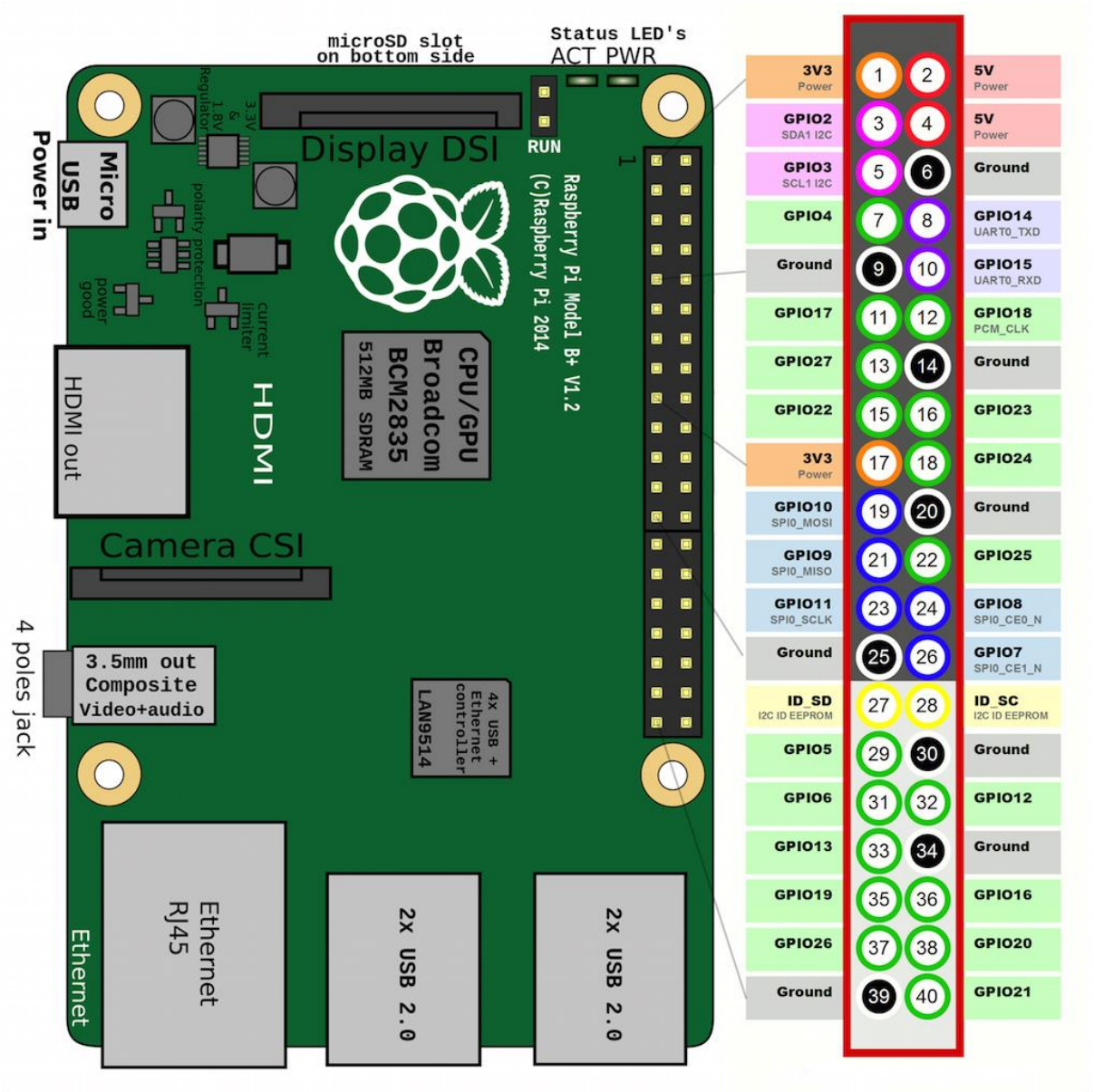
Practical 7

Study and Setup Raspberry Pi board.



Setting up your Raspberry Pi 3 board :

1. Prepare the Necessary Equipment

- **Raspberry Pi 3 board**
- **MicroSD card (at least 8GB, Class 10 or higher recommended)**
- **Micro USB power supply (5V, 2.5A)**
- **Monitor or TV** (with HDMI support)
- **HDMI cable**
- **USB keyboard and mouse**

2. Install the Operating System (OS)

- **Download Raspberry Pi OS**: Raspberry Pi OS (32-bit) is recommended for compatibility and performance. Download it from the official Raspberry Pi website.
- **Flash the OS onto the MicroSD card**: Use software like Raspberry Pi Imager or Balena Etcher. Select the OS, your MicroSD card, and write the image to it.

3. Boot the Raspberry Pi

- Insert the prepared MicroSD card into the Pi.
- Connect it to a display via HDMI, plug in the USB keyboard and mouse, and then power it up using the Micro USB power supply.

4. Initial Setup

- On the first boot, the Raspberry Pi will prompt you through a setup wizard where you'll choose language, time zone, and set up Wi-Fi (if needed).
- **Update the system**: Open the terminal and run `sudo apt update && sudo apt upgrade` to ensure the system is up-to-date.

5. Configure Additional Settings (Optional)

- **Enable SSH**: If you want to access your Raspberry Pi remotely, you can enable SSH under `Raspberry Pi Configuration` > `Interfaces`.
- **Install Packages**: Depending on your project, you may need additional packages like Python, GPIO libraries, etc.

Raspberry Pi GPIO Pin Diagram and Configuration

The Raspberry Pi 3 has a **40-pin GPIO** header. These pins are organized into various types:

1. **Power Pins**:
    - **5V (Pins 2, 4)**: Directly powered by the 5V USB supply.
    - **3.3V (Pins 1, 17)**: Supplies 3.3 volts, useful for powering sensors.
    - **GND (Ground, multiple pins)**: Pins 6, 9, 14, 20, 25, 30, 34, 39 are grounds for completing circuits.
2. **GPIO Pins**:
    - There are 26 GPIO pins on the Raspberry Pi 3, configured as inputs or outputs to control LEDs, motors, sensors, etc.
3. **I2C (Inter-Integrated Circuit)**:
    - **Pins 3 (SDA) and 5 (SCL)**: Used for communication with I2C-compatible devices like temperature sensors or LCD screens.
4. **SPI (Serial Peripheral Interface)**:
    - **Pins 19 (MOSI), 21 (MISO), 23 (SCLK), 24 (CE0), and 26 (CE1)**: Used for fast data communication with devices like ADCs, DACs, and other peripherals.
5. **UART (Universal Asynchronous Receiver/Transmitter)**:
    - **Pins 8 (TXD) and 10 (RXD)**: Used for serial communication, such as connecting to a GPS module or serial monitor.

Common Uses of GPIO Pins

GPIO pins make the Raspberry Pi extremely versatile. Here are a few applications:

1. **Sensors**: Measure environmental parameters with sensors like temperature (DHT11/DHT22), light (LDR), and humidity.
2. **Display Modules**: Control LCD, OLED, and e-paper displays for visual outputs.
3. **Motor Control**: Power and control DC motors, servos, and stepper motors in robotics projects.
4. **LEDs and Buttons**: Create simple circuits for input/output testing with LEDs and buttons.
5. **IoT Projects**: Connect with the internet and sensors to create home automation or monitoring systems.

Programming the Raspberry Pi GPIO

1. Python

- Python is the most popular language for Raspberry Pi, with libraries like **RPi.GPIO** and **gpiozero** simplifying GPIO pin control.
- **Example: Blinking LED with RPi.GPIO**

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
    GPIO.output(18, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(18, GPIO.LOW)
    time.sleep(1)
```

2. C/C++

- The **WiringPi** library for C/C++ enables control over GPIO, commonly used for performance-critical applications.
- **Example: Basic LED Control with WiringPi**

```
#include <wiringPi.h>

int main() {
   wiringPiSetup();
   pinMode(0, OUTPUT);

   while(1) {
      digitalWrite(0, HIGH);
      delay(1000);
      digitalWrite(0, LOW);
      delay(1000);
   }
}
```

3. Scratch and Block-based Languages

- Suitable for beginners, Scratch allows drag-and-drop coding to interact with GPIO pins, ideal for educational purposes.

4. JavaScript (Node.js)

- **onoff** and **pigpio** are libraries that allow JavaScript-based control of GPIO pins, making it easier to integrate with web-based applications.

Advanced GPIO Applications

1. **SPI and I2C Communication**: Communicate with complex devices like ADCs, OLED displays, or multi-sensor modules.
2. **PWM (Pulse Width Modulation)**: Control servo motors, LEDs, and other components that need variable power levels.
3. **Interrupts**: Enable GPIO pins to detect real-time changes for projects requiring responsiveness, like home security systems.