**Internet of Things**

**Lab Report 9**

**Hafiz Ahmad**

**19l-1316**

**Section-7A2**

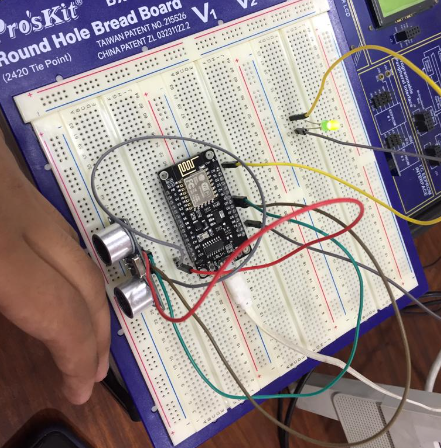
**Using Google Firebase as Real-Time Data Storage for IoT**

**INTRODUCTION:**

A NoSQL database hosted in the cloud, the Firebase Realtime Database lets you store and sync data in real time between your users.NEW:You can store, sync, and query global app data with Cloud Firestore.Firebase can simplify your life if you're one of those people or want your IoT app prototype done quickly.We can use the Realtime Database to connect a smart device to the Internet of Things and have it periodically send data to the application.To get your data and sync it with the majority of databases, you must make HTTP calls.The majority of databases only provide data upon request.Your application is not connecting to Firebase using standard HTTP.A WebSocket connection is being used.WebSockets are significantly quicker than HTTP.You try not to need to settle on individual WebSocket decisions, since one attachment association is bounty.As quickly as your client's network can handle it, that one WebSocket automatically synchronizes all of your data.Firebase immediately sends you updated data.All connected clients receive the updated data almost immediately after your client saves a change to the data.Directly from the client, Firebase Storage offers a straightforward method for saving binary files, most commonly images, to Google Cloud Storage.Firebase Storage has its own set of security rules to keep everyone from accessing your GCloud bucket and give your authenticated clients full write access.

**OBJECTIVES:**

To learn about Using Google Firebase as Real-Time Data Storage for IoT.



**Lab code:**

/\*

    This sketch demonstrates how to set up a simple HTTP-like server.

    The server will set a GPIO pin depending on the request

<http://server_ip/gpio/0> will set the GPIO2 low,

<http://server_ip/gpio/1> will set the GPIO2 high

    server\_ip is the IP address of the ESP8266 module, will be

    printed to Serial when the module is connected.

\*/

#include <ESP8266WiFi.h>

#ifndef STASSID

#define STASSID "EE"

#define STAPSK  "ee123456"

#endif

const char\* ssid = STASSID;

const char\* password = STAPSK;

// Create an instance of the server

// specify the port to listen on as an argument

WiFiServer server(80);

void setup() {

  Serial.begin(115200);

  // prepare LED

  pinMode(LED\_BUILTIN, OUTPUT);

  digitalWrite(LED\_BUILTIN, 0);

  // Connect to WiFi network

  Serial.println();

  Serial.println();

  Serial.print(F("Connecting to "));

  Serial.println(ssid);

  WiFi.mode(WIFI\_STA);

  WiFi.begin(ssid, password);

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

    delay(500);

    Serial.print(F("."));

  }

  Serial.println();

  Serial.println(F("WiFi connected"));

  // Start the server

  server.begin();

  Serial.println(F("Server started"));

  // Print the IP address

  Serial.println(WiFi.localIP());

}

void loop() {

  // Check if a client has connected

  WiFiClient client = server.available();

  if (!client) {

    return;

  }

  Serial.println(F("new client"));

  client.setTimeout(5000); // default is 1000

  // Read the first line of the request

  String req = client.readStringUntil('\r');

  Serial.println(F("request: "));

  Serial.println(req);

  // Match the request

  int val;

  if (req.indexOf(F("/gpio/0")) != -1) {

    val = 0;

  } else if (req.indexOf(F("/gpio/1")) != -1) {

    val = 1;

  } else {

    Serial.println(F("invalid request"));

    val = digitalRead(LED\_BUILTIN);

  }

int toggle, b=0;

toggle = digitalRead(LED\_BUILTIN);

  // Set LED according to the request

 if (val==0)

 {

  toggle = 1;

  digitalWrite(LED\_BUILTIN, toggle);

  delay(2000);

 }

 else if(val==1)

 {

  toggle=0;

  digitalWrite(LED\_BUILTIN, toggle);

   delay(2000);

  }

  else

  {

    }

  // read/ignore the rest of the request

  // do not client.flush(): it is for output only, see below

  while (client.available()) {

    // byte by byte is not very efficient

    client.read();

  }

  // Send the response to the client

  // it is OK for multiple small client.print/write,

  // because nagle algorithm will group them into one single packet

  client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nGPIO is now "));

  client.print((toggle) ? F("high") : F("low"));

  client.print(F("<br><br>Click <a href='http://"));

  client.print(WiFi.localIP());

  client.print(F("/gpio/1'>here</a> to switch LED GPIO on, or <a href='http://"));

  client.print(WiFi.localIP());

  client.print(F("/gpio/0'>here</a> to switch LED GPIO off.</html>"));

  // The client will actually be \*flushed\* then disconnected

  // when the function returns and 'client' object is destroyed (out-of-scope)

  // flush = ensure written data are received by the other side

  Serial.println(F("Disconnecting from client"));

}

**Application:**

According to Firebase Use Cases, these are ten of the best companies and startups using Firebase.

* Half brick.
* Alibaba.
* The New York Times.
* The Economist.
* Taoist.
* Fabulous.
* One football.
* NPR

MongoDB is a more robust document database known for high performance and best-in-class security and has several advantages over Firebase. For example, MongoDB can be operated on-premises or in the cloud (using MongoDB Atlas, or self-managed cloud MongoDB), while Firebase is purely a cloud database service.

**Issues:**

we never find any issue regarding this lab.

**Conclusion:**

In this lab, we interface an ultrasonic sensor with a NodeMCU and save the derived data in Google Firebase. In order to regulate the led, read the data from your database and establish a threshold value. Print the data from the database and the led light's state on the serial monitor using Google Firebase as the real-time data storage for the Internet of Things.