Samsung Innovation Campus

Diplomado Internet de las cosas

Administración de laboratorios escolares durante la pandemia por COVID-19

Integrantes:

Juan Elier Rosales Rosas
Ingeniería en computación y telecomunicaciones,

Jorge Isur Balderas Ramírez
Ingeniería en computación y telecomunicaciones,

Andrea Chavero Hidalgo
Ingeniería en computación y telecomunicaciones,

Asesor:
Hugo Vargas

(Universidad Autónoma Metropolitana)

2022

Índice general

Indice de figuras II

1. Introducción **2**

1.1. Introducción 2

1.2. Objetivos 2

1.2.1. Objetivo general 2

1.2.2. Objetivos especificos 3

1.3. Alcances 3

1.4. Justificación 3

1.5. Materiales 4

2. Desarrollo del proyecto **5**

2.1. Sensores 5

2.1.1. PN532 5

2.1.2. Micro Servo 10

2.1.3. MLX90614 13

2.1.4. MAX30102 17

2.1.5. DHT11 24

2.2. Código de base de datos 28

2.3. JSON de Node Red 34

3. Conclusiones **63**

3.1. Conclusiones 63

Índice de figuras

2.1. Tabla PN532 6
2.2. Conexión PN532 a Raspberry 6
2.3. Diagrama conexiones servo a Raspberry 10
2.4. Diagrama de conexiones MLX90614 13
2.5. Diagrama de conexiones MAX30102 17
2.6. Diagrama de conexiones DHT11 24

Capítulo 1

Introducción

1.1. Introducción

En este taller encontraras los recursos necesarios para la creación de un sistema de administración de acceso y monitoreo de laboratorios en el cual se usarán un conjunto de sensores los cuales serán leídos mediante ESP32-CAM y Raspberrypi 3. Se pasa una tarjeta RFID por un sensor NFC el cual busca en una base de datos si existe la UID de dicha tarjeta, en caso de encontrar un dato relacionado con el UID de la tarjeta RFID, se activa el monitoreo de síntomas covid el cual se realiza con 2 sensores para monitorear temperatura corporal, ritmo cardiaco y nivel de oxigenación, en caso de no presentar síntomas se detona el comportamiento de un micro servo el cual simula la apertura de una cerradura, además, se incluye la medición de temperatura, humedad y el monitoreo por medio de video streaming dentro del laboratorio.

1.2. Objetivos

1.2.1. Objetivo general

Implementar un sistema basado en internet de las cosas para mejorar la administración en laboratorios escolares.

1.2.2. Objetivos especificos

- Monitorear síntomas covid en los usuarios del laboratorio.
- Tener un control de accesos al laboratorio.
- Automatizar el acceso al laboratorio mediante tarjetas RFID.
- Llevar un control de temperatura y humedad dentro del laboratorio.
- Implementar videostreaming dentro de los laboratorios.

1.3. Alcances

- El proyecto mejorara la experiencia del usuario en los laboratorios.
- Tener un monitoreo y control suficiente en cada laboratorio en el que se implemente el proyecto.
- Poder implementar nuevas funciones como un préstamo automático de recursos de los laboratorios.

1.4. Justificación

El desarrollo tecnológico actual con base en las redes del internet de las cosas nos permitió proponer este proyecto que tiene como fin el monitoreo constante de síntomas COVID, lo cual es una necesidad actual constante en los establecimientos públicos derivado de la pandemia. Se implementó un sistema de apertura automático con tecnología RFID incorporada en las credenciales, con el cual se puede llevar un control de las personas que entran al laboratorio, además, el proyecto permite añadir nuevas funcionalidades como un conteo de los usuarios para no superar un aforo máximo, administrar de recursos dentro del mismo, entre otras funciones.

1.5. Materiales

Materiales		
Dispositivo	Cantidad	Precio individual en mxn
Raspberry	2	1065
ESP32-CAM	3	200
DHT11	1	73
MAX30102	1	95
MLX90614	1	290
PN532	1	180
Micro Servo 9g	1	80
Adaptador 5V	1	62.00
Protoboard	2	88
Jumpers	varios	

Cuadro 1.1: Materiales

Capítulo 2

Desarrollo del proyecto

Los sensores trabajan en conjunto comunicados por MQTT, pero los códigos se realizaron pensando en el uso individual para cada uno de los sensores, por lo tanto, se puede usar todo en conjunto o solo ocupar los sensores de tu conveniencia.

2.1. Sensores

2.1.1. PN532

Armado del circuito

Colocar el sensor en el modo MSU, para esto debemos de mover el dipswitch. En este caso como es modo MSU, colocamos el 1 en OFF y el 2 en OFF. (También nos podemos guiar viendo la tabla del sensor, esta se encuentra a un lado del dipswitch como se puede ver en la siguiente imagen)

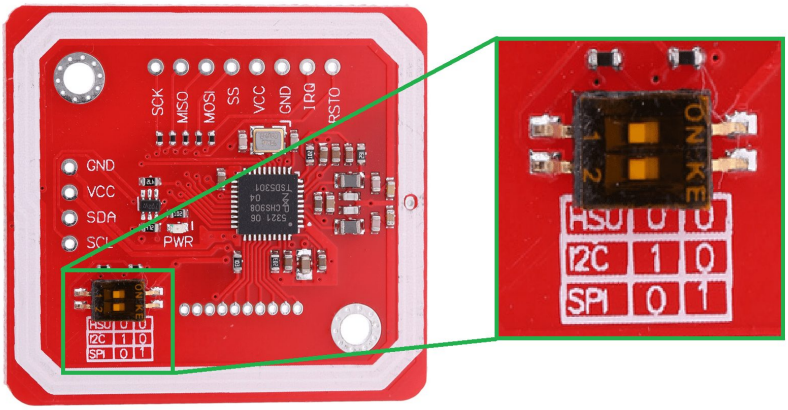

Figura 2.1: Tabla PN532

Conexiones del sensor a Raspberry

Pin en el modulo NFC	Pin Raspberrry
GND	6
VCC	2
SDA	10
SCL	8

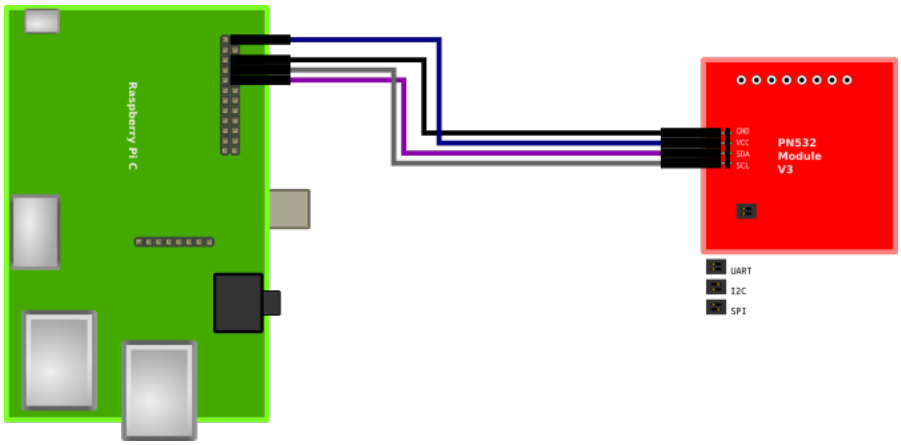

Figura 2.2: Conexión PN532 a Raspberry

Instalación de módulos

Instalamos la librería del modulo con:

```
sudo pip3 install pn532pi
```

Instalamos el módulo paho para poder utilizar MQTT con python.

```
pip3 install paho-mqtt
```

Código

Abrimos Thony Python IDE, pegamos el siguiente código y al final damos clic en Run.

```
"""
Autor: Juan Elier Rosales Rosas
Fecha: 10/02/22
Descripción: Programa para leer un dispositivo NFC mediante
        el sensor PN532.

"""
###################################
#Declaración de modulos
###################################
import paho.mqtt.client as mqtt
import time
import binascii

###################################
#Declaración de bibliotecas
###################################

from pn532pi import Pn532, pn532
from pn532pi import Pn532I2c
from pn532pi import Pn532Spi
from pn532pi import Pn532Hsu

###################################
#Elección de interface a usar.
#Para seleccionar coloca el valor true
###################################
SPI = False
I2C = False
```

```
    HSU = True
    ###################################
    #Comprobación de los valores para la interfaz
    ###################################
    if SPI:
        PN532_SPI = Pn532Spi(Pn532Spi.SS0_GPIO8)
        nfc = Pn532(PN532_SPI)
    # Cuando el número después de #elif es 1, se deberá poner en HSU.
    elif HSU:
        PN532_HSU = Pn532Hsu(0)
        nfc = Pn532(PN532_HSU)

    # Cuando el número después de #if & #elif se configura con
    # 0, se debera poner en modo I2C
    elif I2C:
        PN532_I2C = Pn532I2c(1)
        nfc = Pn532(PN532_I2C)

    ###################################
    #Llamada para cuando el cliente recibe respuesta del servidor
    ###################################
    def on_connect(client, userdata, flags, rc):
        print("Connected with result code "+str(rc))

    client = mqtt.Client()
    client.on_connect = on_connect
    client.connect("187.200.114.169", 1883, 60) #Aquí cambias por la ip o dirección de tu broker

    def setup():
        nfc.begin()

        versiondata = nfc.getFirmwareVersion()
        if not versiondata:
            print("Didn't find PN53x board")
            raise RuntimeError("Didn't find PN53x board")  # halt

        # Confirmación de sensor encontrado
        print("Lector PN5 encontrado {:#x}".format((versiondata >> 24) & 0xFF,
        (versiondata >> 16) & 0xFF, (versiondata >> 8) & 0xFF))
        # Configura el número maximo de intentos para leer alguna tarjeta
        #Esto previene que se espere una tarjeta siempre
        #Está definido por default por el PN532
        nfc.setPassiveActivationRetries(0xFF)

        # Configura para leer los tags RFID
        nfc.SAMConfig()

    def loop():
        # Se espera un tag de tipo ISO14443A (Mifare, etc.). Cuando una es encontrada
        # 'uid'se llena con el UID, y la     longitud del UID indica que
        # si el uid es 4 bytes (Mifare Classic) o 7 bytes (Mifare Ultralight)
```

```
81
82          success, uid = nfc.readPassiveTargetID(pn532.PN532_MIFARE_ISO14443A_106KBPS)
83
84          if (success):
85              print("Tarjeta leida!") #Se leyó con exito la tarjeta
86              uidconv = binascii.b2a_hex(uid);#Regresa de representación hexadecimal a binario
87              uidstr = uidconv.decode();
88              print("UID:",uidstr)
89              #Aquí cambias el cliente/tema con los tuyos
90              client.publish('isur/uid', payload = uidstr, qos=0,retain=False)
91              time.sleep(1)
92              return True
93          else:
94              #Tiempo de espera del PN532
95              print("Timed out waiting for a card")
96              return False
97
98
99
100     if __name__ == '__main__':
101         setup()
102         found = loop()
103         while not found:
104
105         found = loop()
```

Para más información revisar: Repositorio PN532

2.1.2. Micro Servo

Armado del circuito

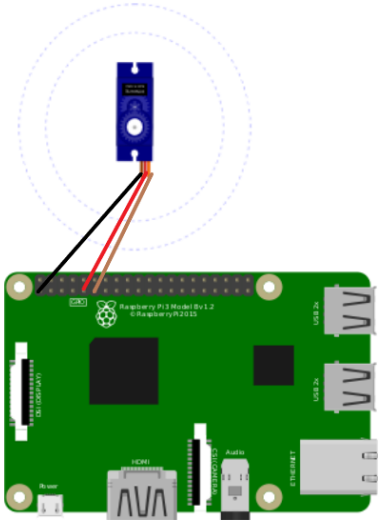

Figura 2.3: Diagrama conexiones servo a Raspberry

Conexiones

Pin del servo	Color del cable	Pin Raspberrry
GND	café o negro	9
VCC	rojo	1
SDA	naranja	11

Instalación de módulos

```
pip3 install paho-mqtt
```

Códigos

Para probar el servo ocuparemos 2 códigos, 1 que es el que se suscribe a MQTT y recibe la instrucción 2 el que publica y envía el mensaje por MQTT.

Código suscripción MQTT

```
#Autor: Juan Elier Rosales Rosas
#Fecha: 11/02/22
#Descripción: Programa para mover un servomotor simulando la apertura de una puerta
####################################
#Declaración de modulos
####################################
import RPi.GPIO as GPIO
import time
import paho.mqtt.client as mqtt
####################################
#Configuración de GPIO
####################################
GPIO.setmode(GPIO.BOARD)

GPIO.setup(11,GPIO.OUT) #Se configura el pin 12 como salida
servo = GPIO.PWM(11, 50) # 12 es el pin y 50 el pulso en Hz
####################################
#Configuración cliente MQTT
####################################
def on_connect(client, userdata, flags, rc):
print("Esperando... ")

    client.subscribe("isur/puerta")

def on_message(client, userdata, msg):
abierto=msg.payload.decode()
acceso(abierto)
####################################
#Función main
####################################
def main():
    client = mqtt.Client()
    client.connect("187.200.112.52", 1883, 60)
    client.on_connect = on_connect
    client.on_message = on_message
    client.loop_forever()
    servo.stop()
    GPIO.cleanup()
    client.disconnect()

def acceso(abierto):
    servo.start(0)
    if abierto=='true':
        print("Puerta abierta")
        servo.ChangeDutyCycle(7)
        time.sleep(0.5)
        servo.ChangeDutyCycle(0)
```

```
48          if abierto=='false':
49              print("Puerta cerrada")
50              servo.ChangeDutyCycle(2)
51              time.sleep(0.5)
52              servo.ChangeDutyCycle(0)
53
54      if __name__ == '__main__':
55          main()
```

Código que publica en MQTT

```
1       import paho.mqtt.client as mqtt
2       global msgstr
3       client = mqtt.Client()
4       #Aquí hay que poner la dirección ip del broker que estemos ocupando
5       client.connect("187.200.112.52", 1883, 60)
6       #El valor entre comillas simples es el tema
7       #ahí se pone tu tema al que quieres mandar el mensaje.
8       #El siguiente valor entre comillas es el valor para actualizar
9       #el estado del servo.
10      client.publish('isur/puerta', "true", qos=1,retain=False)
```

Para más información revisar: Repositorio Mini Servo

2.1.3. MLX90614

Armado del circuito

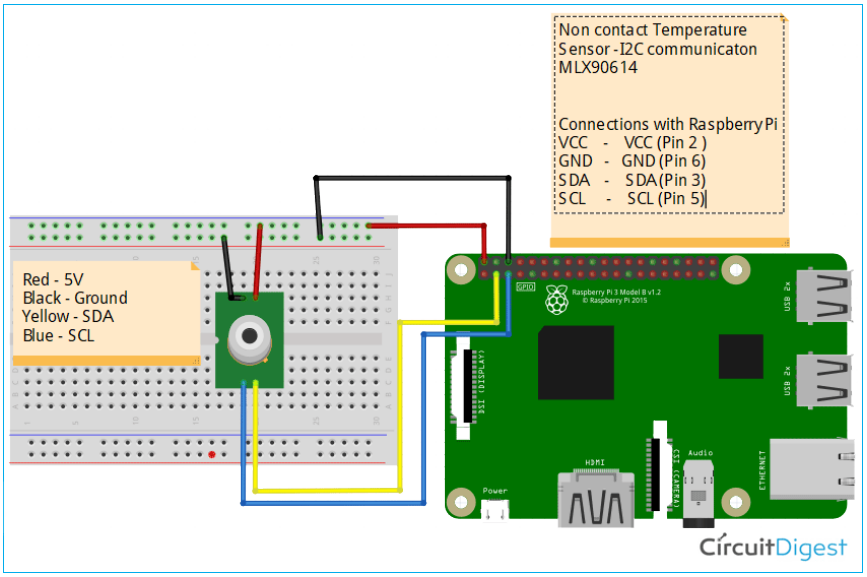

Figura 2.4: Diagrama de conexiones MLX90614

Instalación de módulos

El primer paso es habilitar la interfaz i2c en los ajustes de la raspberry.

Para descargar la librería PyMLX90614 ponemos el siguiente comando el cual descarga el archivo que necesitamos en el directorio donde nos encontremos.

```
wget https://files.pythonhosted.org/packages/67/8a/
443af31ff99cca1e30304dba28a60d3f07d247c8d410822411054e170c9c
/PyMLX90614-0.0.3.tar.gz
```

Para descomprimirlo desde la terminal, ocupamos:

```
tar -xf PyMLX90614-0.0.3.tar.gz
```

Para instalar paho MQTT:

```
pip3 install paho-mqtt
```

Código

```
    #declaracion de modulos
from smbus2 import SMBus #administrar el bus de datos
import paho.mqtt.client as mqtt #manejo de conexiones mqtt
from mlx90614 import MLX90614 #manejo del sensor de temperatura
import time #manejo del tiempo
import RPi.GPIO as gpio #control del acceso GPIO de la raspberry
import random #generacion de numeros aleatorios
import urllib.request as urllib2 #comprobacion de conexion a internet
import datetime as dt
#instancias de objetos
bus = SMBus(1)
sensor = MLX90614(bus, address=0x5A)
#LEDS Y PINES
LED_ENCENDIDO = 12
LED_MQTT = 16
LED_LECTURA = 18
pin_button = 22
#funcion para conectarnos a mqtt
def on_connect(client, userdata, flags, rc):
    for i in range(2):
        gpio.output(LED_MQTT,True)
        gpio.output(LED_MQTT,False)
        gpio.output(LED_MQTT,True)
        gpio.output(LED_MQTT,False)
        gpio.output(LED_MQTT,True)
        gpio.output(LED_MQTT,False)
    print(f"Conectado con codigo de resultado {rc}")
def hayInternet():
    try:
        urllib2.urlopen('http://google.com.mx', timeout=1)
        return True
    except urllib2.URLError as err:
        return False
def standby(LED_ENCENDIDO,LED_MQTT,LED_STANDBY):
    modo = random.randint(0,3)
    if modo == 0:
        gpio.output(LED_ENCENDIDO,False)
        gpio.output(LED_MQTT,False)
        gpio.output(LED_LECTURA,True)
        time.sleep(1)
        gpio.output(LED_ENCENDIDO,False)
        gpio.output(LED_MQTT,True)
        gpio.output(LED_LECTURA,True)
        time.sleep(1)
        gpio.output(LED_ENCENDIDO,True)
        gpio.output(LED_MQTT,True)
        gpio.output(LED_LECTURA,True)
        time.sleep(1)
```

```
49      if modo ==1:
50          gpio.output(LED_ENCENDIDO,True)
51          gpio.output(LED_MQTT,False)
52          gpio.output(LED_LECTURA,False)
53          time.sleep(1)
54          gpio.output(LED_ENCENDIDO,True)
55          gpio.output(LED_MQTT,True)
56          gpio.output(LED_LECTURA,False)
57          time.sleep(1)
58          gpio.output(LED_ENCENDIDO,True)
59          gpio.output(LED_MQTT,True)
60          gpio.output(LED_LECTURA,True)
61          time.sleep(1)
62      if modo ==2:
63          gpio.output(LED_ENCENDIDO,True)
64          gpio.output(LED_MQTT,False)
65          gpio.output(LED_LECTURA,True)
66          time.sleep(1)
67          gpio.output(LED_ENCENDIDO,False)
68          gpio.output(LED_MQTT,True)
69          gpio.output(LED_LECTURA,False)
70          time.sleep(1)
71          gpio.output(LED_ENCENDIDO,True)
72          gpio.output(LED_MQTT,False)
73          gpio.output(LED_LECTURA,True)
74          time.sleep(1)
75      if modo ==3:
76          gpio.output(LED_ENCENDIDO,False)
77          gpio.output(LED_MQTT,True)
78          gpio.output(LED_LECTURA,True)
79          time.sleep(1)
80          gpio.output(LED_ENCENDIDO,False)
81          gpio.output(LED_MQTT,False)
82          gpio.output(LED_LECTURA,True)
83          time.sleep(1)
84          gpio.output(LED_ENCENDIDO,True)
85          gpio.output(LED_MQTT,True)
86          gpio.output(LED_LECTURA,False)
87          time.sleep(1)
88  def parpadeo(LED):
89      gpio.output(LED,True)
90      gpio.output(LED,False)
91      gpio.output(LED,False)
92      time.sleep(1)
93  def mostrarHora():
94      hora = dt.datetime.now().hour
95      minutos = dt.datetime.now().minute
96      dia = dt.datetime.now().day
97      mes = dt.datetime.now().month
98      year = dt.datetime.now().year
99      if minutos<10:
```

```
        minutos = '0'+str(minutos)
    print(f"{dia}/{mes}/{year} a las {hora}:{minutos}")
#programa principal
try:
    gpio.setwarnings(False)
    gpio.setmode(gpio.BOARD)
    gpio.setup(pin_button,gpio.IN,pull_up_down=gpio.PUD_DOWN)
    gpio.setup(LED_ENCENDIDO,gpio.OUT)
    gpio.setup(LED_MQTT,gpio.OUT)
    gpio.setup(LED_LECTURA,gpio.OUT)
    while(hayInternet()==False):
        print("No se encuentra una conexion a Internet.")
        gpio.output(LED_ENCENDIDO,True)
        gpio.output(LED_ENCENDIDO,False)
        print("Reintentando en 5 segundos...")
        time.sleep(5)
    print("Conectado a Internet.")
    gpio.output(LED_ENCENDIDO,True)
    client = mqtt.Client()
    client.on_connect = on_connect
    client.connect("192.168.1.78", 1883, 60)
    while(1):
        if(gpio.input(pin_button)==gpio.HIGH):
            print("Iniciando lectura:\n")
            parpadeo(LED_LECTURA)
            print("Temperatura ambiente:"+str(sensor.get_ambient()))
            parpadeo(LED_LECTURA)
            print("temperatura objeto:"+str(sensor.get_object_1()))
            parpadeo(LED_LECTURA)
            client.publish('isur/temperatura_salon', payload = sensor.get_ambient(),
            qos=0,retain=False)
            parpadeo(LED_MQTT)
            client.publish('isur/temp',payload = sensor.get_object_1(),qos=0,retain=False)
            parpadeo(LED_MQTT)
            time.sleep(5)
        else:
            mostrarHora()
            print("Presione el boton durante 3 segundos para iniciar la lectura.")
            standby(LED_ENCENDIDO,LED_MQTT,LED_LECTURA)
except KeyboardInterrupt:
    print("Finalizando programa.")
    print(f"Apagando leds GPIO{LED_MQTT},GPIO{LED_ENCENDIDO},GPIO{LED_LECTURA}")
    gpio.output(LED_ENCENDIDO,False)
    gpio.output(LED_MQTT,False)
    gpio.output(LED_LECTURA,False)
    bus.close()
```

Para más información revisar : Repositorio MLX90614

2.1.4. MAX30102

Armado del circuito

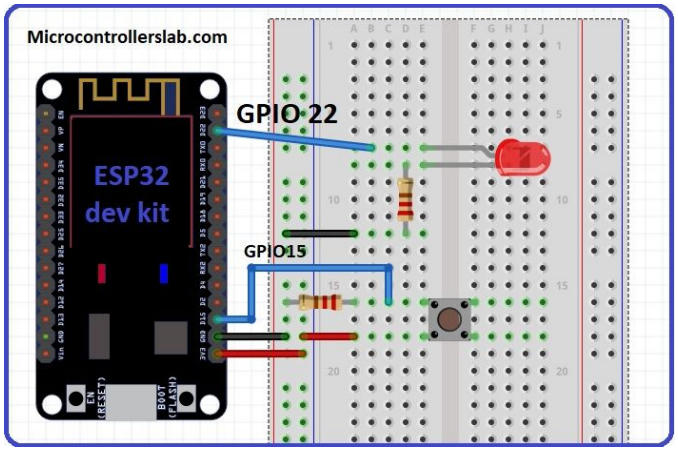

Figura 2.5: Diagrama de conexiones MAX30102

Código

```
/*
AUTOR: JORGE ISUR BALDERAS RAMÍREZ
FECHA: 07-12-2021
DESCRIPCIÓN: PROGRAMA PARA MEDIR SINTOMAS DE COVID MEDIANTE EL SENSOR
             MAX30102
CONEXIONES EN HARDWARE(ESP32CAM)
-5V = 5V (3.3V también funciona)
-GND = GND
-SDA = GPIO15
-SCL = GPIO14
-INT = Not connected

EL MAX30102 SOPORTA I2C LÓGICA EN 5V Y 3.3V, SE RECOMIENDA ENERGIZAR CON 5V
A FIN DE TENER MAYOR PRECISIÓN EN LOS DATOS.
*/
/*****************************************************************************
DECLARACIÓN DE BIBLIOTECAS
*****************************************************************************/
#include <Wire.h> //BIBLIOTECA PARA EL MANEJO DE I2C
#include "MAX30105.h" //BIBLIOTECA PARA EL CONTROL DEL MAX3010X
#include "spo2_algorithm.h" //BIBLIOTECA PARA EL ALGORITMO QUE CÁLCULA OXIGENO EN SANGRE
#include <WiFi.h>  // Biblioteca para el control de WiFi
#include <PubSubClient.h> //Biblioteca para conexion MQTT
/*****************************************************************************
DECLARACIÓN DE INSTANCIA DEL SENSOR
```

```
/*****************************************************************************/
MAX30105 particleSensor;
/*****************************************************************************
DATOS DEL WIFI
*****************************************************************************/
const char* ssid = "INFINITUM3033_2.4";  // Aquí debes poner el nombre de tu red
const char* password = "yYYmteq554";  // Aquí debes poner la contraseña de tu red
/*****************************************************************************
DATOS DEL BROKER MQTT
*****************************************************************************/
const char* mqtt_server = "187.200.114.169"; // Si estas en una red local, coloca la IP
//asignada, en caso contrario, coloca la IP publica
IPAddress server(187,200,218,12);
/*****************************************************************************
OBJETOS
*****************************************************************************/
WiFiClient esp32Client; // Este objeto maneja los datos de conexion WiFi
PubSubClient client(esp32Client); // Este objeto maneja los datos de conexion al broker

#define MAX_BRIGHTNESS 255

#if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
uint16_t irBuffer[100]; //infrared LED sensor data
uint16_t redBuffer[100];  //red LED sensor data
#else
uint32_t irBuffer[100]; //infrared LED sensor data
uint32_t redBuffer[100];  //red LED sensor data
#endif

int32_t bufferLength; //LONGITUD DE LOS DATOS
int32_t spo2; //VALOR DEL OXIGENO EN SANGRE
int8_t validSPO2; //INDICADOR PARA MOSTRAR SI EL VALOR DEL SPO2 ES VALIDO
int32_t heartRate; //VALOR DE PULSO CARDIACO
int8_t validHeartRate; //INDICADOR PARA MOSTRAR SI EL VALOR DEL PULSO CARDIACO ES VALIDO
/*****************************************************************************
LEDS y PINES
*****************************************************************************/
#define LED_OK 12
#define LED_CONEXION_WIFI 13
#define LED_MQTT 2
byte readLED = 33; //PARPADEA CON CADA LECTURA DEL SENSOR
#define push_button 4

void setup()
{
  Serial.begin(115200); // INICIAR LA COMUNICACIÓN SERIAL
  Serial.println("Iniciando...\n");
  Serial.println("LECTOR DE SINTOMAS COVID.\n");
  pinMode(readLED, OUTPUT);
  pinMode(LED_OK, OUTPUT);
  pinMode(LED_CONEXION_WIFI, OUTPUT);
```

```
77    pinMode(LED_MQTT,OUTPUT);
78    pinMode(push_button,INPUT);
79    digitalWrite(LED_OK,HIGH);//EL DISPOSITIVO SE ENCENDIO
80    Serial.print("Conectar a ");
81    Serial.println(ssid);
82
83    WiFi.begin(ssid, password); // Esta es la función que realiz la conexión a WiFi
84
85    while (WiFi.status() != WL_CONNECTED) { // Este bucle espera a que se realice la conexión
86      digitalWrite(readLED, HIGH);
87      digitalWrite(LED_CONEXION_WIFI, HIGH);
88      delay(500); //dado que es de suma importancia esperar a la conexión, debe usarse espera
89      //bloqueante
90      digitalWrite(readLED, LOW);
91      digitalWrite(LED_CONEXION_WIFI,LOW);
92      Serial.print(".");  // Indicador de progreso
93      delay (5);
94    }
95
96    // Cuando se haya logrado la conexión, el programa avanzará, por lo tanto, puede informarse
97    //lo siguiente
98    Serial.println();
99    Serial.println("WiFi conectado");
100   Serial.println("Direccion IP: ");
101   Serial.println(WiFi.localIP());
102
103   // Si se logro la conexión, encender led
104   if (WiFi.status () > 0){
105     digitalWrite(readLED, LOW);
106     digitalWrite(LED_CONEXION_WIFI, HIGH);
107   }
108   delay (1000); // Esta espera es solo una formalidad antes de iniciar la comunicación con el
109   //broker
110   // Conexión con el broker MQTT
111   client.setServer(server, 1883); // Conectarse a la IP del broker en el puerto indicado
112   delay(1500);  // Esta espera es preventiva, espera a la conexión para no perder información
113
114   // Initialize sensor
115   Wire.begin(15,14);
116   if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port, 400kHz speed
117   {
118     Serial.println(F("MAX3010X NO ENCONTRADO. POR FAVOR REVISE EL CABLEADO O LA ALIMENTACIÓN.\n"));
119     while (1);
120   }
121 }
122
123 void loop()
124 {
125   //Verificar siempre que haya conexión al broker
126   Serial.println("Conectando a broker MQTT...");
127   if (!client.connected()) {
```

```
    reconnect();  // En caso de que no haya conexión, ejecutar la función de reconexión
    // definida despues del void setup ()
  }// fin del if (!client.connected())
  Serial.println("Conectado a:");
  Serial.println(server);
  client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante
  //las funciones necesarias para la comunicación con el broker
  digitalWrite(LED_MQTT,HIGH);
  digitalWrite(LED_OK,HIGH);
  int push_button_state = digitalRead(push_button);
  if(push_button_state==HIGH){
    Serial.println(F("Coloque el dedo en el sensor y aplique presión.\n"));
    delay(2000);
    byte ledBrightness = 60; //Options: 0=Off to 255=50mA
    byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
    byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
    byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
    int pulseWidth = 411; //Options: 69, 118, 215, 411
    int adcRange = 4096; //Options: 2048, 4096, 8192, 16384
    //Configure sensor with these settings
    particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange);
    particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.
    bufferLength = 100; //buffer length of 100 stores 4 seconds of samples running at 25sps
    //read the first 100 samples, and determine the signal range
    for (byte i = 0 ; i < bufferLength ; i++){
      while (particleSensor.available() == false){ //do we have new data?
        particleSensor.check(); //Check the sensor for new data
      }
      redBuffer[i] = particleSensor.getRed();
      irBuffer[i] = particleSensor.getIR();
      particleSensor.nextSample(); //We're finished with this sample so move to next sample

      Serial.print(F("red="));
      Serial.print(redBuffer[i], DEC);
      Serial.print(F(", ir="));
      Serial.println(irBuffer[i], DEC);
    }

    //CALCULAR EL PULSO CARDIACO Y OXIGENACIÓN DESPUES DE 100 MUESTRAS(PRIMEROS 4 SEGUNDOS)
    maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
    &validSPO2, &heartRate, &validHeartRate);
    //DESECHANDO LOS PRIMEROS 25 SETS DE MUESTRAS EN MEMORIA Y RECORRIENDO
    //LAS ULTIMAS 75 MUESTRAS AL PRINCIPIO.
    for (byte i = 25; i < 100; i++){
      redBuffer[i - 25] = redBuffer[i];
      irBuffer[i - 25] = irBuffer[i];
    }
    //take 25 sets of samples before calculating the heart rate.
    for (byte i = 75; i < 100; i++)
    {
      while (particleSensor.available() == false){ //¿TENEMOS NUEVAS LECTURAS?
```

```
        particleSensor.check(); //MONITOREA EL SENSOR POR NUEVOS DATOS.
      }
      digitalWrite(readLED, !digitalRead(readLED)); //Blink onboard LED with every data read

      redBuffer[i] = particleSensor.getRed();
      irBuffer[i] = particleSensor.getIR();
      particleSensor.nextSample(); //We're finished with this sample so move to next sample
      Serial.print(F("red="));
      Serial.print(redBuffer[i], DEC);
      Serial.print(F(", ir="));
      Serial.print(irBuffer[i], DEC);
    }
    //After gathering 25 new samples recalculate HR and SPO2
    maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
    &validSPO2, &heartRate, &validHeartRate);
    //send samples and calculation result to terminal program through UART
      Serial.print(F(", PULSO CARDIACO=\n"));
      Serial.print(heartRate, DEC);
      char pulso[10];
      itoa(heartRate,pulso,10);//CONVERTIR INT-->CHAR[] PARA ENVIAR POR MQTT

      //client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante
      //las funciones necesarias para la comunicación con el broker
      //client.publish("isur/pulso",pulso);
      if(!client.publish("isur/pulso",pulso)){
        Serial.print("No sé pudo publicar en isur/pulso.\n");
        Serial.print("Error:");
        Serial.print(client.state()); // Muestra el codigo de error
        digitalWrite(LED_OK,HIGH);
        delay(1000);
        digitalWrite(LED_OK,LOW);
        delay(1000);
        digitalWrite(LED_OK,HIGH);
        delay(1000);
        digitalWrite(LED_OK,LOW);
        delay(2000);
      }
      else{
        Serial.print("Mensaje publicado\n");
        digitalWrite(LED_MQTT,HIGH);
        delay(1000);
        digitalWrite(LED_MQTT,LOW);
        delay(1000);
        digitalWrite(LED_MQTT,HIGH);
        delay(1000);
        digitalWrite(LED_MQTT,LOW);
        delay(2000);
      }
      Serial.print(F(", PULSO CARDIACO VALIDO=\n"));
      Serial.print(validHeartRate, DEC);

```

```
      Serial.print(F(", OXIGENO EN SANGRE=\n"));
      Serial.print(spo2, DEC);
      char oxigeno[10];
      itoa(spo2,oxigeno,10);//CONVERTIR INT-->CHAR[] PARA ENVIAR POR MQTT
      //client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante
      //las funciones necesarias para la comunicación con el broker
      //client.publish("isur/oxigeno",oxigeno);
      if(!client.publish("isur/oxigeno",oxigeno)){
        Serial.print("No se pudo publicar en isur/oxigeno.\n");
        Serial.print("Error:");
        Serial.print(client.state()); // Muestra el codigo de error
        digitalWrite(LED_OK,HIGH);
        delay(1000);
        digitalWrite(LED_OK,LOW);
        delay(1000);
        digitalWrite(LED_OK,HIGH);
        delay(1000);
        digitalWrite(LED_OK,LOW);
        delay(2000);
      }
      else{
        Serial.print("Mensaje publicado\n");
        digitalWrite(LED_MQTT,HIGH);
        delay(1000);
        digitalWrite(LED_MQTT,LOW);
        delay(1000);
        digitalWrite(LED_MQTT,HIGH);
        delay(1000);
        digitalWrite(LED_MQTT,LOW);
        delay(2000);
      }
      Serial.print(F(", OXIGENO EN SANGRE VALIDO=\n"));
      Serial.println(validSPO2, DEC);
      delay(1500);
  }//FIN IF_BUTTON
  else{
    Serial.println("MODO STANDBY");
    Serial.println("Presione el boton por 3 segundos para tomar las medidas");
    digitalWrite(LED_OK,HIGH);
    digitalWrite(LED_CONEXION_WIFI,LOW);
    digitalWrite(LED_MQTT,LOW);
    delay(1000);
    digitalWrite(LED_OK,HIGH);
    digitalWrite(LED_CONEXION_WIFI,HIGH);
    digitalWrite(LED_MQTT,LOW);
    delay(1000);
    digitalWrite(LED_OK,HIGH);
    digitalWrite(LED_CONEXION_WIFI,HIGH);
    digitalWrite(LED_MQTT,HIGH);
    delay(1000);
    digitalWrite(LED_OK,LOW);
```

```
    digitalWrite(LED_CONEXION_WIFI,LOW);
    digitalWrite(LED_MQTT,HIGH);
    delay(1000);
    digitalWrite(LED_OK,LOW);
    digitalWrite(LED_CONEXION_WIFI,HIGH);
    digitalWrite(LED_MQTT,HIGH);
  }
}
// Función para reconectarse
void reconnect() {
  // Bucle hasta lograr conexión
  while (!client.connected()) { // Pregunta si hay conexión
    digitalWrite(LED_MQTT,LOW);
    Serial.print("Tratando de contectarse...");
    // Intentar reconexión
    digitalWrite(LED_MQTT,HIGH);
    if (client.connect("ESP32CAMClient")) { //Pregunta por el resultado del intento de conexión
      Serial.println("Conectado");
    }// fin del  if (client.connect("ESP32CAMClient"))
    else {  //en caso de que la conexión no se logre
      Serial.print("Conexion fallida, Error rc=");
      Serial.print(client.state()); // Muestra el codigo de error
      Serial.println(" Volviendo a intentar en 5 segundos");
      digitalWrite(LED_MQTT,LOW);
      delay(500);
      digitalWrite(LED_MQTT,HIGH);
      delay(500);
      digitalWrite(LED_MQTT,LOW);
      delay(500);
      digitalWrite(LED_MQTT,HIGH);
      delay(500);
      digitalWrite(LED_MQTT,LOW);
      delay(500);
      digitalWrite(LED_MQTT,HIGH);
      delay(500);
      digitalWrite(LED_MQTT,LOW);
      delay(500);
      digitalWrite(LED_MQTT,HIGH);
      delay(500);
      digitalWrite(LED_MQTT,LOW);
      delay(500);
      digitalWrite(LED_MQTT,HIGH);
      delay(500);
      Serial.println (client.connected ()); // Muestra estatus de conexión
    }// fin del else
  }// fin del bucle while (!client.connected())
}// fin de void reconnect()
```

Para más información revisar: Repositorio MAX30102

2.1.5. DHT11

Armado del circuito

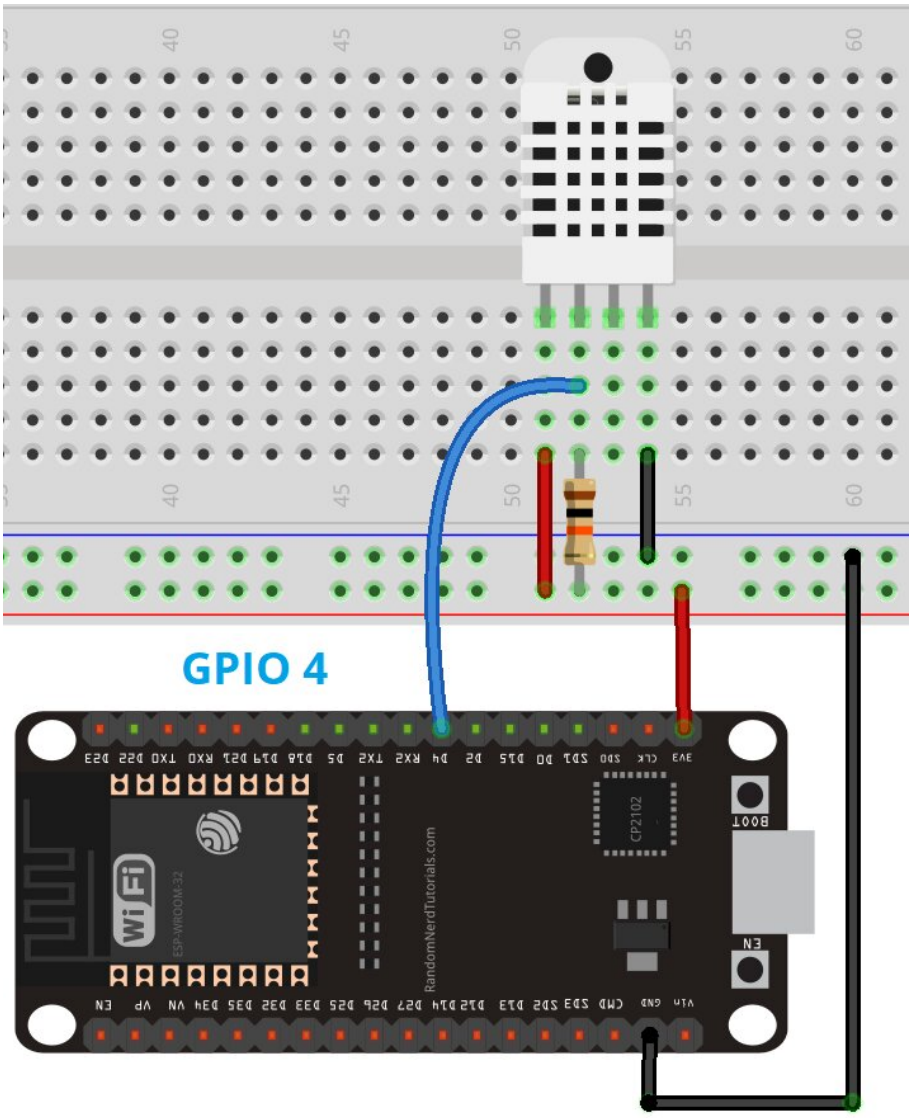

Figura 2.6: Diagrama de conexiones DHT11

Código

```
/*
AUTOR: Jorge Isur Balderas Ramírez
FECHA: 25/08/2021
DISPOSITIVO: ESP32CAM + DHT11
```

```
5     DESCRIPCIÓN: Programa que muestra la temperatura medida por el DHT11
6     GPIO DESCRIPTION:
7     DHT11 DATA---------------->GPIO2
8     DHT11 POWER--------------->3.3V
9     DHT11 GND----------------->GND
10    LED_OK(VERDE)------------>GPIO13
11    LED_WARNING(AMARILLO)--->GPIO14
12    LED_FATAL(ROJO)---------->GPIO15
13    LED_STATUS--------------->GPIO33{
14      INVERSO: ON--->LOW
15               OFF-->HIGH
16      NO EXPUESTO: SOLO SE PUEDE MANIPULAR MEDIANTE SOFTWARE.
17    }
18 */
19 //BIBLIOTECAS
20 #include "DHT.h"
21 #include <WiFi.h>  // Biblioteca para el control de WiFi
22 #include <PubSubClient.h> //Biblioteca para conexion MQTT
23 // DEFINIMOS LOS PINES QUE SE USARÁN
24 #define DHTPIN 2
25 #define DHTTYPE DHT11   // DHT 11
26 #define LED_OK 13
27 #define LED_WARNING 14
28 #define LED_FATAL 15
29 #define LED_STATUS 33
30 //DATOS DEL GUAIFAI
31 const char* ssid = "INFINITUM3033_2.4";//CAMBIAR POR TU NOMBRE DE RED.
32 const char* password = "yYYmteq554"; //tu contraseña de wifi
33 //DATOS DEL BROKER MQTT
34 //en caso de usar broker publico, actualizar la ip.
35 const char* mqtt_server = "18.198.240.106";
36 IPAddress server(18,198,240,106);
37 //objetos
38 WiFiClient esp32Client;
39 PubSubClient client(esp32Client);
40 //inicializar el DHT
41 DHT dht(DHTPIN, DHTTYPE);
42
43 void setup() {
44   Serial.begin(115200);
45   Serial.println(F("Lector de temperatura y humedad iniciado."));
46   pinMode (LED_OK, OUTPUT);//Specify that LED pin is output
47   pinMode (LED_WARNING, OUTPUT);//Specify that LED pin is output
48   pinMode (LED_FATAL, OUTPUT);//Specify that LED pin is output
49   Serial.println();
50   Serial.println();
51   Serial.print("Conectar a ");
52   Serial.println(ssid);
53
54   WiFi.begin(ssid, password); // Esta es la función que realiz la conexión a WiFi
55
```

```
  while (WiFi.status() != WL_CONNECTED) { // Este bucle espera a que se realice la conexión
    digitalWrite (LED_STATUS, HIGH);
    delay(500); //dado que es de suma importancia esperar a la conexión, debe usarse espera bloqueante
    digitalWrite (LED_STATUS, LOW);
    Serial.print("."); // Indicador de progreso
    delay (5);
  }

  // Cuando se haya logrado la conexión, el programa avanzará, por lo tanto, puede informarse lo siguiente
  Serial.println();
  Serial.println("WiFi conectado");
  Serial.println("Direccion IP: ");
  Serial.println(WiFi.localIP());
  delay (1000); // Esta espera es solo una formalidad antes de iniciar la comunicación con el broker
  // Conexión con el broker MQTT
  client.setServer(server, 1883); // Conectarse a la IP del broker en el puerto indicado
  client.setCallback(callback); // Activar función de CallBack, permite recibir mensajes MQTT y ejecutar funciones a p
  delay(1500);  // Esta espera es preventiva, espera a la conexión para no perder información
  dht.begin();
}

void loop() {
  //Verificar siempre que haya conexión al broker
  if (!client.connected()) {
    reconnect();  // En caso de que no haya conexión, ejecutar la función de reconexión, definida despues del void se
  }// fin del if (!client.connected())
  client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante las funciones necesarias para la c
  // Wait a few seconds between measurements.
  delay(5000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  //float f = dht.readTemperature(true);
  String hum = String(h);
  String temp = String(t);
  //conversion a char[]
  char* humedad = hum.toCharArray();
  char* celsius = temp.toCharArray();
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("ERROR AL REGISTRAR LOS DATOS DEL SENSOR!"));
    return;
  }
  if (t<25)
  {
      digitalWrite(LED_OK,HIGH);
      digitalWrite(LED_WARNING,LOW);
```

```
      digitalWrite(LED_FATAL,LOW);
  }
  if (t>=25 && t<30)
  {
      digitalWrite(LED_WARNING,HIGH);
      digitalWrite(LED_FATAL,LOW);
      digitalWrite(LED_OK,LOW);
  }
  else{
      digitalWrite(LED_OK,LOW);
      digitalWrite(LED_WARNING,LOW);
      digitalWrite(LED_FATAL,HIGH);
  }
  Serial.print(F("Humedad: "));
  Serial.print(h);
  Serial.print(F("%  Temperatura: "));
  Serial.print(t);
  Serial.print(F("°C \n"));
  client.publish("isur/humedad",humedad);
  client.publish("isur/temp",temp);
}
// Función para reconectarse
void reconnect() {
  // Bucle hasta lograr conexión
  while (!client.connected()) { // Pregunta si hay conexión
    Serial.print("Tratando de contectarse...");
    // Intentar reconexión
    if (client.connect("ESP32CAMClient")) { //Pregunta por el resultado del intento de conexión
      Serial.println("Conectado");
    }// fin del  if (client.connect("ESP32CAMClient"))
    else {  //en caso de que la conexión no se logre
      Serial.print("Conexion fallida, Error rc=");
      Serial.print(client.state()); // Muestra el codigo de error
      Serial.println(" Volviendo a intentar en 5 segundos");
      // Espera de 5 segundos bloqueante
      delay(5000);
      Serial.println (client.connected ()); // Muestra estatus de conexión
    }// fin del else
  }// fin del bucle while (!client.connected())
}// fin de void reconnect()
```

Para más información revisar : Repositorio DHT11

2.2. Código de base de datos

Código registroBaseDatos

```
"""
Administrador de base de datos: Registro UAM-LERMA
Autor: Jorge Isur Balderas Ramirez
Fecha: 13-12-2021
"""
import pymongo
import time
import paho.mqtt.client as mqtt #manejo de conexiones mqtt
import json
import datetime as dt
miCliente = pymongo.MongoClient("mongodb://localhost:27017/")
base_datos = miCliente["UAM"]
alumnos = base_datos["alumnos"]
admin = base_datos["administrativos"]
def on_connect(client,userdata,flags,rc):
   print(f"Conectado con codigo:{rc}")
   client.subscribe("isur/uid")
def on_message(client,userdata,msg):
   uid = msg.payload
   uid = uid.decode()
   print(f"Tarjeta detectada con UID:{uid}")
   fecha  = dt.datetime.now()
   hora = dt.datetime.now().hour
   minuto = dt.datetime.now().minute
   dia = dt.datetime.now().day
   mes = dt.datetime.now().month
   if hora < 10:
       hora = '0'+str(hora)
   if minuto <10:
       minuto = '0'+str(minuto)
   if mes < 10:
       mes = '0'+str(mes)
   if dia < 10:
       dia = '0'+str(dia)
   print(f"Fecha:{dia}-{mes}-{fecha.year} {hora}:{minuto}")
   client.unsubscribe("isur/uid")
   client.disconnect()
   menu(uid,fecha)
def existeDatabase(db):
    dblist = miCliente.list_database_names()
    if db in dblist:
        print(f"Base de datos {db} encontrada.\n")
        return True
    else:
```

```
45          print(f"Base de datos {db} no existente.\n")
46          return False
47          exit()
48  def existeCollecion(coleccion):
49      listaColeccion = base_datos.list_collection_names()
50      if coleccion in listaColeccion:
51          print(f"Colección {coleccion} encontrada.\n")
52          return True
53      else:
54          print(f"Collecion {db} no existente.\n")
55          return False
56          exit()
57  def registro(uid,fecha):
58      print("************************************************")
59      print("********************REGISTRO********************")
60      opcion=input("Ingrese 0 para regresar al menú y 1 para continuar.\n")
61      if opcion=='0':
62          menu()
63      tipo = input("1. Profesor/Administrativo\n"
64                   "2. Alumno\n")
65      nombre = input("Ingresa el nombre:\t")
66      matricula = input("Ingresa la matricula:\t")
67      info = input("Ingresa la carrera:\t")
68      miRegistro = {"_id":uid,"nombre":nombre,"info":info,"matricula":matricula,"fecha_entrada":fecha,"fecha_salida":"/-
69      if tipo=='1':
70          for dato in admin.find({},{"_id":1}):
71              if dato["_id"]==matricula:
72                  print("Esta matricula ya fue registrada con anterioridad")
73                  exit()
74          push = admin.insert_one(miRegistro)
75          print("************************************************")
76          print("Registro exitoso.\n")
77          print("************************************************")
78          time.sleep(2)
79          ordenamiento = admin.find().sort("nombre",-1)
80      if tipo=='2':
81          for dato in alumnos.find({},{"_id":1}):
82              if dato["_id"]==matricula:
83                  print("Esta matricula ya fue registrada con anterioridad")
84                  exit()
85          push = alumnos.insert_one(miRegistro)
86          print("************************************************")
87          print("Registro exitoso.\n")
88          print("************************************************")
89          ordenamiento = alumnos.find().sort("nombre",-1)
90  def menu(uid,fecha):
91     print("""------------------MENU--------------------
92     1-Registro
93     2-Mostrar todos los alumnos
94     3-Mostrar todos los administrativos/profesores
95     ------------------------------------------""")
```

```
96      opcion = input("Ingresa tu opcion--->")
97      if opcion == '1':
98         registro(uid,fecha)
99      if opcion == '2':
100        print("****************************************")
101        print("*****************ALUMNOS****************")
102        estado_db = existeDatabase("UAM")
103        estado_col = existeCollecion("alumnos")
104        if estado_db and estado_col:
105           for datos in alumnos.find():
106              print("****************************************")
107              print("Nombre:\t"+datos["nombre"])
108              print("UID:\t"+uid)
109              print("Carrera:\t"+datos["info"])
110              print("Matricula:\t"+datos["matricula"])
111              print("Ultima entrada:\t"+str(datos["fecha_entrada"]))
112              print(f"Ultima salida:{datos['fecha_salida']}")
113              print("****************************************")
114     if opcion == '3':
115        print("****************************************")
116        print("*****************ADMIN******************")
117        estado_db2 = existeDatabase("UAM")
118        estado_col2 = existeCollecion("administrativos")
119        if estado_db2 and estado_col2:
120           for datos in admin.find():
121              print("****************************************")
122              print("Nombre:\t"+datos["nombre"])
123              print("UID:\t"+uid)
124              print("Carrera:\t"+datos["info"])
125              print("Matricula:\t"+datos["matricula"])
126              print(f'Ultima entrada: {datos["fecha"]}')
127              print("****************************************")
128 try:
129     client = mqtt.Client("Isur-PC")
130     client.on_connect = on_connect
131     client.on_message = on_message
132     client.connect("192.168.1.78", 1883, 60)
133     print("Acerca la tarjeta al lector.")
134     client.loop_forever()
135 except KeyboardInterrupt:
136     print("\n")
137     print("Finalizando programa.")
```

Código buscarUID

```
1  """
2  Administrador de base de busqueda: Busqueda UAM-LERMA
```

```
3   Autor: Jorge Isur Balderas Ramirez
4   Fecha: 03-02-2021
5   """
6   import pymongo
7   import time
8   import paho.mqtt.client as mqtt #manejo de conexiones mqtt
9   import json
10  import datetime as dt
11
12  miCliente = pymongo.MongoClient("mongodb://localhost:27017/")
13  base_datos = miCliente["UAM"]
14  alumnos = base_datos["alumnos"]
15  admin = base_datos["administrativos"]
16
17  global broker_ip
18  global port
19  global cliente
20  broker_ip = "192.168.1.78"
21  port = 1883
22  cliente = 'Isur-PC'
23  global separador
24  separador ="************************************"
25
26  def on_connect(client,userdata,flags,rc):
27     print(f"Cliente:{cliente}")
28     if rc==0:
29          print("Conexion exitosa")
30          client.subscribe("isur/uid")
31     else:
32         print(f"Conexion con {broker_ip}:{port} fallida, codigo de error: {rc}")
33  def on_message(client,userdata,msg):
34     uid = msg.payload
35     uid = uid.decode()
36     print(f"Tarjeta detectada con UID:{uid}")
37     fecha  = dt.datetime.now()
38     hora = dt.datetime.now().hour
39     minuto = dt.datetime.now().minute
40     dia = dt.datetime.now().day
41     mes = dt.datetime.now().month
42     if hora < 10:
43         hora = '0'+str(hora)
44     if minuto <10:
45         minuto = '0'+str(minuto)
46     if mes < 10:
47         mes = '0'+str(mes)
48     if dia < 10:
49         dia = '0'+str(dia)
50     print(f"Fecha:{dia}-{mes}-{fecha.year} {hora}:{minuto}")
51     client.unsubscribe("isur/uid")
52     client.disconnect()
53     existe = busqueda(uid,fecha)
```

```
    if existe ==False:
        print("Usuario no existente.")
    else:
        print("Usuario encontrado.")
        print("Datos enviados")
def existeDatabase(db):
    dblist = miCliente.list_database_names()
    if db in dblist:
        print(f"Base de busqueda {db} encontrada.\n")
        return True
    else:
        print(f"Base de busqueda {db} no existente.\n")
        return False
        exit()
def existeCollecion(coleccion):
    listaColeccion = base_datos.list_collection_names()
    if coleccion in listaColeccion:
        print(f"Colección {coleccion} encontrada.\n")
        return True
    else:
        print(f"Collecion {db} no existente.\n")
        return False
        exit()
def busqueda(uid,fecha):
    existe = False
    print(separador)
    busqueda = alumnos.find_one(uid)
    if busqueda !=None:
        nombre =busqueda["nombre"]
        print(f"Nombre:{nombre}")
        _id = busqueda["_id"]
        print(f"UID:{_id}")
        carrera = busqueda["info"]
        print(f"Carrera:{carrera}")
        matricula  = busqueda["matricula"]
        print(f"Matricula:{matricula}")
        fecha_entrada = busqueda["fecha_entrada"]
        print(f"Ultima entrada registrada:{fecha_entrada}")
        client.publish("isur/ultima_entrada",payload=str(fecha_entrada),qos=0,retain=False)
        fecha_salida = busqueda["fecha_salida"]
        print(f"Ultima salida registrada: {fecha_salida}")
        client.publish("isur/ultima_salida",payload=str(fecha_salida),qos=0,retain=False)
        estado = entradaSalida(fecha,fecha_entrada,fecha_salida)
        if estado==True:
            query = {"nombre":nombre}
            nuevaFecha = {"$set":{"fecha_entrada":fecha}}
            actualizar=alumnos.update_one(query,nuevaFecha)
            print(f"{actualizar.modified_count} registros actualizados")
        if estado==False:
            query = {"nombre":nombre}
            nuevaFecha = {"$set":{"fecha_salida":fecha}}
```

```
            actualizar = alumnos.update_one(query,nuevaFecha)
            print(f"{actualizar.modified_count} registros actualizados")
        print(separador)
        print("publicando...")
        client.connect("192.168.1.78", 1883, 60)
        client.publish("isur/usuario/nombre",payload=nombre,qos=0,retain=False)
        client.publish("isur/usuario/carrera",payload=carrera,qos=0,retain=False)
        client.publish("isur/usuario/matricula",payload=matricula,qos=0,retain=False)
        client.publish("isur/ultima_entrada",payload=str(fecha_entrada),qos=0,retain=False)
        client.publish("isur/ultima_salida",payload=str(fecha_salida),qos=0,retain=False)
        return True
    resultados = admin.find_one(uid)
    if resultados !=None:
        nombre =resultados["nombre"]
        print(f"Nombre:{nombre}")
        _id = resultados["_id"]
        print(f"UID:{_id}")
        carrera = resultados["info"]
        print(f"Carrera:{carrera}")
        matricula  = resultados["matricula"]
        print(f"Matricula:{matricula}")
        fecha_entrada = busqueda["fecha_entrada"]
        print(f"Ultima entrada registrada:{fecha_entrada}")
        fecha_salida = busqueda["fecha_salida"]
        print(f"Ultima salida registrada: {fecha_salida}")
        estado = entradaSalida(fecha,fecha_entrada,fecha_salida)
        if estado==True:
            query = {"nombre":nombre}
            nuevaFecha = {"$set":{"fecha_entrada":fecha}}
            actualizar=admin.update_one(query,nuevaFecha)
            print(f"{actualizar.modified_count} registros actualizados")
        if estado==False:
            query = {"nombre":nombre}
            nuevaFecha = {"$set":{"fecha_salida":fecha}}
            actualizar = admin.update_one(query,nuevaFecha)
            print(f"{actualizar.modified_count} registros actualizados")
        client.connect("192.168.1.78", 1883, 60)
        client.publish("isur/usuario/nombre",payload=nombre,qos=0,retain=False)
        client.publish("isur/usuario/carrera",payload=carrera,qos=0,retain=False)
        client.publish("isur/usuario/matricula",payload=matricula,qos=0,retain=False)
        return True
def entradaSalida(fecha,fecha_entrada,fecha_salida):
    try:
        if fecha_entrada > fecha_salida:
            print(f"Salio:{fecha}")
            return False
        if fecha > fecha_salida:
            print(f"Entro:{fecha}")
            return True
    except TypeError:
        print("Comparacion no permitida.")
```

```
        print(f"Salio:{fecha}")
        return False
try:
    client = mqtt.Client(cliente)
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(broker_ip, port, 60)
    print("Acerca la tarjeta al lector.")
    print(f"Conectando a {broker_ip}:{port}")
    client.loop_forever()
except KeyboardInterrupt:
    print("\n")
    print("Finalizando programa.")
```

Para más información revisar : Repositorio mongoDB

2.3. JSON de Node Red

```
[
    {
        "id": "43d97d32682ec8d6",
        "type": "tab",
        "label": "Flow 3",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "7bd8e5f4b9cf89c7",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "temperatura",
        "topic": "isur/temp",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 180,
        "y": 80,
        "wires": [
            [
                "3981e5e611260601",
```

```
                "9846f0581858083d",
                "8dca63be913f04e3"
            ]
        ]
    },
    {
        "id": "8f293b58c5771ec5",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "oxigeno",
        "topic": "isur/oxigeno",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 110,
        "y": 140,
        "wires": [
            [
                "74816a4c1cbff5ac",
                "06158c3acb377cc1",
                "bbf8f77038ef22c8"
            ]
        ]
    },
    {
        "id": "74816a4c1cbff5ac",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "74f876ad7860a6ce",
        "order": 1,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Oxigenacion",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "className": "",
        "x": 350,
        "y": 120,
        "wires": []
    },
    {
        "id": "9e01e73598137c09",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/pulso",
```

```
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 100,
        "y": 280,
        "wires": [
            [
                "7345e8202779e282",
                "079883ecdd943b79",
                "ea9246f4b46aee4b"
            ]
        ]
    },
    {
        "id": "65f9b70df85f1ff8",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "Temp Ambiente",
        "topic": "isur/temperatura_salon",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 120,
        "y": 340,
        "wires": [
            [
                "f7d803e5b777ce11"
            ]
        ]
    },
    {
        "id": "f7d803e5b777ce11",
        "type": "ui_gauge",
        "z": "43d97d32682ec8d6",
        "name": "Ambiente",
        "group": "09297ea9730f36a6",
        "order": 1,
        "width": 5,
        "height": 3,
        "gtype": "gage",
        "title": "Temperatura Ambiente",
        "label": "°C",
        "format": "{{value}}",
```

```
        "min": 0,
        "max": "40",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "20",
        "seg2": "30",
        "className": "",
        "x": 360,
        "y": 340,
        "wires": []
    },
    {
        "id": "f0cf5563db53c86c",
        "type": "exec",
        "z": "43d97d32682ec8d6",
        "command": "python3 /home/pi/RASPBERRYPI-CODIGOIOT/PyMLX90614-0.0.3/mlx90614.py",
        "addpay": "",
        "append": "",
        "useSpawn": "false",
        "timer": "",
        "winHide": false,
        "oldrc": false,
        "name": "mlx90614",
        "x": 460,
        "y": 460,
        "wires": [
            [],
            [],
            []
        ]
    },
    {
        "id": "7caa034a49faa628",
        "type": "ui_button",
        "z": "43d97d32682ec8d6",
        "name": "Monitoreo",
        "group": "bb21fdcd96e09c93",
        "order": 1,
        "width": 0,
        "height": 0,
        "passthru": false,
        "label": "Iniciar monitoreo.",
        "tooltip": "",
        "color": "",
        "bgcolor": "",
        "className": "",
        "icon": "",
        "payload": "",
```

```
        "payloadType": "str",
        "topic": "topic",
        "topicType": "msg",
        "x": 240,
        "y": 460,
        "wires": [
            [
                "f0cf5563db53c86c"
            ]
        ]
    },
    {
        "id": "3981e5e611260601",
        "type": "ui_gauge",
        "z": "43d97d32682ec8d6",
        "name": "Corporal",
        "group": "74f876ad7860a6ce",
        "order": 3,
        "width": 6,
        "height": 3,
        "gtype": "gage",
        "title": "Temperatura corporal",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "42",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "36",
        "seg2": "38",
        "className": "",
        "x": 520,
        "y": 100,
        "wires": []
    },
    {
        "id": "467d246c3f24d21b",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/uid",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
```

```
        "x": 70,
        "y": 520,
        "wires": [
            [
                "d1d6ce5e49c3d7f6",
                "a160d5d997580a18"
            ]
        ]
    },
    {
        "id": "f8597d449a3897cc",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
        "order": 1,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Nombre",
        "format": "{{msg.payload}}",
        "layout": "row-spread",
        "className": "",
        "x": 460,
        "y": 580,
        "wires": []
    },
    {
        "id": "5ac4be9b513bf859",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "isur/usuario/nombre",
        "topic": "isur/usuario/nombre",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 110,
        "y": 580,
        "wires": [
            [
                "f8597d449a3897cc"
            ]
        ]
    },
    {
        "id": "981f24d2f4a9d8e4",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
```

```
        "name": "isur/usuario/matricula",
        "topic": "isur/usuario/matricula",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 120,
        "y": 640,
        "wires": [
            [
                "2c2d61faaba5f17f"
            ]
        ]
    },
    {
        "id": "876923e57eb25aaa",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/usuario/carrera",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 110,
        "y": 700,
        "wires": [
            [
                "f5bb69bb47bbddc6"
            ]
        ]
    },
    {
        "id": "1fce39afb392b0b6",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/estado",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
```

```
        "x": 110,
        "y": 780,
        "wires": [
            [
                "950251360d683c14",
                "731d6edef7957b71"
            ]
        ]
    },
    {
        "id": "f52dbb6b6dd07805",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/puerta",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 100,
        "y": 900,
        "wires": [
            [
                "9d54ecc6fe5d2260",
                "faaf816797493bd7"
            ]
        ]
    },
    {
        "id": "8d3d72763908f12a",
        "type": "ui_switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "label": "Puerta",
        "tooltip": "",
        "group": "bb21fdcd96e09c93",
        "order": 4,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "topic",
        "topicType": "msg",
        "style": "",
        "onvalue": "true",
        "onvalueType": "bool",
        "onicon": "",
        "oncolor": "",
```

```
        "offvalue": "false",
        "offvalueType": "bool",
        "officon": "",
        "offcolor": "",
        "animate": false,
        "className": "",
        "x": 630,
        "y": 900,
        "wires": [
            []
        ]
    },
    {
        "id": "950251360d683c14",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
        "order": 7,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Estado del usuario",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "className": "",
        "x": 370,
        "y": 780,
        "wires": []
    },
    {
        "id": "f5bb69bb47bbddc6",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
        "order": 2,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Carrera",
        "format": "{{msg.payload}}",
        "layout": "row-spread",
        "className": "",
        "x": 390,
        "y": 700,
        "wires": []
    },
    {
        "id": "2c2d61faaba5f17f",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
```

```
        "order": 3,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Matricula",
        "format": "{{msg.payload}}",
        "layout": "row-spread",
        "className": "",
        "x": 420,
        "y": 640,
        "wires": []
    },
    {
        "id": "a160d5d997580a18",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
        "order": 4,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "UID",
        "format": "{{msg.payload}}",
        "layout": "row-spread",
        "className": "",
        "x": 590,
        "y": 520,
        "wires": []
    },
    {
        "id": "d1d6ce5e49c3d7f6",
        "type": "trigger",
        "z": "43d97d32682ec8d6",
        "name": "",
        "op1": "true",
        "op2": "true",
        "op1type": "bool",
        "op2type": "bool",
        "duration": "0",
        "extend": false,
        "overrideDelay": false,
        "units": "ms",
        "reset": "",
        "bytopic": "all",
        "topic": "topic",
        "outputs": 1,
        "x": 240,
        "y": 400,
        "wires": [
            [
                "5df1d6da36915ea6"
```

```
            ]
        ]
    },
    {
        "id": "5df1d6da36915ea6",
        "type": "ui_switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "label": "Tarjeta",
        "tooltip": "",
        "group": "bb21fdcd96e09c93",
        "order": 3,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "topic",
        "topicType": "msg",
        "style": "",
        "onvalue": "true",
        "onvalueType": "bool",
        "onicon": "",
        "oncolor": "",
        "offvalue": "false",
        "offvalueType": "bool",
        "officon": "",
        "offcolor": "",
        "animate": false,
        "className": "",
        "x": 450,
        "y": 400,
        "wires": [
            []
        ]
    },
    {
        "id": "f398a28905529942",
        "type": "ui_switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "label": "Temperatura",
        "tooltip": "",
        "group": "d9873314e1d61643",
        "order": 3,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "topic",
        "topicType": "msg",
        "style": "",
```

```
        "onvalue": "true",
        "onvalueType": "bool",
        "onicon": "",
        "oncolor": "",
        "offvalue": "false",
        "offvalueType": "bool",
        "officon": "",
        "offcolor": "",
        "animate": false,
        "className": "",
        "x": 650,
        "y": 20,
        "wires": [
            []
        ]
    },
    {
        "id": "9846f0581858083d",
        "type": "trigger",
        "z": "43d97d32682ec8d6",
        "name": "",
        "op1": "true",
        "op2": "0",
        "op1type": "bool",
        "op2type": "str",
        "duration": "0",
        "extend": false,
        "overrideDelay": false,
        "units": "ms",
        "reset": "",
        "bytopic": "all",
        "topic": "topic",
        "outputs": 1,
        "x": 420,
        "y": 20,
        "wires": [
            [
                "f398a28905529942"
            ]
        ]
    },
    {
        "id": "06158c3acb377cc1",
        "type": "trigger",
        "z": "43d97d32682ec8d6",
        "name": "",
        "op1": "true",
        "op2": "0",
        "op1type": "bool",
        "op2type": "str",
        "duration": "0",
```

```
        "extend": false,
        "overrideDelay": false,
        "units": "ms",
        "reset": "",
        "bytopic": "all",
        "topic": "topic",
        "outputs": 1,
        "x": 360,
        "y": 160,
        "wires": [
            [
                "977915fd173980bd"
            ]
        ]
    },
    {
        "id": "977915fd173980bd",
        "type": "ui_switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "label": "Oxigeno en sangre",
        "tooltip": "",
        "group": "d9873314e1d61643",
        "order": 2,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "topic",
        "topicType": "msg",
        "style": "",
        "onvalue": "true",
        "onvalueType": "bool",
        "onicon": "",
        "oncolor": "",
        "offvalue": "false",
        "offvalueType": "bool",
        "officon": "",
        "offcolor": "",
        "animate": false,
        "className": "",
        "x": 640,
        "y": 140,
        "wires": [
            []
        ]
    },
    {
        "id": "7345e8202779e282",
        "type": "trigger",
        "z": "43d97d32682ec8d6",
```

```
        "name": "",
        "op1": "true",
        "op2": "0",
        "op1type": "bool",
        "op2type": "str",
        "duration": "0",
        "extend": false,
        "overrideDelay": false,
        "units": "ms",
        "reset": "",
        "bytopic": "all",
        "topic": "topic",
        "outputs": 1,
        "x": 340,
        "y": 300,
        "wires": [
            [
                "e1626b421615b47f"
            ]
        ]
    },
    {
        "id": "e1626b421615b47f",
        "type": "ui_switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "label": "Pulso cardiaco",
        "tooltip": "",
        "group": "d9873314e1d61643",
        "order": 1,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "topic",
        "topicType": "msg",
        "style": "",
        "onvalue": "true",
        "onvalueType": "bool",
        "onicon": "",
        "oncolor": "",
        "offvalue": "false",
        "offvalueType": "bool",
        "officon": "",
        "offcolor": "",
        "animate": false,
        "className": "",
        "x": 600,
        "y": 240,
        "wires": [
            []
```

```
        ]
    },
    {
        "id": "079883ecdd943b79",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "74f876ad7860a6ce",
        "order": 2,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Pulso cardiaco",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "className": "",
        "x": 340,
        "y": 260,
        "wires": []
    },
    {
        "id": "8dca63be913f04e3",
        "type": "switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "property": "payload",
        "propertyType": "msg",
        "rules": [
            {
                "t": "gt",
                "v": "37",
                "vt": "num"
            },
            {
                "t": "lt",
                "v": "37",
                "vt": "num"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 2,
        "x": 490,
        "y": 60,
        "wires": [
            [
                "2bb3a6a8ae4e14a2"
            ],
            [
                "b7ceedf0d5f7c56b"
            ]
        ]
```

```
    },
    {
        "id": "e55b0732936264b7",
        "type": "mqtt out",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/estado",
        "qos": "0",
        "retain": "false",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "f0de1995277616c8",
        "x": 1210,
        "y": 160,
        "wires": []
    },
    {
        "id": "2bb3a6a8ae4e14a2",
        "type": "change",
        "z": "43d97d32682ec8d6",
        "name": "enfermo",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "Enfermo",
                "tot": "str"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 960,
        "y": 160,
        "wires": [
            [
                "e55b0732936264b7"
            ]
        ]
    },
    {
        "id": "ea9246f4b46aee4b",
        "type": "switch",
        "z": "43d97d32682ec8d6",
        "name": "",
```

```
        "property": "payload",
        "propertyType": "msg",
        "rules": [
            {
                "t": "gt",
                "v": "100",
                "vt": "num"
            },
            {
                "t": "lte",
                "v": "100",
                "vt": "num"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 2,
        "x": 590,
        "y": 280,
        "wires": [
            [
                "2bb3a6a8ae4e14a2"
            ],
            [
                "b7ceedf0d5f7c56b"
            ]
        ]
    },
    {
        "id": "bbf8f77038ef22c8",
        "type": "switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "property": "payload",
        "propertyType": "msg",
        "rules": [
            {
                "t": "lt",
                "v": "90",
                "vt": "num"
            },
            {
                "t": "gt",
                "v": "90",
                "vt": "num"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 2,
        "x": 350,
```

```
        "y": 200,
        "wires": [
            [
                "2bb3a6a8ae4e14a2"
            ],
            [
                "b7ceedf0d5f7c56b"
            ]
        ]
    },
    {
        "id": "731d6edef7957b71",
        "type": "switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "property": "payload",
        "propertyType": "msg",
        "rules": [
            {
                "t": "eq",
                "v": "Enfermo",
                "vt": "str"
            },
            {
                "t": "eq",
                "v": "Sano",
                "vt": "str"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 2,
        "x": 290,
        "y": 820,
        "wires": [
            [
                "e623d4d8e909e143"
            ],
            [
                "e5d99a330ebf8dde"
            ]
        ]
    },
    {
        "id": "e5d99a330ebf8dde",
        "type": "change",
        "z": "43d97d32682ec8d6",
        "name": "Sano",
        "rules": [
            {
                "t": "set",
```

```
                "p": "payload",
                "pt": "msg",
                "to": "true",
                "tot": "bool"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 510,
        "y": 840,
        "wires": [
            [
                "18c2dcc52e4e2454"
            ]
        ]
    },
    {
        "id": "18c2dcc52e4e2454",
        "type": "mqtt out",
        "z": "43d97d32682ec8d6",
        "name": "abrir",
        "topic": "isur/puerta",
        "qos": "0",
        "retain": "false",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "f0de1995277616c8",
        "x": 750,
        "y": 820,
        "wires": []
    },
    {
        "id": "9d54ecc6fe5d2260",
        "type": "debug",
        "z": "43d97d32682ec8d6",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "false",
        "statusVal": "",
        "statusType": "auto",
        "x": 280,
        "y": 960,
```

```
        "wires": []
    },
    {
        "id": "b7ceedf0d5f7c56b",
        "type": "change",
        "z": "43d97d32682ec8d6",
        "name": "sano",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "Sano",
                "tot": "str"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 950,
        "y": 200,
        "wires": [
            [
                "e55b0732936264b7"
            ]
        ]
    },
    {
        "id": "e623d4d8e909e143",
        "type": "change",
        "z": "43d97d32682ec8d6",
        "name": "Enfermo",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "false",
                "tot": "bool"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 540,
        "y": 800,
        "wires": [
```

```
            [
                "18c2dcc52e4e2454"
            ]
        ]
    },
    {
        "id": "faaf816797493bd7",
        "type": "switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "property": "payload",
        "propertyType": "msg",
        "rules": [
            {
                "t": "eq",
                "v": "true",
                "vt": "str"
            },
            {
                "t": "eq",
                "v": "false",
                "vt": "str"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 2,
        "x": 260,
        "y": 900,
        "wires": [
            [
                "0fc559eabb6eddf9"
            ],
            [
                "9dd2375e7f2a9618"
            ]
        ]
    },
    {
        "id": "0fc559eabb6eddf9",
        "type": "change",
        "z": "43d97d32682ec8d6",
        "name": "Abrir",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "true",
                "tot": "bool"
            }
```

```
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 450,
        "y": 940,
        "wires": [
            [
                "8d3d72763908f12a"
            ]
        ]
    },
    {
        "id": "9dd2375e7f2a9618",
        "type": "change",
        "z": "43d97d32682ec8d6",
        "name": "cerrar",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "false",
                "tot": "bool"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 450,
        "y": 900,
        "wires": [
            [
                "8d3d72763908f12a"
            ]
        ]
    },
    {
        "id": "2fa8af79c6879149",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/humedad",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
```

```
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 110,
        "y": 1040,
        "wires": [
            [
                "3c921acc96e87efa"
            ]
        ]
    },
    {
        "id": "3c921acc96e87efa",
        "type": "ui_gauge",
        "z": "43d97d32682ec8d6",
        "name": "",
        "group": "09297ea9730f36a6",
        "order": 5,
        "width": 5,
        "height": 3,
        "gtype": "gage",
        "title": "Humedad",
        "label": "%",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "40",
        "seg2": "80",
        "className": "",
        "x": 330,
        "y": 1040,
        "wires": []
    },
    {
        "id": "844f769db521d280",
        "type": "mqtt out",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/puerta",
        "qos": "0",
        "retain": "false",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
```

```
        "broker": "f0de1995277616c8",
        "x": 620,
        "y": 1080,
        "wires": []
    },
    {
        "id": "00384d6419662701",
        "type": "ui_switch",
        "z": "43d97d32682ec8d6",
        "name": "",
        "label": "Apertura/Clausura manual",
        "tooltip": "",
        "group": "bb21fdcd96e09c93",
        "order": 5,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "topic",
        "topicType": "msg",
        "style": "",
        "onvalue": "true",
        "onvalueType": "bool",
        "onicon": "",
        "oncolor": "",
        "offvalue": "false",
        "offvalueType": "bool",
        "officon": "",
        "offcolor": "",
        "animate": false,
        "className": "",
        "x": 310,
        "y": 1080,
        "wires": [
            [
                "844f769db521d280"
            ]
        ]
    },
    {
        "id": "b61c3a903c2ca7bb",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/ultima_entrada",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
```

```
        "inputs": 0,
        "x": 130,
        "y": 1280,
        "wires": [
            [
                "548ed3df86ed4e47",
                "c47799459c20319c"
            ]
        ]
    },
    {
        "id": "548ed3df86ed4e47",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
        "order": 5,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Ultima entrada registrada",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "className": "",
        "x": 380,
        "y": 1280,
        "wires": []
    },
    {
        "id": "bd7c16ddfd090404",
        "type": "mqtt in",
        "z": "43d97d32682ec8d6",
        "name": "",
        "topic": "isur/ultima_salida",
        "qos": "2",
        "datatype": "auto",
        "broker": "f0de1995277616c8",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 120,
        "y": 1340,
        "wires": [
            [
                "7ee58748a70570c3",
                "c47799459c20319c"
            ]
        ]
    },
    {
        "id": "7ee58748a70570c3",
```

```
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "b4bde563647d3018",
        "order": 6,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Ultima salida registrada",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "className": "",
        "x": 380,
        "y": 1340,
        "wires": []
    },
    {
        "id": "c47799459c20319c",
        "type": "debug",
        "z": "43d97d32682ec8d6",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "false",
        "statusVal": "",
        "statusType": "auto",
        "x": 380,
        "y": 1420,
        "wires": []
    },
    {
        "id": "3d38033a8349b6f3",
        "type": "ui_text",
        "z": "43d97d32682ec8d6",
        "group": "bb21fdcd96e09c93",
        "order": 2,
        "width": "4",
        "height": "2",
        "name": "Lectura",
        "label": "Informacion",
        "format": "Presione el boton fisico durante 3 segundos, despues de iniciar monitoreo.",
        "layout": "col-center",
        "className": "",
        "x": 650,
        "y": 680,
        "wires": []
    },
    {
        "id": "ec4e1837fb1a3c60",
        "type": "ui_spacer",
```

```
        "z": "43d97d32682ec8d6",
        "name": "spacer",
        "group": "09297ea9730f36a6",
        "order": 2,
        "width": 2,
        "height": 1
    },
    {
        "id": "b56d4159134459fd",
        "type": "ui_spacer",
        "z": "43d97d32682ec8d6",
        "name": "spacer",
        "group": "09297ea9730f36a6",
        "order": 3,
        "width": 2,
        "height": 1
    },
    {
        "id": "415d2d09ed0dae53",
        "type": "ui_spacer",
        "z": "43d97d32682ec8d6",
        "name": "spacer",
        "group": "09297ea9730f36a6",
        "order": 4,
        "width": 2,
        "height": 1
    },
    {
        "id": "2f366e582e4f1e2e",
        "type": "ui_spacer",
        "z": "43d97d32682ec8d6",
        "name": "spacer",
        "group": "09297ea9730f36a6",
        "order": 6,
        "width": 2,
        "height": 1
    },
    {
        "id": "40d6337eeb411d45",
        "type": "ui_spacer",
        "z": "43d97d32682ec8d6",
        "name": "spacer",
        "group": "09297ea9730f36a6",
        "order": 7,
        "width": 2,
        "height": 1
    },
    {
        "id": "83d3ad6b88077d76",
        "type": "ui_spacer",
        "z": "43d97d32682ec8d6",
```

```
        "name": "spacer",
        "group": "09297ea9730f36a6",
        "order": 8,
        "width": 2,
        "height": 1
    },
    {
        "id": "f0de1995277616c8",
        "type": "mqtt-broker",
        "name": "broker isur",
        "broker": "192.168.1.78",
        "port": "1883",
        "clientid": "",
        "autoConnect": true,
        "usetls": false,
        "protocolVersion": "4",
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "birthMsg": {},
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "closeMsg": {},
        "willTopic": "",
        "willQos": "0",
        "willPayload": "",
        "willMsg": {},
        "sessionExpiry": ""
    },
    {
        "id": "74f876ad7860a6ce",
        "type": "ui_group",
        "name": "Signos vitales",
        "tab": "030f73c40792556d",
        "order": 3,
        "disp": true,
        "width": "6",
        "collapse": false,
        "className": ""
    },
    {
        "id": "09297ea9730f36a6",
        "type": "ui_group",
        "name": "Salon",
        "tab": "030f73c40792556d",
        "order": 1,
        "disp": true,
        "width": 7,
```

```
        "collapse": false,
        "className": ""
    },
    {
        "id": "bb21fdcd96e09c93",
        "type": "ui_group",
        "name": "Control",
        "tab": "030f73c40792556d",
        "order": 2,
        "disp": true,
        "width": 6,
        "collapse": false,
        "className": ""
    },
    {
        "id": "b4bde563647d3018",
        "type": "ui_group",
        "name": "Usuario",
        "tab": "030f73c40792556d",
        "order": 5,
        "disp": true,
        "width": "6",
        "collapse": false,
        "className": ""
    },
    {
        "id": "d9873314e1d61643",
        "type": "ui_group",
        "name": "Lectura de sintomas",
        "tab": "030f73c40792556d",
        "order": 4,
        "disp": true,
        "width": "6",
        "collapse": false,
        "className": ""
    },
    {
        "id": "030f73c40792556d",
        "type": "ui_tab",
        "name": "Centro de control",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```


Para más información revisar : Repositorio Node Red

Capítulo 3

Conclusiones

3.1. Conclusiones

Durante la realización de este proyecto, hemos aprendido a usar tecnologías que a día de hoy se utilizan en entornos profesionales reales, lo cuál nos dará una ventaja a la hora de desempeñarnos en el campo laboral. Gracias a las enseñanzas del curso impartido, somos capaces de poder imaginar, diseñar e implementar soluciones altamente eficientes en términos de Internet de las Cosas, Programación y manejo de Sistemas Linux. Nuestro proyecto podrá ayudar a las futuras generaciones de la UAM unidad Lerma a gozar de un entorno de laboratorio totalmente automatizado, donde la dependencia del personal para tareas rutinarias y/o no vitales pueda ser derribada, con el fin de aprovechar los laboratorios al máximo de su capacidad, independientemente del horario, así como de una manera segura con el fin de evitar la propagación del COVID-19.