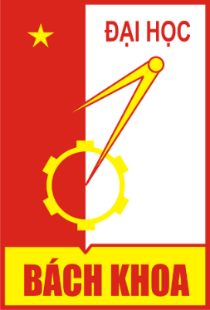
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN ĐIỆN TỬ - VIỄN THÔNG  
\*\*\*\*\* 🙠🕮🙢 \*\*\*\*\***



**BÁO CÁO BÀI TẬP LỚN**

**HỆ THỐNG NHÚNG VÀ THIẾT KẾ GIAO TIẾP NHÚNG**

**ĐỀ TÀI: THEO DÕI VẬT THỂ BẰNG MODULE GPS VÀ XÂY DỰNG WEBAPP ĐỂ HIỂN THỊ VỊ TRÍ THEO THỜI GIAN THỰC**

**Giảng viên hướng dẫn: Thầy Ngô Vũ Đức**

**Mã lớp: 124873 – Nhóm 13**

**Sinh viên thực hiện: Phạm Tiến Dũng 20172501**

**Phùng Tiến Đạt 20172456**

**Nguyễn Thanh Nam 20172701**

***Hà Nội, 6/2021***

**LỜI NÓI ĐẦU**

Từ xưa đến nay, nhân loại đã không ngừng học hỏi, tìm tòi và nghiên cứu để tạo ra những sản phẩm nhằm phục vụ cho sinh hoạt hằng ngày hay phục vụ cho mục đích nghiên cứu. Cùng với sự phát triển nhanh của công nghệ, hàng loạt ứng dụng được nghiên cứu và đưa vào sử dụng. Không chỉ giúp khắc phục các hạn chế về mặt thời gian, không gian, nó còn được xem là trợ thủ đắc lực để chúng ta làm việc nhanh, hiệu quả và chính xác.

Hiện tại, vấn đề định vị vật thể như xe cộ, đồ dùng cá nhân đắt tiền hay định vị trí của con em các phụ huynh học sinh đang không còn xa lạ với xã hội . Các vấn đề trên đang dần được con người tìm ra lời giải hiệu quả để khắc phục việc mất mát hoặc là bị lạc của trẻ nhỏ.

Qua những trải nghiệm thực tế trong suốt quá trình học tập tại trường Đại học Bách Khoa Hà Nội, kèm theo đó là mong muốn được học tập, nghiên cứu và vận dụng kiến thức đã được học kết hợp với vấn đề được nêu ra ở bên trên chúng em đã lựa chọn đề tài: “**Theo dõi vật thể bằng module GPS và xây dựng webapp để hiện thị vị trí theo thời gian thực**” để làm bài tập lớn của nhóm. Với những chức năng mà hệ thống mang lại, nó có thể người dùng có thể định vị được vị trí vật thể mà mình cần tìm kiếm.

Trong quá trình thực hiện đề tài đồ án tốt nghiệp này, nhóm chúng em xin trân thành cảm ơn **thầy Ngô Vũ Đức**, viện Điện tử - Viễn thông, trường Đại học Bách Khoa Hà Nội góp ý, giúp đỡ và chỉnh sửa cho đề tài để nhóm có thể hoàn thành được đề tài

Chúng em xin chân thành cảm ơn!

Mục lục

[TÓM TẮT ĐỀ TÀI 5](#_Toc75476579)

[LÍ DO CHỌN ĐỀ TÀI 6](#_Toc75476580)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 7](#_Toc75476581)

[1.1. Tổng quan về đề tài 7](#_Toc75476582)

[1.1.1. Phân tích đề tài 7](#_Toc75476583)

[1.1.2. Các chức năng của hệ thống 7](#_Toc75476584)

[1.1.3. Các yêu cầu về hệ thông 7](#_Toc75476585)

[1.2. Chuẩn giao tiếp truyền nhận UART 8](#_Toc75476586)

[1.2.1. Tổng quan về UART 8](#_Toc75476587)

[1.2.2. Packet UART 9](#_Toc75476588)

[1.3. Sơ lược về module GPS NEO7M 10](#_Toc75476589)

[1.3.1. Mô tả chung 10](#_Toc75476590)

[1.3.2. Các kiến thức chuyên sâu về GPS Neo7 11](#_Toc75476591)

[1.3.3. NMEA protocol 13](#_Toc75476592)

[1.4. Ngôn ngữ NodeJS 15](#_Toc75476593)

[1.4.1. Tổng quan Nodejs 15](#_Toc75476594)

[1.4.2. Các tính năng NodeJS 16](#_Toc75476595)

[1.4.3. Ứng dụng Nodejs 16](#_Toc75476596)

[1.5. Tổng quan về Socket.IO và Websocket 17](#_Toc75476597)

[1.5.1. Socket.IO 17](#_Toc75476598)

[1.5.2. WebSocket 17](#_Toc75476599)

[CHƯƠNG 2 THIẾT KẾ HỆ THỐNG 20](#_Toc75476600)

[2.1 Sơ đồ khối và chức năng các khối hệ thống 20](#_Toc75476601)

[2.2 Mô hình chi tiết hệ thống và tổng quan phần cứng hệ thống 20](#_Toc75476602)

[2.2.1 GPS Module NEO7M 20](#_Toc75476603)

[2.2.2 Module ESP8266 12E 22](#_Toc75476604)

[CHƯƠNG 3: TRIỂN KHAI HỆ THỐNG 25](#_Toc75476605)

[3.1 Thiết kế phần cứng 25](#_Toc75476606)

[3.2 Đọc dữ liệu từ module GPS NEO7M 26](#_Toc75476607)

[3.3 Đẩy dữ liệu đọc được thông qua ESP8266 lên sever 26](#_Toc75476608)

[3.3.1 Cài đặt esp8266 thành một client socket 26](#_Toc75476609)

[3.3.2 Esp8266 kết nối sever và đẩy lên sự kiện 28](#_Toc75476610)

[3.4 Triển khai Web app hiện thị vị trí trên map 29](#_Toc75476611)

[3.4.1 Server 29](#_Toc75476612)

[3.4.2.Client 31](#_Toc75476613)

[TEST HỆ THỐNG 34](#_Toc75476614)

[KẾT LUẬN 35](#_Toc75476615)

[TÀI LIỆU THAM KHẢO 36](#_Toc75476616)

**DANH MỤC HÌNH VẼ**

[Hình 1‑1 Sơ đồ hệ thống 7](#_Toc75476555)

[Hình 1‑2 Tín hiệu tương đương của UART và RS232 9](#_Toc75476556)

[Hình 1‑3 Packet UART 9](#_Toc75476557)

[Hình 1‑4 GPS NEO 7m 10](#_Toc75476558)

[Hình 1‑5 Block diagram neo 7m 11](#_Toc75476559)

[Hình 1‑6 Receiver config Neo7m 11](#_Toc75476560)

[Hình 1‑7 Operation neo7m 12](#_Toc75476561)

[Hình 1‑8 Diagram of ON/OFF operation 12](#_Toc75476562)

[Hình 1‑9 NMEA frame 14](#_Toc75476563)

[Hình 1‑10 Các chuẩn NMEA 15](#_Toc75476564)

[Hình 1‑11 Mô hình Socket.IO 17](#_Toc75476565)

[Hình 1‑12 websocket protocol 18](#_Toc75476566)

[Hình 2‑1 : sơ đồ khối hệ thống 20](#_Toc75476567)

[Hình 2‑2 output pin NEO7M 22](#_Toc75476568)

[Hình 2‑3 sơ đồ chân ESP8266 và giải thích 24](#_Toc75476569)

[Hình 3‑1 Sơ đồ nguyên lí 25](#_Toc75476570)

[Hình 3‑2 Sơ đồ mạch in 25](#_Toc75476571)

# TÓM TẮT ĐỀ TÀI

Quá trình phát triển đề tài:” **Theo dõi vật thể bằng module GPS và xây dựng webapp để hiện thị vị trí theo thời gian thực**” được diễn ra theo các bước từ thu thập yêu cầu thực tế, thu thập thông tin từ sách báo, tạp chí sau đó phân tích, thiết kế đến triển khai hệ thống. Hệ thống được triển khai thành trang web để thuận tiện cho người dùng có thể theo dõi được vị trí hiện tại.

Nội dung của đồ án này được triển khai làm 3 chương lớn, cụ thể các chương được trình bày như sau:

* Chương 1: Cơ sở lý thuyết

Trình bày một cách khái quát về hệ thống theo dõi vật thể và đưa ra các giải pháp để triển khai hệ thống cũng như đưa ra các cơ sở lý thuyết cần thiết để có thể triển khai hệ thống.

* Chương 2: Thiết kế hệ thống

Tiến hành thiết kế hệ thống về sơ đồ khối và chức năng các khối, đồng thời giới thiệu về những phần cứng cần thiết để triển khai hệ thống.

* Chương 3: Triển khai hệ thống

Tiến hành triển khai hệ thống về phần mềm và các kết nối cần thiết giữa các khối với nhau. Triển khai hệ thống trên server và xây dựng giao diện cho hệ thống.

# LÍ DO CHỌN ĐỀ TÀI

Ngày nay công nghệ đang được phát triển không ngừng như các công nghệ về GPS, viễn thông, Iot, Wifi,… Các công nghệ này giúp cho loài người ngày càng hiện đại và tân tiến hơn. Trong hệ thống định vị, các công nghệ trên giúp giám sát, hiển thị được vị trí vật thể cần tìm kiếm.

Hiện nay, tại Việt Nam, nhu cầu định vị vị trí vật thể ngày càng tăng. Song song với việc hiện đại hóa, đồ dùng cá nhân của con người ngày càng có giá trị, từ đó nhu cầu cần định vị vị trí là vô cùng cần thiết để phòng hờ việc mất cắp hoặc để quên, bên cạnh đó với hệ thống giao thông hiện nay như hệ thống giao thông xe buýt công cộng, việc định vị được vị trí xe buýt giúp các nhà xe có thể giám sát, vận hành các xe buýt được chính xác và hiệu quả hơn

Xuất phát từ nhu cầu đó, một hệ thống định vị sẽ được triển khai, sẽ giải quyết được các vấn đề trên.

Là những sinh viên viện Điện tử - Viễn thông, với những kiến thức đã học trong trường cùng với mong muốn thiết kế được một hệ thống nói trên cùng với sự giúp đỡ của thầy Ngô Vũ Đức, nhóm chúng em đã quyết định chọn đề tài **“**Theo dõi vật thể bằng module GPS và xây dựng webapp để hiện thị vị trí theo thời gian thực**”** làm đề tài bài tập lớn của nhóm

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## Tổng quan về đề tài

### Phân tích đề tài

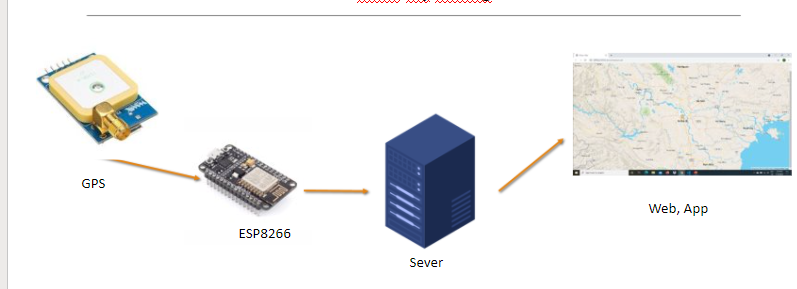
Hệ thống theo dõi vật thể theo thời gian thực sẽ cung cấp vị trí GPS trực tiếp của vật thể (ở đây là ví dụ về xe buýt). Các thông tin sẽ bao gồm vị trí địa lí và thời gian của vật thể. Tất cả thông tin này được truy cập từ người dung bởi client web app.

Thông tin vị trí địa lí gồm kinh độ và vĩ độ được đo chính xác bởi module GPS, các thông tin được được sẽ được đưa lên sever nhờ module ESP8266 12E.

GPS module được kết nối ESP8266 thông qua giao tiếp UART

Sau khi sever nhận được thông tin từ GPS module thông qua ESP thì sẽ hiện thị vị trí vật thể lên webapp

ESP kết nối với sever thông qua socketio sử dụng Websocket.



Hình 1‑1 Sơ đồ hệ thống

### Các chức năng của hệ thống

Hệ thống có chức năng theo dõi vật thể, thể hiện vị trí vật thể lên bản đồ.

Hệ thống cũng có chức năng “last location”: đưa ra vị trí cuối cùng mà khi esp và sever còn kết nối.

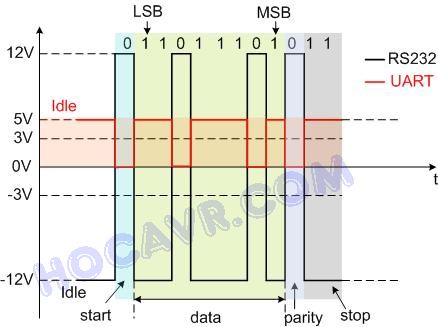
### Các yêu cầu về hệ thông

* Yêu cầu chức năng
* Gửi, nhận vị trí tọa độ​
* Hiển thị object trên giao diện bản đồ Việt Nam​
* Hỗ trợ theo dõi thời gian thực​
* Hỗ trợ thao tác quản lý list objects​
* Lưu lịch sử tọa độ lần cuối trước khi mất tín hiệu​
* Yêu cầu phi chức năng
* Cập nhật nhanh < 5s​
* Giao diện hiển thị trực quan, có thể phóng to thu nhỏ​
* Hỗ trợ iOS và Android​
* Server hỗ trợ tốt cho nhiều thiết bị kết nối cùng một lúc (up to 10 thiết bị, demo tracking trên một xe) ​
* Sai lệch vị trí không quá 50m​

## Chuẩn giao tiếp truyền nhận UART

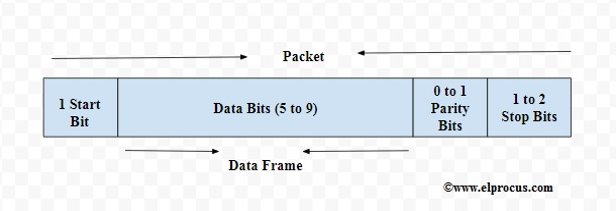
### Tổng quan về UART

Thuật ngữ USART trong tiếng anh là viết tắt của cụm từ Universal Synchronous & Asynchronous serial Reveiver and Transmitter, nghĩa là bộ truyền nhận nối tiếp đồng bộ và không đồng bộ. Cần chú ý rằng khái niệm USART (hay UART nếu chỉ nói đến bộ truyền nhận không đồng bộ) thường để chỉ thiết bị phần cứng (device, hardware), không phải chỉ một chuẩn giao tiếp. USART hay UART cần phải kết hợp với một thiết bị chuyển đổi mức điện áp để tạo ra một chuẩn giao tiếp nào đó. Ví dụ, chuẩn RS232 (hay COM) trên các máy tính cá nhân là sự kết hợp của chip UART và chip chuyển đổi mức điện áp. Tín hiệu từ chip UART thường theo mức TTL: mức logic high là 5, mức low là 0V. Trong khi đó, tín hiệu theo chuẩn RS232 trên máy tính cá nhân thường là -12V cho mức logic high và +12 cho mức low. Chú ý các giải thích này là theo mức logic TTL của USART, không theo RS232.



Hình 1‑2 Tín hiệu tương đương của UART và RS232

### Packet UART

****

Hình 1‑3 Packet UART

* Start bit :

Start bit là bit dùng để đồng bộ dữ liệu. Đây là bit được thêm vào phía trước dữ liệu thực tế. Start bit đánh dấu nơi bắt đầu của gói dữ liệu. Thông trường, trong trạng thái idle, khi không có dữ liệu nào được truyền, mức điện áp trên đường truyền là mức CAO – HIGH (1).Khi bắt đầu truyền dữ liệu, UART truyền sẽ kéo mức điện áp trên bus từ mức CAO xuống mức THẤP (từ 1 xuống 0). UART nhận sẽ phát hiện được sự thay đổi mức điện áp này và sẽ bắt đầu đọc dữ liệu. Thông thường, Start bit chỉ có độ dài 1 bit.​

* ​Stop bit :

Như cái tên của nó, Stop Bit đánh dấu việc kết thúc gói dữ liệu. Nó có độ dài 2 bit nhưng thông thường, người ta chỉ sử dụng 1 bit. Sau khi kết thúc quá trình truyền dữ liệu, mức điện áp trên bus sẽ được giữ ở mức CAO – HIGH (1).

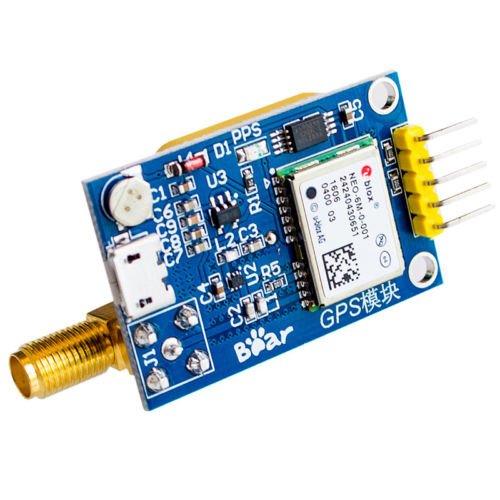
* ​Parity bit:

Parity bit giúp cho thiết bị nhận UART xác định được gói dữ liệu nhận được có chính xác hay không. Parity là kiểu kiểm tra sai sót ở low-level bao gồm 2 biến: Even Parity và Odd Parity. Parity bit là optional và thường ít khi được sử dụng.​

* Data bits:

Là những bits chứa dữ liệu được gửi từ thiết bị truyền sang thiết bị nhận. Độ dài của gói dữ liệu có thể từ 5 đến 9 bits (9 bits nếu như parity bit không được dùng và chỉ có 8 bits khi parity bit được dùng). Thông thường, LSB (bit có giá trị thấp nhất) là bit được truyền đầu tiên.

## Sơ lược về module GPS NEO7M



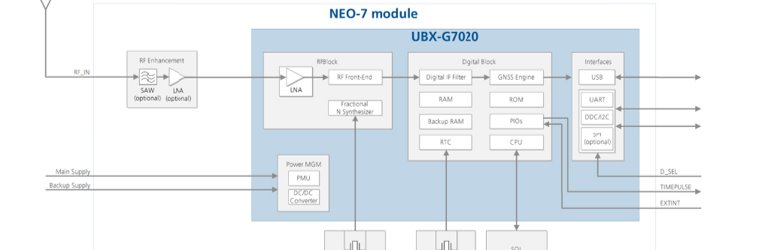
Hình 1‑4 GPS NEO 7m

### Mô tả chung

Module GPS U-blox NEO-7M-0-000 rất cần thiết cho các dự án định vị vị trí và chuyển động, tốc độ cập nhật rất nhanh, trả về tọa độ rất chính xác, kết nối và sử dụng rất đơn giản là những ưu điểm của loại module này.

### Các kiến thức chuyên sâu về GPS Neo7

#### Block diagram

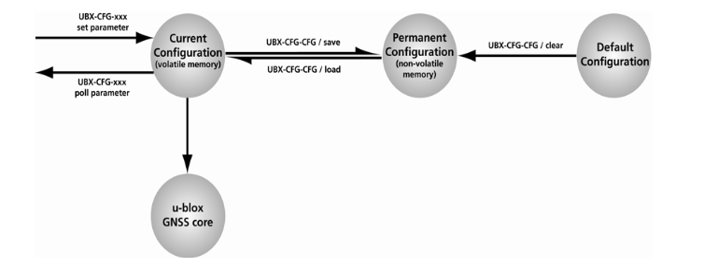


Hình 1‑5 Block diagram neo 7m

Anten được qua 1 bộ lọc, và bộ LNA. Sau đó đưa data vào bộ RF block để đọc tín hiệu anten. Tiếp đó là đưa vào bộ Digital block để convert tín hiệu tương tự thành số

Cuối cùng là bộ interface để gửi tín hiệu đi ra các thiết bị khác

#### Receiver Configuration



Hình 1‑6 Receiver config Neo7m

U-blox có thể cấu hình đầy đủ lại máy thu với giao thức UBX​

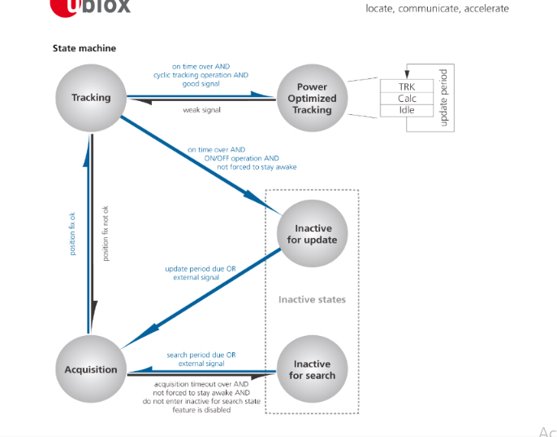
Cấu hình được sử dụng bởi bộ thu GNSS ( được gọi là current configuration)​

Cấu hình có thể được thay đổi bằng  cách gửi tin UBX-CFG-XXX thông qua cổng I/O​

Theo như sơ đồ thì ban đầu sẽ gửi mã UBX-CFG-XXX , mã này sẽ xét các parameter cho máy thu. Các parameter này sẽ trở thành current config . tiếp đó config này sẽ được save lại.

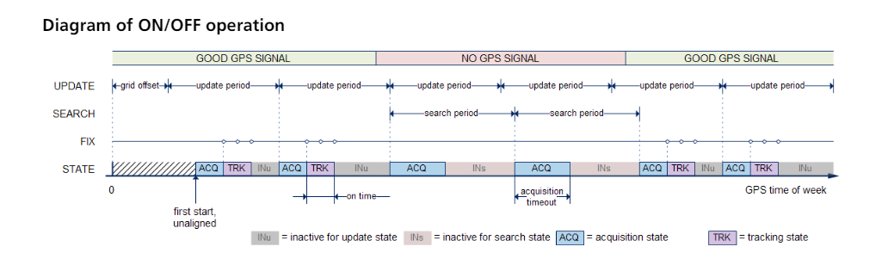
Nếu sử dụng clear thì default config sẽ trở lại config current

#### Operation



Hình 1‑7 Operation neo7m

Neo 7m có 5 giai đoạn lớn là là Acquistion, Tracking, Inactive for update và for search, Power optimize tracking( dành riêng cho lúc tracking: điện năng sẽ được tối ưu hóa)



Hình 1‑8 Diagram of ON/OFF operation

Đây là sơ đồ on/off của operation

Khi bật, đầu tiên sẽ là trạng thái Acquistuon state, nếu có được vị trí hợp lệ với khoảng thời gian fix thì sẽ chuyển trạng thái Tracking. Sau đó sẽ chuyển trạng thái Inactive (không hoạt động) để update Acquistion state. ( vòng lặp sẽ tiếp tục như vậy nếu tín hiệu tốt).

Nếu tín hiệu bị mất thì Acquistion state sẽ được nhận ( giống như không cập nhật state mới mà giữ nguyên state hiện tại) ( trong lúc này sẽ chuyển sang Inactive for search state

### NMEA protocol

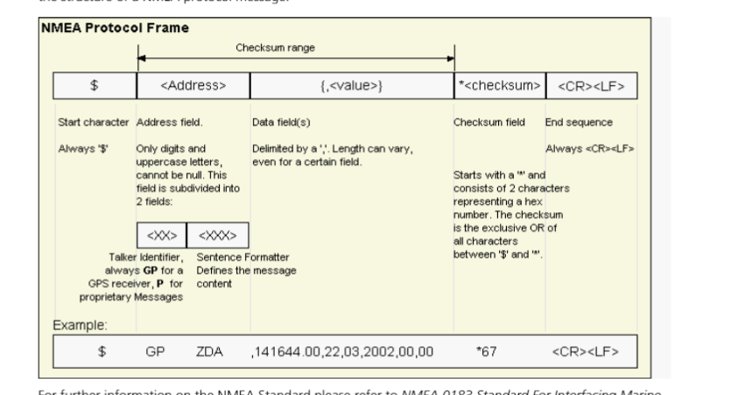
GPS Neo7 sử dụng NMEA protocol.

#### Giới thiệu về NMEA protocol

NMEA 0183 là đặc điểm kỹ thuật điện và dữ liệu kết hợp để giao tiếp giữa các thiết bị điện tử hàng hải như máy đo tiếng vang , máy sonars , máy đo gió , con quay hồi chuyển , máy lái tự động , máy thu GPS và nhiều loại thiết bị khác. Nó đã được xác định bởi, và được kiểm soát bởi Hiệp hội Điện tử Hàng hải Quốc gia . Nó thay thế các tiêu chuẩn NMEA 0180 và NMEA 0182 trước đó . Trong các ứng dụng hàng hải giải trí, nó đang dần được loại bỏ để thay thế cho tiêu chuẩn NMEA 2000 mới hơn [ cần dẫn nguồn] , mặc dù NMEA0183 vẫn là tiêu chuẩn trong vận chuyển thương mại.

Các NMEA 0183 tiêu chuẩn sử dụng một đơn giản ASCII , truyền thông nối tiếp giao thức định nghĩa như thế nào dữ liệu được truyền đi trong một "câu" từ một "talker" cho nhiều "người nghe" tại một thời điểm. Thông qua việc sử dụng các bộ mở rộng trung gian, người nói chuyện có thể trò chuyện một chiều với số lượng người nghe gần như không giới hạn và sử dụng bộ ghép kênh , nhiều bộ cảm biến có thể nói chuyện với một cổng máy tính duy nhất.

#### NMEA frame



Hình 1‑9 NMEA frame

Ví dụ : $ GPGGA, 181908.00.3404.7041778, N, 07044.3966270,​

W, 4,13,1.00.495.144, M, 29.200, M, 0,10,0000 \* 40

Giải thích:

​ Tất cả các data NMEA đều bắt đầu bằng ký tự $ và mỗi trường dữ liệu được phân tách bằng dấu phẩy.​

**GP** thể hiện rằng đó là một vị trí GPS (GL sẽ biểu thị GLONASS).​

**181908.00** là  thời gian: giờ UTC tính bằng giờ, phút và giây.​

**3404.7041778** là vĩ độ ở định dạng DDMM.MMMMM. (decimal minute format) **N** biểu thị vĩ độ bắc.​

**07044.3966270** là kinh độ ở định dạng DDDMM.MMMMM.**W** biểu thị kinh độ tây.​

**4** biểu thị Chỉ số Chất lượng:​

            1 = Tọa độ không hiệu chỉnh​

            2 = Tọa độ đúng sai (ví dụ: WAAS, DGPS)​

            4 = RTK Fix tọa độ (độ chính xác từng cm)​

            5 = RTK Float (độ chính xác decimet.​

**13** biểu thị số lượng vệ tinh được sử dụng trong tọa độ.​

**1,0** biểu thị HDOP (độ pha loãng theo chiều ngang của độ chính xác).​

**495.144** biểu thị độ cao của ăng-ten.​

**M** biểu thị đơn vị độ cao (ví dụ: Mét hoặc Feet)​

**29.200** biểu thị sự phân tách hình học (trừ độ cao này khỏi độ cao của ăng-ten để đến Độ cao trên Ellipsoid (HAE)).​

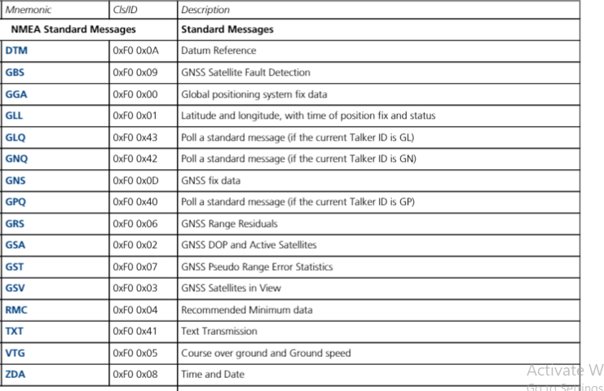
**M** biểu thị các đơn vị được sử dụng bởi sự phân tách geoid.​

**1,0** biểu thị tuổi của việc hiệu chỉnh (nếu có).​

**0000** biểu thị ID trạm sửa (nếu có).​

**\* 40** biểu thị checksum.​

#### Các NMEA Standard



Hình 1‑10 Các chuẩn NMEA

## Ngôn ngữ NodeJS

### Tổng quan Nodejs

* NodeJS là một mã nguồn mở, đa nền tảng, chạy trên môi trường JavaSript, được xây dựng trên V8 JavaScript engine của Chrome - V8 thực thi mã JavaScript bên ngoài trình duyệt. Nó được tạo ra vào năm 2009 đi kèm với một lợi thế chính - NodeJS cho phép thực hiện lập trình bất đồng bộ.
* Ở chế **độ đồng bộ** thực thi từng dòng và tiến hành thực thi dòng tiếp theo khi dòng hiện tại đã thực thi xong.
* Khi **bất đồng bộ** thực thi tất cả dòng code cùng một lúc.
* NodeJS là một nền tảng được xây dựng trên JavaScript runtime của Chrome với mục đích xây dựng các ứng dụng mạng nhanh chóng và có thể mở rộng được một cách dễ dàng hơn. NodeJS sử dụng mô hình I/O lập trình theo sự kiện, non-blocking, do đó nodeJS khá gọn nhẹ và hiệu quả - công cụ hoàn hảo cho các ứng dụng chuyên sâu về dữ liệu theo thời gian thực chạy trên các thiết bị phân tán.
* NodeJS là môi trường runtime mã nguồn mở đa nền tảng, được sử dụng để phát triển các ứng dụng mạng và ứng dụng server-side. Các ứng dụng NodeJS được viết bằng JavaScript và có thể chạy trong NodeJS runtime trên OS X, Microsoft Windows và Linux.
* NodeJS cũng cung cấp một thư viện bao gồm rất nhiều các module JavaScript khác nhau nhằm đơn giản hóa việc phát triển các ứng dụng web, qua đó giảm thiểu tình trạng sử dụng quá nhiều Node.js.

### Các tính năng NodeJS

* **Lập trình hướng sự kiện và không đồng bộ**: Toàn bộ API trong thư viện NodeJS đều không đồng bộ, hay không bị chặn. Về cơ bản điều này có nghĩa là một server sử dụng NodeJS sẽ không bao giờ chờ một API trả về dữ liệu. Server sẽ chuyển sang API kế tiếp sau khi gọi API đó và cơ chế thông báo của Events trong NodeJS giúp server nhận được phản hồi từ lần gọi API trước.
* **Cực kỳ nhanh chóng**: Được xây dựng trên Công cụ JavaScript V8 của Google Chrome, thư viện NodeJS có khả năng xử lý mã vô cùng nhanh.
* **Đơn luồng/Single thread nhưng có khả năng mở rộng cao**: NodeJS sử dụng một mô hình luồng đơn với vòng lặp sự kiện/event. Cơ chế event cho phép máy chủ phản hồi non-blocking và cũng cho phép khả năng mở rộng cao hơn so với các server truyền thống hỗ trợ giới hạn các thread để xử lý yêu cầu. NodeJS sử dụng một chương trình đơn luồng, cùng một chương trình có thể cung cấp dịch vụ cho một số lượng yêu cầu lớn hơn so với các máy chủ truyền thống như Apache HTTP Server.
* **Không có buffer** - Các ứng dụng NodeJS không có vùng nhớ tạm thời (buffer) cho bất kỳ dữ liệu nào. Các ứng dụng này chỉ đơn giản xuất dữ liệu theo khối.
* **License** - NodeJS được phát hành theo giấy phép MIT.

### Ứng dụng Nodejs

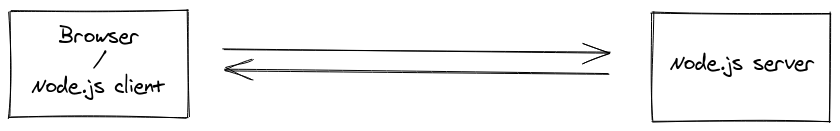
* **Hệ thống Notification** - Giống như facebook hayTwitter
* **Websocket server** - Các máy chủ web socket như là Online Chat, Game Server…
* **Fast File Upload Client** - Các chương trình upload file tốc độ cao.
* **Ad Server** - Các máy chủ quảng cáo.
* **Cloud Services** - Các dịch vụ đám mây.
* **RESTful API** - Những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.
* **Any Real-time Data Application** - Bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực.
* **Ứng dụng Single Page Application (SPA)** - Những ứng dụng này thường request rất nhiều đến server thông qua AJAX
* **Ứng dụng truy vấn tới NoSQL database** - Như MongoDB, CouchDB,…
* **Ứng dụng CLI** - Các công cụ sử dụng command-line.

## Tổng quan về Socket.IO và Websocket

### Socket.IO

Socket.IO là một thư viện cho phép giao tiếp theo thời gian thực, hai chiều và dựa trên sự kiện giữa trình duyệt và máy chủ. Nó bao gồm:

* một máy chủ Node.js
* một thư viện máy khách Javascript cho trình duyệt (cũng có thể chạy từ Node.js)

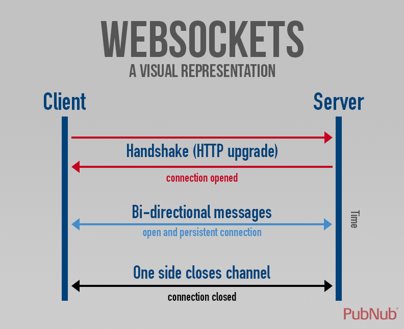


Hình 1‑11 Mô hình Socket.IO

Client sẽ cố gắng thiết lập kết nối Websocket với sever

### WebSocket

Trong giao thức WebSocket, data được truyền theo chuỗi các frame.Để tránh gây nhầm lẫn cho các node trung gian trong mạng internet (máy chủ proxy) và mục đích bảo mật, client phải mask tất cả các frame được gửi tới server. Server sẽ đóng kết nối nếu như nhận được frame không được mask, bằng cách gửi Close frame với status code 1002 (protocol error).Ngược lại, server lại không được mask frame khi gửi xuống cho client. Client sẽ đóng kết nối nếu như nhận được frame được mask, bằng cách gửi Close frame với status code 1002 (protocol error).



Hình 1‑12 websocket protocol

Về bản chất, WebSocket khác với HTTP, mặc dù cả giao thức đều ở trên layer 7 của mô hình OSI, và cùng phụ thuộc vào TCP ở layer 4. ​

Tuy nhiên, WebSocket “được thiết kế để hoạt động trên các cổng HTTP 443 và 80, cũng như để hỗ trợ proxy HTTP và làm trung gian”. ​

Do đó, WebSocket hoàn toàn có khả năng tương thích với giao thức HTTP. ​

Để có được sự tương thích này, WebSocket handshake sẽ sử dụng một header HTTP Upgrade để thay đổi giao thức HTTP thành WebSocket.​

Giao thức handshake sẽ bắt đầu bằng một HTTP request/response, cho phép server xử lý các kết nối HTTP cũng như kết nối WebSocket ở trên cùng một cổng.​

#### Base framing protocol

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-------+-+-------------+-------------------------------+

|F|R|R|R| opcode|M| Payload len | Extended payload length |

|I|S|S|S| (4) |A| (7) | (16/64) |

|N|V|V|V| |S| | (if payload len==126/127) |

| |1|2|3| |K| | |

+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +

| Extended payload length continued, if payload len == 127 |

+ - - - - - - - - - - - - - - - +-------------------------------+

| |Masking-key, if MASK set to 1 |

+-------------------------------+-------------------------------+

| Masking-key (continued) | Payload Data |

+-------------------------------- - - - - - - - - - - - - - - - +

: Payload Data continued ... :

+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +

| Payload Data continued ... |

+---------------------------------------------------------------+

FIN: 1bit: Chỉ ra rằng đây là đoạn cuối cùng trong một tin nhắn​

RSV1, RSV2, RSV3:  1 bit x 3 ​

            Phải bằng 0 trừ khi có phần mở rộng định nghĩa cho giá trị khác 0 ​

Opcode: 4 bit             ​

            Định nghĩa cách diễn giải của phần payload ​

Opcode payload:​

- %x0 biểu thị một continuation frame (sử dụng cho việc gửi message lớn hơn frame size và bị chia thành nhiều frame)​

-  %x1 biểu thị text frame​

- %x2 biểu thị binary frame​

- %x3-7 dự trữ dành cho các non-control frame​

-  %x8 biểu  thị kết thúc của việc kết nối​

- %x9 biểu thị một ping frame​

- %xA biểu thị một pong frame​

- %xB-F dự trữ dành cho các control frame​

Mask: 1 bit​

            Mask = 1 có nghĩa là frame được mask, phải sử dụng masking-key để giải mã payload​

Payload length:  7 bit, 7+16 bit, hoặc 7+64 bit​

            Độ dài của Payload data ,tính bằng byte: 7 bit cho 0-125, 7+16 bit cho 126, 7+64 bit cho 127​

Masking-key: 0 hoặc 4 byte​

Khóa giải mã cho frame được mask gửi từ client​

Payload data: (x+y) byte​

Extension data: x byte (0 byte nếu không có extension. Extension phải được xác định rõ độ dài và mục đích sử dụng trong Handshake request)​

Aplication data: y byte (chiếm phần còn lại của frame sau extension data)​

#### Các frame websocket hay dung

Control frame: Control frame được dùng để thông báo trạng thái của WebSocket.

Close frame: Close Frame được dùng để thông báo đóng kết nối.

Ping frame, pong frame:

Khi nhận được Ping frame, điểm cuối phải gửi một Pong frame trả về, trừ khi đã nhận được Close frame. Một điểm cuối có thể gửi một Ping frame bất kì lúc nào sau khi kết nối được thiết lập và trước khi đóng kết nối.

* Ping frame đóng vai trò xác minh điểm cuối từ xa vẫn đang đáp ứng kết nối
* Pong frame được gửi lại phải có payload data giống với ping frame gửi đi

# CHƯƠNG 2 THIẾT KẾ HỆ THỐNG

Qua quá trình tìm hiểu cơ sở lý thuyết, nghiên cứu và phân tích yêu cầu đề tài, đưa ra thiết kế các khối chức năng cho hệ thống.

## 2.1 Sơ đồ khối và chức năng các khối hệ thống

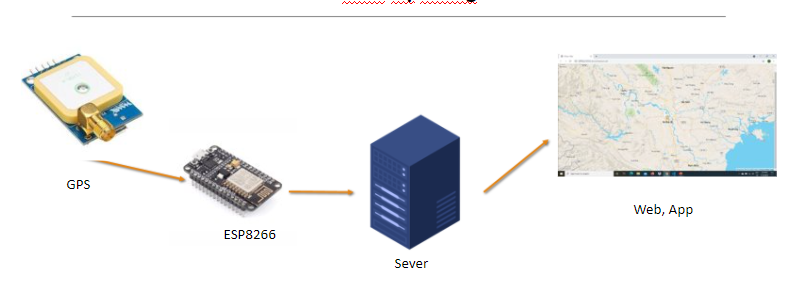
****

Hình 2‑1 : sơ đồ khối hệ thống

Hệ thống gồm 3 phần chính

* Khối Things : đọc dữ liệu data từ module GPS NEO 7m
* Khối kết nối sever : ESP kết nối với sever và đẩy dữ liệu nhận được từ GPS module lên sever. Sever tiếp nhận dữ liệu từ ESP
* Khối ứng dụng : xây dựng webapp để hiện thị vị trí vật thể từ dữ liệu được đẩy lên từ ESP.

## 2.2 Mô hình chi tiết hệ thống và tổng quan phần cứng hệ thống



### 2.2.1 GPS Module NEO7M

Thông số kĩ thuật GPS NEO7M

Hãng Ublox.

IC chính : NEO 7 nguyên chiếc của Ublox.

Nguồn cấp : 3V ~ 3.5V DC

Giao tiếp : Serial (UART TTL)

Chu kỳ cập nhật tọa độ 0,1s đến 1s tùy từng địa hình có thuận lợi hay không.

Phần mềm hỗ trợ với rất nhiều tính năng được Ublox cho tải miễn phí.

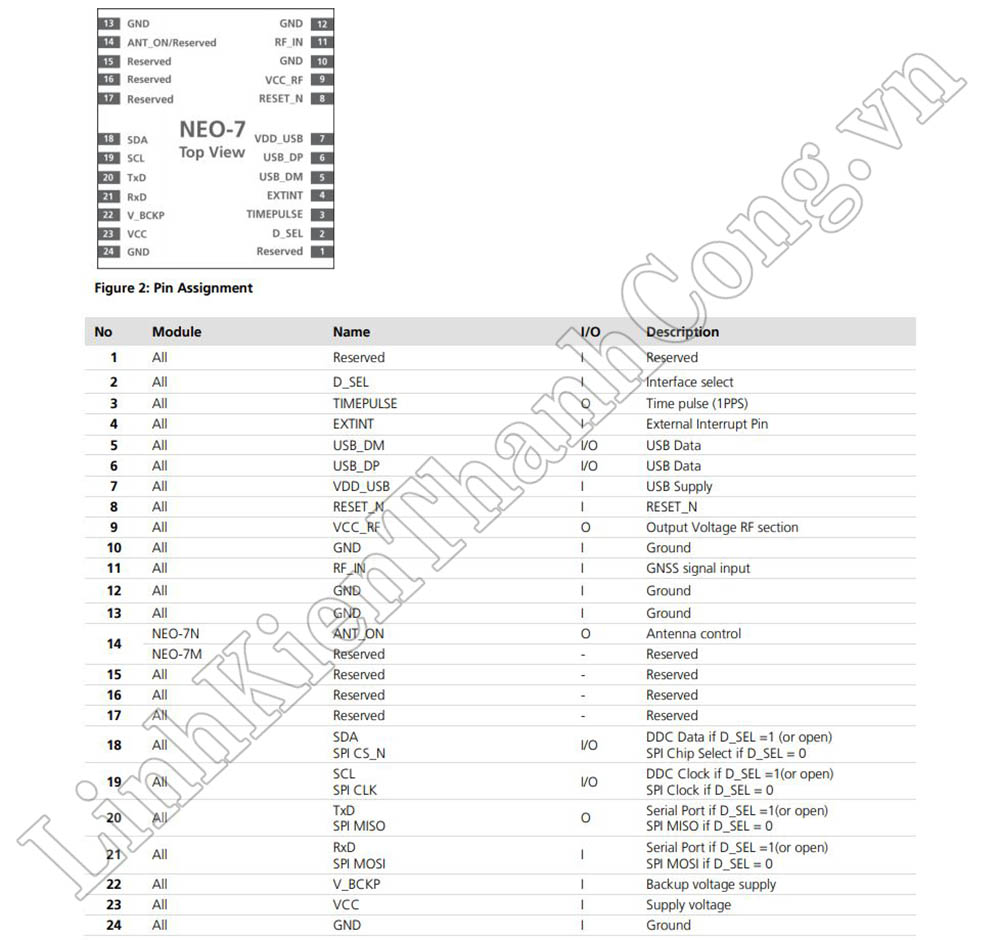
Xác định tọa độ (kinh tuyến, vĩ tuyến) hiện tại của module trên bề mặt trái đất với sai số nhỏ nhất < 1m.

Xác định thời gian quốc tế được cấp bởi đồng hồ nguyên tử trên vệ tinh gửi về. Từ đó bạn cũng có thể suy ra thời gian đồng hồ nơi ở của bạn theo tắc trừ múi giờ. Khỏi cần module RTC.

Chỉ cần 3 vệ tinh là bạn có thể xác định được tọa độ, chỉ cần 4 vệ tinh là bạn có thể xác định được độ cao hiện tại so với mực nước biển.

Có thể tính toán ra tốc độ di chuyển, hướng di chuyển của vật thể được gắn module GPS.

Giải các bài toán về tính toán giữa 2 điểm bất kì, tính diện tích ở một không gian cực kì rộng lớn.



Hình 2‑2 output pin NEO7M

### 2.2.2 Module ESP8266 12E

#### 2.2.2.1 Giới thiệu về ESP8266 12E

[Thu phát wifi ESP8266 12E](https://nshopvn.com/product/thu-phat-wifi-esp8266-12e/) là module wifi giá rẻ và được đánh giá rất cao cho các ứng dụng liên quan đến Internet và Wifi cũng như các ứng dụng truyền nhận sử dụng thay thế cho các module RF khác.

ESP8266 là một chip tích hợp cao, được thiết kế cho nhu cầu của một thế giới kết nối mới, thế giới Internet of thing (IOT). Nó cung cấp một giải pháp kết nối mạng Wi-Fi đầy đủ và khép kín, cho phép nó có thể lưu trữ các ứng dụng hoặc để giảm tải tất cả các chức năng kết nối mạng Wi-Fi từ một bộ xử lý ứng dụng.

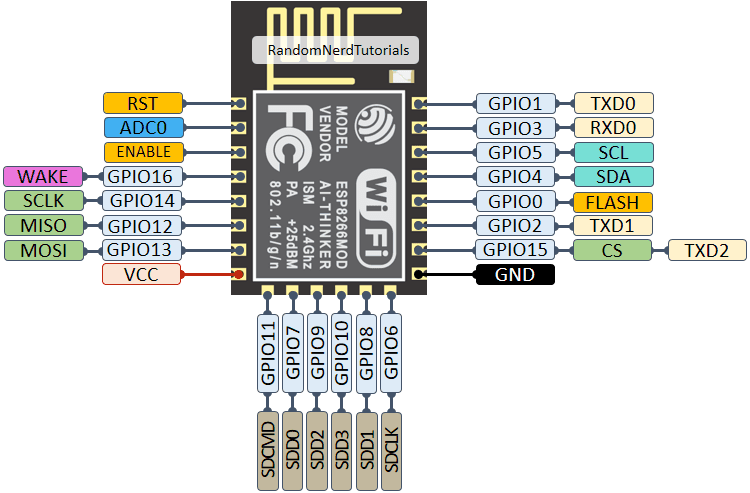
**Mạch thu phát wifi ESP8266 12E** có khả năng xử lý và lưu trữ mạnh mẽ cho phép nó được tích hợp với các bộ cảm biến, vi điều khiển và các thiết bị ứng dụng cụ thể khác thông qua GPIOs với một chi phí tối thiểu và một  PCB tối thiểu.

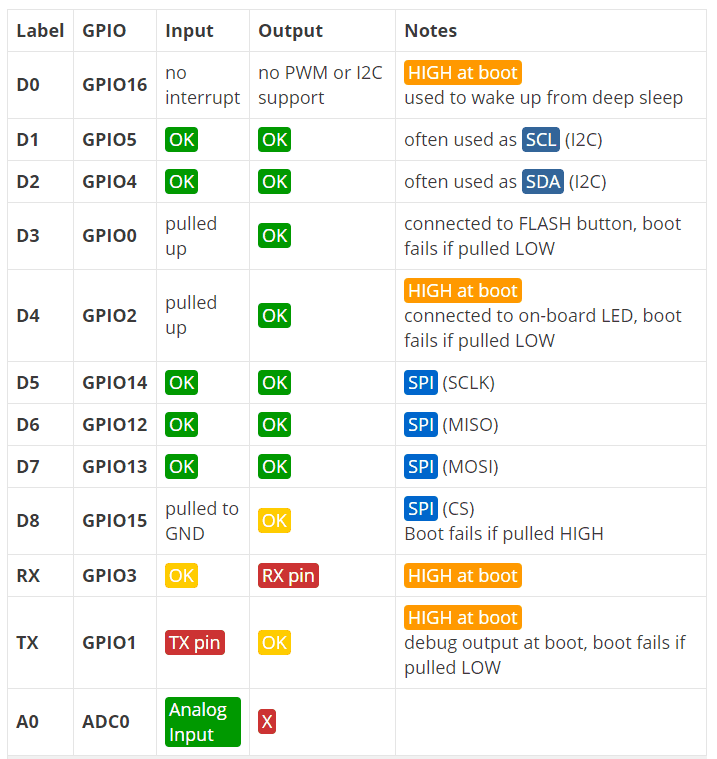
ESP8266 12E có kích thước nhỏ gọn, ra chân đầy đủ của IC **ESP8266**, mạch được thiết kế và gia công chất lượng tốt với vỏ bọc kim loại chống nhiễu và anten Wifi PCB tích hợp cho khoảng các truyền xa và ổn định.

#### 2.2.2.2 Thông số kĩ thuật ESP8266 12E

* IC chính: Wifi SoC ESP8266
* Điện áp sử dụng: 3.0V~3.6V(Optimal 3.3V)
* Working current: ≈70mA(170mA MAX), standby＜200uA
* Dòng tiêu thụ 10 uA, dòng điện năng chờ < 5 uA
* 30 chân (10 GPIO, every GPIO can be PWM, I2C, 1-wire)
* MCU Frequency: 80-160 MHz, 32-bit micro MCU
* SRAM size: 36 KB
* ROM size: 4 MB (SPI External Flash)
* Antena on PCB
* Transmission data rate: 110-460800bps
* 10 bit precision ADC pinout on board (0~1V)
* WiFi @ 2.4 GHz, supports WPA / WPA2 security mode
* Wi-Fi Connectivity (802.11 b/ g/ n)
* Support UART/GPIO data communication interface
* Support STA/AP/STA+AP 3 working modes
* Built-in TCP/IP protocol stack, maximum 5 clients
* Nhiệt độ làm việc: -40℃～＋125℃
* Kích thước : 24 x 16 x 3mm
* Trọng lượng: 4g

#### 2.2.2.3 Sơ đồ chân ESP8266



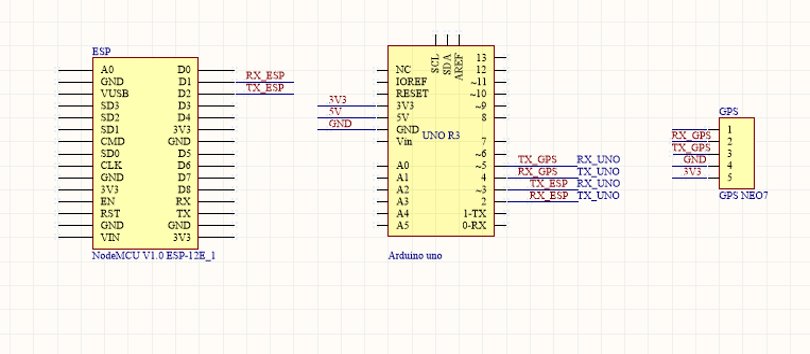


Hình 2‑3 sơ đồ chân ESP8266 và giải thích

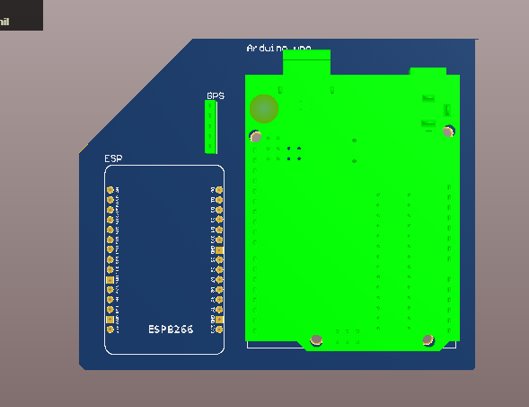
# CHƯƠNG 3: TRIỂN KHAI HỆ THỐNG

## 3.1 Thiết kế phần cứng

Sơ đồ nguyên lí và mạch in:

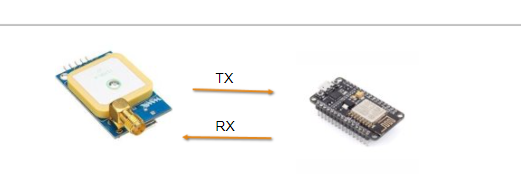


Hình 3‑1 Sơ đồ nguyên lí



Hình 3‑2 Sơ đồ mạch in

## 3.2 Đọc dữ liệu từ module GPS NEO7M



Chân TX của gps được nối với D1 của esp ( RX ảo của esp)

Chân RX của gps được nối với D2 của esp (TX ảo của esp)

Chân VCC của gps nối với chân 3.3V

GND nối GND

DataGps được đọc từ module gps ban đầu ( đọc thông qua 2 chân SoftwareSerial D1,D2) là Raw data NMEA. Chúng ta sử dụng thư viện Tiny Gps để chuyển về data số thực mong muốn

Kết quả quá trình này là lấy được 2 giá trị Long: kinh độ và Lat: vĩ độ

## 3.3 Đẩy dữ liệu đọc được thông qua ESP8266 lên server

### 3.3.1 Cài đặt esp8266 thành một client socket

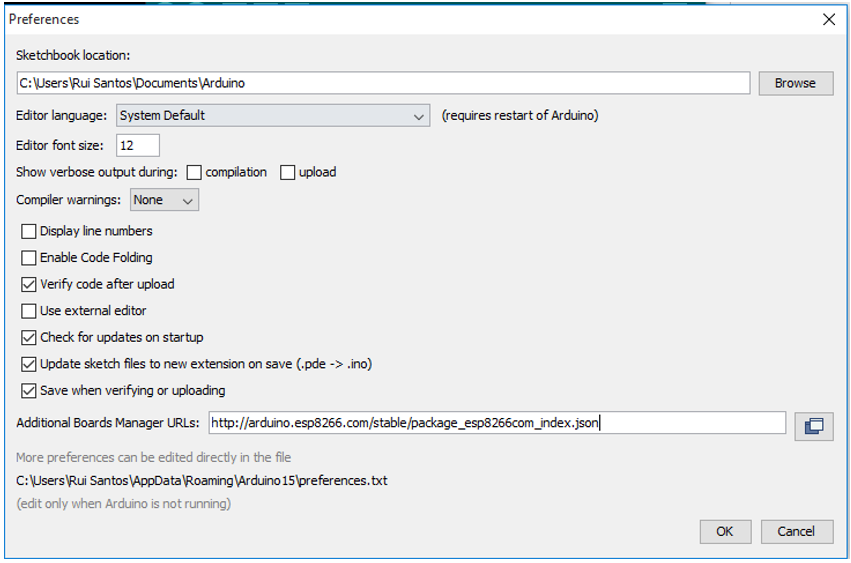
Để cài đặt được cho esp chúng ta sử dụng Arduino IDE

Khi bật Arduino :

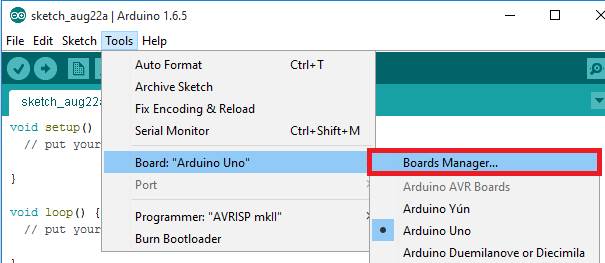
Vào **File→ Preferences**, vào textbox **Additional Board Manager URLs** thêm đường link sau vào

http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

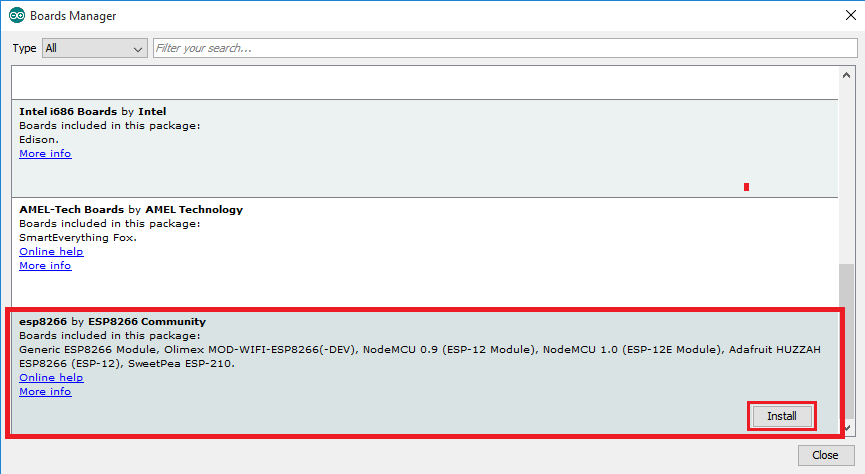
Click **OK** để chấp nhận.



Tiếp theo vào **Tool**→**Board**→**Boards Manager**

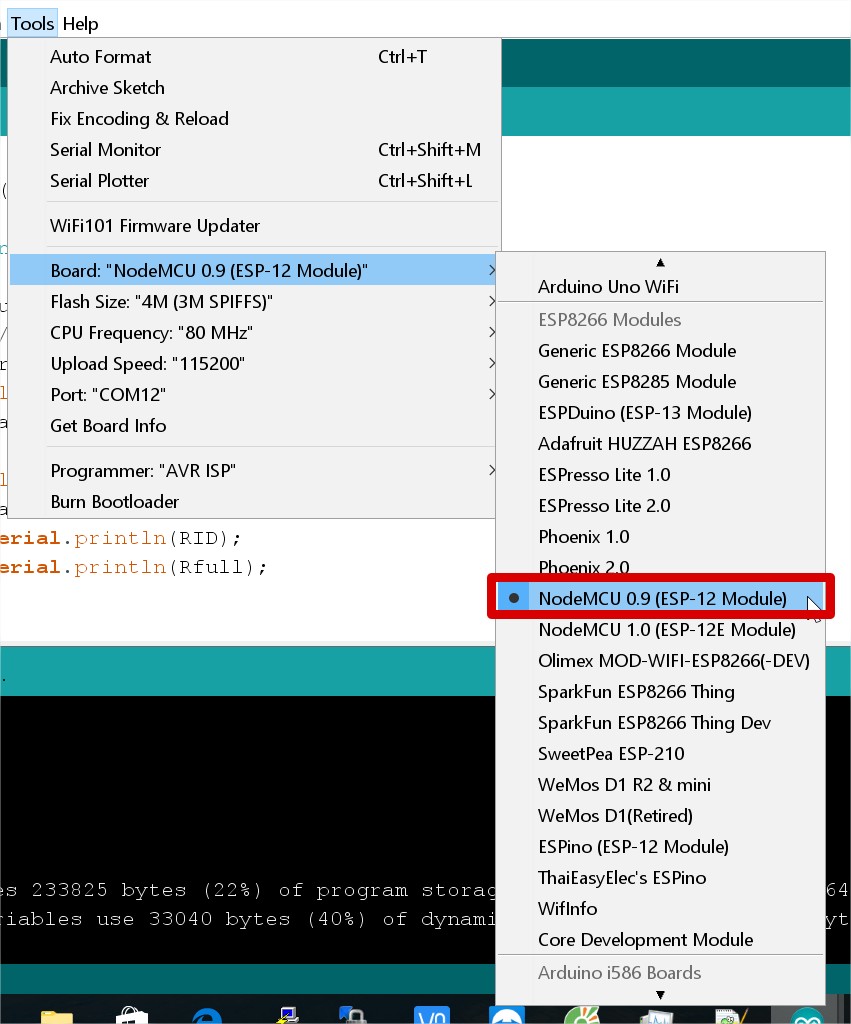


đợi một lát để chương trình tìm kiếm. Ta kéo xuống và click vào **ESP8266 by ESP8266 Community**, click vào **Install**. Chờ phần mềm tự động download và cài đặt.



Chọn Board để lập trình cho ESP8266:

Vào **Tool**→**Board**→**Node MCU 0.9** chọn cổng COM tương ứng với module NodeMCU tương ứng



### 3.3.2 Esp8266 kết nối sever và đẩy lên sự kiện

Sau khi cài đặt esp8266 thành công, ta sẽ kết nối vào mạng wifi với ssid và password của wifi cần kết nối.

Sau đó là viết hàm để có thể đẩy data lên sever

socket có 2 hàm chính đó là "on" và "emit" . Trong đó, hàm on là hàm bắt sự kiện và hàm emit là hàm các sự kiện. Tham số của các hàm lần lượt là:

.on(<tên sự kiện>, function(<danh sách tham số>) {

​<nội dung hàm xử lý

})

emit(<tên sự kiện>, <danh sách tham số>)

Như vậy chúng ta phải tạo hàm .on trên sever để bắt được sự kiện từ Esp và tạo hàm .emit ở Esp để có thể gửi sự kiện trên sever

Nếu sever bắt được đúng tên sự kiện, hàm xử lý sẽ được thực thi

Kết nối Esp tới sever:

Ở đây nhóm sử dụng localhost:4000

Sử dụng hàm SocketIoClient::begin(host[, port, path]) để kết nối tới sever

Hàm emit của esp8266: để gửi sự kiện và data lên sever

SocketIoClient::emit( “ tên sự kiện”, string data)

Tương ứng chúng ta sẽ có hàm .on ở bên phía sever

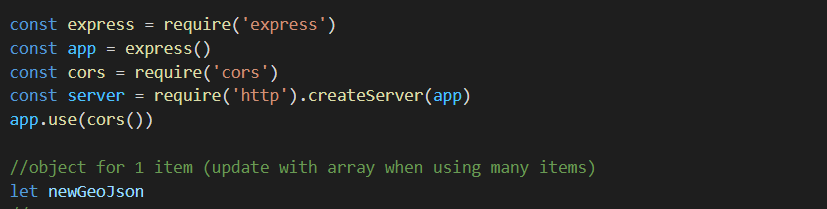
## 3.4 Triển khai Web app hiện thị vị trí trên map

### 3.4.1 Server

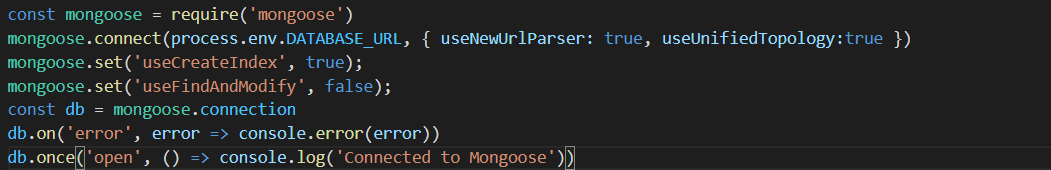
Khởi tạo server sử dụng thư viện express và http.

Sử dụng thư viện cor và sử dụng như một middleware để thực hiện Cross-Origin Resourses Sharing

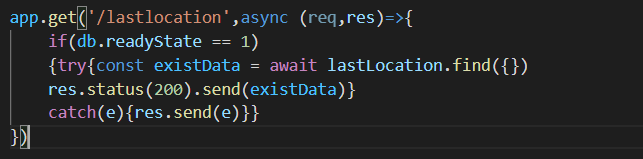
Biến newGeoJson lưu trữ dữ liệu tọa độ hiện tại client gửi lên server.



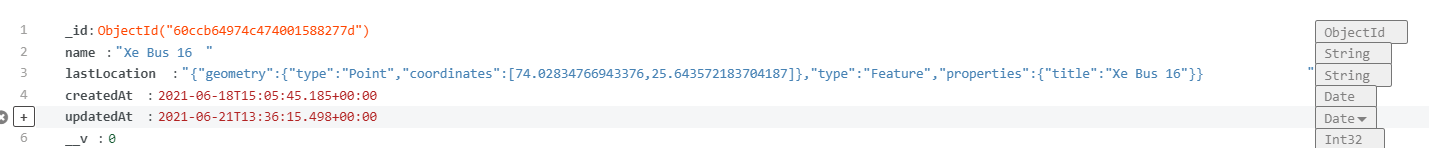
Khởi tạo kết nốt với MongoDB bằng thư viện Mongoose



Định nghĩa API dạng GET Request để lấy dữ liệu tọa độ lưu trong database

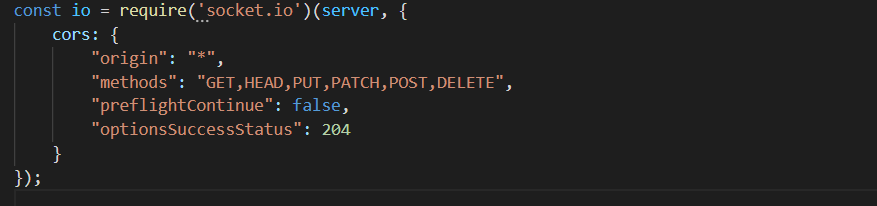


Dữ liệu trong database có dạng như sau:



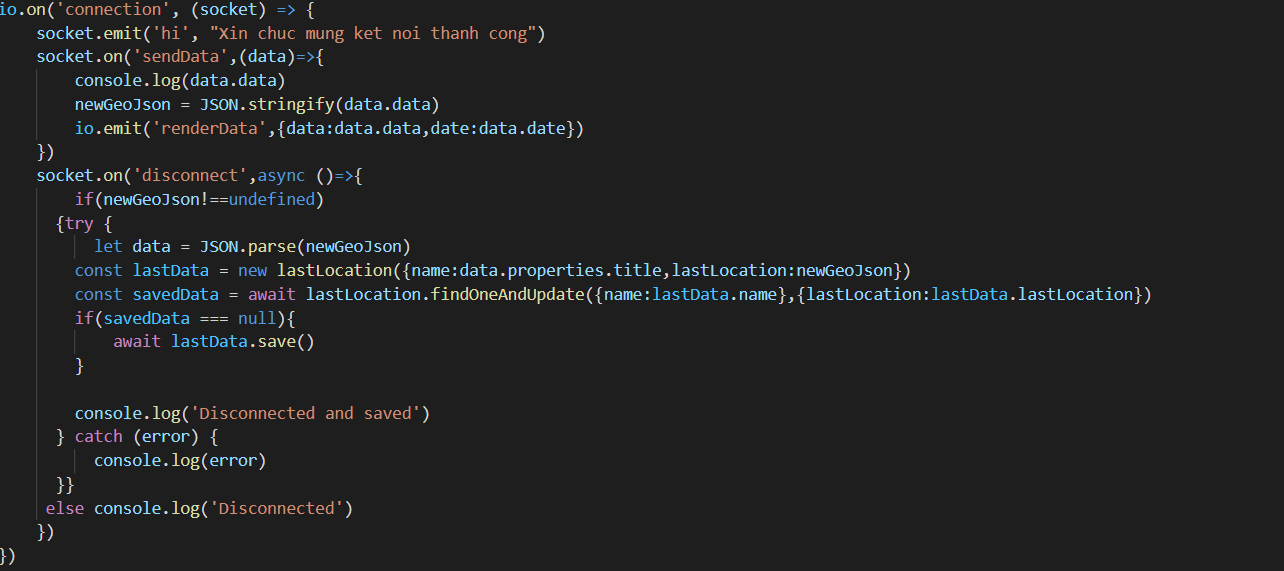
Định dạng dữ liệu trong database được lưu trong file lastlocation.js

Khai báo thư viện Socket.io và định nghĩa kết nối socket, khai báo Cross-Origin Resourses Sharing



Khai báo các trigger:

* Khi có sự kiện đặc biệt ‘connection’ được thông báo (emit) từ client thì server sẽ thực hiện callback
* Bên trong hàm callback định nghĩa các trigger được sử dụng khi kết nối vẫn còn duy trì
* Socket.emit(‘hi’, String) sẽ gửi đi String tới client khi client có hàm lắng nghe sự kiện ‘hi’ từ server
* Socket.on(‘sendData’,callback) sẽ nhận thông báo ‘sendData’ từ client kèm với data được gửi lên và xử lý data với callback. Bên trong hàm callback, server sẽ thông báo sư kiện ‘renderData’ và gửi data tới client chịu trách nhiệm hiển thị tọa độ lên bản đồ
* Khi có sự kiện đặc biệt ‘disconnect’ được thông báo thì data được lưu trong biến newGeoJson sẽ được lưu vào trong database theo định dạng được định nghĩa từ trước



Server hosting tại local sẽ host ở port 4000

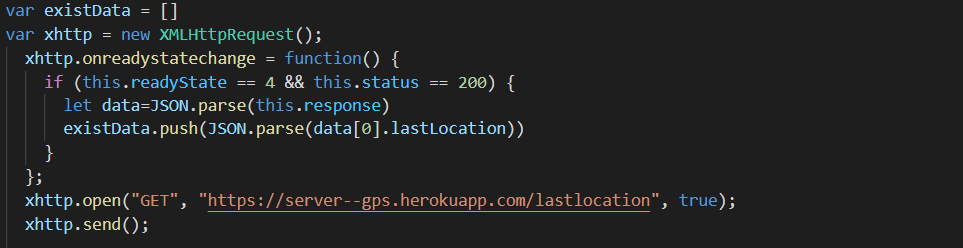


### 3.4.2.Client

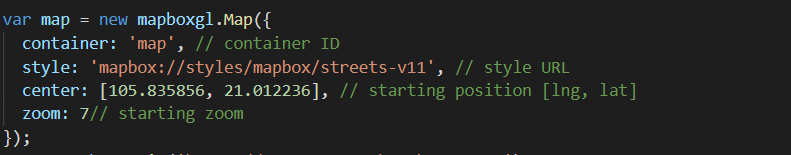
File index.html có nhiệm vụ hiển thị nội dung Website dạng bản đồ

File map.js có nhiệm vụ định nghĩa các hiệu ứng sẽ sử dụng của Website

Mảng existData sẽ chứa dữ liệu được lấy từ database thông qua một GET Request được định nghĩa như bên dưới



Khởi tạo đối tượng Map(bản đồ) có tọa độ trung tâm và mức zoom như bên dưới



Định nghĩa socket cho client, thư viện socket.io được lấy từ server như bên dưới



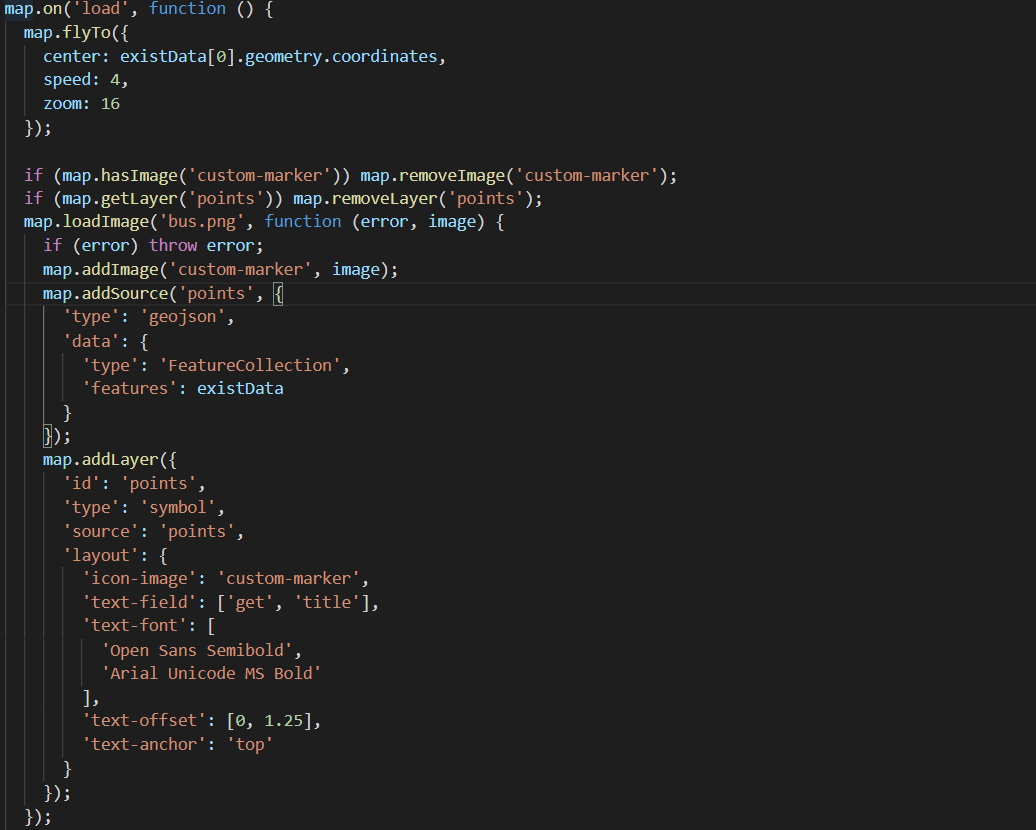
Tiến hành tải bản đồ lên, khi sự kiện ‘load’ xảy ra thì hàm callback bên trong được thực hiện.

Di chuyển bản đồ để cập nhật lại tọa độ trung tâm với data trong mảng existData với hiệu ứng ‘flyTo’, zoom = 16, tốc độ cập nhật bằng 4

Kiểm tra trên bản đồ đã có marker nào có id là ‘point’ và ảnh có id là ‘custom-marker’ chưa, nếu có thì xóa đi.

Tải ảnh bus.png lên và sử dụng như một marker cho tọa độ trung tâm trên bản đồ

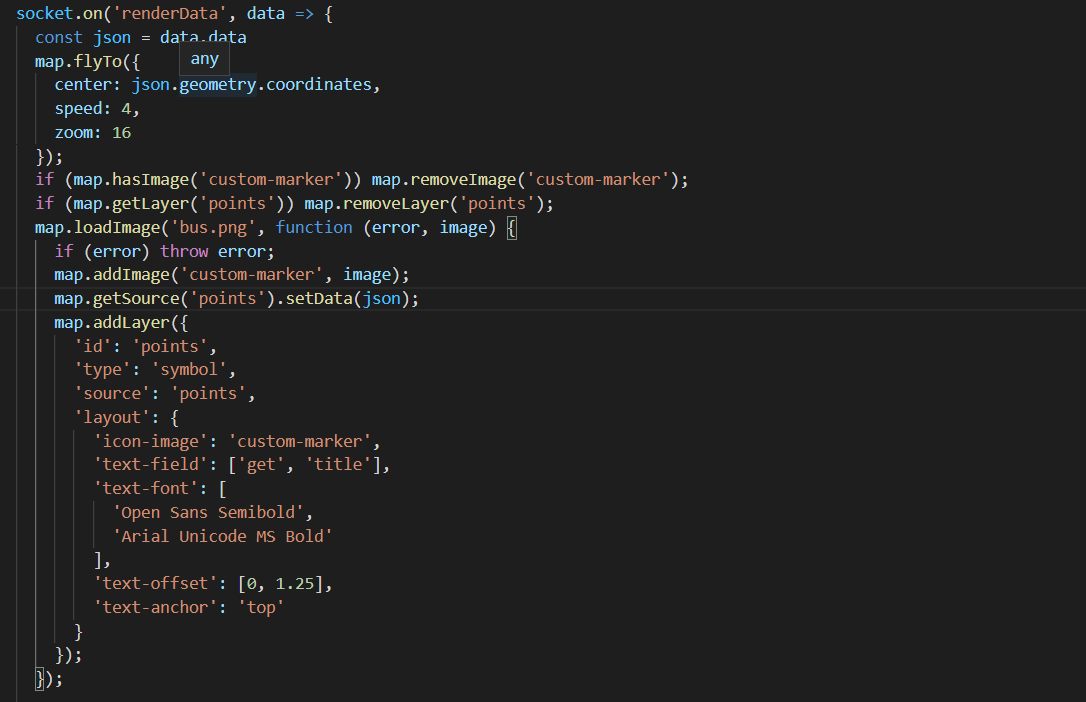
Định nghĩa cho nội dung mà custom-marker hiển thị trên bản đồ với các phương thức addImage, addSource, addLayer



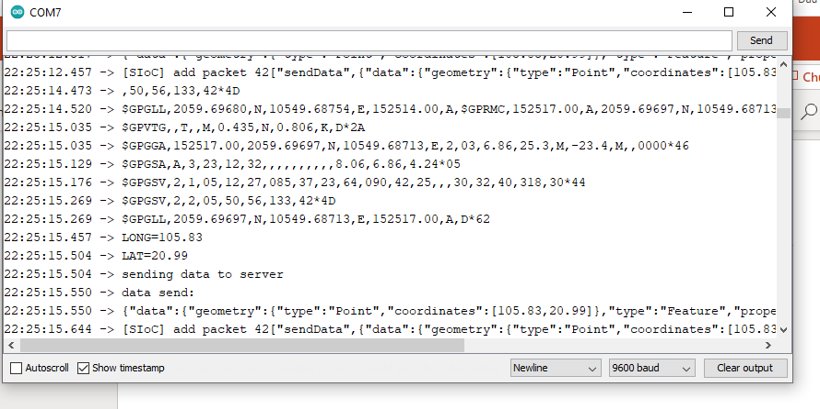
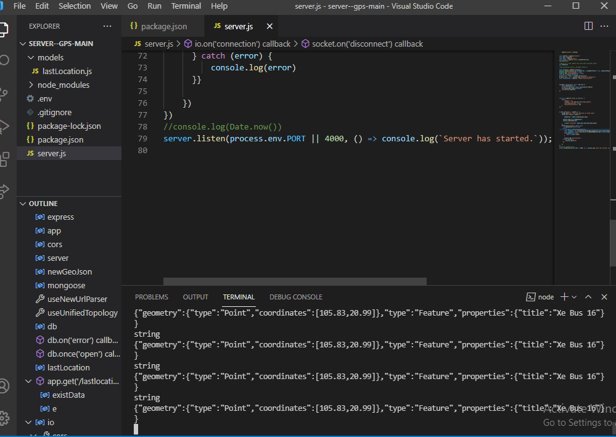
Đoạn code bên dưới thực hiện công việc như bên trên nhưng được đặt trong callback của hàm socket.on(‘renderData’, callback)

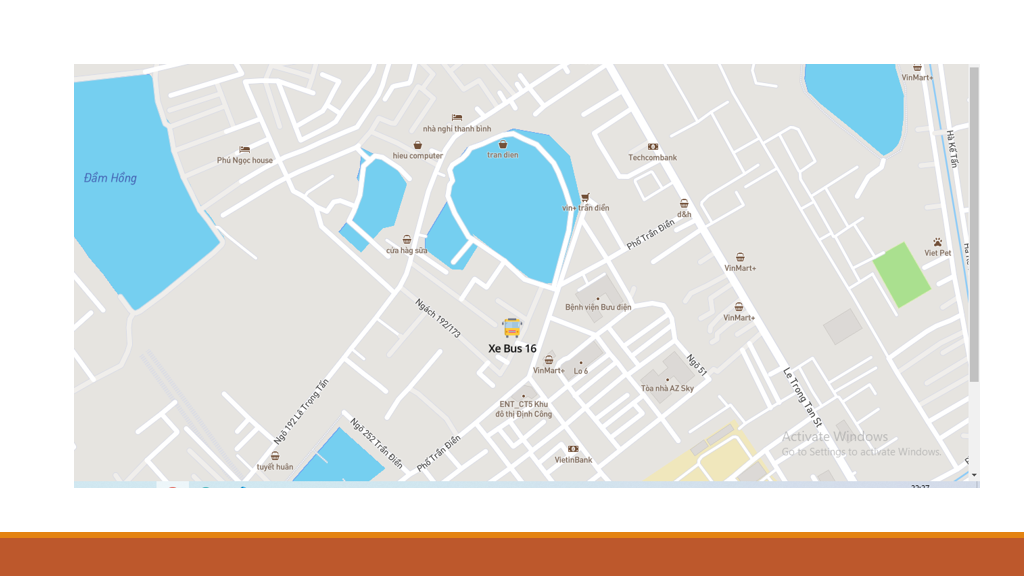
Data được server gửi đến client thông qua việc thông báo sự kiện ‘renderData’ sẽ được xử lý trong hàm callback

Hàm callback khi được kích hoạt sẽ thay đổi tọa độ trung tâm của bản đồ và định nghĩa lại marker hiển thị trên bản đồ nếu có gì được thay đổi



**TEST HỆ THỐNG**



Hệ thống chạy ổn định

# KẾT LUẬN

Trong quá trình thực hiện đề tài đồ án này, chúng em đã xây dựng được một hệ thống có thể thực hiện được các yêu cầu chức năng đặt ra. Hệ thống sau khi được triển khai về cơ bản đã có thể đáp ứng được nhu cầu của người dùng là hiện thị được vị trí lên trên map

Tuy nhiên do hạn chế về thời gian, các bước phân tích, thiết kế vẫn còn thiếu sót nên hệ thống định vị chỉ sử dụng một hệ thống định vị là GPS để xác định vị trí của xe bus, cho nên nếu tín hiệu GPS gặp thời tiết xấu như mưa, sương mù, mây dày cũng sẽ có thể tạo ra độ lệch hoặc trôi vị trí nhất định. Hệ thống cũng đang chỉ được triển khai với 1 vật thể.

Trong tương lai hệ thống có thể phát triển thêm phần giám sát nhiều vật thể hơn, menu lựa chọn tìm kiếm nhiều vật thể, thống kê tọa độ của vật thể,…

Nhóm em xin chân thành cảm ơn thầy Ngô Vũ Đức thời gian qua đã đồng hành, giúp đỡ, góp ý cho nhóm để hoàn thành đề tài. Nhóm em xin chân thành cảm ơn.

# TÀI LIỆU THAM KHẢO

[1] <https://www.u-blox.com/sites/default/files/products/documents/NEO-7_DataSheet_%28UBX-13003830%29.pdf>

[2] <https://www.u-blox.com/sites/default/files/products/documents/NEO-7_ProductSummary_%28UBX-13003342%29.pdf>

[3] <https://github.com/mikalhart/TinyGPS>

[4] <https://github.com/timum-viw/socket.io-client>

[5] <https://topdev.vn/blog/socket-la-gi-websocket-la-gi/>

[6] <http://arduino.vn/tutorial/1482-gioi-thieu-cach-su-dung-module-gps-neo-6-va-neo-7-cua-hang-ublox>