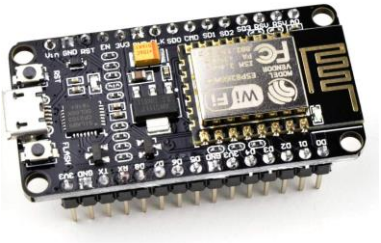
Objective: Create a physical device to get user reviews for the users (for example by clicking on a button). The device send the data to the server via wifi, and can be configurable using Bluetooth.

In this document, we will describe the different board we can use, their pros/cons, the different type of power supply, and an estimate of the battery discharge cycle.

Different boards

There are 3 boards suited for this project:

- The NodeMCU Dev Kit 1 and WeMos D1 mini (cost 2 or 3$):




  - Based on the ESP8266 processor (80Mhz CPU, 64KB/128KB SRAM)
  - Include Wifi that can be enabled/disabled to reduce power consumption.
  - The main difference is that the WeMos include Bluetooth, while some NodeMCU versions do not include Bluetooth.
  - The WeMos is smaller (34x25mm) compared to the NodeMCU (49x26mm)
  - Another difference is in the power consumption, which will be detailed in the Power section.
  - Input voltage: 4.5 to 10v if powered using a battery throught the microUSB port.
  - Or a 3.3v regulated current throught the VIN port.

- The ESP32 Dev Kit 1 (cost ~6$):

  - Based on the ESP32 processor, which is the successor of the ESP8266.
  - The ESP32 includes a Dual Core 160Mhz (up to 240Mhz) with 520KB SRAM.
  - Include Wifi and Bluetooth.
  - Size: 54x28mm.
  - Input voltage: 6-12v.
  - The main advantage of this board is the low consumption and the possibility to use several buttons to wake the board from Deep Sleep mode (6 to 8 buttons); while the NodeMCU can include only 1 button (more details will be given in the next sections).

Power consumption and battery lifetime

If the boards are always in active mode, the user will have to charge the battery everyday such a smartphone.

To save power, we can use those strategies:

- Limit the use of Wifi and Bluetooth by reducing the delay of synchronization with the server (eg: send data once per hour, or 30 minutes).
- Hack the hardware by removing some physical parts from the board consuming a lot of power even during sleep mode. The NodeMCU for example contains an USB-to-UART converter which can be removed, but this means we can't update the code of the board unless we put the UART on again.
- Put the board in Deep Sleep mode, and awake the board only when the user click on the button.
  In the normal consumption for these boards is about 85mA (going up to 200mA when it send/receive data from wifi). Which mean a phone battery of 3500mAh will be drained in one day.
  When we put the NodeMCU in Deep Sleep mode, the board uses a power of 18mA which mean the battery will stay up 1 week.
  The WeMos board has a lower power consumption compared to NodeMCU, it uses around 6mA in the sleep mode. The battery may still up for two weeks.
  The ESP32 has the lower power consumption during Deep Sleep mode (around 10 microA theoretically, which is very low). The battery may still up for few months.

Important: Please note that these numbers are estimates based on calculations, but the real power consumption in production may vary according to several factors such as the current fluctuations, the battery voltage, the number of buttons and leds attached to the board, and the latency of the network.

User Input

During Deep Sleep mode the processor is off, it cannot monitor the user input. The NodeMCU have a pin that can be used to awake the board, when it receives an electric signal from this pin it will put the processor on and quit the Deep Sleep mode.

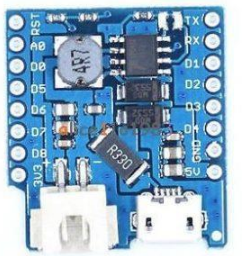
The idea is to link a push button the this pin, when the user click on the button (physical button in the box), the NodeMCU will quit the sleep mode, process the information, and send data to the server. This solution is efficient, but it has a limitation to only know if a user clicked a button or not.

In order to get a 5-star review from the client there are two potential solutions:

1. The client turn a potentiometer to specify the value of his ratting from 1 to 5 stars (basically a potentiometer is the volume button on the radios). Then he clicks a button to validate his choice. This button will awake the NodeMCU which will read the ratting value from the potentiometer and send it to the server.
2. Another alternative is to use the ESP32 board which has several wake pins (6 to 8), which means we can put a button for each ratting value. The user will then have the choice to click on of the 5 buttons to choose from 1 to 5 stars.
3. A third option can be used to use 5 buttons on the NodeMCU, and build an electronic circuit to merge the 5 buttons to 1 input and link it to the wake button. But this require soldering several electronic components such as transistors and diodes.

Different types of battery

- 2xAA standard batteries (e.g: piles rechargeables)
  Need a AA battery holder (~2$), and probably a current regulator.
  The regulator will make the current stable, but it will drain some power.
- LiPo 3.7v battery
  Such as the ones we use in our phones.
  Usually LiPo batteries for IoT boards come with a JST connector, and we use a LiPo battery shield (0.8~1.5$) which allows to use the battery and recharge it simultaneously from USB. In include a Green and Red Led to show when the charging is complete.
  Different batteries can be found with different capacities (from 850mAh to 3500mAh)



- Shield for 18650 battery
  Deliver 5v power via USB port. It can also deliver 3v cables.
  Allows to use and charge the battery using USB.
  Battery protection (overheating and overcharging)
  This shield costs around 2$.

Playing “thank you” message

The best choice is to use ISD1820 Recorder/Playback Module, which cost around 1$. This module contains a built-in microphone to record an audio message up to 20sec, and playing it back from the small button inside, or from the code.

It has an internal audio amplifier that can drive a 8-Ohm 0.5W speakers (cost: 0.5~1$), which is largely sufficient for playing “thank you” message, or a small melody/music.
It uses a power supply of 2.4V to 5.5V DC, and has an automatic power-down mode, which is adapted to our solution. Note that it can record only one audio file.
Dimensions: 80x60x30mm.
There are other alternatives playing audio files from an SD-card, but we have to add the cost of the SD-card, and even the smallest SD-card would be an overkill for just playing a small “thank you” message.

Other components needed

Need several Leds to show the board is running, or alerting when the battery if low.

Push buttons for user input, or potentiometer.

Probably need to add a hard reset button, or the possibility to remove the battery if the box is stuck.