

ESP32 and BLE

Bluetooth Communcation



Bluetooth

Wireless communication protocol to be used over short distances.

BLE

Bluetooth Low Energy

Uses less energy than regular bluetooth (sometimes called bluetooth classic).

Can transmit less data over shorter distances.

Unlike Classic Bluetooth, LE stays in standby mode until its needed.

Can consume 1/100 energy of Classic BT.

Comparisons

	Bluetooth Classic	Bluetooth Low Energy (BLE)
Power Consumption	Higher power consumption	Low power consumption
Data Transfer Rate	Higher data transfer rates	Lower data transfer rates
Range	Longer range	Shorter range
Application Examples	Audio streaming, file transfer	IoT devices, wearables, smart home
Data Transfer	Serial Port Profile (SPP)	Generic Attribute Profile (GATT)

Dabble

App for communicating via BT with Arduino and ESP32 microcontrollers.

It allows you to use your phone to interact with the ESP32 as well as make use of your phone's sensors.

[dabble - Android Apps on Google Play](#)

[Dabble - Bluetooth Controller on the App Store](#)

Arduino Library

Install the DabbleESP32 library.

DabbleESP32 by STEMpedia...
Dabble is a library to interface ESP32 with Dabble Smartphone app on Arduino IDE. Dabble app...
[More info](#)

1.5.1 ▾

INSTALL

Change the ESP32 API to 2.0.17

Tools→Board Manager

Search for ESP32 by Espressif

Change to version 2.0.17 and click Install

Tools→Board Manager→pick version 2.0.17 and press Update

Can not use the ESP32C6 as it requires the latest ESP32 API from Espressif i.e. 3.0.7


Recommended board→ESP32 Dev Module

Code

In the Arduino IDE go to **Examples→DabbleESP32→01LedBrightnessControl**

Modify the setup() function as follows (change the name in Dabble.begin() to include your name....you don't want to confuse your "SSID" with someone else's). This code will run at the start to test your LED.

```
void setup() {  
  Serial.begin(115200);    // make sure your Serial is working  
  Dabble.begin("MyEsp32-K"); //set bluetooth name  
  pinMode(4,OUTPUT);  
  for(int x=0;x<3;x++)  
  {  
    digitalWrite(4,HIGH);  
    delay(250);  
    digitalWrite(4,LOW);  
    delay(250);  
  }  
}  
  
void loop() {  
  Dabble.processInput(); //this function  
  Serial.print("Led:");  
  Serial.print(LedControl.getpinNumber());  
  Serial.print('\t');  
  Serial.print("State:"); //0 means OFF  
  Serial.print(LedControl.getpinState());  
  Serial.print('\t');  
  Serial.print("Brightness:");  
  Serial.println(LedControl.readBrightness());  
}
```



Uncomment the code in the loop() function. This code is there to test the connectivity of the Dabble app to your ESP32. See the results in the Serial Monitor.

Circuit

Add in a resistor controlled LED connected to pin 4(or your selected pin) of your ESP32 Dev Module. You could also use any other available pin. Simply change the pin number referenced in the code.

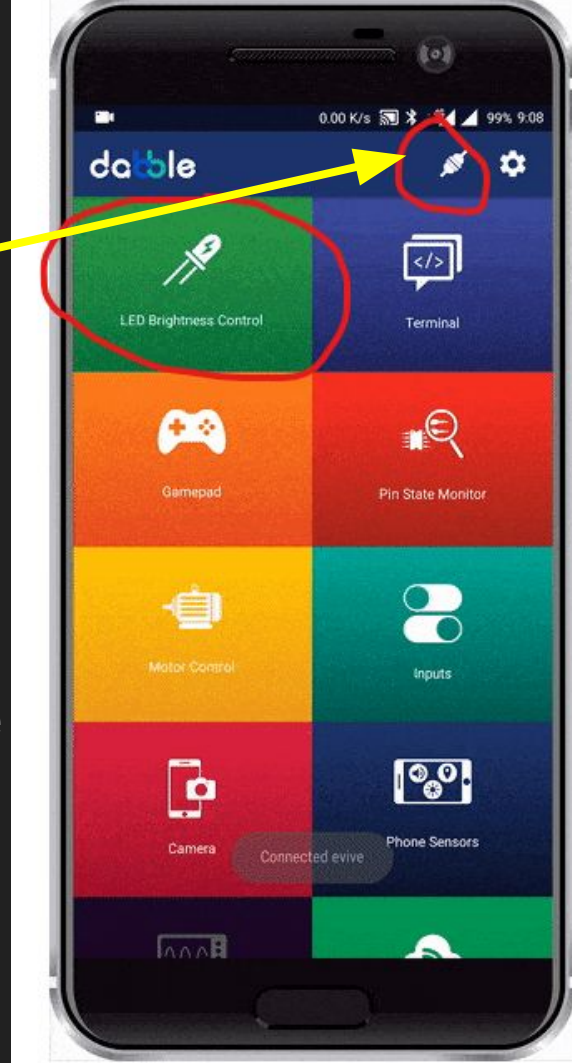
Dabble Control

Start the app.

Connect to your ESP32 Dev Module by clicking the connect icon (allow it to use your location while using the app).

Find and connect to your Bluetooth "SSID".

When connected, start the LED Brightness Control feature by clicking the button.



LED Brightness Control

Enter pin 4(or the pin you selected) and then click the on/off button to turn off/on your LED.

Turn the dial to control the LED brightness.

Open the Serial Monitor to view these changes as printed out via the code in the loop() function.



Terminal



The terminal allows you to send text messages to the ESP32.

Download the 02. Terminal example program.

Modify the Dabble.begin() function to include your name.

Note that the following line of code sends data from the Serial Monitor to the App:

```
Terminal.print(Serialdata);
```

 →you could also use this to send data via code/ESP32 to the App.

This code sends the data from the App to the Serial Monitor:

```
Serial.write(Terminal.read());
```

Terminal Code Modification

Add a new **String** variable called **terminalData**

```
String terminalData="";
```

Modify the last if block in the **loop()** function as follows:

The variable **terminalData** can now be used to determine what message was sent to the ESP32.

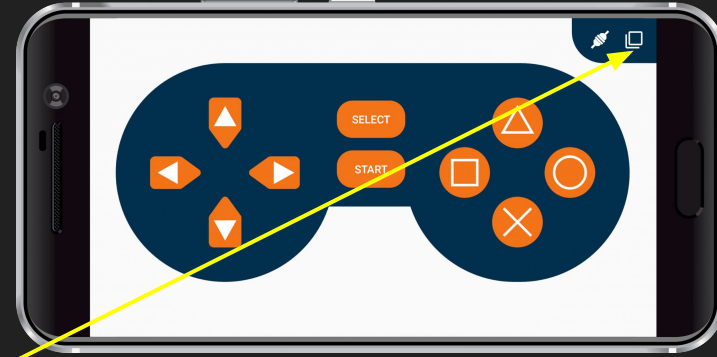
```
if (Terminal.available() != 0)
{
    while (Terminal.available() != 0)
    {
        char data=Terminal.read();
        terminalData+=data;
    }
    Serial.println(terminalData);
    terminalData="";
}
```

Gamepad



Save a copy of the 03.Gamepad example sketch.

Upload and test the Gamepad by opening the Serial monitor.



Once in the Gamepad screen, test the buttons by pressing on them and observing the Serial Monitor. Access the Joystick and Accelerometer Modes. Test the Radius, Angle and x and y axis values by moving the joystick and tilting your phone.

Questions and Exercises

1. What does the `LedControl.getPinNumber()` function return?

`LedControl.getPinNumber()` returns the GPIO pin number that was assigned to a specific function of the LED matrix (e.g., `dataIn`, `clk`, or `load`) during initialization. It tells you which physical pin is being used by the `LedControl` object.

2. What does the `LedControl.getPinState()` function return?

`LedControl.getPinState()` returns the current logical state (HIGH or LOW) of the pin associated with the LED matrix. It indicates whether the pin is currently outputting a high voltage (1) or low voltage (0).

3. What does the `LedControl.readBrightness()` function return?

`LedControl.readBrightness()` returns the current brightness level of the LED display. This value is typically an integer between 0 (dim) and 15 (brightest).

4. What is the purpose of the `Dabble.begin(string)` function?

`Dabble.begin("string")` initializes the `Dabble` library and starts Bluetooth communication using the given name as the Bluetooth device name (the string you provide). For example, `Dabble.begin("ESP32_Dabble")` would make your ESP32 appear with that name in the `Dabble` app.

5. Which variable in the "02. Terminal example program" code is used to store incoming data being sent from the `Dabble` app?

The variable is: `receivedData`

6. Create a circuit with four leds. Turn the leds off and on depending on the tilt of your cell phone. Use a breadboard for this part.

/15 TI

Use the purple ESP32DevKit PCB board. Insert your ESP 32 into the appropriate headers and then program the ESP32 to activate the leds when the gamepad buttons are pressed. When pressing a button i.e. Triangle, sound a piezo buzzer (pin 15). Turn off the buzzer when the button is released. The pins attached to the LEDs on the PCB board are 2,4, 32 and 33.

/15 TI