CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

REPORTE DE PROYECTO INDUSTRIAL

"Desarrollo de una plataforma prototipo de bajo costo para el análisis de contaminación ambiental."

ESPECIALIDAD DE TECNÓLOGO EN MECATRÓNICO

PRESENTA

Lic. David Josafath Brito Díaz

Tutor Académico

Dr. Leonardo Barriga Rodríguez

Santiago de Querétaro, Querétaro. Agosto 2019

Índice general

Índice de Figuras.

ÍNDICE DE TABLAS.

Agradecimiento.

Quiero expresar mi gratitud a Dios, quien con su bendición llena siempre mi vida y a toda mi familia por estar siempre presentes y apoyarme en todos mis proyectos.

Mi profundo agradecimiento a todas las autoridades y personal que hacen el Centro de Ingeniería y Desarrollo Industrial por confiar en mí, abrirme las puertas y permitirme realizar todo el proceso investigativo dentro de su establecimiento educativo.

Finalmente quiero expresar mi más grande y sincero agradecimiento al Dr. Leonardo Barriga Rodríguez, principal colaborador durante todo este proceso, quien con su dirección, conocimiento, enseñanza y colaboración permitió el desarrollo de este trabajo.

David Josafath Brito Díaz

Resumen del Proyecto.

La calidad del aire en los últimos años ha tomado un papel protagónico, ya que los índices de contaminación en las ciudades llegaron a niveles que impactan directamente en la salud de las personas, cada año miles de personas enferman o mueren a casusa de esto. Por lo mencionado anteriormente se desarrolló un prototipo de plataforma crowdsensing para poder monitorear la calidad del aire, este proyecto consta principalmente de 3 partes: sistema electrónico que censa las partículas de contaminación de CO y CO2 suspendidas en aire, una aplicación que muestra las medidas actuales del dispositivo electrónico e incluye un mapa donde se puede observar en que puntos se tomaron las mediciones y una base de datos web.

Capitulo 1: Generalidades de Proyecto.

1.1 Introducción del Proyecto.

En la actualidad, un tema que está tomando importancia es el de monitoreo de la calidad del aire, ya que, en los últimos años se ha visto un incremento en los índices de contaminación. Existen graves riesgos sanitarios no solo por exposición a las partículas, sino también al ozono (O3), el dióxido de nitrógeno (NO2) y el dióxido de azufre (SO2). El ozono es un importante factor de mortalidad y morbilidad por asma, mientras que el dióxido de nitrógeno y el dióxido de azufre pueden tener influencia en el asma, los síntomas bronquiales, las alveolitis y la insuficiencia respiratoria.

Por la situación mencionada anteriormente se decide por realizar una plataforma prototipo crowdsensing, en la cual, se pueda consultar el monitoreo de la calidad del aire, el desarrollo de esta plataforma se dividió principalmente en 3 partes, un dispositivo electrónico capaz de monitorear el incremento de partículas por millón de Dióxido de carbono (CO2) y Monóxido de Carbono(CO), una aplicación para celulares Android con la cual se realiza la comunicación entre el dispositivo electrónico y una base de datos vía web, por último se tiene un mapa en el cual se pueden visualizar los datos de las mediciones.

1.2 Planteamiento del Problema.

La contaminación del aire representa un importante riesgo medioambiental para la salud. Mediante la disminución de los niveles de contaminación del aire los países pueden reducir la carga de morbilidad derivada de accidentes cerebrovasculares, cánceres de pulmón y neumopatías crónicas y agudas, entre ellas el asma [1].

Figura 1.Riesgos por contaminación del aire [1].

Monitorear la calidad del aire se ha vuelto un tema de interés por los gobiernos, ya que se tiene una repercusión directa con la población, entonces al llevar un seguimiento de los índices de contaminación del aire se pueden tomar acciones correctivas o preventivas. Además que, actualmente el monitoreo de la calidad del aire se realiza con estaciones estáticas, las cuales, se encuentran colocadas en locaciones muy lejanas entre una y otra y las mediciones de éstas solo ofrecen una pequeña estimación de las condiciones totales.

1.3 Justificación del Proyecto.

El trabajo se justifica, debido a que esta tecnología existe y está siendo utilizada en otros países, actualmente la mayoría de las personas cuentan con un Smartphone, el cual, puede ser de ayuda para procesar los datos y monitorear en distintas zonas al mismo tiempo, mediante el uso de un aplicación y un dispositivo electrónico, lo cual reduce costos de monitoreo por zonas y mejora la calidad de la base de datos al tener más puntos de medición de la contaminación.

Al contar con una aplicación para el celular, ésta facilita la consulta de la calidad del aire ya que hoy en día se ocupan aplicaciones para todo, ya sea para pedir comida o para transporte, no se necesita de una conexión de internet solo basta con conectar tu Smartphone mediante bluethoot con el dispositivo.

La plataforma puede ser consultada desde cualquier punto de la ciudad que cuente con Internet y así ayudar a las personas a tomar las medidas necesarias en caso de pasar por una zona afectada.

1.4 Hipótesis del Proyecto.

Es posible realizar el monitoreo con una plataforma de bajo costo para mejorar los índices de contaminación del aire.

1.5 Objetivo General del Proyecto.

Desarrollar una plataforma de bajo costo en la cual, se pueda monitorear la concentración de CO y CO2 en el aire.

1.6 Objetivos Específicos del Proyecto.

- Diseñar dispositivo electrónico que monitoree la concentración de CO y CO2 en el aire.
- Diseñar aplicación en Android que permita visualizar las mediciones y realice él envió de éstas a una base de datos web.
- Mostrar los datos en un mapa para poder ser visualizados.

1.7 Alcances.

- Este trabajo tendrá como alcance general entregar el prototipo con capacidad de monitorear la concentración de CO y CO2 en el aire.
- Diseño de una aplicación para visualizar las mediciones a través de una aplicación para Smartphone o a través de una página Web.

1.8 Limitaciones del Proyecto.

- Se implementarán librerías para facilitar la programación del dispositivo electrónico.
- La aplicación solo se desarrolla para la plataforma Android.
- La base de datos que se aloja en un sitio web, tiene un costo pero se elige un periodo de prueba que tiene una vigencia de 15 días.
- Los sensores utilizados en el dispositivo electrónico fueron seleccionados por el precio y la facilidad para adquirirlos.
- Se usa una power bank de 12500 mA/h para alimentar el dispositivo electrónico.

Capitulo 2:

Fundamentos

2.1 Introducción.

Este capítulo presenta los diversos temas que permiten conocer los aspectos que rodean al proyecto, respecto al hardware y software que se utilizaran para desarrollar la plataforma de crowdsensing.

2.2 Smart Cities.

Se define Smart City como aquella ciudad que usa las TIC para hacer que, su infraestructura critica, como sus componentes y servicios públicos ofrecidos, sean más interactivos, eficientes y los ciudadanos puedan ser más conscientes de ellos [2].

Las "Smart cities" es un nuevo termino que está revolucionando la manera en cómo vemos el mundo, su funcionamiento depende mucho de una alta integración y coordinación de objetos de uso común equipados con algún grado de inteligencia. El principal concepto detrás de una ciudad inteligente es el de integrar el mundo físico en un mundo virtual [3].

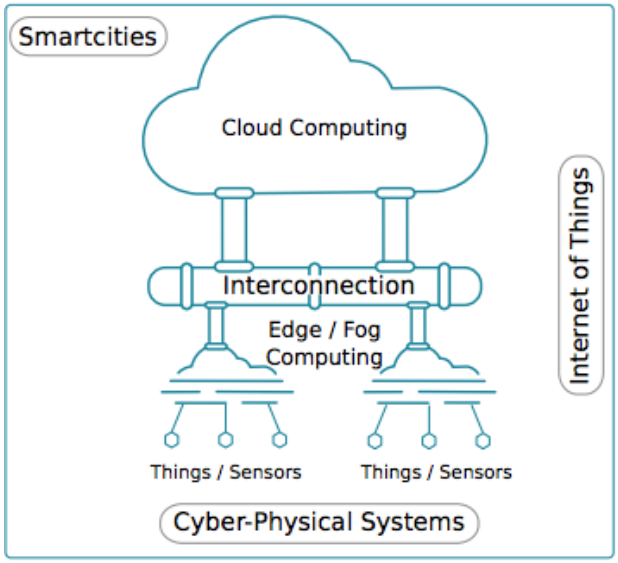


Figura 2. Estructura de una ciudad inteligente [4].

En la **Figura 2** se observa una vista global de cómo funciona una ciudad inteligente, el primer paso para lograr esta estructura es el de equipar a objetos (Things) de uso cotidiano con sensores y capacidad de enviar información, para así compartir los datos y optimizar su funcionamiento. Esto nos introduce otro termino llamado Internet de las cosas (IoT) enfocado en la

intercomunicación de todos los objetos o bien la comunicación de los objetos con servidores de datos, visto de otra manera, desde una perspectiva operacional, como un sistema ciberfísico este es un mecanismo controlado o monitorizado por algoritmos basados en computación y estrechamente integrados con internet. Finalmente visto desde una manera de servicio se tiene el procesamiento de estos datos en servidores centrales o locales.

En una Smartcity se tienen varias sub-áreas de interés por ejemplo, Smart Governance, Smart Mobility, Smart Utilities, Smart Buildings, and Smart Environment, donde si se logra implementar correctamente tendrán un gran beneficio para para la comunidad.

El proceso de monitoreo es uno de los más importantes, ya que estos permiten recabar información de distintos procesos de control por ejemplo el transporte, gestión energética o calidad del aire, etc. Tomando como ejemplo el de controlar la contaminación en las ciudades inteligentes abre un tema muy importante de desarrollo, ya que al monitorear y controlar este proceso se tiene un impacto directo en la salud humana y es en este proceso en el cual, tener un monitoreo correcto ayuda a dar un mejor servicio a los usuarios.

Actualmente el tema de controlar los niveles de contaminación del aire en las ciudades es un tema de interés para la mayoría de los gobiernos, ya que están invirtiendo fuertes cantidades de dinero para prevenir los impactos en la salud que ésta puede generar. El análisis de la calidad del aire recae principalmente en estaciones estáticas, realizar la instalación y el mantenimiento de estas, resulta sumamente costoso. Adicionalmente, la instalación de nuevas estaciones en zonas altamente pobladas resulta difícil por lo que se decide muchas veces instalarlas en locaciones remotas como parques o áreas separadas de la población, lo que nos da solo una pequeña estimación de las condiciones totales de la población.

2.3 Crowdsensing.

Los sensores han sido utilizados en la industria desde hace tiempo. Este tipo de sensores son altamente especializados y requieren de ajustes finos y calibraciones, aparte de ir conectado a un equipo de control mediante cables. Este tipo de sensores tienen un costo alto y un uso poco versátil ya que no podrían funcionar como sensores para monitorear el entorno o caminos de trasporte.

En 1990 se introduce la tecnología wireless, gracias a esta nueva tecnología se desarrollan nuevos sensores de menor tamaño y con la capacidad de trabajar en red, estos son alimentados con baterías. El uso de estos sensores permitió abarcar áreas más grandes ya que no hacía falta el uso de cables y en lugar de tener módulos complejos y costosos para manejar y recolectar los datos, cada sensor cuenta con inteligencia que maneja su funcionamiento y ayuda a realizar la conexión en red. Este tipo de sensores presenta ciertas limitantes en cuanto a su autonomía, ya que su funcionamiento depende de cuánto tiempo dura la batería, por lo cual, el uso de este tipo de sensores en la vida cotidiana no sería practico. La evolución de la capacidad de las baterías ha sido lineal y el costo del desarrollo de sensores estáticos con wireless no ha disminuido como se esperaba.

En los últimos años con la evolución de los celulares, se ha desarrollado una solución complementaria para las redes estáticas de sensores wireless, la detección móvil. Sensores móviles como los teléfonos inteligentes, relojes inteligentes y sistemas vehiculares, representan una nueva infraestructura geográfica de censado centrada en las personas. Este tipo de detección móvil crea una extensa gama de aplicaciones que van desde monitorear el transito hasta el monitoreo del avance de una enfermedad epidémica en tiempo real. Comparados con los sensores wireless estáticos, este tipo de sensores al ser de menor tamaño presenta ventajas en cuanto a consumo energético y mayor capacidad para realizar cálculos complejos además de permitir conexión a internet.

Al tener esta infraestructura formada por la detección móvil se desarrolla a la par, una tecnología llamada crowdsensing. Esta tecnología usa la inteligencia colectiva de las multitudes para resolver problemas que son difíciles de resolver usando computadoras o medios tradicionales costosos como contratar expertos.

Crowdsensing consiste en cinco pasos fundamentalmente, muestreo, filtrado de datos, transferencia de datos, procesamiento de datos y presentación de resultados [4].

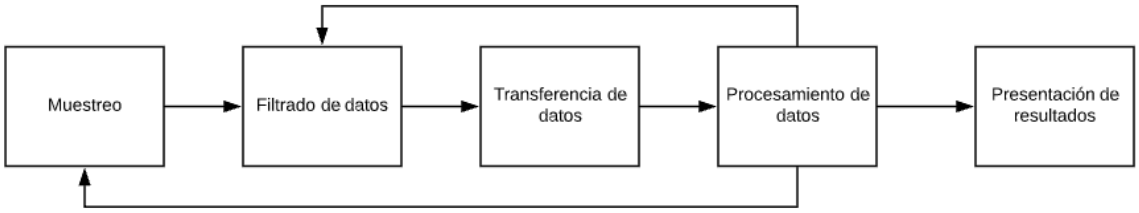


Figura 3. Procesos de para realizar Crowdsensing [4].

Proceso de muestreo, en este proceso se realizan las mediciones de un fenómeno físico, incluye proceso de calibración, donde las señales eléctricas se convierten a unidades, en este caso a unidades de contaminación.

Proceso de filtrado, se eliminan datos redundantes y errores por oscilaciones en lecturas de los sensores, en este proceso también se debe de tomar en cuenta los errores temporales, ajustando las muestras al mismo tiempo temporal.

Proceso de transferencia de datos, este se refiere a subir los datos a un servidor en la web.

Procesamiento de datos se refiere a una técnica de interpolación para crear un mapa de distribución de datos.

Presentación de resultados es el proceso final, en el que se muestra al administrador del sistema una representación gráfica como un mapa, para la zona de interés.

2.3 Arduino.

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines hembra, los que permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables dupont) [6].

Figura 4. Logotipo oficial de Arduino [7].

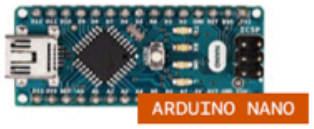
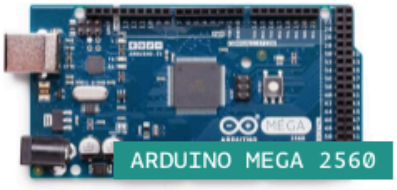
El hardware consiste en una placa con un microcontrolador de la marca Atmel AVR y puertos de entrada/salida. Por parte del software, éste consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.

Alguna de las razones por la cual se opta por elegir esta placa de desarrollo son las siguientes:

- Es una plataforma que puede correr en Windows, Macintosh y Linux.
- Es software y hardware libre
- Tiene la comunidad de usuarios activos más grande, por lo cual se facilita encontrar información acerca de un tema de interés.

A continuación, se muestra en la **Tabla 1** una comparativa de algunos modelos de las placas que ofrece Arduino, como podemos observar las características de cada una cambia, por lo cual debemos de elegir la que más se adapte a las necesidades del proyecto.

Tabla 1 Modelos de placas Arduino [7].

Columna1			
Modelo	Nano	Uno	Mega/Mega2560
Microcontrolador	AVR Atmega 168 ó 328 8bits	AVR Atmega 328 8bits	AVR Atmega2560 8 bits
Frecuencia	16 Mhz	16 Mhz	16 Mhz
Memoria RAM	2KB	2KB	8KB
Memoria EEPROM	1KB	1KB	4KB
Memoria FLASH	16 ó 32 KB	16 ó 32 KB	128 ó 256 KB
Pines digitales Entradas/Salidas	14/14	14/14	54/54
Tensión/corriente pines digitales	5v / 40mA.	5v / 40mA.	5v / 40mA.
Pines analógicos Entradas/Salidas	8/0	6/0	16/0
Tensión/resolución pines analogicos	5v 10 bits	5v 10 bits	5v 10 bits
Pines con interrupción externa	2	2	6
Pines PWM	6	6	15
Conexión Serial/UART	1	1	4KB
Conexión I2C/TWI	1	1	1
Corriente en el pin de 5V.	500mA	500-800mA	500-800mA
Corriente en el pin de 3.3V.	50mA	50mA	50mA

2.4 Sensores.

A continuación, se da una descripción breve de los sensores utilizados para desarrollar el dispositivo electrónico móvil.

2.4.1 Sensor de Temperatura DHT11.

El sensor de temperatura y humedad relativa DHT11 es un sensor digital, fácil de implementar con cualquier microcontrolador. Este sensor está compuesto por un sensor capacitivo de humedad, un termistor y un microcontrolador de 8 bits de alto rendimiento.

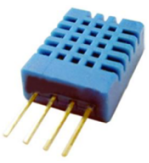


Figura 5. Sensor de temperatura y humedad relativa DHT11 [9].

Las características técnicas del sensor son las siguientes:

Tabla 2 Características técnicas del sensor DHT11 [9].

Alimentación	**3.5 a 5V**
Consumo	2.5mA
Señal de salida	Digital
Temperatura	
Rango	0^o C a 50^o C
Precisión	25^o C ± 2^o C
Resolución	
Humedad	
Rango	20% a 90% RH
Precisión	entre 0^o y 50^o ± 5% RH
Resolución	1% RH

El sensor DHT11 transmite los datos usando un solo bus, la señal se compone de 40 bits transferidos, estos 40 bits transferidos se componen de 8 bits para indicar el valor de la humedad con formato entero, más 8 bits para transmitir la señal de la humedad en decimal, luego 8 bits más para transmitir la temperatura en entero más 8 bits para enviar la señal de temperatura en decimal, y los últimos 8 bits son para enviar un "check sum" que verifica la información anterior por si hubiera errores.

2.4.1 Sensor de Gas serie MQ.

Un sensor de gas es un dispositivo que detecta la presencia de varios gases dentro de un área, usualmente se utilizan en sistemas de seguridad para detectar fugas de gases. Los sensores de gas pueden detectar gases inflamables y tóxicos.

Los detectores de gas pueden ser clasificados de acuerdo a su mecanismo de operación, en este caso los sensores que se ocuparon fueron catalíticos también conocidos como pellistores. Su funcionamiento es por la oxidación del gas vía catalítica.

A continuación, se muestra una tabla de los sensores MQ y cuáles son los gases que puede detectar cada modelo.

Tabla 3 Sensores MQ.

Nombre del Sensor	Gases
MQ-2	Metano, Butano,LPG,Humo
MQ-3	Alcohol, Etanol, Humo
MQ-4	Metano
MQ-5	Gas natural, LPG
MQ-6	LPG, Butano
MQ-7	Monóxido de Carbono
MQ-8	Gas de hidrogeno
MQ-9	Monóxido de Carbono, Gases inflamables
MQ-131	Ozono
MQ-135	Calidad del aire
MQ-136	Gas de sulfuro de hidrogeno
MQ-137	Amoniaco
MQ-138	Benceno, Tolueno, Alcohol, Propano, Hidrogeno
MQ-214	Metano y Gas natural
MQ-216	Gas natural y de carbón
MQ-303A	Alcohol, Etanol y Humo
MQ-306A	LPG, Butano
MQ-307A	Monóxido de Carbono
MQ-309A	Monóxido de Carbono, Gas inflamable

Figura 6. Sensor de gases serie MQ [10].

La estructura interna de los sensores MQ está compuesta principalmente por 2 elementos. Un elemento detector, que contiene un material catalítico sensible a la detección de gases y un elemento compensador de referencia, el cual se encuentra inerte. Los gases detectados solo se quemarán en el elemento sensible, causando un incremento en la temperatura y como consecuencia, un incremento en la resistencia eléctrica. Los gases detectados no se quemarán en el elemento compensador, así que este mantiene su temperatura y resistencia sin cambios.

La estructura y configuración del sensor MQ-135 se muestra en la **Figura 7**, el sensor está compuesto por un micro tubo cerámico de AL_2O_3, una capa sensitiva de Dióxido de Estaño (SnO2), un electrodo de medición y un elemento calefactor, todo esto fijo dentro de una corteza hecha por plástico y una red de acero inoxidable. El sensor cuenta con 6 pines, 4 usados para traer las señales y 2 para proveer la corriente para el calefactor.

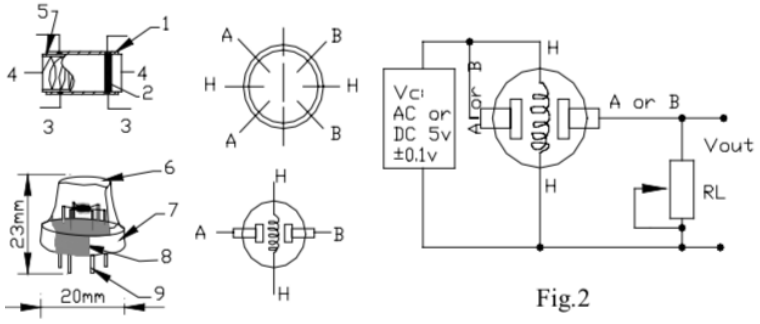


Figura 7. Sensor de gas MQ-135 [11].

Los sensores catalíticos son sensibles y pueden funcionar indeseablemente en presencia de gases inhibidores tales como el dióxido de azufre (SO_2), ácido sulfhídrico (H_2S), etc. También el catalizador puede sufrir envenenamiento si se encuentra en el aire vapores de silicón, grasas, ésteres de fosfato, entre otros.

Antes de tomar lecturas confiables de este tipo de sensores, se requiere de un tiempo de precalentamiento, este tiempo es diferente para cada sensor y viene indicado en la hoja de características, por lo regular este tiempo va de 24 a 48 hrs.

2.5 Modulo GPS Neo-06 [13].

Las siglas “GPS” significan Sistema de Posicionamiento Global, este sistema está compuesto por varios satélites que se encuentran a kilómetros fuera de la atmosfera terrestre y que continuamente están enviando señales a la Tierra por medio de radio frecuencia. El módulo GPS Neo-06 es un receptor de GPS, la antena incluida recibe las señales de los satélites que están alrededor de la Tierra. Es necesario que al menos reciba la señal de 4 satélites para dar un posicionamiento preciso.

El módulo GPS en su modelo GY-GPS6MV2 viene con un módulo de serie U-Blox NEO 6M equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica. También posee los pines o conectores Vcc, Rx, Tx y Gnd por el que se puede conectar a algún microcontrolador mediante una interfaz serial.

Figura 8. Modulo GPS Neo-06.

Especificaciones Técnicas:

- Voltaje de alimentación: 3-5 VDC
- Interface: Serial UART 5V
- Antena cerámica
- EEPROM para guardar datos de configuración cuando el módulo se desenergice
- Batería de respaldo (MS621FE)
- Frecuencia de refresco: 5Hz
- Soporta SBAS (WAAS, EGNOS, MSAS, GAGAN)
- Indicador de señal con LED

- Tamaño de la antena: 25mm x 25mm
- Tamaño del módulo: 25mm x 35mm
- Tamaño de los agujeros de montaje: 3mm
- Baud rate por defecto: 9600bps

2.5 Modulo Micro-SD [13].

El módulo Micro-SD se utiliza cuando la cantidad de datos que se desean guardar excede la capacidad de memoria del microcontrolador con el cual se esta llevando a cabo el programa. Este módulo cuenta con un buffer un regulador de voltaje a 3.3V y un soquet donde se inserta la memoria micro-sd.

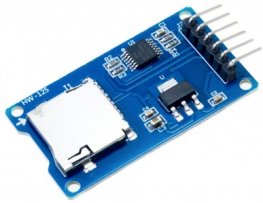


Figura 9. Modulo Micro-SD.

Las características técnicas son las siguientes:

- Voltaje de Operación: 3.3V-5V
- Interfaz: SPI
- Cuenta con todos los pines SPI de la tarjeta SD: MOSI, MISO, SCK, CS
- Te permite almacenar grandes cantidades de datos en memorias SD utilizando Arduino o PIC

2.6 Modulo Bluetooth HC-06 [13].

La comunicación bluetooth se da entre 2 dispositivos: un maestro y un esclavo. El módulo HC-06 viene configurado de fabrica como esclavo, es decir, preparado para escuchar peticiones de conexión.

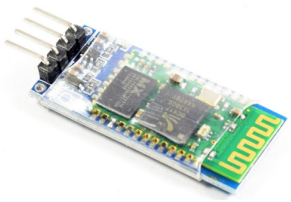


Figura 10. Modulo Bluetooth HC-06.

Especificaciones Técnicas:

- Voltaje de operación: 3.3V - 5VDC
- Corriente de operación: < 40mA
- Corriente modo sleep: < 1mA
- Chip: BC417143
- Bluetooth: V2.0+EDR
- Frecuencia: Banda ISM de 2,4 GHz
- Modulación: GFSK (Gaussian Frequency Shift Keying)
- Potencia de emisión: 4 dBm, clase 2
- Sensibilidad: -84dBm a 0.1% VER
- Alcance 10 metros
- Interfaz de comunicación: Serial UART TTL
- Velocidad de transmisión: 1200bps hasta 1.3Mbps
- Baudrate por defecto: 9600,8,1,n.
- Velocidad asíncrona: 2.1Mbps (máx.) / 160 kbps.
- Velocidad síncrona: 1Mbps/1Mbps
- Seguridad: Autenticación y encriptación
- Compatible con Android
- Dimensiones: 37*16 mm

2.7 Modulo RTC DS1307 [13].

EL módulo de Reloj en Tiempo Real se utiliza principalmente cuando se quiere monitorear el tiempo en el que se realizan las mediciones de los proyectos. Los RTC son de muy bajo consumo por lo que pueden ser alimentados por baterías y de esa forma no perder la sincronización.

El módulo está basado en el RTC DS1307 de MAXIM y la EEPROM AT24C32 de ATMEL. Ambos circuitos integrados comparten el mismo bus comunicación con el Protocolo I2C. La memoria EEPROM AT24C32 te permite almacenar 32Kbits (4K Bytes) de datos de manera permanente.

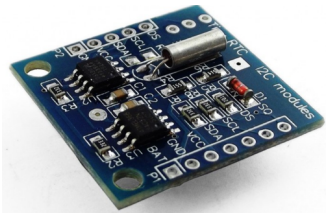


Figura 11. Modulo RTC DS1307.

Especificaciones Técnicas:

- Voltaje de Operación: 3.3V - 5V
- Integrados principales: AT24C32 y DS1307.
- Dirección I2C del DS1307: Read(11010001) Write(11010000)
- Comunicación I2C, solo utiliza 2 cables.
- La batería puede mantener al RTC funcionando por 10 años.
- Conexión para Arduino Uno:
    - SCL-A5
    - SDA-A4
    - VCC-5V
    - GND - GND

Capitulo 3: Desarrollo del Proyecto.

3.1 Introducción.

En el capítulo 3 se explica la manera en que se desarrolló el proyecto, este se divide en 3 partes, diseño del dispositivo electrónico, diseño de la aplicación en Android y por último la implementación de la base de datos.

3.2 Diseño del dispositivo electrónico móvil.

El dispositivo electrónico es capaz de tomar lecturas de las condiciones de calidad del aire junto con los siguientes datos, coordenadas de latitud y longitud de donde se tomó la muestra, hora y fecha. Estos datos se envían mediante la comunicación inalámbrica bluetooth a un dispositivo Android. Adicionalmente, en caso de no estar conectado a un dispositivo bluetooth los datos se guardarán en una memoria Micro-SD para su posterior uso.

Los materiales utilizados en el diseño del dispositivo móvil son los siguientes.

- Arduino Mega 2560.
- Sensor MQ-135
- Sensor Mq-9
- Sensor DHT11
- Modulo GPS Neo-6m
- Tiny RTC Real Time Clock Module DS1307
- Bluetooht HC-06
- Modulo Micro-SD
- Relevador
- Transistor 2n2222a
- Diodo rectificador
- Pila de 1.5V.

- Power Bank de 12500 mA.

3.2.1 Diseño y conexiones del dispositivo electrónico móvil.

Antes de comenzar con la programación se realizó un conteo de las entadas y salidas totales que se necesitan para conectar los sensores y los módulos. A continuación, se asocian las entradas y salidas a la placa de Arduino, quedando de la siguiente manera. Los pines de alimentación no se mencionan en la **Tabla 4**.

Tabla 4 Conexión módulos con la Tarjeta de desarrollo de Arduino.

Nombre del Componente	Pines de la placa Arduino
Sensor de Gas MQ-135	Salida analógica → Pin A1
Sensor de Gas MQ-9	Salida analógica → Pin A0
Sensor de temperatura DHT11	Salida digital → Pin 2
Tiny RTC Real Time Clock Module DS1307	SCL→ 21 SDA→ Pin 20
Bluetooht HC-06	RX→ Pin 1 TX→ Pin 0
Modulo Micro-SD	SCK→ Pin 52 MOSI→ Pin 51 MISO→ Pin 50 CS→ Pin 4
Modulo GPS Neo-6m	RX→ Pin 14 TX→ Pin 15

En la **Figura 12** se muestra como quedaron conectados los dispositivos y sensores a la tarjeta de desarrollo de Arduino, las pilas simbolizan a la Power Bank de 12500 mA, como se puede observar esta se encarga de la alimentación de todos los dispositivos, la batería que se encuentra en la izquierda de la imagen es utilizada para el sensor MQ-9 ya que este necesita conmutar entre dos voltajes para su funcionamiento.

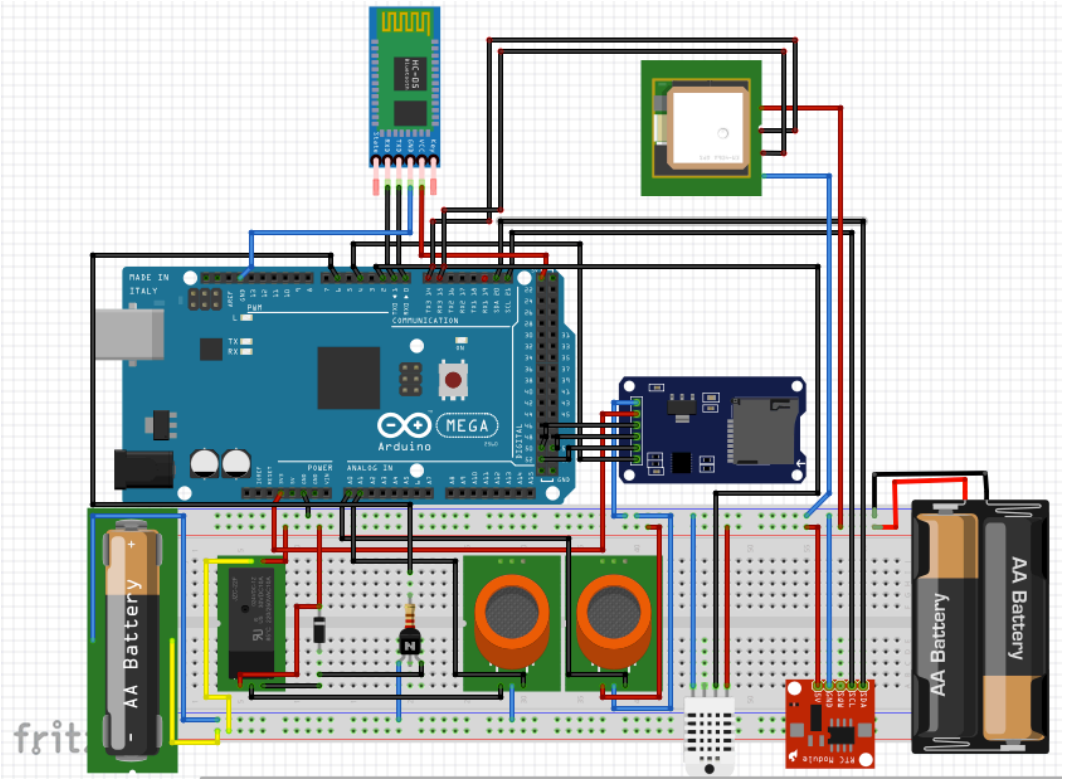


Figura 12. Conexiones del sistema electrónico.

3.2.1.1 Caracterización de los sensores de gas MQ-135 y MQ-09.

Los sensores de gas cuentan con 4 pines, 2 se utilizan para la alimentación, otro es la salida analógica y uno es digital. La salida digital envía señales en alto y en bajo, estas señales dependen del ajuste que se realice con el potenciómetro que viene incluido en la palca del sensor, las señal de salida del sensor entra a un OPAMP con configuración de comparador, este compara la señal de referencia del potenciómetro con la señal de salida del sensor, una vez que la señal del sensor es mayor que la de la referencia, se envía un uno lógico a la salida digital.

El propósito del proyecto consiste en poder medir la cantidad de partículas que se encuentran en el aire por lo que la señal digital de salida no fue utilizada.

La señal de salida analógica se ingresa a un convertidor analógico digital que en este caso fueron los pines A0 para el sensor MQ-9 y A1 para el sensor MQ-1135, una vez que se lee la señal esta necesitamos cambiar la escala de lectura a partículas por millón para ese procedimiento, la hoja

de datos nos proporciona unas gráficas de igualdad. En la **Figura 13** se puede observar el sensor de gas MQ-9.

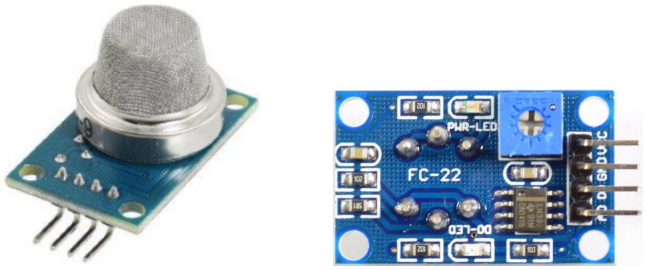


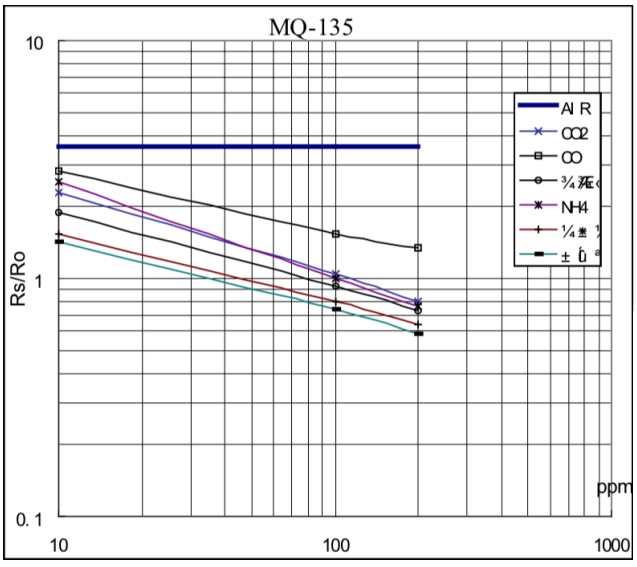
Figura 13. A) Sensor de gas MQ-9 vista delantera, B) Vista posterior del sensor de gas MQ-09.

El procedimiento que se sigue para poder interpretar la lectura analógica del sensor es el siguiente.

- Extraemos los datos de las gráficas que vienen en la hoja de especificaciones del sensor utilizando un programa en la web llamado WebPlotDigitalaizer.
- Realizamos una aproximación por Mínimos Cuadrados.
- Calculamos a Ro.

A continuación, se desarrolla el procedimiento para obtener las partículas por millón.

Sensor MQ-135. Este sensor es capaz de detectar Amoniaco, Óxidos de Nitrógeno, Alcohol, Benceno, Humo, CO2. En este caso obtendremos la gráfica de comportamiento para el CO2.



sensitivity characteristics of the MQ-135 for several gases.
in their: Temp: 20
Humidity: 65%
O_2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 100ppm of NH_3 in the clean air.
Rs:sensor resistance at various concentrations of gases.

Figura 14. Curvas características del sensor MQ-135 [11].

Siguiendo los pasos anteriormente mencionados primero es necesario abrir en un navegador web el programa de WebPlotDigitalaizer, ingresando la siguiente dirección https://apps.automeris.io/wpd/, donde aparece una ventana como la que se muestra en la **Figura 15**, una vez ahí se carga la imagen de la gráfica del sensor MQ-135 **Figura 16**, para cargar la imagen seleccionamos el botón que 'Load Image' y se busca la imagen en la computadora.

Una vez cargada la imagen se abrirá un cuadro como el que aparece en la **Figura 17**, en este cuadro se debe de seleccionar la opción de '2D (X-Y) Plot' y hacer clic en 'Align Axes' inmediatamente aparecerá una ventana indicando las instrucciones de cómo realizar el proceso de alinear los ejes **Figura 18**.

En la **Figura 19** se continúa con el proceso de alineado de ejes, en esta sección se le da valor numérico a los ejes y la escala, para este caso es logarítmica. Posteriormente se procede a seleccionar los puntos de interés de la gráfica, en la **Figura 20** se muestran los puntos utilizados para obtener los datos del gas CO2, una vez terminado se continua a seleccionar la opción de 'View data' y aparece una ventana con los datos de la gráfica **Figura 21**.

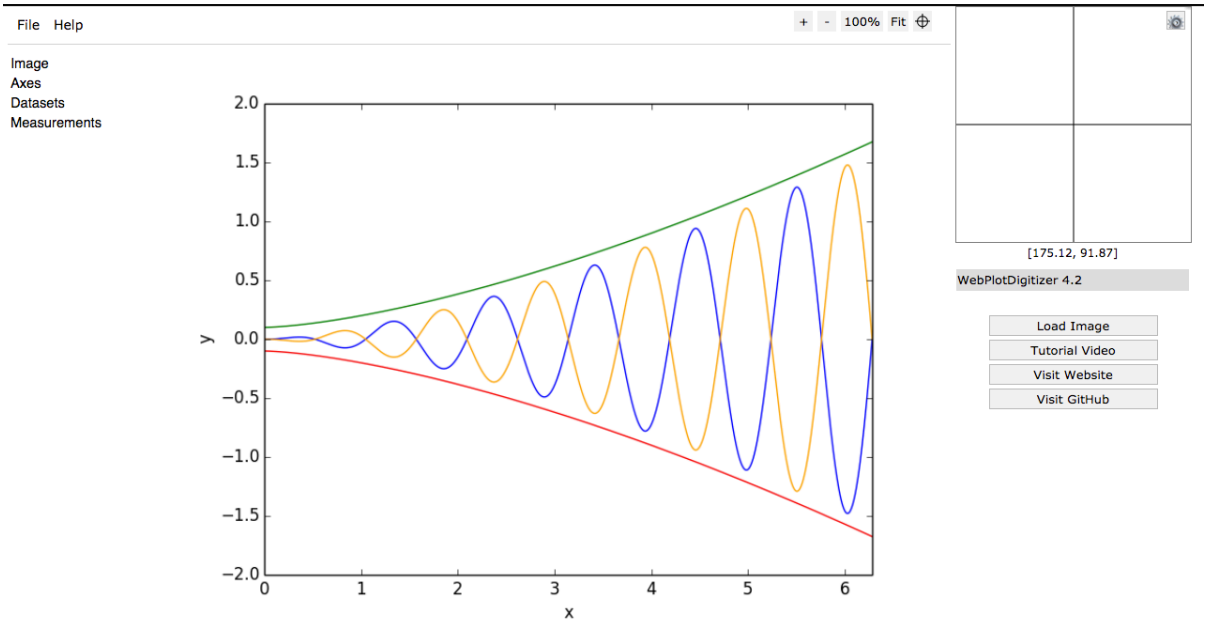


Figura 15. Ventana principal del programa WebPlotDigitalaizer.

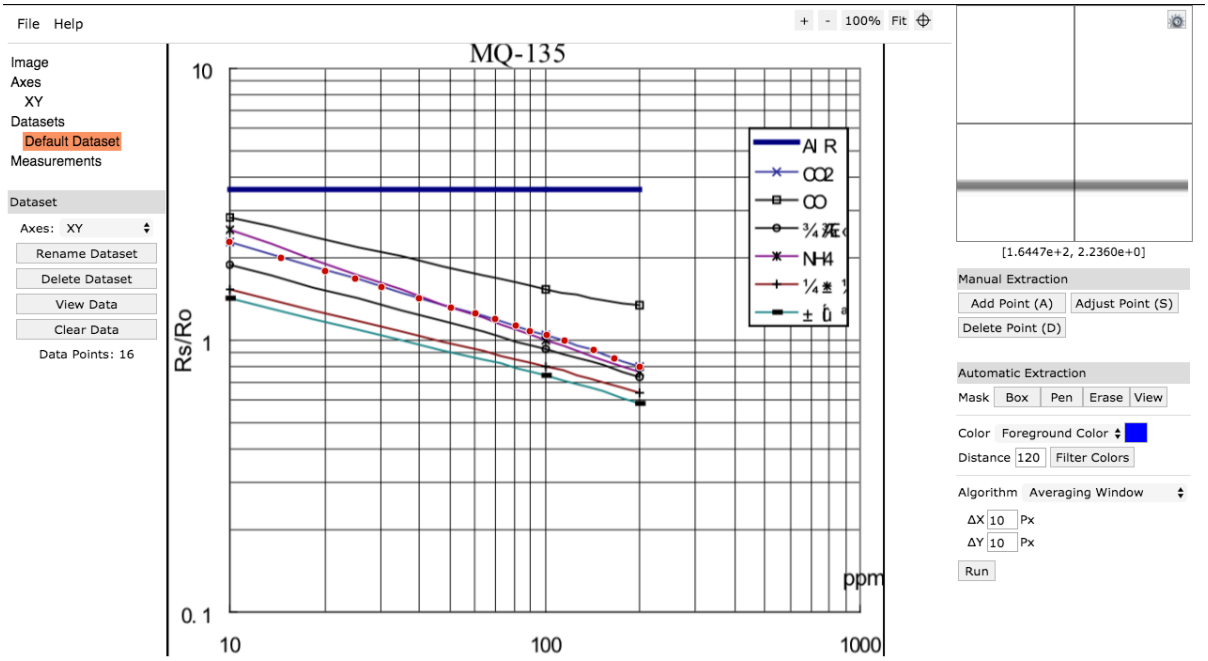


Figura 16. Ventana con la gráfica de datos del sensor MQ-135.

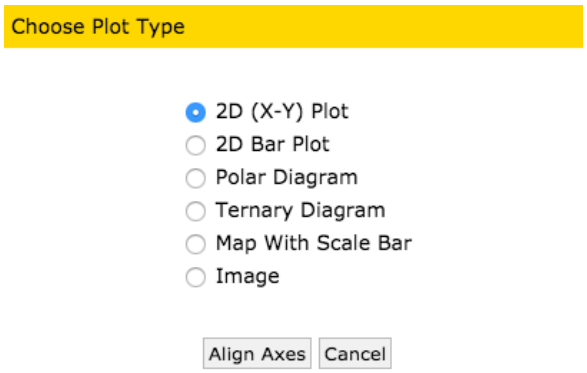


Figura 17. Cuadro de selección tipo de gráfica en WebPlotDigitalaizer.

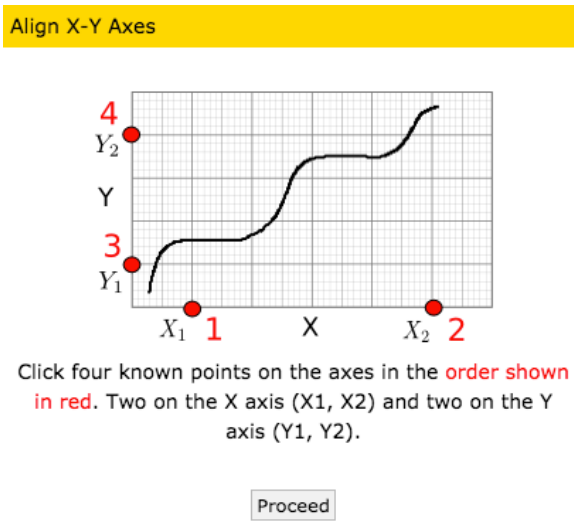


Figura 18. Cuadro de indicaciones WebPlotDigitalaizer.

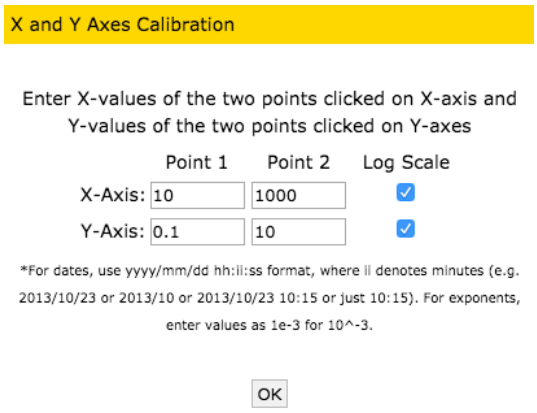


Figura 19. Cuadro de valores numéricos para los ejes WebPlotDigitalaizer.

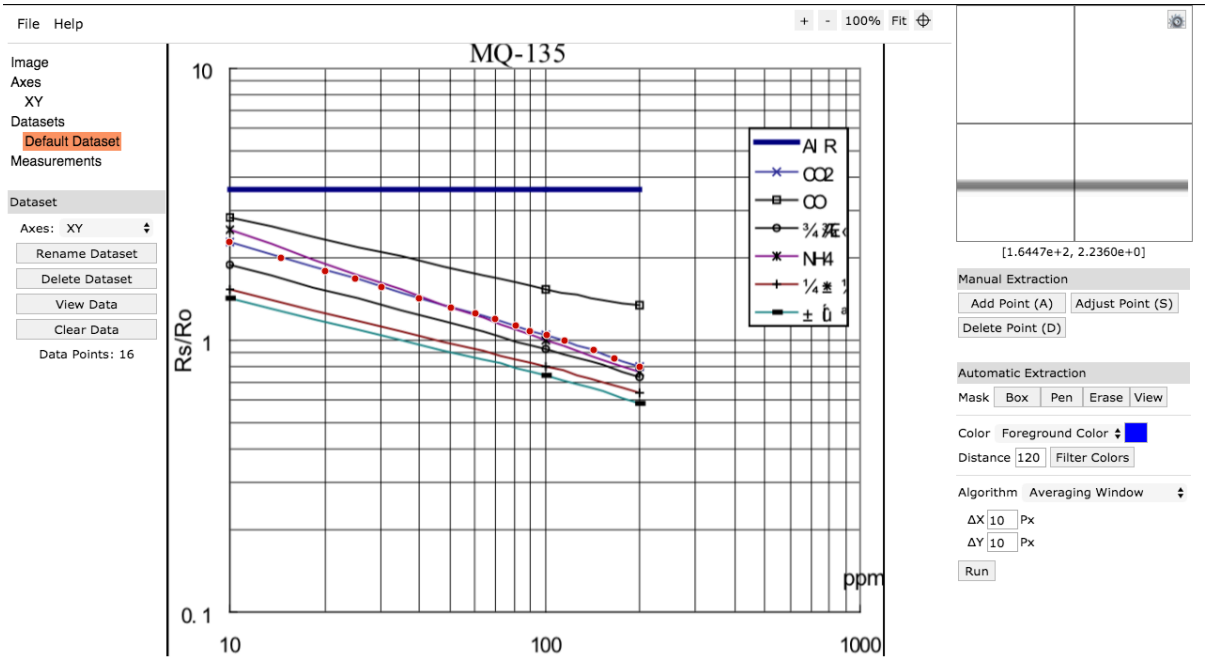


Figura 20. Puntos de interés del sensor MQ-135.

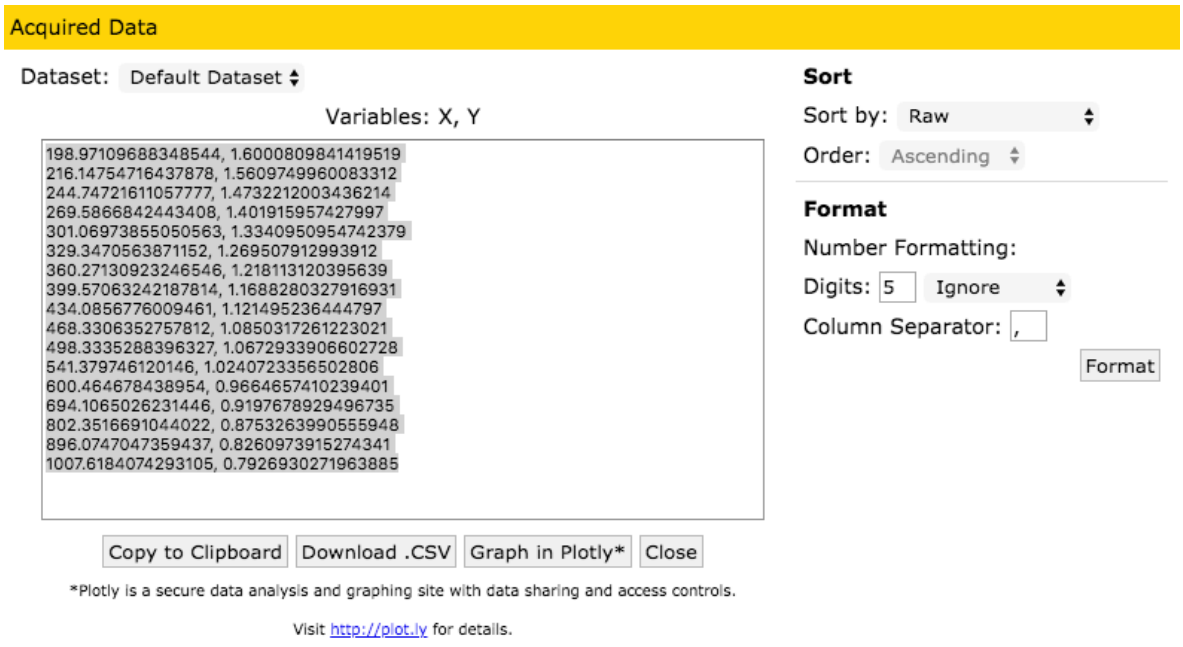


Figura 21. Coordenadas de los puntos de la gráfica.

Obtenidos los puntos se realiza una aproximación por mínimos cuadrados, se ajusta la curva del gas que queremos medir a una función exponencial de tipo $a(x^b)$, se debe de determinar el valor de a y de b.

A continuación, se desarrolla la aproximación para los valores obtenidos del sensor de gas MQ-135.

Tabla 5 Coordenadas de los puntos del sensor MQ-135 para el gas CO2.

X (Eje de las abscisas)	Y (Eje de las ordenadas)
9.931603528	2.308457053
14.48617743	2.016035939
20.00060897	1.803247602
24.91284058	1.691896151
30.1912278	1.574823308
39.72883337	1.431229709
50.17034574	1.321622021
59.97128687	1.259928851
69.26861636	1.20111551
80.5582982	1.13596074
89.29346422	1.082934217
101.034789	1.048965599
115.1072906	1
142.3974576	0.923417124
165.6059765	0.859520136
199.3209306	0.800044579

Utilizando la forma matricial se colocan los datos de la manera siguiente tomando en cuenta que Log ≡ LN y ã=log(a).

$$\begin{pmatrix} 1 & \log(x_1) \\ \vdots & \vdots \\ 1 & \log(x_n) \end{pmatrix} \cdot \begin{pmatrix} \tilde{a} \\ b \end{pmatrix} = \begin{pmatrix} \log(y_1) \\ \vdots \\ \log(y_1) \end{pmatrix}$$

Para resolver el sistema se multiplican ambos lados de la igualdad por la matriz transpuesta de coeficientes, para obtener un sistema con una solución única.

$$\begin{pmatrix} 1 & \cdots & 1 \\ \log(x_1) & \cdots & \log(x_n) \end{pmatrix} \cdot \begin{pmatrix} 1 & \log(x_1) \\ \vdots & \vdots \\ 1 & \log(x_n) \end{pmatrix} \cdot \begin{pmatrix} \tilde{a} \\ b \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 \\ \log(x_1) & \cdots & \log(x_n) \end{pmatrix} \cdot \begin{pmatrix} \log(y_1) \\ \vdots \\ \log(y_1) \end{pmatrix}$$

Ahora se resuelve el sistema con los valores de x_i y y_i extraídos de las gráficas con las curvas características. Con ayuda de Excel se obtienen los siguientes resultados.

$$\begin{pmatrix} 16 & 64.1223 \\ 64.1223 & 268.9480 \end{pmatrix} \cdot \begin{pmatrix} \tilde{a} \\ b \end{pmatrix} = \begin{pmatrix} 3.9598 \\ 11.7221 \end{pmatrix}$$

Resolviendo este sistema de 2 ecuaciones y 2 incógnitas, se obtienen los siguientes valores.

$$\tilde{a} = 1.63621 \quad b = -0{,}34651$$

Utilizando la igualdad de a = $e^{\log(a)} = e^{\tilde{a}} = e^{1.63621}$

$$a = 5.1356$$

quedando la siguiente ecuación

$$y = 5.1356x^{-0.34651}$$

Con la ayuda de MATLAB se grafican los puntos obtenidos de las gráficas y la aproximación por mínimos cuadrados **Figura 22.**

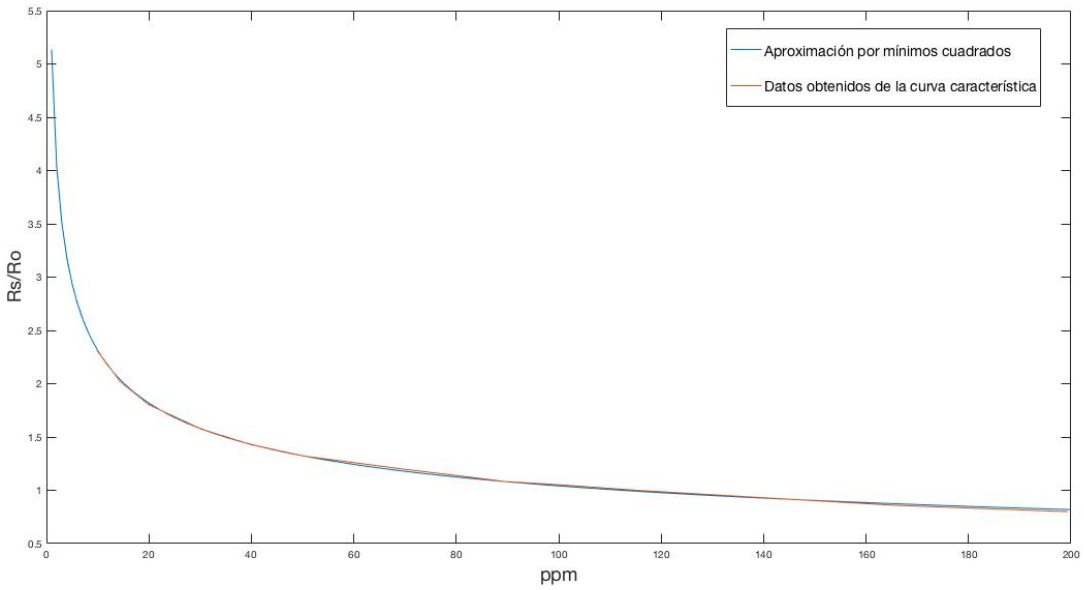


Figura 22. Gráfica obtenida por ajuste de mínimos cuadrados.

La ecuación que se obtuvo obtiene valores para Rs/Ro y lo que se necesita son las partículas por millón (ppm) entonces despejamos a x. Tomando en cuenta las siguientes igualdades Rs/Ro= y, ppm=x.

$$ppm = \left(\frac{\frac{Rs}{Ro}}{5.1356} \right)^{1/-0.34651}$$

Ahora se obtiene los valores para Rs y Ro, la hoja de datos nos da la siguiente ecuación.

$$\frac{Rs}{RL} = \frac{Vc - VRL}{VRL}$$

de la ecuación anterior se despeja a Rs quedando de la siguiente manera. La resistencia Rs tiene que estar entre 30kΩ – 200kΩ según la hoja de datos.

$$Rs = \frac{Vc - VRL}{VRL} RL$$

donde RL es el valor de la resistencia de carga, Vc voltaje de alimentación y VRL voltaje en la resistencia de carga. Para este sensor la RL es de 1kΩ

La resistencia Ro es la resistencia para una cantidad de partículas de gas bajo ciertas condiciones, para lograr medir esta resistencia se necesita de un laboratorio. En este caso ya que no se cuenta con un laboratorio lo que se hace es despejar a Ro, tomando lecturas de un entorno conocido. La cantidad de partículas por millón de CO2 que hay en la atmosfera lamentablemente es de 409.95 [15] tomando en cuenta este valor despejamos a Ro quedando la siguiente igualdad.

$$Ro = \frac{Rs}{5.1356(ppm)^{1/-0.34651}}$$

$$Ro = \frac{Rs}{5.1356(409.95)^{1/-0.34651}}$$

El procedimiento anteriormente descrito es el mismo que se sigue para caracterizar al sensor MQ-9, una diferencia del sensor MQ-9 con el sensor MQ-135 es el principio de operación, mientras que el sensor MQ-135 solamente necesita un voltaje para funcionar, el MQ-9 necesita de dos voltajes. A continuación, se muestra en la **Figura 23** la manera en que trabaja el sensor.

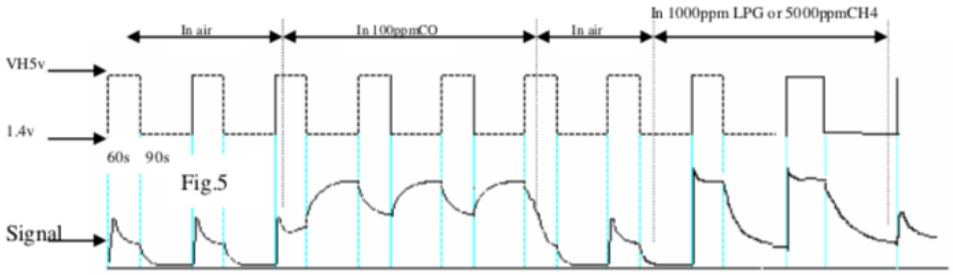


Figura 23. Principio de operación MQ-9[12].

La lectura de la señal se toma después de haber concluido el ciclo en bajo con 1.4V se tiene que ejecutar el ciclo completo para poder obtener una lectura correcta. Este ciclo dura 2.5 minutos.

3.2.2 Programación de la tarjeta de desarrollo Arduino.

La programación se desarrolló en la plataforma de Arduino, se utilizó el software proporcionado por el fabricante ARDUINO IDE versión 1.8.10. Se hizo uso de librerías disponibles para controlar los sensores y módulos. Las librerías utilizadas en el programa son las siguientes:

- TinyGPS++.h
    - La librería TinyGPS++.h se utiliza para controlar el módulo GPS Neo-6m esta librería facilita la lectura de los datos que da el módulo, separando los datos de hora, fecha, longitud, latitud, altitud, numero de satélites conectados. En este caso solo se hizo uso de los datos de latitud y longitud.
- dht11.h
    - La librería dht11.h realiza la comunicación con el sensor de temperatura y humedad, este sensor ocupa la comunicación de un solo bus, separa los datos en 2 variables, uno para la temperatura y otra para la humedad
- SD.h
    - Librería SD.h permite la comunicación con el módulo Micro-SD, esta librería implementa la comunicación SPI entre el Arduino Mega 2560 y el módulo Micro-SD.

- Wire.h
    - Librería Wire.h, esta librería permite la comunicación I2C entre los dispositivos, el módulo Tiny RTC Real Time Clock Module DS1307 utiliza esta comunicación para mandar los datos de fecha y hora.

Una vez que declaradas las variables y definidas las librerías que se utilizaran en el programa se procede con la configuración de pines de entrada/salida e inicialización de protocolos de comunicación.

La rutina principal que se ejecuta cíclicamente controla la dirección a la cual se mandan los datos, estos datos pueden ser mandados por bluetooth a la aplicación desarrollada en Android o guardar los datos en la memoria Micro-SD. A continuación, en la **Figura 24** se presenta el diagrama de flujo de la rutina principal.

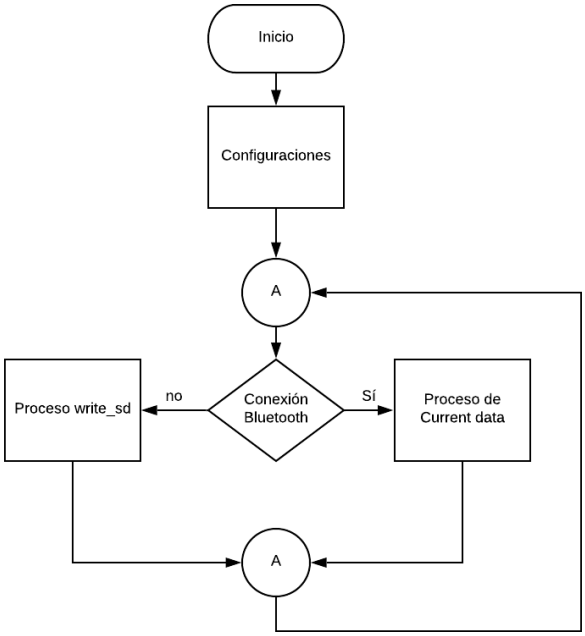


Figura 24. Diagrama de flujo rutina principal.

La rutina de write_sd se utiliza para mandar las lecturas de los sensores y de los módulos a la memoria Micro-SD. El diagrama de flujo de esta subrutina es el siguiente.

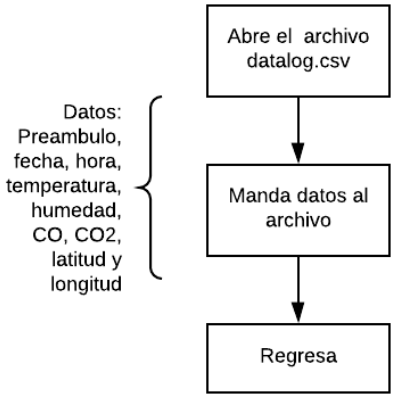


Figura 25. Diagrama de flujo subrutina write_sd.

El diagrama de flujo que se muestra en la **Figura 26** es el de la subrutina Current data, en esta subrutina envían los datos a través de bluetooth utilizando el puerto serial.

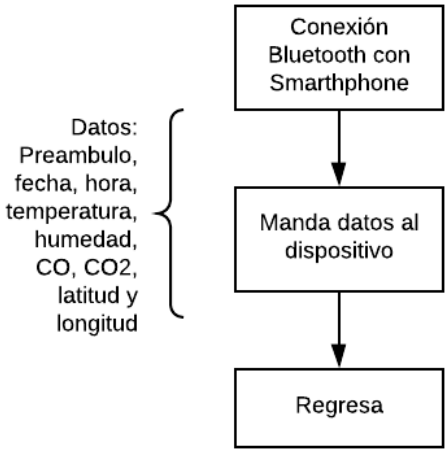


Figura 26. Diagrama de flujo subrutina Current data.

El programa completo se encuentra en el Apéndice B [1].

3.2.3 Diseño de la placa de circuito impreso (PCB).

Una vez probado el sistema electrónico y la programación en tarjeta de prototipos se diseñó una placa de circuito impreso (pcb). Esta se diseñó con la capacidad de poder ser ensamblada como

una tarjeta tipo shield para el Arduino Mega 2560. En la **Figura 27** se pude observar el diseño del pcb.

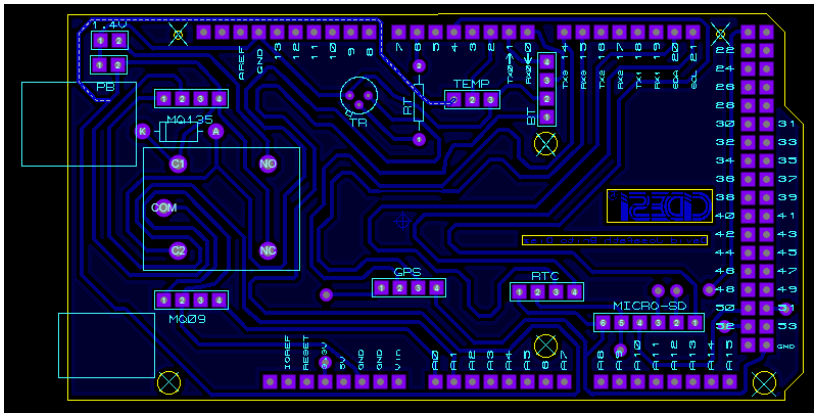


Figura 27. Placa de circuito impreso para el prototipo electrónico.

3.3 Diseño de la aplicación en Android.

La aplicación en Android se desarrolló en la plataforma App Inventor, esta plataforma es impulsada por el Tecnológico de Massachusetts (MIT) con la idea principal de facilitar la programación de aplicaciones en Android. MIT App Inventor es un entorno de programación visual intuitivo que permite construir aplicaciones completamente funcionales para smarthpones y tabletas. La manera en que están diseñados los bloques de programación, permite crear aplicaciones complejas de alto impacto en un tiempo menor, que si se programaran en entornos tradicionales. MIT App Inventor busca facilitar la programación para todas las personas, especialmente los jóvenes impulsando la idea de crear tecnología en vez de consumirla [16].

Un pequeño grupo del Computer Science and Artificial Intelligence Laboratory (CSAIL) y estudiantes, dirigidos por el profesor HAl Abelson , forman el núcleo de un movimiento internacional de inventores. Este equipo mantiene sin costo el entorno de programación, dando le servicio a más de 6 millones de usuarios registrados.

Para hacer uso del entorno de programación se tiene que realizar un registro, en la página http://ai2.appinventor.mit.edu/, una vez realizado el registro se abre una ventana. En esta ventana se selecciona el botón de ‘Comenzar un proyecto nuevo’ **Figura 28**.

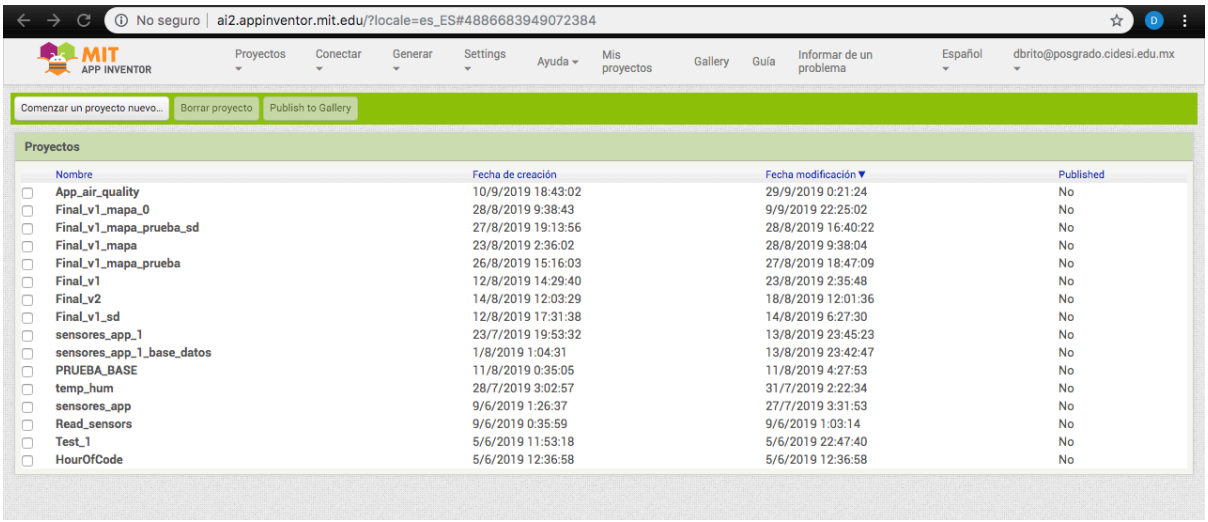


Figura 28. App Inventor ventana de Proyectos.

Una vez creado el proyecto se abre la ventana de diseñador, en esta ventana se desarrolla la interfaz de usuario, en la parte de la izquierda se encuentran la barra de componentes. A continuación, se menciona el contenido de algunas secciones.

- Interfaz de usuario: Se encuentran botones, textos, imágenes, todo lo necesario para poder interactuar con la aplicación.
- Disposición: Se encuentran los bloques con las opciones de disposición para los componentes de la interfaz del usuario ya sea horizontal o vertical.
- Medios: Se encuentran bloques con acceso a cámara, micrófono, reproductor de video, entre otros.
- Sensores: Se pueden encontrar todos los sensores con los cuales cuenta el celular, acelerómetro, sensor de proximidad, etc.
- Social: Se encuentran los objetos necesarios para mandar y escribir archivos de texto a otros usuarios.
- Almacenamiento: Se encuentran los objetos para generar base de datos, ya sea en la nube o en el Smarthphone.
- Conectividad: Se encuentran los bloques como la conexión de bluetooth o servicio web.

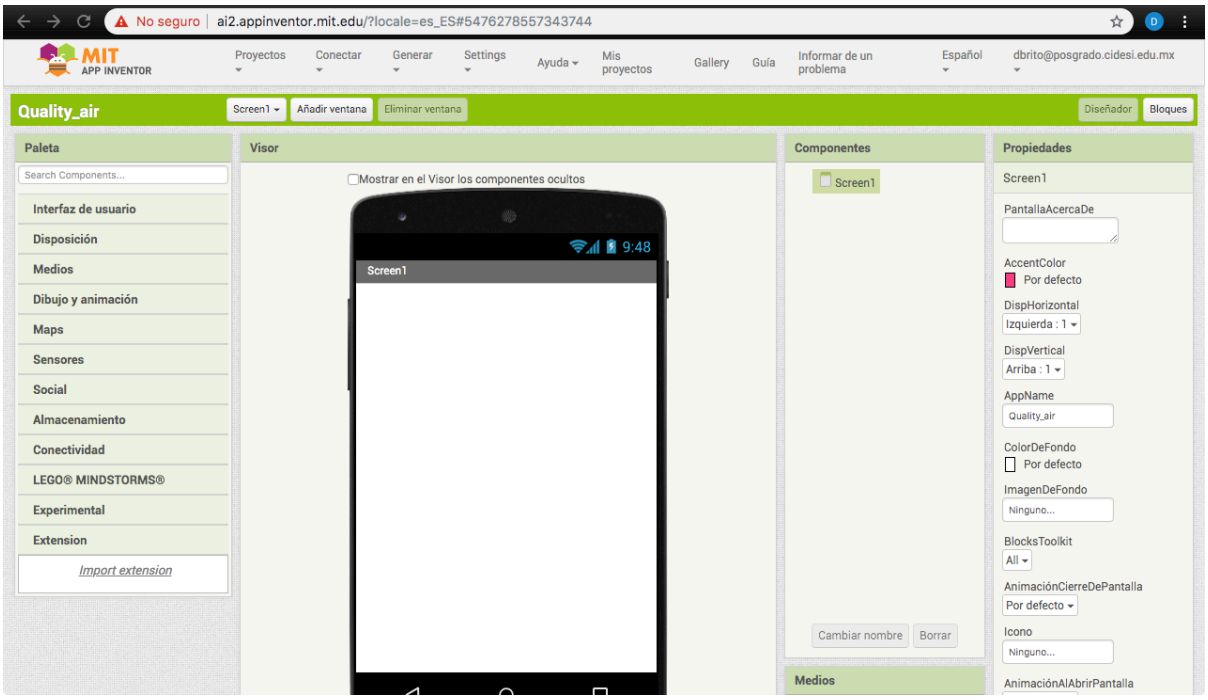


Figura 29. App Inventor entorno de Programación ventana de Diseñador.

Para hacer uso de estos componentes basta con seleccionarlo y arrastrarlo a la ventana del celular, automáticamente se adjunta el componente. Cada vez que se adjunta un componente este aparece en la ventana de la derecha llamada ‘Componentes’. En la ventana de ‘Propiedades’ se editan las características estéticas o funcionales de los componentes que se agregan a la ventana del celular. A continuación, en la **Figura 30** muestra la ventana con la aplicación ya terminada. Como se puede observar se hizo uso de varios componentes de texto, botones y comunicación.

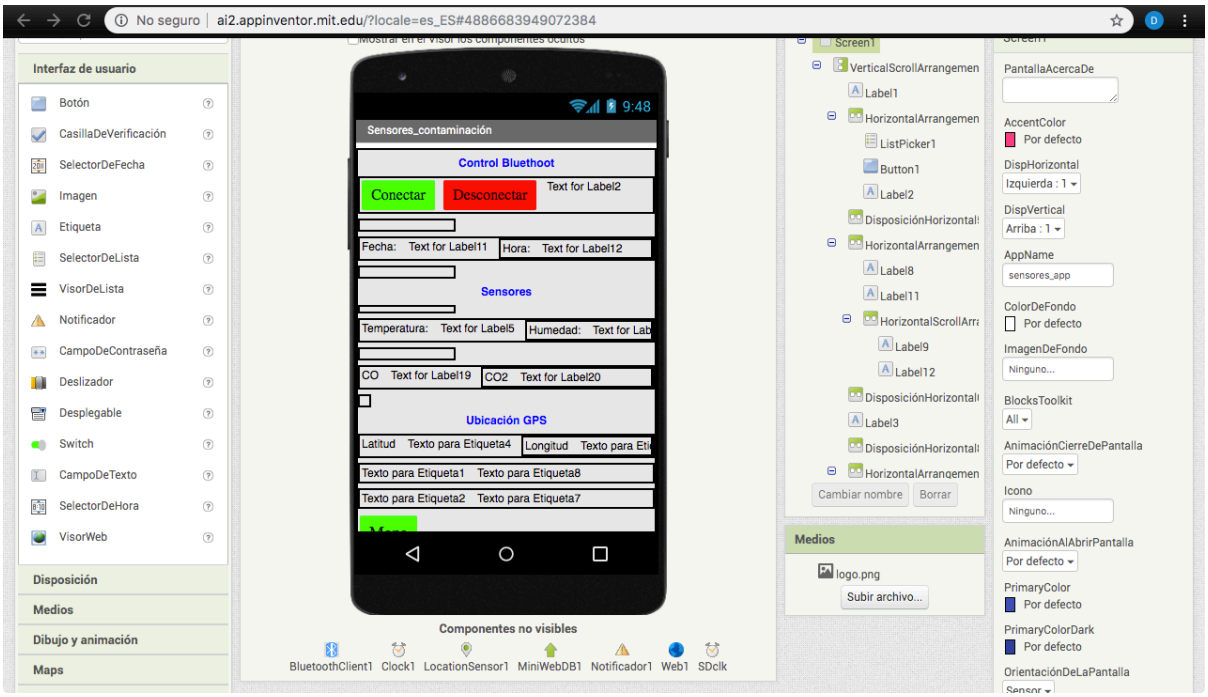


Figura 30. Ventana de Diseñador de la aplicación para el monitoreo de la calidad del aire.

La plataforma de App inventor permiteobservar cómo se ve visualmente la App en los Smartphones mientras se está creando, basta con descargar la aplicación de App inventor llamada MIT AI2 Companion que se encuentra en la Play Store, instalarla en el dispositivo Android y realizar un enlace.

Ahora seleccionamos la ventana de Bloques, en esta ventana se programa con bloques, cada bloque cuenta con distintas funciones, a continuación se da una breve reseña de algunos bloques.

- Bloques de Control: estos se encargan de flujo del programa, se encuentran ciclos tales como if o while.
- Bloque de Lógica: estos se encargan de realizar operaciones lógicas, si o no.
- Bloque de Matemáticas: estos se encargan de las operaciones aritméticas y algunas funciones especiales de operaciones matemáticas.
- Bloque de texto: estos se encargan de realizar modificaciones a los textos.
- Bloque de variables: estos se encargan de crear las variables necesarias para el programa.
- Bloque de procedimiento: estos bloques ayudan a crear subrutinas.

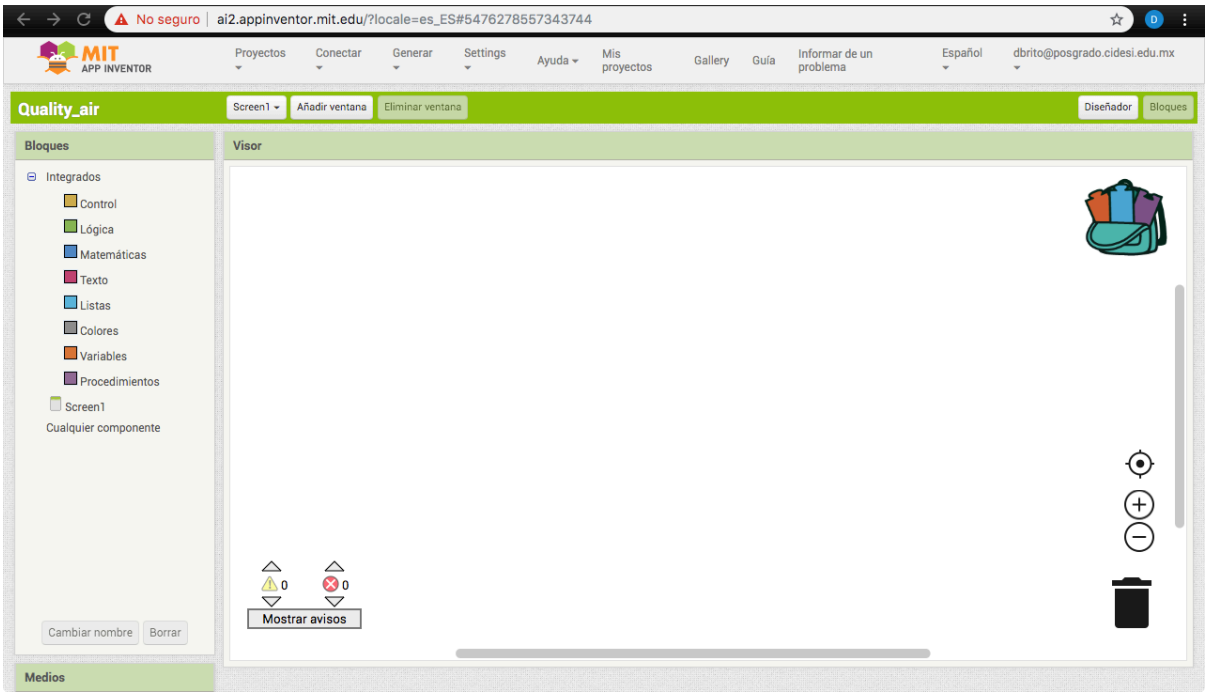


Figura 31. App Inventor entorno de Programación ventana de Bloques.

Al agregar los componentes en la ventana de diseñador, estos automáticamente aparecen en la ventana de bloques como se puede observar en la **Figura 32**, cada objeto cuenta con ciertas opciones, depende de la necesidad de la aplicación cuales son las que se utilizaran en el programa.

En la **Figura 33** se tiene el diagrama de flujo de la aplicación, se inicializan variables después se procede a establecer la comunicación con el bluetooth del dispositivo móvil, en caso de establecer conexión se cargan los datos que envía el dispositivo y estos se despliegan en la pantalla del Smarthphone, por último los datos recibidos se envían a una base de datos web y se reinicia el proceso, por el otro lado en caso de no establecer conexión, se espera hasta que el Smarthphone se conecte al dispositivo.

El programa completo de la aplicación se encuentra en el Apéndice B [2].

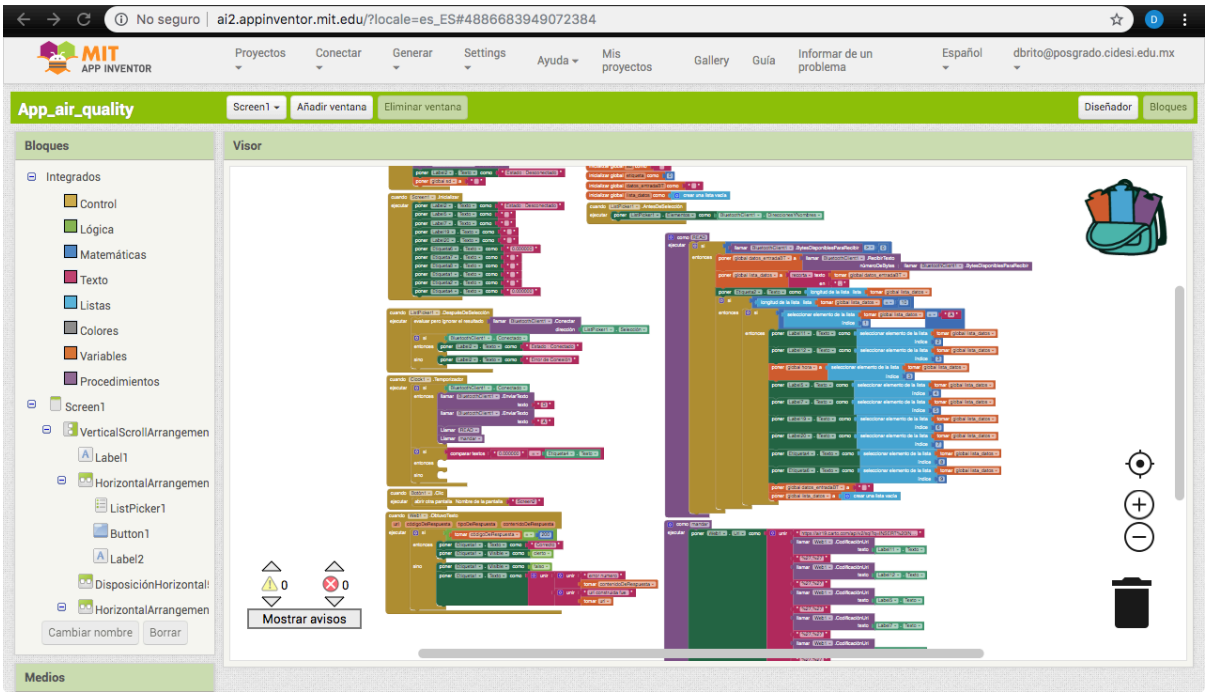


Figura 32. Ventana de Bloques de la aplicación para el monitoreo de la calidad del aire.

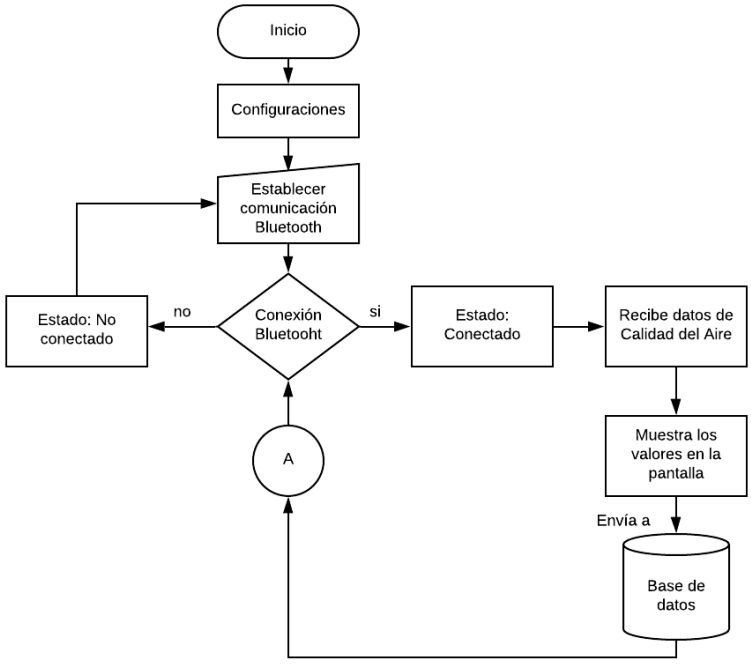


Figura 33. Diagrama de flujo del programa en APP Inventor.

3.4 Implementación de la base de datos.

La base de datos se desarrolló en el servidor de Carto, esta es una herramienta de posicionamiento Geoespacial, se pueden crear bases de datos geoespaciales enormes y a su vez ofrece una herramienta de visualización de datos, facilitando el trabajo de crear mapas dinámicos.

Para poder hacer uso de esta herramienta solo se necesita realizar un registro en la página oficial de CARTO [17]. Una vez realizado el registro se abre una ventana como la que se observa en la **Figura 34.** En la parte superior se encuentran las opciones de 'New map' y 'New dataset', se selecciona dando clic en 'New dataset' para crear la base de datos.

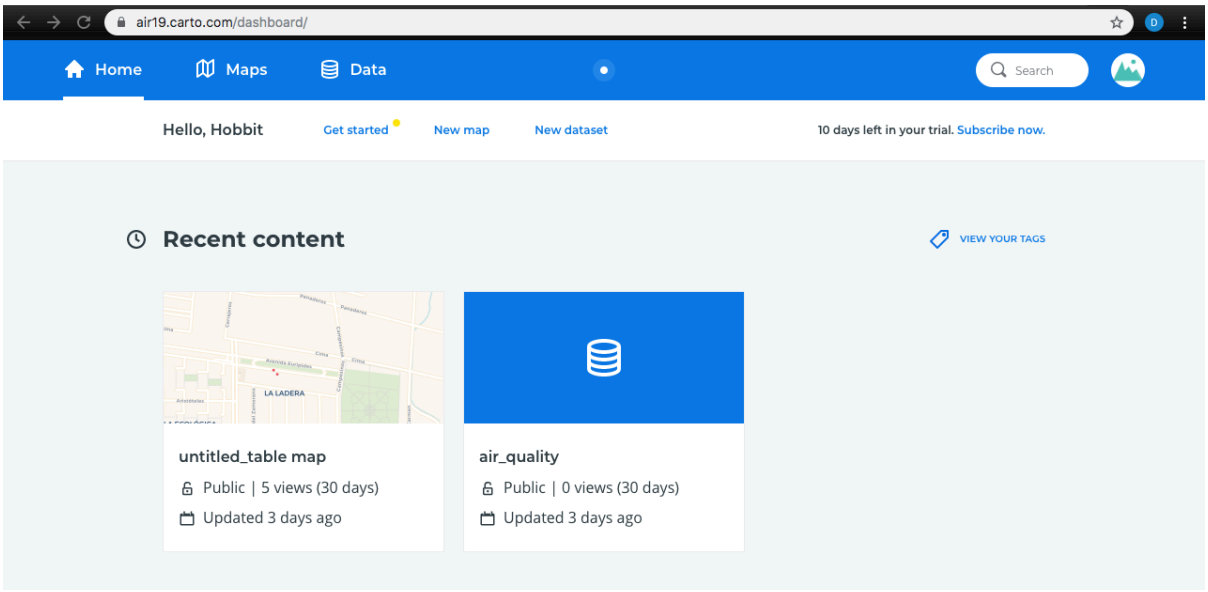


Figura 34. Ventana de inicio CARTO.

En la **Figura 35** se pueden observar las opciones para generar la base de datos, se da la opción de cargar una base de datos desde el ordenador en caso de contar con una, si no, se crea una base de datos vacía seleccionando 'Create empty dataset'. La base de datos que se ha creado cuenta con 4 columnas que vienen por defecto, en la **Figura 36** se observa la ventana con estas columnas, en esta ventana es en donde se agregan todas las columnas necesarias para guardar los datos que se envían desde la aplicación en Android, es importante mencionar que no se debe

de editar ni modificas las columnas de ‘cartodb_id’ y ‘the_geom’ ya que estas son utilizadas para procesar la información a la hora de crear un mapa.

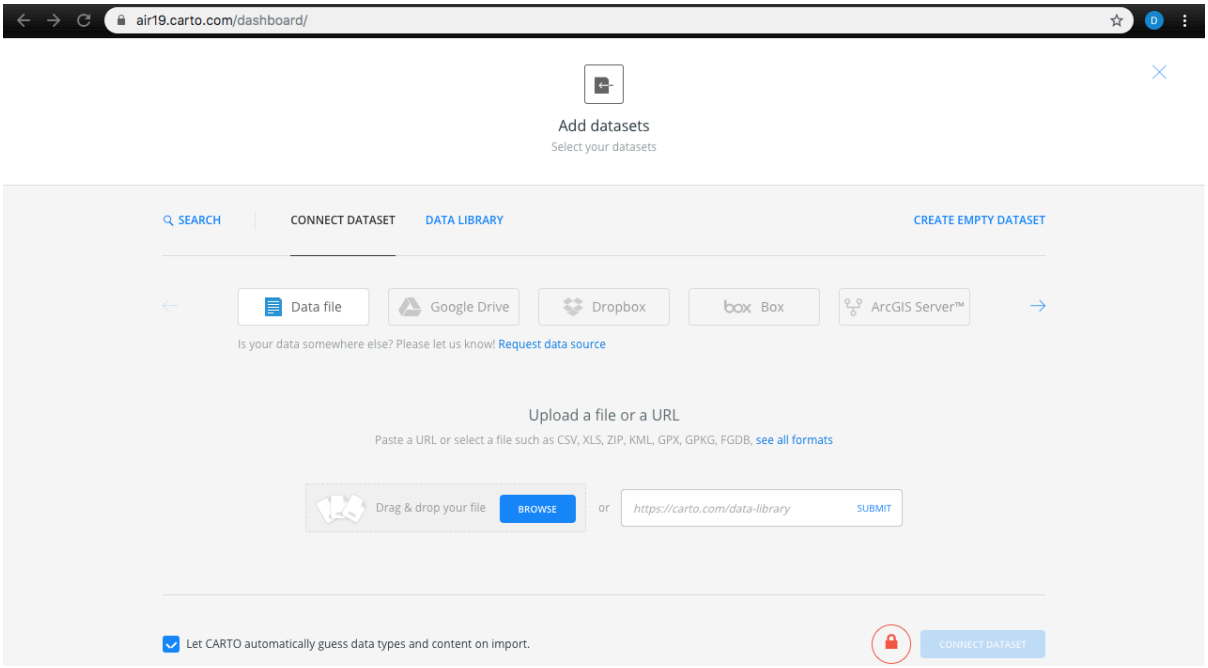


Figura 35. Opciones para base de datos CARTO.

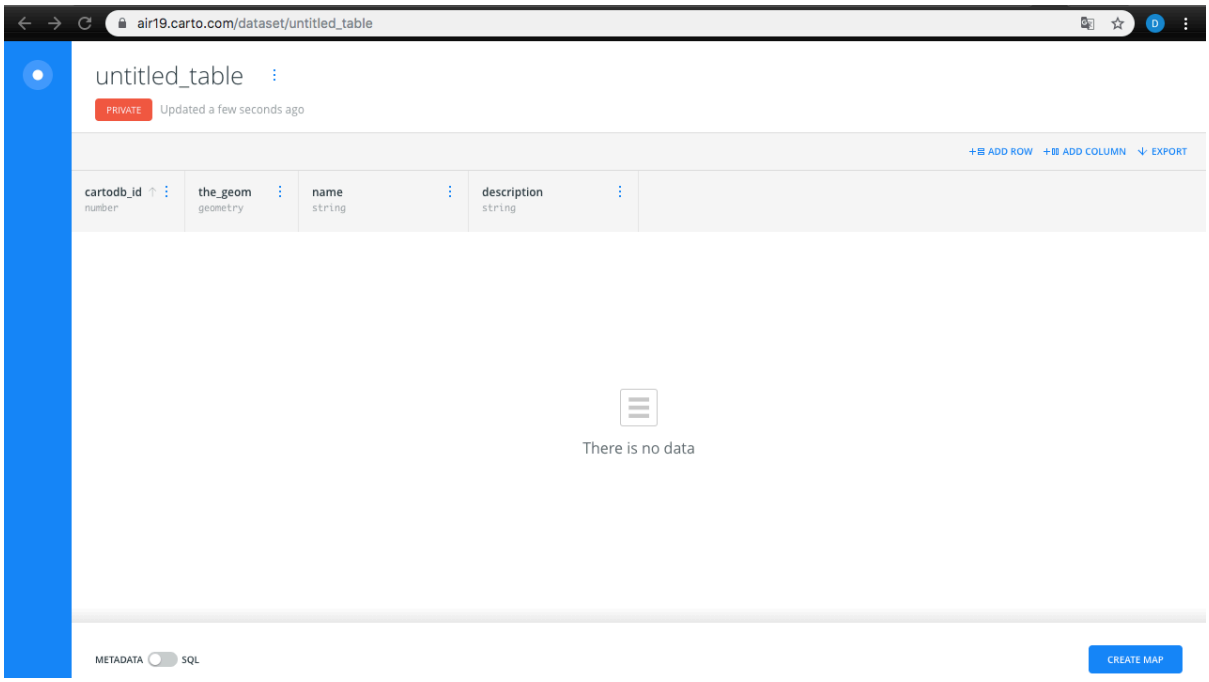


Figura 36. Base de datos CARTO.

Para hacer pública la base de datos y poder interactuar con ella desde cualquier dispositivo, se da clic en el botón rojo que aparece en la parte superior ‘Private’ y seleccionamos la opción de ‘Public’, el botón cambia a color verde después de realizar esta acción **Figura 37**, en la **Figura 38** se observa la base de datos utilizada para la aplicación de Android.

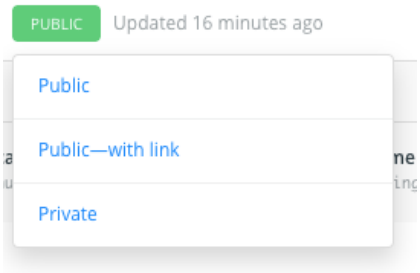


Figura 37. Botón para hacer pública la base de datos CARTO.

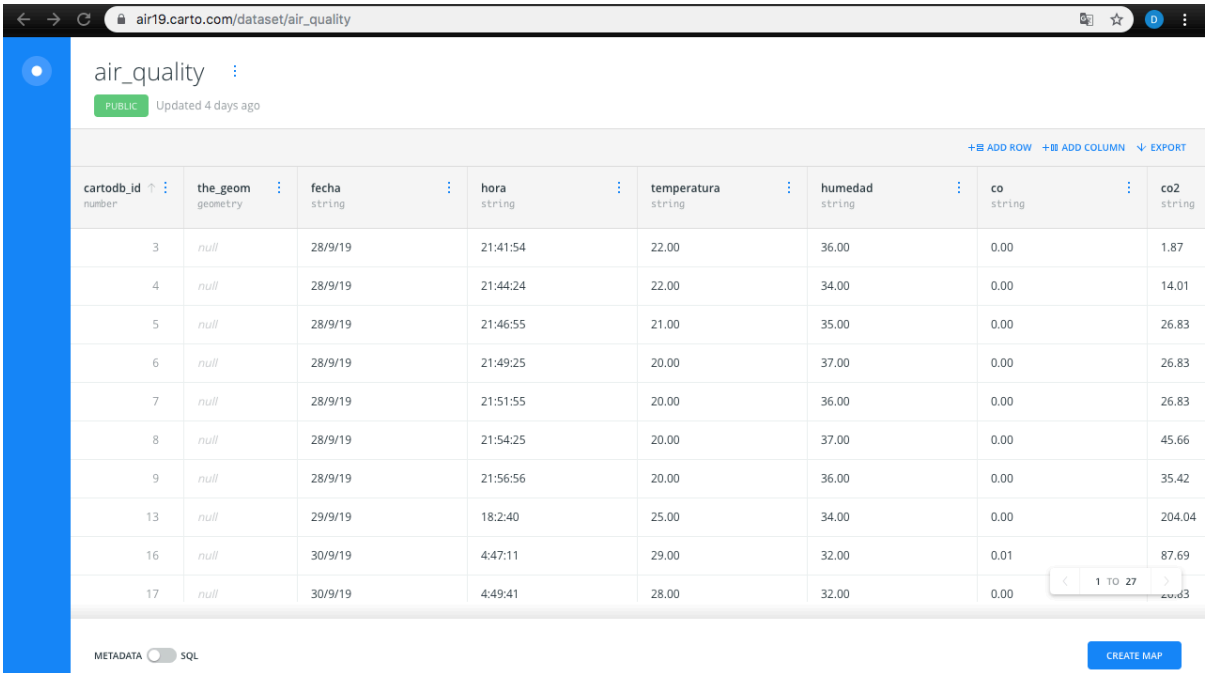


Figura 38. Base de datos para la aplicación de Android.

CARTO cuenta con una herramienta para desplegar la información de la base de datos en un Mapa, para crear este mapa, se selecciona el botón que dice ‘Create Map’ y a continuación se despliega una ventana como la de la **FIGURA 39,** en esta ventana seleccionamos nuestra base de datos y se comienza a configurar los parámetros necesarios para visualizarlos en un mapa. Los datos que utiliza CARTO para geolocalizar los puntos en el mapa son los de latitud y

longitud, estos se tienen que colocar en la opción de ‘Define your parameters’ como se muestra en la **Figura 40**.

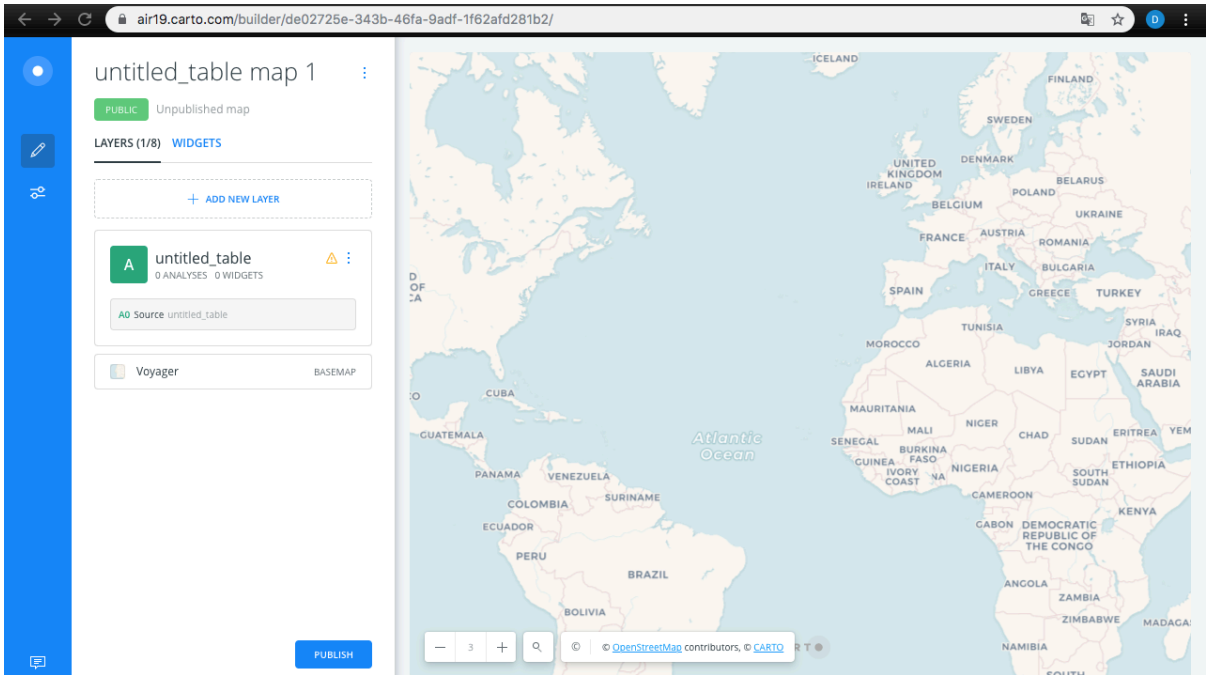


Figura 39. Configuración de mapa para la aplicación de Android.

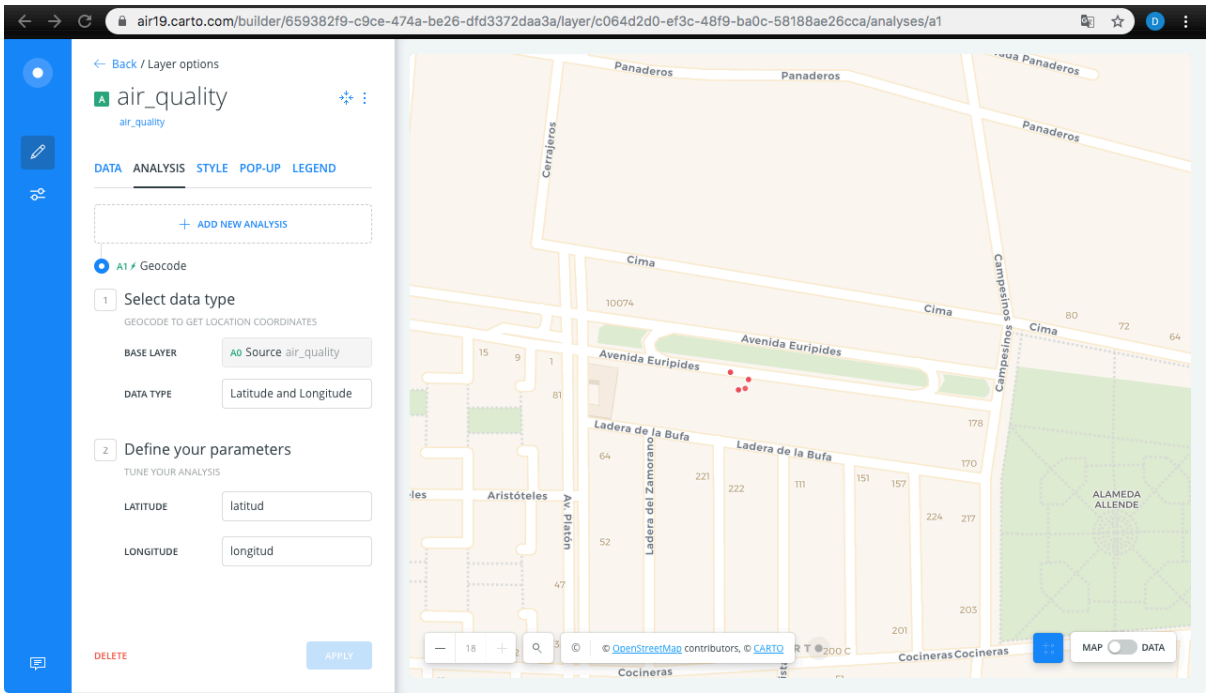


Figura 40. Mapa para la aplicación de Android.

Capitulo 4: Resultados y Conclusiones.

4.1 Resultados.

A continuación, en la **Figura 41,** se presentan imágenes de la placa de circuito impreso con los componentes soldados y ensamblados en la tarjeta de desarrollo de Arduino, como se mencionó anteriormente esta placa fue diseñada como shield para la tarjeta de desarrollo de Arduino Mega 2560, esto quiere decir que se ensambla directo sobre los pines de conexiones de la tarjeta, con esto se hace más compacto el dispositivo y se evitan problemas de desconexión de los módulos y sensores.

Los módulos electrónicos y sensores, no se soldaron directamente a la tarjeta de circuito impreso, ya que se hizo uso de headers, al hacer uso de estos se evita que en caso de una falla en cualquier modulo del dispositivo, se tenga que desoldar el componente de la tarjeta, basta solo con desconectar el módulo de su respectivo header y conectar el nuevo módulo.

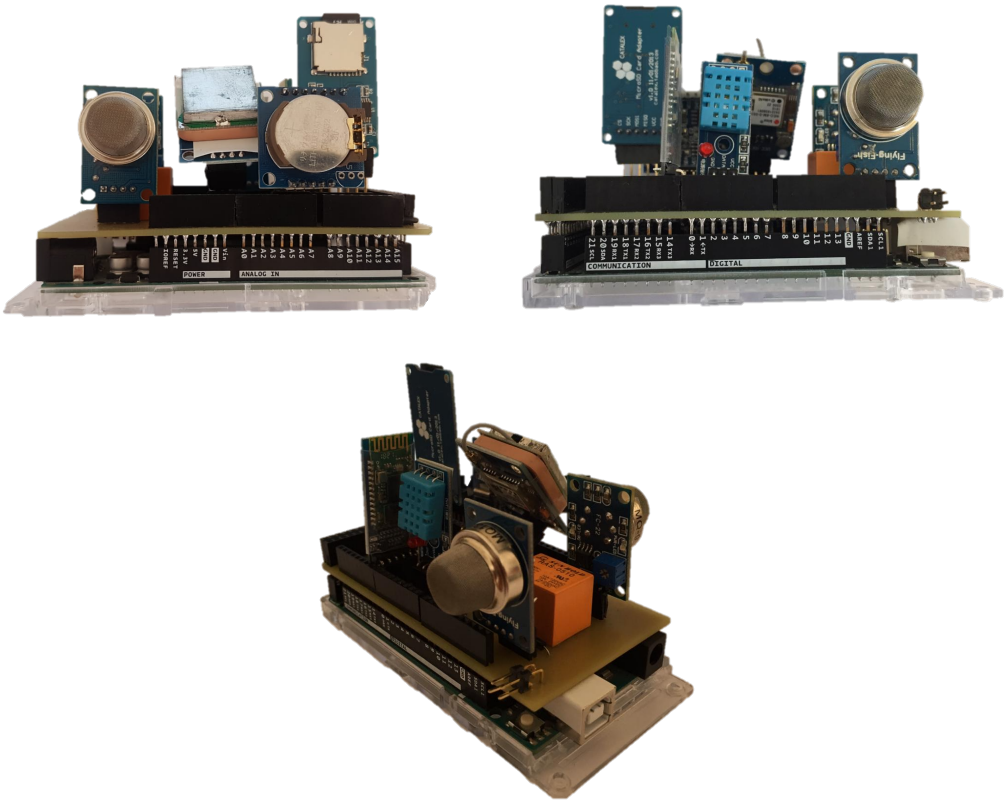


Figura 41. Vistas laterales del dispositivo electrónico.

En la **Figura 42** se observa el dispositivo electrónico energizado, la power bank provee de energía a todo el sistema electrónico y la pila de 1.5V se utiliza para la conmutación de voltaje que necesita el sensor MQ-9 para su correcto funcionamiento.

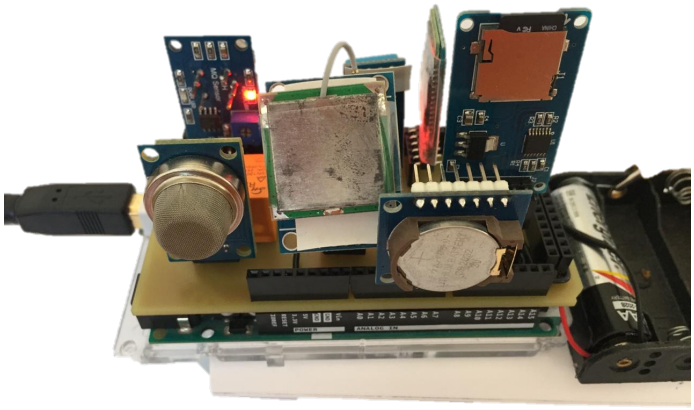


Figura 42. Dispositivo electrónico.

En la **Figura 43** se observa la interfaz de usuario de la aplicación desarrollada en Android, la aplicación está compuesta por 2 ventanas, en la ventana principal se encuentran los botones de control para la conexión del Bluetooth, los datos que envía el dispositivo electrónico, un texto que sirve como indicador para la base de datos y un botón que abre la ventana de mapa. En la ventana de mapa, se encuentra el mapa con puntos en donde se realizaron las mediciones, al seleccionar cualquier punto este despliega la información de la lectura realizada en esas coordenadas.

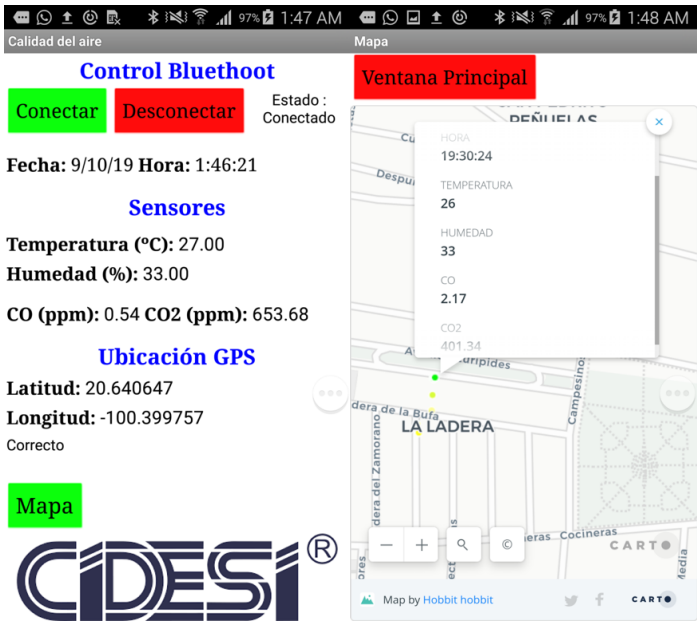


Figura 43. Captura de pantalla del Smartphone a) Ventana Principal b) Ventana mapa.

a) b)

A continuación, se muestra en la **Tabla 6** los costos de los dispositivos electrónicos utilizados.

Tabla 6 Lista de materiales y precio.

Componente.	Precio (pesos mexicano).
Arduino Mega 2560.	652.59
Sensor MQ-135.	66.38
Sensor Mq-9.	66.38
Sensor DHT11.	63.79
Modulo GPS Neo-6m.	360.00
Tiny RTC Real Time Clock Module DS1307.	25.86
Bluetooht HC-06.	102.59
Modulo Micro-SD.	21.55
Relevador.	31.90
Transistor 2n2222a.	13.79
Diodo rectificador.	1.00
Pila de 1.5V.	15.00
Power Bank de 12500 mA.	369.99
Total:	1790.82

4.2 Conclusiones.

Se cumplieron con los objetivos planteados al inicio del proyecto, se desarrolló un dispositivo electrónico con la capacidad de monitorear la calidad del aire, tomando una lectura de CO y CO2 cada dos minutos y medio, a su vez de desarrolló una aplicación en Android con la cual se pueden observar los datos que adquiere el dispositivo electrónico casi en tiempo real y también cuenta con un mapa dinámico, en la cual se pueden consultar los puntos en los que se ha realizado las mediciones. Este mapa se puede consultar desde cualquier dispositivo que tenga acceso a Internet.

El costo final del dispositivo electrónico fue de $1,790.82 pesos mexicanos, con lo cual se cumple que la plataforma tiene que ser de bajo costo, si este precio lo comparamos con nuevas instalaciones de estaciones estáticas, las cuales rondan los miles de pesos.

Algunas consideraciones del sistema a nivel electrónico que se deben de tomar en cuenta son las siguientes, al utilizar los sensores de gas MQ se pudo observar que estos requieren de atenciones especiales ya que tienen un consumo energético considerable y tienden a desajustarse, el funcionamiento del GPS se ve comprometido en lugares cerrados como cuartos, por lo que se debe de tener cerca de una ventana o en espacios abiertos para obtener una buena señal.

A lo largo de este proyecto, me di cuenta de la importancia de los conocimientos adquiridos durante la especialidad en Mecatrónica ya que, estos me sirvieron como bases para poder desarrollar las distintas etapas del proyecto, un reto importante fue realizar las interfaces entre el dispositivo electrónico, la aplicación en Android y la base de datos en plataforma web, porque se tienen que poder interpretar los datos entre cada etapa sin perder información o modificarla en el intercambio de datos entre ellas.

Apendice A:

Programas.

[1] Programa en la plataforma de Arduino.

```
#include <TinyGPS++.h>
#include <dht11.h>
#define DHT11PIN 2
#include <SD.h>
#include <Wire.h>

int lvl;
dht11 DHT11;
File myFile;
TinyGPSPlus gps;
uint8_t second, minute, hour, wday, day, month, year, ctrl;
String LineString = "";
int sd = 0;
int digPin = 6;

void setup()
{
   Serial.begin(9600); // connect serial
   Serial.println("The GPS Received Signal:");
   Serial3.begin(9600); // connect gps sensor
   Wire.begin();
   pinMode(digPin, OUTPUT);
   String LineString = "";
//-------------------------------------------------------
   Serial.print("Iniciando SD ...");
   if (!SD.begin(4))
   {
    Serial.println("No se pudo inicializar");
    return;
   }

   Serial.println("inicializacion exitosa");
   if(!SD.exists("datalog.csv"))
   {
    myFile = SD.open("datalog.csv", FILE_WRITE);
    if (myFile)
    {
     Serial.println("Archivo nuevo, Escribiendo encabezado(fila 1)");
     myFile.println("Preambulo,Fecha,Hora,Temperatura,Humedad,MQ-9,MQ-135,Latitud,Longitud,");
     myFile.close();

    }
    else
    {
     Serial.println("Error creando el archivo datalog.csv");
    }
   }
}

void loop()
{
 while(Serial3.available())   // check for gps data
    {
     if(gps.encode(Serial3.read()))
     {
      int chk = DHT11.read(DHT11PIN);
      char input1 = Serial.read();
```

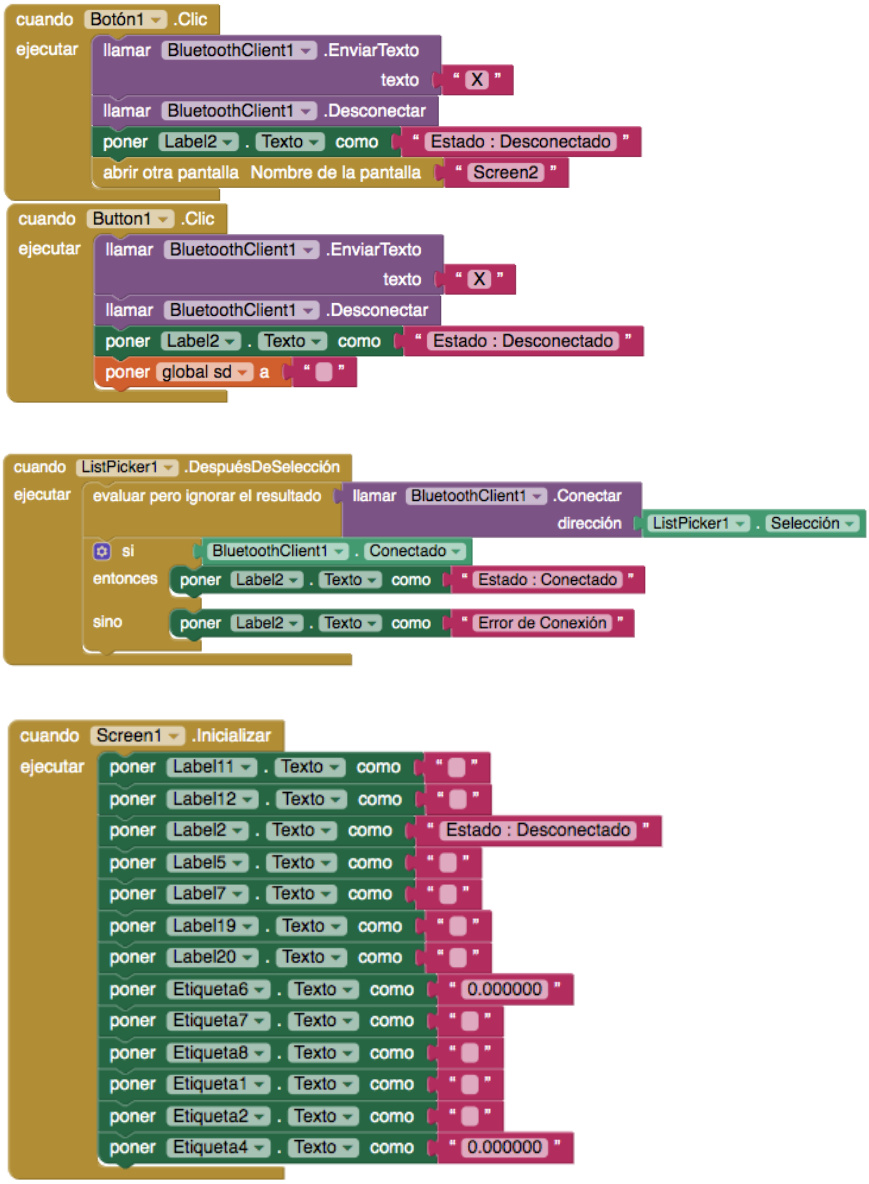
```
        if(input1=='D')
        {
           char input= Serial.read();
           if(input =='M' )
           {
            read_sd();
            input=' ';
           }

           else
           {
            current_data();
            input=' ';
           }
        }
       else
       write_sd();
       input1=' ';
     }

  }
}

////////////////////////////////////////////////////////////////////////////
void read_sd()
{
  myFile = SD.open("datalog.csv", FILE_READ);  // Abre archivo para lectura
  while (myFile.available() != 0) // Lee todos los datos del archivo
      {
        LineString = myFile.readStringUntil('\n');
        Serial.println(LineString);
        delay(200);
       }
   myFile.close();
   Serial.println();
   SD.remove("datalog.csv");

   if(!SD.exists("datalog.csv"))
    {
     myFile = SD.open("datalog.csv", FILE_WRITE);
     if (myFile)
        {
          //Serial.println("Archivo nuevo, Escribiendo encabezado(fila 1)");
          myFile.println("Preambulo,Fecha,Hora,Temperatura,Humedad,MQ-9,MQ-135,Latitud,Longitud,");
          myFile.close();
          }
      else
        {
          Serial.println("Error creando el archivo datalog.csv");
          }
    }
}
////////////////////////////////////////////////////////////////////////////
void current_data()
{
  digitalWrite(digPin, HIGH); // asigna el valor HIGH al pin
  delay(60000);     // espera un cuarto de segundo
  digitalWrite(digPin, LOW); // asigna el valor LOW al pin
  delay(90000);     // espera tres cuartos de segundo
  int adc_MQ09= analogRead(A0);
  digitalWrite(digPin, HIGH);
```

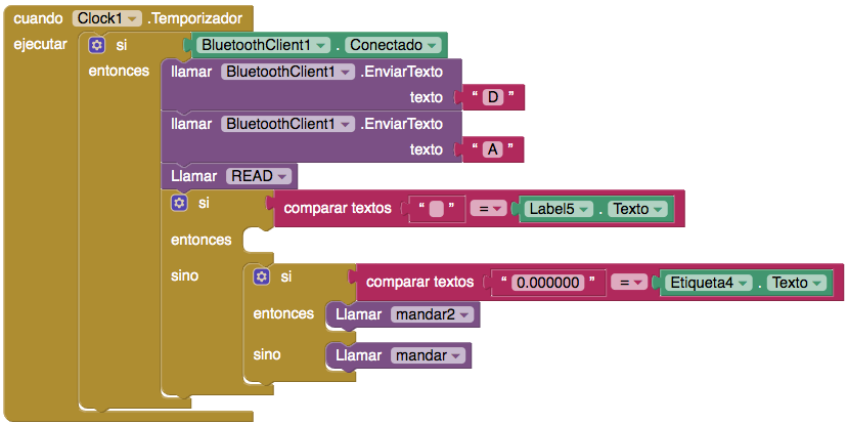
```
  float voltaje2 = adc_MQ09 * (1.4 / 1023.0);
  float Rs2=1000*((1.4-voltaje2)/voltaje2);
  double t1 = ((Rs2/7813.27)/16.5899);
  double ppm1 = pow (t1,-2.2655);
  //-------------------------------------
  int adc_MQ135= analogRead(A1); //Lemos la salida analógica  del MQ
  float voltaje1 = adc_MQ135 * (5.0 / 1023.0); //Convertimos la lectura en un valor de voltaje
  float Rs1=1000*((5-voltaje1)/voltaje1);  //Calculamos Rs con un RL de 1k
  //double alcohol=5.5973*pow(Rs/41000, -.3654); // calculamos la concentración  de alcohol con la ecuación obtenida.
  double t = ((Rs1/143003.07)/5.1356);
  double ppm = pow (t,-2.8859);
  //-------------------------------------
  Serial.print("A");
  Serial.print(",");
  if (read_ds1307()) // Mostrar la fecha y hora
  {
   print_time();
  }
  else // No se puede leer desde le DS1307 (NACK en I2C)
  {
   Serial.println("No se detecta el DS1307, revisar conexiones");
  }
//--------------------Sensor DTH11---------------------------
  Serial.print((float)DHT11.temperature, 2);
  Serial.print(",");
  Serial.print((float)DHT11.humidity, 2);
  Serial.print(",");
//------------------Sensor MQ-135 y MQ-9--------------------
  Serial.print(ppm1);
  Serial.print(",");

  Serial.print(ppm);
  Serial.print(",");
//----------------------GPS---------------------------------
  Serial.print(gps.location.lat(),6);
  Serial.print(",");
  Serial.print(gps.location.lng(),6);
  Serial.print(",");
  Serial.print("\n");
}
////////////////////////////////////////////////////////////////////////////////
void write_sd()
{
   myFile = SD.open("datalog.csv", FILE_WRITE);//abrimos  el archivo
  if (myFile)
  {
  digitalWrite(digPin, HIGH); // asigna el valor HIGH al pin
  delay(60000);     // espera un cuarto de segundo
  digitalWrite(digPin, LOW); // asigna el valor LOW al pin
  delay(90000);     // espera tres cuartos de segundo
  int adc_MQ09= analogRead(A0);
  digitalWrite(digPin, HIGH);
  float voltaje2 = adc_MQ09 * (1.4 / 1023.0);
  float Rs2=1000*((1.4-voltaje2)/voltaje2);
  double t1 = ((Rs2/7813.27)/16.5899);
  double ppm1 = pow (t1,-2.2655);
  //-------------------------------------
  int adc_MQ135= analogRead(A1); //Lemos la salida analógica  del MQ
  float voltaje1 = adc_MQ135 * (5.0 / 1023.0); //Convertimos la lectura en un valor de voltaje
  float Rs1=1000*((5-voltaje1)/voltaje1);  //Calculamos Rs con un RL de 1k
  //double alcohol=5.5973*pow(Rs/41000, -.3654); // calculamos la concentración  de alcoh ol con la ecuación obtenida.
```

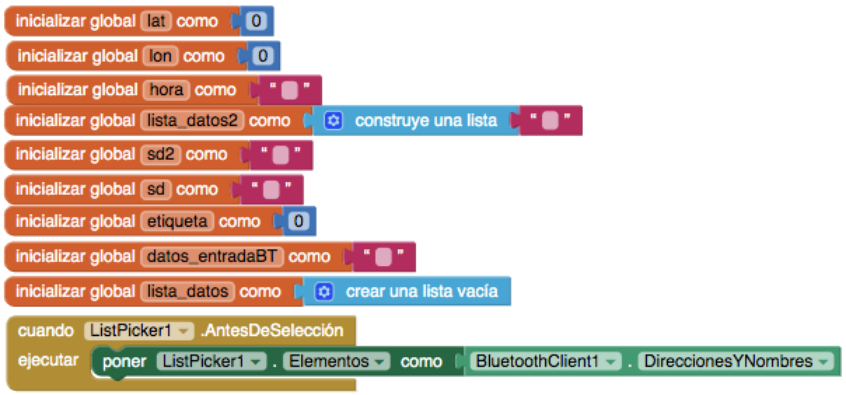
```
  double t = ((Rs1/143003.07)/5.1356);
  double ppm = pow (t,-2.8859);

    myFile.print("A");
    myFile.print(",");
    if (read_ds1307()) // Mostrar la fecha y hora
    {
      print_timesd();
    }
    else // No se puede leer desde le DS1307 (NACK en I2C)
    {
      Serial.println("No se detecta el DS1307, revisar conexiones");
    }
//--------------------Sensor DTH11---------------------------
    myFile.print((float)DHT11.temperature, 2);
    myFile.print(",");
    myFile.print((float)DHT11.humidity, 2);
    myFile.print(",");
//------------------Sensor MQ-135 y MQ-9--------------------
    myFile.print(ppm1);
    myFile.print(",");
    myFile.print(ppm);
    myFile.print(",");
//----------------------GPS--------------------------------
     myFile.print(gps.location.lat(),6);
     myFile.print(",");
     myFile.print(gps.location.lng(),6);
     myFile.print(",");
     myFile.print("\n");
     myFile.close(); //cerramos el archivo
   }
  else
  {
  // if the file didn't open, print an error:
    Serial.println("Error al abrir el archivo");
  }
}
////////////////////////////////////////////////////////////////////////////
bool read_ds1307()
{
  // Iniciar el intercambio de información con el DS1307 (0xD0)
  Wire.beginTransmission(0x68);

  // Escribir la dirección del segundero
  Wire.write(0x00);

  // Terminamos la escritura y verificamos si el DS1307 respondio
  // Si la escritura se llevo a cabo el metodo endTransmission retorna 0
  if (Wire.endTransmission() != 0)
    return false;

  // Si el DS1307 esta presente, comenzar la lectura de 8 bytes
  Wire.requestFrom(0x68, 8);

  // Recibimos el byte del registro 0x00 y lo convertimos a binario
  second = bcd2bin(Wire.read());
  minute = bcd2bin(Wire.read()); // Continuamos recibiendo cada uno de los registros
  hour = bcd2bin(Wire.read());
  wday = bcd2bin(Wire.read());
  day = bcd2bin(Wire.read());
  month = bcd2bin(Wire.read());
```

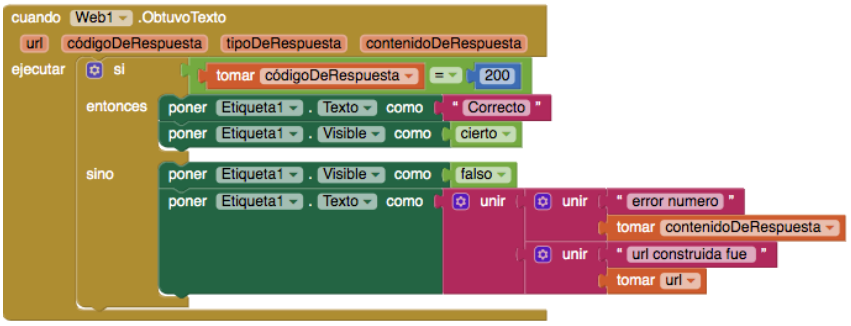
```
  year = bcd2bin(Wire.read());

  // Recibir los datos del registro de control en la dirección 0x07
  ctrl = Wire.read();

  // Operacion satisfactoria, retornamos verdadero
  return true;
}

//-----------Conversion de hora------------
uint8_t bcd2bin(uint8_t bcd)
{
  // Convertir decenas y luego unidades a un numero binario
  return (bcd / 16 * 10) + (bcd % 16);
}

/**
   Imprime la fecha y hora al monitor serial de arduino
*/
void print_time()
{
  //Serial.print("Fecha: ");
  Serial.print(day);
  Serial.print('/');
  Serial.print(month);
  Serial.print('/');
  Serial.print(year);
  Serial.print(",");

  //Serial.print("  Hora: ");
  Serial.print(hour);
  Serial.print(':');
  Serial.print(minute);
  Serial.print(':');
  Serial.print(second);
  Serial.print(",");

  //Serial.println();
}
////////////////////////////////////////////////////////////////////////////
void print_timesd()
{
  //myFile.print("Fecha: ");
  myFile.print(day);
  myFile.print('/');
  myFile.print(month);
  myFile.print('/');
  myFile.print(year);
  myFile.print(",");

  //myFile.print("  Hora: ");
  myFile.print(hour);
  myFile.print(':');
  myFile.print(minute);
  myFile.print(':');
  myFile.print(second);
  myFile.print(",");
  //Serial.println();
}
```

[2] Programa en AppInventor para Android.



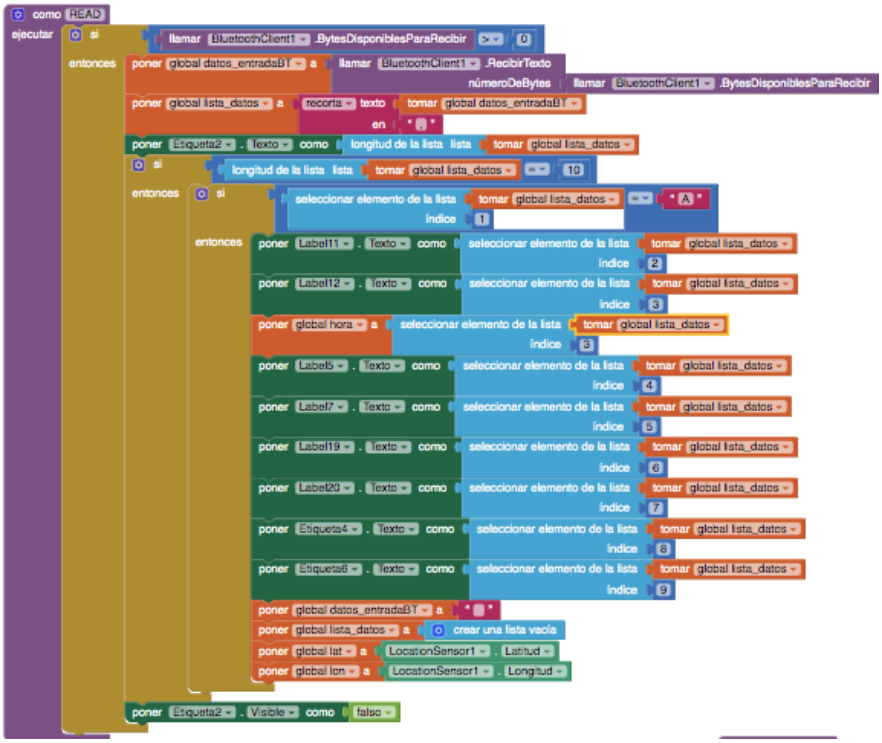

cuando Clock1 .Temporizador
ejecutar si BluetoothClient1 . Conectado
entonces llamar BluetoothClient1 .EnviarTexto
texto " D "
llamar BluetoothClient1 .EnviarTexto
texto " A "
Llamar READ
si comparar textos " " = Label5 . Texto
entonces
sino si comparar textos " 0.000000 " = Etiqueta4 . Texto
entonces Llamar mandar2
sino Llamar mandar


inicializar global lat como 0
inicializar global lon como 0
inicializar global hora como " "
inicializar global lista_datos2 como construye una lista " "
inicializar global sd2 como " "
inicializar global sd como " "
inicializar global etiqueta como 0
inicializar global datos_entradaBT como " "
inicializar global lista_datos como crear una lista vacía
cuando ListPicker1 .AntesDeSelección
ejecutar poner ListPicker1 . Elementos como BluetoothClient1 . DireccionesYNombres


cuando Web1 .ObtuvoTexto
url códigoDeRespuesta tipoDeRespuesta contenidoDeRespuesta
ejecutar si tomar códigoDeRespuesta = 200
entonces poner Etiqueta1 . Texto como " Correcto "
poner Etiqueta1 . Visible como cierto
sino poner Etiqueta1 . Visible como falso
poner Etiqueta1 . Texto como unir unir " error numero "
tomar contenidoDeRespuesta
unir " url construida fue "
tomar url


como mandar
ejecutar
poner Web1 . Url como
unir
" https://air19.carto.com/api/v2/sql?q=INSERT%20IN... "
llamar Web1 .CodificaciónUri
texto Label11 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label12 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label5 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label7 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label19 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label20 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Etiqueta4 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Etiqueta6 . Texto
" %27)&api_key=b9da67df47943caa04ca0f13342d6c75ac4... "
llamar Web1 .Obtener


como mandar2
ejecutar
poner Web1 . Url como
unir
" https://air19.carto.com/api/v2/sql?q=INSERT%20IN... "
llamar Web1 .CodificaciónUri
texto Label11 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label12 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label5 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label7 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label19 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto Label20 . Texto
" %27,%27 "
llamar Web1 .CodificaciónUri
texto tomar global lat
" %27,%27 "
llamar Web1 .CodificaciónUri
texto tomar global lon
" %27)&api_key=b9da67df47943caa04ca0f13342d6c75ac4... "
llamar Web1 .Obtener


como READ
ejecutar
si
llamar BluetoothClient1 .BytesDisponiblesParaRecibir > 0
entonces
poner global datos_entradaBT a llamar BluetoothClient1 .RecibirTexto
número De Bytes llamar BluetoothClient1 .BytesDisponiblesParaRecibir
poner global lista_datos a recorta texto tomar global datos_entradaBT
en " , "
poner Etiqueta2 . Texto como longitud de la lista lista tomar global lista_datos
si
longitud de la lista lista tomar global lista_datos = 10
entonces si
seleccionar elemento de la lista tomar global lista_datos = " A "
índice 1
entonces
poner Label11 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 2
poner Label12 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 3
poner global hora a seleccionar elemento de la lista tomar global lista_datos
índice 3
poner Label5 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 4
poner Label7 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 5
poner Label19 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 6
poner Label20 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 7
poner Etiqueta4 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 8
poner Etiqueta8 . Texto como seleccionar elemento de la lista tomar global lista_datos
índice 9
poner global datos_entradaBT a " "
poner global lista_datos a crear una lista vacía
poner global lat a LocationSensor1 . Latitud
poner global lon a LocationSensor1 . Longitud
poner Etiqueta2 . Visible como falso

Bibliografía.

[1] Organización Mundial de la Salud (2018, Mayo 2), Calidad del aire y salud. Recuperado de https://www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

[2] Fundación Telefónica. (2011). *Smart Cities: un primer paso hacia la internet de las cosas*. Madrid, España: Fundación Telefónica.

[3] Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications, 54*, 1-31.

[4] Alvear, O., Calafate, C. T., Cano, J. C., & Manzoni, P. (2018). Crowdsensing in Smart Cities: Overview, Platforms, and Environment Sensing Issues. *Sensors (Basel, Switzerland)*, *18*(2), 460. doi:10.3390/s18020460

[5] Talasila, M., Curtmola, R., & Borcea, C. (2015). Mobile Crowd Sensing.

[6] Arduni.cl (2019). Recuperado de https://arduino.cl/que-es-arduino/.

[7] Web oficial de Arduino (2019).Recuperado de https://www.arduino.cc/.

[8] Novillo, J., Hernández, D., Mazón, B., Molina, J. y Cárdenas, O. (2018). *Arduino y el Internet de las cosas*. Recuperado el 12 de julio de 2019 de https://books.google.com.mx/books?id=FIlyDwAAQBAJ&lpg=PA4&ots=x887ATBgMp&dq=Arduino%20y%20el%20Internet%20de%20las%20cosas%2C%203Ciencias%2C%202018.&pg=PA4#v=onepage&q=Arduino%20y%20el%20Internet%20de%20las%20cosas,%203Ciencias,%202018.&f=false.

[9] AOSONG, Temperature and Humidity Module DTH11 Product Manual.

[10] Mateev, V., Marinova, I., Kartunov, Z. (2019). *Gas Leakage Source Detection for Li-Ion Batteries by Distributed Sensor Array*. Sensors, 19, 2900.

[11] Technical Data Sensor de gasMQ-135.

[12] Technical Data Sensor de gasMQ-09.

[13] Naylamp Mechatronics, (2019). Recuperado de https://naylampmechatronics.com.

[14] Millan, F. (2016). *Diseño e implementación de un sistema de medida de gases con Arduino* (tesis de licenciatura). Universidad Zaragoza, Teruel, España.

[15] CO2.earth (2019). Recuperado de https://www.co2.earth/.

[16] MIT APP INVENTOR (2019). Recuperado de https://appinventor.mit.edu/.

Bibliografía.

[17] Carto, (2019). Recuperado de https://carto.com/.