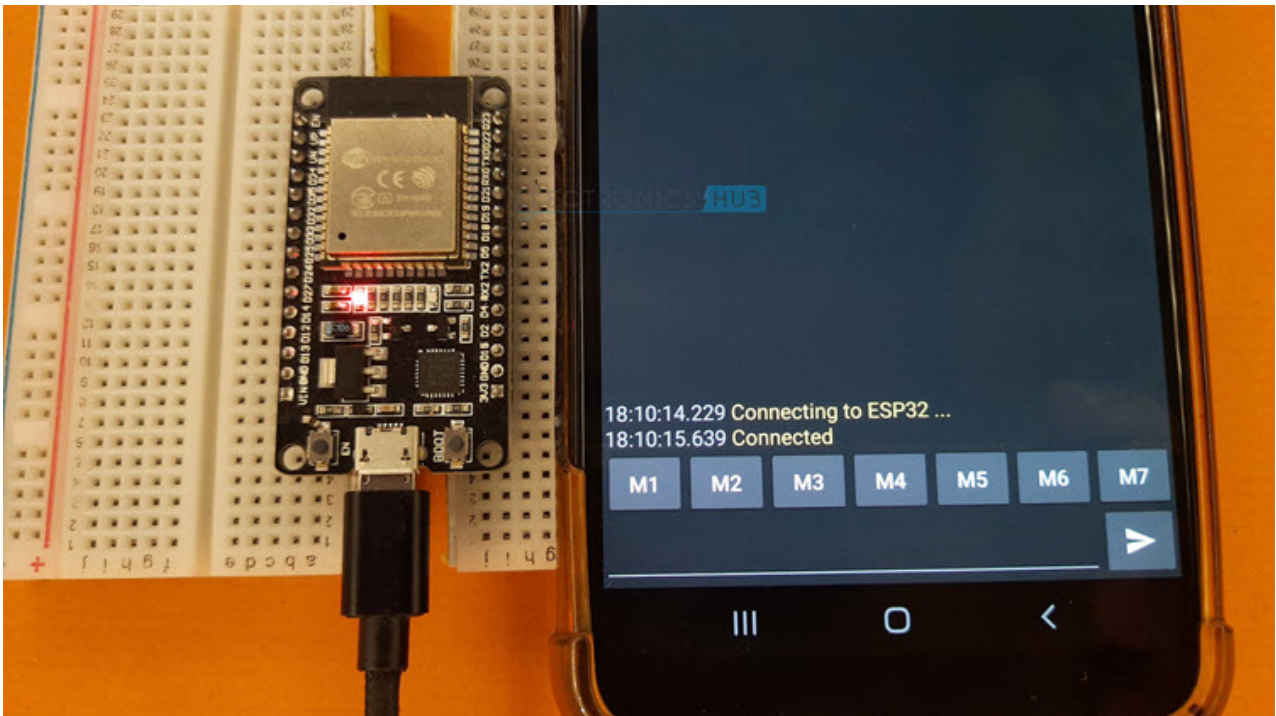


# A Beginner's Tutorial on ESP32 Bluetooth | Learn ESP32 Classic Bluetooth

[electronicshub.org/esp32-bluetooth-tutorial](https://electronicshub.org/esp32-bluetooth-tutorial)

In this tutorial, we will learn about the Bluetooth feature of ESP32. ESP32 supports both the Classic Bluetooth v4.2 as well as the Bluetooth Low Energy (BLE) standards. Let us focus on the Classic Bluetooth in this ESP32 Bluetooth Tutorial. We will learn a little bit about architecture of Bluetooth in ESP32, how to configure, setup and start Bluetooth communication and also a couple of simple projects involving data transfer between ESP32 and a smart phone over Bluetooth Communication.



## A Brief Note of ESP32 Bluetooth

Bluetooth is a great wireless communication technology that has been popular for quite few years. Operating in the unlicensed 2.4 GHz ISM (Industrial, Scientific and Medical) frequency band, Bluetooth is a short-range wireless communication technology with range up to 100 m.

ESP32 SoC integrates both the Bluetooth Link Controller (or Link Manager) and the Baseband into its silicon. Physically, only an external antenna is needed for proper Bluetooth Communication.

Since both Wi-Fi and Bluetooth operate at the same 2.4 GHz ISM frequency, the Wi-Fi Radio and the Bluetooth Radio share the same antenna in ESP32. If you take a look at the pinout of ESP32 SoC, there is only one pin for connecting to antenna (LNA\_IN).

ESP32 supports both the Classic Bluetooth (Classic BT) and Bluetooth Low Energy (BLE) which can be configured with BLUEDROID Bluetooth Stack. ESP32 Bluetooth supports three types of Host Controller Interface (HCI): UART, SPI and VHCI (Virtual HCI) interfaces (only one can be used at a time and UART is the default).

## Getting Started with ESP32 Classic Bluetooth

---

The Classic Bluetooth also known as Bluetooth Base Rate / Enhanced Data Rate, is the original point-to-point network topology designed for one-to-one wireless communication between a master and a slave. Even though multiple slave devices can be connected to a single master, only one slave can be actively communicating with the master. Our Bluetooth keyboards and mouse work with Classic Bluetooth technology. Another simple example is file transfer between two devices (like two mobile phone or a laptop and a mobile phone) over Bluetooth is based on Classic Bluetooth functionality.

BLE or Bluetooth Low Energy on the other hand, as the name suggests, is designed for low power operation and developed with IoT applications as the main target. Bluetooth Specification 4.0 added BLE functionality and is mainly used in battery operated devices like watches, audio devices, health trackers, fitness monitors and data beacons.

Let us make another tutorial on ESP32 BLE and focus on ESP32 Classic Bluetooth for now.

The BLUEDROID Bluetooth Stack communicates with Bluetooth Controller over VHCI (Virtual Host Controller Interface) and at the same time provides APIs for user application.

Bluetooth Profiles determine the functions of each layer of the Bluetooth from PHY to L2CAP while the Bluetooth Protocols define message formats and procedures for data transport, link control etc.

The following is a list of Classic Bluetooth Profiles and Protocols supported by BLUEDROID Bluetooth Stack of ESP32.

### Classic Bluetooth Profiles

---

- GAP
- A2DP (SNK)
- AVRCP (CT)

### Classic Bluetooth Protocols

---

- L2CAP
- SDP
- AVDTP
- AVCTP

The communication between ESP32's Processor and Bluetooth Controller is based on Serial Interface. Let us explore more about ESP32 Bluetooth by using the 'BluetoothSerial' library for Classic Bluetooth.

## ESP32 Classic Bluetooth Serial Communication

---

If you ever worked with Arduino and any Bluetooth device like HC-05, then you might remember that Arduino UNO and HC-05 communicate over Serial Communication. ESP32, which already has a Bluetooth Controller, also has a similar communication between the main Xtensa Processor and the Bluetooth Controller.

What this means is that after receiving data from a Bluetooth device wirelessly, the Bluetooth controller in ESP32 transfers this data to ESP32's Processor over serial communication. Similarly, in order to send data over Bluetooth, the Processor of ESP32 transmits data to the Bluetooth Controller using the serial interface.

We will use this information along with a dedicated 'BluetoothSerial' library to transmit and receive data.

The BluetoothSerial library works similar to the Serial library but it is just within ESP32. Some of the frequently used functions offered by BluetoothSerial library are:

- begin()
- available()
- write()
- read()

Let us write a simple code which transfers data between ESP32 and a Mobile Phone. To view the received data of ESP32, we will print the data on the serial port. Coming to the mobile phone, in order to send and receive data over Bluetooth, we have to use an application.

I tried many Bluetooth Serial Applications for Android but finally went with "Serial Bluetooth Terminal" by Kai Morich. You can download it from this link (or from Play Store).

### Code

---

First, let us see the code and in the process, we can understand the working. The code is very simple. Create an object of class 'BluetoothSerial' and begin the communication using 'begin()' function.

You can pass the name of the ESP32 Bluetooth Device as an argument to the 'begin()' function. If you leave it blank, then the default name i.e., ESP32 is used. Also initialize the normal serial communication with baud rate of 115200.

Then, in the loop function, read data from BluetoothSerial and print it on the Serial Monitor and read data from the Serial Monitor and write it to BluetoothSerial.

When we write data to BluetoothSerial, the Bluetooth Terminal App on the phone receives the data and prints it on the app. When you type data in the app and send it over Bluetooth, the BluetoothSerial will read this data and is printed on Serial Monitor.

```
#include "BluetoothSerial.h"

/* Check if Bluetooth configurations are enabled in the SDK */

/* If not, then you have to recompile the SDK */

#ifndef CONFIG_BT_ENABLED || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);

  /* If no name is given, default 'ESP32' is applied */

  /* If you want to give your own name to ESP32 Bluetooth device, then */

  /* specify the name as an argument SerialBT.begin("myESP32Bluetooth"); */

  SerialBT.begin();

  Serial.println("Bluetooth Started! Ready to pair...");
}

void loop() {
  if (Serial.available())
  {
    SerialBT.write(Serial.read());
  }

  if (SerialBT.available())
  {
    Serial.write(SerialBT.read());
  }
}
```

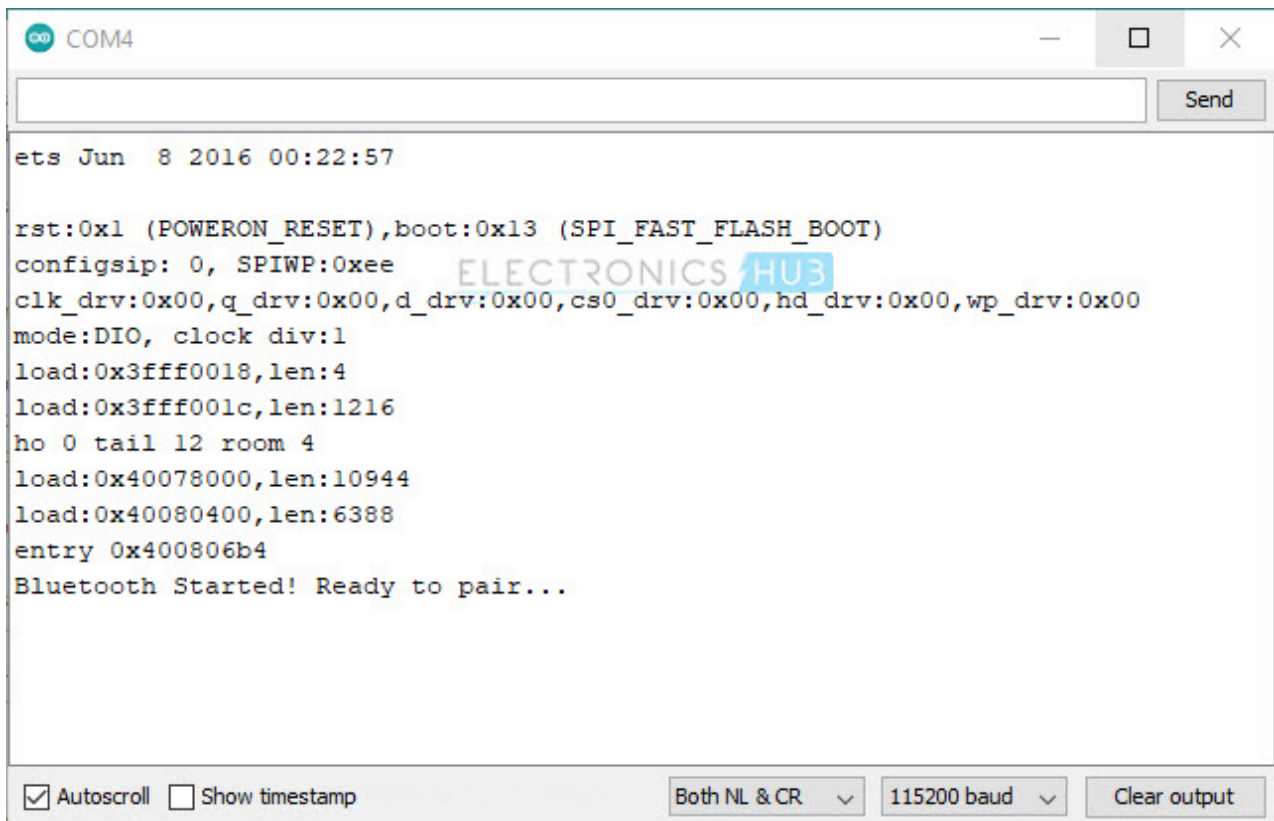
```
delay(20);
```

```
}
```

[view raw ESP32-Bluetooth-Classic-Serial.ino](#) hosted with ❤ by [GitHub](#)

## Uploading the Code and Testing

After uploading the code to ESP32, if you open the serial monitor of Arduino IDE, you can see the ESP32 printing some information about Bluetooth. Also, it displays the 'ready' message.



```
ets Jun 8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config:0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4
Bluetooth Started! Ready to pair...
```

Now, turn on Bluetooth in your smart phone and scan for Bluetooth devices. You should see a list of 'Paired devices' and 'Available devices' and from the available devices, select 'ESP32'.

Your mobile will ask if you want to pair with 'ESP32' and you select yes (or ok). There is no password. Now, open the 'Serial Bluetooth Terminal' app on your phone and click on the three horizontal bars on the top left corner of the screen.

## Bluetooth




Make sure the device you want to connect to is in pairing mode. Your phone (Ravi's-Galaxy) is currently visible to nearby devices.


ELECTRONICS HUB

Paired devices .....

 Myi20

Available devices .....

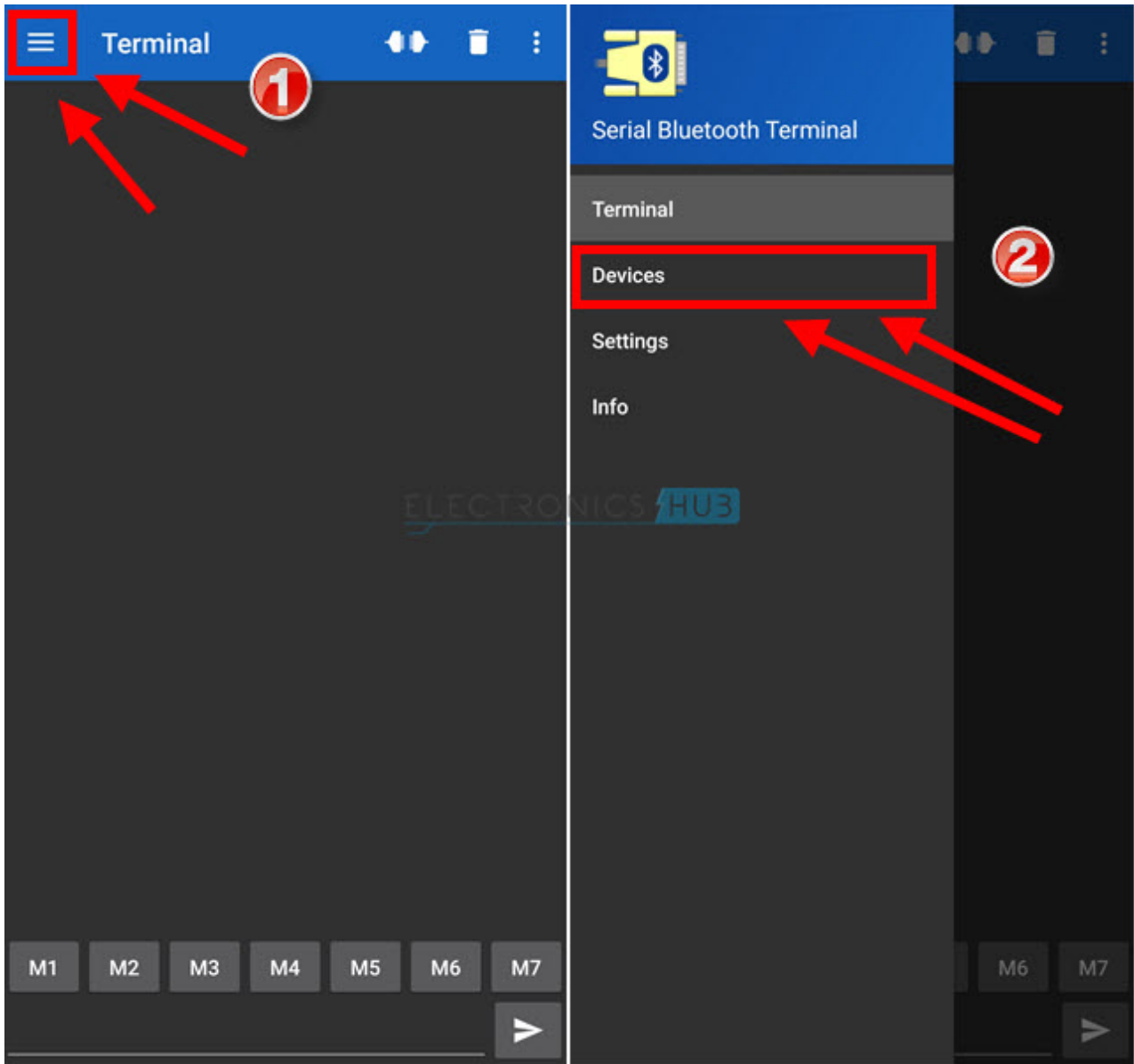
 ESP32

 TV

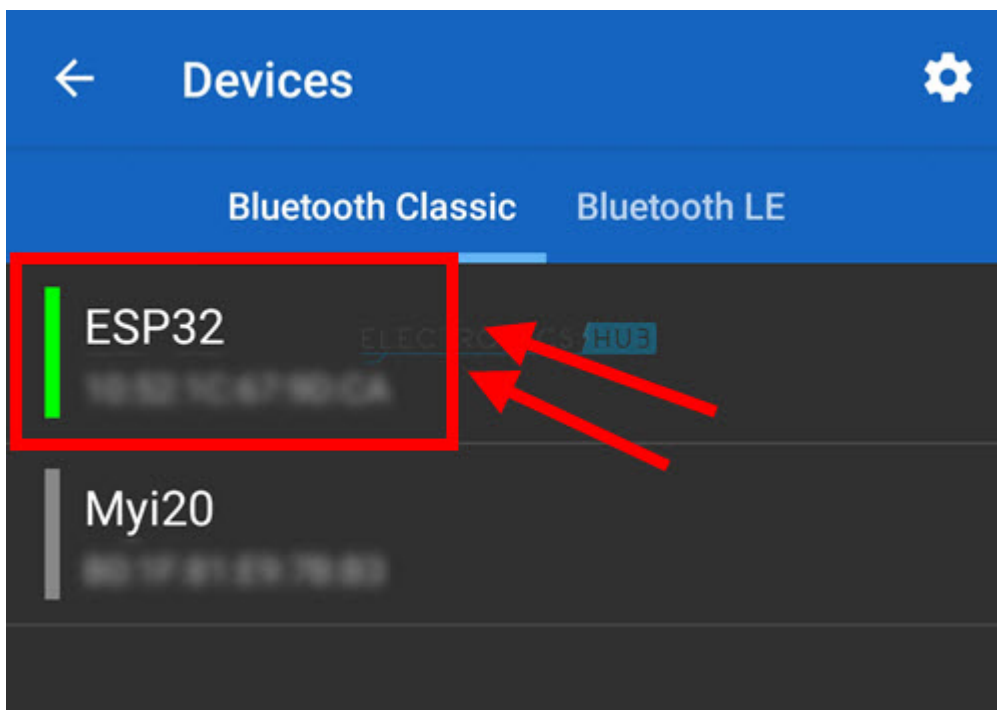
Stop

Done

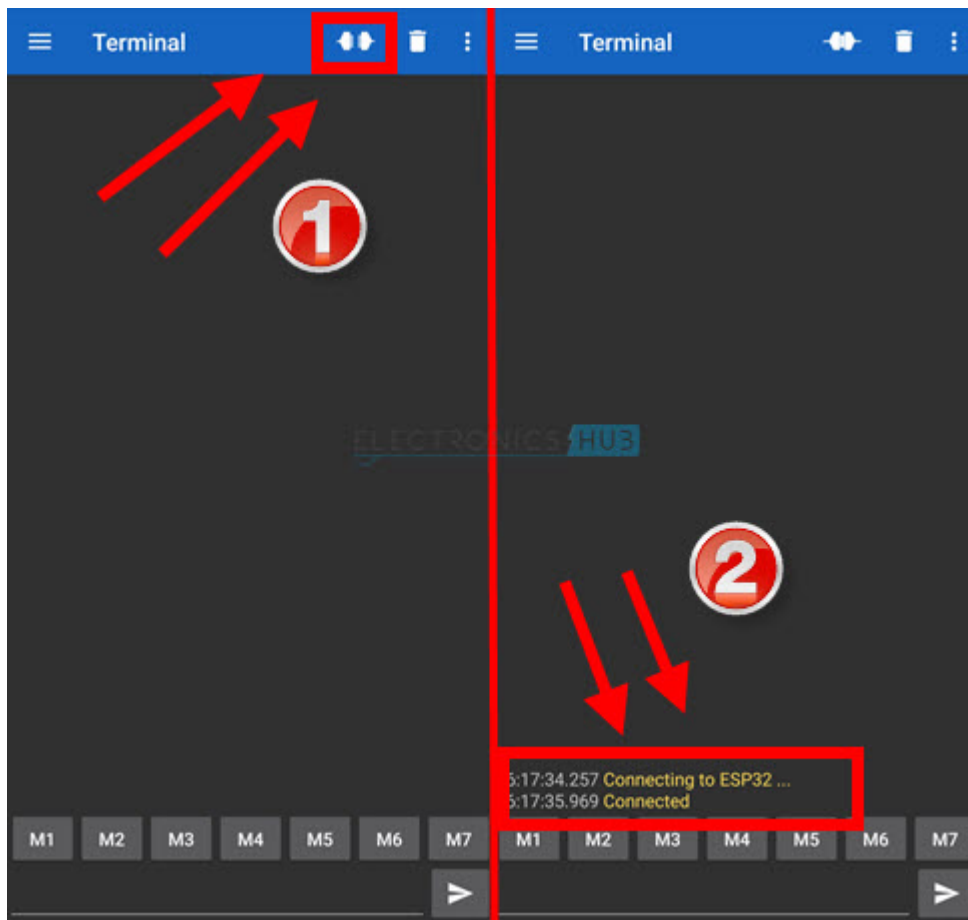




Select 'Devices' tab and select ESP32 from the list.



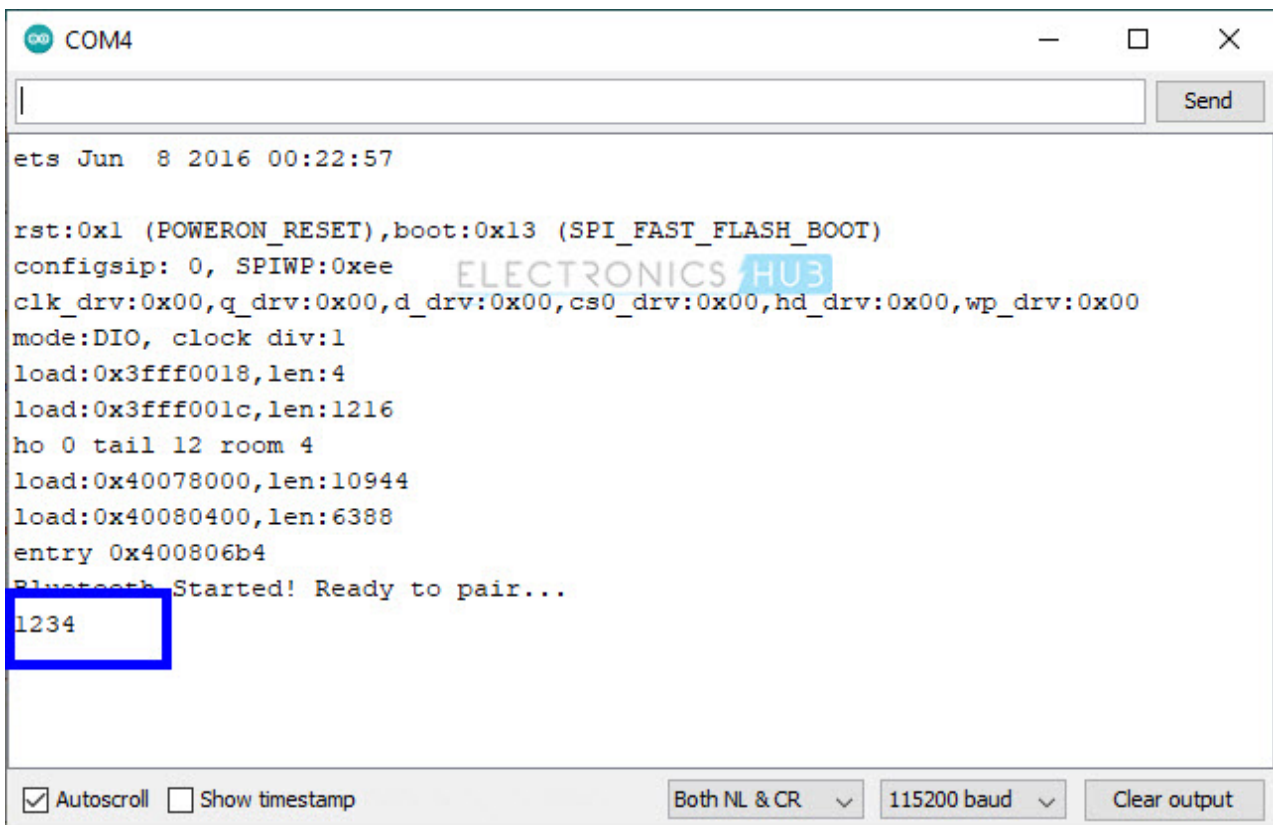
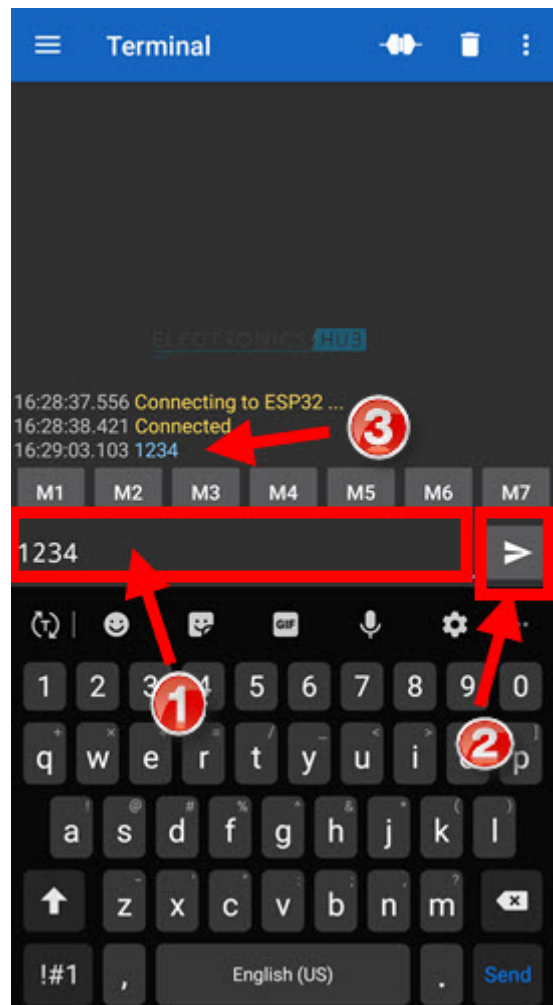
Now, click on the 'link' icon on the top to connect to ESP32 Bluetooth Device. The app will display the status as 'Connecting to ESP32 ...' while making connection and if the connection is successful, it will display 'Connected'.



Below is a space for entering data to transfer over Bluetooth. Type something and click on the send button. The sent data is echoed back on the app. This data is sent to ESP32 over Bluetooth and is received by BluetoothSerial read() function.

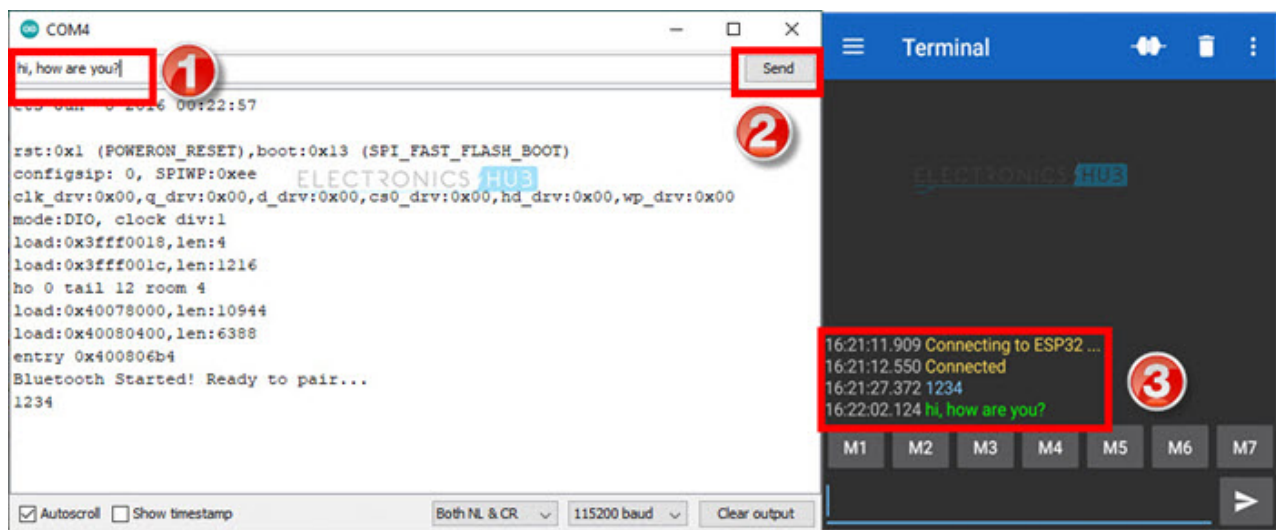
Since we are passing this information to Serial port, you can see the data printed on the serial monitor.





Similarly, you can send data from ESP32 to Mobile Phone. Just type some data in the serial monitor and click on send. This data is sent over Bluetooth to Mobile Phone through the BluetoothSerial write() function.

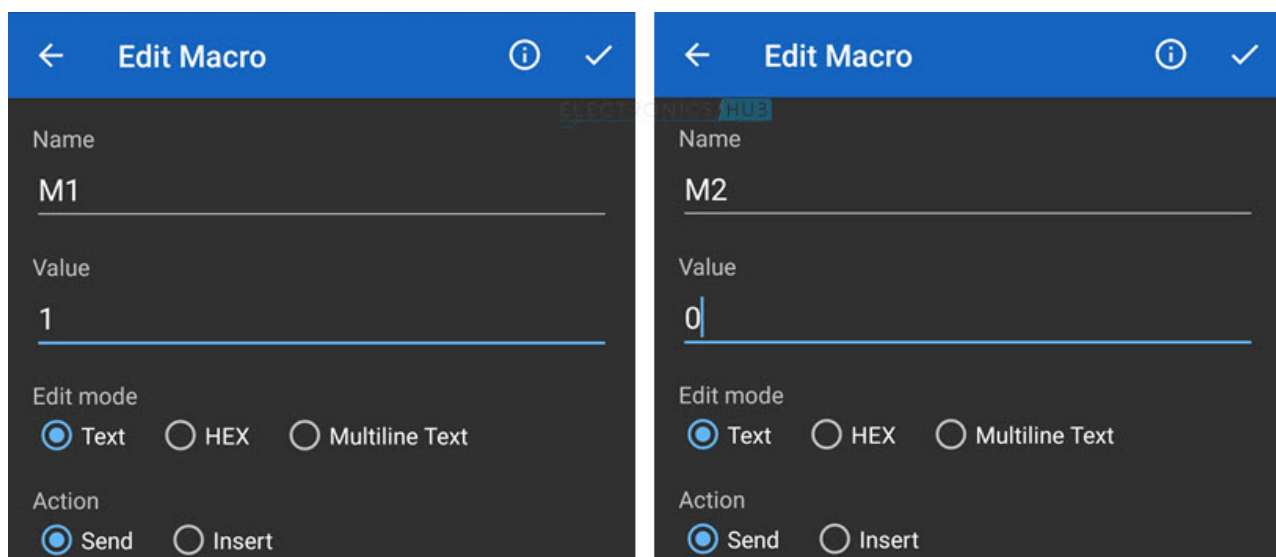
The serial Bluetooth terminal app will read this data and prints it on the app.



## Bluetooth Controlled LED using ESP32

Using the above application, we can modify the code slightly and implement a Bluetooth Controlled LED using ESP32. The aim of this project is to see how easy it is to control GPIO Pins of ESP32 by sending and interpreting the data from Bluetooth.

To keep things simple, let us transmit '1' and '0' from the Mobile Phone App using the macro keys. I assigned '1' for M1 and '0' for M2. You can compare the received data with characters '1' and '0' or their decimal equivalent in ASCII i.e., 49 and 48.



When '1' is received, the LED connected to GPIO 2 will turn ON and if '0' is received, the LED is turned OFF.

Obviously, the LED is just a representation of the GPIO Pin being ON and OFF. You can further improve this application into a Bluetooth Controlled Relay using ESP32.

## Code

```
#include <BluetoothSerial.h>

#define ledPIN 2

BluetoothSerial SerialBT;

byte BTData;

/* Check if Bluetooth configurations are enabled in the SDK */
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

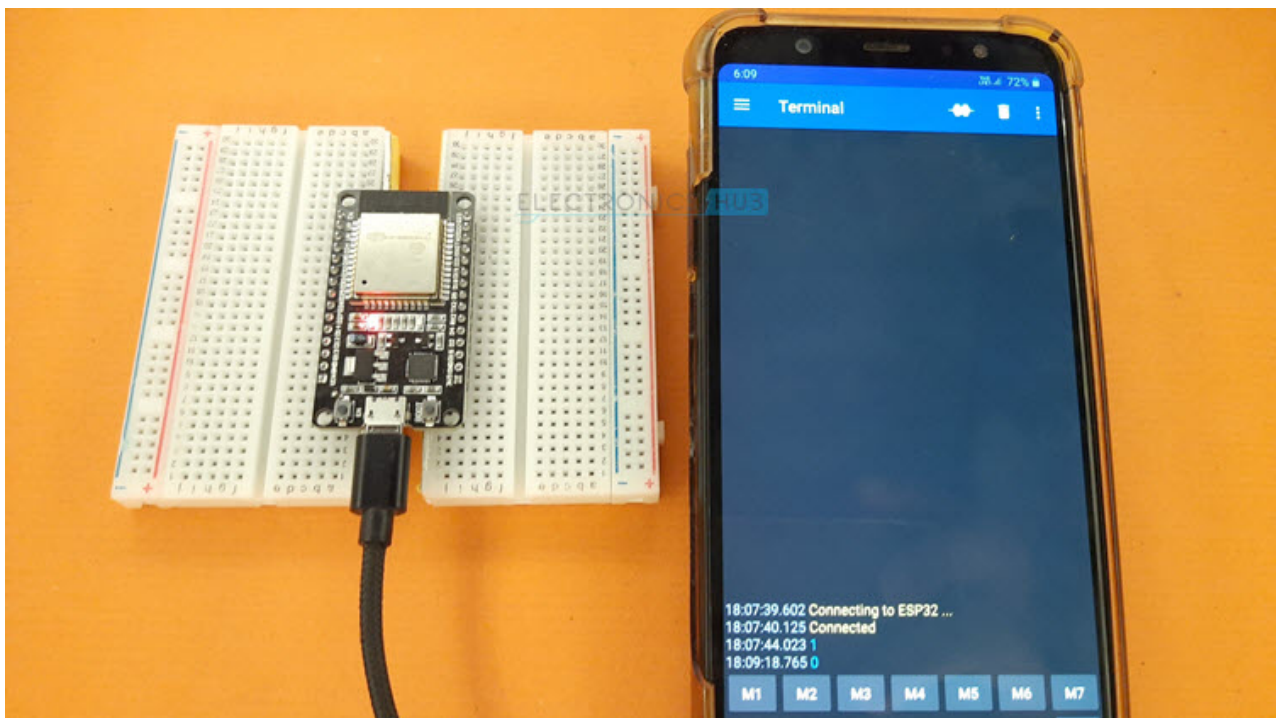
void setup()
{
  pinMode(ledPIN, OUTPUT);
  Serial.begin(115200);
  SerialBT.begin();
  Serial.println("Bluetooth Started! Ready to pair...");
}

void loop()
{
  if(SerialBT.available())
  {
    BTData = SerialBT.read();
    Serial.write(BTData);
  }

  /* If received Character is 1, then turn ON the LED */
  /* You can also compare the received data with decimal equivalent */
  /* 48 for 0 and 49 for 1 */
```

```
/* if(BTData == 48) or if(BTData == 49) */  
if(BTData == '1')  
{  
  digitalWrite(ledPIN, HIGH);  
}  
  
/* If received Character is 0, then turn OFF the LED */  
if(BTData == '0')  
{  
  digitalWrite(ledPIN, LOW);  
}  
}
```

[view raw Bluetooth-Controlled-LED-ESP32.ino](#) hosted with ❤ by [GitHub](#)



## Conclusion

A complete beginner's guide on ESP32 Bluetooth Communication. You learned some important basics of Bluetooth Communication in ESP32 SoC, how to setup Classic Bluetooth in ESP32, transfer data from a smart phone to ESP32 using Bluetooth and an extension project called Bluetooth Controlled LED using ESP32 (which can be easily modified to control a Relay).

## One Response

---

## Leave a Reply

---

Your email address will not be published. Required fields are marked \*