GPIO

Contents

Introduction

Support for the standard GPIO (http://elinux.org/RPi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29) is available in Domoticz.

What is GPIO?

IT stands for "general purpose input output". Many small computers like the Raspberry Pi nowadays have a double row of pins that you can control. You can listen to input on them, or you can send signals to other devices.

- Not all of the pins can be controlled. Any standard Raspberry Pi features 17 digital In/Out (or even 21 on revision 2 boards) that can now be managed by Domoticz.

- The pins can only handle on-off signals. Meaning "on" (3.3v) or "off" (0v)

- You can damage your pi if you put a 5v input on one of these pins. You could buy a PiFace board, which has the advantage of giving a bit of protection to the inputs and of conveniently packing the relays and solderless terminal blocks on a clean board. This is also supported in Domoticz.

- GPIO pin numbers are different from the physical pin number. For example, physical pin 11 is PGIO pin 16. This can be confusing.

- GPIO pins have to be 'exported' by a root user if you want them to be accessible to non-root users. This is done by calling specialized commands which we will get into.

Two options

Domoticz currently supports two ways of accessing the pins:

1. SYSFS GPIO. This is a standardised way of connecting to pins that is used on a lot of devices, and can be used without a driver. **It is the prefered method within Domoticz**.
2. The CLASSIC way is to install a driver, like WiringPi or PiGPIO.

This page explains how you can use both.

Option 1 - SYSFS

SysFS exposes the pins as if they were files on your harddrive. by writing to these files you can send signals. By reading the files you can listen to signals.

You don't have to install any additional software for SYSFS to work. But you will have to setup the pins for your use.

Using to GPIO map here RPI GPIO Maps (https://elinux.org/RPi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29%7C) choose the desired I/O

Example :

1. Find which GPIO pin you want to use. Figure out the GPIO number of the physical pin by looking at the maps here : RPI GPIO Maps (https://elinux.org/RPi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29%7C)
2. Export the GPIO pin (number 4 in this example) by calling this command:
   echo 4 > /sys/class/gpio/export
3. Decide if the pin should be input or output:
   echo out > /sys/class/gpio/gpio4/direction
4. You can manually set the output pins on or off using this command. One is on, zero is off.
   echo 1 > /sys/class/gpio/gpio4/value echo 0 > /sys/class/gpio/gpio4/value

   > **TIP:** You have to give these commands every time you start linux, so it's smart to add these commands to a file like /etc/rc.local which gets called when the Pi boots. ("sudo nano /etc/rc.local"), or to the domoticz.sh file which gets called every time Domoticz starts.

5. set the interrupt if needed choice : rising / falling / both / none
   echo none > /sys/class/gpio/gpio4/edge

6. Now, go to Domoticz' hardware page and install the **Generic SYSFS GPIO** hardware.

   select **auto configure device**, the exported GPIO will appread in the **device** section ;-)

7. 1. first way : Go to the "switches page" and add a new manual switch.

      Select SYSFS as the option in the type dropdown.
      You can set a unique number for the pin inside domoticz using the many dropdowns here. You can try just keeping them all at 0 for your first ever pin.
      The SYSFS dropdown should show the pin you exported earlier. Select it.

   2. second way : Go to menu Setup - Devices

      all the exported GPIO are listed here
      enable the devices (click on the green arrow)

Enjoy ;-)

Option 2 - CLASSIC

The classis way of using GPIO pins in Domoticz requires you to

1. Install a driver
2. Export the pins.

Installation

- The GPIO driver is based on the wiringPi library (http://wiringpi.com/) by Gordon Henderson. Install it like so;

```
sudo apt-get install wiringPi
```

and test the installation with;

```
gpio readall
```

If that doesn't work; try this this procedure (http://wiringpi.com/download-and-install/)

GPIO pins

First, you need to get familiar with the GPIO pins and select which pins are suitable for the goal you want to achieve. I could not explain it better than Gordon, so please refer to those two pages: general considerations (http://wiringpi.com/pins/) and special pin functions (http://wiringpi.com/pins/special-pin-functions/). Then you can select which pins you are going to use and configure them.

Configuration

Before starting Domoticz

The pins you intend to use must be made available to unpriviledged processes (as Domoticz should not run with root priviledges). To do so, you need to run a few commands **before starting Domoticz**. Note that their effect is not permanent, so they should be performed upon each boot before Domoticz starts. Probably the best place is in /etc/rc.local or in the domoticz.sh script. The commands are as follows: (Note: the <BCM PIN #> below refers to the corresponding column on this page (http://wiringpi.com/pins/))

For outputs

```
gpio export <BCM PIN #> out
```

e.g.

```
gpio export 17 out
```

For inputs

```
gpio export <BCM PIN #> in
gpio edge <BCM PIN #> both
```

e.g.

```
gpio export 18 in
gpio edge 18 both
```

Check

Once it's done, you can check the configuration with the following command:

```
gpio exports
```

This should return something like:

```
GPIO Pins exported:
  17: out  0  none
  18: in   0  both
```

Configure at startup

Because the command is not permanently, you likely want to configure the GPIO settings at startup automatically.
You can do so by adding the lines you would like to use to the 'rc.local'.
Type: `sudo nano /etc/init.d/domoticz.sh` to edit the file Go to the end of the file, and just add after ':' place your commands.
In my case the /etc/init.d/domoticz.sh looks like this:

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          domoticz
# Required-Start:    $network $remote_fs $syslog
# Required-Stop:     $network $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Home Automation System
# Description:       This daemon will start the Domoticz Home Automation System
### END INIT INFO

# Do NOT "set -e"

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin
DESC="Domoticz Home Automation System"
NAME=domoticz
USERNAME=pi
DAEMON=/home/$USERNAME/domoticz/$NAME
#DAEMON_ARGS="-daemon -www 8080 -sslwww 443 -log /tmp/domoticz.txt"
#DAEMON_ARGS="-daemon -www 8080 -sslwww 443 -syslog"
DAEMON_ARGS="-daemon -www 8080 -sslwww 443"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.2-14) to ensure that this file is present
# and status_of_proc is working.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
        # Return
        #   0 if daemon has been started
        #   1 if daemon was already running
        #   2 if daemon could not be started
        start-stop-daemon --chuid $USERNAME --start --quiet --pidfile $PIDFILE $
                || return 1
        start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON -- \
                $DAEMON_ARGS \
                || return 2
}

#
# Function that stops the daemon/service
#
do_stop()
{
        # Return
        #   0 if daemon has been stopped
        #   1 if daemon was already stopped
        #   2 if daemon could not be stopped
        #   other if a failure occurred
        start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile $PIDF$
```

```
        RETVAL="$?"
        [ "$RETVAL" = 2 ] && return 2
        # Wait for children to finish too if this is a daemon that forks
        # and if the daemon is only ever run from this initscript.
        # If the above conditions are not satisfied then add some other code
        # that waits for the process to drop all resources that could be
        # needed by services started subsequently.  A last resort is to
        # sleep for some time.
        start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec $D$
        [ "$?" = 2 ] && return 2
        # Many daemons don't delete their pidfiles when they exit.
        rm -f $PIDFILE
        return "$RETVAL"
}

case "$1" in
  start)
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_start
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  stop)
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
        do_stop
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  status)
        status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
        ;;
  restart)
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
          0|1)
                do_start
                case "$?" in
                        0) log_end_msg 0 ;;
                        1) log_end_msg 1 ;; # Old process is still running
                        *) log_end_msg 1 ;; # Failed to start
                esac
                ;;
          *)
                # Failed to stop
                log_end_msg 1
                ;;
        esac
        ;;
  *)
        echo "Usage: $SCRIPTNAME {start|stop|status|restart}" >&2
        exit 3
        ;;
esac

:
#Run export GPIO Raspberry

/usr/bin/gpio export 17 out
/usr/bin/gpio export 27 in
/usr/bin/gpio edge 27 both
/usr/bin/gpio export 22 in
/usr/bin/gpio edge 22 both
```

Active low relays

If you are using a relay board, some of them are active low. This means that writing a 0 will turn the relay on instead of off, so in Domoticz you will see the light bulb off instead of on and vice versa. To solve this we have to add this line at the end of /etc/init.d/domoticz.sh (after the exports)

```
sudo sh -c "echo '1' >> /sys/class/gpio/gpio17/active_low"
```

Copy this for all the relay outputs, and change the bcm pin number (17 in the example) for the pin you are actually using.

Now issue a reboot (`sudo shutdown -r now`)

Inside Domoticz

You need first to declare the GPIO port in Domoticz. To do so, go to the "Hardware" tab, type in a name like "GPIO Port" and select "Raspberry's GPIO Port" as type, then click add.
Next, go to the "Switches" tab, and for each GPIO pin, click "Manual Light/Switch" and proceed as follows:

For outputs

Select the GPIO port hardware you just configured, give it a name, select the "On/Off" switch type and a "GPIO" type. You can now select the pin in the list. Obviously, the ones you configured above should not start with the "! NON EXPORTED" mention. You can test the output if you like (the "Test" button will just set the output to HI), and then click "Add Device".

For inputs

Select the GPIO port hardware you just configured, give it a name, select the "Contact" switch type and a "GPIO" type. You can now select the pin in the list. Obviously, the ones you configured above should not start with the "! NON EXPORTED" mention. Then click "Add Device". The mention "open/closed" corresponds to a switch connected between the pin and the ground, with the pin also connected to a pull-up (as you have read on the wiringPi website, some GPIO have an internal permanent pull-up.

Check

You should now be able to click on the light bulb to toggle the outputs, and you should see the contact status change when you connect the pin to GND or VCC.

Have fun!
