```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <WiFiUdp.h>

unsigned int localPort = 2390; // local port to listen for UDP
packets
IPAddress timeServerIP; // time.nist.gov NTP server address
const char* ntpServerName = "time.nist.gov";
const int NTP_PACKET_SIZE = 48; // NTP time stamp is in the first 48
bytes of the message
byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and
outgoing packets

WiFiUDP udp;

#define FIREBASE_HOST "****"
#define FIREBASE_AUTH "****"
#define WIFI_SSID "****"
#define WIFI_PASSWORD "****"

unsigned long sendNTPpacket(IPAddress& address);

void setup() {

  Serial.begin(9600);
  // Connect to WiFi network
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  udp.begin(localPort);
}

void loop() {

  Serial.print(Firebase.getInt("/Switch_Status/Switch/"));

  // Getting Data from Arduino via Serial Communication
  String data = "";
  String temp = "";
  String humid = "";

  // TX-2(A), RX-3(A)
  while(Serial.available()){
    data += (char)Serial.read();
    if(data.length()==4){
```

```
      temp += data[0];
      temp += data[1];
      humid += data[2];
      humid += data[3];

      // Firebase Library already has ArduinoJson.h in its CORE
      StaticJsonBuffer<200> jsonBuffer;
      JsonObject& root = jsonBuffer.createObject();
      root["Temp"] = temp;
      root["Humid"] = humid;
      root["Day"] = getTime()/86400;
      Firebase.set("/Sensors/", root);
    }
  }

  delay(200);
}

 // Checkout NTPClient Example
 // ESP8266WiFi/examples/NTPClient/NTPClient.ino
unsigned long getTime(){
  //get a random server from the pool
  WiFi.hostByName(ntpServerName, timeServerIP);

  sendNTPpacket(timeServerIP); // send an NTP packet to a time
server

  if (udp.parsePacket()) {
    // We've received a packet, read the data from it
    udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into
the buffer

    //the timestamp starts at byte 40 of the received packet and is
four bytes,
    // or two words, long. First, esxtract the two words:

    unsigned long highWord = word(packetBuffer[40],
packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42],
packetBuffer[43]);
    // combine the four bytes (two words) into a long integer
    // this is NTP time (seconds since Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
    // now convert NTP time into everyday time:
    // Unix time starts on Jan 1 1970. In seconds, that's
2208988800:
    const unsigned long seventyYears = 2208988800UL;
    // subtract seventy years:
    unsigned long epoch = secsSince1900 - seventyYears;
    return epoch;
  }
}

// send an NTP request to the time server at the given address
```

```
unsigned long sendNTPpacket(IPAddress& address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  udp.beginPacket(address, 123); //NTP requests are to port 123
  udp.write(packetBuffer, NTP_PACKET_SIZE);
  udp.endPacket();
}
```