[[1]](#footnote-1)

Athul Raj Vattappilly Sunilkumar

*Email:* [*athul.vattappilly@st.ovgu.de*](mailto:athul.vattappilly@st.ovgu.de)

Baizil MD

*Email: baizil.mulakkampilly@st.ovgu.de*

Scion IoT Smart Home – Smart Smoke Detector

*Abstract*- Internet of things is an evolutionary idea which connects different physical things to the internet. It has huge business opportunity in the regards of home automation, as we become more automated on every other sector. It would be essential for our future to adapt IoT in to the building sector. When considering our homes, providing the household equipment’s the ability to communicate each other, will make it more secure and smarter. Through this project our goal is to design a smart device for smoke detection at home. A smoke detector makes use of different sensors in order to detect a smoke. We have a Raspberry pi with scion server installed on it as a central controlling device, also we have used an ESP module for establishing the communication between the smoke detector and Raspberry Pi. According to the data received from the smoke detector the ESP module will determine whether it is a test alarm or a genuine one, and the whole setup will be constantly in communication with other smart devices through SCION network and will be monitored by a controlled local machine.

# INTRODUCTION

T

he objective of this project is to develop a smart smoke detector by enhancing a conventional photo electric smoke detector available in the market. Unlike the conventional smoke detectors, a smart smoke detector possesses the functionality not only inform the user about a fire incident nearby, but also about the incidents regardless of the location of the user. Therefore, the smart gadget has to be designed to identify and differentiate between an actual smoke that triggers a true alarm, a test device condition that triggers the alarm, lastly a false alarm triggered due to malfunctioning of the device. Moreover, it should identify low battery voltage condition in order for timely battery replacement. The occurrence of any such incidents are to be instantly informed to the user.

# Overview

T

he parts of the Smart Smoke detector could be explained in detail, with the help of a Functional Block diagram as can

be seen in Figure 1. The central part of the device is a Conventional Smoke detector converted to a smart module,

achieved by connecting a Wifi Module to it. The Wifi module

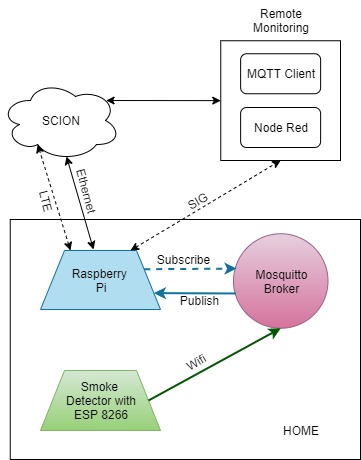


Figure 1. Functional Block Diagram

used here was an ESP 8266 which enables wireless communication and is connected to the to the inner circuit of a conventional smoke detector.

The communication protocol used is the MQTT, a lightweight messaging technique. The central communication point is the MQTT broker installed on a compact and powerful mini computer like Raspberry Pi. Moreover, for end to end communication and routing control the SCION internet architecture was used. This helped ensuring high efficiency and security for packet forwarding.

For remote monitoring a powerful and intelligent machine was used. The SIG (SCION IP Gateway) configured on the same enables connectivity between the same, which is in turn an MQTT client and the Raspberry Pi. To visualize the data received from the smart smoke detector, a flow-based development tool called Node - Red has been installed. Each part in the block diagram is explained in the following sessions.

## Photoelectric Smoke Detector

Those kinds of detectors are mostly reacting more to fires that happen due to long time smoldering. It is also known Optical Smoke detector. The main pasts include an infrared / ultra violet / conventional bulb and a photo diode. The photo diode acts as light receiver. All of these pasts are safely packed in a sensitive chamber. In these type of alarms, light intensity is focused on to the enclosed sensing chamber off the sensor. The smoke due to fire incident enters the chamber, which reflect light onto the sensor, and the alarm set on.

In the scope of this project, the initial goal is to gain enough knowledge about the working of smoke detectors and Microchip modules. The smoke detector used here is based on the MC145010 IC. This Photoelectric Smoke detector IC is an advanced smoke detector component containing sophisticated very low power circuitry. The IC is used with an infrared photoelectric chamber. Detection is accomplished by sensing scattered light from minute smoke particles or other aerosols. When detection occurs, a pulsating alarm is sounded via on-chip push-pull drivers and an external piezoelectric transducer.

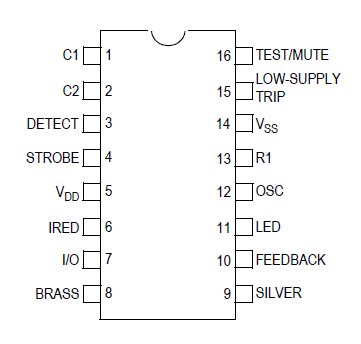


Figure 2. Pin Diagram of MC145010 [8]

Inside the detector, there is a light-sensing chamber. In this chamber, a LED projects light in a straight line, when smoke enters this chamber it cuts the LED straight line inside the chamber which rings the alarm. The whole process is controlled by an IC MC145010.

The MC145010 IC has 16 pins, each one has its own purpose. For this project, the I/O pin is read continuously to ensure smoke detection. When a smoke is detected or tested the I/O pin gives a high output signal.

## Node MCU ESP 8266

The ESP8266 is a cheap Wi-Fi microchip. It possesses “full TCP/IP stack and microcontroller capability. NodeMCU is an open source IoT platform which includes a firmware that runs on the ESP8266 Wi-Fi SoC.

The development board equips the ESP-12E module containing ESP8266 chip having Tensilica Xtensa® 32-bit LX106 RISC microprocessor which operates at 80 to 160 MHz adjustable clock frequency and supports RTOS.

There’s also 128 KB RAM and 4MB of Flash memory for program and data storage. The ESP8266 Integrates 802.11b/g/n HT40 Wi-Fi transceiver, so it can not only connect to a WiFi network and interact with the Internet, but can also set up a network of its own, allowing other devices to connect directly to it.

## B.1 Power Requirement

* Operating Voltage: 2.5V to 3.6V
* On-board 3.3V 600mA regulator
* 80mA Operating Current
* 20 µA during Sleep Mode

Power to the ESP8266 NodeMCU is supplied via the on-board MicroB USB connector or alternatively, a regulated 5V voltage source.

## B.2 Peripherals and I/O

The ESP8266 NodeMCU has total 17 GPIO pins broken out to the pin headers on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including:

* ADC channel – A 10-bit ADC channel.
* UART interface – UART interface is used to load code serially.
* PWM outputs – PWM pins for dimming LEDs or controlling motors.
* SPI, I2C & I2S interface – SPI and I2C interface to hook up all sorts of sensors and peripherals.
* I2S interface – I2S interface if you want to add sound to your project.

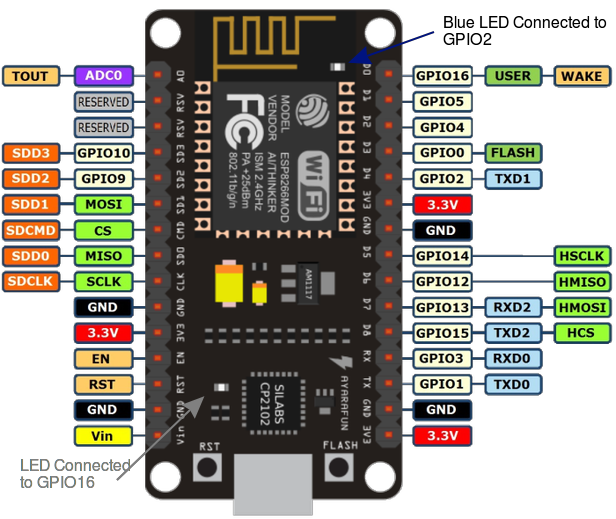


Figure 3. Pinout of ESP 8266 [13]

## B.3 On-board Switches & LED Indicator

The ESP8266 NodeMCU features two buttons. One marked as RST located on the top left corner is the Reset button, used of course to reset the ESP8266 chip. The other FLASH button on the bottom left corner is the download button used while upgrading firmware.

## B.4 Serial Communication

* CP2102 USB-to-UART converter
* 4.5 Mbps communication speed
* Flow Control support

## B.5 ESP 8266 Sleep Modes

In the scope of this project the ESP 8266 is made to operate in two modes, the Station and Access Point modes, where Access point mode allows the ESP to create its own network and on the other hand the station mode allows the ESP to connect to a Wi-Fi network.”[10][11][32] Since our main goal of this project is reduction of power consumptions, its important to explore the different sleeping modes of the chip as below:-

## B.5.1 Modem Sleep

Modem sleep mode is enabled only when the ESP is setup in Station mode. While in modem sleep mode the ESP will disable the modem with maintaining a Wi-Fi connection. The ESP stays connected to modem through DTIM beacon mechanism. The usual DTIM beacon interval of the modem is 100ms to 1000ms. The modem sleep is usually used in applications where the CPU powered on.

## B.5.2 Light Sleep

The light sleep mode is very similar to the modem sleep mode, the difference is that the ESP will also power off clock and suspends the internal CPU in order to reduce the power consumption in modem sleep mode.

## B.5.3 Deep Sleep

In deep sleep mode the system will turn off everything the only working module would be RTC (Real Time Clock). The deep sleep mode can be activated by connecting the RST pin to D0 pin.

## MQTT (MQ Telemetry Transport)

It is a machine-to-machine (M2M) Internet of Things (IoT) connectivity protocol designed as an extremely lightweight publish/subscribe messaging transport. In contrast to HTTP with request/response paradigm, this protocol is event-driven and enables messages to be pushed to clients. The central factor of communication is the MQTT broker, which also handles sending and receiving of all messages between the senders and the matching receivers. Each client publishes a message to the broker, through a topic. The routing information for the broker comes from the topic. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client. Therefore, the clients don’t have to know each other, they only communicate over the topic. Hence even if the clients do not know right match, they still communicate over the right topic. This helps in eliminating the dependencies between the data source and receivers as it ensures scalability solutions.

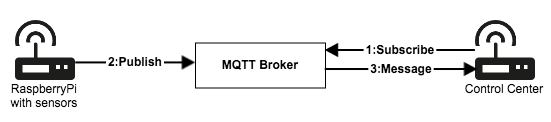


Figure 4. Communication between the sensor client and the control center over MQTT [14]

Here the data from Smoke Detector is transferred to a control center over the internet as an example. The sensors will be connected to a Raspberry Pi, which acts as gateway to the MQTT broker. The control center (remote monitoring) to the opposite side of the MQTT Broker and sensors also acts as an MQTT client which receives data.

## SCION Network and SCION-IP-Gateway (SIG)

Unlike regular communication protocols, this device uses Scalable, Control and Isolation on Next Generation Networks (SCION) for communication. The idea is to make sure that the communication happening between a client and server machine purely goes through SCION network/path creating an environment which purely includes all distinguishing factors of SCION when compared with regular TCP/ IP networks.

The SCION IP Gateway (SIG) enables legacy IP applications to communicate over SCION. Every SCION AS that wants to communicate with the legacy IP world or wants to connect with the legacy hosts inside other SCION ASes must deploy at least one SIG service. This service takes care of encapsulating the IP traffic into the SCION one at the sender side, decapsulating the SCION traffic into the original IP packets at the receiver side and correctly routing the information to the right legacy end-host through the SCION network. The SIG also takes care of mapping in which SCION AS a certain destination IP address belongs to in order to provide correct routing information through SCION.

Interconnecting the legacy Internet with the SCION network through the SIG service brings benefits to the legacy world. More specifically, the SIG’s design provides a way to mitigate DoS and DDoS attacks carried out through the encapsulated legacy traffic transported between one SCION AS to another.

## Node-RED

Node-RED is a programming tool for wiring together hardware devices, APIs and online services. It for wiring together hardware devices, APIs and online services. Flows can be then deployed to the runtime in a single-click. JavaScript functions can be created within the editor using a rich text editor. A built-in library enables to save useful functions, templates or flows for re-use.

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

# Circuit Design

This chapter discuss in detail about the project circuit design. The major challenge was to build a compact circuit in order to integrate with the ready-made conventional Photo Electric Smoke detector and fit inside the same. The circuit design was achieved though a web-based simulation, design, PCB design tool. To explain this in detail this circuit design is as in figure 5 and can be divided into three integral parts as below.

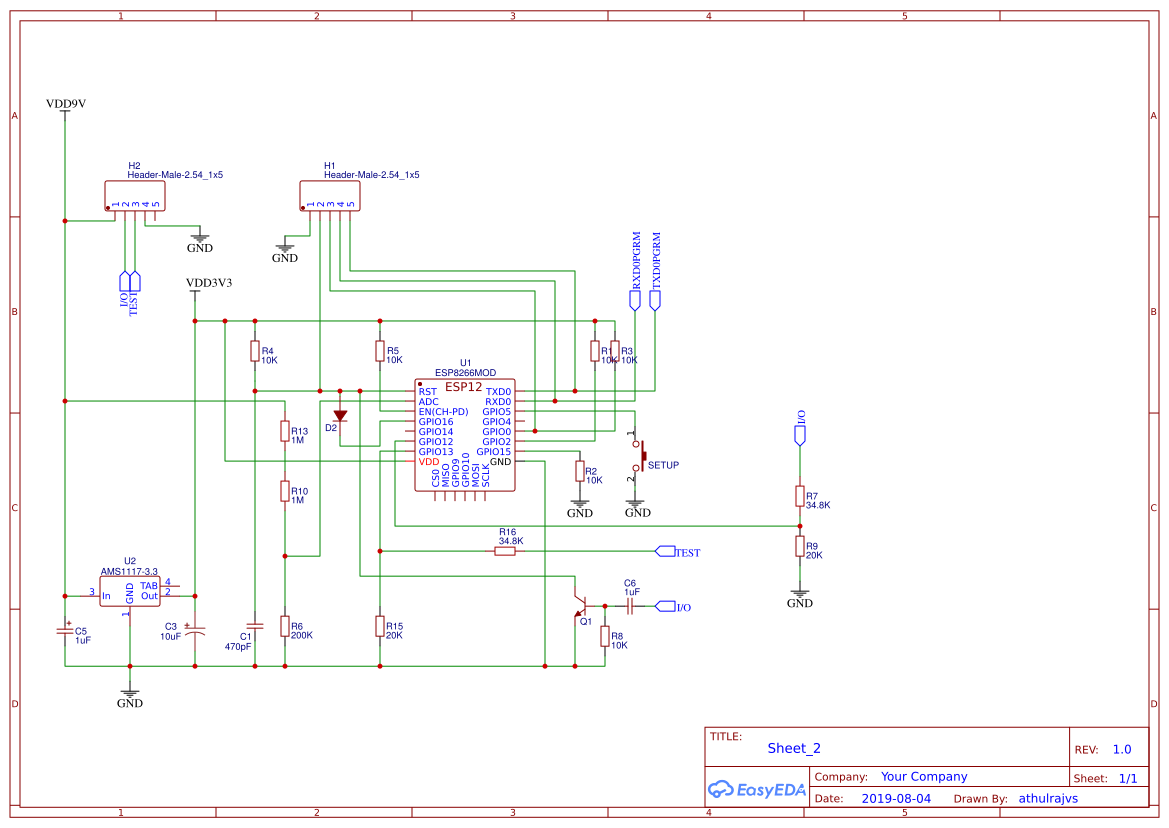


Figure 5. Circuit Diagram

## Interconnection between ESP12 chip and smoke detector

The ESP 12F core is removed from the Node MCU ESP 8266 board. This is because the ready-made chip is as can be seen in Figure 3 is a combination of voltage regulator, peripherals, I/O pins, communication ports, switches etc. Not only does it help in making the circuit compact and fit in the smoke detector but also reduce the overall power consumption drastically. But it should also be seen to it that the chip should still be programmable for which we consider integrating the below circuit (Figure 6) in to the main one.

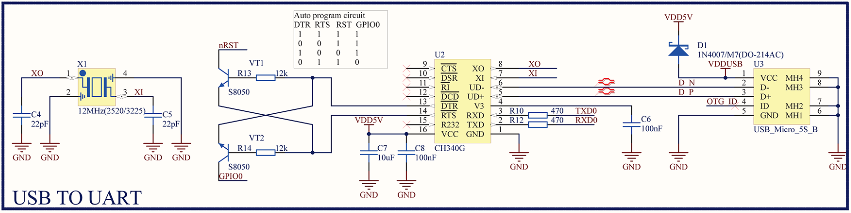


Figure 6. USB to UART (for programming) [13]

The pin connections of the ESP chip are done as in Figure 6 and additionally the The I/O pin from smoke detector IC and Test button from the smoke detector is connected to the GPIO 12 and GPIO 13 of the ESP 12F chip respectively.

## The Circuit setup to wake ESP12 up from Deep Sleep

Due to the power concerns, it is important to run the ESP 12F chip in deep sleep mode and to wake up if triggered externally. This is achieved by using the I/O pin from smoke detector IC as a trigger. The reason behind this is that, the I/O pin is always triggered in case if the Test button is pressed or in case of real smoke. The circuit to wake up the ESP chip from deep sleep mode is as in below figure.

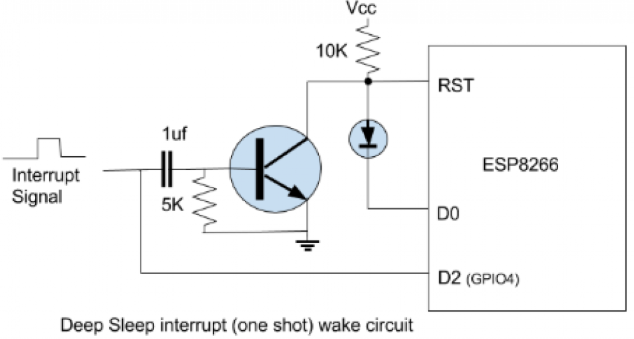


Figure 7. Interrupt circuit to wake from deep sleep [12]

## Voltage regulator circuit, ADC, Setup circuit

The voltage regulator used here is the AMS1117(CD 3.3) series of adjustable and fixed voltage regulator. It is also known as a Low Dropout Voltage Regulator (LDO) designed to provide 1A output current and operate down to 1V input-output differential. The dropout voltage of the device is guaranteed maximum 1.3V at maximum output current, decreasing at lower load currents. The circuit setup is as in Figure 8.

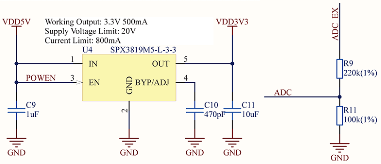


Figure 8. Voltage regulator circuit [13] Figure 9. ADC circuit [13]

One of the major requirement is to check the battery level of the device. The smoke detector gives a warning signal when the actual voltage falls below 6V. The output of the voltage regulator is 3.3V. To check the battery level the LDO output and is fed to a voltage divider and stepped down to 1V range (0.9V approx.). To check the voltage level ADC in the ESP chip is made use of, which can only handle a maximum voltage of 1V. If the voltage level falls under 0.6V then it raises a condition for low battery. The calculation of the battery performance is explained in detail in the Battery Performance Calculation section.

Additionally, a setup button has also been incorporated. This is a push button and is connected to the GPIO5 pin of ESP chip. A press on this button along with a press on the Test button takes the smoke detector to the configuration mode. This enables to reconfigure the smart smoke detector. This is to ensure the re-use of smoke detector for multiple users in different places or environments. This is explained more in the

Wifi Manager section.

# PCB Design

This chapter discusses about the PCB design. As in most of the devices an electronic components or circuits are an integral part. As mentioned earlier, the design of the PCB was done using the online design tool the EasyEDA.

The manufacture and development of the same was achieved with the help of “FLEXtronic Makerlab”[37]. The “planning, circuit design, circuit board design and production” of the PCB was done with various tools and machines that were available in the lab. The main machine system used for this was the Laser Surface Texturing Machine. This helps in manufacturing a real PCB in precise fashion from a design structured through simulation tool. The machine looks as in Figure 10.



Figure 10. Laser Surface Texturing Machine [38]

This system is used to repanel and structure the PCB with an attached laser source. This source U-V radiations at 355 nm wavelength. The whole system contains five main parts that performs vital functions: -

• Lens attached deflection system

• Table for processing

• Laser

• Chill source

• Camera setup

Detailed documentation could be found in references.[38][39]

Figure 11 shows the complete PCB design done using EasyEDA tool. The manufactured PCB was compact and was integrated with the circuit conventional photo – electric smoke detector bought from the market.

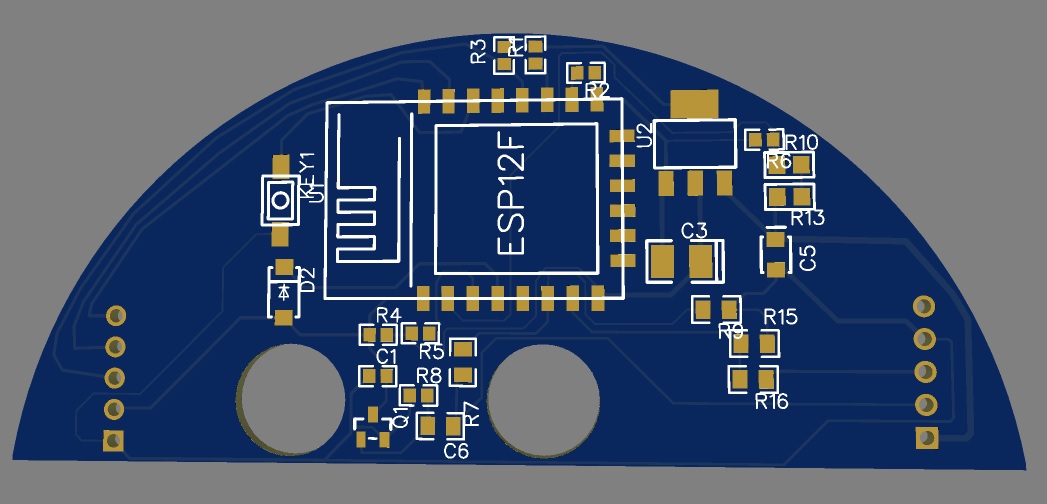


Figure 11. PCB designed using EasyEDA

# Wi-Fi Manager

The ESP 8266 has two modes of operations- Station mode and Access Point mode. Initially, when the ESP 8266 is starts up it is configured in Station mode and tries to connect to a Wi-Fi network which is previously saved in ESP 8266. But in case if the connection is unsuccessful for instance the ESP8266 is brought up into a new Wi-Fi network, it gets configured in to Access Point mode. The ESP 8266 will open up a DNS and a webserver. The ESP 8266 then can be connected to the new network by using a Wi-Fi enabled device.

As mentioned above, when launched in Access point mode, it launches a captive portal. It gives the provision to connect to a list of access points achieved through scanning. Any of the networks could be selected, if protected the password could be entered and connection shall be successful. The Captive Portal (Wi –Fi configuration) is as in Figure 12.

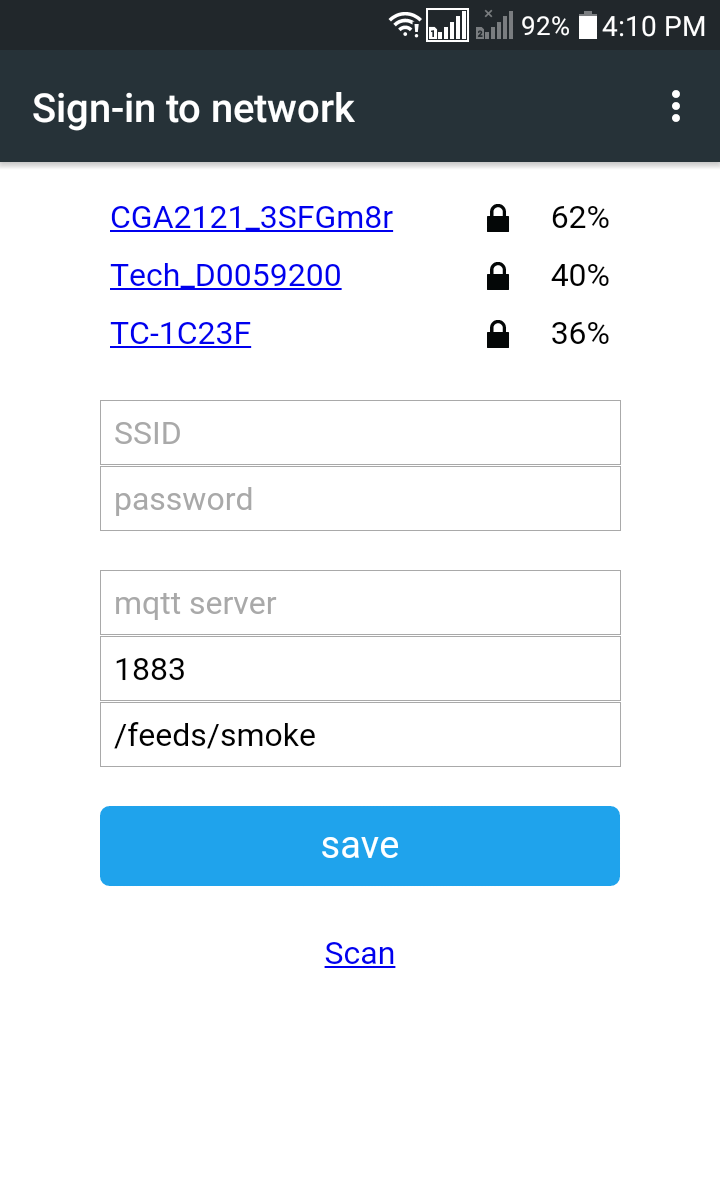
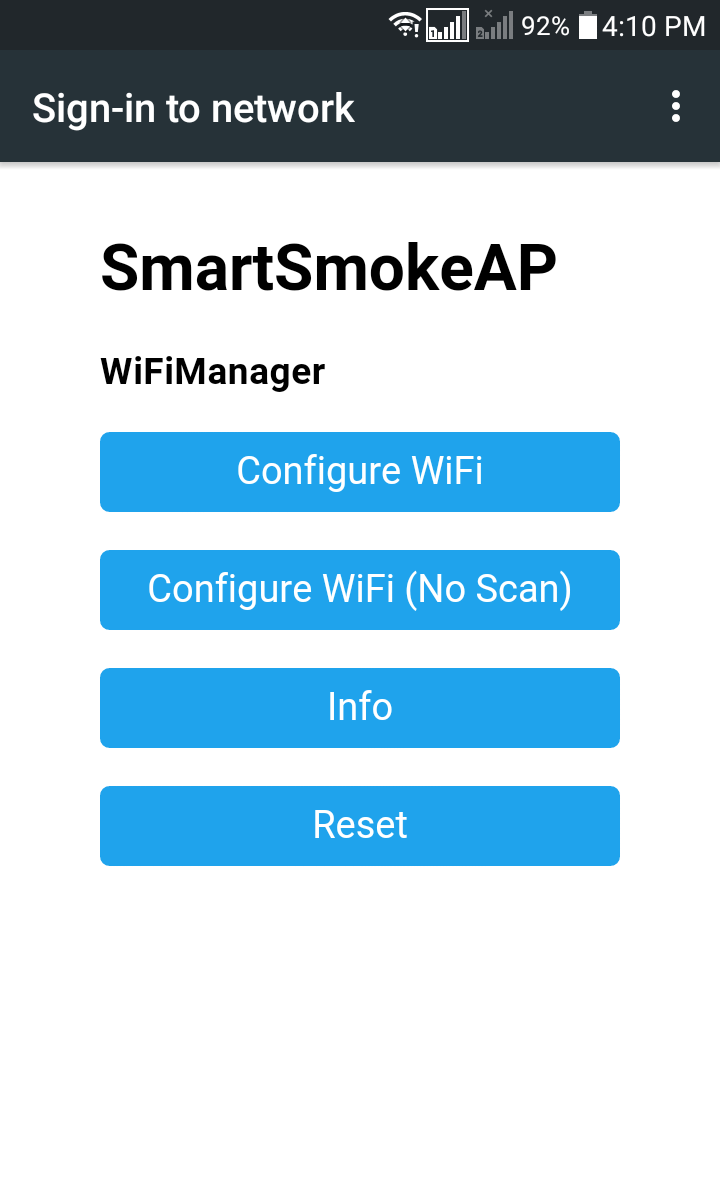


Figure 12. Wi-Fi Configuration [32]

Once the ESP 8266 is connected to a network the credentials would be saved in to the ESP 8266 EPROM. If the module again comes under the same network, it would connect aautomatically through Station mode of operation as the credentials of the previously connected network are stored.

# Battery Performance Calculation

In this projects scopes, making the whole system in to a stand-alone product had great importance. For that the same battery used in the smoke detector should be use to power the ESP 8266. This has a great concern; the Wi-Fi communication would make the battery to drain quickly. Making the ESP8266 to work on deep sleep mode will reduce the battery drainage and reducing the data transmission would also be beneficial. Moreover, since the Node MCU ESP 8266 would consists of several components integrated to it, it definitely draws more current. The power consumption rating was already discussed in detail. Thus, the ESP 12F core was removed from the board. Lastly, The ESP will send the data only when there is a smoke is detected or when the smoke is tested. It would also check the battery percentage once in day. For calculations, the power consumption ratings of the ESP12 chip was used. The battery consumption for this project is as below.

Current consumption while ESP wakes up: 180 mA

Current drawn while ESP 12 chip runs in Deep Sleep: 30 µA

Time (seconds) ESP remains woken up: 10 s

: 0.0027 hrs

Time (seconds) ESP runs in Deep Sleep a day: 86390 s

: 23.9972 hrs

Therefore, current drawn a day can be estimated as:

(180 x 0.0027) + (23.9972 x 0.03)

= (0.486 + 0.7199)

= 1.2059 mAh

Battery capacity: 500 mAh

Days the battery would last: 500 / 1.2059

= 414 days

Therefore, to be precise, the commercially available battery used in the in market would last more than 400 days. For more durability, a battery with a higher capacity could be used.

# Code Logic

## The ESP8266 is using the MQTT protocol for the data transmission. So, for the interaction between the Raspberry and the ESP 8266 a MQTT broker/client environment must be setup. The raspberry would be programmed as a MQTT broker. In MQTT protocol the main important participates are Publishers and Subscribers, a broker can act as both Publisher and Subscriber. In this project the ESP 8266 is assigned as a Publisher which will publish a topic and the Raspberry pi as subscriber which would be subscribed in to that topic.

## The code logic is based on the following Flow Chart (Figure 13) or Internal Block Diagram.

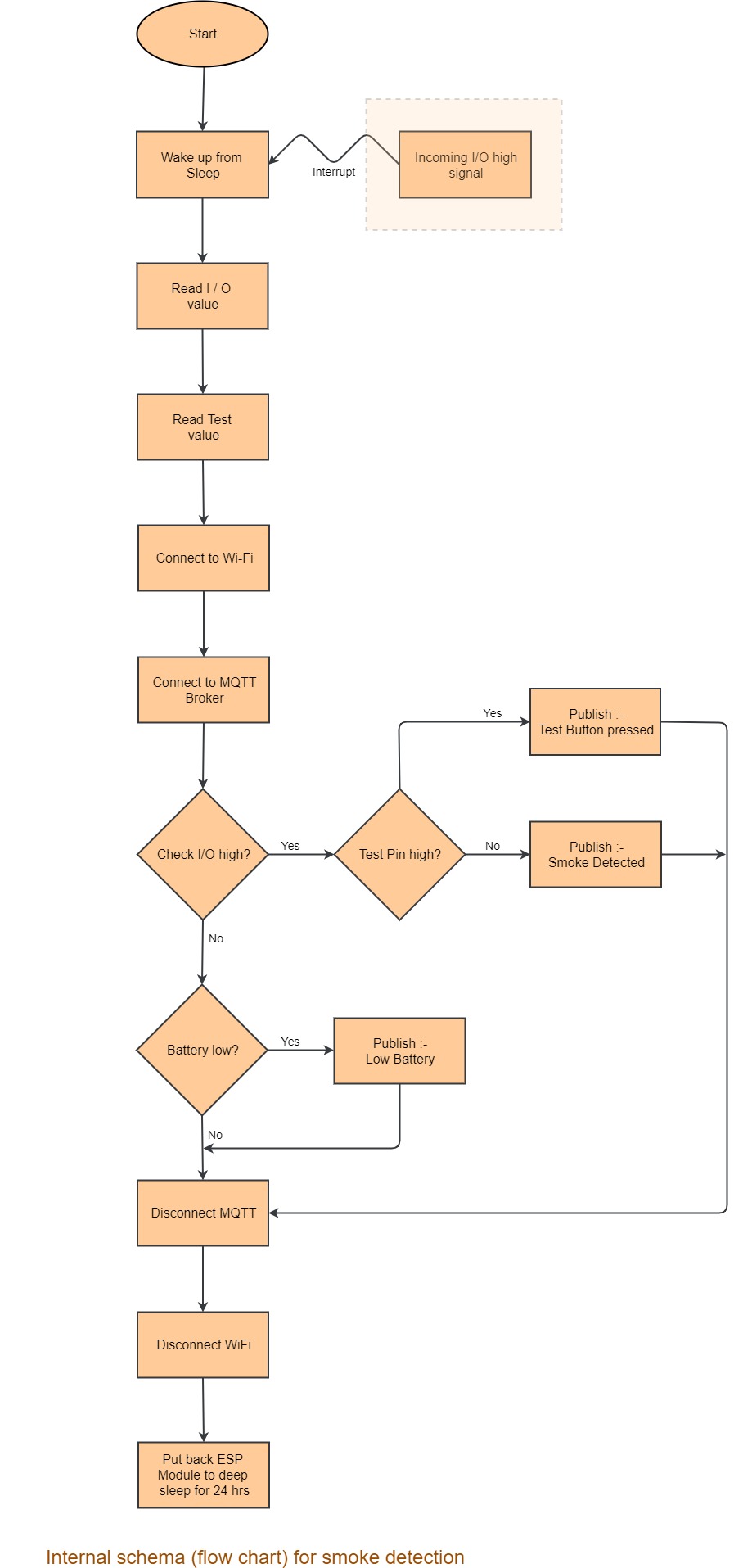


Figure 13. Internal block diagram (program flowchart)

# Node-Red Flow

Node-Red, a flow-based GUI tools based on JavaScript is used to visualize the data received from Smoke Detector. The GUI is developed in a local machine ´which would be useful for the user get information about the current situation.

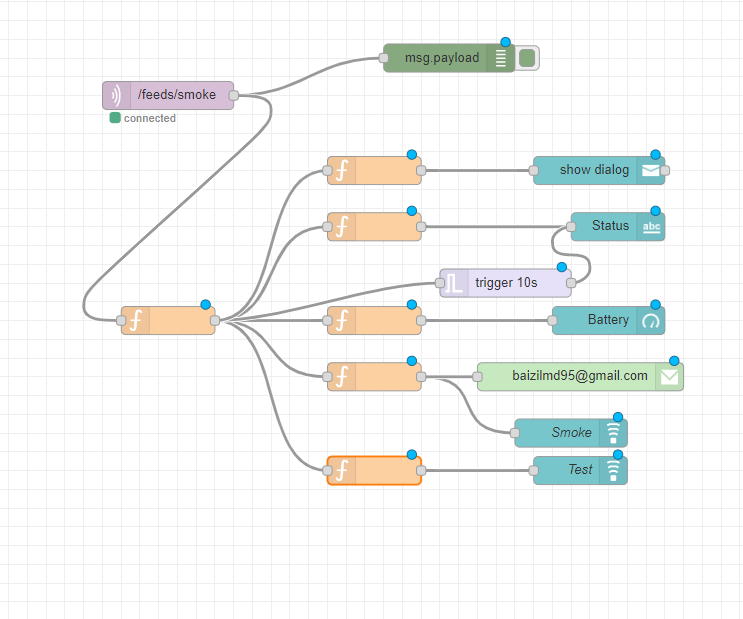


Figure 14. Node-Red Flow

The data from the smoke detector is received through a MQTT- in node. Which is then separated according to the data, each message is displayed in the dashboard. The battery percentage is displayed using a gauge node. When the detector detects smoke, an immediate warning is displayed in the GUI using a notification node along that an Email is sent in to the user’s mail id through mail node in the node red. The Figure 14 demonstrates the Node – Red flow.

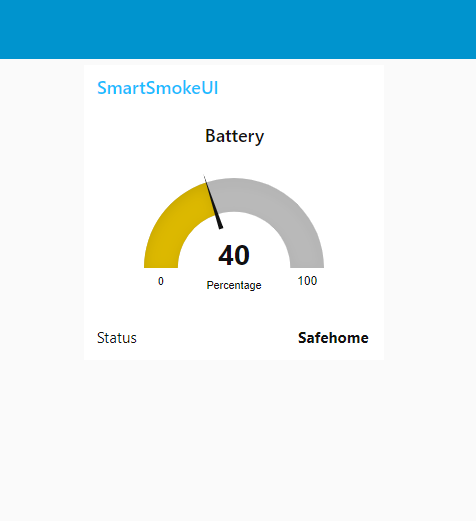
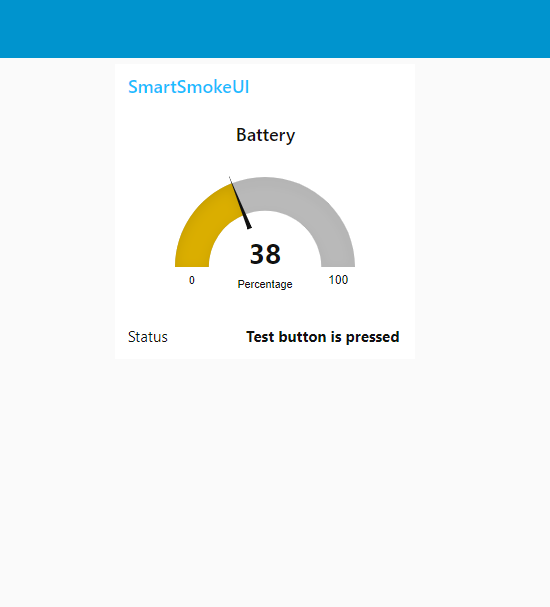
 

Figure 15. Battery percentage visualised using Node-Red

The Figure 15 shows the battery percentage calculation using Node Red tool.

# Conclusion and further works

This purpose of this scientific project is to enhance a conventional smoke detector performance with the help of an ESP8266 and SCION network. On completion of this project, a low-level working model is developed which detects the smoke and test alarms and handover notifications to the user. A further development to this project would be establishing a whole security system for an apartment with similar smoke detectors connected through the scion network.

Another future development would be identifying the difference between a smoke and fire in order to avoid false alarms.

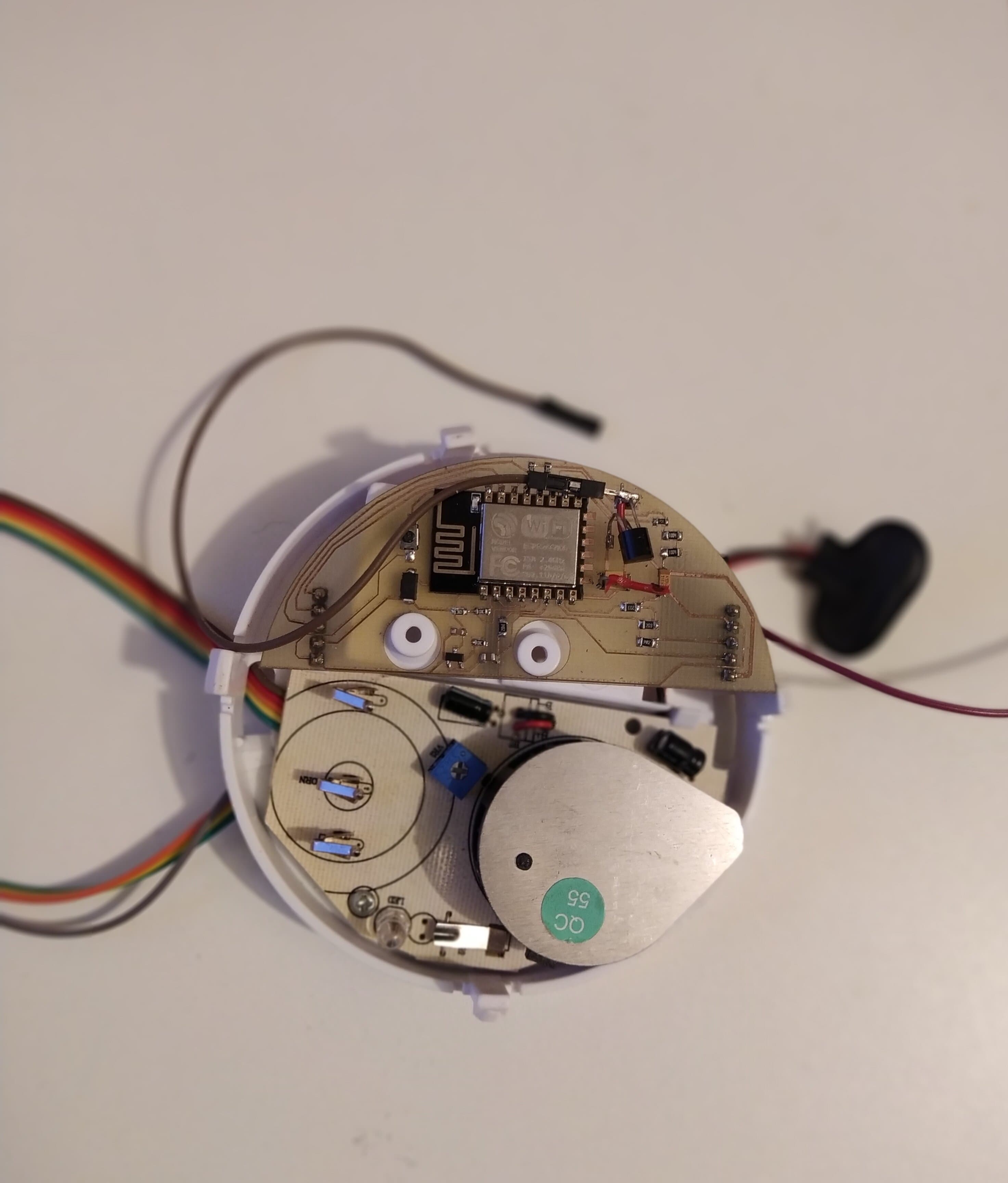
 

Figure 16. Final Product

References

[1] URL: <https://www.firstalert.com/what-is-a-photoelectric-smoke-detector/>

[2] URL: <https://www.nfpa.org/Public-Education/Staying-safe/Safety-equipment/Smoke-alarms/Ionization-vs-photoelectric>

[3] URL: <https://en.wikipedia.org/wiki/Smoke_detector>

[4] Welcome to SCION Tutorials.

Available: <https://docs.scionlab.org/>

[5] Welcome to SCIONLab

Available: <https://www.scionlab.org/>

[6] SCION IP Gateway (SIG)

Available: <https://docs.scionlab.org/content/apps/remote_sig.html>

[7] Piergiacomo De Marchi, “Gateway Architectures for

Interaction between the Current Internet and Future Internet

Architectures”, Padova 2017. Available: <http://tesi.cab.unipd.it/57086/1/UNIPD_Thesis.pdf>

[8] Freescale Semiconductor Technical Data, “Photoelectric Smoke Detector IC with I/O”, May 2006.

Available: <https://www.nxp.com/docs/en/data-sheet/MC145010.pdf>

[9] Insight Into ESP8266 NodeMCU Features & Using It With Arduino IDE,

Available: <https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/>

[10] NodeMCU Documentation

Available: <https://nodemcu.readthedocs.io/en/master/>

[11] ESP32 Deep Sleep with Arduino IDE and Wake Up Sources. Available: <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>

[12] URL: <https://diyprojects.io/esp8266-deep-sleep-mode-test-wake-pir-motion-detector/>

[13] Nodemcu pinout.

Available: <https://circuits4you.com/2017/12/31/nodemcu-pinout/>

[14] Getting Started with MQTT

Available: <https://www.hivemq.com/blog/how-to-get-started-with-mqtt/>

[15] Generic ESP8266 Module

Available: <https://arduino-esp8266.readthedocs.io/en/2.5.1/boards.html#esp-to-serial>

[16] How to Install Mosquitto Broker on Raspberry Pi

Available: <https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/>

[17] Make a Web Server with ESP8266

Available: <https://www.allaboutcircuits.com/projects/make-a-web-server-with-esp8266/>

[18] URL: <http://www.martyncurrey.com/esp8266-and-the-arduino-ide-part-5-adding-wifimanager/>

[19] ESP8266 Web Server with Arduino IDE

Available: <https://randomnerdtutorials.com/esp8266-web-server-with-arduino-ide/>

[20] URL: <https://randomnerdtutorials.com/esp8266-web-server/>

[21] How to use ArduinoJson with PubSubClient

Available: <https://arduinojson.org/v6/how-to/use-arduinojson-with-pubsubclient/>

[22] URL: <https://tttapa.github.io/ESP8266/Chap10%20-%20Simple%20Web%20Server.html>

[23] ESP32 Access point (AP) and Station web server with HTML using Arduino IDE

Available: <https://circuits4you.com/2018/11/21/esp32-access-point-station-ap-html-web-server-arduino/>

[24] ESP8266 Web Server AP (Access Point)

Available: <https://circuits4you.com/2016/12/16/esp8266-web-server-ap/>

[25] URL: <https://circuits4you.com/2016/12/16/esp8266-web-server-html/>

[26] URL: <https://arduinojson.org/v6/api/jsondocument/>

[27] DynamicJsonDocument. Available: <https://arduinojson.org/v6/api/dynamicjsondocument/>

[28] URL: <https://github.com/Nithinshah/ESP8266-HTML-WiFi-Manager>

[29] URL: <https://github.com/esp8266/Arduino>

[30] URL: [https://github.com/wemos/D1\_mini\_Examples/blob/master/ examples/01.Basics/ReadAnalogVoltage/ReadAnalogVoltage.ino](https://github.com/wemos/D1_mini_Examples/blob/master/%20examples/01.Basics/ReadAnalogVoltage/ReadAnalogVoltage.ino)

[31] URL: <https://github.com/tzapu/WiFiManager/issues/65>

[32] URL: <https://github.com/tzapu/WiFiManager>

[33] WiFiManager ESP8266 WiFi connection manager library (Arduino IDE), Available: <https://tzapu.com/esp8266-wifi-connection-manager-library-arduino-ide/>

[34] Sinohara, Hisato. "Laser scribing system and method." U.S. Patent No. 4,861,964. 29 Aug. 1989.

[35] Kovalchenko, Andriy, et al. "The effect of laser surface texturing on transitions in lubrication regimes during unidirectional sliding contact." Tribology International 38.3 (2005): 219-225.

[36] URL: <https://www.researchgate.net/publication/228343441_State_of_the_Art_in_Laser_Surface_Texturing>

[37] URL: <https://forschung-sachsen-anhalt.de/incubator/flextronic-i8793>

[38] URL: <https://data.stimulate.ovgu.de/f/8814c4249e704186a4fd/>

[39] ProtoLaser U4User manual, Available: <https://data.stimulate.ovgu.de/d/d129ec852ba2435caef3/files/?p=/_data/pdf/ProtoLaser%20U4_BA_V2.0_ENG.pdf>

[40] Marcus Prier, LPKF Documentation, Otto-von-Guericke University Magdeburg, Available: <https://data.stimulate.ovgu.de/f/8ac76fbdf95a4539a840/?dl=1>

[41] URL: <https://nodered.org/>

[42] URL: <https://nodered.org/docs/getting-started/raspberrypi>

[43] URL: <https://github.com/node-red/node-red>

1. [↑](#footnote-ref-1)