ESP32 code to fetch GPS coordinates, SMS them to your mobile device, and open the location in Google Maps

The ESP32 code will:

- Initialize the GPS module.
- Read GPS data (latitude, longitude).
- Format the coordinates into a URL suitable for Google Maps.
- Transmit the URL via SMS to your mobile.

Contents

Component requirements:..... 2
Software requirements: 3
Connections: 3
Code: 4
AT Commands: 7
Procedure:..... 7
Output:..... 8

Component requirements:

ESP32-DEVKITC-32E - ESP32-WROOM-32E Development Board 4MB Flash PCB Antenna:

7Semi EC200U-CN LTE 4G GPS GNSS Mini Industrial Modem:

GPS External Active Antenna (3m) – SMA:

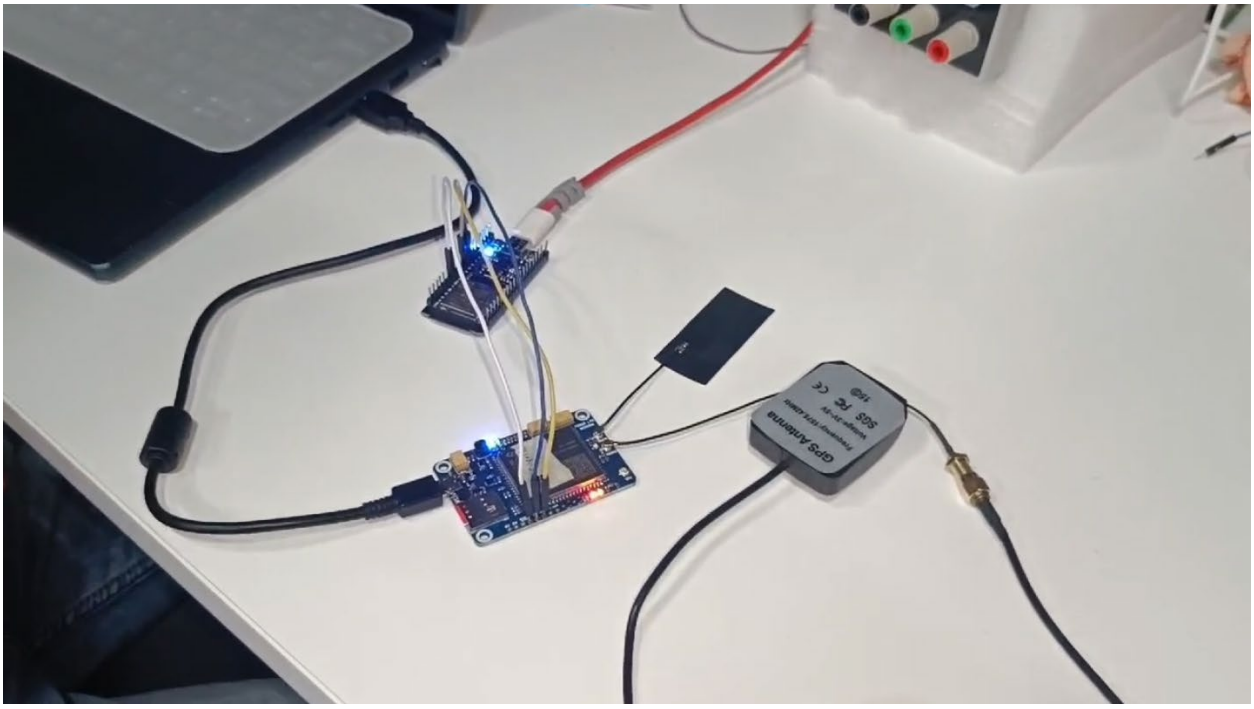
Software requirements:

Arduino IDE 2.3.2

Connections:

ESP 32	EC200U
GND	GND
GPIO 16 (RX2)	TX
GPIO 17 (TX2)	RX

- Antenna to Main of EC200U
- GPS Antenna to GNSS of EC200U
- Power Supply to both ESP32 and EC200U through your system



Code:

```
#include <HardwareSerial.h>

// Define the serial connection to the EC200U module
HardwareSerial ec200u(2); // Using UART2 for EC200U

String phoneNumber = "+123456789"; // Replace with the recipient's phone number

void setup() {
  // Initialize serial communications
  Serial.begin(9600);
  ec200u.begin(115200, SERIAL_8N1, 16, 17); // RX: GPIO 16, TX: GPIO 17 for
EC200U

  // Wait for the EC200U module to initialize
  delay(3000);

  // Test the communication with the EC200U module
  if (!sendATCommand("AT", 1000)) {
    Serial.println("Error: Communication with EC200U module failed.");
    return;
  }

  // Set SMS text mode
  if (!sendATCommand("AT+CMGF=1", 1000)) {
    Serial.println("Error: Failed to set SMS text mode.");
    return;
  }

  // Enable GPS
  if (!sendATCommand("AT+QGPS=1", 2000)) {
    Serial.println("Error: Failed to enable GPS.");
    return;
  }

  Serial.println("EC200U module initialized successfully.");
}

void loop() {
  // Get the GPS data
  String gpsData = sendATCommand("AT+QGPSLOC=2", 10000); // Increased timeout
for GPS data retrieval

  Serial.println("GPS Data: " + gpsData);
```

```
  if (gpsData.indexOf("+QGPSLOC:") != -1) {
    // Parse the GPS data
    String latitude = parseLatitude(gpsData);
    String longitude = parseLongitude(gpsData);

    // Format the GPS data into a Google Maps URL
    String url = "https://maps.google.com/?q=" + latitude + "," + longitude;

    // Send the SMS
    if (sendSMS(phoneNumber, url)) {
      Serial.println("SMS sent successfully.");
    } else {
      Serial.println("Failed to send SMS.");
    }

    // Wait for some time before sending the next SMS
    delay(60000); // Wait for 1 minute
  } else {
    Serial.println("Failed to retrieve GPS data.");
    delay(10000); // Wait for 10 seconds before retrying
  }
}

String sendATCommand(String command, int timeout) {
  ec200u.println(command);
  delay(100); // Small delay for command execution

  String response = "";
  long startTime = millis();

  while (millis() - startTime < timeout) {
    while (ec200u.available()) {
      response += ec200u.readString();
    }
    if (response.length() > 0) {
      break; // Exit loop if response is received
    }
  }

  Serial.println("AT Command: " + command);
  Serial.println("AT Command Response: " + response);

  return response;
}
```

```
String parseLatitude(String gpsData) {
  int startIndex = gpsData.indexOf(",") + 1;
  int endIndex = gpsData.indexOf(",", startIndex);
  String latitude = gpsData.substring(startIndex, endIndex);
  return latitude;
}

String parseLongitude(String gpsData) {
  // Find the start index of the longitude value
  int startIndex = gpsData.indexOf(",", gpsData.indexOf(",") + 1) + 1; // Skip
the first two commas

  // Find the end index of the longitude value
  int endIndex = gpsData.indexOf(",", startIndex); // Find the next comma after
the start index

  // Extract the longitude substring
  String longitude = gpsData.substring(startIndex, endIndex);

  // Trim leading and trailing spaces, if any
  longitude.trim();

  return longitude;
}

bool sendSMS(String phoneNumber, String message) {
  // Start the SMS
  ec200u.println("AT+CMGS=\"" + phoneNumber + "\"");
  delay(1000);

  // Send the SMS message
  ec200u.println(message);
  delay(1000);

  // End the SMS
  ec200u.write(26); // Ctrl+Z to send SMS
  delay(1000);

  // Check for "OK" response
  String response = sendATCommand("", 1000);

  if (response.indexOf("OK") != -1) {
    return true; // SMS sent successfully
  } else {
```

```
    Serial.println("Error: Failed to send SMS.");
    return false;
  }
}
```

AT Commands:

Refer to this link for all the AT Commands and Error codes while executing the code and make changes accordingly,

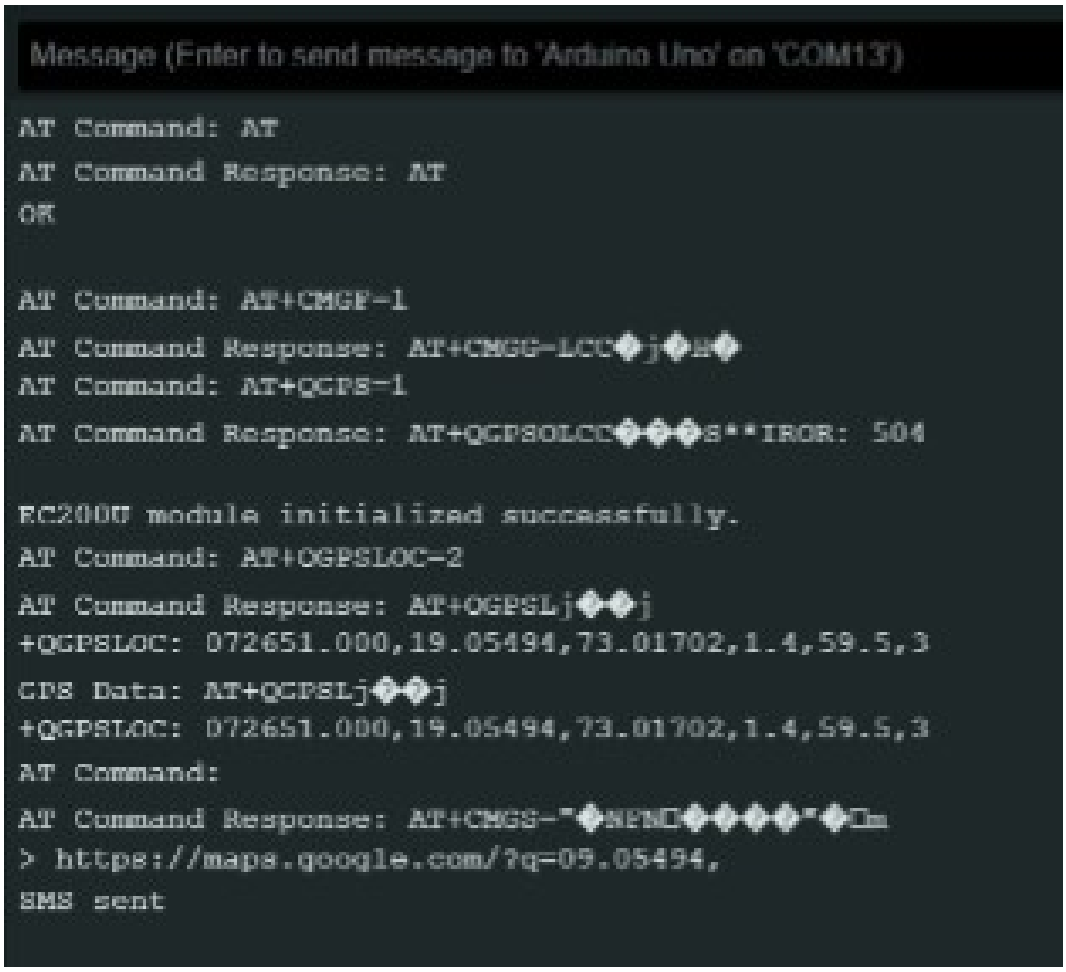
https://evelta.com/content/datasheets/EC200-GNSS-MANUAL.pdf

Procedure:

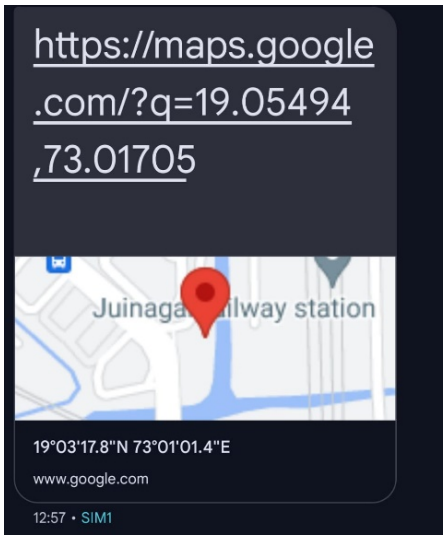
1. Make connections between the components as mentioned above.
2. Apply power supply to ESP32 and EC200U after completing all the connections.
3. Write the Arduino code in Arduino IDE, verify and upload it.
4. Open Serial monitor to see the output.
5. You can observe in the serial monitor that the AT commands are being sent and the corresponding action being taken.
6. An SMS will be sent after every 1 min to the recipient's phone number after the code is executed in a URL format.
7. Click on the link to see the location of the sender in Google Maps.

Output:

Serial Monitor in Arduino IDE:



SMS received in recipient's phone:



Clicking on the URL link redirects you to google maps: