LAB REPORT

IRE 212 : IoT Architecture and Technologies Sessional

PREPARED BY

Mehrin Farzana
ID: 2101013
Session: 2021-2022
Date: 4/12/2024

SUPERVISED BY

Suman Saha
Assistant Professor & Chairman
Department of IRE, BDU

BANGABANDHU SHEIKH MUJIBUR RAHMAN DIGITAL UNIVERSITY

(BDU)

Lab Report-11

Experiment Title: GPS Module Interfacing With Arduino UNO.

Theory:

GPS (Global Positioning System) is a satellite-based navigation system that provides precise location, altitude, and time data. It consists of a network of satellites that transmit signals to GPS receivers. By analysing these signals, a receiver can calculate its exact position on Earth.

The GPS module used in this experiment receives signals from satellites and sends raw NMEA (National Marine Electronics Association) data to the Arduino via a serial connection. The TinyGPS++ library is employed to decode this data into useful information such as latitude, longitude, altitude, and time.

Key Functions of GPS:

- Location Tracking: Determining latitude and longitude coordinates.
- Altitude Measurement: Providing height information above sea level.
- Time Data: Receiving UTC time from satellites.

Components:

Components Required for this Project are:

- Arduino UNO: The microcontroller board used for interfacing and data processing.
- GPS Module (e.g., NEO-6M): Receives signals from satellites and provides NMEA data.
- TinyGPS++ Library: Parses and extracts meaningful data from raw GPS data.
- USB Cable: For powering the Arduino and monitoring serial output.
- Connecting Wires: To establish a connection between the Arduino and GPS module.
- Computer with Arduino IDE: Used for programming and viewing the output.

Circuit Diagram:

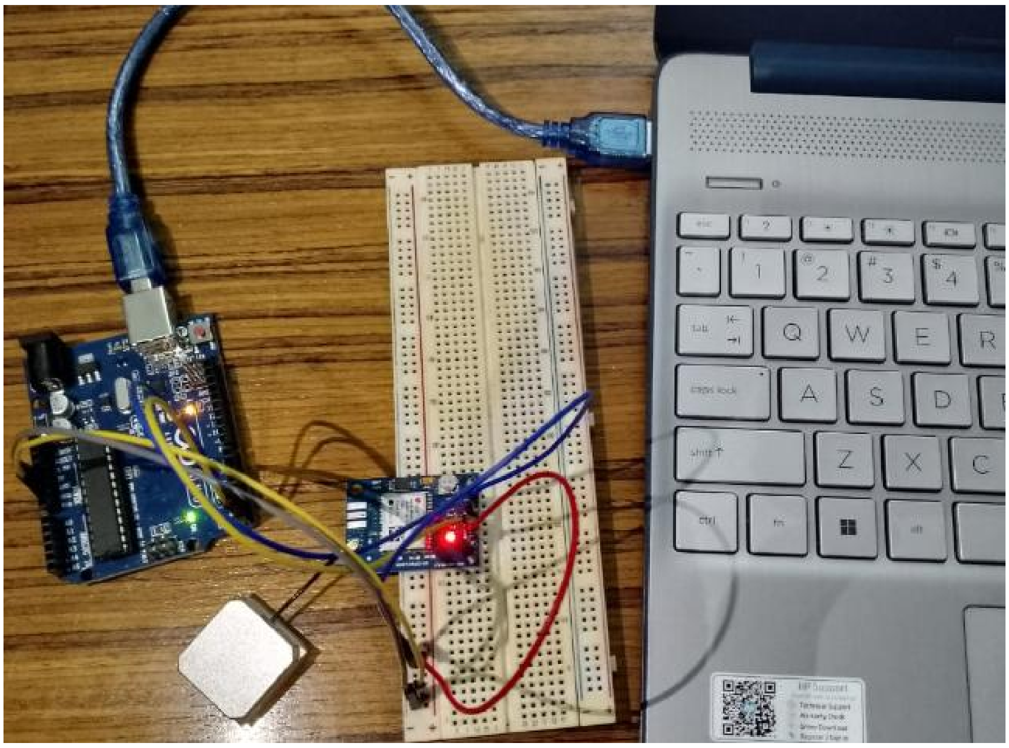


Fig-1: GPS Module Interfacing With Arduino UNO

Code:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
/* Create object named bt of the class SoftwareSerial */
SoftwareSerial GPS_SoftSerial(4, 3);/* (Rx, Tx) */
/* Create an object named gps of the class TinyGPSPlus */
TinyGPSPlus gps;
volatile float minutes, seconds;
volatile int degree, secs, mins;

void setup() {
  Serial.begin(9600); /* Define baud rate for serial communication */
   GPS_SoftSerial.begin(9600);  /*  Define  baud  rate  for  software  serial
communication */
}

void loop() {
        smartDelay(1000); /* Generate precise delay of 1ms */
        unsigned long start;
        double lat_val, lng_val, alt_m_val;
        uint8_t hr_val, min_val, sec_val;
        bool loc_valid, alt_valid, time_valid;
        lat_val = gps.location.lat(); /* Get latitude data */
```

```
         loc_valid = gps.location.isValid(); /* Check if valid location
data is available */
        lng_val = gps.location.lng(); /* Get longtitude data */
          alt_m_val = gps.altitude.meters();  /* Get altitude data in
meters */
         alt_valid = gps.altitude.isValid(); /* Check if valid altitude
data is available */
        hr_val = gps.time.hour(); /* Get hour */
        min_val = gps.time.minute();  /* Get minutes */
        sec_val = gps.time.second();  /* Get seconds */
         time_valid = gps.time.isValid();  /* Check if valid time data
is available */
        if (!loc_valid)
        {
          Serial.print("Latitude : ");
          Serial.println("*****");
          Serial.print("Longitude : ");
          Serial.println("*****");
        }
        else
        {
          DegMinSec(lat_val);
          Serial.print("Latitude in Decimal Degrees : ");
          Serial.println(lat_val, 6);
          Serial.print("Latitude in Degrees Minutes Seconds : ");
          Serial.print(degree);
          Serial.print("\t");
          Serial.print(mins);
          Serial.print("\t");
          Serial.println(secs);
           DegMinSec(lng_val); /* Convert the decimal degree value into
degrees minutes seconds form */
          Serial.print("Longitude in Decimal Degrees : ");
          Serial.println(lng_val, 6);
          Serial.print("Longitude in Degrees Minutes Seconds : ");
          Serial.print(degree);
          Serial.print("\t");
          Serial.print(mins);
          Serial.print("\t");
          Serial.println(secs);
        }
        if (!alt_valid)
        {
          Serial.print("Altitude : ");
          Serial.println("*****");
        }
        else
        {
          Serial.print("Altitude : ");
          Serial.println(alt_m_val, 6);
        }
        if (!time_valid)
        {
          Serial.print("Time : ");
          Serial.println("*****");
        }
        else
        {
          char time_string[32];
              sprintf(time_string, "Time : %02d/%02d/%02d \n", hr_val,
min_val, sec_val);
          Serial.print(time_string);
        }
}

static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
     while (GPS_SoftSerial.available())  /* Encode data read from GPS
while data is available on serial port */
      gps.encode(GPS_SoftSerial.read());
/* Encode basically is used to parse the string received by the GPS and
to store it in a buffer so that information can be extracted from it */
  } while (millis() - start < ms);
}

void DegMinSec( double tot_val)    /* Convert data in decimal degrees
into degrees minutes seconds form */
{
  degree = (int)tot_val;
  minutes = tot_val - degree;
  seconds = 60 * minutes;
  minutes = (int)seconds;
  mins = (int)minutes;
  seconds = seconds - minutes;
  seconds = 60 * seconds;
  secs = (int)seconds;
}
```

Explanation of Code:

Initialization:

Libraries:

The TinyGPS++ library handles parsing and extracting meaningful GPS data from raw NMEA strings received from the GPS module.

The SoftwareSerial library enables serial communication on arbitrary digital pins for GPS data transmission.

Objects and Pins:

GPS_SoftSerial is initialized for software serial communication on pins 4 (Rx) and 3 (Tx).

gps is the object used to decode GPS data.

Data Processing in the Loop:

The loop() function continuously fetches and processes data from the GPS module.

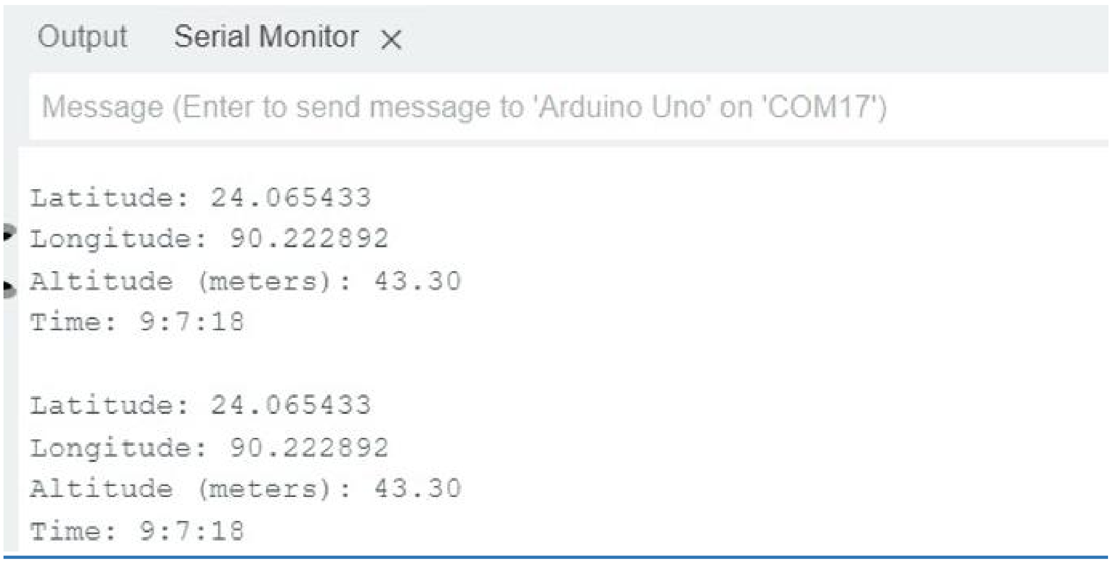
Retrieve and Validate Data:

The gps.location.lat() and gps.location.lng() methods retrieve latitude and longitude in decimal degrees, and their validity is checked.Altitude and time are similarly extracted and validated.

Invalid Data Handling:

For invalid data (e.g., no GPS signal), placeholders ("*****") are displayed for latitude, longitude, altitude, or time.

Output:



```
Latitude: 24.065433
Longitude: 90.222892
Altitude (meters): 43.30
Time: 9:7:18

Latitude: 24.065433
Longitude: 90.222892
Altitude (meters): 43.30
Time: 9:7:18
```

Fig-2: Serial Monitor reading of GPS Module Interfacing

Experiment Title: GPS Module Interfacing With ESP32

Circuit Diagram:

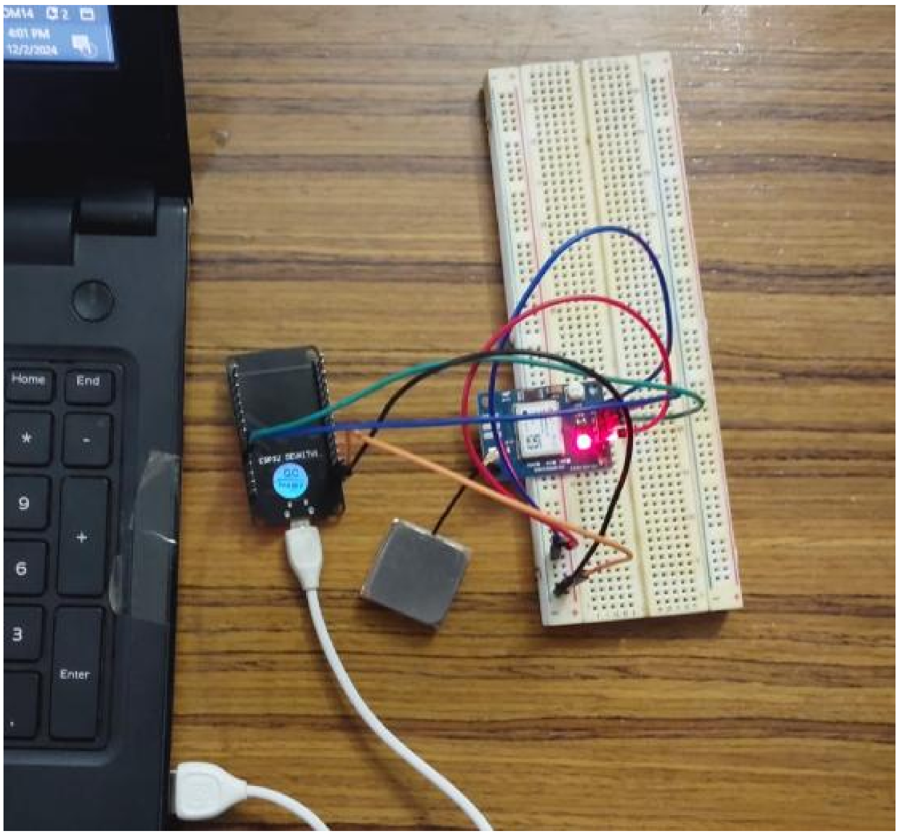


Fig-3: GPS Module Interfacing With ESP32

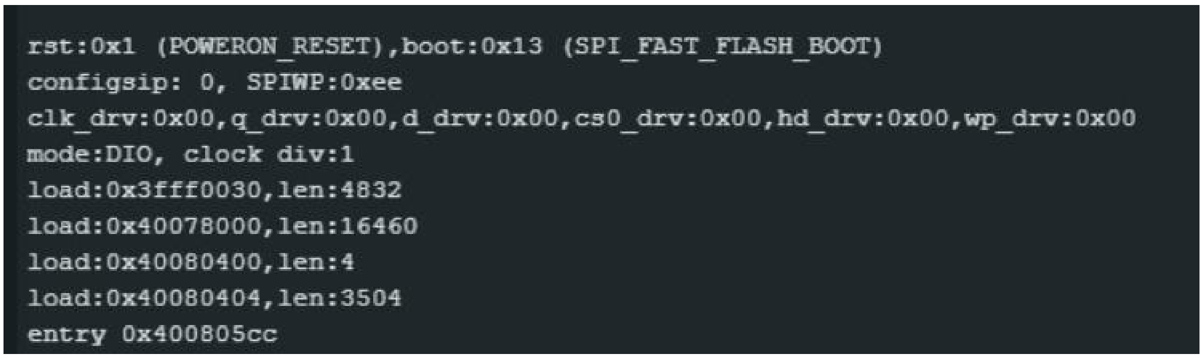
Code:

```
#include <WiFi.h>
#include <TinyGPS++.h>

// GPS Pins
#define RXPin 16
#define TXPin 17
HardwareSerial GPS(1); // Use Serial1 for GPS

// GPS Library
TinyGPSPlus gps;

// Wi-Fi Credentials
const char* ssid = "HCN";
const char* password = "mehedi3311";
```

```
// Server Details
const  char*  server  =  "http://example.com";  //  Replace  with  your
server's endpoint

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  delay(100);

  // Initialize GPS Serial
  GPS.begin(9600, SERIAL_8N1, RXPin, TXPin);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWi-Fi Connected!");
}

void loop() {
  while (GPS.available()) {
    if (gps.encode(GPS.read())) { // Process GPS data
      if (gps.location.isUpdated()) {
        // Get Latitude and Longitude
        String latitude = String(gps.location.lat(), 6);
        String longitude = String(gps.location.lng(), 6);

        Serial.print("Latitude: ");
        Serial.println(latitude);
        Serial.print("Longitude: ");
        Serial.println(longitude);

        // Send data to the server
        sendToServer(latitude, longitude);
      }
    }
  }
}

void sendToServer(String latitude, String longitude) {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;

    if (client.connect(server, 80)) {
      // Prepare data to send
```

```
          String  postData  =  "latitude="  +  latitude  +  "&longitude="  +
longitude;

      // HTTP POST Request
      client.println("POST /gpsdata HTTP/1.1");
      client.println("Host: example.com");
      client.println("Content-Type: application/x-www-form-urlencoded");
      client.print("Content-Length: ");
      client.println(postData.length());
      client.println();
      client.println(postData);

      Serial.println("Data sent to server: " + postData);
    } else {
      Serial.println("Failed to connect to server");
    }
  } else {
    Serial.println("Wi-Fi not connected");
  }
}
```

Output:



```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4832
load:0x40078000,len:16460
load:0x40080400,len:4
load:0x40080404,len:3504
entry 0x400805cc
```

Fig-4: Serial Monitor reading of GPS Module Interfacing