Module 8: IOT Platform

Blynk

Introduction

Blynk is a platform that allows you to create and control IoT applications using a smartphone, tablet, or web browser. You can use Blynk to connect various hardware devices, such as Arduino, Raspberry Pi, ESP8266, ESP32, and others, to the internet and interact with them through a graphical user interface. Blynk provides a widget library that you can drag and drop to design your application, including buttons, sliders, displays, charts, and more. You can also use Blynk to monitor and analyze data from your sensors, send notifications, and integrate with third-party services.

By using Blynk with the ESP32, you can easily and quickly prototype your IoT projects without writing complex code or dealing with network protocols. You can use Blynk to control ESP32 outputs, such as LEDs, relays, motors, and more, as well as read inputs from sensors like temperature, humidity, light, and others. Blynk also allows you to create custom functions and logic for your ESP32, such as timers, triggers, and actions.

Setting Up Blynk

Please download the Blynk app on your smartphone. For more information on how to register an account, create a template, configure datastreams, set up the web dashboard, and configure the mobile dashboard, you can watch this video tutorial.

Blynk Syntax

Blynk events are actions that occur when you interact with your Blynk application or when your hardware sends or receives data from the Blynk server. You can use Blynk events to control the logic and behavior of your ESP32, such as turning devices on or off, reading or writing data, performing calculations, and more. You can handle Blynk events using Blynk library functions and

Arduino sketches.

The Blynk library provides several functions you can use to handle Blynk events, such as:

- **Blynk.begin()**: This function initializes the Blynk connection and connects your ESP32 to your Blynk project. You need to call this function in the `setup()` function of your Arduino sketch and provide your WiFi credentials and Blynk authentication token as parameters. Example:

```
void setup() {
  Serial.begin(115200);
  // Initialize Blynk connection
  Blynk.begin(auth, ssid, pass);
}
```

- **Blynk.run()**: This function runs the Blynk process and handles communication between your ESP32 and your Blynk project. You need to call this function in the `loop()` function of your Arduino sketch, ensuring it is not blocked by long-running code. Example:

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

- **BLYNK_WRITE()**: This function is a macro that defines a callback function executed when a widget writes a value to a pin on your ESP32. You can use this function to handle events triggered by your Blynk app, such as pressing a button, moving a slider, or sending a command. You need to specify the pin number as a parameter and use the `BlynkParam` object to access the value. Example:

```
// Define a callback function for pin V1
BLYNK_WRITE(V1) {
  // Get value from widget
  int value = param.asInt();
  // Do something with the value
  Serial.println(value);
}
```

- **Blynk.virtualWrite()**: This function writes a value to a virtual pin in your Blynk project. You can use this function to send data from your ESP32 to your Blynk app, such as sensor readings, status updates, or custom messages. You need to specify the virtual pin number and the value as parameters. You can also specify the data type, such as `int`, `float`, `string`, or `array`. Example:

```
// Write value to pin V2
Blynk.virtualWrite(V2, 123);
// Write string to pin V3
Blynk.virtualWrite(V3, "Hello Blynk");
// Write array to pin V4
int array[] = {1, 2, 3, 4, 5};
Blynk.virtualWrite(V4, array, 5);
```

- **BLYNK_READ()**: This function is a macro that defines a callback function executed when a widget reads a value from a pin on your ESP32. You can use this function to handle events triggered by your Blynk app, such as sensor value requests, display updates, or chart refreshes. You need to specify the pin number as a parameter and use the `Blynk.virtualWrite()` function to send the value. Example:

```
// Define a callback function for pin V5
BLYNK_READ(V5) {
  // Read value from sensor
  int value = analogRead(A0);
  // Send value to widget
  Blynk.virtualWrite(V5, value);
}
```

Extras

Blynk also offers several advanced features that you can use to enhance your ESP32 project, such as:

- **Virtual Pins**: Virtual pins are pins that do not correspond to physical pins on your ESP32 but are linked to virtual pins in your Blynk project. You can use virtual pins to exchange various types of data between your ESP32 and Blynk app, such as strings, arrays, or custom objects. Virtual pins also allow you to implement custom functions and logic for your ESP32, such as timers, triggers, and actions. To use virtual pins, you need to use the `BLYNK_WRITE()`, `Blynk.virtualWrite()`, and `BLYNK_READ()` functions, as explained in the previous section.
- **Blynk.syncVirtual()**: This function synchronizes the value of a virtual pin between your ESP32 and your Blynk app. You can use this function to update widget values in your Blynk app with values from your ESP32, or vice versa. This function can also trigger the `BLYNK_WRITE()` or `BLYNK_READ()` callback functions for the virtual pin. You need to specify the virtual pin number as a parameter. Example:

```
// Sync the value of pin V6 from ESP32 to the Blynk app
Blynk.syncVirtual(V6);
// Sync the value of pin V7 from the Blynk app to ESP32
// This will also execute the BLYNK_WRITE(V7) callback function
Blynk.syncVirtual(V7);
```

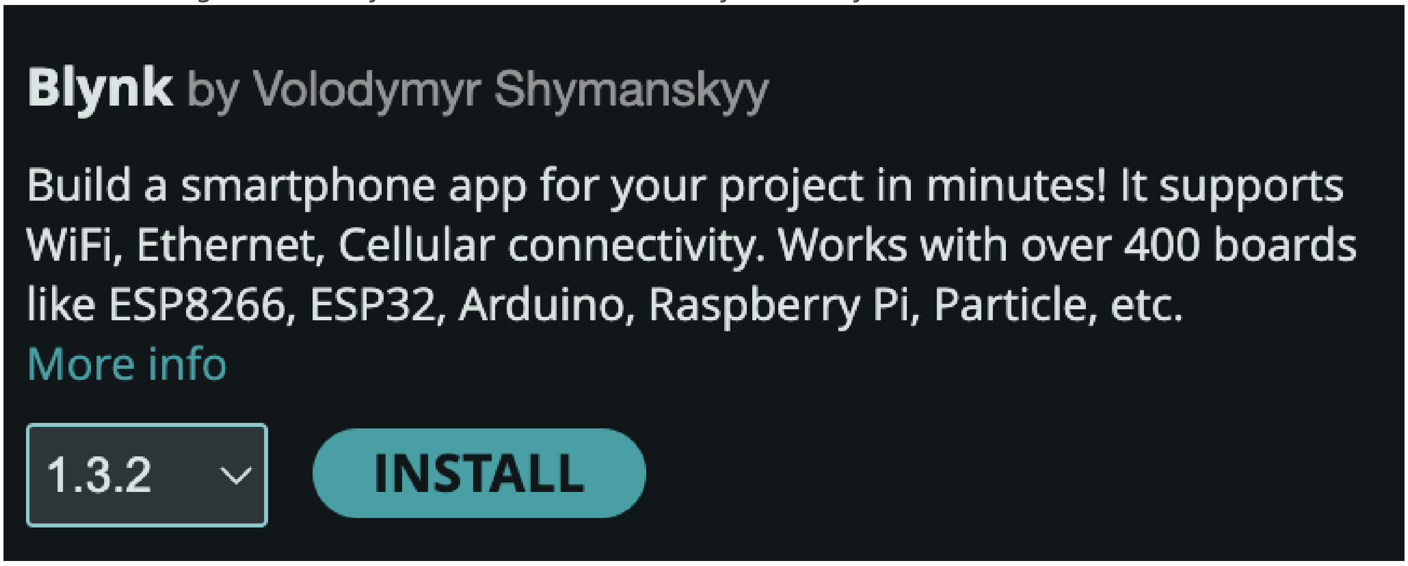
- **Custom Functions**: Custom functions are functions that you can define in your Arduino sketch to perform specific tasks or actions for your ESP32. You can use custom functions to implement complex logic and behavior for your ESP32, such as controlling multiple devices, performing calculations, sending notifications, and more. Custom functions can also handle events from your Blynk app, such as pressing buttons, moving sliders, or sending commands. To use custom functions, use the `BLYNK_WRITE()` function to call the custom function and pass the widget value as a parameter. Example:

```
// Define a custom function to blink an LED
void blinkLED(int value) {
  // Turn on the LED
  digitalWrite(LED_BUILTIN, HIGH);
  // Wait for the specified value in milliseconds
  delay(value);
  // Turn off the LED
  digitalWrite(LED_BUILTIN, LOW);
  // Wait for the specified value in milliseconds
  delay(value);
}

// Call the custom function from pin V8
BLYNK_WRITE(V8) {
  // Get the value from the widget
  int value = param.asInt();
  // Call the custom function
  blinkLED(value);
}
```

Example Code

Before running this code, you need to install the Blynk library in the Arduino IDE.


Blynk by Volodymyr Shymanskyy

Build a smartphone app for your project in minutes! It supports WiFi, Ethernet, Cellular connectivity. Works with over 400 boards like ESP8266, ESP32, Arduino, Raspberry Pi, Particle, etc.
More info

In this example, we will use Blynk and ESP32 to monitor temperature and humidity in a room using the DHT11 sensor (if used, don't forget to install the DHT library first). We will also display the sensor readings on the LCD widget and gauge widget in the Blynk app. Additionally, we will use the LED widget to indicate the connection status of the ESP32. The circuit diagram and code can be seen below:

```
// Include the Blynk and DHT libraries
#include <BlynkSimpleEsp32.h>
#include <DHT.h>
#include <WiFi.h>
#include <WiFiClient.h>

// Define Blynk authentication token
#define BLYNK_TEMPLATE_ID "YourTemplateID"
#define BLYNK_DEVICE_NAME "YourDeviceName"
#define BLYNK_AUTH_TOKEN "YourToken"

// Define WiFi credentials
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

// Define DHT sensor type and pin
#define DHTTYPE DHT11
#define DHTPIN 4

// Create DHT object
DHT dht(DHTPIN, DHTTYPE);
```

```
// Define virtual pins for widgets
#define LED_VPIN V0
#define LCD_VPIN V1
#define GAUGE_VPIN V2

// Define LED pin
#define LED_PIN 25

// Define update interval in milliseconds
#define UPDATE_INTERVAL 2000

// Create a timer object
BlynkTimer timer;

// Define function to read and send sensor data
void sendSensorData() {
  // Read temperature and humidity from sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Check if readings are valid
  if (isnan(temperature) || isnan(humidity)) {
    // Display error message on LCD widget
    Blynk.virtualWrite(LCD_VPIN, "clear");
    Blynk.virtualWrite(LCD_VPIN, 0, 0, "DHT Sensor");
    Blynk.virtualWrite(LCD_VPIN, 0, 1, "error");
  } else {
    // Display readings on LCD widget
    Blynk.virtualWrite(LCD_VPIN, "clear");
    Blynk.virtualWrite(LCD_VPIN, 0, 0, "Temp: " + String(temperature) + " C");
    Blynk.virtualWrite(LCD_VPIN, 0, 1, "Humidity: " + String(humidity) + " %");
    // Display readings on gauge widget
    Blynk.virtualWrite(GAUGE_VPIN, temperature);
  }
}

// Define function to indicate connection status
void indicateConnection() {
  // Check if ESP32 is connected to Blynk
  if (Blynk.connected()) {
    // Turn on LED widget
    Blynk.virtualWrite(LED_VPIN, 255);
  } else {
    // Turn off LED widget
    Blynk.virtualWrite(LED_VPIN, 0);
  }
}

// Define function to handle button events
BLYNK_WRITE(25) {
  // Get the value from the button widget
  int value = param.asInt();
  // Write value to LED pin
  digitalWrite(LED_PIN, value);
}

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Initialize Blynk connection
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  // Initialize DHT sensor
  dht.begin();

  // Initialize LED pin
  pinMode(LED_PIN, OUTPUT);

  // Set LED widget to off
  Blynk.virtualWrite(LED_VPIN, 0);

  // Clear LCD widget
  Blynk.virtualWrite(LCD_VPIN, "clear");

  // Set gauge widget to 0
  Blynk.virtualWrite(GAUGE_VPIN, 0);

  // Set timer to call sendSensorData function every UPDATE_INTERVAL milliseconds
  timer.setInterval(UPDATE_INTERVAL, sendSensorData);
  // Set timer to call indicateConnection function every 100 milliseconds
  timer.setInterval(100, indicateConnection);
}

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

Instructions for the Blynk app:

1. Open the Blynk app and create a new project with the ESP32 device model and WiFi connection type. Copy the authentication token and paste it into the code.
2. Add an LED widget and assign it to pin V0. Set the color to green.
3. Add an LCD widget and assign it to pin V1. Set the advanced mode to ON and the text color to blue.
4. Add a gauge widget and assign it to pin V2. Set the range to 0-50 and the text color to red.
5. Select a button widget. Tap the button widget to open its settings. Set the name to "LED Control" and the color to green. Set the output to digital pin 25 and the mode to switch. This will link the button to the LED pin on the ESP32.
6. Upload the code to the ESP32 and run the project. You should see temperature and humidity readings on the LCD and gauge widgets. You should also see the LED widget light up when the ESP32 is connected to Blynk.

Revision #1
Created 14 November 2024 03:46:32 by AM
Updated 14 November 2024 03:56:30 by AM