EE4178 Laboratory for Microprocessors Systems II

LAB 07

WIFI and IOT: ADC Monitor

Goals:

- Using the provided code, create a soft access point. Use your last name as the ssid and your ID as password.
- Initialize an ADC channel to measure a voltage and visualize it in the provided website.
- Edit the code so that you blink an LED when the user press button 0 and gets data when the user presses the get data button.
- Button 0 must also return the current state of the LED (0 or 1 for on and off)

Bonus:

Display the voltage value rather than the bit value. **+10**

Pre-Lab Questions:

- How can you send stuff back to the server?
- What are the parameters to be use on the function `netconn_write();`?
- How can you find out the size of the string you need to send?

```c
#include <string.h>
#include <math.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "driver/gpio.h"
#include "driver/adc.h"

#include "lwip/sys.h"
#include "lwip/netdb.h"
#include "lwip/api.h"

const static char http_html_hdr[] = "HTTP/1.1 200 OK\r\nContent-type: text/html\r\n\r\n";
const static char http_txt_hdr[] = "HTTP/1.1 200 OK\r\nContent-type: text/plain\r\n\r\n";
const static char http_index_hml[] = R"=====(<!DOCTYPE html><html> <head> <meta charset = UTF-8
name = "viewport" content = "width = device-width initial-scale = 1.0"> <title>Home Page</title> </head>
<body> <div class="header"> <h1>Home Page</h1> </div> <input class = "btn" id = "btn0" type="button"
value = "BTN0" onclick = "sendRequestLed()"> <input class = "btn" id = "btn1" type="button" value = "get
data" onclick = "sendRequestData()"> <div class="sensorVal"> <p>Sensor Value: </p> <p id="sen"></p>
</div> <style> *{margin:0; padding:0;} body {background-color: #D4DCE2;} .header { width:100%;
height:55px; color: white; background-color: #04254D; padding: 0; text-align:center; } .header h1{
color:white; vertical-align:center; font-size:42px; } .btn { margin: 0; margin-top: .5%; background-color:
#FB9541; width:48%; border: none; color: white; padding: 25px 38px; text-align: center; text-decoration:
none; font-size: 16px; } .sensorVal { margin: 0; margin-top: .5%; width:100%; height:70px; color: white;
background-color: #04254D; padding: 0; text-align:center; } .sensorVal p{ color:white; vertical-align:center;
font-size:38px; } </style> <script> function changeButton(value){ var btn =
document.getElementById("btn0"); if(value === "0"){ btn.value = "LED is OFF"; } else{ btn.value = "LED is
ON"; } } function sendRequestLed(){ var http = new XMLHttpRequest(); http.onreadystatechange = (()=>{
if(http.readyState === 4){ if(http.status === 200){ changeButton(http.responseText); } } });
http.open("GET", "0", true); http.send(); } function sendRequestData(){ var http = new XMLHttpRequest();
http.onreadystatechange = (()=>{ if(http.readyState === 4){ if(http.status === 200){
document.getElementById("sen").innerHTML = http.responseText; } } }); http.open("GET", "1", true);
http.send(); } // setInterval(sendRequest1, 1000); </script> </body></html>)=====";

#define EXAMPLE_ESP_WIFI_SSID      ""
#define EXAMPLE_ESP_WIFI_PASS      ""
#define EXAMPLE_MAX_STA_CONN       1

static EventGroupHandle_t s_wifi_event_group;

void wifi_init_softap()
{
  s_wifi_event_group = xEventGroupCreate();

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

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

```c
    wifi_config_t wifi_config = {
      .ap = {
        .ssid = EXAMPLE_ESP_WIFI_SSID,
        .ssid_len = strlen(EXAMPLE_ESP_WIFI_SSID),
        .password = EXAMPLE_ESP_WIFI_PASS,
        .max_connection = EXAMPLE_MAX_STA_CONN,
        .authmode = WIFI_AUTH_WPA_WPA2_PSK
      },
    };
    if (strlen(EXAMPLE_ESP_WIFI_PASS) == 0) {
      wifi_config.ap.authmode = WIFI_AUTH_OPEN;
    }

    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_AP, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());
}

void onURL(struct netconn *conn, char command)
{
    if(command == '0'){
    }
    else if(command == '1'){
    }
    else{
      netconn_write(conn, http_html_hdr, sizeof(http_html_hdr)-1, NETCONN_NOCOPY);
      netconn_write(conn, http_index_hml, sizeof(http_index_hml)-1, NETCONN_NOCOPY);
    }
}

static void http_server_netconn_serve(struct netconn *conn)
{
    struct netbuf *inbuf;
    char *buf;
    u16_t buflen;
    err_t err;

    /* Read the data from the port, blocking if nothing yet there.
     We assume the request (the part we care about) is in one netbuf */
    err = netconn_recv(conn, &inbuf);

    if (err == ERR_OK) {
      netbuf_data(inbuf, (void**)&buf, &buflen);

      /* Is this an HTTP GET command? (only check the first 5 chars, since
       there are other formats for GET, and we're keeping it very simple )*/
      if (buflen>=5 &&
        buf[0]=='G' &&
        buf[1]=='E' &&
        buf[2]=='T' &&
        buf[3]==' ' &&
        buf[4]=='/' ) {
        /* Send the HTML header
             * subtract 1 from the size, since we dont send the \0 in the string
```

Written by Hector Mota. Modified by Dr. Erives & M Elahi September 2021

```
            * NETCONN_NOCOPY: our data is const static, so no need to copy it
        */
                          onURL(conn, buf[5]);
  }

}
netconn_close(conn);
netbuf_delete(inbuf);
}

static void http_server(void *pvParameters)
{
  struct netconn *conn, *newconn;
  err_t err;
  conn = netconn_new(NETCONN_TCP);
  netconn_bind(conn, NULL, 80);
  netconn_listen(conn);
  do {
   err = netconn_accept(conn, &newconn);
   if (err == ERR_OK) {
     http_server_netconn_serve(newconn);
     netconn_delete(newconn);
   }
 } while(err == ERR_OK);
 netconn_close(conn);
 netconn_delete(conn);
}

void setADC()
{

}

void app_main()
{
  nvs_flash_init();
  setADC();
  wifi_init_softap();
  xTaskCreate(&http_server, "http_server", 2048, NULL, 5, NULL);
}
```

Listing 1. Program template for Lab 7.

Written by Hector Mota. Modified by Dr. Erives & M Elahi September 2021