Home automation using DHT11 interfacing with ESP32 & Blink App

Abstract:

In this project, a design is proposed by using ESP32 and Temperature and Humidity (DTH11) sensor for Detecting the temperature and humidity, automation of lights and fans using Arduino with Internet of Things for smart homes. Nowadays we're having automation of every little electrical device in our homes. Internet of Things is the concept of basically connecting any device with an on and off switch to the internet. IOT is more than smart homes and connected appliances; however, it scales up to include smart cities with connected sensors. Using Temperature sensors, the lights will automatically turn on and off according to the intensity of light. Temperature sensors will detect the room temperature and turn on and off fans and also that data is uploaded to the cloud to operate in anywhere by using the Blink app it is a cloud.

Keywords: Arduino, Ethernet, Home Automation,
Internet of Things, WiFi Router

Introduction:

These days, the web need ended up a normal interface that a significant number of gadgets use in place will improve the everyday life about numerous individuals. Web aides us to get quick result for huge number of issues capable to interface starting with any of the remote spots which contributes with general expense decrease furthermore vitality utilization. Home mechanization might make portrayed as introduction for innovation organization in the home environment which provides straight forwardness which is more secure with its occupants. Towards utilizing the innovation of web for Things, those examinations .Furthermore execution about home mechanization have got extra Normal. Different remote advances which has the capacity to help some sort of remote information transfer, sensing also management like Bluetooth, Wi-Fi and other cell division networks would be used to enter abundant levels for discernment inside the the human body presence, those lights and fans will turn on/off. Those temperature sensor need aid associated with the Arduino Uno. By utilizing a Blink app it will stores the data and also operates the fan and light by using temperature and humidity.
The suggested framework may be a greater amount efficient, settled also expense compelling. That Arduino customer in the app sends an blink appeal on arduino that runs the blink app like a cloud. The thought will be that those smartphones sends an blink of the arduino. Those temperature sensor will be utilized to detect the temperature and humidity by that temperature and humidity the automatically lights and fans will ON and OFF.

Hardware:

In this project we used the hardware and software components as describe in the below.
The hardware components are
1. Temperature & Humidity sensor (DHT11)
2. Four pin relay module
3. ESP32 module

Temperature and Humidity Sensor (DHT11):-

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability. This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance.

DHT11 Temperature and Humidity Sensor

Specification:

- Supply Voltage: +5 V
- Temperature range :0-50 °C error of ± 2 °C
- Humidity :20-90% RH ± 5% RH error
- Interface: Digital

Four pin relay module:

A relay is an electromagnetic switch operated by a relatively small electric current that can turn on or off a much larger electric current. The heart of a relay is an electromagnet (a coil of wire that becomes a temporary magnet when electricity flows through it). You can think of a relay as a kind of electric lever: switch it on with a tiny current and it switches on ("leverages") another appliance using a much bigger current. Why is that useful? As the name suggests, many sensors are incredibly sensitive pieces of electronic equipment and produce only small electric currents. But often we need them to drive bigger pieces of apparatus that use bigger currents. Relays bridge the gap, making it possible for small currents to activate larger ones. That means relays can work either as switches (turning things on and off) or as amplifiers (converting small currents into larger ones).

Relay module

ESP32Module:

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations and includes in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.[2] It is a successor to the ESP8266 microcontroller.

ESP32Module

Features:

- Processors:
  - CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
  - Ultra low power (ULP) co-processor
- Memory: 520 KiB SRAM
- Wireless connectivity:
  - Wi-Fi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR and BLE
- Peripheral interfaces:
  - 12-bit SAR ADC up to 18 channels
  - 2 × 8-bit DACs
  - 10 × touch sensors (capacitive sensing GPIOs)
  - 4 × SPI
  - 2 × I²S interfaces
  - 2 × I²C interfaces
  - 3 × UART
  - SDIO/SPI slave controller
  - Ethernet MAC interface with dedicated DMA and IEEE 1588 Precision Time Protocol support
  - CAN bus 2.0
  - Infrared remote controller (TX/RX, up to 8 channels)
  - Motor PWM
  - LED PWM (up to 16 channels)
  - Hall effect sensor
  - Ultra low power analog pre-amplifier
- Security:
  - IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
  - Secure boot

- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographichardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

- Power management:
  - Internal low-dropout regulator
  - Individual power domain for RTC
  - 5µA deep sleep current
  - Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

Specifications:

Parameter	Symbol	Min	Max	Unit
Input low voltage	V_{IL}	-0.3	$0.25\times V_{IO}$	V
Input high voltage	V_{IH}	$0.75\times V_{IO}$	3.3	V
Input leakage current	I_{IL}	-	50	nA
Output low voltage	V_{OL}	-	$0.1\times V_{IO}$	V
Output high voltage	V_{OH}	$0.8\times V_{IO}$	-	V
Input pin capacitance	C_{pad}		2	pF
VDDIO	V_{IO}	1.8	3.3	V
Maximum drive capability	I_{MAX}		12	mA
Storage temperature range	T_{STR}	-40	150	°C

Software:

Arduino IDE:

The **Arduino integrated development environment (IDE)** is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.[5] The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

Blynk:

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, vizualize it and do many other cool things.
There are three major components in the platform:

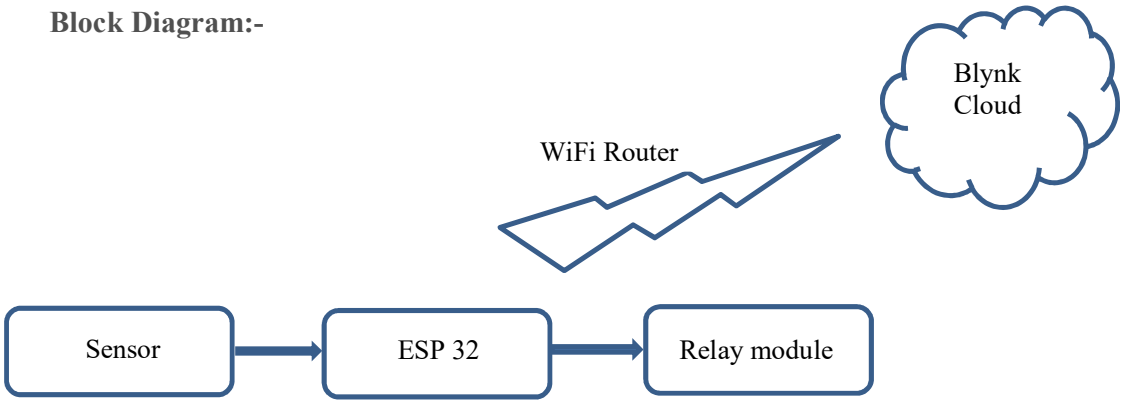
- **Blynk App** - allows to you create amazing interfaces for your projects using various widgets we provide.
- **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Code:

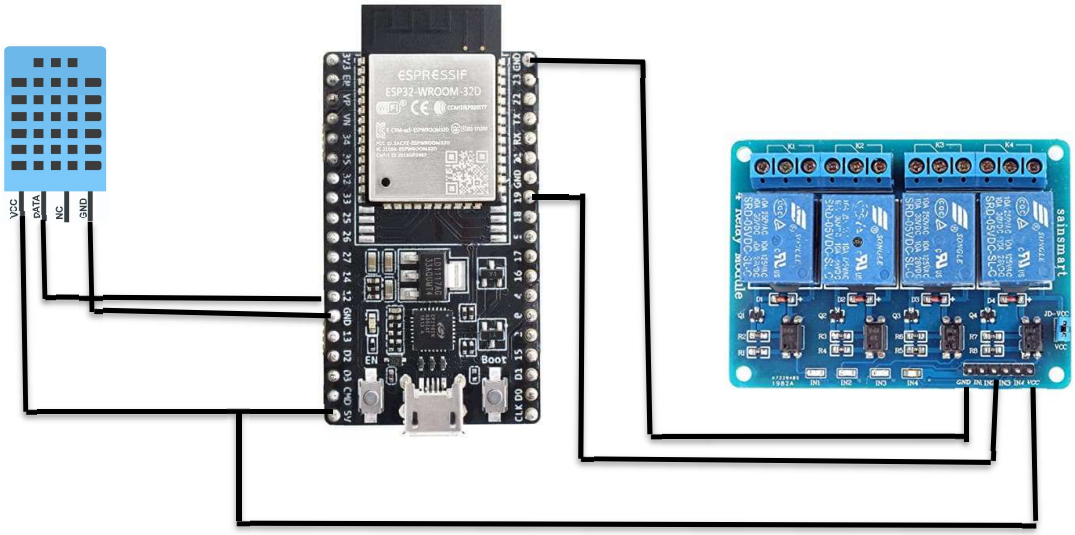
```
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <ThingSpeak.h>
#include "DHT.h"
#define DHTTYPE DHT11
#define Fan 19
#define DHTPIN 12
DHT dht(DHTPIN, DHTTYPE);

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "01179de65dd4493c8f9df4cf307cf160";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "$URYA";
char pass[] = "$urya1242";
void setup()
{
// Debug console
Serial.begin(115200);
pinMode(Fan,OUTPUT);
digitalWrite(Fan,HIGH);
dht.begin();
Blynk.begin(auth, ssid, pass);
// You can also specify server:
//Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);
//Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);
}

void loop()
{
Blynk.run();
float temp=dht.readTemperature();
```

```
float humi = dht.readHumidity();
Blynk.virtualWrite(V4, temp);
Blynk.virtualWrite(V5, humi);
Blynk.virtualWrite(V6, temp);
Blynk.virtualWrite(V7, humi);

if (isnan(humi) || isnan(temp)) {
Serial.println("Failed to read from DHT sensor!");
delay(500);}

if(temp > 25)
{
Serial.print("Temperature: ");
Serial.print(temp);
Serial.print(" *C ");
Serial.println("Temp is High");
digitalWrite(Fan, LOW);
delay(1000);
}
else
{
Serial.print("Temperature: ");
Serial.print(temp);
Serial.print(" *C ");
Serial.println("Temp is Low");
digitalWrite(Fan, HIGH);
// Blynk.virtualWrite(Fan, LOW);
delay(500);
}
// Serial.print("\t");
Serial.println("Humidity: ");
Serial.print(humi);
Serial.print(" %");
Serial.println("");
delay(5000);
}
```

Block Diagram:-



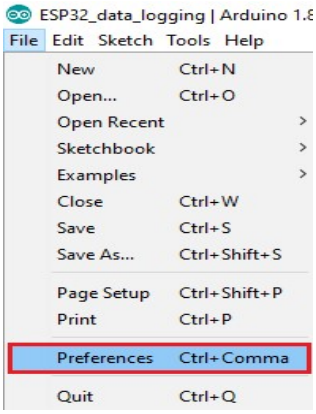
Pin connection:-



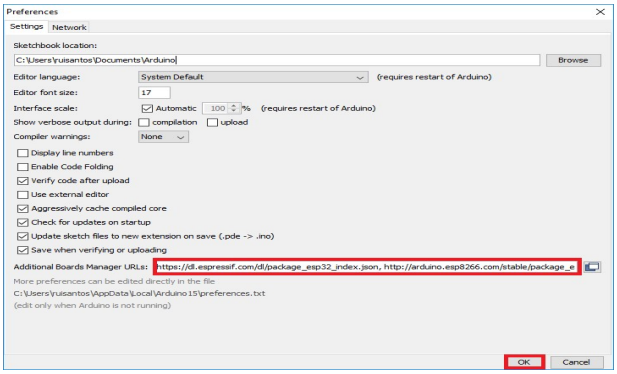
Work Flow:-

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File**> **Preferences**



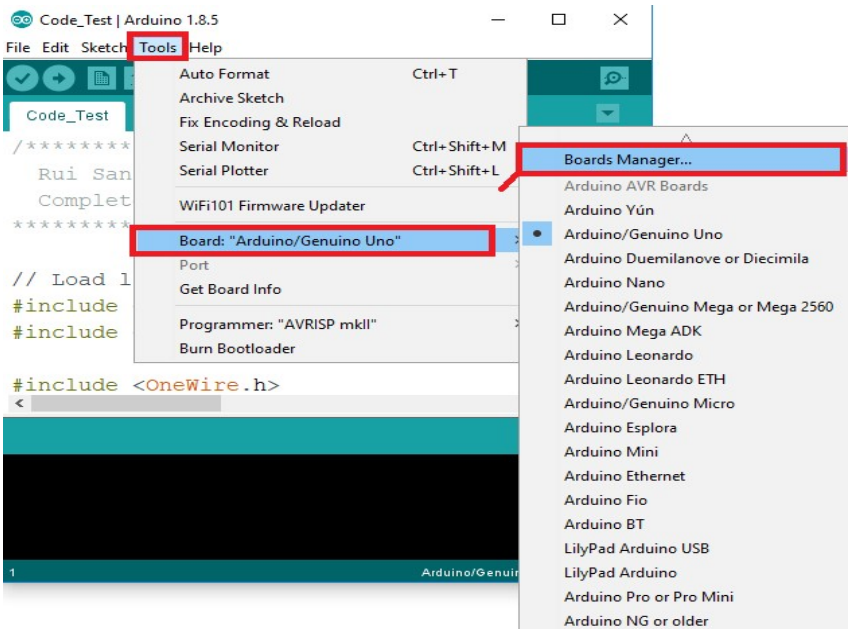
2. Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the "Additional Board Manager URLs" field as shown in the figure below. Then, click the "OK" button:



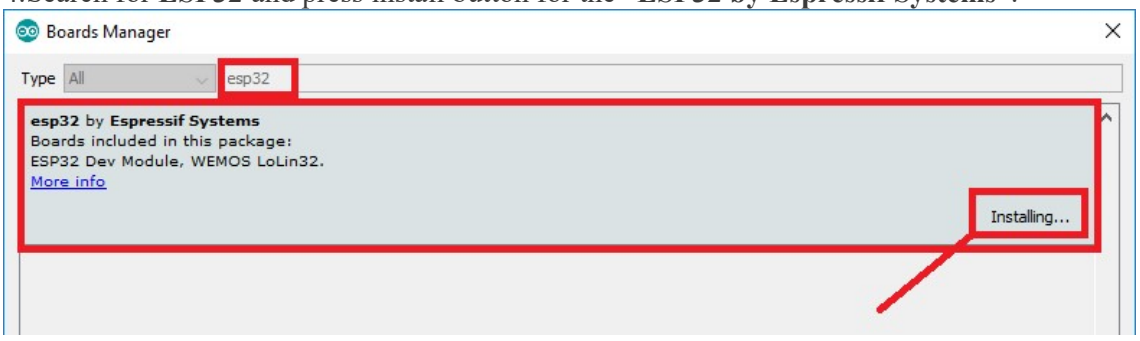
Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:

https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json

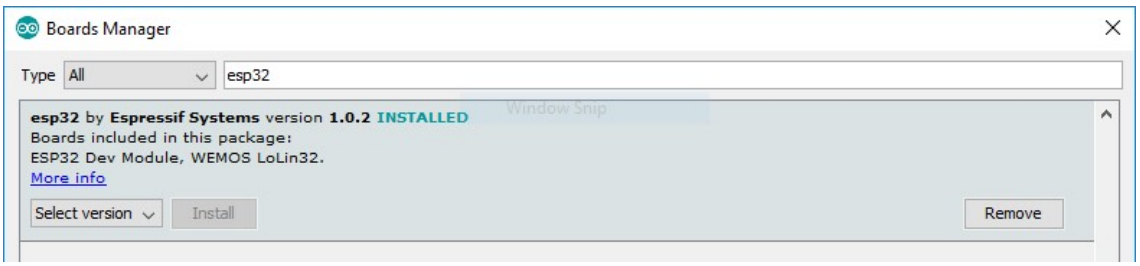
3.Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



4.Search for **ESP32** and press install button for the "**ESP32 by Espressif Systems**":



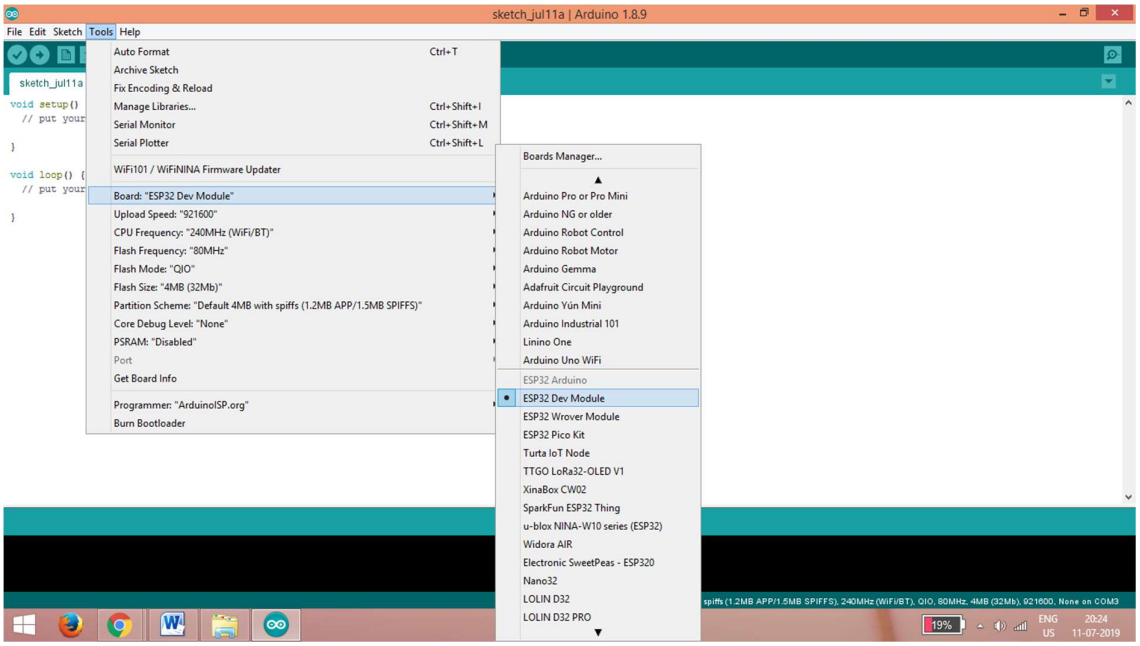
5.That's it. It should be installed after a few seconds.



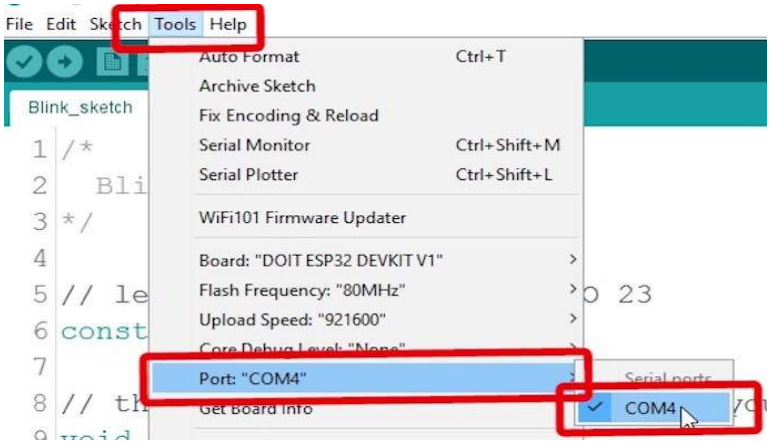
Testing the Installation

Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps:

1. Select your Board in **Tools** > **Board** menu (in my case it's the **DOIT ESP32 DEVKIT V1**)

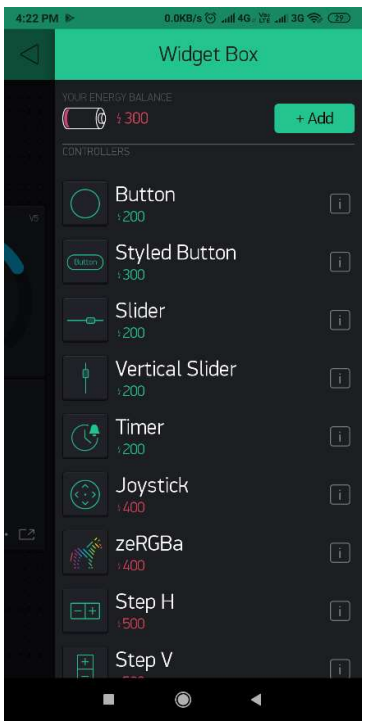


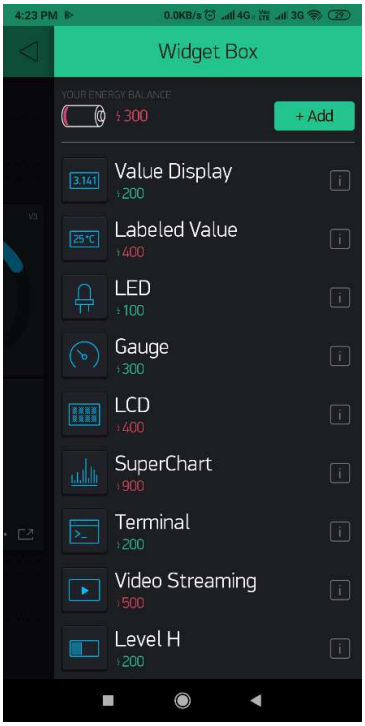
2. Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the CP210x USB to UART Bridge VCP Drivers):

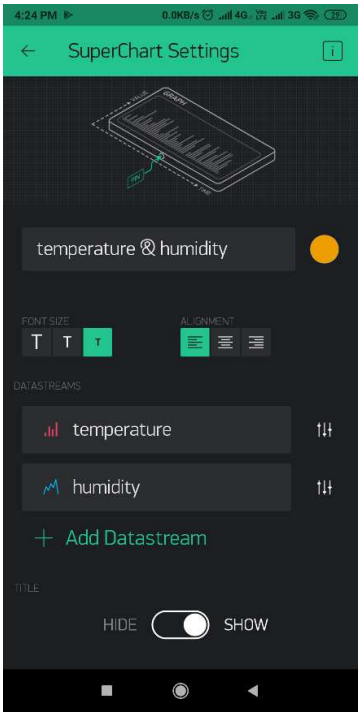


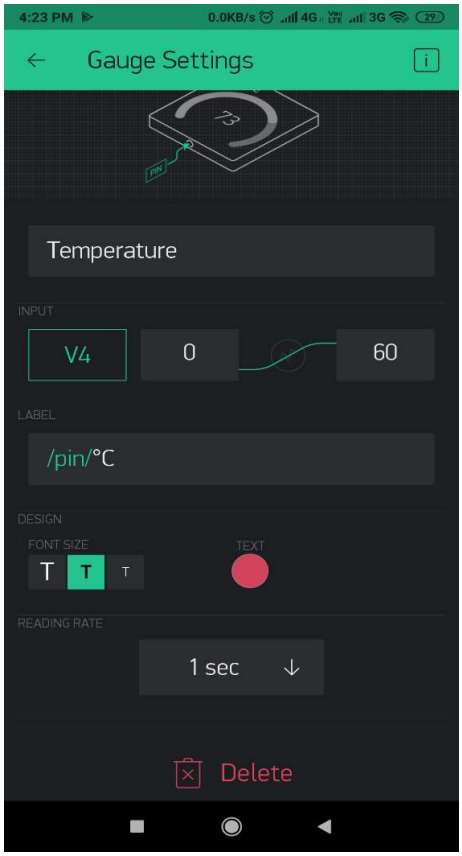
3. Take New file copy the code and compile it.
4. After successful compilation of code upload it.
5. Open Serial monitor There we can see our output values.

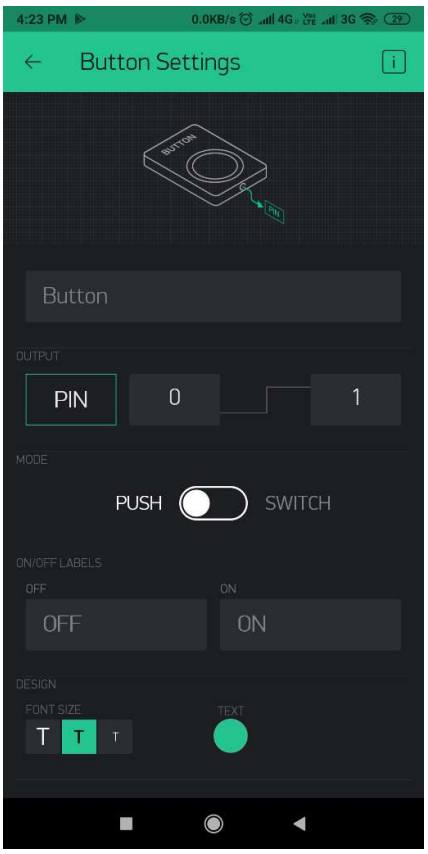
Configuring Blynk App:-











1. Install the Blynk app from Play store.
2. Login With Email Id and a terminal as shown in above fig 1 is displayed select required/suitable widget from the list as shown fig 3.
3. Open widget and configure it with required data fields.
4. Finally upload the code on Arduino and click play button on Blynk app then sensed data is displayed as shown below output.

Result Output:-

Conclusion: -

- Real time Temperature and Humidity values are uploaded to Blynk cloud and visualized in Blynk App.
- If Temperature is more than 25℃ Then fan will automatically turned ON and if it is less than 25℃ then fan will automatically turned OFF.
- We will have virtual control over home appliances via Blynk App. (We can turn On and OFF home appliances using Blynk App.)