```
#include <TinyGPS++.h> // library for GPS module
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
TinyGPSPlus gps;  // The TinyGPS++ object
SoftwareSerial ss(4, 5); // The serial connection to the GPS device

const char* ssid = "ja3"; //ssid of your wifi
const char* password = "12121212"; //password of your wifi
float latitude , longitude,alti ;
int year , month , date, hour , minute , second;
String date_str , time_str , lat_str , lng_str,alt_str;
double alt_m_val;
int pm;
String header;
String output5State = "off";
WiFiServer server(80);

const int output5 = D6;

unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup()
{

  pinMode(output5, OUTPUT);
  digitalWrite(output5, LOW);
  Serial.begin(115200);
  ss.begin(9600);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password); //connecting to wifi
  while (WiFi.status() != WL_CONNECTED)// while wifi not connected
  {
    delay(500);
    Serial.print("."); //print "...."
  }
  Serial.println("");
  Serial.println("WiFi connected");
  server.begin();
```

```
  Serial.println("Server started");
  Serial.println(WiFi.localIP());  // Print the IP address
}


void loop()
{
  while (ss.available() > 0) //while data is available
    if (gps.encode(ss.read())) //read gps data
    {
      if (gps.location.isValid()) //check whether gps location is valid
      {
        latitude = gps.location.lat();
        lat_str = String(latitude , 6); // latitude location is stored in a string
        longitude = gps.location.lng();
        lng_str = String(longitude , 6); //longitude location is stored in a string
      }
      if(gps.altitude.isValid())
      {
        alt_m_val = gps.altitude.meters();
        alt_str = String(alt_m_val,6);
      }

      if (gps.date.isValid()) //check whether gps date is valid
      {
        date_str = "";
        date = gps.date.day();
        month = gps.date.month();
        year = gps.date.year();
        if (date < 10)
          date_str = '0';
        date_str += String(date);// values of date,month and year are stored in a string
        date_str += " / ";

        if (month < 10)
          date_str += '0';
        date_str += String(month); // values of date,month and year are stored in a string
        date_str += " / ";
        if (year < 10)
          date_str += '0';
        date_str += String(year); // values of date,month and year are stored in a string
      }
      if (gps.time.isValid())  //check whether gps time is valid
      {
```

```
      time_str = "";
      hour = gps.time.hour();
      minute = gps.time.minute();
      second = gps.time.second();
      minute = (minute + 30); // converting to IST
      if (minute > 59)
      {
        minute = minute - 60;
        hour = hour + 1;
      }
      hour = (hour + 5) ;
      if (hour > 23)
        hour = hour - 24;   // converting to IST
      if (hour >= 12)  // checking whether AM or PM
        pm = 1;
      else
        pm = 0;
      hour = hour % 12;
      if (hour < 10)
        time_str = '0';
      time_str += String(hour); //values of hour,minute and time are stored in a string
      time_str += " : ";
      if (minute < 10)
        time_str += '0';
      time_str += String(minute); //values of hour,minute and time are stored in a string
      time_str += " : ";
      if (second < 10)
        time_str += '0';
      time_str += String(second); //values of hour,minute and time are stored in a string
      if (pm == 1)
        time_str += " PM ";
      else
        time_str += " AM ";
    }
  }

 WiFiClient client = server.available(); // Check if a client has connected
  if (!client)
  {
    return;
  }
  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n <!DOCTYPE html> <html>
<head> <title>Smart Tag</title> <style>";
```

```
  s += "a:link {background-color: YELLOW;text-decoration: none;}";
  s += "table, th, td </style> </head> <body> <h1  style=";
  s += "font-size:300%;";
  s += " ALIGN=CENTER>Smart Tag</h1>";
  s += "<p ALIGN=CENTER style=""font-size:150%;""";
  s += "> <b>Location Details</b></p> <table ALIGN=CENTER style=";
  s += "width:50%";
  s += "> <tr> <th>Latitude :</th>";
  s += "<td ALIGN=CENTER >";
  s += lat_str;
  s += "</td> </tr> <tr> <th>Longitude :</th> <td ALIGN=CENTER >";
  s += lng_str;
  s += "</td> </tr> <tr> <th>Altitude :</th> <td ALIGN=CENTER >";
  s += alt_str;
  s += "</td> </tr> <tr>  <th>Date :</th> <td ALIGN=CENTER >";
  s += date_str;
  s += "</td></tr> <tr> <th>Time :</th> <td ALIGN=CENTER >";
  s += time_str;
  s += "</td>  </tr> </table> ";

  s += "</body> </html>";

  client.print(s); // all the values are send to the webpage
  delay(100);

  if (client) {                             // If a new client connects,
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the
client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
           /* client.println("HTTP/1.1 200 OK");
```

```
            client.println("Content-type:text/html");
            client.println("Connection: close");*/
            client.println();

            // turns the GPIOs on and off
            if (header.indexOf("GET /5/on") >= 0) {
              Serial.println("GPIO 5 on");
              output5State = "on";
              digitalWrite(output5, HIGH);
            } else if (header.indexOf("GET /5/off") >= 0) {
              Serial.println("GPIO 5 off");
              output5State = "off";
              digitalWrite(output5, LOW);
            }

             // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to fit your
preferences
            client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto;
text-align: center;}");
            client.println(".button { background-color: #195B6A; border: none; color: white; padding:
16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color: #77878A;}</style></head>");

            // Web Page Heading
            client.println("<body><h2> Alert </h2>");

            // Display current state, and ON/OFF buttons for GPIO 5
            client.println("<p> Buzzer & LED - State : " + output5State + "</p>");
            // If the output5State is off, it displays the ON button
            if (output5State=="off") {
              client.println("<p><a href=\"/5/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/5/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }
              client.println("</br></br></br></br><h4>By - Robotix.io  </h4>");
              client.println("</body></html>");
```

```
            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```