



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

Índice de figuras	II
1. Introducción	2
1.1. Introducción	2
1.2. Objetivos	2
1.2.1. Objetivo general	2
1.2.2. Objetivos específicos	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 específicos

- 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)

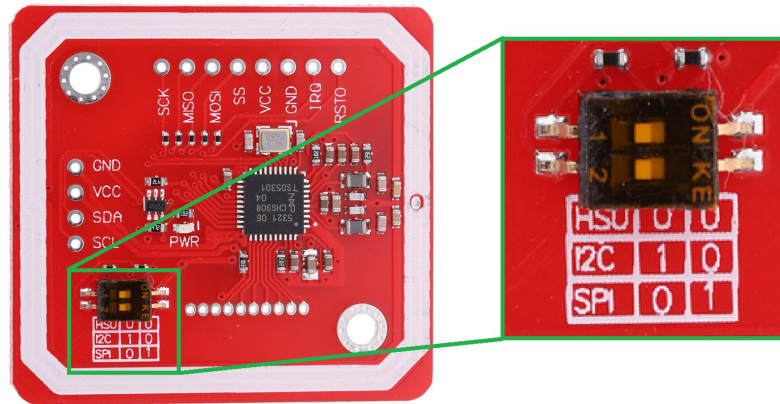


Figura 2.1: Tabla PN532

Conexiones del sensor a Raspberry

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

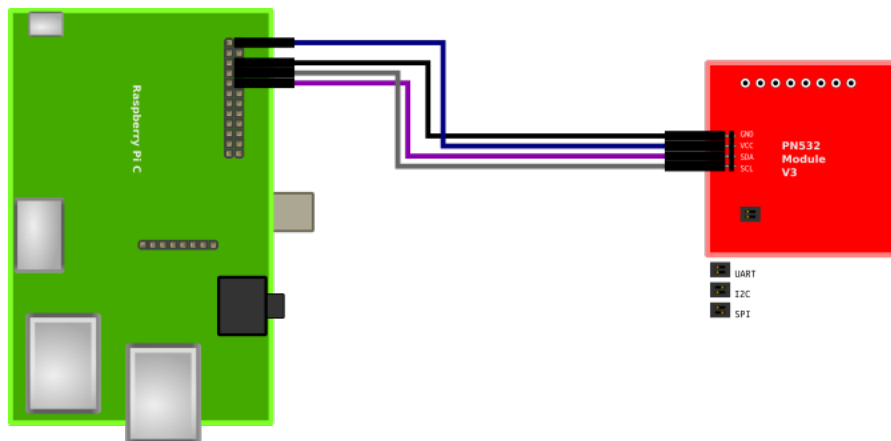


Figura 2.2: Conexión PN532 a Raspberry

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.

```
1      #####
2      Autor: Juan Elier Rosales Rosas
3      Fecha: 10/02/22
4      Descripción: Programa para leer un dispositivo NFC mediante
5                  el sensor PN532.
6
7      #####
8      #####
9      #Declaración de modulos
10     #####
11     import paho.mqtt.client as mqtt
12     import time
13     import binascii
14
15     #####
16     #Declaración de bibliotecas
17     #####
18
19     from pn532pi import Pn532, pn532
20     from pn532pi import Pn532I2c
21     from pn532pi import Pn532Spi
22     from pn532pi import Pn532Hsu
23
24     #####
25     #Elección de interface a usar.
26     #Para seleccionar coloca el valor true
27     #####
28     SPI = False
29     I2C = False
```

```

30 HSU = True
31 #####
32 #Comprobación de los valores para la interfaz
33 #####
34 if SPI:
35     PN532_SPI = Pn532Spi(Pn532Spi.SS0_GPI08)
36     nfc = Pn532(PN532_SPI)
37     # Cuando el número después de #elif es 1, se deberá poner en HSU.
38 elif HSU:
39     PN532_HSU = Pn532Hsu(0)
40     nfc = Pn532(PN532_HSU)
41
42 # Cuando el número después de #if & #elif se configura con
43 # 0, se debera poner en modo I2C
44 elif I2C:
45     PN532_I2C = Pn532I2c(1)
46     nfc = Pn532(PN532_I2C)
47
48 #####
49 #Llamada para cuando el cliente recibe respuesta del servidor
50 #####
51 def on_connect(client, userdata, flags, rc):
52     print("Connected with result code "+str(rc))
53
54 client = mqtt.Client()
55 client.on_connect = on_connect
56 client.connect("187.200.114.169", 1883, 60) #Aquí cambias por la ip o dirección de tu broker
57
58 def setup():
59     nfc.begin()
60
61     versiondata = nfc.getFirmwareVersion()
62     if not versiondata:
63         print("Didn't find PN53x board")
64         raise RuntimeError("Didn't find PN53x board") # halt
65
66     # Confirmación de sensor encontrado
67     print("Lector PN5 encontrado {:#x}".format((versiondata >> 24) & 0xFF,
68         (versiondata >> 16) & 0xFF, (versiondata >> 8) & 0xFF))
69     # Configura el número maximo de intentos para leer alguna tarjeta
70     #Esto previene que se espere una tarjeta siempre
71     #Está definido por default por el PN532
72     nfc.setPassiveActivationRetries(0xFF)
73
74     # Configura para leer los tags RFID
75     nfc.SAMConfig()
76
77 def loop():
78     # Se espera un tag de tipo ISO14443A (Mifare, etc.). Cuando una es encontrada
79     # 'uid' se llena con el UID, y la longitud del UID indica que
80     # 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 éxito 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

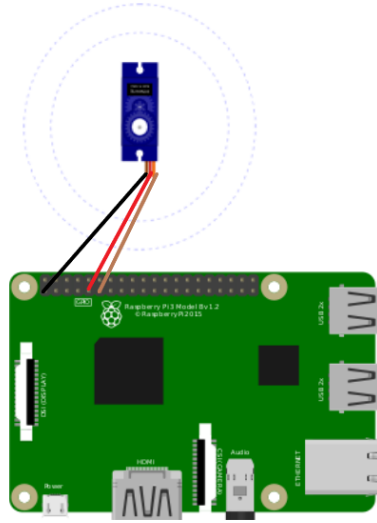


Figura 2.3: Diagrama conexiones servo a Raspberry

Conexiones

Pin del servo	Color del cable	Pin Raspberry
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

```

1      #Autor: Juan Elieir Rosales Rosas
2      #Fecha: 11/02/22
3      #Descripción: Programa para mover un servomotor simulando la apertura de una puerta
4      #####
5      #Declaración de módulos
6      #####
7      import RPi.GPIO as GPIO
8      import time
9      import paho.mqtt.client as mqtt
10     #####
11     #Configuración de GPIO
12     #####
13     GPIO.setmode(GPIO.BOARD)
14
15     GPIO.setup(11,GPIO.OUT) #Se configura el pin 12 como salida
16     servo = GPIO.PWM(11, 50) # 12 es el pin y 50 el pulso en Hz
17     #####
18     #Configuración cliente MQTT
19     #####
20     def on_connect(client, userdata, flags, rc):
21         print("Esperando... ")
22
23         client.subscribe("isur/puerta")
24
25     def on_message(client, userdata, msg):
26         abierto=msg.payload.decode()
27         acceso(abierto)
28         #####
29         #Función main
30         #####
31     def main():
32         client = mqtt.Client()
33         client.connect("187.200.112.52", 1883, 60)
34         client.on_connect = on_connect
35         client.on_message = on_message
36         client.loop_forever()
37         servo.stop()
38         GPIO.cleanup()
39         client.disconnect()
40
41     def acceso(abierto):
42         servo.start(0)
43         if abierto=="true":
44             print("Puerta abierta")
45             servo.ChangeDutyCycle(7)
46             time.sleep(0.5)
47             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

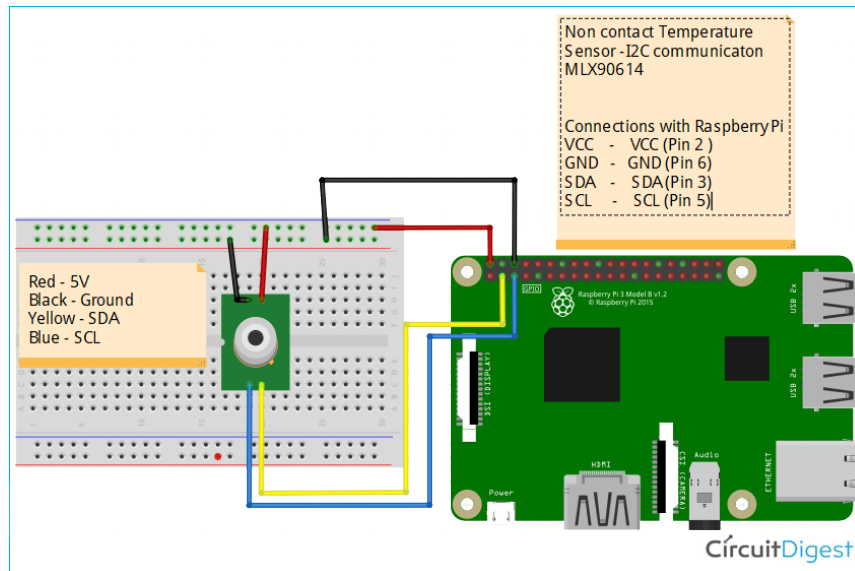


Figura 2.4: Diagrama de conexiones MLX90614

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

```

1  #declaracion de modulos
2  from smbus2 import SMBus #administrar el bus de datos
3  import paho.mqtt.client as mqtt #manejo de conexiones mqtt
4  from mlx90614 import MLX90614 #manejo del sensor de temperatura
5  import time #manejo del tiempo
6  import RPi.GPIO as gpio #control del acceso GPIO de la raspberry
7  import random #generacion de numeros aleatorios
8  import urllib.request as urllib2 #comprobacion de conexion a internet
9  import datetime as dt
10 #instancias de objetos
11 bus = SMBus(1)
12 sensor = MLX90614(bus, address=0x5A)
13 #LEDS Y PINES
14 LED_ENCENDIDO = 12
15 LED_MQTT = 16
16 LED_Lectura = 18
17 pin_button = 22
18 #funcion para conectarnos a mqtt
19 def on_connect(client, userdata, flags, rc):
20     for i in range(2):
21         gpio.output(LED_MQTT,True)
22         gpio.output(LED_MQTT,False)
23         gpio.output(LED_MQTT,True)
24         gpio.output(LED_MQTT,False)
25         gpio.output(LED_MQTT,True)
26         gpio.output(LED_MQTT,False)
27     print(f"Conectado con codigo de resultado {rc}")
28 def hayInternet():
29     try:
30         urllib2.urlopen('http://google.com.mx', timeout=1)
31         return True
32     except urllib2.URLError as err:
33         return False
34 def standby(LED_ENCENDIDO,LED_MQTT,LED_STANDBY):
35     modo = random.randint(0,3)
36     if modo == 0:
37         gpio.output(LED_ENCENDIDO,False)
38         gpio.output(LED_MQTT,False)
39         gpio.output(LED_Lectura,True)
40         time.sleep(1)
41         gpio.output(LED_ENCENDIDO,False)
42         gpio.output(LED_MQTT,True)
43         gpio.output(LED_Lectura,True)
44         time.sleep(1)
45         gpio.output(LED_ENCENDIDO,True)
46         gpio.output(LED_MQTT,True)
47         gpio.output(LED_Lectura,True)
48         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:
```

```

100     minutos = '0'+str(minutos)
101     print(f"{dia}/{mes}/{year} a las {hora}:{minutos}")
102     #programa principal
103     try:
104         gpio.setwarnings(False)
105         gpio.setmode(gpio.BOARD)
106         gpio.setup(pin_button,gpio.IN,pull_up_down=gpio.PUD_DOWN)
107         gpio.setup(LED_ENCENDIDO,gpio.OUT)
108         gpio.setup(LED_MQTT,gpio.OUT)
109         gpio.setup(LED_LECTURA,gpio.OUT)
110         while(hayInternet()==False):
111             print("No se encuentra una conexion a Internet.")
112             gpio.output(LED_ENCENDIDO,True)
113             gpio.output(LED_ENCENDIDO,False)
114             print("Reintentando en 5 segundos...")
115             time.sleep(5)
116         print("Conectado a Internet.")
117         gpio.output(LED_ENCENDIDO,True)
118         client = mqtt.Client()
119         client.on_connect = on_connect
120         client.connect("192.168.1.78", 1883, 60)
121         while(1):
122             if(gpio.input(pin_button)==gpio.HIGH):
123                 print("Iniciando lectura:\n")
124                 parpadeo(LED_LECTURA)
125                 print("Temperatura ambiente:"+str(sensor.get_ambient()))
126                 parpadeo(LED_LECTURA)
127                 print("temperatura objeto:"+str(sensor.get_object_1()))
128                 parpadeo(LED_LECTURA)
129                 client.publish('isur/temperatura_salon', payload = sensor.get_ambient(),
130                               qos=0,retain=False)
131                 parpadeo(LED_MQTT)
132                 client.publish('isur/temp',payload = sensor.get_object_1(),qos=0,retain=False)
133                 parpadeo(LED_MQTT)
134                 time.sleep(5)
135             else:
136                 mostrarHora()
137                 print("Presione el boton durante 3 segundos para iniciar la lectura.")
138                 standby(LED_ENCENDIDO,LED_MQTT,LED_LECTURA)
139     except KeyboardInterrupt:
140         print("Finalizando programa.")
141         print(f"Apagando leds GPIO{LED_MQTT},GPIO{LED_ENCENDIDO},GPIO{LED_LECTURA}")
142         gpio.output(LED_ENCENDIDO,False)
143         gpio.output(LED_MQTT,False)
144         gpio.output(LED_LECTURA,False)
145         bus.close()

```

Para más información revisar : [Repositorio MLX90614](#)

2.1.4. MAX30102

Armado del circuito

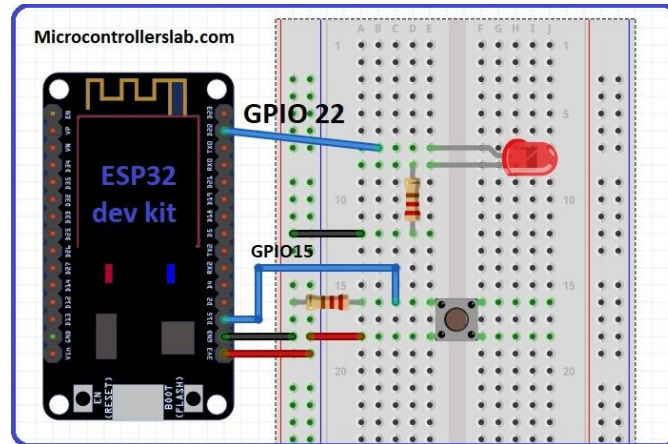


Figura 2.5: Diagrama de conexiones MAX30102

Código

```

1  /*
2  AUTOR: JORGE ISUR BALDERAS RAMÍREZ
3  FECHA: 07-12-2021
4  DESCRIPCIÓN: PROGRAMA PARA MEDIR SINTOMAS DE COVID MEDIANTE EL SENSOR
5  MAX30102
6  CONEXIONES EN HARDWARE(ESP32CAM)
7  -5V = 5V (3.3V también funciona)
8  -GND = GND
9  -SDA = GPIO15
10 -SCL = GPIO14
11 -INT = Not connected
12
13 EL MAX30102 SOPORTA I2C LÓGICA EN 5V Y 3.3V, SE RECOMIENDA ENERGIZAR CON 5V
14 A FIN DE TENER MAYOR PRECISIÓN EN LOS DATOS.
15 */
16 /*****
17 DECLARACIÓN DE BIBLIOTECAS
18 *****/
19 #include <Wire.h> //BIBLIOTECA PARA EL MANEJO DE I2C
20 #include "MAX30105.h" //BIBLIOTECA PARA EL CONTROL DEL MAX3010X
21 #include "spo2_algorithm.h" //BIBLIOTECA PARA EL ALGORITMO QUE CÁLCULA OXIGENO EN SANGRE
22 #include <WiFi.h> // Biblioteca para el control de WiFi
23 #include <PubSubClient.h> //Biblioteca para conexión MQTT
24 /*****
25 DECLARACIÓN DE INSTANCIA DEL SENSOR

```

```

26  *****/
27  MAX30105 particleSensor;
28  /*****/
29  DATOS DEL WIFI
30  *****/
31  const char* ssid = "INFINITUM3033_2.4"; // Aquí debes poner el nombre de tu red
32  const char* password = "yYYmteq554"; // Aquí debes poner la contraseña de tu red
33  /*****/
34  DATOS DEL BROKER MQTT
35  *****/
36  const char* mqtt_server = "187.200.114.169"; // Si estas en una red local, coloca la IP
37  //asignada, en caso contrario, coloca la IP publica
38  IPAddress server(187,200,218,12);
39  /*****/
40  OBJETOS
41  *****/
42  WiFiClient esp32Client; // Este objeto maneja los datos de conexion WiFi
43  PubSubClient client(esp32Client); // Este objeto maneja los datos de conexion al broker
44
45  #define MAX_BRIGHTNESS 255
46
47  #if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
48  uint16_t irBuffer[100]; //infrared LED sensor data
49  uint16_t redBuffer[100]; //red LED sensor data
50  #else
51  uint32_t irBuffer[100]; //infrared LED sensor data
52  uint32_t redBuffer[100]; //red LED sensor data
53  #endif
54
55  int32_t bufferLength; //LONGITUD DE LOS DATOS
56  int32_t spo2; //VALOR DEL OXIGENO EN SANGRE
57  int8_t validSPO2; //INDICADOR PARA MOSTRAR SI EL VALOR DEL SPO2 ES VALIDO
58  int32_t heartRate; //VALOR DE PULSO CARDIACO
59  int8_t validHeartRate; //INDICADOR PARA MOSTRAR SI EL VALOR DEL PULSO CARDIACO ES VALIDO
60  /*****/
61  LEDS y PINES
62  *****/
63  #define LED_OK 12
64  #define LED_CONEXION_WIFI 13
65  #define LED_MQTT 2
66  byte readLED = 33; //PARPADEA CON CADA LECTURA DEL SENSOR
67  #define push_button 4
68
69  void setup()
70  {
71    Serial.begin(115200); // INICIAR LA COMUNICACIÓN SERIAL
72    Serial.println("Iniciando...\n");
73    Serial.println("LECTOR DE SINTOMAS COVID.\n");
74    pinMode(readLED, OUTPUT);
75    pinMode(LED_OK, OUTPUT);
76    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()) {
```

```

128     reconnect(); // En caso de que no haya conexión, ejecutar la función de reconexión
129     // definida despues del void setup ()
130 } // fin del if (!client.connected())
131 Serial.println("Conectado a:");
132 Serial.println(server);
133 client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante
134 //las funciones necesarias para la comunicación con el broker
135 digitalWrite(LED_MQTT,HIGH);
136 digitalWrite(LED_OK,HIGH);
137 int push_button_state = digitalRead(push_button);
138 if(push_button_state==HIGH){
139     Serial.println(F("Coloque el dedo en el sensor y aplique presión.\n"));
140     delay(2000);
141     byte ledBrightness = 60; //Options: 0=Off to 255=50mA
142     byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
143     byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
144     byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
145     int pulseWidth = 411; //Options: 69, 118, 215, 411
146     int adcRange = 4096; //Options: 2048, 4096, 8192, 16384
147     //Configure sensor with these settings
148     particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange);
149     particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.
150     bufferSize = 100; //buffer length of 100 stores 4 seconds of samples running at 25sps
151     //read the first 100 samples, and determine the signal range
152     for (byte i = 0 ; i < bufferSize ; i++){
153         while (particleSensor.available() == false){ //do we have new data?
154             particleSensor.check(); //Check the sensor for new data
155         }
156         redBuffer[i] = particleSensor.getRed();
157         irBuffer[i] = particleSensor.getIR();
158         particleSensor.nextSample(); //We're finished with this sample so move to next sample
159
160         Serial.print(F("red="));
161         Serial.print(redBuffer[i], DEC);
162         Serial.print(F(", ir="));
163         Serial.println(irBuffer[i], DEC);
164     }
165
166     //CALCULAR EL PULSO CARDIACO Y OXIGENACIÓN DESPUES DE 100 MUESTRAS(PRIMEROS 4 SEGUNDOS)
167     maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferSize, redBuffer, &spo2,
168     &validSP02, &heartRate, &validHeartRate);
169     //DESECHANDO LOS PRIMEROS 25 SETS DE MUESTRAS EN MEMORIA Y RECORRIENDO
170     //LAS ULTIMAS 75 MUESTRAS AL PRINCIPIO.
171     for (byte i = 25; i < 100; i++){
172         redBuffer[i - 25] = redBuffer[i];
173         irBuffer[i - 25] = irBuffer[i];
174     }
175     //take 25 sets of samples before calculating the heart rate.
176     for (byte i = 75; i < 100; i++)
177     {
178         while (particleSensor.available() == false){ //¿TENEMOS NUEVAS LECTURAS?

```

```

179     particleSensor.check(); //MONITOREA EL SENSOR POR NUEVOS DATOS.
180 }
181 digitalWrite(readLED, !digitalRead(readLED)); //Blink onboard LED with every data read
182
183 redBuffer[i] = particleSensor.getRed();
184 irBuffer[i] = particleSensor.getIR();
185 particleSensor.nextSample(); //We're finished with this sample so move to next sample
186 Serial.print(F("red="));
187 Serial.print(redBuffer[i], DEC);
188 Serial.print(F(", ir="));
189 Serial.print(irBuffer[i], DEC);
190 }
191 //After gathering 25 new samples recalculate HR and SPO2
192 maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
193 &validSPO2, &heartRate, &validHeartRate);
194 //send samples and calculation result to terminal program through UART
195 Serial.print(F(", PULSO CARDIACO=\n"));
196 Serial.print(heartRate, DEC);
197 char pulso[10];
198 itoa(heartRate,pulso,10);//CONVERTIR INT-->CHAR[] PARA ENVIAR POR MQTT
199
200 //client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante
201 //las funciones necesarias para la comunicación con el broker
202 //client.publish("isur/pulso",pulso);
203 if(!client.publish("isur/pulso",pulso)){
204     Serial.print("No sé pudo publicar en isur/pulso.\n");
205     Serial.print("Error:");
206     Serial.print(client.state()); // Muestra el código de error
207     digitalWrite(LED_OK,HIGH);
208     delay(1000);
209     digitalWrite(LED_OK,LOW);
210     delay(1000);
211     digitalWrite(LED_OK,HIGH);
212     delay(1000);
213     digitalWrite(LED_OK,LOW);
214     delay(2000);
215 }
216 else{
217     Serial.print("Mensaje publicado\n");
218     digitalWrite(LED_MQTT,HIGH);
219     delay(1000);
220     digitalWrite(LED_MQTT,LOW);
221     delay(1000);
222     digitalWrite(LED_MQTT,HIGH);
223     delay(1000);
224     digitalWrite(LED_MQTT,LOW);
225     delay(2000);
226 }
227 Serial.print(F(", PULSO CARDIACO VALIDO=\n"));
228 Serial.print(validHeartRate, DEC);
229

```

```

230     Serial.print(F(", OXIGENO EN SANGRE=\n"));
231     Serial.print(spo2, DEC);
232     char oxigeno[10];
233     itoa(spo2,oxigeno,10);//CONVERTIR INT-->CHAR[] PARA ENVIAR POR MQTT
234     //client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante
235     //las funciones necesarias para la comunicación con el broker
236     //client.publish("isur/oxigeno",oxigeno);
237     if(!client.publish("isur/oxigeno",oxigeno)){
238         Serial.print("No se pudo publicar en isur/oxigeno.\n");
239         Serial.print("Error:");
240         Serial.print(client.state()); // Muestra el código de error
241         digitalWrite(LED_OK,HIGH);
242         delay(1000);
243         digitalWrite(LED_OK,LOW);
244         delay(1000);
245         digitalWrite(LED_OK,HIGH);
246         delay(1000);
247         digitalWrite(LED_OK,LOW);
248         delay(2000);
249     }
250     else{
251         Serial.print("Mensaje publicado\n");
252         digitalWrite(LED_MQTT,HIGH);
253         delay(1000);
254         digitalWrite(LED_MQTT,LOW);
255         delay(1000);
256         digitalWrite(LED_MQTT,HIGH);
257         delay(1000);
258         digitalWrite(LED_MQTT,LOW);
259         delay(2000);
260     }
261     Serial.print(F(", OXIGENO EN SANGRE VALIDO=\n"));
262     Serial.println(validSPO2, DEC);
263     delay(1500);
264 }//FIN IF_BUTTON
265 else{
266     Serial.println("MODULO STANDBY");
267     Serial.println("Presione el boton por 3 segundos para tomar las medidas");
268     digitalWrite(LED_OK,HIGH);
269     digitalWrite(LED_CONEXION_WIFI,LOW);
270     digitalWrite(LED_MQTT,LOW);
271     delay(1000);
272     digitalWrite(LED_OK,HIGH);
273     digitalWrite(LED_CONEXION_WIFI,HIGH);
274     digitalWrite(LED_MQTT,LOW);
275     delay(1000);
276     digitalWrite(LED_OK,HIGH);
277     digitalWrite(LED_CONEXION_WIFI,HIGH);
278     digitalWrite(LED_MQTT,HIGH);
279     delay(1000);
280     digitalWrite(LED_OK,LOW);

```



```
281     digitalWrite(LED_CONEXION_WIFI,LOW);
282     digitalWrite(LED_MQTT,HIGH);
283     delay(1000);
284     digitalWrite(LED_OK,LOW);
285     digitalWrite(LED_CONEXION_WIFI,HIGH);
286     digitalWrite(LED_MQTT,HIGH);
287 }
288 }
289 // Función para reconectarse
290 void reconnect() {
291     // Bucle hasta lograr conexión
292     while (!client.connected()) { // Pregunta si hay conexión
293         digitalWrite(LED_MQTT,LOW);
294         Serial.print("Tratando de conectarse...");
295         // Intentar reconexión
296         digitalWrite(LED_MQTT,HIGH);
297         if (client.connect("ESP32CAMClient")) { //Pregunta por el resultado del intento de conexión
298             Serial.println("Conectado");
299         } // fin del if (client.connect("ESP32CAMClient"))
300         else { //en caso de que la conexión no se logre
301             Serial.print("Conexion fallida, Error rc=");
302             Serial.print(client.state()); // Muestra el codigo de error
303             Serial.println(" Volviendo a intentar en 5 segundos");
304             digitalWrite(LED_MQTT,LOW);
305             delay(500);
306             digitalWrite(LED_MQTT,HIGH);
307             delay(500);
308             digitalWrite(LED_MQTT,LOW);
309             delay(500);
310             digitalWrite(LED_MQTT,HIGH);
311             delay(500);
312             digitalWrite(LED_MQTT,LOW);
313             delay(500);
314             digitalWrite(LED_MQTT,HIGH);
315             delay(500);
316             digitalWrite(LED_MQTT,LOW);
317             delay(500);
318             digitalWrite(LED_MQTT,HIGH);
319             delay(500);
320             digitalWrite(LED_MQTT,LOW);
321             delay(500);
322             digitalWrite(LED_MQTT,HIGH);
323             delay(500);
324             Serial.println (client.connected ()); // Muestra estatus de conexión
325         } // fin del else
326     } // fin del bucle while (!client.connected())
327 } // fin de void reconnect()
```

Para más información revisar: [Repositorio MAX30102](#)

2.1.5. DHT11

Armado del circuito

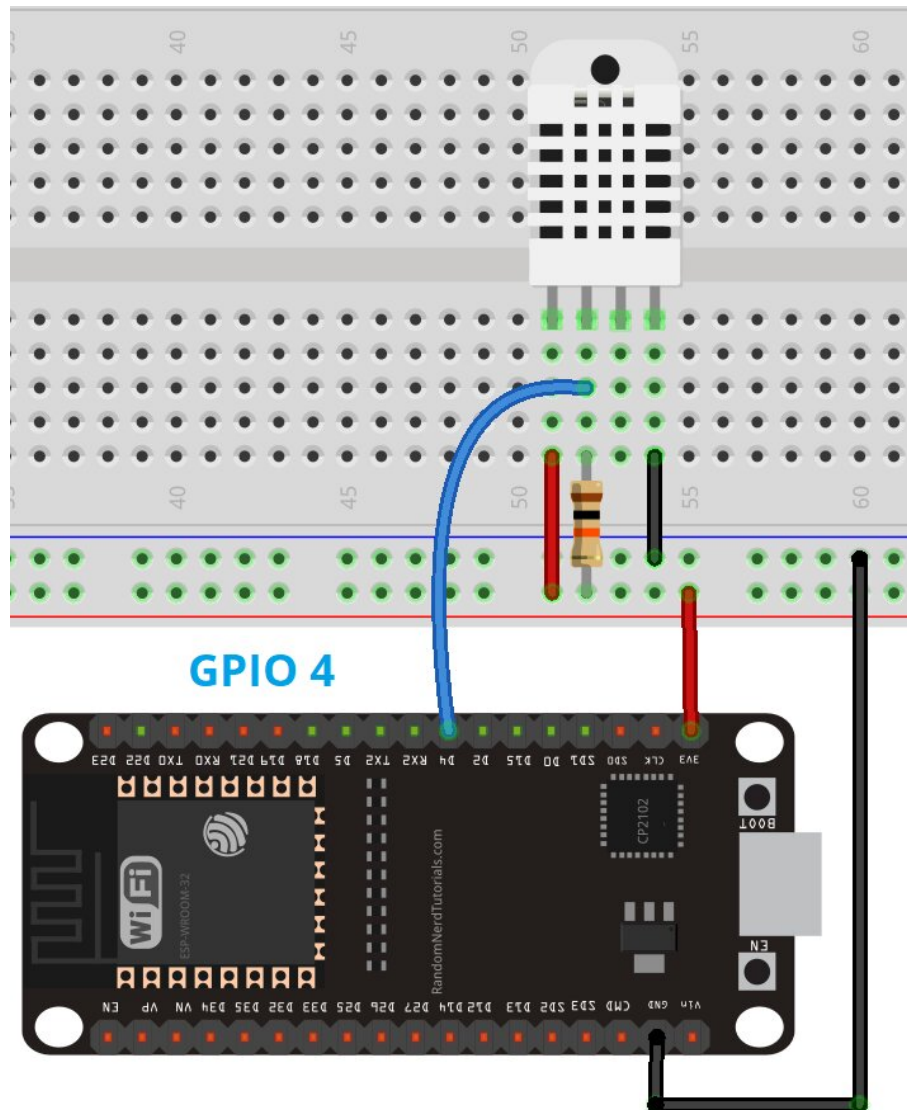


Figura 2.6: Diagrama de conexiones DHT11

Código

```
1  /*  
2  AUTOR: Jorge Isur Balderas Ramírez  
3  FECHA: 25/08/2021  
4  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

```

```

56 while (WiFi.status() != WL_CONNECTED) { // Este bucle espera a que se realice la conexión
57     digitalWrite (LED_STATUS, HIGH);
58     delay(500); //dado que es de suma importancia esperar a la conexión, debe usarse espera bloqueante
59     digitalWrite (LED_STATUS, LOW);
60     Serial.print("."); // Indicador de progreso
61     delay (5);
62 }
63
64 // Cuando se haya logrado la conexión, el programa avanzará, por lo tanto, puede informarse lo siguiente
65 Serial.println();
66 Serial.println("WiFi conectado");
67 Serial.println("Direccion IP: ");
68 Serial.println(WiFi.localIP());
69 delay (1000); // Esta espera es solo una formalidad antes de iniciar la comunicación con el broker
70 // Conexión con el broker MQTT
71 client.setServer(server, 1883); // Conectarse a la IP del broker en el puerto indicado
72 client.setCallback(callback); // Activar función de CallBack, permite recibir mensajes MQTT y ejecutar funciones a p
73 delay(1500); // Esta espera es preventiva, espera a la conexión para no perder información
74 dht.begin();
75 }
76
77 void loop() {
78     //Verificar siempre que haya conexión al broker
79     if (!client.connected()) {
80         reconnect(); // En caso de que no haya conexión, ejecutar la función de reconexión, definida despues del void se
81     } // fin del if (!client.connected())
82     client.loop(); // Esta función es muy importante, ejecuta de manera no bloqueante las funciones necesarias para la c
83     // Wait a few seconds between measurements.
84     delay(5000);
85
86     // Reading temperature or humidity takes about 250 milliseconds!
87     // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
88     float h = dht.readHumidity();
89     // Read temperature as Celsius (the default)
90     float t = dht.readTemperature();
91     // Read temperature as Fahrenheit (isFahrenheit = true)
92     //float f = dht.readTemperature(true);
93     String hum = String(h);
94     String temp = String(t);
95     //conversion a char[]
96     char* humedad = hum.toCharArray();
97     char* celsius = temp.toCharArray();
98     // Check if any reads failed and exit early (to try again).
99     if (isnan(h) || isnan(t)) {
100         Serial.println(F("ERROR AL REGISTRAR LOS DATOS DEL SENSOR!"));
101         return;
102     }
103     if (t<25)
104     {
105         digitalWrite(LED_OK,HIGH);
106         digitalWrite(LED_WARNING,LOW);

```

```
107     digitalWrite(LED_FATAL,LOW);
108 }
109 if (t>=25 && t<30)
110 {
111     digitalWrite(LED_WARNING,HIGH);
112     digitalWrite(LED_FATAL,LOW);
113     digitalWrite(LED_OK,LOW);
114 }
115 else{
116     digitalWrite(LED_OK,LOW);
117     digitalWrite(LED_WARNING,LOW);
118     digitalWrite(LED_FATAL,HIGH);
119 }
120 Serial.print(F("Humedad: "));
121 Serial.print(h);
122 Serial.print(F("% Temperatura: "));
123 Serial.print(t);
124 Serial.print(F("°C \n"));
125 client.publish("isur/humedad",humedad);
126 client.publish("isur/temp",temp);
127 }
128 // Función para reconectarse
129 void reconnect() {
130     // Bucle hasta lograr conexión
131     while (!client.connected()) { // Pregunta si hay conexión
132         Serial.print("Tratando de conectarse...");
133         // Intentar reconexión
134         if (client.connect("ESP32CAMclient")) { //Pregunta por el resultado del intento de conexión
135             Serial.println("Conectado");
136         } // fin del if (client.connect("ESP32CAMclient"))
137         else { //en caso de que la conexión no se logre
138             Serial.print("Conexion fallida, Error rc=");
139             Serial.print(client.state()); // Muestra el código de error
140             Serial.println(" Volviendo a intentar en 5 segundos");
141             // Espera de 5 segundos bloqueante
142             delay(5000);
143             Serial.println (client.connected ()); // Muestra estatus de conexión
144         } // fin del else
145     } // fin del bucle while (!client.connected())
146 } // fin de void reconnect()
```

Para más información revisar : [Repositorio DHT11](#)

2.2. Código de base de datos

Código registroBaseDatos

```
1  """
2  Administrador de base de datos: Registro UAM-LERMA
3  Autor: Jorge Isur Balderas Ramirez
4  Fecha: 13-12-2021
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 miCliente = pymongo.MongoClient("mongodb://localhost:27017/")
12 base_datos = miCliente["UAM"]
13 alumnos = base_datos["alumnos"]
14 admin = base_datos["administrativos"]
15 def on_connect(client,userdata,flags,rc):
16     print(f"Conectado con codigo:{rc}")
17     client.subscribe("isur/uid")
18 def on_message(client,userdata,msg):
19     uid = msg.payload
20     uid = uid.decode()
21     print(f"Tarjeta detectada con UID:{uid}")
22     fecha = dt.datetime.now()
23     hora = dt.datetime.now().hour
24     minuto = dt.datetime.now().minute
25     dia = dt.datetime.now().day
26     mes = dt.datetime.now().month
27     if hora < 10:
28         hora = '0'+str(hora)
29     if minuto <10:
30         minuto = '0'+str(minuto)
31     if mes < 10:
32         mes = '0'+str(mes)
33     if dia < 10:
34         dia = '0'+str(dia)
35     print(f"Fecha:{dia}-{mes}-{fecha.year} {hora}:{minuto}")
36     client.unsubscribe("isur/uid")
37     client.disconnect()
38     menu(uid,fecha)
39 def existeDatabase(db):
40     dblist = miCliente.list_database_names()
41     if db in dblist:
42         print(f"Base de datos {db} encontrada.\n")
43         return True
44     else:
```

```

45     print(f"Base de datos {db} no existente.\n")
46     return False
47     exit()
48 def existeColeccion(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"Coleccion {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 = existeColeccion("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 = existeColeccion("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)

```

```

54     if existe ==False:
55         print("Usuario no existente.")
56     else:
57         print("Usuario encontrado.")
58         print("Datos enviados")
59 def existeDatabase(db):
60     dblist = miCliente.list_database_names()
61     if db in dblist:
62         print(f"Base de busqueda {db} encontrada.\n")
63         return True
64     else:
65         print(f"Base de busqueda {db} no existente.\n")
66         return False
67         exit()
68 def existeColeccion(coleccion):
69     listaColeccion = base_datos.list_collection_names()
70     if coleccion in listaColeccion:
71         print(f"Colección {coleccion} encontrada.\n")
72         return True
73     else:
74         print(f"Coleccion {db} no existente.\n")
75         return False
76         exit()
77 def busqueda(uid,fecha):
78     existe = False
79     print(separador)
80     busqueda = alumnos.find_one(uid)
81     if busqueda !=None:
82         nombre =busqueda["nombre"]
83         print(f"Nombre:{nombre}")
84         _id = busqueda["_id"]
85         print(f"UID:{_id}")
86         carrera = busqueda["info"]
87         print(f"Carrera:{carrera}")
88         matricula = busqueda["matricula"]
89         print(f"Matricula:{matricula}")
90         fecha_entrada = busqueda["fecha_entrada"]
91         print(f"Ultima entrada registrada:{fecha_entrada}")
92         client.publish("isur/ultima_entrada",payload=str(fecha_entrada),qos=0,retain=False)
93         fecha_salida = busqueda["fecha_salida"]
94         print(f"Ultima salida registrada: {fecha_salida}")
95         client.publish("isur/ultima_salida",payload=str(fecha_salida),qos=0,retain=False)
96         estado = entradaSalida(fecha,fecha_entrada,fecha_salida)
97         if estado==True:
98             query = {"nombre":nombre}
99             nuevaFecha = {"$set":{"fecha_entrada":fecha}}
100             actualizar=alumnos.update_one(query,nuevaFecha)
101             print(f"{actualizar.modified_count} registros actualizados")
102         if estado==False:
103             query = {"nombre":nombre}
104             nuevaFecha = {"$set":{"fecha_salida":fecha}}

```

```

105         actualizar = alumnos.update_one(query,nuevaFecha)
106         print(f"{actualizar.modified_count} registros actualizados")
107     print(separador)
108     print("publicando...")
109     client.connect("192.168.1.78", 1883, 60)
110     client.publish("isur/usuario/nombre",payload=nombre,qos=0,retain=False)
111     client.publish("isur/usuario/carrera",payload=carrera,qos=0,retain=False)
112     client.publish("isur/usuario/matricula",payload=matricula,qos=0,retain=False)
113     client.publish("isur/ultima_entrada",payload=str(fecha_entrada),qos=0,retain=False)
114     client.publish("isur/ultima_salida",payload=str(fecha_salida),qos=0,retain=False)
115     return True
116 resultados = admin.find_one(uid)
117 if resultados !=None:
118     nombre =resultados["nombre"]
119     print(f"Nombre:{nombre}")
120     _id = resultados["_id"]
121     print(f"UID:{_id}")
122     carrera = resultados["info"]
123     print(f"Carrera:{carrera}")
124     matricula = resultados["matricula"]
125     print(f"Matricula:{matricula}")
126     fecha_entrada = busqueda["fecha_entrada"]
127     print(f"Ultima entrada registrada:{fecha_entrada}")
128     fecha_salida = busqueda["fecha_salida"]
129     print(f"Ultima salida registrada: {fecha_salida}")
130     estado = entradaSalida(fecha,fecha_entrada,fecha_salida)
131     if estado==True:
132         query = {"nombre":nombre}
133         nuevaFecha = {"$set":{"fecha_entrada":fecha}}
134         actualizar=admin.update_one(query,nuevaFecha)
135         print(f"{actualizar.modified_count} registros actualizados")
136     if estado==False:
137         query = {"nombre":nombre}
138         nuevaFecha = {"$set":{"fecha_salida":fecha}}
139         actualizar = admin.update_one(query,nuevaFecha)
140         print(f"{actualizar.modified_count} registros actualizados")
141     client.connect("192.168.1.78", 1883, 60)
142     client.publish("isur/usuario/nombre",payload=nombre,qos=0,retain=False)
143     client.publish("isur/usuario/carrera",payload=carrera,qos=0,retain=False)
144     client.publish("isur/usuario/matricula",payload=matricula,qos=0,retain=False)
145     return True
146 def entradaSalida(fecha,fecha_entrada,fecha_salida):
147     try:
148         if fecha_entrada > fecha_salida:
149             print(f"Salio:{fecha}")
150             return False
151         if fecha > fecha_salida:
152             print(f"Entro:{fecha}")
153             return True
154     except TypeError:
155         print("Comparacion no permitida.")

```

```
156     print(f"Salio:{fecha}")
157     return False
158 try:
159     client = mqtt.Client(cliente)
160     client.on_connect = on_connect
161     client.on_message = on_message
162     client.connect(broker_ip, port, 60)
163     print("Acerca la tarjeta al lector.")
164     print(f"Conectando a {broker_ip}:{port}")
165     client.loop_forever()
166 except KeyboardInterrupt:
167     print("\n")
168     print("Finalizando programa.")
```

Para más información revisar : [Repositorio mongoDB](#)

2.3. JSON de Node Red

```
1  [
2    {
3      "id": "43d97d32682ec8d6",
4      "type": "tab",
5      "label": "Flow 3",
6      "disabled": false,
7      "info": "",
8      "env": []
9    },
10   {
11     "id": "7bd8e5f4b9cf89c7",
12     "type": "mqtt in",
13     "z": "43d97d32682ec8d6",
14     "name": "temperatura",
15     "topic": "isur/temp",
16     "qos": "2",
17     "datatype": "auto",
18     "broker": "f0de1995277616c8",
19     "nl": false,
20     "rap": true,
21     "rh": 0,
22     "inputs": 0,
23     "x": 180,
24     "y": 80,
25     "wires": [
26       [
27         "3981e5e611260601",
```

```
28         "9846f0581858083d",
29         "8dca63be913f04e3"
30     ]
31 ]
32 },
33 {
34     "id": "8f293b58c5771ec5",
35     "type": "mqtt in",
36     "z": "43d97d32682ec8d6",
37     "name": "oxigeno",
38     "topic": "isur/oxigeno",
39     "qos": "2",
40     "datatype": "auto",
41     "broker": "f0de1995277616c8",
42     "nl": false,
43     "rap": true,
44     "rh": 0,
45     "inputs": 0,
46     "x": 110,
47     "y": 140,
48     "wires": [
49         [
50             "74816a4c1cbff5ac",
51             "06158c3acb377cc1",
52             "bbf8f77038ef22c8"
53         ]
54     ]
55 },
56 {
57     "id": "74816a4c1cbff5ac",
58     "type": "ui_text",
59     "z": "43d97d32682ec8d6",
60     "group": "74f876ad7860a6ce",
61     "order": 1,
62     "width": 0,
63     "height": 0,
64     "name": "",
65     "label": "Oxigenacion",
66     "format": "{{msg.payload}}",
67     "layout": "col-center",
68     "className": "",
69     "x": 350,
70     "y": 120,
71     "wires": []
72 },
73 {
74     "id": "9e01e73598137c09",
75     "type": "mqtt in",
76     "z": "43d97d32682ec8d6",
77     "name": "",
78     "topic": "isur/pulso",
```

```
79     "qos": "2",
80     "datatype": "auto",
81     "broker": "f0de1995277616c8",
82     "nl": false,
83     "rap": true,
84     "rh": 0,
85     "inputs": 0,
86     "x": 100,
87     "y": 280,
88     "wires": [
89       [
90         "7345e8202779e282",
91         "079883ecdd943b79",
92         "ea9246f4b46aee4b"
93       ]
94     ]
95   },
96   {
97     "id": "65f9b70df85f1ff8",
98     "type": "mqtt in",
99     "z": "43d97d32682ec8d6",
100    "name": "Temp Ambiente",
101    "topic": "isur/temperatura_salon",
102    "qos": "2",
103    "datatype": "auto",
104    "broker": "f0de1995277616c8",
105    "nl": false,
106    "rap": true,
107    "rh": 0,
108    "inputs": 0,
109    "x": 120,
110    "y": 340,
111    "wires": [
112      [
113        "f7d803e5b777ce11"
114      ]
115    ]
116  },
117  {
118    "id": "f7d803e5b777ce11",
119    "type": "ui_gauge",
120    "z": "43d97d32682ec8d6",
121    "name": "Ambiente",
122    "group": "09297ea9730f36a6",
123    "order": 1,
124    "width": 5,
125    "height": 3,
126    "gtype": "gage",
127    "title": "Temperatura Ambiente",
128    "label": "°C",
129    "format": "{{value}}",
```

```
130     "min": 0,
131     "max": "40",
132     "colors": [
133         "#00b500",
134         "#e6e600",
135         "#ca3838"
136     ],
137     "seg1": "20",
138     "seg2": "30",
139     "className": "",
140     "x": 360,
141     "y": 340,
142     "wires": []
143 },
144 {
145     "id": "f0cf5563db53c86c",
146     "type": "exec",
147     "z": "43d97d32682ec8d6",
148     "command": "python3 /home/pi/RASPBERRYPI-CODIGOIOT/PyMLX90614-0.0.3/mlx90614.py",
149     "addpay": "",
150     "append": "",
151     "useSpawn": "false",
152     "timer": "",
153     "winHide": false,
154     "oldrc": false,
155     "name": "mlx90614",
156     "x": 460,
157     "y": 460,
158     "wires": [
159         [],
160         [],
161         []
162     ]
163 },
164 {
165     "id": "7caa034a49faa628",
166     "type": "ui_button",
167     "z": "43d97d32682ec8d6",
168     "name": "Monitoreo",
169     "group": "bb21fdcd96e09c93",
170     "order": 1,
171     "width": 0,
172     "height": 0,
173     "passthru": false,
174     "label": "Iniciar monitoreo.",
175     "tooltip": "",
176     "color": "",
177     "bgcolor": "",
178     "className": "",
179     "icon": "",
180     "payload": "",
```

```
181     "payloadType": "str",
182     "topic": "topic",
183     "topicType": "msg",
184     "x": 240,
185     "y": 460,
186     "wires": [
187       [
188         "f0cf5563db53c86c"
189       ]
190     ],
191   },
192   {
193     "id": "3981e5e611260601",
194     "type": "ui_gauge",
195     "z": "43d97d32682ec8d6",
196     "name": "Corporal",
197     "group": "74f876ad7860a6ce",
198     "order": 3,
199     "width": 6,
200     "height": 3,
201     "gtype": "gage",
202     "title": "Temperatura corporal",
203     "label": "°C",
204     "format": "{{value}}",
205     "min": 0,
206     "max": "42",
207     "colors": [
208       "#00b500",
209       "#e6e600",
210       "#ca3838"
211     ],
212     "seg1": "36",
213     "seg2": "38",
214     "className": "",
215     "x": 520,
216     "y": 100,
217     "wires": []
218   },
219   {
220     "id": "467d246c3f24d21b",
221     "type": "mqtt in",
222     "z": "43d97d32682ec8d6",
223     "name": "",
224     "topic": "isur/uid",
225     "qos": "2",
226     "datatype": "auto",
227     "broker": "f0de1995277616c8",
228     "nl": false,
229     "rap": true,
230     "rh": 0,
231     "inputs": 0,
```



```
232     "x": 70,  
233     "y": 520,  
234     "wires": [  
235         [  
236             "d1d6ce5e49c3d7f6",  
237             "a160d5d997580a18"  
238         ]  
239     ]  
240 },  
241 {  
242     "id": "f8597d449a3897cc",  
243     "type": "ui_text",  
244     "z": "43d97d32682ec8d6",  
245     "group": "b4bde563647d3018",  
246     "order": 1,  
247     "width": 0,  
248     "height": 0,  
249     "name": "",  
250     "label": "Nombre",  
251     "format": "{{msg.payload}}",  
252     "layout": "row-spread",  
253     "className": "",  
254     "x": 460,  
255     "y": 580,  
256     "wires": []  
257 },  
258 {  
259     "id": "5ac4be9b513bf859",  
260     "type": "mqtt in",  
261     "z": "43d97d32682ec8d6",  
262     "name": "isur/usuario/nombre",  
263     "topic": "isur/usuario/nombre",  
264     "qos": "2",  
265     "datatype": "auto",  
266     "broker": "f0de1995277616c8",  
267     "nl": false,  
268     "rap": true,  
269     "rh": 0,  
270     "inputs": 0,  
271     "x": 110,  
272     "y": 580,  
273     "wires": [  
274         [  
275             "f8597d449a3897cc"  
276         ]  
277     ]  
278 },  
279 {  
280     "id": "981f24d2f4a9d8e4",  
281     "type": "mqtt in",  
282     "z": "43d97d32682ec8d6",
```

```
283     "name": "isur/usuario/matricula",
284     "topic": "isur/usuario/matricula",
285     "qos": "2",
286     "datatype": "auto",
287     "broker": "f0de1995277616c8",
288     "nl": false,
289     "rap": true,
290     "rh": 0,
291     "inputs": 0,
292     "x": 120,
293     "y": 640,
294     "wires": [
295       [
296         "2c2d61faaba5f17f"
297       ]
298     ]
299   },
300   {
301     "id": "876923e57eb25aaa",
302     "type": "mqtt in",
303     "z": "43d97d32682ec8d6",
304     "name": "",
305     "topic": "isur/usuario/carrera",
306     "qos": "2",
307     "datatype": "auto",
308     "broker": "f0de1995277616c8",
309     "nl": false,
310     "rap": true,
311     "rh": 0,
312     "inputs": 0,
313     "x": 110,
314     "y": 700,
315     "wires": [
316       [
317         "f5bb69bb47bbddc6"
318       ]
319     ]
320   },
321   {
322     "id": "1fce39afb392b0b6",
323     "type": "mqtt in",
324     "z": "43d97d32682ec8d6",
325     "name": "",
326     "topic": "isur/estado",
327     "qos": "2",
328     "datatype": "auto",
329     "broker": "f0de1995277616c8",
330     "nl": false,
331     "rap": true,
332     "rh": 0,
333     "inputs": 0,
```

```
334     "x": 110,
335     "y": 780,
336     "wires": [
337         [
338             "950251360d683c14",
339             "731d6edef7957b71"
340         ]
341     ]
342 },
343 {
344     "id": "f52dbb6b6dd07805",
345     "type": "mqtt in",
346     "z": "43d97d32682ec8d6",
347     "name": "",
348     "topic": "isur/puerta",
349     "qos": "2",
350     "datatype": "auto",
351     "broker": "f0de1995277616c8",
352     "nl": false,
353     "rap": true,
354     "rh": 0,
355     "inputs": 0,
356     "x": 100,
357     "y": 900,
358     "wires": [
359         [
360             "9d54ecc6fe5d2260",
361             "faaf816797493bd7"
362         ]
363     ]
364 },
365 {
366     "id": "8d3d72763908f12a",
367     "type": "ui_switch",
368     "z": "43d97d32682ec8d6",
369     "name": "",
370     "label": "Puerta",
371     "tooltip": "",
372     "group": "bb21fdcd96e09c93",
373     "order": 4,
374     "width": 0,
375     "height": 0,
376     "passthru": true,
377     "decouple": "false",
378     "topic": "topic",
379     "topicType": "msg",
380     "style": "",
381     "onvalue": "true",
382     "onvalueType": "bool",
383     "onicon": "",
384     "oncolor": "",
```

```
385     "offvalue": "false",
386     "offvalueType": "bool",
387     "officon": "",
388     "offcolor": "",
389     "animate": false,
390     "className": "",
391     "x": 630,
392     "y": 900,
393     "wires": [
394       []
395     ]
396   },
397   {
398     "id": "950251360d683c14",
399     "type": "ui_text",
400     "z": "43d97d32682ec8d6",
401     "group": "b4bde563647d3018",
402     "order": 7,
403     "width": 0,
404     "height": 0,
405     "name": "",
406     "label": "Estado del usuario",
407     "format": "{{msg.payload}}",
408     "layout": "col-center",
409     "className": "",
410     "x": 370,
411     "y": 780,
412     "wires": []
413   },
414   {
415     "id": "f5bb69bb47bbddc6",
416     "type": "ui_text",
417     "z": "43d97d32682ec8d6",
418     "group": "b4bde563647d3018",
419     "order": 2,
420     "width": 0,
421     "height": 0,
422     "name": "",
423     "label": "Carrera",
424     "format": "{{msg.payload}}",
425     "layout": "row-spread",
426     "className": "",
427     "x": 390,
428     "y": 700,
429     "wires": []
430   },
431   {
432     "id": "2c2d61faaba5f17f",
433     "type": "ui_text",
434     "z": "43d97d32682ec8d6",
435     "group": "b4bde563647d3018",
```

```
436     "order": 3,
437     "width": 0,
438     "height": 0,
439     "name": "",
440     "label": "Matricula",
441     "format": "{{msg.payload}}",
442     "layout": "row-spread",
443     "className": "",
444     "x": 420,
445     "y": 640,
446     "wires": []
447   },
448   {
449     "id": "a160d5d997580a18",
450     "type": "ui_text",
451     "z": "43d97d32682ec8d6",
452     "group": "b4bde563647d3018",
453     "order": 4,
454     "width": 0,
455     "height": 0,
456     "name": "",
457     "label": "UID",
458     "format": "{{msg.payload}}",
459     "layout": "row-spread",
460     "className": "",
461     "x": 590,
462     "y": 520,
463     "wires": []
464   },
465   {
466     "id": "d1d6ce5e49c3d7f6",
467     "type": "trigger",
468     "z": "43d97d32682ec8d6",
469     "name": "",
470     "op1": "true",
471     "op2": "true",
472     "op1type": "bool",
473     "op2type": "bool",
474     "duration": "0",
475     "extend": false,
476     "overrideDelay": false,
477     "units": "ms",
478     "reset": "",
479     "bytopic": "all",
480     "topic": "topic",
481     "outputs": 1,
482     "x": 240,
483     "y": 400,
484     "wires": [
485       [
486         "5df1d6da36915ea6"
```

```
487     ]
488   ]
489 },
490 {
491   "id": "5df1d6da36915ea6",
492   "type": "ui_switch",
493   "z": "43d97d32682ec8d6",
494   "name": "",
495   "label": "Tarjeta",
496   "tooltip": "",
497   "group": "bb21fdcd96e09c93",
498   "order": 3,
499   "width": 0,
500   "height": 0,
501   "passthru": true,
502   "decouple": "false",
503   "topic": "topic",
504   "topicType": "msg",
505   "style": "",
506   "onvalue": "true",
507   "onvalueType": "bool",
508   "onicon": "",
509   "oncolor": "",
510   "offvalue": "false",
511   "offvalueType": "bool",
512   "officon": "",
513   "offcolor": "",
514   "animate": false,
515   "className": "",
516   "x": 450,
517   "y": 400,
518   "wires": [
519     []
520   ]
521 },
522 {
523   "id": "f398a28905529942",
524   "type": "ui_switch",
525   "z": "43d97d32682ec8d6",
526   "name": "",
527   "label": "Temperatura",
528   "tooltip": "",
529   "group": "d9873314e1d61643",
530   "order": 3,
531   "width": 0,
532   "height": 0,
533   "passthru": true,
534   "decouple": "false",
535   "topic": "topic",
536   "topicType": "msg",
537   "style": "",
```

```
538     "onvalue": "true",
539     "onvalueType": "bool",
540     "onicon": "",
541     "oncolor": "",
542     "offvalue": "false",
543     "offvalueType": "bool",
544     "officon": "",
545     "offcolor": "",
546     "animate": false,
547     "className": "",
548     "x": 650,
549     "y": 20,
550     "wires": [
551       []
552     ]
553   },
554   {
555     "id": "9846f0581858083d",
556     "type": "trigger",
557     "z": "43d97d32682ec8d6",
558     "name": "",
559     "op1": "true",
560     "op2": "0",
561     "op1type": "bool",
562     "op2type": "str",
563     "duration": "0",
564     "extend": false,
565     "overrideDelay": false,
566     "units": "ms",
567     "reset": "",
568     "bytopic": "all",
569     "topic": "topic",
570     "outputs": 1,
571     "x": 420,
572     "y": 20,
573     "wires": [
574       [
575         "f398a28905529942"
576       ]
577     ]
578   },
579   {
580     "id": "06158c3acb377cc1",
581     "type": "trigger",
582     "z": "43d97d32682ec8d6",
583     "name": "",
584     "op1": "true",
585     "op2": "0",
586     "op1type": "bool",
587     "op2type": "str",
588     "duration": "0",
```

```
589     "extend": false,
590     "overrideDelay": false,
591     "units": "ms",
592     "reset": "",
593     "bytopic": "all",
594     "topic": "topic",
595     "outputs": 1,
596     "x": 360,
597     "y": 160,
598     "wires": [
599       [
600         "977915fd173980bd"
601       ]
602     ]
603   },
604   {
605     "id": "977915fd173980bd",
606     "type": "ui_switch",
607     "z": "43d97d32682ec8d6",
608     "name": "",
609     "label": "Oxigeno en sangre",
610     "tooltip": "",
611     "group": "d9873314e1d61643",
612     "order": 2,
613     "width": 0,
614     "height": 0,
615     "passthru": true,
616     "decouple": "false",
617     "topic": "topic",
618     "topicType": "msg",
619     "style": "",
620     "onvalue": "true",
621     "onvalueType": "bool",
622     "onicon": "",
623     "oncolor": "",
624     "offvalue": "false",
625     "offvalueType": "bool",
626     "officon": "",
627     "offcolor": "",
628     "animate": false,
629     "className": "",
630     "x": 640,
631     "y": 140,
632     "wires": [
633       []
634     ]
635   },
636   {
637     "id": "7345e8202779e282",
638     "type": "trigger",
639     "z": "43d97d32682ec8d6",
```



```
640     "name": "",
641     "op1": "true",
642     "op2": "0",
643     "op1type": "bool",
644     "op2type": "str",
645     "duration": "0",
646     "extend": false,
647     "overrideDelay": false,
648     "units": "ms",
649     "reset": "",
650     "bytopic": "all",
651     "topic": "topic",
652     "outputs": 1,
653     "x": 340,
654     "y": 300,
655     "wires": [
656       [
657         "e1626b421615b47f"
658       ]
659     ],
660   },
661   {
662     "id": "e1626b421615b47f",
663     "type": "ui_switch",
664     "z": "43d97d32682ec8d6",
665     "name": "",
666     "label": "Pulso cardiaco",
667     "tooltip": "",
668     "group": "d9873314e1d61643",
669     "order": 1,
670     "width": 0,
671     "height": 0,
672     "passthru": true,
673     "decouple": "false",
674     "topic": "topic",
675     "topicType": "msg",
676     "style": "",
677     "onvalue": "true",
678     "onvalueType": "bool",
679     "onicon": "",
680     "oncolor": "",
681     "offvalue": "false",
682     "offvalueType": "bool",
683     "officon": "",
684     "offcolor": "",
685     "animate": false,
686     "className": "",
687     "x": 600,
688     "y": 240,
689     "wires": [
690       []
```

```
691     ]
692   },
693   {
694     "id": "079883ecdd943b79",
695     "type": "ui_text",
696     "z": "43d97d32682ec8d6",
697     "group": "74f876ad7860a6ce",
698     "order": 2,
699     "width": 0,
700     "height": 0,
701     "name": "",
702     "label": "Pulso cardiaco",
703     "format": "{msg.payload}",
704     "layout": "col-center",
705     "className": "",
706     "x": 340,
707     "y": 260,
708     "wires": []
709   },
710   {
711     "id": "8dca63be913f04e3",
712     "type": "switch",
713     "z": "43d97d32682ec8d6",
714     "name": "",
715     "property": "payload",
716     "propertyType": "msg",
717     "rules": [
718       {
719         "t": "gt",
720         "v": "37",
721         "vt": "num"
722       },
723       {
724         "t": "lt",
725         "v": "37",
726         "vt": "num"
727       }
728     ],
729     "checkall": "true",
730     "repair": false,
731     "outputs": 2,
732     "x": 490,
733     "y": 60,
734     "wires": [
735       [
736         "2bb3a6a8ae4e14a2"
737       ],
738       [
739         "b7ceedf0d5f7c56b"
740       ]
741     ]
742   }
743 ]
```

```
742 },
743 {
744   "id": "e55b0732936264b7",
745   "type": "mqtt out",
746   "z": "43d97d32682ec8d6",
747   "name": "",
748   "topic": "isur/estado",
749   "qos": "0",
750   "retain": "false",
751   "respTopic": "",
752   "contentType": "",
753   "userProps": "",
754   "correl": "",
755   "expiry": "",
756   "broker": "f0de1995277616c8",
757   "x": 1210,
758   "y": 160,
759   "wires": []
760 },
761 {
762   "id": "2bb3a6a8ae4e14a2",
763   "type": "change",
764   "z": "43d97d32682ec8d6",
765   "name": "enfermo",
766   "rules": [
767     {
768       "t": "set",
769       "p": "payload",
770       "pt": "msg",
771       "to": "Enfermo",
772       "tot": "str"
773     }
774   ],
775   "action": "",
776   "property": "",
777   "from": "",
778   "to": "",
779   "reg": false,
780   "x": 960,
781   "y": 160,
782   "wires": [
783     [
784       "e55b0732936264b7"
785     ]
786   ],
787 },
788 {
789   "id": "ea9246f4b46aee4b",
790   "type": "switch",
791   "z": "43d97d32682ec8d6",
792   "name": "",
```

```
793     "property": "payload",
794     "propertyType": "msg",
795     "rules": [
796       {
797         "t": "gt",
798         "v": "100",
799         "vt": "num"
800       },
801       {
802         "t": "lte",
803         "v": "100",
804         "vt": "num"
805       }
806     ],
807     "checkall": "true",
808     "repair": false,
809     "outputs": 2,
810     "x": 590,
811     "y": 280,
812     "wires": [
813       [
814         "2bb3a6a8ae4e14a2"
815       ],
816       [
817         "b7ceedf0d5f7c56b"
818       ]
819     ]
820   },
821   {
822     "id": "bbf8f77038ef22c8",
823     "type": "switch",
824     "z": "43d97d32682ec8d6",
825     "name": "",
826     "property": "payload",
827     "propertyType": "msg",
828     "rules": [
829       {
830         "t": "lt",
831         "v": "90",
832         "vt": "num"
833       },
834       {
835         "t": "gt",
836         "v": "90",
837         "vt": "num"
838       }
839     ],
840     "checkall": "true",
841     "repair": false,
842     "outputs": 2,
843     "x": 350,
```

```
844     "y": 200,
845     "wires": [
846       [
847         "2bb3a6a8ae4e14a2"
848       ],
849       [
850         "b7ceedf0d5f7c56b"
851       ]
852     ]
853   },
854   {
855     "id": "731d6edef7957b71",
856     "type": "switch",
857     "z": "43d97d32682ec8d6",
858     "name": "",
859     "property": "payload",
860     "propertyType": "msg",
861     "rules": [
862       {
863         "t": "eq",
864         "v": "Enfermo",
865         "vt": "str"
866       },
867       {
868         "t": "eq",
869         "v": "Sano",
870         "vt": "str"
871       }
872     ],
873     "checkall": "true",
874     "repair": false,
875     "outputs": 2,
876     "x": 290,
877     "y": 820,
878     "wires": [
879       [
880         "e623d4d8e909e143"
881       ],
882       [
883         "e5d99a330ebf8dde"
884       ]
885     ]
886   },
887   {
888     "id": "e5d99a330ebf8dde",
889     "type": "change",
890     "z": "43d97d32682ec8d6",
891     "name": "Sano",
892     "rules": [
893       {
894         "t": "set",
```

```
895         "p": "payload",
896         "pt": "msg",
897         "to": "true",
898         "tot": "bool"
899     }
900 ],
901     "action": "",
902     "property": "",
903     "from": "",
904     "to": "",
905     "reg": false,
906     "x": 510,
907     "y": 840,
908     "wires": [
909         [
910             "18c2dcc52e4e2454"
911         ]
912     ]
913 },
914 {
915     "id": "18c2dcc52e4e2454",
916     "type": "mqtt out",
917     "z": "43d97d32682ec8d6",
918     "name": "abrir",
919     "topic": "isur/puerta",
920     "qos": "0",
921     "retain": "false",
922     "respTopic": "",
923     "contentType": "",
924     "userProps": "",
925     "correl": "",
926     "expiry": "",
927     "broker": "f0de1995277616c8",
928     "x": 750,
929     "y": 820,
930     "wires": []
931 },
932 {
933     "id": "9d54ecc6fe5d2260",
934     "type": "debug",
935     "z": "43d97d32682ec8d6",
936     "name": "",
937     "active": true,
938     "tosidebar": true,
939     "console": false,
940     "tostatus": false,
941     "complete": "false",
942     "statusVal": "",
943     "statusType": "auto",
944     "x": 280,
945     "y": 960,
```

```
946     "wires": []
947   },
948   {
949     "id": "b7ceedf0d5f7c56b",
950     "type": "change",
951     "z": "43d97d32682ec8d6",
952     "name": "sano",
953     "rules": [
954       {
955         "t": "set",
956         "p": "payload",
957         "pt": "msg",
958         "to": "Sano",
959         "tot": "str"
960       }
961     ],
962     "action": "",
963     "property": "",
964     "from": "",
965     "to": "",
966     "reg": false,
967     "x": 950,
968     "y": 200,
969     "wires": [
970       [
971         "e55b0732936264b7"
972       ]
973     ]
974   },
975   {
976     "id": "e623d4d8e909e143",
977     "type": "change",
978     "z": "43d97d32682ec8d6",
979     "name": "Enfermo",
980     "rules": [
981       {
982         "t": "set",
983         "p": "payload",
984         "pt": "msg",
985         "to": "false",
986         "tot": "bool"
987       }
988     ],
989     "action": "",
990     "property": "",
991     "from": "",
992     "to": "",
993     "reg": false,
994     "x": 540,
995     "y": 800,
996     "wires": [
```

```
997         [
998             "18c2dcc52e4e2454"
999         ]
1000     ],
1001 },
1002 {
1003     "id": "faaf816797493bd7",
1004     "type": "switch",
1005     "z": "43d97d32682ec8d6",
1006     "name": "",
1007     "property": "payload",
1008     "propertyType": "msg",
1009     "rules": [
1010         {
1011             "t": "eq",
1012             "v": "true",
1013             "vt": "str"
1014         },
1015         {
1016             "t": "eq",
1017             "v": "false",
1018             "vt": "str"
1019         }
1020     ],
1021     "checkall": "true",
1022     "repair": false,
1023     "outputs": 2,
1024     "x": 260,
1025     "y": 900,
1026     "wires": [
1027         [
1028             "0fc559eabb6eddf9"
1029         ],
1030         [
1031             "9dd2375e7f2a9618"
1032         ]
1033     ]
1034 },
1035 {
1036     "id": "0fc559eabb6eddf9",
1037     "type": "change",
1038     "z": "43d97d32682ec8d6",
1039     "name": "Abrir",
1040     "rules": [
1041         {
1042             "t": "set",
1043             "p": "payload",
1044             "pt": "msg",
1045             "to": "true",
1046             "tot": "bool"
1047         }
1048     ]
1049 }
```



```
1048     ],
1049     "action": "",
1050     "property": "",
1051     "from": "",
1052     "to": "",
1053     "reg": false,
1054     "x": 450,
1055     "y": 940,
1056     "wires": [
1057       [
1058         "8d3d72763908f12a"
1059       ]
1060     ]
1061   },
1062   {
1063     "id": "9dd2375e7f2a9618",
1064     "type": "change",
1065     "z": "43d97d32682ec8d6",
1066     "name": "cerrar",
1067     "rules": [
1068       {
1069         "t": "set",
1070         "p": "payload",
1071         "pt": "msg",
1072         "to": "false",
1073         "tot": "bool"
1074       }
1075     ],
1076     "action": "",
1077     "property": "",
1078     "from": "",
1079     "to": "",
1080     "reg": false,
1081     "x": 450,
1082     "y": 900,
1083     "wires": [
1084       [
1085         "8d3d72763908f12a"
1086       ]
1087     ]
1088   },
1089   {
1090     "id": "2fa8af79c6879149",
1091     "type": "mqtt in",
1092     "z": "43d97d32682ec8d6",
1093     "name": "",
1094     "topic": "isur/humedad",
1095     "qos": "2",
1096     "datatype": "auto",
1097     "broker": "f0de1995277616c8",
1098     "nl": false,
```

```
1099     "rap": true,
1100     "rh": 0,
1101     "inputs": 0,
1102     "x": 110,
1103     "y": 1040,
1104     "wires": [
1105       [
1106         "3c921acc96e87efa"
1107       ]
1108     ]
1109   },
1110   {
1111     "id": "3c921acc96e87efa",
1112     "type": "ui_gauge",
1113     "z": "43d97d32682ec8d6",
1114     "name": "",
1115     "group": "09297ea9730f36a6",
1116     "order": 5,
1117     "width": 5,
1118     "height": 3,
1119     "gtype": "gage",
1120     "title": "Humedad",
1121     "label": "%",
1122     "format": "{{value}}",
1123     "min": 0,
1124     "max": "100",
1125     "colors": [
1126       "#00b500",
1127       "#e6e600",
1128       "#ca3838"
1129     ],
1130     "seg1": "40",
1131     "seg2": "80",
1132     "className": "",
1133     "x": 330,
1134     "y": 1040,
1135     "wires": []
1136   },
1137   {
1138     "id": "844f769db521d280",
1139     "type": "mqtt out",
1140     "z": "43d97d32682ec8d6",
1141     "name": "",
1142     "topic": "isur/puerta",
1143     "qos": "0",
1144     "retain": "false",
1145     "respTopic": "",
1146     "contentType": "",
1147     "userProps": "",
1148     "correl": "",
1149     "expiry": "",
```

```

1150     "broker": "f0de1995277616c8",
1151     "x": 620,
1152     "y": 1080,
1153     "wires": []
1154 },
1155 {
1156     "id": "00384d6419662701",
1157     "type": "ui_switch",
1158     "z": "43d97d32682ec8d6",
1159     "name": "",
1160     "label": "Apertura/Clausura manual",
1161     "tooltip": "",
1162     "group": "bb21fdcd96e09c93",
1163     "order": 5,
1164     "width": 0,
1165     "height": 0,
1166     "passthru": true,
1167     "decouple": "false",
1168     "topic": "topic",
1169     "topicType": "msg",
1170     "style": "",
1171     "onvalue": "true",
1172     "onvalueType": "bool",
1173     "onicon": "",
1174     "oncolor": "",
1175     "offvalue": "false",
1176     "offvalueType": "bool",
1177     "officon": "",
1178     "offcolor": "",
1179     "animate": false,
1180     "className": "",
1181     "x": 310,
1182     "y": 1080,
1183     "wires": [
1184         [
1185             "844f769db521d280"
1186         ]
1187     ]
1188 },
1189 {
1190     "id": "b61c3a903c2ca7bb",
1191     "type": "mqtt in",
1192     "z": "43d97d32682ec8d6",
1193     "name": "",
1194     "topic": "isur/ultima_entrada",
1195     "qos": "2",
1196     "datatype": "auto",
1197     "broker": "f0de1995277616c8",
1198     "nl": false,
1199     "rap": true,
1200     "rh": 0,

```

```
1201     "inputs": 0,
1202     "x": 130,
1203     "y": 1280,
1204     "wires": [
1205         [
1206             "548ed3df86ed4e47",
1207             "c47799459c20319c"
1208         ]
1209     ]
1210 },
1211 {
1212     "id": "548ed3df86ed4e47",
1213     "type": "ui_text",
1214     "z": "43d97d32682ec8d6",
1215     "group": "b4bde563647d3018",
1216     "order": 5,
1217     "width": 0,
1218     "height": 0,
1219     "name": "",
1220     "label": "Ultima entrada registrada",
1221     "format": "{{msg.payload}}",
1222     "layout": "col-center",
1223     "className": "",
1224     "x": 380,
1225     "y": 1280,
1226     "wires": []
1227 },
1228 {
1229     "id": "bd7c16ddfd090404",
1230     "type": "mqtt in",
1231     "z": "43d97d32682ec8d6",
1232     "name": "",
1233     "topic": "isur/ultima_salida",
1234     "qos": "2",
1235     "datatype": "auto",
1236     "broker": "f0de1995277616c8",
1237     "nl": false,
1238     "rap": true,
1239     "rh": 0,
1240     "inputs": 0,
1241     "x": 120,
1242     "y": 1340,
1243     "wires": [
1244         [
1245             "7ee58748a70570c3",
1246             "c47799459c20319c"
1247         ]
1248     ]
1249 },
1250 {
1251     "id": "7ee58748a70570c3",
```

```

1252     "type": "ui_text",
1253     "z": "43d97d32682ec8d6",
1254     "group": "b4bde563647d3018",
1255     "order": 6,
1256     "width": 0,
1257     "height": 0,
1258     "name": "",
1259     "label": "Ultima salida registrada",
1260     "format": "{{msg.payload}}",
1261     "layout": "col-center",
1262     "className": "",
1263     "x": 380,
1264     "y": 1340,
1265     "wires": []
1266   },
1267   {
1268     "id": "c47799459c20319c",
1269     "type": "debug",
1270     "z": "43d97d32682ec8d6",
1271     "name": "",
1272     "active": true,
1273     "tosidebar": true,
1274     "console": false,
1275     "tostatus": false,
1276     "complete": "false",
1277     "statusVal": "",
1278     "statusType": "auto",
1279     "x": 380,
1280     "y": 1420,
1281     "wires": []
1282   },
1283   {
1284     "id": "3d38033a8349b6f3",
1285     "type": "ui_text",
1286     "z": "43d97d32682ec8d6",
1287     "group": "bb21fdcd96e09c93",
1288     "order": 2,
1289     "width": "4",
1290     "height": "2",
1291     "name": "Lectura",
1292     "label": "Informacion",
1293     "format": "Presione el boton fisico durante 3 segundos, despues de iniciar monitoreo.",
1294     "layout": "col-center",
1295     "className": "",
1296     "x": 650,
1297     "y": 680,
1298     "wires": []
1299   },
1300   {
1301     "id": "ec4e1837fb1a3c60",
1302     "type": "ui_spacer",

```

```
1303     "z": "43d97d32682ec8d6",
1304     "name": "spacer",
1305     "group": "09297ea9730f36a6",
1306     "order": 2,
1307     "width": 2,
1308     "height": 1
1309   },
1310   {
1311     "id": "b56d4159134459fd",
1312     "type": "ui_spacer",
1313     "z": "43d97d32682ec8d6",
1314     "name": "spacer",
1315     "group": "09297ea9730f36a6",
1316     "order": 3,
1317     "width": 2,
1318     "height": 1
1319   },
1320   {
1321     "id": "415d2d09ed0dae53",
1322     "type": "ui_spacer",
1323     "z": "43d97d32682ec8d6",
1324     "name": "spacer",
1325     "group": "09297ea9730f36a6",
1326     "order": 4,
1327     "width": 2,
1328     "height": 1
1329   },
1330   {
1331     "id": "2f366e582e4f1e2e",
1332     "type": "ui_spacer",
1333     "z": "43d97d32682ec8d6",
1334     "name": "spacer",
1335     "group": "09297ea9730f36a6",
1336     "order": 6,
1337     "width": 2,
1338     "height": 1
1339   },
1340   {
1341     "id": "40d6337eeb411d45",
1342     "type": "ui_spacer",
1343     "z": "43d97d32682ec8d6",
1344     "name": "spacer",
1345     "group": "09297ea9730f36a6",
1346     "order": 7,
1347     "width": 2,
1348     "height": 1
1349   },
1350   {
1351     "id": "83d3ad6b88077d76",
1352     "type": "ui_spacer",
1353     "z": "43d97d32682ec8d6",
```

```
1354     "name": "spacer",
1355     "group": "09297ea9730f36a6",
1356     "order": 8,
1357     "width": 2,
1358     "height": 1
1359   },
1360   {
1361     "id": "f0de1995277616c8",
1362     "type": "mqtt-broker",
1363     "name": "broker isur",
1364     "broker": "192.168.1.78",
1365     "port": "1883",
1366     "clientid": "",
1367     "autoConnect": true,
1368     "usetls": false,
1369     "protocolVersion": "4",
1370     "keepalive": "60",
1371     "cleansession": true,
1372     "birthTopic": "",
1373     "birthQos": "0",
1374     "birthPayload": "",
1375     "birthMsg": {},
1376     "closeTopic": "",
1377     "closeQos": "0",
1378     "closePayload": "",
1379     "closeMsg": {},
1380     "willTopic": "",
1381     "willQos": "0",
1382     "willPayload": "",
1383     "willMsg": {},
1384     "sessionExpiry": ""
1385   },
1386   {
1387     "id": "74f876ad7860a6ce",
1388     "type": "ui_group",
1389     "name": "Signos vitales",
1390     "tab": "030f73c40792556d",
1391     "order": 3,
1392     "disp": true,
1393     "width": "6",
1394     "collapse": false,
1395     "className": ""
1396   },
1397   {
1398     "id": "09297ea9730f36a6",
1399     "type": "ui_group",
1400     "name": "Salon",
1401     "tab": "030f73c40792556d",
1402     "order": 1,
1403     "disp": true,
1404     "width": 7,
```

```
1405     "collapse": false,
1406     "className": ""
1407   },
1408   {
1409     "id": "bb21fdcd96e09c93",
1410     "type": "ui_group",
1411     "name": "Control",
1412     "tab": "030f73c40792556d",
1413     "order": 2,
1414     "disp": true,
1415     "width": 6,
1416     "collapse": false,
1417     "className": ""
1418   },
1419   {
1420     "id": "b4bde563647d3018",
1421     "type": "ui_group",
1422     "name": "Usuario",
1423     "tab": "030f73c40792556d",
1424     "order": 5,
1425     "disp": true,
1426     "width": "6",
1427     "collapse": false,
1428     "className": ""
1429   },
1430   {
1431     "id": "d9873314e1d61643",
1432     "type": "ui_group",
1433     "name": "Lectura de sintomas",
1434     "tab": "030f73c40792556d",
1435     "order": 4,
1436     "disp": true,
1437     "width": "6",
1438     "collapse": false,
1439     "className": ""
1440   },
1441   {
1442     "id": "030f73c40792556d",
1443     "type": "ui_tab",
1444     "name": "Centro de control",
1445     "icon": "dashboard",
1446     "disabled": false,
1447     "hidden": false
1448   }
1449 ]
```

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.