ESP32-M1 Reach Out Internet Communication Setup and Test With NAT

Perth, Western Australia

`

Document history

Version	Reason for Revision	Date
1.0	New document	27-04-2021

Contents

Acronyms

AP – Access Point

NAT – Network Address Translation

IP – Internet Protocol

IoT – Internet of Things

RF – Radio Frequency

RFFE – RF Front End

RX – Receiver

STA – Station

TX - Transmitter

1. Introduction

1.1. Objective

This document serves as a setup guide for ESP32-M1 board to connect to a router to access the internet services. This document explains the codes in general, the steps to put the board into AP/STA mode, connect devices to ESP32-M1 board and access the internet. The documentation also shows the steps to connect a ESP32-M1 as a repeater to extend the range for internet access. The speed test performed on ESP32-M1 as AP and as Repeater.

1.2. Background

The ESP32-M1 board is an IoT device, at the broader spectrum of the board primary application is for remote monitoring and sensing, drone radios, long-range video streaming, mesh networking and many other IoT applications, as a secondary application the board also able to support internet access which helps in the situation where accessing the internet is an issue. The ESP32-M1 connect to the internet with the NAT implementation. The NAT can translate the source IP addresses in a private network into a single IP address to reach the desired destination.

1.3. Related Documents

Please refer to below document for technical guide for ESP32-M1,

https://github.com/bisonscience/ESP32-M1-Reach-Out/blob/main/Datasheet/BS%20ESP32-M1.pdf

and ESP32-M1 Datasheet

https://github.com/bisonscience/ESP32-M1-Reach-Out/blob/main/Datasheet/BS%20ESP32-M1.pdf

1.4. Version

The code used in this document is referred to API release v3.3.

https://github.com/espressif/esp-idf/tree/release/v3.3

and the project guide and code for ESP-IDF-NAT example is written by GitHub ID: *jonask1337.* Please obtain the project file from below link.

https://github.com/jonask1337/esp-idf-nat-example/tree/esp-idf-v3.3

Note: There is newer version of code for NAT example available, this example is used because of the simplicity of the code, uncomplicated workflow of the example and main purpose is to test the hardware performance.

2. Setup Guide for NAT Router

2.1. Setup Block Diagram

Figure 2.1 shows the setup for ESP32-M1. The setup consists of a device accessing the internet, for example, a PC or a mobile device as an STA connecting to ESP32-M1 as an AP, the ESP32-M1 connected to the router as an STA. The router connected to internet services.

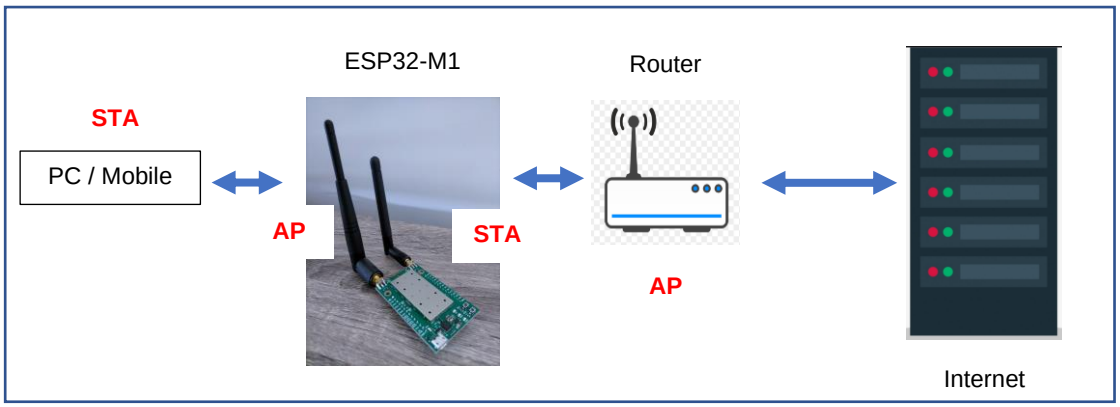


Fig 2.1: Setup of ESP32-M1 as AP and STA.

To implement NAT for ESP32, please follow these steps;

1. Download the repository of the project using the following command,

```
git clone https://github.com/jonask1337/esp-lwip.git
```

2. Rename or delete the in the directory ***esp-idf/component/lwip/lwip***

3. Move the downloaded lwIP library with NAT repository folder from Step 2 to ***esp-idf/component/lwip/***
4. Rename the lwIP library with the NAT repository folder from ***esp-lwip*** to ***lwip***.

5. Open the project in ESP-IDF and follow the below step in Section 2.2.

2.2. Build and Flash.

Open ***make menuconfig*** tool and proceed with the below steps,

1. Go to ***Component config -> LWIP*** see the red box below in Figure 2.2 and set "*Enable copy between Layer2 and Layer3 packets*" option.

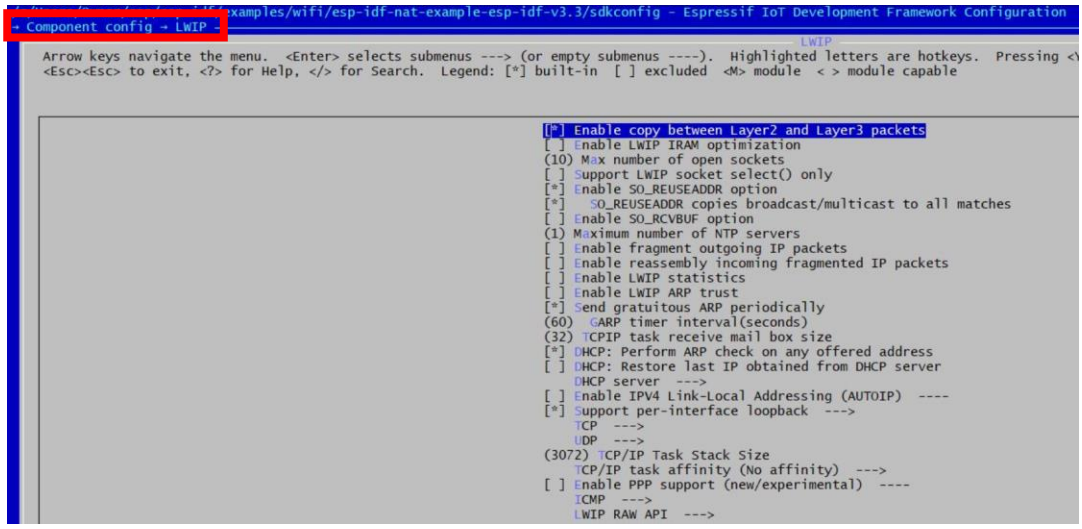


Fig 2.2: Component config section.

2. In ***Example Configuration*** see the red box below in Figure 2.3,

- For ***WiFi SSID*** provide Router SSID which the ESP32-M1 will connect to. This can be home, office router or mobile hotspot.
- For ***STA Password*** provide the Router Password.
- For ***AP SSID*** give a name for the ESP32-M1. This is the SSID for ESP32-M1.

- And provide a password for ESP32-M1 in ***AP Password***.

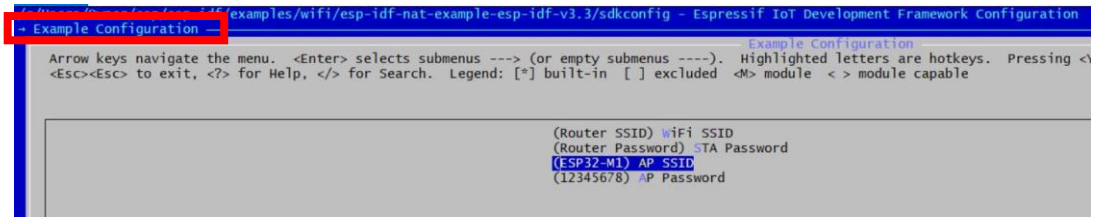


Fig 2.3: Example Configuration section.

3. Once the above configuration is saved and serial port configured in ***make menuconfig***, build, and flash the code discussed in Section 2.

2.3. Repeater Setup

The ESP32-M1 board can perform as a repeater. The function of a repeater is to receive the existing Wi-Fi signal and transmit the signal to a different area for wider coverage. Figure 2.4 shows the operation of the repeater with two ESP32-M1 boards. The initial setup for ESP32-M1 stays as in Figure 2.1 and the ESP32-M1_Repeater now connected to ESP32-M1. The PC now connect to ESP32-M1_Repeater for internet access.

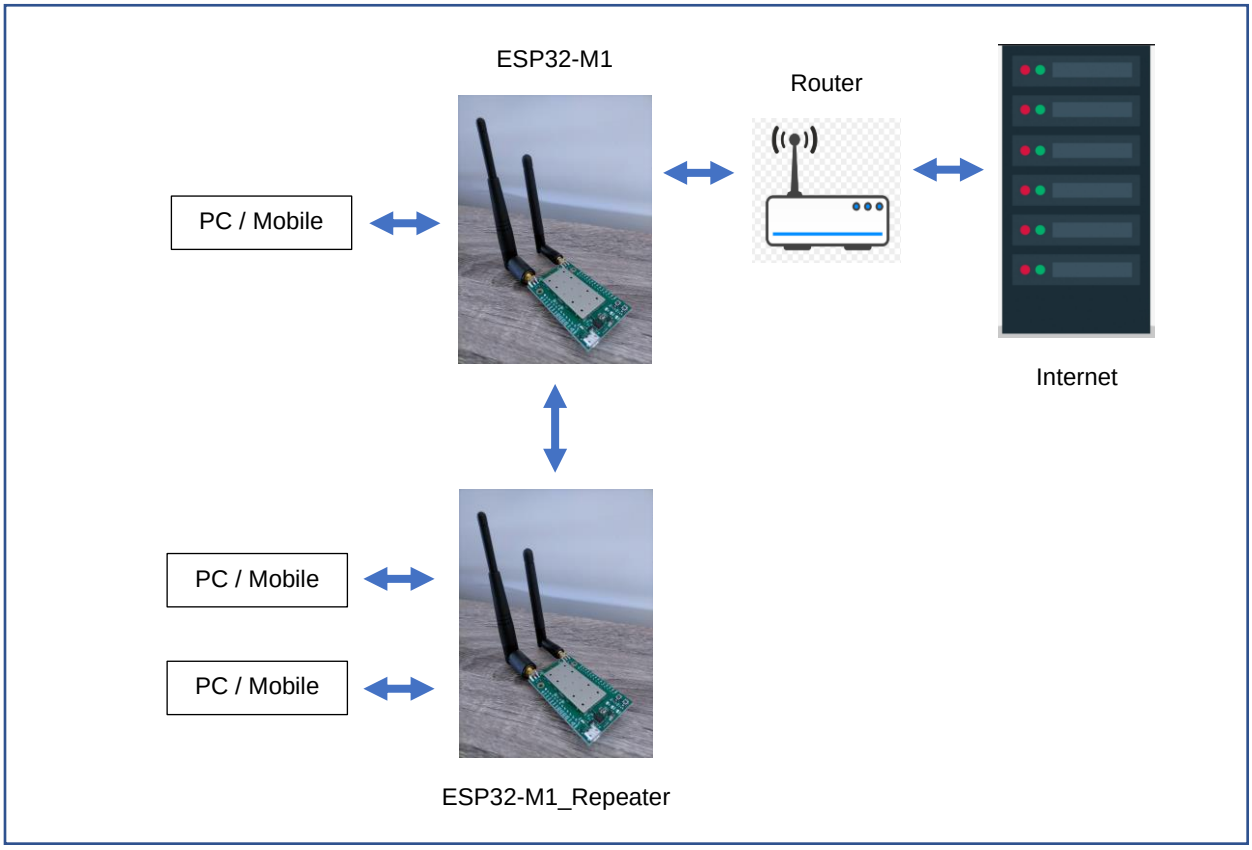


Fig 2.4: Setup of ESP32-M1 as a repeater.

To setup as a repeater for ESP32-M1_Repeater board use the same code and steps from Section 2.1. The only change needed is for ***Example Configuration*** in ***make menuconfig.***

- For ***WiFi SSID*** provide ESP32-M1 SSID which the ESP32-M1_Repeater will connect to.
- For ***STA Password*** provide the ESP32-M1 Password.
- For ***AP SSID*** give a name for the ESP32-M1_Repeater (ESP32-M1_Rep used in Figure 2.5). This is the SSID for ESP32-M1_Repeater.
- And provide a password for ESP32-M1_Repeater in ***AP Password***.

Figure 2.5 shows the configuration for ***Example Configuration*** for ESP32-M1_Repeater. Once configured build and flash.

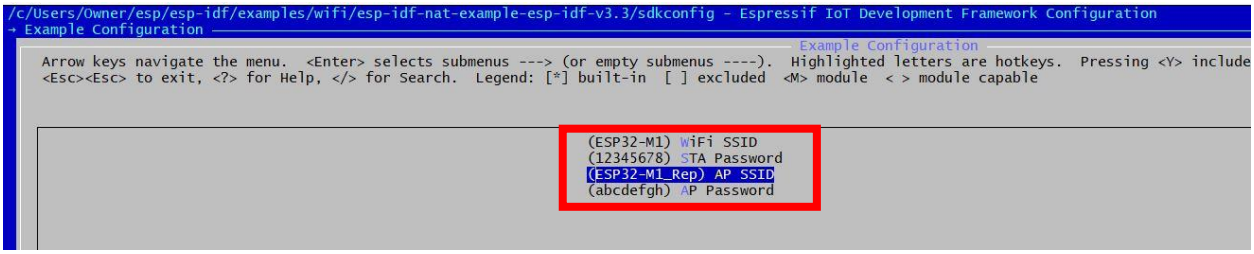


Fig 2.5: Setup of ESP32-M1_Rep as Repeater.

2.4. IP Address

The IP Addresses for connection with ESP32-M1s mapped below.

The ESP32-M1 in the below setup Figure 2.6 is an AP and STA at the same time. The ESP32-M1 connected to the router has an IP address of 192.168.1.196 is an STA, on the other side the ESP32-M1 is an AP with gateway IP address 192.168.4.1 for the PC, the PC IP address is 192.168.4.2.

The ESP32-M1_Rep similar to ESP32-M1 is an AP and STA at the same time. The ESP32-M1_Rep IP address once connected to ESP32-M1 is 192.168.4.4 is an STA and 192.168.1.1 is an AP and the PC connected to ESP32-M1_Rep is 192.168.4.2.

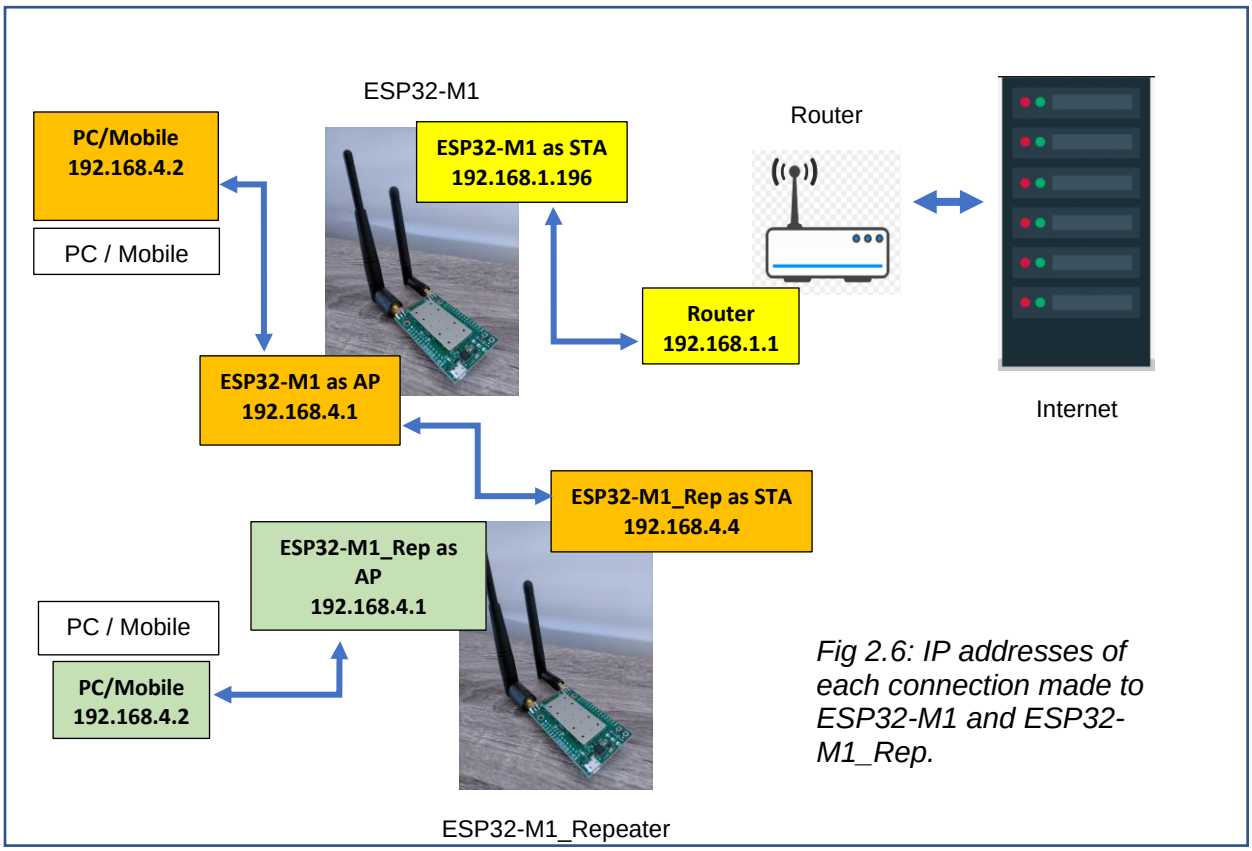


Fig 2.6: IP addresses of each connection made to ESP32-M1 and ESP32-M1_Rep.

3. Speed Test

3.1. Indoor Speed Test

The indoor speed test performed by connecting to ESP32-M1 board to the home router, and ESP32-M1_Rep repeater to ESP32-M1 with below setup;

- Antenna – 3dBi Dipole antenna, model BS Dip 3-24.pdf.
- Antenna in vertical position.
- Tested Range – 15m, a wall in between.
- Speed test performed using https://www.speedtest.net/
- One PC connected to ESP32-M1.
- One PC connected to ESP32-M1_Rep

The speed test performed on the home router initially to obtain the base speed and compared to ESP32-M1 and ESP32-M1_Rep.

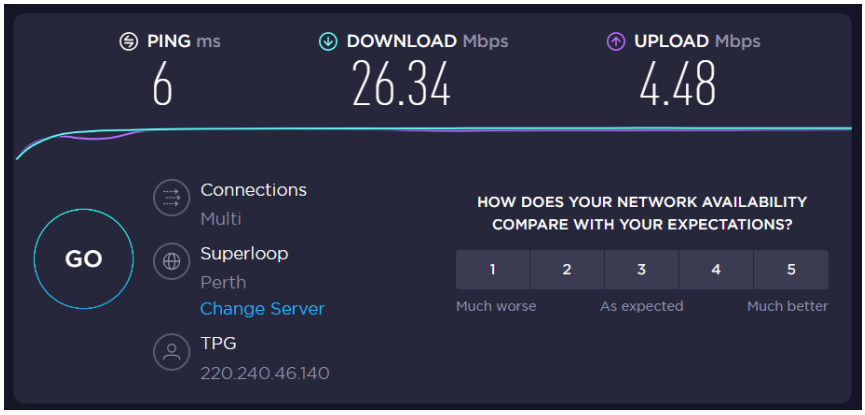
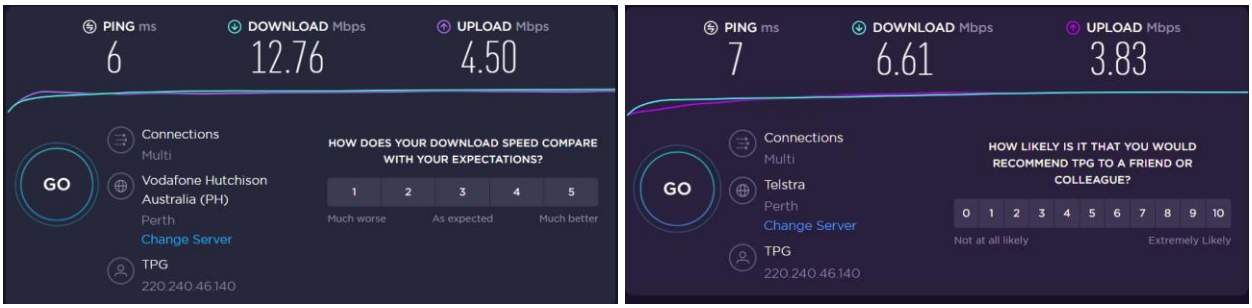


Fig 3.1: Home router speed test.



(a). *(b).*

Fig 3.2: (a). ESP32-M1 speed test connected to home router, b. ESP32-M1 _Rep speed test connected to ESP32-M1.

The above speed test Figure 3.1 shows the home router has a Download speed of 26.34Mbps and Upload speed of 4.48Mbps. The ESP32-M1, Figure 3.2(a) download speed is almost half the speed of a home router and cannot be beyond home router speed, but the Upload speed is almost similar. The ESP32-M1_Rep Figure 3.2(b) speed is half the speed of ESP32-M1 and the Upload speed is 3.83Mbps lower than the ESP32-M1.

3.2. Outdoor Speed Test

The ESP32-M1 positioned on the road outside of the house and connected to the home router in the middle of the house. A laptop is connected to ESP32-M1 for internet access.

The test conducted by walking the laptop further from the ESP32-M1 (increasing the range of the laptop from ESP32-M1) and measure the speed of the connection. The laptop can reach 380m to 400m close to the line of sight and the speed ESP32-M1 to the laptop is 4.19Mbps Download speed and 1.87Mbps Upload speed, Figure 3.3 shows this results. At this distance internet is accessible and YouTube video able to play. Beyond this distance, there is no line of sight for this test area, speed is dropping, and internet accessibility is slow.

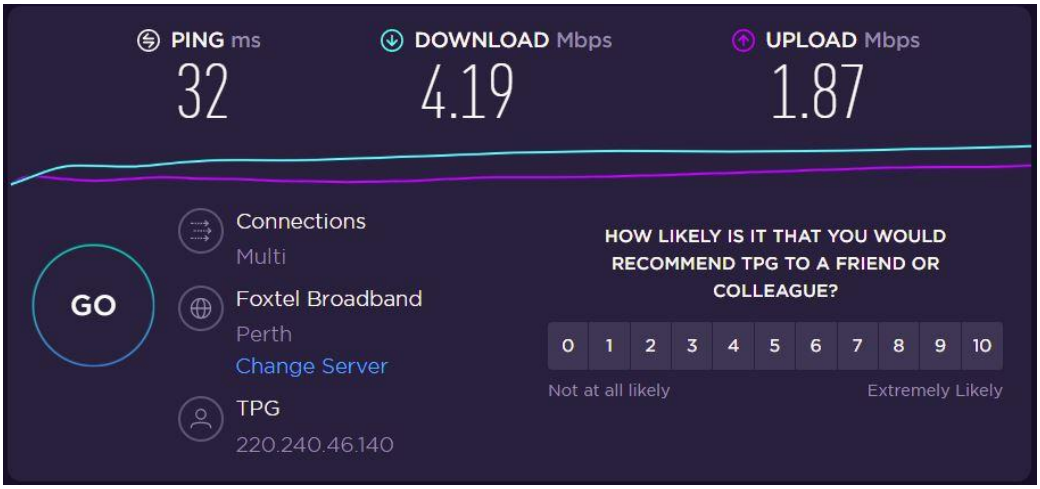


Fig 3.3: Outdoor speed at approximately 380m to 400m from ESP32-M1 to a laptop accessing internet.

Note: On 27th of April 2021 the normal straight road that used as a test area is not permitted to perform testing because of the COVID-19 lockdown restriction. The repeater is not tested at this stage. Once the lockdown restriction is lifted the repeater will be tested.

4. Code

The code below is from ***main.c*** of the project file from Section 1.4 (same linked given below). Some portions of codes are added for ESP32-M1 board.

https://github.com/jonask1337/esp-idf-nat-example/tree/esp-idf-v3.3

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_wpa2.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "driver/gpio.h"
#include "lwip/opt.h"
#include "esp_event_loop.h"

#if IP_NAPT
#include "lwip/lwip_napt.h"
#endif

#include "lwip/err.h"
#include "lwip/sys.h"

#define MY_DNS_IP_ADDR 0x08080808 // 8.8.8.8

// WIFI CONFIGURATION
#define ESP_AP_SSID CONFIG_ESP_AP_SSID
#define ESP_AP_PASS CONFIG_ESP_AP_PASSWORD

#define EXAMPLE_ESP_WIFI_SSID      CONFIG_STA_SSID
#define EXAMPLE_ESP_WIFI_PASS      CONFIG_STA_PASSWORD

#define EXAMPLE_ESP_MAXIMUM_RETRY  3

/* FreeRTOS event group to signal when we are connected*/
static EventGroupHandle_t s_wifi_event_group;

/* The event group allows multiple bits for each event, but we only care about one event
 * - are we connected to the AP with an IP? */
const int WIFI_CONNECTED_BIT = BIT0;

static const char *TAG = "wifi apsta";

static int s_retry_num = 0;

static esp_err_t event_handler(void *ctx, system_event_t *event)
{
  switch(event->event_id) {
  case SYSTEM_EVENT_STA_START:
    esp_wifi_connect();
    break;
  case SYSTEM_EVENT_STA_GOT_IP:
    ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->event_info.got_ip.ip_info.ip));
    s_retry_num = 0;
```

Include the required lwip, see Section 2.1.

```
    xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
    break;
  case SYSTEM_EVENT_STA_DISCONNECTED:
    {
      if (s_retry_num < EXAMPLE_ESP_MAXIMUM_RETRY) {
        esp_wifi_connect();
        xEventGroupClearBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
        s_retry_num++;
        ESP_LOGI(TAG,"retry to connect to the AP");
      }
      ESP_LOGI(TAG,"connect to the AP failed");
      break;
    }
  case SYSTEM_EVENT_AP_STACONNECTED:
    ESP_LOGI(TAG,"station connected");
    break;
  case SYSTEM_EVENT_AP_STADISCONNECTED:
    ESP_LOGI(TAG,"station disconnected");
    break;
  default:
    break;
  }
  return ESP_OK;
}

void wifi_init_sta()
{
    ip_addr_t dnsserver;
    //tcpip_adapter_dns_info_t dnsinfo;

    s_wifi_event_group = xEventGroupCreate();

    tcpip_adapter_init();
    ESP_ERROR_CHECK(esp_event_loop_init(event_handler, NULL) );

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    /* ESP STATION CONFIG */
    wifi_config_t wifi_config = {
        .sta = {
            .ssid = EXAMPLE_ESP_WIFI_SSID,
            .password = EXAMPLE_ESP_WIFI_PASS
        },
    };

    /* ESP AP CONFIG */
    wifi_config_t ap_config = {
        .ap = {
            .ssid = ESP_AP_SSID,
            .channel = 0,
            .authmode = WIFI_AUTH_WPA2_PSK,
            .password = ESP_AP_PASS,
            .ssid_hidden = 0,
            .max_connection = 8,
            .beacon_interval = 100
        }
    };

    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_APSTA) );
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config) );
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_AP, &ap_config) );

    // Enable DNS (offer) for dhcp server
    dhcps_offer_t dhcps_dns_value = OFFER_DNS;
    dhcps_set_option_info(6, &dhcps_dns_value, sizeof(dhcps_dns_value));
```

```
    // Set custom dns server address for dhcp server
    dnsserver.u_addr.ip4.addr = htonl(MY_DNS_IP_ADDR);
    dnsserver.type = IPADDR_TYPE_V4;
    dhcps_dns_setserver(&dnsserver);

    //tcpip_adapter_get_dns_info(TCPIP_ADAPTER_IF_AP, TCPIP_ADAPTER_DNS_MAIN, &dnsinfo);
    //ESP_LOGI(TAG, "DNS IP:" IPSTR, IP2STR(&dnsinfo.ip.u_addr.ip4));

    ESP_ERROR_CHECK(esp_wifi_start());

    ESP_LOGI(TAG, "wifi_init_apsta finished.");
    ESP_LOGI(TAG, "connect to ap SSID: %s ",
             EXAMPLE_ESP_WIFI_SSID);
}

void gpio_out_tx_status()
{
        wifi_ant_gpio_config_t config_gpio;
        config_gpio.gpio_cfg[0].gpio_select = 1;
        config_gpio.gpio_cfg[0].gpio_num = 23;
        config_gpio.gpio_cfg[1].gpio_select = 1;
        config_gpio.gpio_cfg[1].gpio_num = 18;
        wifi_ant_config_t config_ant = {
                        .rx_ant_mode = WIFI_ANT_MODE_ANT1,
                        .rx_ant_default = WIFI_ANT_MODE_ANT1,
                        .tx_ant_mode = WIFI_ANT_MODE_ANT0,
                        .enabled_ant0 = 1,
                        .enabled_ant1 = 2
        };

        esp_wifi_set_ant_gpio(&config_gpio);
        esp_wifi_set_ant(&config_ant);
}

void app_main()
{
  // Initialize NVS
  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);

  // Setup WIFI
  wifi_init_sta();

#if IP_NAPT
  u32_t napt_netif_ip = 0xC0A80401; // Set to ip address of softAP netif (Default is 192.168.4.1)
  ip_napt_enable(htonl(napt_netif_ip), 1);
  ESP_LOGI(TAG, "NAT is enabled");
#endif

  int8_t i = 60;
  int8_t j;

  //Enable WiFi
  gpio_pad_select_gpio(21);
  gpio_set_direction(21, GPIO_MODE_OUTPUT);
  gpio_set_level(21, 1);//Set 1 to enable WiFi and 0 for disable.

  //Disable BT
  gpio_pad_select_gpio(22);
  gpio_set_direction(22, GPIO_MODE_OUTPUT);
  gpio_set_level(22, 0);//Set 1 to enable BT and 0 for disable.
```

Setting the TX and RX switch RFFE

Setting TX RF power

Enable WiFi path

Disable BT path

```
  //GPIO pull down initially for TX
  gpio_pad_select_gpio(23);
  gpio_set_direction(23, GPIO_MODE_OUTPUT);
  gpio_set_level(23, 0);

  //GPIO pull down initially for RX
  gpio_pad_select_gpio(18);
  gpio_set_direction(18, GPIO_MODE_OUTPUT);
  gpio_set_level(18, 0);


  //TX and RX Selection, Pin 23-TX and 18-RX
  gpio_out_tx_status();

  //Setting up WiFi power. Set for 60 for 29.5dBm to 30dBm RF output power.
  ESP_ERROR_CHECK(esp_wifi_set_max_tx_power(i));
  ESP_LOGI(TAG, "Wifi power set to %d.", i);
  ESP_ERROR_CHECK(esp_wifi_get_max_tx_power(&j));
  ESP_LOGI(TAG, "\t\t\tWifi power get is %d.", j);

}
```

GPIO 23 is pulled low

GPIO 18 is pulled low

Setting GPIO TX and RX switch

Setting RF Power

----------END OF DOCUMENT----------