Engineering Capstone Project

ESP8266 Wireless Bridge

Project Report

By

Anubhab de (7749260)

Sanjay ji Kshatriya (7843154)

Zankrut Shukla (7661200)

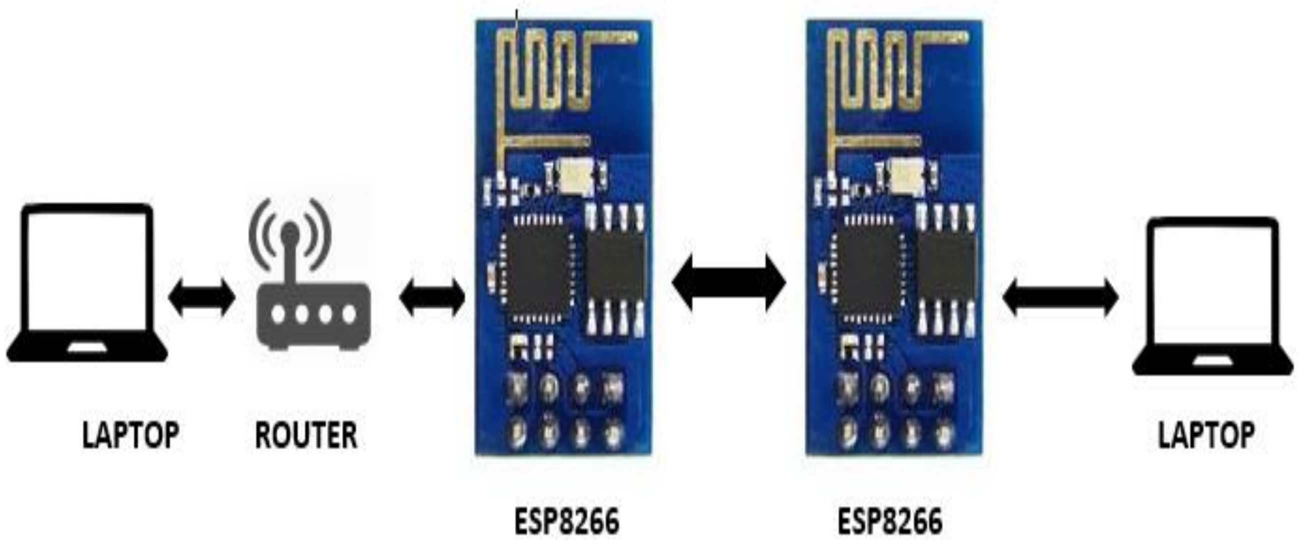
Table of Contents

Page no

INTRODUCTION

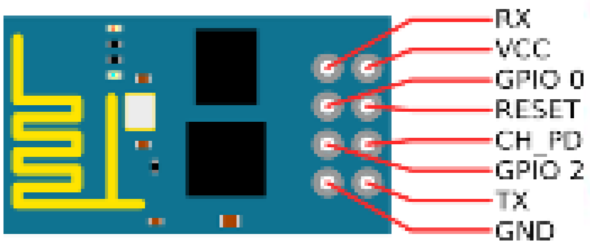
ESP8266 is a popular low cost WIFI module that is used in most IOT based application. The purpose of this project is to involve multiple ESP8266 modules and create a wireless bridge. This is done for testing the feasibility of using these multiple ESP8266 wireless modules and extend the range of a local area network by distances on the order of ∼1km.

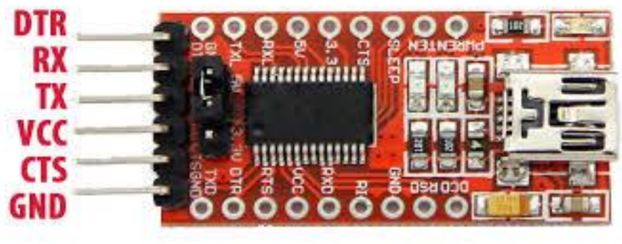
A wireless bridge is an arrangement of devices (minimum two) that link two wired network segments, wirelessly.ESP8266 wireless bridge is based on the idea that ESP8266 is capable of working in station and AP mode at the same time. We are trying to use the module to send and receive the data and transmit it to other ESP8266 module and create a bidirectional connection. This bidirectional connection between multiple ESP modules will create a bridge and we can extend the range of a local area network by distances on the order of ∼1km.



SPECIFICATION

The goal of this project is to create a Wireless bridge using ESP8266.
We have decided to code in Eclipse Kepler IDE on JDK 10 platform with Mingw GCC compiler.
We use the ESP8266 flasher to flash tool to flash our module
We are also going to use “Espressif” development kit to code for our ESP8266 module.
We need a serial terminal to check the output. We are using “Putty” 64 bit for out serial terminal.
All the software’s were decided as per our convince to complete our project.





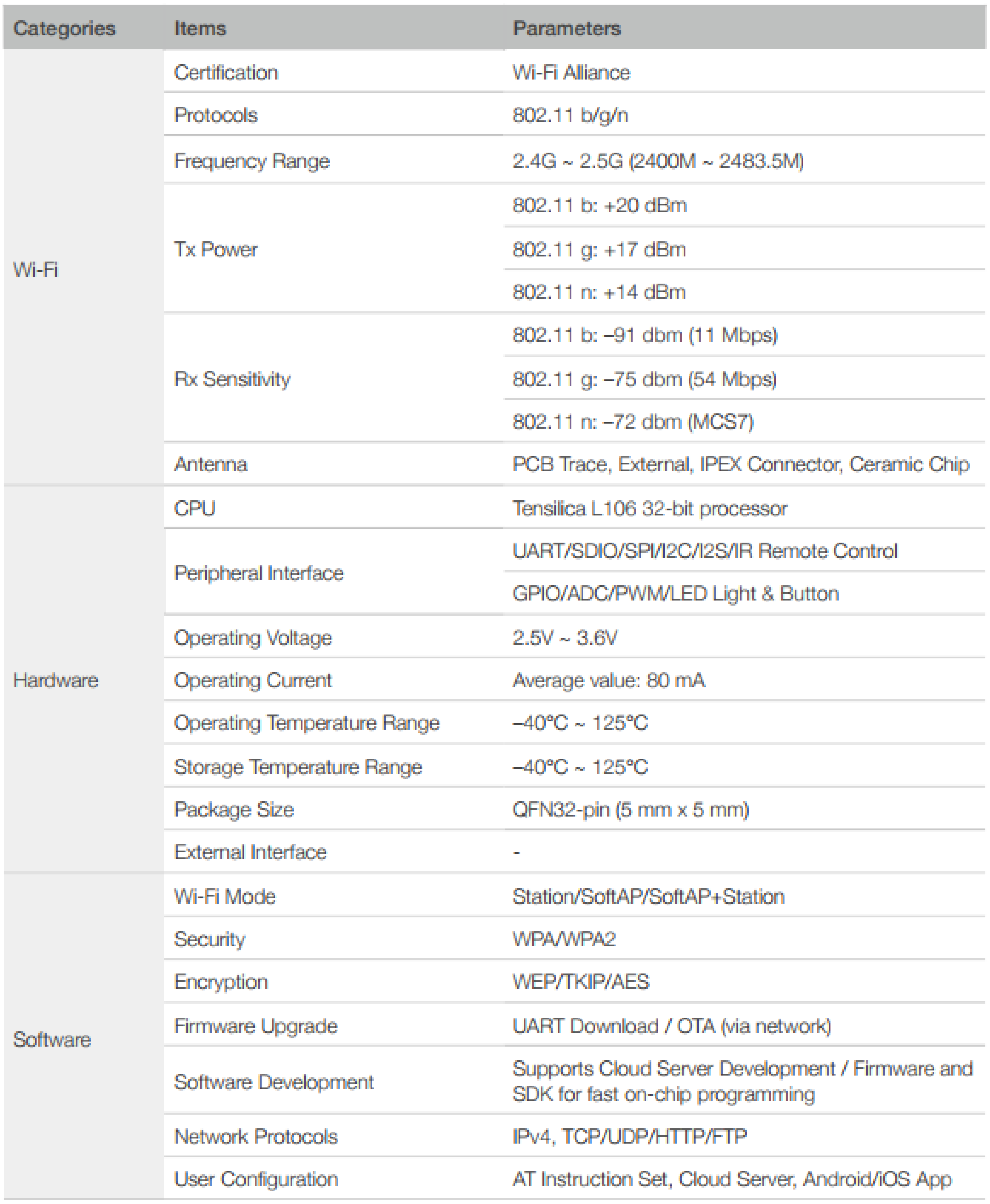
ESP8266 Pin out FT232RL USB TTL

There are 3 modes of operation for ESP8266 module

1. **Dual station mode**
2. **Soft Access Point mode**
3. **Station mode**

In station mode the ESP8266 module will connect to main server or router and in soft AP mode module will work as a router. We shall utilize both the modes for our project.
We should set up the tool chain and shall start programming the ESP8266 module. Using the ESP8266 flash tool we shall first try to connect the module to network in STA mode. We connect the Arduino to USB TTL and then to ESP8266 module. ESP8266 will be programmed through USB TTL.
We shall first connect one module to the server and test the system then will start adding mode modules as per the distance requirement.
We have to search a lot for understanding the tools. We first thought of Linux for coding and debugging but we do not have enough experience with Linux so we choose Eclipse to work on our code.

Specifications of ESP8266:

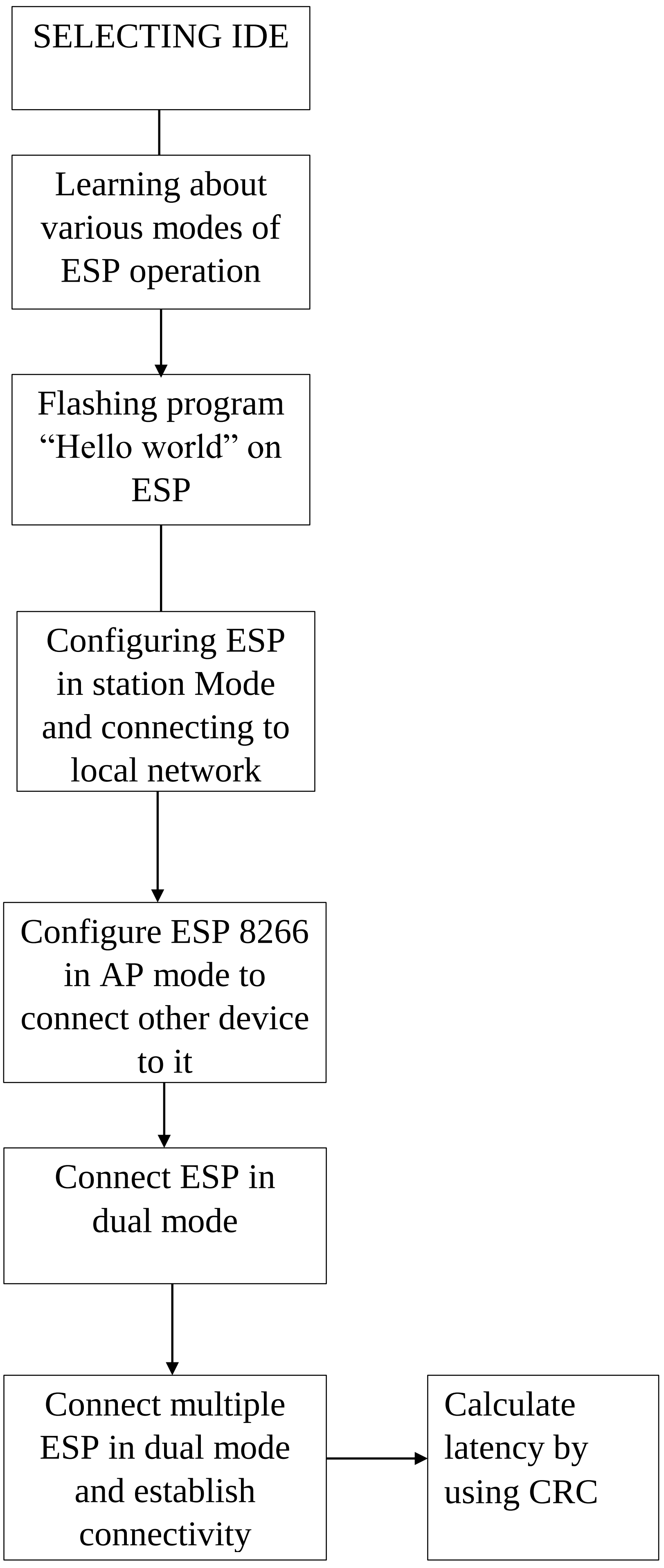


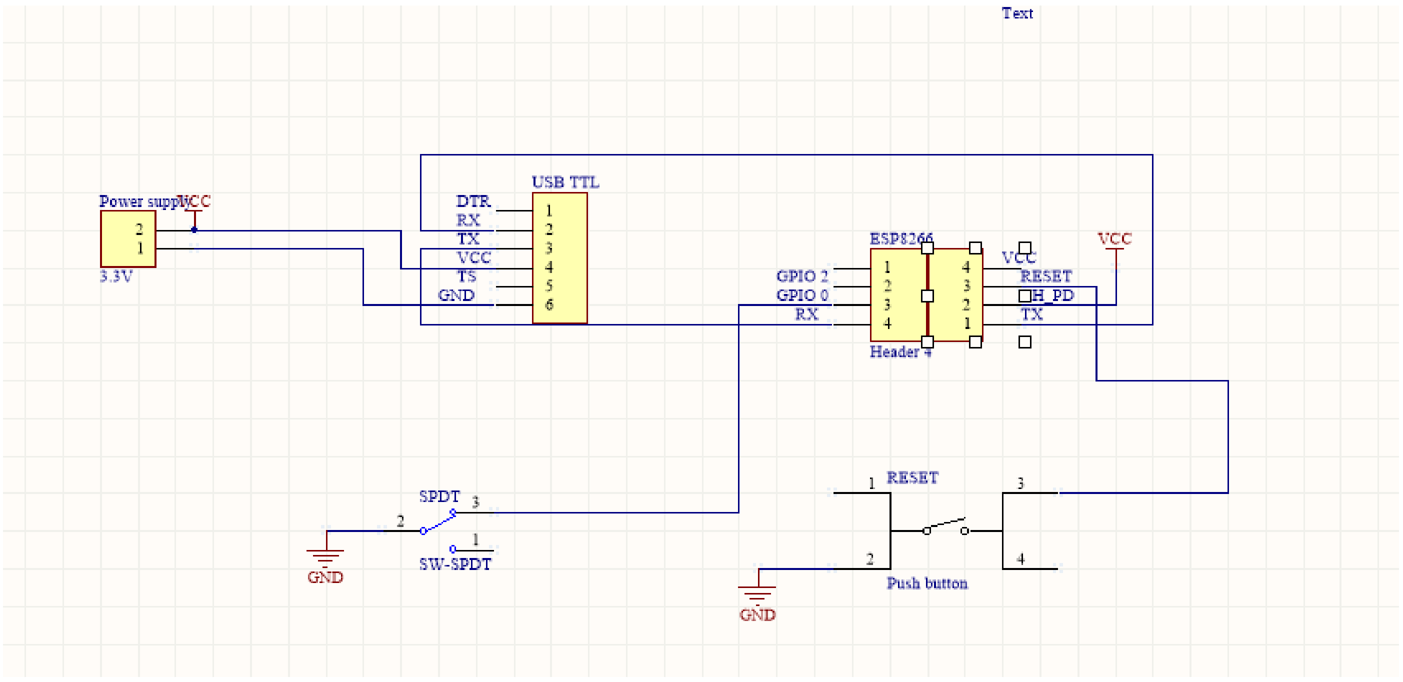
Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: –91 dbm (11 Mbps)
		802.11 g: –75 dbm (54 Mbps)
		802.11 n: –72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5V ~ 3.6V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	–40°C ~ 125°C
	Storage Temperature Range	–40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

The following are the equipment that was provided for the project:

1. 1X Addicore breadboard prototyping kit.
2. 1X Strip of male to female headers.
3. 5X ESP8266 wireless modules.
4. 5X 3 AA battery holder.
5. 1X Arduino with USB cable.
6. ~8 3.3v regulators with heatsink.
7. 1X USB serial TTL module (in case if we want to flash without Arduino).

FLOW DIAGRAM


SELECTING IDE
Learning about various modes of ESP operation
Flashing program "Hello world" on ESP
Configuring ESP in station Mode and connecting to local network
Configure ESP 8266 in AP mode to connect other device to it
Connect ESP in dual mode
Connect multiple ESP in dual mode and establish connectivity
Calculate latency by using CRC



Circuit diagram

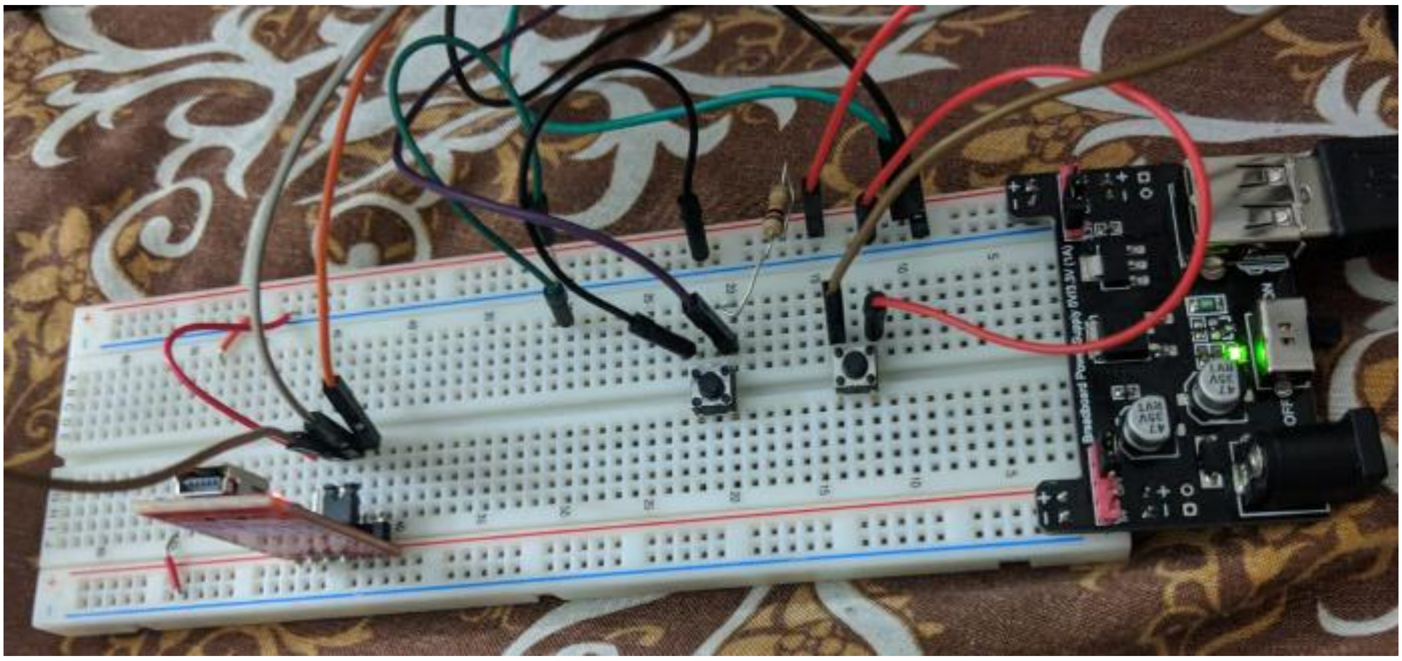
TIMELINE

We were assigned this project in the month of May 2018 with **Mr. Robert Elder** as our client by our advisor **Mr. Rudy**.

Project Roles:
Anubhab will be our project leader and shall be working software part for the project. Sanjay will be coordinating the projects and shall be working with Anubhab. Sanjay also will be debugging the issues while doing his research about the components. Sanjay will also create reports for the project. Zankrut will be working with circuits and shall be collecting the data for the project. Zankrut shall also be taking care of documentation and ppt required for presentation of project.

The following are our activities that we followed in achieving the wireless bridge with ESP8266 modules.

Start Time: 20:48 Hrs **Date: 05/17/2018**



Circuit built for ESP8266 module

We started to done some basic research about ESP8266 and wireless bridge.

While going through the several videos and tutorials like the ones available on https://www. HYPERLINK "https://www.nodemcu.readthedocs.io/en/master/en/" HYPERLINK "https://www.nodemcu.readthedocs.io/en/master/en/" HYPERLINK "https://www.nodemcu.readthedocs.io/en/master/en/"nodemcu.readthedocs.io/en/master/en/ and https://www.youtube.com/watch?v=p06NNRq5NTU

We got to understand that the ESP8266 can be programmed to have three different modes of operations. They are

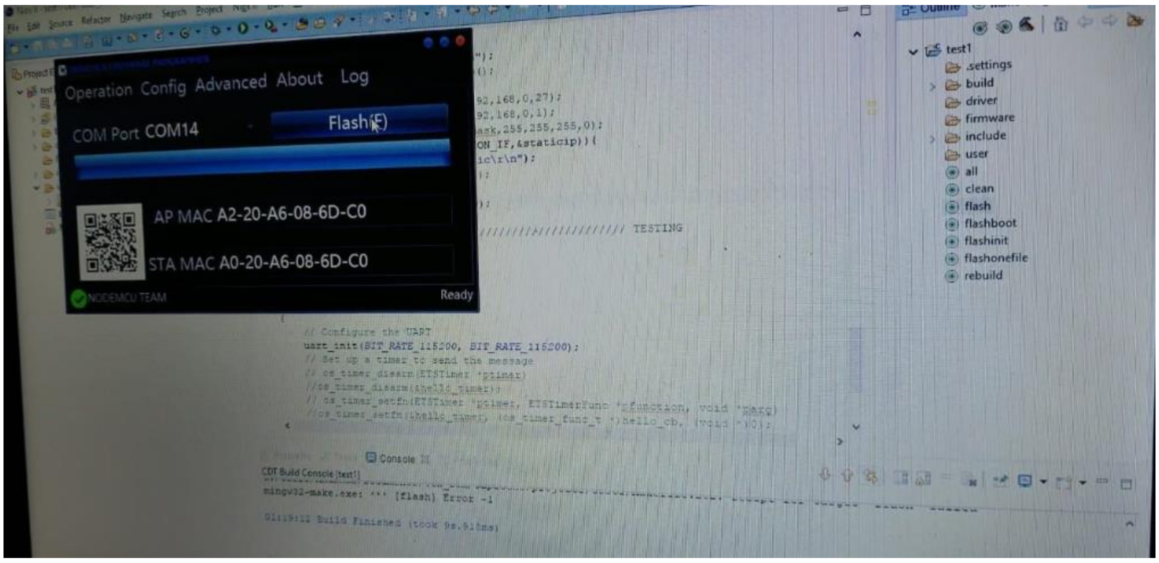
- Station Mode : In this mode the ESP8266 acts as a device and connects to an existing access point

- Soft AP Mode : Access Point mode where the ESP itself acts as AP & other devices like Mobile can connect to it.
- Dual or AT+CW Mode :In this mode the ESP can act both as a station as well as an access point at the same time.

In order to build a wireless bridge using ESP8266 modules the ESP WiFi modules needs to be configured in the Dual or AT+CW Mode as it needs to connect to the local network or another ESP8266 in the Slave configuration while at the same time it should also act as a Master and provide access points for other ESP modules or other devices to connect to it.

So, we decided to initially configure the ESP8266 in station mode and start flashing my firmware on the device. To do that we had to download and setup a handful of softwares from the internet. We have chosen Eclipse Keppler IDE to develop the firmware for the ESP8266 modules. Here are the list of the software's that we have downloaded :

- ECLIPSE KEPPLER (IDE) – This is the development environment that we have chosen to develop my firmware for ESP8266. We have chosen this IDE because this IDE because Anubhab have already had the experience of working using Eclipse Environment as we have used Atollic in the first semester.\
- ESPRESSIF DEVELOPMENT KIT – This is the development tool provided by the manufacturers of ESP8266 which can be easily integrated into Eclipse. It contains many basic examples to start working with initially.
- MINGW GCC Compiler : This compiler is needed as windows doesnot have a built in GCC compiler and we need a GCC Compiler to compile the C/C++ codes written Eclipse IDE.
- Java Development Kit 10 (JDK 10) – The JDK is required to provide the run-time environment for eclipse.
- PUTTY 64-Bit – The putty provides the serial terminal needed to display the occurrence of events and other information via the serial comm. port. This is required as the ESP8266 communicates through TCP/IP protocol where its Rx/Tx and pins are used to send and receive data through the serial port.
- ESP8266 Flasher : The ESP8266 flasher is a very essential tool which is needed to flash the firmwares written in eclipse on to the device.



COM Port COM14

AP MAC A2-20-A6-08-6D-C0

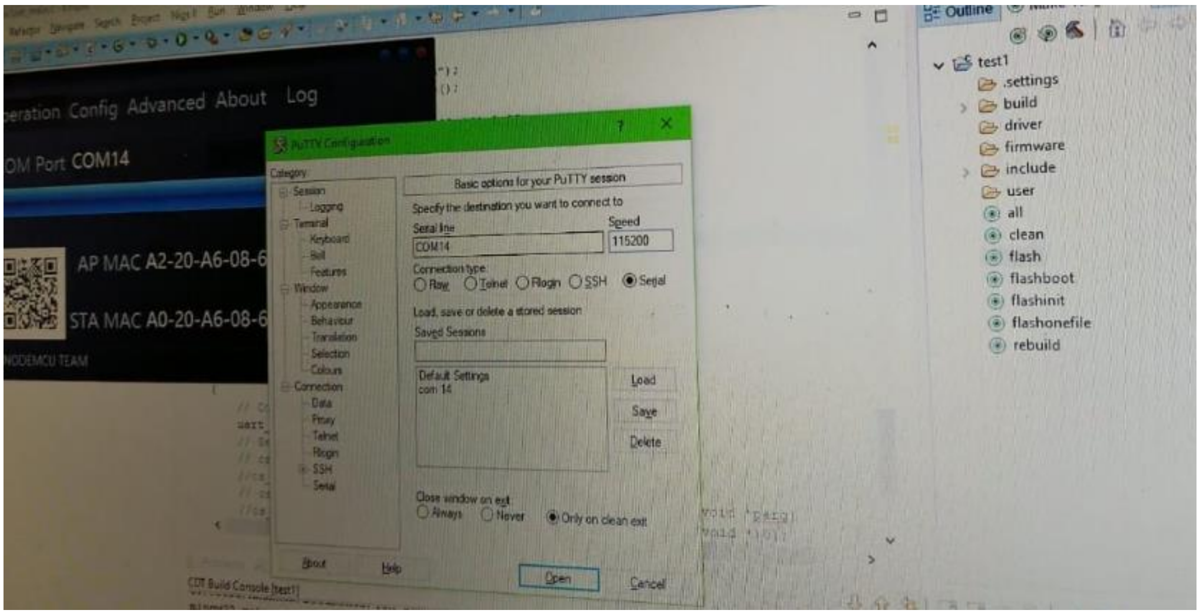
STA MAC A0-20-A6-08-6D-C0

```
// Configure the UART
uart_init(BIT_RATE_115200, BIT_RATE_115200);
// Set up a timer to send the message
// os_timer_disarm(ETSTimer *ptimer)
//os_timer_disarm(&hello_timer);
// os_timer_setfn(ETSTimer *ptimer, ETSTimerFunc *pfunction, void *parg)
//os_timer_setfn(&hello_timer, (os_timer_func_t *)hello_cb, (void *)0);
```

CDT Build Console [test1]

```
mingw32-make.exe: *** [flash] Error -1

01:19:12 Build Finished (took 9s.512ms)
```

Flash tool for Esp8266


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

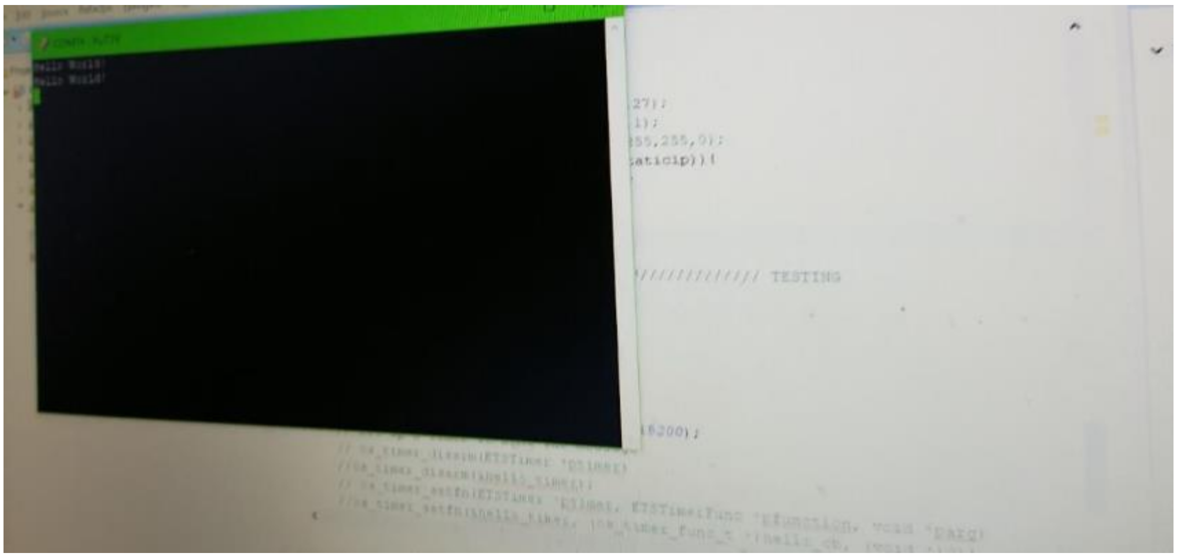
Serial line: COM14 Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Saved Sessions: Default Settings, com 14

Close window on exit: Always Never [✓ Only on clean exit]

Putty” 64 bit for out serial terminal


```
Hello World!
Hello World!
```

Serial terminal

End Time: 00:45 Hrs

Start Time 15:10 Hrs **Date: 05/18/2018**

Today our primary aim will be to flash the ESP8266 using the ESP8266 flashing tool and write the basic lines of code to print “Hello World” over the Serial Terminal while working on building the circuit to flash the ESP8266.

The sample line of codes are provided in the ESPRESSIF DEVELOPMENT KIT to flash the ESP8266 module and display “Hello World” on the terminal.

Anubhab have imported the “Hello World” example provided by ESPRESSIF into Eclipse as my starting point. The process is mentioned below:

Download ESPRESSIF DEVELOPMENT KIT from the link https://www.espressif.com/en/support/download/sdk

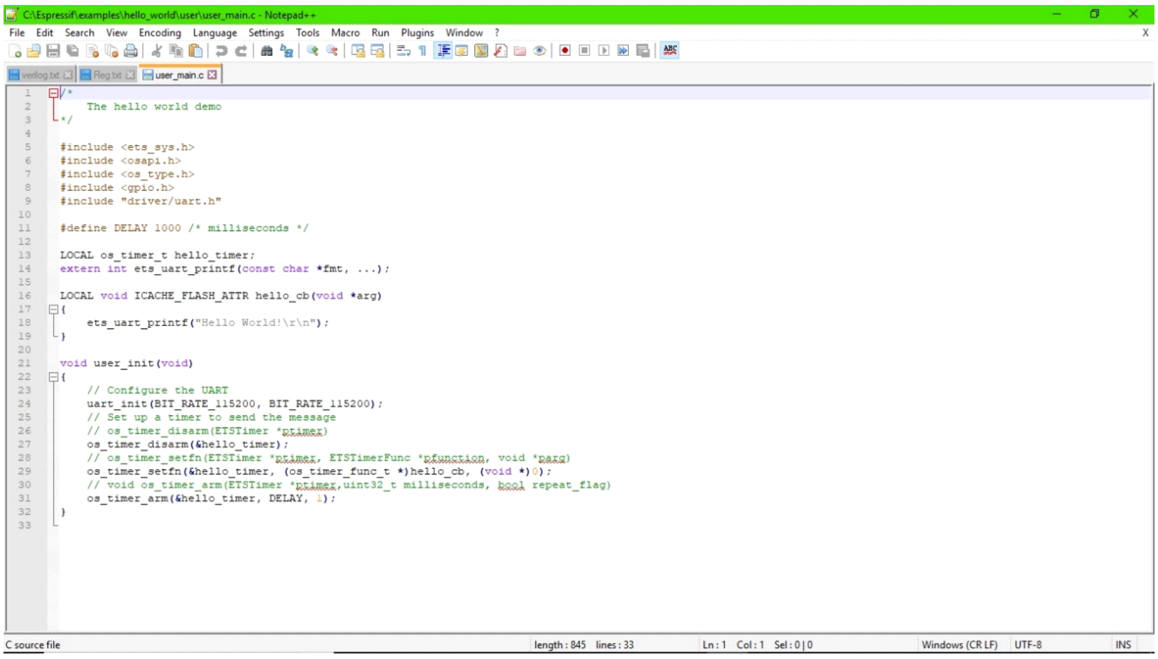
Install the application and a folder by the name of ESPRESSIF will show up in the c:/program files after successfully installation.

Then I opened Eclipse Keppler (Eclipse Oxygen can also be used) and create anew workspace.

Then click on File -> Import -> Existing Projects -> Select root directory -> Browse ->Local Disk (C:) ->Program Files -> Espressif -> Examples ->hello_world.

Import the entire project directory to Eclipse.

Then expanded the Hello_World project directory click on user folder and open the user_main.c file. This was our starting point.


C:\Espressif\examples\hello_world\user\user_main.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
verilog.txt
Reg.txt
user_main.c

```c
/*
    The hello world demo
*/

#include <ets_sys.h>
#include <osapi.h>
#include <os_type.h>
#include <gpio.h>
#include "driver/uart.h"

#define DELAY 1000 /* milliseconds */

LOCAL os_timer_t hello_timer;
extern int ets_uart_printf(const char *fmt, ...);

LOCAL void ICACHE_FLASH_ATTR hello_cb(void *arg)
{
    ets_uart_printf("Hello World!\r\n");
}

void user_init(void)
{
    // Configure the UART
    uart_init(BIT_RATE_115200, BIT_RATE_115200);
    // Set up a timer to send the message
    // os_timer_disarm(ETSTimer *ptimer)
    os_timer_disarm(&hello_timer);
    // os_timer_setfn(ETSTimer *ptimer, ETSTimerFunc *pfunction, void *parg)
    os_timer_setfn(&hello_timer, (os_timer_func_t *)hello_cb, (void *)0);
    // void os_timer_arm(ETSTimer *ptimer,uint32_t milliseconds, bool repeat_flag)
    os_timer_arm(&hello_timer, DELAY, 1);
}
```

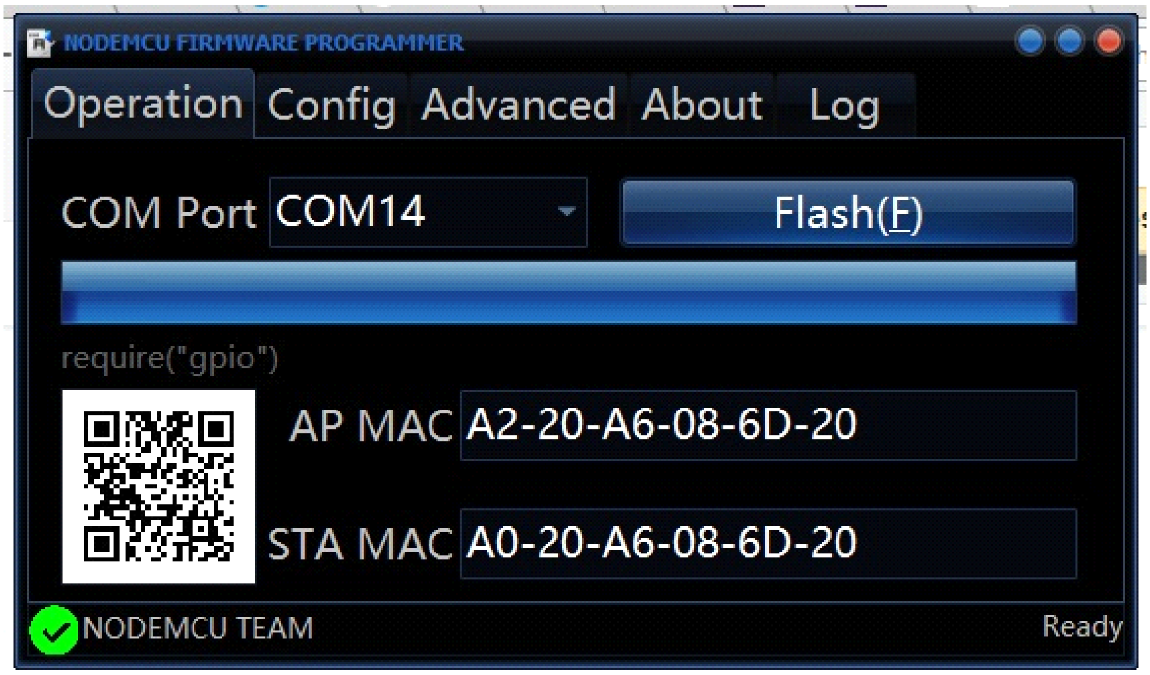
```
C source file    length : 845    lines : 33    Ln : 1    Col : 1    Sel : 0 | 0    Windows (CR LF)    UTF-8    INS
```

After this we started ESP Flashing Tool application and connected the ESP8266 to USB-TTL converter.The Rx pin of ESP8266 is connected to the Tx pin of USB-TTL converter and vice-versa. The GPIO 0 pin of ESP-8266 is grounded when flashing the firmware because that drives the ESP-8266 into booting or flashing mode. The USB-TTL converter is connected to the laptop via USB cable. Constant supply of 3.3V is provided from the micro-controller that is Arduino UNO provided to us by Robert.

After starting up the ESP-Flashing Tool and supplying the power from the Arduino UNO evaluation board the ESP-Flashing Tool detects the MAC address of the connected ESP8266 module.

After this we have followed the following steps to build my project in Eclipse IDE.

Click on Projects -> Build Project.



COM Port COM14

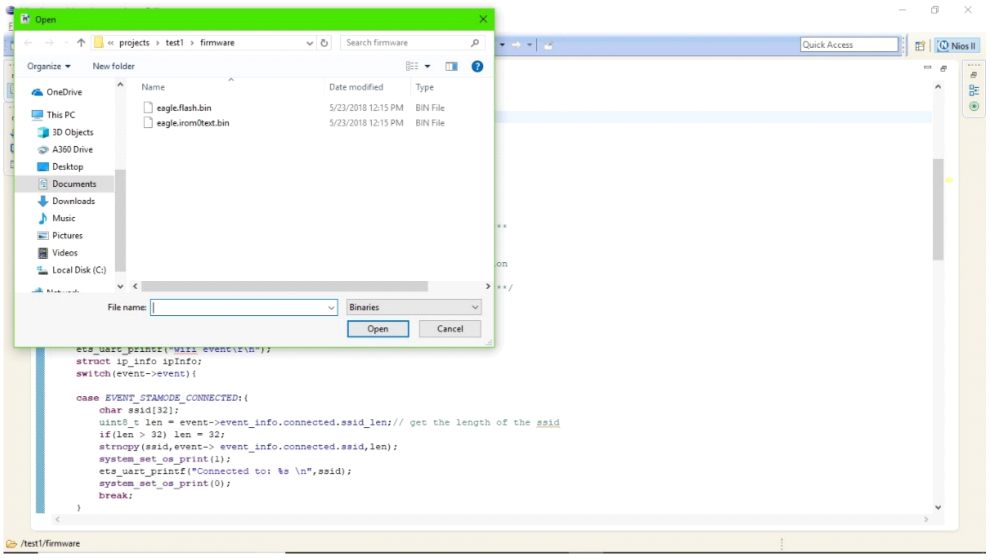
require("gpio")

AP MAC A2-20-A6-08-6D-20

STA MAC A0-20-A6-08-6D-20

This sample code compiled without any errors . Then to reconfirm that the build is successful we clicked on Project Explorer -> expand test1 -> firmware .

Inside the firmware folder we found two binary files with .bin extension. These are the files that are to be flashed onto the core of ESP8266.



Name	Date modified	Type
eagle.flash.bin	5/23/2018 12:15 PM	BIN File
eagle.irom0text.bin	5/23/2018 12:15 PM	BIN File

File name: | Binaries

```
struct ip_info ipInfo;
switch(event->event){

case EVENT_STAMODE_CONNECTED:{
    char ssid[32];
    uint8_t len = event->event_info.connected.ssid_len;// get the length of the ssid
    if(len > 32) len = 32;
    strncpy(ssid,event-> event_info.connected.ssid,len);
    system_set_os_print(1);
    ets_uart_printf("Connected to: %s \n",ssid);
    system_set_os_print(0);
    break;
}
```

/test1/firmware

Once the flashing is done the ESP Flashing Tool will show flashing completed successfully.

The following are the settings of Putty:

Click on the Putty desktop icon ->select serial as connection type ->Serial line should be the comm. Port in which the USB-TTL is connected ->set the speed to 115200.

The Hello World will print on the terminal with a delay 1000ms.

End Time: 23:50 Hrs.

Start Time: 12:38 Hrs **Date : 05/23/2018**

After successfully being able to print Hello World and being able to take the initial step our next goal is to configure ESP8266 in Station Mode so that it can connect to any available Local network. This needs to be done so as to understand how ESP8266 connects to available Local network and how it transmits data to the laptop via the local network.

We started going through the examples of how to configure the ESP8266 in Station Mode available on the internet .These are the few links from which we have taken reference to develop my firmware:

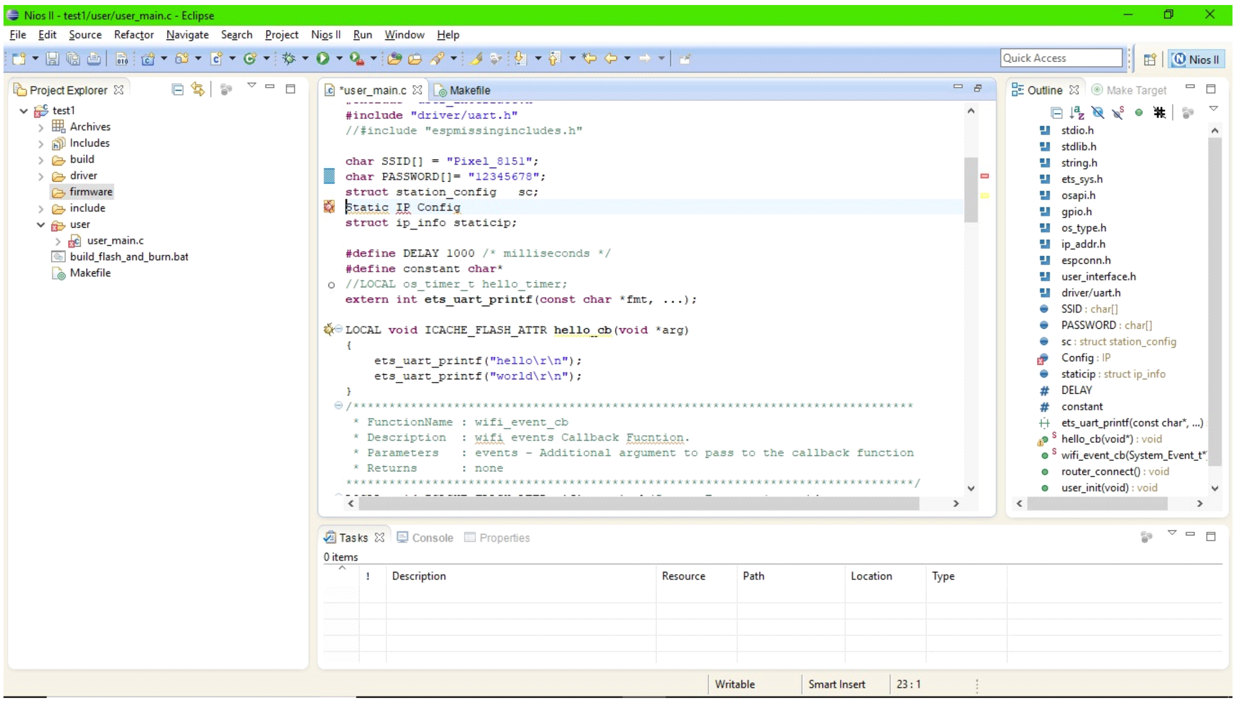
http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/station-class.html

http://fab.cba.mit.edu/classes/865.15/people/dan.chen/esp8266/

We also took reference from the example codes provided by ESPRESSIF.

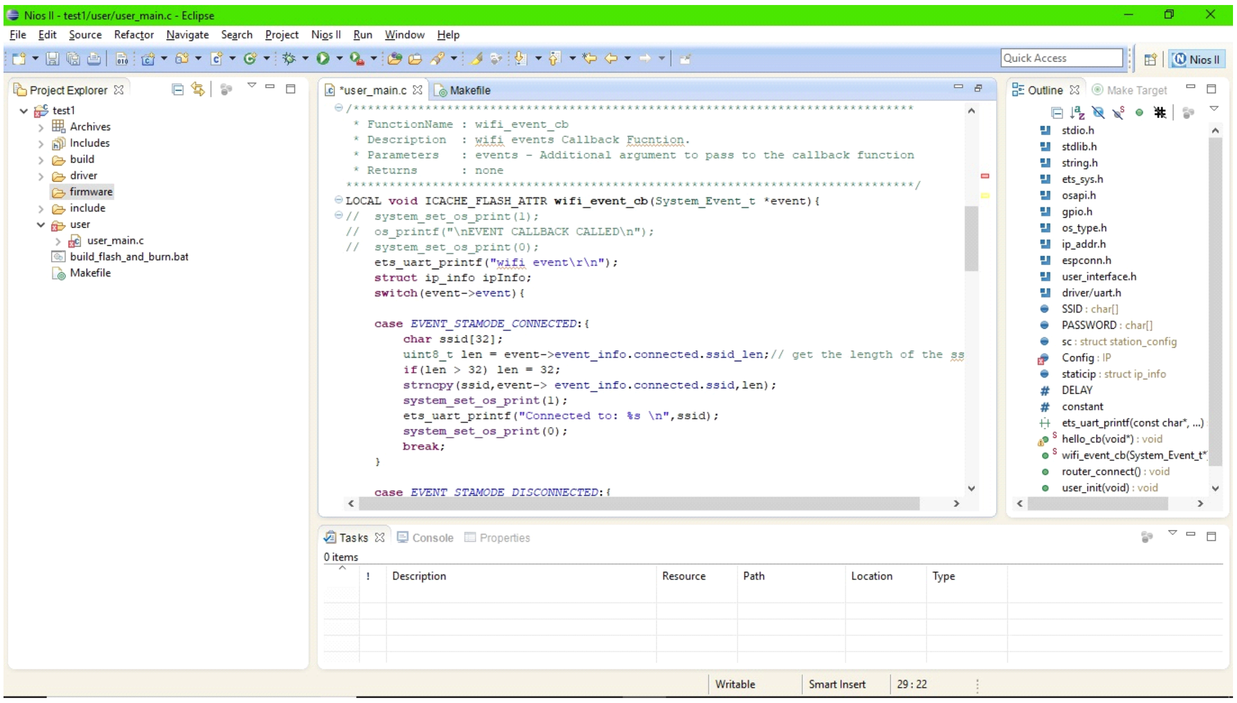
My PC ->Local Disk (C:) ->Program Files -> Espressif -> Examples.

Following are the lines of code that we have written to configure ESP8266 in Station mode.

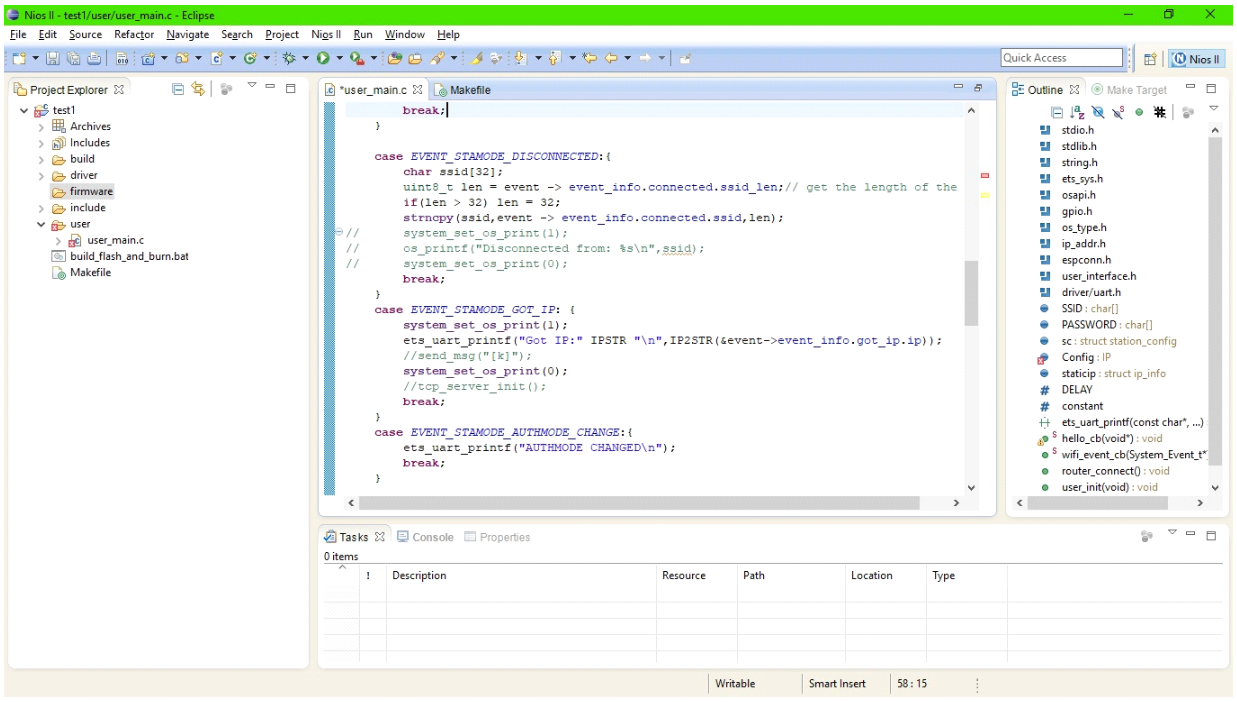


```
#include "driver/uart.h"
//#include "espmissingincludes.h"

char SSID[] = "Pixel_8151";
char PASSWORD[]= "12345678";
struct station_config   sc;
Static IP Config
struct ip_info staticip;

#define DELAY 1000 /* milliseconds */
#define constant char*
//LOCAL os_timer_t hello_timer;
extern int ets_uart_printf(const char *fmt, ...);

LOCAL void ICACHE_FLASH_ATTR hello_cb(void *arg)
{
    ets_uart_printf("hello\r\n");
    ets_uart_printf("world\r\n");
}
/**************************************************************************
 * FunctionName : wifi_event_cb
 * Description  : wifi events Callback Fucntion.
 * Parameters   : events - Additional argument to pass to the callback function
 * Returns      : none
***************************************************************************/
```

We included all the header files that was in the sample program for the “hello_world”. Then SSID and password is declared as character arrays. The ESP8266 is configured in station mode and is given a static IP. The linker to store the instructions in flash memory is also declared.

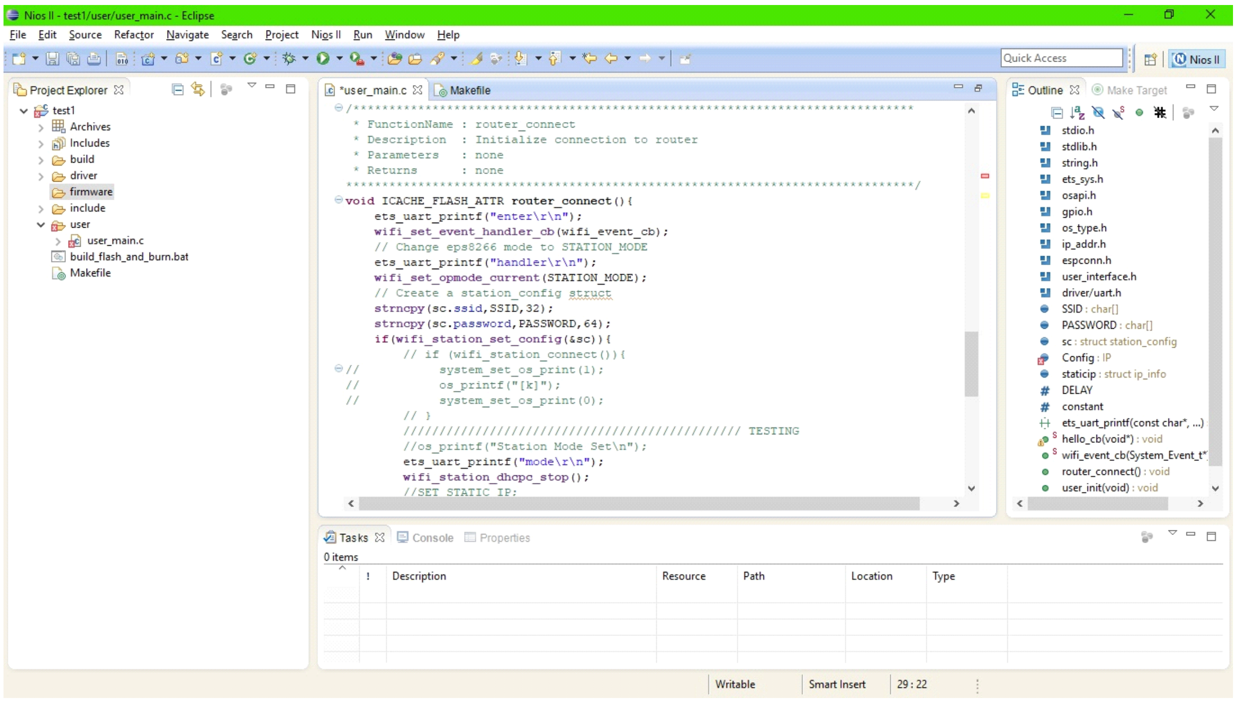


```
/*******************************************************************************
 * FunctionName : wifi_event_cb
 * Description  : wifi events Callback Fucntion.
 * Parameters   : events - Additional argument to pass to the callback function
 * Returns      : none
*******************************************************************************/
LOCAL void ICACHE_FLASH_ATTR wifi_event_cb(System_Event_t *event){
//  system_set_os_print(1);
//  os_printf("\nEVENT CALLBACK CALLED\n");
//  system_set_os_print(0);
    ets_uart_printf("wifi event\r\n");
    struct ip_info ipInfo;
    switch(event->event){

    case EVENT_STAMODE_CONNECTED:{
        char ssid[32];
        uint8_t len = event->event_info.connected.ssid_len;// get the length of the ss
        if(len > 32) len = 32;
        strncpy(ssid,event-> event_info.connected.ssid,len);
        system_set_os_print(1);
        ets_uart_printf("Connected to: %s \n",ssid);
        system_set_os_print(0);
        break;
    }

    case EVENT_STAMODE_DISCONNECTED:{
```

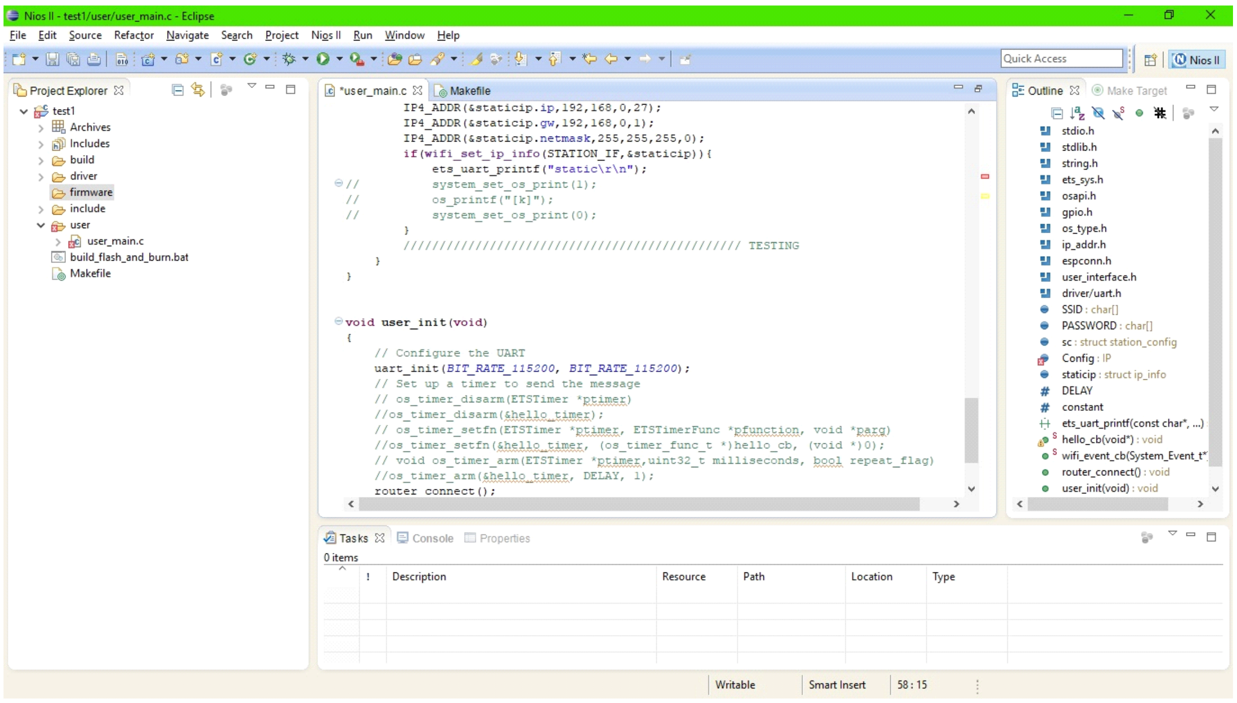


```
        break;
    }

    case EVENT_STAMODE_DISCONNECTED:{
        char ssid[32];
        uint8_t len = event -> event_info.connected.ssid_len;// get the length of the
        if(len > 32) len = 32;
        strncpy(ssid,event -> event_info.connected.ssid,len);
//      system_set_os_print(1);
//      os_printf("Disconnected from: %s\n",ssid);
//      system_set_os_print(0);
        break;
    }
    case EVENT_STAMODE_GOT_IP: {
        system_set_os_print(1);
        ets_uart_printf("Got IP:" IPSTR "\n",IP2STR(&event->event_info.got_ip.ip));
        //send_msg("[k]");
        system_set_os_print(0);
        //tcp_server_init();
        break;
    }
    case EVENT_STAMODE_AUTHMODE_CHANGE:{
        ets_uart_printf("AUTHMODE CHANGED\n");
        break;
    }
```

We have created a function called wifi_event_cb() which has the event as its parameter. We have used switch-case statement to check the if the device is connected on the network, is disconnected from the network, if the ESP received the IP of the local network which acts as the Master in this configuration and to check if the authentication has changed.



```
/*******************************************************************************
 * FunctionName : router_connect
 * Description  : Initialize connection to router
 * Parameters   : none
 * Returns      : none
*******************************************************************************/
void ICACHE_FLASH_ATTR router_connect(){
    ets_uart_printf("enter\r\n");
    wifi_set_event_handler_cb(wifi_event_cb);
    // Change eps8266 mode to STATION_MODE
    ets_uart_printf("handler\r\n");
    wifi_set_opmode_current(STATION_MODE);
    // Create a station_config struct
    strncpy(sc.ssid,SSID,32);
    strncpy(sc.password,PASSWORD,64);
    if(wifi_station_set_config(&sc)){
        // if (wifi_station_connect()){
//          system_set_os_print(1);
//          os_printf("[k]");
//          system_set_os_print(0);
        // }
        /////////////////////////////////////////////// TESTING
        //os_printf("Station Mode Set\n");
        ets_uart_printf("mode\r\n");
        wifi_station_dhcpc_stop();
        //SET STATIC IP;
```



```
        IP4_ADDR(&staticip.ip,192,168,0,27);
        IP4_ADDR(&staticip.gw,192,168,0,1);
        IP4_ADDR(&staticip.netmask,255,255,255,0);
        if(wifi_set_ip_info(STATION_IF,&staticip)){
            ets_uart_printf("static\r\n");
//          system_set_os_print(1);
//          os_printf("[k]");
//          system_set_os_print(0);
        }
        /////////////////////////////////////////////// TESTING
    }
}


void user_init(void)
{
    // Configure the UART
    uart_init(BIT_RATE_115200, BIT_RATE_115200);
    // Set up a timer to send the message
    // os_timer_disarm(ETSTimer *ptimer)
    //os_timer_disarm(&hello_timer);
    // os_timer_setfn(ETSTimer *ptimer, ETSTimerFunc *pfunction, void *parg)
    //os_timer_setfn(&hello_timer, (os_timer_func_t *)hello_cb, (void *)0);
    // void os_timer_arm(ETSTimer *ptimer,uint32_t milliseconds, bool repeat_flag)
    //os_timer_arm(&hello_timer, DELAY, 1);
    router_connect();
```

We wrote a function called router_connect() to initialize connection to the router. The SSID and Password that was defined above is used to connect to the wifi network. Then the IP, Gateway and net mask are set for the ESP8266 to connect to the router.

While writing these lines of codes we got to learn about a few new header files and syntaxes.

- #include “os_type.h”-> this header file contains
    - os_signal_t ETSSignal –> It provides the signal for occurrence of an event

    - os_param__t ETSParam –> It contains the event parameters.
    - os_event_t ETSEvent –> It posts an event to a task
- char SSID = “ “ -> It is used to define the name of the local network
- char PASSWORD = “ “ -> password for the local network is defined here.
- struct_station_config sc -> This is used to configure ESP8266 in station mode.
- struct ip_info staticip -> ESP is configured with a static IP
- ICACHE_FLASH_ATTR -> its an instruction for the linker to specify that the instructions are to be stored in the flash memory instead of RAM
- wifi_set_opmode_current (STATION MODE) -> sets the current operation mode to Station mode.
- wifi_station_dhcpc_stop() –DHCP client is implemented with this command

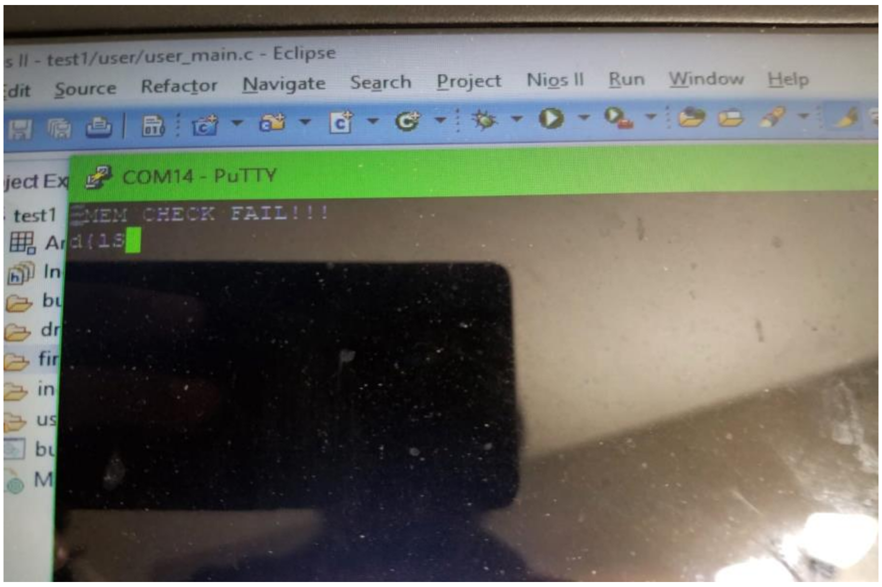
What is DHCP?

Dynamic Host Configuration Protocol (**DHCP**) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configuration information such as the subnet mask and default gateway.

Reference : www.google.com

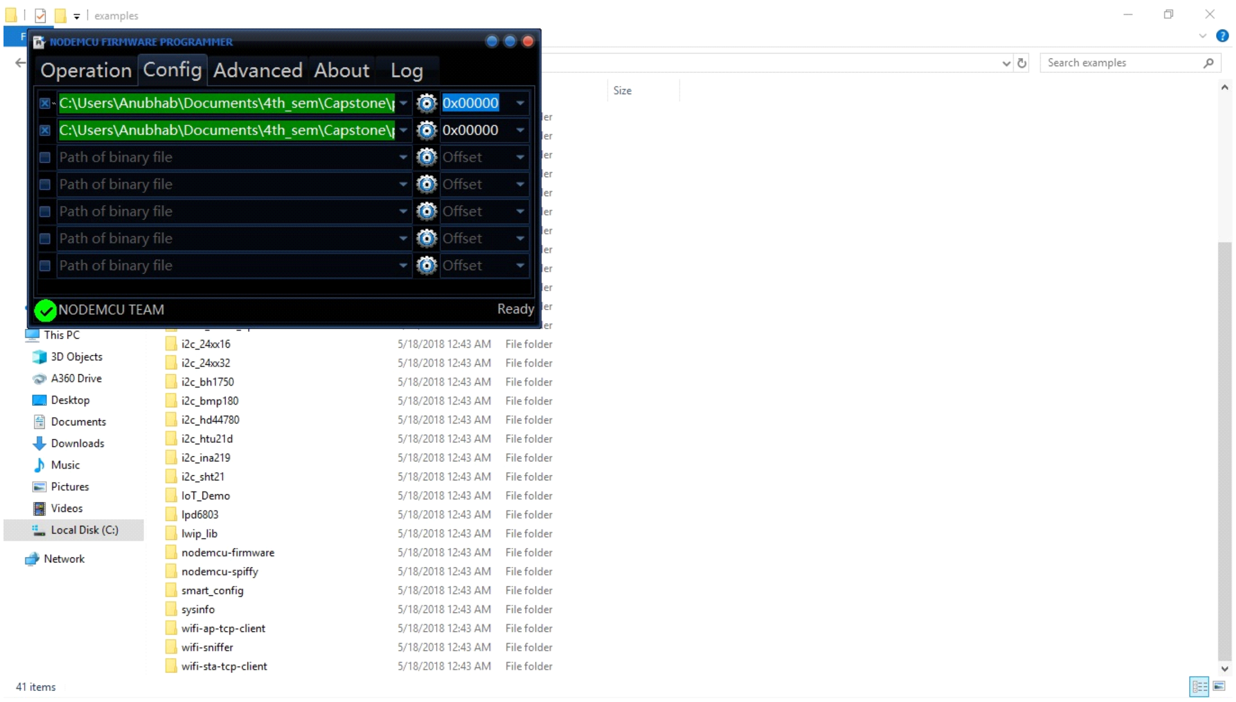
ERRORS WHILE DEBUGGING AND FLASHING:

- Static IP Config: The error message showed that “Type ‘IP’ couldnot be resolved”. Just by commenting this line we solved the error.
- ESP Flashing Tool: Two files contains the same memory address “mem_check_fail”.


s II - test1/user/user_main.c - Eclipse
dit Source Refactor Navigate Search Project Nios II Run Window Help
ject Ex
test1
COM14 - PuTTY

```
MEM CHECK FAIL!!!
d(15
```

We searched on www.google.com to find an answer to the error and found out that this occurs if the two binary files are flashed at the same memory address.

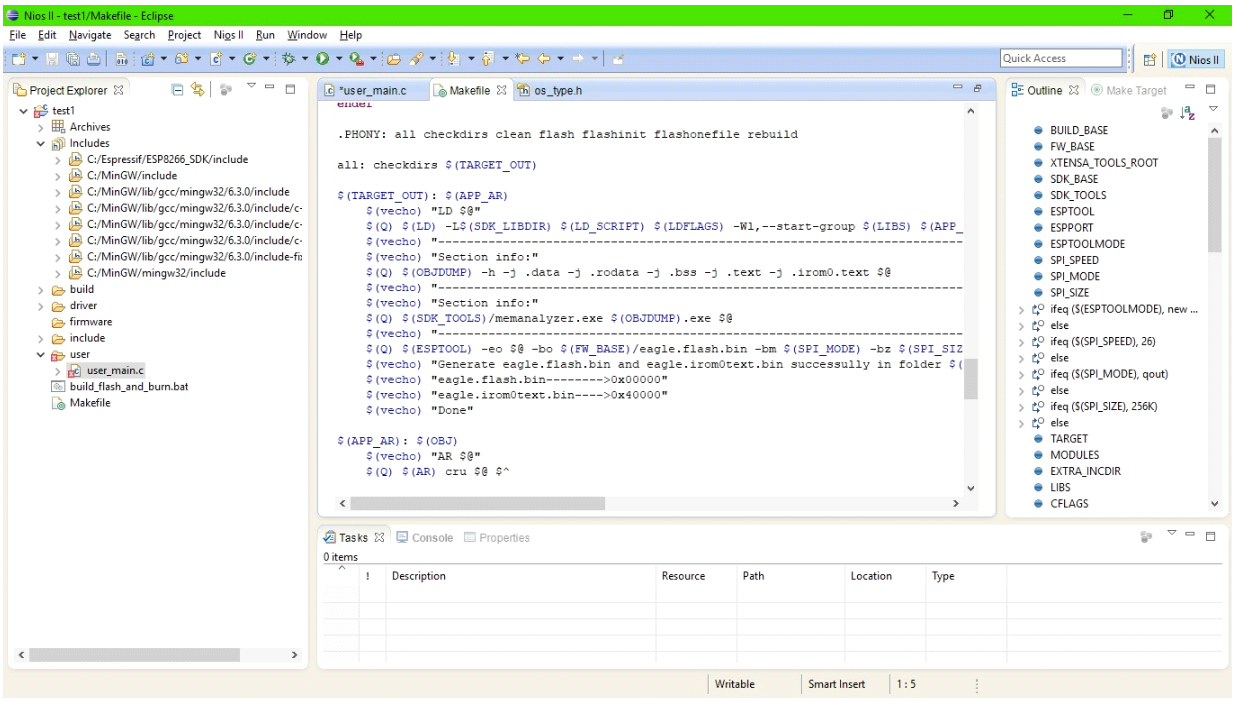

NODEMCU FIRMWARE PROGRAMMER

Operation | Config | Advanced | About | Log

```
[x] C:\Users\Anubhab\Documents\4th_sem\Capstone\   0x00000
[x] C:\Users\Anubhab\Documents\4th_sem\Capstone\   0x00000
[ ] Path of binary file                            Offset
[ ] Path of binary file                            Offset
[ ] Path of binary file                            Offset
[ ] Path of binary file                            Offset
[ ] Path of binary file                            Offset
NODEMCU TEAM                                        Ready
```

```
i2c_24xx16            5/18/2018 12:43 AM   File folder
i2c_24xx32            5/18/2018 12:43 AM   File folder
i2c_bh1750            5/18/2018 12:43 AM   File folder
i2c_bmp180            5/18/2018 12:43 AM   File folder
i2c_hd44780           5/18/2018 12:43 AM   File folder
i2c_htu21d            5/18/2018 12:43 AM   File folder
i2c_ina219            5/18/2018 12:43 AM   File folder
i2c_sht21             5/18/2018 12:43 AM   File folder
IoT_Demo              5/18/2018 12:43 AM   File folder
lpd6803               5/18/2018 12:43 AM   File folder
lwip_lib              5/18/2018 12:43 AM   File folder
nodemcu-firmware      5/18/2018 12:43 AM   File folder
nodemcu-spiffy        5/18/2018 12:43 AM   File folder
smart_config          5/18/2018 12:43 AM   File folder
sysinfo               5/18/2018 12:43 AM   File folder
wifi-ap-tcp-client    5/18/2018 12:43 AM   File folder
wifi-sniffer          5/18/2018 12:43 AM   File folder
wifi-sta-tcp-client   5/18/2018 12:43 AM   File folder
41 items
```

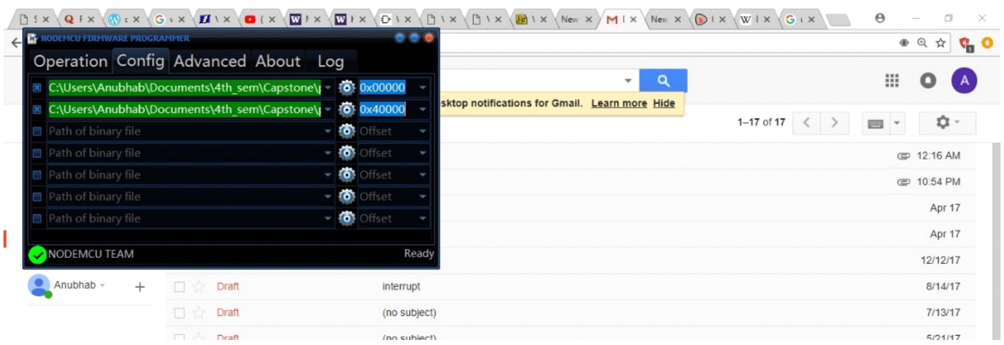
Then we discovered while looking at the makefile that the memory address for the two binary files are provided inside the makefile.


Nios II - test1/Makefile - Eclipse

```
.PHONY: all checkdirs clean flash flashinit flashonefile rebuild

all: checkdirs $(TARGET_OUT)

$(TARGET_OUT): $(APP_AR)
	$(vecho) "LD $@"
	$(Q) $(LD) -L$(SDK_LIBDIR) $(LD_SCRIPT) $(LDFLAGS) -Wl,--start-group $(LIBS) $(APP_
	$(vecho) "------------------------------------------------------------------------
	$(vecho) "Section info:"
	$(Q) $(OBJDUMP) -h -j .data -j .rodata -j .bss -j .text -j .irom0.text $@
	$(vecho) "------------------------------------------------------------------------
	$(vecho) "Section info:"
	$(Q) $(SDK_TOOLS)/memanalyzer.exe $(OBJDUMP).exe $@
	$(vecho) "------------------------------------------------------------------------
	$(Q) $(ESPTOOL) -eo $@ -bo $(FW_BASE)/eagle.flash.bin -bm $(SPI_MODE) -bz $(SPI_SIZ
	$(vecho) "Generate eagle.flash.bin and eagle.irom0text.bin successully in folder $(
	$(vecho) "eagle.flash.bin-------->0x00000"
	$(vecho) "eagle.irom0text.bin---->0x40000"
	$(vecho) "Done"

$(APP_AR): $(OBJ)
	$(vecho) "AR $@"
	$(Q) $(AR) cru $@ $^
```

So we changed the memory address of the second file to 0x40000 from 0x00000 and it solved the error. The process to do it is as follows:

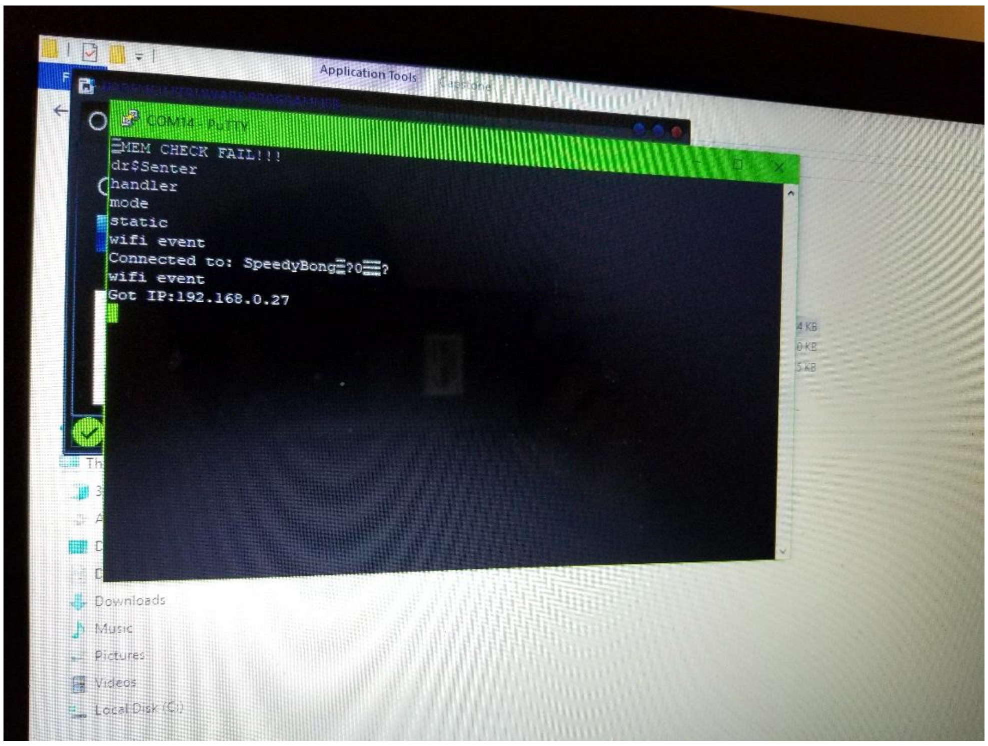
Open NODEMCU FIRMWARE PROGRAMMER -> Config -> 0x40000


NODEMCU FIRMWARE PROGRAMMER
Operation Config Advanced About Log
C:\Users\Anubhab\Documents\4th_sem\Capstone\ 0x00000
C:\Users\Anubhab\Documents\4th_sem\Capstone\ 0x40000
Path of binary file Offset
NODEMCU TEAM
Ready

After solving the error we again re-built the project in Eclipse and it got built without any errors. Then we used the ESP Flashing Tool to flash the firmware onto the ESP8266 module.

Then after opening putty and configuring it as mentioned earlier we got the desired output.

It connected perfectly to my home wifi network and we got the IP, the wifi name displayed on the terminal.



```
MEM CHECK FAIL!!!
dr$Senter
handler
mode
static
wifi event
Connected to: SpeedyBong?0?
wifi event
Got IP:192.168.0.27
```

End Time: 21:35 Hrs

Start Time: 19:30 Hrs **Date: 25/05/2018**

Our main focus for this week is to configure the ESP-8266 WiFi module in Soft AP mode in which it will act as a router or host to other devices connected to it. To configure the ESP-8266 we referred to the example line of codes provided in the ESPRESSIF .

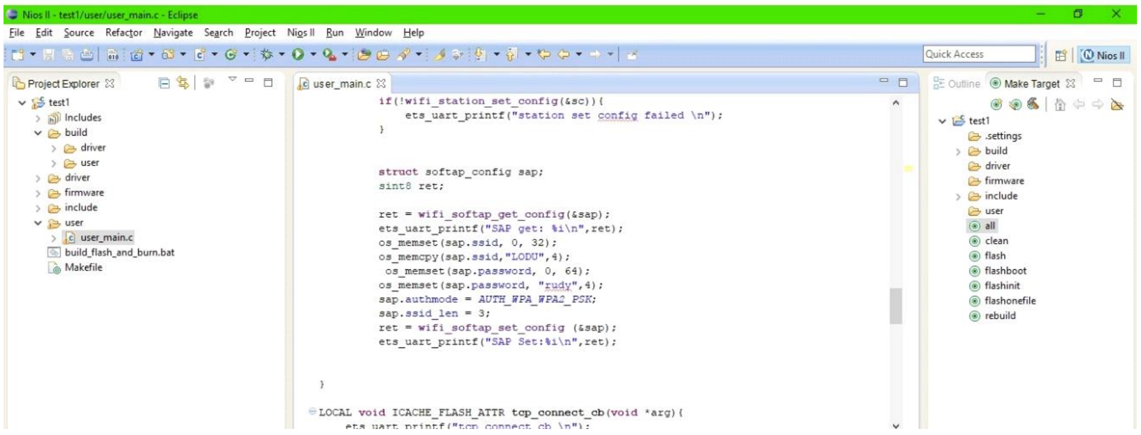
My PC ->Local Disk (C:) ->Program Files -> Espressif -> Examples -> wifi-ap-tcp-client -> user -> user_main.c

We opened this file in notepad++ because we are implementing the changes in the file we have already written to configure the ESP-8266 in station mode.

To start with we called two separate structure from the ESP library in my existing lines of code to configure the ESP-8266 in soft AP mode.

Struct ip_info softAP_staticip;

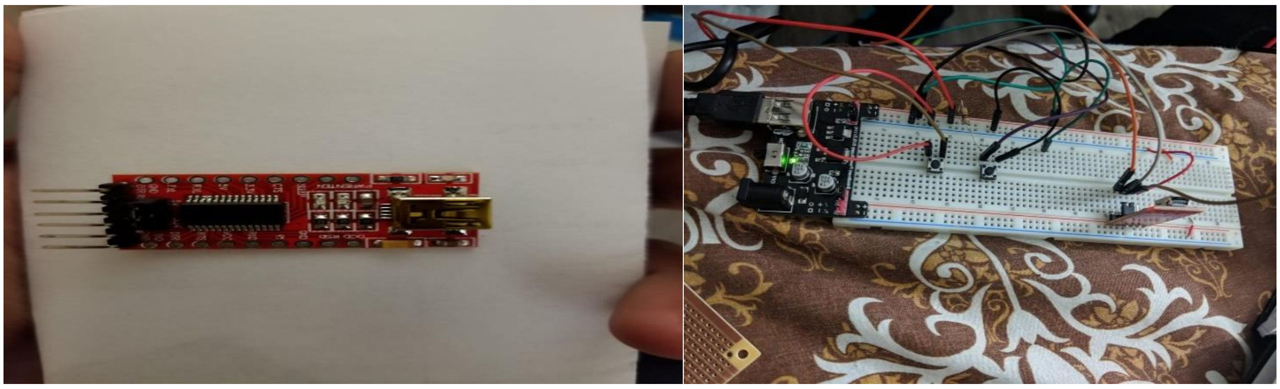
struct softap_config sap;



```
            if(!wifi_station_set_config(&sc)){
                ets_uart_printf("station set config failed \n");
            }


            struct softap_config sap;
            sint8 ret;

            ret = wifi_softap_get_config(&sap);
            ets_uart_printf("SAP get: %i\n",ret);
            os_memset(sap.ssid, 0, 32);
            os_memcpy(sap.ssid,"LODU",4);
             os_memset(sap.password, 0, 64);
            os_memset(sap.password, "rudy",4);
            sap.authmode = AUTH_WPA_WPA2_PSK;
            sap.ssid_len = 3;
            ret = wifi_softap_set_config (&sap);
            ets_uart_printf("SAP Set:%i\n",ret);


    }

LOCAL void ICACHE_FLASH_ATTR tcp_connect_cb(void *arg){
```

This structure contains the lines of codes necessary to initially configure the ESP-8266 module in soft AP mode. we have set the set the authentication mode to AUTH_WPA_WPA2_PSK where WPA_WPA2_PSK means WiFi Protected Access 2- Pre Shared Key. In this method the ESP-8266 is setup as a router without an encryption key but rather with a plain English passphrase.

Anubhab is working with code to connect the module in dual mode while Sanjay and Zankrut are working on soldering the components to the board.

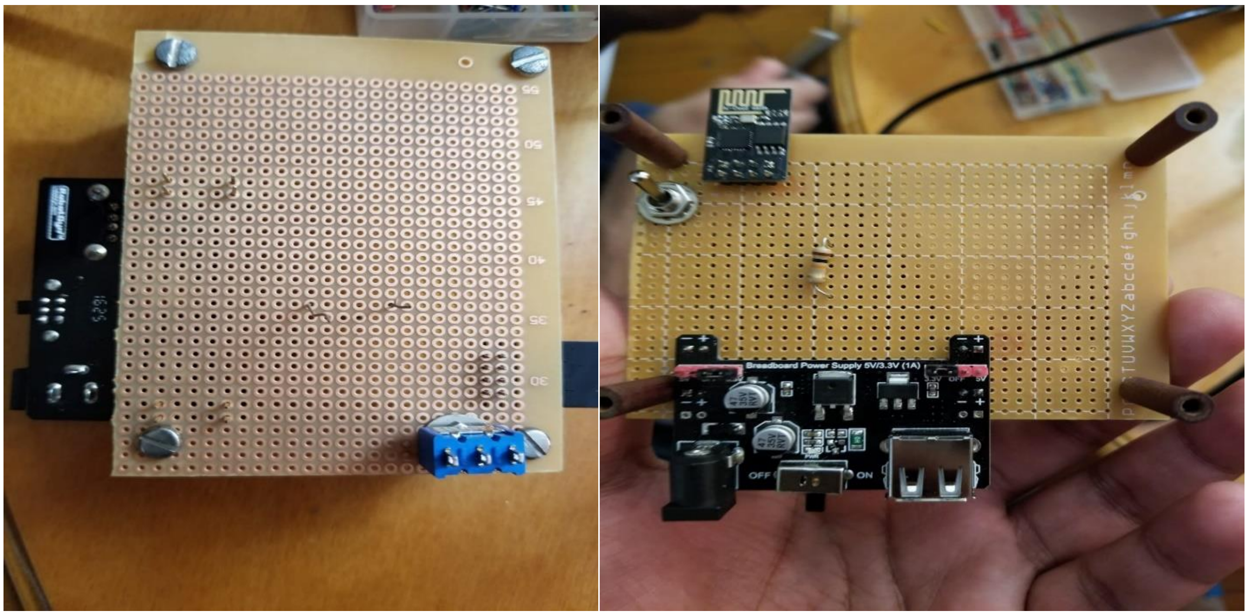


Before soldering the components on board

We got the components and equipment's for soldering the components to PCB from the tool room. The PCB that was provided by Robert was not ideal from our purpose, so we got a PCB from tool room.

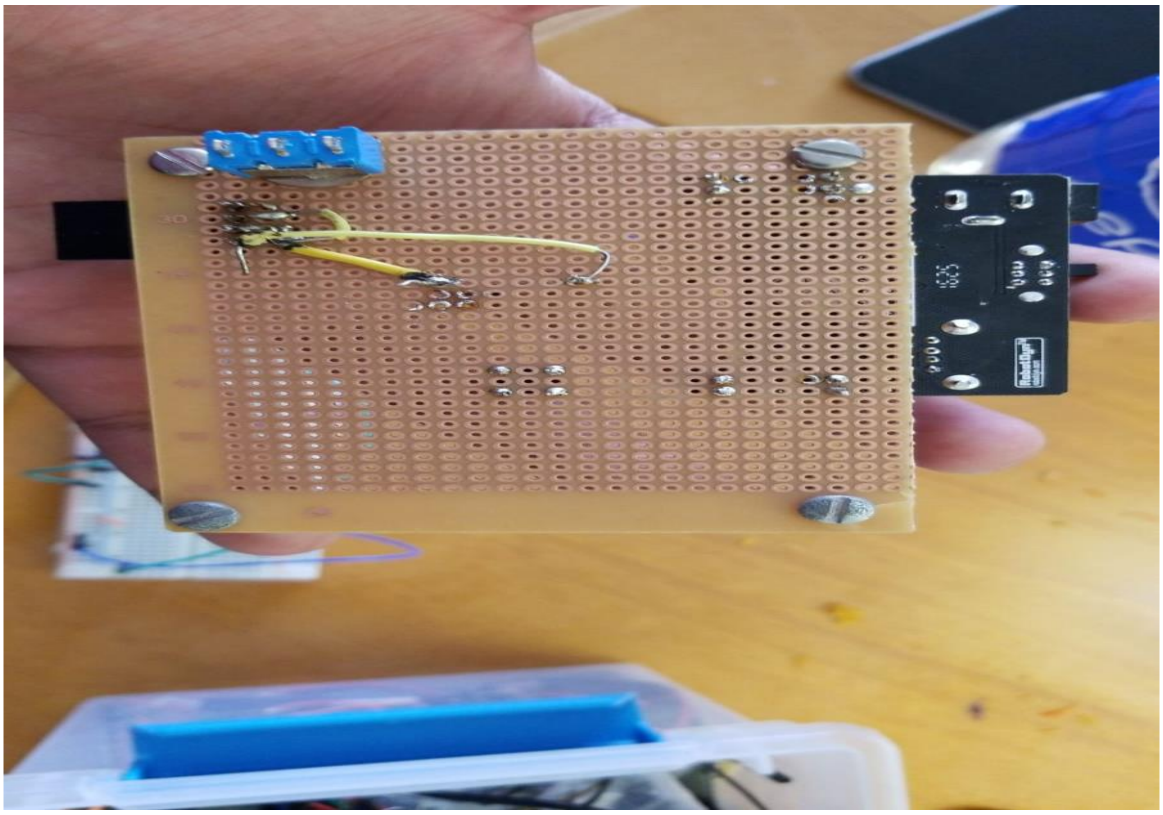
We tested the circuit with the bread board first then we started soldering the components to the PCB.

We must drill few holes to the PCB as we planned to provide the PCB board with a stand. We used the drill machine that was available in the lab.



Board to be soldered

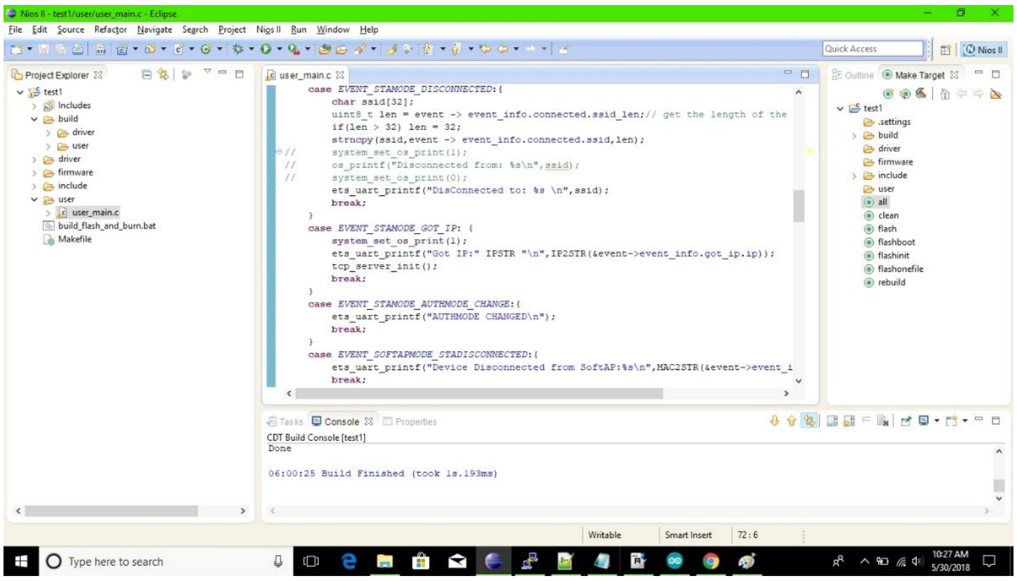
We have soldered all the components and started testing



End Time: 00:45 Hrs **Date: 26/05/2018**

Then we created two more events in the previously declared wifi_event_cb() function to display if any device is connected or disconnected to the ESP module. The two events that created are

- EVENT_SOFTAPMODE_STADISCONNECTED to display if the device is disconnected
- EVENT_SOFTAPMODE_STACONNECTED to display if a device is connected.

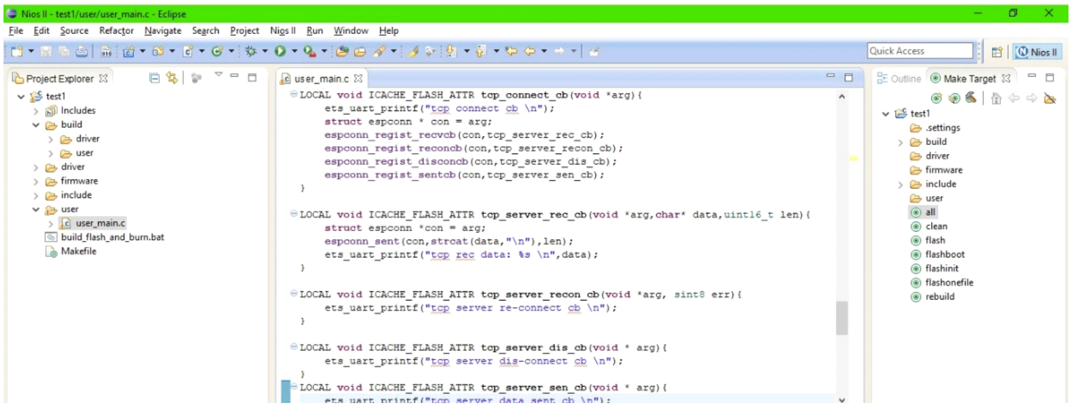


```
    case EVENT_STAMODE_DISCONNECTED:{
        char ssid[32];
        uint8_t len = event -> event_info.connected.ssid_len;// get the length of the
        if(len > 32) len = 32;
        strncpy(ssid,event -> event_info.connected.ssid,len);
//      system_set_os_print(1);
//      os_printf("Disconnected from: %s\n",ssid);
//      system_set_os_print(0);
        ets_uart_printf("DisConnected to: %s \n",ssid);
        break;
    }
    case EVENT_STAMODE_GOT_IP: {
        system_set_os_print(1);
        ets_uart_printf("Got IP:" IPSTR "\n",IP2STR(&event->event_info.got_ip.ip));
        tcp_server_init();
        break;
    }
    case EVENT_STAMODE_AUTHMODE_CHANGE:{
        ets_uart_printf("AUTHMODE CHANGED\n");
        break;
    }
    case EVENT_SOFTAPMODE_STADISCONNECTED:{
        ets_uart_printf("Device Disconnected from SoftAP:%s\n",MAC2STR(&event->event_i
        break;
```

```
CDT Build Console [test1]
Done

06:00:25 Build Finished (took 1s.193ms)
```

After this we started writing the code to configure the TCP_IP port of the ESP-8266. The ESP-8266 has five TCP_IP port for communication between the host and the client. It has got five ports for communication out of which I will be using four ports for the time being. We wrote the local function LOCAL void ICACHE_FLASH_ATTR tcp_server_init() in which we setup the TCP_IP as following:

- tcp_conn.type = ESPCONN_TCP;
- tcp_conn.state = ESPCONN_MODE;
- tcp_conn,proto.tcp = &esptcp;
- tcp_conn.proto.tcp->local_port = 2345;



```
LOCAL void ICACHE_FLASH_ATTR tcp_connect_cb(void *arg){
    ets_uart_printf("tcp connect cb \n");
    struct espconn * con = arg;
    espconn_regist_recvcb(con,tcp_server_rec_cb);
    espconn_regist_reconcb(con,tcp_server_recon_cb);
    espconn_regist_disconcb(con,tcp_server_dis_cb);
    espconn_regist_sentcb(con,tcp_server_sen_cb);
}

LOCAL void ICACHE_FLASH_ATTR tcp_server_rec_cb(void *arg,char* data,uint16_t len){
    struct espconn *con = arg;
    espconn_sent(con,strcat(data,"\n"),len);
    ets_uart_printf("tcp rec data: %s \n",data);
}

LOCAL void ICACHE_FLASH_ATTR tcp_server_recon_cb(void *arg, sint8 err){
    ets_uart_printf("tcp server re-connect cb \n");
}

LOCAL void ICACHE_FLASH_ATTR tcp_server_dis_cb(void * arg){
    ets_uart_printf("tcp server dis-connect cb \n");
}
LOCAL void ICACHE_FLASH_ATTR tcp_server_sen_cb(void * arg){
    ets_uart_printf("tcp server data sent cb \n");
```

After this we compiled the entire code in ECLIPSE KEPPLER IDE and to our surprise it compiled with two minor errors which was rectified in a couple of minutes.

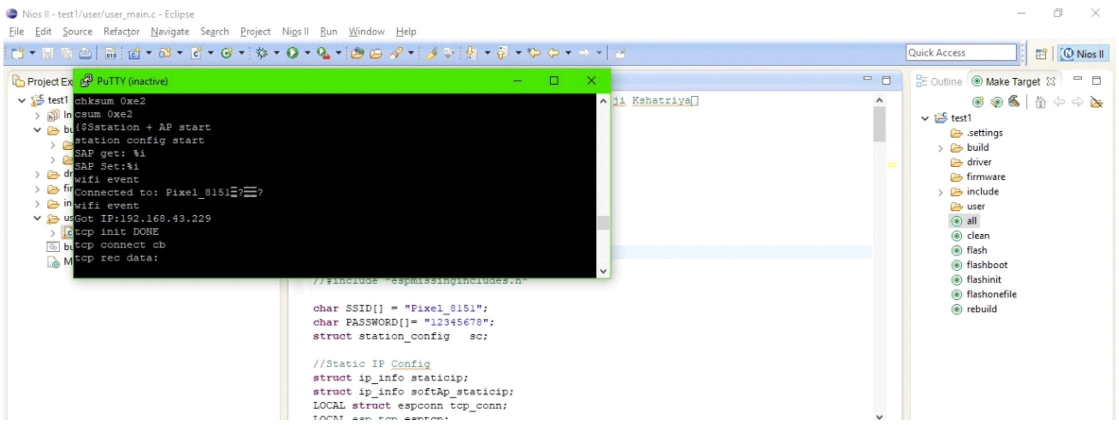
The first error that popped up missing “;” or “}”. So we checked for the closing of braces and we found that we have one missing and the error was solved.

The next one was due to not including the espconn.h header file

Then we again open the ESP-FLASHING TOOL and added the two binary files created during the compilation of the code in ECLIPSE.

After flashing of the firmware on the ESP module we launched putty in windows. Now for the IP setup we declared the ip as 192.198.43.229 and the port as 2345.

Then we restarted the ESP-module again and on the terminal window we saw that it showed that it was ready to be connected to and when checking for wifi devicces on mobile and laptop the ESP-8266 was available as an open wifi network.


```
chksum 0xe2
csum 0xe2
{$Sstation + AP start
station config start
SAP get: %i
SAP Set:%i
wifi event
Connected to: Pixel_8151☰?☰?
wifi event
Got IP:192.168.43.229
tcp init DONE
tcp connect cb
tcp rec data:
```

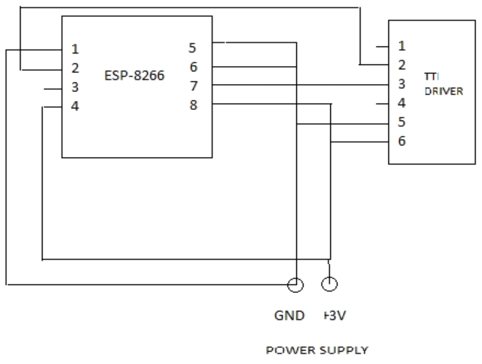
```
//#include "espmissingincludes.h"

char SSID[] = "Pixel_8151";
char PASSWORD[]= "12345678";
struct station_config   sc;

//Static IP Config
struct ip_info staticip;
struct ip_info softAp_staticip;
LOCAL struct espconn tcp_conn;
```

On the hardware part we are still waiting to design a circuit on the pcb for flashing the firmware on the NODEMCU that is ESP 8266.

End Time: 17:30 Hrs.

Start Time: 20:16 **Date: 30/05/2018**

Today in the lab I am helping out Zankrut in completing the soldering for the firmware flashing circuit . The firmware flashing circuit diagram is provided below.



<u>CIRCUIT DIAGRAM FOR FLASHING CIRCUIT</u>

To build the circuit we used female headers and soldered them on the zero PCB so that we can easily place and remove the ESP-8266 module and TTL-Driver. To solder the components we borrowed out the soldering station from the tool room.

We have completed building the flashing during the class hours. Now we are going to meet Keiran and collect the parts that we have ordered.

End Time: 16:00 Hrs

Start Time: 13:05 Hrs **Date: 15/06/2018**

While we were suffering from viral fever my capstone group mate Sanjay was working on the programming part to try and enable sending and receiving data when configuring one ESP 8266 module in dual mode (Station Mode/ SoftAP Mode). In this configuration the ESP-8266 will act as a client in Station mode as well as a Host to other devices in SoftAP Mode. As he was not able to figure it out how to configure the ESP8266 in Dual Mode I am now working on the software.

First we changed the name of the ESP-8266 from AI-THINKER to ESP8266 and setup a password as well for other devices to connect to it.

```
os_memset(config.ssid, 0, 32);
os_memset(config.password, 0, 64);
os_memcpy(config.ssid, "ESP", 3);
os_memcpy(config.password, "12345678", 8);
```

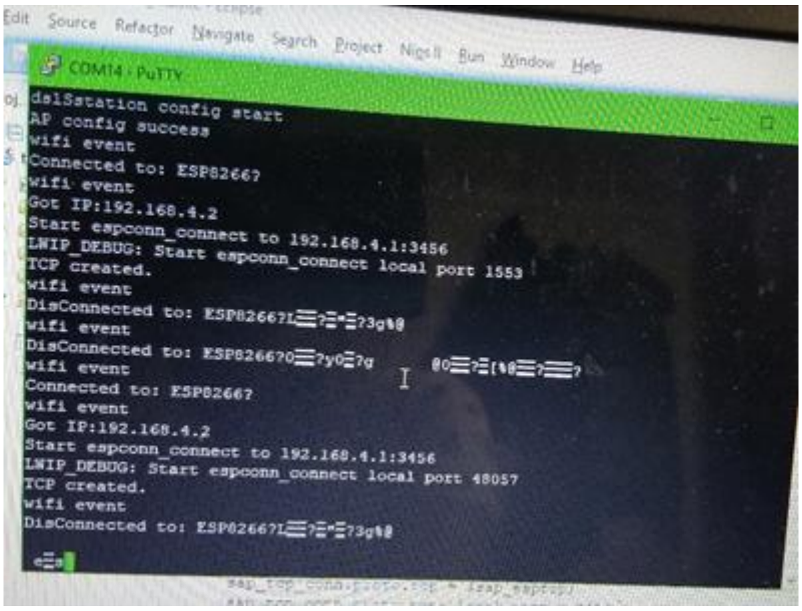
We also defined how many stations can connect to ESP8266 at SoftAP Mode at maximum.

```
 config.max_connection = 4;
```

We am now writing the code to send data to the device connected to the ESP-8266 module using its SoftAP mode. This would enable me to send data to the device. We have already written the code before to receive data in the softAP mode.

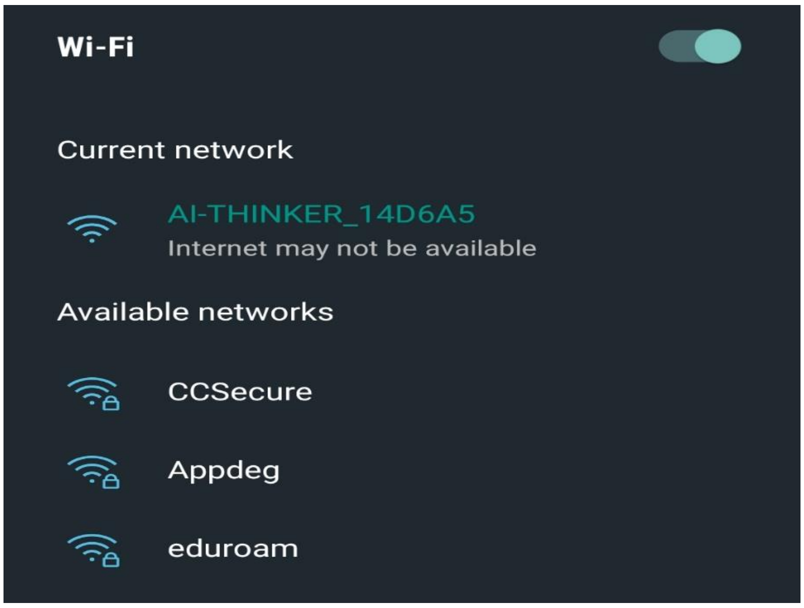
```
LOCAL void ICACHE_FLASH_ATTR tcp_server_rec_cb(void *arg,char* data,uint16_t len){
struct espconn *con = arg;
espconn_sent(con,strcat(data,"\n"),len);
//send this data to sap tcpserver
ets_uart_printf("tcp rec data: %s \n",data); // this will go to sap send data
}
```

With these lines of code added to my previous code we can now configure one ESP module in both Station and SoftAP mode at the same time as well as send and receive data and view it on the terminal.



```
dslSstation config start
AP config success
wifi event
Connected to: ESP82667
wifi event
Got IP:192.168.4.2
Start espconn_connect to 192.168.4.1:3456
LWIP_DEBUG: Start espconn_connect local port 1553
TCP created.
wifi event
DisConnected to: ESP82667L☰?☰•☰?3g%@
wifi event
DisConnected to: ESP82667O☰?yO☰?g          @O☰?☰(%@☰?☰?
wifi event
Connected to: ESP82667
wifi event
Got IP:192.168.4.2
Start espconn_connect to 192.168.4.1:3456
LWIP_DEBUG: Start espconn_connect local port 48057
TCP created.
wifi event
DisConnected to: ESP82667L☰?☰•☰?3g%@

e☰s
```

We were successfully able to run the program in dual mode



Wi-Fi

Current network

AI-THINKER_14D6A5
Internet may not be available

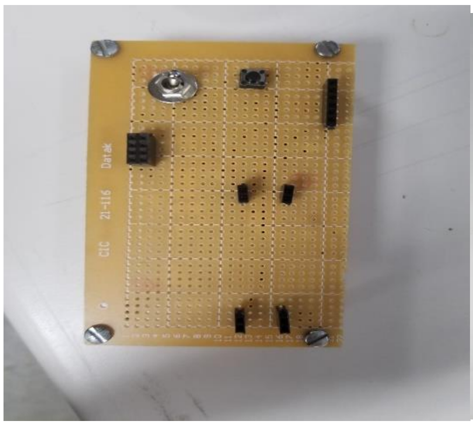
Available networks

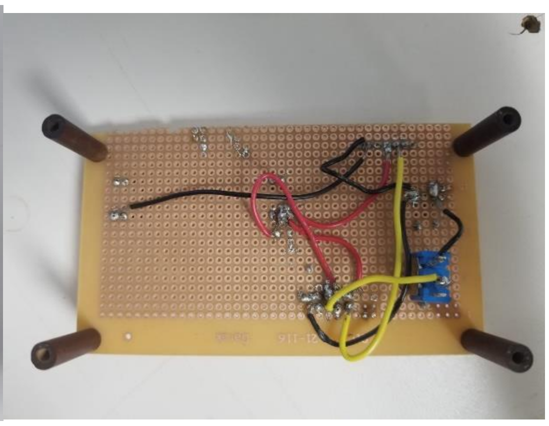
CCSecure

Appdeg

eduroam

After that we started working on our circuit board for ESP8266 as we were not satisfied with previous circuit.





End Time: 21:48

Start Time: 12:15 Hrs **Date: 20/06/2018**

Today we are having a meeting about our progress till date and the work that is still left for us to complete. We asked Sanjay to take the ESP8266 module outside the classroom and verify its range. On its datasheet it is mentioned approximately 400 metres but to our surprise we found that there is problem in connectivity even when the ESP module is taken to the ground floor. This was happening due to the presence of concrete walls which was disrupting the line of sight and hindering the connectivity. When we took our entire setup to the open field behind our campus we found that with a proper line of sight we were not able to get anywhere close to mentioned connectivity range in the datasheet. We got connectivity at a maximum range of around 250m.
Thereafter we decided to connect external antennas to the wifi-module to increase its range. Sanjay took up the responsibility of finding how to connect the external antennas that we have ordered to ESP8266 module.\

End Time: 15:35 Hrs

Start Time: 20:10 Hrs **Date: 20/06/2018**

We are now working on the code on how to build a wireless bridge between two ESP8266 modules. For that we have flashed the previous firmware that we developed on one ESP8266 module. Now we have to write the firmware for the second ESP8266 module which again has to be configured in both Station and SoftAP mode.

We used the code that we wrote previously and started making changes to it.
The first step that we did was to change the SSID and password of the Station Mode. By doing this it will now connect to the other ESP8266 module and not the home wifi network.

```
char SSID[] = "ESP8266";
char PASSWORD[]= "12345678";
struct station_config  sc;
```

Then we wrote the code to establish a connection and obtain the MAC Address from the first module. It is very necessary to obtain the MAC Address because without the MAC Address it won't be possible to successfully establish a link between the two modules even after knowing the IP address of the first module.

```
    _STAMODE_GOT_IP: {
    system_set_os_print(1);
    ets_uart_printf("Got IP:" IPSTR "\n",IP2STR(&event->event_info.got_ip.ip));
    tcp_client_init();
    SoftAP:%s\n",MAC2STR(&event->event_info.sta_disconnected.mac));
```

We are now writing the code to configure the TCP/IP port which will be used for communication.

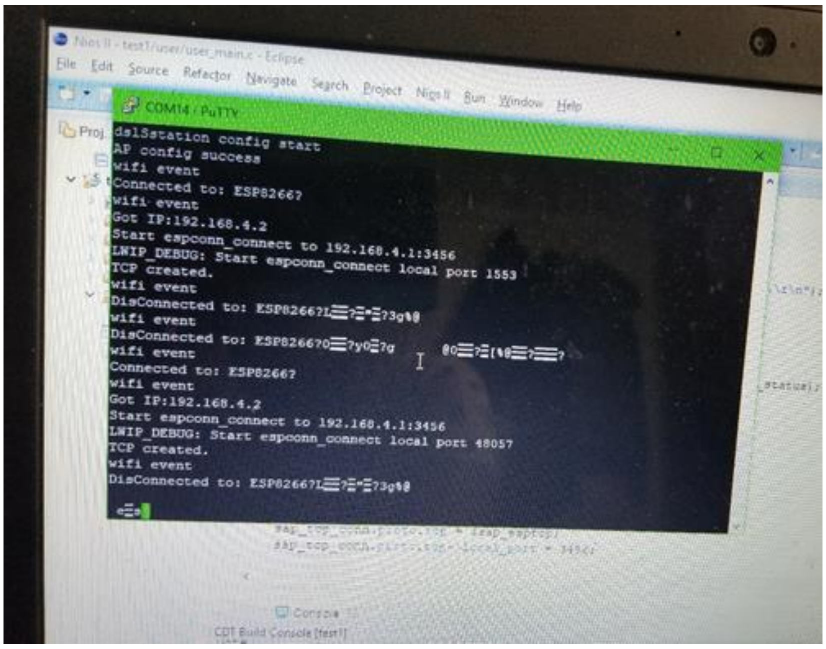
```
    LOCAL void ICACHE_FLASH_ATTR sap_tcp_server_init(){
    ets_uart_printf("SAP tcp init START \n");
    sap_tcp_conn.type = ESPCONN_TCP;
    sap_tcp_conn.state = ESPCONN_NONE;
    sap_tcp_conn.proto.tcp = &sap_esptcp;
    sap_tcp_conn.proto.tcp->local_port = 3456;

    espconn_regist_connectcb(&sap_tcp_conn,sap_tcp_connect_cb);
    espconn_regist_disconcb(&sap_tcp_conn,sap_tcp_disconnect_cb);
    espconn_regist_reconcb(&sap_tcp_conn,sap_tcp_reconnect_cb);

    int ret = espconn_accept(&sap_tcp_conn);
    espconn_regist_time(&sap_tcp_conn,7200,0);
    ets_uart_printf("SAP tcp init DONE \n");
}
```

We are using port 3456 of ESP8266 as the TCP/IP port in the softAP mode.
Now configuring the UART

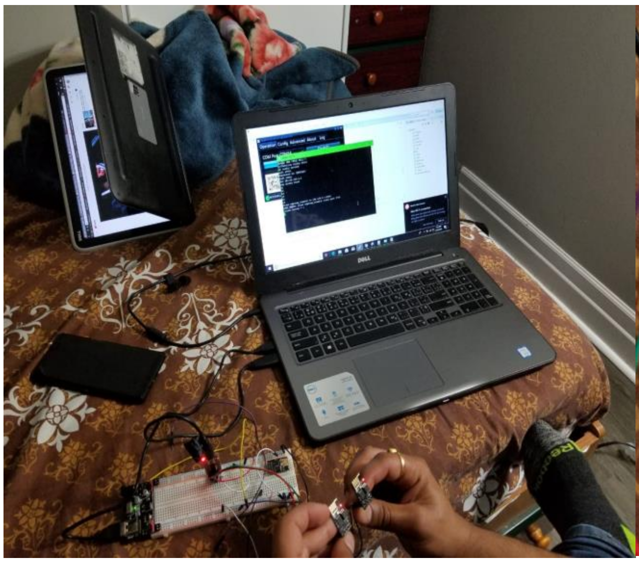
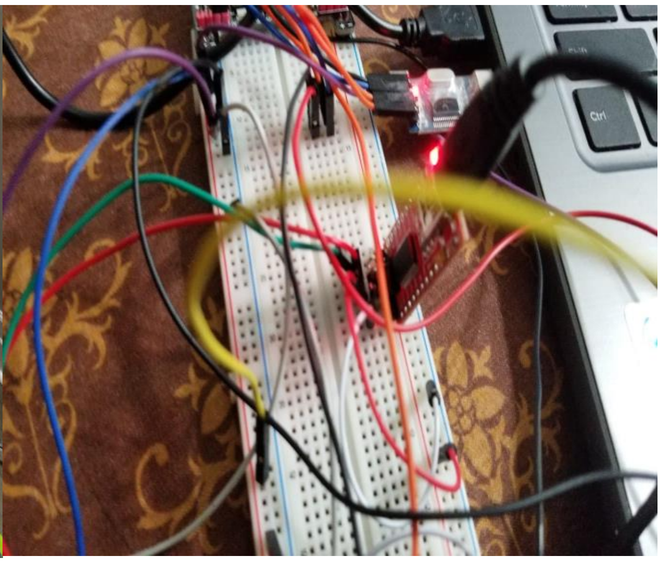
```
uart_init(BIT_RATE_115200, BIT_RATE_115200);
```

Here 115200 is the Baudrate. Baudrate is a common measure of the speed of communication over a data channel.

We have compiled the code and it compiles with no errors. Now using the ESP8266 flashing kit to flash the firmware onto the second ESP8266 module.


Nios II - test1/user/user_main.c - Eclipse
File Edit Source Refactor Navigate Search Project Nios II Run Window Help
COM14 - PuTTY
Proj
Console
CDT Build Console [test1]

```
ds1Sstation config start
AP config success
wifi event
Connected to: ESP8266?
wifi event
Got IP:192.168.4.2
Start espconn_connect to 192.168.4.1:3456
LWIP_DEBUG: Start espconn_connect local port 1553
TCP created.
wifi event
DisConnected to: ESP8266?L[illegible]
wifi event
DisConnected to: ESP8266?0[illegible]
wifi event
Connected to: ESP8266?
wifi event
Got IP:192.168.4.2
Start espconn_connect to 192.168.4.1:3456
LWIP_DEBUG: Start espconn_connect local port 48057
TCP created.
wifi event
DisConnected to: ESP8266?L[illegible]
```

Though we being able to connect to ESP modules and build the bridge we are not being able to successfully send and receive data over the bridge.

We have connected my mobile phone to the second ESP8266 module while my first ESP866 module is connected to my home WI-FI network . Whenever we are sending data from my mobile phone using TELNET application for android the data is getting transferred and is being printed on the terminal window of my laptop but whenever we are trying to send any data from my laptop to mobile phone the data is not getting transmitted.

This implies that we have been partially successful in building the bridge and not fully successful.

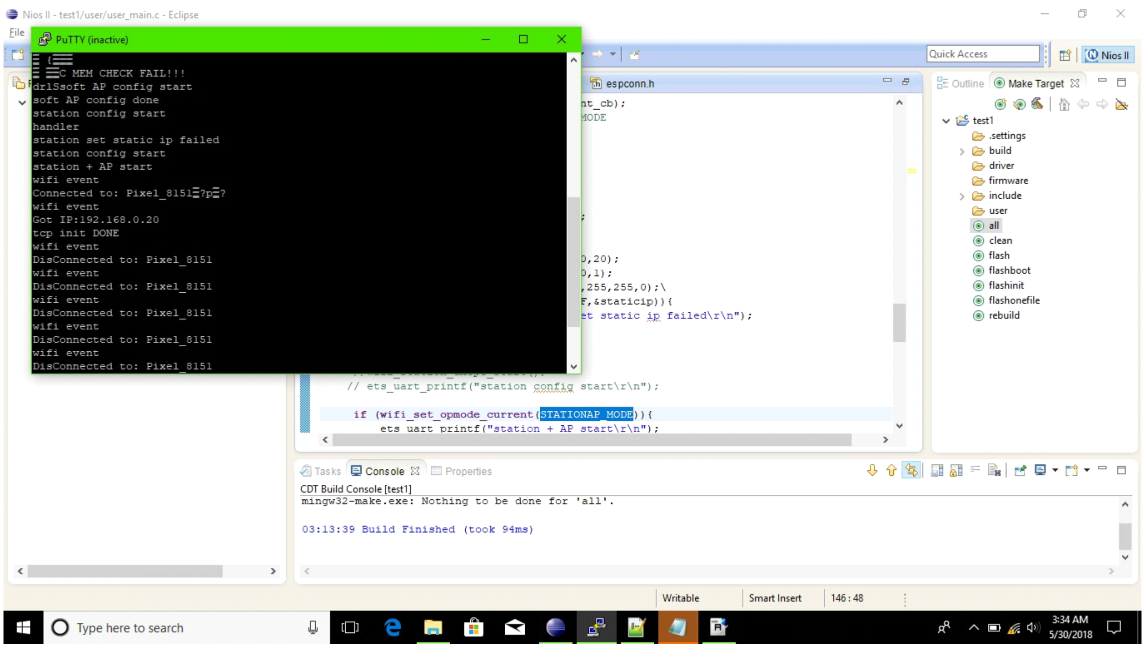
End Time: 01:50 Hrs

Start Time: 21:45 Hrs **Date: 21/06/2018**

We are starting to look forward to fixing the issues with bidirectional communication. While looking at the puuty terminal we found that the **espconn_sent()** is returning a value of -12.So we started searching for what the value -12 returned by the espconn_sent() function is. we went through the espconn.h file and found that -12 has been defined there as an error - **#define** ESPCONN_ARG -12 /* Illegal argument.*/. We made some changes in my code by trying to harcode my IP instead of using the dhcps_start() function to assign a floating IP to the esp module. So we physically harcoded the AP of the first esp module with a static IP. We used the following lines to implement the static IP for the first esp module.

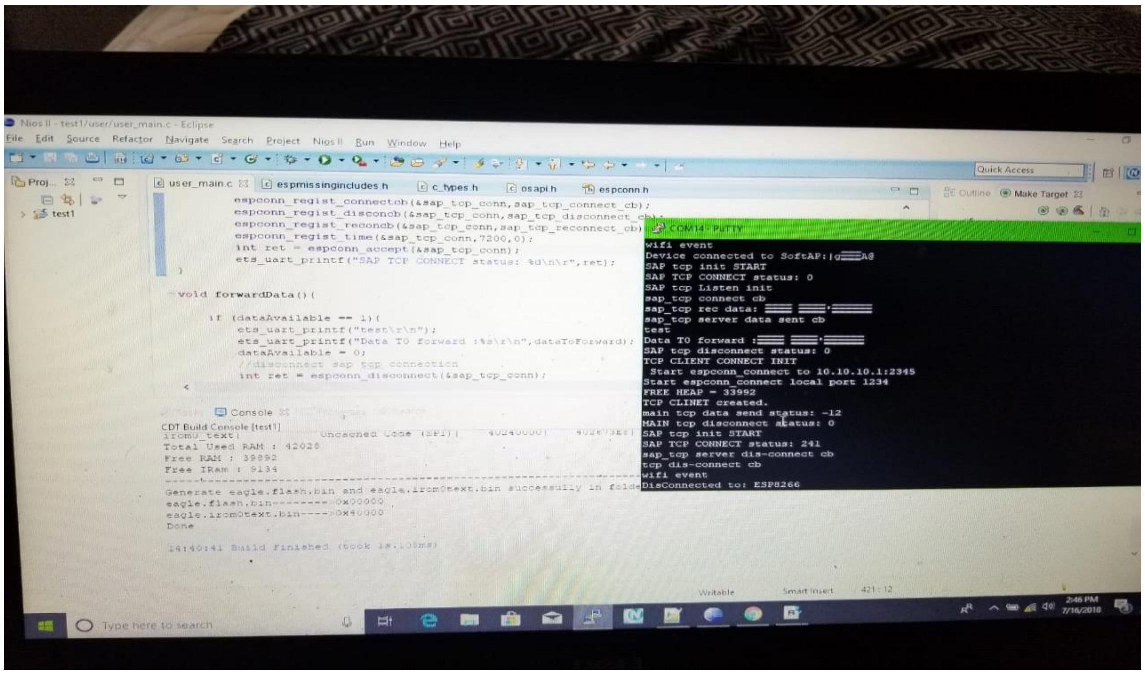
```
wifi_softap_dhcps_stop();

IP4_ADDR(&ipinfo.ip, 10, 10, 10, 1);

IP4_ADDR(&ipinfo.gw, 10, 10, 10, 1);

IP4_ADDR(&ipinfo.netmask, 255, 255, 255, 0);

int ret  = wifi_set_ip_info(SOFTAP_IF, &ipinfo);
```

Then we compiled the code and the code compiled without any compilation error but after flashing the code on the esp module which acts as the server or rather connects to the second esp module as a host, We find that although the IP address has changed and still the error persists.



```
 {
C MEM CHECK FAIL!!!
drlSsoft AP config start
soft AP config done
station config start
handler
station set static ip failed
station config start
station + AP start
wifi event
Connected to: Pixel_8151?p?
wifi event
Got IP:192.168.0.20
tcp init DONE
wifi event
DisConnected to: Pixel_8151
wifi event
DisConnected to: Pixel_8151
wifi event
DisConnected to: Pixel_8151
wifi event
DisConnected to: Pixel_8151
wifi event
DisConnected to: Pixel_8151
```

```
mingw32-make.exe: Nothing to be done for 'all'.

03:13:39 Build Finished (took 94ms)
```



```
wifi event
Device connected to SoftAP:|gA@
SAP tcp init START
SAP TCP CONNECT status: 0
SAP tcp Listen init
sap_tcp connect cb
sap_tcp rec data:
sap_tcp server data sent cb
test
Data TO forward :
SAP tcp disconnect status: 0
TCP CLIENT CONNECT INIT
 Start espconn_connect to 10.10.10.1:2345
Start espconn_connect local port 1234
FREE HEAP = 33992
TCP CLINET created.
main tcp data send status: -12
MAIN tcp disconnect status: 0
SAP tcp init START
SAP TCP CONNECT status: 241
sap_tcp server dis-connect cb
tcp dis-connect cb
wifi event
DisConnected to: ESP8266
```

Our initial gut feeling was that, the error was occuring due to conflict in IP, as the IP was being set by the router using the dhcps_start() function , but now with the implementation of the static IP that problem shouldnt exist. So now we realize that error is not occuring due to the conflict of IP's.

End Time: 04:15 Hrs **Date: 21/06/2078**

Start Time: 18:00 Hrs **Date: 23/06/2018**

While looking at the espconn.h file more closely and also referring to the espconn.h file in the git repository https://github.com/espressif/esp8266-dual-cloud/blob/master/ESP8266_RTOS_SDK/include/espressif/espconn.h we realized that the error is occurring because the esp cant find the corresponding network for transmission according to the structure espconn.

We looked at the espconn structure in espmissingincludes.h and found that it has two parameters "type" and "state" where type refers to the type of connection that is whether UDP or TCP protocol is used for connecting and the state of the esp module.

Now we are having no clue why this error is coming up because it seems to me that the arguments we are passing in espconn_sent() function matches with the espconn_struct.

End Time: 22:50 Hrs

Start Time: 19:42 Hrs **Date: 27/06/2018**

To find out where the error was occuring we implemented a switch-case statement inside the tcp_client init(). We used the value returned by the esp_connect() to switch between the cases.

```
int espcon_status = espconn_connect(&TcpSTAClientConn);
switch(espcon_status)
{
        case ESPCONN_OK:
        ets_uart_printf("TCP CLINET created.\r\n");
        ets_uart_printf ("%s\n\r",data);
        ets_uart_printf("%d\n\r", &TcpSTAClientConn);
        int strLen=0;
        strLen=strlen(data);
        int a =espconn_sent(&TcpSTAClientConn,data,strlen(data));
        ets_uart_printf("main tcp data send status: %d.\r\n",a);
        tcpClientFalg = 1;
        break;

        case ESPCONN_RTE:
        ets_uart_printf("Error connection, routing problem.\r\n        \r");
        break;

        case ESPCONN_TIMEOUT:
        ets_uart_printf("Error connection, timeout.\r\n\r");
        break;

        case ESPCONN_MEM:
        ets_uart_printf("OUT OF MEMORY.\r\n\r");
        break;

        case ESPCONN_INPROGRESS:
        ets_uart_printf("in progress.\r\n\r");
        break;

        case ESPCONN_ABRT:
        ets_uart_printf("ABORT.\r\n\r");
        break;

        case ESPCONN_RST:
        ets_uart_printf("RESET.\r\n\r");
        break;

        case ESPCONN_CLSD:
        ets_uart_printf("CLOSED \r\n\r");
        break;

        case ESPCONN_CONN:
        ets_uart_printf("NOT CONNECTED\r\n\r");
        break;

        case ESPCONN_ARG:
        ets_uart_printf("ILLEGAL ARGUMENT.\r\n\r");
        break;

        case ESPCONN_ISCONN:
        ets_uart_printf("ALREADY CONNECTED.\r\n\r");
        break;

        default:
        ets_uart_printf("Connection error: %d\r\n\r", (int)    espcon_status);
        }
```

We did this to navigate through all the possible errors mentioned in espconn.h file.The errors listed in espconn.h are as follows

```
/* Definitions for error constants. */


#define ESPCONN_OK          0    /* No error, everything OK. */
#define ESPCONN_MEM        -1    /* Out of memory error.     */
#define ESPCONN_TIMEOUT    -3    /* Timeout.                 */
#define ESPCONN_RTE        -4    /* Routing problem.         */
#define ESPCONN_INPROGRESS  -5   /* Operation in progress    */


#define ESPCONN_ABRT       -8    /* Connection aborted.      */
#define ESPCONN_RST        -9    /* Connection reset.        */
#define ESPCONN_CLSD      -10    /* Connection closed.       */
#define ESPCONN_CONN      -11    /* Not connected.           */


#define ESPCONN_ARG       -12    /* Illegal argument.        */
#define ESPCONN_ISCONN    -15    /* Already connected.       */
```

After compilation of the code and flashing it on the esp module we find that Illegal Argument error still exists but the esp_connect function returns a value of -1 or espconn_ok that is it returns a ok status. So even after returning a OK status that error still exists.

END TIME: 03:40 Hrs **Date: 28/06/2018**

Start Time: 18:05 Hrs **Date: 02/07/2018**

We are being unable to find any proper solutions to the error. We have google for this error but still of no avail. These are the links we have visited:

https://www.esp8266.com/viewtopic.php?f=160 HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965" HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160 HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965"& HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965"t=17965" HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965"& HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965" HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160 HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965"& HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965"t=17965" HYPERLINK "https://www.esp8266.com/viewtopic.php?f=160&t=17965"t=17965

https://github.com/espressif/esp8266-dual-cloud/.../ESP8266.../espressif/espconn.h

https://www.espressif.com/sites/.../2c-esp8266_non_os_sdk_api_reference_en.pdf

After referring to the documentation of esp8266 provided by its developers Espressif we tried implenting remote_ip and remote_port as it is mentioned for espconn_sent_callback function definition.

4.1.6. espconn_sent_callback()

Function Callback after the data are sent.

Prototype void espconn_sent_callback (void *arg)

Parameter

void *arg: pointer corresponding structure espconn. This pointer may be different in different callbacks, please don't use this pointer directly to distinguish one from another in multiple connections, use remote_ip and remote_port in espconn instead.

Return none

So we implemented the changes accordingly

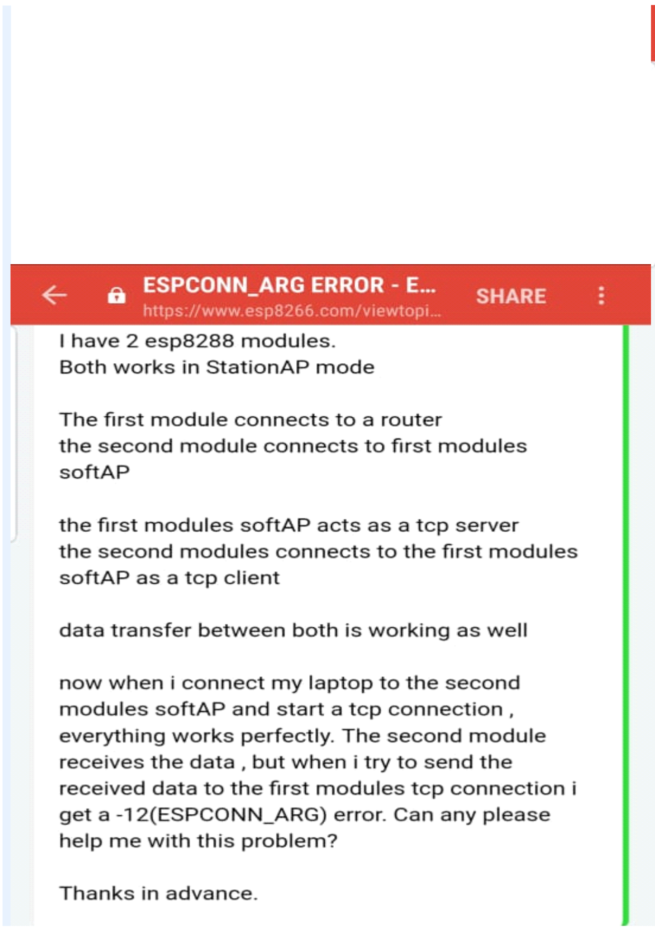
```
os_memcpy(TcpSTAClientConn.proto.tcp->remote_ip, &ip, 4);

TcpSTAClientConn.proto.tcp->remote_port = 2345;
```

We compiled my code and like before it compiled without any error **but after flashing** it on esp8266 the same error still existed.

End Time: 05:30 Hrs **Date: 03/07/2018**

Start Time: 20:05 Hrs **Date: 07/07/2017**

As we are having no clue on how to solve the error we posted this on ESP8266 forum



ESPCONN_ARG ERROR - E...
https://www.esp8266.com/viewtopi...
SHARE

I have 2 esp8288 modules.
Both works in StationAP mode

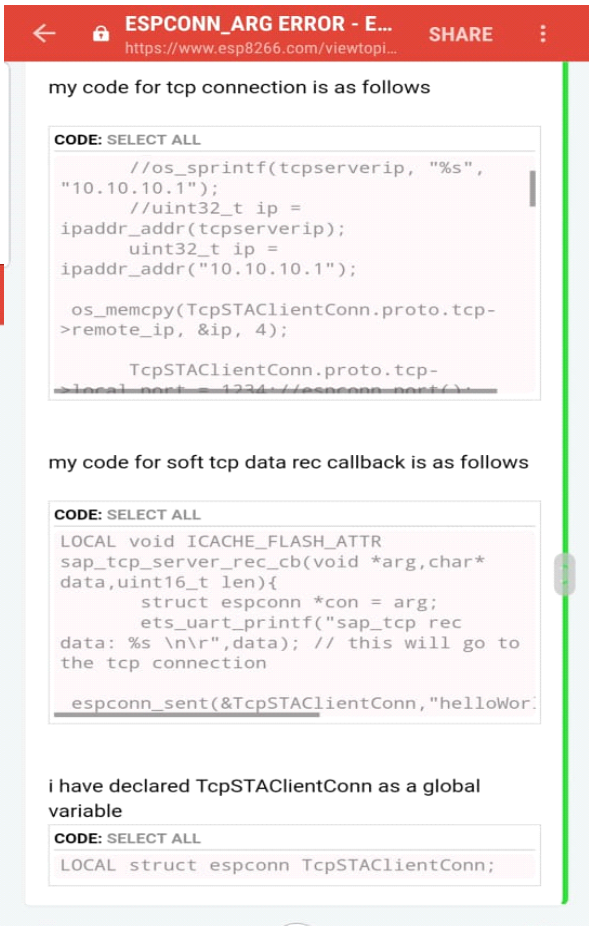
The first module connects to a router
the second module connects to first modules softAP

the first modules softAP acts as a tcp server
the second modules connects to the first modules softAP as a tcp client

data transfer between both is working as well

now when i connect my laptop to the second modules softAP and start a tcp connection , everything works perfectly. The second module receives the data , but when i try to send the received data to the first modules tcp connection i get a -12(ESPCONN_ARG) error. Can any please help me with this problem?

Thanks in advance.



ESPCONN_ARG ERROR - E...
https://www.esp8266.com/viewtopi...
SHARE

my code for tcp connection is as follows

CODE: SELECT ALL

```
      //os_sprintf(tcpserverip, "%s",
"10.10.10.1");
      //uint32_t ip =
ipaddr_addr(tcpserverip);
      uint32_t ip =
ipaddr_addr("10.10.10.1");

 os_memcpy(TcpSTAClientConn.proto.tcp-
>remote_ip, &ip, 4);

      TcpSTAClientConn.proto.tcp-
>local_port = 1234;//espconn_port();
```

my code for soft tcp data rec callback is as follows

CODE: SELECT ALL

```
LOCAL void ICACHE_FLASH_ATTR
sap_tcp_server_rec_cb(void *arg,char*
data,uint16_t len){
       struct espconn *con = arg;
       ets_uart_printf("sap_tcp rec
data: %s \n\r",data); // this will go to
the tcp connection

 espconn_sent(&TcpSTAClientConn,"helloWor
```

i have declared TcpSTAClientConn as a global variable

CODE: SELECT ALL

```
LOCAL struct espconn TcpSTAClientConn;
```

So now we are waiting for some insight on how to solve the error.

We also have gone through the document provided by Espressif to learn more about the espconn struct.The espconn struct is as follows.

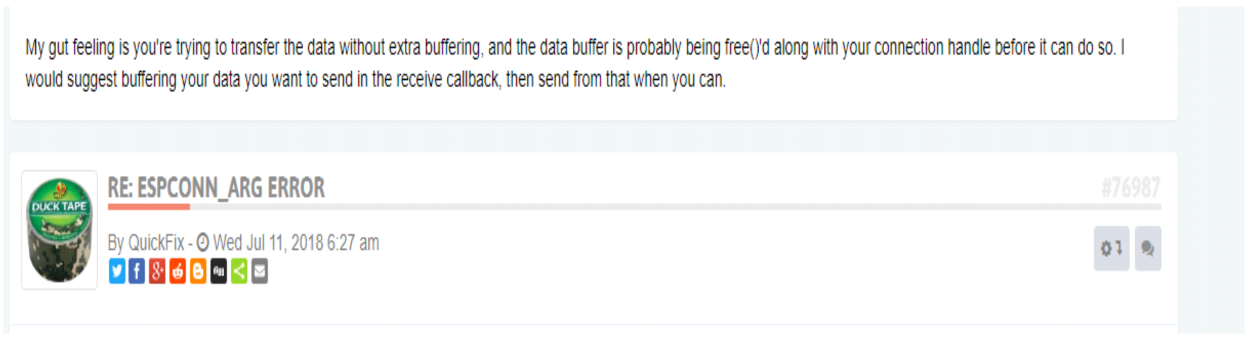
```
struct espconn {
   /** type of the espconn (TCP, UDP) */
   enum espconn_type type;
   /** current state of the espconn */
   enum espconn_state state;
   union {
      esp_tcp *tcp;
      esp_udp *udp;
   } proto;
   /** A callback function that is informed about events for this espconn */
```

```
    espconn_recv_callback recv_callback;
    espconn_sent_callback sent_callback;
    uint8 link_cnt;
    void *reverse;
};
```

Here the enum espconn_type checks whether the connection is a UDP or a TCP while the enum espconn_state checks for the current state of the esp that is whether it is connected,disconnected, reconnecting or writing to a device. It also stores the local_port, remote_port, local_ip and remote_ip. This also i got know from the espconn.h file. The structures are as follows

```
typedef struct _esp_tcp {
    int remote_port;
    int local_port;
    uint8 local_ip[4];
    uint8 remote_ip[4];
    espconn_connect_callback connect_callback;
    espconn_reconnect_callback reconnect_callback;
    espconn_connect_callback disconnect_callback;
    espconn_connect_callback write_finish_fn;
} esp_tcp;
```

END TIME: 00:30 Hrs **Date: 08/07/2018**


My gut feeling is you're trying to transfer the data without extra buffering, and the data buffer is probably being free()'d along with your connection handle before it can do so. I would suggest buffering your data you want to send in the receive callback, then send from that when you can.

RE: ESPCONN_ARG ERROR #76987

By QuickFix - Wed Jul 11, 2018 6:27 am

END TIME: 17:50 Hrs

START TIME: 15:30 Hrs **Date: 12/07/2018**

Even after buffering the data in the receive callback function that is receive_cb it shows up the same error. So we have decided to implement a delay of a few milliseconds between the time the ESP8266 receives the data from the device connected at its Access Point and forwarding it ahead to the other ESP8266 connected at its station, so that the receive and transmit of data doesn’t happen simultaneously and the memory buffer doesn’t get cleared instantly. But this also didn’t change things in anyways as i am stuck exactly where i was before.

So now our only options are to resort to my professors and my seniors who have worked with ESP8266 previously for help.

END TIME: 19:Hrs

START TIME: 02:00 Hrs **Date: 16/07/2018**

We have mailed Professor Mike regarding this issue. The following was our email to Mike:

Hello Mike,

This mail is regarding the error that we are facing with our project and we need an experts advice to guide through our issue. We hope to get your valuable inputs to complete our project ESP8266 bridge.We are trying to solve the error from past three weeks and are struck at this point.We are sending you our code as well as the part that we are facing error with.We would kindly request you to please provide your suggestion in solving this issue and guide us in completing our project .

....

===

I have 2 esp modules. Both work on station AP mode

===

===

module A (in station mode) connects to a router

module A (in softAP mode) works as a tcp server

module B (in station mode) connects to module A's softAP

module B (in softAP mode) acts as a tcp server

===

===

sending receiving data via TCP between a device connected to the router and module A works perfectly

sending receiving data via TCP between a device connected to the module A's softAP and module A works perfectly

===

===

now module B connects (as station) to module A's softAP and a TCP connection is created with module A as server and module B as client. The data transfer is working. Data received from the client, by the server, is mirrored i.e if 'a' is received , 'a' is send back.

==

==

module B's softAP acts as a tcp server and connecting my laptop to module B's softAp as a tcp client is working as well. Data sending and receiving is working.

The problem arises when i try to send the data received via module B's softAp to module A. In the data receive callback of module B's softAP, i send the received data to module B's station TCP connection and i get erroe code (-12) which is illegal argument error. I need your help with resolving it. Once it is sorted the project will be almost done.

==

```
int espcon_status = espconn_connect(&TcpSTAClientConn);
    //#ifdef PLATFORM_DEBUG
    switch(espcon_status)
    {
      case ESPCONN_OK:
        ets_uart_printf("TCP CLINET created.\r\n");
        tcpClientFalg = 1;


        break;
      case ESPCONN_RTE:
        ets_uart_printf("Error connection, routing problem.\r\n\r");
        break;
      case ESPCONN_TIMEOUT:
        ets_uart_printf("Error connection, timeout.\r\n\r");
        break;
      case ESPCONN_MEM:
        ets_uart_printf("OUT OF MEMORY.\r\n\r");
        break;
      case ESPCONN_INPROGRESS:
        ets_uart_printf("in progress.\r\n\r");
```

```
            break;
         case ESPCONN_ABRT:
            ets_uart_printf("ABORT.\r\n\r");
            break;
         case ESPCONN_RST:
            ets_uart_printf("RESET.\r\n\r");
            break;
         case ESPCONN_CLSD:
            ets_uart_printf("CLOSED \r\n\r");
            break;
         case ESPCONN_CONN:
            ets_uart_printf("NOT CONNECTED\r\n\r");
            break;
         case ESPCONN_ARG:
            ets_uart_printf("ILLEGAL ARGUMENT.\r\n\r");
            break;
         case ESPCONN_ISCONN:
            ets_uart_printf("ALREADY CONNECTED.\r\n\r");
            break;
         default:
            ets_uart_printf("Connection error: %d\r\n\r", (int)espsscon_status);
      }
```

We are getting Connection error: 1. none of the errors mentioned in the espconn.h is received. i have mentioned all the errors in the switch case but i still get error code 1.

my code for tcp connection is as follows

```
char tcpserverip[10];
    TcpSTAClientConn.proto.tcp = &ConnTcp;
    TcpSTAClientConn.type = ESPCONN_TCP;
    TcpSTAClientConn.state = ESPCONN_NONE;
    //os_sprintf(tcpserverip, "%s", "10.10.10.1");
    //uint32_t ip = ipaddr_addr(tcpserverip);
    uint32_t ip = ipaddr_addr("10.10.10.1");
    os_memcpy(TcpSTAClientConn.proto.tcp->remote_ip, &ip, 4);


    TcpSTAClientConn.proto.tcp->local_port = 1234;//espconn_port();
    TcpSTAClientConn.proto.tcp->remote_port = 2345;


    espconn_regist_connectcb(&TcpSTAClientConn, tcp_connect_cb);
    espconn_regist_reconcb(&TcpSTAClientConn, tcp_reconnect_cb);
    espconn_regist_disconcb(&TcpSTAClientConn, tcp_disconnect_cb);
    //#ifdef PLATFORM_DEBUG
    ets_uart_printf("Start espconn_connect to " IPSTR ":%d\r\n", IP2STR(TcpSTAClientConn.proto.tcp-
>remote_ip), TcpSTAClientConn.proto.tcp->remote_port);
    //#endif
    //#ifdef LWIP_DEBUG
    ets_uart_printf("Start espconn_connect local port %u\r\n", TcpSTAClientConn.proto.tcp->local_port);
    //#endif
    ets_uart_printf("FREE HEAP = %d\r\n",system_get_free_heap_size());
    int espcon_status = espconn_connect(&TcpSTAClientConn);
    //#ifdef PLATFORM_DEBUG
    switch(espcon_status)
    {
      case ESPCONN_OK:
        ets_uart_printf("TCP CLINET created.\r\n");
        tcpClientFalg = 1;


        break;
      case ESPCONN_RTE:
        ets_uart_printf("Error connection, routing problem.\r\n\r");
        break;
      case ESPCONN_TIMEOUT:
        ets_uart_printf("Error connection, timeout.\r\n\r");
        break;
      case ESPCONN_MEM:
        ets_uart_printf("OUT OF MEMORY.\r\n\r");
```

```
            break;
          case ESPCONN_INPROGRESS:
            ets_uart_printf("in progress.\r\n\r");
            break;
          case ESPCONN_ABRT:
            ets_uart_printf("ABORT.\r\n\r");
            break;
          case ESPCONN_RST:
            ets_uart_printf("RESET.\r\n\r");
            break;
          case ESPCONN_CLSD:
            ets_uart_printf("CLOSED \r\n\r");
            break;
          case ESPCONN_CONN:
            ets_uart_printf("NOT CONNECTED\r\n\r");
            break;
          case ESPCONN_ARG:
            ets_uart_printf("ILLEGAL ARGUMENT.\r\n\r");
            break;
          case ESPCONN_ISCONN:
            ets_uart_printf("ALREADY CONNECTED.\r\n\r");
            break;
          default:
            ets_uart_printf("Connection error: %d\r\n\r", (int)espsscon_status);
        }
```

my code for soft tcp data rec callback is as follows

```
LOCAL void ICACHE_FLASH_ATTR sap_tcp_server_rec_cb(void *arg,char* data,uint16_t len){
    struct espconn *con = arg;
    ets_uart_printf("sap_tcp rec data: %s \n\r",data); // this will go to the tcp connection
```

```
espconn_sent(&TcpSTAClientConn,"helloWorld\n\r",strlen("helloWorld\n\r"));
```

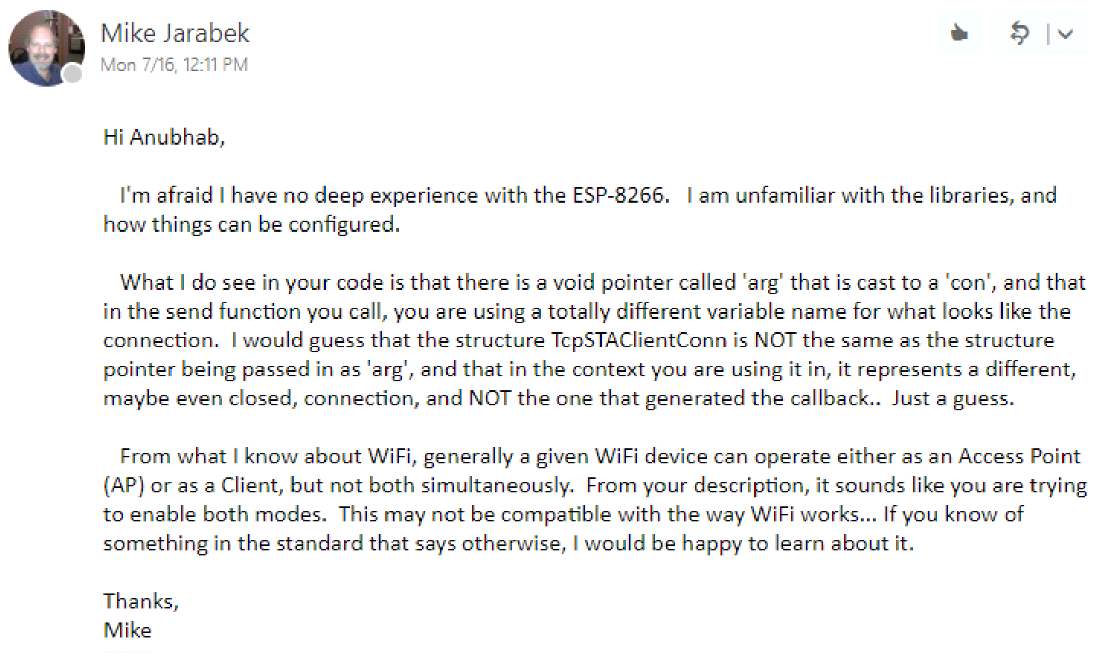
Above line gives us the -12 error and sometimes 1.

I hope you will look into this and help us completing our project.

Thanking you,

Abubhab de.

We got a reply from Mike where he provided us with his opinion that there may be a mismatch in the argument that we are passing. He suggested that the structure TcpSTAClientConn is not the same as the pointer 'arg' that is being passed. He also had the view that a WiFi can only be configured as a Staion or an Access Point but not both simultaneously.


Mike Jarabek
Mon 7/16, 12:11 PM

Hi Anubhab,

I'm afraid I have no deep experience with the ESP-8266. I am unfamiliar with the libraries, and how things can be configured.

What I do see in your code is that there is a void pointer called 'arg' that is cast to a 'con', and that in the send function you call, you are using a totally different variable name for what looks like the connection. I would guess that the structure TcpSTAClientConn is NOT the same as the structure pointer being passed in as 'arg', and that in the context you are using it in, it represents a different, maybe even closed, connection, and NOT the one that generated the callback.. Just a guess.

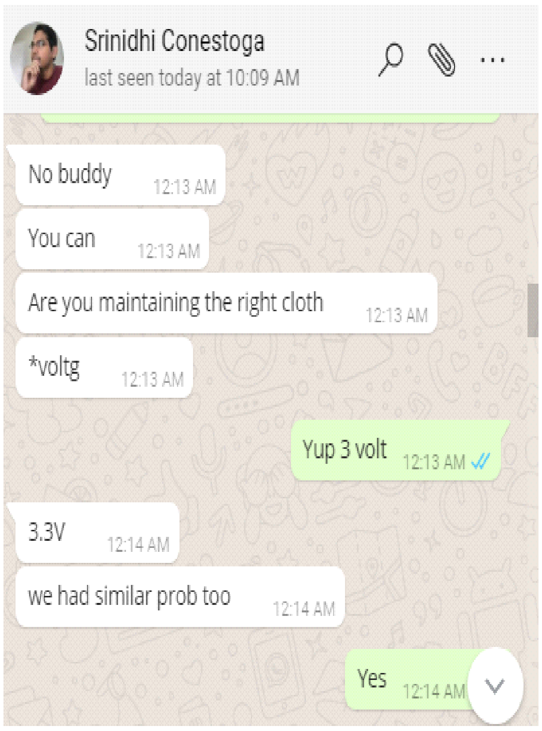
From what I know about WiFi, generally a given WiFi device can operate either as an Access Point (AP) or as a Client, but not both simultaneously. From your description, it sounds like you are trying to enable both modes. This may not be compatible with the way WiFi works... If you know of something in the standard that says otherwise, I would be happy to learn about it.

Thanks,
Mike

We are now trying to implement the changes. First we have changed the argument that we were passing in the callback function and then we implemented a switchcase to switch between the modes of operation that is between the Station and the SoftAp mode. According to the new changes that we have implemented we have used a flag called receiveDataflag which would be set once the ESP8266 receives data on its access point. Once the flag is set, the ESP8266 will terminate its SoftAp connection mode and switch to its Statin mode to transmit the data stored in the data buffer to the preceeding ESP8266 module or to the router.

Even after applying these changes the code returns the same error.

We have also contacted our senior Srinidhi Krishnan regarding this issue and he told us that this sometimes happen if the ESP8266 doesnt get the desired 3.3V at its input.


Srinidhi Conestoga
last seen today at 10:09 AM

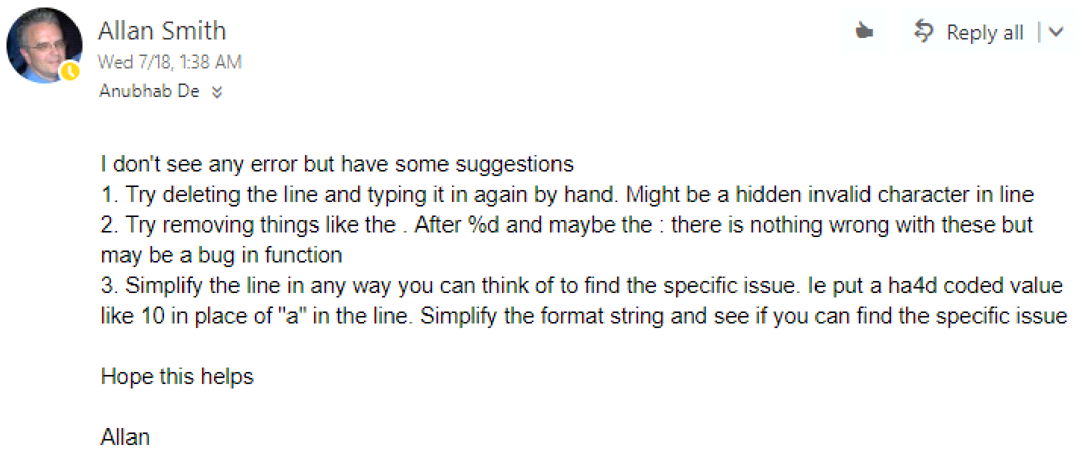
No buddy 12:13 AM
You can 12:13 AM
Are you maintaining the right cloth 12:13 AM
*voltg 12:13 AM
Yup 3 volt 12:13 AM
3.3V 12:14 AM
we had similar prob too 12:14 AM
Yes 12:14 AM

In order to ensure that i was getting 3.3V at its input we checked with the multimeter at the output of the power suplly for the ESP8266 where i got a voltage of 3.37V on the multimeter and on the other end which is connected to the ESP8266 Vcc pin. This also gave me a reading of 3.34V on the multimeter. So this clearly defines that there are no issues regarding the supply voltage to the ESP8266 module.

END TIME: 15:00 Hrs

START TIME: 07:00 Hrs **Date: 18/07/2018**

We have emailed to Professor Alan Smith regarding the issue as well and he gave us a reply which is as follows:


Allan Smith
Wed 7/18, 1:38 AM
Anubhab De

I don't see any error but have some suggestions
1. Try deleting the line and typing it in again by hand. Might be a hidden invalid character in line
2. Try removing things like the . After %d and maybe the : there is nothing wrong with these but may be a bug in function
3. Simplify the line in any way you can think of to find the specific issue. Ie put a ha4d coded value like 10 in place of "a" in the line. Simplify the format string and see if you can find the specific issue

Hope this helps

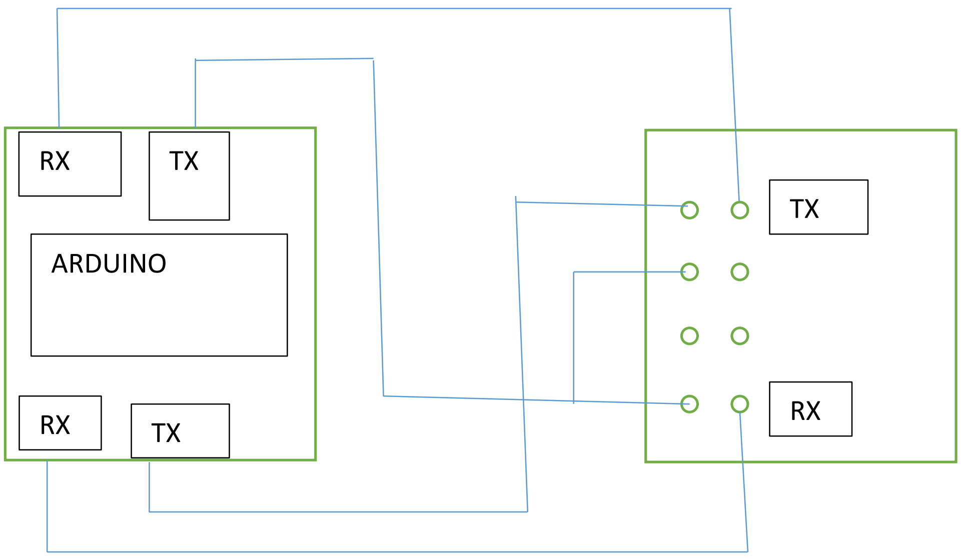
Allan

This didn't help either and we are still stuck at the same place with now getting wrinkles on our forehead on how to complete the project in time or rather if at all we would be able to get our project working

END TIME: 10:50Hrs

START TIME: 13:00 Hrs **Date: 20/07/2018**

Having no success still now we are going to try out writing the firmware in in Arduino IDE as well. We will be trying to use Arduino sketches for Arduino UNO development board. We have downloaded the Arduino IDE to write the firmware.



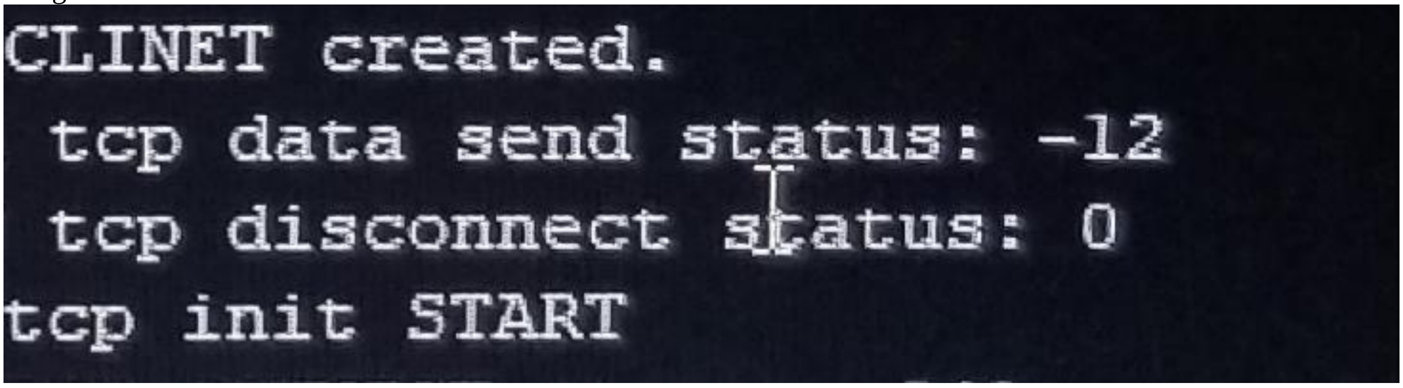
CONNECTING DIAGRAM WITH ARDUINO

Budrate – 115200
Inbuilt AT command set
32 bit processor
26-52 MHZ processor
64 kb of RAM
96 kb of ROM
Arduino IDE downloaded
Download ESP8266 library for Arduino
Create a new sketch in Arduino to run the code
Write the code for establishing ESP8266 in dual mode
Compile the sketch and upload the sketch to ESP8266 using Arduino

CODE:
```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <EEPROM.h>
#include <string.h>

char ssid[100] = "STA MODE"; //SSID STATION MODE
char password[100] = "1234567890"; //Password STATION MODE

const char* myssid = "AP MODE"; //SSID AP MODE
const char* mypassword = "1234567890"; //Password AP MODE

WiFiServer server(80);
ESP8266WebServer serverAP(8000);

long timeout=0;
char estado=0;

#define OUT1 5

void handleRoot()
```

```
{
serverAP.send(200, "text/html", "<h1>You are connected</h1>");
}

void setup()
{
pinMode(OUT1, OUTPUT);
digitalWrite(OUT1, 0);

WiFi.mode(WIFI_AP_STA);

//AP
IPAddress myIP =WiFi.softAP(myssid, mypassword);
serverAP.on("/", handleRoot);
serverAP.begin();

//Station
WiFi.begin(ssid, password);
}

void loop()
{

serverAP.handleClient();

if(estado==0)
{
if (WiFi.status() == WL_CONNECTED)
{
server.begin();
server.setNoDelay(true);
estado=1;
}
}

if(estado==1)
{
WiFiClient client = server.available();

if (!client)
{
return;
}

timeout=0;
while(!client.available() && timeout<=1000)
{
delay(1);
timeout++;
}

String req = client.readStringUntil('\r');
client.flush();

String resposta = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n ";

if (req.indexOf("/A") != -1)
{
digitalWrite(OUT1,1);
resposta += "OUT1 ON";
}
else if (req.indexOf("/a") != -1)
{
digitalWrite(OUT1,0);
resposta += "OUT1 OFF";
}

client.flush();
resposta += "</html>\n";
client.print(resposta);
delay(1);
}
}
```

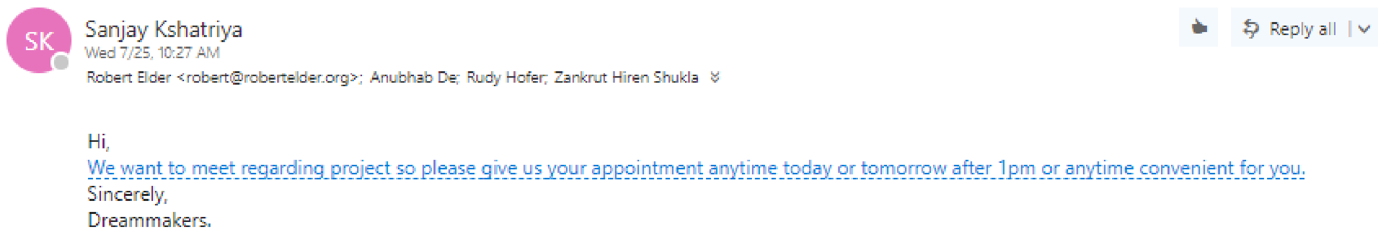
We get the same error.



```
CLINET created.
 tcp data send status: -12
 tcp disconnect status: 0
tcp init START
```

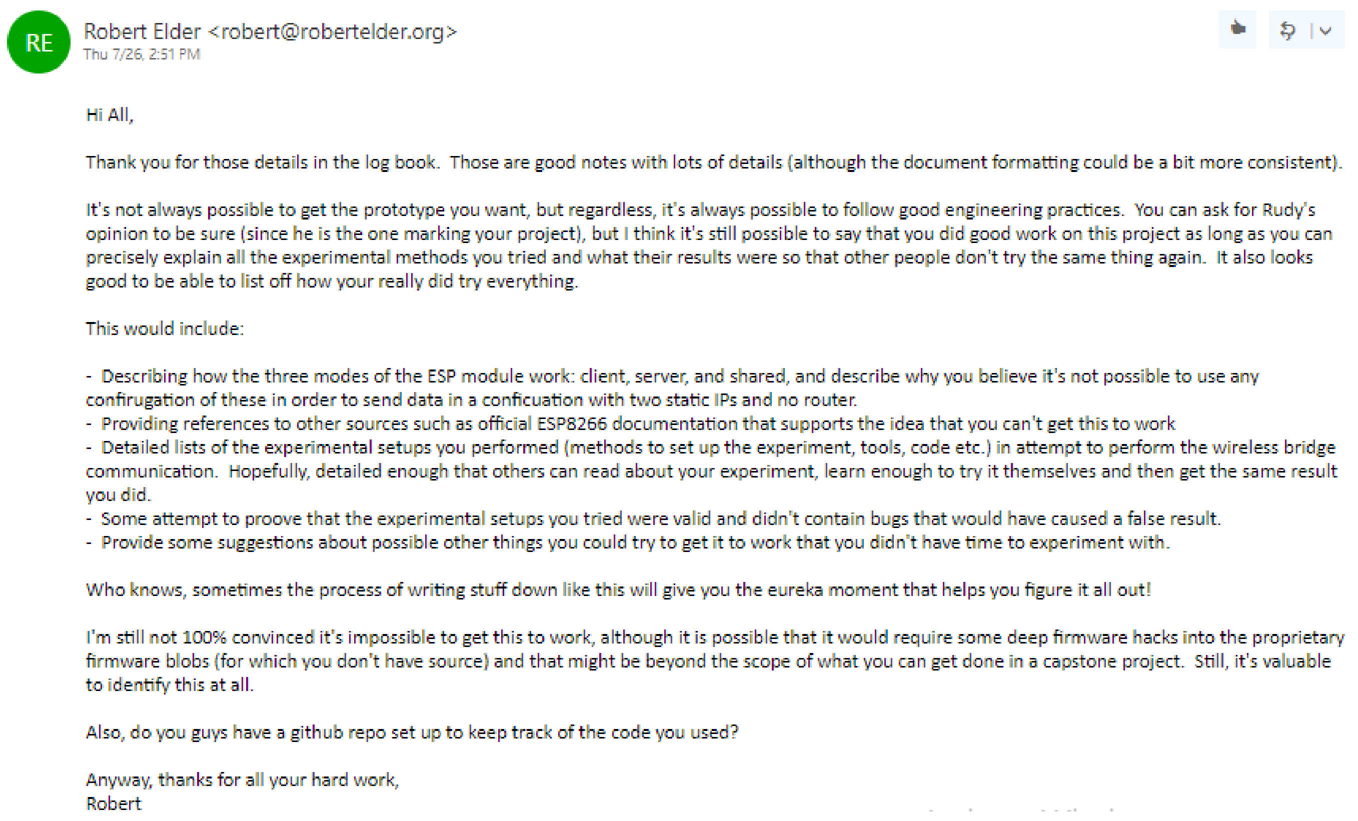
We have also used AT commands but still error persist.

Finally we contacted our client Mr. Robert and gave him the information regarding our progress


Sanjay Kshatriya
Wed 7/25, 10:27 AM
Robert Elder <robert@robertelder.org>; Anubhab De; Rudy Hofer; Zankrut Hiren Shukla

Hi,
We want to meet regarding project so please give us your appointment anytime today or tomorrow after 1pm or anytime convenient for you.
Sincerely,
Dreammakers.

This was Mr. Robert Elders reply


Robert Elder <robert@robertelder.org>
Thu 7/26, 2:51 PM

Hi All,

Thank you for those details in the log book. Those are good notes with lots of details (although the document formatting could be a bit more consistent).

It's not always possible to get the prototype you want, but regardless, it's always possible to follow good engineering practices. You can ask for Rudy's opinion to be sure (since he is the one marking your project), but I think it's still possible to say that you did good work on this project as long as you can precisely explain all the experimental methods you tried and what their results were so that other people don't try the same thing again. It also looks good to be able to list off how your really did try everything.

This would include:

- Describing how the three modes of the ESP module work: client, server, and shared, and describe why you believe it's not possible to use any confirugation of these in order to send data in a conficuation with two static IPs and no router.
- Providing references to other sources such as official ESP8266 documentation that supports the idea that you can't get this to work
- Detailed lists of the experimental setups you performed (methods to set up the experiment, tools, code etc.) in attempt to perform the wireless bridge communication. Hopefully, detailed enough that others can read about your experiment, learn enough to try it themselves and then get the same result you did.
- Some attempt to proove that the experimental setups you tried were valid and didn't contain bugs that would have caused a false result.
- Provide some suggestions about possible other things you could try to get it to work that you didn't have time to experiment with.

Who knows, sometimes the process of writing stuff down like this will give you the eureka moment that helps you figure it all out!

I'm still not 100% convinced it's impossible to get this to work, although it is possible that it would require some deep firmware hacks into the proprietary firmware blobs (for which you don't have source) and that might be beyond the scope of what you can get done in a capstone project. Still, it's valuable to identify this at all.

Also, do you guys have a github repo set up to keep track of the code you used?

Anyway, thanks for all your hard work,
Robert

REASON AND ANALYSIS

Additional sites discussing same issue

1. https://forum.arduino.cc/index.php?topic=415359.0
2. https://arduino.stackexchange.com/questions/18449/esp8266-and-arduino-bidirectional-streaming-from-and-to-a-website
3. https://www.esp8266.com/viewtopic.php?f=32&t=11421

Wireless bridge with ESP8266 is not possible as we are not able to establish a bidirectional communication between two modules working in dual mode and connected to WIFI.
The error lies in the argument being passed to ESPCONN tcp_connect structure. There are no references regarding the type of arguments this structure can accept.
Internal memory of ESP8266 is 512 bits and even this may be one of the reasons why bridge was not established

CONCLUSION

ESP8266 could not be operated in dual mode transferring data directionally while connected to WIFI because of which Wireless WIFI bridge is not been established. As suggested by Mr. Robert it might be possible that Wireless Bridge through ESP8266 would require some deep firmware hacks into the proprietary firmware blobs (for which we don't have source).