ADVANCE PYTHON PROGRAMMING

REENA SRI G
22MID0009

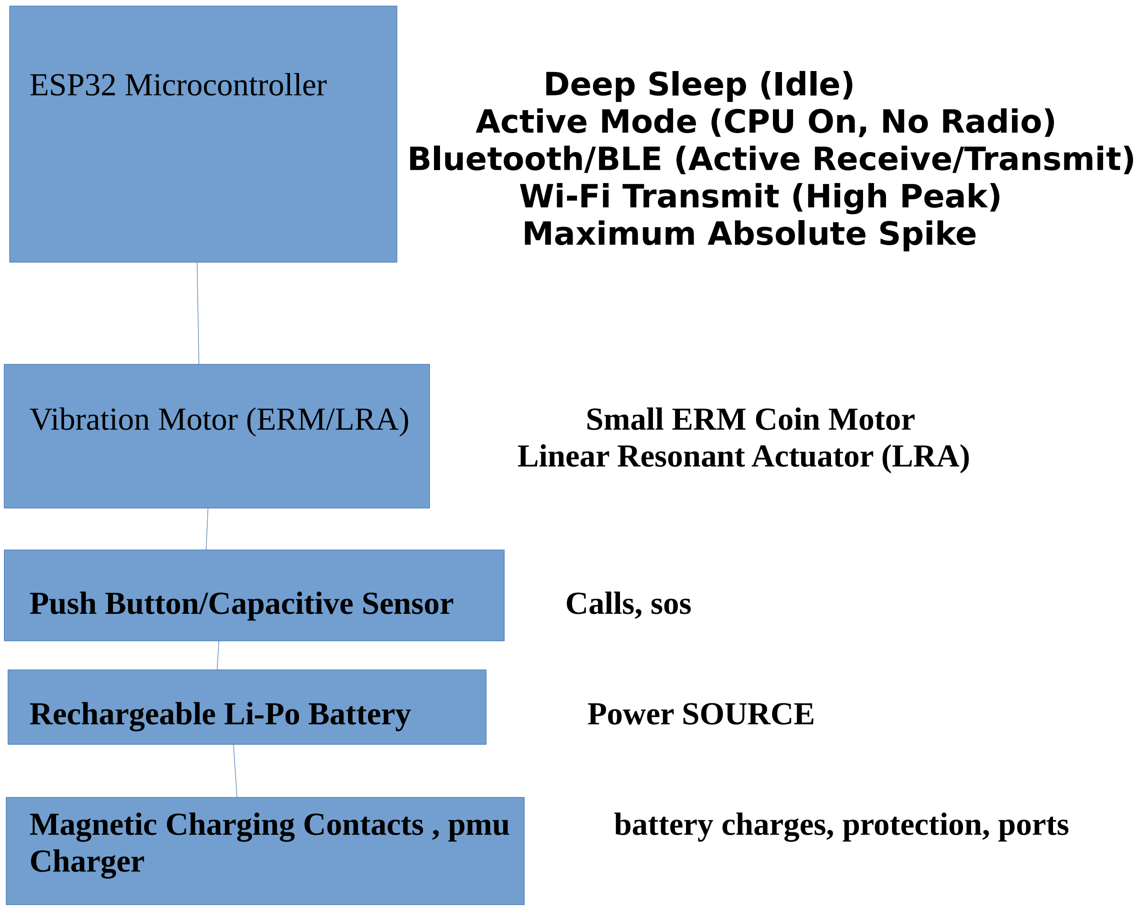
IOT -DEVICE APPLICATION
CLOTH BUTTONS

ENERGY SOURCE:

As the clothing button is less dimensionality it is important to consider the **energy density-to-weight ratio**, meaning a small, thin battery can hold a good charge. Lithium-Polymer (Li-Po) is considered as the suitable one as they are highly customisable, making them ideal for the **small, rounded button casing** you envision. These are generally rechargeable and provide the nominal **3.7V** required by many ESP32 boards.

For a small button, a capacity between **50 mAh and 200 mAh** (milliampere-hours) might be practical for size constraints, but the specific choice depends on the final enclosure volume.The single biggest power drain will be the **vibrating motor** when it's active, followed by the **ESP32 when Wi-Fi/Bluetooth is connected**. We must choose a battery (Lipo) capable of safely delivering the **peak current** required for the motor's vibration .By incorporating **magnetic charging contacts** (pogo pins) into the clip or its base would allow for a seamless, water-resistant way to recharge the battery without a USB port on the button itself. This supports your water-resistant design.

DEVICE SETUP :


ESP32 Microcontroller
Deep Sleep (Idle)
Active Mode (CPU On, No Radio)
Bluetooth/BLE (Active Receive/Transmit)
Wi-Fi Transmit (High Peak)
Maximum Absolute Spike
Vibration Motor (ERM/LRA)
Small ERM Coin Motor
Linear Resonant Actuator (LRA)
Push Button/Capacitive Sensor
Calls, sos
Rechargeable Li-Po Battery
Power SOURCE
Magnetic Charging Contacts , pmu Charger
battery charges, protection, ports

CONNECTION LAYER :

Here Bluetooth Low Energy (BLE) is ideal for this case. Because its focus on **wearability, low power consumption, and short-range communication** with a personal device (the user's smartphone). It has two main devices : Smart Button (ESP32) and User's Smartphone (App) for Peripheral and central GATT ways. This way the data is communicated through vibrations through notification and button commands writes the data , connectiong to phone. For eg , 1 to attend calls, 2 press to raise volumns.

DATA STORAGE LAYER :

The Data Storage Layer for smart button IoT project is essential for two main aspects: Local Device Storage (on the ESP32) and Remote Cloud Storage (for advanced features like monitoring and SOS alerts).

1. Local Device Storage (On-Button)
This storage is physically on the ESP32 chip or a tiny, connected memory module. Its primary purpose is to hold the device's operational settings and immediate state.

A. Non-Volatile Memory (NVS)

- **Location:** Integrated Flash memory on the ESP32.
- **Purpose:** Stores data that must persist even when the device is powered off or in Deep Sleep.
- **What to Store:**
    - **Configuration Settings:** User-defined **vibration patterns** (e.g., call = 3 long pulses, message = 2 short pulses), sensitivity levels, and volume settings for the mic/speaker (if used).
    - **BLE Pairing Keys:** Necessary to quickly re-establish a secure connection with the paired smartphone without repeating the full pairing process.
    - **Device ID/Credentials:** Unique identifier and temporary tokens needed to authenticate with the Cloud Server (if the cloud link is used).
    - **Current State:** The last known battery level before shutdown.

B. Random Access Memory (RAM)

- **Location:** Built-in to the ESP32 chip.
- **Purpose:** Stores data required for immediate operation while the device is awake and running. This data is **lost** when the device enters Deep Sleep or loses power.
- **What to Store:**
    - **Active Data Buffers:** Temporary storage for incoming BLE packets (notifications) and outgoing sensor/command data.
    - **Runtime Variables:** Current connection status, time since last successful sync, and counter for button presses (to detect double-click/long-press).

Apart from these , in Data storage Layer. As we connect to the mobile phones , we can actually store the data and their logs in the phone itself or some server connected to the phone.

APPLICATION LAYER :

Mobile App is much suitable in this case . AS it ideally focuses on connecting the button to the phone. The main objective is to answer phone calls, sense meaage in the mobile through vibrations in buttons. This is useful for deaf and blind people. As a result , This layer adds to the objective and goal of the project.

Smartphone Application (The Source) ----------> ESP32 Firmware (The Logic) ---------> Cloud Server/Database (Data Storage & Monitoring)

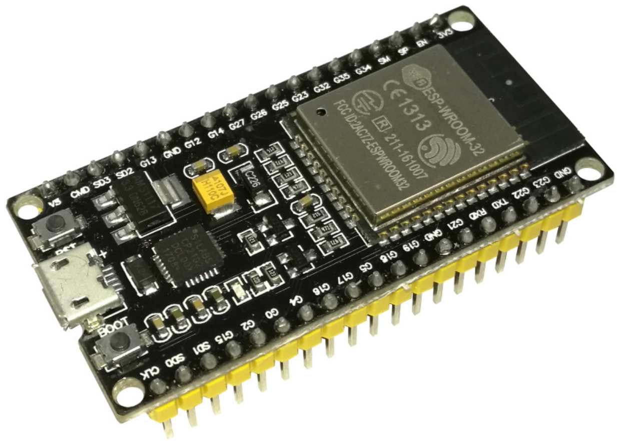
Data flow in Application Layer :

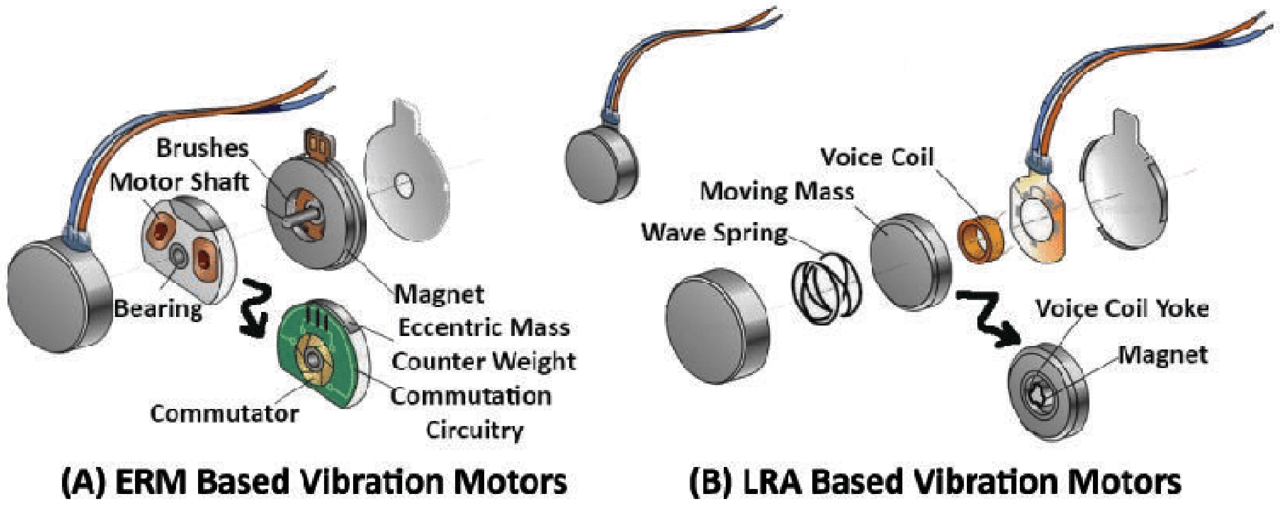
1.**Smartphone Application (The Source)**
- **Notification Interception**
- **Data Translation**
- **BLE Transmission**
- **Settings and Customization**

2. ESP32 Firmware (The Logic)
- **Command Listener**
- **Pattern Mapping (Call. SOS, Mesasge)**
- **Power/Sleep Management**
- **Input Handling**

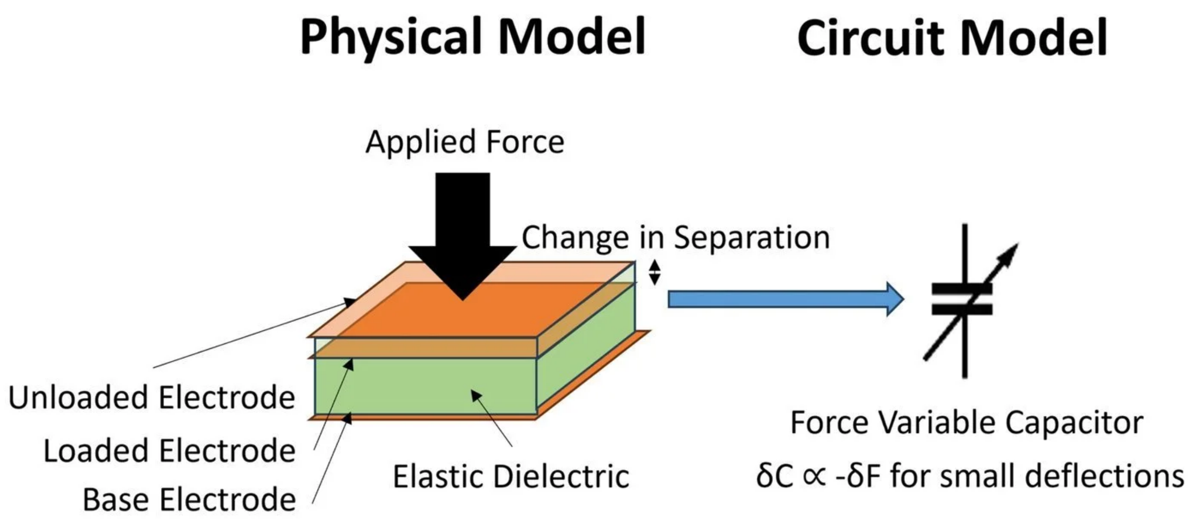
DEVICES :

ESP-32 BOARD





VIBRATION MOTORS



This diagram explains the working of the Push or capacitive sensors working. How the force is captured and activating the signals for transmissin.