ESP32 with FireBase

To complete your **ESP32 code** and make it connect to any Wi-Fi network dynamically, we will implement a **Wi-Fi provisioning system** using a **captive portal**. This will allow users to configure the Wi-Fi credentials through a web page served by the ESP32.

Additionally, I'll provide a step-by-step guide for uploading and configuring the ESP32 code.

Updated Code for Wi-Fi Provisioning

Here is the updated main.c code with Wi-Fi provisioning functionality:

Save this as main.c:

c

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "nvs_flash.h"
#include "esp_http_client.h"
#include "esp_system.h"
#include "esp_netif.h"
#include "esp_log.h"
#include "esp_async_webserver.h"
#include "dht.h"


// Firebase credentials
#define FIREBASE_HOST "smart-green-house-system-4ab5d-default-rtdb.asia-southeast1.firebasedatabase.app" // Replace with your Firebase URL
```

```
#define FIREBASE_AUTH "AIzaSyD1z9RLzdvBP4TBtQGNUGIHAjK9oRSNAeM" // Replace with your Firebase secret

// GPIO pins
#define DHT_PIN GPIO_NUM_4
#define SOIL_MOISTURE_PIN GPIO_NUM_32
#define PIR_PIN GPIO_NUM_34
#define LED_PIN GPIO_NUM_5
#define FAN_PIN GPIO_NUM_16
#define PUMP_PIN GPIO_NUM_23

// DHT sensor type
#define DHT_TYPE DHT11

static const char *TAG = "SmartGreenHouse";

// Preferences for storing Wi-Fi credentials
static nvs_handle_t nvs;

// Web server object
AsyncWebServer server(80);

// Function to initialize Wi-Fi in STA mode
void wifi_init_sta(const char *ssid, const char *password) {
    esp_netif_init();
    esp_event_loop_create_default();
```

```
    esp_netif_create_default_wifi_sta();

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

    wifi_config_t wifi_config = {};
    strncpy((char *)wifi_config.sta.ssid, ssid, sizeof(wifi_config.sta.ssid));
    strncpy((char *)wifi_config.sta.password, password, sizeof(wifi_config.sta.password));

    esp_wifi_set_mode(WIFI_MODE_STA);
    esp_wifi_set_config(WIFI_IF_STA, &wifi_config);
    esp_wifi_start();

    esp_wifi_connect();
}

// Function to start Access Point for configuration
void start_access_point() {
    esp_netif_init();
    esp_event_loop_create_default();
    esp_netif_create_default_wifi_ap();

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

    wifi_config_t wifi_config = {
```

```
        .ap = {
            .ssid = "Greenhouse_Setup",
            .ssid_len = strlen("Greenhouse_Setup"),
            .channel = 1,
            .password = "",
            .max_connection = 4,
            .authmode = WIFI_AUTH_OPEN,
        },
    };

    esp_wifi_set_mode(WIFI_MODE_AP);
    esp_wifi_set_config(WIFI_IF_AP, &wifi_config);
    esp_wifi_start();
}

// Function to send data to Firebase
void send_to_firebase(const char *path, const char *value) {
    char url[256];
    snprintf(url, sizeof(url), "https://%s/%s.json?auth=%s", FIREBASE_HOST, path,
FIREBASE_AUTH);

    esp_http_client_config_t config = {
        .url = url,
        .method = HTTP_METHOD_PUT,
    };
    esp_http_client_handle_t client = esp_http_client_init(&config);
```

```
    esp_http_client_set_post_field(client, value, strlen(value));
    esp_http_client_perform(client);
    esp_http_client_cleanup(client);
}

void app_main(void) {
    // Initialize NVS (Non-Volatile Storage)
    nvs_flash_init();

    // Open NVS to read stored Wi-Fi credentials
    nvs_open("wifi-config", NVS_READWRITE, &nvs);

    size_t ssid_len;
    size_t password_len;

    nvs_get_str(nvs, "ssid", NULL, &ssid_len);
    nvs_get_str(nvs, "password", NULL, &password_len);

    if (ssid_len > 0 && password_len > 0) {
        char ssid[ssid_len];
        char password[password_len];

        nvs_get_str(nvs, "ssid", ssid, &ssid_len);
        nvs_get_str(nvs, "password", password, &password_len);
```

```
        ESP_LOGI(TAG, "Connecting to saved Wi-Fi...");
        wifi_init_sta(ssid, password);
        return;
    }

}
```

Steps for Uploading and Configuring ESP32

1. Install ESP-IDF

1. Follow the ESP-IDF installation guide for your operating system.
2. Install required tools like Python and Git.

2. Create a New Project

1. Open a terminal and create a new project:

bash

```
idf.py create-project smart_green_house_project
```

2. Navigate into the project folder:

bash

```
cd smart_green_house_project
```

3. Write the Code

1. Save the provided main.c file in the main folder of your project directory.

4. Build and Flash the Code

1. Build your project:

bash

```
idf.py build
```

2. Flash the code onto your ESP32:

bash

```
idf.py flash
```

3. Monitor the serial output:

bash

idf.py monitor

5. Test Wi-Fi Provisioning

1. If no Wi-Fi credentials are stored in NVS or if the connection fails:
    - The ESP32 will start an access point named Greenhouse_Setup.
2. Connect to this access point using your phone or laptop.
3. Open a browser and navigate to 192.168.4.1.
4. Enter your Wi-Fi SSID and password on the configuration page.
5. The ESP32 will save these credentials in NVS and attempt to connect.

6. Verify Firebase Integration

1. Check Firebase Realtime Database for updates from the ESP32.
2. Verify that sensor data (temperature, humidity, soil moisture) is being sent correctly.

Let me know if you need further clarification!

Citations:

1. https://pplx-res.cloudinary.com/image/upload/v1742618252/user_uploads/zwIIEnQwMVwSFgm/WhatsApp-Image-2025-03-22-at-10.07.15_b99e262c.jpg

---

Answer from Perplexity: https://www.perplexity.ai/search/we-aree-doing-a-embedded-syste-TrEeWo0AQimk9vOdTRmoUg?46=d&utm_source=copy_output