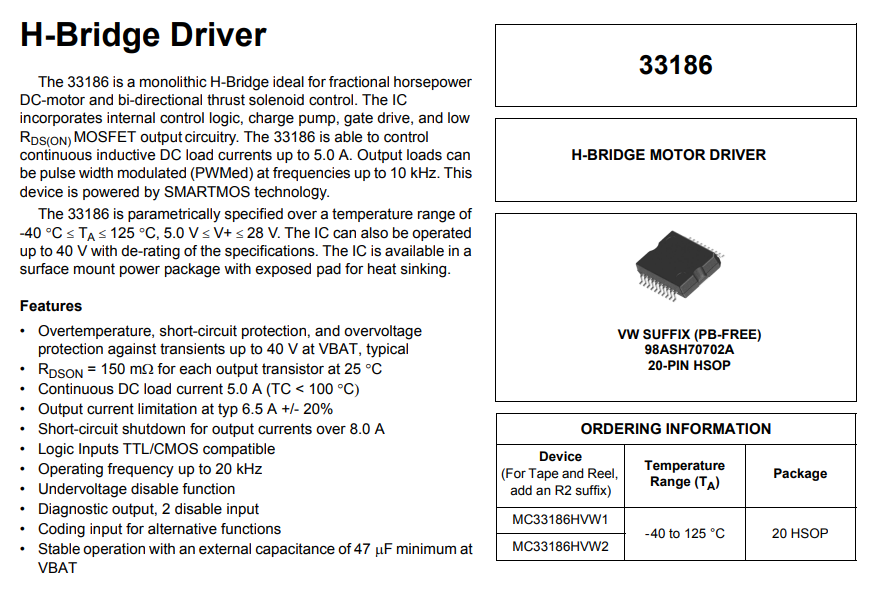
# Doxygen:

Doxygen is a documentation generator that works with many programming languages. It extracts information from specially-formatted source code comments and saves the information in one of various supported formats. Doxygen supports static analysis of a codebase.

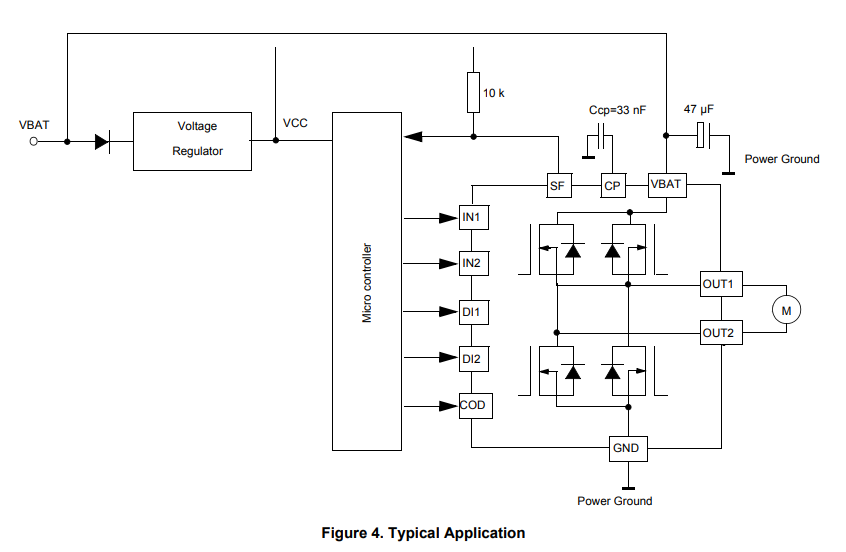
# Automotive DC motor drivers (used for driving electronic throttle bodies)

Driving DC motors using discrete circuits is a bad idea because of its complexity. As an example, a simple H-bridge with no protection circuitry requires four power transistors and a lot of gate drive circuitry while still not being able to provide any protection against over-current, over-temperature conditions.

Research indicates that ICs which include all of the mentioned functionalities exist. Take MC33186 as an example. A fully integrated and high-power DC motor driver.



The datasheet also provides typical usage figures.



# Linked lists in programming

# Measuring thermistor temp sensor

## What is a thermistor?

A thermistor is a semiconductor type of resistor in which the resistance is strongly dependent on temperature. The varying resistance with temperature allows these devices to be used as temperature sensors, or to control current as a function of temperature.

The most common type of thermistor used in engines is a thermistor water temperature sensor shown below:

This sensor is responsible for measuring the temperature of the engine coolant temperature and it helps the ecu decide how much fuel and ignition timing is should use to achieve optimal performance in different climate conditions.

## Steinhart–Hart equation

This equation shows the relationship between the resistance and temperature of the thermistor.

where a, b and c are called the Steinhart–Hart parameters and must be specified for each device. T is the absolute temperature, and R is the resistance.

However, there is a problem! We need to find the Steinhart-Hart equation parameters.

There is a good resource on the web that describes this in detail: [https://www.thinksrs.com/downloads/PDFs/ApplicationNotes/LDC%20Note%204%20NTC%20Calculatorold.pdf](https://www.thinksrs.com/downloads/PDFs/ApplicationNotes/LDC%20Note%204%20NTC%20Calculatorold.pdf%20)

A code needed to be developed first to measure resistances and then a code was developed to translate that measured resistance into a correct temperature reading.

Relative files: ‘thermistor.h’ and ‘resistor.h’

# Tuning for the end user

The end use is most certainly not advanced in c language skills, which means that there is a need to provide an easy-to-use interface for the user to be able to adjust parameters and settings in the SGC.

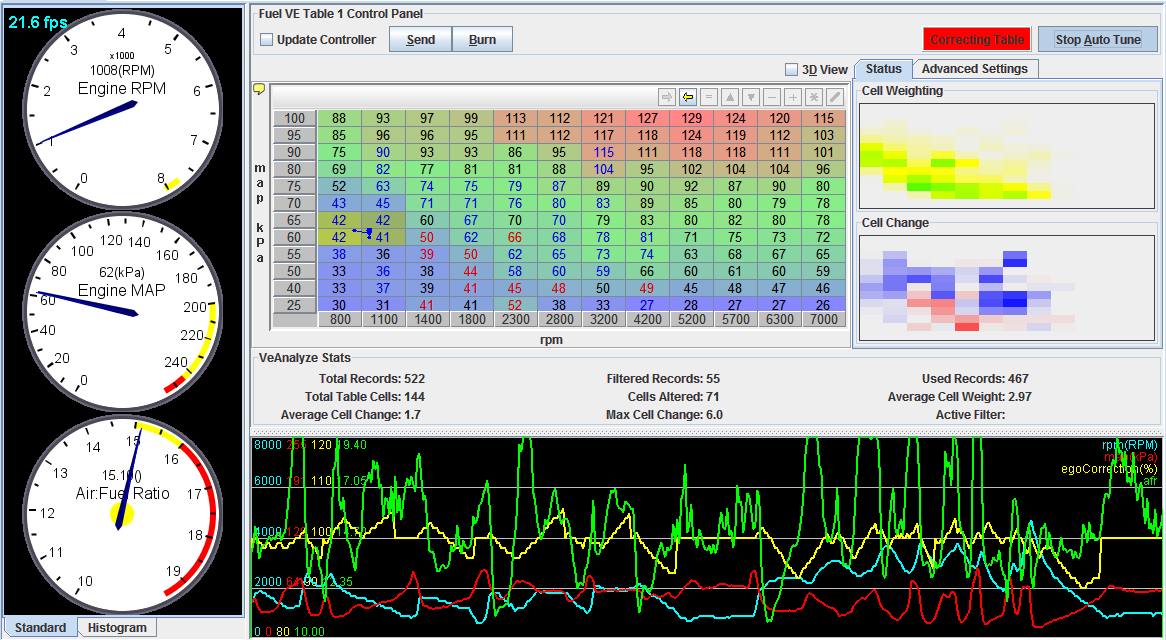
However, developing a windows or Linux application form scratch that can connect to the SGC would require a lot of effort and the end result would certainly not be pleasing to the eye or good to use.

Considering the above, I have decided to use Tuner Studio, which is an open-source software developed for open-source engine control units. Considering that it is used by a few successful projects that have been put into production, I can be assured that the end result will at least be pretty to look at and easy to use.

The down side is that I will need to learn how to code the configuration required for the software, meaning that I will have to make tuner studio and my controller speak the same language. This is most certainly going to be very challenging but I’m going to put in the effort.

Tuner Studio coding guide: <https://www.efianalytics.com/TunerStudio/docs/EFI%20Analytics%20ECU%20Definition%20files.pdf>

This is how Tuner Studio looks like:



# Storing user configuration

Storing user configuration on the SGC requires a non-volatile memory which keeps user data even after the power is cut off. There are multiple approaches to this problem, including, using a micro-controller using a built-in EEPROM which stands for electrically erasable read only memory, using an external EEPROM IC or as I just found out, EEPROM emulation.

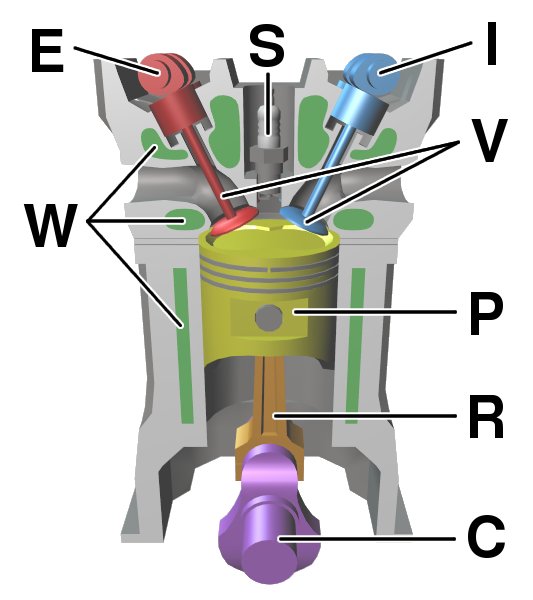
EEPROM emulation allows us to not have an external EEPROM chip which adds extra cost and complexity to the controller. By erasing sectors of flash memory in the microcontroller and carefully writing data on those sectors. We can store anything we want, effectively.

A software driver has been written by engineers at ST, using it seems relatively easy and there is a good resource on this matter on the ST’s website, AN3969.

# PID control implementation

# How does a four-stroke engine work?

## Parts of a four-stroke spark-ignition engine:



The parts are:

C – crankshaft.

E – exhaust camshaft.

I – inlet camshaft.

P – piston.

R – connecting rod.

S – spark plug.

V – valves. red: exhaust, blue: intake.

W – cooling water jacket.

gray structure – engine block.

### The engine operates in four distinct strokes of the piston within the cylinder:

#### Intake Stroke

The intake valve opens.

The piston moves down, drawing in a mixture of air and fuel.

In electronically controlled engines, the amount of air is managed precisely by an Electronic Throttle Control (ETC) system.

#### Compression Stroke

The intake valve closes.

The piston moves up, compressing the mixture. This brings the molecules close together for a better burn.

#### Power Stroke

A spark plug ignites the compressed mixture.

The heat and pressure force the piston downward, producing power.

The ECU precisely controls the ignition timing using data from sensors (e.g., crankshaft position, intake air temperature sensor, intake air pressure sensor etc.) and from the ignition table.

#### Exhaust Stroke

The exhaust valve opens.

The piston moves up, expelling exhaust gases.

The control system has to precisely control the amount of fuel getting into the engine because:

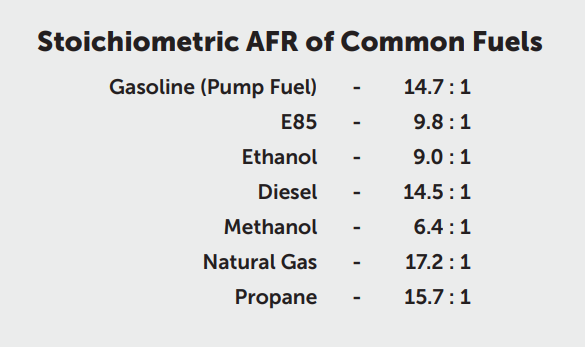
When the load on the generator increases, more power—and therefore more fuel—is needed. When the load drops, fuel delivery must decrease to avoid waste and maintain stable operation.

The controller uses data from sensors (such as engine speed, air intake pressure and temperature and exhaust oxygen levels) to:

* **Maintain a stable RPM** to keep output frequency constant.
* **Optimize fuel efficiency** based on real-time load demand.
* **Ensure clean combustion** for lower emissions, especially in regulated environments.
* **Prevent engine overload or stalling** by responding quickly to changes in demand.

Without electronic fuel control, the generator might run too rich (wasting fuel and producing more emissions) or too lean (risking unstable operation or damage).

# Calculating the amount of fuel that needs to enter the engine

The ratio of air to fuel varies depending on the fuel type and also the requirements of the engine. More on this later. The table below shows the stochiometric ratio of air to fuel for different types of fuel.

The stoichiometric air fuel ratio is the ratio of air to fuel that can enter the engine and burn completely in an idea situation.

The air fuel ratio has a major impact of the combustion process of the engine and specially it’s fuel economy.

Generally, a higher air to fuel ratio will result in more fuel economy but this does not mean that we can run any engine with a high air to fuel ratio. The reason being that combustion temperatures tend to rise as we increase the air fuel ratio and approach stoichiometric condition. This also results in an increase in the NOx emissions of the engine.

The table below illustrates this phenomenon:

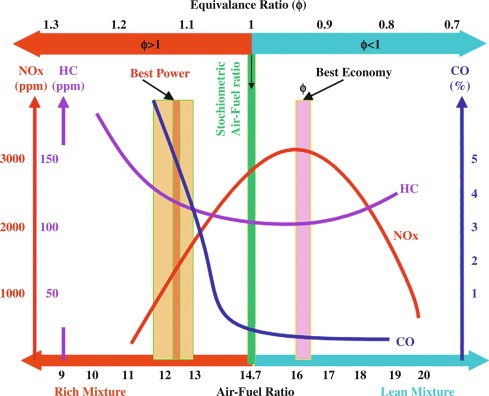
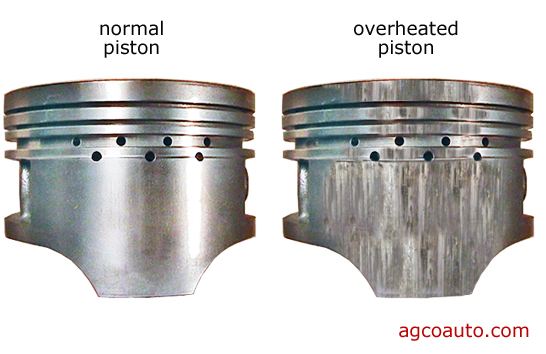


Figure Effect of air-fuel ratio on various factors

It is important to note that the catalytic converter in cars easily gets rid of most NOx emissions, this allows automotive manufactures to run their engines at high air to fuel ratios, providing more fuel economy for the final user but, the situation is different for generator engines as they usually do not have catalytic converters. Moreover, in case of backup generators, fuel economy is usually less of a concern.

My research and experience also indicate that running an engine closer to stoichiometric ratio is possible only in low-load conditions, where the engine is not under much stress and does not produce much heat. This is because leaner air to fuel mixtures – i.e. higher air to fuel ratios – result in more combustion temperatures which could risk damaging the engine components such as pistons and valves.



**Figure 2 Piston damage due to improper fueling**

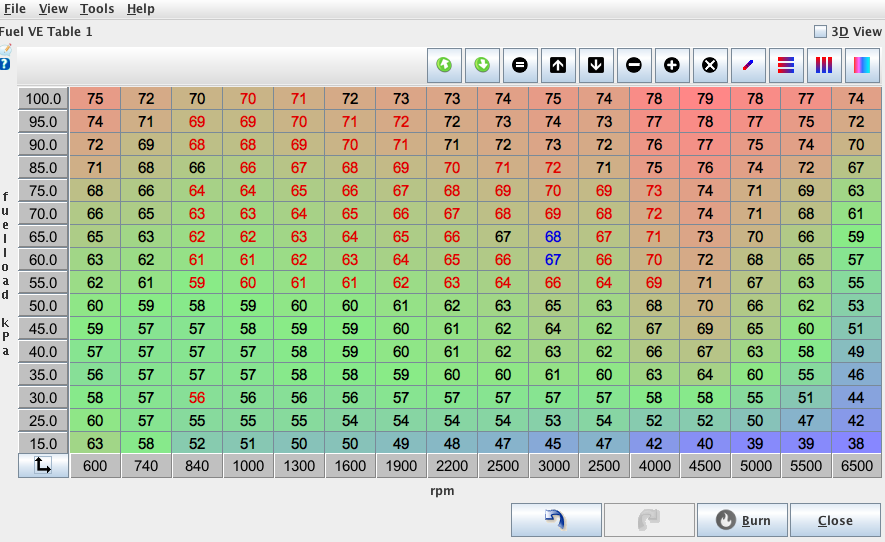
This means that we usually have to operate the engine using a rich air fuel mixture when it is under significant load. In order to adapt to these varying conditions, engine management systems use a table that dictates how much fuel should enter the engine.

This process is not as simple as it seems because every engine is different and the amount of air mass entering the engine varies widely with rpm, air pressure, throttle opening, and temperature.

In order to model the capacity of an engine to ingest air, we use a table known as volumetric efficiency (VE) table which describes how much air enters the engine vs how much air could theoretically enter the engine. This is expressed as a percentage and it is often less than 100%.

Knowing the VE of the engine at any point and intake air temperature, intake air pressure and rpm allows us to calculate the amount of air mass entering the engine, this then enables us to calculate the required amount of fuel that needs to be mixed with this air to have the correct air fuel ratio.

Further corrections can be applied to the amount of fuel to account for different operating conditions.

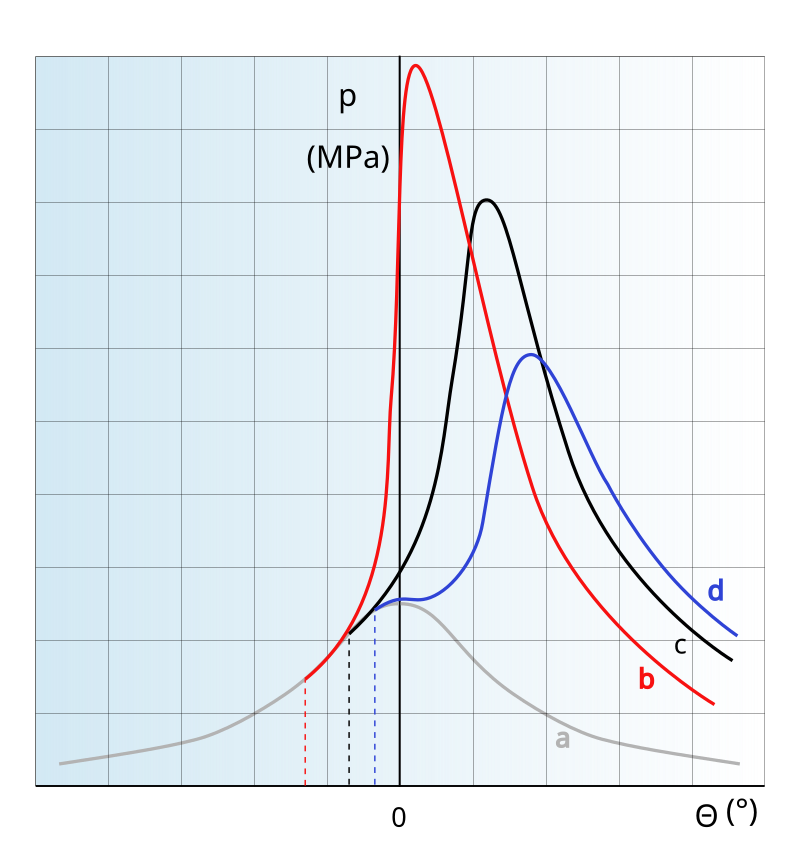


**Figure 3 Volumetric efficiency table in an ECU**

# The challenge of firing the spark at the right time

The timing of the spark in an engine impacts the performance of that engine to a large degree. Not to mention the effect it has on engine reliability. Therefore, a modern ecu must precisely calculate the exact time each spark plug needs to be fired based on information coming from the crankshaft position sensor, ignition table (more about this later), manifold absolute pressure and intake air temperature at a minimum.

Important note: Spark is fired before the piston reaches top dead center (TDC). This is because combustion takes time and by the time full combustion takes place, the piston is past TDC and in the correct position to receive the combustion pressure.

This is apparent when we look at how the pressure in the combustion chamber changes vs spark plug timing.

**Figure 4 Effects of spark timing**

(a) misfire

(b) too soon

(c) optimal

(d) too late

**Figure 4** shows the effects of spark timing on cylinder pressure for a specific engine rpm. This means that everything changes based on the speed of the engine.

One might ask: don’t generators run at a constant speed?

To which I can reply: they do, but they need to start up first and reach that constant engine speed and also the speed isn’t really constant and changes to some extent when a load is suddenly taken off or put on the generator.

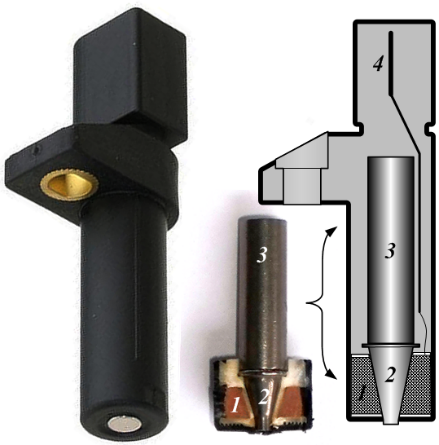
# Calculation and spark execution

## Crankshaft position and speed

We must first know where the engine is exactly. This is done using a component known as the trigger wheel and the crankshaft position sensor.

A crankshaft position sensor is a type of variable reluctance sensor (VR), that generates a voltage when the magnetic reluctance in front of the sensor changes. A typical sensor can be seen in **Figure 5**.

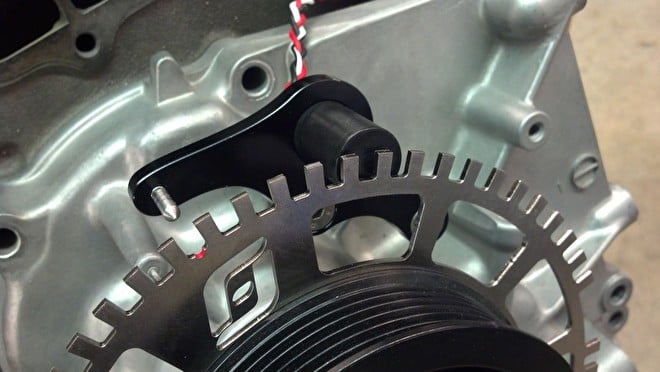
This voltage is the interpreted by the controller to extract crankshaft position and speed.

In order to change the reluctance in front of the sensor, a trigger wheel is used. It is basically a wheel with teeth all around its perimeter with a few teeth missing.

**Figure 5 Typical crankshaft position sensor**

When a tooth passes in front of the sensor, the reluctance decreases substantially and when the tooth is not there the reluctance increases. This generates a voltage that varies in intensity depending on the engine speed.

These teeth are usually directly built into the engine’s flywheel. A heavy rotating weight, used to smooth the output power of the engine. **Figure 6**.

**Figure 6 Flywheel with teeth on the bottom half. The missing teeth can also be seen**

**Figure 7 A different style of trigger wheel and sensor combination. The missing teeth can also be seen**

Now the question becomes, why are some teeth missing?



**Figure 8 Effects of a missing tooth on the sensor output**

As I have illustrated in the figure above. The output signal shape changes when a missing tooth passes in front of the sensor. This allows us to count the number of teeth after that and by knowing the number of teeth in total we can calculate the position of the crankshaft.

However, this is easier said than done, as implementing it in code is quite a challenge.

The complete trigger wheel implementation can be seen in files “trigger.h” and “trigger.c”, but it is a bit complex to understand since the problem is complex. I can provide a full explanation in person by showing the code and talking about how it works.

Now that we know the position of the crankshaft at any moment and its speed, we must know the load on the engine at any moment.

## Engine load

The best indicator of engine load is the mass of air entering the engine. The more air mass entering the engine the higher the load. The problem is that measuring air mass directly is not easy, but we have information from multiple sensors which can help us calculate the air mass.

These sensors are: