**Internet of Things**

**Lab Report 11**

**Hafiz Ahmad**

**19l-1316**

**Section-7A2**

**Programming ESP8266 development boards Over-the-air (OTAupdates)**

**INTRODUCTION:**

Mobile phones, set-top boxes, and secure voice communication equipment may all be updated with new software, configuration settings, and even encryption keys through a process known as over-the-air programming (OTA) (encrypted 2-way radios). The fact that an update may be sent to all users from a single central place, that update cannot be rejected, defeated, or changed, and that the update takes effect immediately for everyone on the channel, is a key characteristic of OTA. The "channel management" might "kick them off" the channel immediately if a user "refused" OTA.

**OBJECTIVES:**

To learn Programming ESP8266 development boards Over-the-air (OTA

updates).

**Lab code:**

#include <ESP8266WiFi.h>

#include <ESP8266mDNS.h>

#include <WiFiUdp.h>

#include <ArduinoOTA.h>

const char\* ssid = "..........";

const char\* password = "..........";

void setup() {

Serial.begin(115200);

Serial.println("Booting");

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.waitForConnectResult() != WL\_CONNECTED) {

Serial.println("Connection Failed! Rebooting...");

delay(5000);

ESP.restart();

}

// Port defaults to 8266

// ArduinoOTA.setPort(8266);

// Hostname defaults to esp8266-[ChipID]

// ArduinoOTA.setHostname("myesp8266");

// No authentication by default

// ArduinoOTA.setPassword("admin");

// Password can be set with it's md5 value as well

// MD5(admin) = 21232f297a57a5a743894a0e4a801fc3

// ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

ArduinoOTA.onStart([]() {

String type;

if (ArduinoOTA.getCommand() == U\_FLASH)

type = "sketch";

else // U\_SPIFFS

type = "filesystem";

// NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()

Serial.println("Start updating " + type);

});

ArduinoOTA.onEnd([]() {

Serial.println("\nEnd");

});

ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {

Serial.printf("Progress: %u%%\r", (progress / (total / 100)));

});

ArduinoOTA.onError([](ota\_error\_t error) {

Serial.printf("Error[%u]: ", error);

if (error == OTA\_AUTH\_ERROR) Serial.println("Auth Failed");

else if (error == OTA\_BEGIN\_ERROR) Serial.println("Begin Failed");

else if (error == OTA\_CONNECT\_ERROR) Serial.println("Connect Failed");

else if (error == OTA\_RECEIVE\_ERROR) Serial.println("Receive Failed");

else if (error == OTA\_END\_ERROR) Serial.println("End Failed");

});

ArduinoOTA.begin();

Serial.println("Ready");

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

}

void loop() {

ArduinoOTA.handle();

}

Upload to air:

#include <ESP8266WiFi.h>

#include <ESP8266mDNS.h>

#include <WiFiUdp.h>

#include <ArduinoOTA.h>

const char\* ssid = "..........";

const char\* password = "..........";

//variabls for blinking an LED with Millis

const int led = D0; // ESP8266 Pin to which onboard LED is connected

unsigned long previousMillis = 0; // will store last time LED was updated

const long interval = 1000; // interval at which to blink (milliseconds)

int ledState = LOW; // ledState used to set the LED

void setup() {

pinMode(led, OUTPUT);

Serial.begin(115200);

Serial.println("Booting");

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.waitForConnectResult() != WL\_CONNECTED) {

Serial.println("Connection Failed! Rebooting...");

delay(5000);

ESP.restart();

}

// Port defaults to 8266

// ArduinoOTA.setPort(8266);

// Hostname defaults to esp8266-[ChipID]

// ArduinoOTA.setHostname("myesp8266");

// No authentication by default

// ArduinoOTA.setPassword("admin");

// Password can be set with it's md5 value as well

// MD5(admin) = 21232f297a57a5a743894a0e4a801fc3

// ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

ArduinoOTA.onStart([]() {

String type;

if (ArduinoOTA.getCommand() == U\_FLASH)

type = "sketch";

else // U\_SPIFFS

type = "filesystem";

// NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()

Serial.println("Start updating " + type);

});

ArduinoOTA.onEnd([]() {

Serial.println("\nEnd");

});

ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {

Serial.printf("Progress: %u%%\r", (progress / (total / 100)));

});

ArduinoOTA.onError([](ota\_error\_t error) {

Serial.printf("Error[%u]: ", error);

if (error == OTA\_AUTH\_ERROR) Serial.println("Auth Failed");

else if (error == OTA\_BEGIN\_ERROR) Serial.println("Begin Failed");

else if (error == OTA\_CONNECT\_ERROR) Serial.println("Connect Failed");

else if (error == OTA\_RECEIVE\_ERROR) Serial.println("Receive Failed");

else if (error == OTA\_END\_ERROR) Serial.println("End Failed");

});

ArduinoOTA.begin();

Serial.println("Ready");

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

}

void loop() {

ArduinoOTA.handle();

//loop to blink without delay

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

// save the last time you blinked the LED

previousMillis = currentMillis;

// if the LED is off turn it on and vice-versa:

ledState = not(ledState);

// set the LED with the ledState of the variable:

digitalWrite(led, ledState);

}

}

**Application:**

Open-source, low-cost IoT platform NodeMCUIt started out with hardware based on the ESP-12 module and firmware that runs on Espressif Systems' ESP8266 Wi-Fi SoC.Later, the ESP32 32-bit MCU received support.The only drawback is that in order to use OTA in the subsequent update, you must include an OTA code with each sketch you upload. The ESP-07 is based on Popular ESP8266 WiFi chip and features an on-board ceramic antenna as well as uFL connector allowing an external antenna to be connected for greater range.

**Issues:**

we never find any issue regarding this lab.

**Conclusion:**

In this lab we learn OTA programming lets you update/upload a new program to the ESP8266 over Wi-Fi without having to connect the ESP8266 to the computer via USB. The OTA functionality comes in handy when there is no physical access to the ESP module.