MQTT TCP

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <stddef.h>
#include "esp_wifi.h"
#include "esp_system.h"
#include "nvs_flash.h"
#include "esp_event.h"
#include "tcpip_adapter.h"
#include "protocol_examples_common.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"
#include "freertos/queue.h"
#include "lwip/sockets.h"
#include "lwip/dns.h"
#include "lwip/netdb.h"
#include "esp_log.h"
#include "mqtt_client.h"


Static const char *TAG = "MQTT_EXAMPLE";


Static esp_err_t mqtt_event_handler_cb(esp_mqtt_event_handle_t event){
    esp_mqtt_client_handle_t client = event -> client;
    int msg_id;
    switch (event -> event_id) {
```

```
case MQTT_EVENT_CONNECTED:
    ESP_LOGI (TAG, "MQTT_EVENT_CONNECTED");
    msg_id = esp_mqtt_client_publish
        (client, "/topic/qos1", "data_3", 0, 1, 0);
    ESP_LOGI (TAG, "sent publish successful, msg_id=%d", msg_id);
    ~~msg~~msg_id = esp_mqtt_client_subscribe (client, "/topic/qos0", 0);
    ESP_LOGI (TAG, "sent subscribe successful, msg_id=%d", msg_id);
    msg_id = esp_mqtt_client_subscribe (client, "/topic/qos1", 1);
    ESP_LOGI (TAG, "sent subscribe successful, msg_id = %d", msg_id);
    msg_id = esp_mqtt_client_unsubscribe (client, "/topic/qos1");
    ESP_LOGI (TAG, "sent unsubscribe successful, msg_id=%d", msg_id);
    break;

  case MQTT_EVENT_DISCONNECTED:
    ESP_LOGI (TAG, "MQTT_EVENT_DISCONNECTED");
    break;

case MQTT_EVENT_SUBSCRIBED:
    ESP_LOGI (TAG, "~~msq~~MQTT_EVENT_SUBSCRIBED, msg_id =%d", event->msg_id);
    msg_id = esp_mqtt_client_publish (client, "/topic/qos0", "data", 0, 0, 0);
    ESP_LOGI (TAG, "sent publish successful, msg_id = %d", msg_id);
    break;

case MQTT_EVENT_UNSUBSCRIBED:
    ESP_LOGI (TAG, "MQTT_EVENT_UNSUBSCRIBED, msg_id=%d", event->msg_id);
    break;

case MQTT_EVENT_PUBLISHED:
    ESP_LOGI (TAG, "MQTT_EVENT_PUBLISHED, msg_id=%d", event->msg_id);
    break;
```

```
case MQTT-EVENT-DATA:
   ESP_LOGI(TAG, "MQTT-EVENT_DATA");
   printf("TOPIC = %.*s\r\n", event->topic_len, event->topic);
   printf("DATA = %.*s\r\n", event->data_len, event->data);
   break;

case MQTT_EVENT-ERROR:
   ESP_LOGI("TAG, "MQTT_EVENT_ERROR");
   break;
~~def~~
default:
    ESP_LOGI(TAG, "other event id: %d", event->event_id);
    break;
 }

  return ESP_OK;
}


static void mqtt_event_handler(void *handler_args,
   esp_event_base_t base, int32_t event_id, void *event_data) {
   ESP_LOGD(TAG, "event dispatched from event loop base=%s,
       event_id=%d", base, event_id);
   mqtt_event_handler_cb(event_data);
}
```

BUT WHY: V? X?

```
static void mqtt_app_start(void) {
  esp_mqtt_client_config_t mqtt_cfg = {
    .uri = CONFIG_BROKER_URL,
  };

#if CONFIG_BROKER_URL_FROM_STDIN
   char line[128];
   if (strcmp(mqtt_cfg.uri, "FROM_STDIN") == 0) {
     int count = 0;
     printf("please enter url of mqtt broker \n");
     while (count < 128) {
       int c = fgetc(stdin);
       if (c=='\n') {
          line[count] = '\0';
          break;
       } else if (c>0 && c<127) {
         line[count] = c;
         ++count;
       }
       vTaskDelay(10 / portTICK_MPERIOD_MS);
     }

    mqtt_cfg.uri = line;
    printf("broker url:" %s\n", line);
  } else {
    ESP_LOGE (TAG, "configuration mismatch: wrong broker url");
    abort();
  }
#endif
```

```
    esp_mqtt_client_handle_t client = esp_mqtt_client_init(&mqtt_cfg);
    esp_mqtt_client_register_event(client, ESP_EVENT_ANY_ID,
               mqtt_event_handler, client);
     esp_~~client~~mqtt_client_start(~~&~~client);
}


void app_main() {
    ESP_LOGI(TAG, "[APP] Startup...");
    ESP_LOGI(TAG, "[APP] free memory: %d bytes", esp_get_free_heap_size(
    ESP_LOGI(TAG, "[APP] IDF version: %s", esp_get_idf_version());

     esp_log_level_set("*", ESP_LOG_INFO);
     esp_log_level_set("MQTT_CLIENT", ESP_LOG_VERBOSE);
     esp_log_level_set("MQTT_EXAMPLE", ESP_LOG_VERBOSE);
     esp_log_level_set("TRANSPORT_TCP", ESP_LOG_VERBOSE);
     esp_log_level_set("TRANSPORT_SSL", ESP_LOG_VERBOSE);
     esp_log_level_set("TRANSPORT", ESP_LOG_VERBOSE);
     esp_log_level_set("OUTBOX", ESP_LOG_VERBOSE);

      ESP_ERROR_CHECK(nvs_flash_init());
      tcpip_adapter_init();
      ESP_ERROR_CHECK(esp_event_loop_create_default());
      ESP_ERROR_CHECK(example_connect());
      mqtt_app_start();
 }
```