INSTITUT JULIO ANTONIO MÓRA D'EBRE	**CFGM SMIX**	Project 15: Synthesis Project Report

Real-time GPS Tracking with Wappsto:bit

Akram Amkeched Ouriaghli - Enric Navarro. CFGM Sistemes Microinformàtics i Xarxes

Tutor: David Caminero

Curs 23-24

Abstract

This project focuses on the practical implementation of a localization system for a theoretical 5G-supported autonomous vehicle. Utilizing a BBC micro:bit paired with a Wappsto:bit extension, we developed a web-based application that opens a WebSocket to the Wappsto API. This setup retrieves geolocation data from the micro:bit and displays it on a map in real-time. The core of this project involved programming the micro:bit to send location data to the Wappsto:bit, which then communicated with our web application. We designed the web interface to provide a user-friendly visualization of the geolocation data, ensuring that the information was accurate and up-to-date. Throughout the development process, we explored the significant impact of 5G technology, particularly its role in enhancing data transmission speed and reliability, which is crucial for autonomous systems. The project highlights the integration of IoT devices with web technologies, showcasing how the micro:bit's flexibility and Wappsto:bit's enhanced connectivity can be leveraged for advanced tracking and data visualization purposes.

Resum

Aquest projecte se centra en la implementació pràctica d'un sistema de localització per a un vehicle autònom teòricament suportat per 5G. Utilitzant un BBC micro:bit combinat amb una extensió Wappsto:bit, hem desenvolupat una aplicació web que obre un WebSocket a l'API de Wappsto. Aquest sistema recupera dades de geolocalització del micro:bit i les mostra en un mapa en temps real. El nucli d'aquest projecte va consistir a programar el micro:bit per enviar dades de localització al Wappsto:bit, que després es comunicava amb la nostra aplicació web. Vam dissenyar la interfície web per oferir una visualització amigable de les dades de geolocalització, assegurant-nos que la informació fos precisa i actualitzada. Durant el procés de desenvolupament, vam explorar l'impacte significatiu de la tecnologia 5G, especialment el seu paper en la millora de la velocitat i la fiabilitat de la transmissió de dades, la qual cosa és crucial per als sistemes autònoms. El projecte destaca la integració de dispositius IoT amb tecnologies web, mostrant com la flexibilitat del micro:bit i la connectivitat millorada del Wappsto:bit poden ser utilitzades per al seguiment avançat i la visualització de dades.

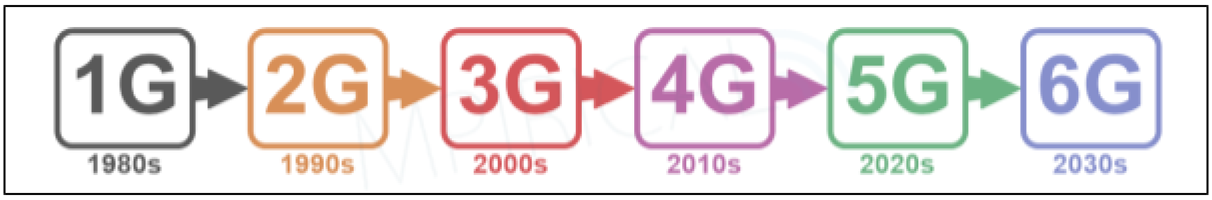
Acknowledgements

I would like to express my sincere gratitude to several individuals who have played a crucial role in the completion of our research project. First and foremost, I extend my heartfelt thanks to Professor Adrian for his unwavering support and guidance throughout this journey. His continuous encouragement and regular check-ins have been invaluable in keeping us on track and motivated. I am also deeply grateful to Professor Joan for his assistance whenever we encountered obstacles along the way. His expertise and willingness to help have been instrumental in overcoming challenges. I extend my appreciation to Tutor Mireia for her timely provision of resources, which gave us a significant head start in our project. Additionally, I would like to acknowledge the contributions of my classmate David Pujals, whose assistance with the alpha build of our website's backend provided the momentum we needed to get started. Lastly, I extend my thanks to Wappsto's technical support team, particularly Andreas Bomholtz, for their prompt responses, clarifications, and solutions to our inquiries regarding Wappsto's functioning. Their assistance has been invaluable in resolving issues efficiently and effectively.

Index

1. What’s 5G?

5G is something we've all heard about at some point, but many people don't understand how it works. 5G is neither 5 gigabytes nor 5 gigahertz, 5G is the fifth GENERATION of mobile phone technologies. The first generation of mobile phone technologies started with the invention of the standard AMPS (Advanced Mobile Phone System) in the United States and the NMT (Nordic Mobile Telephone) System in Europe and NTT in Japan. 1G used analogue communication channels and served exclusively to transmit voice.



Network Evolution, Kevin Moore
Published by mpirical.com, 11th of August 2023

Figure 1

Fast-forward to today, we are currently in the 4G. More specifically, in the end of the same. In other words, the dawn of 5G. For the majority of consumers, a new generation means faster download speed or less lag in game. But that's not all, 5G networks can support a larger number of devices than previous generations and can also handle more data-intensive applications like Virtual and Augmented Reality, Self-Driving Vehicles, and the IoT (Internet of Things). But how does it work? First, we need to know how current generation technology works.

1.1. How does 4G work?

Currently, the generation most present on our phones is the 4G. 4G uses a radio wireless system to translate data through the air and communicate to our phones. In this technology, we have cells[1] that connect our phones with the internet using an ip-based protocol. 4G operates on a completely IP-based network, meaning all data, even voice calls, are converted into packages for transmission. This, combined with a more advanced radio technology, allows 4G to squeeze more data into cellular signals. This translates to significantly faster download speeds and the ability to handle more users on the network at once. Additionally, 4G boasts lower latency, reducing the time it takes for data to travel. In essence, 4G is a significant upgrade over previous generations, offering a faster and more responsive mobile data experience.

[1] A cell is the geographic area that is covered by a single base station in a cellular network.

1.2. 5G First Steps: 5G no Stand-alone

5G non-standalone (NSA) is a cost-effective way for carriers to introduce 5G by utilizing their existing 4G core network. It uses the 5G frequency[2] but with the 4G core that speeds up deployment and reduces investment compared to a complete overhaul. While 5G NSA offers advantages like faster speeds and lower latency, it doesn't unlock the full potential of 5G, particularly in areas requiring ultra-reliable low-latency communication. Think of it as a stepping stone, bridging the gap between 4G and the future of more advanced 5G stand-alone technology.

1.3. The Peak of 5G: 5G Stand-alone

5G Stand-alone (SA) operates on a completely new network architecture compared to its predecessor, 5G Non-Standalone (NSA).

Core Network: 5G SA utilizes a cloud-native 5G Core (NG Core), enabling features not possible with the legacy Evolved Packet Core (EPC) used in 4G and 5G NSA. This NG Core offers greater scalability, flexibility, and programmability for handling the demands of 5G applications.

Radio Access Network (RAN): Both 5G SA and NSA use the 5G New Radio (NR) standard for the air interface, providing faster data speeds and improved spectral efficiency. However, 5G SA decouples the NR from the EPC entirely, relying solely on the NG Core for control and management.

Network Slicing: A key advantage of 5G SA is its ability to support advanced network slicing. The NG Core enables the creation of virtual networks on top of the physical infrastructure, catering to diverse use cases with specific requirements for bandwidth, latency, and security. Each slice can be independently configured and managed, optimizing network resources for applications like Internet of Things (IoT) or Ultra-Reliable Low-Latency Communication (uRLLC).

Basically, 5G SA unlocks the full potential of 5G technology by offering a truly stand-alone network designed to address the low-latency, high-capacity, and mission-critical needs of future applications.

1.4. The State of 5G in Finland

Imagine living in a city where traffic lights magically adjust to the flow of traffic, waste collection happens exactly when it's needed, and the environment is monitored effortlessly.

Let's start with smart cities and infrastructure. Picture this: you're driving through a busy intersection, and instead of waiting at a red light for what feels like an eternity, the traffic light changes just as you approach. How does it know? Well, with 5G's lightning-fast speed

[2] The wave length

and low latency, it can communicate with sensors in real-time to adjust the timing of traffic lights based on congestion levels. This not only saves you time, but it also reduces traffic and makes the whole city more efficient.

But that's not all. Imagine garbage trucks that only pick up bins when they're full, thanks to IoT sensors. This means no more unnecessary trips and a more sustainable waste management system. And with environmental monitoring, we can keep a close eye on things like air quality and noise levels, making our cities healthier and more liveable.

Now, let's talk about industrial automation and manufacturing. In a factory, every minute of downtime can cost a fortune. But with 5G, we can monitor equipment in real-time and catch any potential issues before they become major problems. This means less unexpected downtime and more efficient production. Plus, imagine robots and machines that are connected and working together seamlessly to optimize the manufacturing process. This leads to increased productivity and better products for all of us.

When it comes to healthcare, 5G has the potential to revolutionize remote consultations and telemedicine. With its high bandwidth, we can have high-quality video calls with doctors, no matter where we are. This is especially important for people in remote areas who may not have easy access to healthcare facilities. And with real-time monitoring of patients with chronic conditions, doctors can catch any changes or issues early on, improving patient care and outcomes.

Last but not least, let's talk about logistics and supply chain management. Imagine if we could track the location and condition of goods in real-time using IoT sensors. This data, combined with 5G's speed, allows us to optimize logistics, reduce delivery times, and improve overall efficiency in the supply chain. This means faster delivery of goods to customers and less waste along the way.

Now, let's look at some real-life examples of how Finland is leading the way in implementing these technologies. Elisa, a telecom operator, partnered with the city of Oulu to create a smart city testbed. They're using 5G and IoT sensors to manage traffic flow, optimize waste collection, and monitor environmental conditions. This project is a great example of how these technologies can be applied in a real-world setting.

Another example is Wärtsilä, an engineering giant that is using 5G and IoT sensors to remotely monitor the performance of ships at sea. This allows for predictive maintenance and improved operational efficiency, ensuring that ships are always in top shape and reducing the risk of breakdowns.

And let's not forget about OuluHealth's remote patient monitoring project. With 5G and wearable devices, they're able to monitor patients with chronic conditions remotely. This means early detection of potential health issues and better care for patients, even when they're not in the hospital.

These are just a few examples of how Finland is embracing the power of 5G and IoT. With their focus on research and development, they're constantly exploring new applications and staying at the forefront of this technological revolution. So, get ready for a future where our cities are smarter, our factories are more efficient, our healthcare is more accessible, and our supply chains are optimized. The possibilities are endless, and Finland is leading the way.

1.5. The State of 5g in Spain

Spain was actively rolling out 5G technology across its major cities and urban centers. Telecommunications companies like Telefonica, Vodafone, and Orange were leading the charge in deploying 5G infrastructure.

However, the pace and extent of 5G deployment can vary by region and city, with some areas having more comprehensive coverage than others. Major cities like Madrid, Barcelona, Valencia, and Seville likely had significant 5G coverage, while rural areas might still be in the early stages of adoption.

The Spanish government had been supportive of 5G development, recognizing its potential to drive economic growth and innovation across various sectors such as manufacturing, transportation, healthcare, and entertainment.

2. Self-Driving Vehicles

2.1. Introduction

Since the appearance of the first driver assistance feature: The Automatic Highway Traveller[3] in 1958. The idea of full self-driving cars has been floating around, and honestly, it's just a matter of time until we achieve a fully reliable autonomous vehicle. Vehicle autonomy refers to the ability of a vehicle to drive autonomously without the need for human intervention. It's true that currently we are closer than ever to achieve this goal. But expectations are beginning to shift as the dream of fully autonomous cars is proving to be far more complex than we anticipated.

We can simplify the information processing of autonomous vehicles into layers, at the bottom of this stack lies navigation or path planning, which is the most robustly developed area of automated driving technology. Because this technology is an extension of human use navigation systems, adapting it to autonomous vehicles has proven to be relatively easy. The best example of this is modern vehicles, which all come included with a refined GPS navigation system (More of this in the practical part).

The next challenge to meet is the environmental perception and car control. The process of detecting and mapping the environment around the vehicle for obstacle avoidance. The primary mechanism of environmental perception are laser[4], radar and visual navigation, more on this later. But in simple terms, laser navigation is limiting for mass production. Radar navigation lacks the level of detail needed to be used as a primary sensor or a task as sensitive as this one. And finally we got visual navigation, which is by far the best candidate for this task. Combined with the recent explosion of AI and massive amounts of data it could be the solution for our problems. Nonetheless, we can't ignore the fact that this navigation system can be easily confused by small animals, weirdly shaped objects and weather conditions. This fatal flaw has put us in a stationary state, where the only thing we do is to train our AI until we get o a point where it's on par with human comprehension.

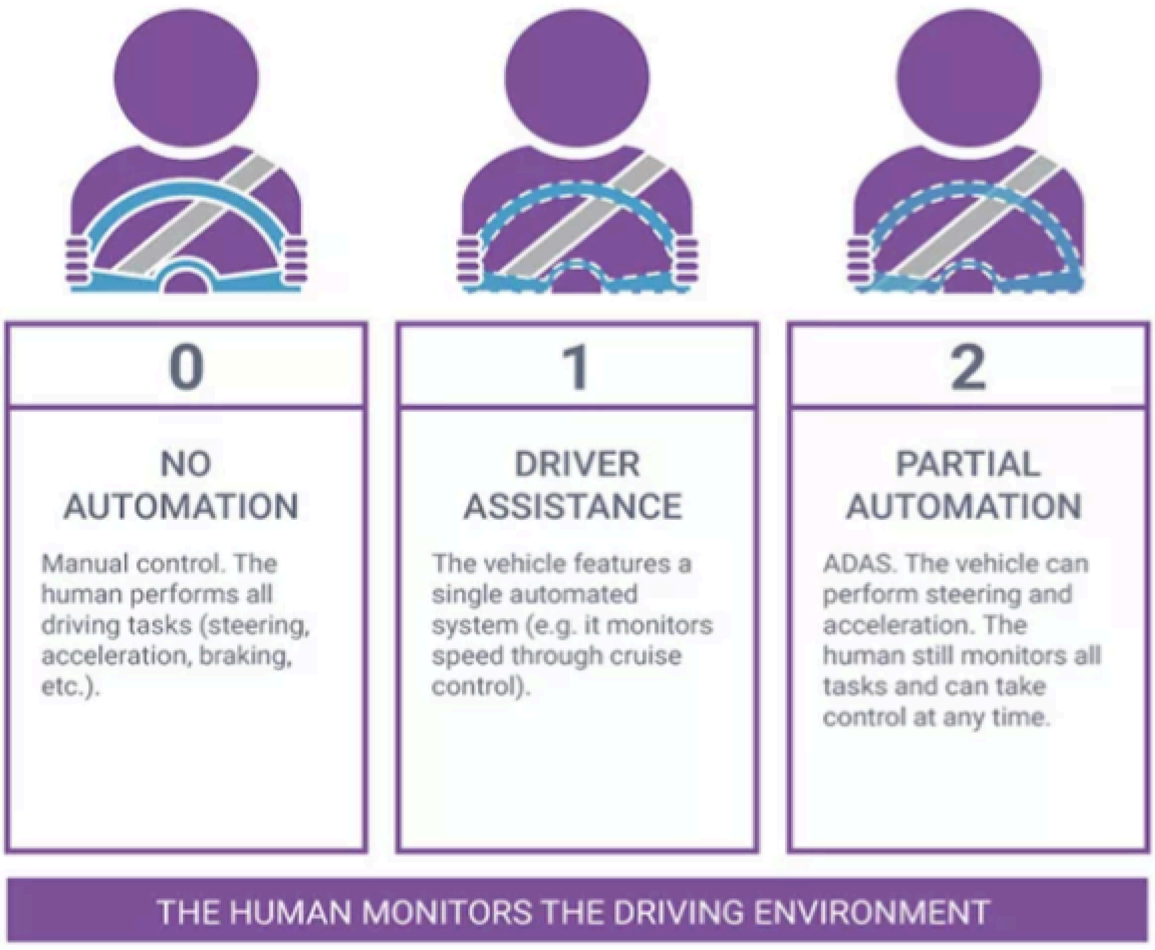
Considering this, the European Union (EU), regulations for autonomous vehicles focus on ensuring their safe and responsible deployment on European roads. This includes adherence to comprehensive safety standards outlined in regulations such as the General Safety Regulation (EU) 2019/2144 and compliance with type approval processes for vehicle design, construction, and safety features. The EU aligns its regulations with United Nations Economic Commission for Europe (UNECE) standards where applicable and provides guidelines and recommendations for the development, testing, and deployment of autonomous vehicles, addressing issues such as safety, cybersecurity, data protection, and liability. Additionally, the EU promotes the deployment of Cooperative Intelligent Transport Systems (C-ITS) to enable vehicle-to-vehicle and vehicle-to-infrastructure communication, facilitating various safety and efficiency applications (More on this later).

[3] A system that used radar to detect other vehicles and adjust speed accordingly

[4] Laser navigation can be categorized as either single line, multiline and omnidirectional. The more lines, the more information perceived and the slower is the generation of real-time output

2.2. Levels of Vehicle Autonomy

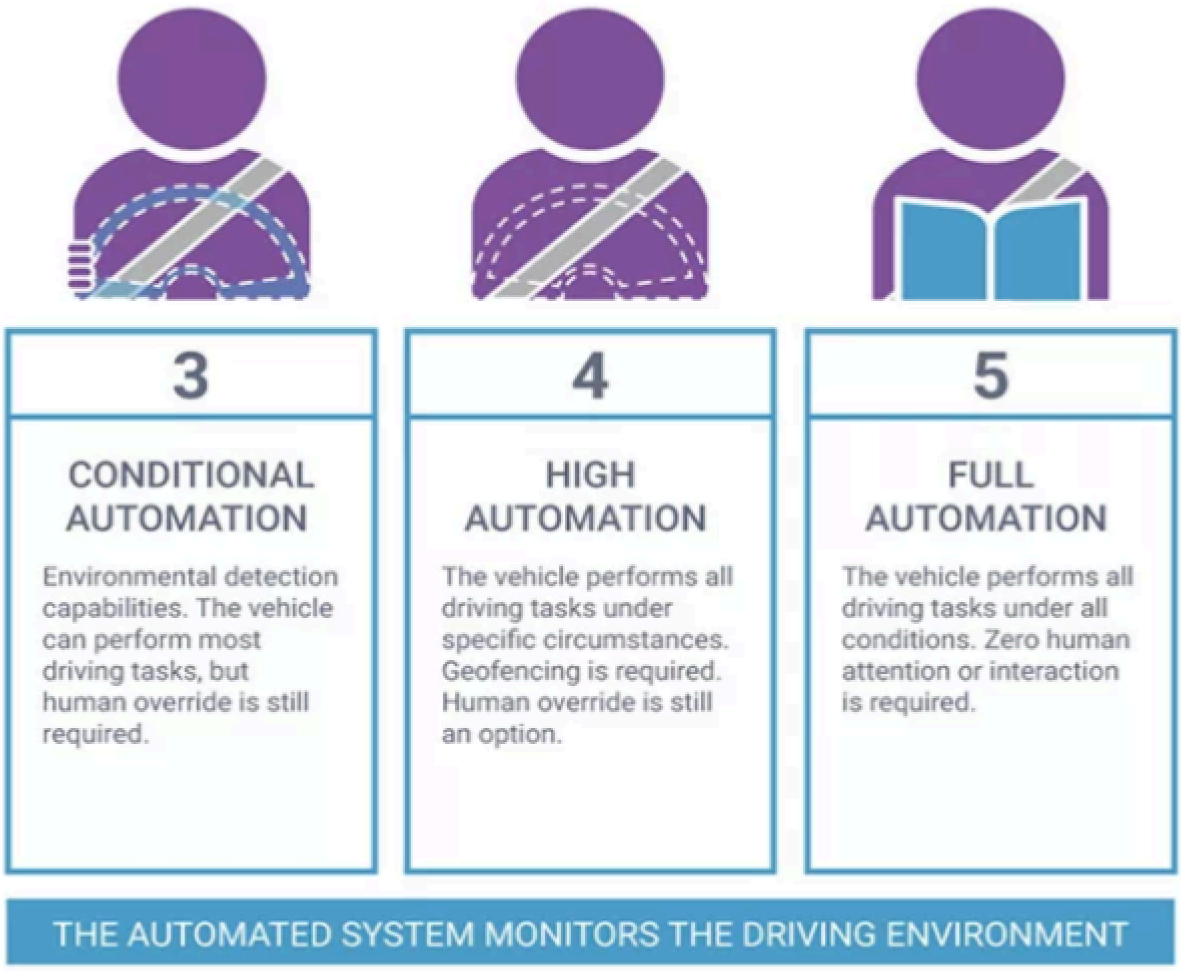
The Society of Automotive Engineers (SAE) has established six levels of driving automation, ranging from Level 0 (fully manual) to Level 5 (fully autonomous), which have been adopted by the U.S. Department of Transportation. At Level 0, vehicles are manually controlled, but because of how this level is described, even a 2019 Ford Edge is included in level 0 despite having Emergency braking and cruise control, while Level 1 features single automated systems such as lane assist or adaptive cruise control. Level 2 involves partial automation, where the vehicle can control steering and acceleration, but a human driver must remain in control. Here is where the commercially available cars like Tesla stop.


0
NO AUTOMATION
Manual control. The human performs all driving tasks (steering, acceleration, braking, etc.).
1
DRIVER ASSISTANCE
The vehicle features a single automated system (e.g. it monitors speed through cruise control).
2
PARTIAL AUTOMATION
ADAS. The vehicle can perform steering and acceleration. The human still monitors all tasks and can take control at any time.
THE HUMAN MONITORS THE DRIVING ENVIRONMENT

Levels of Driving Automation Figure 2
Published by Synopsys, Inc, 16th of August 2019

Level 3 marks a significant technological advancement, with vehicles capable of making informed decisions but still requiring human intervention at some point or another, but they are legally allowed to not pay attention to the road. The jump from level 2 to level 3 is so difficult because at this point the automated system monitors the driving environment. To mitigate the risk that this level imposes, autonomy at this level is conditional to certain driving conditions that the manufacturer defines. These conditions are referred to as *Operation Design Domain* or *ODD* for short. The official definition for *ODD* is:

> «Operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics.» SAE J3016, 2021.


3

CONDITIONAL AUTOMATION

Environmental detection capabilities. The vehicle can perform most driving tasks, but human override is still required.

4

HIGH AUTOMATION

The vehicle performs all driving tasks under specific circumstances. Geofencing is required. Human override is still an option.

5

FULL AUTOMATION

The vehicle performs all driving tasks under all conditions. Zero human attention or interaction is required.

THE AUTOMATED SYSTEM MONITORS THE DRIVING ENVIRONMENT

Levels of Driving Automation Figure 3
Published by Synopsys, Inc, 16th of August 2019

Audi introduced the world's first production Level 3 vehicle, the 2019 Audi A8L, featuring Traffic Jam Pilot. However, due to shifting regulatory processes in the U.S., it's classified as Level 2 there but as Level 3 in Europe.

Moving up to Level 4, vehicles can intervene in emergencies or system failures without human input, but human override remains an option. These vehicles operate in self-driving mode within limited areas, usually urban environments with lower speeds, known as geofencing. Examples include NAVYA's Level 4 shuttles and Alphabet's Waymo Level 4 self-driving taxi service. Companies like Magna and Volvo are also investing in Level 4 technology for applications in ride-sharing[5] and the robotaxi market, demonstrating ongoing advancements in autonomous vehicle technology and deployment.

Finally, we have level 5, as you probably guessed, level 5 vehicles are fully autonomous and can operate everywhere and under all driving conditions. The system should be able to comfortably navigate through the environment with zero intervention required.

2.3. The Importance of 5G in the autonomy of vehicles

The integration of 5G technology is playing a crucial role in advancing autonomous vehicles, offering improved connectivity, speed, and reliability. This technology enables seamless communication between vehicles, infrastructure, and other road users, enhancing the real-time data exchange necessary for autonomous driving systems to function effectively and safely.

The main challenge of fully autonomous vehicles is the unpredictability of the other vehicles around it. This is the main focus of 5G implementation in it and the IoT. Being able to communicate the destination and the path taken to other vehicles can be a revolutionizing advancement that might save the life of hundreds if not thousands of drivers and pedestrians

With its lower latency and higher bandwidth capabilities, 5G empowers vehicles to communicate with each other and with traffic signals, leading to enhanced road safety and more efficient traffic management systems.

The ongoing debate between 5G and Wi-Fi for vehicle connectivity highlights the benefits and limitations of each technology. While Wi-Fi has traditionally been used for vehicle-to-vehicle communication, the superior performance of 5G in terms of latency, coverage, and reliability is driving its adoption for future autonomous vehicles. The transition to 5G is seen as essential for the evolution of connected and autonomous vehicles, enabling advanced features such as real-time mapping, sensor data sharing, and remote vehicle monitoring. Countries like China are leading the way in implementing cellular technology for vehicles, shaping the global automotive industry's standard for future vehicle connectivity.

[5] Refers to a transportation service where individuals share a vehicle for a journey

3. DPC (Data Processing Center)

Of course, every IT centered project or even every office needs a network infrastructure. This provides total control over all of your devices. You might be thinking, a project about autonomous vehicles won't need that big of an infrastructure, right? Well, we went the extra mile and made a complete and functioning network environment.

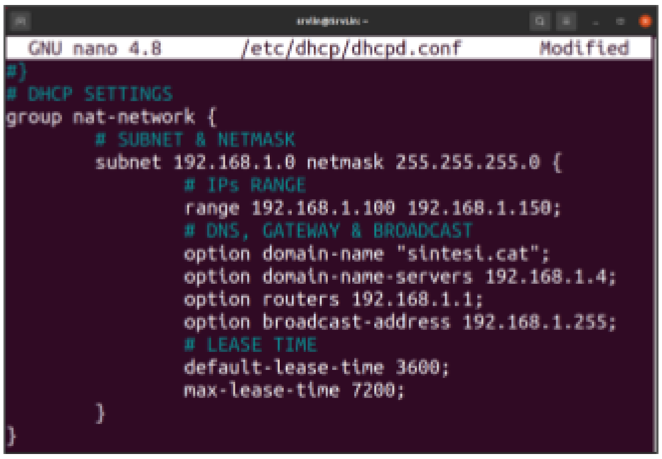
First, we started with the basic network services like DHCP and DNS, then we linked the different users with an active directory domain, and finally we installed a shared storage NAS service. Next, we will go through the different virtual machines and services we installed and how we installed them. Bear in mind that this environment is operating on a NAT network, which insures internet connection for all the devices, and most importantly, provides communication between the different machines

3.1. SrvLin (Ubuntu Desktop 20.03)

For this task, we have chosen Ubuntu distro for convenience, it is famous for its beginner-friendly interface. This virtual machine will be the head controller of our network. It will consist of a DHCP service to assign IP addresses, a DNS service for translating human-friendly domain names into machine-readable IP addresses, a Proxy service for surveying HTTP and HTTPS requests and denying access to unwanted websites and, finally, a shared printing service. Next, we will dive in every service individually:

- ***DHCP (isc)***

ISC-DHCP-Server, provided by the Internet Systems Consortium (ISC), facilitates the automatic assignment of IP addresses and network configuration parameters to devices within a network. To configure ISC-DHCP-Server on our Ubuntu server, we'll primarily work with the /etc/dhcp/dhcpd.conf file.

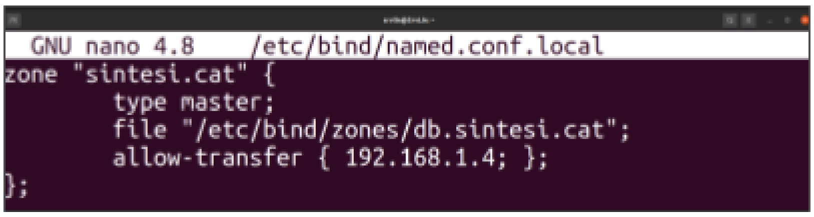


```
  GNU nano 4.8          /etc/dhcp/dhcpd.conf          Modified
#}
# DHCP SETTINGS
group nat-network {
        # SUBNET & NETMASK
        subnet 192.168.1.0 netmask 255.255.255.0 {
                # IPs RANGE
                range 192.168.1.100 192.168.1.150;
                # DNS, GATEWAY & BROADCAST
                option domain-name "sintesi.cat";
                option domain-name-servers 192.168.1.4;
                option routers 192.168.1.1;
                option broadcast-address 192.168.1.255;
                # LEASE TIME
                default-lease-time 3600;
                max-lease-time 7200;
        }
}
```

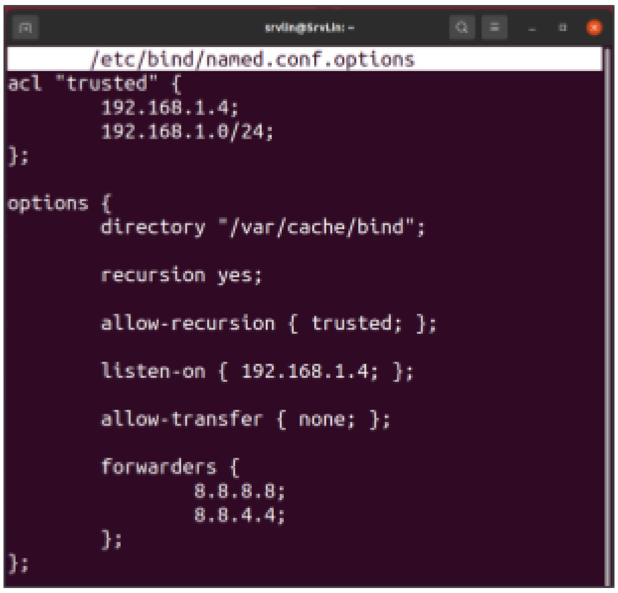
Here, we define essential DHCP settings, including subnet details, IP address range allocations, default gateway, DNS server addresses, lease durations, and any custom options. Additionally, ISC-DHCP-Server allows us to set up DHCP reservations, ensuring specific devices consistently receive the same IP addresses, which can be crucial for network management and device identification purposes.

- ***DNS (bind)***

Bind9, developed by the Internet Systems Consortium (ISC), is a robust DNS server software widely used for translating domain names into IP addresses. Configuration of Bind9 on our Ubuntu server primarily involves setting up forward and reverse lookup zones, typically managed through the /etc/bind/named.conf.local file.



```
  GNU nano 4.8     /etc/bind/named.conf.local
zone "sintesi.cat" {
        type master;
        file "/etc/bind/zones/db.sintesi.cat";
        allow-transfer { 192.168.1.4; };
};
```



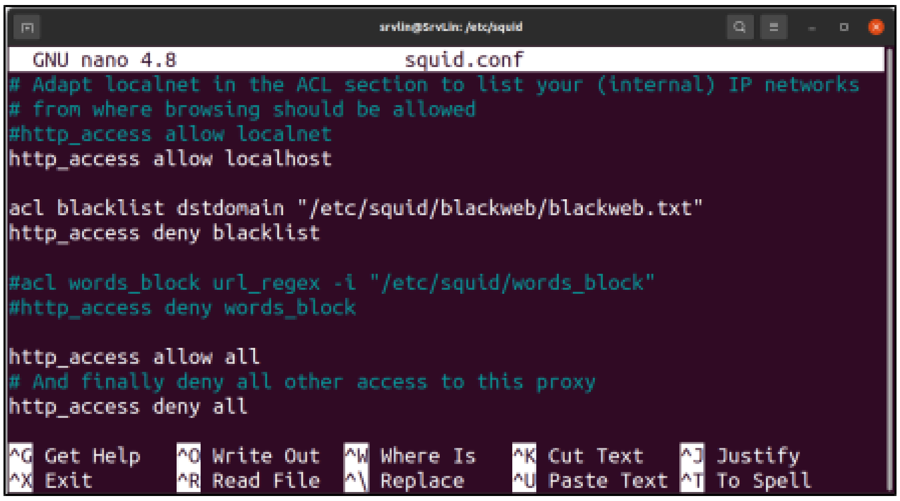
```
            /etc/bind/named.conf.options
acl "trusted" {
        192.168.1.4;
        192.168.1.0/24;
};

options {
        directory "/var/cache/bind";

        recursion yes;

        allow-recursion { trusted; };

        listen-on { 192.168.1.4; };

        allow-transfer { none; };

        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
};
```

DNS 192.168.1.4

Here, we define zone records such as A records for mapping domain names to IP addresses and PTR records for reverse DNS lookup. Additionally, configuration of global DNS options, including DNS forwarders and caching settings, is specified in the /etc/bind/named.conf.options file. Bind9's flexibility and extensive feature set make it a preferred choice for DNS service deployment in diverse network environments.

- ***Proxy (squid)***

Squid, an open-source caching proxy server, offers powerful capabilities for monitoring and controlling HTTP and HTTPS traffic. Developed by the Squid Project, Squid is highly configurable and adaptable to various network architectures. Setting up Squid on our Ubuntu server involves editing the /etc/squid/squid.conf configuration file.

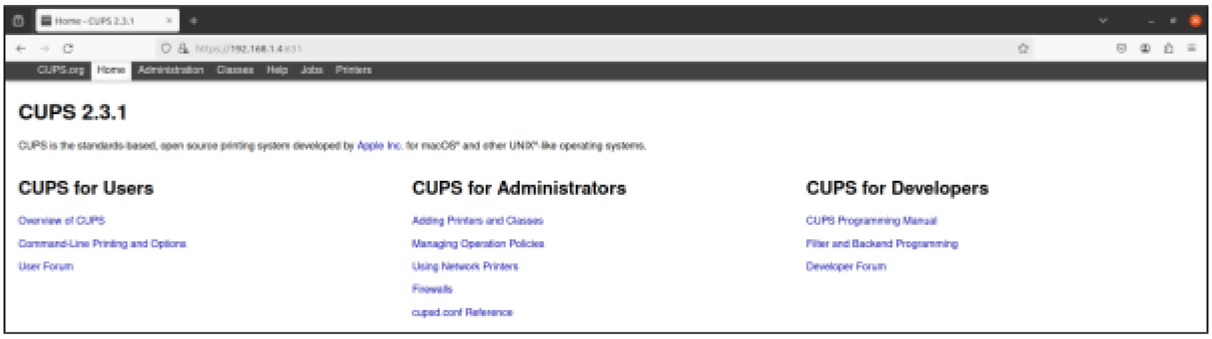


```
srvlin@SrvLin: /etc/squid
  GNU nano 4.8                      squid.conf
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
#http_access allow localnet
http_access allow localhost

acl blacklist dstdomain "/etc/squid/blackweb/blackweb.txt"
http_access deny blacklist

#acl words_block url_regex -i "/etc/squid/words_block"
#http_access deny words_block

http_access allow all
# And finally deny all other access to this proxy
http_access deny all

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell
```

Here, we define Access Control Lists (ACLs) to specify allowed or denied access for clients and destinations. Squid supports features such as caching, SSL/TLS interception, authentication, and logging, providing granular control over web traffic. Moreover, Squid's transparent proxying capabilities make it suitable for intercepting and filtering network traffic without requiring client-side configuration changes.

- ***Printing Service (cups)***

The Common Unix Printing System (CUPS) is the de facto printing system used on Unix-like operating systems, including Ubuntu. Developed by Apple Inc., CUPS provides a flexible and reliable printing solution for networked environments. Configuring CUPS on our Ubuntu server is primarily done through its web interface (http://localhost:631).



CUPS 2.3.1

CUPS is the standards-based, open source printing system developed by Apple Inc. for macOS® and other UNIX®-like operating systems.

CUPS for Users

Overview of CUPS
Command-Line Printing and Options
User Forum

CUPS for Administrators

Adding Printers and Classes
Managing Operation Policies
Using Network Printers
Firewalls
cupsd.conf Reference

CUPS for Developers

CUPS Programming Manual
Filter and Backend Programming
Developer Forum

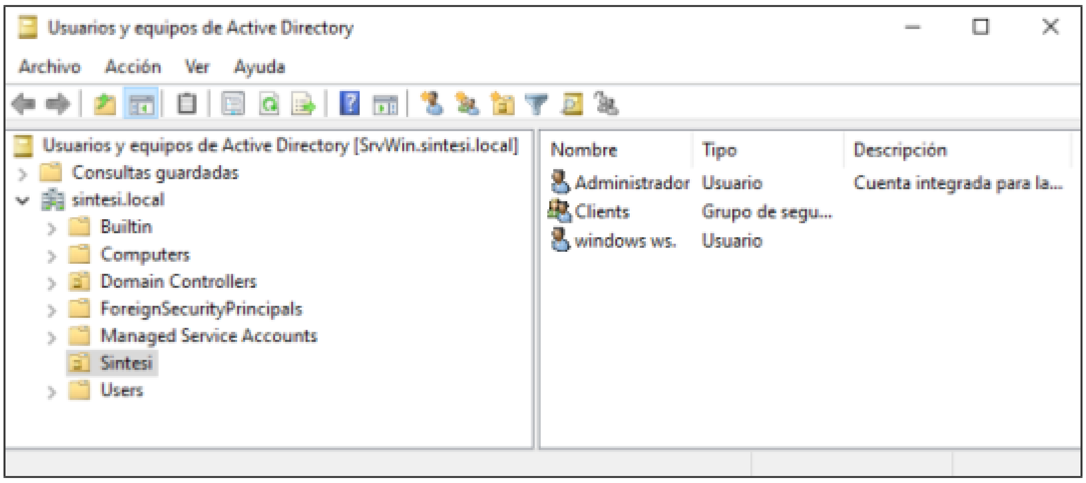
Here, we can add printers, configure printer options such as paper size and print quality, and enable printer sharing for network access. CUPS supports a variety of printing protocols, including IPP, LPD, and SMB, allowing seamless integration with different types of printers and operating systems. Additionally, CUPS offers advanced features such as printer access control, job management, and support for printer drivers, making it a comprehensive printing solution for our network environment.

3.2. *SrvWin (Windows Server 2022 GUI)*

For our Windows Server 2022 GUI, our primary focus will be on deploying Active Directory (AD) and DNS services.

- ***Active Directory***

Active Directory, developed by Microsoft, serves as the backbone for centralized authentication, authorization, and management of network resources within a Windows-based environment.



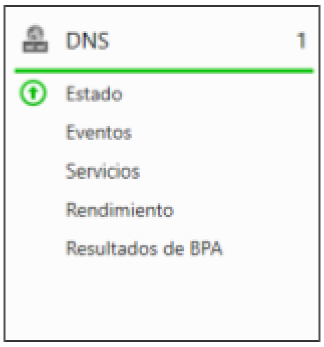
Usuarios y equipos de Active Directory [SrvWin.sintesi.local]
- Consultas guardadas
- sintesi.local
  - Builtin
  - Computers
  - Domain Controllers
  - ForeignSecurityPrincipals
  - Managed Service Accounts
  - Sintesi
  - Users

Nombre	Tipo	Descripción
Administrador	Usuario	Cuenta integrada para la...
Clients	Grupo de segu...	
windows ws.	Usuario	

With Active Directory, we can create and manage user accounts, group policies, and security settings from a single, centralized location. Setting up Active Directory involves promoting the Windows Server to a domain controller, configuring domain settings, and creating organizational units (OUs) to organize network resources.

- ***DNS***

Additionally, integrating DNS with Active Directory ensures proper name resolution for domain-joined clients and facilitates domain controller location and replication. More on this later.

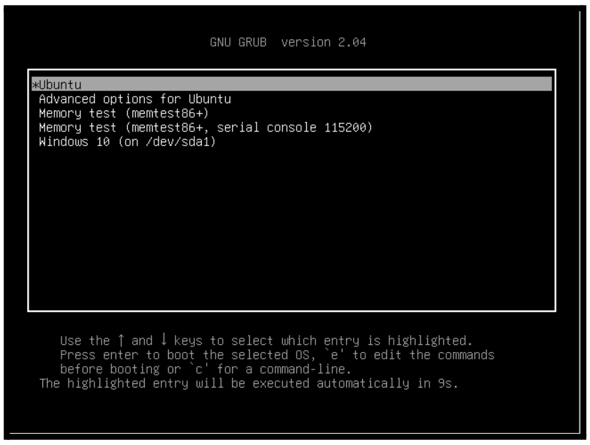


3.3. *Client (Windows 11 / Ubuntu Desktop 20.03)*

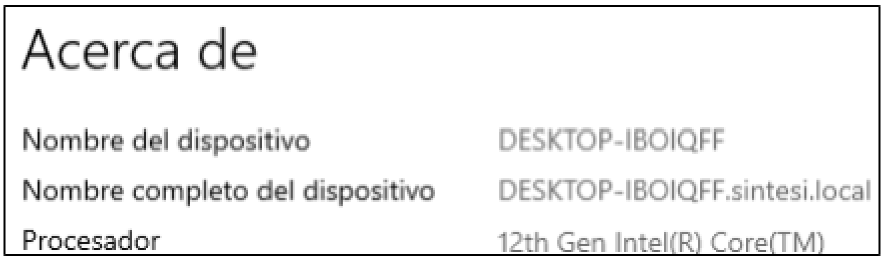
Our client machine, configured with both Ubuntu and Windows operating systems in a dual-boot configuration, offers flexibility and versatility for end-user computing needs. To enable seamless integration with Active Directory (AD) from our dual-boot Ubuntu and Windows client machine, we configured both operating systems to authenticate against the AD domain hosted on our Windows Server.

- ***Windows***

Windows offers compatibility with a vast ecosystem of commercial software and hardware devices, making it suitable for various business and personal computing tasks.



```
GNU GRUB  version 2.04

*Ubuntu
 Advanced options for Ubuntu
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)
 Windows 10 (on /dev/sda1)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line.
 The highlighted entry will be executed automatically in 9s.
```



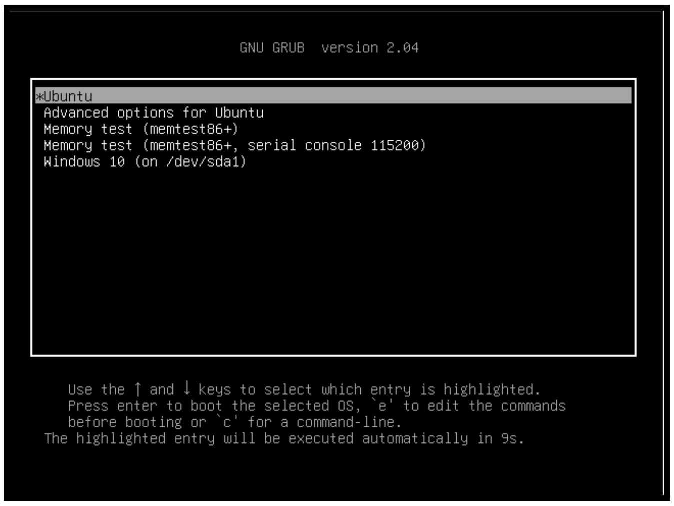
Acerca de

Nombre del dispositivo	DESKTOP-IBOIQFF
Nombre completo del dispositivo	DESKTOP-IBOIQFF.sintesi.local
Procesador	12th Gen Intel(R) Core(TM)

Users can join the domain directly from the Control Panel or Settings interface, allowing them to log in with their domain credentials and access network resources seamlessly. DNS plays a crucial role in facilitating domain name resolution and ensuring seamless communication with domain controllers and other network resources.

- ***Ubuntu***

On the other hand, in Ubuntu, users can access a wide range of free and open-source software, including productivity tools, development environments, and multimedia applications.

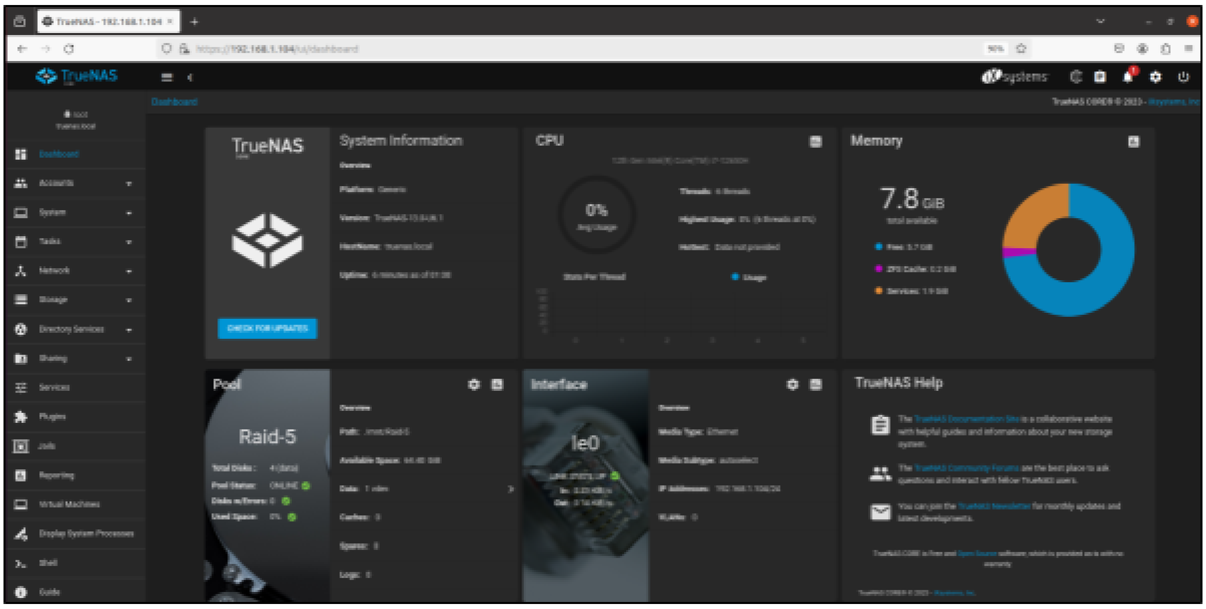


```
GNU GRUB  version 2.04

*Ubuntu
 Advanced options for Ubuntu
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)
 Windows 10 (on /dev/sda1)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line.
The highlighted entry will be executed automatically in 9s.
```

We configured the Samba software suite to enable integration with AD, allowing Ubuntu users to authenticate against the domain controller and access shared network resources using their AD credentials.

3.4. SrvNAS (TrueNAS)

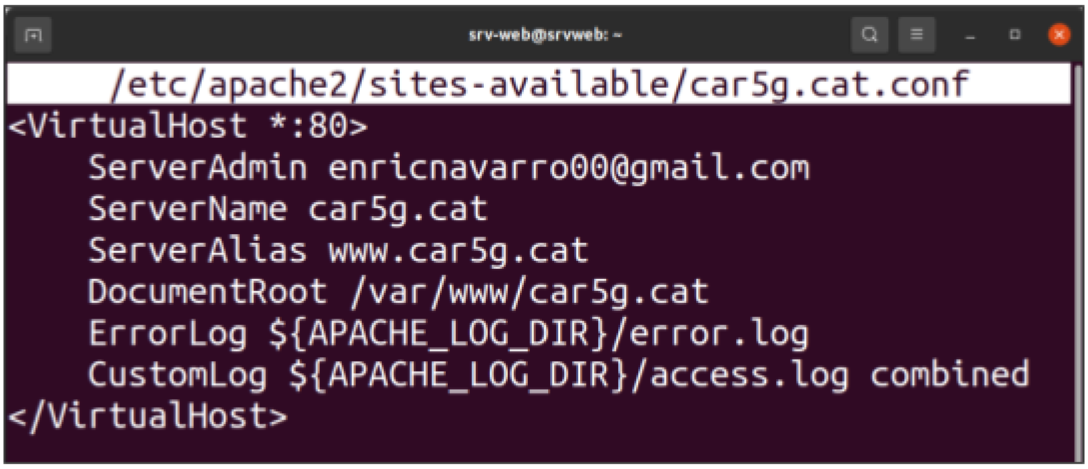
TrueNAS, based on the open-source FreeNAS project, provides a robust and scalable network-attached storage (NAS) solution for our network environment. TrueNAS offers features such as data deduplication, encryption, snapshotting, and replication, making it ideal for storing and sharing large volumes of data across the network.



Setting up TrueNAS involves installing the TrueNAS operating system on dedicated hardware or as a virtual machine, configuring storage pools and datasets, and setting up sharing protocols such as SMB/CIFS, NFS, and FTP. With TrueNAS, we can centralize data storage, ensure data integrity and availability, and implement backup and disaster recovery strategies to safeguard critical data assets.

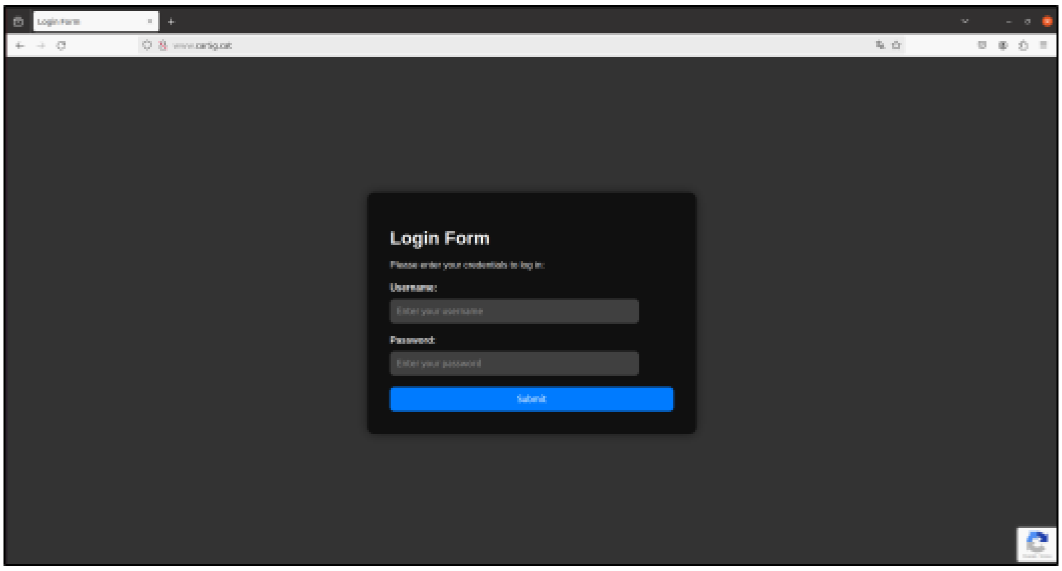
3.5. SrvWeb (Ubuntu Desktop 20.03)

Our SrvWeb, running Ubuntu Server, serves as a pivotal component in our network architecture. Primarily, it hosts the Apache web server along with PHP, forming the backbone of our web-based applications and services. Apache's versatility and robustness make it an ideal choice for serving dynamic content, handling HTTP requests, and hosting websites with PHP support. With PHP installed, our server can execute server-side scripts and interact with databases, enabling dynamic content generation and data processing.



```
srv-web@srvweb: ~
      /etc/apache2/sites-available/car5g.cat.conf
<VirtualHost *:80>
    ServerAdmin enricnavarro00@gmail.com
    ServerName car5g.cat
    ServerAlias www.car5g.cat
    DocumentRoot /var/www/car5g.cat
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

The primary function of our SrvWeb is to communicate with wappsto's API.



Our SrvWeb hosts a website that displays the geolocation data of the micro:bit in an intuitive and user-friendly manner. This website, powered by Apache and PHP, provides a graphical interface for users to visualize and interact with the collected data. Using HTML, CSS, and JavaScript, we design a responsive and engaging interface that presents the geolocation data in maps, charts, or tables, depending on user preferences. By integrating the website with backend Javascript scripts, we enable real-time updates and data retrieval, ensuring that users have access to the latest information about the micro:bit's whereabouts.

4. The practical part - The making of the localization system

The time has come to talk about the main objective of this project. As you might be thinking, making a fully autonomous vehicle is very complex, and covering all the parts that make said vehicle is no easy task. Considering the time window that we work with, we have decided to indulge in the localization system of a theoretical 5g supported autonomous vehicle.

4.1. The beating heart of the Localization system - Micro:Bit / Wappsto

For the making of such a system, we'll be using a BBC micro:bit board, an educational computing platform, that has seen tremendous popularity due to its versatility and ease of use in teaching programming and electronics to learners of all ages. With its compact size and intuitive interface, the micro:bit serves as an excellent tool for introducing students to coding concepts and fostering creativity in building digital projects.

BBC micro:bit, Nicholas H.Tollervey Figure 4
Published by ntoll.org, 20th of October 2015

One of the standout features of the micro:bit is its extensive array of built-in sensors and components, including an accelerometer, magnetometer, and LED matrix. These features enable users to create a wide variety of interactive projects, from simple games and animations to more complex IoT applications. The micro:bit's compatibility with multiple programming languages, such as Python and JavaScript, further enhances its appeal by catering to users with different levels of coding proficiency.

Using the block-based programming tool that the company itself provides, we'll program it to generate coordinates of its location. The micro:bit itself does not have built-in geolocation capabilities. However, being this very malleable and flexible tool, it has access to a ton of external extensions and plug-ins that make it go beyond its capabilities. Enter Wappsto:bit.

Wappsto:bit is an innovative expansion board designed to enhance the capabilities of the BBC micro:bit. Developed by Wappsto, it extends the functionality of the micro:bit by adding features like Wi-Fi connectivity, additional sensors, and expanded input/output options. This expansion opens up a wide range of possibilities for creating IoT (Internet of Things) projects and exploring advanced programming concepts.

Wappsto:bit, Akram & Enric Figure 5

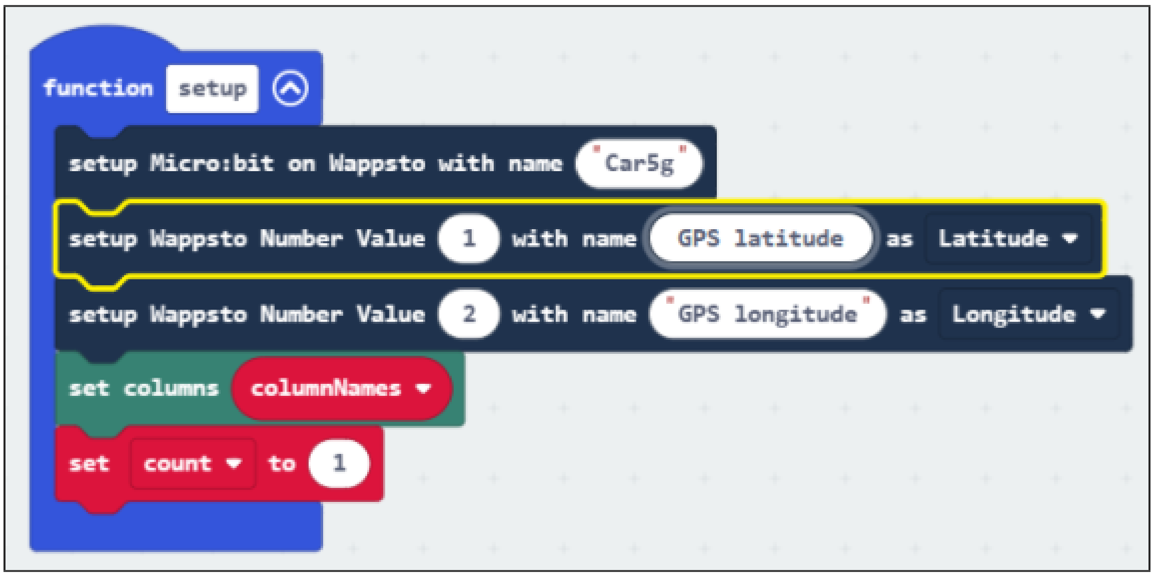
One of the key features of Wappsto:bit is its integration with the Wappsto platform, which provides a user-friendly interface for managing and controlling connected devices. This platform enables users to create custom dashboards, automate tasks, and visualize data from their micro:bit projects in real-time.

As we mentioned previously, micro:bit has its own block-based code editor, and that is exactly what we will be using to write the foundation code that will be the beating heart of this device.

4.2. Micro:bit code deep dive

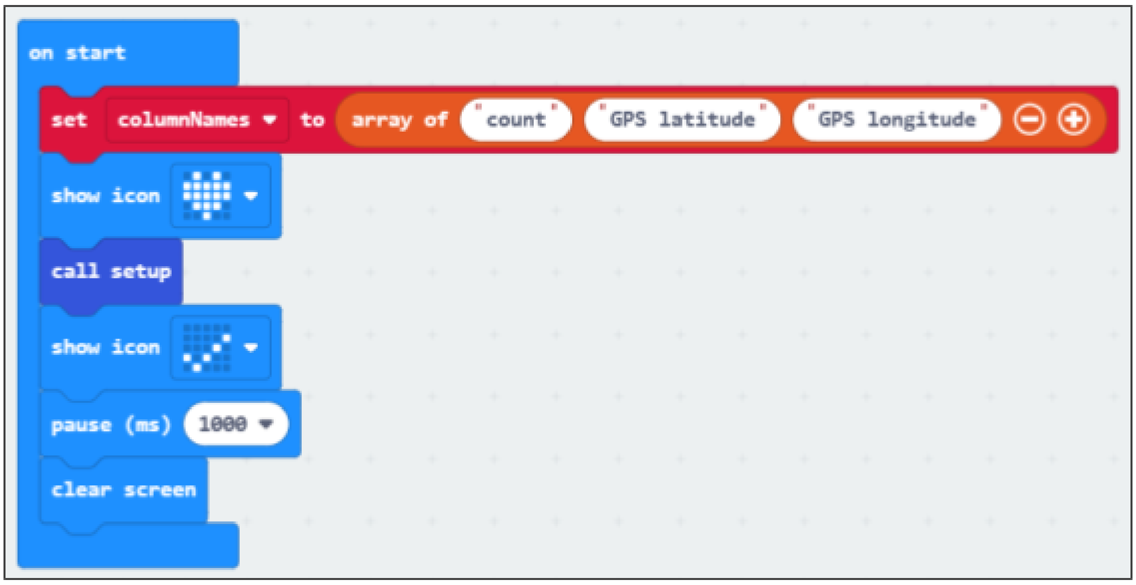
before we start, we need to understand what we want the micro:bit to do. Ultimately, the final goal is to generate a dataflow containing location data. As we are at it, we'll make it have a log to have a physical log table inside the micro:bit memory. To achieve all of this we will be using two extensions: Data Logger and Wappsto.

First, we'll create a function called "setup", this well initiate the micro:bit with the name "Car5g" and get the latitude and longitude coordinates. Also, it will create the log table.



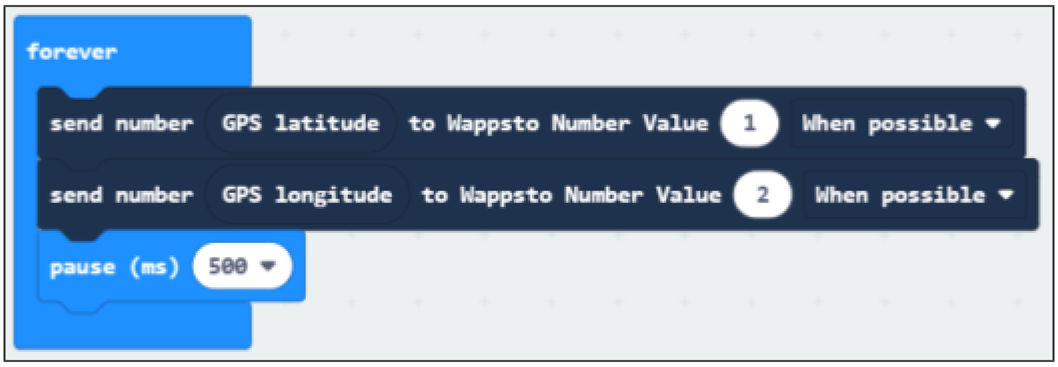
```
function setup
    setup Micro:bit on Wappsto with name "Car5g"
    setup Wappsto Number Value 1 with name "GPS latitude" as Latitude
    setup Wappsto Number Value 2 with name "GPS longitude" as Longitude
    set columns columnNames
    set count to 1
```

On start, we will add the columns "count", "GPS latitude" and GPS longitude. For some visual feedback, we'll make the micro:bit show a heart icon. Then we'll call the setup function, and if it's successful, it will show a check mark symbol. We add a 1 sec pause before clearing the screen.



```
on start
    set columnNames to array of "count" "GPS latitude" "GPS longitude"
    show icon
    call setup
    show icon
    pause (ms) 1000
    clear screen
```

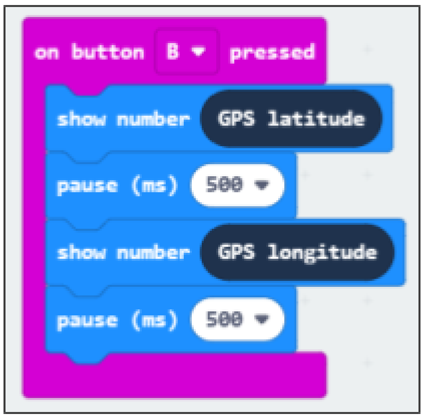
To initialize the data flow, we simply send the coordinates whenever it's possible with a forever block.



```
forever
    send number GPS latitude to Wappsto Number Value 1 When possible
    send number GPS longitude to Wappsto Number Value 2 When possible
    pause (ms) 500
```

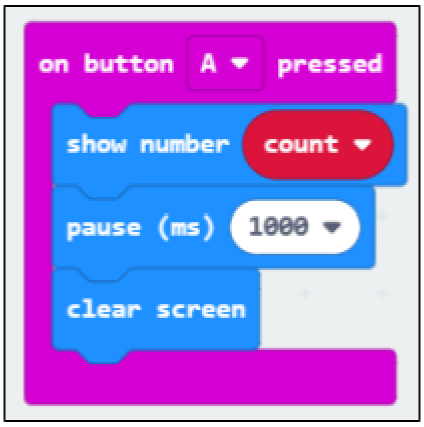
Technically speaking, the core code is done for. The micro:bit can be up and running, generating the coordinates and sending it to the wappsto interface via wappsto:bit. But we'll add more functionalities.

Let's give the buttons a use, for the B button, it will display the coordinates when pressed.



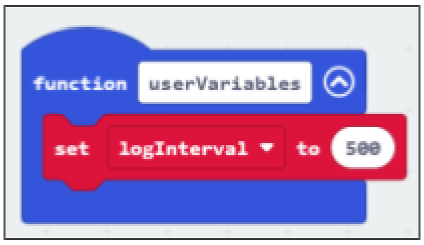
```
on button B pressed
    show number GPS latitude
    pause (ms) 500
    show number GPS longitude
    pause (ms) 500
```

For the A button, it will display the log count.



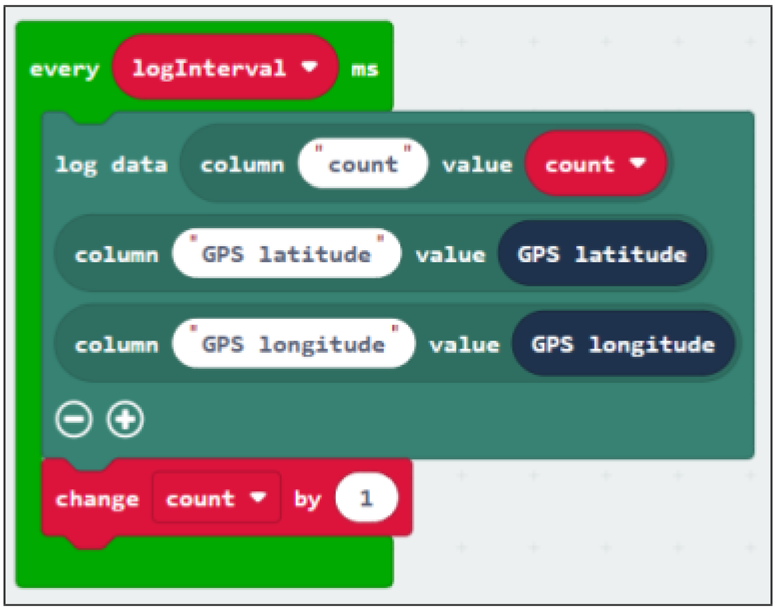
```
on button A pressed
    show number count
    pause (ms) 1000
    clear screen
```

Talking about the log, let's programme the log system. First, well set the logging interval with the function "userVariables".



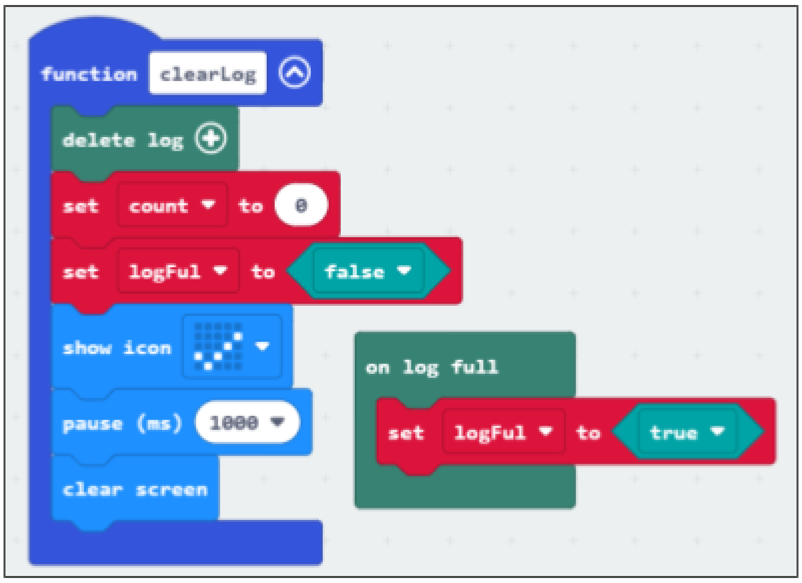
```
function userVariables
    set logInterval to 500
```

With this block, we'll be logging all the data we get from the main code into the table we created.



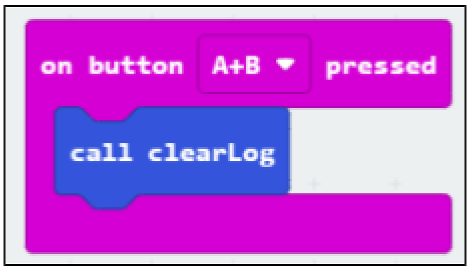
```
every logInterval ms
    log data column "count" value count
        column "GPS latitude" value GPS latitude
        column "GPS longitude" value GPS longitude
    change count by 1
```

We don't want to be logging infinitely, so we'll create a "clearLog" function. It will delete the whole log table and display a visual feedback when cleared.



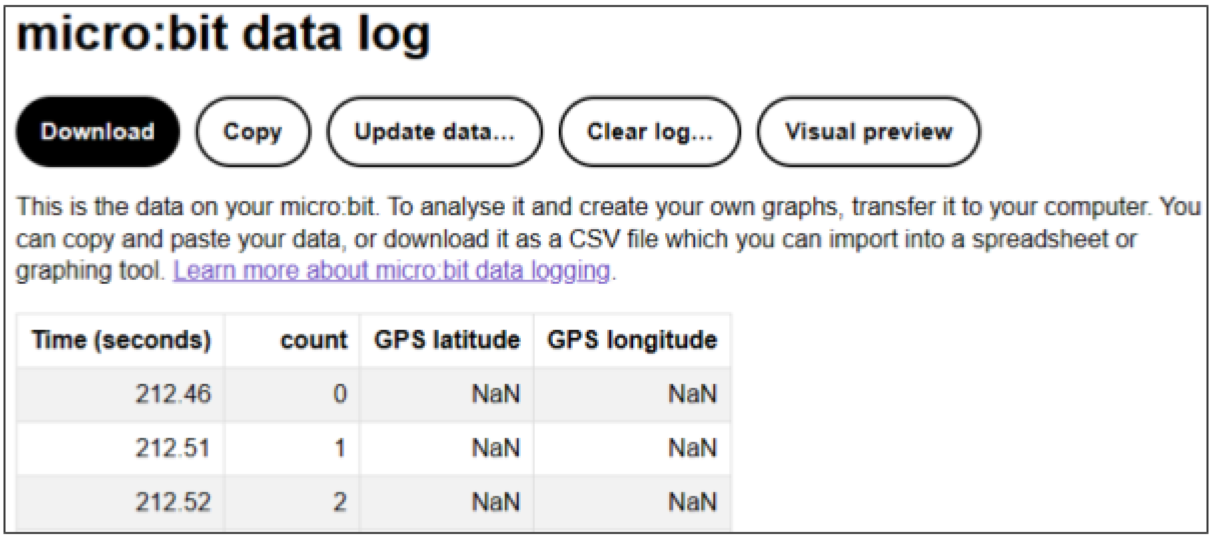
```
function clearLog
    delete log
    set count to 0
    set logFul to false
    show icon
    pause (ms) 1000
    clear screen

on log full
    set logFul to true
```

This function will be called when pressing A and B simultaneously.



```
on button A+B ▾ pressed
    call clearLog
```

The log table will be something like this:



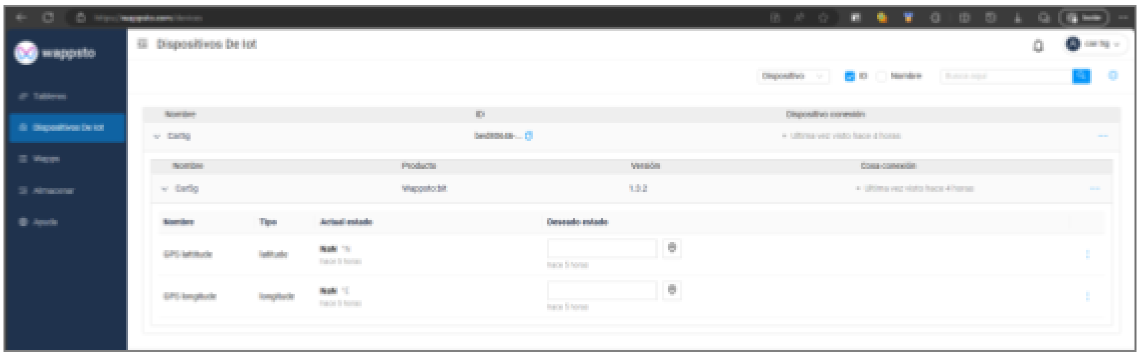
micro:bit data log

This is the data on your micro:bit. To analyse it and create your own graphs, transfer it to your computer. You can copy and paste your data, or download it as a CSV file which you can import into a spreadsheet or graphing tool. Learn more about micro:bit data logging.

Time (seconds)	count	GPS latitude	GPS longitude
212.46	0	NaN	NaN
212.51	1	NaN	NaN
212.52	2	NaN	NaN

4.3. *The wappsto chronicles*

At this point, we have the location data we need being sent to the official wappsto website, so naturally, we started studying this tool, little did we know that this is when our project will hit a huge wall.



Nombre	Tipo	Actual estado	Deseado estado
GPS latitude	latitude	NaN	
GPS longitude	longitude	NaN	

It may be hard to believe that across all the tools that the wappsto provides, from extensions to native dashboards, none has resulted being useful for our cause. Our goal is to monitor the movement of our device and track it. The only tool close to this objective is the map dashboard that only displays the current location of the wappsto:bit, but half the time, it doesn't even work!

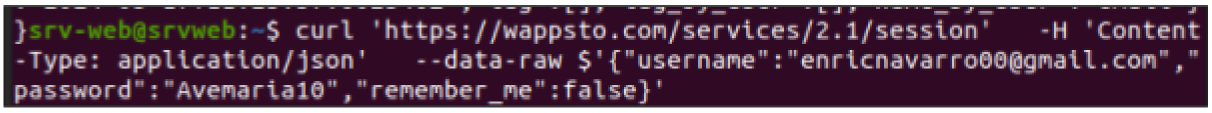
So we decided to migrate the data to an external server so we can manipulate the data. This implies working with wappsto's API. Now, for the majority of people, this doesn't represent an issue. But taking into account our experience with APIs, which is none by the way, and also the fact that wappsto's API is not made for inexperienced developers. This resulted in a huge boulder standing in our journey towards our goal.

After several days of research, going on and on trying to find a way to send the data from wappsto or fetch it fom there and store it in a Linux server. We learned the following: we need to open a *WebSocket*. A WebSocket is a communication protocol that provides a full-duplex, bidirectional communication channel over a single, long-lived TCP connection. It enables real-time, low-latency data exchange between a client (in this case our Linux machine) and a server (wappsto). Unlike traditional HTTP connections, which are stateless and involve sending requests and receiving responses, WebSocket connections remain open after the initial handshake, allowing data to be sent and received asynchronously at any time without the overhead of repeated handshakes. And that's exactly what we are aiming for.

That's the theory of it, how can we put it into practice. One simple way to do it is with the "wscat" tool in Linux terminal. It seems to need some kind of authorization, going through the API documentation, we find out that we need to request a session ID. After some research, we found out that we need to send a JSON containing our wappsto account credentials via POST. This could be done using "curl", but unfortunately the API doesn't specify the exact format of the request, so we reached out to the wappsto support team. They were so cooperative, and they answered a lot of questions we had.

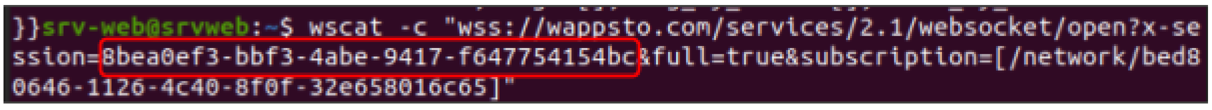
After all this time, we successfully made a connection with our wappsto:bit. The breakdown of it is the following:

- First, we send the following JSON containing our credentials using "curl":



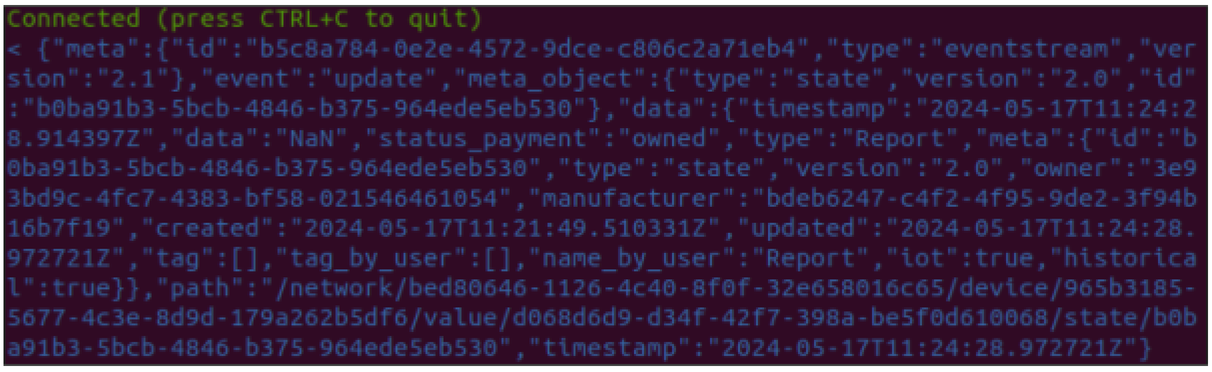
```
}srv-web@srvweb:~$ curl 'https://wappsto.com/services/2.1/session'   -H 'Content
-Type: application/json'   --data-raw $'{"username":"enricnavarro00@gmail.com","
password":"Avemaria10","remember_me":false}'
```

- Then, since we got the session ID that we needed, we open a websocket with wappsto to receive the data flow generated by our micro:bit code:



```
}}srv-web@srvweb:~$ wscat -c "wss://wappsto.com/services/2.1/websocket/open?x-se
ssion=8bea0ef3-bbf3-4abe-9417-f647754154bc&full=true&subscription=[/network/bed8
0646-1126-4c40-8f0f-32e658016c65]"
```

- The outcome of this is a continuos data flow in JSON format.



```
Connected (press CTRL+C to quit)
< {"meta":{"id":"b5c8a784-0e2e-4572-9dce-c806c2a71eb4","type":"eventstream","ver
sion":"2.1"},"event":"update","meta_object":{"type":"state","version":"2.0","id"
:"b0ba91b3-5bcb-4846-b375-964ede5eb530"},"data":{"timestamp":"2024-05-17T11:24:2
8.914397Z","data":"NaN","status_payment":"owned","type":"Report","meta":{"id":"b
0ba91b3-5bcb-4846-b375-964ede5eb530","type":"state","version":"2.0","owner":"3e9
3bd9c-4fc7-4383-bf58-021546461054","manufacturer":"bdeb6247-c4f2-4f95-9de2-3f94b
16b7f19","created":"2024-05-17T11:21:49.510331Z","updated":"2024-05-17T11:24:28.
972721Z","tag":[],"tag_by_user":[],"name_by_user":"Report","iot":true,"historica
l":true}},"path":"/network/bed80646-1126-4c40-8f0f-32e658016c65/device/965b3185-
5677-4c3e-8d9d-179a262b5df6/value/d068d6d9-d34f-42f7-398a-be5f0d610068/state/b0b
a91b3-5bcb-4846-b375-964ede5eb530","timestamp":"2024-05-17T11:24:28.972721Z"}
```

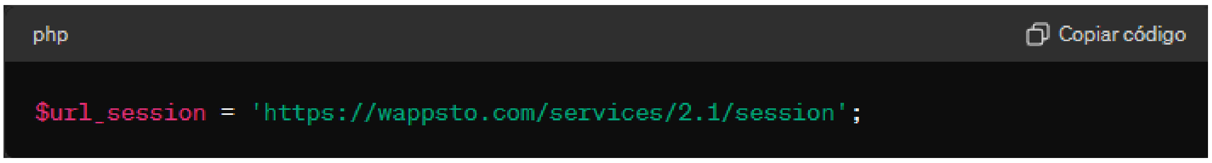
This is huge for us, finally we were able to overcome this challenge. But, having this flow of data in the terminal is not very useful, at least not in a way we can think of. So, the plan is to try to open the WebSocket in a code environment to be able to store the data in variables and use it. This will be done by creating our own *web page*.

4.4. *The website's back-end development*

The journey to develop this website is a long one, so stick around because we'll be doing a full deep dive, meticulously explaining each line of code, every version of the code and all the challenges we faced.

At first, we thought about using PHP, being a scripting language for web development that can be embedded into HTML, it sounded like a great idea.

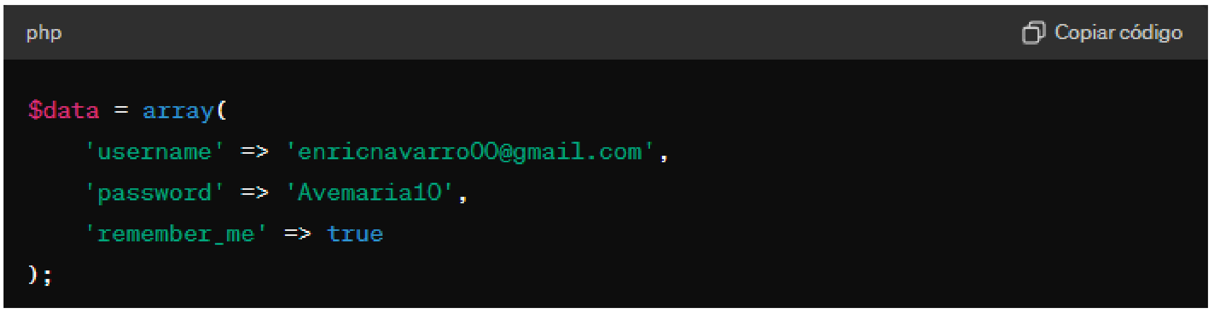
- ***URL Setup:***



```php
$url_session = 'https://wappsto.com/services/2.1/session';
```

This line defines the URL to which the script will send a request to establish a session.

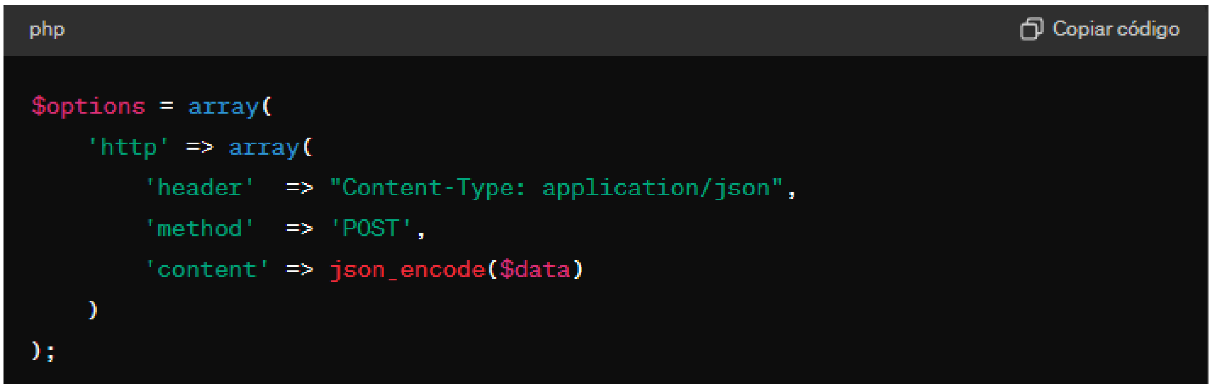
- ***Data Array:***



```php
$data = array(
    'username' => 'enricnavarro00@gmail.com',
    'password' => 'Avemaria10',
    'remember_me' => true
);
```

Here, an array named "$data" is created, containing the username, password, and a flag for remembering the user's session.

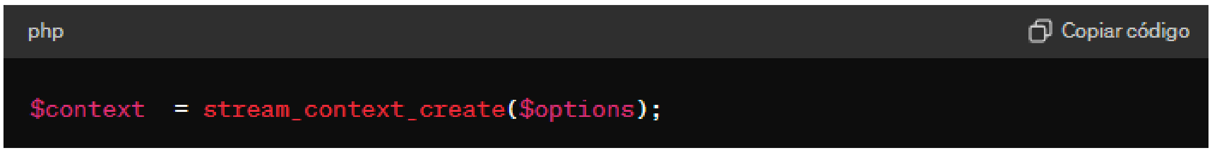
- ***HTTP Options:***



```php
$options = array(
    'http' => array(
        'header'  => "Content-Type: application/json",
        'method'  => 'POST',
        'content' => json_encode($data)
    )
);
```

This section prepares the options for the HTTP request. It specifies that the request will be a POST method with a JSON content type and includes the encoded data.

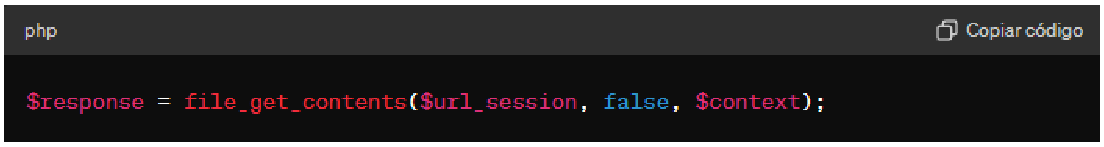
- ***Create Context:***



```php
$context  = stream_context_create($options);
```

Here, the context for the HTTP request is created using the options defined earlier.

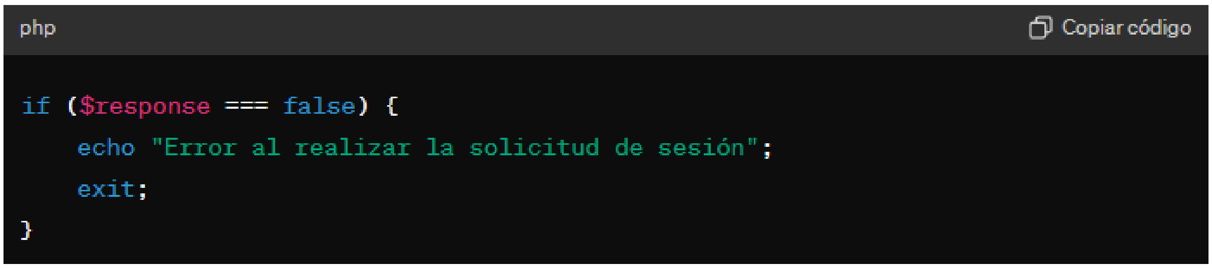
- ***Send Request:***



```php
$response = file_get_contents($url_session, false, $context);
```

This line sends the HTTP request to the specified URL ($url_session) using the context created earlier, and it stores the response in the $response variable.

- ***Error Handling:***



```php
if ($response === false) {
    echo "Error al realizar la solicitud de sesión";
    exit;
}
```

This part checks if the response is false (indicating an error in the request). If so, it prints an error message and terminates the script.

- ***Decode Response:***

```php
$response_data = json_decode($response, true);
```

Here, the JSON response is decoded into an associative array ($response_data).

- ***Response Validation:***

```php
if ($response_data === null || !isset($response_data['meta']['id'])) {
    echo "Error: No se pudo obtener el sessionId de la respuesta JSON";
    exit;
}
```

This section verifies whether the decoded JSON response contains the required data. If not, it prints an error message and terminates the script.

- ***Extract Session ID:***

```php
$session_id = $response_data['meta']['id'];
```

This line extracts the session ID from the decoded JSON response. It accesses the id element within the meta array of $response_data

- ***Device ID Setup:***

```php
$device_id = "bed80646-1126-4c40-8f0f-32e658016c65";
```

Here, our micro:bit ID is specified.

- ***WebSocket URL Construction:***

```php
$url_websocket = "wss://wappsto.com/services/2.1/websocket/open?x-session=$session_
```

This line constructs a WebSocket URL using the session ID and device ID previously obtained.

- ***Output URL:***

```php
echo $url_websocket;
```

Finally, the constructed WebSocket URL is echoed out, for further use in the application.

To put it in a few words, The PHP code segment establishes a session with wappsto, retrieves a session ID using the credentials of our account and wappsto:bit, and constructs a WebSocket URL for real-time data communication. This is the backend of the page, the frontend will be made with JavaScript. Next we'll be doing the same, giving a block by block explanation of the code. Take into account that this code will be inside the main HTML file, so we won't have to deal with multiple files at once.

This JavaScript code is meant to be executed when the window has finished loading. Let's break it down step by step:

- ***Window Onload Event:***

```javascript
window.onload = function () {
    // Code to execute when the window has loaded
};
```

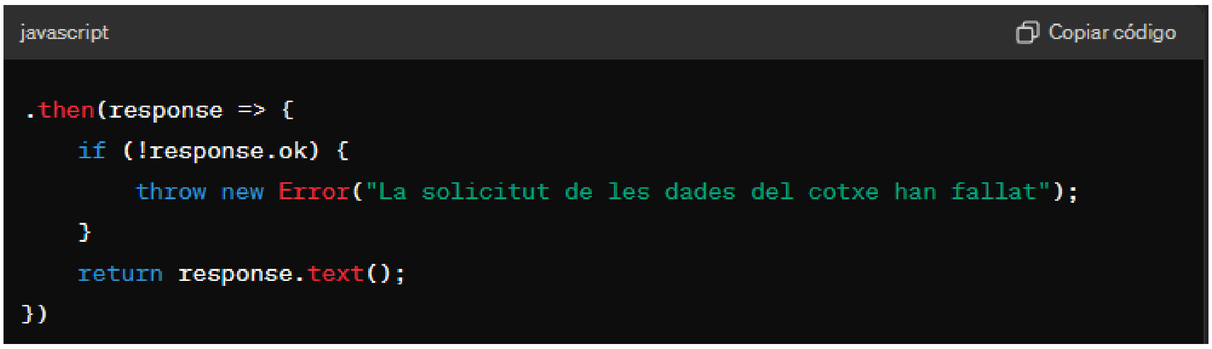
This sets up an event handler that triggers when the window has finished loading.

- ***Fetch Backend Data:***

```javascript
fetch("backend.php")
```

This initiates a fetch request to the "backend.php" file, to retrieve the Session ID.

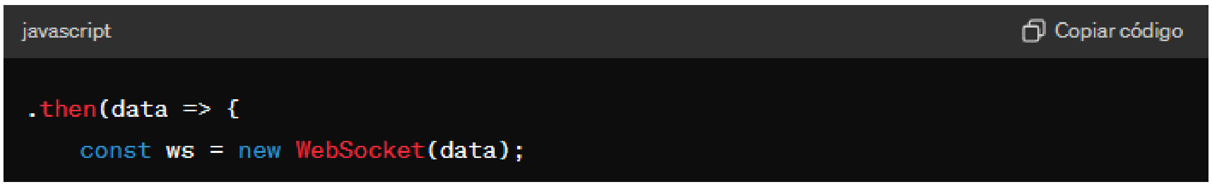
- ***Handling Response:***



```javascript
.then(response => {
    if (!response.ok) {
        throw new Error("La solicitut de les dades del cotxe han fallat");
    }
    return response.text();
})
```

This section checks if the fetch response is successful. If it's not, it throws an error. If successful, it converts the response to text format.

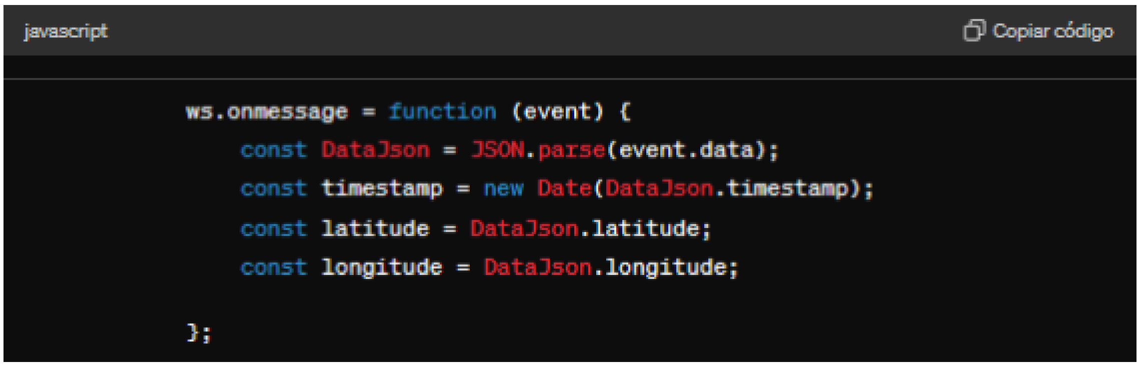
- ***WebSocket Connection:***



```javascript
.then(data => {
    const ws = new WebSocket(data);
```

Upon receiving the response from the fetch request, it creates a new WebSocket connection using the retrieved data (presumably a WebSocket URL).

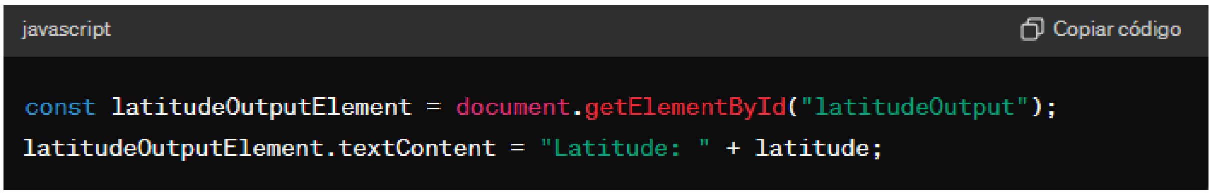
- ***WebSocket Message Handler:***



```javascript
ws.onmessage = function (event) {
    const DataJson = JSON.parse(event.data);
    const timestamp = new Date(DataJson.timestamp);
    const latitude = DataJson.latitude;
    const longitude = DataJson.longitude;

};
```

This sets up a handler for WebSocket messages. When a message is received, it parses the JSON data and extracts information like timestamp, latitude, and longitude.

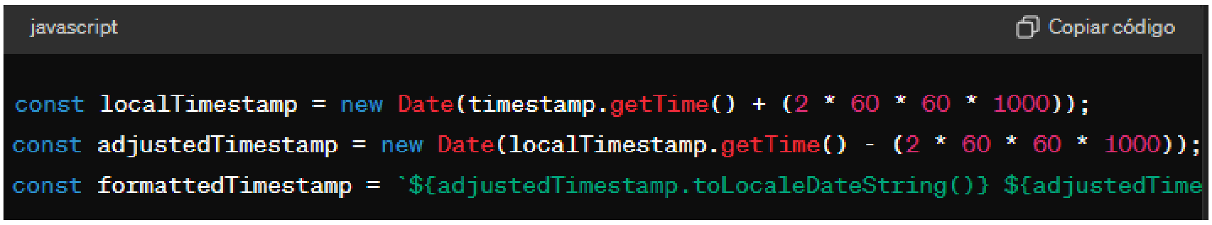
- ***Updating HTML Elements:***



```javascript
const latitudeOutputElement = document.getElementById("latitudeOutput");
latitudeOutputElement.textContent = "Latitude: " + latitude;
```

These lines update specific HTML elements with the latitude and longitude data received from the WebSocket.

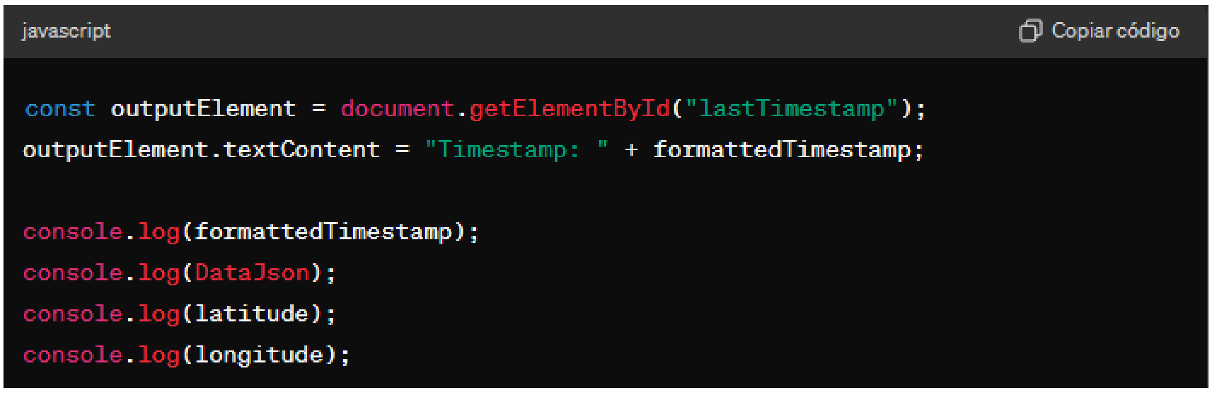
- ***Timestamp Adjustment:***



```javascript
const localTimestamp = new Date(timestamp.getTime() + (2 * 60 * 60 * 1000));
const adjustedTimestamp = new Date(localTimestamp.getTime() - (2 * 60 * 60 * 1000));
const formattedTimestamp = `${adjustedTimestamp.toLocaleDateString()} ${adjustedTime
```

This section adjusts the timestamp received from the WebSocket to the local time and formats it.

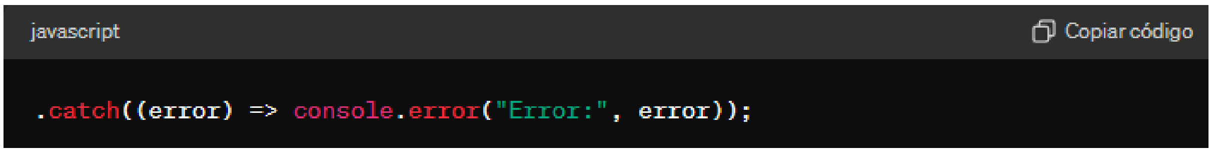
- ***Displaying Timestamp and Data:***



```javascript
const outputElement = document.getElementById("lastTimestamp");
outputElement.textContent = "Timestamp: " + formattedTimestamp;

console.log(formattedTimestamp);
console.log(DataJson);
console.log(latitude);
console.log(longitude);
```

These lines display the formatted timestamp on the webpage and log various pieces of data to the console for debugging purposes.

- ***Error handling:***



```javascript
 .catch((error) => console.error("Error:", error));
```

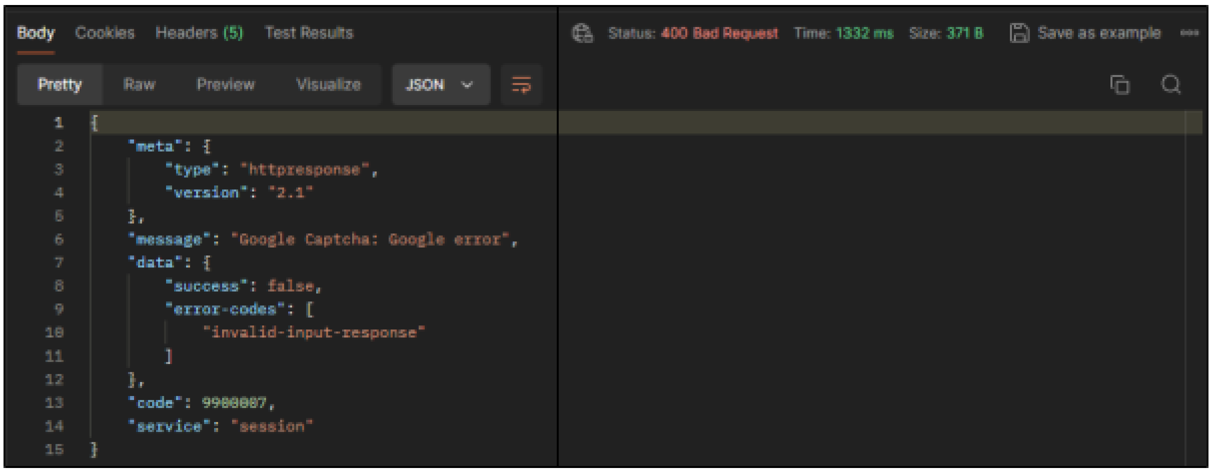
This catches any errors that occur during the fetch request or WebSocket connection and logs them to the console.

This code as a whole executes when the webpage loads. It fetches data from the backend PHP script, then establishes a WebSocket connection using the received URL. This WebSocket connection allows for real-time communication with wappsto to retrieve and display live data, such as the wappsto:bit location and timestamp. The JavaScript code updates HTML elements with the received data and handles errors gracefully, ensuring smooth operation of the real-time data display on the webpage.

These two codes combined make the alpha build of our website. It does a fairly simple job: Requesting a session ID from the wappsto API, using it to pen a websocket and displaying the coordinates of the wappsto:bit. At least that is what it did until we hit the next bump in our progress: *Bad Request 400.*

The thing about this error is that it's pretty generic. It is a status code that indicates that the server cannot process the request because it is incorrect or corrupt. This can happen for various reasons, such as: Incorrectly typed URL, corrupted files in the browser cache and even problems with expired or corrupted browser cookies. We double-checked the URLs and reset the browser cookies, but nothing changed. Here comes another tool discovery: *Postman*, which is a popular API platform used by developers to build, test, and collaborate on APIs. It simplifies each step of the API lifecycle

This tool allows us to make POST and GET request and get the responses in a very visual and comprehensive way. So we can see the exact response we are getting from the wappsto server.

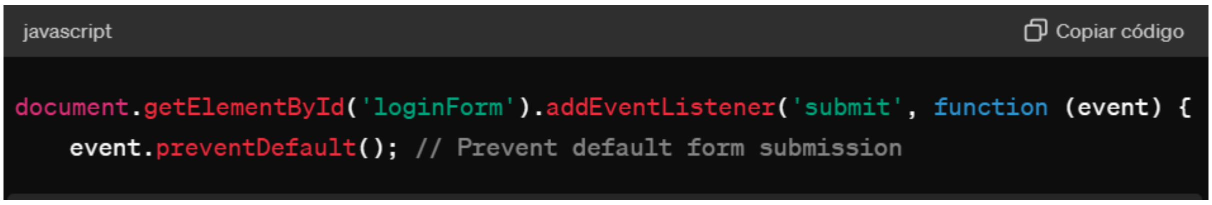


```
{
    "meta": {
        "type": "httpresponse",
        "version": "2.1"
    },
    "message": "Google Captcha: Google error",
    "data": {
        "success": false,
        "error-codes": [
            "invalid-input-response"
        ]
    },
    "code": 9900007,
    "service": "session"
}
```

Turns out it's a "Google captcha" error for "Failed authentication". For further clarification, we reached out again to wappsto support, and it seems that this error happens when logging in with invalid credentials for 3 or more times. The account gets suspended until you log in to the wappsto official page. A pretty simple problem that costed us two work days.

The next step was to migrate the PHP script to JavaScript, this way we'll be dealing with one programming language instead of two. And we thought about making it handles the submission of a login form by making an asynchronous request to wappsto, obtaining the session ID, and then redirecting the user upon successful login. More on the redirection destination later. Here's a breakdown of its functionality:

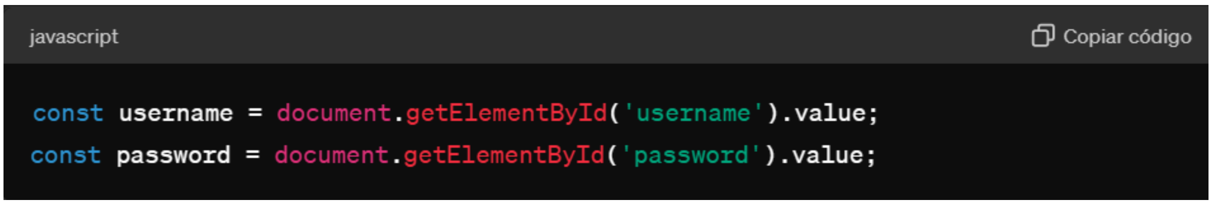
- ***Form Submission Event Listener***



```javascript
document.getElementById('loginForm').addEventListener('submit', function (event) {
    event.preventDefault(); // Prevent default form submission
```

This line sets up an event listener for the form with the ID “loginForm”, preventing the default form submission behaviour.

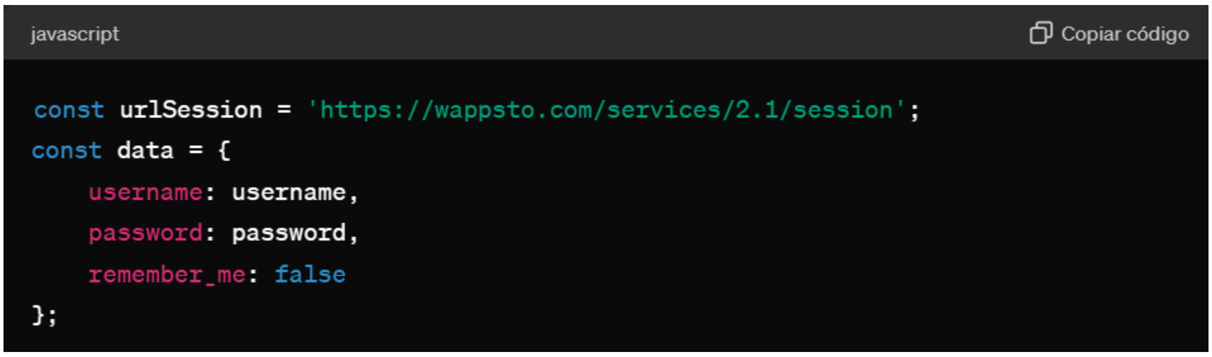
- ***Retrieve Form Values:***



```javascript
const username = document.getElementById('username').value;
const password = document.getElementById('password').value;
```

These lines get the values entered by the user in the username and password fields.

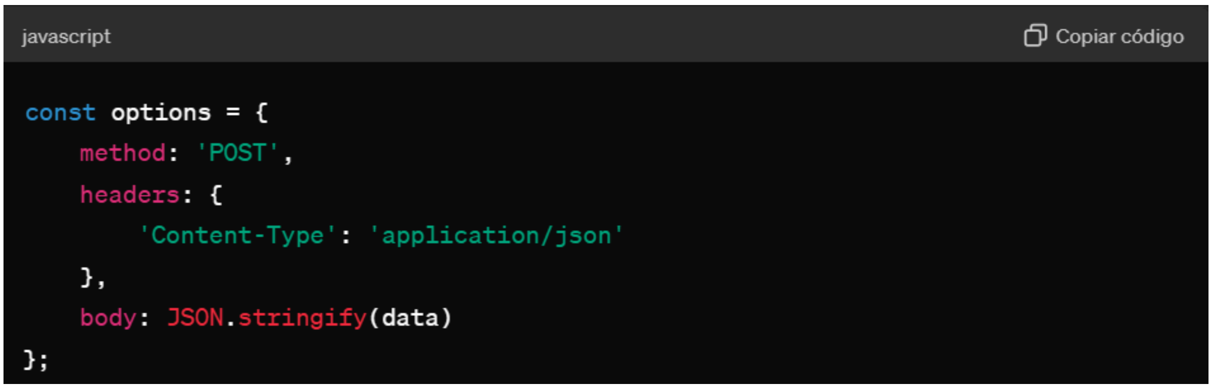
- ***Define Session URL and Data:***



```javascript
const urlSession = 'https://wappsto.com/services/2.1/session';
const data = {
    username: username,
    password: password,
    remember_me: false
};
```

This sets the URL for the session request and creates a data object containing the username, password, and a flag to indicate whether the session should be remembered.

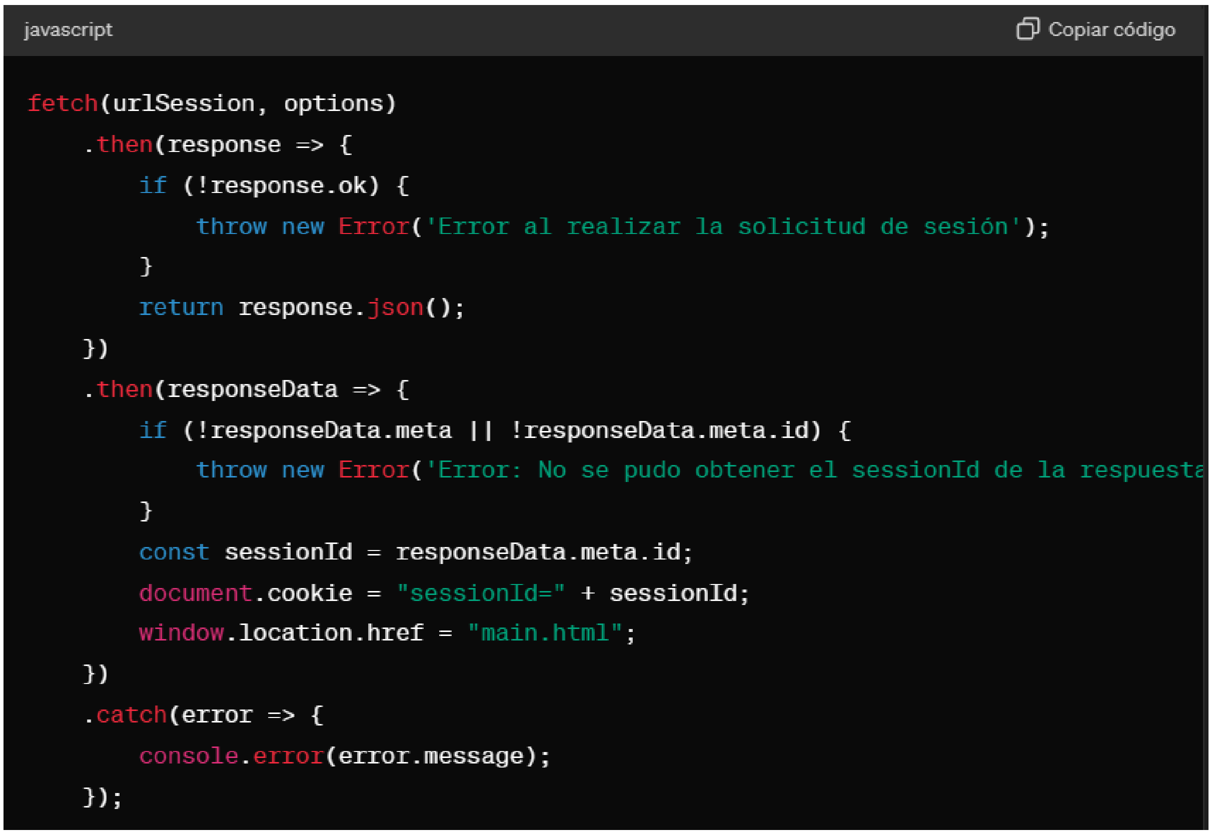
- ***Fetch Options:***



```javascript
const options = {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(data)
};
```

Here, the options for the fetch request are defined, including the HTTP method (POST), headers (indicating JSON content), and the request body (stringified data object).

- Fetch Request:

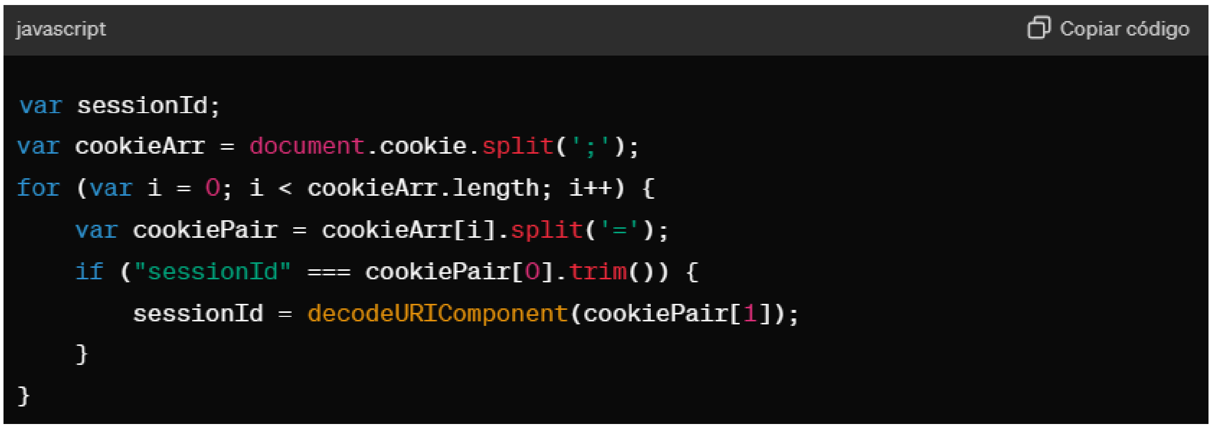


```javascript
fetch(urlSession, options)
    .then(response => {
        if (!response.ok) {
            throw new Error('Error al realizar la solicitud de sesión');
        }
        return response.json();
    })
    .then(responseData => {
        if (!responseData.meta || !responseData.meta.id) {
            throw new Error('Error: No se pudo obtener el sessionId de la respuesta
        }
        const sessionId = responseData.meta.id;
        document.cookie = "sessionId=" + sessionId;
        window.location.href = "main.html";
    })
    .catch(error => {
        console.error(error.message);
    });
```

This sequence sends the fetch request with the specified options to the session URL. It handles the response by checking if it's OK and then parsing the JSON data. If the response data includes a valid session ID, it sets this session ID in a *cookie* and redirects the user to "main.html". If any errors occur, they are caught and logged to the console.

The cookie part is very important, before we settled with this code we found our selves before a new boulder. This time the issue is that we'll have two HTML files, one for the login form and the other one for displaying the data. But in order for the login form to work, it has to process the websocket and session ID request within it and then do redirection to the other page, doing that shuts down the websocket connection. To be more specific, it just doesn't pass the connection to the other page. So we had to find a way to pass out the necessary information to the other page in order to establish the connection again.

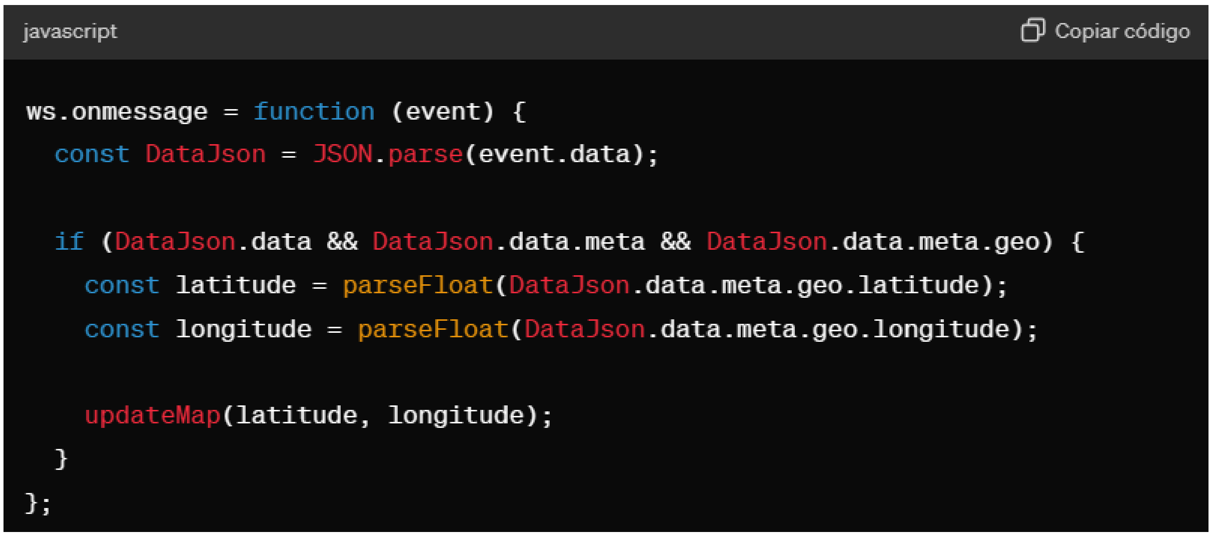
After another grind, and the help of third-party members, we figured out the cookie method. Cookies are small pieces of data that are stored on the user's device by their web browser. They are used to remember information about the user, such as login sessions, preferences, and other data that should persist across different pages or visits. The cookie mechanism in this code is used to store the session ID provided by the server. In the other page we handle the cookie like this:



```javascript
var sessionId;
var cookieArr = document.cookie.split(';');
for (var i = 0; i < cookieArr.length; i++) {
    var cookiePair = cookieArr[i].split('=');
    if ("sessionId" === cookiePair[0].trim()) {
        sessionId = decodeURIComponent(cookiePair[1]);
    }
}
```

We retrieve the session ID from the browser's cookies to establish a WebSocket connection for real-time updates. We declare a sessionId variable, split the document.cookie string into an array of individual cookies, and iterates through this array. Each cookie is split into its name and value, and the code checks if the name matches "sessionId". If a match is found, the value is decoded using decodeURIComponent and assigned to sessionId.

Once it's done, we can construct the WebSocket URL. As we stated earlier, the only data we need are the longitude and latitude coordinates. Using this code we establish the websocket, check if the data is available, and then we store it in two variables.



```javascript
ws.onmessage = function (event) {
  const DataJson = JSON.parse(event.data);

  if (DataJson.data && DataJson.data.meta && DataJson.data.meta.geo) {
    const latitude = parseFloat(DataJson.data.meta.geo.latitude);
    const longitude = parseFloat(DataJson.data.meta.geo.longitude);

    updateMap(latitude, longitude);
  }
};
```

As you can see, we call an undefined function "updateMap()", at least for now. This is the second part of our website: *The data manipulation and display*.

4.5. The website's Front-end development

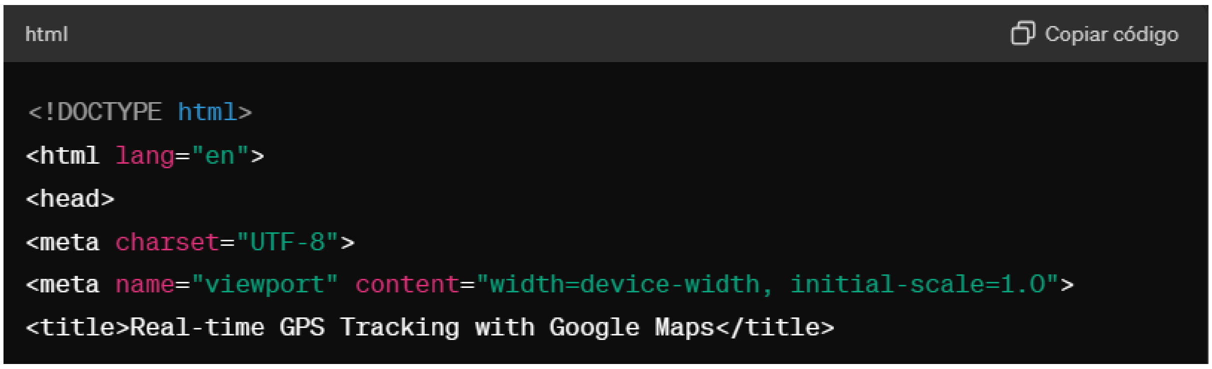
This project, being a geolocating device, having a map somewhere in the content of is a given. So this is exactly what we are doing next. The search for an API that provides us with the tool we need began.

Our first option was *Mapbox,* Mapbox is a comprehensive mapping and location platform that provides a wide array of tools for developers to integrate maps and location-based services into their applications. One of its primary features is the ability to create highly customizable maps. Developers can choose from a variety of map styles and themes, tailoring them to fit the aesthetic and functional requirements of their applications. The use of vector tiles allows for smooth, high-performance rendering of maps, which can be dynamically styled on the client side, ensuring a responsive and engaging user experience.

So we tried implementing it, first we made an account, pitched our API Token and started writing the code. The mapbox API's documentation was fairly simple and comprehensive, something we can't say about wappsto's. We were able to render a world map in our website and display our location, emphasis on our location, this was a huge let-down. The thing is, map box had no option to track a device that is not the same device running the code. We even tried to trick into thinking the micro:bit is the main device with brute force, but it seems off limit. At this point, our only option was to step aside from trying to use a not so known programme, which is Mapbox, and go with a main stream alternative that we are sure you thought about: *Google Maps*.

Fortunately, this was the last jig saw piece we needed to complete the project. So next we will do a break-down of our definitive website, then we will talk about some changes we did since we implemented Google Maps API. The file is an HTML file that also contains some JavaScript code.

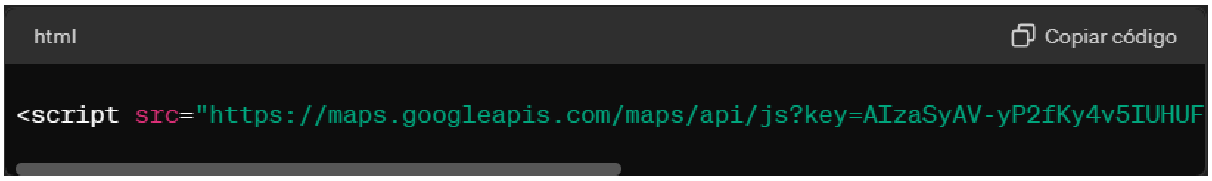
- ***Basic HTML Document Setup:***



```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Real-time GPS Tracking with Google Maps</title>
```

Sets the document type, language, character encoding, viewport settings, and title.

- ***Google Maps API:***



```html
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAV-yP2fKy4v5IUHUF
```

Asynchronously loads the Google Maps JavaScript API with necessary libraries (geometry and places) and specifies a callback function (initMap) to initialize the map.

- ***Map Container:***

```html
<style>
  #map {
    height: 100vh;
    width: 100%;
  }
</style>
</head>
<body>
<div id="map"></div>
```

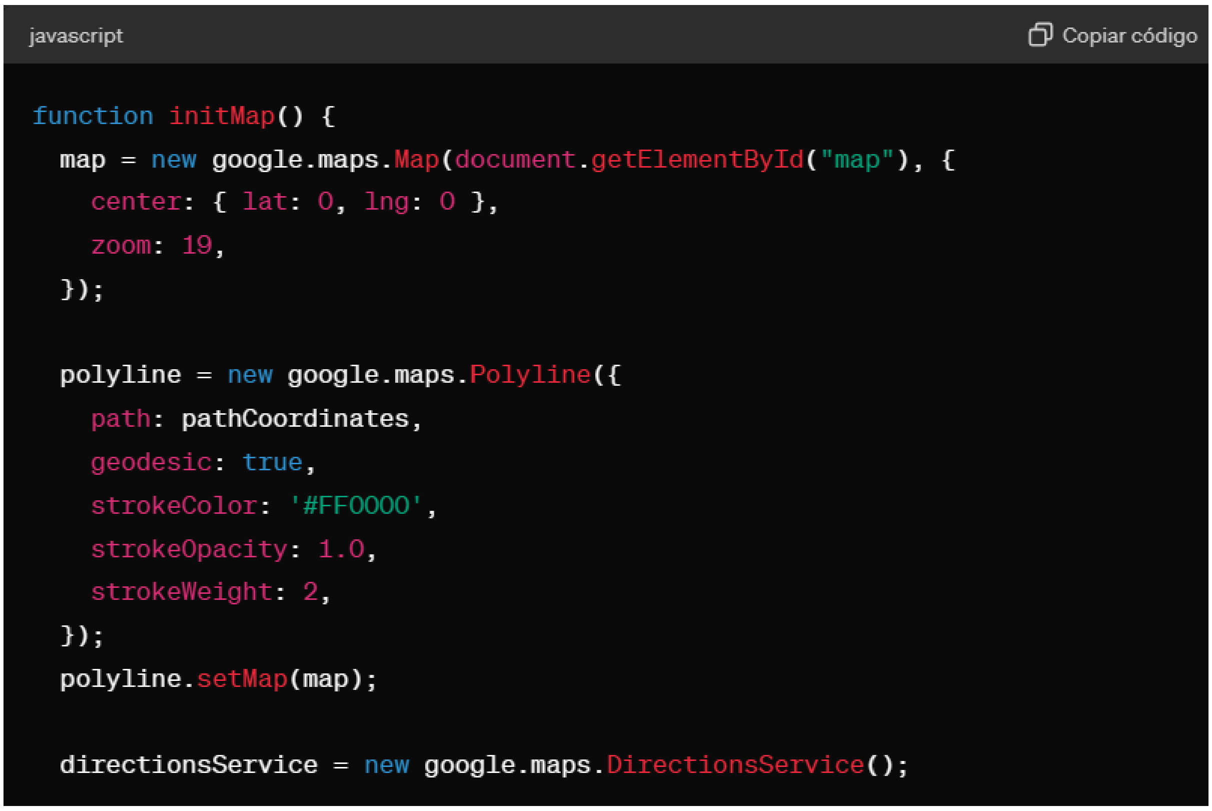
Defines a container for the map with full viewport height and width.

- ***Global Variables:***

```javascript
let map;
let pathCoordinates = [];
let polyline;
let lastPosition = null;
let directionsService;
let marker = null;
let thresholdDistance = 10; // 10 meters threshold distance
```

Initializes global variables for the map, coordinates, polyline, last position, directions service, marker, and a threshold distance for location updates.

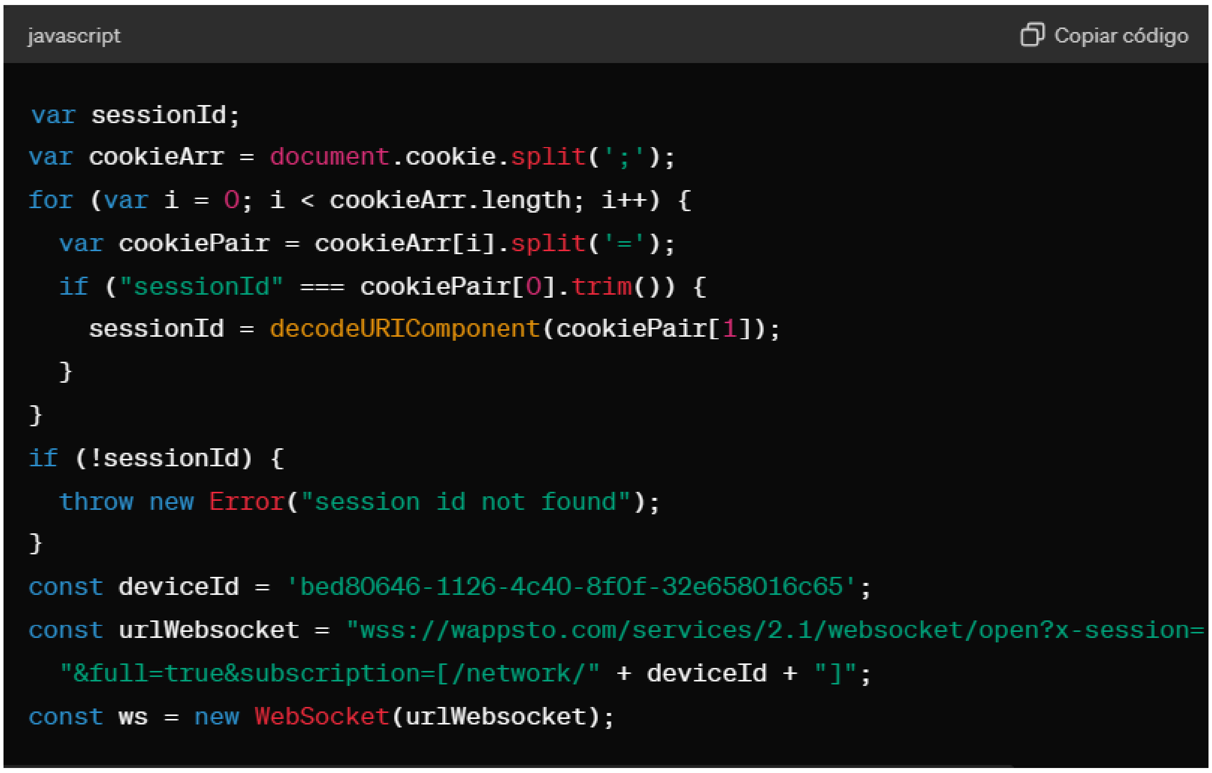
- ***Map Initialization:***



```javascript
function initMap() {
  map = new google.maps.Map(document.getElementById("map"), {
    center: { lat: 0, lng: 0 },
    zoom: 19,
  });

  polyline = new google.maps.Polyline({
    path: pathCoordinates,
    geodesic: true,
    strokeColor: '#FF0000',
    strokeOpacity: 1.0,
    strokeWeight: 2,
  });
  polyline.setMap(map);

  directionsService = new google.maps.DirectionsService();
```

Initializes the Google Map centred at coordinates (0,0) with a zoom level of 19. A polyline is created to display the path, and the directions service is initialized.

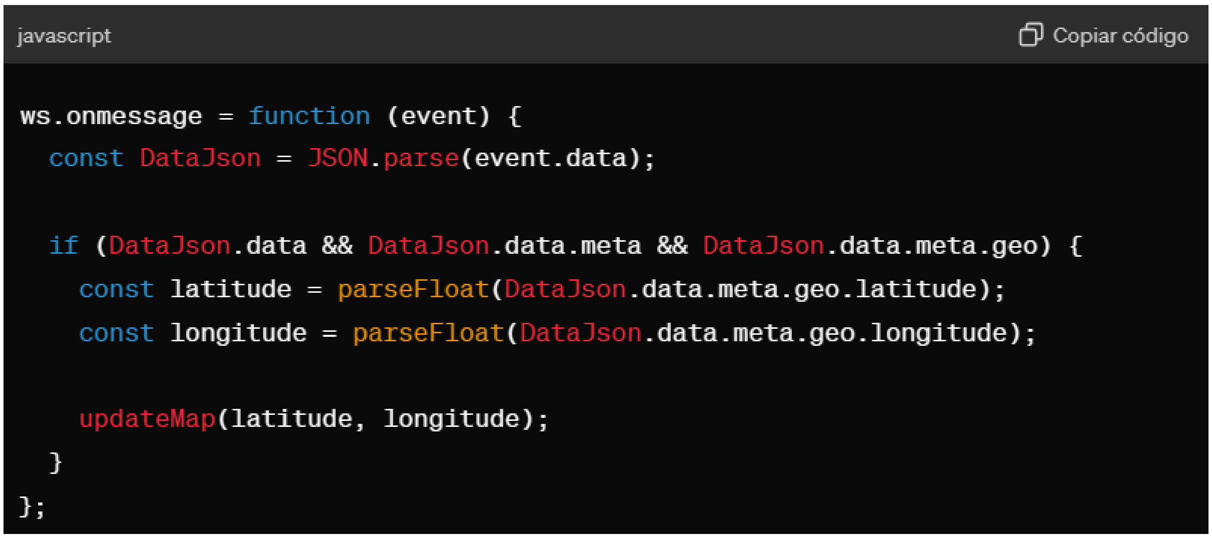
- ***Websocket Setup:***



```javascript
var sessionId;
var cookieArr = document.cookie.split(';');
for (var i = 0; i < cookieArr.length; i++) {
  var cookiePair = cookieArr[i].split('=');
  if ("sessionId" === cookiePair[0].trim()) {
    sessionId = decodeURIComponent(cookiePair[1]);
  }
}
if (!sessionId) {
  throw new Error("session id not found");
}
const deviceId = 'bed80646-1126-4c40-8f0f-32e658016c65';
const urlWebsocket = "wss://wappsto.com/services/2.1/websocket/open?x-session=
  "&full=true&subscription=[/network/" + deviceId + "]";
const ws = new WebSocket(urlWebsocket);
```

Retrieves the session ID from the cookies and constructs a WebSocket URL for real-time updates. Opens a WebSocket connection.

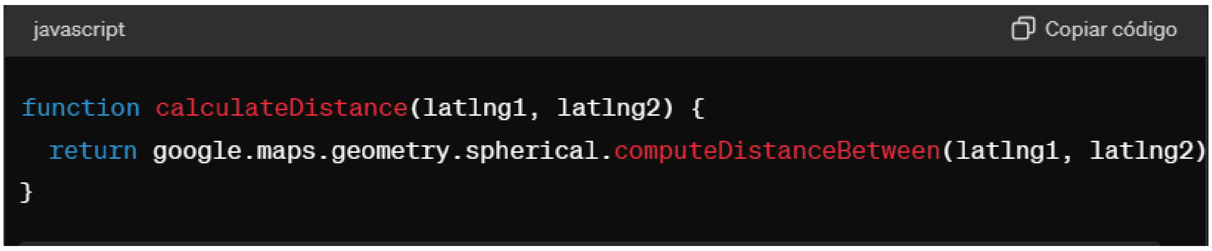
- ***Handling Websocket Messages:***



```javascript
ws.onmessage = function (event) {
  const DataJson = JSON.parse(event.data);

  if (DataJson.data && DataJson.data.meta && DataJson.data.meta.geo) {
    const latitude = parseFloat(DataJson.data.meta.geo.latitude);
    const longitude = parseFloat(DataJson.data.meta.geo.longitude);

    updateMap(latitude, longitude);
  }
};
```

Defines a function to handle incoming WebSocket messages, extracting latitude and longitude from the received data, and calls updateMap with these coordinates.

- ***Distance Calculation Function:***

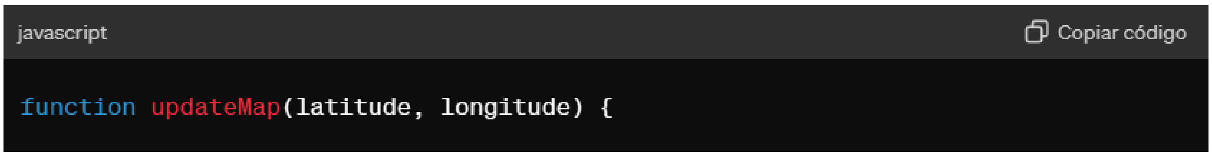


```javascript
function calculateDistance(latlng1, latlng2) {
  return google.maps.geometry.spherical.computeDistanceBetween(latlng1, latlng2)
}
```

Utilizes the Google Maps geometry library to calculate the distance between two LatLng points.

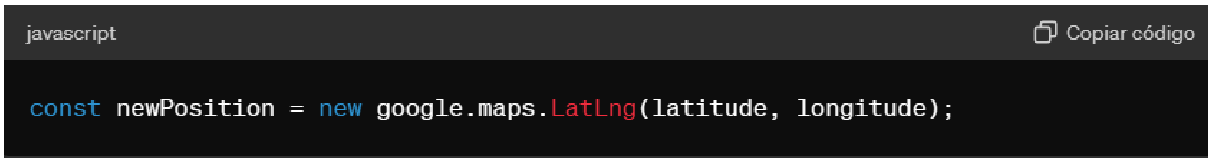
- ***Map Update Function:***

The updateMap function is a crucial part of the real-time GPS tracking application using Google Maps. It updates the map with new coordinates received from a WebSocket connection, manages the drawing of the path, and calculates distances to ensure accurate tracking. Let's break down this function step-by-step.



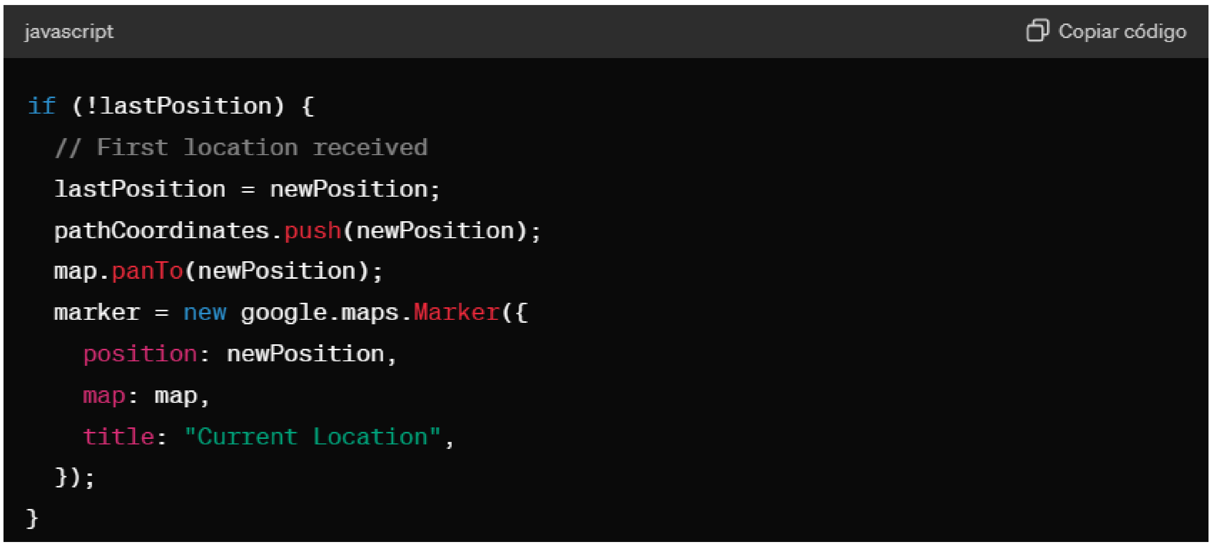
```javascript
function updateMap(latitude, longitude) {
```

The function updateMap takes two parameters: latitude and longitude. These represent the new geographic coordinates received for the current position.



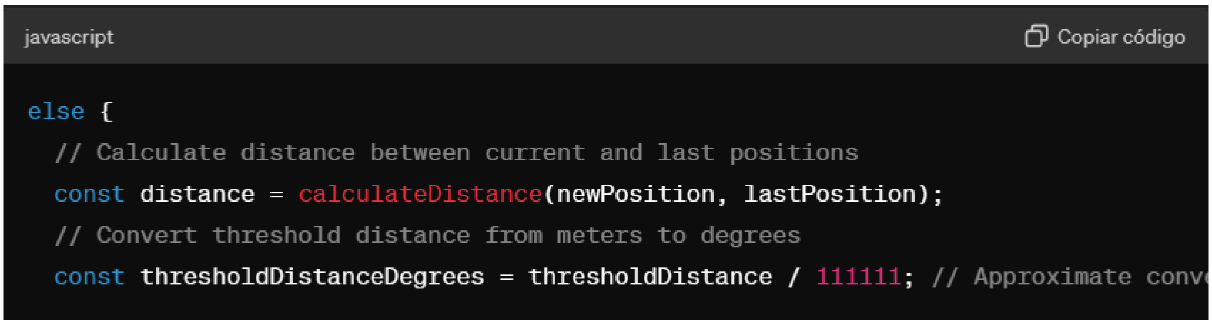
```javascript
const newPosition = new google.maps.LatLng(latitude, longitude);
```

newPosition is a new google.maps.LatLng object created with the provided latitude and longitude. This object represents the new GPS position on the map.



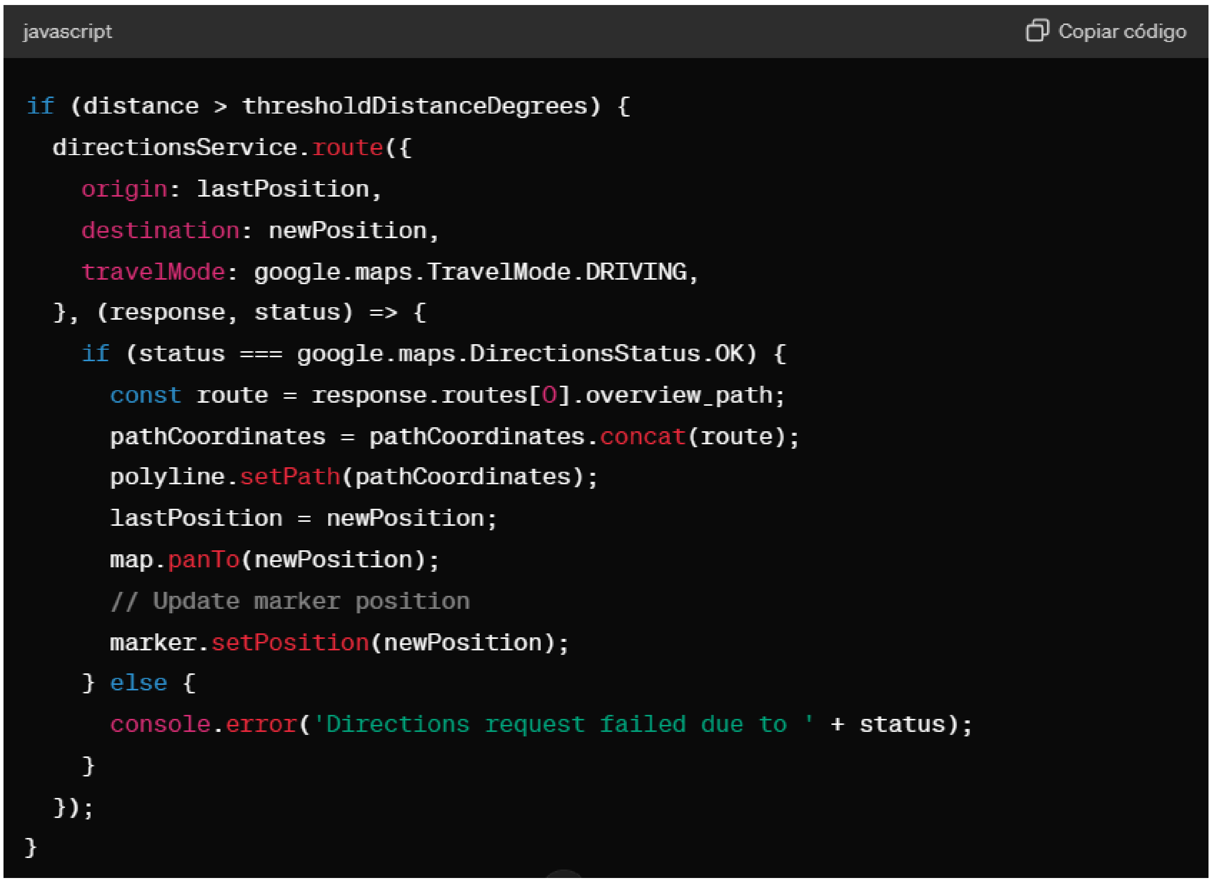
```javascript
if (!lastPosition) {
  // First location received
  lastPosition = newPosition;
  pathCoordinates.push(newPosition);
  map.panTo(newPosition);
  marker = new google.maps.Marker({
    position: newPosition,
    map: map,
    title: "Current Location",
  });
}
```

- If lastPosition is null (indicating that no previous position has been set), the function handles this as the first GPS position received:
    - It assigns newPosition to lastPosition.
    - Adds newPosition to the pathCoordinates array, which stores the sequence of positions to be drawn on the map.
    - Pans the map to center on newPosition.
    - Creates a new marker at newPosition to represent the current location on the map.



```javascript
else {
  // Calculate distance between current and last positions
  const distance = calculateDistance(newPosition, lastPosition);
  // Convert threshold distance from meters to degrees
  const thresholdDistanceDegrees = thresholdDistance / 111111; // Approximate conv
```

- If lastPosition is already set, the function handles updates for subsequent positions:
    - It calculates the distance between newPosition and lastPosition using the calculateDistance function.
    - Converts the threshold distance from meters to degrees for comparison. The value 111111 is an approximate conversion factor from meters to degrees of latitude.

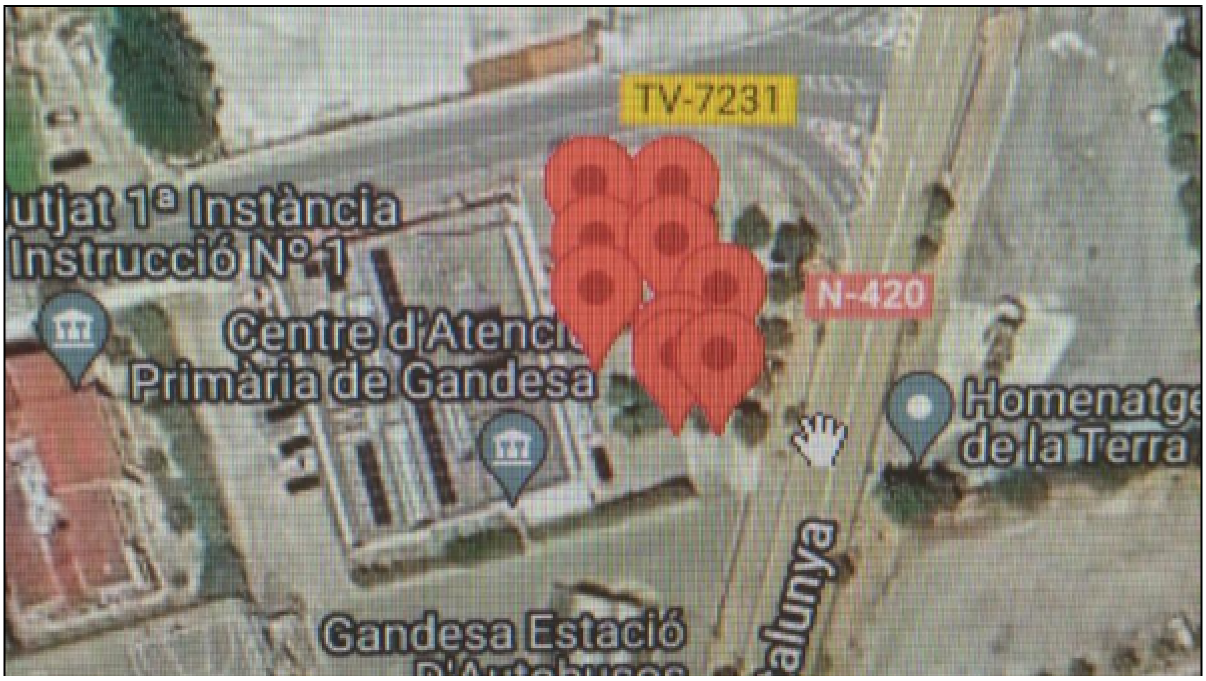


```javascript
if (distance > thresholdDistanceDegrees) {
  directionsService.route({
    origin: lastPosition,
    destination: newPosition,
    travelMode: google.maps.TravelMode.DRIVING,
  }, (response, status) => {
    if (status === google.maps.DirectionsStatus.OK) {
      const route = response.routes[0].overview_path;
      pathCoordinates = pathCoordinates.concat(route);
      polyline.setPath(pathCoordinates);
      lastPosition = newPosition;
      map.panTo(newPosition);
      // Update marker position
      marker.setPosition(newPosition);
    } else {
      console.error('Directions request failed due to ' + status);
    }
  });
}
```

- If the distance between newPosition and lastPosition is greater than the threshold distance:
    - A route request is sent to the directionsService with lastPosition as the origin and newPosition as the destination, using driving mode.
    - If the request is successful (status === google.maps.DirectionsStatus.OK):
        - The route's overview path is extracted from the response.
        - pathCoordinates is updated to include the new route segment.
        - The polyline on the map is updated to reflect the new path.
        - lastPosition is updated to newPosition.
        - The map is panned to the new position.
        - The marker's position is updated to the new position.
    - If the request fails, an error is logged to the console.

The updateMap function efficiently updates the map with new GPS coordinates by managing the path and marker on the map. It handles both the initial positioning and subsequent updates by calculating distances and using Google Maps services to dynamically draw the route. This ensures accurate real-time tracking on the map.

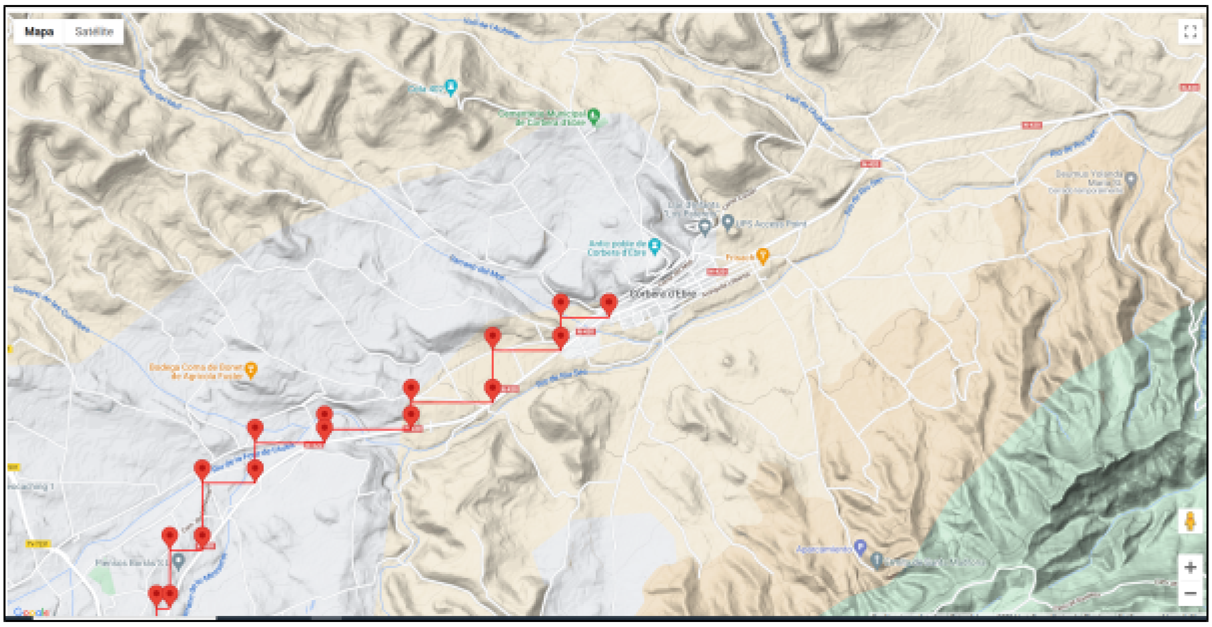
Before we settled down with this version, we ran into several versions, mainly because the display of the track wasn’t as we draw it in our head. The first one was the off-road tracking:



As this geolocating device is made thinking about autonomous cars, we think that tracking the device when it is off-road is unnecessary. The way we did it was by imploementing the Google Maps directions and rout API, this allows us to only draw the polyline on streets (on-rout). The way this method works is not so direct, because now the polyline is more of a rout rather than a tracking line of historical location.

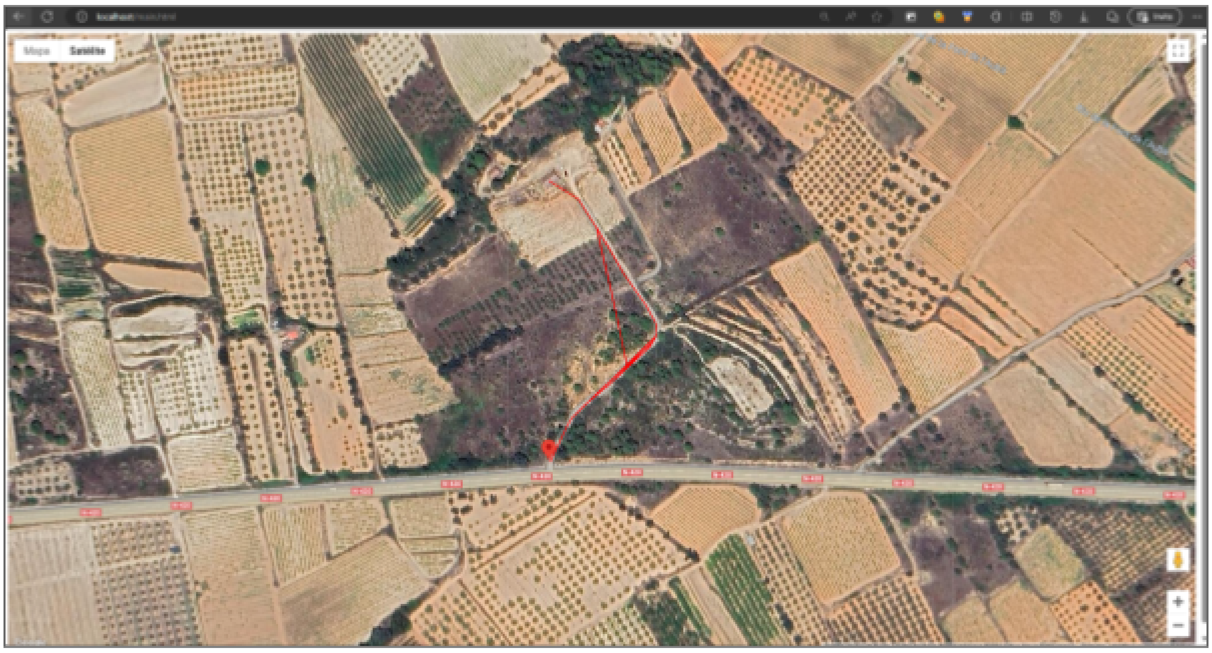
The side effect of doing it this way made it so a rout to the current location of the device is displayed, even if the change in location was minimal. So we implemented a segment of code that calculates the distance between every instance of location change, and checks if it surpasses a specific limit (10 meters). If true, the code doesn't sraw a route and waits for another instance.

Another hiccup was the fact that the code made it so it displays a marker every time the coordinates are changed. As stated before, we made a segment of code that put a marker only at the last position of the device, A.K.A. its current location.



As you can observe, the other issue was the order in which the two variables longitude and latitude are registered. It displays the change in the longitude coordinates, and then it displays the latitude change, instead of using the two as one. Luckily, the change that we did from drawing a polyline between the location to display the directions from point B to point A solved the problem.

As a result of this, we can get outputs as this one:



And with this final edit, we got the project finished and ready to work in some real life products.

Conclusion

The project successfully demonstrates the feasibility and practicality of using micro:bit and Wappsto:bit for real-time geolocation tracking via a web application. Despite initial challenges with Wappsto's native tools and API, we managed to establish a continuous data flow from the micro:bit to our custom web server using WebSockets. This real-time data was then visualized on a map, providing a clear and interactive way to monitor the device's location. The experience underscored the importance of 5G in enhancing IoT applications, offering improved data transmission speeds and reliability. The project also highlighted the need for robust and flexible API interactions to ensure seamless communication between IoT devices and web applications. Future improvements could focus on refining the API interactions, enhancing the user interface, and expanding the system's capabilities to include more complex autonomous vehicle functions, thereby leveraging the full potential of 5G technology in IoT ecosystems.

Bibliography

Moore, K. (2023, August 11). *From 1G to 5G: The Evolution of Mobile Communications - Mpirical*. Mpirical. https://www.mpirical.com/blog/the-evolution-of-mobile-communication

Chandler, N. (2023, March 8). *How 4G works*. HowStuffWorks. https://electronics.howstuffworks.com/4g.htm

Elisa granted €3.9m by Finnish gov't to roll out virtual power plant. (2023, February 16). DCD. https://www.datacenterdynamics.com/en/news/elisa-granted-39m-by-finnish-govt-to-roll-out-virtual-power-plant/

Wärtsilä Expert Insight. (n.d.). [Video]. Wartsila.com. https://www.wartsila.com/energy/services/lifecycle-solutions/remote-support-and-data-management

Home - OuluHealth. (2022, October 18). OuluHealth. https://ouluhealth.fi/

The 6 levels of vehicle autonomy explained | Synopsys Automotive. (n.d.). https://www.synopsys.com/automotive/autonomous-driving-levels.html

J3259 (WIP) Taxonomy & Definitions for Operational Design Domain (ODD) for Driving Automation Systems - SAE International. (n.d.). https://www.sae.org/standards/content/j3259/

New Mind. (2022, November 2). *The truth about self driving cars* [Video]. YouTube. https://www.youtube.com/watch?v=d5TiaIYdug4

MegaLag. (2021, May 25). *Autonomous Self-Driving Vehicle levels explained* [Video]. YouTube. https://www.youtube.com/watch?v=LIP4eJAECvU

Bloomberg Technology. (2019, August 1). *5G and the Future of Connected Cars* [Video]. YouTube. https://www.youtube.com/watch?v=x6DfzkeQpQ0

GPS tracking data logging device with micro:bit and wappsto:bit. (2022, May 27). OKdo. https://www.okdo.com/project/gps-tracking-data-logging-device-with-microbit-and-wappstobit/