University College London

ELEC0033: Internet Of Things

Final project report

Authors:

Sreethyan Aravinthan
SN: 17034996

Constantina Papavasileiou
SN: 17001295

Mindaugas Jarmolovičius
SN: 17139494

Miriam Robledo
SN: 17080614

April 30, 2020

Contents

1 Executive Summary

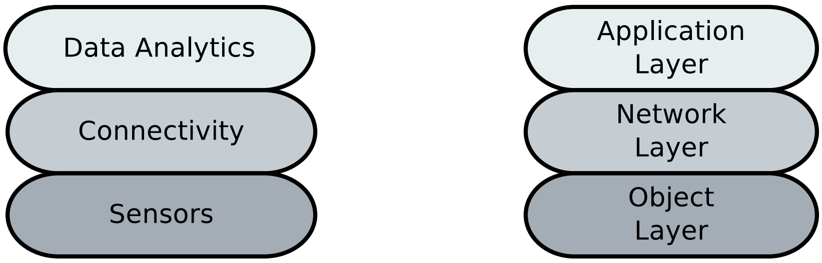
2 Business Case

3 Design Methodology and Implementation

3.1 The Problem

3.2 Background Theory

3.2.1 IoT Stack



(a) *Model 1* ***(b)*** *Model 2*

Figure 3.2.1: *IoT conceptual models*

Model 1
Model 2
Chosen Model

3.2.2 Characteristics of Sensors

Range
Maximum and minimum value range over which a sensor works well. Sensors may work well outside this range, but require additional calibration. e.g. the output may no longer be linear.

Accuracy
How well the sensor measures the environment in an absolute sense, i.e. how good the data is when compared with a recognized standard. e.g. a temperature sensor accurate to 0.001oC is expected to agree within 0.001oC with a known temperature standard. This is what you want to compare results with other observations.

Resolution
The ability of a sensor to see small differences in readings. e.g. a temperature sensor may have a resolution of 0.000,01^{o}C, but only be accurate to 0.001^{o}C. Can detect relatively small changes in temperature, smaller than the accuracy of the sensor. Resolution in often controlled by the quantisation in digitising the signal

so is not a function of the sensor itself, but of the sampling process.

Repeatability
This is the ability of a sensor to repeat a measurement when put back in the same environment. It is often directly related to accuracy, but a sensor can be inaccurate, yet be repeatable in making observations.

Drift/Stability
This is the low frequency change in a sensor with time, i.e., with a given input you always get the same output. It is often associated with electronic aging of components or reference standards in the sensor.

Response time
A simple estimate of the frequency response of a sensor assuming a change in the measurement.

Output
What output is given for a change in the parameter being measured. For example, a voltage range e.g. 0 to 5 volts for an input range of 0 to 30^{o}C.

Power Consumption
What is needed to power the sensor, quite often specified as the current draw.

Setting Time
After being switched on, how long before a valid measurement is ready.

Sampling time required
How often do we need to repeat measurements to get an accurate picture of the phenomenon being measured.

3.2.3 Communication Protocol for sensor data

SPI
I^{2}C

3.2.4 Methods for accessing external devices

Polling
Interrupts

3.2.5 Low Power Mode

3.3 System Design

This section describes overall design of our Internet of Things (IoT) device. The IoT core component is `Feather HUZZAH ESP8266` module containing `ESP8266` microcontroller with built-in Wi-Fi. This module also has built-in Lithium Polymer (LiPo) battery charging circuit, USB connectivity with serial converter to charge and program microcontroller; linear converter to supply 3.3V rail. Air Quality `SGP30` and temperature-humidity `BME280` sensor modules are connected via I^2C bus. An additional `IO16` pin is connected to reset pin enabling deep-sleep mode wakeup timer. Circuit diagram is shown in Figure 3.3.1.

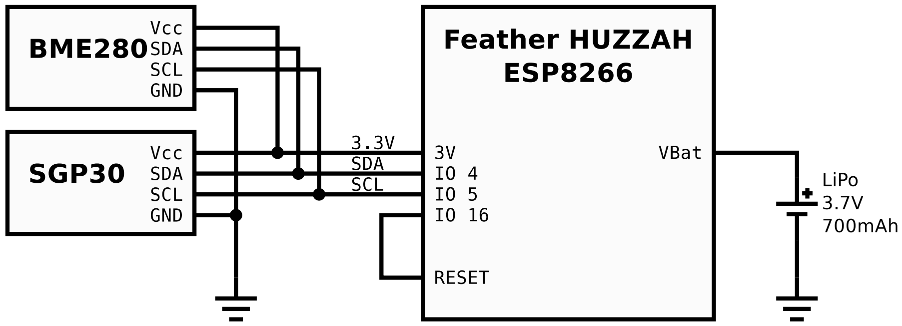


Figure 3.3.1: *Circuit of IoT device.*

With this configuration, even in deep-sleep mode both sensor modules are still powered as 3.3V bus is not affected. This is intentional as both sensors have low-power modes. `BME280` sensor that include temperature, humidity and pressure readings has three of such modes, sleep which does no operation and is at the lowest power; forced which perform one mesurement, store results and return to sleep; and normal mode for perpetual cycling of measurements. Additional oversampling and IIR filter settings can be set. Weather monitoring configuration recommended by datasheet was used which uses forced mode with one sample per minute without oversampling and IIR filter. This achieves reasonable noise and performance with average current consumption of $160nA$.

`SGP30` sensor is equiped to measure Carbon Dioxide (from 400 ppm), Total Volatile Organic Compounds (in ppb range), relative H_2 and Ethanol values. After power up, sensor low power sleep mode is activated and high current measurement mode is only actived when microprocessor sends a measurement request. Sensor internally takes samples at 1Hz in order to calculate baseline. During experimentation it was discovered that it takes around 16 seconds from power up to reading an accurate value. This lead to decision to not disconnect power from this sensor during sleep. In addition, this sensor features on-chip humidity compensation calculation, however it requires absolute humidity value to be provided by microprocessor. This value is calculated from `BME280` sensor readings with Equation 1:

$$AH = 216.7 \times \frac{\frac{RH}{100} \times 6.112^{\frac{17.62t}{243.12+t}}}{273.15 + t} \tag{1}$$

Where AH is absolute humidity in g/m^3, RH is relative humidity in percent, and t is temperature in $°C$.

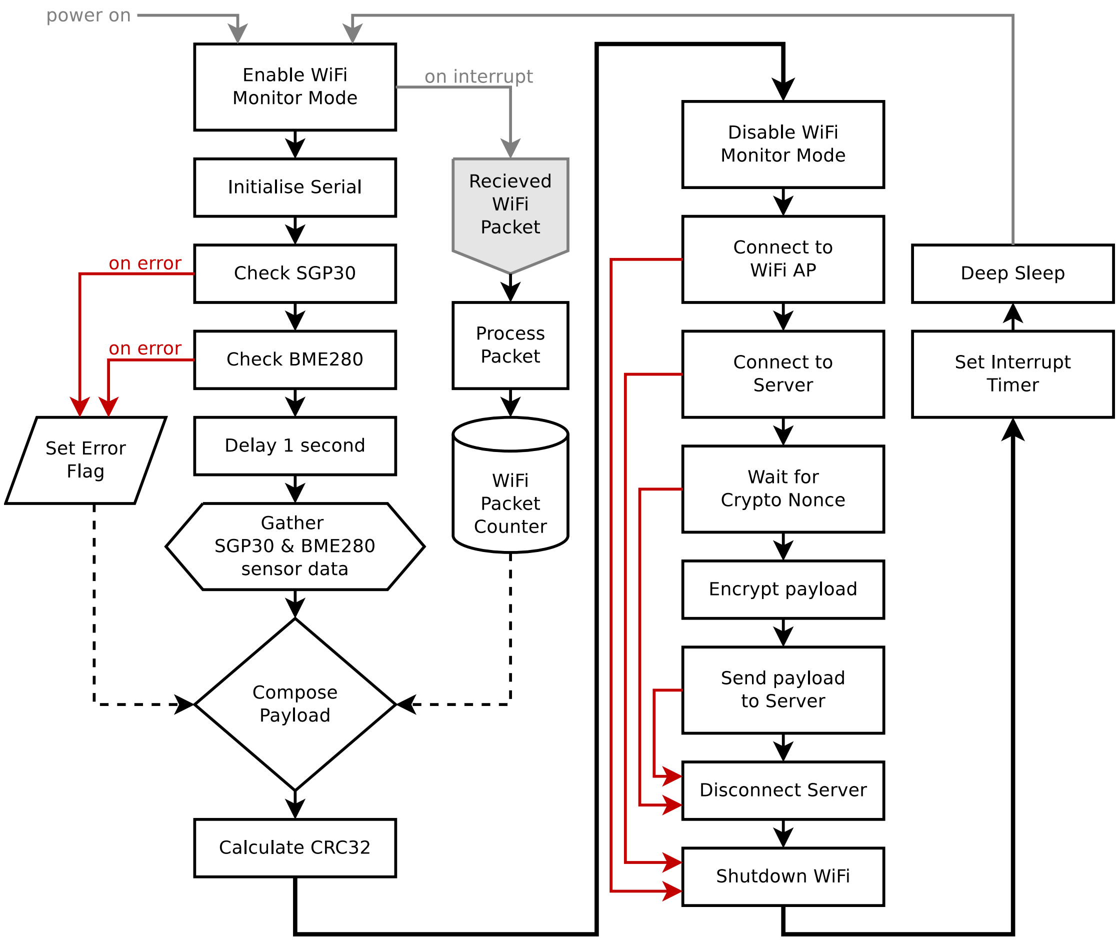


Figure 3.3.2: *Functional diagram of IoT device program.*

Figure 3.3.2 show a function diagram of implemented software in IoT device. It starts at enabling WiFi monitor mode which executes an interrupt routine on every time microcontroller's WiFi module has observed a WiFi packet. This packet is processed and WiFi packet counter in increased. The main routine then follows serial and sensor initialisation, if either of sensors do not respond, a error flag is set high for that sensor. Main routine then sleeps for one second in order for WiFi monitor routine to gather enough packets. Main routine then gathers sensor readings, composes a payload message with CRC32 checksum. Then a WiFi mode is changed to connect to Access Point, and further connect to server. Upon a new TCP socket server sends a cryptographic nonce which microcontroller uses to encrypt payload and send it to server. Afterwards server and WiFi connections are gradually disconnected and timer in deep sleep mode is set upon which program is reset.

	Content Name	**Size and type**
	Device ID	8bits
	Payload length*	8bit unsigned integer
Encrypted Payload **24-bytes**	Temperature Reading	32bit float
	Humidity Reading	32bit float
	Pressure Reading	32bit float
	WiFi Managed packets	16bit unsigned integer
	WiFi Control packets	16bit unsigned integer
	WiFi Data packets	16bit unsigned integer
	CO_2 Reading	16bit unsigned integer
	H_2 Reading	16bit unsigned integer
	TVOC Reading	16bit unsigned integer
	Ethanol Reading	16bit unsigned integer
	Status Flags	8bits
	reserved.	8bits
	CRC32 Checksum	32bits

Table 3.3.1: *List of message content sent from IoT device to server via TCP socket. * Payload length is represented in multiples of 12bits, value of 2 in this case.*

Message content sent to server is shown in Table 3.3.1. Sending data in raw binary was chosen as suppose to encoded with data interchange format such as JSON or XML due to bandwidth efficiency, computation time and simplicity of implementation with C. Checksum was required to ensure that all values sent are correct as there is no way to ensure that raw values were not corrupted. ChaCha20 cipher was used for encryption with 12 byte nonce. This nonce size requires payload to be in multiples of 12 bytes blocks which is the reason for having "reserved" byte in payload. Message authentication code (such as Poly1305) was not used due to simplcity reasons. Instead Device ID is not encrypted and used by server to lookup symmetric key to decrypt the payload.

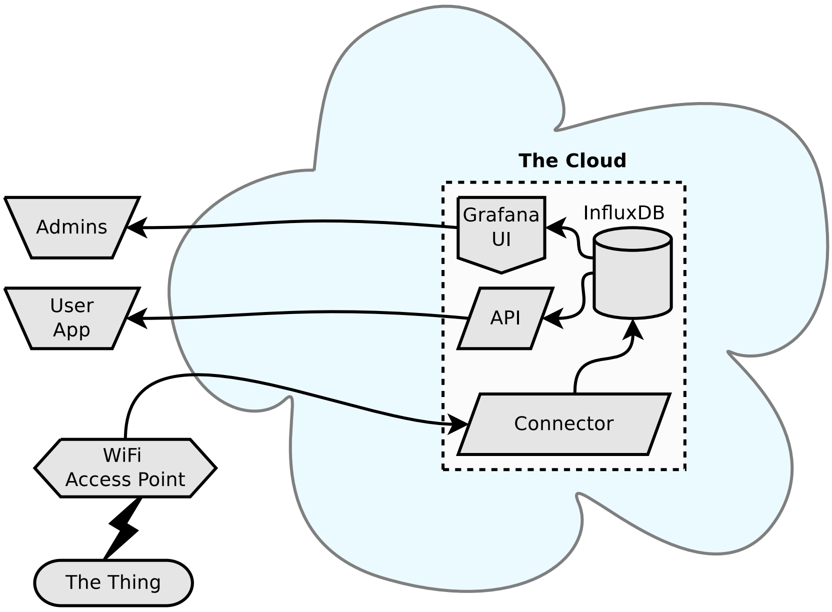


Figure 3.3.3: *General flow diagram of whole system connectivity.*

3.3.1 Layer 1 approach

3.3.2 Layer 2 approach

3.3.3 Layer 3 approach

4 Experimental Results

5 Engineering Trade-offs

Appendices