Contents

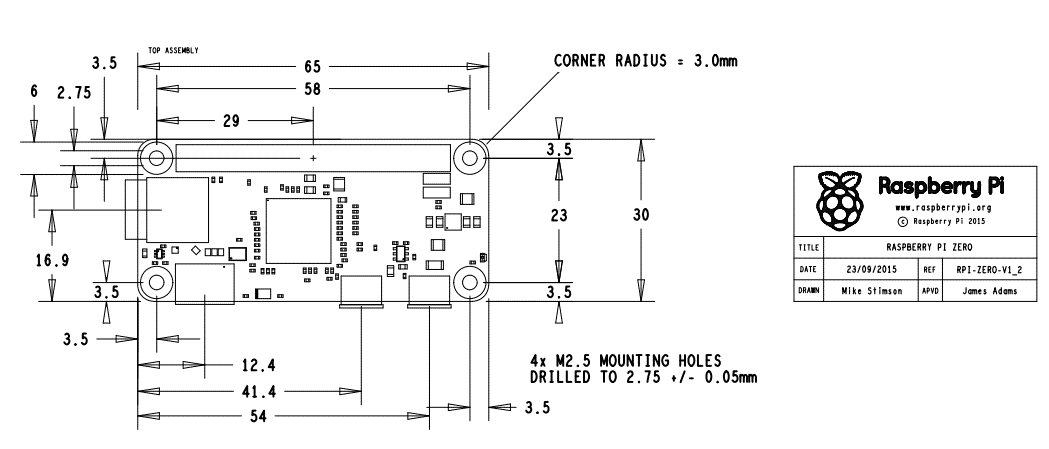
[1. Raspberry Pi layout 2](#_Toc119863246)

[2. ESP8266 for door sensor 3](#_Toc119863247)

[*2.1.* *Hardware Design* 3](#_Toc119863248)

[2.2. Firmware design 3](#_Toc119863249)

# Raspberry Pi layout



|  |  |  |  |
| --- | --- | --- | --- |
| **Item Name** | **User Visibility** | **Power source** |  |
| Power supply | No | 24vac |  |
| RPI zero | No | 5vdc |  |
| Camera | Yes | Internal |  |
| Audio | No | Internal |  |
| Switch | Yes | GND |  |
| Motion sensor | Yes | 3.3Vdc , GND |  |
| LED | Yes | 3.3vdc/GND |  |

Length: 33(motion sensor) + 25 (camera Module) + 16 (Button) =

Height : Audio board + pi zero == 28mm

Inner 86x41

External 90\*45

Motion sensor dia= 23.25 mm holding rest pillar height 7mm from top

Camera lenses dia= 14.02 , holding rest pillar height 9mm from top

Button dia= 13.25mm

# ESP8266 for door sensor

# Hardware design

# ESP8266 pin ratings

* The ESP8266 is a 3.3V microcontroller, so its I/O operates at 3.3V as well. The pins are **not 5V tolerant, applying more than 3.6V on any pin will kill the chip.**
* The maximum current that can be drawn from a single GPIO pin is **12mA**
* **PWM**

To enable PWM on a certain pin, use analogWrite(pin, value); where pin is the digital pin, and value a number between 0 and 1023.

You can change the range (bit depth) of the PWM output by using analogWriteRange(new\_range);

The frequency can be changed by using analogWriteFreq(new\_frequency);. new\_frequency should be between 100 and 1000Hz.

* ADC

Just like on an Arduino, you can use analogRead(A0) to get the analog voltage on the analog input. (0 = 0V, 1023 = 1.0V).

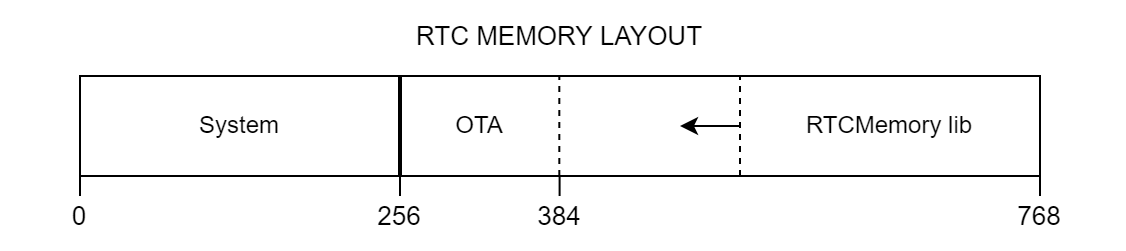
The ESP can also use the ADC to measure the supply voltage (VCC). To do this, include ADC\_MODE(ADC\_VCC); at the top of your sketch, and use ESP.getVcc(); to actually get the voltage.  
If you use it to read the supply voltage, you can’t connect anything else to the analog pin.

# ESP8266 RTC Memory structure

ESP8266 has a total amount of 768 bytes of RTC memory. However, the underlying SDK (System)reserves the first 256 bytes, leaving the remaining 512 bytes to the user application.

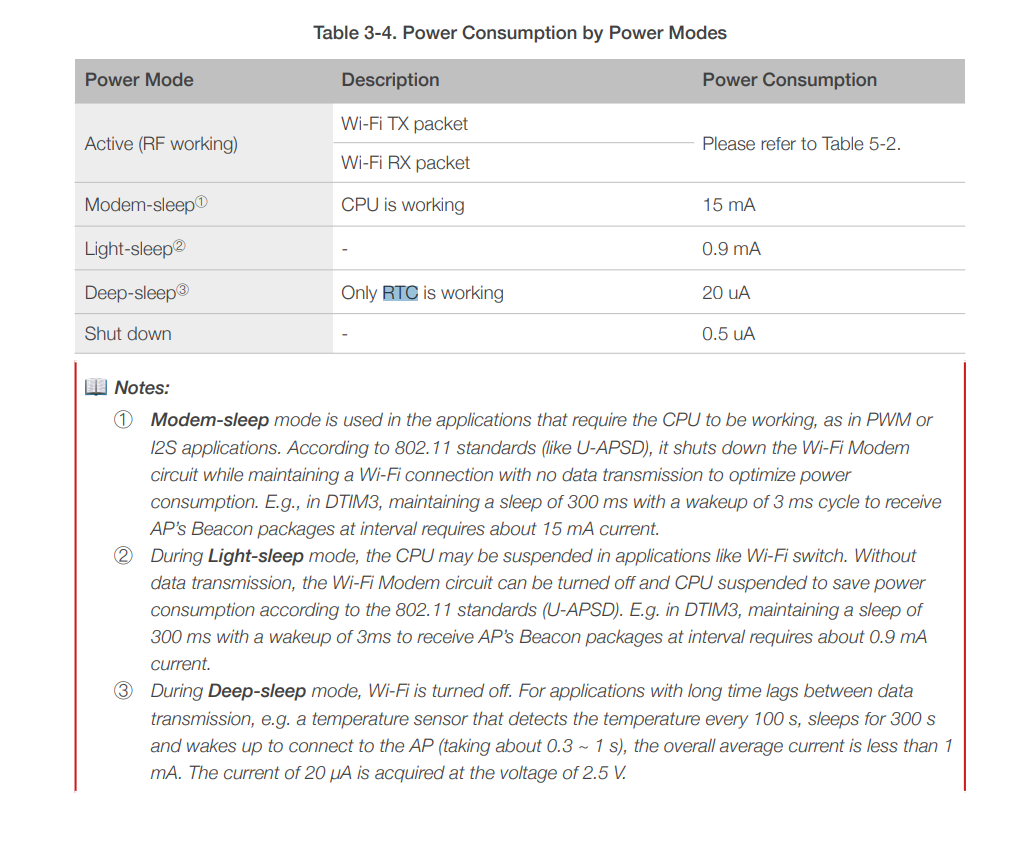
Moreover, much ESP8266 firmware use OTA functionality, which requires the first 128 bytes of user memory.

In this condition, to avoid OTA and RTCMemory overlapping each other, RTCMemory must limit its size to 384 bytes.



If OTA is not used then User memory can be started from the byte 256.

# ESP8266 power saving mode



# Hardware block diagram



# Firmware design

