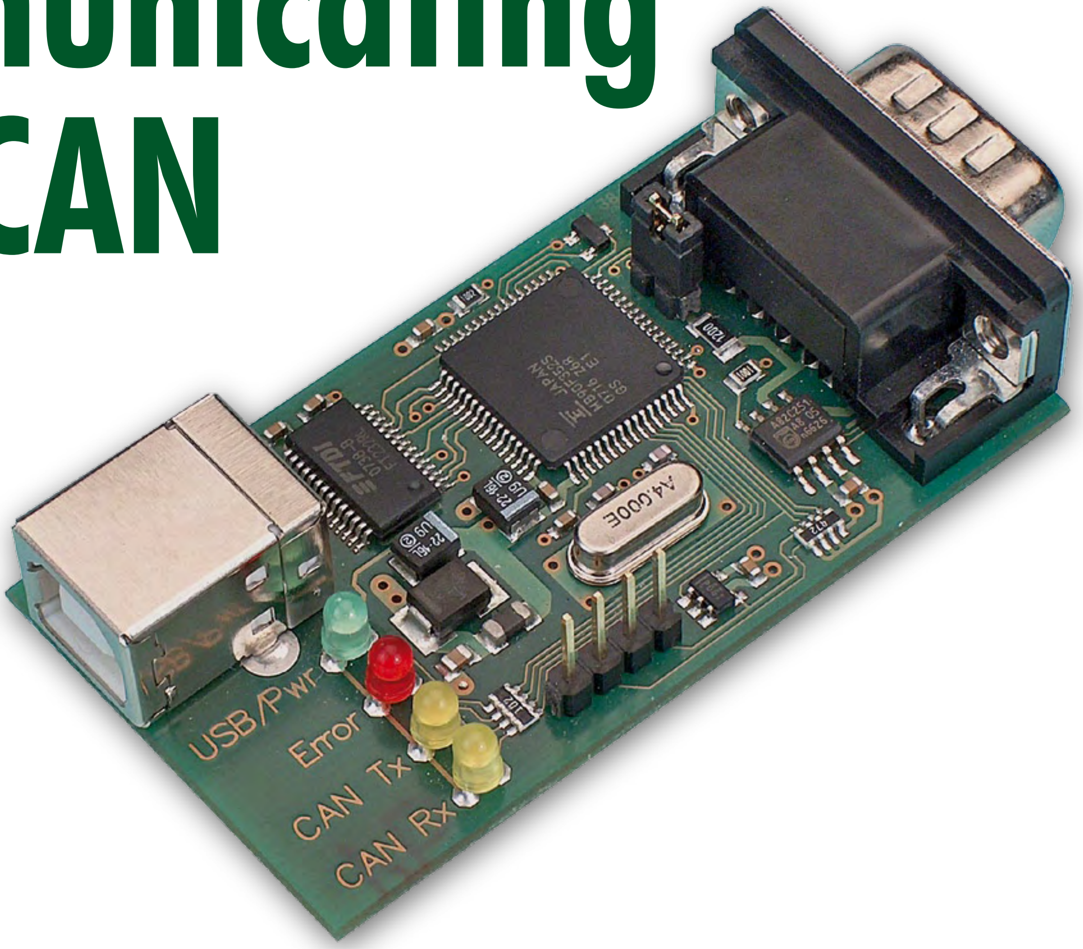
Communicating with CAN



Compact USB-CAN Adapter

Klaus Demlehner

Despite the fact that the CAN protocol is a serial protocol, it can't just be connected to (the serial port of) a computer. The all-round USB-CAN adapter described here provides a compact and simple solution. With the help of the accompanying software you can follow all data communications taking place and carry out operations such as filtering and storage at the flick of a (mouse) switch.

The CAN (Controller Area Network) protocol was originally developed for use in the automotive sector. It is now over 20 years old, but is still frequently used these days. The protocol was invented by Bosch in order to let microcontrollers and other electronic devices communicate with each other.

It was specially designed for use in environments where you have a lot of electromagnetic interference. Because of this it was decided to use differential signalling, all of which made CAN especially suitable for use in the automotive sector.

The design

The USB-CAN adapter presented here makes it very easy to communicate with the CAN bus. The data present on the CAN bus can be read via a USB connection, which can be found on virtually every PC these days. You can of course also transmit data. The recommended software, *Tiny CAN View*, has a handy and clearly laid out user interface for this.

Apart from the recommended software, the USB-CAN adapter can also be used with other 'third party' software such as *CANopen Device Monitor* and *CAN-REport*. After installing the device driver for the FTDI USB interface chip used in the adapter it can be easily accessed from either Windows or Linux operating systems. When a firmware update for the microcontroller is required this can also be easily done via the same USB connection.

The circuit diagram

The size of the circuit diagram (**Figure 1**) is certainly not reflected in the size of the final PCB. The microcontroller in particular is a lot smaller in real life, mainly because of its SMD packaging. The other functional blocks in the circuit diagram stand out very clearly: nearly every block represents an IC.
The USB interface makes use of a USB-to-serial converter chip (IC1). This chip is the well-known FT232RL made by FTDI. It is widely supported in both the Windows and Linux operating systems. The only external component required by the FT232RL is a capacitor (C6), which is used to stabilise the internal 3.3V supply voltage.
The 16-bit microcontroller made by Fujitsu (IC3) comes with integrated

Hardware highlights

- Electrically isolated
- External power supply of 9 to 48 V
- Protection circuitry for CAN and external supply, especially for automotive applications
- Hardware based transmit buffer using an interval timer, with a capacity for up to 16 CAN messages
- Microcontroller supervised by 'Hardware Watchdog'
- Firmware for the module can be updated via the USB bus.

Software highlights

- Data reception in either polled or event-driven mode (CallBack function)
- Selection of received messages via filter rules
- Transmit buffer with an interval timer
- Supports saving of log files

CAN support and forms the heart of the circuit. From the controller we use the serial and CAN interfaces, and the controller also has a built-in 15-channel 10-bit A/D converter. However, the latter is not used in this circuit.

To show the status of the USB-CAN adapter we have connected four LEDs (LD1 to LD4) to the controller. With the help of **Table 1** you will be able to determine the current status of the controller.

The microcontroller (IC3) can be programmed via the USB interface. D1 is used here to protect against over-voltages. Components C1, L1, C4 and C5 make sure that any RF interference picked up by the USB cable stays out-

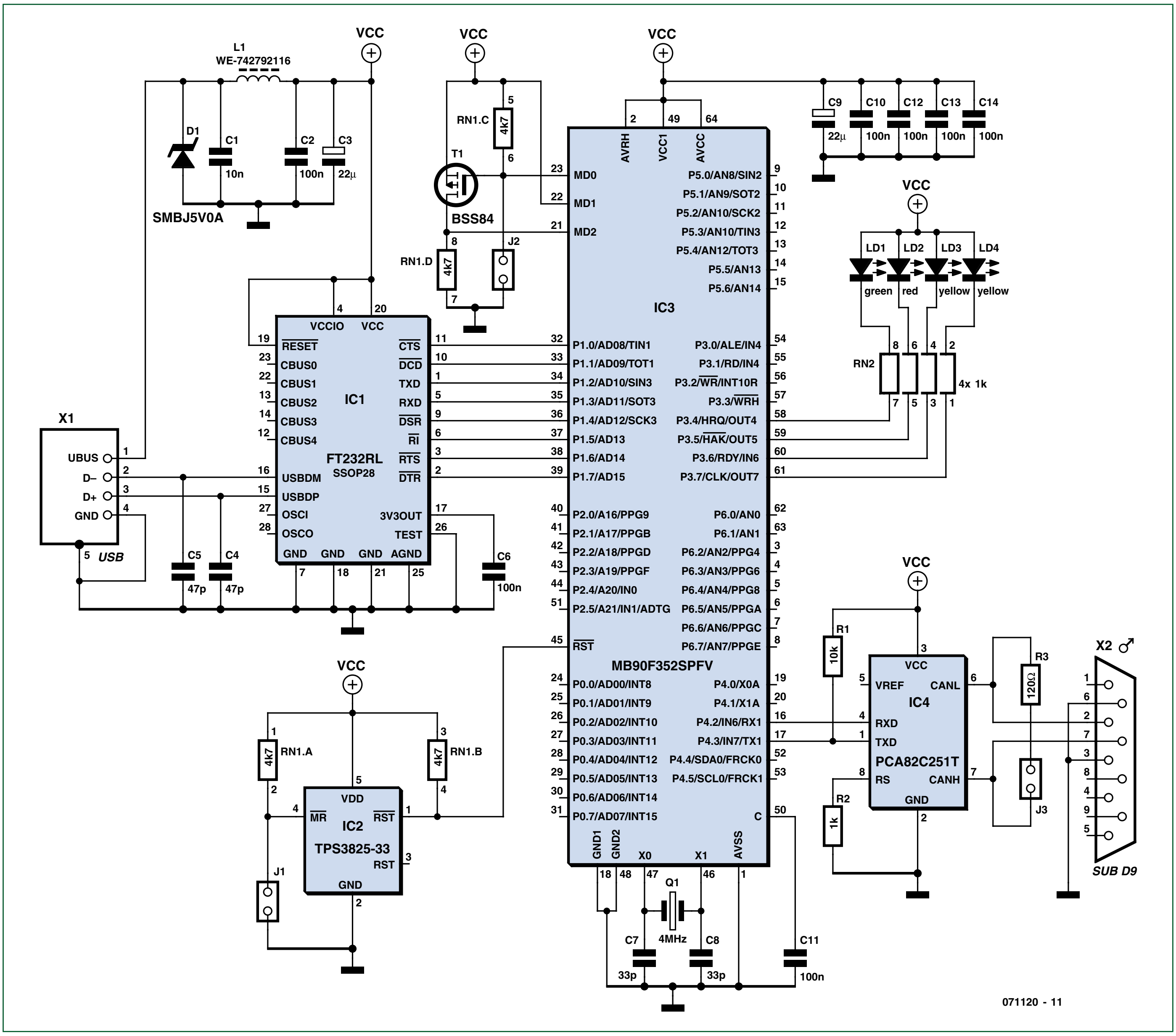
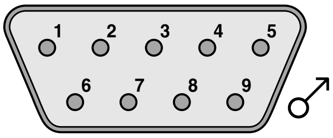


Figure 1. The circuit diagram seems much larger than the PCB. It is clear that the microcontroller made by Fujitsu (IC3) plays the main role in this circuit.

9-way sub-D plug



Pin	Signal	Description
1	-	reserved
2	CAN-L	CAN bus signal, Low
3	CAN-GND	CAN ground
4	-	reserved
5	CAN Shield	not wired
6	GND	CAN ground (optional)
7	CAN-H	CAN bus signal, High
8	-	reserved
9	CAN-V+	not wired

071120 - 12

Figure 2. This shows the pinout of the 9-way sub-D plug used for connections to the CAN bus.

side the circuit. In that way they also protect IC1.

When programming jumper (J2) is not plugged in, pin 23 (MD0) is connected to the positive supply via RN1C and pin 21 is connected to ground via RN1D. The controller is then in RUN mode. With the programming jumper in place, pin 23 (MD0) is connected to ground. Transistor T1 will then conduct and present a logic High level to pin 21 (MD2). The controller is then in its programming mode.

IC2 is a reset controller, which together with jumper J1 serves as an external reset circuit for the microcontroller.

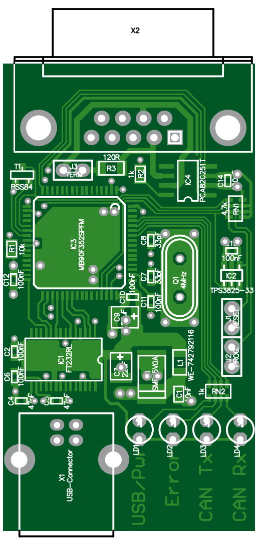


Figure 3. PCB component layout as seen from above. The size of the PCB shows just how compact this project is.

As with all CAN circuits, this project requires a CAN transceiver. In our case this is IC4, a PCA82C251, which conforms to the ISO-11898 standard. This IC has a similar function in this circuit to that of a MAX232 used in conjunction with a PC: it converts the 24 V CAN signals to TTL levels and vice versa.
As far as the other components are concerned, R1 makes sure that the microcontroller can't block the CAN bus during the initialisation. R2 manages the 'slope control', although this function isn't used here. When jumper J3 is plugged on, the CAN bus is terminated by R3 (120Ω). C2, C3, C9, C10, C12, C13 and C14 are decoupling capacitors. **Figure 2** shows the pinout for the CAN connector. The PCB component layout is shown in **Figure 3**.

Software

The Tiny CAN View monitoring program is based on the GTK+ library. On a Windows based system this library needs to be installed first.
When Tiny CAN View is started for the first time, the program warns that it can't find a configuration file. After a click on 'OK' you should therefore first enter a few settings.

In the main window (see **Figure 4**) you can see all information at a glance. In (1) the received messages are displayed. This requires that the trace function has been enabled first. In (2) the filtered messages are displayed, (3) shows the macro list and (4) the transmit list. In this case a macro is a stored CAN message, which makes it an easy and fast way to transmit messages. Macros can be created easily via the *macro* menu.

When required, the transmit list can be expanded to several lines via the setup menu (*Options* -> *Setup*, *transmit* tab). The filtering of messages can be set up

COMPONENTS LIST

Resistors
R1 = 10kΩ
R2 = 1kΩ
R3 = 120Ω
RN1 = 4-way 4kΩ7 SIL array
RN2 = 4-way 1kΩ SIL array

Capacitors
C1 = 10nF
C2,C6, C10-C14 = 100nF
C3,C9 = 22µF
C4,C5 = 47pF
C7,C8 = 33pF

Semiconductors
D1 = SMBJ5V0A
IC1 = FT232RL
IC2 = TPS3825-33
IC3 = MB90F352SPFV
IC4 = PCA82C251T
LD1 = LED, 3mm, low power, green
LD2 = LED 3mm, low power, red
LD3,LD4 = LED 3mm, low power, yellow
T1 = BSS84

Miscellaneous
Q1 = 4MHz quartz crystal
L1 = WE-742792116 SMD ferrite (Würth Electronics)
X1 = USB 2.0 type B in-equipment connector
X2 = 9-way sub-D plug (male)
J1,J2 = 4-way pinheader
J3 = 2-way pinheader
Kit of parts comprising PCB with premounted SMD parts and all other parts: Elektor SHOP no. 071120-71 (www.elektor.com)

Table 1. LED status indicators

LEDs		Description
LD1	**LD2**	
off	on	Firmware on the module has been started.
on	–	Module ready, no communications with the PC.
flickering	–	Active connection with the PC.
–	flashing	CAN bus status: 'Error Warning' - the FIFO receive buffer is full.
–	on	CAN bus status: 'Bus Off'.
LD3	**LD4**	
flickering/on	–	CAN bus message successfully received.
–	flickering/on	CAN bus message successfully sent.

and adapted via the *Filter* sub-menu. Messages can be filtered in three different ways:

- *single*: a CAN message with a certain Id is extracted from the data stream.
- *range*: messages with an Id between two programmable values ('Id start' and 'Id stop') are displayed.
- *masked*: the Id is filtered using a mask. Only those bits that have a '1' in the mask field (see **Figure 5**) are compared. The values of the other bits in the received message are ignored.

In the transmit list the values for the CAN Id and other data can be represented in several ways. A prefix is used to show how the data is displayed: 'x' stands for hexadecimal, 'd' stands for decimal, 'b' for binary and 'c' for ASCII. To change the display method you should click on the prefix with the mouse.

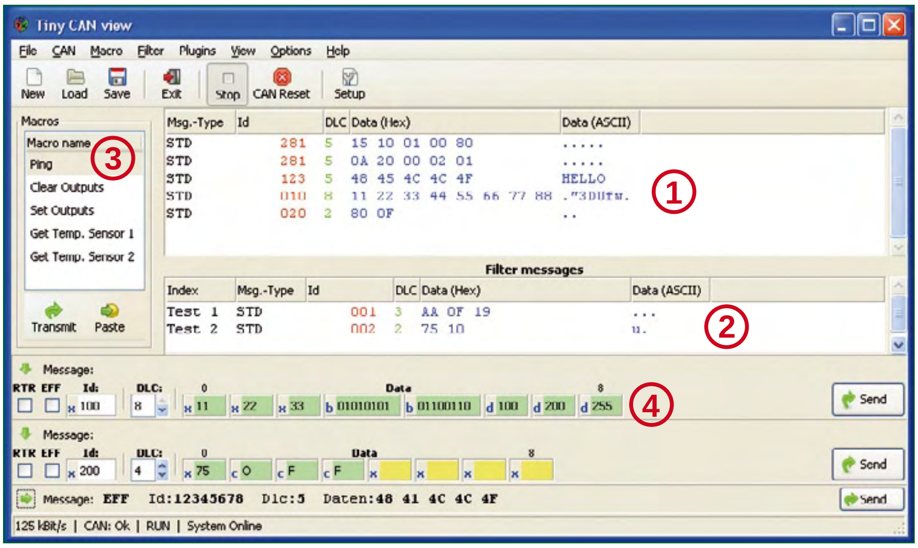


Figure 4. The relevant information is clearly laid out in the main window of the software.

Let's get started!

The module is supplied with all SMD components already mounted. Only the through-hole components need to be soldered on the board (**Figure 6**). When the USB connector has been mounted the controller can be programmed. But before you can do this, the driver for the FTDI chip (USB interface) has to be installed. Until this has happened you should **not** connect the module to the USB port. The most up-to-date driver can be downloaded from the FTDI website [1]. At the time of writing these are version 1.35r1 for Linux and version 2.04.06 for Windows, for which you can also download a *'setup executable'* called *CDM 2.04.04.exe*.

To program the microcontroller you first need to plug on programming jumper J2. Only then should you connect the USB cable (please note: J2 should NEVER be plugged on or removed while the USB cable is connected!). The computer will then detect the new hardware (in the case of Windows) and show it as a USB serial port.

Download the software from the Elektor website and extract the files from the zip file. Next run the program *TCanFirst* in the folder .../Tiny-CAN/fu_down/TCanFirst. This programs the Flash Bios of the module. After a message has appeared saying the flashing has completed successfully you can unplug the USB cable and remove the programming jumper (J2).
After reconnecting the USB cable the red LED should light up. You are now at the stage where you can program the actual firmware. For this you need to run the program *TCanProg* that is found in the folder .../Tiny-CAN/fu_down/TCanProg. When the green LED lights up you know that everything has completed successfully. For a future firmware update you only need to carry out the last action again (run TCanProg). The CAN monitor program can now be started.
Tiny CAN View is a CAN monitor program available for both Windows and

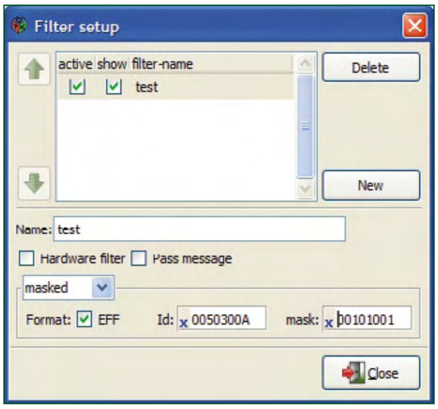


Figure 5. The filter parameters can be easily modified in the filter setup window.

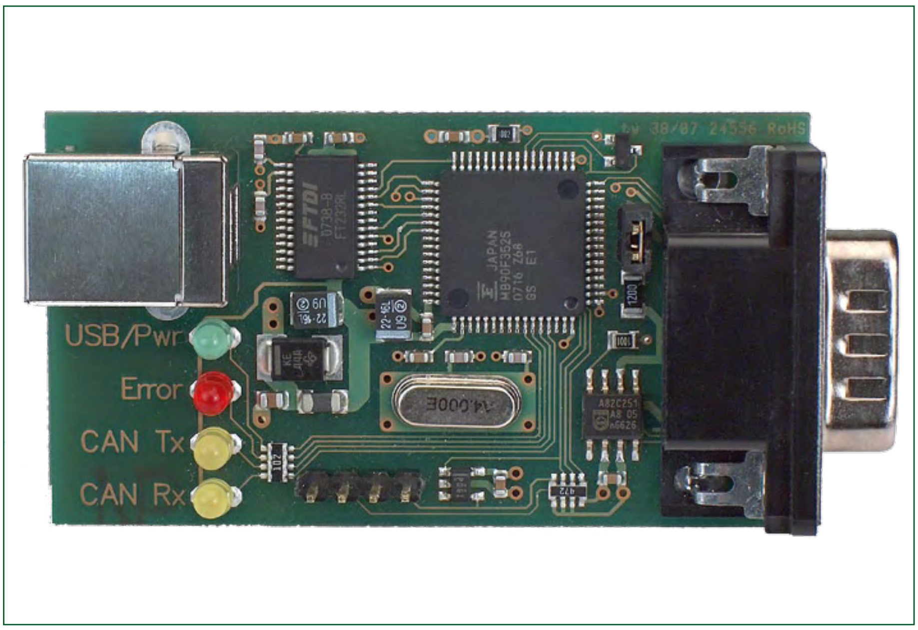


Figure 6. The module is really very compact and tidily laid out. The status LEDs show the current operational status.

Linux. It can be downloaded from the Elektor website via the link on the project page for this article. The program is a GNU Open Source project and has been written in C with MinGW/Gtk+ and it makes use of the GTK+ library, as we mentioned earlier. This can be downloaded from the link in [2]. Choose the *Development Environment Revision* and install the library.

Tiny-CAN View makes an automatic connection with the USB-CAN adapter when mhstcan1.dll is chosen as the driver. In the *CAN* tab in the setup menu you can select maximum data rate. The other tabs aren't required for the first run and can be ignored for now. As we mentioned earlier, the program makes use of filters to keep the data stream to a manageable level. The received data can also be stored in a file and CAN messages can be transmitted. Support for Standard (11-bit IDs) and Extended Frames (29-bit IDs) is built in.

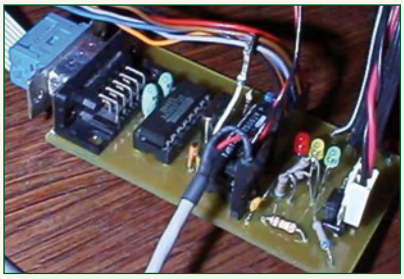
All required links and programs have been grouped together on the project page for this article on the Elektor website. From there it is easy to find all the software and drivers. The PCB layout for the circuit can also be found there.

For those of you who find the SMD packages too small to solder, the module is also available as a kit of parts from the SHOP section of our website www.elektor.com. This kit consists of a board that has al SMD components already mounted. You are therefore left to solder only the through-hole ('leaded') components.

(071120-I)

Some projects using CAN

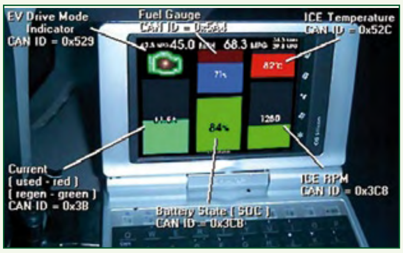
Home automation



http://caraca.sourceforge.net — CARACA stands for CAN Remote Automation and Control with the AVR. CARACA is a home automation project based on a network of individually programmable circuits. These circuits can carry out different tasks, such as switching devices on or off, decoding signals from IR remote controls, controlling thermostats, and so on. Each node in the network can communicate with any other node via the robust CAN protocol and the status can be monitored on a PC, which in turn can be connected to the Internet.

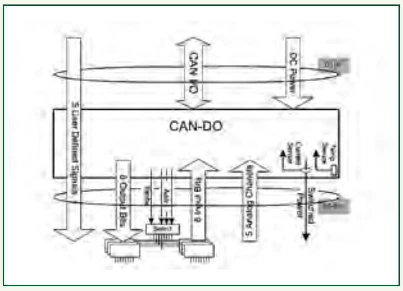
Toyota Prius

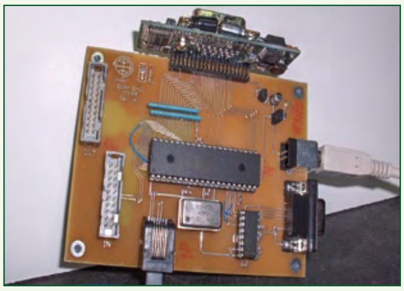
www.eaa-phev.org/wiki/Prius_PHEV_User_Interfaces — On this page the possibilities are discussed regarding the modification of the user interface and the State Of Charge Manipulation in the Toyota Prius. Such a device should be able to deal with the logic as described in the Prius PHEV Pseudo Code.



Satellite

http://can-do.moraco.info — CAN-Do! is a microcontroller (widget) that was designed for use as an interface to the wiring harness of a satellite and use this network to gain access to the integrated Housekeeping Unit that manages the different satellite subsystems. The primary aims are the reduction in the required amount of cabling and the simplification of the integration in a space ship.





Temperature control

www.ece.usu.edu/experiences/5770_projects/zone_heating_system_sp03/index.htm – On this site a system is described that has been developed by a group of students, and which controls the temperature in a number of separate rooms within a house. A computer program is used to set the temperature for several 'zones' and to view the current status. A dedicated controller keeps track of the status of each zone and turns on the heating or air conditioning when required.

The main controller communicates via a CAN bus with the zone controllers, which control the valves in the central heating system and which return the temperature to the main controller. The PC communicates with the main controller using a standard serial link.

Internet Links

[1] www.ftdichip.com

[2] http://gladewin32.sourceforge.net

[3] www.mhs-elektronik.de/tiny-can