RasPICER

Your automation partner for the **Raspberry Pi**

User manual

Contents

1. INTRODUCTION

The Raspberry Pi is a powerful Linux computer that has gained wide acceptance among technicians and hobbyists. However, in the automation arena this system has several drawbacks that hinder the integration possibilities.

First, although the Raspberry out of the box has a number of GPIO and a serial port, its use is limited since all these signals have TTL levels, requiring some kind of signal conditioning. Second, the Raspberry powers from a +5 VDC supply and has no power management.

The RasPICER board has been designed to provide a complete interface to the automation world. **RasPICER** name stands for **Ras**pberry **P**rogrammable **I**ndustrial **C**ontroller with **E**nergy **R**esources.

Communication between the Raspberry Pi and the RasPICER board is accomplished through the onboard SPI serial interface.

The RasPICER board includes:

- 8 digital inputs (12-24 VDC)
- 4 relay outputs
- 4 NPN transistor outputs
- 2 analog inputs (0-20 mA)
- 2 analog outputs (0-20 mA)
- 1 RS485 serial port
- 1 RS232 serial port
- Power manager (12-24 VDC)
- Rechargeable backup battery (3000 mAh)
- Real time clock
- Watchdog functions

The RasPICER power manager controls the Raspberry Pi supply. The entire system is powered supplying a voltage from 12 VDC to 24 VDC to the terminals VDC and GND, or by means of the backup battery.

This power manager is fully configurable to fit every need.

The intended working for the manager is to provide power to the Raspberry whenever a valid voltage is detected. When power is lost, the Raspberry will continue powered until the control application decides that is safe to power down, by issuing a Linux poweroff command. When the poweroff procedure is completed, the RasPICER hardware will detect it and remove power to the Raspberry. The system status will continue unpowered until the main voltage is restored.

A push button is present on the RasPICER board to allow a power control like a PC ATX supply. You can power on or off the Raspberry with this button.

The watchdog function, when active, monitors the Raspberry Pi and removes power when no activity is detected, ensuring this way that the main control application is always up and running.

For support please contact info@ferrariehijos.com or http://www.ferrariehijos.com/RasPICER.

2. REQUIREMENTS

The RasPICER board is compatible with the following Raspberry Pi versions:

- Raspberry Pi 1 A+
- Raspberry Pi 1 B+
- Raspberry Pi 2 B
- Raspberry Pi 3 B **(NOTE)**

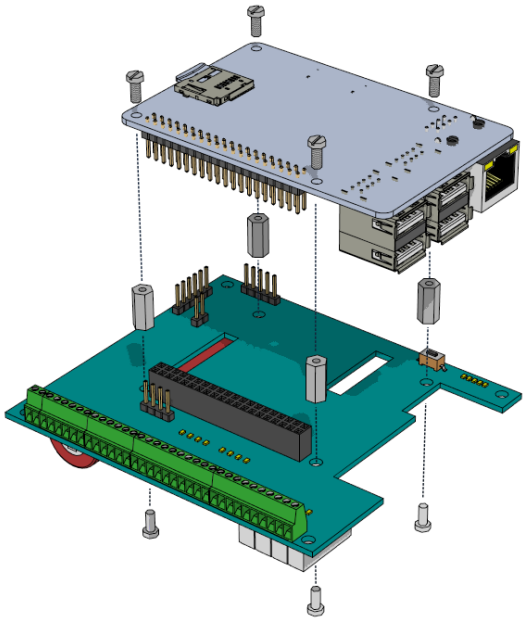
All software is compiled for the RASPBIAN JESSIE Linux distribution. Other distributions may be supported on request. Please contact us for other Linux flavors.

Important note for the Raspberry Pi 3 B: The RasPICER board needs TTL serial port signals (**serial0)** available in GPIO header in order to detect Raspberry power. On **Raspberry Pi 3 B** models, this port is disabled by default. **You need to enable this serial port BEFORE connecting it to the RasPICER board**. When this port is disabled, the RasPICER board will not detect Raspberry power ON state, resulting in an endless power cycle (power ON/OFF) loop. You can enable these port signals by appending **enable_uart=1** option in /boot/config.txt file, before installing the RasPICER board or with the Raspberry USB power connected.

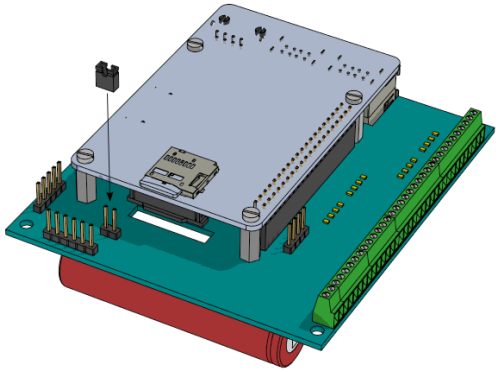
3. INSTALLATION GUIDE

NOTE: On Raspberry Pi 3 B model you **MUST** enable serial port **serial0** before connecting the RasPICER board. This port is disabled by default. Refer to section **2. Requirements** for details.

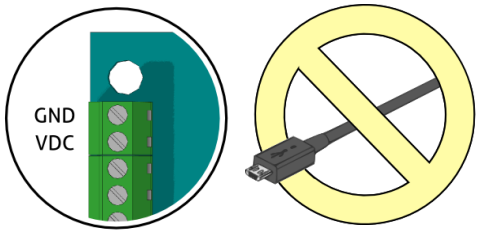
1 Mount the Raspberry Pi as indicated, using the included spacers and screws.

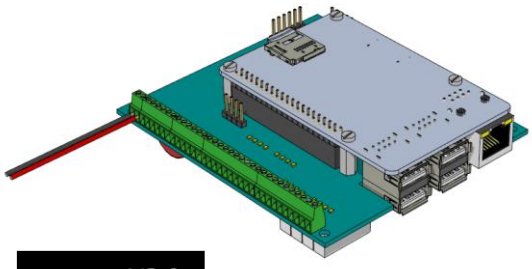


2 Put BAT jumper. BAT jumper is disconnected before shipping for security.



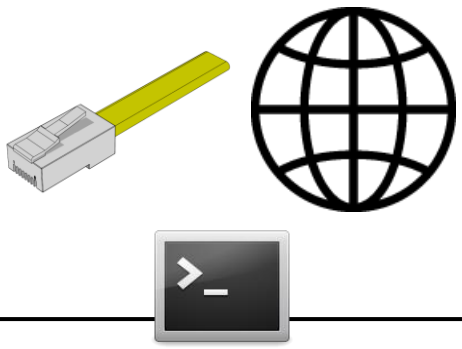
3 Power the RasPICER board using terminals 1 and 2. Do not power the Raspberry Pi from USB.





12...24 VDC

4 Provide Internet connection to the Raspberry Pi. Open a terminal and follow instructions.



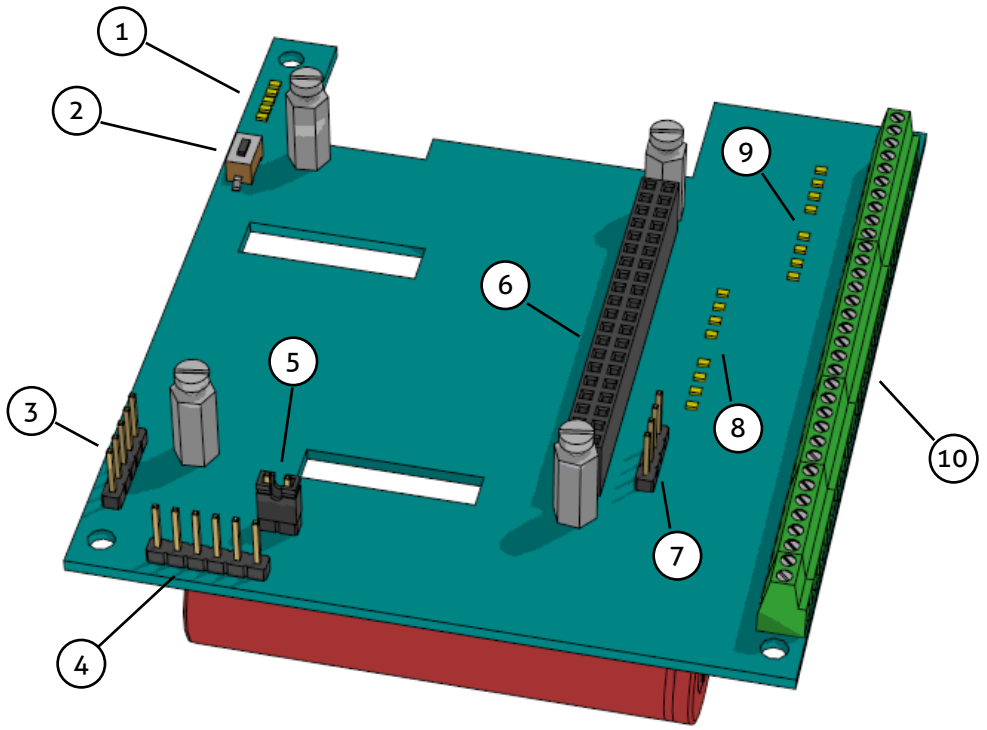
```
cd
wget http://www.ferrariehijos.com/raspsetup
chmod +x raspsetup
sudo ./raspsetup
```

< or install easyLadder PLC engine >

```
cd
wget http://www.ferrariehijos.com/easyladdersetup
chmod +x easyladdersetup
sudo ./easyladdersetup
```

4. SYSTEM HARDWARE

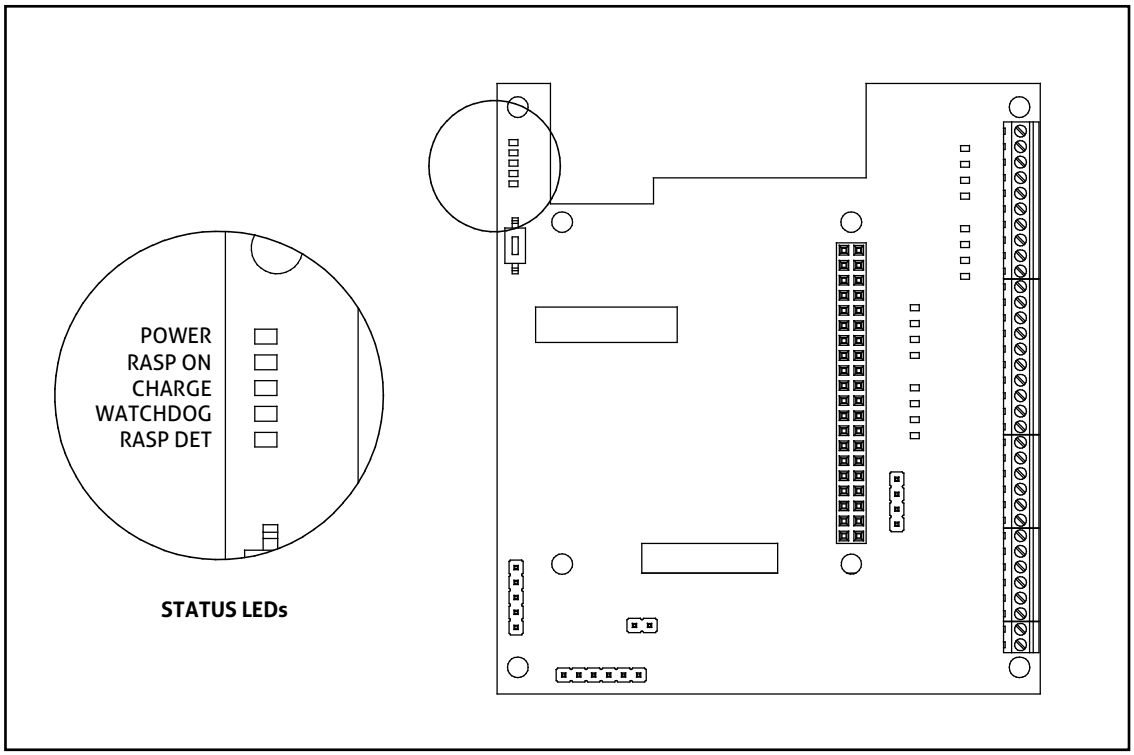
The following image shows important components on your RasPICER board:



Pos.	Description	Refer to section
1	Status LEDs	**section 4.1**
2	Power button	**section 4.12**
3	Expansion header	**section 4.2**
4	RasPICER ICSP header (for factory programming, do not use)	**section 4.3**
5	Battery header	**section 4.4**
6	Raspberry GPIO connector	
7	Raspberry ttyAMA0 TTL to RS232 header	**section 4.5**
8	Input monitoring LEDs	**section 4.8**
9	Output monitoring LEDs	**section 4.7**
10	RasPICER terminal block	**section 4.6**

4.1. Status LEDs

These LEDs monitors the status of the RasPICER board.



STATUS LEDs

POWER This LED indicates that the RasPICER board is powered through external VDC and GND terminals.

RASP ON Raspberry Pi power indicator. The +5VDC power supply is enabled (Raspberry Pi supply).

CHARGE Battery is charging from VDC power.

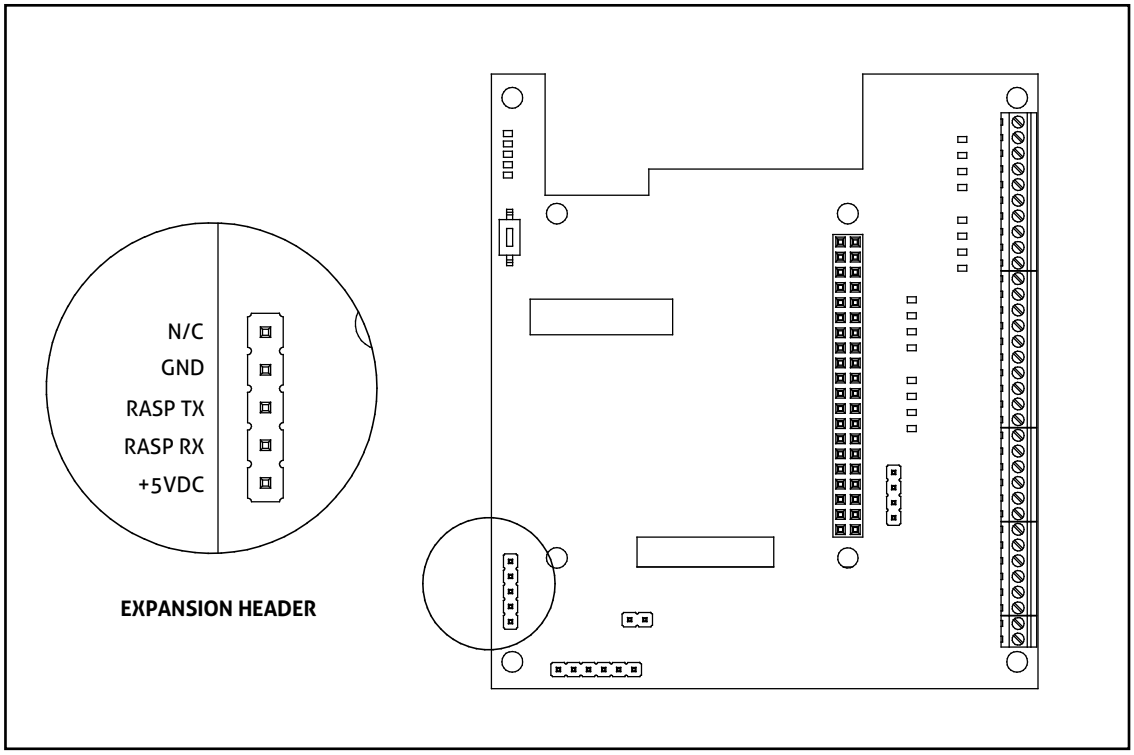
WATCHDOG This LED indicates serial SPI activity between RasPICER and Raspberry Pi. A blinking LED signals a temporally disabled watchdog function.

RASP DET Raspberry Pi run detection. When lit, the Raspberry is powered and running (i.e. not in a **poweroff** state).

4.2. Expansion header

The expansion header gives access to the Raspberry Pi TTL serial port (**ttyAMA0**) and **+5VDC** supply.

This header is compatible with **4D SYSTEMS** Intelligent Display Modules connector. You can connect directly any **4D SYSTEMS** Intelligent Module to this header, or other custom hardware requiring **+5VDC** power and serial TTL signals.



N/C Unconnected terminal.

GND Main GROUND signal.

RASP TX Raspberry serial port (ttyAMA0) 3V3 TTL TX signal.

RASP RX Raspberry serial port (ttyAMA0) 3V3 TTL RX signal.

+5VDC Main +5VDC power supply. This source powers the Raspberry Pi. This power supply is rated as +5VDC 2A.

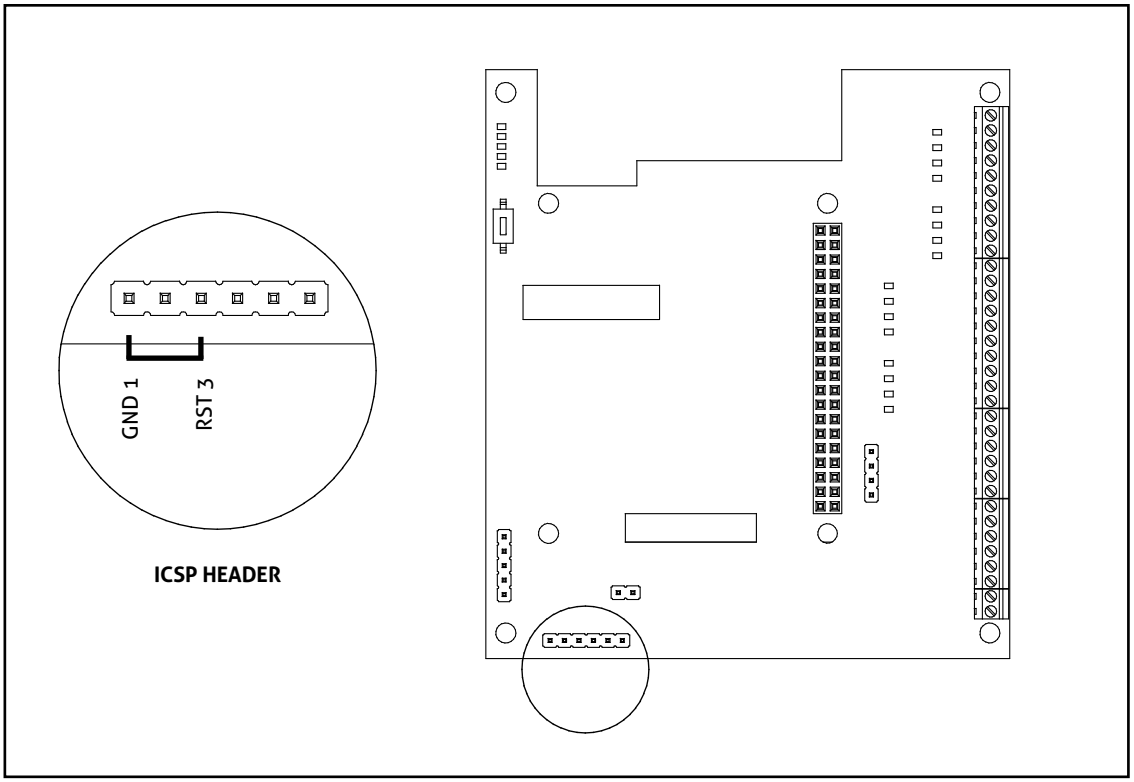
4.3. RasPICER ICSP header

This header is used for RasPICER CPU factory programming. **Do not use this header unless you want to recover a bricked RasPICER board**.

If your RasPICER board is not starting due to a firmware corruption, you can use this header to reset the RasPICER CPU. When starting the RasPICER after a system reset, if the Raspberry Pi is detected (powered by the USB header), the CPU will enter the **bootloader** mode for 5 seconds.

Using the **bootloader** mode you can reflash your RasPICER using a firmware utility. Please refer to http://www.ferrariehijos.com/RasPICER for firmware download.

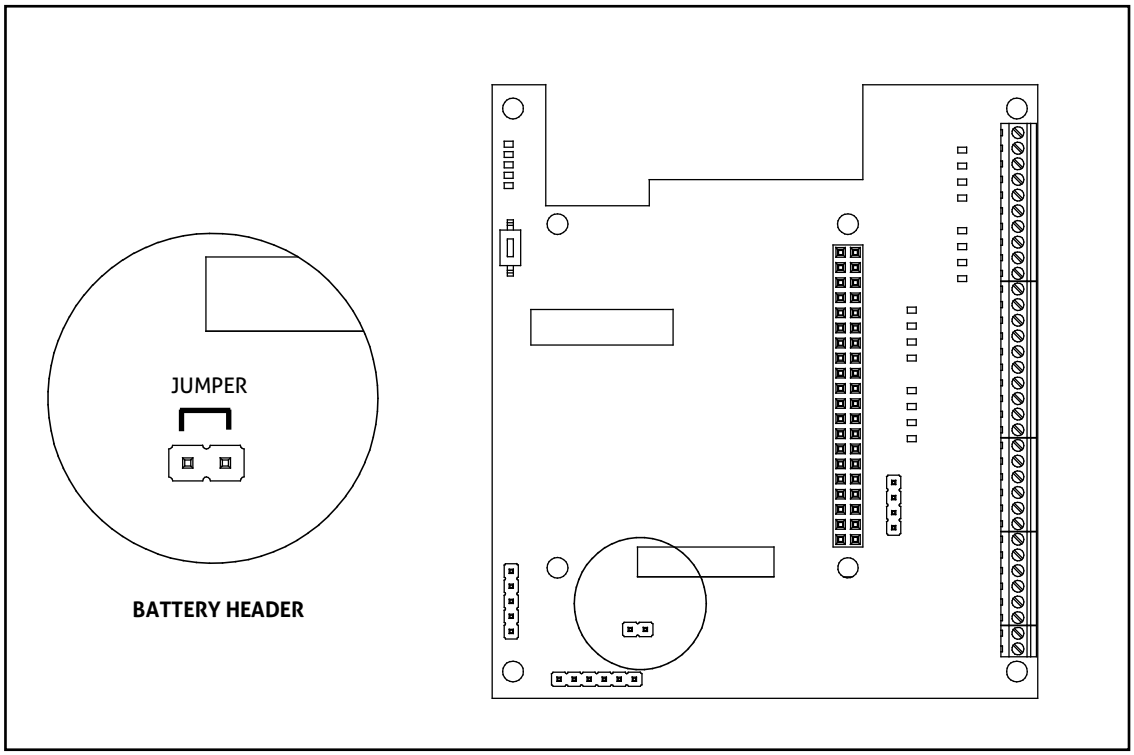
You can reset the RasPICER CPU by shorting pins 1 (GND) and 3 (RST).



4.4. Battery header

This header is used to connect and disconnect the RasPICER battery. Short these terminals using a jumper to apply battery to the RasPICER.

Take out the jumper if you need to stock your RasPICER board for a long time. This jumper is removed before shipment for safety reasons. **Please connect the jumper before use**.

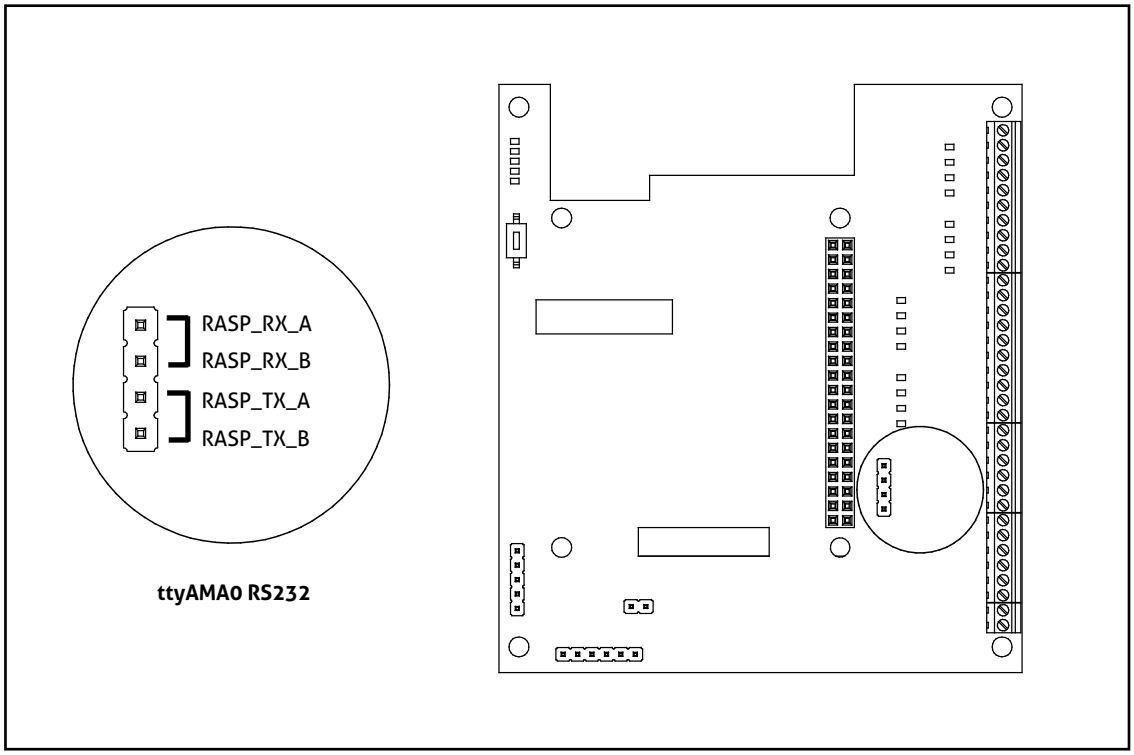


4.5. Raspberry ttyAMA0 TTL to RS232 header

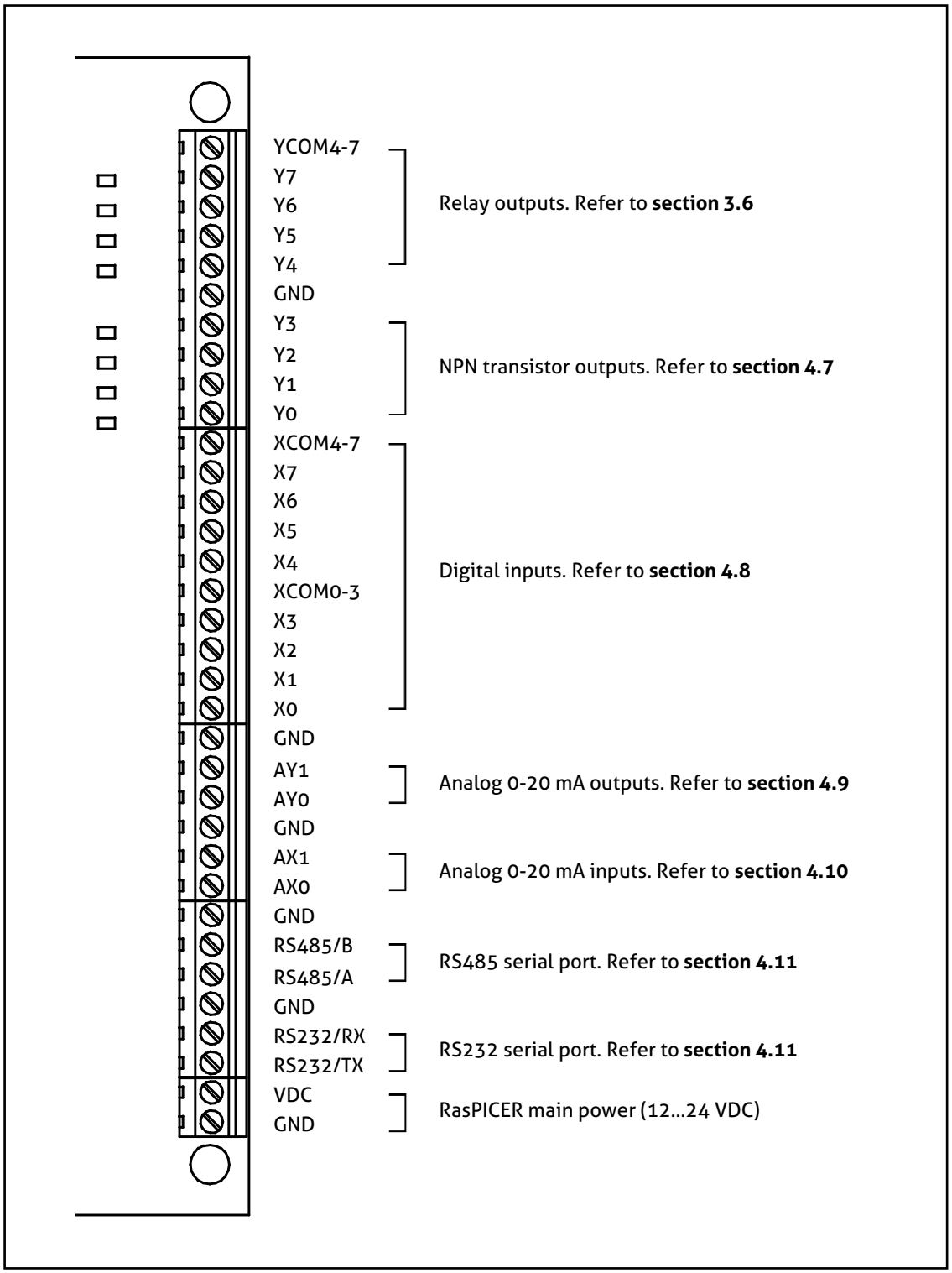
The RasPICER incorporates two serial ports: one RS485 port and one RS232 port. These ports are available in the terminal block and are fully independent from the existing serial port on the Raspberry Pi (ttyAMA0). You can access the ttyAMA0 port with 3V3 TTL levels using the Expansion port. Refer to section **4.2. Expansion Header** for more information.

Using this TTL to RS232 header, you can route TTL ttyAMA0 port signals to the RasPICER RS232 driver. Doing so the ttyAMA0 Raspberry port will have RS232 levels and will be present in the RasPICER RS232 terminals, but the additional RasPICER RS232 port will not be available anymore. **If you use (open) the RasPICER RS232 port when the ttyAMA0 is routed to the RS232 driver, serial data corruption will occur.** Nevertheless, this fact will not damage the hardware.

In order to convert ttyAMA0 TTL to RS232 levels, you must put a jumper between RASP_RX_A and RASP_RX_B and another jumper between RASP_TX_A and RASP_TX_B, as indicated in the image.



4.6. Terminal block



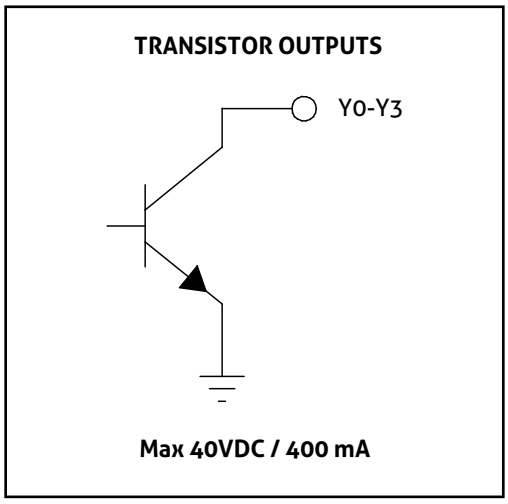
4.7. Digital outputs

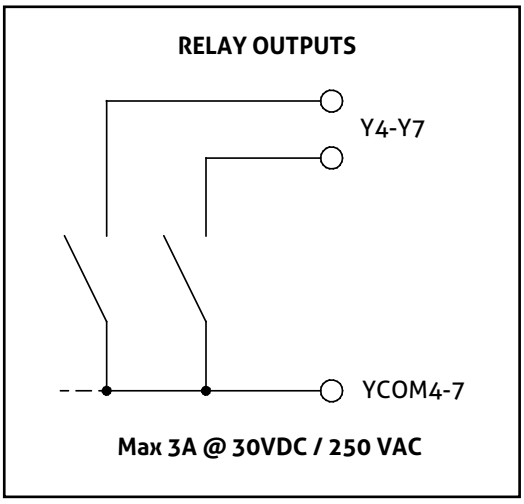
The board includes eight digital outputs, referred as **Y0** to **Y7**.

Outputs **Y0** to **Y3** are NPN transistor outputs. The image bellow shows a simplified schematic of the output circuits. Transistor max open voltage is **40 VDC**, and max current is **400 mA**.

Outputs **Y4** to **Y7** are relay outputs. All four outputs share the same common **YCOM4-7**. Relay simplified schematic is shown bellow. Relay is rated max **3A** at **30 VDC / 250 VAC**.

The LED close to the terminal indicates each output state.



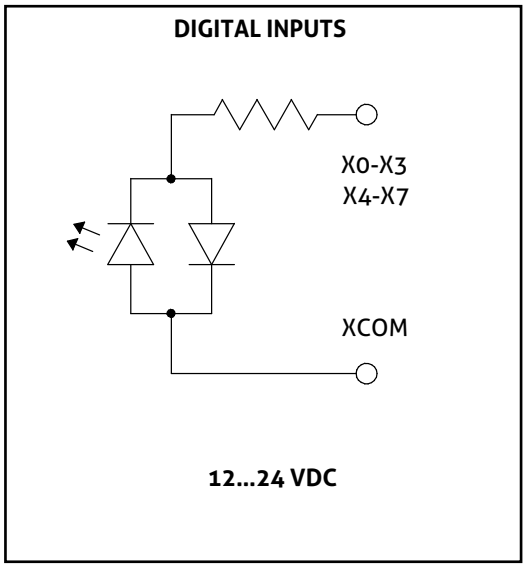


4.8. Digital inputs

The board includes eight optoisolated digital inputs, referred as **X0** to **X7**.

All inputs are dual NPN / PNP inputs. You can select NPN or PNP operation separately for inputs **X0** to **X3** and **X4** to **X7**. This layout gives you enhanced flexibility for mixing PNP and NPN sensors according to your needs. You can use 8 NPN inputs, 8 PNP inputs or 4 PNP and 4 NPN inputs, This selection is accomplished connecting **XCOM** either to VDC or to GND. Applying the opposite voltage to the **Xn** terminal energizes the input.

A simplified schematic for the input circuit is offered at the right.



Every input has a configurable input filter and provides a hardware low speed counter. The counter speed is limited by the input filter time.

The LED close to the terminal indicates each input state.

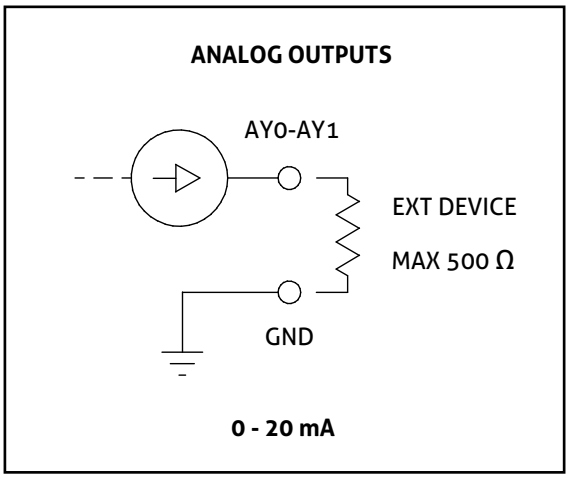
4.9. Analog outputs

The board includes two analog current outputs **AY0** and **AY1**.

These outputs can generate a settable current between 0 and 20 mA for governing external devices. Of course, you can use it to generate 4 to 20 mA also, if your external device requires it.

Outputs are controlled by a 12 bit DAC, and can drive a maximum impedance of **500 Ω**.

WARNING: Analog outputs are referenced to the MAIN GND. Please consider it when wiring the outputs.



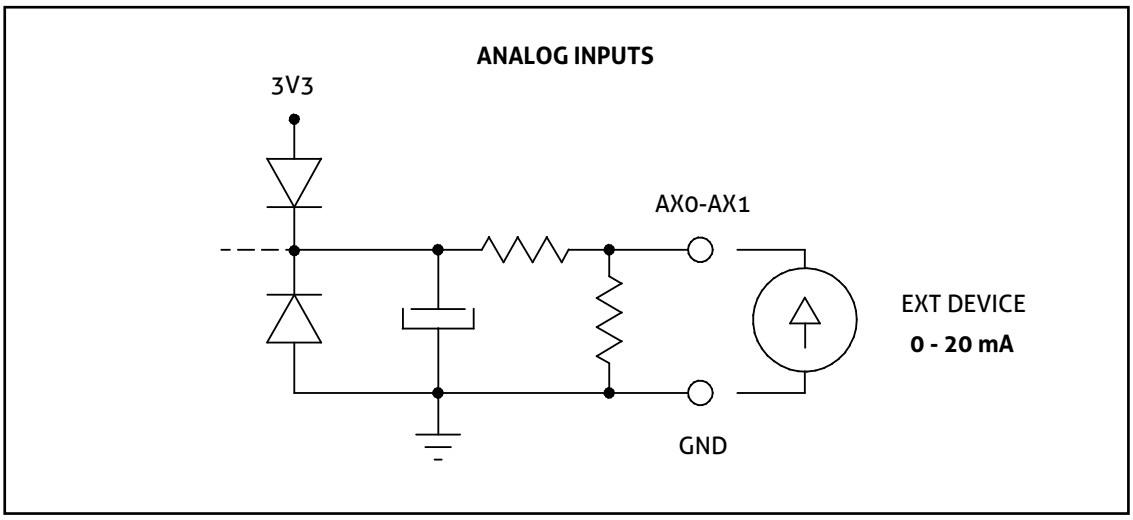
4.10. Analog inputs

The board includes two analog current inputs **AX0** and **AX1**.

These inputs can measure currents between 0 and 20 mA from external devices.

Analog inputs are captured by a 12 bit ADC. The RasPICER incorporates a simple but effective digital filter to suppress noises from 50Hz/60Hz power surges. You can shield the cable to GND at one end to provide additional noise immunity. Response time is 12,5 ms, but the filter settles in 100 ms after a large input change.

WARNING: Analog inputs are referenced to the MAIN GND. Please consider it before wiring. Inputs are protected from overvoltage and undervoltage with clamp diodes as indicated in the included schematic.



4.11. Serial ports

The RasPICER incorporates two serial ports: one RS485 port and one RS232 port. These ports are available in the terminal block and are fully independent from the existing serial port on the Raspberry Pi (ttyAMA0). You can access the ttyAMA0 port with 3V3 TTL levels using the Expansion port. Refer to section **3.2. Expansion Header** for more information.

These ports are configurable and accessible through the RasPICER support software. You can configure baudrates from **150** baud to **115200** baud, 7 or 8 bits, with or without parity control.

Both ports include a separate 250 bytes buffer for sending and 250 bytes buffer for receiving to allow some delay between port polling without data drop.

For your convenience, it is possible to convert ttyAMA0 TTL levels to RS232 levels, but losing access to the RasPICER RS232 port. In order to do so, you must put jumpers to the **ttyAMA0 header** as indicated in section **3.4. Raspberry ttyAMA0 TTL to RS232 header**.

4.12. Power manager

The RasPICER power manager controls the Raspberry Pi supply, ensuring in every case that the Raspberry and the controlling application is up and running.

When connected to the RasPICER, the Raspberry Pi no longer needs power from the included USB power adapter. Do not power the Raspberry from the USB connector unless you need to force power to the Raspberry for a RasPICER firmware upgrade or an incorrect watchdog setting. The hardware will not be damaged if you connect both supplies.

The onboard rechargeable 3000 mAh Li-ION battery ensures up to 6 hours of operation without power supply. This time varies depending on the attached equipment and workload on the Raspberry.

The battery is recharged when the RasPICER is connected to external power.

Complementing the hardware power manager, a *daemon* application for the Raspberry Pi is supplied as a part of the freely available support software. Some power manager features require this *daemon* running in the Raspberry Pi.

The power manager (hardware and software) includes the following features:

AUTO POWER ON When enabled, the Raspberry Pi will be automatically powered when a valid voltage is detected between VDC and GND terminals.

AUTO POWER OFF When enabled, the RasPICER will detect the Raspberry Pi **poweroff** state. When the **poweroff** procedure is completed, the RasPICER will remove power to the Raspberry.

AUTO SHUTDOWN When enabled and main power supply is removed from the RasPICER board, the Raspberry Pi will automatically issue a **poweroff** command to shut down the system, allowing any running application to save essential data before power down. This controlled power down is accomplished through the backup battery, and requires the included *daemon* or a custom application to issue the **poweroff** command.

POWER BUTTON

The RasPICER includes a power button (section **3. System Hardware**). This button is provided to control the supply of the Raspberry Pi.

When the system is switched off, this button starts the Raspberry.

When the system is on, this button informs the *daemon* or custom application to start a system shutdown by issuing the **poweroff** command.

A long press in this button will force a system power off, by removing power to the Raspberry.

Pressing the button three times will disable the RasPICER watchdog until a system power down or another triple press is accomplished.

The button double press is reserved for custom application usage. The RasPICER will inform the custom application that a double press was made, so it can perform some action.

WATCHDOG

You can configure the RasPICER to remove power from the Raspberry when a watchdog timeout occurs.

There are two options for reloading the watchdog: **SPI activity** or **Special watchdog reload** command.

When **SPI activity** is selected, any SPI command will reload the watchdog.

When **Special watchdog reload** command is selected, the watchdog will be reloaded only when this command is issued.

About watchdog timeout, you can configure two watchdog times:

The **startup time** is the time needed for the Raspberry Pi system to start, thus the maximum time after a power up required to issue the first watchdog reload. If no watchdog reload occurs passed this time from the power up, the system will restart. This time must be set according to your system, depending on the time required to launch your control application or *daemon*.

The **watchdog time** is the maximum time required between watchdog reloads, after the first (startup) reload. If no watchdog reload occurs passed this time, the system will restart.

When the watchdog is active and you need to stop the control application or *daemon*, or when your watchdog times are too short and your Raspberry is resetting continuously, you can temporally disable the watchdog engine by pushing the power button three times or by using the supplied **raspicer utility**. When the watchdog is temporally disabled the watchdog LED will blink. You can reactivate the watchdog by pressing the power button three times again, or with the **raspicer utility**

The watchdog function is factory disabled, to allow software install.

4.13. Real Time Clock (RTC)

The RasPICER includes a Real Time Clock engine, powered with the main Li-Ion battery. You can use this RTC to transfer a reliable clock to the Raspberry Pi, even without NTC servers.

The **raspicer utility**, supplied as part of the support package, lets you transfer the clock to or from the RasPICER board.

You can also use the **raspicer** *daemon* to automatically synchronize both clocks. During *daemon* launch, if synchronize option is enabled, the system will check if the Raspberry clock is synchronized with the NTC servers. If so, the time will be transferred to the RasPICER board. If not synchronized, the system time will be get from the RasPICER. After this startup, the *daemon* will monitor the system time to detect changes due to NTC update or manual clock change. When detected, and there's a significant change to the system clock, the new time will be transferred to the RasPICER board, ensuring an always synchronized RTC in the RasPICER board.

5. USING THE RASPICER

This section helps you to setup the Raspberry with the software needed to control your RasPICER.

The RasPICER software support package is designed to fit every scenario. Before you begin, it is very important to select your desired method to control the board.

The RasPICER software package includes a *daemon*. The *daemon* use is optional but is highly recommended. The **raspicer** *daemon* is an application that generally loads during system initialization as an *init.d* service.

This *daemon* helps you to control, when required, some functions of the RasPICER such as:

- **AUTO SHUTDOWN** feature (section **3.11. Power manager**)
- **POWER BUTTON** poweroff function (section **3.11. Power manager**)
- **RTC** automatic synchronization (section **3.12. Real Time Clock**)
- **WATCHDOG** automatic reload (section **3.11. Power manager**)

The most important function of the *daemon* is to access low-level registers to operate the RasPICER board. For this reason, the *daemon* must run with superuser privileges.

When the *daemon* is running, other applications using RasPICER functions do not need low-level access anymore. Communication with the RasPICER is done through this *daemon* by means of UNIX domain sockets. So superuser privileges are not needed for the controlling applications. Furthermore, several controlling applications can access the RasPICER simultaneously by means of the *daemon*.

To control the board features you can choose one of the following methods:

- Use the **raspicer utility** from the command line (terminal), with or without *daemon*. Using this utility you can configure the board, set outputs and get inputs.
- Create your own program in **C** or **Python** using the included source code and libraries, either accessing low-level registers or through the *daemon*.
- Convert your RasPICER and Raspberry into a full featured PLC with our **easyLadder** software. **easyLadder** is a complete PLC software engine for the Raspberry with a graphical ladder editor designed for *Microsoft Windows*.

  easyLadder includes his own *daemon*, replacing the general *raspicer daemon*. This *daemon* lets you to access every RasPICER feature and additionally gives control of the embedded PLC variables (devices). It is also possible to use our libraries and source code to write a program to interact with the PLC program.

  You can find information about **easyLadder** at http://www.ferrariehijos.com/easyLadder.

5.1. Software setup

If your Raspberry Pi has connection to the Internet, you can download our software installer to ease the entire process. To get the installer, open a local terminal or a remote SSH session to your Raspberry and type the following commands:

```
cd
wget http://www.ferrariehijos.com/raspsetup
chmod +x raspsetup
sudo ./raspsetup
```

The installer will query for your desired configuration and download the required files. Support package will be installed in **/opt/effesoftware**.

All files are compiled for the RASPBIAN JESSIE, our preferred Linux distribution. Please contact us if you want to use other Linux distribution.

If your Raspberry is not connected to the Internet, or you want to do a manual setup, you can download all files and sample source code from:

http://www.ferrariehijos.com/RasPICER

5.2. The raspicer utility

The **raspicer** utility is provided as a part of the board software support package. This is a command line utility used to monitor the status of the board, set outputs, adjust clock, configure hardware values and run the *daemon*.

When using the *raspsetup* installer, raspicer utility is copied to the **/opt/raspicer** directory.

The utility will use direct access to the low-level registers when no *daemon* is running. For this reason you must run it with superuser privileges (using *sudo* command, for example) unless the *daemon* is already running on the system.

You can use the **raspicer** utility to load the *daemon*, but the recommended procedure is to load the *daemon* as an *init.d* service using the provided *raspicerd* script file. The *raspsetup* installer copies this script to the **/etc/init.d** directory for you, and configures the system to load it on system startup, if you selected to use *daemon* during install.

You can start or stop manually the *raspicer daemon* service using:

```
sudo service raspicerd start
-or-
sudo service raspicerd stop
```

Please note that stopping the *daemon* when the watchdog function is active can result in a system poweroff due to a watchdog timeout. Disable the watchdog before stopping using **-w0**

parameter or doing three power button presses. The watchdog LED indicator will blink when watchdog is temporally disabled.

Basically, on a default setup, the *raspicerd* script runs the *daemon* by issuing the following command:

/opt/effesoftware/raspicer -q -p3 -y --startdaemon

This command line indicates to hide messages (**-q**), enable **POWER BUTTON** and **AUTO SHUTDOWN** features (**-p 3**), synchronize clocks automatically (**-y**) and start the daemon (**--startdaemon**). You can edit the */etc/init.d/raspicerd* script if you need to change these default parameters by modifying DAEMON_ARGS variable at the start of the file.

The available command line parameters for the raspicer utility are:

Usage: **raspicer** [option ...]

General options:

-s, --status Displays board I/O or PLC status information.

-q, --quiet Runs raspicer in quiet mode. No text is displayed.

-h, --help Shows command line help information.

-w0, --watchdog0 Disables RasPICER shutdown due to watchdog timeout. When the watchdog is active and you need to stop the control application or daemon, you can temporarily disable the watchdog engine by using this command. When the watchdog is temporarily disabled the watchdog LED will blink. You can reactivate the watchdog issuing the –w1 command. Refer to section **3.11. Power manager** for more information.

-w1, --watchdog1 Re-enables RasPICER shutdown due to watchdog timeout. Refer to section **3.11. Power manager** for more information.

-r, --reloadwd Reloads RasPICER special watchdog counter. Refer to section **3.11. Power manager** for more information.

-p PAR, --powerctrl PAR Sets RasPICER power control options (Daemon is required). Refer to section **3.11. Power manager** for more information.

PAR value is set according to the following bits:

bit0: Controls RasPICER **POWER BUTTON** feature. When set, system will shutdown (poweroff) on RasPICER button press.

bit1: Controls RasPICER **AUTO SHUTDOWN** feature. When set, system will shutdown (poweroff) if no power is detected.

bit2: Automatic RasPICER contact. When set, RasPICER daemon will automatically contact RasPICER board every second in order to reload normal watchdog, without need of external applications. Setting either bit0 or bit1 has the same effect because the *daemon* will query the RasPICER board to know the status of the button or power supply.

WARNING: When watchdog is enabled and no power option is configured (PAR=0), the system will power down if no external

application is reloading the watchdog, even when a Daemon is running.

-c, --viewconfig Displays RasPICER flash configuration parameters. See parameters bellow.

-f CFG, --setconfig CFG Sets RasPICER flash configuration parameters. New parameters are saved to flash memory.

Use '--setconfig parameter1=value1,parameter2=value2, ...' format.

Valid parameters are:

DisableAutoPowerOn (0-1). This parameter configures the RasPICER board to power the Raspberry when external VDC power is applied to the terminals (**AUTO POWER ON** power manager function). A value of 1 disables the function.

DisableAutoPowerOff (0-1). This parameter configures the RasPICER to remove power to the Raspberry when the **poweroff** state is detected (**AUTO POWER OFF** power manager function). A value of 1 disables the function.

DisableWatchdog (0-1). This parameter configures the RasPICER to remove power to the Raspberry when a watchdog timeout occurs (**WATCHDOG** power manager function). A value of 1 disables the function.

UseWatchdogCommand (0-1). This parameter configures the preferred method to reload the watchdog. Using a value of 0, any SPI command reloads the watchdog. Using a value of 1, the special watchdog reload command is required.

StartupTime (0-65535). This parameter specifies the time (in 100 ms units) needed for the Raspberry Pi system to start, thus the maximum time after a power up required to issue the first watchdog reload. If no watchdog reload occurs passed this time from the power up, the system will restart. This time must be set according to your system, depending on the time required to launch your control application or daemon.

WatchdogTime (0-65535). This parameter specifies the maximum time (in 100 ms units) required between watchdog reloads, after the first (startup) reload. If no watchdog reload occurs passed this time, the system will restart.

InputFilter (0-255). This parameter specifies the time filter value (in 1,54 ms units) for digital inputs.

CalibOutCurrent0 and **CalibOutCurrent1** (0-65535). These parameters are used to calibrate analog current outputs AY0 and AY1. In order to calibrate the outputs, set the analog output to generate 20 mA. Increasing the parameter decreases the output current.

CalibInCurrent0 and **CalibInCurrent1** (0-65535). These parameters are used to calibrate analog current inputs AX0 and AX1. In order to calibrate the inputs, connect a 20mA analog source to the input. Increasing the parameter decreases the read current.

--startdaemon Starts RasPICER controller daemon.

--stopdaemon Stops RasPICER controller daemon.

RasPICER I/O options:

--yN VAL Set value for RasPICER board digital output N.

N is a number between 0 and 7. VAL is 0 or 1.

--ayN VAL Set value for RasPICER board analog output N.

N is a number between 0 and 1.

--rcxN Reset counter value for RasPICER board digital input N.

N is a number between 0 and 7.

-d VAL, --do VAL Sets value for entire RasPICER board digital output.

VAL is a byte representing the required status for the output. Bit 0 represents Output 0 and so on.

Example: 'raspicer -do 255' sets all outputs to ON state.

RasPICER RTC options:

-v, --viewrtc View current clock from RasPICER RTC.

-g, --getrtc Transfer RasPICER RTC to system clock.

-t, --setrtc Transfer system clock to RasPICER RTC.

-y, --syncrtc Automatic RTC synchronization.

The Automatic RTC sync allows an easy and automated way to synchronize both the system clock and the RasPICER RTC to obtain an accurate clock in every case.

First, when this command is executed, the system synchronizes both clocks based on the status of the NTP. When NTP is not synchronized, the RTC from the RasPICER is transferred to the system clock. When NTP is synchronized, the system clock is transferred to the RasPICER RTC.

After that, when any *daemon* is available, the system clock is continuously monitored to evaluate changes (due to manual clock change or NTP updates). If a large change is detected, the new system clock is transferred to the RasPICER RTC, so it will be available on subsequent system restarts, even when there is no NTP server available.

easyLadder options (a running easyLadder PLC engine is required):

--plcrun	Places easyLadder PLC in RUN mode.
--plcstop	Places easyLadder PLC in STOP mode.
--plcsetbit PAR	Sets PLC bit device value. Use '--plcsetbit Device1=Value1,Device2=Value2, ...' format.
--plcsetword PAR	Sets PLC word device value. Use '--plcsetword Device1=Value1,Device2=Value2, ...' format.
--plcgetbit PAR	Gets PLC bit device value. Use '--plcgetbit Device1,Device2, ...' format.
--plcgetword PAR	Gets PLC word device value. Use '--plcgetword Device1,Device2, ...' format.
--unblocktcpip	Disables PLC TCP/IP security. When you restricted the allowed incoming TCP/IP addresses for easyLadder Studio, and the easyLadder engine is no longer accessible, you can execute this option, so the TCP/IP security will be disabled.

5.3. Programming library

The RasPICER software package contains libraries and source code to ease your custom development project. **C** and **Python** languages are supported.

With these libraries, you can contact the RasPICER hardware either directly (with superuser privileges) or communicate with the *daemon* using UNIX domain sockets. This choice is made transparently to the programmer during library initialization depending on the availability of the *daemon*.

When using **C** language, you will need to add the **raspicer.c** source file to your project and include a **raspicer.h** header. The only requirement is to call **raspicerInit ()** function before using the library.

When using **Python**, you will find the source code for the Python module, written in C language. To build and install this module follow these instructions:

```
cd /opt/effesoftware/python
python setup.py install --user
```

In order traspicero build and install this module for all users follow these instructions:

```
cd /opt/effesoftware/python
sudo python setup.py install
```

To use the raspicer module, simply **import raspicer** in your Python program.

Refer to http://www.ferrariehijos.com/RasPICER for more details about available functions.

5.4. Programming and watchdog considerations

When designing your control application probably you will consider the use of the watchdog feature. Thanks to the watchdog you can ensure that your control code is always up and running. You could use the *daemon* and let it recharge the watchdog, but this does not guarantee the operation of your control loop. Alternatively, you can use two approaches:

- Do not use the *daemon* and configure the RasPICER to reload watchdog with any SPI command (**UseWatchdogCommand** flash parameter to 0). In this scenario you have to use a thread or other multitasking approach to continuously poll the status of the inputs and control yourself the **POWER BUTTON** and **POWER SUPPLY** to shutdown the system, if required, as shown is this pseudocode:

```
initsystem ();
RUN=1;
while (RUN) {
        if (...) processinputs ();
        if (...) setoutputs ();
        getraspicerstate ();
        if (!POWERDETECTED) || (BUTTONPRESSED) RUN=0;
        sleep (...);
}
savedata ();
poweroff ();
```

- Use the *daemon* and configure the RasPICER to reload watchdog with the special watchdog reload command (**UseWatchdogCommand** flash parameter to 1). In this scenario you have to use a thread or other multitasking approach to continuously reload the watchdog. You can let the *daemon* control the **POWER BUTTON** and **POWER SUPPLY** to shutdown the system, as shown is this pseudocode:

```
initsystem ();
RUN=1;
while (RUN) {
        if (...) processinputs ();
        if (...) setoutputs ();
        reloadwatchdog ();
        if (SIGTERM)  RUN=0;
        sleep (...);
}
savedata ();
return;  // exit app
```

ferrari e hijos, s.a.

http://www.ferrariehijos.com/easyLadder
info@ferrariehijos.com