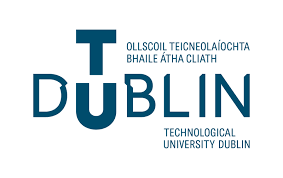
******

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**Bachelor of Engineering (Hons) BE in Electrical & Electronic / Comp & Comm Eng**

**Programme Code: (DT021A)**

**<YEAR 4>**

**Name of Module (The Internet of Things, COMP4000)**

**Name of Lecturer (Kelly, Paula, Core, Michael)**

Design Document

**Student Name** \_Sajjad Ullah, C17344483\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_**

**Title \_**IOT NodeJS and BLE– Assignment 2\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

**DECLARATION**

I hereby certify that the material, which is submitted in this assignment, is entirely my own work and has not been submitted for any academic assessment other than as part fulfilment of the assessment procedures.

Signature of student: ……Sajjad ullah……………….

Date: ……4/12/21…………………………

Contents

[Introduction 4](#_Toc89623689)

[Application Explained 4](#_Toc89623690)

[Architecture Diagram 5](#_Toc89623691)

[Flowchart 6](#_Toc89623692)

[Component Description 7](#_Toc89623693)

[References 9](#_Toc89623694)

[Appendix A: gateway.js code 10](#_Toc89623695)

[Appendix B: Program Results 14](#_Toc89623696)

[Appendix C: micro bit connected to raspberry pi for power 16](#_Toc89623697)

# Introduction

In this assignment we are required to develop a system using a gateway communicating with an end device over Bluetooth Low Energy (BLE).

A BLE central device application will be running the raspberry Pi that requires the use of BLE services and characteristics configured on a BBC Micro Bit BLE peripheral device to measure and report environmental variables.

Objectives:

* The peripheral device should have a BLE service that provides sensor data (such as accelerometer values) to the central device gateway and another BLE service (such as a LED) that can be activated via the gateway.
* The gateway should read the sensor values from the peripheral device, send it to a MQTT broker and then subscribe to the broker for messages to send back to the peripheral to e.g. turn on/off the LED on it.
* The gateway code should be written using NodeJS.
* Communication to the cloud broker should use MQTT as a transport.

The environmental variable we will use to report on is the co2 sensor on the SCD30.

# Application Explained

We will use the co2 sensor on the SCD30 to report carbon dioxide level from the peripheral device to the MQTT broker where they will be published in the topic “co2/pub”.

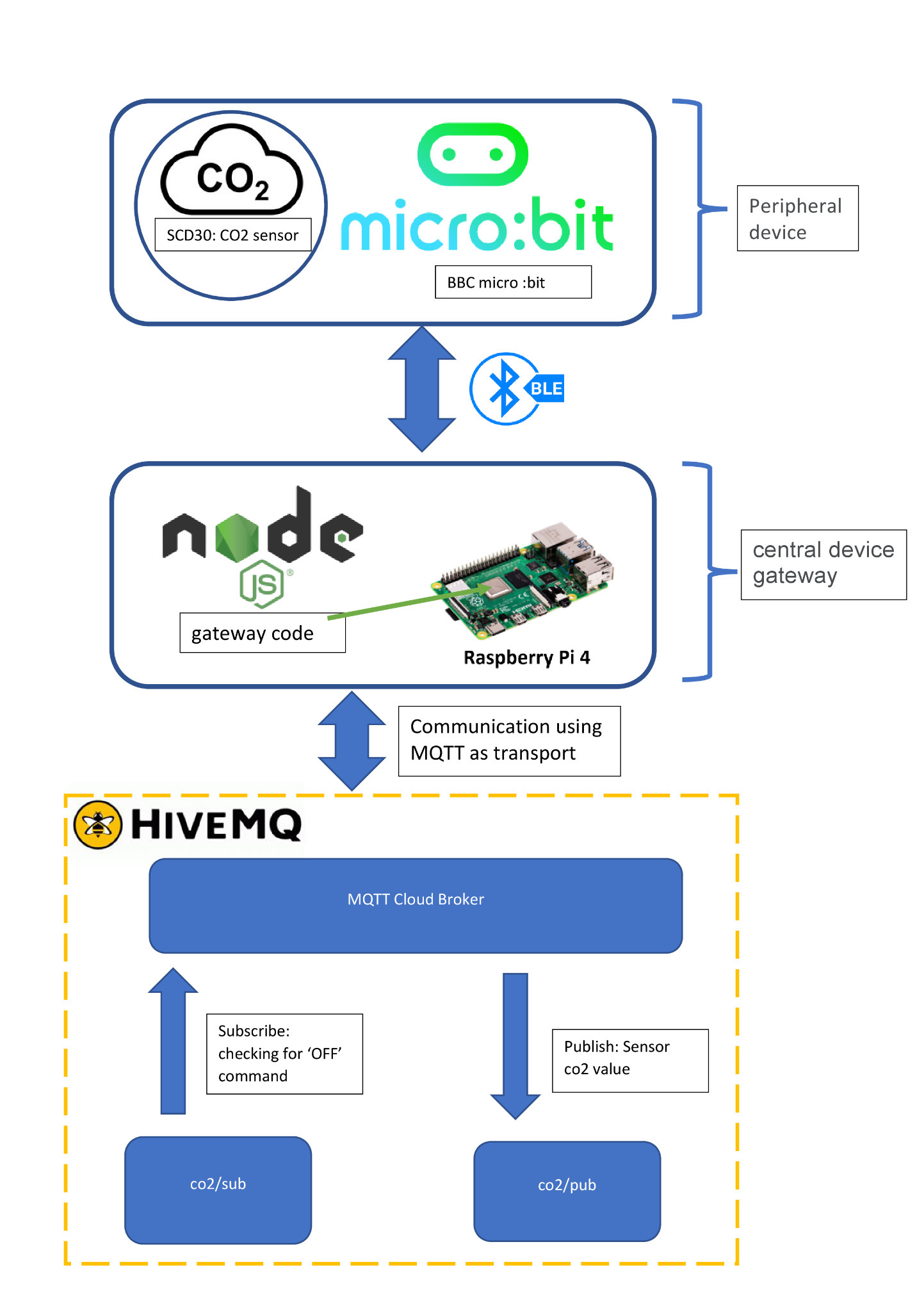
During this time the SCD30 will be connected to the BBC micro bit and using its LED matrix to display a happy emoji face if co2 levels are below a set threshold eg 600 ppm , if above the threshold it will display a unhappy emoji face.

The device will subscribe to the MQTT broker for messages on the topic “co2/sub” and checking for the command “off”, if this is sent then for the purpose of this assignment this will turn off the LED Matrix.

Overall Innovation:

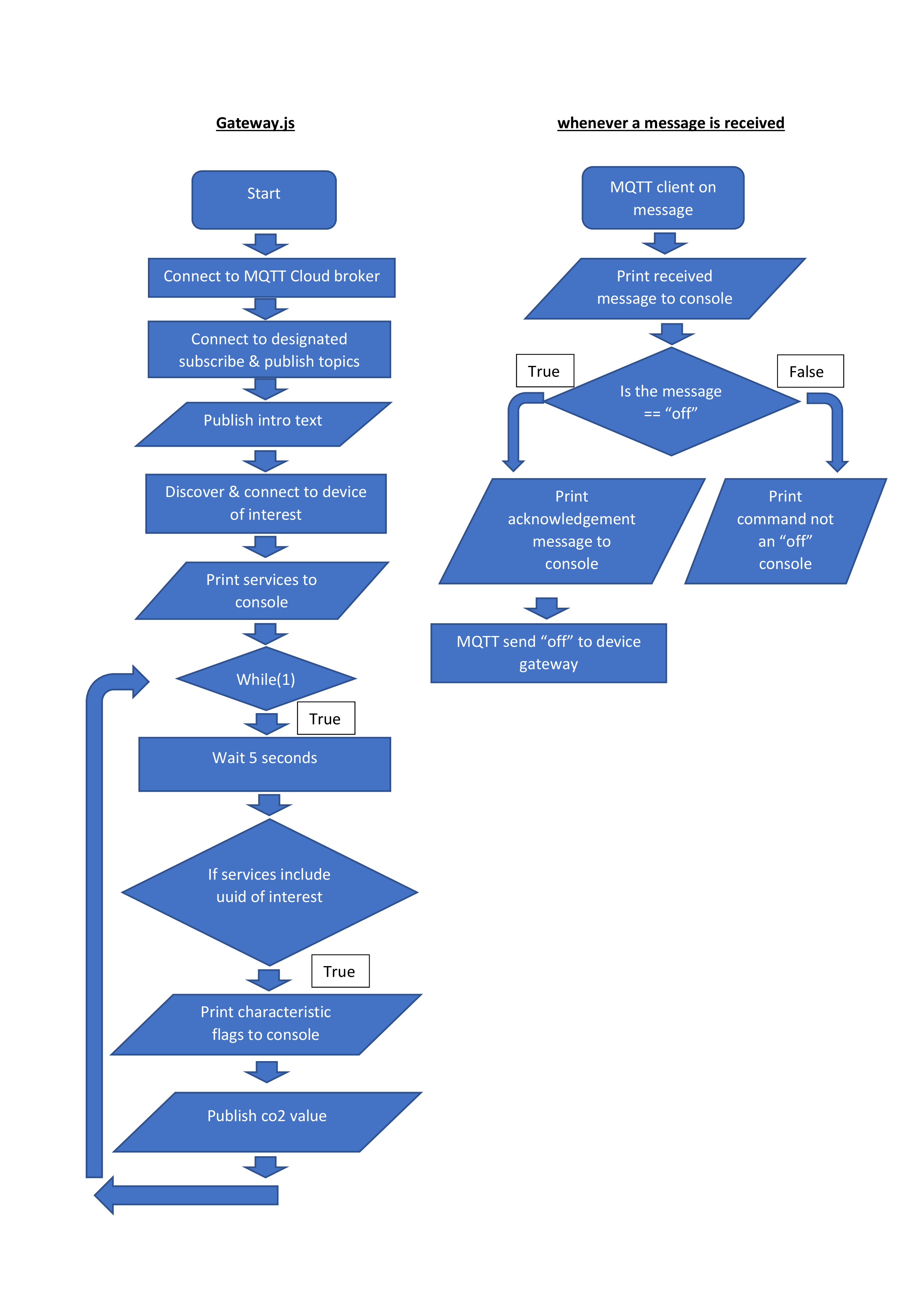
When this is working, we can envision that instead of turning off the LED matrix we could attach a buzzer and turn this on, alerting the occupants of the room the need to take action such as opening window. Or perhaps turn on the air filter.

# Architecture Diagram



# Flowchart

Below is the flowchart for the gateway.js code and flowchart for when a message is received.



# Component Description

**SCD30**: This is a sensor by Sensirion containing CO₂, humidity, and temperature sensing capabilities. The Co2 is based on Nondispersive Infrared (NDIR) technology which enables it to detect 400 to 10000ppm with an accuracy of ±30ppm [1].

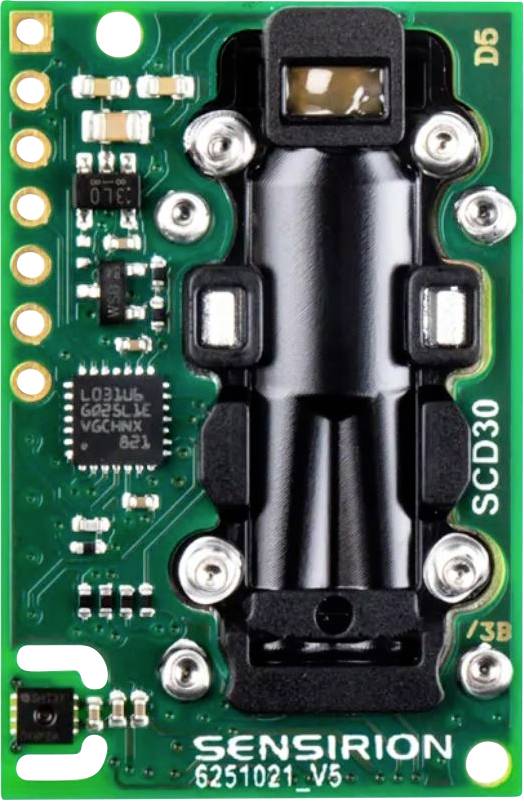


Figure SCD30 Sensor [1]

In figure 2, The gas which in our case is the surrounding air goes into the chamber and will absorb a certain amount of the IR source. By measuring the amount absorbed at the particular wavelength for the specific gas such as CO2, a NDIR detector is able to measure the concentration of CO2[3].

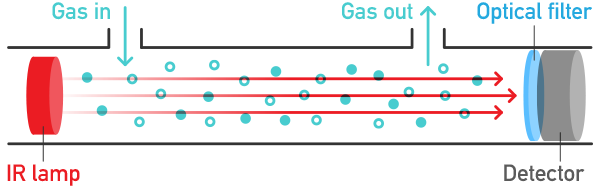


Figure Working principal of NIDAR [2]

**Micro bit** : The microbit is a compact device with onboard an LED light display, buttons, sensors, and multiple input/output (I/O) features which is used for learning digital hardware and software skills or for projects requiring common sensors and I/O.

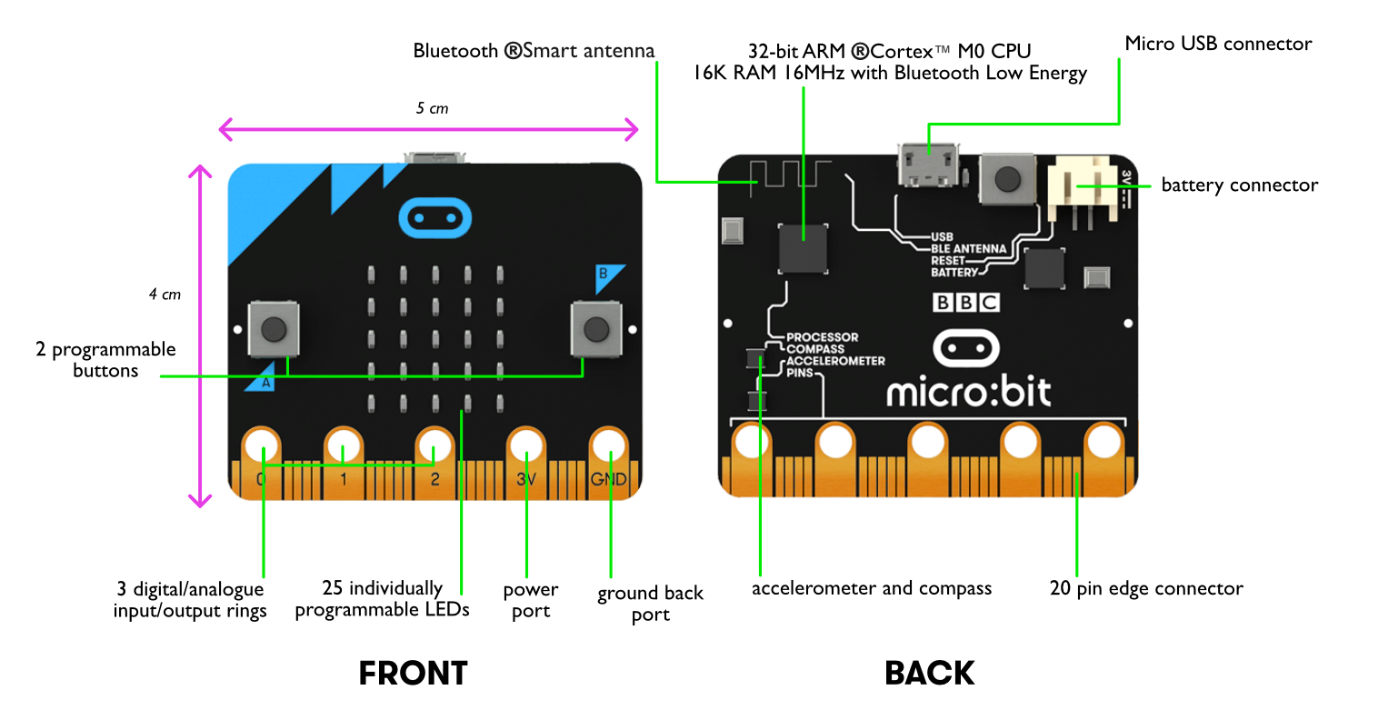


Figure micro-bit front and back [4]

It is clocked at 64 MHz with the version used for this project V2 which has 128 KB of memory RAM. Containing modern radio technology, 2.4 GHz radio antenna as well as Bluetooth 5.1 it can be used in conjunction with any Internet of Things (IOT) applications.

The micro-bit has the following features[5]:

* 25 individually programmable LEDs
* 2 programmable buttons
* Physical connection pins
* Light and temperature sensors
* Motion sensors (accelerometer and compass)
* Wireless Communication, via Radio and Bluetooth
* USB interface

# References

[1]"Sensirion Gas sensor module SCD30", *Google.com*. [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.conrad.com%2Fp%2Fsensirion-gas-sensor-module-scd30-suitable-for-gases-carbon-dioxide-l-x-w-x-h-35-x-23-x-7-mm-2345259&psig=AOvVaw3MJaAzipVFWNTxpvwLzhUP&ust=1638720852954000&source=images&cd=vfe&ved=0CAkQjhxqFwoTCMiTuITFyvQCFQAAAAAdAAAAABAE. [Accessed: 01- Dec- 2021].

[2]"What is the NDIR sensor and why is it preferred over other types of sensors?", *Google.com*. [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Fforum.aranet.com%2Fall-about-aranet4%2Fwhat-is-the-ndir-sensor-and-why-is-it-preferred-over-other-types-of-sensors%2F&psig=AOvVaw2hVObzockDUonuF0WMZymy&ust=1638720638177000&source=images&cd=vfe&ved=0CAkQjhxqFwoTCNjav73EyvQCFQAAAAAdAAAAABAJ. [Accessed: 03- Dec- 2021].

[3]"NDIR Principle", *Cambustion*. [Online]. Available: https://www.cambustion.com/products/knowledgebase/ndir-principle. [Accessed: 02- Dec- 2021].

[4]"OSOYOO Basic Kit for Micro Bit User Guide", *Osoyoo.com*. [Online]. Available: https://osoyoo.com/2018/09/21/osoyoo-basic-kit-for-micro-bit-user-guide/. [Accessed: 04- Dec- 2021].

[5]"BBC MicroBit — Zephyr Project Documentation", *Docs.zephyrproject.org*. [Online]. Available: https://docs.zephyrproject.org/latest/boards/arm/bbc\_microbit/doc/index.html. [Accessed: 04- Dec- 2021].

# Appendix A: gateway.js code

var mqtt = require('mqtt')

var mqttClient = mqtt.connect('mqtt://broker.mqttdashboard.com');

var topicToPublishTo="co2/pub"

var topicToSubscribeTo="co2/sub"

const deviceOfInterest = 'C4:26:87:A4:D6:FA'//device friendly name is C17344483\_SU\_BLE

const serviceOfInterestUuid\_SDC30\_primary\_service = '00000001-0002-0003-0004-000000000000'//uuid of co2 service

const characteristicOfInterestUuid\_co2\_rw = '00000001-0002-0003-0004-000000000001'

//MQTT events and handlers

mqttClient.on('connect', connectCallback); //when a 'connect' event is received call the connectCallback listener function

function connectCallback() {

console.log("\nconnected to cloud MQTT broker"); //print acknowledgement of connection to console

mqttClient.subscribe(topicToSubscribeTo, mqttSubscribeCallback); //subscribe to topic "topicToSubscribeTo" defined at top, call the mqttSubscribeCallback function

mqttClient.publish(topicToPublishTo, 'published values from microbit will appear here', publishCallback);//publish intro text to topic "topicToPublishTo" defined at top, call the publishCallback function

}

function mqttSubscribeCallback(error, granted) { //checking if an error occured when subscribing to topic

if (error) {

console.log("!!!!!!!Error subscribing to topic!!!!!!!");

} else {

console.log("\nsubscribed to and awaiting messages on topic '" + topicToSubscribeTo + "'"); //print acknowledgement text with the topic name of where it will appear. this is helpful to user

}

}

//an async function is a function declared with the async keyword

//enable asynchronous, promise-based behaviour

//this will be used for the messageEventHandler

const messageEventHandler = async (topic, message, packet) => {

try {

console.log("\nReceived message'" + message + "' on topic '" + topic + "'");//print received msg text with the topic name of where it will appear

if (message.toString().toLowerCase() == "off") { //cheking if message from broker is an 'off' (not case sensitive)

console.log("\n+++++++++command received from MQTT broker to turn LED off++++++");//print acknowledgement text if msg == 'off' to console

await charact.writeValue(new Buffer.from("off")) //write the value "off" to the co2 read write uuid

//await charact.write(new Buffer.from("off"), false, writeDataCallback); //(not used)write the value "off" to the co2 read write uuid

} else {

console.log("\n++++++++++command received from MQTT broker was not an OFF command+++++++ ");//print this text if msg is anything else to console

}

}catch(error) {// Handling rejection here

console.log("\n\n\n\n\n\nRejcton occured at messageEventHandler")

}

};

mqttClient.on('message', messageEventHandler);

function writeDataCallback(error, data) { //function to be called checks if data cannot be written to

if (error) {

console.log("error writing data"); //print text to inform user of the error in console

} else {

//disconnect the central device from the peripheral device

console.log("--disconnecting---");

peripheralGlobal.disconnect(disconnectCallback); // disconnect the peripheral

}

}

//this callback function is called when a message has been published to the broker

function publishCallback(error) {

if (error) {

console.log("error publishing data");

} else {

console.log("Message is published to topic '" + topicToPublishTo+ "'");//print the topic name of where it will appear. this is helpful to user

}

}

const main = async() => {//the entry point of the application, using async for promises, asynchronous code

const {createBluetooth}=require('node-ble') //nodejs ble module/library

const { bluetooth, destroy } = createBluetooth()

// get bluetooth adapter

const adapter = await bluetooth.defaultAdapter() //get an available Bluetooth adapter

await adapter.startDiscovery() //using the adapter, start a device discovery session

console.log('discovering')

// look for a specific device

const device = await adapter.waitDevice(deviceOfInterest)

console.log('got device', await device.getAddress()) // await device.getAddress())

const deviceName = await device.getName() //get device name

console.log('got device remote name', deviceName) //print to console the remote name

console.log('got device user friendly name', await device.toString())//print to console the user friendly name i.e. device name

await adapter.stopDiscovery() //stop discovery mode

//connect to the specific device

await device.connect()

console.log("connected to device : " + deviceName)

const gattServer = await device.gatt()

services = await gattServer.services()//getting the gatt services in device

console.log("services are " + services)//print the gatt services to console

while(1){//used to loop indefinitely

await new Promise(resolve => setTimeout(resolve, 5000)).catch();

//;//delay for 5 seconds, to slow down the rate it displays the below code, for readability

if (services.includes(serviceOfInterestUuid\_SDC30\_primary\_service)) { //check if the service has the uuid of the service of interest

console.log('\n---------------------------------------------------------------')

console.log('\nGot the co2, temp , humidity service')

//Primary Service: represents the primary functionality of a device.

const primaryNotifyService = await gattServer.getPrimaryService(serviceOfInterestUuid\_SDC30\_primary\_service)//getting the Primary Service

charact = await primaryNotifyService.getCharacteristic(characteristicOfInterestUuid\_co2\_rw)//save primaryNotifyService as 'charact'

console.log("characteristic flags are : " + await charact.getFlags())//Print the flags of the uuid

cval = await charact.readValue()//saving co2 value to cval

console.log('\nPublishing to mqtt')

mqttClient.publish(topicToPublishTo,"co2 val hex:"+ cval.toString('hex'), publishCallback);//publish to 'topicToPublishTo' (defined at top) with cval in hex, call the publishCallback function

console.log('\nDone Publishing to mqtt')

console.log('\n---------------------------------------------------------------')

}

}

await device.disconnect()

console.log('disconnected')

process.exit()

}

main()

.then()

.catch(console.error)

# Appendix B: Program Results

Graphical user interface, application

Description automatically generatedUsing HIVEMQ, connected and subscribed to the two topics:

Text

Description automatically generatedconnected to BLE device:

Text

Description automatically generatedPublishing co2 values to ‘co2/pub’:

Graphical user interface, text, application, email

Description automatically generatedTyping “keep led matrix on” to the topic ‘co2/sub’

Text

Description automatically generatedResponse at gateway:

Typing “off” to the topic ‘co2/sub’ (using the new messageEventHandler function)

Text

Description automatically generated

Note: LED matrix does indeed turn off

# Appendix C: micro bit connected to raspberry pi for power

A close-up of a car's headlight

Description automatically generated with medium confidence