CAN Bus Sniffer Explained (2021)

All modern vehicles today are controlled by multiple Electronic Control Units (ECUs), which you can think of as small computers controlling all electrical components in your car.

Using the OBD-II port and an AutoPi it is possible to communicate with the ECUs.

One of the ECU's is called the Engine Control Module (ECM). This is responsible for communication with a lot of subsystems, like transmission, power steering, windows and doors.

These subsystems communicates on a network bus called Controller Area Network (CAN), by broadcasting messages on the bus. A message could look like this:

CAN bus output

The '024' is the sender ID of the message, and the rest is the

data containing the actual data. For example it could be a message containing instruction to roll down a window.

With access to the CAN bus through the AutoPi, it is possible to send commands to your vehicle, and thereby inject commands on the CAN bus. The only problem is that individual commands to control specific parts of your vehicle, is not easily available.

A common method to identify commands is to listen/record the data flow on the bus, manually perform the action in your car you want to identify (like rolling down a window) and then identify the command on the recorded data stream.

This is commonly known as CAN sniffing and is normally a very advanced way to discover hidden metrics in your car. But with AutoPi this has been reduced to only a few steps, that you need to run from the AutoPi Cloud and everyone can expand the possibilities of their car. The steps are:

From the AutoPi Cloud, start CAN monitoring
In your car, perform the action you want to record 5 times or more
Let AutoPi Cloud analyze the recorded data in order to identify your action
Name and save the action in the AutoPi library for everyone to use in the future

It couldn't be easier. In this blog we will go over the details and explain what you can do with your new metrics.

Monitoring the CAN bus is complicated, because so much data flows in modern cars. So identifying individual messages from the stream of data is difficult.

A common way of identifying a specific message is called "divide and conquer". The idea came from a classic computer algorithm, of same name, used in many aspect (like sorting). Easily explained, you know that the message you are looking for can be compared to finding a needle in a haystack.

With the divide and conquer algorithm, you divide the haystack in two equal parts and look through half the haystack for the needle.

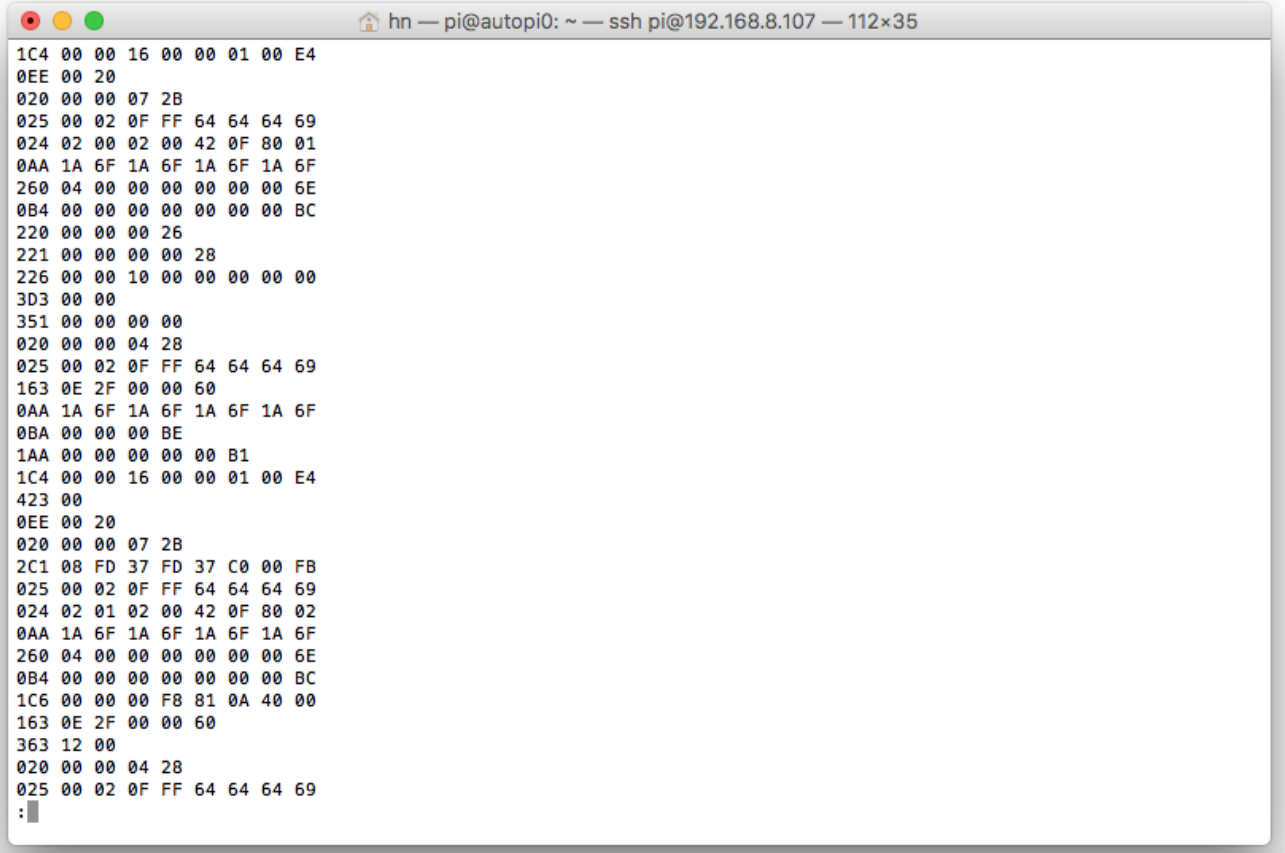
If you don't find the needle, you break up the remaining half in two new stacks, and looking in one of these. You continue this process until you find the needle.

You can apply this to finding CAN messages as well. Let's say you want to discover the command for controlling the power windows, then the steps involved are:

Record messages flowing while pressing the window switch in the car.
Divide the recorded messages into two halfs.

Replay/send all data from one part to the car's CAN Bus. If the window rolls down you have the right part.
Keep doing this procedure until you have identified the exact message you are looking for.

The following CAN data dump is 34 lines out of 1797 lines, for a 5 second recording:



Large CAN bus export

Once you identified the specific command, you need to replay/send it to make sure you have the right one.

All of this sounds very complicated (which it can be), but with AutoPi all of this has been automated in a simple tool

you can access from the AutoPi Cloud. The AutoPi Telematics Unit can be considered as a CAN Bus Sniffer due to its wide variety of functionalities it offers.

Benefits of CAN bus reverse engineering

CAN bus reverse engineering has lots of benefits, however, we will tell you more about the most common ones.

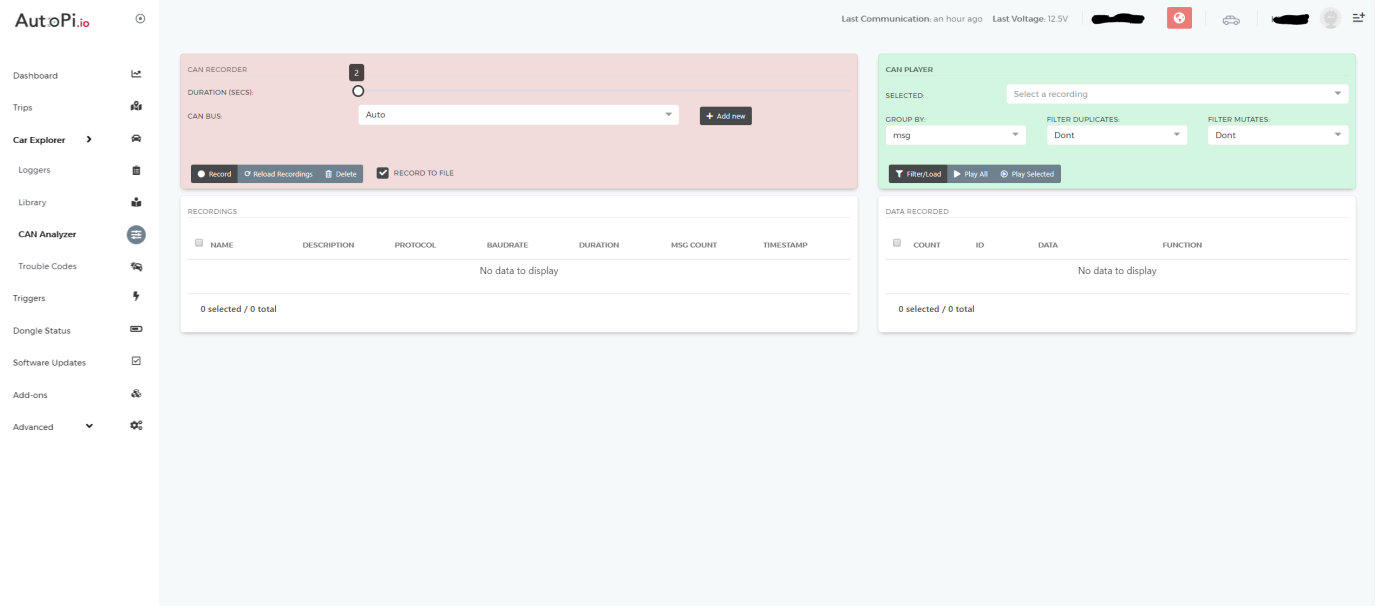
Decoding data - also known as CAN bus hacking, stands for the ability to enable decoding of proprietary CAN IDs. Then it is able to analyze data from cars, trucks, machinery and so on.
Give commands - you are able to give commands to your car. This can be done by enabling control of vehicles via commands such as toggle locks, lights and so on.
Applications - You are able to log state-of-charge (SoC) from electric vehicles (EV).
DBC databases extension - CAN bus decoder tool can definitely help you with reverse engineer missing CAN messages and signals.

CAN bus reverse engineering brings you lots of useful lifehacks, that's why it is also called CAN bus hacking. Read more about it below and find out how can AutoPi help you with CAN bus in your car.

Using AutoPi to discover new

commands in my car

With the AutoPi Cloud, the tedious process of finding CAN commands for your vehicle has been automated in a simple user interface.



Above image shows the user interface used to discover new CAN commands. The steps are very simple. You need to physically be in your car to do the discovery:

Open the CAN explorer. It will tell you if there is connection to the car
Press the record button
Now perform the function you want to record 5 times. Like pressing the “left front window” switch
Press the stop record button

The system will now analyze the data received and output

the possible commands found. Because of the large data amount, the system will sometimes output a few number of possible commands. You now have to possibility to narrow it down by replaying the found commands one-by-one.

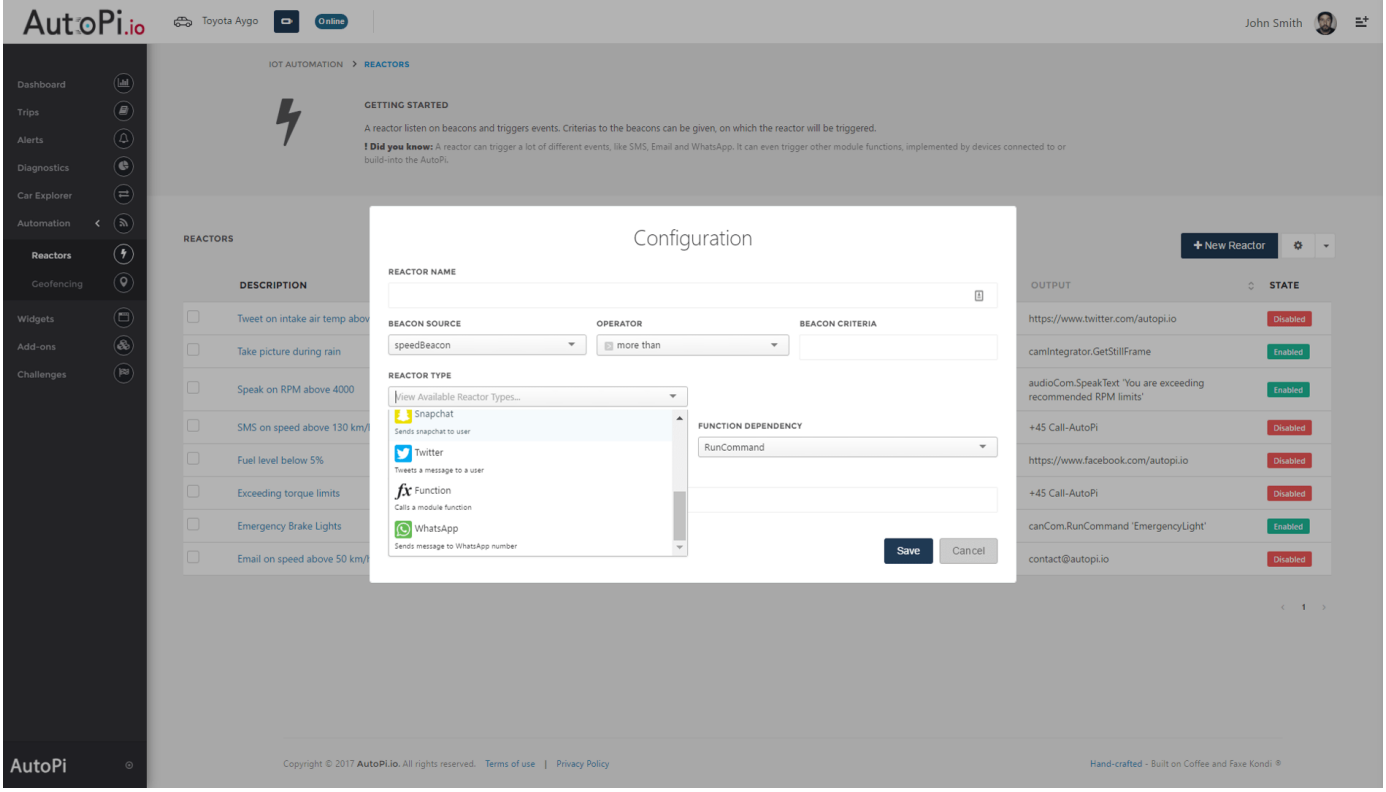
When you found the command you were looking for, you simply press the save button. You newly found command is stored for future use by you and for all other users with a similar car.

Decoding raw CAN messages might be a bit complicated and confusing, especially if you are new to that. We have prepared a guide on how to log raw CAN messages, in order to make the whole process easier for you.

Using discovered CAN commands to build something cool

So, what can you do with your new CAN command? Why not use it to tie up a cool speech command to your car.

The AutoPi Cloud comes built in with an If-This-Then-That trigger system. Using this system you can add your own triggers and trigger the new found CAN command.



The image above shows the interface for adding new triggers to the system. Adding a new trigger is very simple.