValueRight == LOW) {
    // Neither sensor detects the line, stop or take corrective action

    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, LOW);

        } else if (sensorValueLeft == LOW && sensorValueRight == HIGH) {

    // Only right sensor detects the line, adjust left to turn left
    digitalWrite(motorPin1, HIGH);
```

```
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin4, HIGH);

 } else if (sensorValueLeft == HIGH && sensorValueRight == LOW) {

  // Only left sensor detects the line, adjust right to turn right
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, HIGH);
  digitalWrite(motorPin3, HIGH);
  digitalWrite(motorPin4, LOW);

 } else if (sensorValueLeft == HIGH && sensorValueRight == HIGH) {

  // Both sensors detect the line, move forward
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, HIGH);
  digitalWrite(motorPin4, LOW);
 }
 delay(100);  }
```

Python Code:

```
import RPi.GPIO as GPIO
import time

# Define GPIO pins
motorPin1 = 17  # Example GPIO pin numbers; adjust as per your wiring
motorPin2 = 18
motorPin3 = 27
motorPin4 = 22
irSensorPinLeft = 37
irSensorPinRight = 14

# Setup GPIO mode and pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(motorPin1, GPIO.OUT)
GPIO.setup(motorPin2, GPIO.OUT)
GPIO.setup(motorPin3, GPIO.OUT)
GPIO.setup(motorPin4, GPIO.OUT)
GPIO.setup(irSensorPinLeft, GPIO.IN)
GPIO.setup(irSensorPinRight, GPIO.IN)

try:
    while True:
        # Read sensor values
```

```
            sensorValueLeft = GPIO.input(irSensorPinLeft)
            sensorValueRight = GPIO.input(irSensorPinRight)

            # Print sensor readings (optional)
            print(f"Left Sensor: {sensorValueLeft} | Right Sensor: {sensorValueRight}")

            # Determine line-following behavior based on sensor readings
            if sensorValueLeft == GPIO.LOW and sensorValueRight == GPIO.LOW:
                # Neither sensor detects the line, stop or take corrective action
                GPIO.output(motorPin1, GPIO.LOW)
                GPIO.output(motorPin2, GPIO.LOW)
                GPIO.output(motorPin3, GPIO.LOW)
                GPIO.output(motorPin4, GPIO.LOW)

            elif sensorValueLeft == GPIO.LOW and sensorValueRight == GPIO.HIGH:
                # Only right sensor detects the line, adjust left to turn left
                GPIO.output(motorPin1, GPIO.HIGH)
                GPIO.output(motorPin2, GPIO.LOW)
                GPIO.output(motorPin3, GPIO.LOW)
                GPIO.output(motorPin4, GPIO.HIGH)

            elif sensorValueLeft == GPIO.HIGH and sensorValueRight == GPIO.LOW:
                # Only left sensor detects the line, adjust right to turn right
                GPIO.output(motorPin1, GPIO.LOW)
                GPIO.output(motorPin2, GPIO.HIGH)
                GPIO.output(motorPin3, GPIO.HIGH)
                GPIO.output(motorPin4, GPIO.LOW)

            elif sensorValueLeft == GPIO.HIGH and sensorValueRight == GPIO.HIGH:
                # Both sensors detect the line, move forward
                GPIO.output(motorPin1, GPIO.HIGH)
                GPIO.output(motorPin2, GPIO.LOW)
                GPIO.output(motorPin3, GPIO.HIGH)
                GPIO.output(motorPin4, GPIO.LOW)

            time.sleep(0.1)  # Adjust delay as needed

    except KeyboardInterrupt:
        print("Program stopped by user")
    finally:
        GPIO.cleanup()
```

Experiment No – 9 : To Study Pan and Tilt Mechanism

Title: Integrating Pan and Tilt Services into Robot Object using ESP32

Aim: To add pan and tilt functionalities to an existing robot object using an ESP32 microcontroller and verify their functionality through controlled movements.

Hardware Requirements:

- Robot object with existing motor controls
- Pan servo motor
- Tilt servo motor
- ESP32 development board (e.g., ESP32 DevKitC)
- Breadboard and jumper wires
- Power supply (battery pack or adapter)

Software Requirements:

- Arduino IDE or PlatformIO with ESP32 support
- ESP32 Servo library

Theory

Servo Motor

Servo motor (or servomechanism): This type of motor combines a gear motor with a sensor and a built-in controller as shown in Figure 2.5. A signal to a controller states a motor position, and the controller uses feedback from the sensor to try to reach this position. Servo motors are used in pan and tilt mechanisms, along with robot arms and limbs

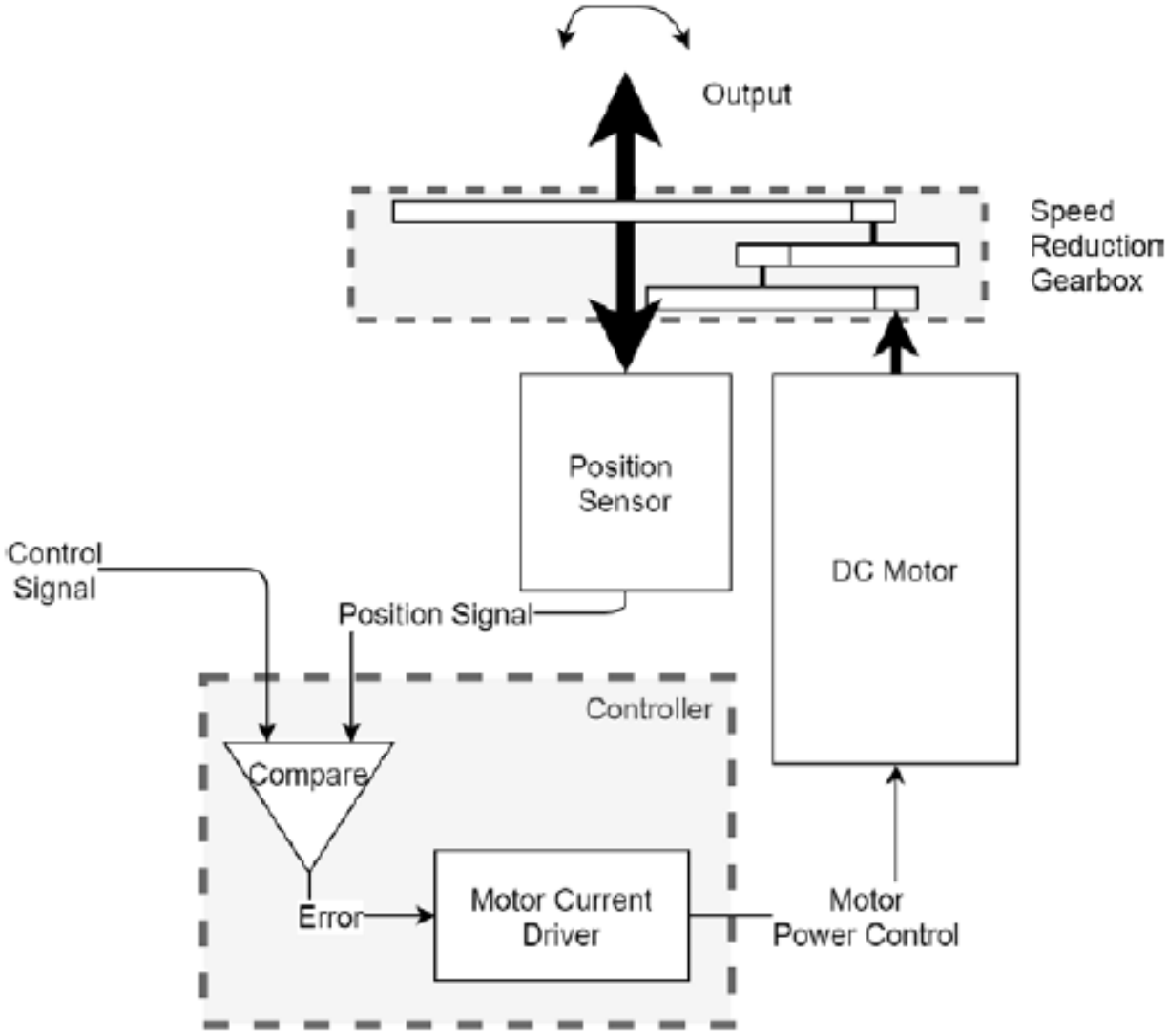


Fig. Block Diagram of Servo Motor

Pan and Tilt mechanism,

shown in *Fig*, moves a sensor (or anything else) through two axes under servo motor control.

Pan means to turn left or right. *Tilt* means to tilt up or down:

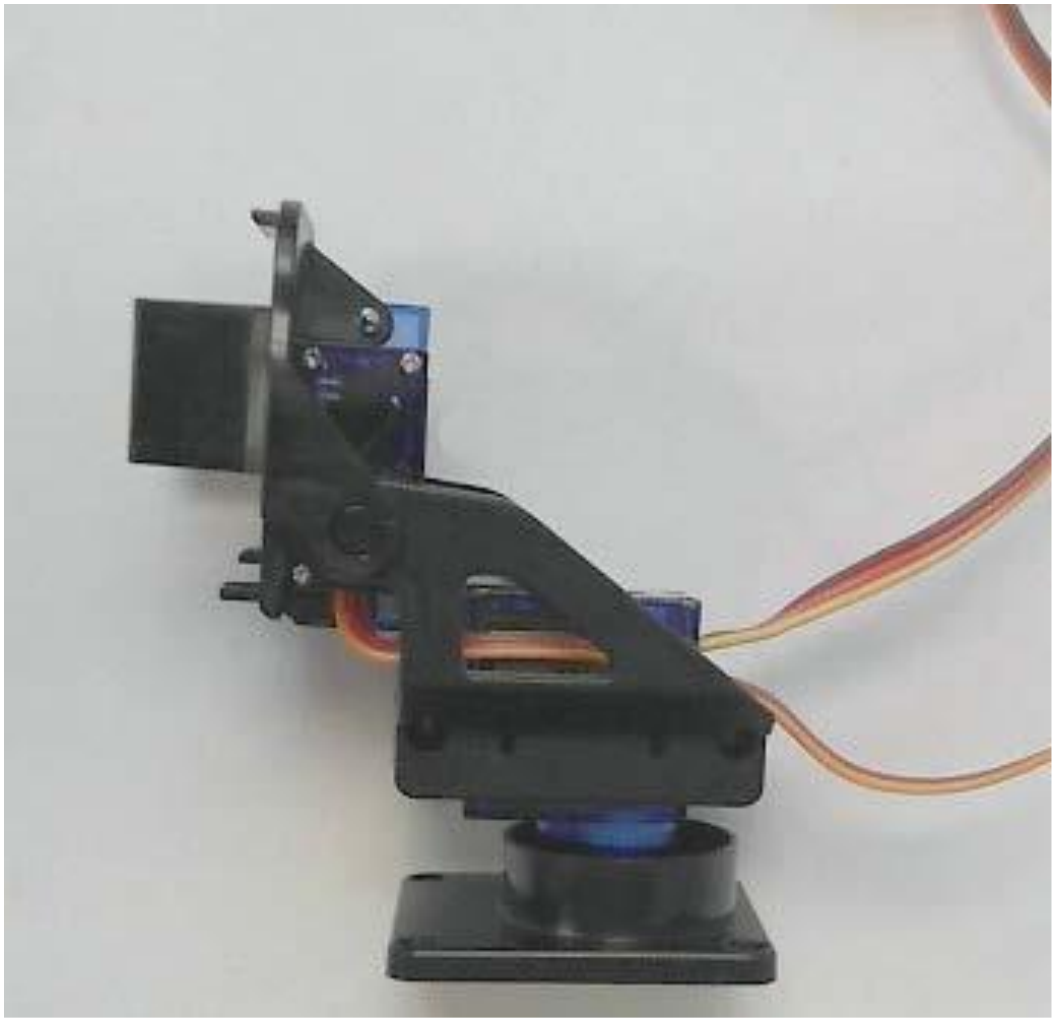


Fig. Typical Pan and Tilt Mechanism

Our robot block diagram with the servos looks like *Fig*:

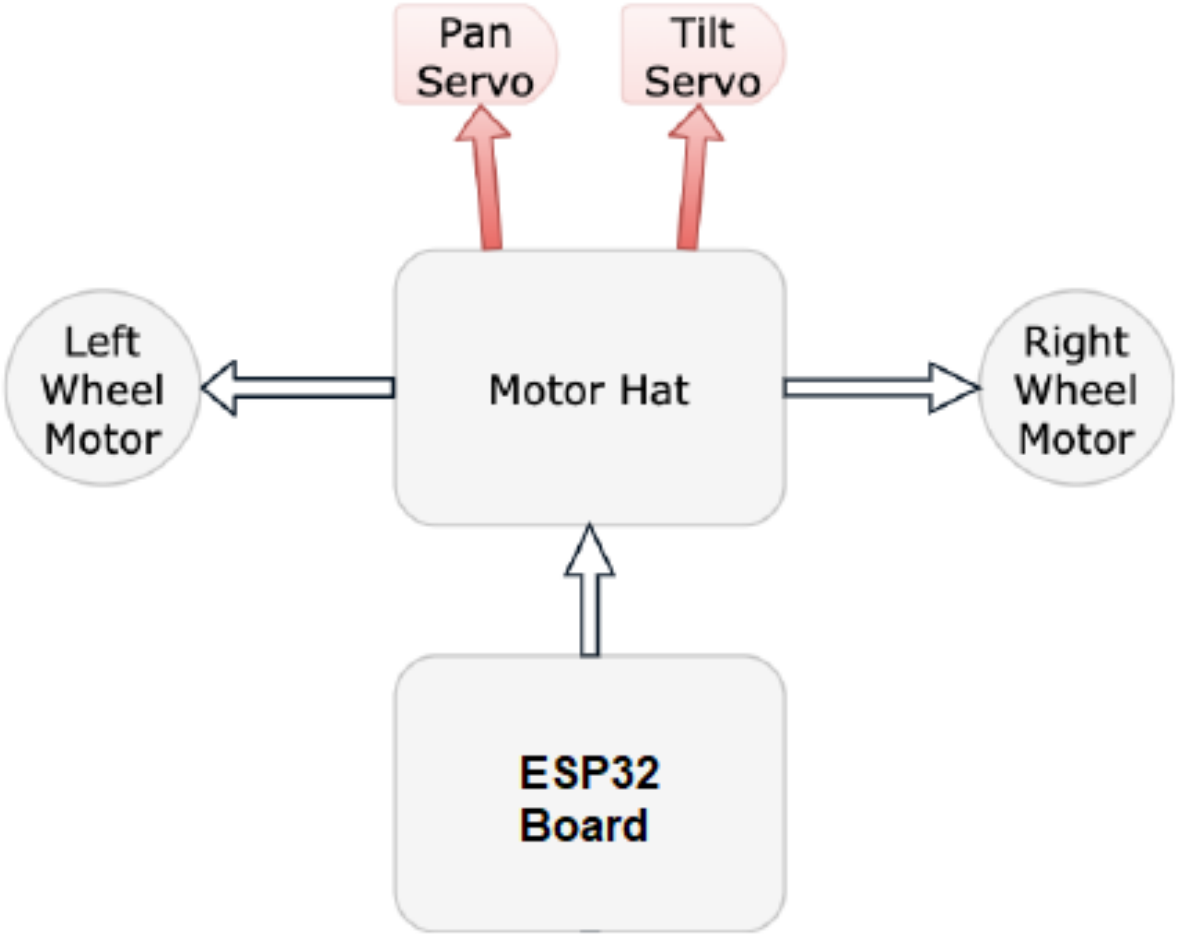
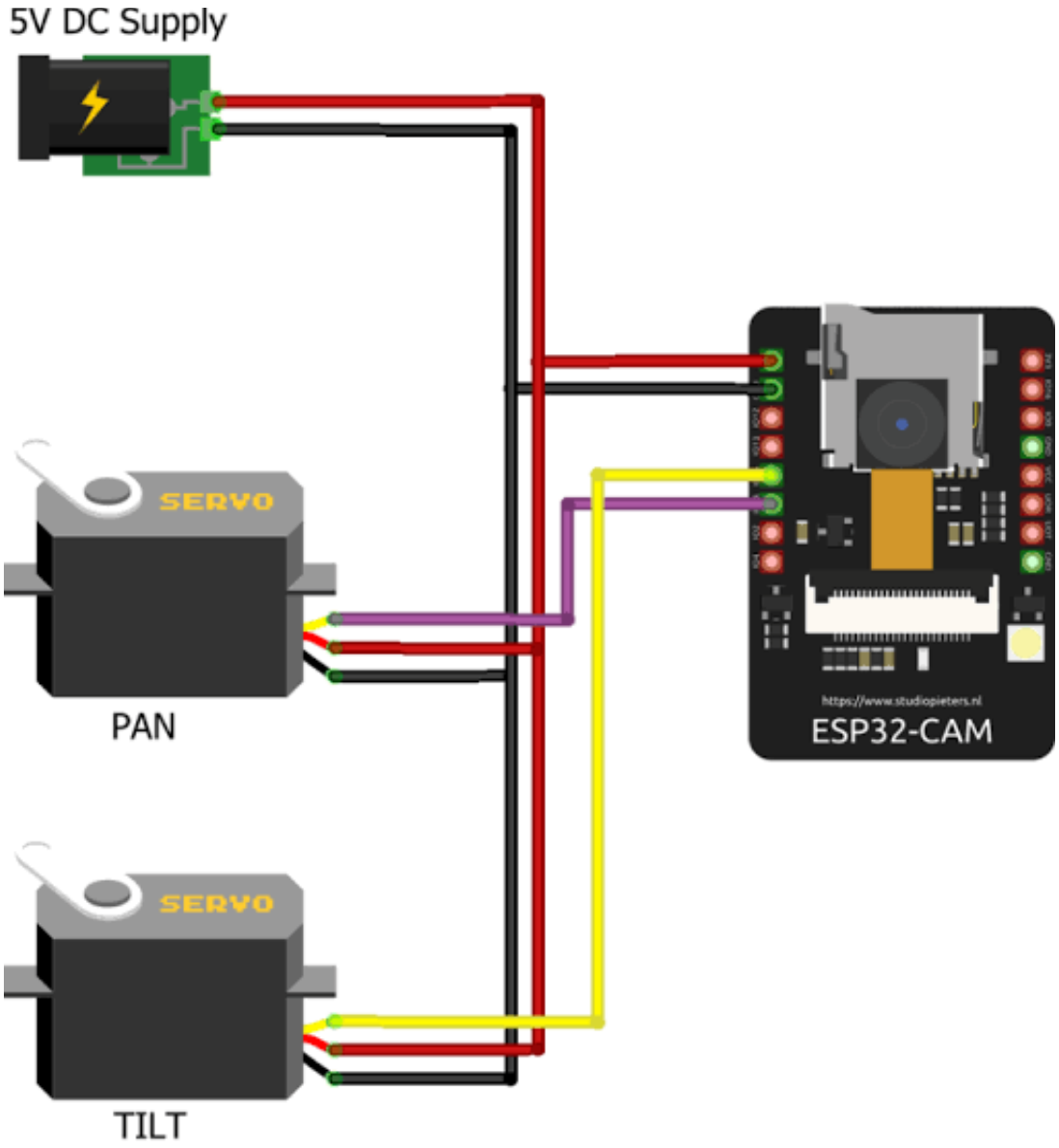


Fig. Block Diagram of Pan and Tilt Mechanism

Circuit Diagram:



Circuit Diagram for Pan and Tilt mechanism with ESP32

C- Code:

```
#include <Servo.h>

                                        // Define servo objects

Servo panServo;

Servo tiltServo;
```

```
                                        // Define servo pin numbers
int panPin = 21; // Example GPIO pin for pan servo
int tiltPin = 22; // Example GPIO pin for tilt servo

void setup() {
                                    // Attach servo objects to respective pins
  panServo.attach(panPin);
  tiltServo.attach(tiltPin);
}

void loop() {
                                // Example usage: move pan servo to 90 degrees
  panServo.write(90);
  delay(1000); // Wait for 1 second

                                // Example usage: move tilt servo to 45 degrees
  tiltServo.write(45);
  delay(1000); // Wait for 1 second

  // Add more movements or control logic as needed
   // Loop will repeat continuously
}
```

Explanation:

1. **Servo Library:** Includes the Servo library which simplifies the code for controlling servo motors.
2. **Servo Objects:** panServo and tiltServo are initialized as servo objects.

3. **Setup Function:** In the setup() function, the servo objects are attached to their respective GPIO pins (panPin and tiltPin). Adjust the GPIO pins according to your ESP32 board's pinout.
4. **Loop Function:** The loop() function contains example code to move the pan and tilt servos to specific angles (90 degrees for pan and 45 degrees for tilt). Adjust the angles and delays as per your specific requirements.

Experiment No 10 : Edge Following

TITLE: Edge-Following Robot with ESP32

AIM: To create an edge-following robot using IR sensors with an ESP32 for maintaining a consistent distance from an edge.

HARDWARE REQUIREMENTS

- ESP32 Development Board
- 2x IR Sensors
- 2x DC Motors
- Motor Driver (if applicable)
- Robot Chassis
- Connecting wires
- Breadboard (if needed)

SOFTWARE REQUIREMENTS

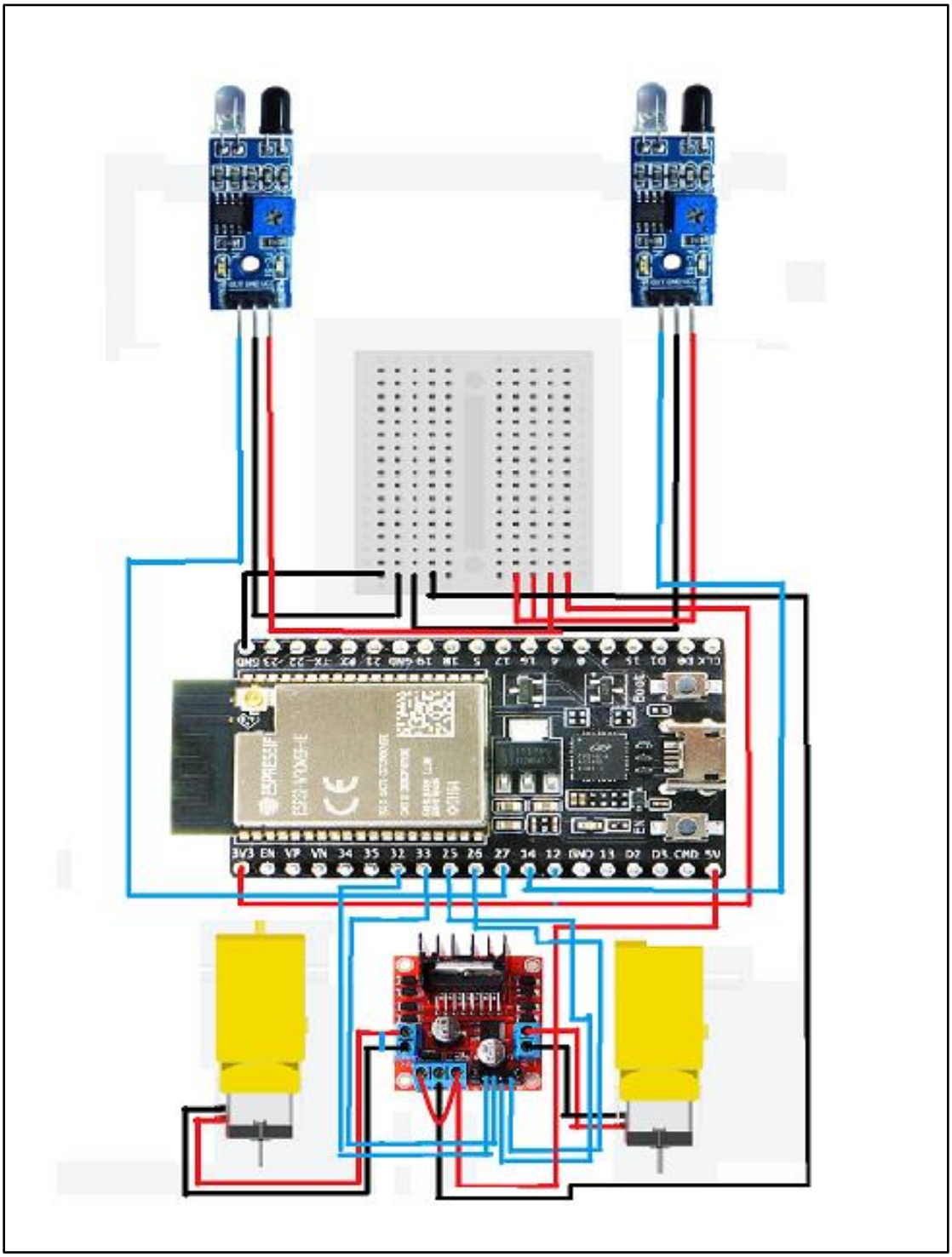
- Arduino IDE with ESP32 board support

THEORY

1. **ESP32 Development Board:** A versatile microcontroller with built-in Wi-Fi and Bluetooth capabilities, ideal for IoT projects due to its dual-core processor and extensive peripheral support.
2. **IR Sensors:** These detect infrared radiation to sense proximity or edges in robotics, comprising an emitter and a receiver for reflective sensing.
3. **DC Motors:** Convert electrical energy into mechanical motion for robot propulsion, controlled via voltage polarity or PWM signals.
4. **Motor Driver:** Interfaces between the microcontroller and DC motors, managing power delivery and direction control.
5. **Robot Chassis:** Provides structural support and configuration for mounting motors, sensors, and microcontrollers, crucial for defining the robot's physical design and mobility capabilities.

6. **Connecting Wires and Breadboard:** Essential for temporary connections and prototyping, enabling flexible circuit assembly and testing without permanent soldering.

CIRCUIT DIAGRAM



C- CODE

```
                // Define motor control pins
int m11 = 13;
int m12 = 12;
int m21 = 14;
int m22 = 27;

                // Define IR sensor pins
int ir_1 = 26;
int ir_2 = 25;

                // Variables to store sensor readings
int obsense1;
int obsense2;

void setup() {
                // Set motor control pins as outputs
  pinMode(m11, OUTPUT);
  pinMode(m12, OUTPUT);
  pinMode(m21, OUTPUT);
  pinMode(m22, OUTPUT);

                  // Set IR sensor pins as inputs
  pinMode(ir_1, INPUT);
  pinMode(ir_2, INPUT);
}

void loop() {
                        // Read sensor values
  obsense1 = digitalRead(ir_1);
  obsense2 = digitalRead(ir_2);

                        // Decision making based on sensor readings
  if (obsense1 == HIGH && obsense2 == LOW) {
    // Turn right to adjust position
    digitalWrite(m11, HIGH);
    digitalWrite(m12, LOW);
    digitalWrite(m21, LOW);
    digitalWrite(m22, HIGH);
```

```
  }
else if (obsense1 == LOW && obsense2 == HIGH) {
                // Turn left to adjust position
   digitalWrite(m11, LOW);
   digitalWrite(m12, HIGH);
   digitalWrite(m21, HIGH);
   digitalWrite(m22, LOW);
  }
else if (obsense1 == LOW && obsense2 == LOW) {
               // Move forward if both sensors are off the edge
   digitalWrite(m11, HIGH);
   digitalWrite(m12, LOW);
   digitalWrite(m21, HIGH);
   digitalWrite(m22, LOW);
  }
else {
               // Stop if both sensors detect the edge or are off
   digitalWrite(m11, LOW);
   digitalWrite(m12, LOW);
   digitalWrite(m21, LOW);
   digitalWrite(m22, LOW);
  }
}
```

- If only the first sensor (ir_1) detects the edge, the robot will turn right to adjust its position.
- If only the second sensor (ir_2) detects the edge, the robot will turn left to adjust its position.
- If both sensors detect the edge or both sensors are off the edge, the robot will stop.
- If neither sensor detects the edge, the robot will move forward.

PYTHON CODE

```
import RPi.GPIO as GPIO
import time

# Define motor control pins (GPIO numbering)
m11 = 17
m12 = 18
```

```
m21 = 27
m22 = 22

# Define IR sensor pins (GPIO numbering)
ir_1 = 26
ir_2 = 25

# Initialize GPIO settings
GPIO.setmode(GPIO.BCM)
GPIO.setup(m11, GPIO.OUT)
GPIO.setup(m12, GPIO.OUT)
GPIO.setup(m21, GPIO.OUT)
GPIO.setup(m22, GPIO.OUT)
GPIO.setup(ir_1, GPIO.IN)
GPIO.setup(ir_2, GPIO.IN)

def stop_motors():
    GPIO.output(m11, GPIO.LOW)
    GPIO.output(m12, GPIO.LOW)
    GPIO.output(m21, GPIO.LOW)
    GPIO.output(m22, GPIO.LOW)

try:
    while True:
        # Read sensor values
        obsense1 = GPIO.input(ir_1)
        obsense2 = GPIO.input(ir_2)

        # Decision making based on sensor readings
        if obsense1 == GPIO.HIGH and obsense2 == GPIO.LOW:
            # Turn right to adjust position
            GPIO.output(m11, GPIO.HIGH)
            GPIO.output(m12, GPIO.LOW)
            GPIO.output(m21, GPIO.LOW)
            GPIO.output(m22, GPIO.HIGH)
        elif obsense1 == GPIO.LOW and obsense2 == GPIO.HIGH:
            # Turn left to adjust position
            GPIO.output(m11, GPIO.LOW)
            GPIO.output(m12, GPIO.HIGH)
            GPIO.output(m21, GPIO.HIGH)
            GPIO.output(m22, GPIO.LOW)
```

```
        elif obsense1 == GPIO.LOW and obsense2 == GPIO.LOW:
            # Move forward if both sensors are off the edge
            GPIO.output(m11, GPIO.HIGH)
            GPIO.output(m12, GPIO.LOW)
            GPIO.output(m21, GPIO.HIGH)
            GPIO.output(m22, GPIO.LOW)
        else:
            # Stop if both sensors detect the edge or are off
            stop_motors()

        time.sleep(0.1)  # Adjust delay as needed for your application

except KeyboardInterrupt:
    print("Stopping the program...")
finally:
    GPIO.cleanup()
```