OVMS

Open Vehicle Monitoring System

www.openvehicles.com

OVMS SimpleConsole Howto v1.0 (2014/02/16)

History

v1.0	2014/02/16	Initial version

Recent changes

- -

Table of contents

Welcome!

The OVMS (Open Vehicle Monitoring System) team is a group of enthusiasts who are developing a means to remotely communicate with our cars, and are having fun while doing it.

The OVMS module is a low-cost hardware device that you install in your car simply by installing a SIM card, connecting the module to your car's Diagnostic port connector, and positioning a cellular antenna. Once connected, the OVMS module enables remote control and monitoring of your car.

There are normally only indirect ways for users to communicate with the OVMS module: either by SMS text messages or by using the smartphone App.

This guide outlines how to extend the OVMS (V1/V2) by a simple hardware console offering four push buttons and/or switches and four LEDs to enable a direct user control method for some functions.

The console needs software support in the OVMS firmware to work. At the moment, the Renault Twizy firmware supports using the console to select and inform about controller configuration profiles.

Warning!

OVMS is a hobbyist project, not a commercial product. It was designed by enthusiasts for enthusiasts. Installation and use of this module requires some technical knowledge, and if you don't have that we recommend you contact other users in your area to ask for assistance.

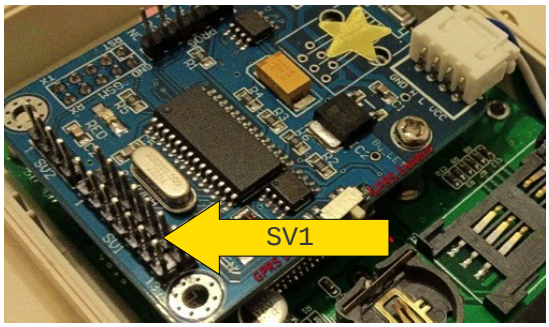
Legal disclaimer: by using the OVMS you agree to do so completely on your own risk. Being a hobbyist project, the OVMS has neither CE approval nor undergone any official EMC tests. It has no ECE approval, so depending on your country may not be legal on public roads.

How does it work

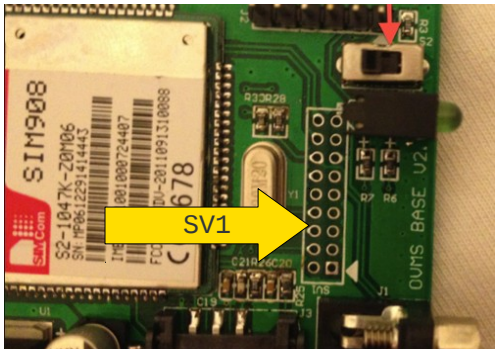
The OVMS V1 and V2 modules include an extension connector on the PCB labeled "**SV1**" enabling access to eight bidirectional I/O ports which can be used for custom purposes.

The console simply connects four of these ports to switches and the other four to LEDs.

So any combination of simultaneous key presses / switch states and LED states is possible as each switch and LED has a dedicated I/O port.

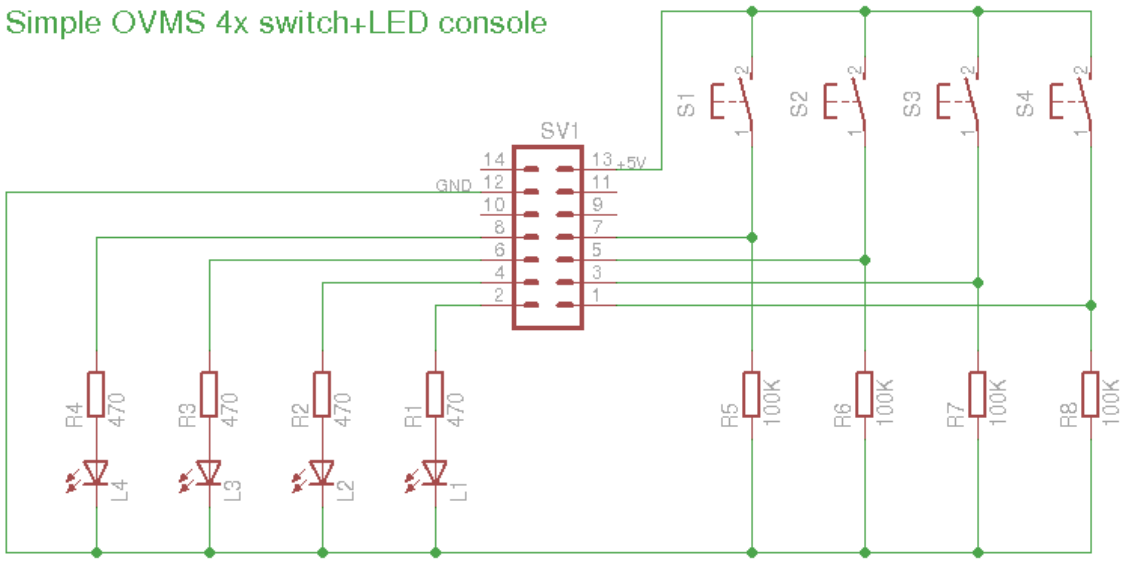


OVMS V1



OVMS V2

Scheme



EAGLE format: `/vehicle/Car Module/SimpleConsole/OVMS-SimpleConsole.sch`

There's no PCB layout as this can be easily mapped onto all-purpose experimentation PCBs.

Parts needed

- 4 switches and/or push buttons (closing)
- 4 LEDs (separately or integrated into the switches)
- 4 resistors 470 Ohm
- 4 resistors 100K Ohm (or similar)
- PCB socket & connector 2x7 (see below) (**ATT**: **2.0 mm grid**!)
- Cable (10 lines)
- PCB & case

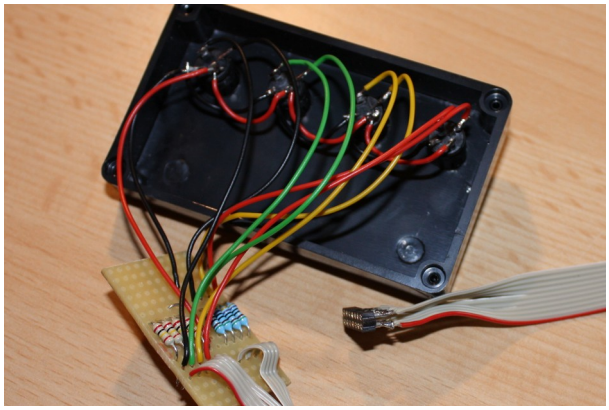
The SV1 port has 14 pins (2x7) on V1 and 18 pins (2x9) on V2 modules. Pins 1-14 allocation is identical for both versions, pins 15-18 are not used by this console. Also the V2 board has not much room at the high end, as the onboard LEDs are mounted very close. So you may decide to use a 14 pin (2x7) connection on V2 modules as well.

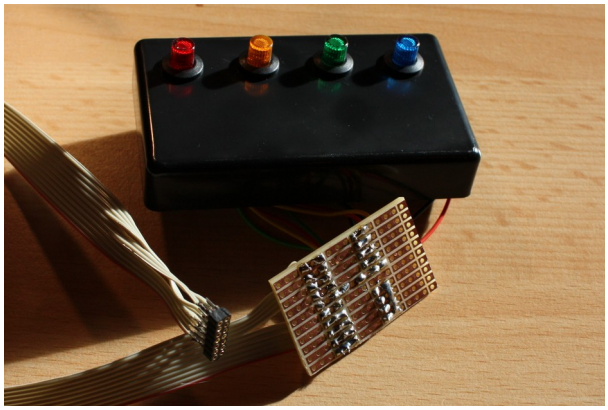
Check if your OVMS module already is equipped with a socket for the SV1 connector, it seems V1 hardware does include a socket while V2 does not.

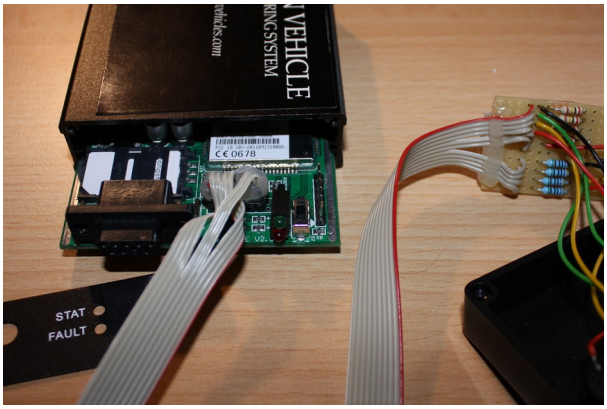
All components can be selected according to your personal likes, just make sure everything fits nice and secure.

Example build

The following example shows a console using four push buttons with integrated LEDs. This setup matches the requirements of the Renault Twizy firmware, where pushing a button maps to selecting a controller configuration, the button LED then reflects the currently active configuration.







The console is attached to the glove box cover using Velcro® tape.

How to read and access

The OVMS framework "input" module provides these utility functions to access single I/O ports (single bits/lines):

```
unsigned char input_gpi2(void);
unsigned char input_gpi3(void);
unsigned char input_gpi4(void);
unsigned char input_gpi5(void);

unsigned char output_gpo0(unsigned char onoff);
unsigned char output_gpo1(unsigned char onoff);
unsigned char output_gpo2(unsigned char onoff);
unsigned char output_gpo3(unsigned char onoff);
```

Alternatively you may read and write the PIC registers PORTA (switches) and PORTC (LEDs). The framework automatically configures PORTA for input and PORTC for output on init.

Just remember inputs are bits 2-5, outputs bits 0-3. So for example to just echo all key presses back to the LEDs, do:

```
// Read input port:
keys = (PORTA & 0b00111100) >> 2;

// Echo keys to LEDs:
PORTC = (PORTC & 0b11110000) | keys;
```

For a more complex example take a look at the Twizy `vehicle_twizy_state_ticker10th()` function.

Caveats

The I/O ports on the PIC are mapped also to other functions depending on the PIC state. In effect during OVMS boot / init / modem reset, output bit 3 (LED 4) will be enabled. The state gets cleared once the OVMS bootup is done.

Contact / Feature requests

If you need help, want to give some feedback, find bugs, have an idea on improving or miss some feature, please don't hesitate to post on the OVMS forum:

> http://www.openvehicles.com/forum

If you want to take part in the development in any way, please subscribe to the OVMS developers mailing list:

> http://lists.teslaclub.hk/mailman/listinfo/ovmsdev

Remember, this is a community project, any help is appreciated :-)

Thank you!