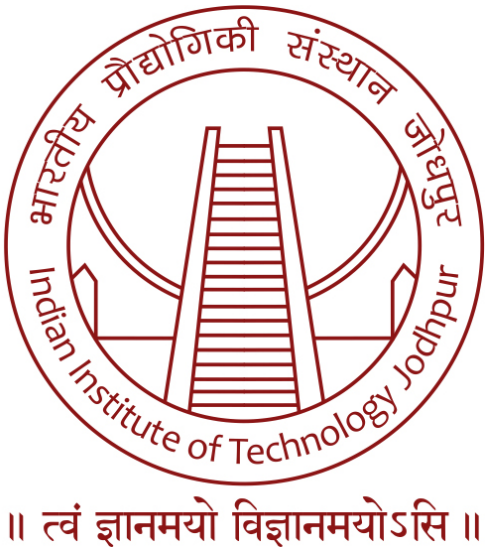
EEL 7170 : Introduction to IoT

Lab Report



Name: **Anushkaa Ambuj**
Roll Number: **B21ES006**
Program: **B.Tech in ES**

Lab 2: Interfacing OTA Updates

1 InLab

1.1 Objective

The experiment focuses on using Over-the-Air (OTA) updates to modify the NodeMCU's firmware without a wired connection, ensuring real-time updates as if the microcontroller were part of an integrated system.

1.2 Components Used

- NodeMCU (ESP8266)
- DHT11 Temperature and Humidity Sensor
- Jumper Wires (female to female)
- Power Bank/External Power Source
- Wi-Fi Hotspot
- Arduino IDE with OTA capability
- DHT11 Library (DFRobot DHT11)

1.3 Procedure

Part 1: Interfacing with DHT11 Sensor

- **Install DHT11 Library:**
  - Open the Arduino IDE and search for the DFRobot DHT11 library in the Library Manager.
  - Install the library for interfacing with the DHT11 sensor.
- **Connections:** Use female-to-female jumper wires to connect the DHT11 sensor to the NodeMCU:
  - DATA pin to D7 on the NodeMCU.
  - VCC pin to 3V.
  - GND pin to G (ground).
- **Upload the Code:**
  - Open the provided `DHT11_interfacing.ino` file.
  - Upload the code to the NodeMCU to start receiving temperature and humidity data from the sensor.
- **Verify Output:** Check the output on the serial monitor for real-time readings of temperature and humidity.

Part 2: Interfacing with LIDAR (VL53L0X) Sensor

- **Install LIDAR Library:**
  - Open the Arduino IDE and search for the Adafruit_VL53L0X library in the Library Manager.
  - Install the library for interfacing with the LIDAR sensor.
- **Connections:** Use female-to-female jumper wires to connect the LIDAR sensor to the NodeMCU:
  - SCL pin to D1 on the NodeMCU.
  - SDA pin to D2.
  - VIN pin to 3V.
  - GND pin to G (ground).
- **Upload the Code:**
  - Open the provided `vl53l0x.ino` file.
  - Upload the code to the NodeMCU to start receiving data on the object's distance from the sensor.
- **Verify Output:** Check the output on the serial monitor for real-time readings of distance.

Part 3: Over-The-Air (OTA) Update

- **Setting Up OTA:**
  - To enable OTA (Over-The-Air) updates, use the BasicOTA example provided in the Arduino IDE (`File > Examples > ArduinoOTA > BasicOTA`).
- **Wi-Fi Configuration:**
  - Modify the OTA code to include your system's Wi-Fi credentials, enabling wireless firmware updates.

Figure 1: Mobile Hotspot Credentials

```
// WiFi credentials
#ifndef STASSID
#define STASSID "IDEAPAD"
#define STAPSK "Hello123"
#endif
const char* ssid = STASSID;
const char* password = STAPSK;
```

Figure 2: Change the Wifi Credentials in the code

  - Upload the OTA code to the NodeMCU initially through a wired connection.

- **Upload the Updated Code via OTA:**
  - Disconnect the NodeMCU from the system and connect it to an external power source.
  - Ensure the Wi-Fi hotspot is on, and the NodeMCU will appear as a device in the Arduino IDE's device list for wireless uploads.

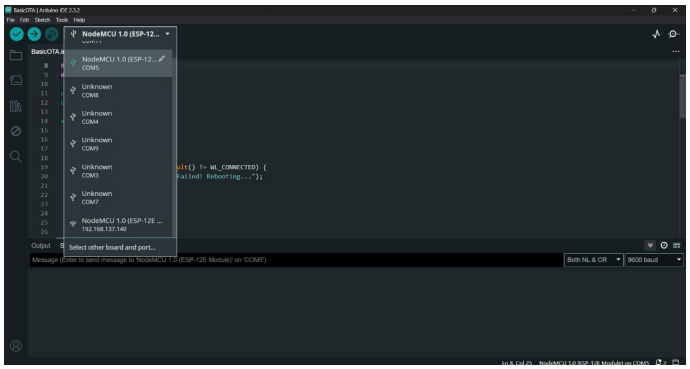


Figure 3: You will see a new device option.

  - Notice: The IP Address of NodeMCU is 192.168.137.140
  - Modify the code to handle additional requirements.

- **Modify Code via OTA:**
  - Modify the code by adding a blinking LED function in the `setup()` and `loop()`.
  - Re-upload the updated code wirelessly by entering the default password (123).
  - Verify if the onboard LED of the NodeMCU starts blinking.

1.4 Code

1.4.1 Part 1: Interfacing with DHT11 Sensor

```
#include "DHT.h"

#define DHTPIN D7     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // Reading temperature or humidity takes about 250 milliseconds!
  float humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float temperature = dht.readTemperature();
  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
```

```
  }
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" *C");
  delay(2000); // Delay between readings.
}
```

[language=C++]

Part 2: Interfacing with LIDAR (VL53L0X) Sensor

```
#include "Adafruit_VL53L0X.h"

Adafruit_VL53L0X lox = Adafruit_VL53L0X();

// 3V --> VIN
// D1 --> SCL
// D2 --> SDA
// G  --> GND

void setup() {
  Serial.begin(115200);

  // wait until serial port opens for native USB devices
  while (! Serial) {
    delay(1);
  }

  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
  // power
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
}


void loop() {
  VL53L0X_RangingMeasurementData_t measure;

  Serial.print("Reading a measurement... ");
  lox.rangingTest(&measure, false); // pass in 'true' to get debug data
     ↪ printout!

  if (measure.RangeStatus != 4) {  // phase failures have incorrect data
    Serial.print("Distance (mm): "); Serial.println(measure.
       ↪ RangeMilliMeter);
  } else {
    Serial.println(" out of range ");
  }

```

```
  delay(100);
}
```

[language=C++]

Part 3: Over-The-Air (OTA) Update

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

#ifndef STASSID
#define STASSID "your-ssid"
#define STAPSK "your-password"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

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_FS
␣␣␣␣␣␣type␣=␣"filesystem";
␣␣␣␣}

␣␣␣␣//␣NOTE:␣if␣updating␣FS␣this␣would␣be␣the␣place␣to␣unmount␣FS␣using␣FS
    ↪ .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();
}
```

[language=C++]

1.5 Observations

- The DHT11 sensor successfully reads temperature and humidity values after proper connection and configuration.

```
Humidity: 88.00 %       Temperature: 32.10 *C
Humidity: 86.00 %       Temperature: 32.10 *C
Humidity: 87.00 %       Temperature: 32.10 *C
Humidity: 89.00 %       Temperature: 32.10 *C
Humidity: 90.00 %       Temperature: 32.10 *C
Humidity: 90.00 %       Temperature: 32.20 *C
Humidity: 88.00 %       Temperature: 32.20 *C
Humidity: 86.00 %       Temperature: 32.30 *C
Humidity: 87.00 %       Temperature: 32.30 *C
Humidity: 89.00 %       Temperature: 32.40 *C
Progress: 100%
Humidity: 89.00 %       Temperature: 32.50 *C
Humidity: 85.00 %       Temperature: 32.50 *C
Humidity: 83.00 %       Temperature: 32.60 *C
Humidity: 82.00 %       Temperature: 32.60 *C
Humidity: 82.00 %       Temperature: 32.60 *C
Humidity: 81.00 %       Temperature: 32.60 *C
Humidity: 80.00 %       Temperature: 32.60 *C
Humidity: 80.00 %       Temperature: 32.60 *C
Humidity: 80.00 %       Temperature: 32.60 *C
Humidity: 80.00 %       Temperature: 32.60 *C
Humidity: 80.00 %       Temperature: 32.60 *C
Humidity: 80.00 %       Temperature: 32.60 *C
Humidity: 79.00 %       Temperature: 32.60 *C
Humidity: 79.00 %       Temperature: 32.60 *C
```

- The LIDAR sensor successfully reads the distance.

```
Reading a measurement... Distance (mm): 299
Reading a measurement... Distance (mm): 297
Reading a measurement... Distance (mm): 305
Reading a measurement... Distance (mm): 302
Reading a measurement... Distance (mm): 300
Reading a measurement... Distance (mm): 293
Reading a measurement... Distance (mm): 297
Reading a measurement... Distance (mm): 298
Reading a measurement... Distance (mm): 304
Reading a measurement... Distance (mm): 301
Reading a measurement... Distance (mm): 301
Reading a measurement... Distance (mm): 302
Reading a measurement... Distance (mm): 305
Reading a measurement... Distance (mm): 303
Reading a measurement... Distance (mm): 301
Reading a measurement... Distance (mm): 305
Reading a measurement... Distance (mm): 304
Reading a measurement... Distance (mm): 303
Reading a measurement... Distance (mm): 304
Reading a measurement
```

- OTA updates provide flexibility for remote firmware modifications without physical access to the NodeMCU.
- When using OTA, the NodeMCU may take time to appear in the device list, requiring patience or network changes.

1.6 Results

- **Part 1:** The interfacing of the DHT11 sensor with NodeMCU was successful, providing accurate temperature and humidity readings.
- **Part 2:** OTA updates were enabled, and the NodeMCU was successfully updated wirelessly, with the onboard LED indicating successful code modification.

---

2 Assignment

2.1 Objective

The objective of this assignment is to:

1. Interface a DHT11 temperature and humidity sensor with NodeMCU and fetch real-time data.
2. Use Over-The-Air (OTA) updates to wirelessly upload new firmware to NodeMCU.
3. Upload sensor data to the ThingSpeak platform for real-time visualization.
4. Generate and upload a random number calculated based on admission year and roll number.

2.2 Components Used

- NodeMCU (ESP8266)
- DHT11 Temperature and Humidity Sensor
- Jumper Wires (Female-to-Female)
- USB Cable and External Power Source
- LiDAR Sensor (VL53L0X)
- Wi-Fi Hotspot
- Arduino IDE
- ThingSpeak API

2.3 Procedure

Interfacing with DHT11 Sensor

Over-The-Air (OTA) Update

Uploading Data to ThingSpeak

- Create a ThingSpeak channel and add three fields: Temperature, Humidity, and Random Number.
- Fetch the Write API Key from ThingSpeak and add it to the Arduino code.

```
18  // ThinkSpeak API details
19  String serverName_send = "http://api.thingspeak.com/update?api_key=OJIOH52E5SUPMIM1";
20
```

Figure 4: Thingspeak API Key

- Calculate the random number as the product of your admission year and roll number, ie. 6*21 = 126.
- Upload the DHT11 data and random number to ThingSpeak for visualization.

```
109   // Generate a random number
110   int randomNumber = 126;
111
112   // Create ThinkSpeak URL with DHT sensor data and random number
113   WiFiClient client;
114   HTTPClient http;
115
116   String serverPath_send = serverName_send + "&field1=" + String(temperature)
117                                            + "&field2=" + String(humidity)
118                                            + "&field3=" + String(random(randomNumber));
119
120   // Send the HTTP GET request to ThinkSpeak
121   http.begin(client, serverPath send.c str());
```

2.4 Code

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <Arduino_JSON.h>
#include "DHT.h"
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

// WiFi credentials
#ifndef STASSID
#define STASSID "IDEAPAD"
#define STAPSK "Hello123"
#endif
const char* ssid = STASSID;
const char* password = STAPSK;

// ThinkSpeak API details
String serverName_send = "http://api.thingspeak.com/update?api_key=
   ↪ OJIOH52E5SUPMIM1";

// DHT sensor details
#define DHTPIN D7
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);

// Timer delay (20 seconds)
unsigned long lastTime = 0;
unsigned long timerDelay = 20000;

void setup() {
  Serial.begin(115200);

  // Initialize DHT sensor
  dht.begin();

  // Connect to WiFi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection␣Failed!␣Rebooting...");
    delay(5000);
    ESP.restart();
  }

  Serial.print("Connected␣to␣WiFi␣network␣with␣IP␣Address:␣");
  Serial.println(WiFi.localIP());

  // Setup OTA updates
  ArduinoOTA.onStart([]() {
    String type = (ArduinoOTA.getCommand() == U_FLASH) ? "sketch" : "
       ↪ filesystem";
```

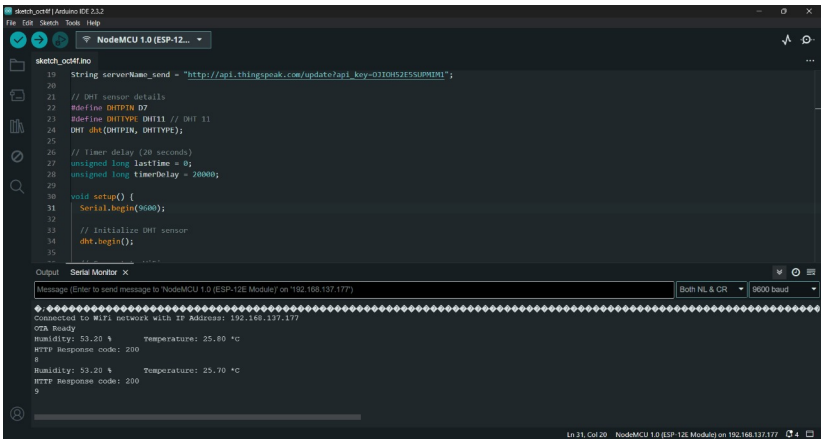
```
    Serial.println("Start updating " + type);
  });

  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("OTA Ready");
}

void loop() {
  // Handle OTA updates
  ArduinoOTA.handle();

  // Check if 20 seconds have passed
  if ((millis() - lastTime) > timerDelay) {
    // Check WiFi connection status
    if (WiFi.status() == WL_CONNECTED) {

      // Read data from the DHT sensor
      float humidity = dht.readHumidity();
      float temperature = dht.readTemperature(); // Celsius

      // Check if any reads failed
      if (isnan(humidity) || isnan(temperature)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
      }

      // Debug print DHT sensor values
      Serial.print("Humidity: ");
      Serial.print(humidity);
      Serial.print(" %\t");
      Serial.print("Temperature: ");
```

```
106       Serial.print(temperature);
107       Serial.println("␣*C");
108
109       // Generate a random number
110       int randomNumber = 126;
111
112       // Create ThinkSpeak URL with DHT sensor data and random number
113       WiFiClient client;
114       HTTPClient http;
115
116       String serverPath_send = serverName_send + "&field1=" + String(
              ↪ temperature)
117                                                + "&field2=" + String(
                                                     ↪ humidity)
118                                                + "&field3=" + String(
                                                     ↪ random(randomNumber))
                                                     ↪ ;
119
120       // Send the HTTP GET request to ThinkSpeak
121       http.begin(client, serverPath_send.c_str());
122       int httpResponseCode = http.GET();
123
124       // Handle HTTP response
125       if (httpResponseCode > 0) {
126         Serial.print("HTTP␣Response␣code:␣");
127         Serial.println(httpResponseCode);
128         String payload = http.getString();
129         Serial.println(payload);
130       } else {
131         Serial.print("Error␣code:␣");
132         Serial.println(httpResponseCode);
133       }
134
135       // Free resources
136       http.end();
137     } else {
138       Serial.println("WiFi␣Disconnected");
139     }
140
141     // Update lastTime to current time
142     lastTime = millis();
143   }
144
145   // You can also add other functionality like blinking an LED
146   digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED on
147   delay(100);                       // Wait for a short period
148   digitalWrite(LED_BUILTIN, LOW);   // Turn the LED off
149   delay(100);
150 }
```

[language=C++]

2.5 Observations

- The DHT11 sensor successfully provided temperature and humidity readings after proper connection and configuration.



- OTA updates allowed for seamless wireless code updates without needing physical access to the NodeMCU, & Data was successfully uploaded to ThingSpeak, and the graphs for temperature, humidity, and random numbers were generated.

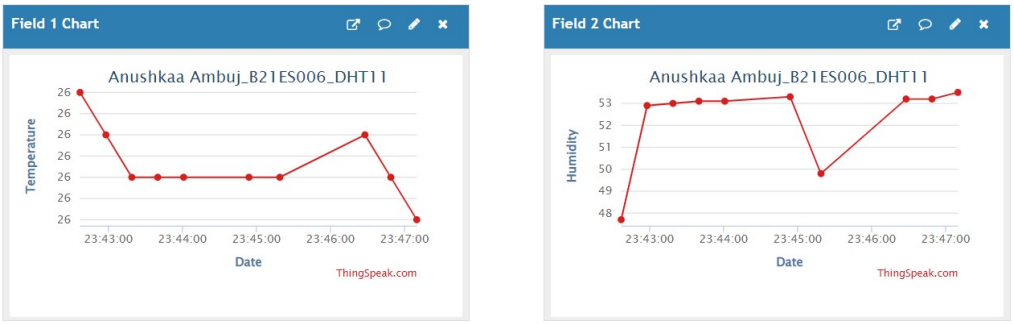


Figure 5: Temperature and Humidity Graph Plots

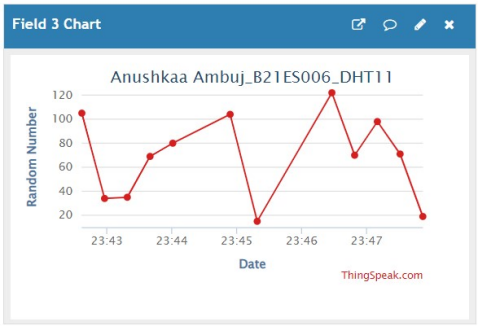


Figure 6: Plot of Random Number Generated

2.6 Results

OTA updates were implemented successfully.