COMUNICACIÓN INÁLAMBRICA BLUETOOTH POR MEDIO DE LA TARJETA DE DESARROLLO ESP32

Las herramientas o software por utilizar son las siguientes:

- Arduino IDE 1.8.19 ó posterior
- Plataforma App Inventor
- Tarjeta de desarrollo ESP WROOM 32

Introducción

En esta presente guía que tiene como finalidad realizar la comunicación vía Bluetooth de un dispositivo hacia una aplicación móvil, se brindan las herramientas necesarias para monitorizar variables físicas, biológicas o químicas.

Las principales ventajas de la comunicación Bluetooth son las siguientes:

1. Fácil implementación: La tecnología Bluetooth se ha vuelto muy común en dispositivos de consumo y existen muchos módulos Bluetooth disponibles que facilitan la implementación en sistemas embebidos.
2. Comunicación bidireccional: Bluetooth permite la comunicación bidireccional, lo que significa que los dispositivos conectados pueden enviar y recibir datos entre sí.
3. La tecnología Bluetooth consume muy poca energía, lo que la hace ideal para aplicaciones de bajo consumo de energía en microcontroladores que pueden estar alimentados por baterías.

La tecnología Bluetooth se utiliza en una amplia variedad de aplicaciones en sistemas embebidos y microcontroladores. Se puede utilizar para la comunicación inalámbrica entre dispositivos, como el control remoto de dispositivos, sistemas de automatización del hogar, monitoreo de salud, sistemas de seguimiento y localización, aplicaciones de automoción y comunicación inalámbrica de datos. En general, la tecnología Bluetooth ofrece una forma flexible y confiable de comunicación inalámbrica para sistemas embebidos y microcontroladores en diversas aplicaciones.

En esta ocasión se permitirá hacer uso de la tarjeta de desarrollo ESP32, el cual tiene como característica peculiar la conectividad inalámbrica; cuenta con la integración de los módulos Wifi y Bluetooth, además de ser de muy bajo costo a comparación de otras tarjetas que realizan menos funciones. Además de ello, se usará la plataforma App Inventor, el cual tiene la peculiaridad de desarrollar aplicaciones móviles a través de la programación basada a flujos, el cuál permite al desarrollador del sistema tener un mayor enfoque en la programación del hardware.

Instalación de software, firmware y librería.

Para comenzar, es necesario contar con el entorno de Arduino IDE, el cual puede descargarse a través de la siguiente dirección: https://www.arduino.cc/en/software

Elaboró:
Samuel Moisés Flores Aguirre

Actualmente se encuentra disponible la versión 1.8.19 de Arduino IDE, sin embargo, recientemente se lanzó una versión mejorada, la versión 2.0.4, que también puede ser descargada en la misma dirección antes mencionada.

Una vez descargado el entorno de Arduino, es necesario contar con el firmware de la tarjeta de desarrollo ESP32, para ello es necesario abrir la aplicación de Arduino IDE y dirigirse hacia la sección de Archivo > Preferencias y en el campo de texto "Gestor de URLs Adicionales de Tarjetas" se ingresa la siguiente dirección: https://dl.espressif.com/dl/package_esp32_index.json, como se puede observar en la Figura 1.

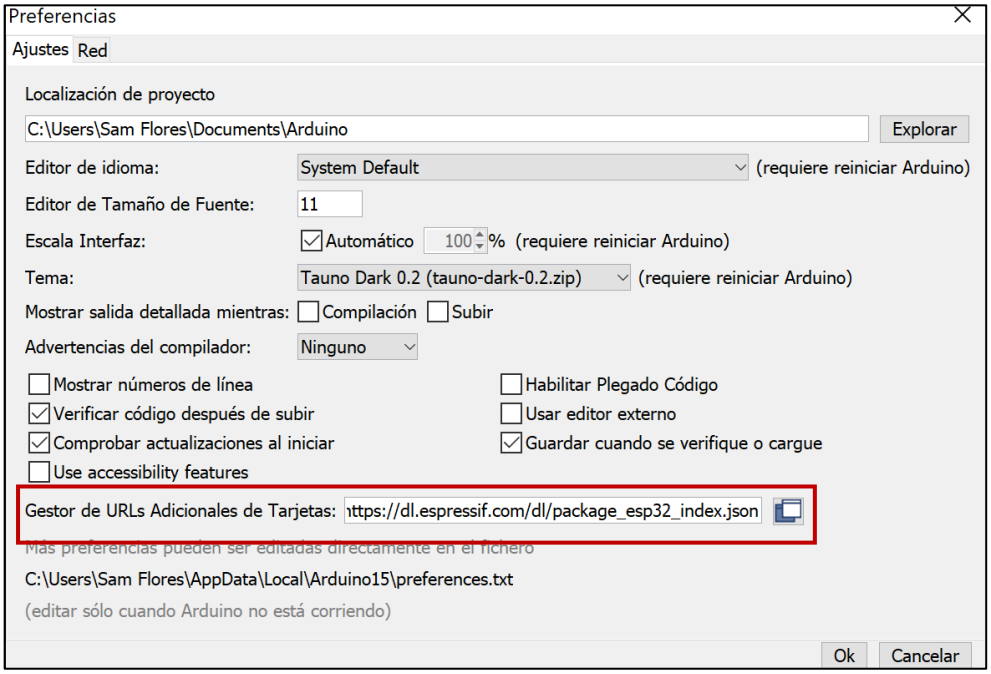


Figura 1

Una vez que se haya ingresado la dirección en el "Gestor de URLs" se selecciona el botón "Ok" para guardar las modificaciones efectuadas.

Con la Url ya ingresada, se puede seleccionar la tarjeta a utilizar, que en este caso es la tarjeta de desarrollo ESP32. Para seleccionarla, se debe dirigir hacia los apartados o secciones del IDE de Arduino. En la sección de "Herramientas" y en donde se encuentra "Placa" se selecciona y se debe dirigir hacia el "Gestor de tarjetas". Ver Figura 2.

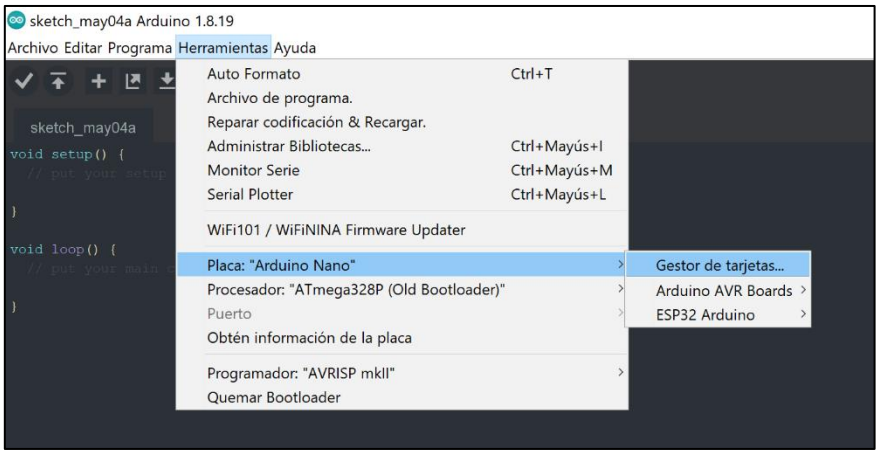


Figura 2

Elaboró:
Samuel Moisés Flores Aguirre

En el “Gestor de tarjetas“, se selecciona el buscador y se escribe la palabra “esp32”. Se procede a instalar el paquete y se espera a que termine el proceso de instalación, una vez terminado, se selecciona el botón “cerrar”. Ver Figura 3.

Figura 3

Creación de cuenta en App Inventor

Para ingresar a la plataforma de App Inventor y diseñar la aplicación móvil es necesario crear una cuenta de usuario, para ello se ingresa a la siguiente dirección web: https://appinventor.mit.edu/ y se selecciona en el botón de “Create Apps!”. Ver Figura 4

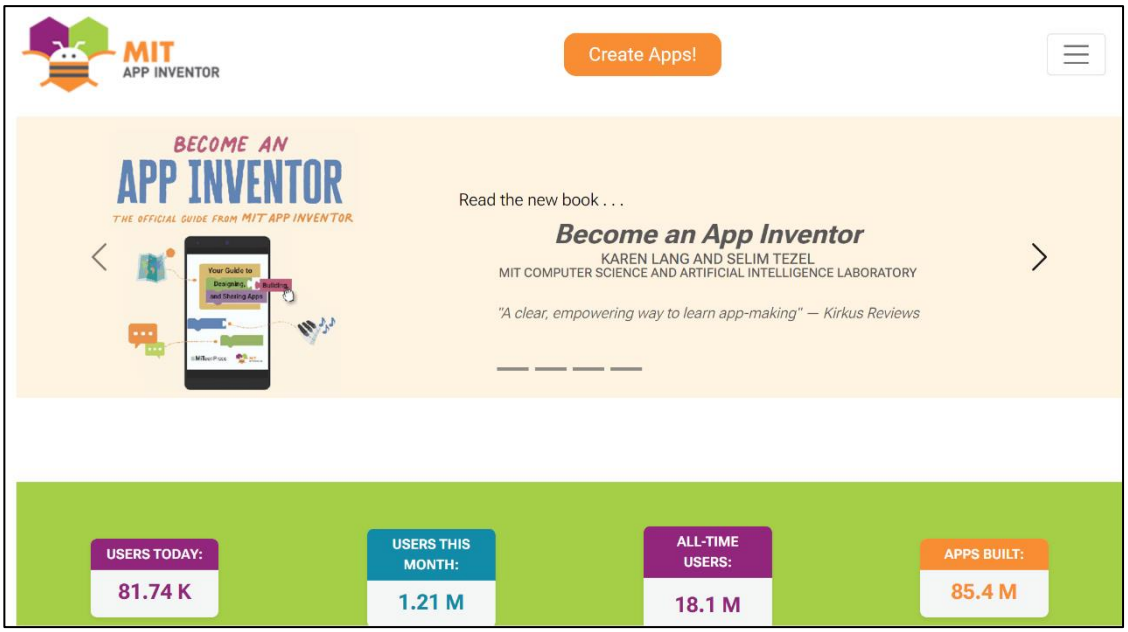


Figura 4

Elaboró:
Samuel Moisés Flores Aguirre

Se redirecciona para iniciar sesión mediante su cuenta de Google, puede seleccionar cualquier cuenta que tenga disponible para registrarse en App Inventor. Ver Figura 6.

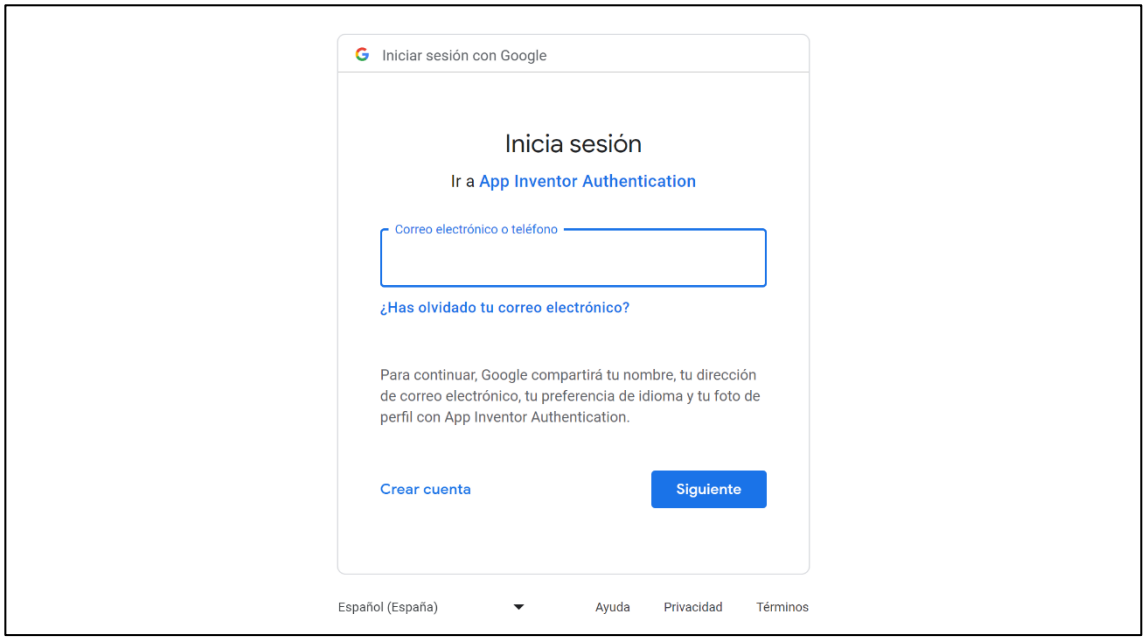


Figura 6

Después de que se haya registrado con su cuenta, podrá observar un panel como el que se encuentra en la Figura 7. Conforma vaya creando nuevos proyectos, podrá visualizarlos en ese miso menú.

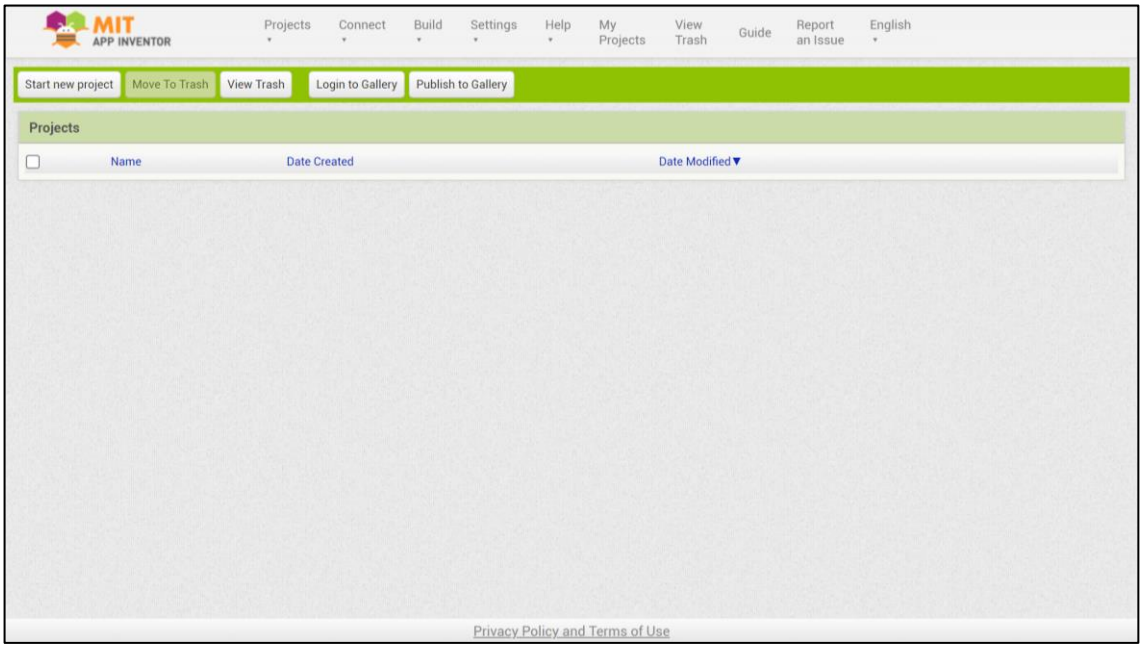


Figura 7

Elaboró:
Samuel Moisés Flores Aguirre

Desarrollo de código en Arduino para comunicación Bluetooth

Para la comunicación Bluetooth, es necesario contar con una librería especial que otorga Arduino al momento de instalar la biblioteca del dispositivo. En este caso, al momento de que se instalo el firmware del ESP32, se instalaron algunas librerías como lo es el caso de la librería "BluetoothSerial". El código base propuesto, se puede observar en las siguientes líneas:

```
#include <BluetoothSerial.h>


#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

void setup() {
       Serial.begin(9600);
       SerialBT.begin("BluetoothESP32");
}


void loop() {
             float valor1,valor2;
             if (isnan(valor1) || isnan(valor2)) {
                return;
             }

              SerialBT.print(valor1);
              SerialBT.print(";");
              SerialBT.print(valor2);
              SerialBT.println(";");
}
```

A continuación, se explicará con mayor detalle lo que realiza cada línea de código para su mayor comprensión.

Elaboró:
Samuel Moisés Flores Aguirre

Se inserta la librería “BluetoothSerial.h” en el código para utilizar las funciones y realizar la comunicación Bluetooth, para este caso, no es necesario descargar e incluir ninguna librería, al momento de instalar el firmware del ESP32, viene por defecto.

```
#include <BluetoothSerial.h>
```

En la siguiente línea se verifica si dos macros (CONFIG_BT_ENABLED y CONFIG_BLUEDROID_ENABLED) están definidos. Si alguna de las macros no está definida, el preprocesador genera un error con el mensaje "Bluetooth is not enabled! Please run make menuconfig to and enable it".

CONFIG_BT_ENABLED es una macro que indica si el soporte de Bluetooth está habilitado o no en la plataforma. Si esta macro no está definida o tiene un valor de 0, significa que el soporte de Bluetooth no está habilitado.

CONFIG_BLUEDROID_ENABLED es una macro que indica si la pila de protocolos Bluetooth BlueDroid está habilitada o no en la plataforma

```
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
```

BluetoothSerial SerialBT; es una declaración de variable que crea un objeto BluetoothSerial llamado SerialBT. Este objeto se utiliza para establecer una conexión Bluetooth con otro dispositivo y enviar y recibir datos a través de esa conexión.

```
BluetoothSerial SerialBT;
```

En la función void setupo() se realiza la comunicación serial a 9600 baudios, mientras que SerialBT.begin(“BluetoothESP32”) es una llamada al método begin() del objeto SerialBT de la clase BluetoothSerial. El método begin() se utiliza para iniciar la comunicación Bluetooth serie y configurar la conexión.

```
void setup() {
      Serial.begin(9600);
      SerialBT.begin("BluetoothESP32");
}
```

En este caso, se está pasando una cadena de caracteres "BluetoothESP32" como parámetro del método begin(). Este parámetro se utiliza para establecer el nombre del dispositivo Bluetooth visible para otros dispositivos.

Elaboró:
Samuel Moisés Flores Aguirre

En la función void loop, se crean las variables donde se almacenarán los valores sensados, que de acuerdo al usuario, estos serán definidos de acuerdo al sistema propuesto.

```
void loop() {
  float valor1,valor2;
  if (isnan(valor1) || isnan(valor2)) {
    return;
  }

  SerialBT.print(valor1);
  SerialBT.print(";");
  SerialBT.print(valor2);
  SerialBT.println(";");
}
```

La condición isnan(t) || isnan(h) comprueba si cualquiera de las dos variables es NaN. Si alguna de las dos variables es NaN, el código devuelve inmediatamente a la función que lo llamó utilizando la instrucción return. Esto significa que el código no ejecutará ninguna acción adicional si alguna de las dos variables es NaN.

Estas líneas de código utilizan el objeto SerialBT para enviar datos a través de la conexión Bluetooth serie. Se envían los valores sensados.

NOTA: La función isnan() es una función matemática que se utiliza para verificar si un número es "no un número" (NaN). En este caso, se está utilizando para verificar si los valores de t y h son NaN.

Ejemplo: Monitoreo de sensor DHT11

Teniendo en cuenta lo antes mencionado, para este ejemplo se realizará el monitoreo del sensor DHT11 que cuenta con la peculiaridad de leer las variables físicas de la temperatura y la humedad. Las conexiones se pueden visualizar en el siguiente esquema. Ver Figura 8.

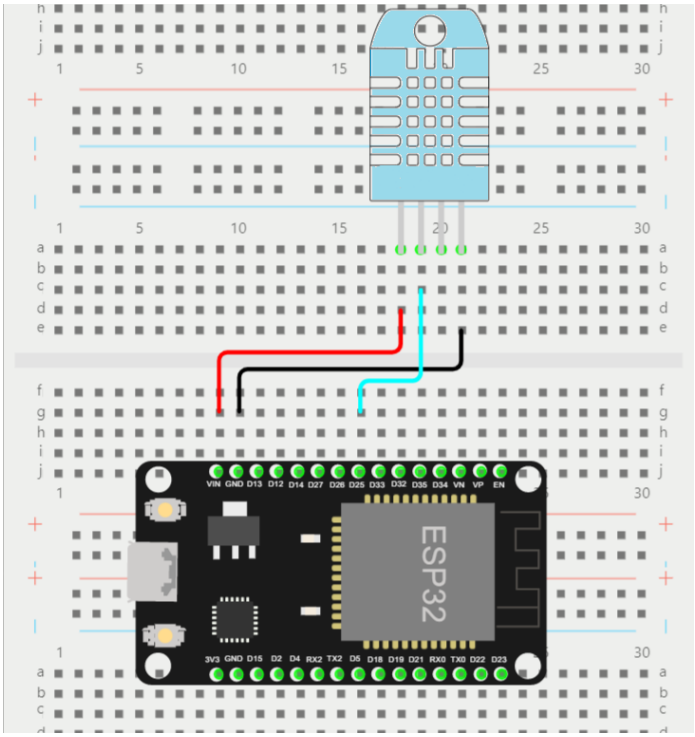


Figura 8

Elaboró:
Samuel Moisés Flores Aguirre

Diseño de Aplicación móvil en App Inventor

Se crea un proyecto nuevo en la sección que dice “Start new project” y se le escoge un nombre a la aplicación. Ver Figura 9.

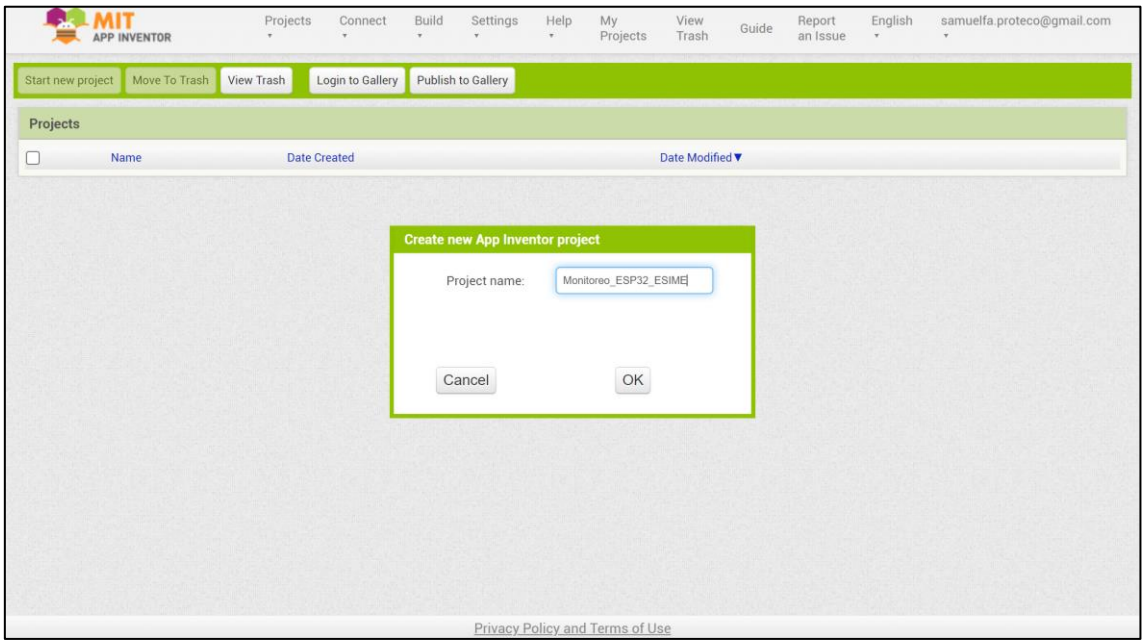


Figura 9

Al momento de crear el proyecto podrá visualizar entorno de desarrollo, donde podrá seleccionar los componentes necesarios para su aplicación. Ver Figura 10.

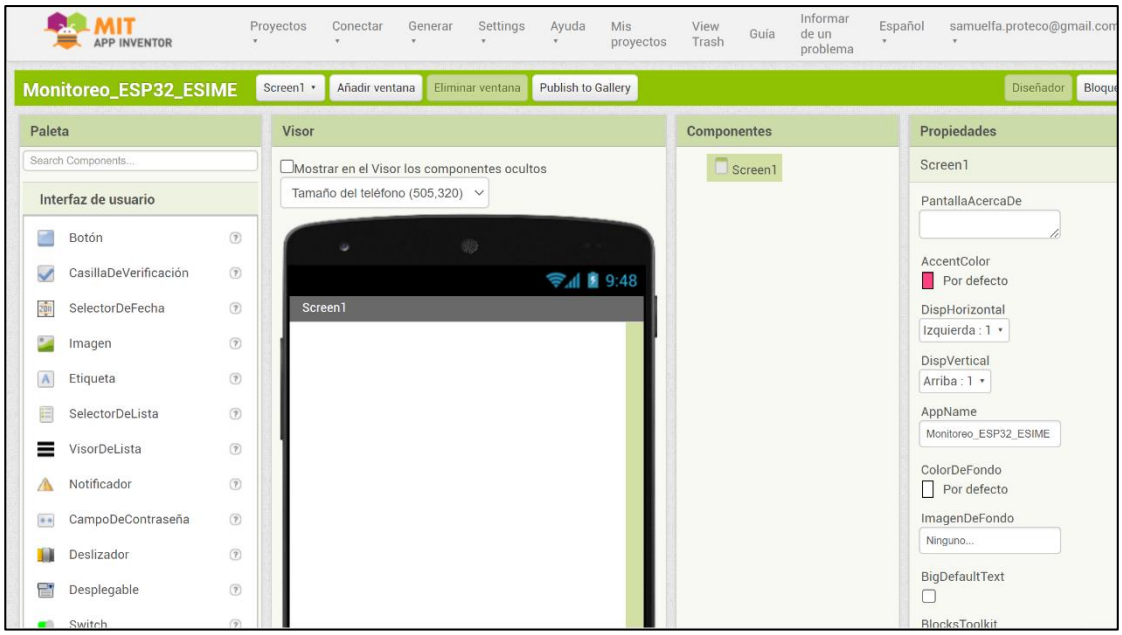


Figura 10

Elaboró:
Samuel Moisés Flores Aguirre

Se inserta un elemento de disposición horizontal. Ver Figura 11.

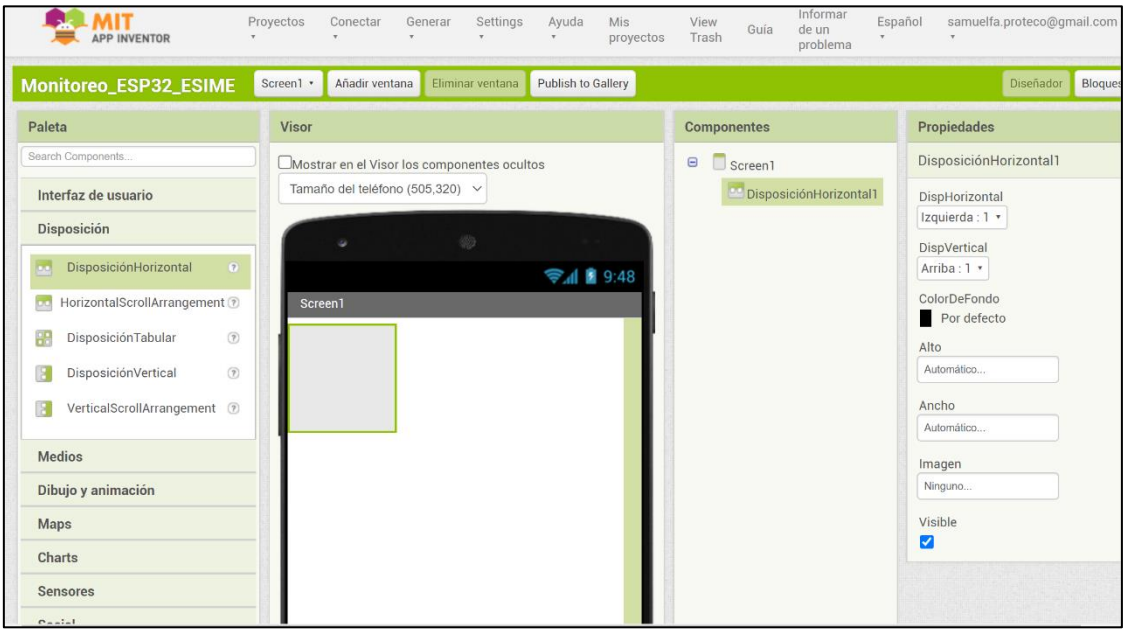


Figura 11

Se coloca una etiqueta en la disposición, se puede cambiar el color de letra, el tipo de letra y el tamaño. Ver Figura 12.

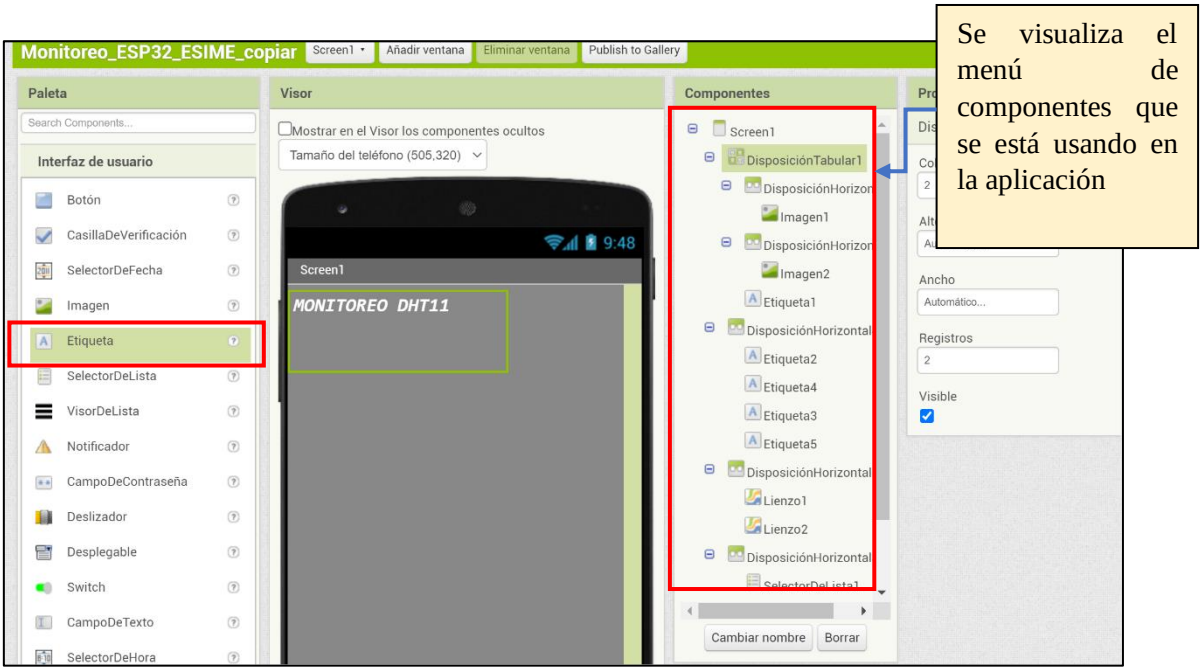


Figura 12

Elaboró:
Samuel Moisés Flores Aguirre

Se cambia el color de fondo en el componente Screen1, el cuál representa la pantalla completa de la app. Ver Figura 13.

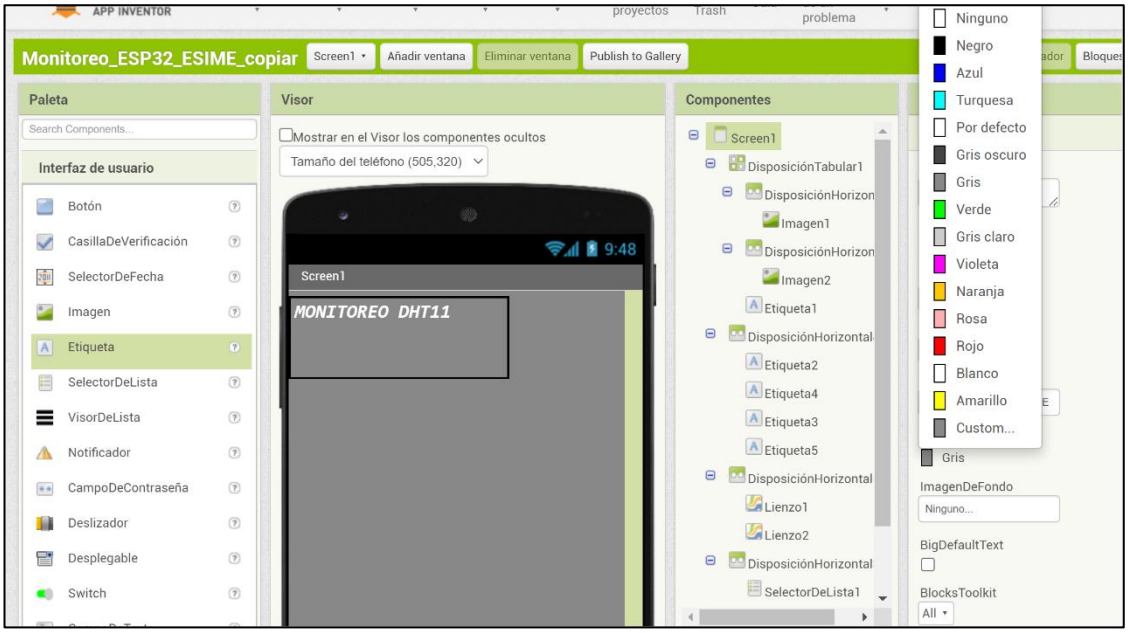


Figura 13

Se agrega otra disposición horizontal dentro de la disposición horizontal antes añadida. Ver Figura 14.

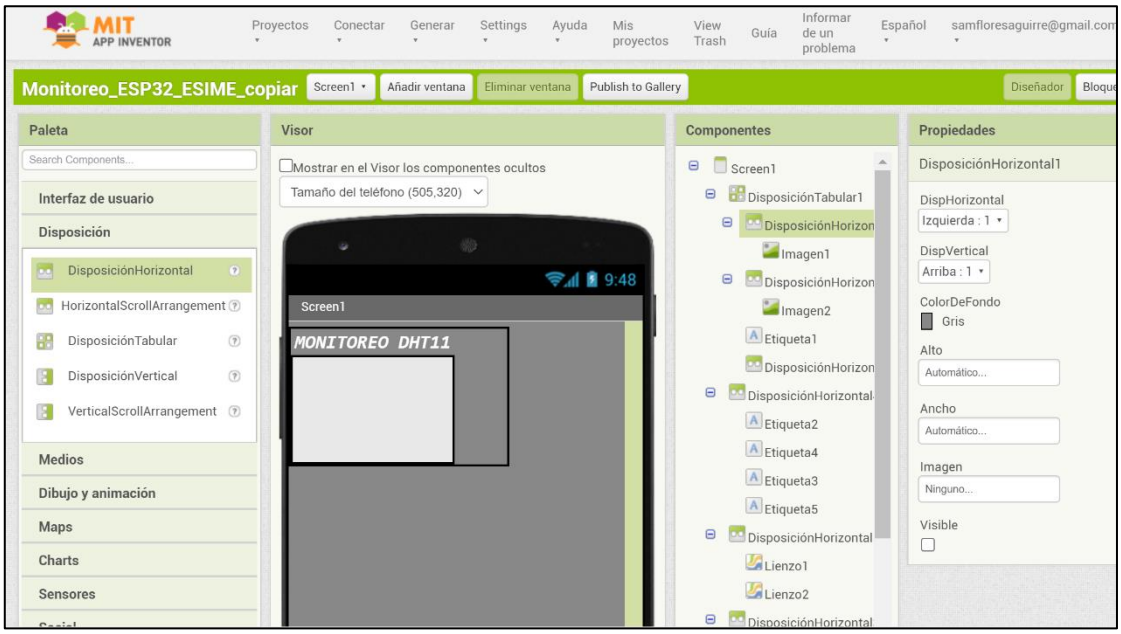


Figura 14

Elaboró:
Samuel Moisés Flores Aguirre

Se agrega el componente imagen a la disposición horizontal creada y se sube un archivo, posteriormente se selecciona. Ver Figura 15.

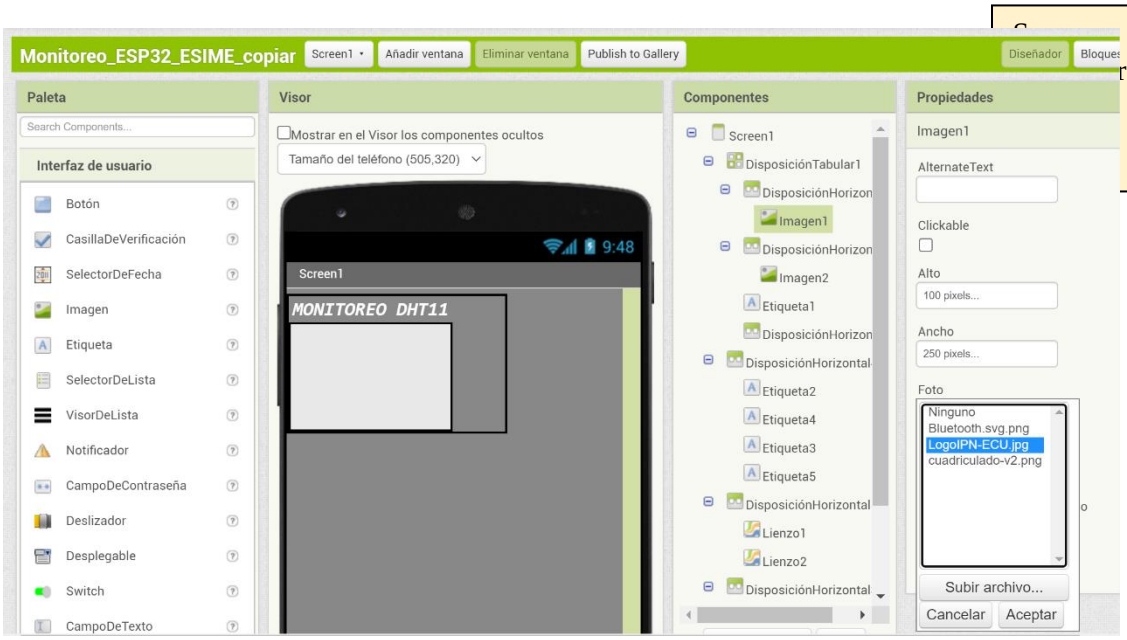


Figura 15

Se selecciona la imagen y al modificar su tamaño, se podrá visualizar en la app. Ver Figura 16.

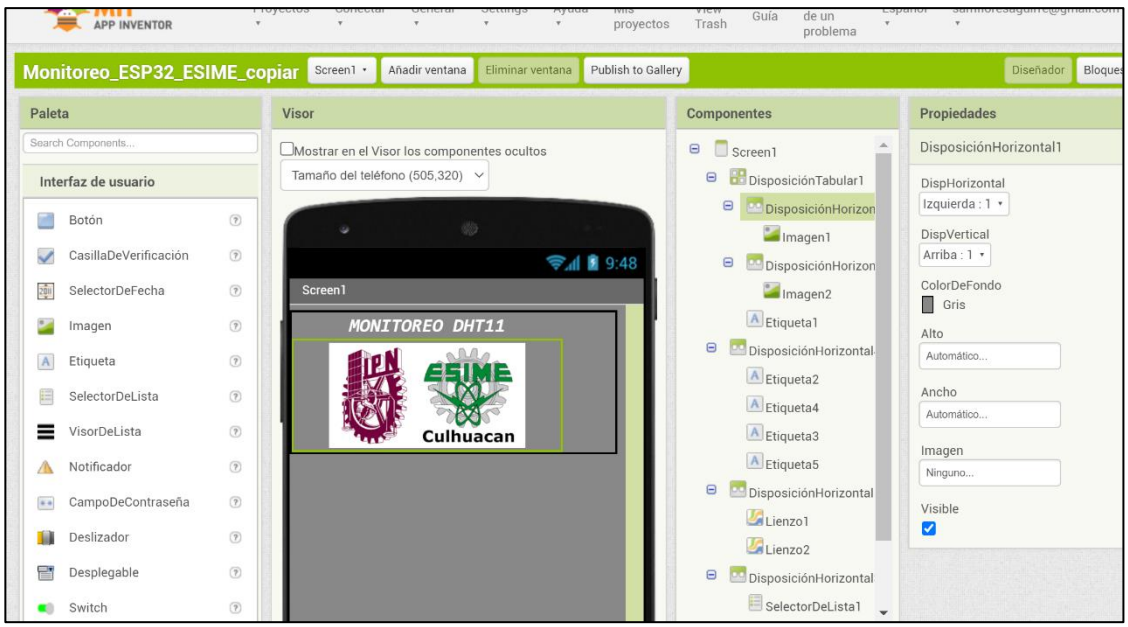


Figura 16

Elaboró:
Samuel Moisés Flores Aguirre

Se inserta otra disposición horizontal para agregar una segunda imagen. Ver Figura 17 y 18.

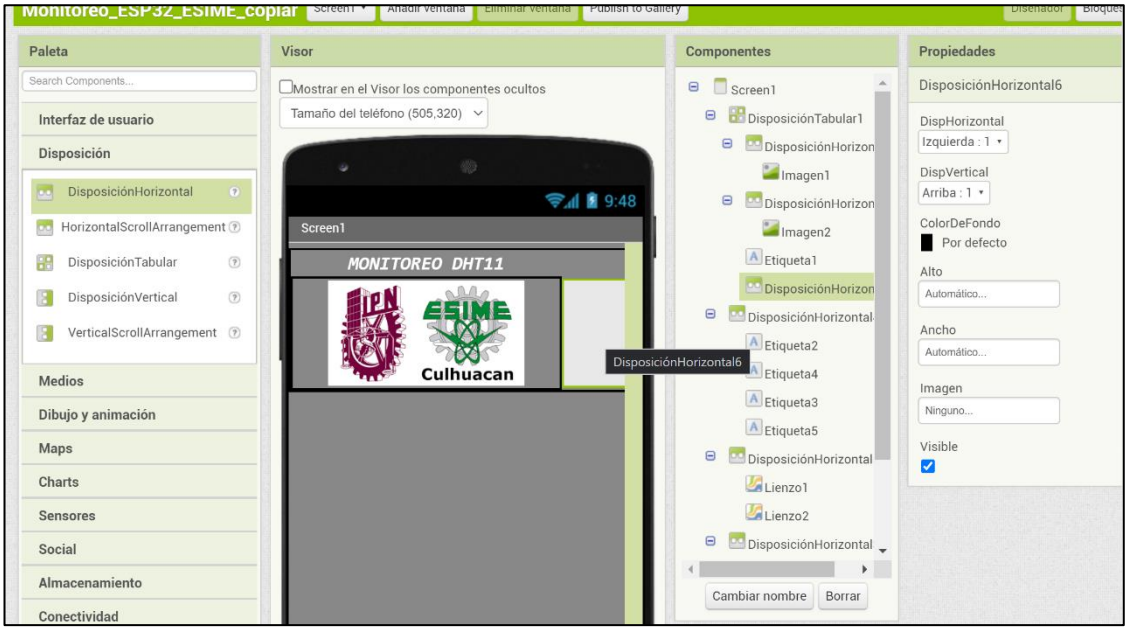


Figura 17

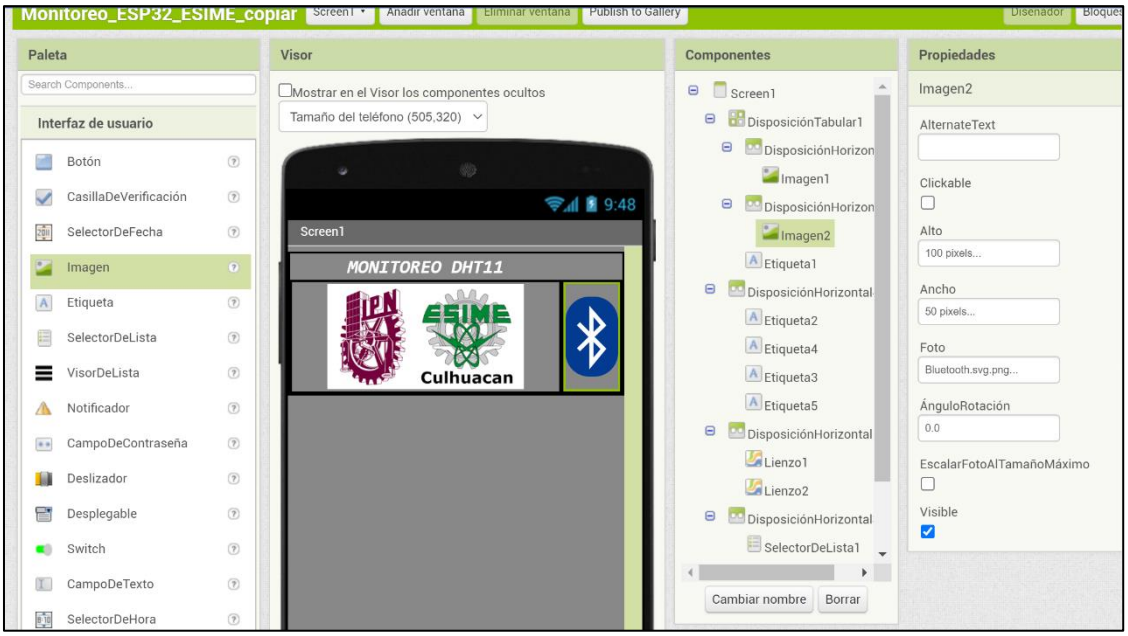


Figura 18

Elaboró:
Samuel Moisés Flores Aguirre

Se inserta otra disposición para colocar etiquetas donde se muestren las variables que se estén monitoreando. Ver Figura 19.

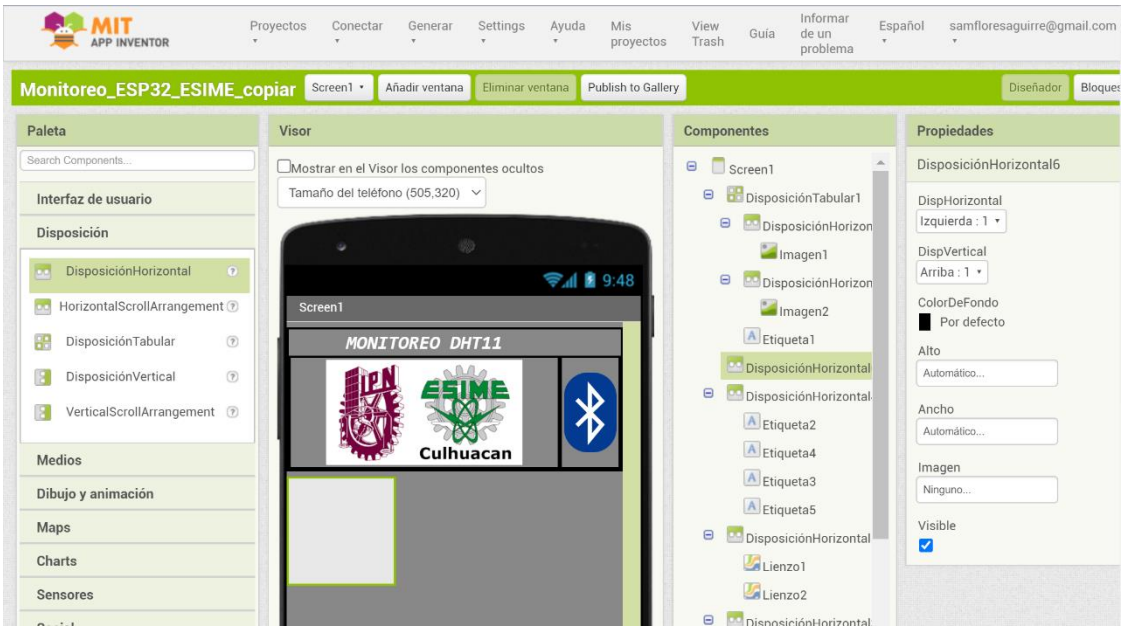


Figura 19

Se agregan cuatro etiquetas o labels, dos que muestren en formato texto el nombre de las dos variables y otras dos donde se muestren los valores sensados. Ver Figura 20.

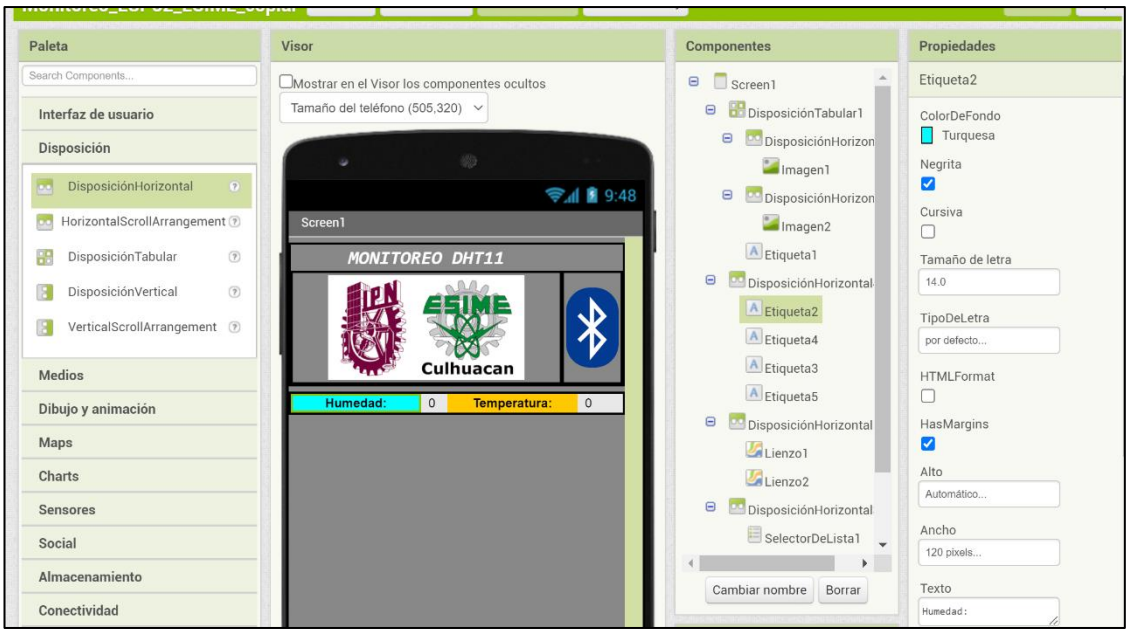


Figura 20

Elaboró:
Samuel Moisés Flores Aguirre

Debajo de las cuatro etiquetas, se inserta una nueva disposición para los canvas o lienzos. Ver Figura 21.

Figura 21

Se inserta dos lienzos dentro de la disposición, se encuentran en la sección de dibujo y animación. Los lienzos permitirán dibujar de manera gráfica los resultados que se vayan obteniendo del sensor. Ver Figura 22.

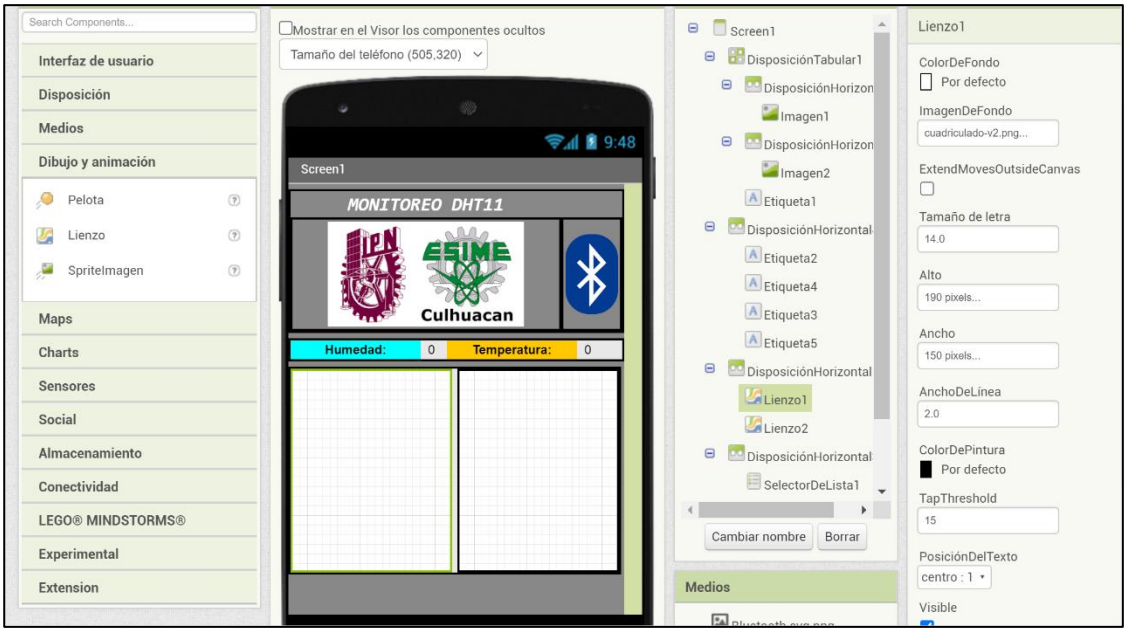


Figura 22

Elaboró:
Samuel Moisés Flores Aguirre

Se agrega una nueva disposición para agregar dos botones, uno para conectar y otra para desconectar. El botón para conectar sirve como selector de lista mientras que el otro componente es un botón común. Ver Figura 23 y 23.

Figura 23

Figura 24

Elaboró:
Samuel Moisés Flores Aguirre

Se agregan los componentes de Reloj y Bluetooth y por default se agregan de forma no visible. Ver Figura 25

Figura 25

Código a bloques en App Inventor

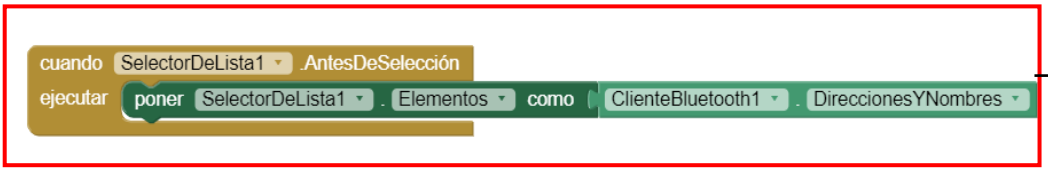
Después de ya tener el diseño realizado, se procede a realizar el programa en bloques, para ello se selecciona en la esquina superior derecha, en la sección “bloques”. Ver Figura 26.

Figura 26

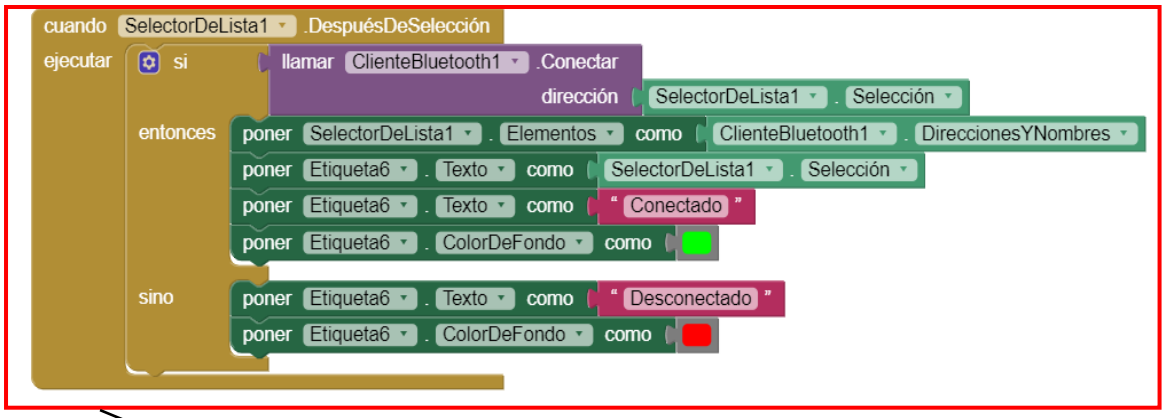
Elaboró:
Samuel Moisés Flores Aguirre

El programa propuesto es el siguiente. Se explicará algunos bloques de manera general:

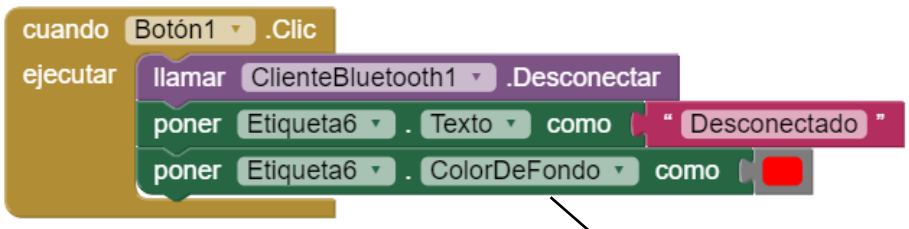
Se crean y se inicializan las variables para los ejes x y para los lienzos. Los lienzos se pueden caracterizar como un plano, en el cual se dibujará las variaciones que tengan los resultados. Como son dos lienzos, y se necesitan de una coordenada origen y una final, se tendrán cuatros variables para cada lienzo.



Antes de realizar la selección, se genera una lista con el nombre de los dispositivos y permite seleccionar alguno que este disponible.



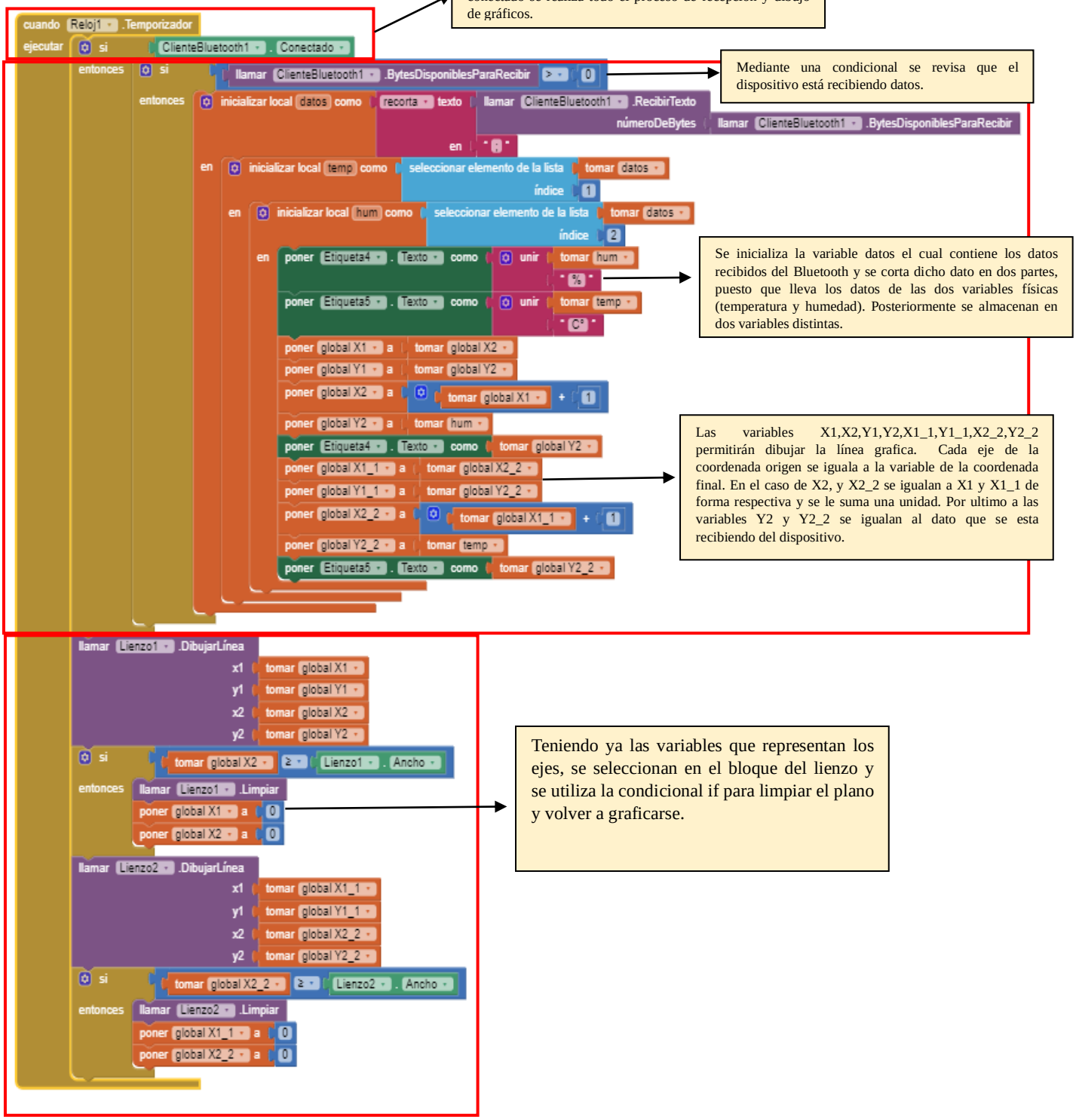
Cuando se presiona el botón de conectar, este genera una lista con los dispositivos disponibles por Bluetooth y se selecciona uno dependiendo el usuario. Después de seleccionar, la etiqueta 6 representará el estado de conexión con el lema “Conectado” y pintado de color verde, en caso contrario, dirá “Desconectado” en color rojo.



Al seleccionar el botón de “Desconectar” la etiqueta 6 dirá el estado como “Desconectado” en color rojo y se desconectará en automático el bluetooth.

Elaboró:
Samuel Moisés Flores Aguirre

El bloque de Reloj permitirá la sincronización del envío de los datos. Para ello si el dispositivo Bluetooth aparece conectado se realiza todo el proceso de recepción y dibujo de gráficos.



Mediante una condicional se revisa que el dispositivo está recibiendo datos.

Se inicializa la variable datos el cual contiene los datos recibidos del Bluetooth y se corta dicho dato en dos partes, puesto que lleva los datos de las dos variables físicas (temperatura y humedad). Posteriormente se almacenan en dos variables distintas.

Las variables X1,X2,Y1,Y2,X1_1,Y1_1,X2_2,Y2_2 permitirán dibujar la línea grafica. Cada eje de la coordenada origen se iguala a la variable de la coordenada final. En el caso de X2, y X2_2 se igualan a X1 y X1_1 de forma respectiva y se le suma una unidad. Por ultimo a las variables Y2 y Y2_2 se igualan al dato que se esta recibiendo del dispositivo.

Teniendo ya las variables que representan los ejes, se seleccionan en el bloque del lienzo y se utiliza la condicional if para limpiar el plano y volver a graficarse.

Elaboró:
Samuel Moisés Flores Aguirre

Código en Arduino

Para este caso, se tomo la misma plantilla antes explicada y se utilizó la biblioteca de DHT11 para recuperar y leer los valores de las variables de Temperatura y Humedad.

```
#include <DHT11.h>
#include <BluetoothSerial.h>

int pin=25;
DHT11 dht11(pin);

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

void setup()
   {
       Serial.begin(9600);
       SerialBT.begin("BluetoothESP32");
   }

void loop()
   {
       int err;
       float temp, hum;
       if((err = dht11.read(hum, temp)) == 0)
          {
             Serial.print("Temperatura: ");
             Serial.print(temp);
             Serial.print(" Humedad: ");
             Serial.print(hum);
             Serial.println();

             if (isnan(temp) || isnan(hum)) {
                return;
             }

              SerialBT.print(temp);
              SerialBT.print(";");
              SerialBT.print(hum);
              SerialBT.println(";");
          }
       else
          {
             Serial.println();
             Serial.print("Error Num :");
             Serial.print(err);
             Serial.println();
          }
       delay(1000);
   }
```

Elaboró:
Samuel Moisés Flores Aguirre

Prueba de la aplicación en App Inventor

Ya teniendo el código creado en Arduino cargado en el ESP32, se realizan las pruebas correspondientes, para ello se debe instalar una app creada por App Inventor para realizar las pruebas o en su caso utilizar el simulador que viene en la plataforma, sin embargo, para este caso se utilizará la app de Google Play Store. Ver Figura 27.

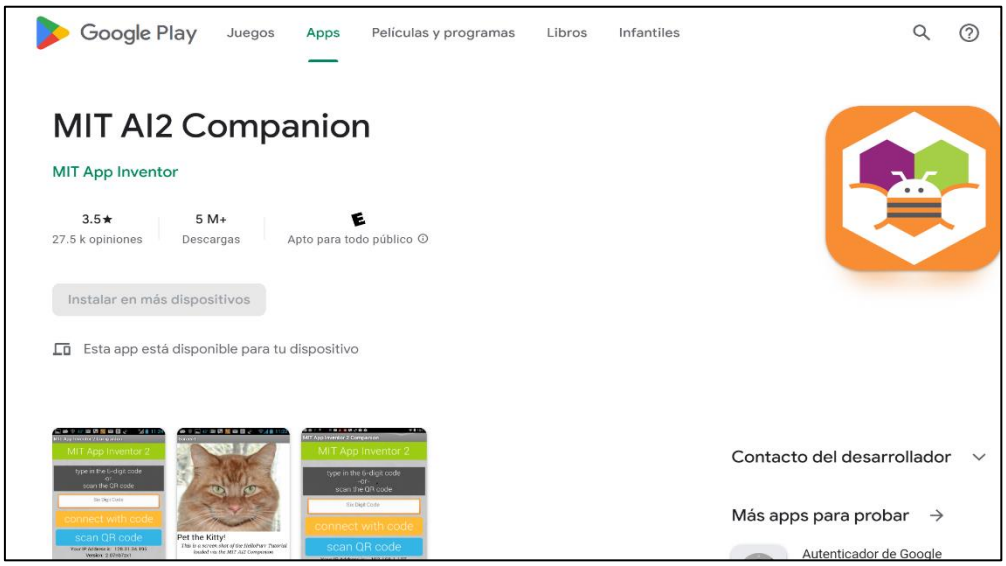


Figura 27

Una vez que se tiene descargada la app en el dispositivo, se conecta de App Inventor mediante un código. Ver Figura 28 y 29.

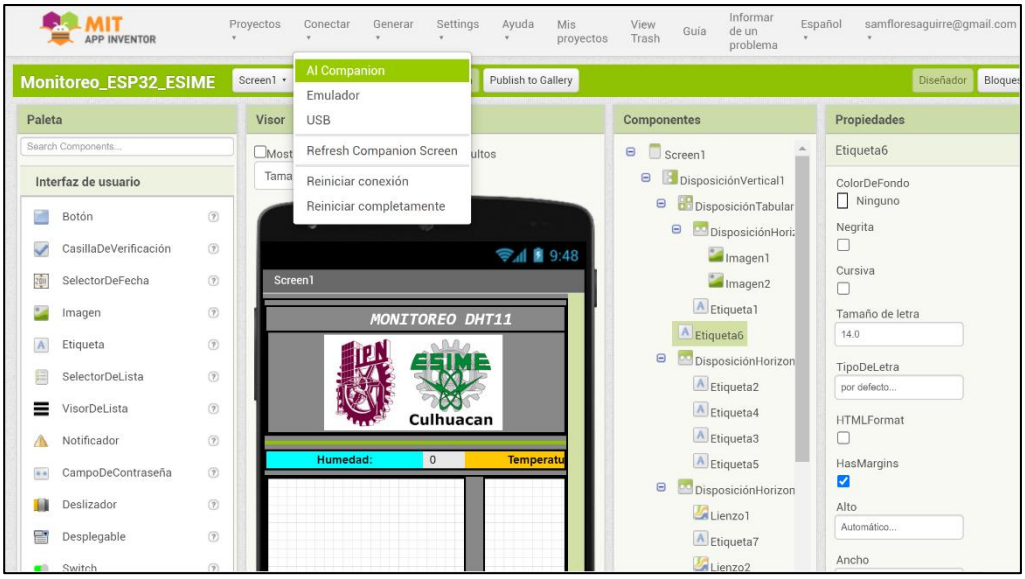


Figura 28

Elaboró:
Samuel Moisés Flores Aguirre

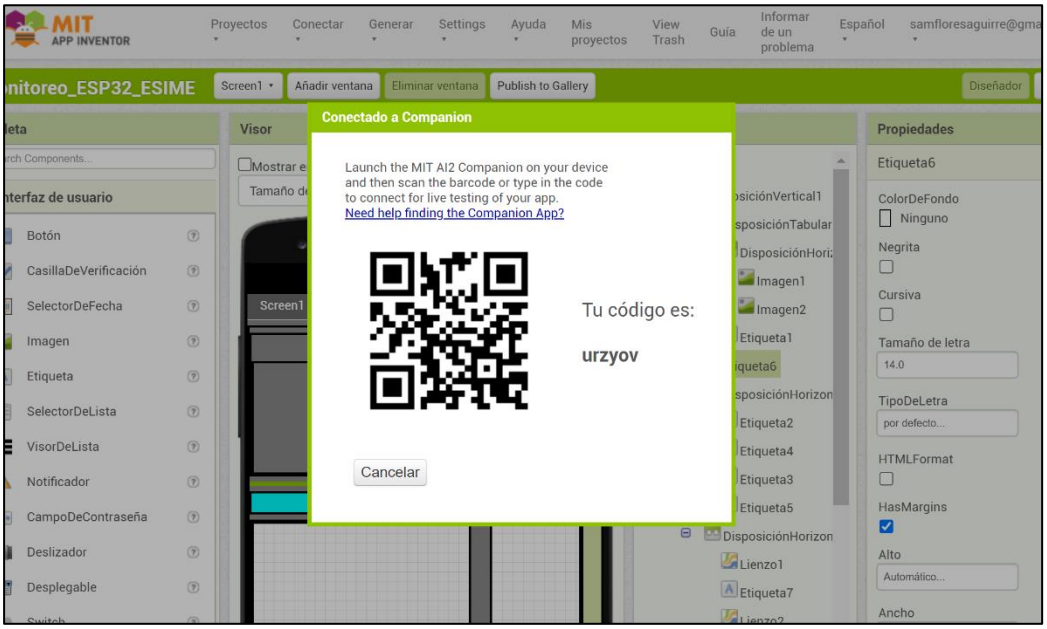


Figura 29

Ya con la aplicación conectada con la plataforma App Inventor, la aplicación creada aparecerá en el dispositivo móvil. La interfaz será idéntica a la que se vio en las imágenes anteriores, y para probarla solo basta con presionar el botón “Conectar” y se visualizará algo similar a lo de la Figura 30. El dispositivo por seleccionar es “BluetoothESP32”, es el nombre que anteriormente le dimos en el código de Arduino.

Figura 30

Elaboró:
Samuel Moisés Flores Aguirre

Una vez que se conecto con el dispositivo, y teniendo el ESP32 conectado al sensor DHT11 y a la computadora o fuente para alimentar los 5V, se podrá visualizar las variables sensadas. Ver Figura 31.

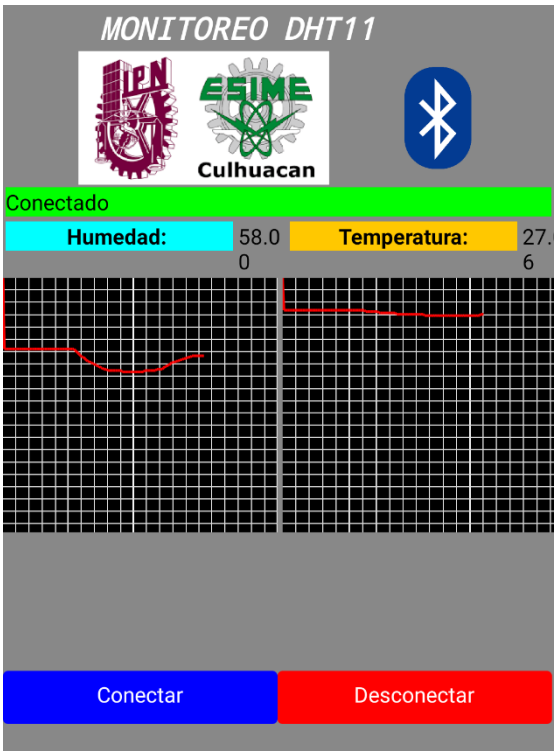


Figura 32

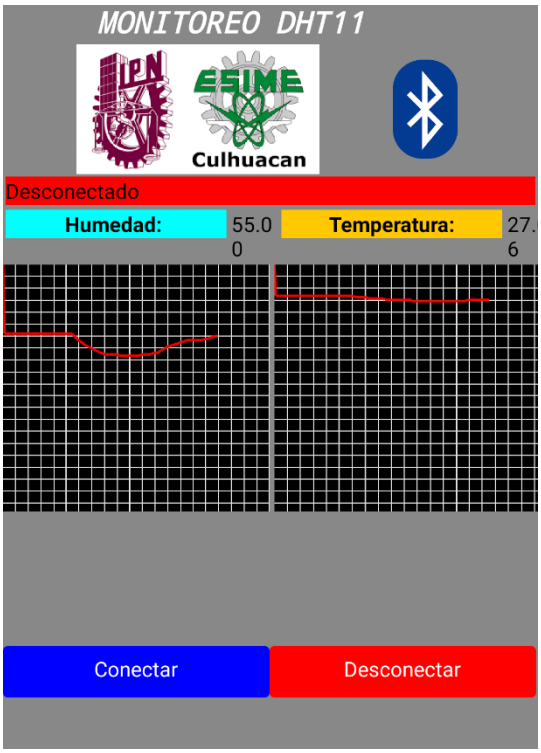


Figura 33

En la Figura 32 se puede visualizar a la aplicación conectada, mientras que en la Figura 33 se encuentra en modo “Desconectado”.

Nota: Dependiendo el tamaño de pantalla del dispositivo se tiene que tomar a consideración para el desarrollo de la app, también puede considerar desarrollar una app de tipo responsive.

Conclusión

El ESP32 es una tarjeta de desarrollo muy versátil que permite una amplia gama de aplicaciones, desde proyectos de IoT hasta robótica y automatización industrial.

En este caso se realizó una demostración de forma genérica de como llevar datos de un sensor a través de un microcontrolador mediante la comunicación Bluetooth utilizando como interfaz grafica una aplicación móvil. Por flexibilidad y facilidad del estudiante se realizo desde una plataforma publica (App Inventor), el cual permite que mediante su programación en bloques sea más accesible su desarrollo y diseño.

Repositorio en Github

Para facilidad del estudiante, puede revisar el siguiente repositorio con los códigos utilizados en la presente guía: https://github.com/samfag/Comunicacion_Inalambrica_ESP32.git

Elaboró:
Samuel Moisés Flores Aguirre