

**SISTEMAS EMBEBIDOS II**

**PROYECTO**

**Airpurifier IOT**

**Maestro:**

Luis Gerardo Garza Garza

**Integrantes:**

Arturo Ortiz Moncada 1686487

**Fecha:**

4 jun. 2020

****Indice****

**Contenido**

[Descripción del Proyecto 3](#_Toc38134068)

[Materiales 7](#_Toc38134069)

[Diagrama Fritzing 8](#_Toc38134070)

[Código del nodeMCU 9](#_Toc38134071)

[Código página web 24](#_Toc38134072)

[Diagrama de Gantt 26](#_Toc38134073)

Fotografías……………………………………………………………………27

Bibliografía…………………………………………………………………..28

****Descripción del Proyecto****

Los contaminantes del aire tales como las partículas y gases nocivos son perjudiciales para la salud humana. En el caso de la industria, por otro lado, las altas concentraciones de gases como el metano, propano, o el monóxido de carbono resultantes de malos procesos de combustión, pueden presentar un riesgo inmediato para la seguridad. Para superar estos problemas, una amplia gama de grupos como los propietarios, operadores de edificios comerciales o zonas industriales, de los ayuntamientos y organismos ambientales necesitan tener acceso a equipo de monitoreo de la calidad del aire y detectar la presencia de diversos gases.

Por tal razón presento una introducción con este Proyecto de que se puede elaborar la conceptualización y la forma en que los Sistemas Operativos modernos ayudan al mejoramiento y controles estándares de Calidad de Aire en nuestro entorno.

El purificador consta de una estructura de un filtro de papel el cual en su interior cuenta con fragmentos pequeños de carbón activado y encima del filtro un cubo de cartón para aislar el aire cargado de carbón y que a su vez se expulsa por encima por un pequeño ventilador que se activa o desactiva por comandos de voz o por la página web, además de que al lado del ventilador se encuentra el sensor mq-135 para monitorear la calidad del aire actual y que así se vea oportuno o no encender el ventilador.

Con el nodeMCU se mandaron los parámetros de la calidad del aire y del estado del purificador en 2 json diferentes: el primero constaba de un objeto calidadAire que tenía los parámetros de calidadNivel; el cual trae consigo el dato arrojado por el sensor mq-135, un string de fecha actual en formato dd/MM/yyyy hh:mm:ss el cual se obtuvo desde internet aprovechando el módulo wifi esp8266 y apoyándome de la librería time.h de Arduino, claro que implementando a parte una función para darle el formato a la fecha, y por último el objeto estado el cual cuenta con 3 parámetros; el string de fecha actual mencionado anteriormente, la variable purificadorEncendido que es un boolean para controlar si el purificador se encuentra encendido o apagado y otro boolean llamado ventiladorEncendido para controlar el estado del ventilador, se usaron 2 variables de estado ya que quise que, en caso de apagar el purificador se apague también el ventilador ya que no tiene caso mantener el ventilador encendido si lo que se busca es monitorear la calidad del aire (ya que al apagar el purificador se apaga el sensor mq-135) y que se pudiera controlar por separado el ventilador para casos en donde no es necesario que expulse carbón activado del purificador, el ventilador no se puede encender a menos que el purificador esté encendido

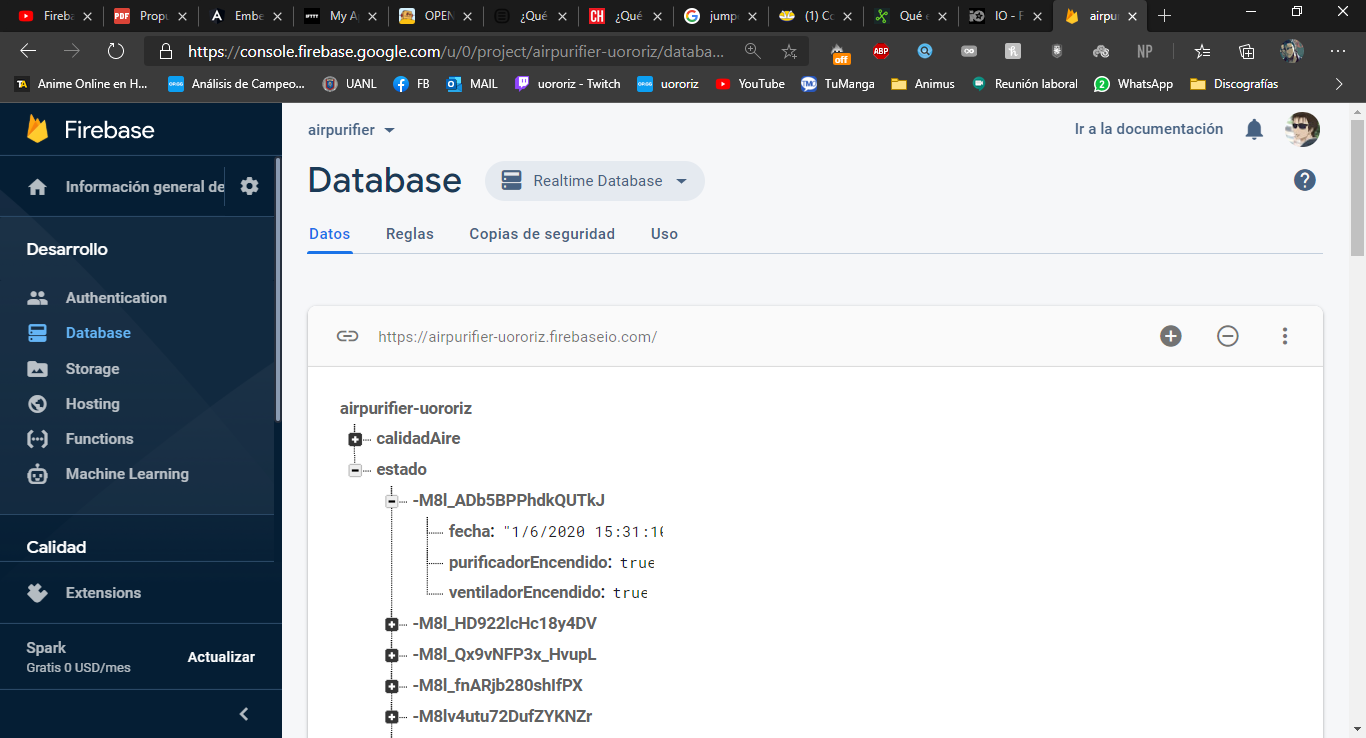
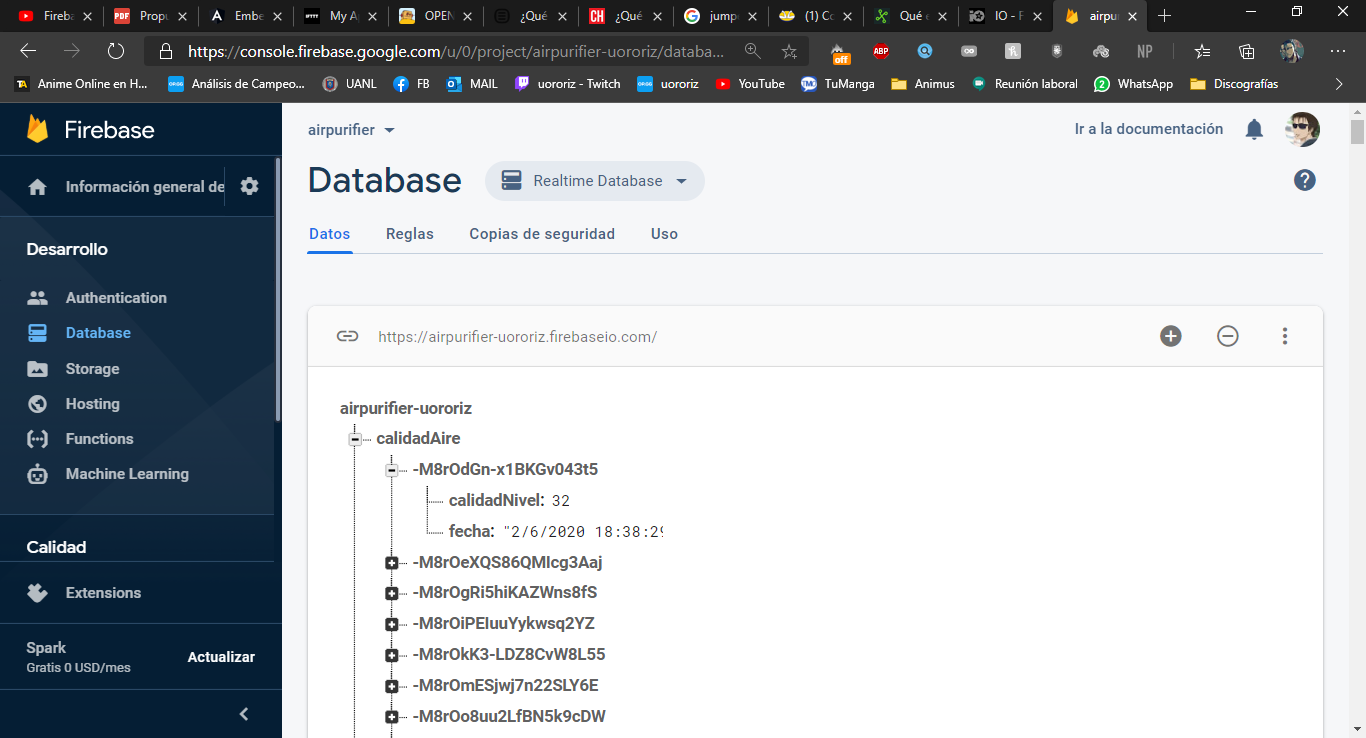
# Firebase

Es una plataforma digital que se utiliza para facilitar el desarrollo de aplicaciones web o móviles de una forma efectiva, rápida y sencilla, la cual es utilizada por sus diversas funciones como una técnica de Marketing Digital para aumentar la base de usuarios y generar mayores beneficios económicos.

Su principal objetivo, es mejorar el rendimiento de las apps mediante la implementación de diversas funcionalidades que van a hacer de la aplicación en cuestión, mucho más manejable, segura y de fácil acceso para los usuarios.

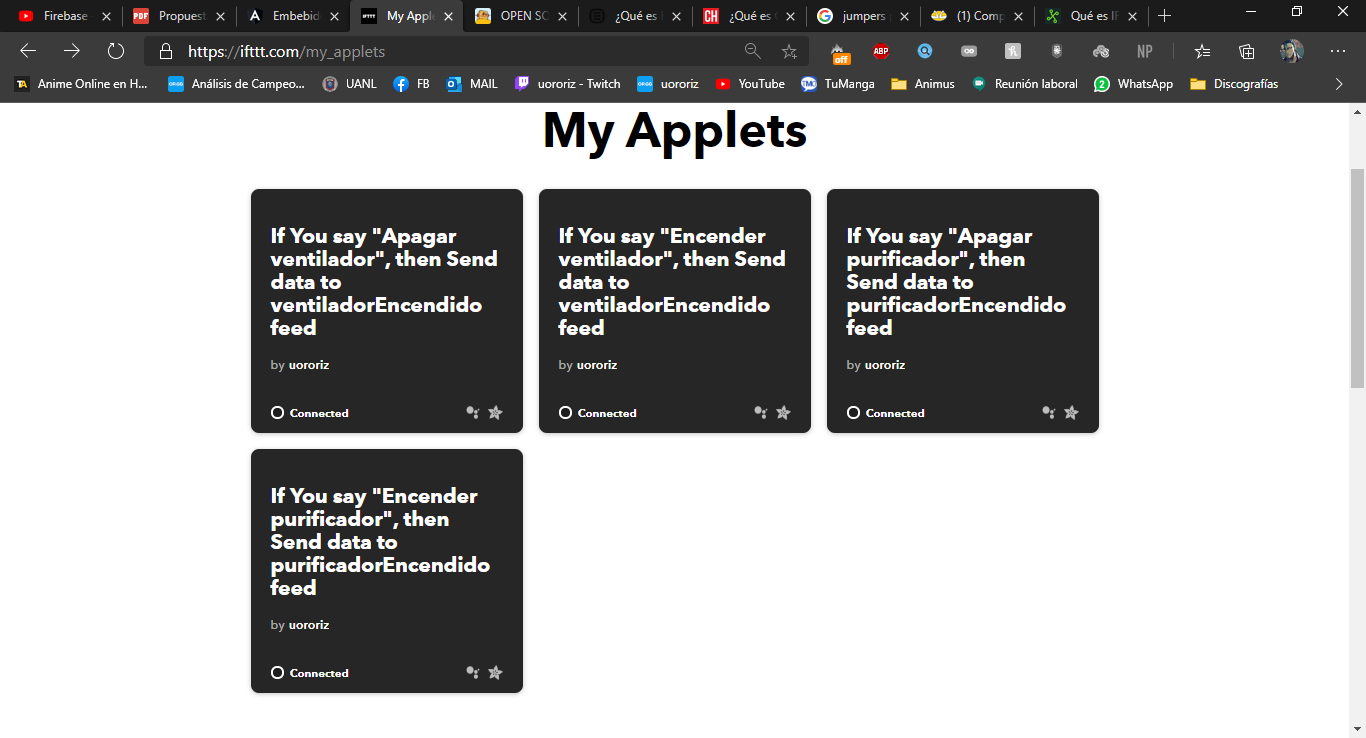
Con Firebase podemos:

* lograr una integración dinámica de los usuarios usando Firebase Authentication;
* lograr que nuestras aplicaciones sean visualizadas y utilizadas utilizando la herramienta de compartir o Dynamic Links;
* enviar notificaciones a varias plataformas con Cloud Messaging;
* crear análisis de resultados con Analytics;



# IFTTT

# Las siglas de IFTTT significan "IF This, Then That", lo que traducido al español significaría "Si esto, entonces aquello". Con este nombre, se refiere a que es una página con la que puedes crear y programar acciones entre diferentes aplicaciones, de manera que, si haces o pasa determinada cosa con una aplicación, entonces automáticamente realiza una acción con ella. Esta plataforma se utilizó para recibir los comandos de voz mandados desde el teléfono por Google assistant. Los comandos disponibles son los siguientes:

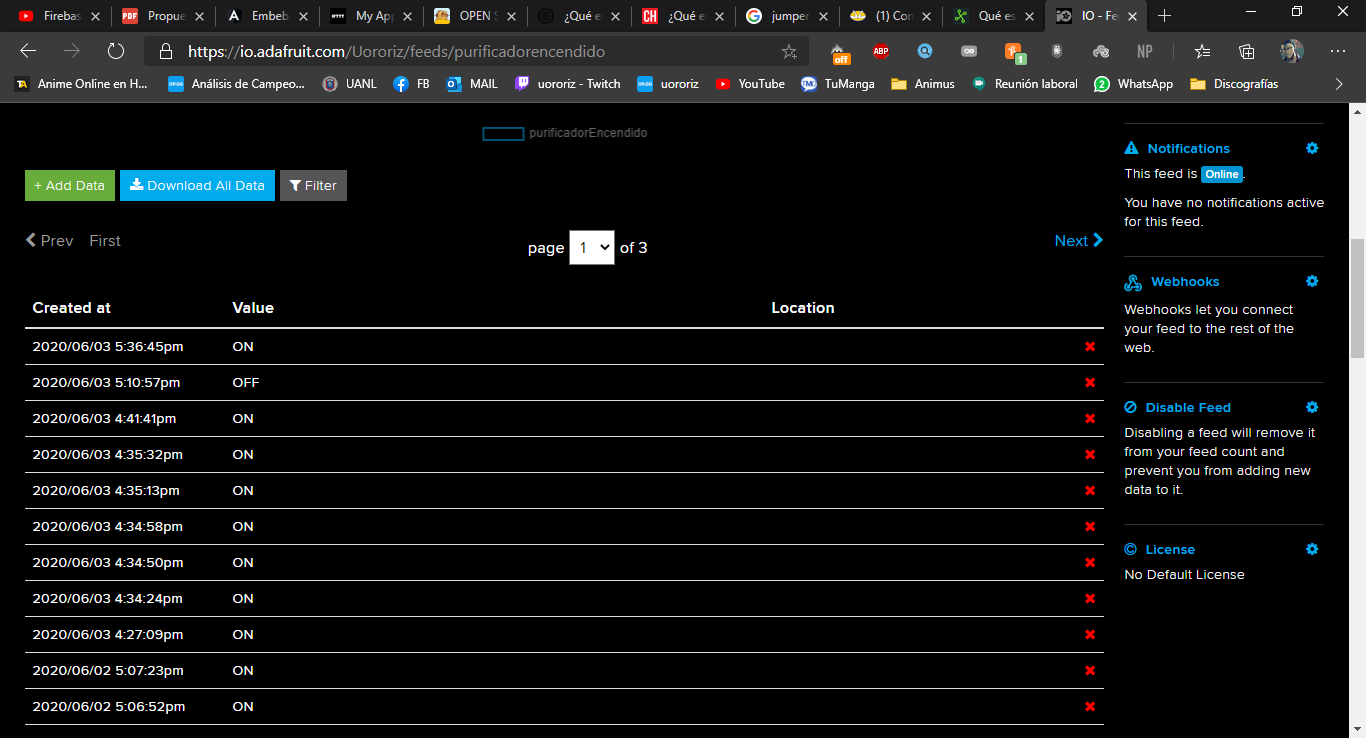


# AdaFruit IO

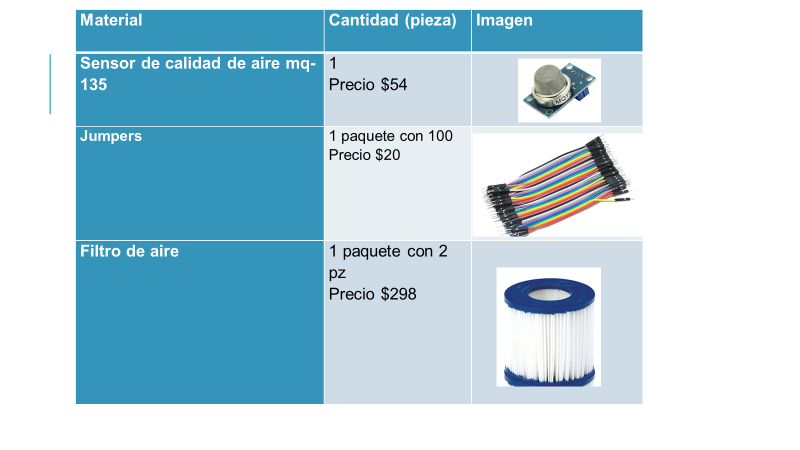
Adafruit.io es un servicio en la nube, eso solo significa que lo ejecutamos por usted y no tiene que administrarlo. Puede conectarse a través de Internet. Está destinado principalmente para almacenar y luego recuperar datos, ¡pero puede hacer mucho más que eso!

Adafruit.io se utilizó para que, con IFTT, el estado del ventilador y el purificador se cambiaran y que el nodeMCU pudiera recibir ese valor para gestionarlo dependiendo del mismo.

Valores recibidos desde IFTT:



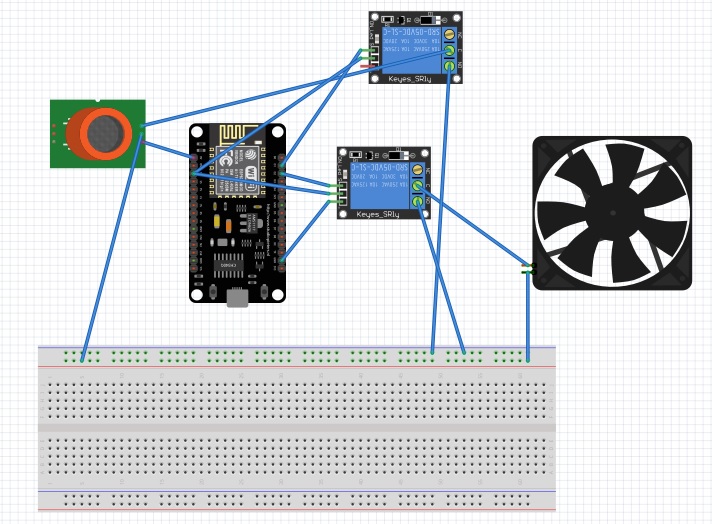
****Materiales****

****

****

**Diagrama Fritzing**

Mq-315

****

Código del nodeMCU

# #include "FirebaseESP8266.h"

# #include <ESP8266WiFi.h>

# #include <WiFiClient.h>

# #include "Adafruit\_MQTT.h"

# #include "Adafruit\_MQTT\_Client.h"

# #include <AdafruitIO.h>

# #include <time.h>

# /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# #define AIO\_SERVER "io.adafruit.com" // io.adafruit.com

# #define AIO\_SERVERPORT 1883 // use 8883 for SSL

# #define AIO\_USERNAME "Uororiz"

# #define AIO\_KEY "aio\_Ijtz32qBi85yNplegsV1izjHhKf1"

# #define FIREBASE\_HOST "airpurifier-uororiz.firebaseio.com"

# #define FIREBASE\_AUTH "GKcusIaEqbmrZhN7lifpFMpxU2XAYcPnC8CpWxhX"

# /\*\*\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# // Create an ESP8266 WiFiClient class to connect to the MQTT server.

# WiFiClient client;

# // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

# Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_USERNAME, AIO\_KEY);

# /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Feeds \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# // Setup a feed called 'photocell' for publishing.

# // Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>

# Adafruit\_MQTT\_Publish calidadAire = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/calidadaire");

# Adafruit\_MQTT\_Publish purificador = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/purificadorencendido");

# Adafruit\_MQTT\_Publish ventilador = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/ventiladorencendido");

# Adafruit\_MQTT\_Subscribe purificadorSubscribe = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/purificadorencendido");

# Adafruit\_MQTT\_Subscribe ventiladorSubscribe = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/ventiladorencendido");

# /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sketch Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# void digitalCallback(char \*data, uint16\_t len) {

# Serial.print("Hey we're in a onoff callback, the button value is: ");

# Serial.println(data);

# String message = String(data);

# message.trim();

# if (message == "ON") {

# digitalWrite(12, HIGH);

# }

# if (message == "OFF") {

# digitalWrite(12, LOW);

# }

# }

# void analogicaCallback(char \*data, uint16\_t len) {

# Serial.print("Hey we're in a onoff callback, the button value is: ");

# Serial.println(data);

# String message = String(data);

# message.trim();

# analogWrite(13, message.toInt());

# }

# //-------------------VARIABLES GLOBALES--------------------------

# int contconexion = 0;

# const char \*ssid = "Uororiz";

# const char \*password = "casa12345";

# unsigned long previousMillis = 0;

# boolean ventiladorEncendido = false;

# boolean ventiladorEncendidoUltimo = false;

# boolean purificadorEncendido = false;

# boolean purificadorEncendidoUltimo = false;

# const int a0 = A0;

# const int VENTILADOR = 5;

# const int SENSOR = 4;

# // time set variables para obtener la hora por internet

# int timezone = -5 \* 3600;

# int dst = 0;

# String pathCalidadAire = "/calidadAire";

# String pathEstado = "/estado";

# String keyPath;

# //objetos de la libreria de firebaseesp8266

# FirebaseData firebaseData;

# FirebaseJson json;

# FirebaseJson json2;

# //-------------------------------------------------------------------------

# // Callback function header

# void setup() {

# // Inicia Serial

# Serial.begin(115200);

# Serial.println("");

# pinMode(A0, INPUT);

# pinMode(VENTILADOR, OUTPUT);

# pinMode(SENSOR, OUTPUT);

# digitalWrite(VENTILADOR, HIGH);

# digitalWrite(SENSOR, HIGH);

# // Conexión WIFI

# WiFi.begin(ssid, password);

# while (WiFi.status() != WL\_CONNECTED and contconexion < 50) { //Cuenta hasta 50 si no se puede conectar lo cancela

# ++contconexion;

# delay(500);

# Serial.print(".");

# }

# if (contconexion < 50) {

# //para usar con ip fija

# IPAddress ip(192, 168, 1, 156);

# IPAddress gateway(192, 168, 1, 1);

# IPAddress subnet(255, 255, 255, 0);

# WiFi.config(ip, gateway, subnet);

# Serial.println("");

# Serial.println("WiFi conectado");

# Serial.println(WiFi.localIP());

# }

# else {

# Serial.println("");

# Serial.println("Error de conexion");

# }

# mqtt.subscribe(&purificadorSubscribe);

# mqtt.subscribe(&ventiladorSubscribe);

# Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

# Firebase.reconnectWiFi(true);

# //Establezca el tiempo de espera de lectura de la base de datos en 1 minuto (máximo 15 minutos)

# Firebase.setReadTimeout(firebaseData, 1000 \* 60);

# //Tamaño y tiempo de espera de escritura, tiny (1s), small (10s), medium (30s) and large (60s).

# //tiny, small, medium, large and unlimited.

# Firebase.setwriteSizeLimit(firebaseData, "tiny");

# configTime(timezone, dst, "pool.ntp.org", "time.nist.gov");

# Serial.println("\nWaiting for Internet time");

# while (!time(nullptr)) {

# Serial.print("\*");

# delay(1000);

# }

# Serial.println("Time response....OK!");

# }

# bool activado;

# //--------------------------LOOP--------------------------------

# void loop() {

# MQTT\_connect();

# unsigned long currentMillis = millis();

# if (keyPath != "" ) { //obtiene la ruta del ultimo estado ingresado desde firebase para que se pueda encender y apagar los componentes desde la página web

# delay(1000);

# Firebase.getBool(firebaseData, pathEstado + "/" + keyPath + "/purificadorEncendido");

# activado = firebaseData.boolData();

# digitalWrite(SENSOR, !activado);

# delay(1000);

# Firebase.getBool(firebaseData, pathEstado + "/" + keyPath + "/ventiladorEncendido");

# activado = firebaseData.boolData();

# digitalWrite(VENTILADOR, !activado);

# }

# while ((subscription = mqtt.readSubscription(1000))) {//lee las suscripciones desde adafruit.io que a su vez éste las recibe desde IFTT con el comando de voz de google assistant

# if (subscription == &purificadorSubscribe) {

# Serial.print(F("Got: "));

# Serial.println(((char \*)purificadorSubscribe.lastread));

# if ((strcmp((char \*)purificadorSubscribe.lastread, "ON") == 0)) {//si IFTT le envia un ON a adafruit.io entonces se enciende, en dado caso de recibir un OFF se apaga el pin

# digitalWrite(SENSOR, LOW);

# Serial.println("PURIFICADOR ENCENDIDO");

# digitalWrite(VENTILADOR, LOW);

# Serial.println("VENTILADOR ENCENDIDO");

# } else {

# digitalWrite(SENSOR, HIGH);

# Serial.println("PURIFICADOR APAGADO");

# digitalWrite(VENTILADOR, HIGH);

# Serial.println("VENTILADOR APAGADO");

# }

# }

# if (subscription == &ventiladorSubscribe) {

# Serial.print(F("Got: "));

# Serial.println(((char \*)ventiladorSubscribe.lastread));

# if ((strcmp((char \*)ventiladorSubscribe.lastread, "ON") == 0)) {//si IFTT le envia un ON a adafruit.io entonces se enciende, en dado caso de recibir un OFF se apaga el pin

# digitalWrite(VENTILADOR, LOW);

# Serial.println("VENTILADOR ENCENDIDO");

# } else {

# digitalWrite(VENTILADOR, HIGH);

# Serial.println("VENTILADOR APAGADO");

# }

# }

# }

# if ((digitalRead(SENSOR) == 0)) {//lee el estado del pin que controla el relevador del sensor, si éste es 0 entonces está encendido, de lo contrario está apagado

# purificadorEncendido = true;

# ventiladorEncendido = true;

# } else {

# purificadorEncendido = false;

# ventiladorEncendido = false;

# }

# if (digitalRead(VENTILADOR) == 0) {//lee el estado del pin que controla el relevador del ventilador, si éste es 0 entonces está encendido, de lo contrario está apagado

# ventiladorEncendido = true;

# } else {

# ventiladorEncendido = false;

# }

# if ((currentMillis - previousMillis >= 5000) && purificadorEncendido) { //envia la calidad de aire cada 5 segundos y solo si el purificador está encendido

# previousMillis = currentMillis;

# int analog = analogRead(A0);

# float calidad = analog;

# Serial.print(F("\nSending calidad val "));

# Serial.println(calidad);

# Serial.println("...\n");

# if (! calidadAire.publish(calidad)) {//si se logra publicar la calidad en adafruit.io entonces procede a crear el json y hace push a firebase

# Serial.println(F("Failed"));

# } else {

# String fecha = obtenerFecha();//llama la función de fecha

# json.set("calidadNivel", double(calidad));//se castea la variable por errores de firebase

# json.set("fecha", fecha);

# Firebase.pushJSON(firebaseData, pathCalidadAire, json);//se envia el json calidadAire a firebase

# Serial.println(F("OK!"));

# }

# }

# if ((ventiladorEncendido != ventiladorEncendidoUltimo) || (purificadorEncendido != purificadorEncendidoUltimo)) { //envia el estado del purificador y ventilador solamente cuando cambia.

# ventiladorEncendidoUltimo = ventiladorEncendido;

# purificadorEncendidoUltimo = purificadorEncendido;

# Serial.println(F("\nEstado ventilador val "));

# Serial.println(ventiladorEncendido);

# Serial.println("...");

# Serial.print(F("\nEstado purificador val "));

# Serial.println(purificadorEncendido);

# Serial.println("...");

# String fecha = obtenerFecha();//se manda a llamar la función de fecha

# json2.set("purificadorEncendido", purificadorEncendido);

# json2.set("ventiladorEncendido", ventiladorEncendido);

# json2.set("fecha", fecha);

# Firebase.pushJSON(firebaseData, pathEstado, json2);//se hace push del json estado a firebase

# keyPath = firebaseData.pushName();//se obtiene la nueva ruta del ultimo estado ingresado a firebase

# Serial.println(F("OK!"));

# delay(1000);

# }

# // this is our 'wait for incoming subscription packets and callback em' busy subloop

# // try to spend your time here:

# mqtt.processPackets(500);

# // ping the server to keep the mqtt connection alive

# // NOT required if you are publishing once every KEEPALIVE seconds

# // if(! mqtt.ping()) {

# // mqtt.disconnect();

# // }

# }

# // Function to connect and reconnect as necessary to the MQTT server.

# // Should be called in the loop function and it will take care if connecting.

# void MQTT\_connect() {

# int8\_t ret;

# Adafruit\_MQTT\_Subscribe \*subscription;

# // Stop if already connected.

# if (mqtt.connected()) {

# return;

# }

# Serial.print("Connecting to MQTT... ");

# uint8\_t retries = 3;

# while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

# Serial.println(mqtt.connectErrorString(ret));

# Serial.println("Retrying MQTT connection in 10 seconds...");

# mqtt.disconnect();

# delay(10000); // wait 10 seconds

# retries--;

# if (retries == 0) {

# // basically die and wait for WDT to reset me

# while (1);

# }

# }

# Serial.println("MQTT Connected!");

# }

# String obtenerFecha() {//función que obtiene la fecha y hora actual desde internet

# String fecha;

# time\_t now = time(nullptr);

# struct tm\* p\_tm = localtime(&now);

# //se obtiene cada valor de fecha por separado

# String iday = String(p\_tm->tm\_mday);

# String imonth = String(p\_tm->tm\_mon + 1);

# String iyear = String(p\_tm->tm\_year + 1900);

# String ihour = String(p\_tm->tm\_hour);

# String imin = String(p\_tm->tm\_min);

# String isec = String(p\_tm->tm\_sec);

# fecha = iday + '/' + imonth + '/' + iyear + ' ' + ihour + ':' + imin + ':' + isec;//se aplica el formato deseado para la fecha

# Serial.println(fecha);//se imprime en consola con fines de monitoreo

# return fecha;//regresa el string de fecha formateado

# }

Código página web

**Gráficos de dona y de puntos**

/////tabla

<div class="row">

  <div class="row">

    <div class="col-12">

      <app-pastel

        [calidadAire] ="this.calidadActual"

      ></app-pastel>

    </div>

  </div>

  <div class="row">

    <div class="col-12">

      <app-linea

        [calidadArray]="this.calidadArray"

        [fechaCalidadArray]="this.fechaCalidadArray"

        [estatusPurificadorArray]="this.estatusPurificadorArray"

        [estatusVentiladorArray]="this.estatusVentiladorArray"

        [fechaEstatusArray]="this.fechaEstatusArray"

      ></app-linea>

    </div>

  </div>

  <div class="col-12">

    <div class="scrolltable col-10" style="margin: 0 auto;">

      <table class="table">

        <tr>

          <th \*ngFor="let label of lineChartLabels">{{ label }}</th>

        </tr>

        <tr

          \*ngFor="let d of lineChartData; let i = index"

          [class]="'line-' + i"

        >

          <td \*ngFor="let label of lineChartLabels; let j = index">

            {{ d && d.data[j] }}

          </td>

        </tr>

      </table>

    </div>

  </div>

  <div class="row">

    <button \*ngIf="!purificadorEncendido" (click) = "encenderPurificador()" type="button" class="btn btn-success">

      Encender Purificador

    </button>

    <button \*ngIf="purificadorEncendido" (click) = "encenderPurificador()" type="button" class="btn btn-danger">

      Apagar Purificador

    </button>

    <button \*ngIf="!ventiladorEncendido" (click) = "encenderVentilador()" type="button" class="btn btn-info" [disabled] ="!this.purificadorEncendido" >

      Encender Ventilador

    </button>

    <button \*ngIf="ventiladorEncendido" (click) = "encenderVentilador()" type="button" class="btn btn-warning">

      Apagar Ventilador

    </button>

  </div>

</div>

////grafica de puntos

<div class="row">

  <div class="col-md-12">

    <div style="display: block;">

      <canvas

        baseChart

        width="400"

        height="400"

        [datasets]="lineChartData"

        [labels]="lineChartLabels"

        [options]="lineChartOptions"

        [colors]="lineChartColors"

        [legend]="lineChartLegend"

        [chartType]="lineChartType"

        [plugins]="lineChartPlugins"

        (chartHover)="chartHovered($event)"

        (chartClick)="chartClicked($event)"

      ></canvas>

    </div>

  </div>

</div>

////grafica de dona

<div class="row">

  <div class="col-md-12">

    <div style="display: block;">

      <canvas

        baseChart

        width="400"

        height="400"

        [data]="doughnutChartData"

        [labels]="doughnutChartLabels"

        [chartType]="doughnutChartType"

      >

      </canvas>

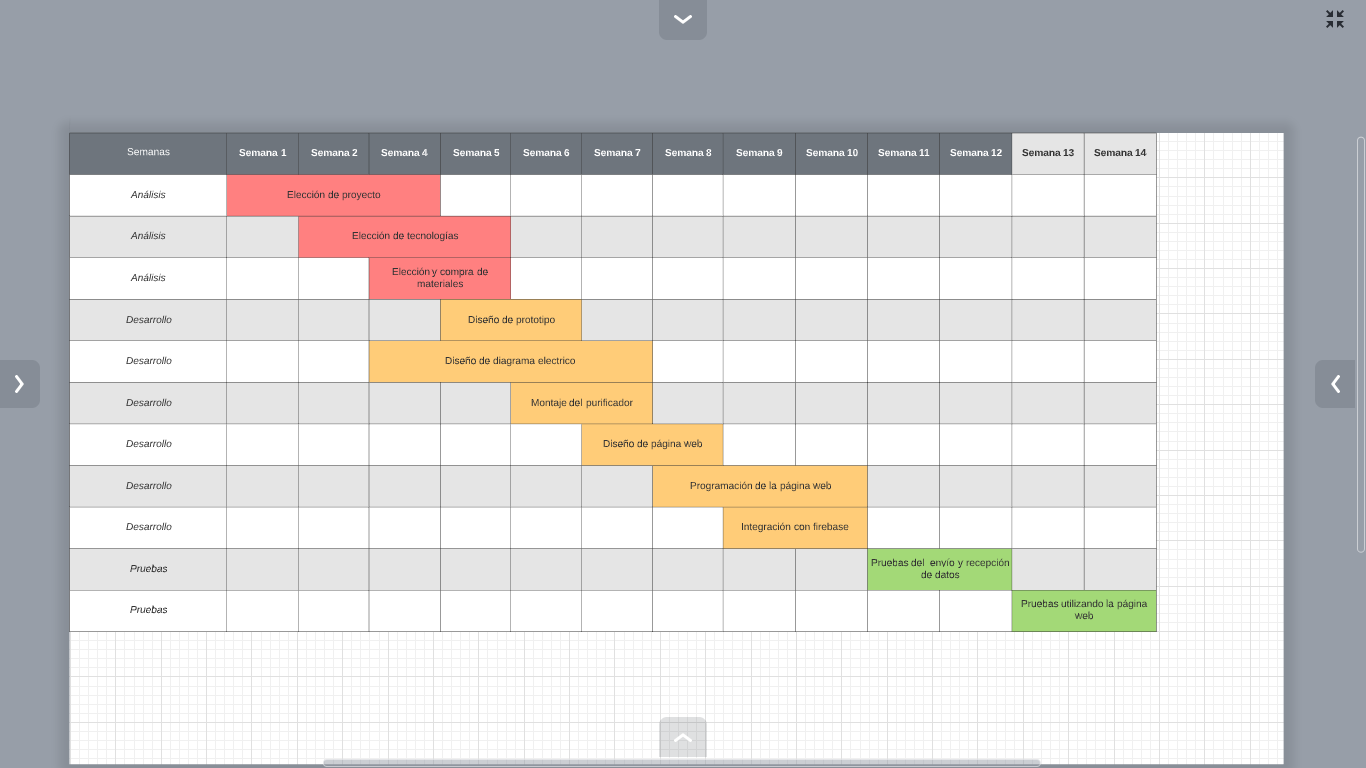
    </div>

  </div>

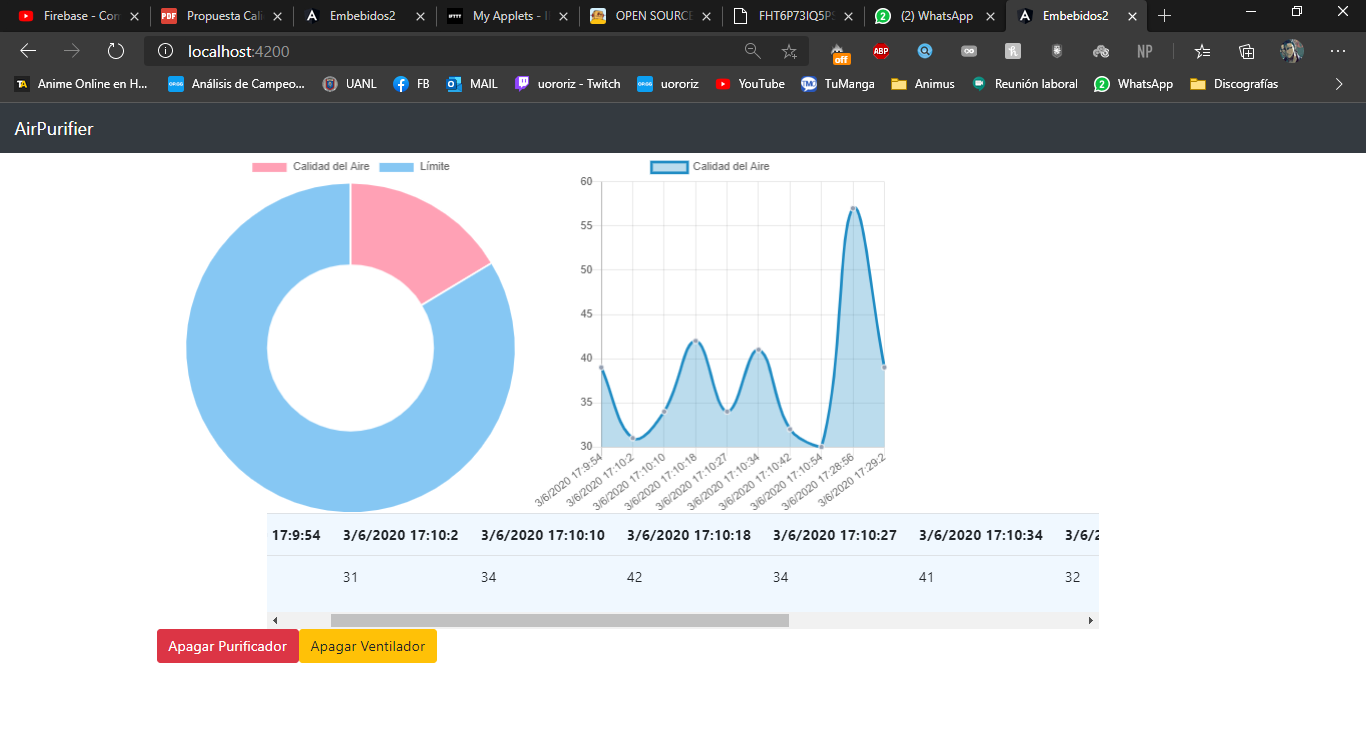
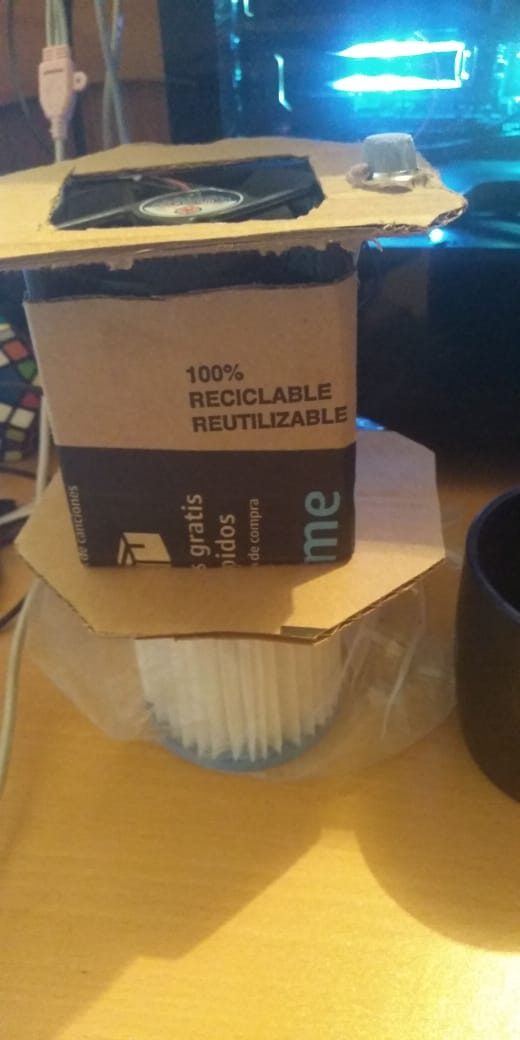
</div>

El código completo (incluyendo lo de typescript) se incluye en un paquete a parte ya que es más extenso.

****Diagrama de Gantt****



Fotografías del proyecto:



Bibliografía

*Introducción a Arduino con firebase*

<https://create.arduino.cc/projecthub/electropeak/connecting-arduino-to-firebase-to-send-receive-data-cd8805>

*Documentación de firebase-arduino*

<https://firebase-arduino.readthedocs.io/en/latest/#class_firebase_arduino_1a2ffac88673914b1127500fcaf4ad9187>

*Documentación de firebase-esp8266*

<https://github.com/mobizt/Firebase-ESP8266/blob/master/README.md>

*Vídeo de cómo obtener la fecha y hora actual desde internet*

<https://www.youtube.com/watch?v=EQysbOq_4go>

*Gráficos de dona y de puntos en angular con ng2-charts*

<https://valor-software.com/ng2-charts/#/GeneralInfo>

*IFTT plataforma utilizada para “escuchar” las peticiones desde Google assistant*

[https://ifttt.com/](https://ifttt.com/my_applets)

*Adafruit io, cloud service utilizado para recibir el ON/OFF desde IFTT*

<https://io.adafruit.com/>