Practical IoT (Internet of Things)

BSCIS – DCIS, PIEAS

Lab 08 - OTA Updates for IoT Devices

Objective

- Implement and compare different OTA update methods using ESP8266/ESP32 and control updates via Node-RED

Prerequisites

- Pre-configured Node-RED server (one per group)
- Arduino IDE with ESP8266WiFi and DHT sensor libraries
- Lecture on OTA updates completed.
- Arduino IDE with ESP8266/ESP32 board support.
- Node-RED with `node-red-dashboard` and `node-red-contrib-http-request` installed.
- Stable Wi-Fi network with shared credentials.

Lab Setup

Hardware

- ESP8266 (e.g., WeMos D1 / NodeMCU) or ESP32 development board.
- USB cable for initial flashing.
- Wi-Fi network (stable, you may use your mobile hotspot)

Software

- Arduino IDE with the following libraries:
  - `ArduinoOTA` (pre-installed with ESP8266/ESP32 boards).
  - `ESPAsyncWebServer, AsyncTCP` (ESP32), `ESPAsyncTCP` (ESP8266), `ElegantOTA`.
- Node-RED (local or cloud instance) with dashboard and HTTP request nodes.
- Python 3 for hosting firmware files (`http.server`).

Pre-Lab Preparation (For each Task)

- Prepare two sample .bin files:
  - `blink_v1.bin`: LED blinks every 1 second.
  - `blink_v2.bin`: LED blinks every 0.5 seconds (to demonstrate update).
- Wi-Fi credentials (SSID, password).
- Provide server IP for firmware hosting (e.g., `192.168.1.100:8000`).

- Ensure boards have ≥1MB flash memory for OTA partitions.

Lab Tasks 1: Manual OTA Updates with ArduinoOTA

Install ArduinoOTA Library

- Already included with ESP8266/ESP32 board support in Arduino IDE.
- Verify in Sketch > Include Library > ArduinoOTA.

Flash Initial Firmware

- Use the following sketch for ESP8266 (modify for ESP32 by replacing `ESP8266WiFi.h` with `WiFi.h`): Please check your WiFi credentials

```
#include <ESP8266WiFi.h>
#include <ArduinoOTA.h>
const char* ssid = "IoT-Class";
const char* password = "iotclass1";
void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi Connected. IP: " + WiFi.localIP().toString());
  ArduinoOTA.setHostname("ESP-OTA");
  ArduinoOTA.begin();
}
void loop() {
  ArduinoOTA.handle();
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Blink every 1 second (v1)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

- Upload via USB and open Serial Monitor to confirm Wi-Fi connection and IP address.

Prepare Firmware for Update:

- Compile blink_v2.bin (0.5-second blink) in Arduino IDE:
- Modify delay(1000) to delay(500) in the above sketch.
- Export via Sketch > Export Compiled Binary.

Perform Manual OTA Update:

- In Arduino IDE, go to Tools > Port > Network Ports, select the ESP's IP (e.g., "ESP-OTA at 192.168.1.x"). The IP Should be matched as displayed in Serial Monitor.
- Upload newly compiled sketch using Upload button.
- Monitor Serial Monitor for update progress and verify the LED blinks faster (0.5 seconds).

Lab Tasks 2: Manual OTA Updates with ElegantOTA

Install Required Libraries

- Install following libraries via Arduino Library Manager:
    - `ESPAsyncWebServer`
    - `AsyncTCP (ESP32) or ESPAsyncTCP (ESP8266)`
    - `ElegantOTA`

Flash ElegantOTA Firmware

- Use the following sketch for ESP8266 (modify for ESP32 by replacing `ESP8266WiFi.h` with `WiFi.h`): Please check your WiFi credentials

```
#include <ESP8266WiFi.h>
#include <ElegantOTA.h>
const char* ssid = "IoT-Class";
const char* password = "iotclass1";
ESP8266WebServer server(80);
// Non-blocking LED blink
static unsigned long previousMillis = 0; // Store last time LED was updated
const long interval = 1000; // Interval for 1-second blink (on/off)
static bool ledState = false; // Track LED state (HIGH/LOW)

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWiFi Connected. IP: " + WiFi.localIP().toString());
  ElegantOTA.begin(&server);
  server.begin();
}
```

```
void loop() {
  server.handleClient();
  ElegantOTA.loop();

  unsigned long currentMillis = millis(); // Get current time
  // Check if it's time to toggle the LED
  if (currentMillis - previousMillis >= interval) {
    ledState = !ledState; // Toggle state
    digitalWrite(LED_BUILTIN, ledState ? HIGH : LOW); // Update LED
    previousMillis = currentMillis; // Save the current time
  }
}
```

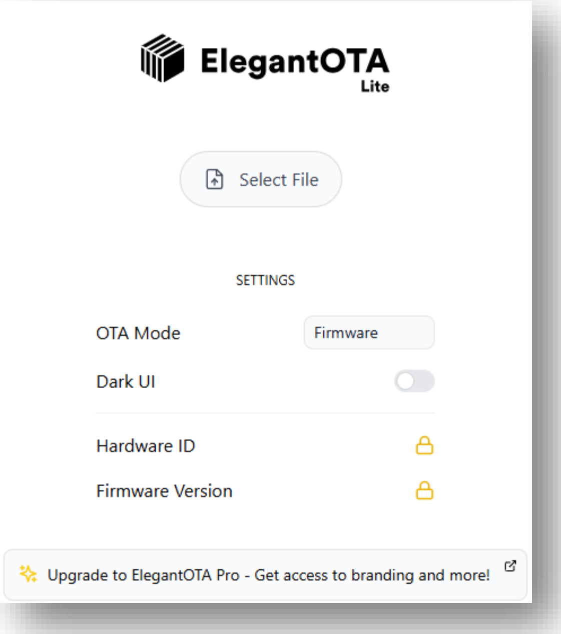
- Upload via USB and note the ESP's IP in Serial Monitor.

Prepare Firmware for Update:

- Compile blink_v2.bin (0.5-second blink) in Arduino IDE:
- Modify constant variable `interval` from 1000 to 500 in the above sketch.
- Export via Sketch > Export Compiled Binary.

Access ElegantOTA Web Interface:

- Open a browser and go to **`http://<ESP_IP>/update`**.
- You'll see a web interface with options to upload firmware or filesystem.



SETTINGS

OTA Mode: Firmware

Dark UI

Hardware ID

Firmware Version

Upgrade to ElegantOTA Pro - Get access to branding and more!

Perform Manual OTA Update:

- Use the newly created blink_v2.bin (0.5-second blink).
- In the ElegantOTA interface, click “Choose File”, select blink_v2.bin, and click “Open”.
- Update starts automatically
- Watch Serial Monitor and Web UI for progress.
- Verify the LED blinks every 0.5 seconds.

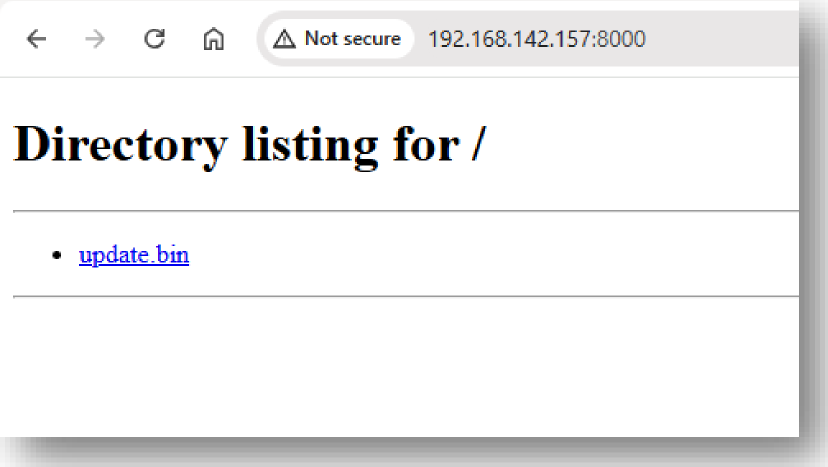
Lab Tasks 3: Automated OTA Updates with Node-RED

Set Up Firmware Server:

- On a group member’s laptop, create a new directory, and place `blink_v2.bin` in the directory.
- Change to your newly created directory and host `blink_v2.bin` using Python:

```
python -m http.server 8000
```

- Note the server IP (e.g., 192.168.1.100:8000).
- Make sure that webserver is accessible in browser like using above URL.



Directory listing for /

- update.bin

Flash Modified ElegantOTA Firmware:

- Modify the ElegantOTA sketch to accept HTTP POST requests for automated updates:

```
#include <ESP8266WiFi.h>
#include <ElegantOTA.h>
#include <ESP8266HTTPClient.h> // Use <HTTPClient.h> for ESP32
#include <ESP8266httpUpdate.h> // Use <HTTPUpdate.h> for ESP32
const char* ssid = "IoT-Class";
const char* password = "iotclass1";
const char* firmwareUrl = "http://192.168.142.157:8000/update.bin"; // firmware URL
```

```
ESP8266WebServer server(80);
WiFiClient client; // Create WiFiClient instance for HTTPClient

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWiFi Connected. IP: " + WiFi.localIP().toString());
  ElegantOTA.begin(&server);
  server.on("/trigger-update ", []() {
    server.send(200, "text/plain", "Update triggered.");
    // Perform automated OTA update
    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(client, firmwareUrl);
      t_httpUpdate_return ret = ESPhttpUpdate.update(http, "");

      switch (ret) {
        case HTTP_UPDATE_FAILED:
          Serial.printf("Update Failed: %s\n", ESPhttpUpdate.getLastErrorString().c_str());
          server.send(500, "text/plain", "Update Failed");
          break;
        case HTTP_UPDATE_NO_UPDATES:
          Serial.println("No Updates Available");
          server.send(304, "text/plain", "No Updates");
          break;
        case HTTP_UPDATE_OK:
          Serial.println("Update Success");
          server.send(200, "text/plain", "Update Success");
          break;
      }
      http.end();
    } else {
      Serial.println("Wi-Fi Disconnected");
      server.send(503, "text/plain", "Wi-Fi Disconnected");
    }
  });

  server.begin();
}

// Non-blocking LED blink
static unsigned long previousMillis = 0; // Store last time LED was updated
const long interval = 1000; // Interval for 1-second blink (on/off)
static bool ledState = false; // Track LED state (HIGH/LOW)

void loop() {
  server.handleClient();
  ElegantOTA.loop();

  unsigned long currentMillis = millis(); // Get current time
  // Check if it's time to toggle the LED
  if (currentMillis - previousMillis >= interval) {
    ledState = !ledState; // Toggle state
    digitalWrite(LED_BUILTIN, ledState ? HIGH : LOW); // Update LED
    previousMillis = currentMillis; // Save the current time
  }
}
```

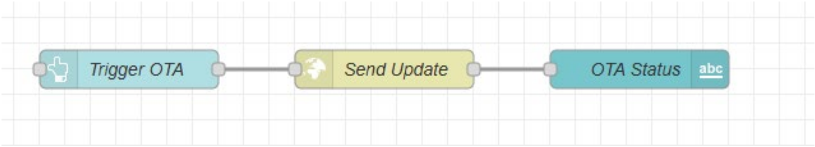
- Upload via USB and note the ESP's IP in Serial Monitor.

Prepare Firmware for Update:

- Compile blink_v2.bin (0.5-second blink) in Arduino IDE:
- Modify constant variable `interval` from 1000 to 500 in the above sketch.
- Export via Sketch > Export Compiled Binary.
- Replace the file in your web server directory (if firmware changed)

Create Node-RED Flow:

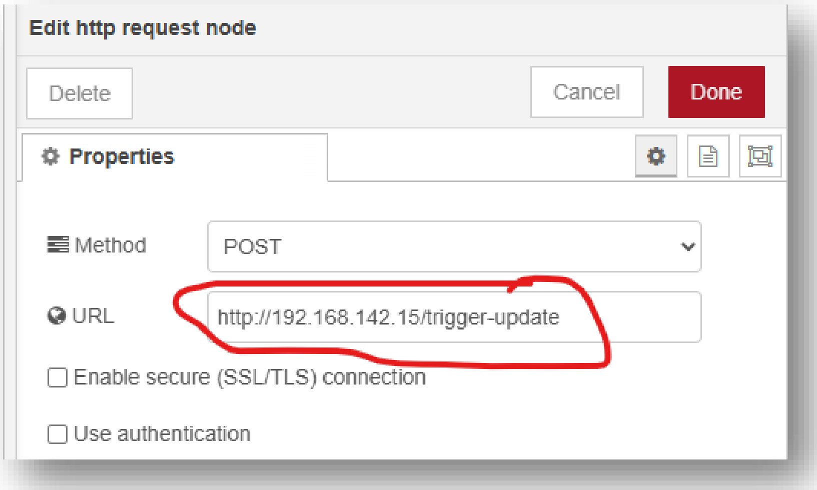
- Open Node-RED and create a flow:
- Button Node: Triggers the update.
- HTTP Request Node: Sends a POST request to `http://<ESP_IP>/trigger-update`.
- Text Node: Displays status on the dashboard.
- Import this flow (replace `<ESP_IP>` with your ESP's IP):



- Go to Menu > Import and copy the following to your Clipboard to create your Node-RED flow. This will create all the flow nodes.

```
[
    {
        "id": "node1",
        "type": "ui_button",
        "z": "6c4985abbf6d31c9",
        "name": "Trigger OTA",
        "group": "32afd9f10e5c45ba",
        "order": 1,
        "width": "",
        "height": "",
        "passthru": false,
        "label": "Start OTA Update",
        "tooltip": "",
        "color": "",
        "bgcolor": "",
        "className": "",
        "icon": "",
        "payload": "",
        "payloadType": "str",
        "topic": "",
        "topicType": "str",
        "x": 110,
        "y": 60,
        "wires": [
            [
                "node2"
            ]
        ]
    },
    {
        "id": "node2",
        "type": "http request",
        "z": "6c4985abbf6d31c9",
        "name": "Send Update",
        "method": "POST",
        "ret": "txt",
        "paytoqs": "ignore",
        "url": "http://<ESP_IP>/trigger-update",
        "tls": "",
        "persist": false,
        "proxy": "",
        "insecureHTTPParser": false,
        "authType": "",
        "senderr": false,
        "headers": [],
        "x": 310,
        "y": 60,
        "wires": [
            [
                "node3"
            ]
        ]
    },
    {
        "id": "node3",
        "type": "ui_text",
        "z": "6c4985abbf6d31c9",
        "group": "32afd9f10e5c45ba",
        "order": 2,
        "width": "",
        "height": "",
        "name": "OTA Status",
        "label": "Status",
        "format": "",
        "layout": "",
        "className": "",
        "style": false,
        "font": "",
        "fontSize": "",
        "color": "#000000",
        "x": 510,
        "y": 60,
        "wires": []
    },
    {
        "id": "32afd9f10e5c45ba",
        "type": "ui_group",
        "name": "Default",
        "tab": "65cf22eab423bc27",
        "order": 1,
        "disp": true,
        "width": 6,
        "collapse": false,
        "className": ""
    },
    {
        "id": "65cf22eab423bc27",
        "type": "ui_tab",
        "name": "Home",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```

- Dual verify your HTTP node does contain the IP of your IoT device (double click node and check properties).



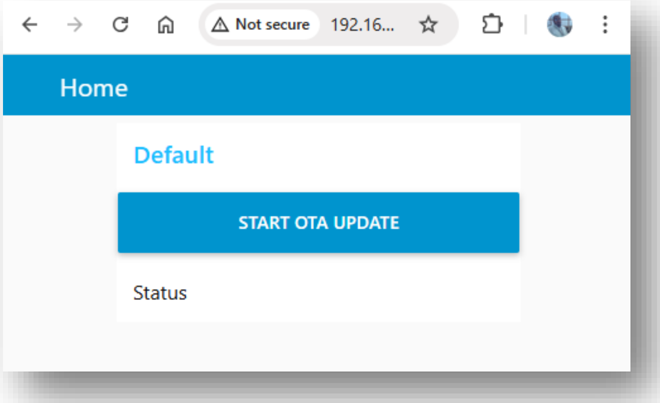
Method: POST

URL: http://192.168.142.15/trigger-update

□ Enable secure (SSL/TLS) connection

□ Use authentication

- Deploy the flow and access the dashboard at `http://<Node-RED IP>:1880/ui`
- Click the "START OTA UPDATE" button on dashboard to trigger automatic update to device.



- The device will start updating automatically.
- Verify the LED blinks every 0.5 seconds

Submission Requirements:

Submit a single PDF containing:

- **Screenshots:**
  - Task 1: ArduinoOTA update with LED blink change.
  - Task 2: ElegantOTA update via web interface.
  - Task 3: Node-RED dashboard triggering an update.
- **Codes:** Wemos code and Node-RED flow JSON. (You can export any Node-RED flow as json).
- **Analysis:** Explain the code working. You need understand and submit working of code for all 3 parts (must thing).
- **Short Description:**
  - The steps you followed.
  - How you verified the functionality.
  - Any challenges you encountered and how you addressed them.