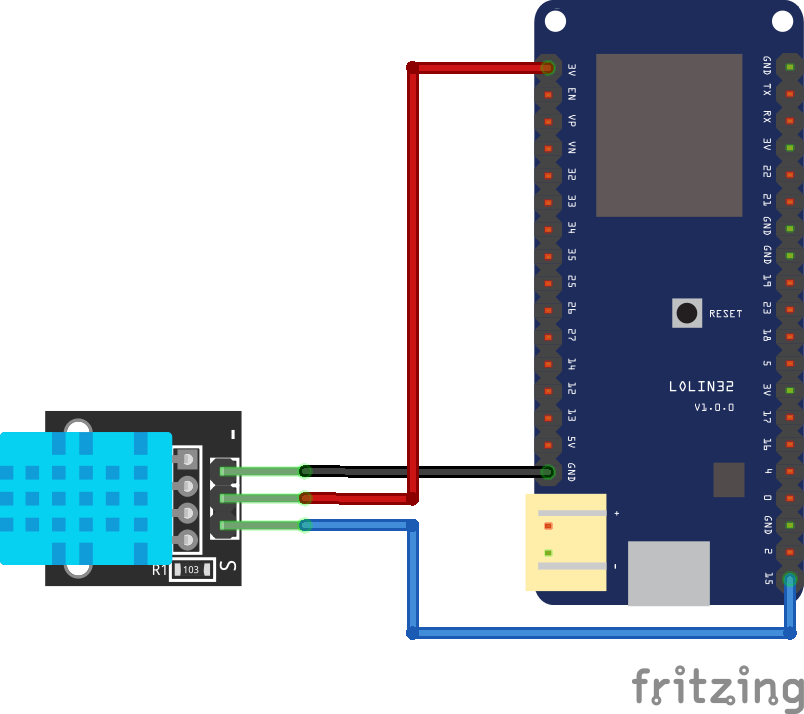
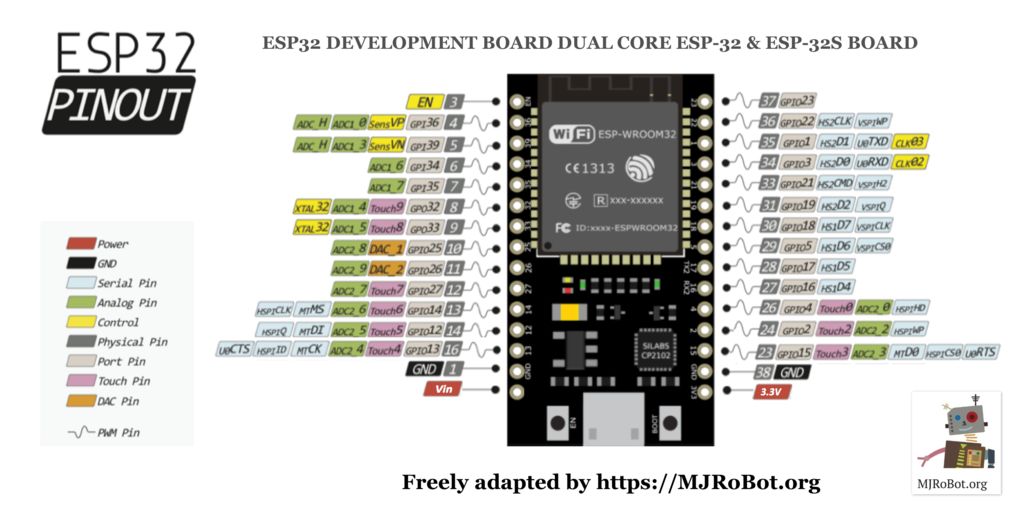
Reading the value of DHT11 Sensor via ESP32 Board using BLE Scanner(app)

Requirements:

* ESP32 Board
* DHT11 Sensor
* Jumper Wires
* Install Arduino IDE

Step 1: Connections ===>

|  |  |
| --- | --- |
| **ESP 32 Pin number** | **DHT11 sensor** |
| GND | -ve |
| 3v | +ve |
| D13 | Data |



**Step 2**: In Arduino IDE :

Select Board – DOIT ESP32 DEVKIT

**Step 3**: Download the DHT11 library and include it

Link: <https://codeload.github.com/adafruit/DHT-sensor-library/zip/master>

**Step 4**: Write program as follows:

/\*

Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-snippets/blob/master/cpp\_utils/tests/BLE%20Tests/SampleServer.cpp

Ported to Arduino ESP32 by Evandro Copercini

\*/

#include <BLEDevice.h>

#include <BLEUtils.h>

#include <BLEServer.h>

/\*

Download "DHT.h" library from https://codeload.github.com/adafruit/DHT-sensor-library/zip/master

\*/

#include "DHT.h"

#include <stdlib.h>

#include <string.h>

// Define the type of DHT you have i.e. DHT11, DHT22.

#define DHTTYPE DHT11

//Set pin for DATA in ESP32 (For more details check PinOut)

const int DHTPin = 13;

// Initialize DHT sensor.

DHT dht(DHTPin, DHTTYPE);

//Global Variables to get/read data from/to mobile.

char inputBuffer[10];

char tx[10];

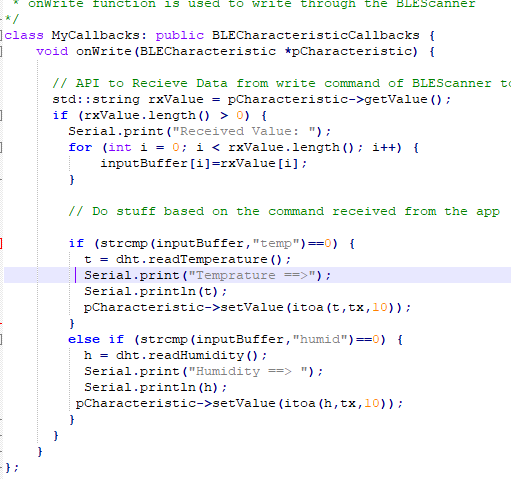
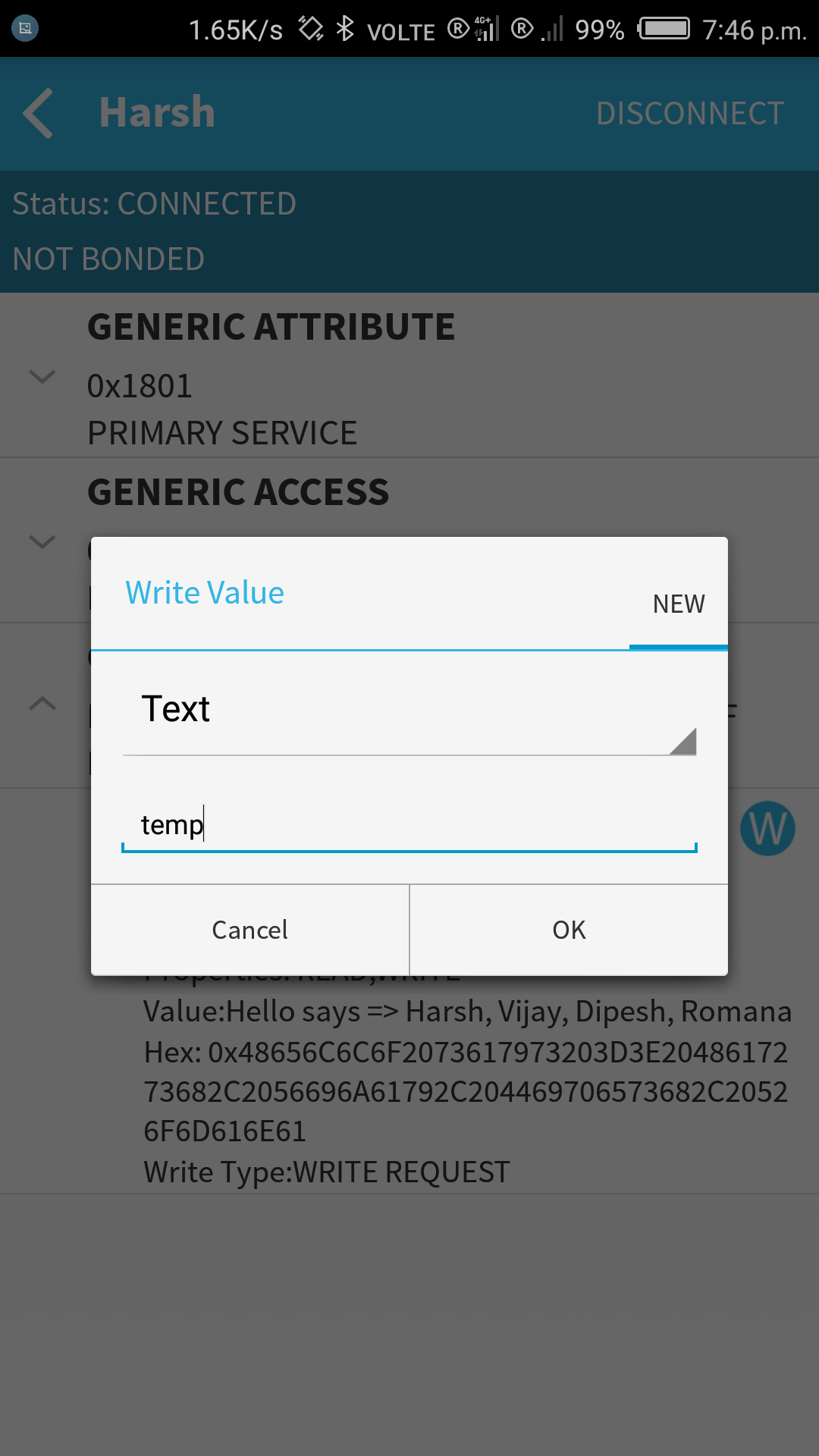
int h, t;

**Step 5**: Generate UUIDs from <https://www.uuidgenerator.net/> and define them as follows

#define SERVICE\_UUID "df620f6f-8907-47b1-aa61-1e81d2db5f6f"

#define CHARACTERISTIC\_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

**Step 6:** Define the class MyCallback to create the services and in class write a “onWrite” funtion which is going to call when write command comes from BLEScanner as shown below:



**Step 7:** SetCallbacks is an API used to create class object MyCallbacks()

void setup() {

Serial.begin(115200);

Serial.println("Starting BLE work!");

dht.begin();

BLEDevice::init("Harsh");

BLEServer \*pServer = BLEDevice::createServer();

BLEService \*pService = pServer->createService(SERVICE\_UUID);

BLECharacteristic \*pCharacteristic = pService->createCharacteristic(

CHARACTERISTIC\_UUID,

BLECharacteristic::PROPERTY\_READ |

BLECharacteristic::PROPERTY\_WRITE

);

// setCallbacks is an API used to create class object MyCallbacks()

pCharacteristic->setCallbacks(new MyCallbacks());

pCharacteristic->setValue("Hello says => Harsh, Vijay, Dipesh, Romana");

pService->start();

BLEAdvertising \*pAdvertising = pServer->getAdvertising();

pAdvertising->start();

Serial.println("Characteristic defined! Now you can read it in your phone!");

}

void loop() {

// put your main code here, to run repeatedly:

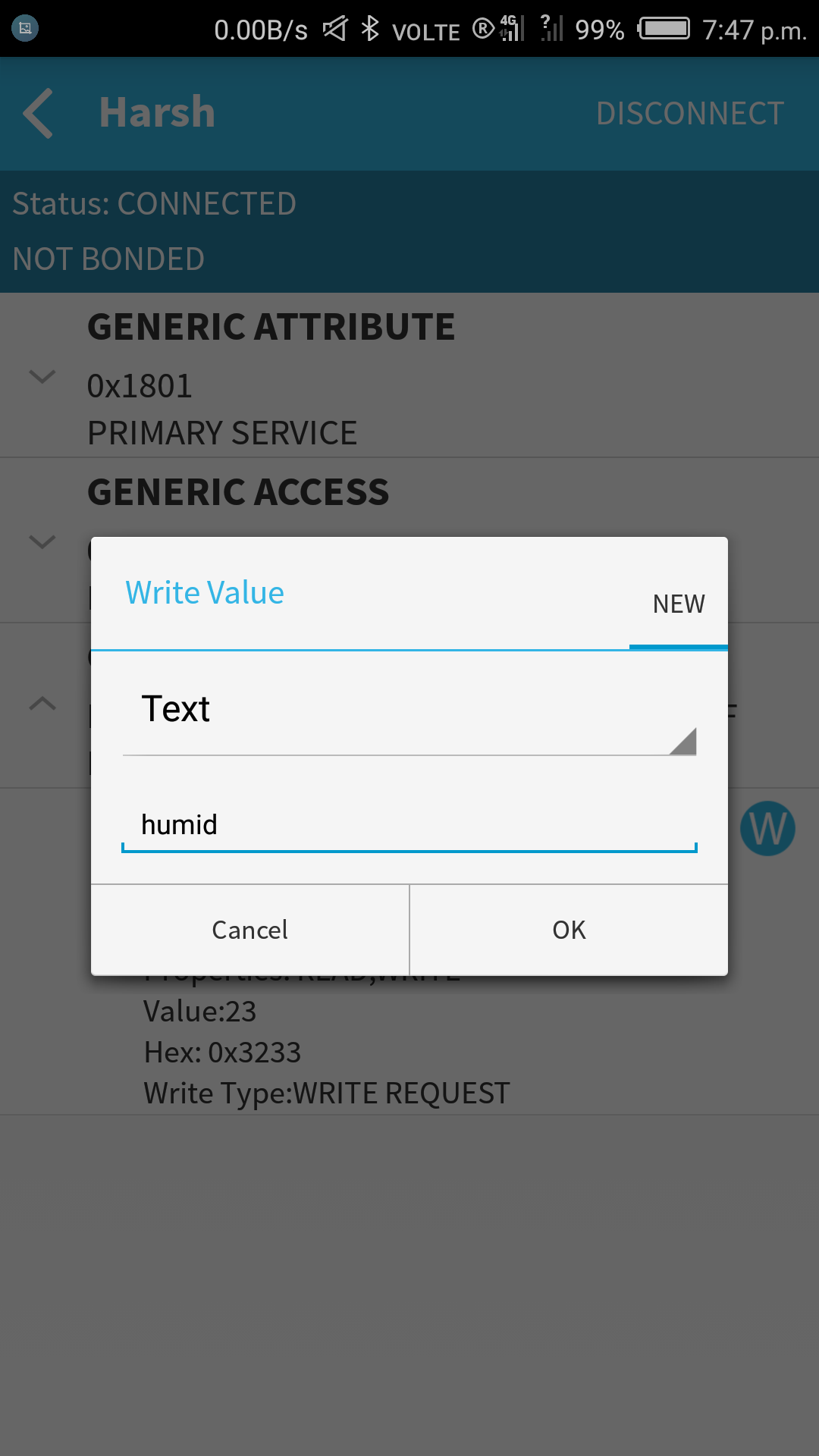
delay(2000);

}

**Step 8**: Now upload the code and open Serial Monitor.

**Step 9**:Now open BLE Scanner in phone and connect the bluetooth .

**Step 10**: In Custom Characteristics when you write “humid” following API is going to call.



else if (strcmp(inputBuffer,"humid")==0) {

h = dht.readHumidity();

Serial.print("Humidity ==> ");

Serial.println(h);

pCharacteristic->setValue(itoa(h,tx,10));

}

**Step 11**: Same as for “temp” , following API is going to call when temp command is send via BLE

if (strcmp(inputBuffer,"temp")==0) {

t = dht.readTemperature();

Serial.print("Temprature ==>");

Serial.println(t);

pCharacteristic->setValue(itoa(t,tx,10));

}

**Result**: Hence communication via BLE is successfully setup and you can see the ouput in Serial monitor as well as on your Phone too.

**THANK YOU**