Pol Cirera Jordana

CAN Datalogger for EV Digital twins

Master's Thesis
Supervised by Dr. Enric Vidal Idiarte

Master's Degree in Electric Vehicle Technologies

UNIVERSITAT ROVIRA I VIRGILI

Tarragona
2024

Índex

1 Introduction

1.1 Main goals and objectives

The main goals and objectives of this TFM are the following:

Deep dive on the digital twins definition, history, and actual state of the art.

Develop a CAN datalogger able to provide information for a digital twin.

Analyse the algorithms available for creating a digital twin

1.1.1 A peak of history

The term digital twin is gaining popularity in the last 5 years but the first concept was proposed by Michael Grives in 2002 while talking about PLM (Product Lifecycle Management). The main goal of Grieves was a model with 3 components, real space, virtual space and linking mechanisms for the data/information flow in between both spaces. He refer it as "Mirrored spaces model". Even thought similar approaches were proposed previously, like "Mirror worlds" in 1991 by David Gelernter, Grieves focused on the mechanism for data transfer to be bidirectional and having multiple virtual spaces for a single real space enabling the testing of different ideas/approaches in parallel. At that time the computer power and devices connectivity and networks were not powerful enough to transform the concept into a reality.

It's not until 2010 when NASA first forged the term Digital twin, being this also referred as "Virtual Digital Fleet leader".

If you like cinema you might remember "Apollo 13" movie with Tom Hanks as the main character. In the movie we can see that Nasa has built a real twin of the space capsule and they use it to test it on earth and then give exact instructions to the astronauts in order to avoid damaging the spaceship in orbit, hence avoiding an even worst situation. This is the Nasa main goal, create a digital model accurate enough to avoid having to build an exact costly replica. They described as: *"an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin"*.

In the following years US Air Force followed Nasa steps in order to design simulate maintain and predict the physical and mechanical properties of their aircrafts calling it Airframe Digital Twin, "ADT"

Since then and with the boom on information technologies the research around digital twins is growing exponentially, a quick search of term "Digital Twin" on LENS.org highlights the following trend:

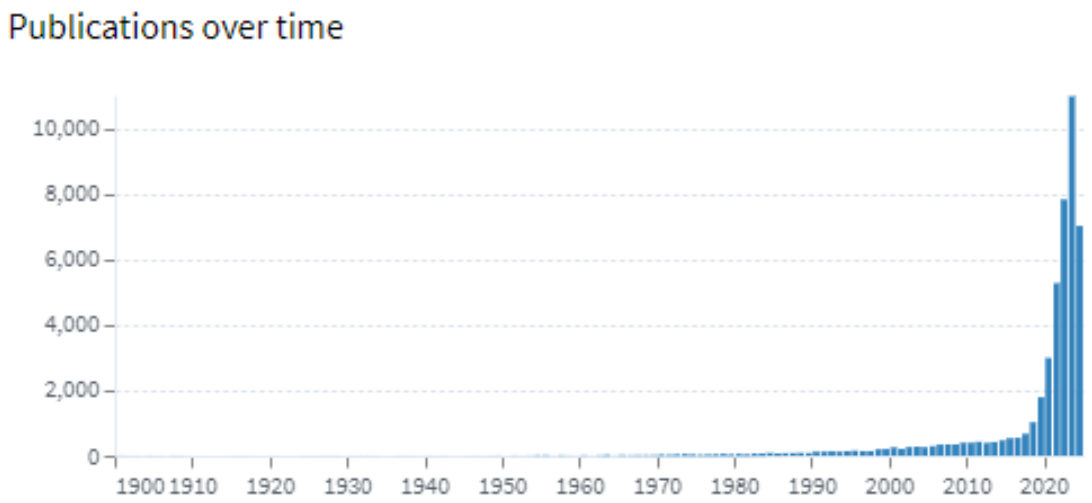

Figure 1. Screenshoot from Lens.org of the number of publications including "Digital Twin" [17]

This trend (Figure 1) even though promising for the sector still highlights that it's a new technology with a lot to grow and where there is not a clear expertise or a commercial solution that can reach the professional market.

Obviously, there are available commercial solutions but the focus of this solution is mainly to provide a data collection environment, not a turn key solution with all the hardware needed and integration needed. Below some examples:

1. Siemens Digital Industries Software (Mindsphere and Simcenter)
   Siemens offers robust solutions for creating digital twins in industries like manufacturing, energy, and automation.
   Mindsphere is an IoT platform that allows for data collection and integration with the digital twin.
   Simcenter is used for advanced simulations, enabling the modeling of physical behavior in complex environments.
2. PTC ThingWorx
   ThingWorx is an IoT and AR platform that facilitates the creation of digital twins by combining real-time data with simulations.
   It offers tools for connecting devices, analyzing data, and creating interactive 3D visualizations, making it suitable for industries like manufacturing, healthcare, and industrial automation.
3. Dassault Systèmes (3DEXPERIENCE)
   3DEXPERIENCE is a powerful platform that integrates design, simulation, and real-time data to create digital twins.
   It uses tools like CATIA, SIMULIA, and DELMIA to design and simulate digital twins in industries such as automotive, aerospace, and construction.
4. Ansys Twin Builder
   Ansys Twin Builder is a specialized solution for creating digital twins based on multiphysics simulations.
   It excels in modeling complex systems using real-time data and detailed simulations, making it widely used in advanced engineering and electronics.
5. Microsoft Azure Digital Twins
   Azure Digital Twins is a platform that allows you to model digital twins using Microsoft's cloud infrastructure.

It offers native integration with IoT, data analytics, machine learning, and simulations, making it particularly useful for smart cities, buildings, and asset management.

6. IBM Maximo Application Suite
   IBM offers tools for creating digital twins, especially in the realm of asset management and predictive maintenance.
   Maximo focuses on equipment monitoring and condition-based maintenance, which is useful in sectors like manufacturing and energy.
7. GE Digital (Predix)
   Predix from General Electric is a software platform designed for industry, enabling digital twins in sectors like energy, manufacturing, and aviation.
   It offers advanced capabilities for modeling, analyzing, and optimizing operations through digital twins.
8. Simul8
   Simul8 is a simulation tool focused on manufacturing processes and logistics operations. It allows for process modeling and scenario analysis.
   Although it's not a dedicated digital twin solution, it can be adapted to simulate real-time operations and make adjustments based on data.
9. Unity Reflect and VisualLive
   These tools are popular for creating immersive 3D visualizations and simulations, especially in architecture, construction, and smart buildings.
   Unity Reflect allows integration of BIM (Building Information Modeling) models into an interactive visual environment.
10. Amazon Web Services (AWS) IoT TwinMaker
    AWS IoT TwinMaker helps you create and manage digital twins using Amazon's cloud services. It facilitates integration with IoT sensors, data analytics, and 3D modeling, making it suitable for a variety of industrial applications.

And if we focus on the hardware side that enable us provide data to the beforementioned services the options are even less, mainly focused in the industrial sector and most of times needing gateways or edge device that can handle the data transfer to the server.

1.1.2 Main characteristics of a DT

After seeing the beginings of this technology, let's see what are the main characteristics that define a DT:

- **High fidelity**: A DT needs to be a copy as accurate as possible of his physical counterpart.
- **Dynamic**: As the physical model is changing over time, the DT must have the capability to adapt and change accordingly.
- **Self-evolving**: As seen before the dynamic characteristic leads that the DT have the ability to evolve over time.
- **Identifiable**: Unique data cause tailor fit model hence unique DT.
- **Multiscale and multiphysical**: A DT as a complex system includes several parts, so each of this parts have its own focus and DT at each scale model with each characteristics. If we take an Electric vehicle DT as example, the motor will have its own DT, the inverter as well, the battery, the chassis, the suspension… and the aggregate of all the small DT's of each part will create the DT of the EV. Obviously, the characterization of a battery or the chassis it's completely different hence the multphysical and multiscale characteristics

- **Multidisciplinary**: The creation of a digital twin involves several disciplines, from computer science to mechanical engineering.
- **Hierarchical**: The hierarchical nature of a DT comes from the fact that a complex system can have DT's of each of it's part, and the addition of all the small DT's creates the whole. In the example of an EV, we could have a DT of the battery, another of the motor… and the addition of each DT will build the DT of the EV.

1.1.3 Level of integration

This is probably one of the key parts that will determine and constrain our DT, and it's related with how the information flows:

- **Digital model:** The data between models are not automatically updated.
- **Digital shadow:** The data from the physical object flows automatically to the digital but not vice versa.
- **Digital twin:** The data from both physical and digital flows to the other automatically.

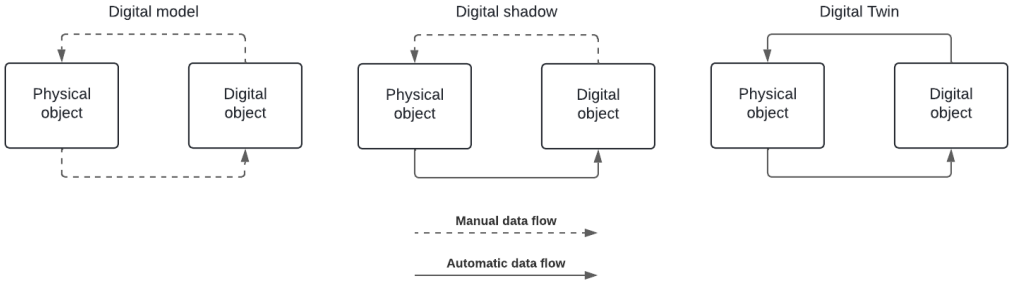

Figure 2: Detail of the level of integration [2]

The reason why the level of integration is so important it's because in the automotive industry the digital twin can only be achieved by OEM's that know the system very well or by Tier 1 that provides a major component and even though the automatically update of the physical model can possibly affect the safety level of the system. So we reach some important conclusions:

- The safety of the system cannot be compromised.
- The update on the physical level must be constrained and have clear boundaries.
- If the boundary limits are surpassed, then probably there is some issue on the part.

There are more challenges associated with the DT's for vehicles and the data flow, for example, the network availability. Most of the time the car will have network availability but there needs to be mechanisms to handle the data while there is no network available. This is one of the reasons that the Datalogger will secure all data on the SD card first and then upload it.

1.1.4 Application

A digital twin can also be categorized by the use that is going to have and here we fall into 3 categories:

Product DT: Used mainly for prototyping, it's an assurance that the production part is going to be/perform as desired and any iteration is done on the digital space instead of the physical space.

Production DT: It's main use is to validate processes by simulating them even before the actual production.

Performance DT: It's a combination of the product and production DT with the goal of improving the decision making process.

1.1.5 Hierarchy

As commented before one of the main characteristics of a DT is the hierarchy, this means that one DT can be nested into another as the complexity of a big system cannot be dealt with in global. So from less to top we have 3 levels of hierarchy:

Unit level: Is the smallest DT unit, usually pointing to a single component, material. Like for example a battery cell.

System level: It's a group of several DT's that are interconnected and are collaborating. Following the previous example, here we would have the battery pack, that will include for sure the battery cells, the BMS, can include if available the coolant pump in case of liquid cooling or the fan if air cooling… The EV itself would be considered a system.

System of Systems: It's the biggest group of System level DT, the goal of the SoS is to take high level decision on company level, supply chain, service, aftersales….

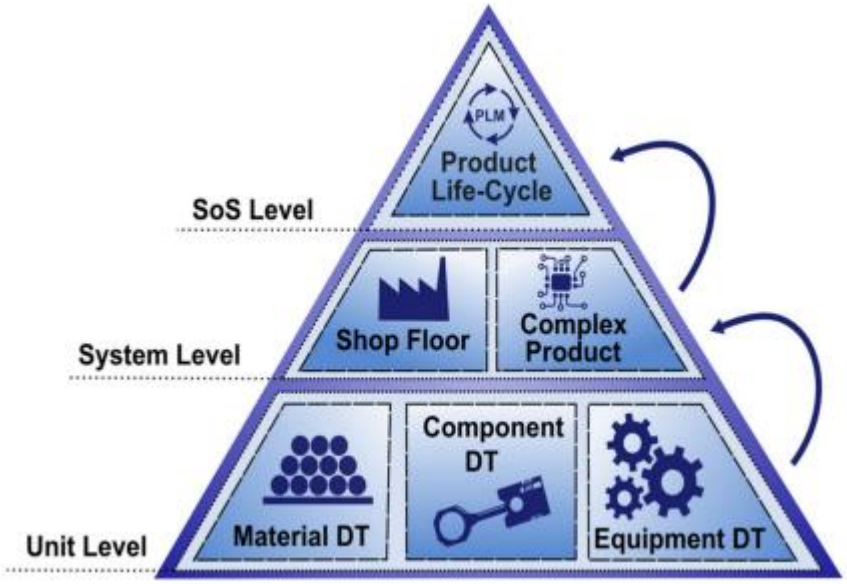

Figure 3 Pyramid of the DT's hierarchy [2]

1.1.6 Time frame

This is also a key part of a digital twin as will redefine the whole approach and infrastructure needed.

Interrogative DT: The interrogative DT is basically to recollect data and have some way to ask for events, current or past state. It's basically an up-to-date database of your model

Predictive DT: On the contrary the predictive is a look in the future, it aims to answer or predict the behaviour of the physical model by simulating it. To do so there are 2 main approaches:

Model based: When creating a predictive model of your DT if the element is not complex or it's very well-known this can be modelled. Using this method the outcome of the inputs is known from the beginning and if needed a fine tuning can be added upon the data delivery.

Data driven: In high complexity models the data can be the main source for the modelling of the DT. There are several ways to do it according to the inputs and the data but for a time series input as our CAN Datalogger an Advance Neural Network will fit fine as a predictive model.

1.2 Data structure

The whole idea on the CAN datalogger for Digital twins is to follow the next data structure. This example is done using amazon structure but can be done using any other supplier or your own servers

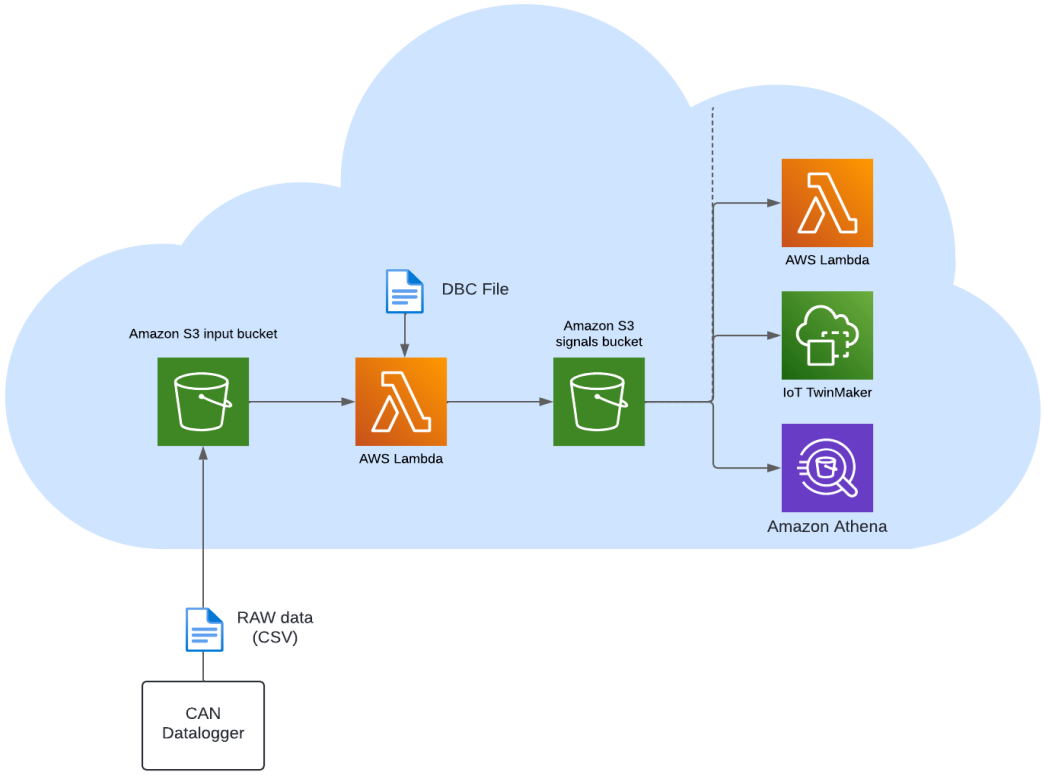

Figure 4 Cloud data structure

1. **Step 1: Record the RAW data:** The CAN Datalogger will store the data and save it in rows coded like MF4 but using CSV files format. This is because MF4 format cannot be processed easily on time on a microcontroller. All the raw data will be following the MF4 format for the data but the structure that defines the MF4 on the file itself will not be created until the CSV file reaches the server.
2. **Step 2: Upload the RAW data:** The next step is to upload the data to the server, here there are 3 strategies:
    a. **Uploading each CAN message:** This requires from a good internet connection and a microprocessor able to handle at the required speed both communications, CAN and Network. Knowing that the network (WLAN, WAN, GSM LTE…) is not deterministic, neither under our control, the difficulty increases in terms of having a large enough buffer to handle the discarded messages. This is the ideal solution that provides a real time approach and certainly useful for static sensors but not ours that is going to be on a moving vehicle.
    b. **Batch uploading:** Similar than before but now the upload is made at information in data batches. A good limit criterion is the maximum size of the server protocol we are communicating with. Again, there needs to be some handling of the errors or network issue that ensure not losing information. This approach is a mid-solution between real time and file-uploading.
    c. File uploading: Even though it's the worst option in terms of real time handling it's the easiest in terms of error proofing. The data is being collected, stored locally and when there is connection, you upload it to the server. This will be the approach taken for developing the CAN Datalogger.

    The data once has reached the server can be stored like this without processing it and create a copy of the file for further processing, so the original data can always be retrieved.
3. **Step 3: Convert the data**: In the case of Amazon, we could use a Lambda function but basically there are 2 conversions to made.
    a. First: Convert the raw data into an MF4 file format. This can be done by running a simple python script using "asdmf" library or open-source code.
    b. Second: Now that we have the data in MF4 using the vehicle dbc file, we convert the RAW data into signal data.
    Here the main drawback that we are facing is that the dbc file is only provided by the manufacturer of the vehicle(OEM). Hence to decode it we most certainly must sign a NDA with the OEM or use reverse engineering methods.
4. **Step 4: Signals data lake:** Now that we have our raw data decrypted, we can store it on another bucket or data lake, this time using the CAN decoded signals. It's important to remember that each vehicle must have its own bucket/data lake as the modelling of the digital twin depends on individual data.
    It's important to realize that now that the data is in this format handling it is much more easier. We have servers with plenty of power, several scripts can run in parallel and basically the restrictions on an embedded system have disappeared.
5. **Step 5: Application level:** Now that we have the data in the server we have to handle it using any service available. In the picture there are 3:
    a. **AWS Athena:** allows you to analyze data directly in Amazon S3 using standard SQL, without the need to set up or manage servers. Athena is a serverless tool, meaning you don't have to worry about infrastructure, and you only pay for the queries you run.
    b. **AWS IoT TwinMaker** is a service that helps create digital twins of physical environments, integrating data from IoT sensors, 3D models, and other sources to

visualize, simulate, and analyze operations in real-time. It enables businesses to optimize processes, and enhance decision-making. TwinMaker supports seamless integration with AWS services, allowing for custom applications and real-time insights.

c. **AWS Lambda** is a serverless compute service that lets you run code in response to events without provisioning or managing servers. It automatically scales and executes your code only when triggered, charging only for compute time used. Lambda integrates with various AWS services, enabling you to build scalable, event-driven applications effortlessly.

2 Datalogger

Now that we have seen the characteristics of a DT and introduced how the information must be saved, let's define the main requirements that we want to achive on our datalogger and from this point move forward on defining the system architecture, software architecture, hardware design etc.

2.1 Main requirements

Identifier	Subsystem	Requirement
DL-EL-01	Electrical	The Datalogger will be powered up using the OBD-II connector
DL-EL-02	Electrical	The Datalogger will include a DC-DC to adapt the voltage from the vehicle OBDII port (12V) to 3,3V
DL-MEC-01	Mechanical	The Datalogger will be self contained
DL-MEC-02	Mechanical	The Datalogger enclosure will be made of plastic to avoid blocking the communication and GPS signals.
DL-MEC-03	Mechanical	The Datalogger will have a DB-9 connector for interfering with the vehicle
DL-SW-01	Software	The Datalogger will have the capability to store data on a MicroSD card
DL-MEC-04	Mechanical	The MicroSD card will be accessible/removable with the datalogger enclosure closed
DL-SW-02	Software	The Datalogger will store all the received data locally into the MiscroSD
DL-SW-03	Software	The Datalogger will not start working if the MicroSD card is not connected
DL-SW-04	Software	The Datalogger will timestamp the data
DL-SW-05	Software	The Datalogger will store data from GPS and CAN in the same file
DL-SW-06	Software	The Datalogger will save the data in file format .CSV
DL-SW-07	Software	The Datalogger will create a new file every time there is a power-up, a new record pushbutton press or there is a file upload to a server
DL-SW-08	Software	The Datalogger will have the capability to upload the recorded files to a web server manually or automatically

Identifier	Subsystem	Requirement
DL-SW-09	Software	The automatic file upload to a server will only be possible if there is an autorized wifi network available
DL-SW-10	Software	The Datalogger will have 2 buttons accessible by the user with the enclosure closed
DL-SW-11	Software	The Datalogger will have 2 LED's for user notification
DL-SW-12	Software	The Datalogger will work on a plug and play manner, starting the recording of CAN and GPS data upon power up.
DL-SW-13	Software	The Datalogger will have the capability to receive CAN Bus 2.0 B.
DL-SW-14	Software	The Datalogger must secure CAN data integrity as the main priority
DL-SW-15	Software	The Datalogger will record RAW CAN data
DL-SW-16	Software	The CAN data record must be standarized format
DL-SW-17	Software	The Datalogger will have the capability to receive GPS data.
DL-SW-18	Software	The GPS data must be recorded at a rate of 1 Hz
DL-SW-19	Software	The Datalogger will record RAW GPS data
DL-SW-20	Software	The GPS data to record must be the phrase $GPGGA
DL-SW-21	Software	The Datalogger will have the capability to connect to a local network using WiFi
DL-SW-22	Software	The Datalogger will only connect to the predefined WiFi networks
DL-SW-23	Software	The Datalogger will have the capability to upload the files to an online server

2.2 System architecture & components

Below we have the system architecture of the datalogger. We can see that it's made of separate modules, and all the inputs and power supply came from the DB9 connector that is going to connect to the vehicle by means of the OBDII port.

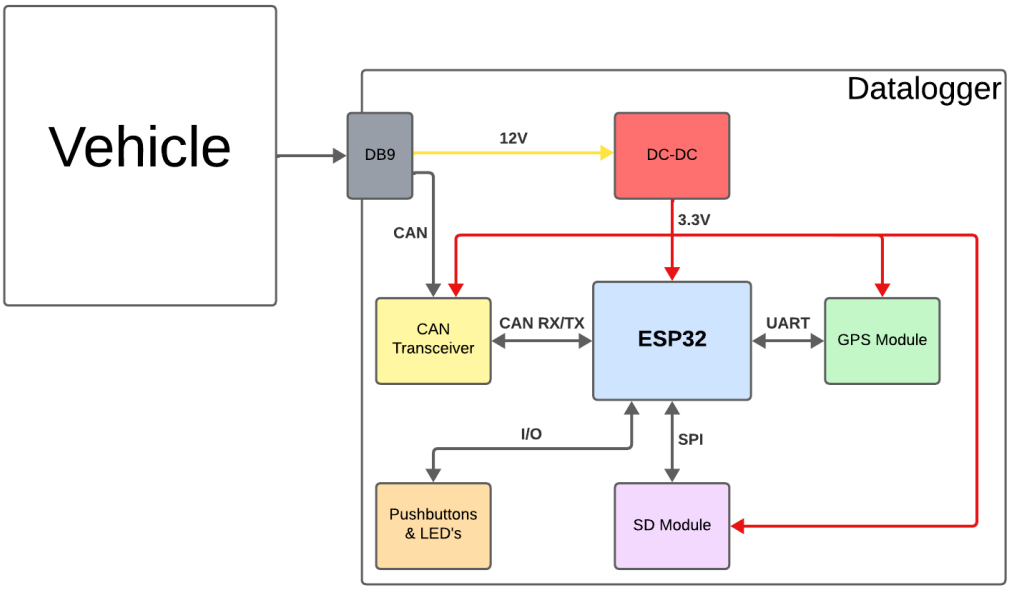

Figure 5 System architecture diagram

Main microcontroller: ESP32-WROOM-32 Devkit board

GPS module: NEO-6M-0-00

CAN Transceiver using VP230 from TI

SD module enabling SPI connectivity.

DC-DC based on LM2596

2.3 Components cost

Below there is the overall components cost of the datalogger:

Part	Cost
ESP32	3,71 €
CAN module	1,61 €
GPS module	2,79 €
DC-DC module	0,69 €
PCB	2,36 €
DB9 conector	1,19 €
Case	2,14 €
Micro SD card	8,49 €
LED x2	0,12€
Resistors x4	0,08€
Microswitches x2	0,34€
Total	**23,52 €**

2.4 Internal connection diagram

Here is the full schematic of the datalogger

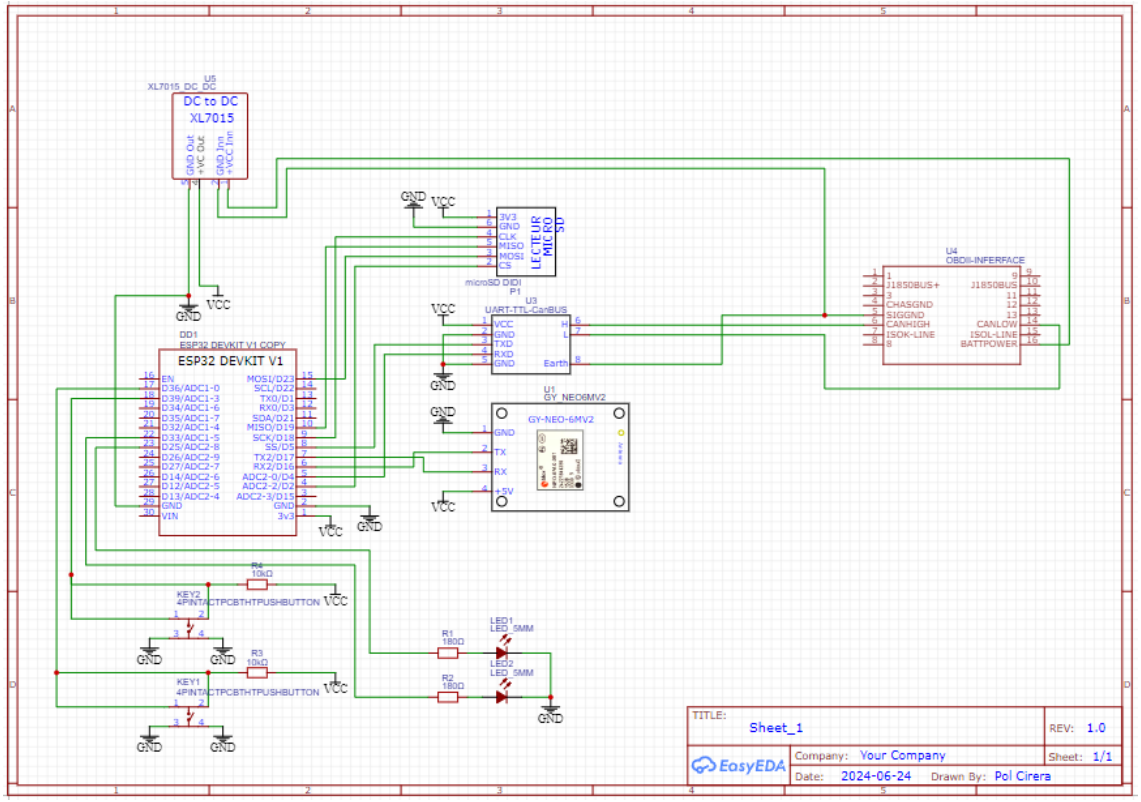

Figure 6: Schematic of the Datalogger PCB

Below there is the PCB design. In red we can see the top layer and in blue the bottom layer.

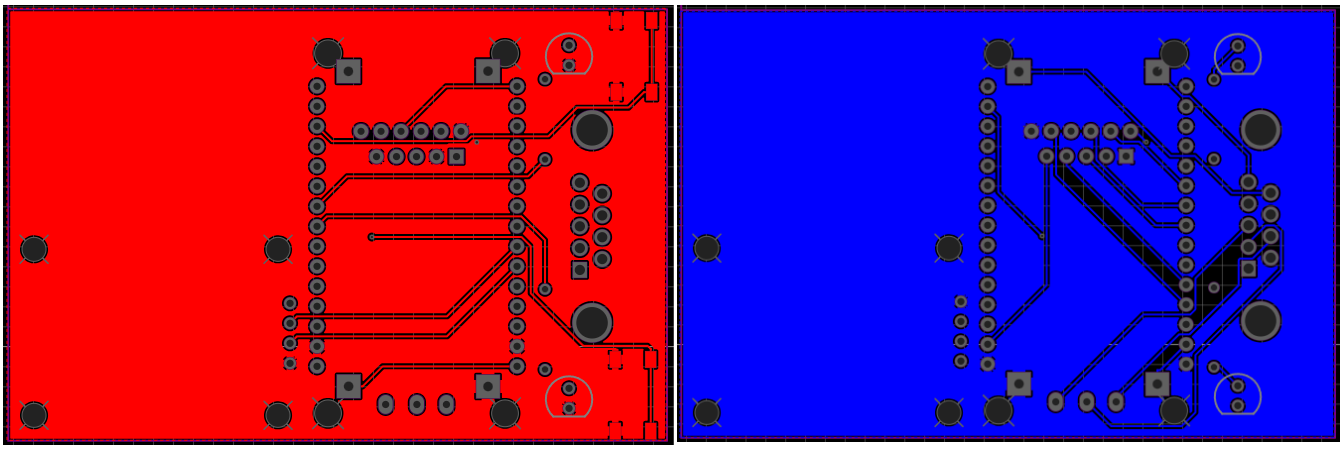

Figure 7 PCB traces detail from EDA software

2.5 Hardware Design

Once the PCB is done the casing has been designed in CAD Software Fusion 360 to be 3D printed.

The case is splited in two parts, top and bottom, that are attached between them using heated inserts in the bottom part and screws on top.

On top part we can also see the holes for the LED's, and just below the pushbuttons marked with U for Uploading and R for Recording. On the lateral is noticeable the fingertip space for extracting the MicroSD card.

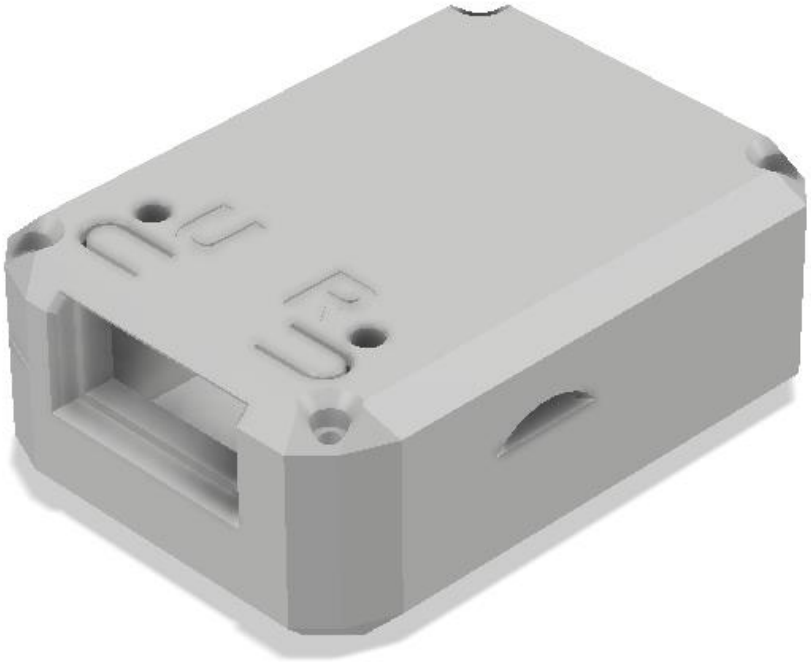

Figure 8 Render of the enclosure closed front

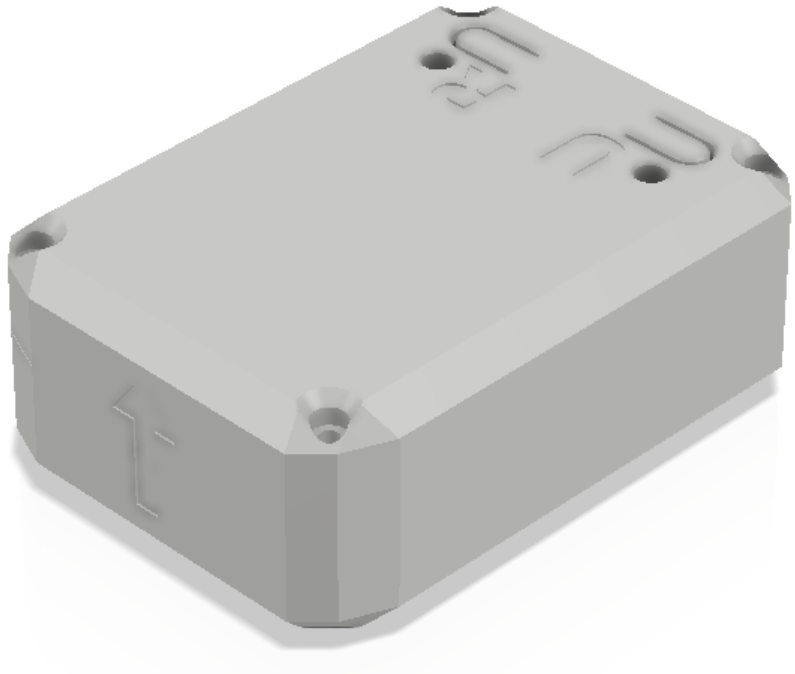

Figure 9 Render of the enclosure closed back

Here we have the detail of the case opened from front orthogonal and below from the opposite.

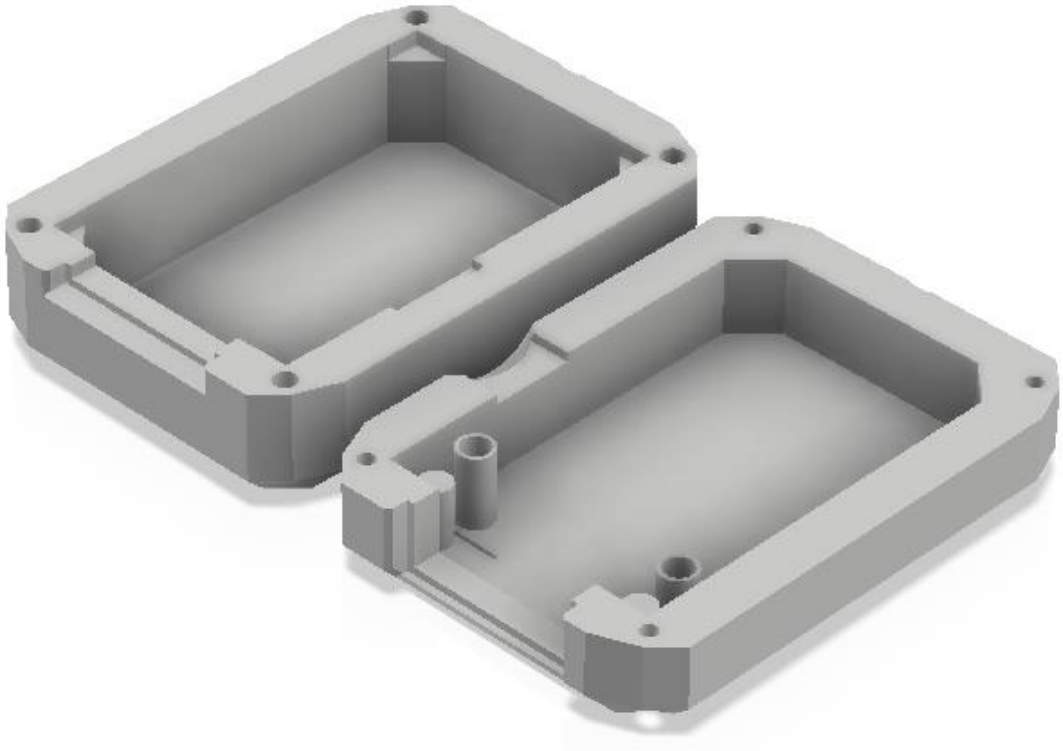

Figure 10 Render of the enclosure open front

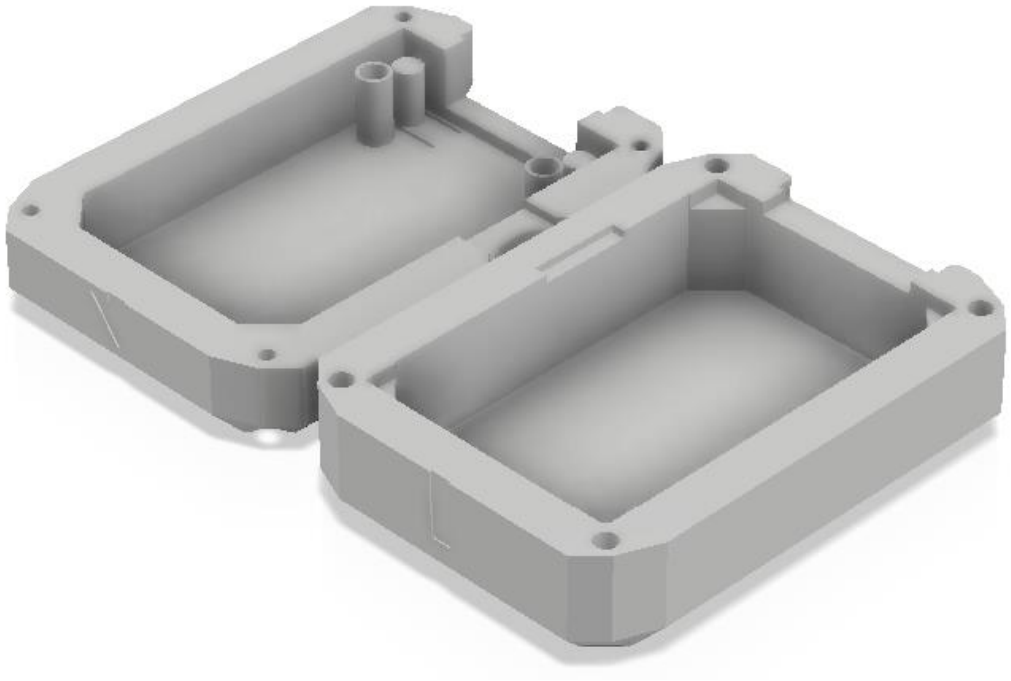

Figure 11 Render of the enclosure open back

2.6 Final hardware pictures

Final PCB design top and bottom

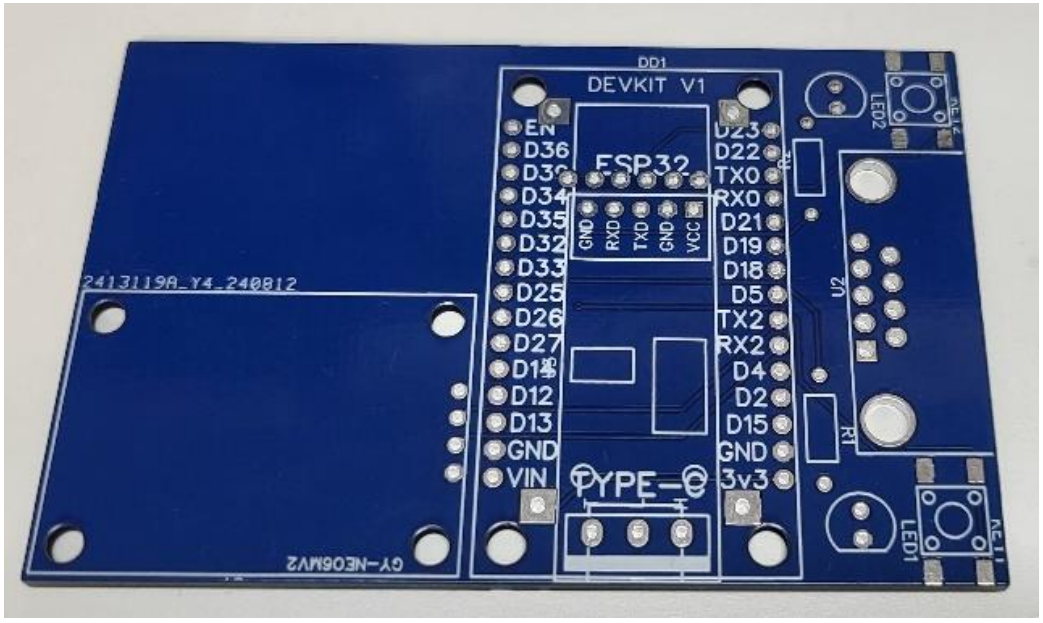

Figure 12 PCB top view

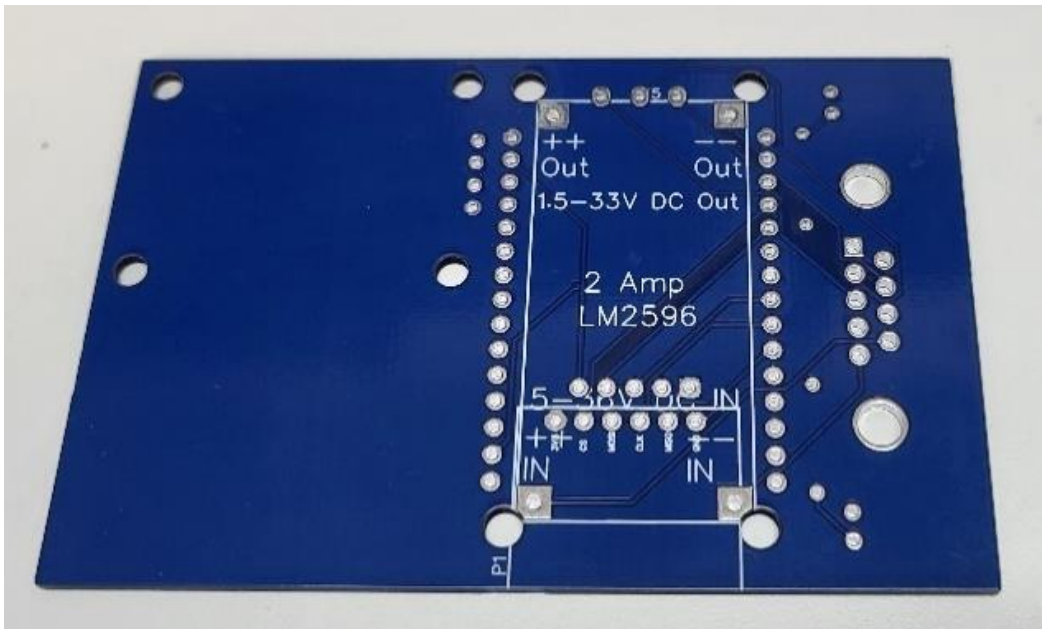

Figure 13 PCB bottom view

Figure 14 All electronic components disassembled

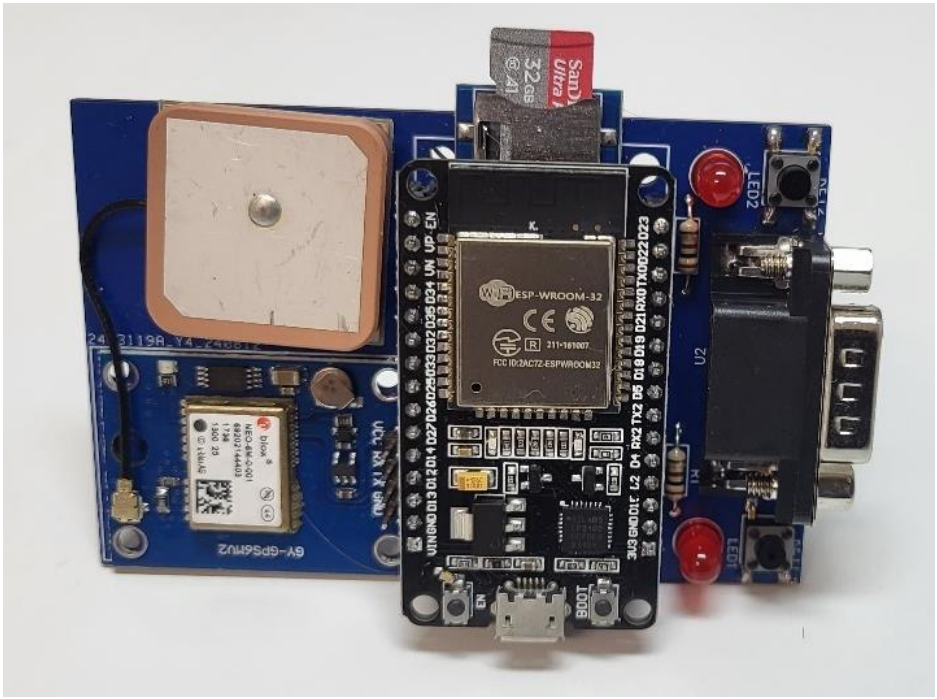

Figure 15 All electronic components assembled

Figure 16 Final parts detail

Figure 17 Final parts assembly

Figure 18 Final product front view

Figure 19 Final product front isometric view

Figure 20 Final product fingertip detail

Figure 21 Final product back isomètric view

Figure 22 Final product with OBDII cable

2.7 Software architecture

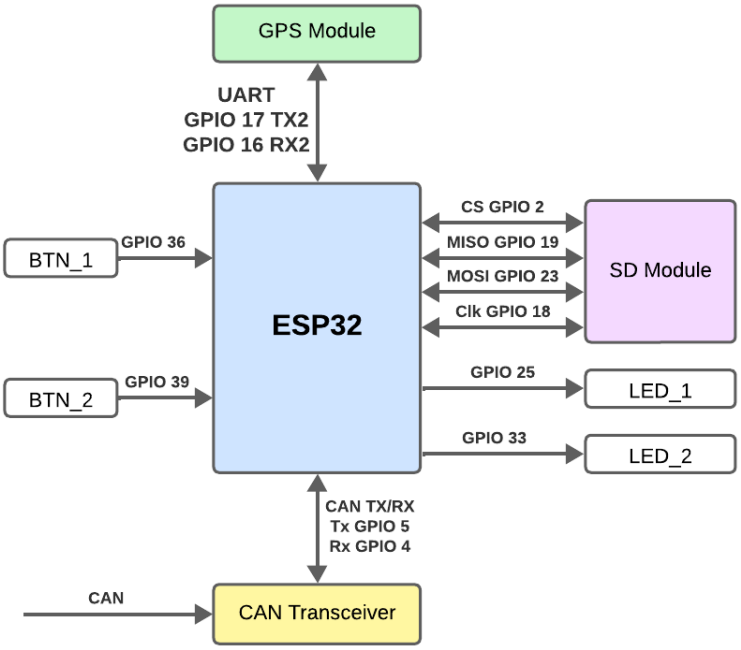

Figure 23 Software architecture

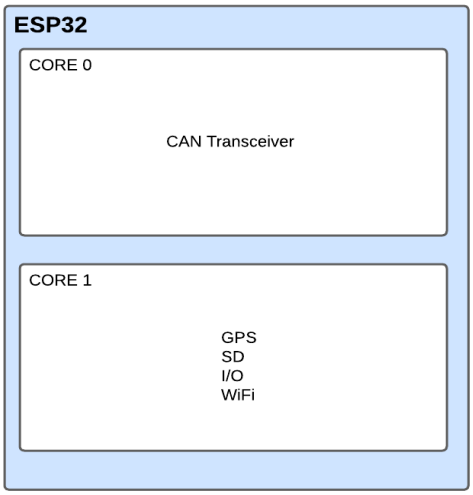

Figure 24 Task Core distribution

As seen in Figure 24, the idea is that the CAN task run alone on core 0 and the rest of the tasks run on Core 1 as are not so time critical.

Actually the usual way to configure the microcontroller is that all the communication buses/protocols runs on a core and the rest of the program/application on the other

2.8 Software design

2.8.1 Overall

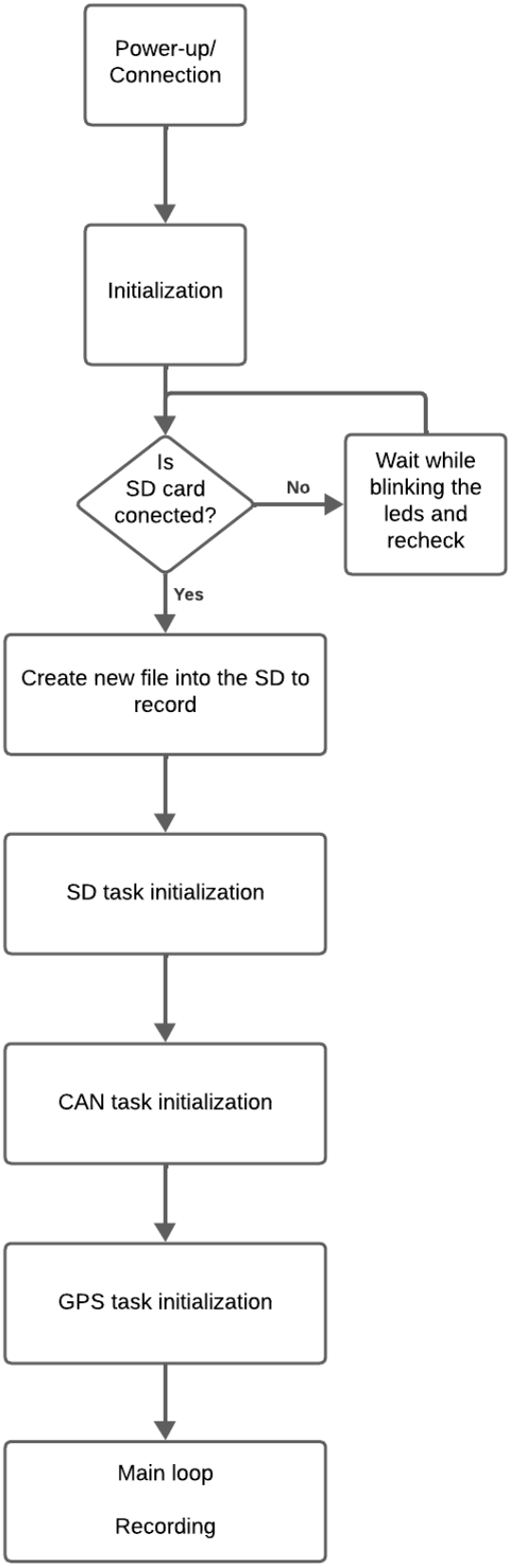

Figure 25 Overall workflow

1. **Power-up/Connection**: The process starts when the device is powered on or connected to the OBDII port

2. **Initialization**: The device goes through an initial setup or booting process.

3. **Is the SD card connected?**: The device checks whether an SD card is properly inserted.
    a. If **No**, it waits while blinking the LEDs, then rechecks for the SD card to be connected.
    b. If **Yes**, meaning there is an SD card connected, then proceed to the next step.

4. **Create a new file on the SD card to record**: A new file is created on the SD card to store the recorded data. The creation of the new file takes into account the actual file names into the SD card in order to not overwrite any existing file.

5. **SD task initialization**: The task related to storing data to the SD card is initialized.

6. **CAN task initialization**: The task related with the CAN bus is initialized

7. **GPS task initialization**: The GPS-related task is initialized.

8. **Main loop - Recording**: The device enters its main operational loop where the user inputs does take effect on the behaviour of the datalogger. Entering the main loop it's done in the recording state.

2.8.2 Main loop

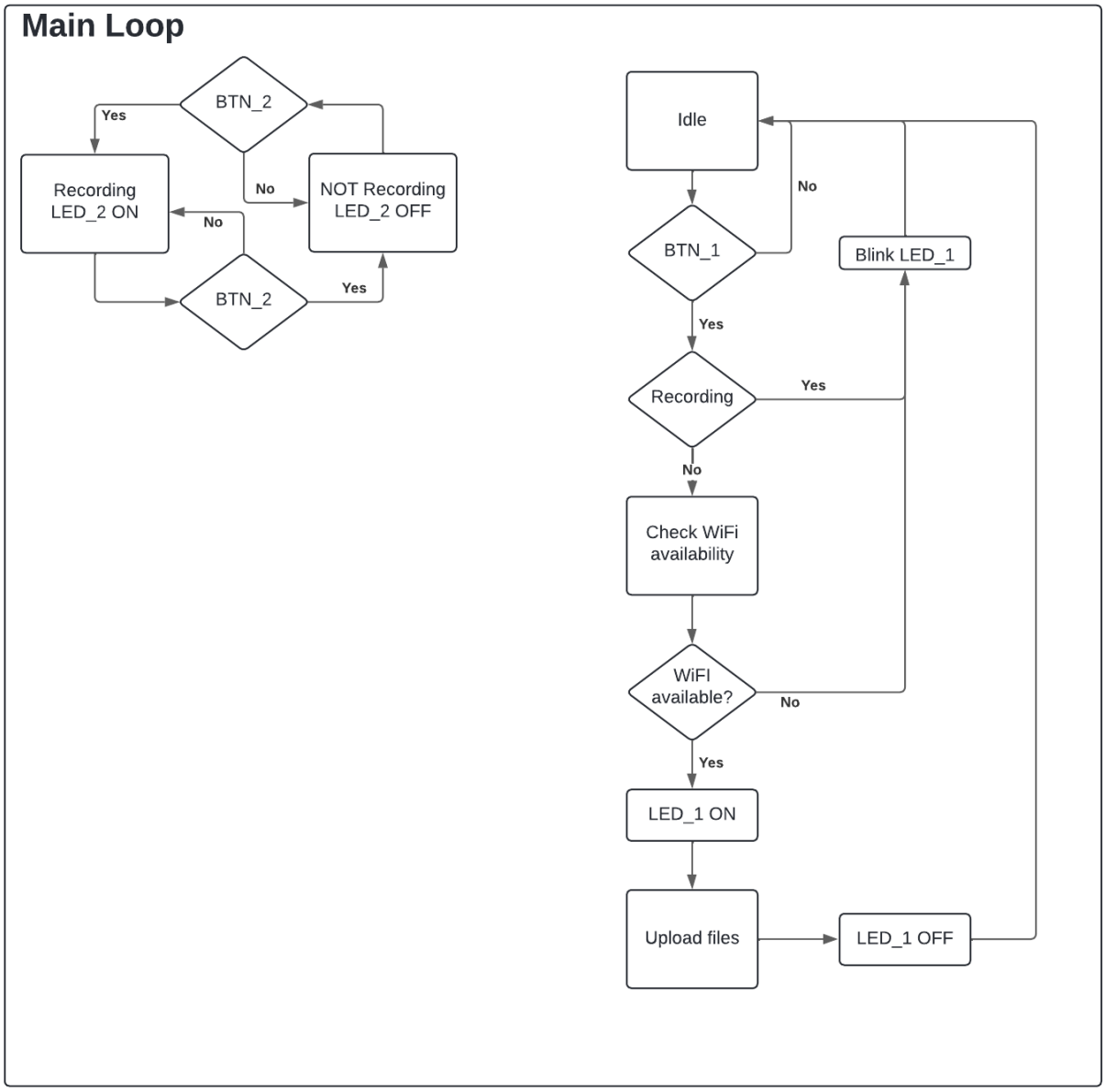

Figure 26 Main loop workflow

The main loop is where the program runs, and the user can interacts with the datalogger via the push buttons. In the following lines its function is explained in detail.

2.8.2.1 Recording Control (left side):

By default the datalogger starts recording upon connection, so on the main loop we will always start at the recording state.

1. **Start in Recording State (LED_2 ON)**: By default, the datalogger starts on recording state upon connection, indicated by LED_2 ON.
    a. **BTN_2 Pressed (Yes)**: When BTN_2 is, the system exits the recording mode. **Switching to NOT Recording (LED_2 OFF)**, pausing the GPS and CAN handling task and raising the saving flag and after a delay pausing the SD task
2. In NOT Recording State (LED_2 OFF):
    a. Now, with recording stopped (LED_2 OFF), if BTN_2 is pressed again, it **switches to Recording (LED_2 ON)**: The system turns LED_2 back on to indicate that recording is active. And starts resuming the tasks, this time in inverse order, first the SD, then the CAN and finally the GPS
3. BTN_2 Not Pressed (No):
    a. If BTN_2 is **not** pressed, the system maintains its current state:
        i. If the system was recording (LED_2 ON), it continues recording.
        ii. If the system was not recording (LED_2 OFF), it remains idle and does not record.

2.8.2.2 Wi-Fi File Upload Control (right side):

1. Idle State: The system starts in an idle state, waiting for further actions.
2. BTN_1 Pressed (Yes): If the button BTN_1 is pressed:
    - The system checks if it is currently recording.
        - If it is recording, it stays in the recording mode and as no further action can be taken the LED 1 blinks as a consequence of not being able to act
        - If it is not recording, it checks for Wi-Fi availability.
3. Check Wi-Fi Availability:
    - If Wi-Fi is available (Yes):
        - The system turns LED_1 ON to indicate the start of the file upload process.
        - All the files on the SD card are then uploaded.
        - After the upload is complete, LED_1 is turned off.
    - If Wi-Fi is not available (No):
        - The system blinks LED_1 to indicate the action cannot be completed.
4. Return to Idle: After completing the process, the system returns to the idle state.

2.8.3 CAN task

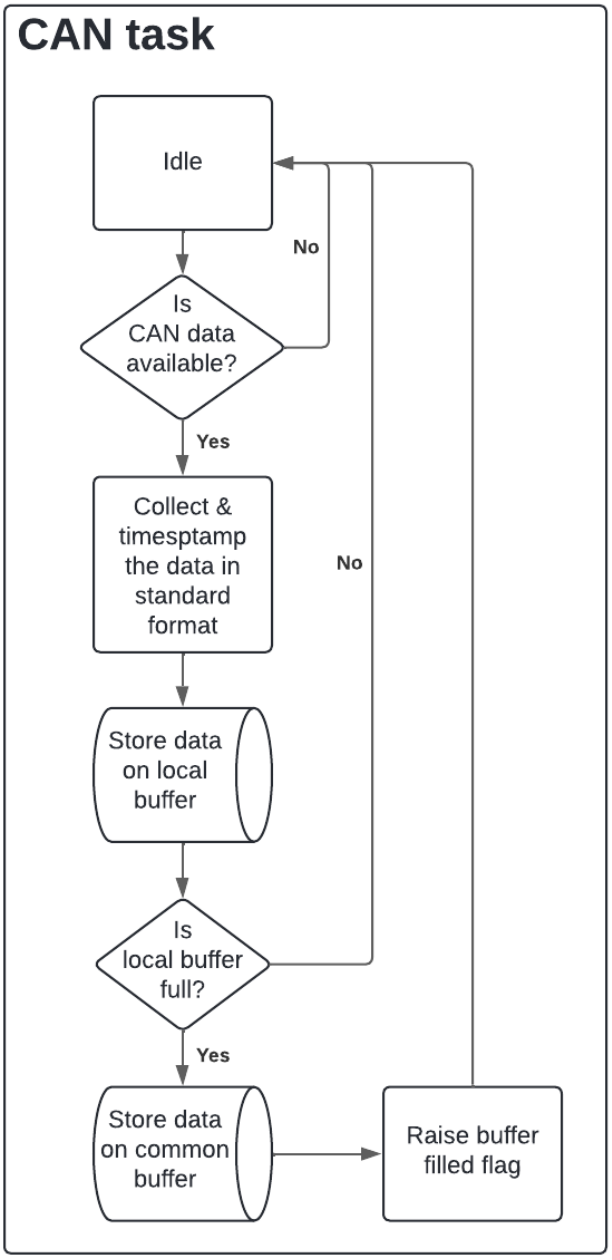

Figure 27 CAN task workflow

1 **Idle State**: The process starts in an idle state, waiting for new CAN data to become available.

2 **Is CAN Data Available?:** The system checks whether any CAN data is incoming.
   a. If **No**, the process returns to idle state and keeps checking for data.
   b. If **Yes**, it moves forward to collect the data.

3 **Collect & Timestamp the Data in Standard Format:** The system gathers the available CAN data and timestamps it, storing the information in a standardized format like MF4 than afterwards is going to be converted to.

4 **Store Data on Local Buffer:** The collected data is temporarily stored in a local buffer.

5 **Is Local Buffer Full?**: The system checks whether the local buffer has reached its capacity. The capacity of this buffer has been calculated based on the saving time of the SD card.
   a. If **No**, the process continues gathering more data until the buffer is full.
   b. If **Yes**, it proceeds to store the data in a common buffer.

6 **Store Data on Common Buffer**: Once the local buffer is full, its data is transferred to a common buffer, this being a global variable that can be accessed by the SD task.

7 **Raise Buffer Filled Flag:** After the data is stored in the common buffer, the system raises a flag indicating that the buffer is filled.

8 **Return to Idle:** The system then returns to the idle state, awaiting the next CAN data frame to be received

2.8.4 GPS task

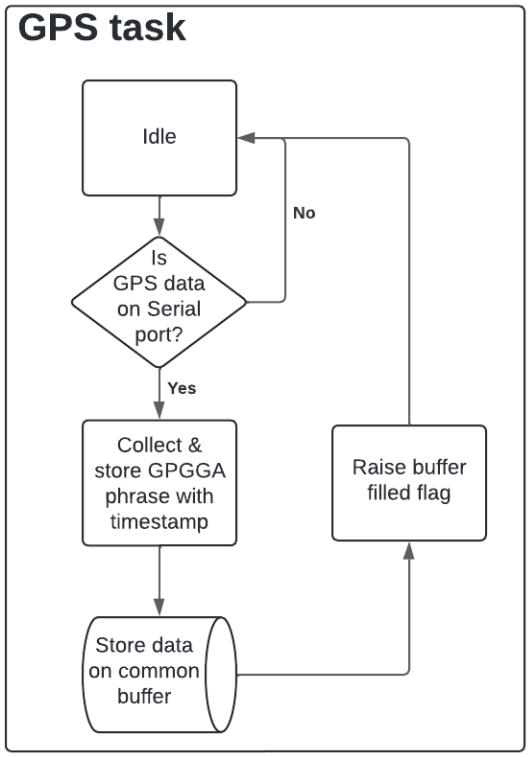

Figure 28 GPS task

1 **Idle State:** The task starts in an idle state, waiting for GPS data.

2 **Is GPS Data on the Serial Port?:** The system checks whether there is incoming GPS data available on the serial port.
   a. If No, it returns to the idle state and continues checking for data.
   b. If Yes, it proceeds to the next step.

3 **Collect & Store GPGGA Phrase with Timestamp**: When GPS data is detected, the system collects the relevant NMEA $GPGGA phrase (GPS sentence format that provides location, altitude, and time). It then adds a timestamp to the data.

4 **Store Data in Common Buffer:** The collected GPS data, along with the timestamp, is stored in a common buffer. This buffer is shared using global variables, so other tasks of the system can access to it.

5 **Raise Buffer Filled Flag:** Once the data is stored, a flag is raised to indicate that the buffer has been filled. This flag is a global variable that will be used by SD task and signals that the GPS data is ready to be stored into the SD.

6 **Loop Back to Idle:** After raising the flag, the task loops back to the idle state and repeats the process, continuously monitoring for new GPS data.

2.8.5 SD task

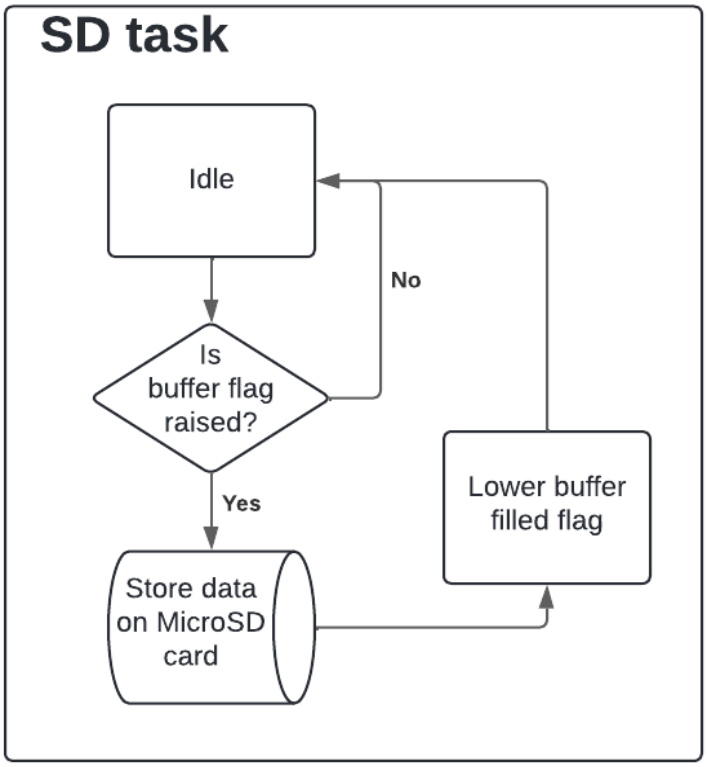

Figure 29 SD task

1 **Idle State:** The task starts in an idle state, waiting for a signal to begin.

2 **Is the Buffer Flag Raised?:** The system checks whether the buffer flag is raised. Buffer flags are the global variables raised by GPS and CAN tasks
   a. If **No**, the system returns to idle state.
   b. If **Yes**, it indicates that the buffer is filled with data that needs to be stored.

3 **Store Data on MicroSD Card**: If the buffer flag is raised, the system proceeds to store the data from the buffer onto the microSD card. Opening the actual file, append the new information and closing the file. As this process is slower than the CAN bus reading a buffer must be used.

4 **Lower Buffer Filled Flag:** After storing the data, the system lowers the buffer filled flag to indicate that the data has been saved and the buffer is now free to use for other tasks.

5 **Return to Idle:** Once the flag is lowered, the system returns to the idle state, ready to repeat the process when new data becomes available.

2.9 Software coding

Here is all the code inside the datalogger to make it work. For security reasons the wifi password and AWS credentials have been eliminated

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <ArduinoHttpClient.h>
#include <WiFiClientSecure.h>
#include <AsyncUDP.h>
#include <SD.h>
#include <SPI.h>
#include <driver/can.h>
#include <esp_system.h>
#include <stdio.h>
#include <stdlib.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <WiFiUdp.h>
#include <mbedtls/md.h>
#include <NTPClient.h>

// IO Parameters
const int Led_1 = 25;
const int Led_2 = 33;
const int PBTN_1 = 36;
const int PBTN_2 = 39;
bool Actbuttonstate1=0;
bool Actbuttonstate2=0;
bool Prevbuttonstate1=0;
bool Prevbuttonstate2=0;
bool RecStatus=0;
bool UplStatus=0;

// Configuración de AWS
const char* accessKey “”
const char* secretKey = "”;
const char* region = "eu-north-1";
const char* service = "s3";
const char* bucketName = "";
const char* awsEndpoint = "s3.eu-north-1.amazonaws.com";
// URL del API Gateway
const char* apiEndpoint = "”;
const int httpsPort = 443;
const String apiPath = "/Tests/esp32dataloggercanbucket"; // Esta es la
ruta que sigue al dominio
int ApiGatewayFailConnectionCount =0;
bool ApiGatewayFailConnection =0;
```

```
    // Configuración del servidor NTP
    WiFiUDP ntpUDP;
    NTPClient timeClient(ntpUDP, "pool.ntp.org", 0, 60000);
    String NTPCurrentDate="";

    //CAN parameters
    String BufferMessageCAN="";
    void canReceiveTask(void *pvParameters);
    TaskHandle_t canReceiveTaskHandle;
    #define BUFFER_SIZE_CAN 2048
    String CAN_BusChannel="1";
    String CAN_IDE="0";
    String CAN_Dir="0";
    String CAN_EDL="0";
    String CAN_BRS="0";

    //Wifi parameteres
    bool wifiInitialized = false;
    #define WIFI_SSID ""
    #define WIFI_PASSWORD ""

    //GPS
    #define GPS_BAUDRATE 115200
    String buffer = "";  // Variable para almacenar los datos GPGGA recibidos
    String RAWbuffer = "";  // Variable para almacenar los datos recibidos
    TaskHandle_t GPSSerialTaskHandle;

    //SD Parameters
    #define SD_CS 2
    #define SD_MISO 19
    #define SD_MOSI 23
    #define SD_SCK 18
    String uniqueFilename ="";
    String SDfilename ="";
    TaskHandle_t BufferToSDTaskHandle;
    String BuffertoSD [4]={"","","",""};
    bool BufferControl [4]={0,0,0,0};


    String generateUniqueFilename(String baseName) {
        int fileIndex = 0;
        String currentFilename = baseName;
        int longitudString=baseName.length()-4-3;
        baseName=baseName.substring(0,longitudString);
        // Verifica si el archivo existe y genera un nuevo nombre si es
necesario
        while (SD.exists(currentFilename)) {
            if (fileIndex<10){
```

```
                currentFilename = baseName + "00" + String(fileIndex) +
".csv";
                }
            else if (fileIndex<100){
                currentFilename = baseName + "0" + String(fileIndex) +
".csv";
                }
            else {
                currentFilename = baseName +  String(fileIndex) + ".csv";
                }
            fileIndex=fileIndex+1;
            }
        return currentFilename;
    }


    void uploadFileToAPIGateway() {
        String AWSFileName="";

        // Leer el archivo de la tarjeta SD
        File file = SD.open(SDfilename.c_str());
        if (!file) {
            Serial.println("Fallo al abrir el archivo");
            return;
        }

        // Leer el contenido del archivo
        String fileContent;
        while (file.available()) {
            fileContent += (char)file.read();
        }
        file.close();

        // Establecer la conexión segura
        WiFiClientSecure client;
        client.setTimeout(10000);
        client.setInsecure();  // No valida el certificado
        if (!client.connect(apiEndpoint, httpsPort)) {
            Serial.println("Conexión fallida");
            ApiGatewayFailConnectionCount=ApiGatewayFailConnectionCount+1;
            ApiGatewayFailConnection=1;
            return;
        }
        ApiGatewayFailConnection=0;
        int longString=SDfilename.length();
        AWSFileName= "/"+
NTPCurrentDate+"_"+SDfilename.substring(1,longString);

        // Crear la solicitud HTTP
```

```
        client.println("PUT " + apiPath + AWSFileName + " HTTP/1.1");
        Serial.println("PUT " + apiPath + AWSFileName + " HTTP/1.1");
        client.println("Host: " + String(apiEndpoint));
        client.println("Content-Type: text/csv");
        client.println("Content-Length: " + String(fileContent.length()));
        client.println();
        client.print(fileContent);

        // Leer la respuesta del servidor
        while (client.connected()) {
            String line = client.readStringUntil('\n');
            if (line == "\r") {
                break;
            }
            Serial.print(line);
        }

        // Leer el cuerpo de la respuesta
        String responseBody = client.readString();
        Serial.println(responseBody);

        // Cerrar la conexión
        client.stop();
    }

    void writeBufferToSDTask (void *pvParameters) {
    //Serial.println("Task started SD");
    String MessageToSave ="";
    bool DataToSave=false;
        while (true){
            for (size_t i = 0; i < 4; i++){
                if (BufferControl[i]==1){
                    MessageToSave=MessageToSave+BuffertoSD [i];
                    BufferControl[i]=0;
                    DataToSave=true;
                }
            }
            if (DataToSave==true){
                File dataFile = SD.open(uniqueFilename.c_str(), FILE_APPEND);
                    if (dataFile) {
                        dataFile.print(MessageToSave);
                        dataFile.close();
                        DataToSave=false;
                        MessageToSave="";
                    }
                    else {
                        Serial.println("Error abriendo el archivo para
escribir");
                    }
```

```
            }
        vTaskDelay(10 / portTICK_PERIOD_MS);  // Pausa breve para evitar
sobrecarga de CPU
        }
    }

    void canReceiveTask(void *pvParameters) {
      //Serial.println("Task started CAN");
      can_message_t rx_frame;
      BufferMessageCAN.reserve(BUFFER_SIZE_CAN);
      BufferMessageCAN="";
      int i=0;
        while (true) {
            if (can_receive(&rx_frame, pdMS_TO_TICKS(10)) == ESP_OK) {
                unsigned long timestamp = millis();
                //Timestamp;BusChannel;ID;IDE;DLC;DataLength;Dir;EDL;BRS;Data
Bytes
                String dataLine =   String(timestamp) + ";" +
                                    CAN_BusChannel + ";" +
                                    "0x"+ String(rx_frame.identifier, HEX) +
";" +
                                    //CAN_IDE + ";" +
                                    rx_frame.flags + ";" +
                                    String(rx_frame.data_length_code) + ";" +
                                    String(rx_frame.data_length_code) + ";" +
                                    CAN_Dir + ";" +
                                    CAN_EDL + ";" +
                                    CAN_BRS + ";"
                                    ;
                for (int i = 0; i < rx_frame.data_length_code; i++) {
                    char buf[4]; // Para almacenar el byte formateado como
HEX
                    sprintf(buf, "%02X ", rx_frame.data[i]);
                    dataLine += buf;
                }
                dataLine += "\n";
                BufferMessageCAN= BufferMessageCAN+dataLine;
                }

            if (BufferMessageCAN.length() >= BUFFER_SIZE_CAN) {
                i=0;
                if (BufferControl[i]==0) {
                    BuffertoSD [i]=BufferMessageCAN;
                    BufferControl[i]=1;
                    BufferMessageCAN="";
                }
                else if (BufferControl[i+1]==0){
                    BuffertoSD [i+1]=BufferMessageCAN;
                    BufferControl[i+1]=1;
```

```
                    BufferMessageCAN="";
                }
                else if (BufferControl[i+2]==0){
                    BuffertoSD [i+2]=BufferMessageCAN;
                    BufferControl[i+2]=1;
                    BufferMessageCAN="";
                }
                else{
                    Serial.println("Buffer CAN overflow");
                }
            }
            vTaskDelay(1 / portTICK_PERIOD_MS);  // Pausa breve para evitar
sobrecarga de CPU
        }
    }

    void GPSserialTask(void *parameter) {
      Serial.println("Task started GPS");
      unsigned long timestamp;
      char c;

      while (true) {
        RAWbuffer="";
        timestamp = millis();
        while (Serial2.available() ){
            c = Serial2.read();  // Leer el siguiente caracter del puerto
serie
            RAWbuffer=RAWbuffer+c;
            }
        int startIndex = RAWbuffer.indexOf("$GPGGA");
        if (startIndex != -1) {
            // Encuentra la posición del siguiente $
            int endIndex = RAWbuffer.indexOf('$', startIndex + 6); // +6 para
pasar $GPGGA
            if (endIndex != -1) {
                // Extraer la cadena entre $GPGGA y el siguiente $
                buffer = String(timestamp) + ";3;;;;;;;;;"
+RAWbuffer.substring(startIndex, endIndex);
                //Serial.print(buffer);
                BuffertoSD [3]=buffer;
                BufferControl[3]=1;
            }
        }
        vTaskDelay(1000 / portTICK_PERIOD_MS);  // Pausa breve para evitar
sobrecarga de CPU
      }
    }

    void setup_can_driver(){
```

```
      can_general_config_t general_config = {
            .mode = CAN_MODE_NORMAL,
            .tx_io = (gpio_num_t)GPIO_NUM_5,
            .rx_io = (gpio_num_t)GPIO_NUM_4,
            .clkout_io = (gpio_num_t)CAN_IO_UNUSED,
            .bus_off_io = (gpio_num_t)CAN_IO_UNUSED,
            .tx_queue_len = 100,
            .rx_queue_len = 65,
            .alerts_enabled = CAN_ALERT_NONE,
            .clkout_divider = 0
        };
        can_timing_config_t timing_config = CAN_TIMING_CONFIG_500KBITS();
        can_filter_config_t filter_config = CAN_FILTER_CONFIG_ACCEPT_ALL();
        esp_err_t error;

        error = can_driver_install(&general_config, &timing_config,
&filter_config);
        if (error == ESP_OK){
            Serial.println("CAN Driver installation success...");
            }
        else{
            Serial.println("CAN Driver installation fail...");
            return;
            }

        // start CAN driver
        error = can_start();
        if (error == ESP_OK){
            Serial.println("CAN Driver start success...");
            }
        else{
            Serial.println("CAN Driver start FAILED...");
            return;
            }
    }

    void CreateNewFile(){
      uniqueFilename = generateUniqueFilename(uniqueFilename);
      Serial.print("uniqueFilename: ");
      Serial.println(uniqueFilename);

      // Verifica si se puede abrir un archivo en la tarjeta SD
      File testFile = SD.open(uniqueFilename.c_str(), FILE_WRITE);
      if (testFile) {
        //testFile.println("CAN_Datalogger");
        testFile.println("Timestamp;BusChannel;ID;IDE;DLC;DataLength;Dir;EDL;
BRS;DataBytes;GPS");
        testFile.close();
      } else {
```

```
      Serial.println("Error opening file on SD card.");
    }
  }

  void SD_Inserted_Check(){
    while (!SD.begin(SD_CS, SPI, 40000000)) {
      Serial.println("Esperando que se inserte la tarjeta SD...");
      delay(250);
      digitalWrite(Led_2,LOW);
      digitalWrite(Led_1,LOW);
      delay(250);
      digitalWrite(Led_2,HIGH);
      digitalWrite(Led_1,HIGH);
      delay(250);
      digitalWrite(Led_2,LOW);
      digitalWrite(Led_1,LOW);
      delay(250);
      digitalWrite(Led_2,HIGH);
      digitalWrite(Led_1,HIGH);
    }
    Serial.println("Tarjeta SD inicializada exitosamente.");
  }

  void START_Recording(){
    //SD_Inserted_Check();
    digitalWrite(Led_2,HIGH);
    CreateNewFile();
    vTaskResume(BufferToSDTaskHandle);
    vTaskResume(canReceiveTaskHandle);
    vTaskResume(GPSSerialTaskHandle);
    RecStatus=1;
  }

  void STOP_Recording(){
    digitalWrite(Led_2,LOW);
    vTaskSuspend(canReceiveTaskHandle);
    while (eTaskGetState(canReceiveTaskHandle)!= eSuspended){
      vTaskSuspend(canReceiveTaskHandle);
      delay(1);
      }
    vTaskSuspend(GPSSerialTaskHandle);
    for (size_t i = 0; i < 4; i++){
        BufferControl[i]=1;
        }
    delay(30);
    vTaskSuspend(BufferToSDTaskHandle);
    RecStatus=0;
  }
```

```
void iniciarWiFi() {
  Serial.println("Iniciando conexión WiFi...");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  bool Wificonnection=false;
  bool WifiTimeout=false;
  int i =0;
  while (!Wificonnection and !WifiTimeout) {
    if (WiFi.status() != WL_CONNECTED){Wificonnection=true;}
    if (i>30){WifiTimeout=true;}
    delay(1000);
    Serial.println("Conectando...");
    i=i+1;
  }

  if (Wificonnection){
    Serial.println("Conectado a WiFi!");
    wifiInitialized = true;
  }
  else if (WifiTimeout){
    Serial.println("Timeout WiFi!");
    wifiInitialized = false;
  }

}

void blinkerrorLED1(){
  digitalWrite(Led_1,HIGH);
    delay(250);
    digitalWrite(Led_1,LOW);
    delay(250);
    digitalWrite(Led_1,HIGH);
    delay(250);
    digitalWrite(Led_1,LOW);
    delay(250);
    digitalWrite(Led_1,HIGH);
    delay(250);
    digitalWrite(Led_1,LOW);
}

String getValidDate() {
    String currentDate;
    bool Incorrectdate=true;

    timeClient.begin();

    while(Incorrectdate){
        // Actualizar la hora desde el NTP
        timeClient.update();
```

```
            // Obtener la hora en formato epoch
            unsigned long epochTime = timeClient.getEpochTime();
            struct tm* ptm = gmtime((time_t*)&epochTime);

            // Verificar si el año es válido
            int currentYear = ptm->tm_year + 1900;
            if (currentYear > 2020) {
                // Formatear la fecha y hora
                char dateBuffer[20];
                sprintf(dateBuffer, "%04d-%02d-%02d_%02d:%02d:%02d",
                        currentYear,
                        ptm->tm_mon + 1,
                        ptm->tm_mday,
                        ptm->tm_hour,
                        ptm->tm_min,
                        ptm->tm_sec);

                // Convertir a String y devolver
                currentDate = String(dateBuffer);
                Incorrectdate=false;
            } else {
                Serial.println("Fecha no válida recibida, intentando
nuevamente...");
                delay(1000);  // Esperar un momento antes de intentar de
nuevo
            }
        }
        return currentDate;
    }

    void UploadFilesOnSD(File dir, String path) {
        while (true) {
            File entry = dir.openNextFile();
            if (!entry) {
                // No más archivos
                break;
            }

            String currentPath = path +  entry.name();

            if (entry.isDirectory()) {
                // Si es un directorio, lo recorremos recursivamente
                Serial.println("Directorio: " + currentPath);
                UploadFilesOnSD(entry, currentPath);
            } else {
                // Si es un archivo, lo subimos
                Serial.println("Archivo: " + currentPath);
                SDfilename=currentPath;
                ApiGatewayFailConnectionCount=0;
```

```
                ApiGatewayFailConnection=0;
                uploadFileToAPIGateway();
                delay(20);
                while(ApiGatewayFailConnection &&
ApiGatewayFailConnectionCount<20){
                    uploadFileToAPIGateway();
                    delay(20);
                }
                if (ApiGatewayFailConnection){
                    Serial.println("Exceeded retries");
                }else{
                    Serial.println ("Removing file: " + SDfilename);
                    SD.remove(SDfilename);
                }
            }
            entry.close();
        }
    }

    void setup() {
      Serial.begin(115200);
      Serial2.begin(115200);

      btStop();

      pinMode(Led_1,OUTPUT);
      pinMode(Led_2,OUTPUT);
      pinMode(PBTN_1,INPUT_PULLUP);
      pinMode(PBTN_2,INPUT_PULLDOWN);
      RecStatus=1;
      digitalWrite(Led_2,HIGH);
      UplStatus=0;
      digitalWrite(Led_1,LOW);

      SPI.begin(SD_SCK, SD_MISO, SD_MOSI, SD_CS);
      SD_Inserted_Check();

      uniqueFilename = "/Can_Data_000.csv";
      CreateNewFile();


      xTaskCreatePinnedToCore(
        writeBufferToSDTask,    // Función de la tarea
        "writeBufferToSDTask",  // Nombre de la tarea
        4096,                   // Tamaño de la pila
        NULL,                   // Parámetros de la tarea
        2,                      // Prioridad de la tarea
        &BufferToSDTaskHandle,  // Handle de la tarea
        1                       // Núcleo en el que se ejecutará
```

```
    );

    setup_can_driver();

    xTaskCreatePinnedToCore(
      canReceiveTask,         // Función de la tarea
      "CAN Receive Task",     // Nombre de la tarea
      4096,                   // Tamaño de la pila
      NULL,                   // Parámetros de la tarea
      1,                      // Prioridad de la tarea
      &canReceiveTaskHandle,  // Handle de la tarea
      0                       // Núcleo en el que se ejecutará
    );

    xTaskCreatePinnedToCore(
      GPSserialTask,          // Función de la tarea
      "GPSSerialTask",        // Nombre de la tarea
      4096,                   // Tamaño de la pila
      NULL,                   // Parámetros de la tarea
      2,                      // Prioridad de la tarea
      &GPSSerialTaskHandle,   // Handle de la tarea
      1                       // Núcleo en el que se ejecutará
    );
  }

  void loop() {

  Prevbuttonstate1=Actbuttonstate1;
  Prevbuttonstate2=Actbuttonstate2;
  Actbuttonstate1=!digitalRead(PBTN_1);
  Actbuttonstate2=!digitalRead(PBTN_2);

  if (Actbuttonstate1==1 && Prevbuttonstate1==0){
    if(RecStatus==0){
        if (UplStatus==0){
            UplStatus=1;
            digitalWrite(Led_1,HIGH);
            iniciarWiFi();
            if (wifiInitialized){
                SD_Inserted_Check();

                NTPCurrentDate=getValidDate();
                Serial.println("Fecha y hora actuales: " + NTPCurrentDate);

                File root = SD.open("/");
                if (!root) {
                    Serial.println("Fallo al abrir el directorio raíz");
                }else{
                    UploadFilesOnSD(root, "");
```

```
                    UploadFilesOnSD(root, "");
                    root.close();
                }

            }else{
                blinkerrorLED1();
            }
            UplStatus=0;
            digitalWrite(Led_1,LOW);
        }
    }
    if (RecStatus==1){
        blinkerrorLED1();
    }
}

if (Actbuttonstate2==1 && Prevbuttonstate2==0){
  if (RecStatus==0 && UplStatus==0){
    START_Recording();
  }else if (RecStatus ==1 && UplStatus==0){
    STOP_Recording();
  }
}

delay(50);
}
```

3 Data processing

3.1 Server set-up

To update the data to a server I've tried several things:

Assembling an **FTP server** on my WLAN: by using FileZilla Server I did create a server easily. And by using FileZilla client I did manage to check that everything is properly working, connecting and uploading files but I could not manage to code properly the access to the same with the ESP32. Also, even though it's a good initial step it lacks the ability to being access via WAN so if there is no FTP server configured on the WLAN the connection and data upload is not possible.

InfluxDB, it's a database based on AWS that simplifies the connection by sharing the ESP32 code and enabling "fast connections". I did manage to upload data there, but the refresh rate was not so fast and the whole idea was to upload events, not files. After some coding I was uploading a GPS frame every second and loosing 2 out of 3 messages on the transmission, then this option was discarded too.

Next step was trying to connect to **AWS** S3 bucket directly and certainly this did not work either, what did work though was to create an API into AWS to PUT the files into an S3 bucket using HTTP, and this is the procedure:

First we have to log and create an account into AWS. Once we are logged in the next steps must be followed:

3.1.1 S3 Bucket Creation

This is the first and easiest step, search for S3 on the search bar, and click on it:

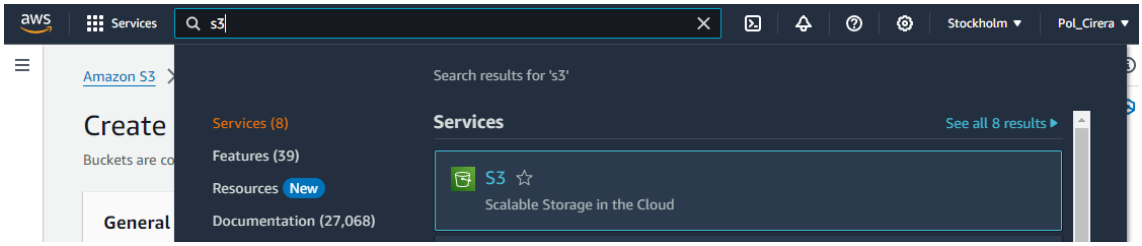

Figure 30 AWS search bar S3 [18]

Then go to "Create Bucket":

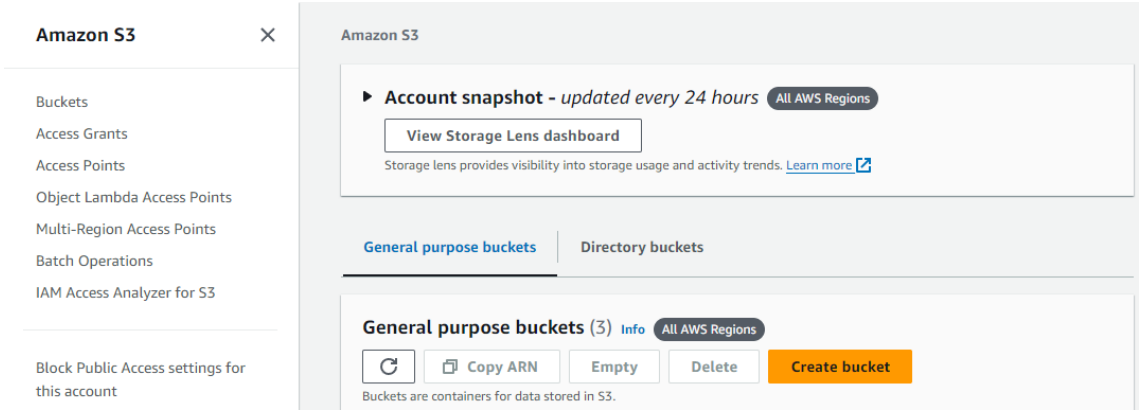

Figure 31 AWS S3 [18]

We put the desired name, in our case "esp32dataloggercanbucket"

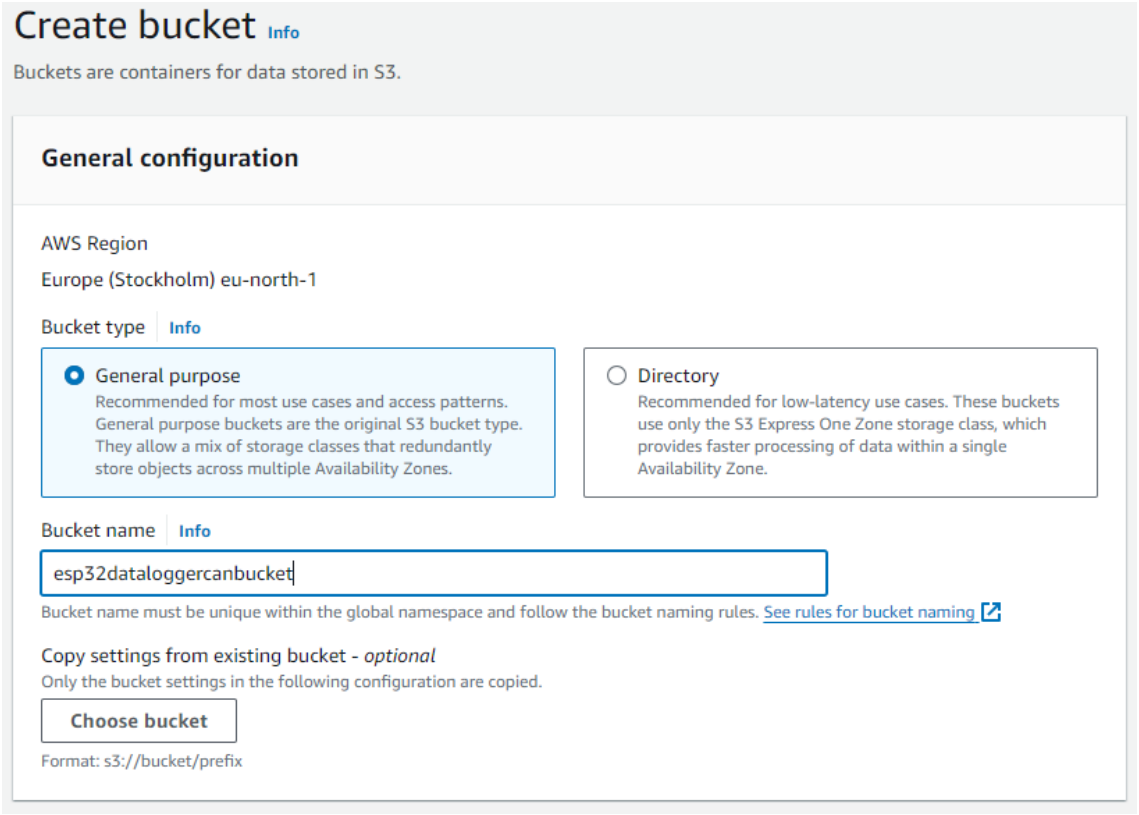

Figure 32 S3 Bucket general configuration [18]

We enable bucket versioning:

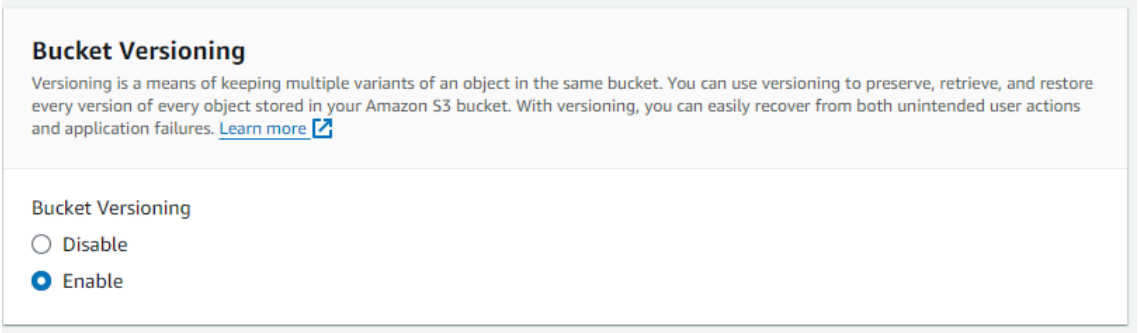

Figure 33 S3 Bucket versioning [18]

Leave the rest of the options as default and create bucket:

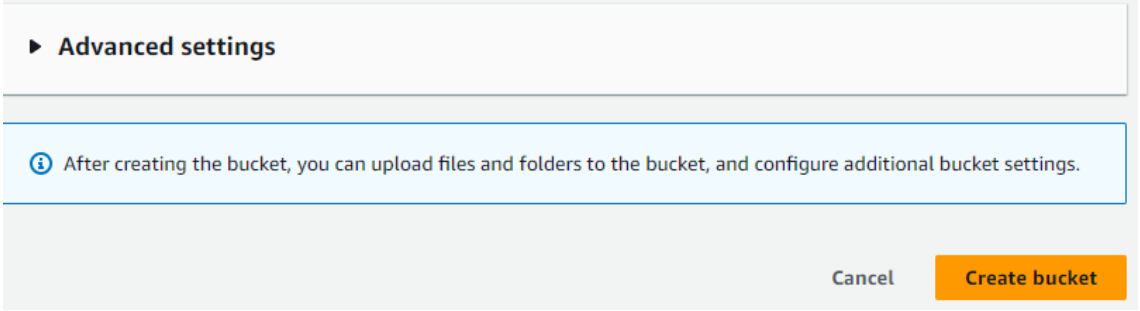

Figure 34 S3 Bucket creation [18]

And our bucket is now created and can be seen from the S3 console.

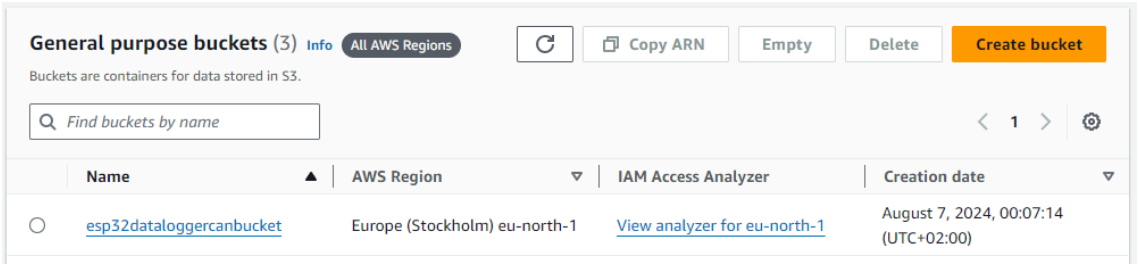

Figure 35 S3 Bucket checkout [18]

3.1.2 IAM Role Creation

Next step is to create a Role, so let's search for "IAM" on the searcher bar and click it.

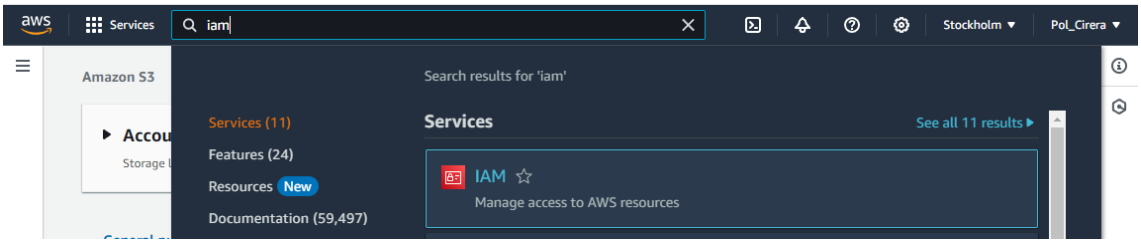

Figure 36 IAM role search bar [18]

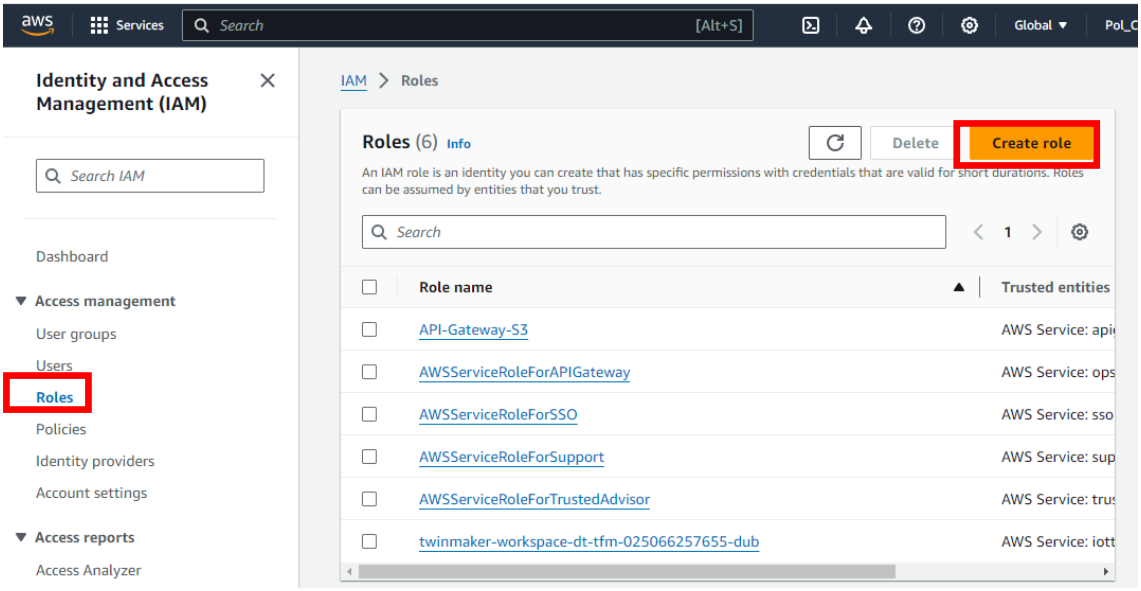

Figure 37 IAM role creation [18]

Then go to "Roles" and "Create a role":

Select "AWS Service" and search for "API Gateway" and click next

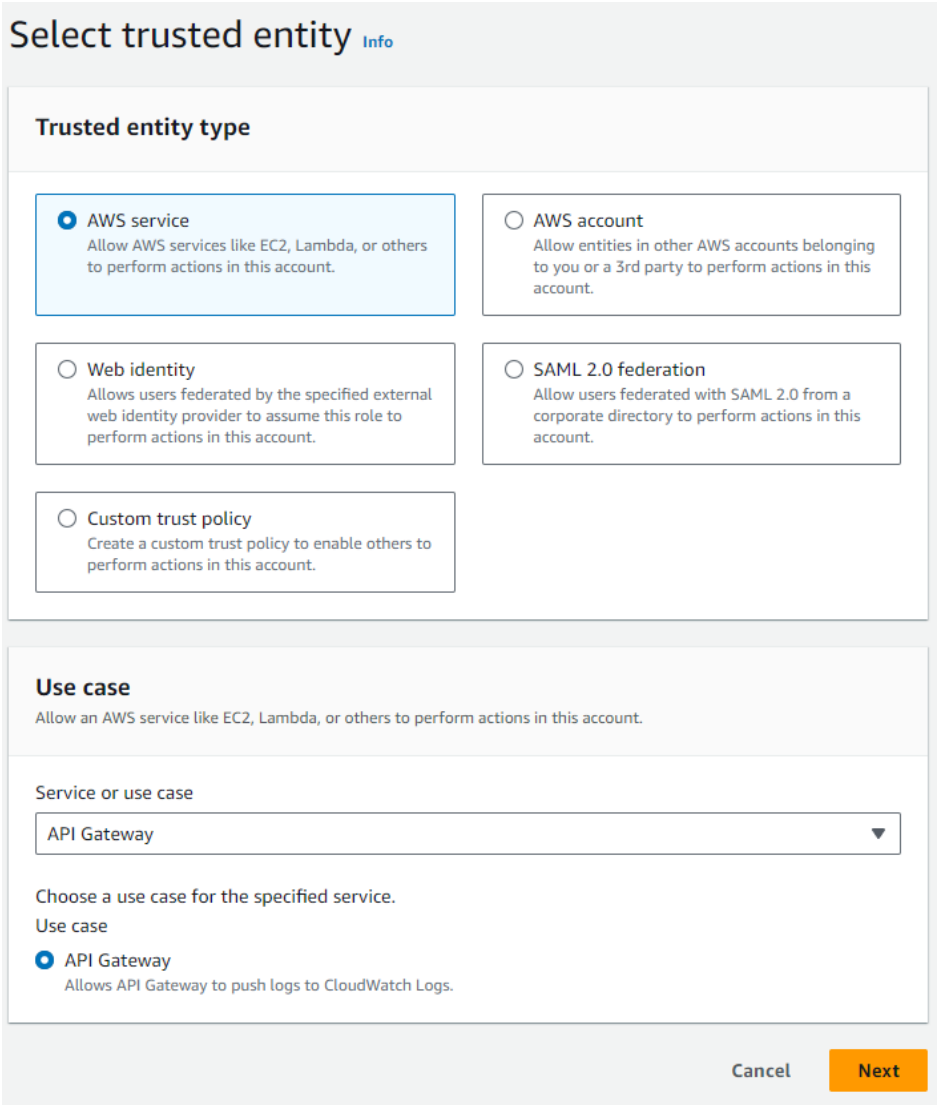

Figure 38 Role configuration [18]

On the following window we just click "Next", leave it as default:

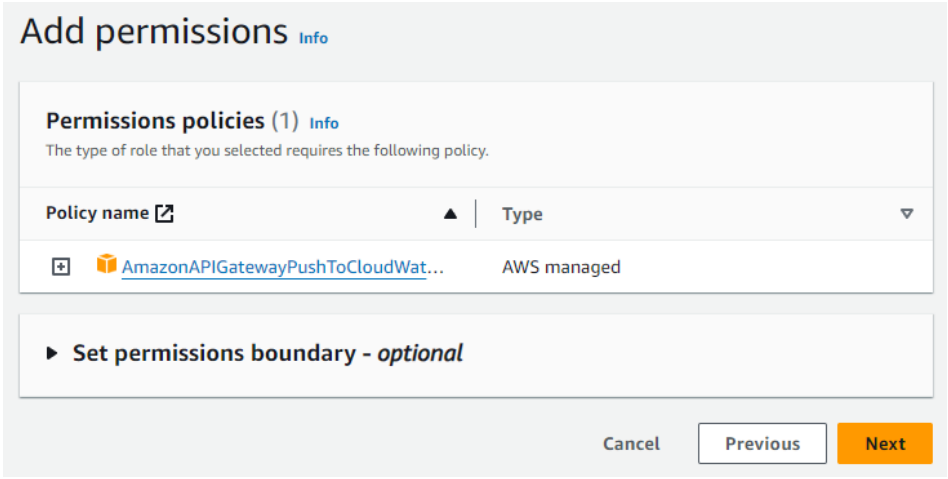

Figure 39 Role permissions [18]

To finalize, the Role name must be filled in our case with "API-Gateway-S3", a description can be included and then scroll down and click "Next".

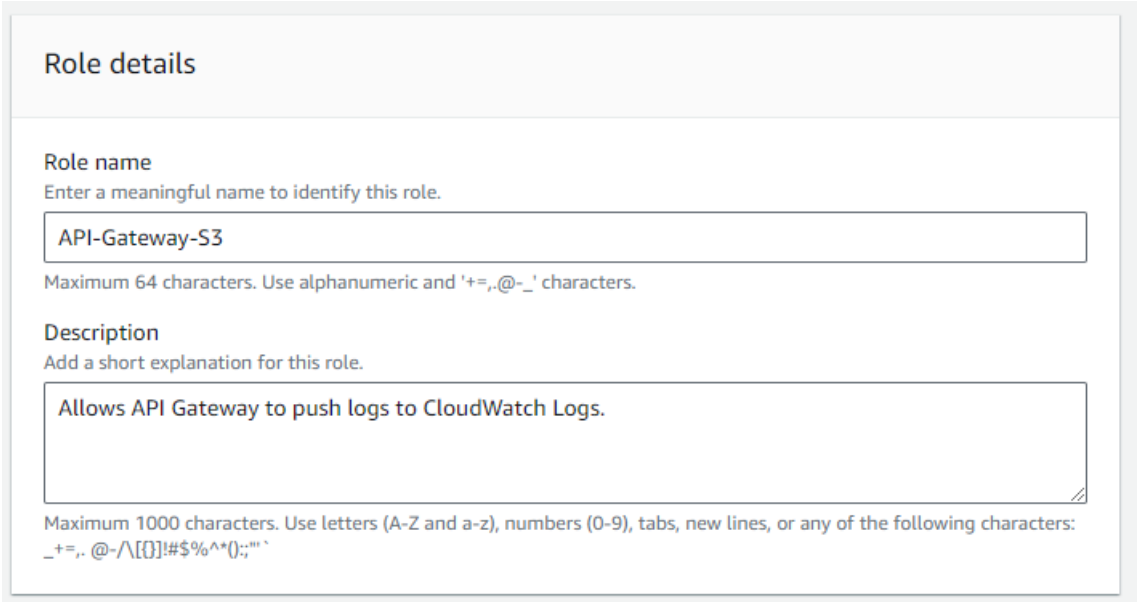

Figure 40 Role details [18]

Now our role is created as seen on the below screenshot:

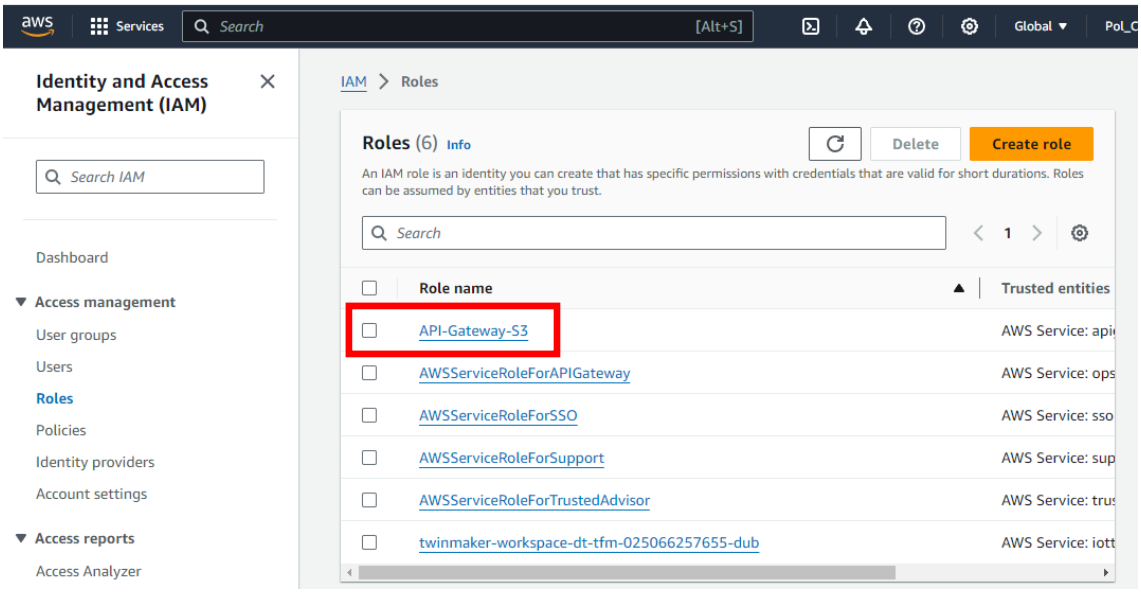

Figure 41 Roles outcome [18]

3.1.3 IAM Policy Creation

We then go to "Policies" in the lateral menu, and "Create policy"

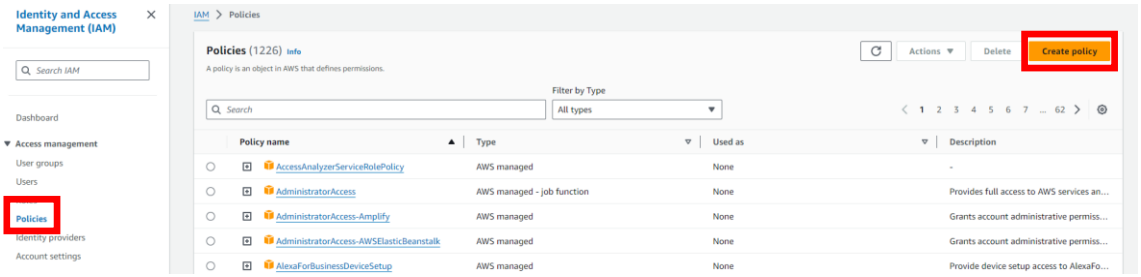

Figure 42 Policy menu [18]

On next screen we search for "S3" service

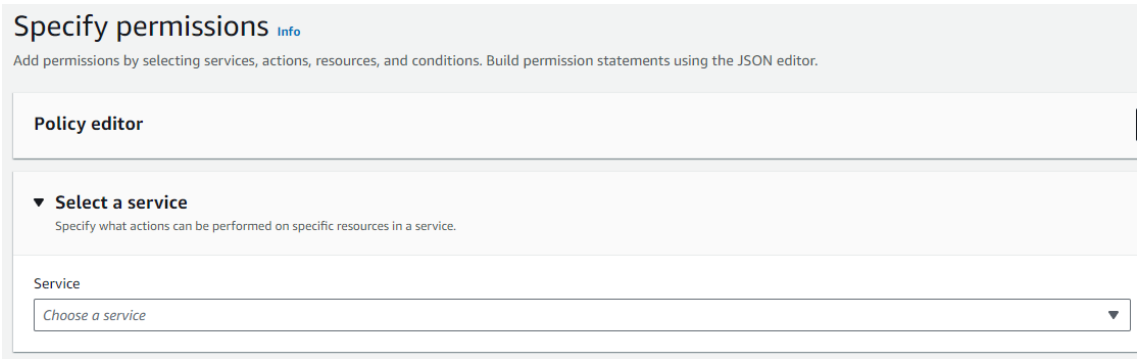

Figure 43 Policies service [18]

And add action "PutObject" using the searchbar. Once PutObject is selected the screen will be shown as below and we will move to "Add ARNs".

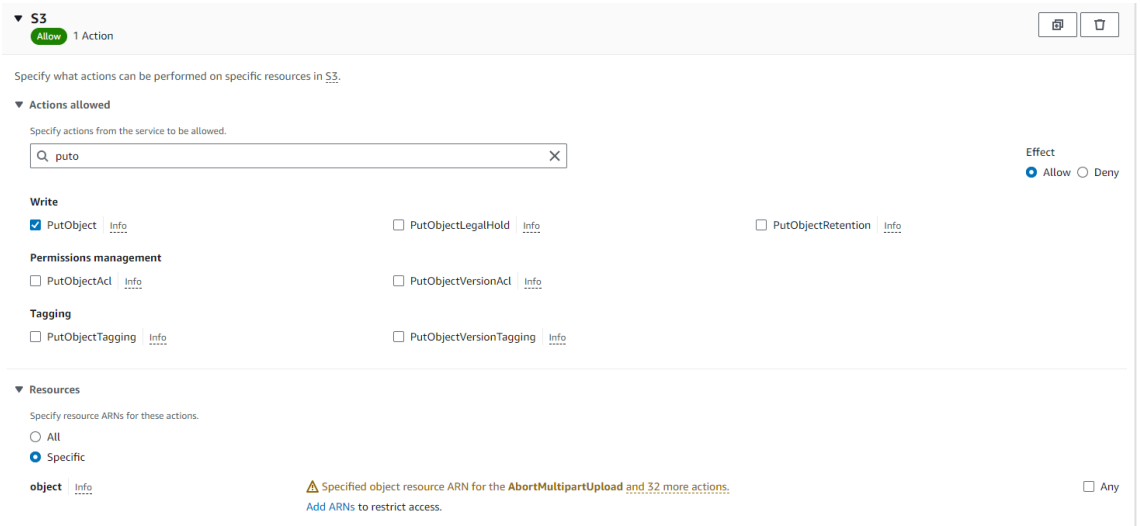

Figure 44 Policy configuration [18]

The following window will pop-up and this must be filled with our bucket name ("esp32dataloggercanbucket") and marking "Any object name". Finally "Add ARNs".

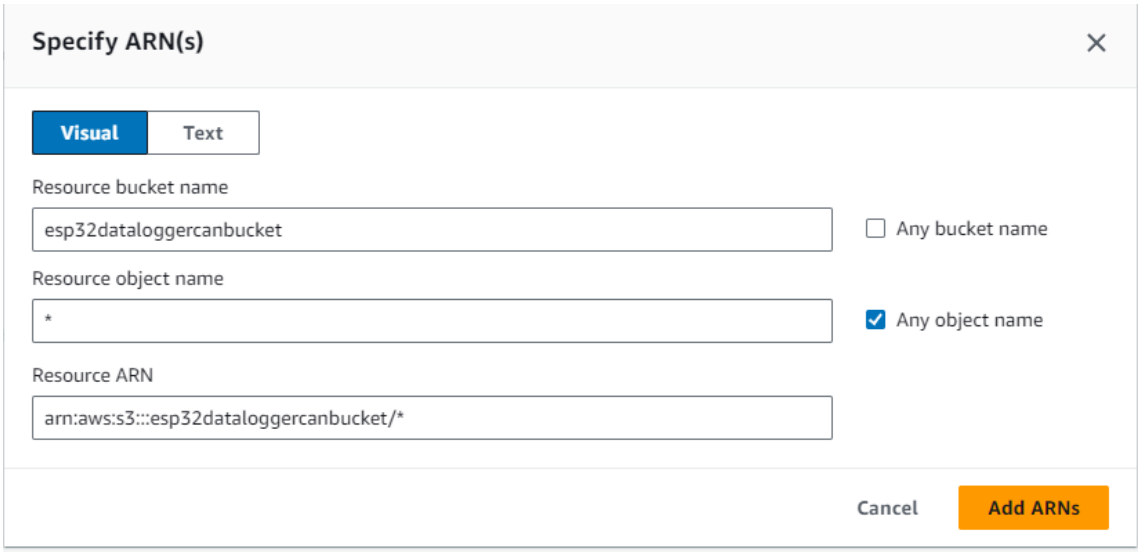

Figure 45 Policy link to S3 bucket [18]

Once the ARN is done, we can click next, we will enter the last step window. We enter the policy name "s3-bucket_policy_CAN" and click "Create policy"

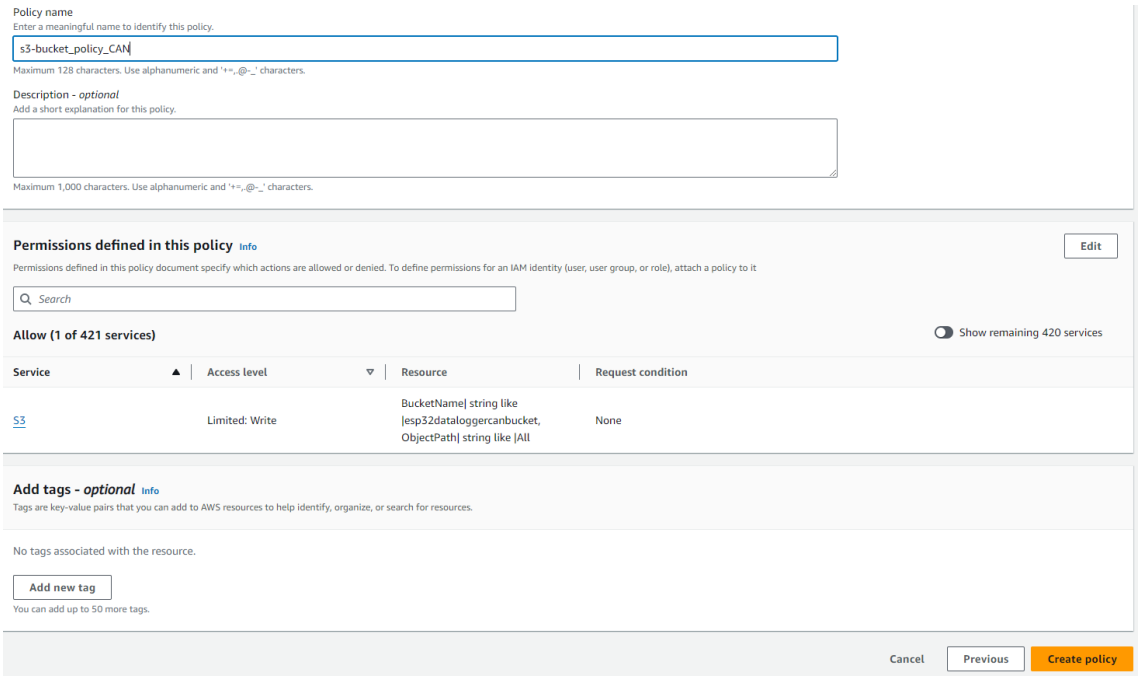

Figure 46 Policy creation [18]

Now that our policy is created, we will return to the policies main screen search for our policy, select it, click "Actions" and "Attach" with the intention to link it to our previously created role.

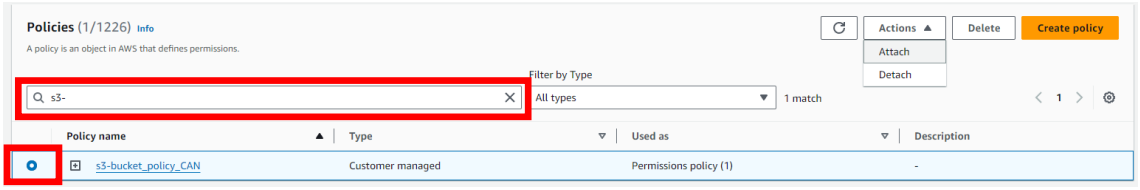

Figure 47 Policies main screen search [18]

On next screen we select our previous role "api-gateway-s3" and click "Attach policy".

Now before moving forward, save the arn direction that can be found on the api gateway role:

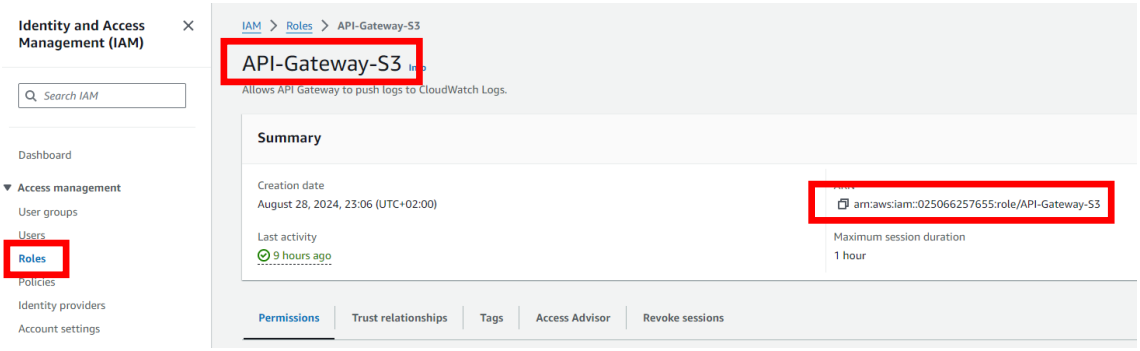

Figure 48 Roles ARN detail [18]

In our case: "*arn:aws:iam::025066257655:role/API-Gateway-S3*"

3.1.4 API Gateway Creation

To continue let's search on the search bar "API Gateway" and click on "Create API"

Figure 49 API Gateway menu [18]

On the next menu will select "REST API" and "Build" and the configuration will be as follows:

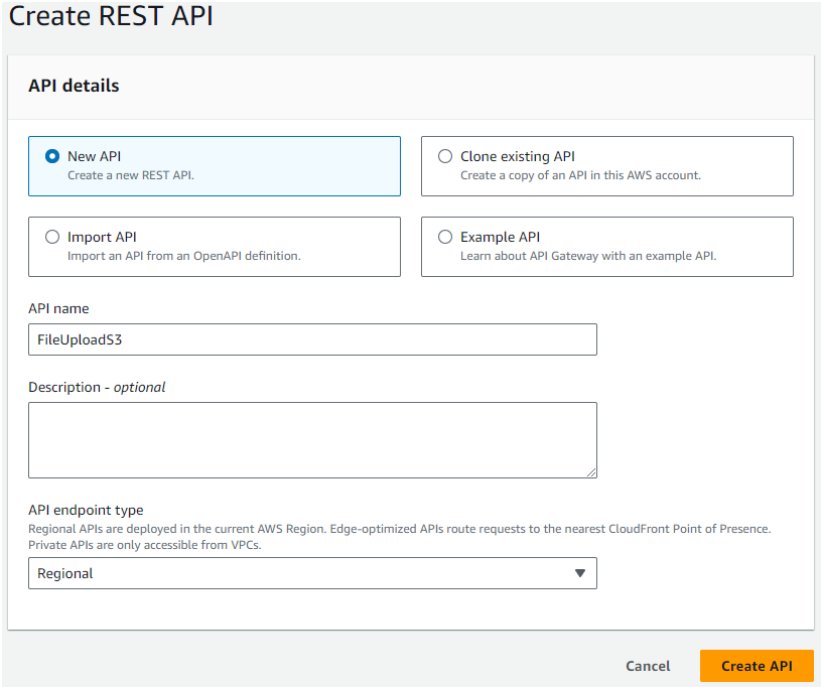

Figure 50 REST API creation [18]

Once the API is created we will click it.

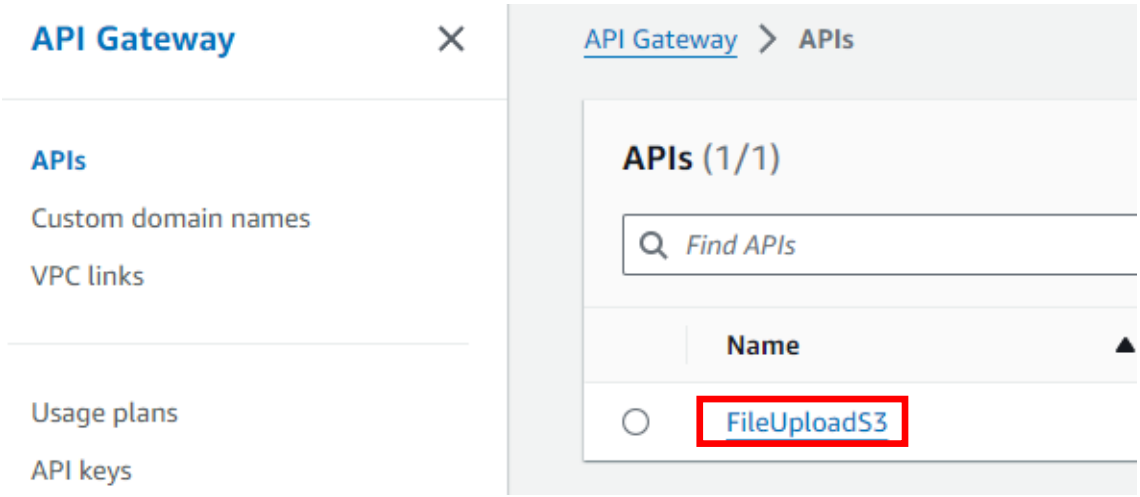

Figure 51 API search [18]

This will lead us to the following window and then will create a resource:

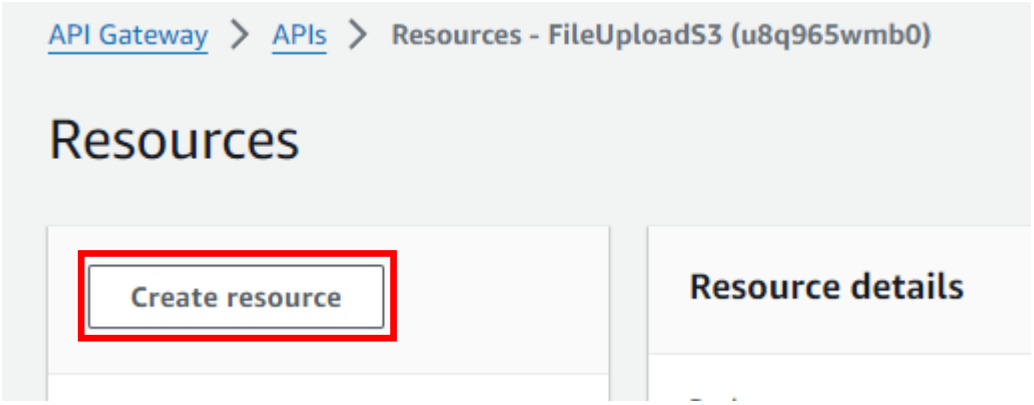

Figure 52: API resources [18]

Now we create our first resource, the bucket, the configuration must be as follows, and it's important to put the brackets otherwise it will not work. To finish we click on "Create resource"

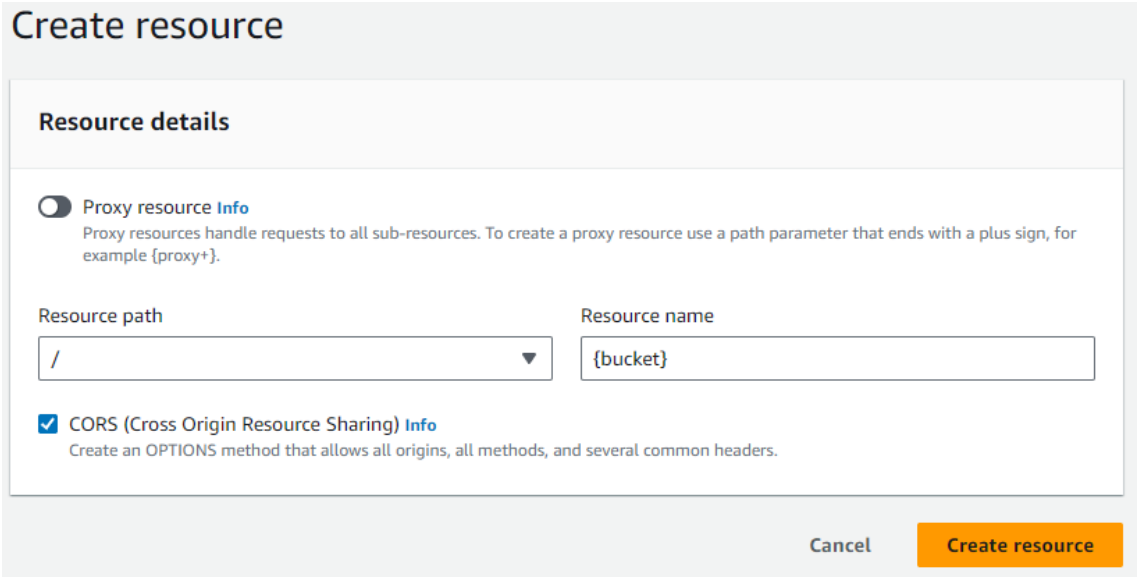

Figure 53 Bucket resource path [18]

Now that our resource has been created we will see it on our API menu as can be seen on the below image. So will click on bucket and create resource

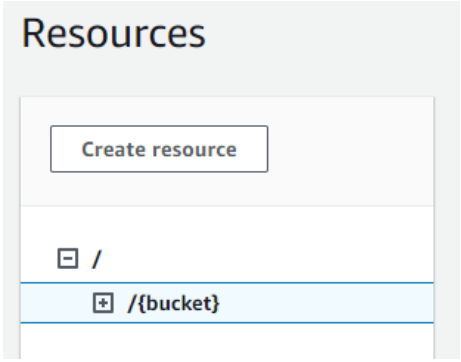

Figure 54 Bucket resource path outcome [18]

The following screen will appear, and we fill it with {filename}

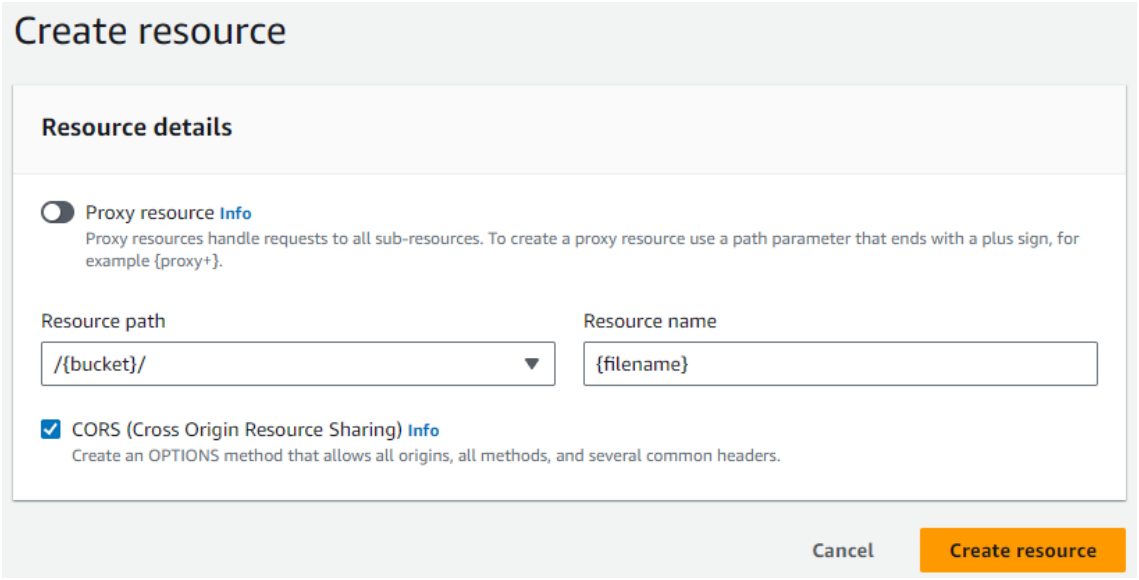

Figure 55 Filename resource path [18]

Now we can see the filename created, we will select it and click on "Create method"

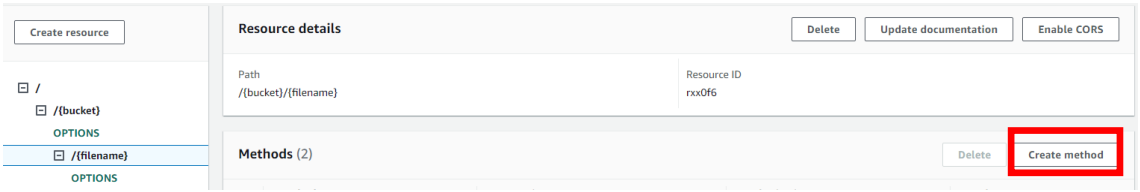

Figure 56 Method overview [18]

We will configure the method as follows and click on "Create method"

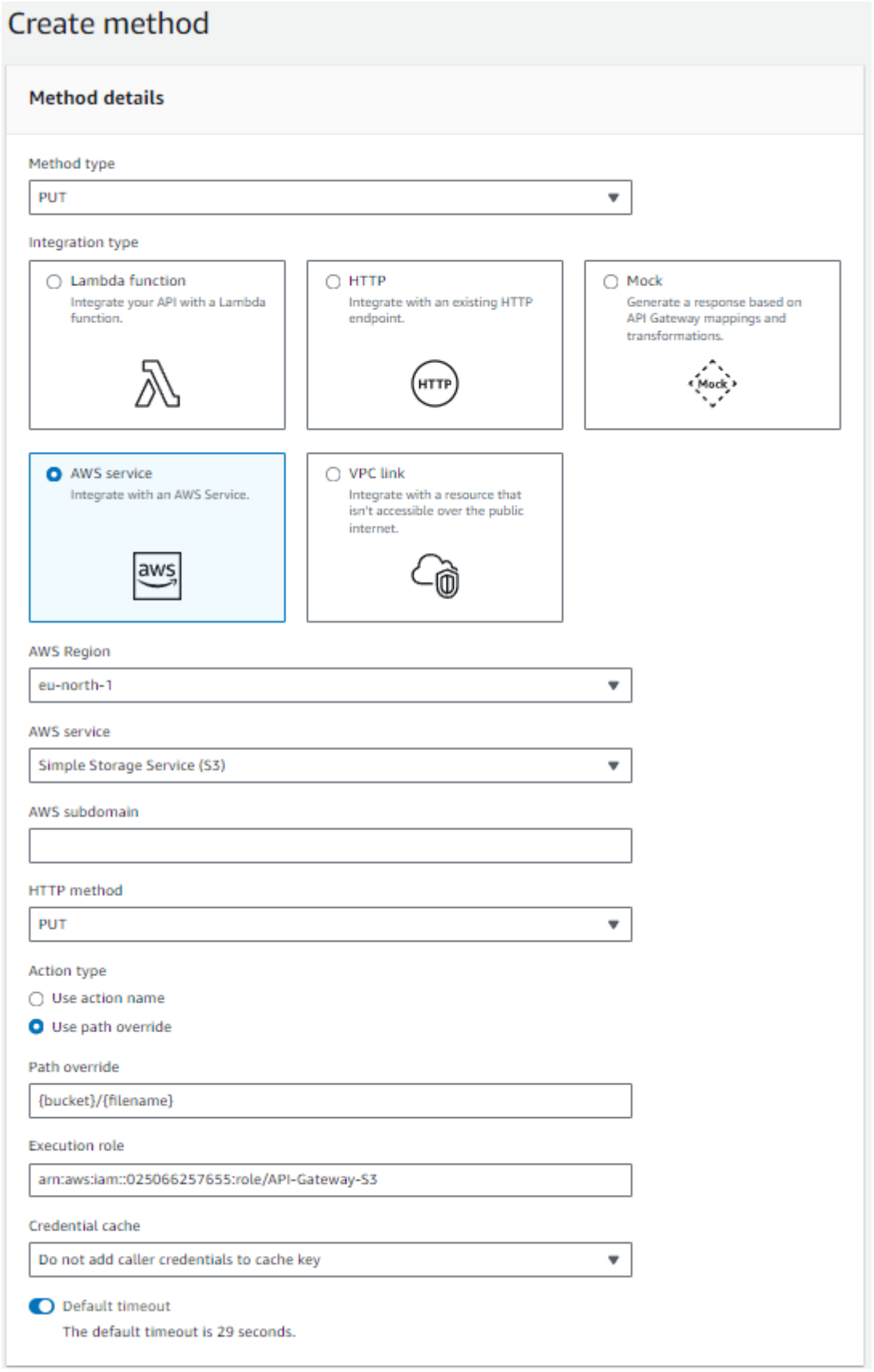

Figure 57 Method configuration [18]

If we continue to scroll down, we need to fill "URL path parameters" with the following:

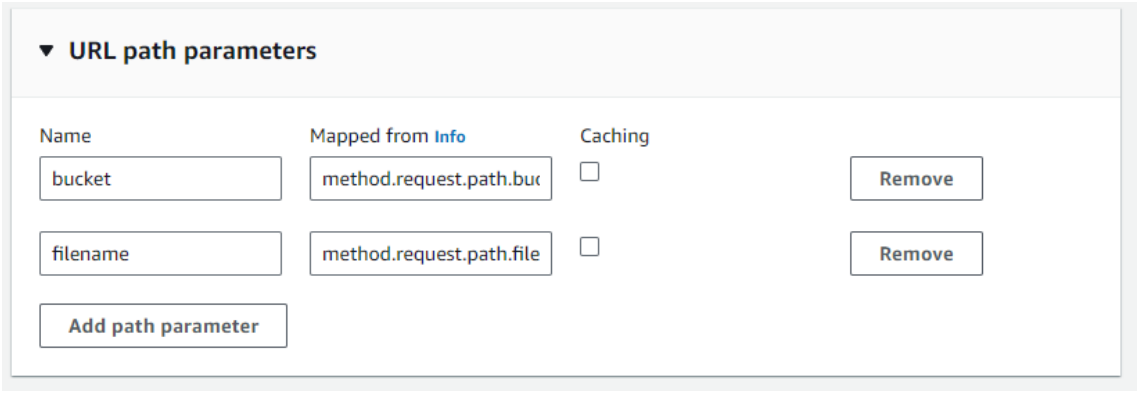

Figure 58 Method URL parameters [18]

Finally we will configure the type of files accepted by this API by going to "API Settings" "Binary media types" and "Manage media types":

Figure 59 API settings [18]

We will input "*/*" that means that all the files are accepted, and "Save changes".

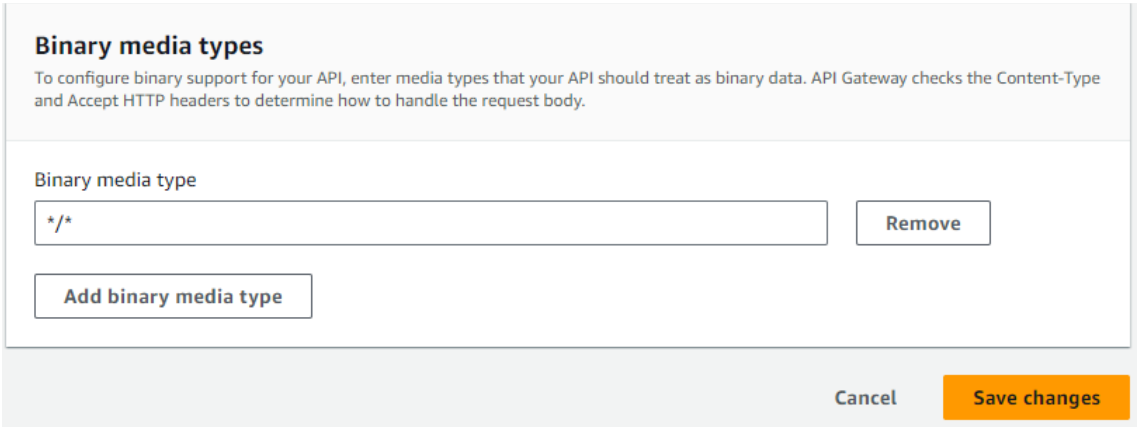

Figure 60 File types authorization [18]

To finish, we will return to the API main screen and click on "Deploy API".

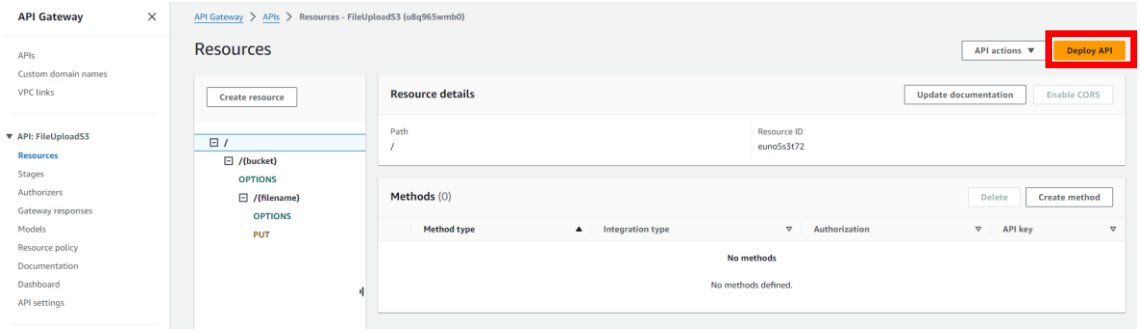

Figure 61 API deployment [18]

A pop-up screen will appear. We fill it with as seen below and click "Deploy":

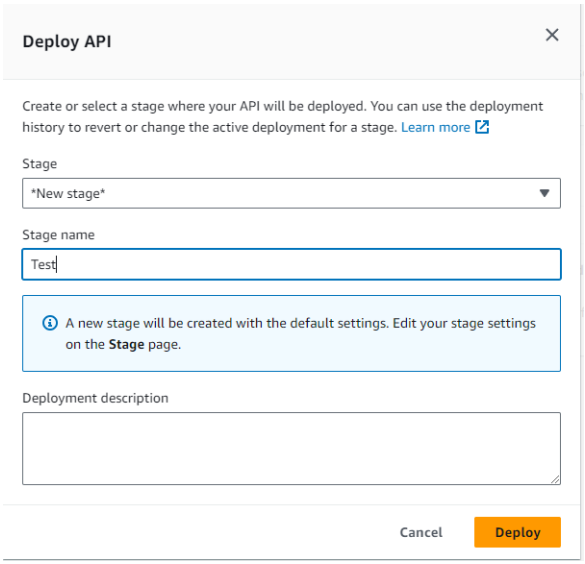

Figure 62 API Deployment stage [18]

The deployment will take a minute or so, but before being able to test it we go to the PUT method and we copy the URL:

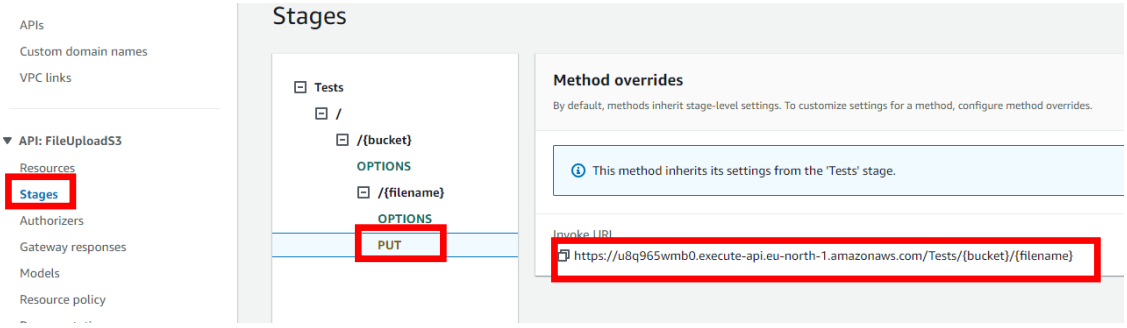

Figure 63 API URL [18]

As this URL is then entered into our code

```
    // URL del API Gateway
    const char* apiEndpoint = "u8q965wmb0.execute-api.eu-north-
1.amazonaws.com";
    const int httpsPort = 443;
    const String apiPath = "/Tests/esp32dataloggercanbucket";
```

4 Results

The initial goal was to make the system work offline, so make sure that the CAN and GPS data was stored properly. Once this major milestone was achieved the sending of the data to AWS was handled.

4.1 GPS

For simple refresh rate the GPS data was easier and less critical and as the module search automatically the signal, and just provide a serial data string with full or empty GPS data.

Here there are some examples of GPS data:

No GPS data:

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1233	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
2243	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
3251	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
4259	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
5269	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
6282	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
7290	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
8298	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
9308	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
10317	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
11325	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
12340	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
13349	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
14364	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
15373	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
16381	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
17389	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
18397	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
19406	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
20414	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
21422	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
22430	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48
23438	3									$GPGGA,,,,,,0,00,99.99,,,,,,*48

Figure 64 GPS data input without satellite connection

GPS Data

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1168	3									$GPGGA,170340.00,4123.26430,N,00136.55698,E,1,07,1.50,304.4,M,49.5,M,,*51
2181	3									$GPGGA,170341.00,4123.26445,N,00136.55735,E,1,07,1.50,304.4,M,49.5,M,,*54
3194	3									$GPGGA,170342.00,4123.26448,N,00136.55743,E,1,07,1.50,304.5,M,49.5,M,,*5A
4207	3									$GPGGA,170343.00,4123.26447,N,00136.55730,E,1,07,1.50,304.5,M,49.5,M,,*50
5220	3									$GPGGA,170344.00,4123.26406,N,00136.55697,E,1,07,1.50,304.5,M,49.5,M,,*5E
6233	3									$GPGGA,170345.00,4123.26405,N,00136.55696,E,1,07,1.50,304.5,M,49.5,M,,*5D
7246	3									$GPGGA,170346.00,4123.26400,N,00136.55683,E,1,07,1.50,304.6,M,49.5,M,,*5C
8259	3									$GPGGA,170347.00,4123.26410,N,00136.55685,E,1,07,1.50,304.5,M,49.5,M,,*59
9272	3									$GPGGA,170348.00,4123.26422,N,00136.55691,E,1,07,1.50,304.5,M,49.5,M,,*52
10285	3									$GPGGA,170349.00,4123.26447,N,00136.55711,E,1,07,1.50,304.5,M,49.5,M,,*59
11298	3									$GPGGA,170350.00,4123.26453,N,00136.55717,E,1,07,1.50,304.5,M,49.5,M,,*52
12311	3									$GPGGA,170351.00,4123.26411,N,00136.55675,E,1,07,1.50,304.5,M,49.5,M,,*50

Figure 65 GPS data input with satellite connection

4.2 CAN

Next step was to check that CAN was working, here the setup was a little bit more complex. First we need a CAN transceiver. In our case a Peak CAN copy; this interfaces with the original PCAN-VIEW program. It's like the original certified tool but at much lower price.

Using PCAN-View we can connect to our can transceiver and send a CAN message to our datalogger. This is the setup seen on figure 66 and in the screen there is the message that is going to be send over and over again.

First steps were the reception of the message.

Second, send the message automatically every 100ms, 10ms, 2 ms

Finally top speed. The PCAN-View maximum speed is 1ms per message.

This is the case that led me to create the buffers to handle the data as the writing of the SD card is slower and cannot keep up with the CAN so I was losing CAN frames, and this is one of the requirements of the system, not lose any CAN frames.

Once the buffer is implemented, and with a capability of 30ms of data at 1ms rate there is no loose of information as the writing of the buffer to the MicroSD card is done in around 12ms

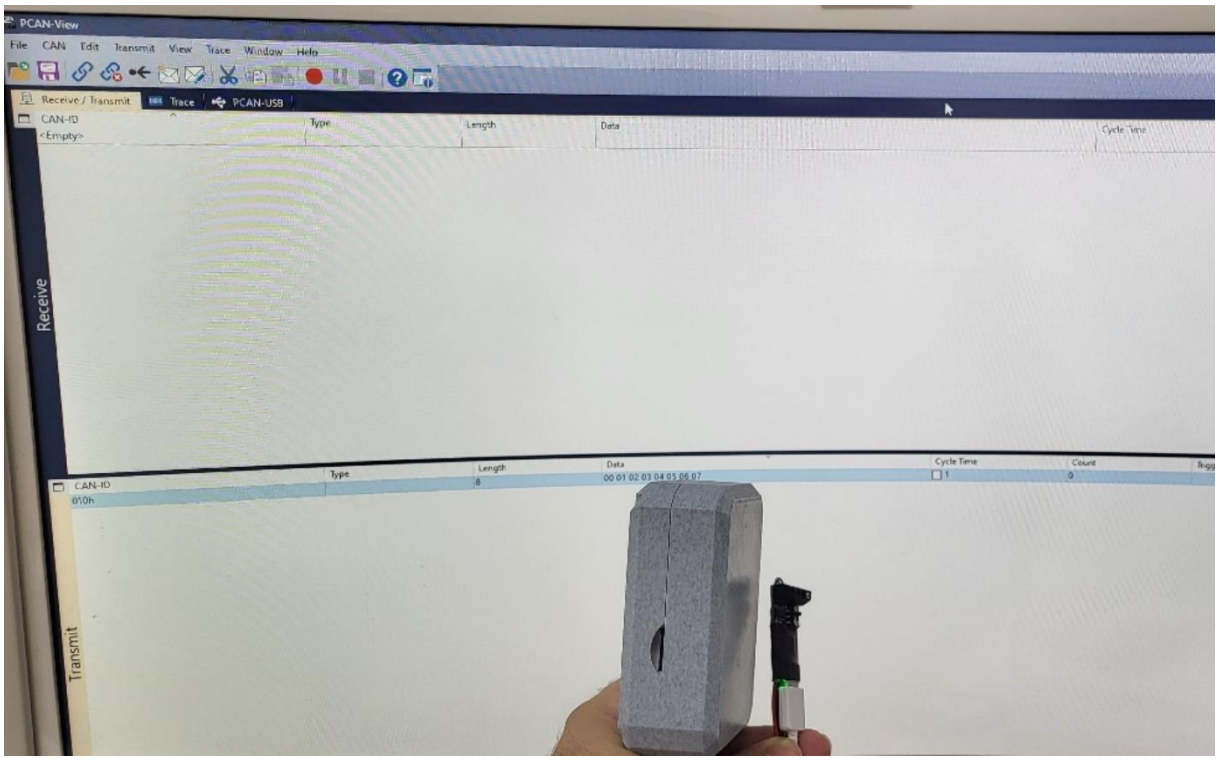

Figure 66 PCAN-View + PCAN copy + CAN datalogger

Here there is a screenshoots of the recorded data at 1 ms:

	A	B	C	D	E	F	G	H	I	J	
1	Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	G
2	172	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
3	174	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
4	174	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
5	175	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
6	176	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
7	177	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
8	178	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
9	179	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
10	179	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
11	181	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
12	182	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
13	183	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
14	184	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
15	185	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
16	186	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
17	187	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
18	188	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
19	189	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
20	190	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
21	191	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
22	192	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
23	193	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
24	194	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
25	195	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
26	196	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
27	197	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
28	198	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
29	199	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
30	200	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
31	202	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	
32	202	1	0x10	0	8	8	0	0	0	00 01 02 03 04 05 06 07	

Figure 67 PCAN-View injected data 1 ms

To finish we connected the datalogger to our car using a OBDII cable

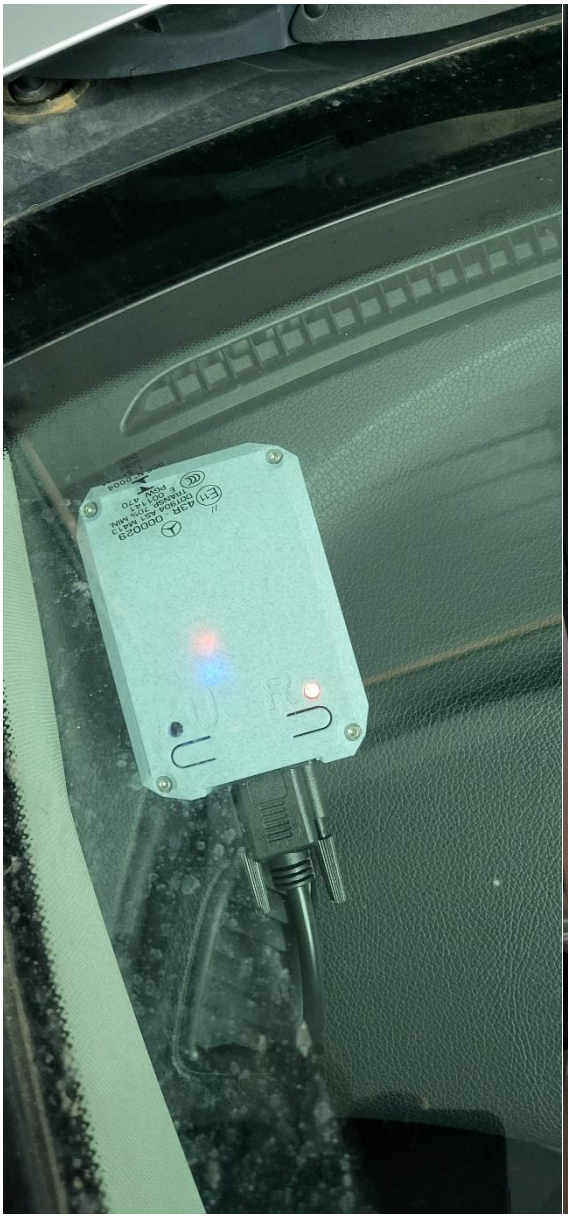

Figure 68 Datalogger from the windshield

Figure 69 Datalogger from Inside the car

And recorded the following data:

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538260	3									$GPGGA,161648.00,4121.04572,N,00142.31739,E,1,04,1.82,212.9,M,49.5,M,,*55
1538260	1	0x87	0	8	8	0	0	0	FB 0C 00 00 00 40 10 38	
1538262	1	0xb3	0	8	8	0	0	0	2E 07 0E 00 FF 00 00 20	
1538264	1	0x141	0	8	8	0	0	0	00 07 00 00 3C 61 F9 6B	
1538265	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1538270	1	0xa1	0	8	8	0	0	0	DE AD 37 42 18 37 C2 7E	
1538272	1	0xa3	0	8	8	0	0	0	06 AD 38 42 C6 35 C2 43	
1538273	1	0x21b	0	8	8	0	0	0	0A 0D 20 02 8D 33 80 7F	
1538274	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1538275	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1538277	1	0xae	0	8	8	0	0	0	16 45 A2 01 17 00 A4 14	
1538278	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538279	1	0x469	0	8	8	0	0	0	C8 86 C8 B2 21 00 04 00	
1538280	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538281	1	0x87	0	8	8	0	0	0	A4 0D 00 00 00 40 10 38	
1538282	1	0xb3	0	8	8	0	0	0	52 08 0E 00 FF 00 00 20	
1538284	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1538289	1	0x14d	0	8	8	0	0	0	00 1C 00 00 20 00 FF FF	
1538290	1	0xa1	0	8	8	0	0	0	82 AE 38 42 21 35 C2 87	
1538292	1	0xa3	0	8	8	0	0	0	05 AE 37 42 CF 35 C2 4C	
1538292	1	0x21b	0	8	8	0	0	0	76 0E 20 02 8D 32 80 7F	
1538294	1	0x149	0	2	2	0	0	0	46 02	
1538295	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538296	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1538297	1	0xae	0	8	8	0	0	0	6F 46 A2 01 13 00 A4 10	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538299	1	0x14b	0	8	8	0	0	0	02 37 32 A4 23 00 80 00	
1538300	1	0x20	0	8	8	0	0	0	DD CE 41 0C 00 00 00 00	
1538301	1	0x87	0	8	8	0	0	0	DC 1E 00 00 00 40 10 38	
1538302	1	0xb3	0	8	8	0	0	0	38 09 0E 00 FF 00 00 20	
1538304	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1538309	1	0x377	0	8	8	0	0	0	C0 03 FF FF FF F3 3F 00	
1538310	1	0x15a	0	8	8	0	0	0	49 72 55 55 E0 03 0F 00	
1538311	1	0xa1	0	8	8	0	0	0	05 AF 37 42 2A 35 C2 90	
1538312	1	0xa3	0	8	8	0	0	0	DD AF 36 42 D8 34 C2 55	
1538313	1	0x21b	0	8	8	0	0	0	4F 0F 20 02 8D 43 80 7F	
1538314	1	0x22b	0	8	8	0	0	0	B3 0A 00 00 00 00 00 60	
1538315	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1538316	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	
1538317	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538318	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 71 81	
1538319	1	0xae	0	8	8	0	0	0	8E 47 A2 01 10 00 A4 10	
1538320	1	0x122	0	8	8	0	0	0	87 21 04 0A 00 00 00 00	
1538321	1	0x87	0	8	8	0	0	0	E1 5F 01 00 06 40 10 38	
1538322	1	0xb3	0	8	8	0	0	0	86 0A 0E 00 FF 00 00 20	
1538324	1	0x1e1	0	8	8	0	0	0	E4 08 00 F9 08 01 00 00	
1538325	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1538326	1	0x225	0	8	8	0	0	0	AC 0B 24 00 00 80 00 00	
1538329	1	0xd5	0	8	8	0	0	0	00 00 32 00 21 01 00 00	
1538330	1	0xa1	0	8	8	0	0	0	FF A0 36 42 33 35 C2 99	
1538331	1	0x351	0	8	8	0	0	0	11 2E 54 AF FF FF F1 00	
1538332	1	0xa3	0	8	8	0	0	0	22 A0 38 42 E1 34 C2 5E	
1538333	1	0x21b	0	8	8	0	0	0	B7 00 20 02 8D 40 80 7F	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538334	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 A0 4E	
1538335	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538336	1	0xb1	0	8	8	0	0	0	33 09 7B 00 00 84 07 1F	
1538337	1	0xae	0	8	8	0	0	0	9D 48 A2 01 0F 00 A4 0C	
1538338	1	0x209	0	8	8	0	0	0	63 45 BF BE 2D 1B 0D 5F	
1538339	1	0x2f0	0	8	8	0	0	0	BC 05 00 FF B6 05 04 00	
1538340	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538341	1	0x87	0	8	8	0	0	0	E4 50 10 00 0C 40 10 38	
1538342	1	0xb3	0	8	8	0	0	0	EC 0B 0E 00 FF 00 00 20	
1538344	1	0x3d	0	8	8	0	0	0	D4 07 00 00 00 00 00 FF	
1538350	1	0xa1	0	8	8	0	0	0	0D A1 36 42 3D 32 C2 A2	
1538352	1	0xa3	0	8	8	0	0	0	0B A1 36 42 EA 32 C2 67	
1538352	1	0x21b	0	8	8	0	0	0	53 01 20 02 8D 4A 80 7F	
1538354	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF 34 04	
1538355	1	0xb1	0	8	8	0	0	0	8D 0B 7B 00 00 84 07 1F	
1538356	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538357	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538358	1	0xae	0	8	8	0	0	0	51 49 A2 01 0C 00 A4 08	
1538360	1	0x87	0	8	8	0	0	0	2B 51 1A 00 12 40 10 38	
1538362	1	0xb3	0	8	8	0	0	0	E7 0C 0E 00 FF 00 00 20	
1538364	1	0x141	0	8	8	0	0	0	00 07 00 00 3C 61 F9 6B	
1538365	1	0x3d	0	8	8	0	0	0	C2 09 00 00 00 00 00 FF	
1538370	1	0x46b	0	8	8	0	0	0	00 18 3C 50 FF 00 00 00	
1538371	1	0xa1	0	8	8	0	0	0	DE A2 35 42 46 31 C2 AB	
1538372	1	0xa3	0	8	8	0	0	0	EE A2 34 42 F3 32 C2 70	
1538373	1	0x21b	0	8	8	0	0	0	20 02 20 02 8D 3C 80 7F	
1538374	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538375	1	0xb1	0	8	8	0	0	0	52 0D 7B 00 00 84 07 1F	
1538377	1	0xae	0	8	8	0	0	0	FC 4A A2 01 02 00 A4 00	
1538378	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538379	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538380	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538381	1	0x87	0	8	8	0	0	0	89 52 1F 00 1A 40 10 38	
1538382	1	0xb3	0	8	8	0	0	0	8D 0D 0E 00 FF 00 00 20	
1538384	1	0x3d	0	8	8	0	0	0	16 0B 00 00 00 00 00 FF	
1538389	1	0x14d	0	8	8	0	0	0	00 24 00 00 28 00 FF FF	
1538390	1	0xa1	0	8	8	0	0	0	31 A3 35 42 4F 32 C2 B4	
1538392	1	0xa3	0	8	8	0	0	0	86 A3 35 42 FC 32 C2 79	
1538393	1	0x21b	0	8	8	0	0	0	18 03 20 02 8D 44 80 7F	
1538394	1	0x149	0	2	2	0	0	0	46 02	
1538395	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538396	1	0xb1	0	8	8	0	0	0	EC 0F 7B 00 00 84 07 1F	
1538397	1	0xae	0	8	8	0	0	0	77 4B A2 01 00 00 A4 00	
1538400	1	0x20	0	8	8	0	0	0	40 CF 41 0C 00 00 00 00	
1538401	1	0x87	0	8	8	0	0	0	6A 53 22 00 20 40 10 38	
1538402	1	0x351	0	8	8	0	0	0	11 2E 54 AF AF EA F1 00	
1538403	1	0xb3	0	8	8	0	0	0	33 0E 0E 00 FF 00 00 20	
1538404	1	0x3d	0	8	8	0	0	0	77 0D 00 00 00 00 00 FF	
1538410	1	0x15a	0	8	8	0	0	0	49 72 55 55 E8 03 0F 00	
1538411	1	0xa1	0	8	8	0	0	0	A5 A4 36 42 58 32 C2 BD	
1538412	1	0x369	0	8	8	0	0	0	00 00 00 00 00 00 00 00	
1538413	1	0xa3	0	8	8	0	0	0	07 A4 34 42 06 32 C2 82	
1538414	1	0x21b	0	8	8	0	0	0	05 04 20 02 8D 45 80 7F	
1538415	1	0x22b	0	8	8	0	0	0	D9 0B 00 00 00 00 00 60	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538416	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1538417	1	0xb1	0	8	8	0	0	0	F1 01 7B 00 00 84 07 1F	
1538418	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538419	1	0xae	0	8	8	0	0	0	F7 4C A2 01 00 00 A4 00	
1538420	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 70 81	
1538421	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538422	1	0x68	0	1	1	0	0	0	02	
1538423	1	0x122	0	8	8	0	0	0	87 21 04 0A 00 00 00 00	
1538424	1	0x87	0	8	8	0	0	0	8C 54 24 00 24 40 10 38	
1538425	1	0xb3	0	8	8	0	0	0	59 0F 0E 00 FF 00 00 20	
1538426	1	0x1e1	0	8	8	0	0	0	A8 09 00 B5 09 01 00 00	
1538427	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1538428	1	0x225	0	8	8	0	0	0	2C 0C 24 00 00 80 00 00	
1538430	1	0xa1	0	8	8	0	0	0	D7 A5 36 42 61 31 C2 C6	
1538432	1	0xa3	0	8	8	0	0	0	FA A5 33 42 0F 30 C2 8B	
1538433	1	0x21b	0	8	8	0	0	0	A0 05 20 02 8D 3C 80 7F	
1538434	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 EC 4F	
1538435	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538436	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1538437	1	0xae	0	8	8	0	0	0	A8 4D A2 01 00 00 A4 00	
1538438	1	0x209	0	8	8	0	0	0	1A 0D 19 06 F4 03 29 1F	
1538440	1	0x87	0	8	8	0	0	0	11 55 24 00 28 40 10 38	
1538442	1	0xb3	0	8	8	0	0	0	25 00 0E 00 FF 00 00 20	
1538444	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1538450	1	0xa1	0	8	8	0	0	0	DE A6 35 42 6A 2F C2 CF	
1538452	1	0xa3	0	8	8	0	0	0	2A A6 34 42 18 2F C2 94	
1538453	1	0x21b	0	8	8	0	0	0	26 06 20 02 8D 44 80 7F	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538454	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF 17 04	
1538455	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1538456	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538457	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538458	1	0xae	0	8	8	0	0	0	49 4E A2 01 00 00 A4 00	
1538459	1	0x14b	0	8	8	0	0	0	02 38 32 A4 23 00 80 00	
1538460	1	0x87	0	8	8	0	0	0	45 56 24 00 2C 40 10 38	
1538462	1	0xb3	0	8	8	0	0	0	4F 01 0E 00 FF 00 00 20	
1538464	1	0x141	0	8	8	0	0	0	00 07 00 00 3C 61 F9 69	
1538465	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1538470	1	0xa1	0	8	8	0	0	0	B2 A7 34 42 73 31 C2 D8	
1538472	1	0xa3	0	8	8	0	0	0	51 A7 34 42 21 2F C2 9D	
1538472	1	0x21b	0	8	8	0	0	0	8C 07 20 02 8D 4A 80 7F	
1538474	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1538475	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	
1538477	1	0xae	0	8	8	0	0	0	16 4F A2 01 00 00 A4 00	
1538478	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538479	1	0x469	0	8	8	0	0	0	C8 86 C8 B2 21 00 04 00	
1538480	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538481	1	0x87	0	8	8	0	0	0	CE 57 24 00 2E 40 10 38	
1538482	1	0xb3	0	8	8	0	0	0	F1 02 0E 00 FF 00 00 20	
1538484	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1538489	1	0x14d	0	8	8	0	0	0	00 2C 00 00 30 00 FF FF	
1538490	1	0xa1	0	8	8	0	0	0	48 A8 33 42 7C 30 C2 E1	
1538492	1	0xa3	0	8	8	0	0	0	1B A8 33 42 2A 2F C2 A6	
1538492	1	0x21b	0	8	8	0	0	0	81 08 20 02 8D 47 80 7F	
1538494	1	0x149	0	2	2	0	0	0	46 02	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538495	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538496	1	0xb1	0	8	8	0	0	0	33 09 7B 00 00 84 07 1F	
1538497	1	0xae	0	8	8	0	0	0	54 40 A2 01 00 00 A4 00	
1538499	1	0x14b	0	8	8	0	0	0	02 38 32 A4 23 00 80 00	
1538500	1	0x20	0	8	8	0	0	0	28 C0 41 0C 00 00 00 00	
1538501	1	0x87	0	8	8	0	0	0	29 58 25 00 30 40 10 38	
1538502	1	0xb3	0	8	8	0	0	0	9B 03 0E 00 FF 00 00 20	
1538504	1	0x3d	0	8	8	0	0	0	D4 07 00 00 00 00 00 FF	
1538510	1	0x15a	0	8	8	0	0	0	49 72 55 55 E0 03 0F 00	
1538511	1	0xa1	0	8	8	0	0	0	C7 A9 33 42 85 2E C2 EA	
1538512	1	0xa3	0	8	8	0	0	0	3C A9 33 42 33 2E C2 AF	
1538513	1	0x21b	0	8	8	0	0	0	F8 09 20 02 8D 4C 80 7F	
1538514	1	0x22b	0	8	8	0	0	0	D2 0C 00 00 00 00 00 60	
1538515	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1538516	1	0xb1	0	8	8	0	0	0	8D 0B 7B 00 00 84 07 1F	
1538517	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538518	1	0xae	0	8	8	0	0	0	0B 41 A2 01 00 00 A4 00	
1538520	1	0x122	0	8	8	0	0	0	87 21 04 0A 00 00 00 00	
1538521	1	0x87	0	8	8	0	0	0	24 59 26 00 34 40 10 38	
1538522	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 6E 81	
1538523	1	0xb3	0	8	8	0	0	0	90 04 0E 00 FF 00 00 20	
1538524	1	0x1e1	0	8	8	0	0	0	7C 0A 00 61 0A 01 00 00	
1538525	1	0x3d	0	8	8	0	0	0	C2 09 00 00 00 00 00 FF	
1538526	1	0x225	0	8	8	0	0	0	73 0D 24 00 00 80 00 00	
1538529	1	0xd5	0	8	8	0	0	0	00 00 32 00 21 01 00 00	
1538530	1	0xa1	0	8	8	0	0	0	4D AA 31 42 8E 2D C2 F3	
1538532	1	0xa3	0	8	8	0	0	0	EC AA 31 42 3C 2D C2 B8	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538532	1	0x21b	0	8	8	0	0	0	73 0A 20 02 8D 51 80 7F	
1538534	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 4F 40	
1538535	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538536	1	0xb1	0	8	8	0	0	0	52 0D 7B 00 00 84 07 1F	
1538537	1	0xae	0	8	8	0	0	0	EA 42 A2 01 00 00 A4 00	
1538538	1	0x209	0	8	8	0	0	0	63 45 FF 1A CB 14 DE DF	
1538539	1	0x2f0	0	8	8	0	0	0	30 06 00 FF 0C 06 04 00	
1538540	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538541	1	0x87	0	8	8	0	0	0	4C 5A 27 00 36 40 10 38	
1538542	1	0xb3	0	8	8	0	0	0	FA 05 0E 00 FF 00 00 20	
1538544	1	0x3d	0	8	8	0	0	0	16 0B 00 00 00 00 00 FF	
1538550	1	0xa1	0	8	8	0	0	0	1F AB 30 42 97 2D C2 FC	
1538552	1	0xa3	0	8	8	0	0	0	BE AB 31 42 45 2C C2 C1	
1538553	1	0x21b	0	8	8	0	0	0	95 0B 20 02 8D 49 80 7F	
1538554	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF FE 03	
1538555	1	0xb1	0	8	8	0	0	0	EC 0F 7B 00 00 84 07 1F	
1538556	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538557	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538558	1	0xae	0	8	8	0	0	0	B5 43 A2 01 00 00 A4 00	
1538560	1	0x87	0	8	8	0	0	0	04 5B 29 00 38 40 10 38	
1538562	1	0xb3	0	8	8	0	0	0	44 06 0E 00 FF 00 00 20	
1538564	1	0x141	0	8	8	0	0	0	00 07 00 00 3C 61 F9 68	
1538565	1	0x3d	0	8	8	0	0	0	77 0D 00 00 00 00 00 FF	
1538570	1	0x46b	0	8	8	0	0	0	00 18 3C 50 FF 00 00 00	
1538571	1	0xa1	0	8	8	0	0	0	F2 AC 30 42 A0 2B C2 06	
1538572	1	0xa3	0	8	8	0	0	0	D0 AC 30 42 4E 2B C2 CA	
1538573	1	0x21b	0	8	8	0	0	0	32 0C 20 02 8D 4B 80 7F	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538574	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1538575	1	0xb1	0	8	8	0	0	0	F1 01 7B 00 00 84 07 1F	
1538577	1	0xae	0	8	8	0	0	0	35 44 A2 01 00 00 A4 00	
1538578	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538579	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538580	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538581	1	0x87	0	8	8	0	0	0	BC 5C 2A 00 3D 40 10 38	
1538582	1	0xb3	0	8	8	0	0	0	2E 07 0E 00 FF 00 00 20	
1538584	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1538589	1	0x14d	0	8	8	0	0	0	00 34 00 00 38 00 FF FF	
1538590	1	0xa1	0	8	8	0	0	0	64 AD 2F 42 A9 29 C2 0F	
1538592	1	0xa3	0	8	8	0	0	0	B7 AD 2E 42 57 2A C2 D2	
1538592	1	0x21b	0	8	8	0	0	0	F0 0D 20 02 8D 4A 80 7F	
1538594	1	0x149	0	2	2	0	0	0	46 02	
1538594	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538595	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1538597	1	0xae	0	8	8	0	0	0	6A 45 A2 01 00 00 A4 00	
1538600	1	0x20	0	8	8	0	0	0	B5 C1 41 0C 00 00 00 00	
1538601	1	0x87	0	8	8	0	0	0	13 5D 2B 00 44 40 10 38	
1538602	1	0x351	0	8	8	0	0	0	11 2E 54 AF FF FF F1 00	
1538603	1	0xb3	0	8	8	0	0	0	52 08 0E 00 FF 00 00 20	
1538604	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1538605	1	0x341	0	8	8	0	0	0	00 00 28 21 10 89 59 00	
1538610	1	0x15a	0	8	8	0	0	0	4D 7B D5 DD E8 03 0F 00	
1538611	1	0x335	0	8	8	0	0	0	03 C0 FC FF 01 00 00 00	
1538612	1	0xa1	0	8	8	0	0	0	08 AE 2E 42 B2 2A C2 18	
1538613	1	0xa3	0	8	8	0	0	0	DE AE 2C 42 60 2A C2 DB	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538614	1	0x21b	0	8	8	0	0	0	11 0E 20 02 8D 4A 80 7F	
1538615	1	0x22b	0	8	8	0	0	0	B8 0D 00 00 00 00 00 60	
1538616	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1538617	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1538618	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538619	1	0x409	0	8	8	0	0	0	6F 67 5F 61 00 00 10 00	
1538620	1	0xae	0	8	8	0	0	0	8B 46 A2 01 00 00 A4 00	
1538621	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538622	1	0x68	0	1	1	0	0	0	02	
1538623	1	0x122	0	8	8	0	0	0	87 21 04 0A 00 00 00 00	
1538625	1	0x87	0	8	8	0	0	0	E3 5E 2D 00 48 40 10 38	
1538626	1	0xb3	0	8	8	0	0	0	38 09 0E 00 FF 00 00 20	
1538627	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 6B 81	
1538628	1	0x1e1	0	8	8	0	0	0	30 0B 00 2D 0B 01 00 00	
1538629	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1538630	1	0x225	0	8	8	0	0	0	92 0E 24 00 00 80 00 00	
1538631	1	0x4af	0	4	4	0	0	0	0F 07 F8 3F	
1538632	1	0x2fb	0	8	8	0	0	0	55 19 31 00 C9 D7 98 28	
1538633	1	0x337	0	4	4	0	0	0	50 0F 00 00	
1538634	1	0xa1	0	8	8	0	0	0	11 AF 2D 42 BB 28 C2 20	
1538635	1	0xa3	0	8	8	0	0	0	D9 AF 2C 42 69 28 C2 E4	
1538636	1	0x21b	0	8	8	0	0	0	41 0F 20 02 8D 3D 80 7F	
1538637	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 03 41	
1538638	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538639	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	
1538640	1	0xae	0	8	8	0	0	0	D4 47 A2 01 00 00 A4 00	
1538641	1	0x209	0	8	8	0	0	0	A2 0D 59 06 9C 03 29 1F	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538642	1	0x2f1	0	8	8	0	0	0	5F 11 00 A0 76 00 30 00	
1538643	1	0x87	0	8	8	0	0	0	5B 5F 2E 00 48 40 10 38	
1538644	1	0xb3	0	8	8	0	0	0	86 0A 0E 00 FF 00 00 20	
1538645	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1538647	1	0x35d	0	4	4	0	0	0	23 00 70 00	
1538649	1	0x37f	0	8	8	0	0	0	40 20 8C 87 2A 58 FF FF	
1538650	1	0xa1	0	8	8	0	0	0	6E A0 2E 42 C4 28 C2 29	
1538652	1	0xa3	0	8	8	0	0	0	4C A0 2B 42 72 27 C2 ED	
1538652	1	0x21b	0	8	8	0	0	0	9E 00 20 02 8D 3C 80 7F	
1538654	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF DB 03	
1538655	1	0xb1	0	8	8	0	0	0	33 09 7B 00 00 84 07 1F	
1538656	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538657	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538658	1	0xae	0	8	8	0	0	0	96 48 A2 01 00 00 A4 00	
1538659	1	0x14b	0	8	8	0	0	0	02 39 32 A4 23 00 80 00	
1538660	1	0x87	0	8	8	0	0	0	4F 50 2F 00 4F 40 10 38	
1538662	1	0xb3	0	8	8	0	0	0	EC 0B 0E 00 FF 00 00 20	
1538664	1	0x141	0	8	8	0	0	0	00 07 00 00 3C 61 F9 67	
1538665	1	0x3d	0	8	8	0	0	0	D4 07 00 00 00 00 00 FF	
1538668	1	0x379	0	7	7	0	0	0	FF FF FF FF FF FF 0F	
1538670	1	0xa1	0	8	8	0	0	0	F1 A1 2D 42 CD 28 C2 32	
1538672	1	0xa3	0	8	8	0	0	0	69 A1 2A 42 7B 27 C2 F6	
1538673	1	0x21b	0	8	8	0	0	0	E6 01 20 02 8D 3E 80 7F	
1538674	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1538675	1	0xb1	0	8	8	0	0	0	8D 0B 7B 00 00 84 07 1F	
1538677	1	0xae	0	8	8	0	0	0	C9 49 A2 01 00 00 A4 00	
1538677	1	0x4b8	0	8	8	0	0	0	FF FF FF FF FF FF 0F 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538678	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538679	1	0x469	0	8	8	0	0	0	C8 86 C8 B2 21 00 04 00	
1538680	1	0x2f3	0	8	8	0	0	0	5B 11 00 04 00 00 00 00	
1538681	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538682	1	0x87	0	8	8	0	0	0	B8 51 30 00 55 40 10 38	
1538683	1	0xb3	0	8	8	0	0	0	E7 0C 0E 00 FF 00 00 20	
1538684	1	0x3d	0	8	8	0	0	0	C2 09 00 00 00 00 00 FF	
1538689	1	0x353	0	8	8	0	0	0	07 07 83 FE F5 F5 99 55	
1538689	1	0x14d	0	8	8	0	0	0	00 3C 00 FF FF 00 FF 03	
1538690	1	0xa1	0	8	8	0	0	0	03 A2 29 42 D5 26 C2 3B	
1538692	1	0xa3	0	8	8	0	0	0	D7 A2 2B 42 84 24 C2 00	
1538692	1	0x21b	0	8	8	0	0	0	94 02 20 02 8D 41 80 7F	
1538694	1	0x149	0	2	2	0	0	0	46 02	
1538695	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538696	1	0xb1	0	8	8	0	0	0	52 0D 7B 00 00 84 07 1F	
1538697	1	0xae	0	8	8	0	0	0	28 4A A2 01 00 00 A4 00	
1538699	1	0x14b	0	8	8	0	0	0	02 39 32 A4 23 00 80 00	
1538700	1	0x34b	0	1	1	0	0	0	00	
1538701	1	0x20	0	8	8	0	0	0	0F C2 41 0C 00 00 00 00	
1538702	1	0x87	0	8	8	0	0	0	AA 52 2F 00 5B 40 10 38	
1538703	1	0xb3	0	8	8	0	0	0	8D 0D 0E 00 FF 00 00 20	
1538704	1	0x3d	0	8	8	0	0	0	16 0B 00 00 00 00 00 FF	
1538710	1	0x15a	0	8	8	0	0	0	69 72 5D 55 E0 03 0F 00	
1538711	1	0xa1	0	8	8	0	0	0	A7 A3 29 42 DE 25 C2 44	
1538712	1	0xa3	0	8	8	0	0	0	66 A3 2A 42 8D 23 C2 08	
1538713	1	0x21b	0	8	8	0	0	0	57 03 20 02 8D 49 80 7F	
1538714	1	0x22b	0	8	8	0	0	0	06 0E 00 00 00 00 00 60	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538715	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1538716	1	0xb1	0	8	8	0	0	0	EC 0F 7B 00 00 84 07 1F	
1538717	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538718	1	0xae	0	8	8	0	0	0	77 4B A2 01 00 00 A4 00	
1538719	1	0x357	0	8	8	0	0	0	30 FF C0 FF 00 00 00 00	
1538720	1	0x122	0	8	8	0	0	0	87 21 04 0A 00 00 00 00	
1538721	1	0x87	0	8	8	0	0	0	AD 53 2F 00 63 40 10 38	
1538722	1	0x2f5	0	8	8	0	0	0	F6 51 30 90 51 12 00 00	
1538723	1	0xb3	0	8	8	0	0	0	33 0E 0E 00 FF 00 00 20	
1538724	1	0x1e1	0	8	8	0	0	0	C9 0C 00 D4 0C 01 00 00	
1538725	1	0x3d	0	8	8	0	0	0	77 0D 00 00 00 00 00 FF	
1538726	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 68 81	
1538727	1	0x225	0	8	8	0	0	0	CD 0F 24 00 00 80 00 00	
1538728	1	0x2c4	0	8	8	0	0	0	0F 80 4D 2A 29 08 95 01	
1538729	1	0xd5	0	8	8	0	0	0	00 00 32 00 21 01 00 00	
1538730	1	0xa1	0	8	8	0	0	0	A8 A4 2A 42 E7 25 C2 4C	
1538732	1	0xa3	0	8	8	0	0	0	D9 A4 28 42 96 23 C2 11	
1538732	1	0x21b	0	8	8	0	0	0	6E 04 20 02 8D 51 80 7F	
1538734	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 D7 42	
1538735	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538736	1	0xb1	0	8	8	0	0	0	F1 01 7B 00 00 84 07 1F	
1538737	1	0xae	0	8	8	0	0	0	F7 4C A2 01 00 00 A4 00	
1538738	1	0x209	0	8	8	0	0	0	DA 44 FF 06 BB 04 59 1F	
1538739	1	0x2f0	0	8	8	0	0	0	BF 07 00 FF 91 07 04 00	
1538740	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538741	1	0x87	0	8	8	0	0	0	8E 54 2F 00 69 40 10 38	
1538742	1	0xb3	0	8	8	0	0	0	59 0F 0E 00 FF 00 00 20	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538744	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1538750	1	0x33f	0	8	8	0	0	0	00 5A 00 00 0F 55 02 00	
1538751	1	0xa1	0	8	8	0	0	0	96 A5 26 42 F0 24 C2 55	
1538752	1	0xa3	0	8	8	0	0	0	99 A5 27 42 9E 20 C2 1A	
1538753	1	0x21b	0	8	8	0	0	0	7E 05 20 02 8D 5C 80 7F	
1538754	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF B2 03	
1538755	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1538756	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538757	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538758	1	0xae	0	8	8	0	0	0	A8 4D A2 01 00 00 A4 00	
1538760	1	0x2f7	0	8	8	0	0	0	3B 51 03 2A 0F 23 20 1B	
1538761	1	0x87	0	8	8	0	0	0	B0 55 2F 00 6F 40 10 38	
1538762	1	0xb3	0	8	8	0	0	0	25 00 0E 00 FF 00 00 20	
1538764	1	0x141	0	8	8	0	0	0	00 08 00 00 3C 91 F9 5F	
1538765	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1538768	1	0x3ed	0	8	8	0	0	0	B7 03 66 08 FF FF FF FF	
1538770	1	0x46b	0	8	8	0	0	0	00 18 3C 50 FF 00 00 00	
1538771	1	0xa1	0	8	8	0	0	0	6A A6 25 42 F9 22 C2 5E	
1538772	1	0xa3	0	8	8	0	0	0	C0 A6 26 42 A7 20 C2 23	
1538773	1	0x21b	0	8	8	0	0	0	03 06 20 02 8D 54 80 7F	
1538774	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 45 7C 86 18	
1538775	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1538777	1	0xae	0	8	8	0	0	0	49 4E A2 01 00 00 A4 00	
1538778	1	0x4bb	0	1	1	0	0	0	00	
1538779	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538780	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538781	1	0x71	0	8	8	0	0	0	00 57 44 44 32 30 35 32	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538782	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538783	1	0x87	0	8	8	0	0	0	7D 56 2F 00 73 40 10 38	
1538784	1	0xb3	0	8	8	0	0	0	4F 01 0E 00 FF 00 00 20	
1538785	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1538789	1	0x14d	0	8	8	0	0	0	00 04 45 00 08 57 FF FF	
1538790	1	0x3e7	0	1	1	0	0	0	02	
1538791	1	0xa1	0	8	8	0	0	0	F2 A7 25 42 03 1F C2 67	
1538792	1	0xa3	0	8	8	0	0	0	13 A7 23 42 B0 1F C2 2B	
1538793	1	0x21b	0	8	8	0	0	0	E5 07 20 02 8D 4C 80 7F	
1538794	1	0x149	0	2	2	0	0	0	46 02	
1538795	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538796	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	
1538797	1	0xae	0	8	8	0	0	0	16 4F A2 01 00 00 A4 00	
1538800	1	0x20	0	8	8	0	0	0	92 C3 41 0C 00 00 00 00	
1538801	1	0x71	0	8	8	0	0	0	01 34 37 31 46 32 36 39	
1538802	1	0x87	0	8	8	0	0	0	72 57 30 00 77 40 10 38	
1538803	1	0x351	0	8	8	0	0	0	11 2E 54 AF AF EA F1 00	
1538804	1	0xb3	0	8	8	0	0	0	F1 02 0E 00 FF 00 00 20	
1538805	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1538809	1	0x377	0	8	8	0	0	0	C0 03 FF FF FF F3 3F 00	
1538810	1	0x15a	0	8	8	0	0	0	49 72 55 55 E8 03 0F 00	
1538811	1	0x341	0	8	8	0	0	0	00 00 28 20 10 89 59 00	
1538812	1	0xa1	0	8	8	0	0	0	E7 A8 24 42 0B 1D C2 6F	
1538813	1	0xa3	0	8	8	0	0	0	C6 A8 21 42 B9 1D C2 34	
1538814	1	0x21b	0	8	8	0	0	0	3A 08 20 02 8D 4D 80 7F	
1538815	1	0x22b	0	8	8	0	0	0	6C 0F 00 00 00 00 00 60	
1538816	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538817	1	0xb1	0	8	8	0	0	0	33 09 7B 00 00 84 07 1F	
1538818	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538819	1	0xae	0	8	8	0	0	0	54 40 A2 01 00 00 A4 00	
1538820	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538821	1	0x68	0	1	1	0	0	0	02	
1538822	1	0x71	0	8	8	0	0	0	02 31 34 36 FF FF FF FF	
1538823	1	0x122	0	8	8	0	0	0	87 20 04 0A 00 00 00 00	
1538824	1	0x87	0	8	8	0	0	0	A4 58 31 00 7C 40 10 38	
1538825	1	0xb3	0	8	8	0	0	0	9B 03 0E 00 FF 00 00 20	
1538826	1	0x1e1	0	8	8	0	0	0	85 0D 00 98 0D 01 00 00	
1538827	1	0x3d	0	8	8	0	0	0	D4 07 00 00 00 00 00 FF	
1538828	1	0x225	0	8	8	0	0	0	8F 00 24 00 00 80 00 00	
1538829	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 65 81	
1538830	1	0x4bd	0	1	1	0	0	0	03	
1538831	1	0x3ef	0	8	8	0	0	0	E8 07 08 16 10 10 30 11	
1538832	1	0xa1	0	8	8	0	0	0	53 A9 21 42 14 1D C2 78	
1538833	1	0xa3	0	8	8	0	0	0	15 A9 20 42 C1 1D C2 3D	
1538834	1	0x21b	0	8	8	0	0	0	B7 09 20 02 8D 41 80 7F	
1538835	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 9B 43	
1538836	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538837	1	0xb1	0	8	8	0	0	0	8D 0B 7B 00 00 84 07 1F	
1538838	1	0xae	0	8	8	0	0	0	0B 41 A2 01 00 00 A4 00	
1538839	1	0x209	0	8	8	0	0	0	12 0E 99 06 F4 03 29 1F	
1538840	1	0x3f7	0	8	8	0	0	0	E4 FF FF FF FF FF FF FF	
1538841	1	0x3a4	0	8	8	0	0	0	00 00 00 00 00 00 00 00	
1538842	1	0x87	0	8	8	0	0	0	F3 59 32 00 83 40 10 38	
1538843	1	0xb3	0	8	8	0	0	0	90 04 0E 00 FF 00 00 20	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538844	1	0x37f	0	8	8	0	0	0	40 20 8C 86 2A 58 FF FF	
1538845	1	0x3d	0	8	8	0	0	0	C2 09 00 00 00 00 00 FF	
1538850	1	0xa1	0	8	8	0	0	0	32 AA 21 42 1D 1B C2 81	
1538852	1	0xa3	0	8	8	0	0	0	F2 AA 1E 42 CA 1B C2 45	
1538852	1	0x21b	0	8	8	0	0	0	C5 0A 20 02 8D 3E 80 7F	
1538854	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF 82 03	
1538855	1	0xb1	0	8	8	0	0	0	52 0D 7B 00 00 84 07 1F	
1538856	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538857	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538858	1	0xae	0	8	8	0	0	0	EA 42 A2 01 00 00 A4 00	
1538859	1	0x3f3	0	8	8	0	0	0	8C 12 0B 15 04 00 00 00	
1538860	1	0x14b	0	8	8	0	0	0	02 3A 32 A4 23 00 80 00	
1538861	1	0x87	0	8	8	0	0	0	59 5A 33 00 8D 40 10 38	
1538862	1	0xb3	0	8	8	0	0	0	FA 05 0E 00 FF 00 00 20	
1538864	1	0x141	0	8	8	0	0	0	00 08 00 00 3C 91 F9 57	
1538865	1	0x3d	0	8	8	0	0	0	16 0B 00 00 00 00 00 FF	
1538870	1	0xa1	0	8	8	0	0	0	AB AB 20 42 26 19 C2 89	
1538872	1	0xa3	0	8	8	0	0	0	D0 AB 1C 42 D3 18 C2 4E	
1538873	1	0x21b	0	8	8	0	0	0	2E 0B 20 02 8D 43 80 7F	
1538874	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1538875	1	0xb1	0	8	8	0	0	0	EC 0F 7B 00 00 84 07 1F	
1538877	1	0xae	0	8	8	0	0	0	B5 43 A2 01 00 00 A4 00	
1538878	1	0x3a2	0	8	8	0	0	0	00 00 00 00 00 00 00 00	
1538879	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538880	1	0x469	0	8	8	0	0	0	C8 86 C8 B2 21 00 04 00	
1538881	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538882	1	0x87	0	8	8	0	0	0	7B 5B 34 00 94 40 10 38	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538883	1	0xb3	0	8	8	0	0	0	44 06 0E 00 FF 00 00 20	
1538884	1	0x3d	0	8	8	0	0	0	77 0D 00 00 00 00 00 FF	
1538889	1	0x14d	0	8	8	0	0	0	00 0C 03 00 10 00 FF FF	
1538890	1	0xa1	0	8	8	0	0	0	CB AC 1D 42 2E 18 C2 92	
1538892	1	0xa3	0	8	8	0	0	0	94 AC 1D 42 DB 18 C2 56	
1538892	1	0x21b	0	8	8	0	0	0	E0 0C 20 02 8D 47 80 7F	
1538894	1	0x149	0	2	2	0	0	0	46 02	
1538895	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538896	1	0xb1	0	8	8	0	0	0	F1 01 7B 00 00 84 07 1F	
1538897	1	0xae	0	8	8	0	0	0	35 44 A2 01 00 00 A4 00	
1538899	1	0x14b	0	8	8	0	0	0	02 3A 32 A4 23 00 80 00	
1538900	1	0x20	0	8	8	0	0	0	66 C4 41 0C 00 00 00 00	
1538901	1	0x87	0	8	8	0	0	0	95 5C 36 00 96 40 10 38	
1538902	1	0xb3	0	8	8	0	0	0	2E 07 0E 00 FF 00 00 20	
1538904	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1538907	1	0x407	0	8	8	0	0	0	56 56 57 56 5E 6B 00 00	
1538910	1	0x15a	0	8	8	0	0	0	49 72 55 55 E0 0B 2F 00	
1538911	1	0xa1	0	8	8	0	0	0	86 AD 1B 42 37 17 C2 9A	
1538912	1	0x369	0	8	8	0	0	0	00 00 00 00 00 00 00 00	
1538913	1	0xa3	0	8	8	0	0	0	26 AD 1C 42 E4 16 C2 5F	
1538914	1	0x21b	0	8	8	0	0	0	23 0D 20 02 8D 4F 80 7F	
1538915	1	0x22b	0	8	8	0	0	0	10 00 00 00 00 00 00 60	
1538916	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1538917	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1538918	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1538919	1	0xae	0	8	8	0	0	0	6A 45 A2 01 00 00 A4 00	
1538920	1	0x458	0	8	8	0	0	0	7C FF 7F FF FF FF 00 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538921	1	0x3f1	0	8	8	0	0	0	7A A9 67 5D 24 70 36 81	
1538922	1	0x122	0	8	8	0	0	0	87 20 04 0A 00 00 00 00	
1538923	1	0x87	0	8	8	0	0	0	E0 5D 37 00 9E 40 10 38	
1538924	1	0xb3	0	8	8	0	0	0	52 08 0E 00 FF 00 00 20	
1538925	1	0x1e1	0	8	8	0	0	0	51 0E 00 4C 0E 01 00 00	
1538926	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1538927	1	0x225	0	8	8	0	0	0	D0 01 24 00 00 80 00 00	
1538928	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 60 81	
1538929	1	0xd5	0	8	8	0	0	0	00 00 32 00 21 01 00 00	
1538930	1	0xa1	0	8	8	0	0	0	BD AE 1A 42 3F 15 C2 A3	
1538932	1	0xa3	0	8	8	0	0	0	16 AE 1A 42 ED 14 C2 68	
1538933	1	0x21b	0	8	8	0	0	0	5C 0E 20 02 8D 55 80 7F	
1538934	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 62 44	
1538935	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1538936	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1538937	1	0xae	0	8	8	0	0	0	8B 46 A2 01 00 00 A4 00	
1538938	1	0x209	0	8	8	0	0	0	D3 45 3F C2 4B 14 E7 5F	
1538939	1	0x2f0	0	8	8	0	0	0	39 08 00 FF F9 08 04 00	
1538940	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538941	1	0x87	0	8	8	0	0	0	6F 5E 38 00 A7 40 10 38	
1538942	1	0xb3	0	8	8	0	0	0	38 09 0E 00 FF 00 00 20	
1538944	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1538950	1	0xa1	0	8	8	0	0	0	8D AF 19 42 48 13 C2 AB	
1538952	1	0xa3	0	8	8	0	0	0	84 AF 18 42 F5 14 C2 70	
1538952	1	0x21b	0	8	8	0	0	0	D0 0F 20 02 8D 50 80 7F	
1538954	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF 4D 03	
1538955	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538956	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1538957	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1538958	1	0xae	0	8	8	0	0	0	D4 47 A2 01 00 00 A4 00	
1538959	1	0x3db	0	8	8	0	0	0	7A 0C 04 04 F3 03 59 1F	
1538960	1	0x87	0	8	8	0	0	0	93 5F 38 00 AD 40 10 38	
1538962	1	0xb3	0	8	8	0	0	0	86 0A 0E 00 FF 00 00 20	
1538964	1	0x141	0	8	8	0	0	0	00 08 00 00 3C 91 F9 4F	
1538965	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1538970	1	0x46b	0	8	8	0	0	0	00 18 3C 50 FF 00 00 00	
1538971	1	0xa1	0	8	8	0	0	0	0D A0 17 42 51 11 C2 B4	
1538972	1	0xa3	0	8	8	0	0	0	CB A0 16 42 FE 11 C2 79	
1538973	1	0x21b	0	8	8	0	0	0	40 00 20 02 8D 5C 80 7F	
1538974	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1538975	1	0xb1	0	8	8	0	0	0	33 09 7B 00 00 84 07 1F	
1538976	1	0x30b	0	8	8	0	0	0	12 10 2D 16 08 18 12 FF	
1538977	1	0xae	0	8	8	0	0	0	96 48 A2 01 00 00 A4 00	
1538978	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1538979	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1538980	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1538981	1	0x87	0	8	8	0	0	0	97 50 38 00 B0 40 10 38	
1538982	1	0xb3	0	8	8	0	0	0	EC 0B 0E 00 FF 00 00 20	
1538984	1	0x3d	0	8	8	0	0	0	D4 07 00 00 00 00 00 FF	
1538989	1	0x14d	0	8	8	0	0	0	00 14 00 00 18 00 FF FF	
1538990	1	0xa1	0	8	8	0	0	0	C5 A1 15 42 59 11 C2 BC	
1538992	1	0xa3	0	8	8	0	0	0	95 A1 15 42 07 0F C2 81	
1538992	1	0x21b	0	8	8	0	0	0	83 01 20 02 8D 54 80 7F	
1538994	1	0x149	0	2	2	0	0	0	46 02	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1538994	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1538995	1	0xb1	0	8	8	0	0	0	8D 0B 7B 00 00 84 07 1F	
1538997	1	0xae	0	8	8	0	0	0	C9 49 A2 01 00 00 A4 00	
1539000	1	0x20	0	8	8	0	0	0	FB C5 41 0C 00 00 00 00	
1539001	1	0x87	0	8	8	0	0	0	21 51 37 00 B4 40 10 38	
1539002	1	0x351	0	8	8	0	0	0	11 2E 54 AF FF FF F1 00	
1539003	1	0xb3	0	8	8	0	0	0	E7 0C 0E 00 FF 00 00 20	
1539004	1	0x3d	0	8	8	0	0	0	C2 09 00 00 00 00 00 FF	
1539007	1	0x3eb	0	8	8	0	0	0	4A AC 67 5D 6A 70 36 81	
1539010	1	0x15a	0	8	8	0	0	0	49 72 55 55 E8 03 0F 00	
1539011	1	0xa1	0	8	8	0	0	0	DB A2 13 42 62 0D C2 C5	
1539012	1	0xa3	0	8	8	0	0	0	62 A2 12 42 10 0D C2 89	
1539013	1	0x21b	0	8	8	0	0	0	D8 02 20 02 8D 57 80 7F	
1539014	1	0x22b	0	8	8	0	0	0	7A 01 00 00 00 00 00 60	
1539015	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1539016	1	0xb1	0	8	8	0	0	0	52 0D 7B 00 00 84 07 1F	
1539017	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1539018	1	0x341	0	8	8	0	0	0	00 00 28 1F 10 89 59 00	
1539019	1	0xae	0	8	8	0	0	0	28 4A A2 01 00 00 A4 00	
1539020	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1539021	1	0x68	0	1	1	0	0	0	02	
1539022	1	0x122	0	8	8	0	0	0	87 1F 04 0A 00 00 00 00	
1539023	1	0x87	0	8	8	0	0	0	C0 52 37 00 B4 40 10 38	
1539024	1	0xb3	0	8	8	0	0	0	8D 0D 0E 00 FF 00 00 20	
1539025	1	0x1e1	0	8	8	0	0	0	1D 0F 00 00 0F 01 00 00	
1539026	1	0x3d	0	8	8	0	0	0	16 0B 00 00 00 00 00 FF	
1539027	1	0x225	0	8	8	0	0	0	31 02 24 00 00 80 00 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539030	1	0xa1	0	8	8	0	0	0	E7 A3 11 42 6A 0A C2 CD	
1539031	1	0x309	0	8	8	0	0	0	12 E6 15 88 0B 35 5B 81	
1539032	1	0xa3	0	8	8	0	0	0	46 A3 0F 42 18 0B C2 92	
1539033	1	0x21b	0	8	8	0	0	0	3D 03 20 02 8D 54 80 7F	
1539034	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 2E 45	
1539035	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1539036	1	0xb1	0	8	8	0	0	0	EC 0F 7B 00 00 84 07 1F	
1539037	1	0xae	0	8	8	0	0	0	77 4B A2 01 00 00 A4 00	
1539038	1	0x209	0	8	8	0	0	0	0E B5 61 1E 00 00 4E E0	
1539040	1	0x87	0	8	8	0	0	0	C2 53 36 00 B4 40 10 38	
1539042	1	0xb3	0	8	8	0	0	0	33 0E 0E 00 FF 00 00 20	
1539044	1	0x37f	0	8	8	0	0	0	40 20 8C 87 2A 58 FF FF	
1539044	1	0x3d	0	8	8	0	0	0	77 0D 00 00 00 00 00 FF	
1539050	1	0xa1	0	8	8	0	0	0	6C A4 0E 42 73 08 C2 D6	
1539052	1	0xa3	0	8	8	0	0	0	14 A4 0D 42 21 0B C2 9A	
1539052	1	0x21b	0	8	8	0	0	0	F3 04 20 02 8D 50 80 7F	
1539054	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF 14 03	
1539055	1	0xb1	0	8	8	0	0	0	F1 01 7B 00 00 84 07 1F	
1539056	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1539057	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1539058	1	0xae	0	8	8	0	0	0	F7 4C A2 01 00 00 A4 00	
1539059	1	0x14b	0	8	8	0	0	0	C2 38 32 A4 23 00 80 00	
1539060	1	0x87	0	8	8	0	0	0	CF 54 35 00 B5 40 10 38	
1539062	1	0xb3	0	8	8	0	0	0	59 0F 0E 00 FF 00 00 20	
1539064	1	0x141	0	8	8	0	0	0	00 08 00 00 3C 91 F9 47	
1539065	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1539070	1	0xa1	0	8	8	0	0	0	83 A5 0E 42 7B 09 C2 DE	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539072	1	0xa3	0	8	8	0	0	0	0C A5 0C 42 29 08 C2 A3	
1539073	1	0x21b	0	8	8	0	0	0	58 05 20 02 8D 57 80 7F	
1539074	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1539075	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1539077	1	0xae	0	8	8	0	0	0	A8 4D A2 01 00 00 A4 00	
1539078	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1539079	1	0x469	0	8	8	0	0	0	C8 86 C8 B2 21 00 04 00	
1539080	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1539081	1	0x87	0	8	8	0	0	0	90 55 35 00 B5 40 10 38	
1539082	1	0xb3	0	8	8	0	0	0	25 00 0E 00 FF 00 00 20	
1539084	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1539089	1	0x14d	0	8	8	0	0	0	00 1C 00 00 20 00 FF FF	
1539090	1	0xa1	0	8	8	0	0	0	75 A6 0C 42 83 05 C2 E6	
1539092	1	0xa3	0	8	8	0	0	0	B6 A6 0A 42 32 06 C2 AB	
1539093	1	0x21b	0	8	8	0	0	0	D3 06 20 02 8D 4A 80 7F	
1539094	1	0x149	0	2	2	0	0	0	46 02	
1539095	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1539096	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1539097	1	0xae	0	8	8	0	0	0	49 4E A2 01 00 00 A4 00	
1539099	1	0x14b	0	8	8	0	0	0	C2 38 32 A4 23 00 80 00	
1539100	1	0x20	0	8	8	0	0	0	41 C6 41 0C 00 00 00 00	
1539101	1	0x87	0	8	8	0	0	0	71 56 35 00 B5 40 10 38	
1539102	1	0xb3	0	8	8	0	0	0	4F 01 0E 00 FF 00 00 20	
1539104	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1539110	1	0x15a	0	8	8	0	0	0	4D 7B D5 DD E0 03 0F 00	
1539111	1	0x335	0	8	8	0	0	0	03 C0 FC FF 01 00 00 00	
1539112	1	0xa1	0	8	8	0	0	0	A5 A7 09 42 8C 04 C2 EE	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539113	1	0xa3	0	8	8	0	0	0	ED A7 08 42 3A 04 C2 B3	
1539114	1	0x21b	0	8	8	0	0	0	E6 07 20 02 8D 57 80 7F	
1539115	1	0x22b	0	8	8	0	0	0	C4 02 00 00 00 00 00 60	
1539116	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1539117	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	
1539118	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1539119	1	0xae	0	8	8	0	0	0	16 4F A2 01 00 00 A4 00	
1539120	1	0x122	0	8	8	0	0	0	87 1F 04 0A 00 00 00 00	
1539121	1	0x87	0	8	8	0	0	0	73 57 34 00 B5 40 10 38	
1539122	1	0xb3	0	8	8	0	0	0	F1 02 0E 00 FF 00 00 20	
1539124	1	0x1e1	0	8	8	0	0	0	BE 00 00 A3 00 01 00 00	
1539125	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1539126	1	0x225	0	8	8	0	0	0	6E 03 24 00 00 80 00 00	
1539127	1	0x4af	0	4	4	0	0	0	0F 07 F8 3F	
1539129	1	0xd5	0	8	8	0	0	0	00 00 32 00 21 01 00 00	
1539130	1	0x2fb	0	8	8	0	0	0	55 19 31 00 C9 D7 98 28	
1539131	1	0xa1	0	8	8	0	0	0	E9 A8 06 42 94 02 C2 F7	
1539132	1	0xa3	0	8	8	0	0	0	07 A8 07 42 42 02 C2 BB	
1539133	1	0x21b	0	8	8	0	0	0	EB 08 20 02 8D 5A 80 7F	
1539134	1	0x309	0	8	8	0	0	0	13 E6 15 88 0B 35 55 81	
1539135	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 FA 46	
1539136	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1539137	1	0xb1	0	8	8	0	0	0	33 09 7B 00 00 84 07 1F	
1539138	1	0xae	0	8	8	0	0	0	54 40 A2 01 00 00 A4 00	
1539139	1	0x209	0	8	8	0	0	0	E2 0E D9 06 9C 03 59 1F	
1539140	1	0x2f0	0	8	8	0	0	0	B6 09 00 FF 64 09 04 00	
1539141	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539142	1	0x87	0	8	8	0	0	0	31 58 34 00 B5 40 10 38	
1539143	1	0xb3	0	8	8	0	0	0	9B 03 0E 00 FF 00 00 20	
1539144	1	0x37f	0	8	8	0	0	0	40 20 8C 86 2A 58 FF FF	
1539145	1	0x3d	0	8	8	0	0	0	D4 07 00 00 00 00 00 FF	
1539147	1	0x35d	0	4	4	0	0	0	23 00 70 00	
1539150	1	0xa1	0	8	8	0	0	0	47 A9 04 42 9C 00 C2 00	
1539152	1	0xa3	0	8	8	0	0	0	0E A9 06 42 4A FF C1 C3	
1539152	1	0x21b	0	8	8	0	0	0	FD 09 20 02 8D 61 80 7F	
1539154	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF E6 02	
1539155	1	0xb1	0	8	8	0	0	0	8D 0B 7B 00 00 84 07 1F	
1539156	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	
1539157	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1539158	1	0xae	0	8	8	0	0	0	0B 41 A2 01 00 00 A4 00	
1539159	1	0x37f	0	8	8	0	0	0	40 20 8C 86 2A 58 FF FF	
1539160	1	0x87	0	8	8	0	0	0	E0 59 33 00 B5 40 10 38	
1539162	1	0xb3	0	8	8	0	0	0	90 04 0E 00 FF 00 00 20	
1539164	1	0x141	0	8	8	0	0	0	00 08 00 00 3C 91 F9 42	
1539165	1	0x3d	0	8	8	0	0	0	C2 09 00 00 00 00 00 FF	
1539170	1	0x46b	0	8	8	0	0	0	00 18 3C 50 FF 00 00 00	
1539171	1	0xa1	0	8	8	0	0	0	F7 AA 03 42 A5 FE C1 08	
1539172	1	0xa3	0	8	8	0	0	0	4B AA 02 42 53 FE C1 CC	
1539173	1	0x21b	0	8	8	0	0	0	53 0A 20 02 8D 6C 80 7F	
1539174	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1539175	1	0xb1	0	8	8	0	0	0	52 0D 7B 00 00 84 07 1F	
1539177	1	0xae	0	8	8	0	0	0	EA 42 A2 01 00 00 A4 00	
1539178	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1539179	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539180	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1539181	1	0x87	0	8	8	0	0	0	E6 5A 30 00 B5 40 10 38	
1539182	1	0xb3	0	8	8	0	0	0	FA 05 0E 00 FF 00 00 20	
1539184	1	0x3d	0	8	8	0	0	0	16 0B 00 00 00 00 00 FF	
1539189	1	0x14d	0	8	8	0	0	0	00 24 00 00 28 00 FF FF	
1539190	1	0xa1	0	8	8	0	0	0	05 AB 00 42 AD FB C1 10	
1539192	1	0xa3	0	8	8	0	0	0	B9 AB 01 42 5B FB C1 D4	
1539193	1	0x21b	0	8	8	0	0	0	FA 0B 20 02 8D 79 80 7F	
1539194	1	0x149	0	2	2	0	0	0	46 02	
1539195	1	0x21d	0	8	8	0	0	0	00 90 FF 64 32 4D F4 01	
1539196	1	0xb1	0	8	8	0	0	0	EC 0F 7B 00 00 84 07 1F	
1539197	1	0xae	0	8	8	0	0	0	B5 43 A2 01 00 00 A4 00	
1539200	1	0x20	0	8	8	0	0	0	DC C7 41 0C 00 00 00 00	
1539201	1	0x87	0	8	8	0	0	0	8F 5B 29 00 B5 40 10 38	
1539202	1	0x34b	0	1	1	0	0	0	00	
1539203	1	0x351	0	8	8	0	0	0	11 2E 54 AF AF EA F1 00	
1539204	1	0xb3	0	8	8	0	0	0	44 06 0E 00 FF 00 00 20	
1539205	1	0x3d	0	8	8	0	0	0	77 0D 00 00 00 00 00 FF	
1539210	1	0x15a	0	8	8	0	0	0	69 72 5D 55 E8 03 0F 00	
1539211	1	0xa1	0	8	8	0	0	0	B9 AC FE 41 B5 FA C1 18	
1539212	1	0xa3	0	8	8	0	0	0	DC AC 00 42 63 F8 C1 DC	
1539213	1	0x21b	0	8	8	0	0	0	78 0C 20 02 8D 6B 80 7F	
1539214	1	0x22b	0	8	8	0	0	0	AE 03 00 00 00 00 00 60	
1539215	1	0x147	0	8	8	0	0	0	00 00 FF 00 D6 02 E2 04	
1539216	1	0xb1	0	8	8	0	0	0	F1 01 7B 00 00 84 07 1F	
1539217	1	0x1f3	0	8	8	0	0	0	00 00 00 86 00 00 77 00	
1539218	1	0xae	0	8	8	0	0	0	35 44 A2 01 00 00 A4 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539219	1	0x14b	0	8	8	0	0	0	01 00 42 00 60 2D 80 00	
1539220	1	0x122	0	8	8	0	0	0	87 1F 04 0A 00 00 00 00	
1539221	1	0x68	0	1	1	0	0	0	02	
1539222	1	0x87	0	8	8	0	0	0	67 5C 23 00 B5 40 10 38	
1539223	1	0xb3	0	8	8	0	0	0	2E 07 0E 00 FF 00 00 20	
1539224	1	0x1e1	0	8	8	0	0	0	F2 01 00 EF 01 01 00 00	
1539225	1	0x3d	0	8	8	0	0	0	A3 0F 00 00 00 00 00 FF	
1539226	1	0x225	0	8	8	0	0	0	EE 04 24 00 00 80 00 00	
1539230	1	0xa1	0	8	8	0	0	0	FF AD FC 41 BD F7 C1 20	
1539232	1	0xa3	0	8	8	0	0	0	EA AD FD 41 6B F8 C1 E4	
1539232	1	0x21b	0	8	8	0	0	0	4F 0D 20 02 8D 64 80 7F	
1539234	1	0x2cf	0	8	8	0	0	0	00 00 10 00 04 00 B6 47	
1539235	1	0x401	0	8	8	0	0	0	00 70 00 00 00 FF C0 00	
1539236	1	0xb1	0	8	8	0	0	0	4F 03 7B 00 00 84 07 1F	
1539237	1	0x309	0	8	8	0	0	0	13 E6 15 88 0B 35 4E 81	
1539238	1	0xae	0	8	8	0	0	0	6A 45 A2 01 00 00 A4 00	
1539239	1	0x209	0	8	8	0	0	0	D3 45 7F 1A AB 13 2D DF	
1539240	1	0x87	0	8	8	0	0	0	A6 5D 1F 00 AC 40 10 38	
1539242	1	0xb3	0	8	8	0	0	0	52 08 0E 00 FF 00 00 20	
1539244	1	0x3d	0	8	8	0	0	0	B5 01 00 00 00 00 00 FF	
1539250	1	0x33f	0	8	8	0	0	0	00 5A 00 00 0F 55 02 00	
1539251	1	0xa1	0	8	8	0	0	0	CD AE FA 41 C5 F6 C1 28	
1539252	1	0xa3	0	8	8	0	0	0	54 AE FB 41 73 F5 C1 EC	
1539253	1	0x21b	0	8	8	0	0	0	5A 0E 20 02 8D 63 80 7F	
1539254	1	0x1e9	0	8	8	0	0	0	00 00 00 00 0F FF C2 02	
1539255	1	0xb1	0	8	8	0	0	0	90 05 7B 00 00 84 07 1F	
1539256	1	0x133	0	8	8	0	0	0	02 00 00 00 00 00 00 00	

Timestamp	BusChannel	ID	IDE	DLC	DataLength	Dir	EDL	BRS	DataBytes	GPS
1539257	1	0x18ef0704	1	8	8	0	0	0	40 24 22 00 01 00 00 00	
1539258	1	0xae	0	8	8	0	0	0	8B 46 A2 01 00 00 A4 00	
1539259	1	0x14b	0	8	8	0	0	0	02 37 32 A4 23 00 80 00	
1539260	1	0x87	0	8	8	0	0	0	0A 5E 1B 00 9B 40 10 38	
1539262	1	0xb3	0	8	8	0	0	0	38 09 0E 00 FF 00 00 20	
1539264	1	0x141	0	8	8	0	0	0	00 08 00 00 3C 91 F9 3F	
1539265	1	0x3d	0	8	8	0	0	0	61 03 00 00 00 00 00 FF	
1539270	1	0xa1	0	8	8	0	0	0	3F AF F9 41 CD F3 C1 30	
1539272	1	0xa3	0	8	8	0	0	0	4A AF F9 41 7C F3 C1 F4	
1539272	1	0x21b	0	8	8	0	0	0	03 0F 20 02 8D 55 80 7F	
1539274	1	0x2b9	0	8	8	0	0	0	FF 00 00 4B 46 7C 86 18	
1539275	1	0xb1	0	8	8	0	0	0	2E 07 7B 00 00 84 07 1F	
1539277	1	0xae	0	8	8	0	0	0	D4 47 A2 01 00 00 A4 00	
1539278	1	0x137	0	8	8	0	0	0	E4 FF FD 40 07 7E 00 00	
1539279	1	0x469	0	8	8	0	0	0	C8 86 C8 B2 21 00 00 00	
1539280	1	0x339	0	8	8	0	0	0	F3 10 88 00 64 FF 30 00	
1539281	1	0x87	0	8	8	0	0	0	61 5F 17 00 8D 40 10 38	
1539282	1	0xb3	0	8	8	0	0	0	86 0A 0E 00 FF 00 00 20	
1539284	1	0x3d	0	8	8	0	0	0	00 05 00 00 00 00 00 FF	
1539285	3									$GPGGA,161649.00,4121.04126,N,00142.31385,E,1,04,1.82,213.3,M,49.5,M,,*59

This is just a small screening between 2 GPS frames that means ~1 second.

During this second 747 messages have been read from 74 different ID's.

Overall, this files was a 25 minutes 40 seconds trip storing 1087882 registers(lines) and 53.1MB of CSV file

Entering the last GPS GPGGA data string in this web (https://rl.se/gprmc)we can see that the location is correct:

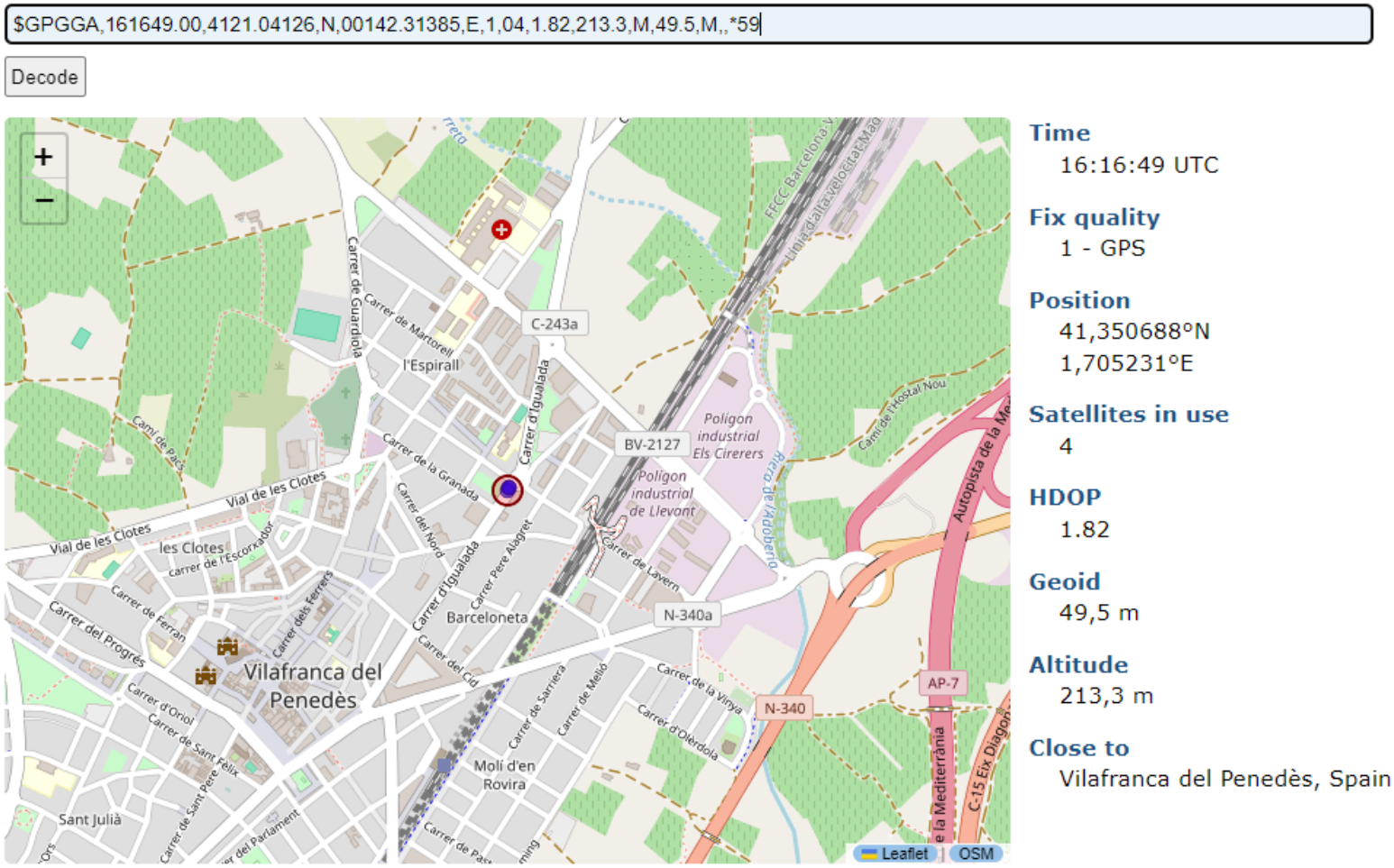

Figure 70 GPS location from the CAN datalogger

4.3 File uploading

Probably the hardest part of all, and as well the most satisfying one. Connecting to AWS was not easy but in the following screenshoot we can see from the terminal of VS Code the file that has been just uploaded and the file on the server proving that the upload is successful.

The data upload is done by changing the name and timestamping the UTC time at the beginning of the CSV file at this way we have our file located in time.

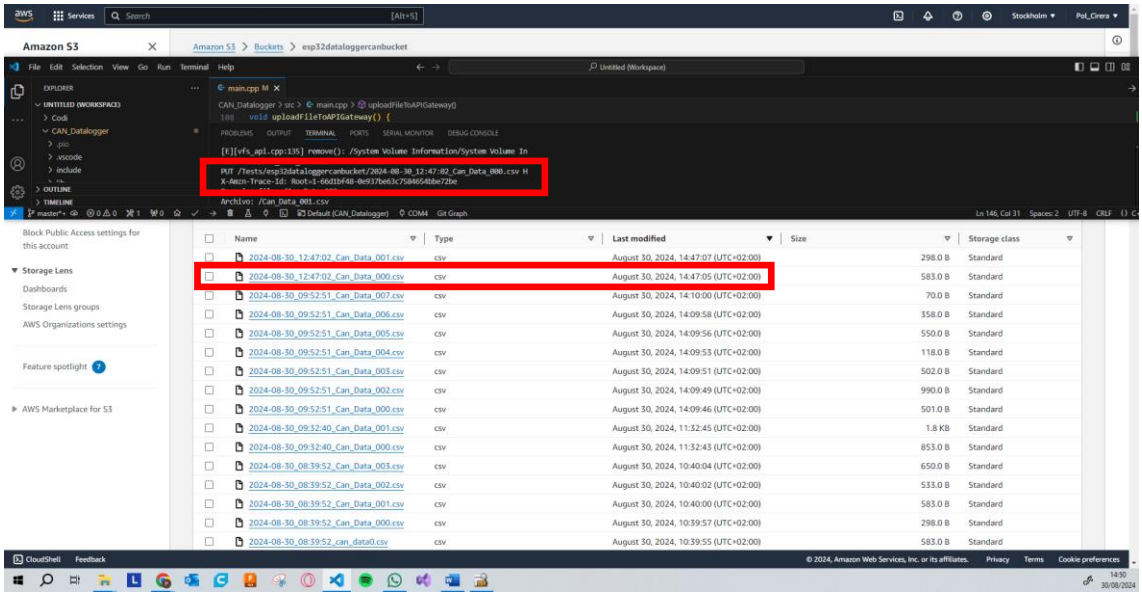

Figure 71 AWS connection results

5 Conclusions

The development of this project has led me to the following conclusions:

DT technology is an emerging technology with lot of potential growth but as well with some uncertainties of its applications.

It's a complex product involving a lot of fields and certainly there is no commercial solution that handles everything on a closed environment or plug and play solution. You must provide the data to a platform; the sensors must be purchased to another manufacturer and the integration must be done by another third party.

In general, and thinking in factories smart cities etc, as smaller is the system easier it is to being able to model it but with few advantages. As you go higher in the hierarchy, the system of systems, more interesting is the possible information outcome to take strategic decisions.

I think that big data and AI is going to take an important role on the further development of the technology and data analysis by searching patterns and improvements on the actual data lakes.

Looking on our application the use of a datalogger CAN for creating a DT is interesting as there is no technology deployment. We use all the vehicle sensors as recollecting points and the data is already converted, filtered and sent for us to recollect. So even though we are not recollecting the data by itself, we are sniffing a communication bus, the quality of the data is good and otherwise is not available by any other means on most of the vehicles.

Analysing the CAN Datalogger, it's obvious that there is a lot to improve, like the following points:

- The PCB must be done selecting the components, and not premanufactured modules. This will reduce the footprint and increase the control over the same as well as the manufacturability.
- The GPS must change to a more updated one (GNSS) with better reception configurability and faster refresh rate.
- The CAN transceiver must include the decoding of CAN FD and CAN XL.
- Including a second CAN or LIN channel could be an option
- The DC-DC must be optimized using different components that causes less interferences.
- A RTC clock with its own battery is a must to timestamp the data properly using epoch method.
- Adding GSM capabilities is needed even though it has some drawbacks and increase the system complexity
- The connection with AWS must improve in terms of speed, flexibility of use and implementation.
- The code can be improved in terms of security and configuration options. Wi-Fi and AWS keys are hardcoded not being able to update them without the source code.
- The actual application / data recording is not handling the actual vehicle installation/identification.
- External application must be developped to enable configuration and data visualization in local or remote.

6 References & webography

[1] D. M. Botín-Sanabria, D. A. Santiesteban-Pozas, G. Sáenz-González, R. A. Ramírez-Mendoza, M. A. Ramírez-Moreno, and J. de J. Lozoya-Santos. (2021, November). Digital Twin for a Vehicle: ElectroBus Case Study. Proceedings of the International Conference on Industrial Engineering and Operations Management, Monterrey, Mexico, pp. 2971–2981.

[2] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. (2017, March). Digital twin-driven product design, manufacturing, and service with big data. Int. J. Adv. Manuf. Technol. [Online]. 94(10). pp. 3563–3576.

[3] C. Boje, et al. (2020). Distribution of studies on BIM uses in construction. Automation in Construction, 114, 103179.

[4] F. Tao, M. Zhang, and A.Y.C. Nee. (2019). Applications of Digital Twin. In Digital Twin Driven Smart Manufacturing, Academic Press: Cambridge, MA, USA, pp. 29–62.

[5] M. Anwar, M.K. Alam, S.E. Gleason, J. Setting. (2023). Digital Twin models of power electronic converters using dynamic neural networks. Sensors, 23, 1414. [Online].

[6] https://randomnerdtutorials.com/esp32-microsd-card-arduino/

[7] https://randomnerdtutorials.com/esp8266-nodemcu-influxdb/

[8] https://randomnerdtutorials.com/esp32-neo-6m-gps-module-arduino/

[9] https://www.waveshare.com/wiki/UART_GPS_NEO-6M

[10] https://synerflight.com/kwad-documentation-2/gps-configuration/

[11] https://www.peak-system.com/PCAN-View.242.0.html?&L=1

[12] https://docs.aws.amazon.com/freertos/latest/userguide/getting_started_espressif.html

[13] https://aws.amazon.com/es/blogs/compute/patterns-for-building-an-api-to-upload-files-to-amazon-s3/

[14] https://blog.learnesp32.com/post/esp-and-aws-s3-integration-part-1-create-an-aws-bucket/

[15] https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[16] https://docs.espressif.com/projects/esp-idf/en/v4.1/api-reference/index.html

[17] https://www.lens.org/

[18] https://eu-north-1.console.aws.amazon.com/