**RMIT UNIVERSITY**

# Report – GPS NEO8M

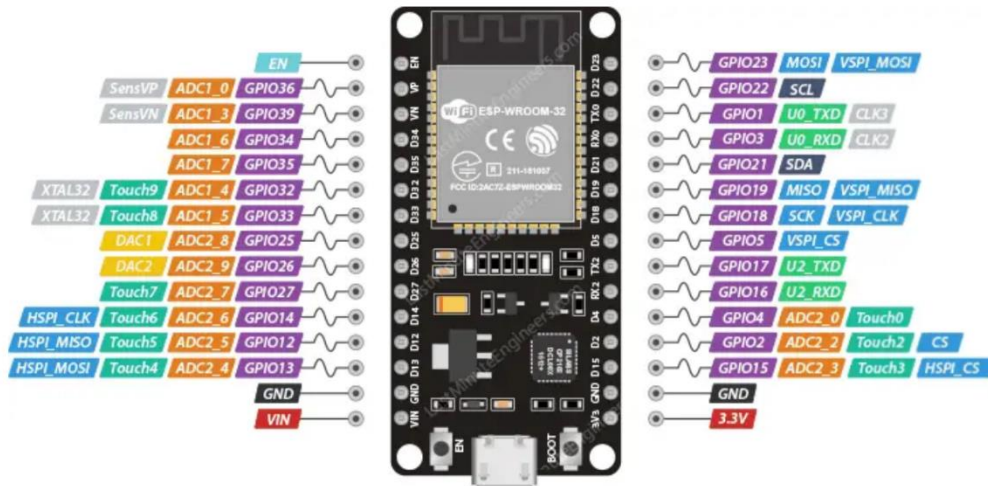October 2023

Dinh-Son Vu

School of Science, Engineering, and Technology

What's next...

# Table of Contents

# 1. Introduction

This document introduces how to use the GPS module NEO-8M. This sensor is essential to navigate the drone autonomously.

# 2. ESP32 Pinout



| Label | GPIO | Safe to use? | Reason |
|-------|------|--------------|--------|
| D0 | 0 | ⚠️ | must be HIGH during boot and LOW for programming |
| TX0 | 1 | ❌ | Tx pin, used for flashing and debugging |
| D2 | 2 | ⚠️ | must be LOW during boot and also connected to the on-board LED |
| RX0 | 3 | ❌ | Rx pin, used for flashing and debugging |
| D4 | 4 | ✅ | |
| D5 | 5 | ⚠️ | must be HIGH during boot |
| D6 | 6 | ❌ | Connected to Flash memory |
| D7 | 7 | ❌ | Connected to Flash memory |
| D8 | 8 | ❌ | Connected to Flash memory |
| D9 | 9 | ❌ | Connected to Flash memory |
| D10 | 10 | ❌ | Connected to Flash memory |
| D11 | 11 | ❌ | Connected to Flash memory |
| D12 | 12 | ⚠️ | must be LOW during boot |
| D13 | 13 | ✅ | |
| D14 | 14 | ✅ | |
| D15 | 15 | ⚠️ | must be HIGH during boot, prevents startup log if pulled LOW |

| Label | GPIO | Safe to use? | Reason |
|-------|------|--------------|--------|
| D15 | 15 | ⚠️ | must be HIGH during boot, prevents startup log if pulled LOW |
| RX2 | 16 | ✅ | |
| TX2 | 17 | ✅ | |
| D18 | 18 | ✅ | |
| D19 | 19 | ✅ | |
| D21 | 21 | ✅ | |
| D22 | 22 | ✅ | |
| D23 | 23 | ✅ | |
| D25 | 25 | ✅ | |
| D26 | 26 | ✅ | |
| D27 | 27 | ✅ | |
| D32 | 32 | ✅ | |
| D33 | 33 | ✅ | |
| D34 | 34 | ⚠️ | Input only GPIO, cannot be configured as output |
| D35 | 35 | ⚠️ | Input only GPIO, cannot be configured as output |
| VP | 36 | ⚠️ | Input only GPIO, cannot be configured as output |
| VN | 39 | ⚠️ | Input only GPIO, cannot be configured as output |

# 3. GPS NEO-8M Pinout

The GPS module uses the UART communication, which is another name for serial communication, just like the USB connection. The ESP32 has three UART communication channel. UART0 is for programming and debugging (USB port with the computer). UART1 can be used, but is sometimes used for other tasks (SPI flashing). Thus, use UART2 preferably.

The connection with ESP32 is slightly non-intuitive: the TX of the ESP32 must be connected to the RX of the GPS module and vice-versa. For example, for the I2C communication with the MPU6050 module, the SDA pin of the ESP32 is connected to the SDA pint of the MPU6050.

The GPS module does not work indoor and may take some time to get a satellite signal. You may enter the location data on Google Maps to see the accuracy of the sensor.
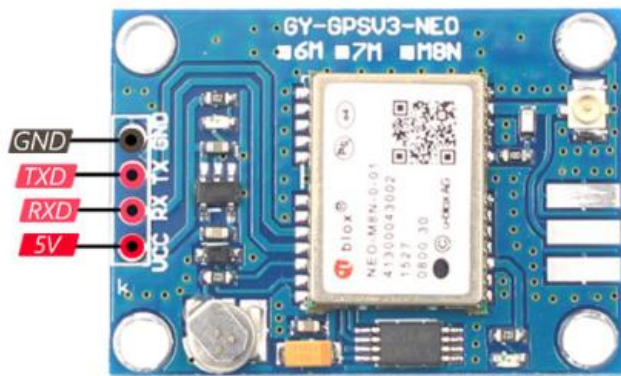
| GPS | ESP32 |
|-----|-------|
| VCC | 3V3 |
| GND | GND |
| TX | 16 |
| RX | 17 |

```cpp
#include <Arduino.h>
#include <TinyGPSPlus.h>
// =================================================================
// Variable declaration
// =================================================================
#define RXD2 16
#define TXD2 17
TinyGPSPlus gps;
// =================================================================
// Function Declaration
// =================================================================
void Init_Serial(); // Init the serial monitor
void Get_GPSData(); // Get the GPS data
void displayInfo(); // Display info from the GPS
// =================================================================
// Setup function
// =================================================================
void setup()
{
  Init_Serial();
  // Serial2.begin(baud-rate, protocol, RX pin, TX pin);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
}
// =================================================================
// Loop function
// =================================================================
void loop()
{
  Get_GPSData();
}
// =================================================================
// Function Definition
// =================================================================
void Init_Serial()
{
  Serial.begin(115200);
  while (!Serial)
    ;
}
// =================================================================
void Get_GPSData()
{
  while (Serial2.available() > 0)
    if (gps.encode(Serial2.read()))
      displayInfo();
}
// =================================================================
```

```
void displayInfo()
{
  Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F("  Altitude: "));
  if (gps.altitude.isValid())
  {
    Serial.print(gps.altitude.meters());
    Serial.print(F("m"));
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F("  Date/Time: "));
  if (gps.date.isValid())
  {
    Serial.print(gps.date.month());
    Serial.print(F("/"));
    Serial.print(gps.date.day());
    Serial.print(F("/"));
    Serial.print(gps.date.year());
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F(" "));
  if (gps.time.isValid())
  {
    if (gps.time.hour() < 10)
      Serial.print(F("0"));
    Serial.print(gps.time.hour());
    Serial.print(F(":"));
    if (gps.time.minute() < 10)
```

```
      Serial.print(F("0"));
    Serial.print(gps.time.minute());
    Serial.print(F(":"));
    if (gps.time.second() < 10)
      Serial.print(F("0"));
    Serial.print(gps.time.second());
    Serial.print(F("."));
    if (gps.time.centisecond() < 10)
      Serial.print(F("0"));
    Serial.print(gps.time.centisecond());
  }
  else
  {
    Serial.print(F("INVALID"));
  }
  Serial.println();
}
```