Faculty of Engineering and Technology Electrical and
Computer Engineering Department

ENCS5140

Real-Time System and Interfacing Techniques
Laboratorys

ESP32 and Blynk IoT Platform

An Overview of Features, Applications, and
Implementation Techniques

2024/2025

Contents

List of Figures

1 Introduction

1.1 ESP32 Overview

The **ESP32** is a powerful and versatile microcontroller designed by *Espressif Systems*. It is widely used in IoT applications due to its extensive features and robust capabilities.

1.1.1 Key Features of ESP32

- **Wi-Fi and Bluetooth Connectivity**:
  - Built-in 802.11 b/g/n Wi-Fi for internet connectivity.
  - Bluetooth 4.2 and BLE (Bluetooth Low Energy) for communication with nearby devices.
- **High Processing Power**:
  - Dual-core 32-bit processor with speeds up to 240 MHz.
  - 520 KB SRAM and optional external flash memory.
- **Low Power Consumption**:
  - Multiple power modes to optimize energy efficiency for battery-operated devices.
- **Rich Peripherals**:
  - GPIOs, ADC, DAC, PWM, SPI, I2C, UART, and more for connecting sensors, displays, and actuators.
- **Open-Source SDK**:
  - Supported by Espressif's development frameworks like Arduino, ESP-IDF, and Micropython.

1.2 ESP32 Pinout

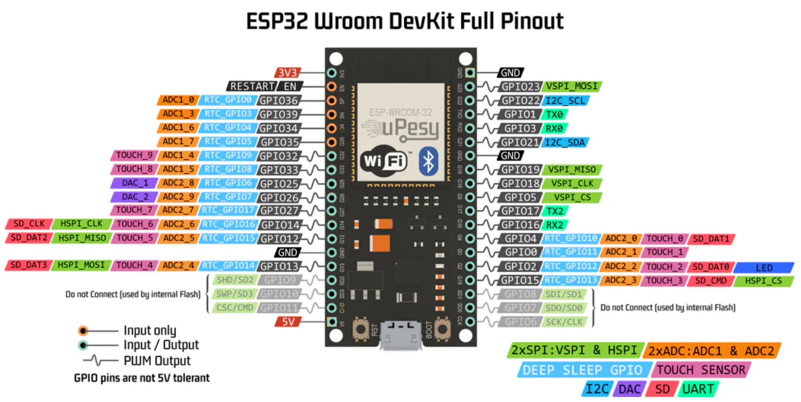

ESP32 Wroom DevKit Full Pinout
3V3
RESTART / EN
ADC1_0 / RTC_GPIO0 / GPIO36
ADC1_3 / RTC_GPIO3 / GPIO39
ADC1_6 / RTC_GPIO4 / GPIO34
ADC1_7 / RTC_GPIO5 / GPIO35
TOUCH_9 / ADC1_4 / RTC_GPIO9 / GPIO32
TOUCH_8 / ADC1_5 / RTC_GPIO8 / GPIO33
DAC_1 / ADC2_8 / RTC_GPIO6 / GPIO25
DAC_2 / ADC2_9 / RTC_GPIO7 / GPIO26
TOUCH_7 / ADC2_7 / RTC_GPIO17 / GPIO27
SD_CLK / HSPI_CLK / TOUCH_6 / ADC2_6 / RTC_GPIO16 / GPIO14
SD_DAT2 / HSPI_MISO / TOUCH_5 / ADC2_5 / RTC_GPIO15 / GPIO12
GND
SD_DAT3 / HSPI_MOSI / TOUCH_4 / ADC2_4 / RTC_GPIO14 / GPIO13
SHD/SD2 / GPIO9
SWP/SD3 / GPIO10
CSC/CMD / GPIO11
Do not Connect (used by internal Flash)
5V
GND
GPIO23 / VSPI_MOSI
GPIO22 / I2C_SCL
GPIO1 / TX0
GPIO3 / RX0
GPIO21 / I2C_SDA
GND
GPIO19 / VSPI_MISO
GPIO18 / VSPI_CLK
GPIO5 / VSPI_CS
GPIO17 / TX2
GPIO16 / RX2
GPIO4 / RTC_GPIO10 / ADC2_0 / TOUCH_0 / SD_DAT1
GPIO0 / RTC_GPIO11 / ADC2_1 / TOUCH_1
GPIO2 / RTC_GPIO12 / ADC2_2 / TOUCH_2 / SD_DAT0 / LED
GPIO15 / RTC_GPIO13 / ADC2_3 / TOUCH_3 / SD_CMD / HSPI_CS
GPIO8 / SDI/SD1
GPIO7 / SDO/SD0
GPIO6 / SCK/CLK
Do not Connect (used by internal Flash)
ESP-WROOM-32
uPesy
Wi Fi
RST
BOOT
Input only
Input / Output
PWM Output
GPIO pins are not 5V tolerant
2xSPI:VSPI & HSPI
2xADC:ADC1 & ADC2
DEEP SLEEP GPIO
TOUCH SENSOR
I2C
DAC
SD
UART

Figure 1: ESP32 Pinout [1]

1.2.1 Applications of ESP32

- Home automation systems.
- Smart agriculture and IoT weather stations.
- Wearable technology and health monitoring.
- Industrial automation and monitoring.

1.3 Blynk IoT Platform Overview

Blynk is a platform that simplifies the development of IoT applications by providing tools for connecting devices, managing them remotely, and visualizing data in real-time.

1.3.1 Key Features of Blynk

- **Cross-Platform Compatibility**:
  - Supports iOS, Android apps, and web dashboards for device monitoring and control.
- **Cloud Connectivity**:
  - Allows IoT devices to connect to the **Blynk Cloud** for data exchange and remote access.
- **Widgets for Visualization**:
  - Create dashboards with widgets like buttons, sliders, charts, and labels.
- **Easy Integration**:
  - Works with popular IoT hardware like ESP32, ESP8266, Arduino, and Raspberry Pi.
- **Template-Based Development**:
  - Supports reusable templates to quickly set up projects and manage multiple devices.
- **Secure Communication**:
  - TLS encryption ensures secure communication between devices and the cloud.

1.3.2 How Blynk Works

1. **Blynk Cloud**: Acts as the central hub for managing devices and data. Devices connect to the cloud using an authentication token.
2. **Mobile App**: Provides a graphical interface for users to monitor and control their devices remotely.
3. **Device Firmware**: Runs on the microcontroller (e.g., ESP32) to communicate with sensors and actuators and send/receive data from the Blynk Cloud.

1.3.3 ESP32 with Blynk: Key Details

- **Setup Process**:
  - Install the Blynk library in the Arduino IDE.
  - Configure the device with Wi-Fi credentials, authentication token, and Blynk template details.
- **Authentication Token**: A unique string provided by Blynk that connects the ESP32 to the cloud project.
- **Template Configuration**: Templates in the Blynk Cloud define the structure of the data and the interface in the app.
- **Programming Flow**:
  - Initialize Wi-Fi and Blynk libraries in the firmware.
  - Use `Blynk.run()` in the main loop to handle communication with the Blynk Cloud.
- **Supported Sensors and Actuators**: ESP32 can interface with temperature sensors, motion detectors, relays, LEDs, and more. Data from these devices can be visualized and controlled via the Blynk app.

1.3.4 Advantages of Using ESP32 with Blynk

- **Rapid Development**: Simplifies complex IoT tasks like cloud integration and app creation.
- **Remote Access**: Monitor and control devices from anywhere using the Blynk app or web dashboard.
- **Cost-Effective**: ESP32 is affordable and Blynk offers free and subscription-based plans.
- **Scalable**: Suitable for single-device hobby projects or large-scale IoT deployments.

1.3.5 Example Use Case: IoT-Controlled LED via Blynk

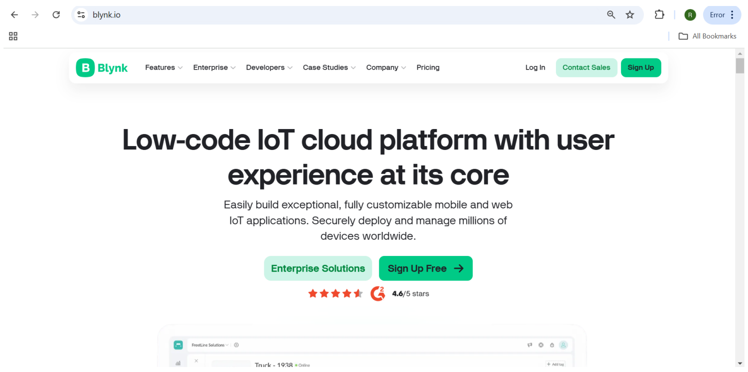
- **Hardware**: ESP32, LED, and a resistor.
- **Software**: Blynk app.
- **Implementation**:
  - Configure the ESP32 to connect to Wi-Fi and the Blynk Cloud.
  - Create a virtual button widget in the Blynk app to control the LED.
  - Use the ESP32 firmware to control the LED's state based on commands from the Blynk app (toggle between ON and OFF).
  - Set up a virtual LCD widget in the Blynk app to display the current state of the LED (ON/OFF).

2 Getting Started with Blynk

In this section, we will guide you through the steps required to set up your Blynk account and the app for controlling your IoT devices.

2.1 Creating an Account on the Blynk Website

First, we need to create an account on the Blynk website. You can visit their official site at https://blynk.io/ to sign up.



Low-code IoT cloud platform with user experience at its core

Easily build exceptional, fully customizable mobile and web IoT applications. Securely deploy and manage millions of devices worldwide.

Figure 2: Blynk Website

Once you've signed up, you can log in to your account and start building your IoT projects.

2.2 Downloading the Blynk App

Next, download the Blynk app. It is available for both Android and iOS devices. Simply search for "Blynk" in the Google Play Store or Apple App Store.

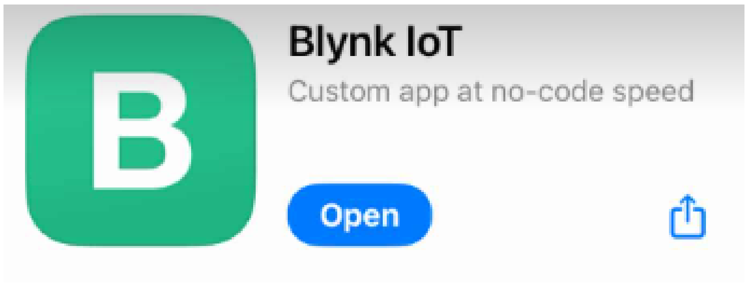


Figure 3: Blynk App

Install the app on your smartphone, and once it's installed, open the app.

2.3 Logging in to the App

After opening the Blynk app, log in with the same credentials you used to sign up on the Blynk website. This ensures that your app is connected to the same account, and you can sync your projects between the app and the website.

3 Setting Up the Blynk Project

3.1 Creating a New Template

First, we need to create a new template on the Blynk platform. A Blynk template is a reusable structure that contains your IoT project settings, such as datastreams, devices, and widgets. Templates allow for easier management and scalability of your IoT solutions.

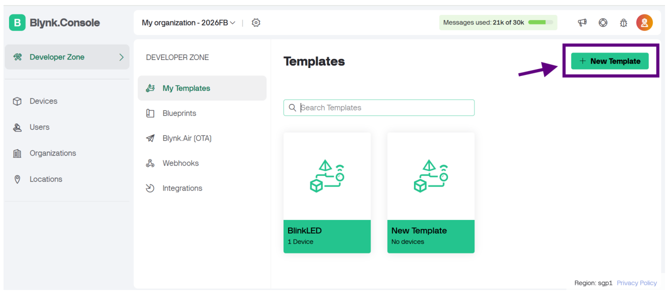
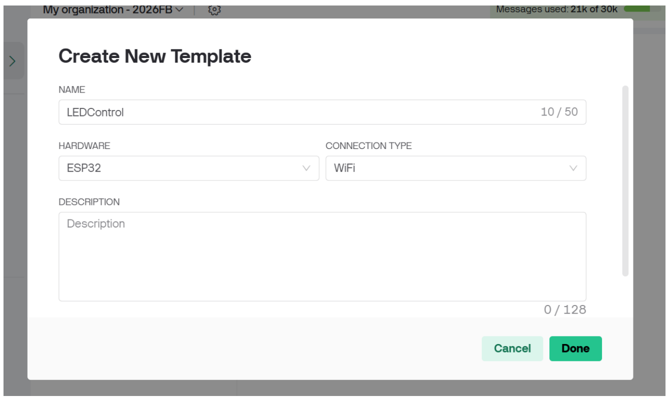


Figure 4: Creating a New Template

Once the new template is created, fill in the required information such as the template name, device type, and connection type (e.g., Wi-Fi for ESP32).



Create New Template

NAME: LEDControl 10 / 50

HARDWARE: ESP32

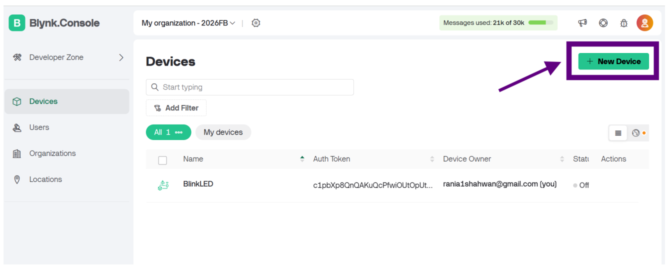
CONNECTION TYPE: WiFi

DESCRIPTION: Description 0 / 128

Figure 5: Template Options

3.2 Creating a New Device

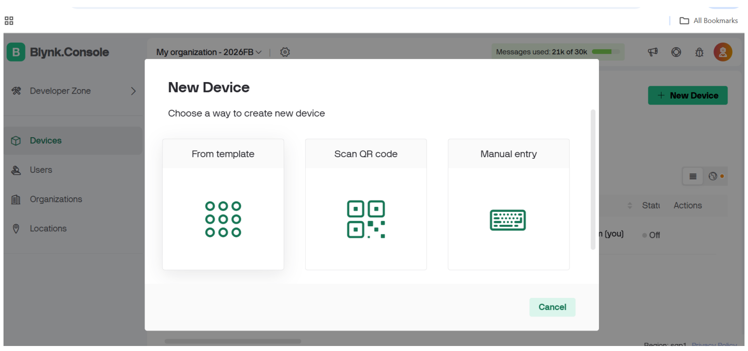
After defining the template, create a new device based on this template.



Name	Auth Token	Device Owner	Stat	Actions
BlinkLED	c1pbXp8QnQAKuQcPfwiOUtOpUt...	rania1shahwan@gmail.com (you)	Off	

Figure 6: Adding a New Device

Select the "From Template" option to link the new device to the template you created earlier.

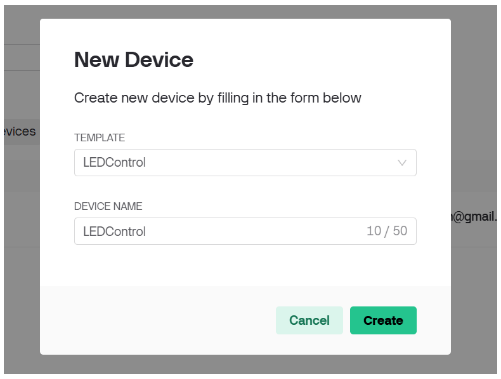


New Device

Choose a way to create new device

From template | Scan QR code | Manual entry

Figure 7: Selecting the "From Template" Option



New Device

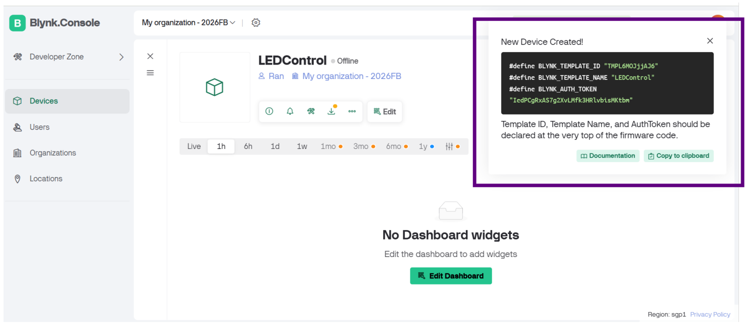
Create new device by filling in the form below

TEMPLATE: LEDControl

DEVICE NAME: LEDControl 10 / 50

Figure 8: Link with the Template

Once the new device is created, the platform will generate and display the Template ID, Template Name, and Auth Token. Copy these details to use in your code later.



New Device Created!

```
#define BLYNK_TEMPLATE_ID "TMPL6MOJjjAJ6"
#define BLYNK_TEMPLATE_NAME "LEDControl"
#define BLYNK_AUTH_TOKEN
"IedPCgRxAS7g2XvLMfk3HRlvbisMKtbm"
```

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

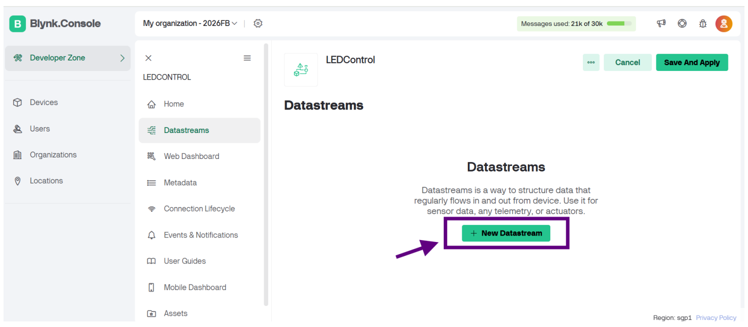
Figure 9: Retrieving Template Information

The device is an IoT-enabled microcontroller (ESP32) that can be connected to the Blynk cloud platform, allowing users to control and monitor hardware components such as LEDs, buttons, and other peripherals through a mobile app or web interface.

3.3 Adding Datastreams for LCD and Button

In Blynk, a datastream is a channel that defines how data is exchanged between your IoT device and the Blynk platform. It specifies the pin (virtual, digital, or analog), data type (e.g., integer, string), and direction (input or output), enabling features like controlling devices or displaying data on the app/dashboard.
Create new datastreams for the virtual LCD and the button:

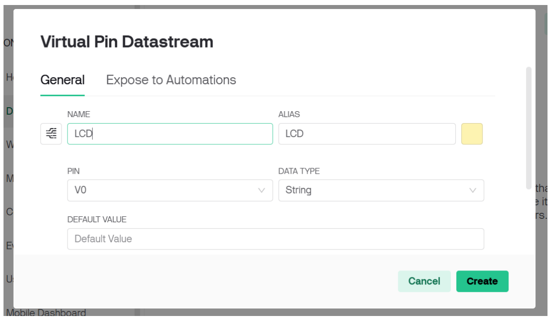
- Virtual LCD: Assign to a virtual pin (e.g., V0) with data type set to 'String'.
- Virtual Button: Assign to a virtual pin (e.g., V1) with data type 'Integer'.



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

Figure 10: Creating New Datastreams

Assign a virtual pin for the LCD (e.g., V0) and configure the button on another pin (e.g., V1).

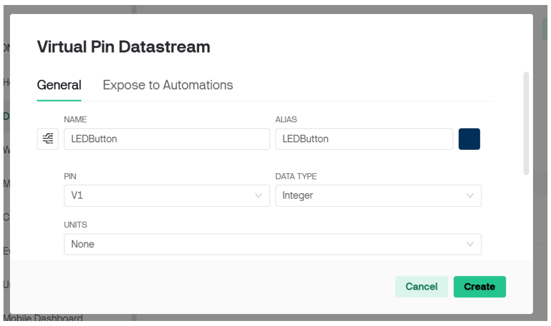


Virtual Pin Datastream

General | Expose to Automations

NAME: LCD
ALIAS: LCD
PIN: V0
DATA TYPE: String
DEFAULT VALUE: Default Value

Figure 11: Virtual LCD Configuration



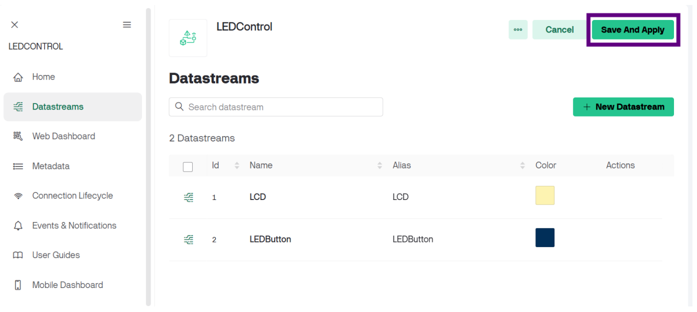
Virtual Pin Datastream

General | Expose to Automations

NAME: LEDButton
ALIAS: LEDButton
PIN: V1
DATA TYPE: Integer
UNITS: None

Figure 12: Virtual Button Configuration

Once all datastreams are configured, the datastream page will look like the following. Don't forget to **save and apply** your changes.



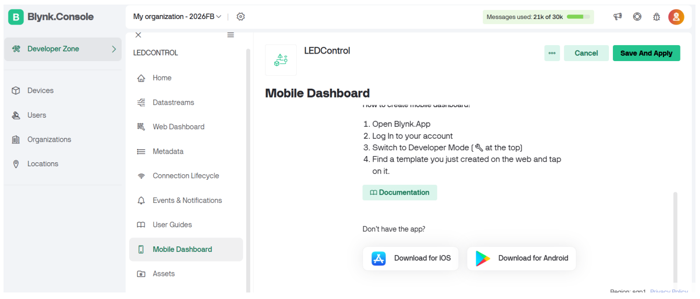
2 Datastreams

Id	Name	Alias	Color	Actions
1	LCD	LCD	🟨	
2	LEDButton	LEDButton	🟦	

Figure 13: Final Datastream Setup

3.4 Accessing the Mobile App

The Blynk mobile app can be used to control and monitor your device. You can find the app link on the mobile dashboard.



1. Open Blynk.App
2. Log In to your account
3. Switch to Developer Mode (at the top)
4. Find a template you just created on the web and tap on it.

Figure 14: Mobile App

3.5 Editing the Dashboard and Configuring the Switch

After creating the necessary datastreams, we will edit the Blynk dashboard to add a switch widget for controlling the LED.

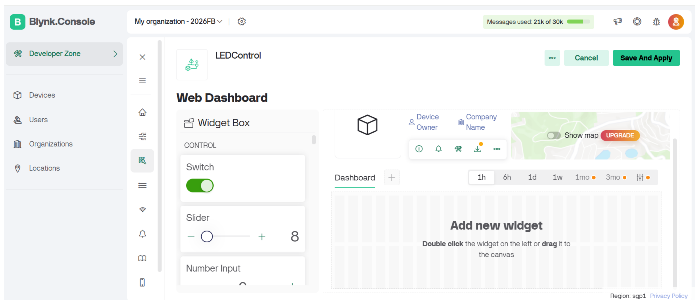
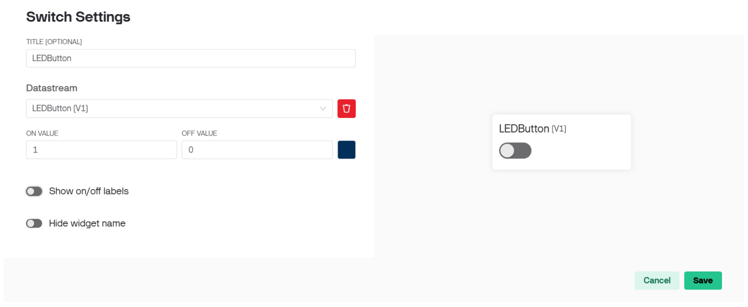


Figure 15: Adding a Switch to the Dashboard

For the switch settings, we need to specify the datastream we created earlier (the LED button). This ensures the switch on the dashboard will control the LED connected to the ESP32.



Switch Settings

TITLE (OPTIONAL): LEDButton

Datastream: LEDButton (V1)

ON VALUE: 1

OFF VALUE: 0

Figure 16: Configuring Switch Settings

Make sure to **Save & Apply** the settings after making the changes.

3.6 Adding Hotspot Name for Network Configuration

Next, we need to add the hotspot name for the ESP32 to connect to your local network. To do this, click on the **Info & Metadata** option in the dashboard.

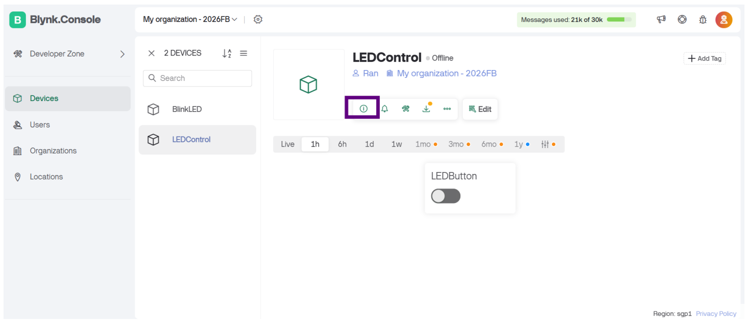
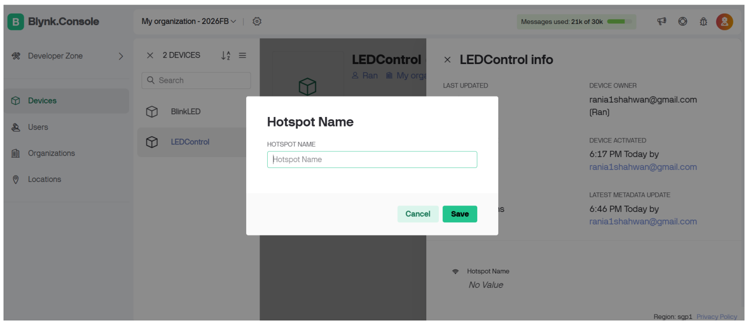


Figure 17: press on info and meta

Then, input your **hotspot name** (SSID) in the appropriate field.



LAST UPDATED

DEVICE OWNER: rania1shahwan@gmail.com (Ran)

DEVICE ACTIVATED: 6:17 PM Today by rania1shahwan@gmail.com

LATEST METADATA UPDATE: 6:46 PM Today by rania1shahwan@gmail.com

Hotspot Name: No Value

Figure 18: Adding Hotspot Name

After these steps, your device will be configured to connect to the specified Wi-Fi network and communicate with the Blynk app.

4 Configuring Widgets in the Blynk App

Next, we will open the Blynk application and add the necessary widgets. Since there are many widget options available, we will start by choosing the template we will be working with. After selecting the template, click on **Edit** to begin customization.

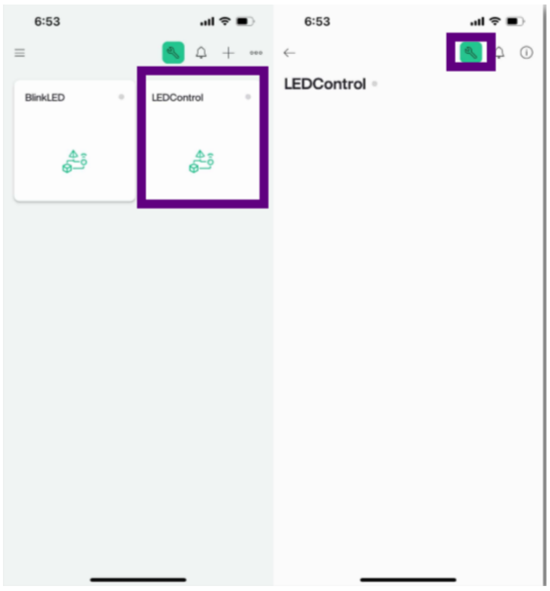


Figure 19: Select Template and Edit

Once in the edit mode, we will add a **Button** widget. This button will control the LED based on the datastream we created earlier. You can also explore the display settings the button's appearance.

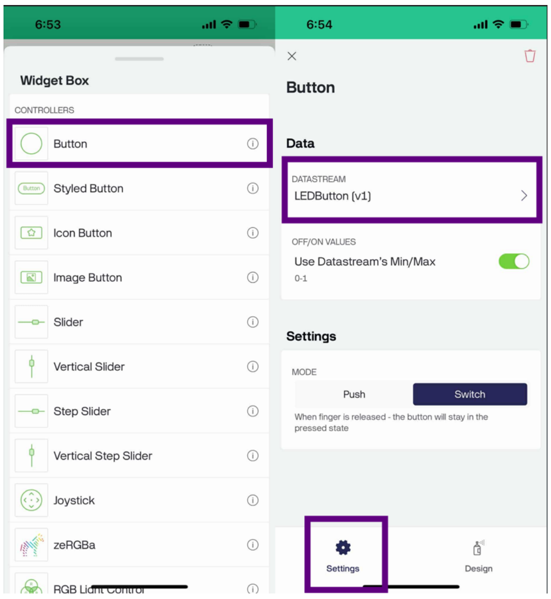
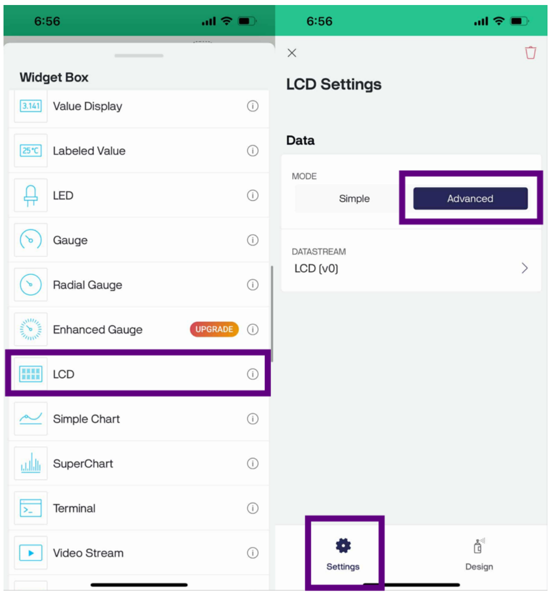


Figure 20: Button Widget Configuration

After adding the button, we will proceed to add an **LCD** widget. This widget will display information about the LED state, and its datastream must be configured to the previously created LCD datastream. Again, you can explore the display settings to adjust the appearance and layout of the LCD widget.



MODE
Simple [✓ Advanced]

DATASTREAM
LCD (v0)

Figure 21: LCD Widget Configuration

At this stage, our app interface will look like the following, with the button and LCD widgets properly configured to control and display the LED state.

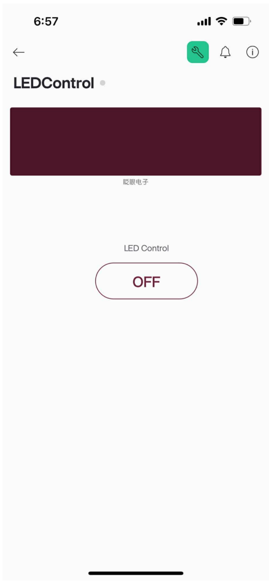


Figure 22: Initial Interface Layout

5 ESP32 Connection and Code

Now, let's connect the LED to the ESP32. The LED will be connected to GPIO pin 2, along with a pull-up resistor. Note that there is also a built-in LED on pin 2 of the ESP32, which can be controlled through the same GPIO pin.

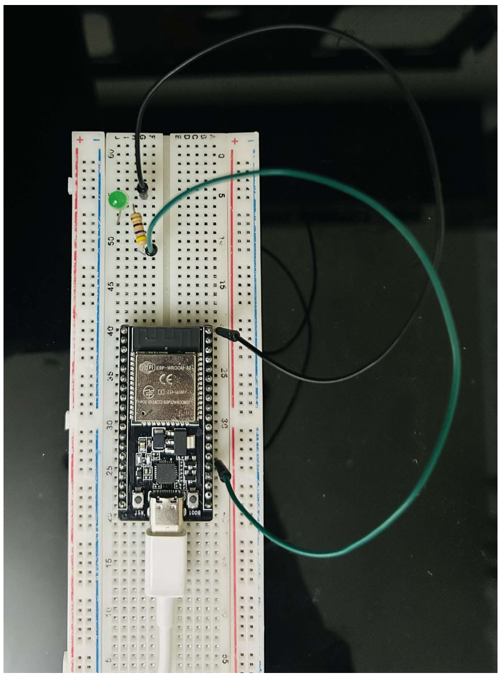


Figure 23: ESP32 Pin Connection for LED

Next, we will write the code in Arduino IDE, ensuring that the necessary ESP32 and Blynk libraries are installed. The code begins by defining the template ID, template name, and authentication token, which were generated during the creation of the template in the Blynk app.
The logic of the code is designed to control an LED on the ESP32 using a virtual button in the Blynk app.The logic behind the system is as follows (the full code is provided in the appendix):

1. **Setup:**
    - The code begins by including necessary libraries for Wi-Fi and Blynk communication.
    - The ESP32 is connected to a Wi-Fi network using the provided SSID and password.
    - A virtual LCD widget is initialized on virtual pin V0, which will display the LED state (ON/OFF).
2. **Button Press Event (BLYNK_WRITE(V1)):**
    - The Blynk app sends a signal to the ESP32 when the virtual button on virtual pin V1 is pressed.
    - The function `BLYNK_WRITE(V1)` is triggered, which reads the button's state (either 0 for OFF or 1 for ON).

3. **LED Control:**
   - If the button is pressed (state is 1), the LED is turned ON by writing a HIGH value to GPIO pin 2.
   - If the button is not pressed (state is 0), the LED is turned OFF by writing a LOW value to GPIO pin 2.
4. **LCD Update:**
   - The LCD widget is updated to display the current state of the LED. When the LED is ON, the LCD displays "LED State: ON". When the LED is OFF, it displays "LED State: OFF".
5. **Looping:**
   - The `loop()` function continuously runs the `Blynk.run()` command to maintain communication with the Blynk server and update the device's state.

6 Testing the System

Let's test the system. After uploading the code to the ESP32, we will be able to control its state (ON or OFF) using the Blynk app. We will notice that the app interface updates in real time as we toggle the button.

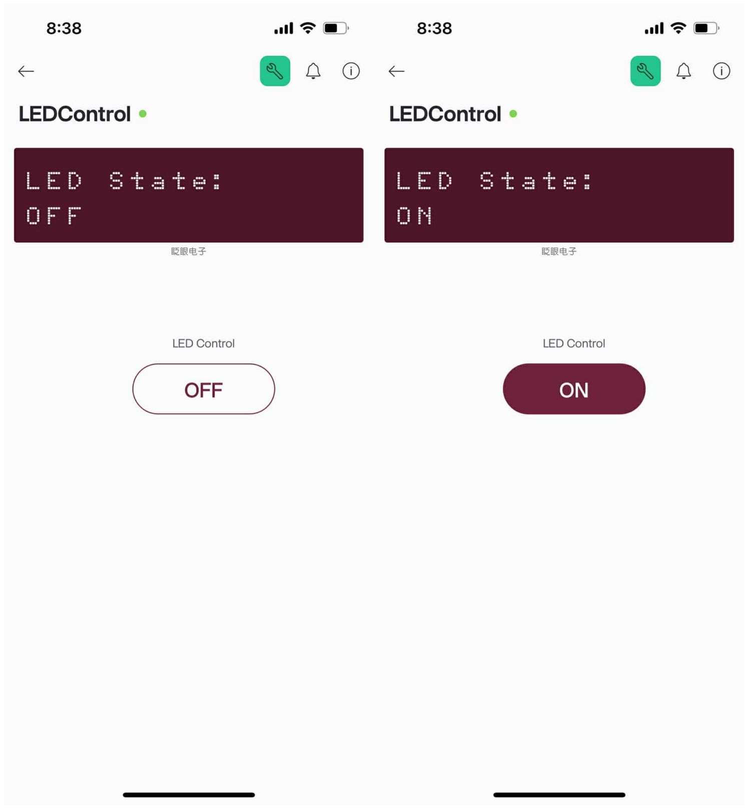


Figure 24: App ON/OFF states

For instance, when we press the "ON" button, the switch on the Blynk website will also be updated to "ON," and the LED on the ESP32 will turn on.

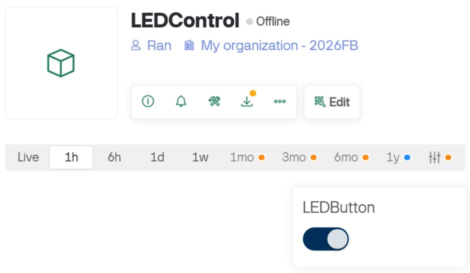


Figure 25: Website switch updated to ON

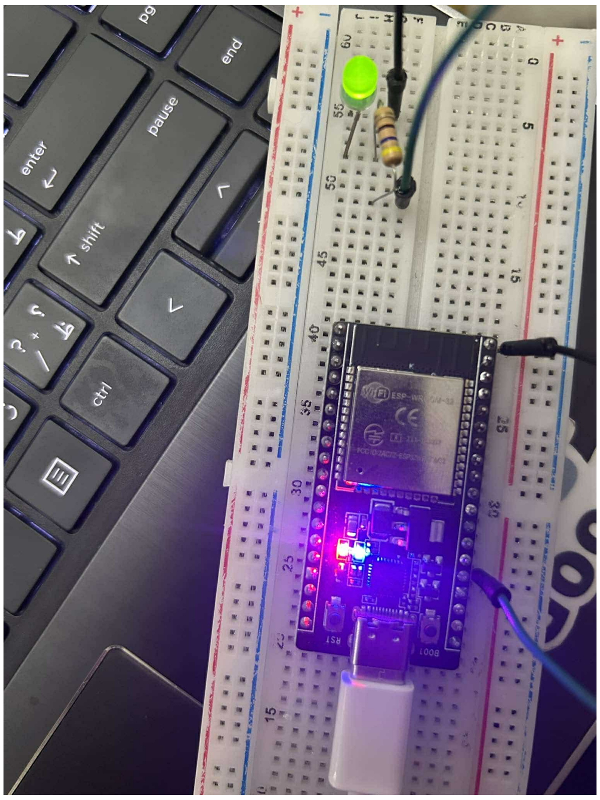


Figure 26: LED turned ON on ESP32

7 References

[1] https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide , Last accessed: 12/12/2024

[2] https://www.nabto.com/guide-to-iot-esp-32/ , Last accessed: 12/12/2024

[3] https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/ , Last accessed: 12/12/2024

[4] https://docs.blynk.io/en , Last accessed: 13/12/2024

[5] https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/google$_vignette, Lastaccessed$: 13/12/2024

8 Appendix

8.1 Code

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL6MOJjjAJ6"
#define BLYNK_TEMPLATE_NAME "LEDControl"
#define BLYNK_AUTH_TOKEN "IedPCgRxAS7g2XvLMfk3HRlvbisMKtbm"

// Include the library files
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

// Enter your Auth token
char auth[] = BLYNK_AUTH_TOKEN;

// Enter your WIFI SSID and password
char ssid[] = "Rania";
char pass[] = "rania123" ;

// Define LED pin
const int ledPin = 2;

// Virtual LCD widget on V0
WidgetLCD lcd(V0);

void setup() {
  // Debug console
  Serial.begin(9600);
  // Configure LED pin as output
  pinMode(ledPin, OUTPUT);
  // Start Blynk
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
}

// Function to handle the virtual button on V1
BLYNK_WRITE(V1) {
  int buttonState = param.asInt(); // Read button state (0 or 1)

  if (buttonState == 1) {
    // Turn LED ON
    digitalWrite(ledPin, HIGH);
    lcd.print(0, 0, "LED State:");
    lcd.print(0, 1, "ON ");
  } else {
    // Turn LED OFF
    digitalWrite(ledPin, LOW);
    lcd.print(0, 0, "LED State:");
    lcd.print(0, 1, "OFF");
  }
}

void loop() {
  // Run Blynk process
  Blynk.run();
}
```

Listing 1: ESP32 Code for LED Control