Controlling devices remotely using Bluetooth link, WiFi link

~By Uday, Shweta

LinkedIn

https://www.linkedin.com/in/uday-yeshi-79240127a//

https://www.linkedin.com/in/shweta-shinde-018b8b27a//

Mail

uday.yeshi23@spit.ac.in

shweta.shinde23@spit.ac.in

By using this system, you can monitor and control devices remotely with ease. The real-time data is available on the Blynk cloud server, accessible from your PC, mobile phone, or web browser. This feature significantly enhances user experience with a better UI and advanced functionalities. The impact of this method is substantial, offering seamless device management and real-time monitoring, leading to improved efficiency and convenience in various applications.

Home Water Tank Management: Automatically control the water pump to fill household water tanks, preventing overflow and ensuring a constant water supply.

Agricultural Irrigation Systems: Monitor and manage water levels in irrigation systems, ensuring crops receive the right amount of water.

Industrial Water Storage: Maintain optimal water levels in industrial storage tanks, crucial for processes that depend on consistent water supply.

Flood Monitoring and Prevention: Use in flood-prone areas to monitor rising water levels and activate pumps or alarms to prevent flooding.

Aquarium Management: Ensure proper water levels in aquariums to maintain a healthy environment for aquatic life.

Water Treatment Plants: Monitor and control water levels in various stages of the treatment process to ensure efficient operation.

Swimming Pool Management: Automatically refill swimming pools to maintain desired water levels and prevent overflow.

Smart City Infrastructure: Integrate with smart city systems to monitor and manage water levels in public water bodies, reservoirs, and storage tanks.

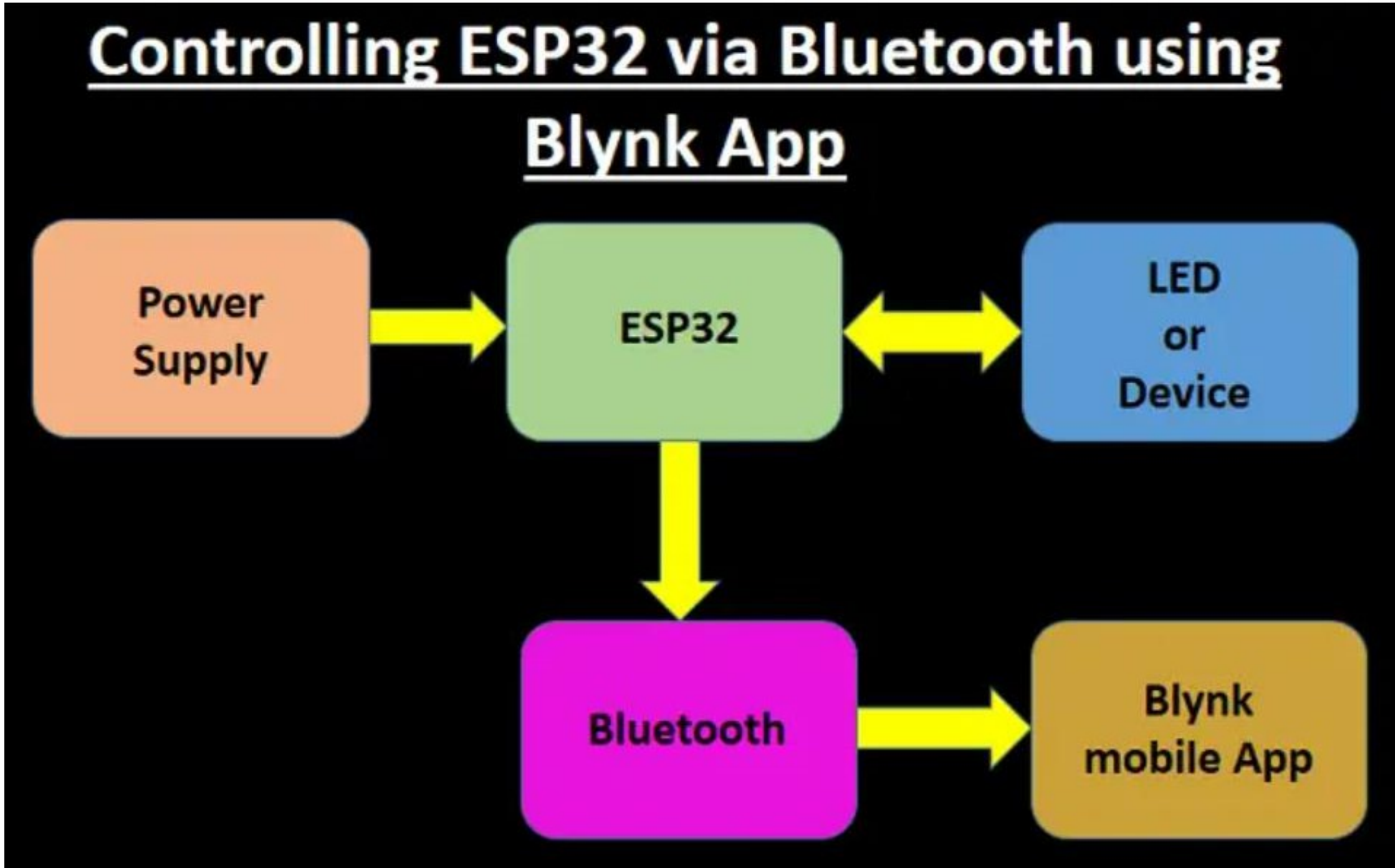
Rainwater Harvesting Systems: Efficiently manage water levels in rainwater harvesting tanks, ensuring optimal use of collected rainwater.

Remote Monitoring of Wells: Keep track of water levels in remote wells and control pumping operations to maintain water availability.

Firefighting Water Reservoirs: Ensure that water reservoirs for firefighting are always at optimal levels, ready for emergency use.

Greenhouse Management: Monitor and control water levels in tanks used for watering plants in greenhouses, maintaining ideal growing conditions.

- With Blynk Library you can connect **over 400 hardware models** (including ESP8266, ESP32, NodeMCU, all Arduinos, Raspberry Pi, Particle, Texas Instruments, etc.)to the Blynk Cloud.


Controlling ESP32 via Bluetooth using Blynk App
Power Supply
ESP32
LED or Device
Bluetooth
Blynk mobile App

Blynk Protocol

The Blynk Protocol is designed to facilitate communication between IoT (Internet of Things) devices, such as ESP8266, ESP32, Arduino, Raspberry Pi, and the Blynk mobile app. It enables real-time interaction and control of these devices through a graphical user interface (GUI) created in the Blynk app. Here's how the Blynk Protocol works:

1. **Device Initialization**:
   - You begin by programming your IoT device (e.g., ESP8266) using the Blynk library or framework compatible with your platform.
   - During initialization, you provide the device with a unique authentication token. This token is generated when you create a project in the Blynk app and is used to identify and authenticate your device with the Blynk cloud server.
2. **Connection to Blynk Cloud**:
   - The device establishes a connection to the Blynk Cloud server over the Internet using the Blynk Protocol. The cloud server serves as an intermediary between your device and the Blynk app.
3. **Blynk App Configuration**:
   - On your mobile device, you install the Blynk app and create a new project.
   - You design the GUI of your project by adding various widgets (buttons, sliders, displays, graphs, etc.) to the project interface. Each widget is associated with a virtual pin.
4. **Mapping Widgets to Virtual Pins:**
   - You link the widgets in the Blynk app to specific virtual pins (e.g., V0, V1, V2, etc.).
   - These virtual pins act as channels through which data is sent and received between the app and the IoT device.
5. **Bi-Directional Communication:**
   - The IoT device can send data to the Blynk Cloud server by writing data to specific virtual pins.
   - The Blynk app can send data to the Blynk Cloud server by interacting with the widgets on the app.
   - Data flows bi-directionally between the IoT device and the app via these virtual pins.

Blynk Protocol

6. **Interaction and Control**:
   - In the Blynk app, you can interact with the widgets to control your IoT device or monitor sensor data in real-time.
   - For example, you can turn on/off lights, adjust the position of a servo motor, or view temperature sensor readings.
7. **Cloud-Based Processing**:
   - The Blynk Cloud server processes incoming data and relays it to the appropriate virtual pins.
   - It maintains the connection with your IoT device and ensures that data is routed correctly.
8. **Device Response:**
   - The IoT device receives data sent to its virtual pins and responds accordingly.
   - For example, if you pressed a button widget in the Blynk app to turn on a light, the device would receive the command and activate the light.
9. **Real-Time Feedback**:
   - The app receives real-time feedback from the IoT device. For instance, if the IoT device is a temperature sensor, the app can display the latest temperature readings on a widget.
10. **Security and Authentication**:
    - The authentication token you provided during device initialization ensures that only authorized devices can connect to your specific Blynk project.
11. **End-to-End Communication**:
    - The Blynk Protocol ensures end-to-end communication between your IoT device and the Blynk app via the Blynk Cloud server.

blynk

Search on PyPI

<INSTALL>
adafruit-board-toolkit
argparse
astroid
asttokens
bcrypt
bitstring
blynklib
certifi
cffi
charset-normalizer
colorama
coverage
cryptography
dill
docutils
ecdsa
esp
esp-flasher

blynklib

Installed version: 0.2.6
Installed to: c:\users\lenovo\appdata\roaming\thonny\plugins\python310\site-packages

Latest stable version: 0.2.6
Summary: Blynk Python/Micropython library
Author: Anton Morozenko
License: MIT
Homepage: https://github.com/blynkkk/lib-python
PyPI page: https://pypi.org/project/blynklib/

Steps for programming

- Install BlynkAPP in windows and mobile
- Make new
- Include BlynkLib.py from moodle to Micropython device

- Create new template
- Search-> get device -> from template
- Templates-> device->edit

My organization - 1872CT

MY TEMPLATES

My Templates

BLUEPRINTS BETA

All Blueprints

Templates

Search Templates

DHT11
No devices

ESP32
No devices

Quickstart Template
2 Devices

New Device

Choose a way to create new device

From template	Scan QR code	Manual entry
		

Point on the cards to see instructions

Cancel

My Devices

+ New Device

New Device

Create new device by filling in the form below

TEMPLATE

DHT11

DEVICE NAME

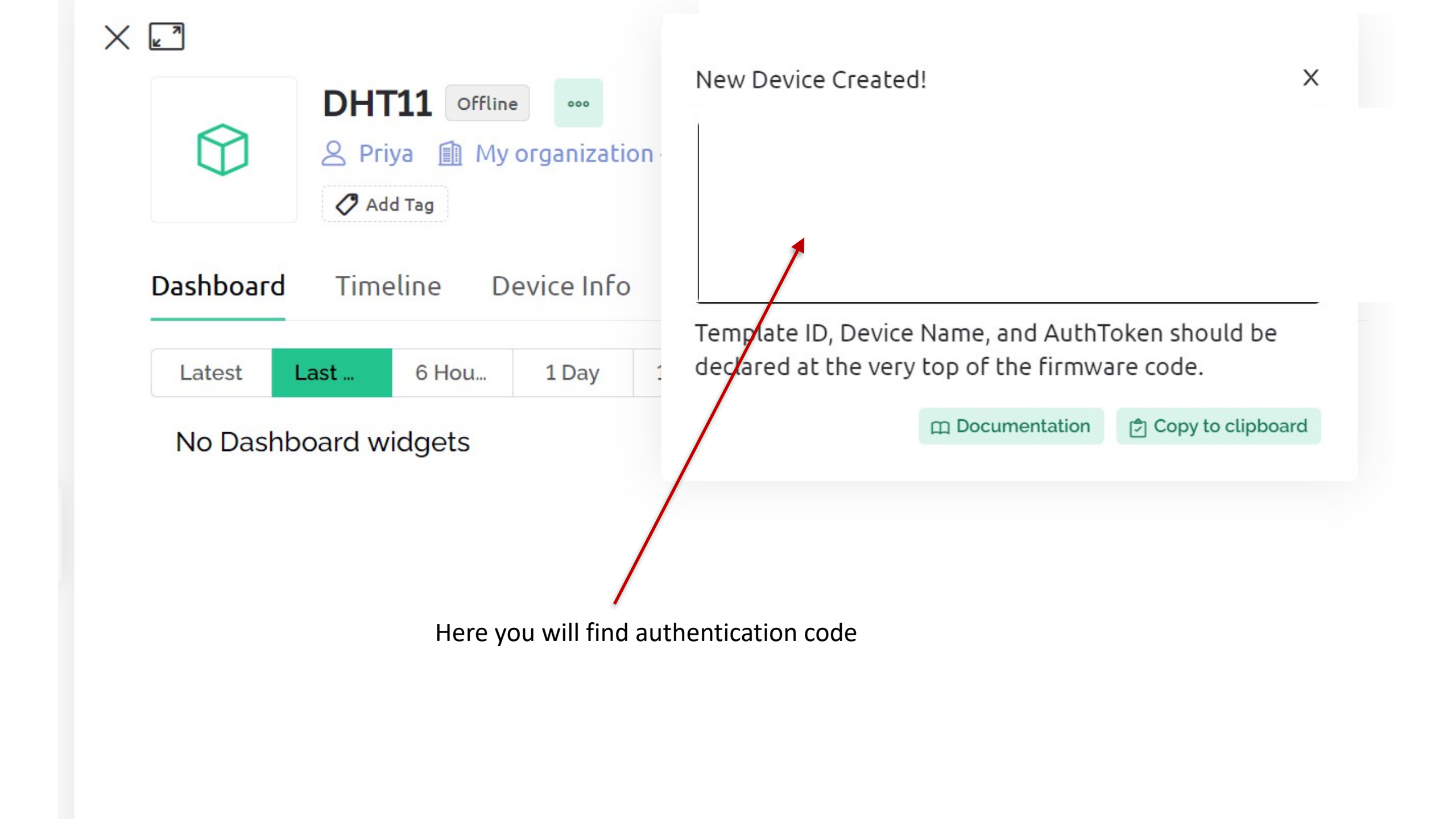
DHT11

Use letters, digits and spaces only

Cancel Create

Device Owner Actions

priya.chimurkar@gmail.com


DHT11
Offline
Priya
My organization
Add Tag
Dashboard
Timeline
Device Info
Latest
Last ...
6 Hou...
1 Day
No Dashboard widgets
New Device Created!
Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.
Documentation
Copy to clipboard
Here you will find authentication code

Add Tag

Dashboard Timeline **Device Info** Metadata Actions Log

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "T
#define BLYNK_TEMPLATE_NAME
#define BLYNK_AUTH_TOKEN "oN
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware

STATUS

Online

LAST UPDATED

5:41 PM Yesterday

DEVICE ACTIVATED

7:40 PM Aug 20, 2023

LATEST METADATA UPDATE

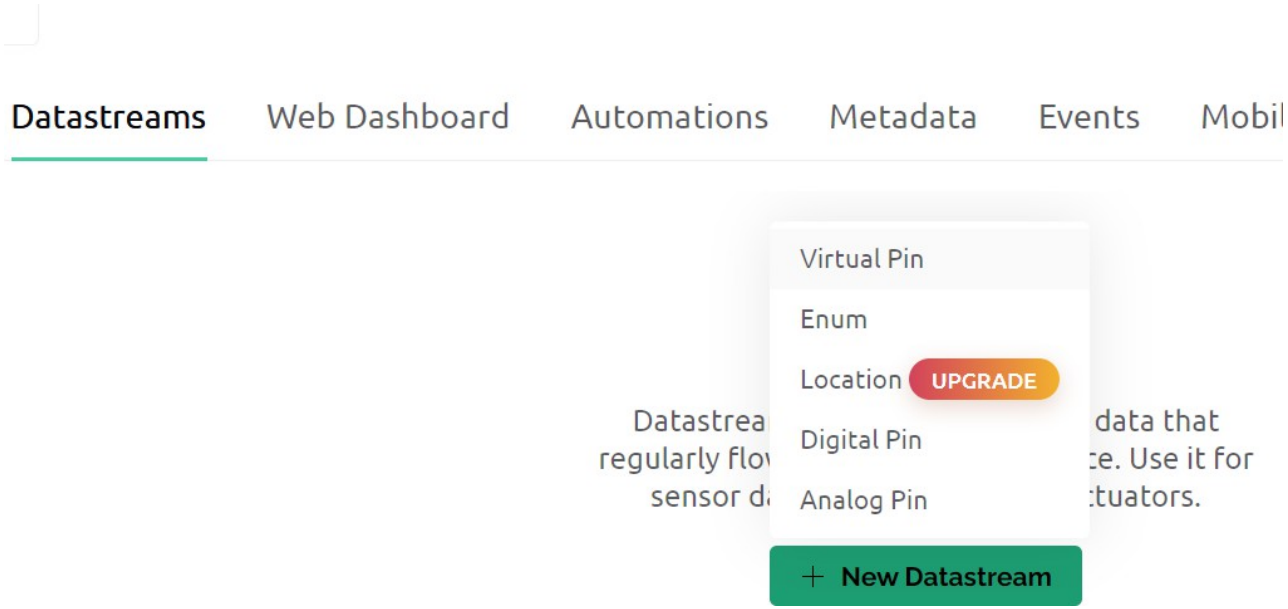
3:26 PM Today

Home Datastreams Web Dashboard Automations Metadata Events Mobile Dashboard

Datastreams

Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.

+ New Datastream



Virtual Pin Datastream

NAME

Humidity

ALIAS

Humidity

PIN

V0

DATA TYPE

Integer

UNITS

Percentage, %

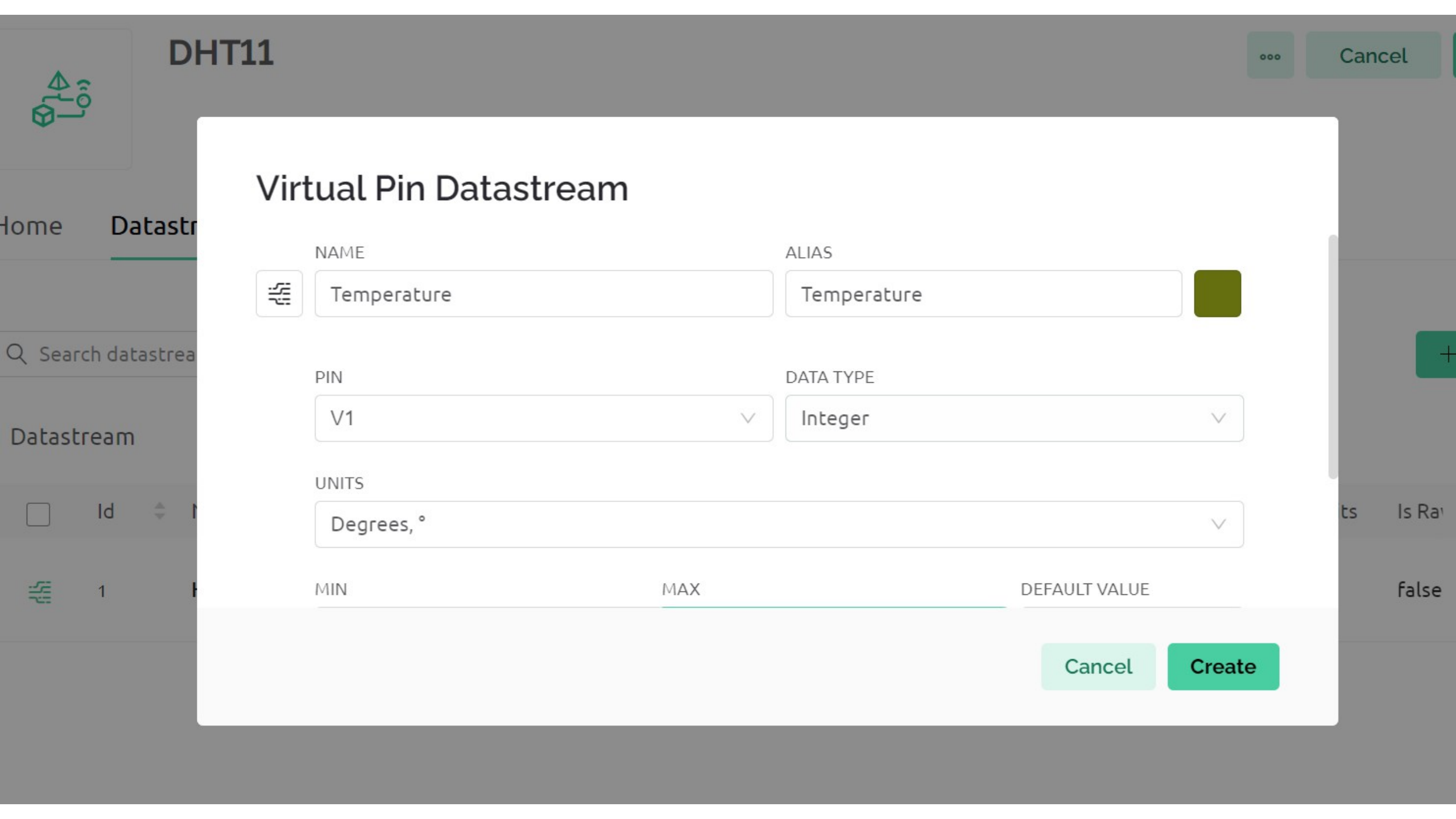
MIN

MAX

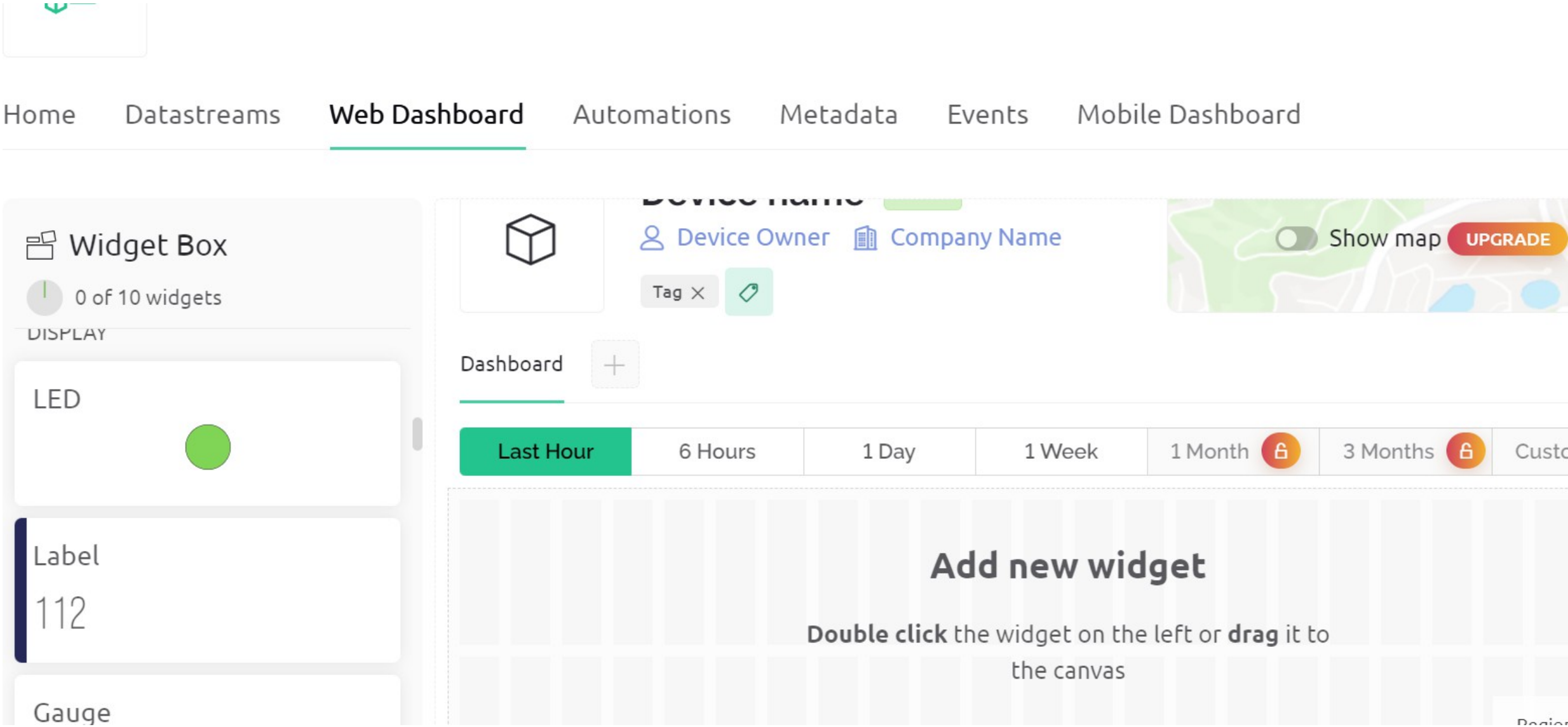
DEFAULT VALUE

Cancel

Create


DHT11
Cancel
Home
Datastr
Search datastrea
Datastream
Id
1
false
Virtual Pin Datastream
NAME
Temperature
ALIAS
Temperature
PIN
V1
DATA TYPE
Integer
UNITS
Degrees, °
MIN
MAX
DEFAULT VALUE
Cancel
Create

Add widget of your choice



Label Settings

TITLE (OPTIONAL)

Humidity

Datastream

Humidity (V0)

CONTENT ALIGNMENT

WIDGET BACKGROUND

Change color based on value

Humidity (V0)

--

Cancel

Save

Boot.py

```
import usocket as socket
import network
from machine import Pin
import dht

import esp
esp.osdebug(None)


ssid = ‘SSID'
password = ‘PASSWORD'

station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())


sensor = dht.DHT11(Pin(5))
```

Main.py

```
import network
import machine
import time

from BlynkLib import Blynk
import dht
# Your Blynk authentication token
auth_token = 'oNTuKCpTT5t7fturLcpX5oKIAtmQ5_S9'

# Set up Wi-Fi connection
ssid = ‘SSID'
password = ‘PASSWORD'

station = network.WLAN(network.STA_IF)

if not station.isconnected():
    print('Connecting to WiFi...')
    station.active(True)
    station.connect(ssid, password)
    while not station.isconnected():
        pass

print('Connected to WiFi:', ssid)
```

```
# Initialize Blynk
blynk = Blynk(auth_token)

# Create a DHT sensor object on pin 5
dht_sensor = dht.DHT11(machine.Pin(5))

# Function to read and send DHT sensor data to Blynk
def read_dht_and_send_to_blynk():
    dht_sensor.measure()
    temperature = dht_sensor.temperature()
    humidity = dht_sensor.humidity()

    # Send temperature to virtual pin V0
    blynk.virtual_write(1, temperature)
    # Send humidity to virtual pin V1
    blynk.virtual_write(0, humidity)
# Main loop
while True:
    try:
        blynk.run()

        # Read and send DHT sensor data to Blynk every 5 seconds
        read_dht_and_send_to_blynk()

        time.sleep(5)
    except KeyboardInterrupt:
        break
    except Exception as e:
        print("Exception:", e)
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Traceback (most recent call last):
  File "boot.py", line 14, in <module>
OSError: Wifi Internal Error


    ___ __                __
   / _ )/ /_ _____  ___  / /__
  / _  / / // / _ \/ _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ for Python v1.0.0 (esp32)

Connected to WiFi: Amol 303
Connecting to blynk.cloud:443...
Exception: [Errno 116] ETIMEDOUT
```