HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

DEPARTMENT OF MECHANICAL AND MEDICAL ENGINEERING

SMART SYSTEMS

SCIENTIFIC PROJECT REPORT

SENSOR VALUES TO AWS IOT USING MQTT PROTOCOL

UNDER THE GUIDANCE OF

DR. PROFESSOR. ROBERT HÖNL

SUBMITTED BY

ASHISH GADAM ARUN KUMAR (264376)

VINAYAKA BHAT (264363)

08/08/2020

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

1.IoT – Internet of Things

2.AWS – Amazon Web Services

3.MQTT – Message Queuing Telemetry Transport

4.VOC – Volatile Organic Compounds

5.HVAC - Heating ventilation and Air conditioning

6.CS – Chip Select

7.I2C – Inter Integrated Circuit

8.SPI – Serial Peripheral Interface

9.USB – Universal Serial Bus

10.UART – universal asynchronous Receiver Transmitter

11.GPIO – General Purpose Input Output

12.HTML – Hyper Text Mark Up Language

13.CSS – Cascading Style Sheets

14.JS – Java Script

CHAPTER 1

ABSTRACT

Internet of Things (IoT) is expected to play a major role in our lives through pervasive systems of sensor networks encompassing our environment. These systems are designed to monitor vital physical phenomena generating data which can be transmitted and saved at cloud from where this information can be accessed through applications and further actions can be taken.

This report presents the implementation and results of the surrounding monitoring system which employs sensors for temperature, humidity and pressure of the surrounding area. This data can be used to trigger short term actions such as remotely controlling heating or cooling devices or long-term statistics. The sensed data is uploaded to cloud storage and a web page or an application which links to the cloud and presents the results to the end users.

The system employs BME680 and ESP8266 Wi-Fi module which transmits data to AWS IoT Core cloud services using MQTT Protocol.

The whole idea of report is to summarize such device and elaborating the working principle and showing the real time report of such device which makes use of Internet of Things and its benefits.

CHAPTER 2

MOTIVATION

Internet of Things (IOT) has become quite an essential technology of 21st century and has revolutionized the world. This change in technology has affected our lives and the same has created sophistication.

Controlling such parameters according to our need is very well needed as well as it reduces the human efforts to be spent. Temperature control in a medical store where few medicines need adequate and optimum temperature to last long, gas leakage in chemical plant can cause a catastrophe etc. So, In order to maintain these parameters in an optimum range a device is required which can measure these quantities which will alert human to take care of the situation when necessary.

In order to make it work we needed to feed the data using physical device to demonstrate and develop the functionality and this motivated us to kick start the project.

CHAPTER 3

INTRODUCTION

Internet of Things (IoT) is expected to transform the world by making it possible to monitor and control important environmental or surrounding phenomena using the physical devices or sensors capable of capturing, processing and transmitting the data wirelessly to the remote storage like the cloud services which stores, analyse and exposes this data as a meaningful information. This information can be accessed over internet through various front-end user interfaces such as a web page or a mobile application, depending on the need and the intended goal.

Internet plays a critical role in this transformation by making it more efficient and reliable, and ensuring a smooth and swift communication of data from these IoT devices to the cloud and from the cloud to the users.

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

The “things” are capable of sensing, capturing and sending the data such as temperature, pressure, humidity, VOC etc.

Surrounding monitoring system is an important IoT application which involves monitoring and controlling the surrounding environment and presenting this data for effective actions such as remotely controlling the heating or cooling devices, sending notifications about the current status and a long-term data analysis.

The sensors are placed at various locations to collect the data in order to predict the behaviour of a particular area of interest. The main goal is to design and implement an efficient monitoring system through which the required parameters are monitored remotely using internet of things and the data gathered from the sensors are stored in the cloud and to project the estimated trend on a web page or mobile application.

The system consists of an environment sensor BME680 which is interfaced with the ESP8266 Wi-fi module (Node MCU) which transmits the sensed data through Internet to a remote cloud storage.

This is a low-cost system which gives insight into the design and implementation of a complete IoT application involving all aspects from sensing and wireless transmission to cloud storage and data retrieval from cloud via a web page or mobile application.

It involves a detailed study and deployment of Node MCU and BME680 and its interfacing with input and output modules such as sensors and Wi-Fi module, the usage of API for sending data to the cloud and development of a web page using HTML, CSS and JS.

The results of the project show the real-time monitoring of temperature, humidity and pressure levels from any location in the world and its statistical analysis. This system can be extended to enable remote controlling of various appliances based on the sensed data.

Our proposed system uses the Broadcast based approach wherein multiple devices subscribe for a MQTT Topic with the Broker. Temperature, Humidity and Pressure values are received from BME680 sensor and processed by the ESP8266 Node MCU. Sensor readings from Node MCU are published to the AWS IOT core that maintains the Shadow table for this Topic.

CHAPTER 4

PRINCIPLE OF WORKING

Block diagram

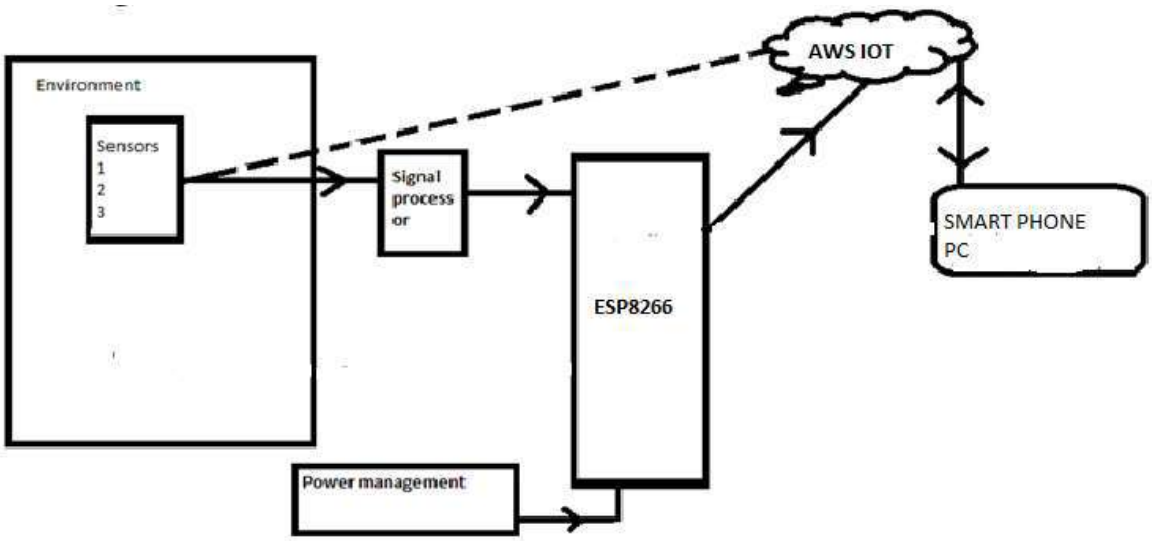


Figure 1: Block Diagram of our model

The field devices like sensors will collect the data from the surroundings and will feed the data to the signal processor which will be given to the Microcontroller .The data received in the controller will be converted to normal number format and will be displayed in the serial monitor first and then the data received will be extracted using the PLX-DAQ spread sheet and helps in uploading the same data to the web or cloud.

The same data can be seen, accessed or controlled by the user using Smart phones. The main concept we are using is the Internet Of Things to overcome the challenges and making life easy.

We are making using of Amazon Web services IOT core for this purpose.

In our project, we create a web page as a prototype for the user to see and access the data as well as control it.

CHAPTER 5

HARDWARE DESCRIPTION

In our project we are making use of both software as well has hardware components, First let us look at the hardware components in detail:

5.1 BME680 SENSOR

There are so many companies which manufactures BME680 environment sensor and out of these, Bosch and Adafruit package is popular and trust worthy to buy. In our project we are making use of Adafruit BME680 sensor.

The main advantage of using BME680 is it combines a gas sensor with air pressure, humidity, and ambient air temperature sensing functions within a single package. This replaces the use of separate sensors to measure each of the physical quantities.

The combo MEMS solution enables multiple new capabilities for portable and mobile devices such as air quality measurement, home automation and other applications for the Internet of Things (IoT).

The BME680 sensor offers high accuracy and low power consumption in an efficient and cost-saving manner.

Applications include:

- Smart homes
- Smart offices and buildings
- Smart energy
- Smart transportation
- Heating ventilation and Air conditioning (HVAC)
- Health monitoring
- Sports and Fitness.

BME680 gives us the results very precisely. Humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy.

Because pressure changes with altitude, and the pressure measurements are so good, we can also use it as an altimeter with ±1 meter or better accuracy.

Pin out configuration of BME680 sensor is as shown below:

Figure 2: Simple BME680 sensor [1]

The BME680 takes these sensors to the next step and it contains a small MOX sensor. The heated metal oxide changes resistance based on the volatile organic compounds (VOC) in the air, so it can be used to detect gasses.

The advanced version of BME680 looks like this:

Figure 3: Advanced BME680 sensor [1]

POWER PINS**:**

1.Vin - This is the power pin. Since the sensor chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down.To power the board, give it the same power as the logic level of your microcontroller

.

2.3Vo - This is the 3.3V output from the voltage regulator, we can grab up to 100mA.

3.GND - Common ground for power and logic. [1]

SPI PINS LOGIC:

All pins going into the breakout have level shifting circuitry to make them 3-5v logic level safe.

1. SCK - This is the SPI Clock pin, its an input to the chip

2. SDO - This is the Serial Data Out / Microcontroller In Sensor Out pin, for data sent from the BME680 to your processor

3. SDI - This is the Serial Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the BME680

4. CS - This is the Chip Select pin, drop it low to start an SPI transaction. It is an input to the chip [1]

If we want to connect multiple BME680's to one microcontroller, have them share the SDI, SDO and SCK pins. Then we assign each one a unique CS pin.

I2C LOGIC PINS:

- SCK - This is also the I2C clock pin, connect to the microcontroller I2C clock line.
- SDI - This is also the I2C data pin, connect to your microcontroller I2C data line. [1]

For the data transfer we make use of the SPI protocol or I2C protocol. There is a lot of difference between these two protocols. Let us have a look at it.

Features of BME680 sensor:

- Digital interfaces I2C, SPI
- Operating voltage 3-5V
- Dimensions 30 x 14 x 10mm
- Weight 10g

Humidity accuracy

- Response speed 8s
- Tolerance ± 3%
- Hysteresis ≤ 1.5%

Pressure accuracy

- Pressure range 300 - 1100 hPa
- Relative accuracy ± 0.12 hPa
- Absolute accuracy ± 1 hPa

Temperature accuracy

- Working range -40 ° C - 85 ° C
- Full accuracy 0 ° C - 65 ° C

I2C PROTOCOL

An I2C protocol is one of the serial communication protocol that is used for the chip to chip communication. Similar to the I2C protocol, SPI and UART also used for the chip to chip communication.

The I2C is the short form of Inter-Integrated Circuit, is a type of bus, which was designed and developed by Philips in 1980 for inter-chip communication.I2C is adopted by a lot of vendor companies for the chip to chip communication.

It is a multi-master and multi-slave serial communication protocol. It means that we have the freedom to attach multiple IC at a time with the same bus. In I2C protocol, communication is always started by the master and in the case of multi-master, only one master has the ownership of the bus. [2]

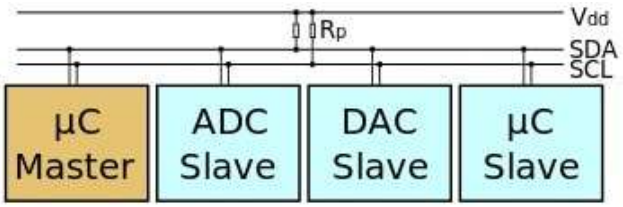


Figure 4: I2C protocol [2]

I2C is a serial communication protocol. It provides the good support to the slow devices, for example, EEPROM, ADC, and RTC etc.

I2c is not only used with the single board but also used with the other external components which have connected with boards through the cables.

I2C is basically two-wire communication protocol. It uses only two wire for the communication. In which one wire is used for the data (SDA) and other wire is used for the clock (SCL).

In I2C, both buses are bidirectional, which means master will able to send and receive the data from the slave. The clock bus is controlled by the master but in some cases slave is also able to suppress the clock signal,[2]

Advantages of I2C communication protocol

There is a lot of advantage of I2C protocol which makes the user to use the I2C protocol in many applications.

- It is the synchronous communication protocol, so no need of precise oscillators for the master and slave.
- It requires only two-wire, one wire for the data (SDA), and other wire for the clock (SCL).
- It provides the flexibility to the user to select the transmission rate as per the requirements.
- In I2C Bus, each device on the bus is independently addressable.
- It follows the master and slave relationships.
- It has the capability to handle multiple masters and multiple slaves on the I2C Bus.
- I2C has some important features like arbitration, clock synchronization, and clock stretching.
- I2C provides ACK/NACK (acknowledgment/ Not-acknowledgement) features that provide help in error handling.[2]

Disadvantages of I2C communication protocol

An I2C protocol has a lot of advantage but beside it, I2C has a few limitations.

- It consumes more power than other serial communication busses due to open-drain topology.
- It is good only for a short distance.
- I2C protocol has some limitation for the number of slaves, the number of the slave depends on the capacitance of the I2C bus.
- It only provides a few limited communication speed like 100 kbit/s,400 kbit/s, etc.
- In I2C, devices can set their communication speed, slower operational devices can delay the operation of faster speed devices.[2]

SPI PROTOCOL:

The serial peripheral interface is four wire-based full-duplex communication protocol these wire generally known as MOSI (master out slave in), MISO (master in slave out), SCL (a serial clock which produces by the master) and SS (slave select line which use to select specific slave during the communication).

Figure 5: SPI Protocol [2]

SPI follows the master and slave architecture and communication is always started by the master. Like I2C it is also a synchronous communication protocol because the clock is shared by the master and slave.

SPI is supported only multi-slave does not support multi-master and slaves are selected by the slave select signal. In SPI during the communication data is shifted out from the master and shifted into the slave vice-versa through the shift register.

Advantages of SPI communication protocol:

- There is no start and stop bits, so the data can be streamed continuously without interruptions. It supports full-duplex.
- No need for precision oscillators in slave devices as it uses a master's clock.
- No complicated slave addressing system like I2C.
- Higher data transfer rate than I2C (almost twice as fast).
- Separate MISO and MOSI lines, so data can be sent and received at the same time.
- Simple software implementation. [2]

Disadvantages of SPI communication protocol

- If there is more than one slave in communication then the wiring will be complex.
- Uses four wires (I2C and UARTs use two).
- No acknowledgment that the data has been successfully received (I2C has this).
- No form of error checking like the parity bit in UART.
- It only allows for a single master. [2]

Overall, SPI is better for high speed and low power applications while I2C is better suited for communication with a large number of peripherals, as well as in situations involving dynamic changing of the master device role among peripherals on the I2C bus. In our project we are making use of I2C protocol.

5.2 ESP8266 MODULE (NODE MCU)

The ESP8266 WiFi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers.

The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area.

The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts.

ESP8266 is a microcontroller with Wi-Fi capability. it requires external flash memory and some antenna to work. Some development boards use basic esp8266 modules and some integrate the chip, flash memory and the antenna on the PCB.

Node MCU is a development board with esp8266 and a firmware with the same name. [3]

This firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for the project and build a firmware tailored according to the needs.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. [3]

The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

Figure 6: ESP8266 Module [3]

ESP8266 module looks like as shown in the figure above. Main advantage is its in-built Wi-Fi module.

ESP-12E Module

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**.**

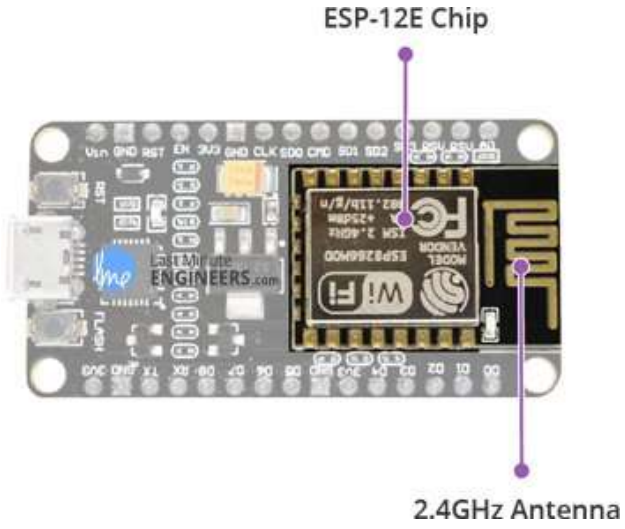


Figure 7: 12E Module

There's also 128 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays.

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 it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8266 NodeMCU even more versatile.

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.

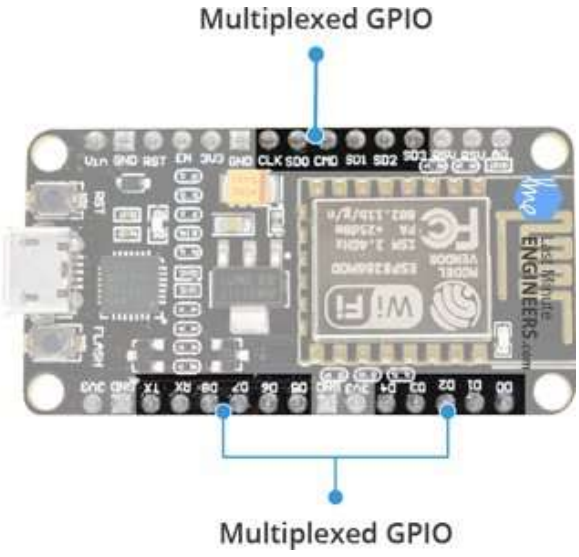


Figure 8: Multiplexed GPIO

Node MCU has a special feature - ESP8266's pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin).Means that a single GPIO pin can act as PWM/UART/SPI.

Onboard switches and 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.

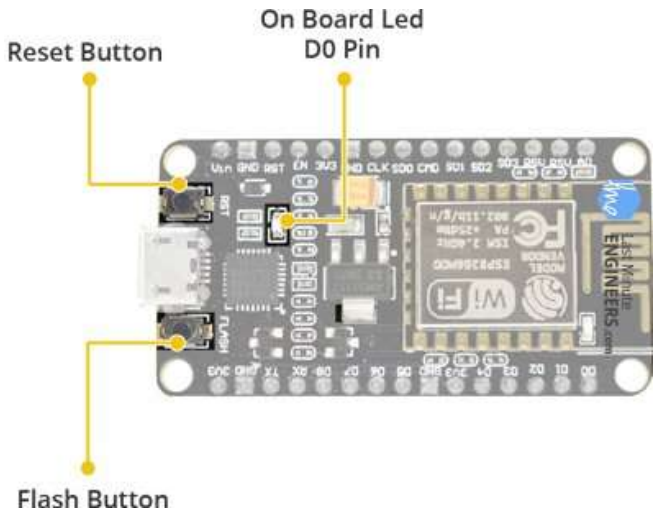


Figure 9: LED indicator and RST button

The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

Serial Communication:

The board includes CP2102 USB-to-UART Bridge Controller from Silicon Labs, which converts USB signal to serial and allows your computer to program and communicate with the ESP8266 chip.

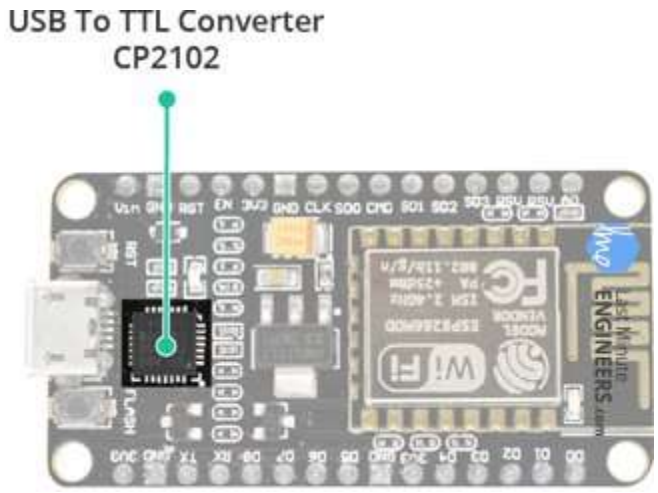


Figure 10: USB to TTL Converter

Let us have a look at the pin diagram of ESP8266 Node MCU module:

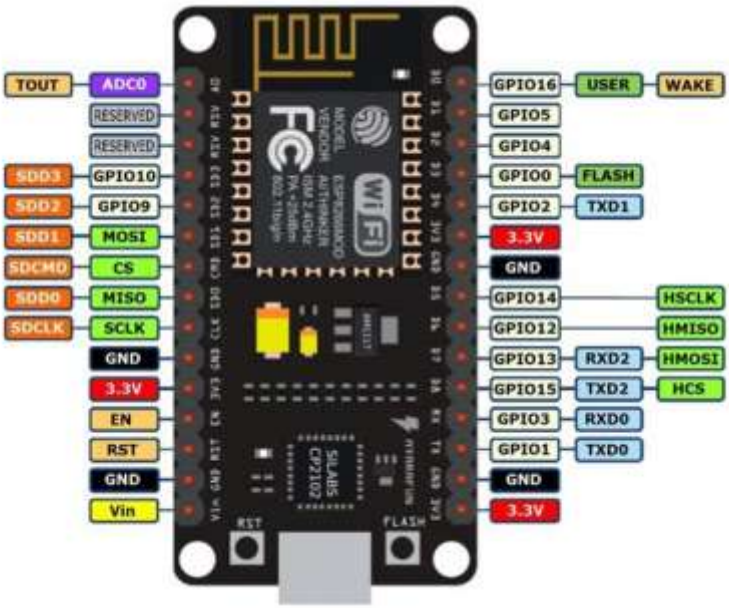


Figure 11: ESP8266 Pin configuration [3]

Power Pins There are four power pins. VIN pin and three 3.3V pins.

- VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin

- 3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components. [4]

GND are the ground pins of NodeMCU/ESP8266

I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO

SDIO Pins NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).

Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

- EN**:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power
  .
- RST**:** RST pin is used to reset the ESP8266 chip.

- WAKE**:** Wake pin is used to wake the chip from deep-sleep. [4]

NodeMCU ESP8266 Specifications & Features

- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna [3]

Applications of Node MCU

- Prototyping of IoT devices
- Low power battery operated applications
- Network projects
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities [3]

5.3 POWER CABLE

Power to the ESP8266 NodeMCU is supplied via the on-board MicroB USB connector. Alternatively, if you have a regulated 5V voltage source, the VIN pin can be used to directly supply the ESP8266 and its peripherals.

5.4 JUMPING WIRES AND BREAD BOARD

All the circuit connections between ESP8266 and BME689 are made on the bread board with the help of connecting wires known as jumping wires.

CHAPTER 6

SOFTWARE DESCRIPTION

Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.

Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users.

Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments.

Arduino is a key tool to learn new things.

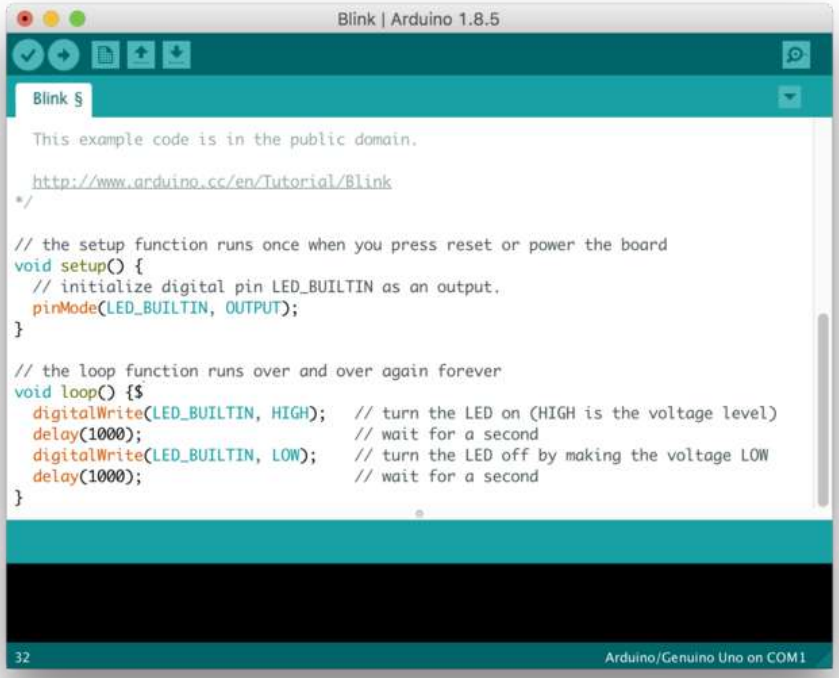


Figure 12: Arduino IDE with simple LED Blink Program

Why Arduino IDE?

- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. [5]

- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.

- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.

- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.

- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.

- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.

- This environment supports both C and C++ languages. [5]

CHAPTER 7

INTERFACING ESP8266 and BME680

Before uploading the code, it is necessary to make the connections between the Micro controller and the sensor.

BME680	ESP8266 Module
GND	GND
VCC	3V3
SCL	D1
SDA	D2

Figure 13: Pin connections

To depict the connection we made use of Fritzing Software. It took use a while to understand it and we successfully made it. The Interfacing is as shown below:

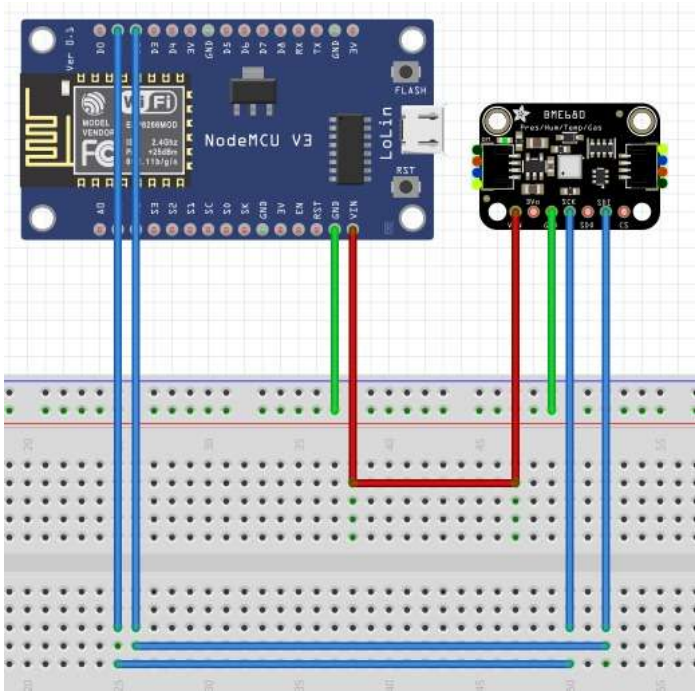


Figure 14: Interfacing ESP8266 and BME680

The controller is given with 3V power supply and the same power is given to the sensor as supply voltage. The ground of the sensor is connected to the ground pin of the controller.

SDA pin of the sensor is given to the D2 pin of the controller and same is triggered in the code as well. SCL pin of the sensor is interfaced with the D1 pin of the controller. The whole circuitry is made to send the data from environment sensor to the controller.

We use the I2C connection for the sensor

After the connections are made it is time to build the code and run the code in Arduino IDE.

But before that we need to import the Adafruit sensor and BME680 libraries to the Arduino IDE.

The Osoyoo Node MCU comes pre-programmed with Lua interpreter, but we don't have to use it! Instead, we can use the Arduino IDE which may be a great starting point for Arduino lovers to familiarize ourselves with the technologies surrounding the IoT.

It must be noted that when we use the Node MCU board with the Arduino IDE, It will write directly to the firmware, erasing the Node MCU firmware.

So, if we want to back to Lua SDK, we use the "flasher" to re-install the firmware.

The Node MCU programming can be as easy as in Arduino, the main difference is the distribution of pins in the Node MCU board.

Following are the steps to be followed to import the libraries to the Arduino IDE:

Step 1: Use the USB cable to connect your NodeMCU to the computer, you will see the blue onboard LED flicker when powered up

.

Step 2: In order to upload code to the ESP8266 and use the serial console, connect any data-capable micro USB cable to ESP8266 IOT Board and the other side to your computer's USB port.

Step 3: Install the ESP8266 board package

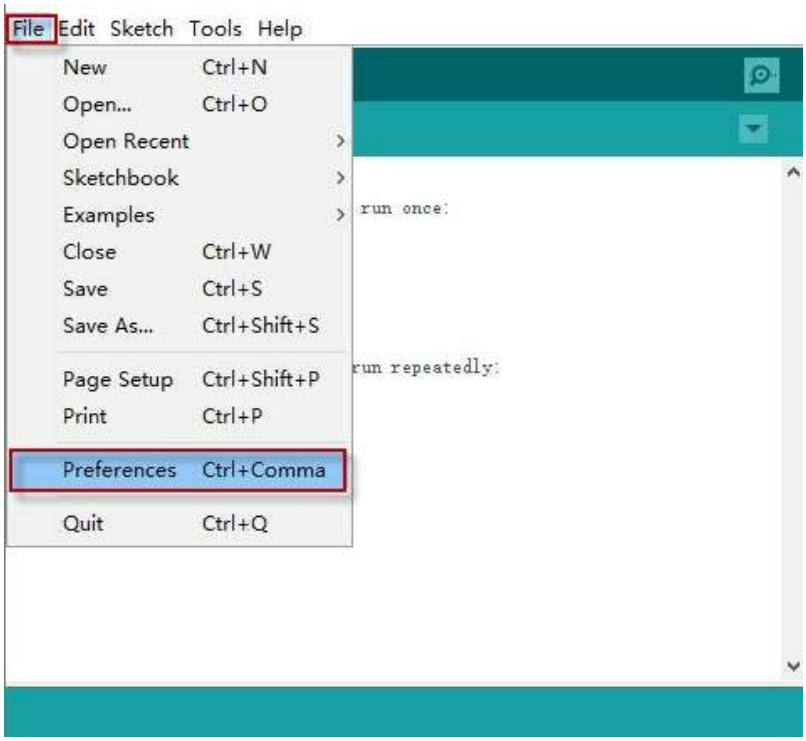


Figure 15: Preferences Tab

Enter http://arduino.esp8266.com/stable/package_esp8266... into Additional Board Manager URLs field in the Arduino v1.6.4+ preferences (Open Arduino IDE–>File–>Preferences–>Settings). Enter the link and click "OK" to save your changes.

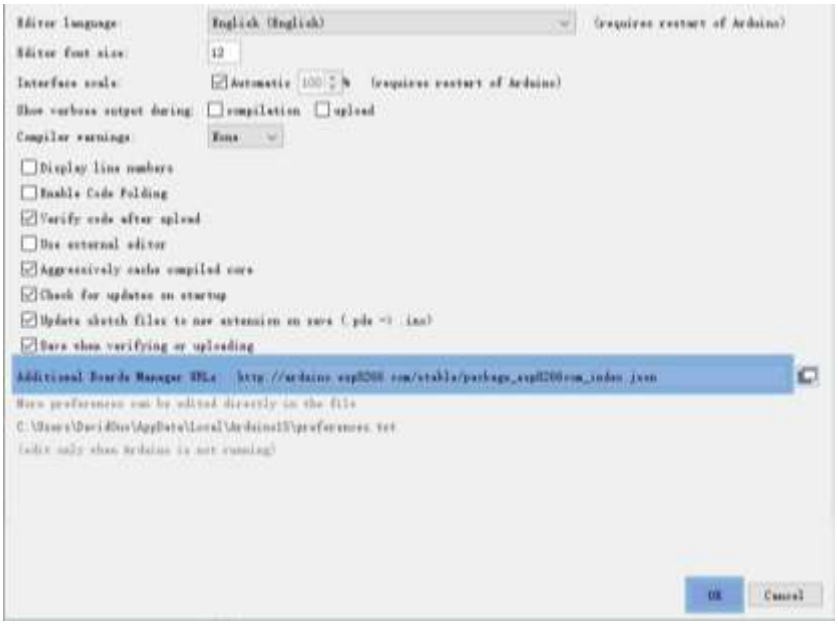


Figure 16: Enter the Link

Next, use the Board Manager to install the ESP8266 package Enter the Boards Manager and find the board type as below: Scroll the Broads Manager screen down to the bottom, you will see A module called "esp8266 by esp8266 Community"

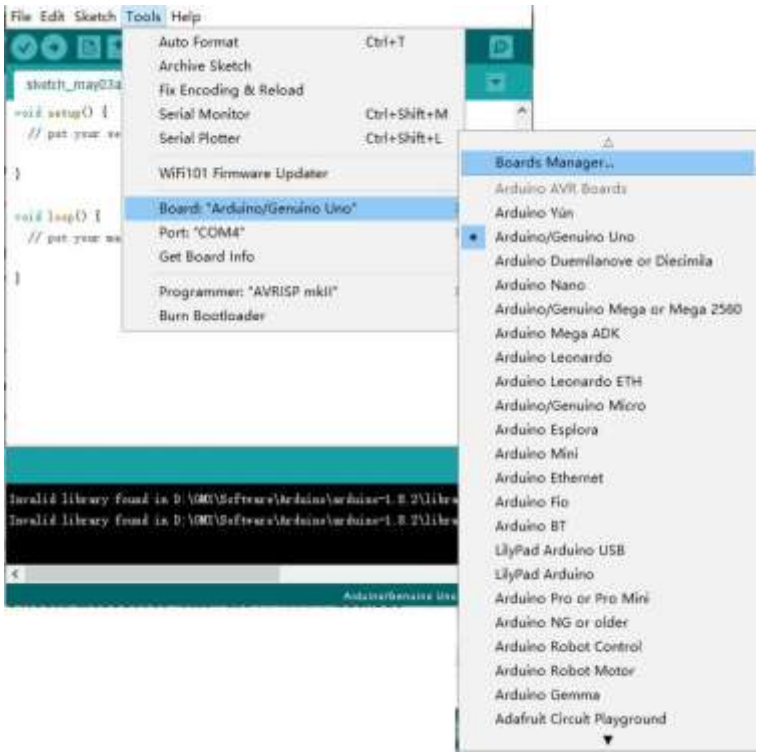


Figure 17: Board manager

Select the latest version and click on Install button. The ESP8266 package has been installed successfully. It is better to close the Arduino IDE and restart it again.

Figure 18: Installed the latest version

The default address for the sensor is 0x77 and we have to make use of it in the code. The code and output will is as shown in the Appendix

We can see from the Appendix that the BME680 sensor is working properly and is collecting the data from the environment.

Now our major task is to extend and use the concept of Internet of Things in our project and send the data to the Cloud and this data can be accessed and controlled by the user with the help of web page or mobile application.

For our prototype we have created a simple web page. For sending and receiving the data from the cloud we have used Amazon Web Services (AWS) IOT Core under certain protocols.

Before going through all that, It is good to have a knowledge about all these terms which will be explained further.

CHAPTER 8

INTERNET OF THINGS (IOT)

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data.

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

It is all possible because of the arrival of super-cheap computer chips and the ubiquity of wireless networks. From something as small as a pill to something as big as an aeroplane is a part of the IoT.

Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate real-time data without involving a human being.

A **Thing** in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analysed or analysed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data. [6]

Figure 19: Simple Model of IOT [6]

The connectivity, networking and communication protocols used with these web enabled devices largely depend on the specific IoT applications deployed.

IoT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic.

Tech analyst company IDC predicts that in total there will be 41.6 billion connected IoT devices by 2025, or "things." It also suggests industrial and automotive equipment represent the largest opportunity of connected "things,", but it also sees strong adoption of smart home and wearable devices in the near term.

Building automation – like connected lighting – will be the fastest growing sector, followed by automotive (connected cars) and healthcare (monitoring of chronic conditions).

IOT benefits to organizations

The internet of things offers several benefits to organizations. Some benefits are industry-specific, and some are applicable across multiple industries. Some of the common benefits of IoT enable businesses to:

- monitor their overall business processes
- improve the customer experience
- save time and money
- enhance employee productivity
- integrate and adapt business models
- make better business decisions
- generate more revenue. [6]

IOT encourages companies to rethink the ways they approach their businesses and gives them the tools to improve their business strategies.

Generally, IOT is most abundant in manufacturing, transportation and utility organizations, making use of sensors and other IOT devices. However, it has also found use cases for organizations within the agriculture, infrastructure and home automation industries, leading some organizations towards Digital transformation.

IOT can benefit farmers in agriculture by making their job easier. Sensors can collect data on rainfall, humidity, temperature and soil content, as well as other factors, that would help automate farming techniques.

The ability to monitor operations surrounding infrastructure is also a factor that IOT can help with. Sensors, for example, could be used to monitor events or changes within structural buildings, bridges and other infrastructure. This brings benefits with it, such as cost saving, saved time, quality-of-life workflow changes and paperless workflow.

A home automation business can utilize IOT to monitor and manipulate mechanical and electrical systems in a building. On a broader scale, smart cities can help citizens reduce waste and energy consumption.

IOT touches every industry, including businesses within healthcare, finance, retail and manufacturing.

The IoT generates vast amounts of data: from sensors attached to machine parts or environment sensors, or the words we shout at our smart speakers. That means the IoT is a significant driver of big-data analytics projects because it allows companies to create vast data sets and analyse them. Giving a manufacturer vast amounts of data about how its components behave in real-world situations can help them to make improvements much more rapidly, while data culled from sensors around a city could help planners make traffic flow more efficiently.

CHAPTER 9

USER INTERFACE

We are making use of HTML, CSS and Java script to design a web page which is the user interface. We are creating a simple web page for the prototype. Let us see what these scripting languages mean briefly.

Hyper Text Mark-up Language (HTML), Cascading Style Sheets (CSS), and JavaScript are the languages that run the web. They're very closely related, but they're also designed for very specific tasks. [7]

- HTML is for adding meaning to raw content by marking it up.
- CSS is for formatting that marked up content.
- JavaScript is for making that content and formatting interactive.

HTML, CSS, and JavaScript are totally different languages, but they all refer to one another in some way. Most websites rely on all three, but the appearance of *every* website is determined by HTML and CSS. [7]

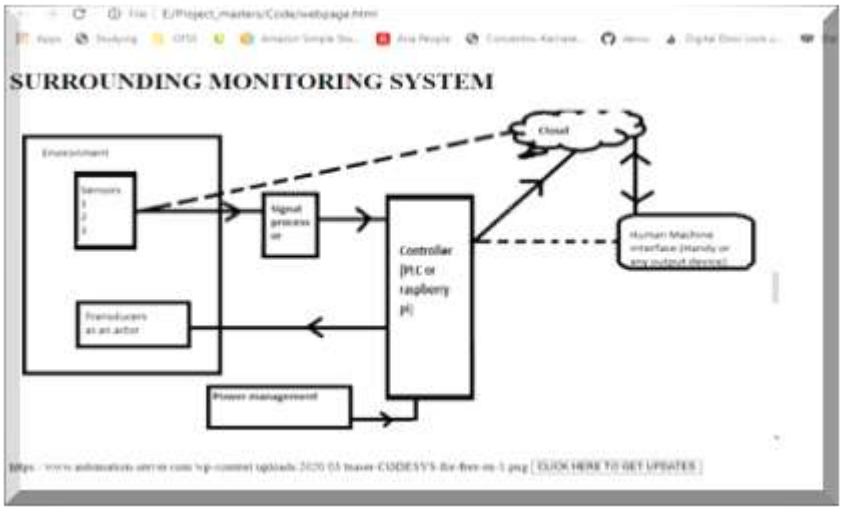


Figure 20: Web Page for the User to Interact with cloud

The user can click on the "CLICK HERE TO UPDATE" button to access the data from the cloud.

The code for the User Interface is as shown in the Appendix

CHAPTER 10

AMAZON WEB SERVICES (AWS)

Amazon Web Services is the market leader in IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) for cloud ecosystems, which can be combined to create a scalable cloud application without worrying about delays related to infrastructure provisioning (compute, storage, and network) and management. [8]

With AWS we can select the specific solutions needed, and faster time to value without sacrificing application performance or user experience.

Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers multitude of businesses in 190 countries around the world.

Types of Clouds

There are three types of clouds − Public, Private, and Hybrid cloud.

1.Public Cloud- In public cloud, the third-party service providers make resources and services available to their customers via Internet. Customer's data and related security is with the service providers' owned infrastructure.

2.Private Cloud- A private cloud also provides almost similar features as public cloud, but the data and services are managed by the organization or by the third party only for the customer's organization. In this type of cloud, major control is over the infrastructure so security related issues are minimized.

3.Hybrid Cloud- A hybrid cloud is the combination of both private and public cloud. The decision to run on private or public cloud usually depends on various parameters like sensitivity of data and applications, industry certifications and required standards, regulations, etc. [8]

Cloud Service Models

There are two types of service models in cloud – IaaS and PaaS

1.IaaS - IaaS stands for Infrastructure as a Service. It provides users with the capability to provision processing, storage, and network connectivity on demand. Using this service model, the customers can develop their own applications on these resources.

2.PaaS - PaaS stands for Platform as a Service. Here, the service provider provides various services like databases, queues, workflow engines, e-mails, etc. to their customers. The customer can then use these components for building their own applications. The services, availability of resources and data backup are handled by the service provider that helps the customers to focus more on their application's functionality.

Security is the major issue in cloud computing. The cloud service providers implement the best security standards and industry certifications, however, storing data and important files on external service providers always bears a risk. [8]

Why AWS?

1. Cost savings - With AWS, companies pay for what they use. There's no upfront cost to build a storage system and no need to estimate usage. AWS customers use what they need and their costs are scaled automatically and accordingly.

2. Scalable and Adaptable - AWS cloud infrastructure is designed to be the most flexible and secured cloud network. It provides scalable and highly adaptable platform that enables customers to deploy applications and data quickly and securely.

3.Security and Reliability - Amazon Web Services is much more secure than a company hosting its own website or storage. AWS currently has dozens of data centers across the globe which are continuously monitored and strictly maintained. The data centers and all the data contained therein are safe from intrusions, and, with Amazon's experience in cloud services, outages and potential attacks can be quickly identified and easily remedied, 24 hours a day.

AWS provides many services like AWS Lambda, Amazon Glacier, Amazon Kinesis etc. Clients can choose any service based on the application and work they are dealing with.

In our case we are making use of AWS IOT Core service.

AWS IOT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices.

AWS IOT Core can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely.

It enables us to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, enables applications to interact with devices even when they are offline.

Amazon Web services Set up:

Our device must be registered with AWS IoT to communicate with the AWS Cloud. To register with the AWS IOT, the following are needed:

1.An AWS IoT policy

The AWS IoT policy grants your device permissions to access AWS IoT resources. It is stored in the AWS Cloud.

2.An AWS IoT thing

An AWS IoT thing allows you to manage your devices in AWS IoT. It is stored in the AWS Cloud.

3.A private key and X.509 certificate

The private key and certificate allow your device to authenticate with AWS IoT.

Steps to be followed for the full setup:

1.Creating a policy

- Browse to the AWS IoT console.
- In the navigation pane, choose Secure, choose Policies, and then choose Create.
- Enter a name to identify your policy.
- In the Add statements section, choose Advanced mode
- Choose Create.

This policy grants all AWS IoT resources access to all AWS IoT actions. This policy is convenient for development and testing purposes.

1.Creating a Thing, Private Key and Certificate for device

- Browse to the AWS IoT console
- In the navigation pane, choose Manage, and then choose Things.
- If you do not have any things registered in your account, the You don't have any things yet page is displayed. If you see this page, choose Register a thing. Otherwise, choose Create.
- On the Creating AWS IoT things page, choose Create a single thing.
- On the Add your device to the thing registry page, enter a name for your thing, and then choose Next.
- On the Add a certificate for your thing page, under One-click certificate creation, choose Create certificate.
- Download your private key and certificate by choosing the Download links for each.
- Choose Activate to activate your certificate. Certificates must be activated prior to use.
- Choose Attach a policy to attach a policy to your certificate that grants your device access to AWS IoT operations.
- Choose the policy you just created, and then choose Register thing**.**

After all these, we attach the policy to the Certificates.

CHAPTER 11
MQTT PROTOCOL

- Message Queuing Telemetry Transport (MQTT) is a lightweight application-layer messaging protocol based on the publish/subscribe (pub/sub) model. HTTP is based on request-response model which is one of the reasons why it does not fulfil the needs of IOT applications.
- In the pub/sub model, multiple clients (sensors) can connect to a central server called a broker and subscribe to topics that they are interested in. Clients can also publish messages to specific topics of their interest through the broker.
- The broker is a common interface for sensor devices to connect to and exchange data. Another important point to note about MQTT is that it utilizes TCP connection on the transport layer for connections between sensors and broker which makes the communication reliable.
- Messages in MQTT are always published on Topics, which basically represents the destination address for that message. A client may subscribe as well as publish to multiple topics. Every client subscribed to a topic receives all the messages published to that topic.
- As a standard practice, topics should follow a hierarchy using a slash (/) as a separator. This allows for logical grouping/arrangement for a network of sensors. [9]

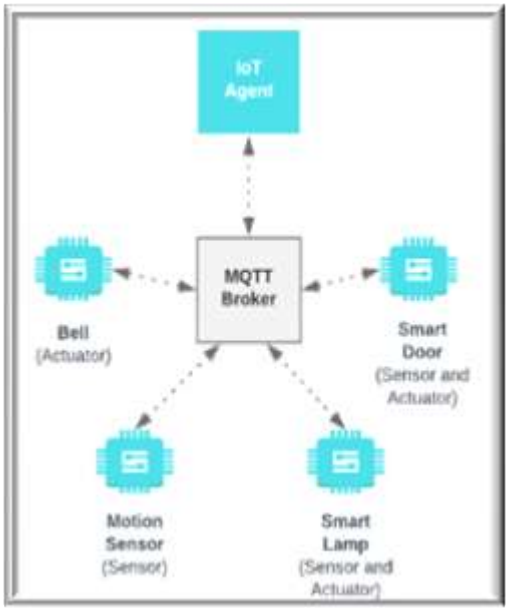


Figure 21: MQTT Model [9]

Other than acting as a router for routing the messages, the Broker has some additional responsibilities like:

- Quality of Service: The Quality of Service (QoS) feature allows the MQTT protocol to provide additional messaging qualities of service depending on the requirement (severity level) of the data or application. The QoS feature ensures that the message in transit is delivered as required by the

service.

- Store and Forward**:** MQTT provides support for storing persistent messages on the broker. When publishing messages, clients may request that the broker persists the message. When a client subscribes to a topic, any persisted message will be sent to the client when this feature is used. Only the most recent persistent message is stored. MQTT broker does not allow these persisted messages to back up inside the server.

- Security**:** MQTT broker will require username and password authentication from clients to connect for security. To ensure privacy of messages in transit, the TCP connection may be encrypted with SSL/TLS. [9]

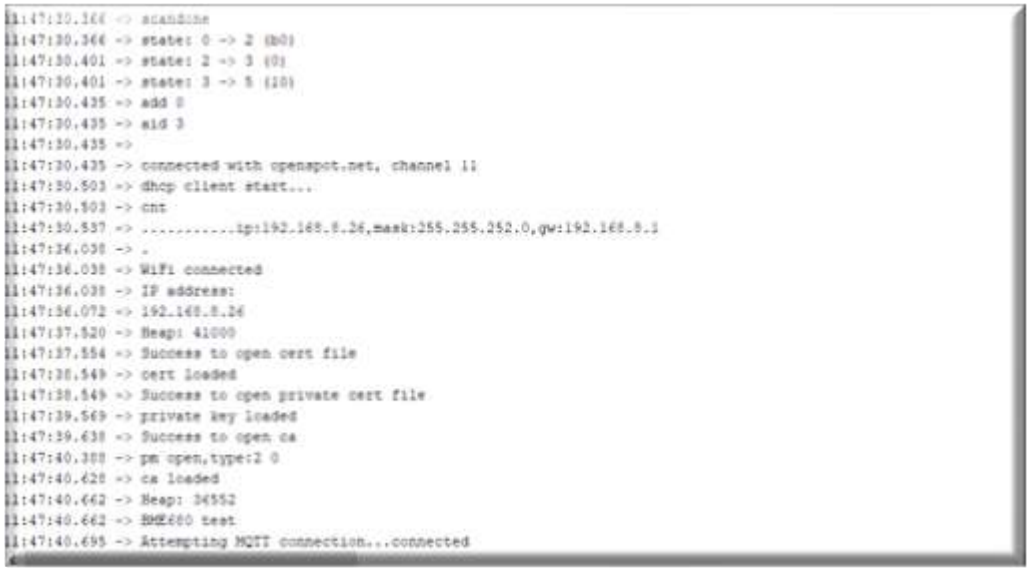
Now that we have a knowledge in all the circuitry and tools that we are using in this project. We make use of AWS IOT core and pass the functions in the Code and run it on the Arduino IDE.

We enter the SSID, Wi-Fi name, Wi-Fi Password and the credentials like Private key which we got when we created the Thing in the code as shown in the appendix

CHAPTER 12

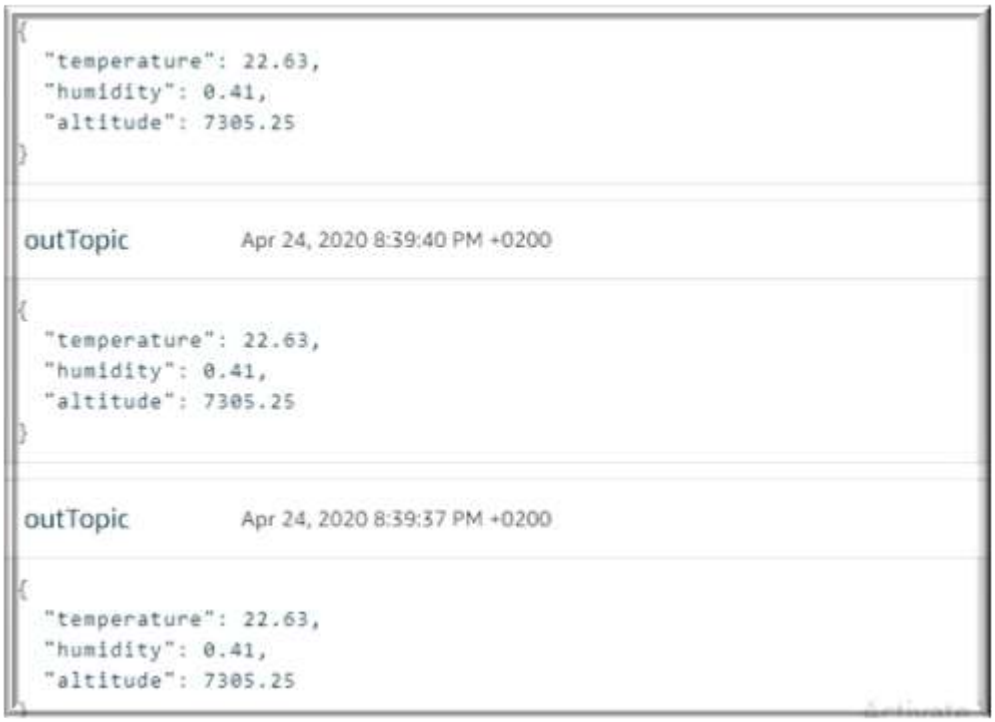
RESULTS

Wi-Fi connection establishment



We can see that the Wi-fi connection is established successfully and it takes in the IP address, Heap area and the Route CA certificate , private key will be loaded.

Output Values:



Publish Window

So we have given the topic name as outTopic and we click on the Publish to Topic button to publish the output.

CHAPTER 13
CONCLUSION

The prototype of Surrounding Monitoring System is made using BME680 Sensor and aESP8266 module. We could measure the physical quantities like Temperature, Humidity and Altitude. These values are then sent to the Cloud using MQTT protocol which can then be accessed by users with the help of User Interface and can thus control these quantities based on the need and application.

CHAPTER 14
WORK PLANNING

14.1 SEQUENCE OF COMPLETION OF PROJECT

1.Ordering and collection of components

2.Interfacing sensors with Micro controller

3.Uploading the data to cloud

4. Monitoring the data constantly for a tenure

5. Do the simulation

6. Comparing the simulated and hardware results

14.2WORK DIVISION AND EXPECTED DATE OF COMPLETION

NUMBER	TASK	EXPECTED TIME (Hours)	DATE
1	SYNOPSIS SUBMISSION	10	05.05.2020
2	INTERFACING OF SENSORS AND CONTROLLER	30	15.05.2020
3	PROGRAMMING AND INTERFACING TESTING	30	25.05.2020
4	CLOUD SET UP	40	30.05.2020
5	AWS IOT WORKING	10	07.06.2020
6	MQTT PROTOCOL CERTIFICATE CONVERSION	10	14.06.2020
7	PROGRAMMING FOR CLOUD DATA TRANSMISSIO	25	20.06.2020
8	MATLAB SIMULATION	30	26.06.2020
9	HARDWARE /SOFTWARE RESULTS COMPARISON	25	28.06.2020
10	REPORT SUBMISSION	20	30.05.2020

APPENDIX

Code1 - Test

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME680 bme; // I2C

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println(F("BME680 test"));

  if (!bme.begin(0x76))
  {
    Serial.println("Could not find a valid BME680 sensor, check wiring!");
    while (1);
  }

  // Set up oversampling and filter initialization
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
}

void loop()
{
  if (! bme.performReading())
  {
    Serial.println("Failed to perform reading :(");
    return;
  }
  Serial.print("Temperature = ");
  Serial.print(bme.temperature);
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bme.pressure / 100.0);
  Serial.println(" hPa");

  Serial.print("Humidity = ");
  Serial.print(bme.humidity);
  Serial.println(" %");

  Serial.print("Gas = ");
  Serial.print(bme.gas_resistance / 1000.0);
  Serial.println(" KOhms");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.println();
  delay(2000);
}
```

Code 2 – User Interface

Code 3- Final Code:

```
#include "FS.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <Wire.h>
#include <SPI.h>
#include <WiFiClientSecure.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME680 bme;

const char* ssid = "openspot.net";


WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org");

const char* AWS_endpoint = "a2zdoystkuvcdf-ats.iot.eu-central-1.amazonaws.com";

void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i = 0; i < length; i++) {
Serial.print((char)payload[i]);
}
Serial.println();
```

```
}
WiFiClientSecure espClient;
PubSubClient client(AWS_endpoint, 8883, callback, espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi() {

delay(10);
// We start by connecting to a WiFi network
espClient.setBufferSizes(512, 512);
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid);

while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

timeClient.begin();
while(!timeClient.update()){
timeClient.forceUpdate();
}

espClient.setX509Time(timeClient.getEpochTime());

}

void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
// Attempt to connect
if (client.connect("ESPthing")) {
Serial.println("connected");
// Once connected, publish an announcement...

// ... and resubscribe
client.subscribe("inTopic");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");

char buf[256];
espClient.getLastSSLError(buf,256);
Serial.print("WiFiClientSecure SSL error: ");
Serial.println(buf);

// Wait 5 seconds before retrying
delay(5000);
}
}
}
```

```
void setup() {

Serial.begin(9600);
Serial.setDebugOutput(true);
setup_wifi();
delay(1000);
if (!SPIFFS.begin()) {
Serial.println("Failed to mount file system");
return;
}

Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());

// Load certificate file
File cert = SPIFFS.open("/cert.der","r"); //replace cert.crt eith your uploaded file name
if (!cert) {
Serial.println("Failed to open cert file");
}
else
Serial.println("Success to open cert file");

delay(1000);

if (espClient.loadCertificate(cert))
Serial.println("cert loaded");
else
Serial.println("cert not loaded");
delay(1000);
// Load private key file
File private_key = SPIFFS.open("/private.der", "r"); //replace private eith your uploaded file name
if (!private_key) {
Serial.println("Failed to open private cert file");
}
else
Serial.println("Success to open private cert file");

delay(1000);

if (espClient.loadPrivateKey(private_key))
Serial.println("private key loaded");
else
Serial.println("private key not loaded");




// Load CA file
File ca = SPIFFS.open("/ca.der", "r"); //replace ca eith your uploaded file name
if (!ca) {
Serial.println("Failed to open ca ");
}
else
Serial.println("Success to open ca");

delay(1000);

if(espClient.loadCACert(ca))
Serial.println("ca loaded");
else
Serial.println("ca failed");

Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());
```

```
while (!Serial);
Serial.println(F("BME680 test"));
if (!bme.begin(0x77))
{
Serial.println("Could not find a valid BME680 sensor, check wiring!");
while (1);
}
bme.setTemperatureOversampling(BME680_OS_8X);
bme.setHumidityOversampling(BME680_OS_2X);
bme.setPressureOversampling(BME680_OS_4X);
bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
bme.setGasHeater(320, 150); //

//test furtwangen ends

}

void loop() {

  if (! bme.performReading())
{
Serial.println("Failed to perform reading :(");
return;
}
Serial.print("Temperature = ");
Serial.print(bme.temperature-15);
Serial.println(" *C");
Serial.print("Pressure = ");
Serial.print(bme.pressure / 100.0);
Serial.println(" hPa");
Serial.print("Humidity = ");
Serial.print(bme.humidity);
Serial.println(" %");

Serial.print("Gas = ");
Serial.print(bme.gas_resistance / 1000.0);
Serial.println(" KOhms");
Serial.print("Approx. Altitude = ");
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(" ft");
Serial.println();
delay(2000);

float t=bme.temperature-15;
float h=0.41;
float a=bme.readAltitude(SEALEVELPRESSURE_HPA);
if (!client.connected()) {
reconnect();
}
client.loop();
String temp = "{\"temperature\":" + String(t) + ", \"humidity\":" + String(h) + ", \"altitude\":" + String(a) +"}";
    int len = temp.length();
    char msg[len+1];
    temp.toCharArray(msg, len+1);

Serial.print("Publish message: ");
Serial.println(msg);
client.publish("outTopic", msg);
Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());
}
```

REFERENCES

[1] Learn.adafruit [Online].

Available at: https://learn.adafruit.com/adafruitbme680humidity-temperature-barometic-pressure-voc-gas/pinouts

[2] Aticleworld [Online].

https://aticleworld.com/difference-between-i2c-and spi/#:~:text=SPI%20does%20not%20have%20a,is%20better%20for%20long%2Ddistance.

[3] NodeMCU ESP8266 [Online].

https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet

[4] NodeMCU ESP8266 Detailed Review [Online].

Available at: https://www.make-it.ca/nodemcu-arduino/nodemcu-details-specifications/

[5] Introduction to Arduino IDE [Online].

Available at: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html

[6] Internet of Things [Online].

Available at: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[7] HTM, CSS and JavaScript [Online]

Available at: https://www.internetingishard.com/html-and-css/introduction/

[8] Tutorialspoint [Online]

Available at: https://www.tutorialspoint.com/amazon_web_services/amazon_web_services_cloud_computing.htm

[9] Introduction to MQTT Protocol for IOT applications [Online]

Available at: https://www.concurrency.com/blog/june-2019/introduction-to-mqtt-protocol-for-iot-applications