**GETTING STARTED WITH ESP32 and ESP-IDF || BLUETOOTH DATA ON OLED DISPLAY**

In this tutorial we will discuss about how to display the data which send from Bluetooth on OLED with ESP-IDF using ESP32. In this tutorial we’ll learn about display the BT data on OLED. So, let’s started and try to understand in much better way...

As we know that we will be using the ESP32 WROOM-32 Dev-kit, for this project which is shown in the pic below.

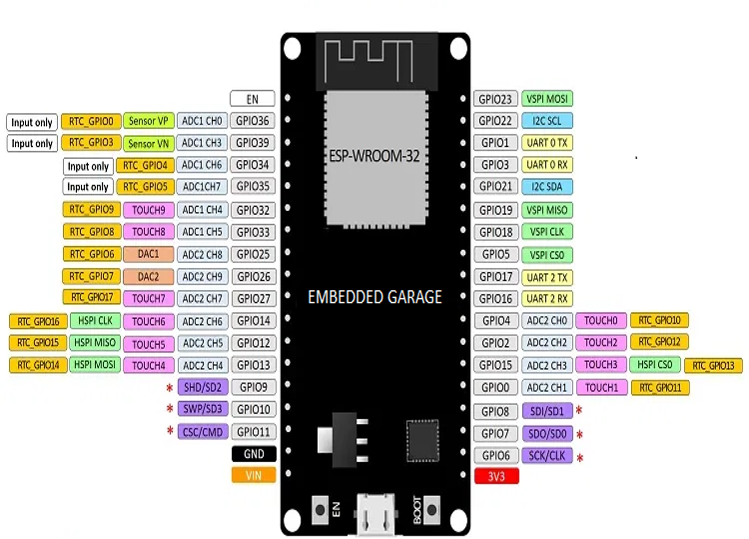


Fig.no 1

**CREATE THE PROJECT FIRST**

Note: we are using VS code for programming and we also installed in the ESP-IDF extensions in VS code. So, its depend on you that which platform you like most VS code or Espressif-IDE. Then after we need to...

Create a new Project where you set the ESP\_IDF path. Then give some name of the project, then open the project in VS Code.

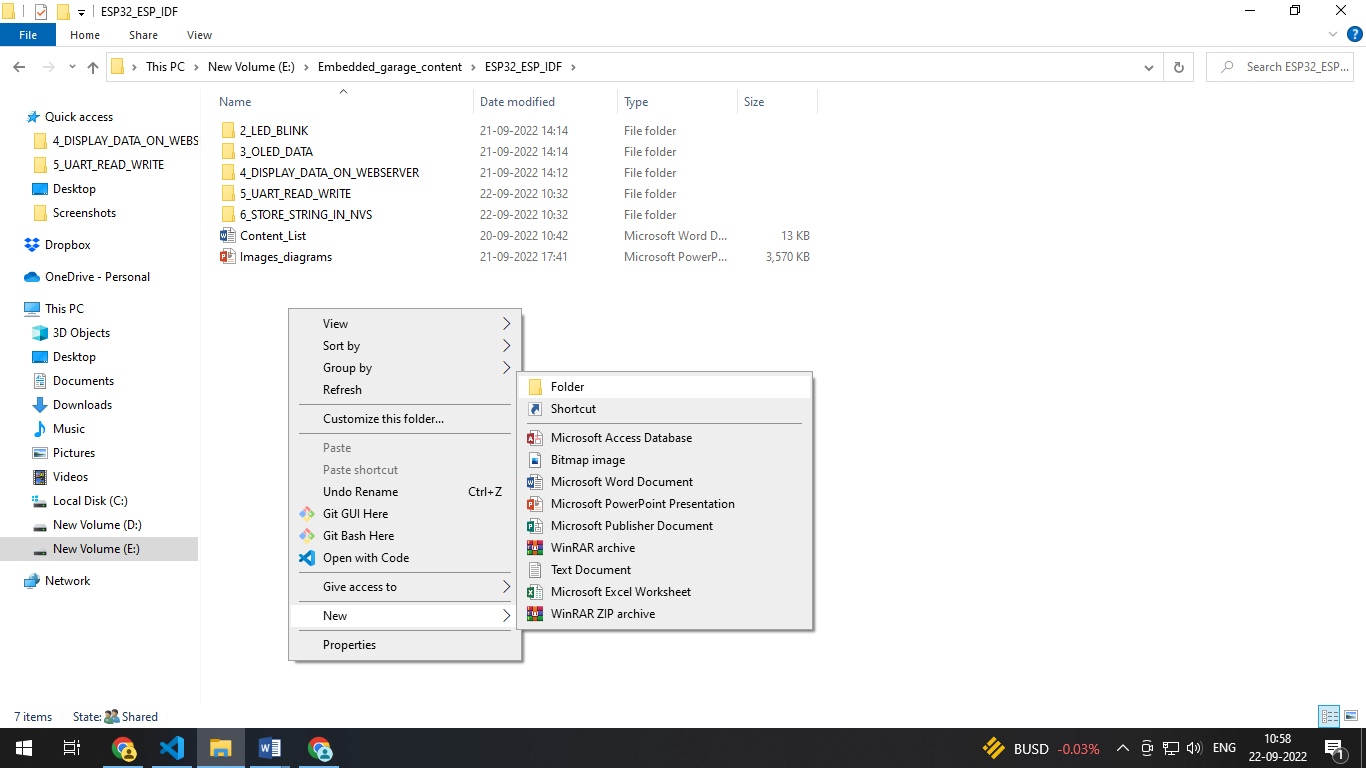


Fig.no 2

After creating the project firstly, we need to write the program of how to display the Bluetooth data on OLED screen. So, we need to define the libraries which are important in the code/programs.

So, lets write the code....

**THE CODE**

Firstly, we adding the Libraries which we used in the code:

#include <stdio.h>

#include "esp\_log.h"

#include "driver/i2c.h"

#include "sdkconfig.h"

#include "OLEDDisplay.h"

#include <stdint.h>

#include <string.h>

#include <stdbool.h>

#include "nvs.h"

#include "nvs\_flash.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "esp\_bt.h"

#include "esp\_bt\_main.h"

#include "esp\_gap\_bt\_api.h"

#include "esp\_bt\_device.h"

#include "esp\_spp\_api.h"

#include "driver/gpio.h"

#include "time.h"

#include "sys/time.h"

Then define the tag.

static const char \*tag = "oled\_Bluetooth-example";

Now define the OLED functionality

#define \_I2C\_NUMBER(num) I2C\_NUM\_##num

#define I2C\_NUMBER(num) \_I2C\_NUMBER(num)

#define I2C\_MASTER\_SCL\_IO 22               /\*!< gpio number for I2C master clock \*/

#define I2C\_MASTER\_SDA\_IO 21               /\*!< gpio number for I2C master data  \*/

#define I2C\_MASTER\_NUM I2C\_NUMBER(CONFIG\_I2C\_MASTER\_PORT\_NUM) /\*!< I2C port number for master dev \*/

SemaphoreHandle\_t print\_mux = NULL;

OLEDDisplay\_t \*oled;

Define the Bluetooth Functionality and some variable

#define SPP\_SERVER\_NAME "SPP\_SERVER"

#define EXAMPLE\_DEVICE\_NAME "ESP32\_BT"

#define SPP\_SHOW\_DATA 0

#define SPP\_SHOW\_SPEED 1

#define SPP\_SHOW\_MODE SPP\_SHOW\_DATA /\*Choose show mode: show data or speed\*/

static const esp\_spp\_mode\_t esp\_spp\_mode = ESP\_SPP\_MODE\_CB;

static struct timeval time\_new, time\_old;

static long data\_num = 0;

static const esp\_spp\_sec\_t sec\_mask = ESP\_SPP\_SEC\_AUTHENTICATE;

static const esp\_spp\_role\_t role\_slave = ESP\_SPP\_ROLE\_SLAVE;

char value;

char bt\_data[10] = "";

int ii;

Now create a function esp\_spp\_cb and put the event in switch case and initialize them.

static void esp\_spp\_cb(esp\_spp\_cb\_event\_t event, esp\_spp\_cb\_param\_t \*param, void \*arg)

{

    // OLEDDisplay\_t \*oled = OLEDDisplay\_init(I2C\_MASTER\_NUM,0x78,I2C\_MASTER\_SDA\_IO,I2C\_MASTER\_SCL\_IO);

    switch (event)

    {

    case ESP\_SPP\_INIT\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_INIT\_EVT");

        esp\_spp\_start\_srv(sec\_mask, role\_slave, 0, SPP\_SERVER\_NAME);

        break;

    case ESP\_SPP\_DISCOVERY\_COMP\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_DISCOVERY\_COMP\_EVT");

        break;

    case ESP\_SPP\_OPEN\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_OPEN\_EVT");

        break;

    case ESP\_SPP\_CLOSE\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_CLOSE\_EVT");

        break;

    case ESP\_SPP\_START\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_START\_EVT");

        esp\_bt\_dev\_set\_device\_name(EXAMPLE\_DEVICE\_NAME);

        esp\_bt\_gap\_set\_scan\_mode(ESP\_BT\_CONNECTABLE, ESP\_BT\_GENERAL\_DISCOVERABLE);

        break;

    case ESP\_SPP\_CL\_INIT\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_CL\_INIT\_EVT");

        break;

    case ESP\_SPP\_DATA\_IND\_EVT:

     #if (SPP\_SHOW\_MODE == SPP\_SHOW\_DATA)

        ESP\_LOGI(tag, "ESP\_SPP\_DATA\_IND\_EVT len=%d handle=%d",

                 param->data\_ind.len, param->data\_ind.handle);

After that we store the data which comes from the Bluetooth in **“value”** variable and display them on **OLED display**.

if (param->data\_ind.len < 1023) {

        esp\_log\_buffer\_hex("", param->data\_ind.data, param->data\_ind.len);

        printf("received length: %d", param->data\_ind.len);

        printf("Received info: %s\n", param->data\_ind.data);

        }

        for (ii = 0; ii < (param->data\_ind.len)-2; ii++)

        {

           value = param->data\_ind.data[ii];

           bt\_data[ii] = value;

        }

    bt\_data[ii] = '\0';

    OLEDDisplay\_clear(oled);

    OLEDDisplay\_setTextAlignment(oled,TEXT\_ALIGN\_CENTER);

    OLEDDisplay\_setFont(oled,ArialMT\_Plain\_24);

    OLEDDisplay\_drawString(oled,64,20,bt\_data);

    OLEDDisplay\_display(oled);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);

    // }

    printf("\n");

    esp\_spp\_write(param->data\_ind.handle, param->data\_ind.len, param->data\_ind.data);

#else

        gettimeofday(&time\_new, NULL);

        data\_num += param->data\_ind.len;

        if (time\_new.tv\_sec - time\_old.tv\_sec >= 3)

        {

            print\_speed();

        }

#endif

        break;

    case ESP\_SPP\_CONG\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_CONG\_EVT");

        break;

    case ESP\_SPP\_WRITE\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_WRITE\_EVT");

        break;

    case ESP\_SPP\_SRV\_OPEN\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_SRV\_OPEN\_EVT");

        gettimeofday(&time\_old, NULL);

        break;

    case ESP\_SPP\_SRV\_STOP\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_SRV\_STOP\_EVT");

        break;

    case ESP\_SPP\_UNINIT\_EVT:

        ESP\_LOGI(tag, "ESP\_SPP\_UNINIT\_EVT");

        break;

    default:

        break;

    }

}

Then after create a esp\_bt\_gap\_cb function and put the event in the switch case and authenticate the Bluetooth functionality.

void esp\_bt\_gap\_cb(esp\_bt\_gap\_cb\_event\_t event, esp\_bt\_gap\_cb\_param\_t \*param)

{

    switch (event)

    {

    case ESP\_BT\_GAP\_AUTH\_CMPL\_EVT:

    {

        if (param->auth\_cmpl.stat == ESP\_BT\_STATUS\_SUCCESS)

        {

            ESP\_LOGI(tag, "authentication success: %s", param->auth\_cmpl.device\_name);

            esp\_log\_buffer\_hex(tag, param->auth\_cmpl.bda, ESP\_BD\_ADDR\_LEN);

        }

        else

        {

            ESP\_LOGE(tag, "authentication failed, status:%d", param->auth\_cmpl.stat);

        }

        break;

    }

    case ESP\_BT\_GAP\_PIN\_REQ\_EVT:

    {

        ESP\_LOGI(tag, "ESP\_BT\_GAP\_PIN\_REQ\_EVT min\_16\_digit:%d", param->pin\_req.min\_16\_digit);

        if (param->pin\_req.min\_16\_digit)

        {

            ESP\_LOGI(tag, "Input pin code: 0000 0000 0000 0000");

            esp\_bt\_pin\_code\_t pin\_code = {0};

            esp\_bt\_gap\_pin\_reply(param->pin\_req.bda, true, 16, pin\_code);

        }

        else

        {

            ESP\_LOGI(tag, "Input pin code: 1234");

            esp\_bt\_pin\_code\_t pin\_code;

            pin\_code[0] = '1';

            pin\_code[1] = '2';

            pin\_code[2] = '3';

            pin\_code[3] = '4';

            esp\_bt\_gap\_pin\_reply(param->pin\_req.bda, true, 4, pin\_code);

        }

        break;

    }

#if (CONFIG\_BT\_SSP\_ENABLED == true)

    case ESP\_BT\_GAP\_CFM\_REQ\_EVT:

        ESP\_LOGI(tag, "ESP\_BT\_GAP\_CFM\_REQ\_EVT Please compare the numeric value: %d", param->cfm\_req.num\_val);

        esp\_bt\_gap\_ssp\_confirm\_reply(param->cfm\_req.bda, true);

        break;

    case ESP\_BT\_GAP\_KEY\_NOTIF\_EVT:

        ESP\_LOGI(tag, "ESP\_BT\_GAP\_KEY\_NOTIF\_EVT passkey:%d", param->key\_notif.passkey);

        break;

    case ESP\_BT\_GAP\_KEY\_REQ\_EVT:

        ESP\_LOGI(tag, "ESP\_BT\_GAP\_KEY\_REQ\_EVT Please enter passkey!");

        break;

#endif

    case ESP\_BT\_GAP\_MODE\_CHG\_EVT:

        ESP\_LOGI(tag, "ESP\_BT\_GAP\_MODE\_CHG\_EVT mode:%d", param->mode\_chg.mode);

        break;

    default:

    {

        ESP\_LOGI(tag, "event: %d", event);

        break;

    }

    }

    return;

}

Now come to the main function, in the main function initialize the OLED functionality and finally we can display the BT data on OLED.

void app\_main(void)

{

    print\_mux = xSemaphoreCreateMutex();

    ESP\_LOGI(tag,"Running");

    oled = OLEDDisplay\_init(I2C\_MASTER\_NUM,0x78,I2C\_MASTER\_SDA\_IO,I2C\_MASTER\_SCL\_IO);

    // xTaskCreate(i2c\_test\_task, "i2c\_test\_task\_0", 1024 \* 2, (void \*)0, 10, NULL);

    esp\_err\_t ret = nvs\_flash\_init();

    if (ret == ESP\_ERR\_NVS\_NO\_FREE\_PAGES || ret == ESP\_ERR\_NVS\_NEW\_VERSION\_FOUND)

    {

        ESP\_ERROR\_CHECK(nvs\_flash\_erase());

        ret = nvs\_flash\_init();

    }

    ESP\_ERROR\_CHECK(ret);

    ESP\_ERROR\_CHECK(esp\_bt\_controller\_mem\_release(ESP\_BT\_MODE\_BLE));

    esp\_bt\_controller\_config\_t bt\_cfg = BT\_CONTROLLER\_INIT\_CONFIG\_DEFAULT();

    if ((ret = esp\_bt\_controller\_init(&bt\_cfg)) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s initialize controller failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

    if ((ret = esp\_bt\_controller\_enable(ESP\_BT\_MODE\_CLASSIC\_BT)) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s enable controller failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

    if ((ret = esp\_bluedroid\_init()) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s initialize bluedroid failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

    if ((ret = esp\_bluedroid\_enable()) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s enable bluedroid failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

    if ((ret = esp\_bt\_gap\_register\_callback(esp\_bt\_gap\_cb)) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s gap register failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

    if ((ret = esp\_spp\_register\_callback(esp\_spp\_cb)) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s spp register failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

    if ((ret = esp\_spp\_init(esp\_spp\_mode)) != ESP\_OK)

    {

        ESP\_LOGE(tag, "%s spp init failed: %s\n", \_\_func\_\_, esp\_err\_to\_name(ret));

        return;

    }

#if (CONFIG\_BT\_SSP\_ENABLED == true)

    /\* Set default parameters for Secure Simple Pairing \*/

    esp\_bt\_sp\_param\_t param\_type = ESP\_BT\_SP\_IOCAP\_MODE;

    esp\_bt\_io\_cap\_t iocap = ESP\_BT\_IO\_CAP\_IO;

    esp\_bt\_gap\_set\_security\_param(param\_type, &iocap, sizeof(uint8\_t));

#endif

    /\*

     \* Set default parameters for Legacy Pairing

     \* Use variable pin, input pin code when pairing

     \*/

    esp\_bt\_pin\_type\_t pin\_type = ESP\_BT\_PIN\_TYPE\_VARIABLE;

    esp\_bt\_pin\_code\_t pin\_code;

    esp\_bt\_gap\_set\_pin(pin\_type, 0, pin\_code);

}

**RESULT**

Finally, after written the code need to follow some important steps.

* Install the Serial Bluetooth Application.
* Build the code first.
* After successful build up need to select correct port.
* Then, choose the flash method as (UART).
* And finally flash the code to the ESP32.
* See the output (result).

Install the Serial Bluetooth Application.

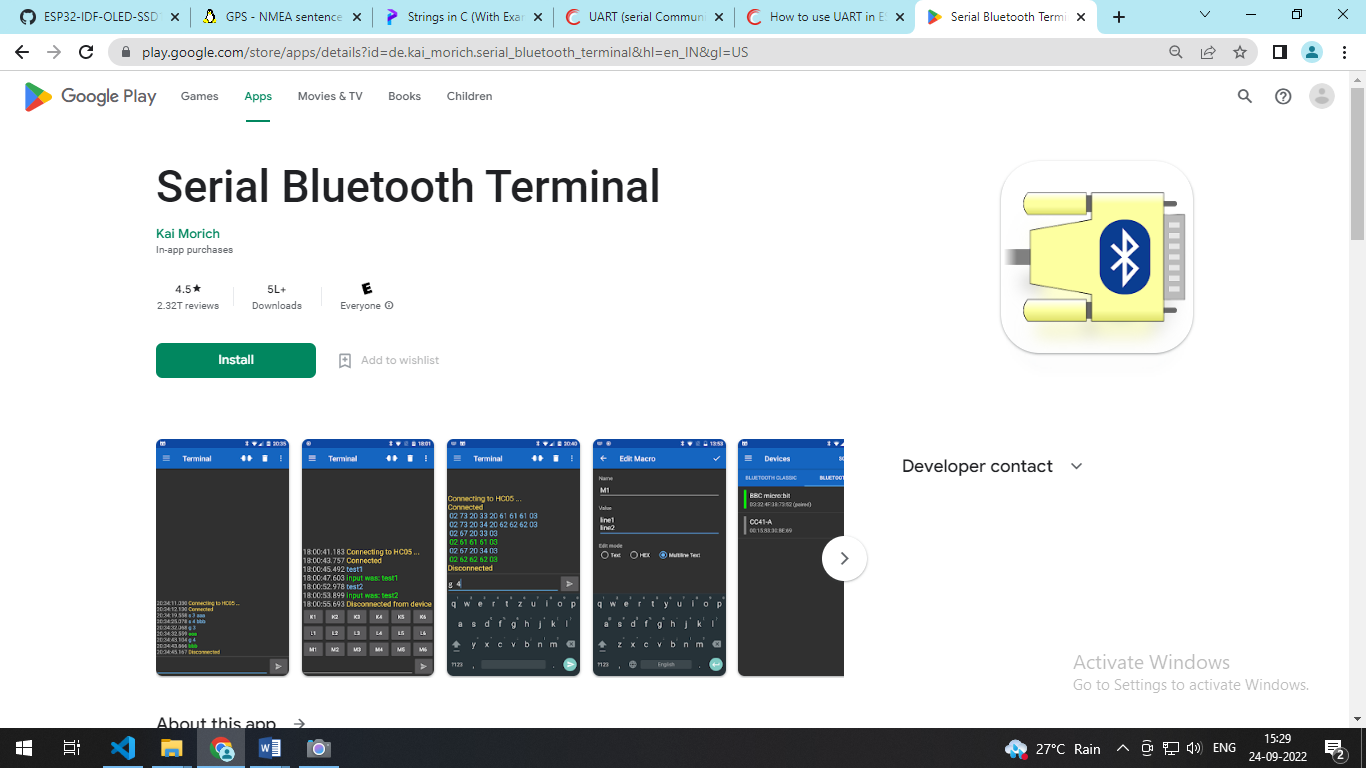


Fig.no 3

Here, my device name is **“ESP32\_BT” as you can see below:**

#define EXAMPLE\_DEVICE\_NAME "ESP32\_BT"

Now you can see in the pic below the Bluetooth is initialized and start means the device (esp32) is ready to pair with Bluetooth. So in your mobile turn on the Bluetooth (open the BT) and search the device whatever the name which you put in the code as your device name and pair with that.

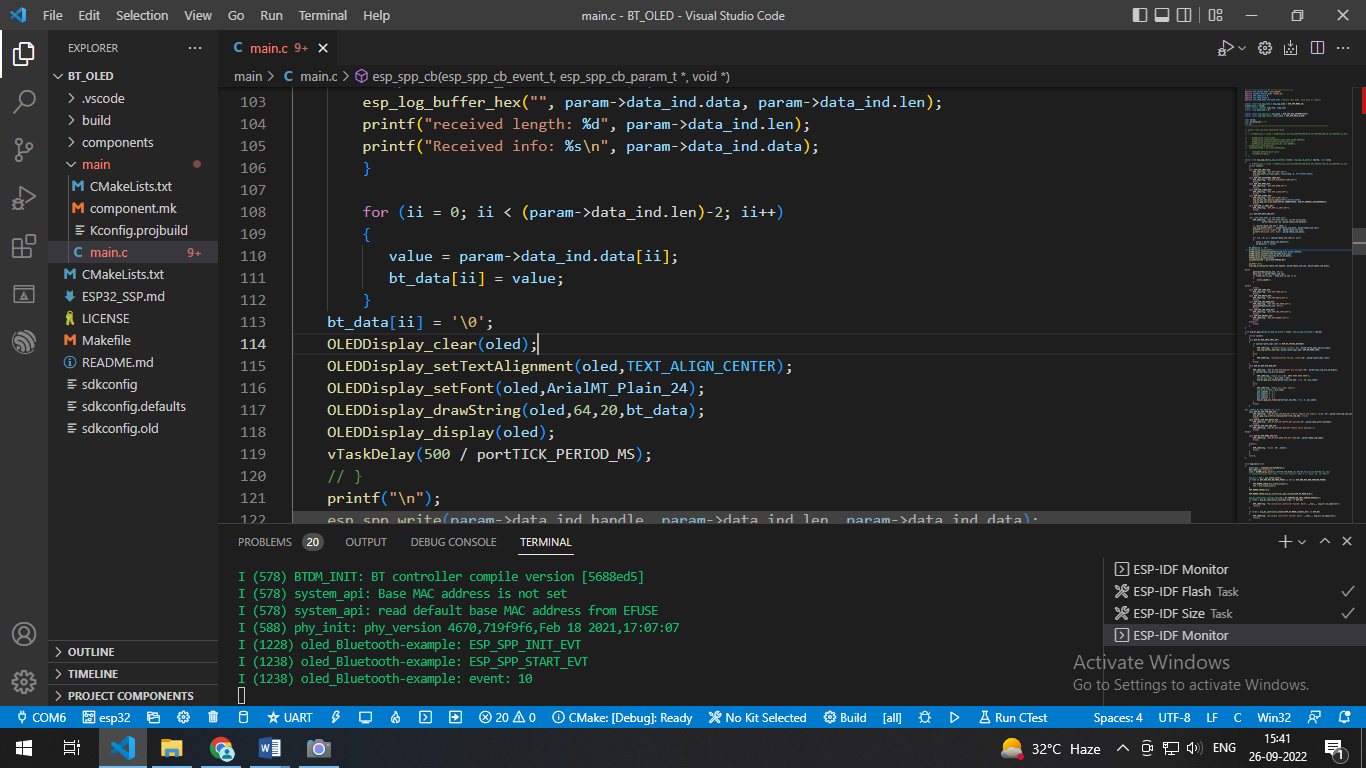


Fig.no 4

Here you can see that when the esp32 is connected then we received a msg.

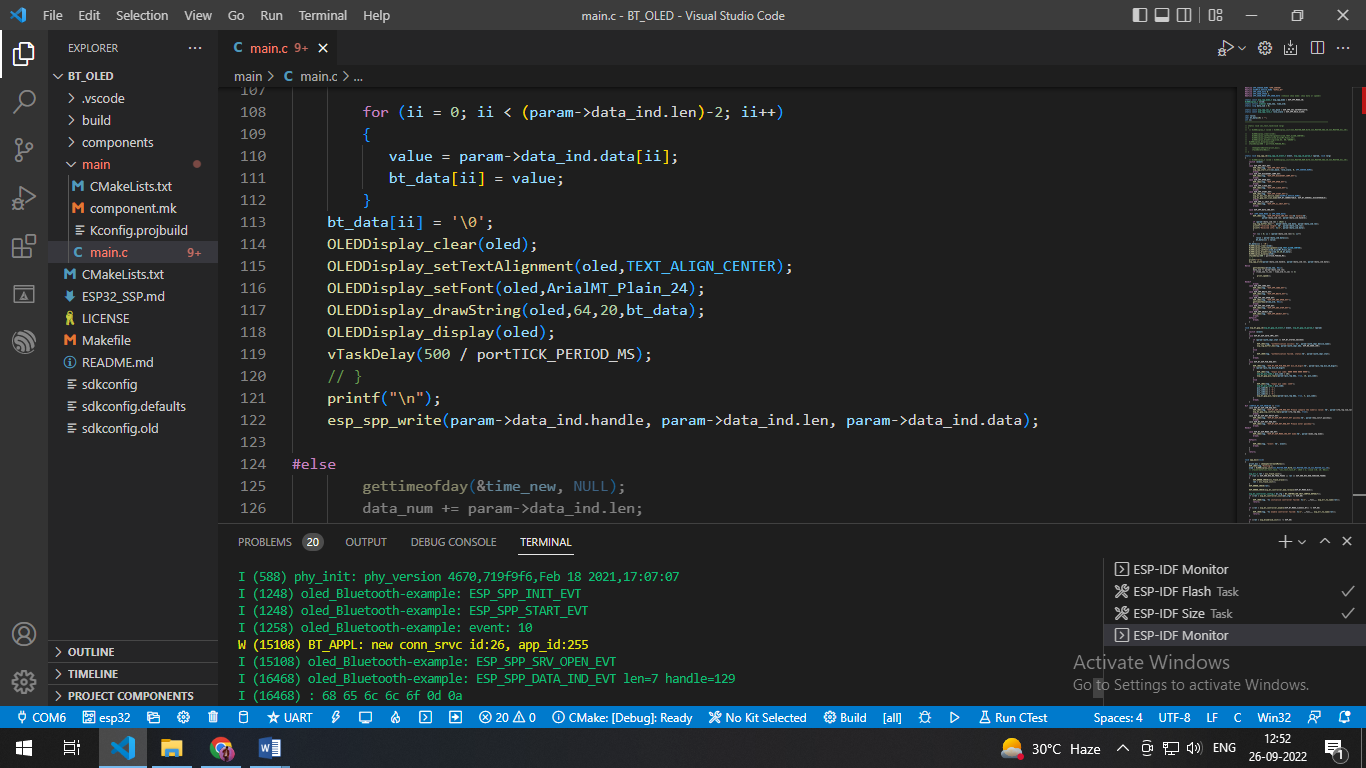


Fig.no 5

Now here you can see the result in the pic below (pic1, pic2), if we sending any data from Bluetooth then the same data is received and display into the OLED screen.



Fig.no 6



Fig.no 7

**Hope you like this tutorial! Thank you!**