Recent site activity

Nivel Intermedio
edited by Holguer Becerra

Modulo SVGA
edited by Holguer Becerra
attachment removed by Holguer Becerra

Nivel Avanzado
edited by Holguer Becerra

DE0-NANO
attachment from Holguer Becerra
edited by Holguer Becerra
attachment removed by Holguer Becerra

STRENGTH CONTROL SYSTEM AND HAPTIC INTERFACE FOR A PROTOTYPE OF HAND ELECTROMYOGRAPHIC PROSTHESIS
attachment from Rodrigo Mancipe

View All

How to use GPIOs on raspberry pi (Simple I/O, PWM and UART)

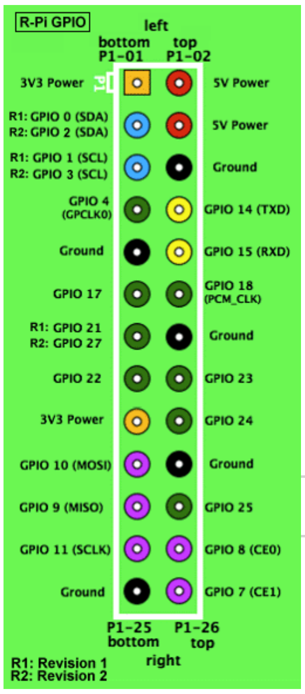
How to use GPIOs on raspberry pi (Raspbian-Wheezy)

Taka a look at the GPIO header information of the raspberry pi, you can find it in the next links:

```
http://elinux.org/RPi_Low-level_peripherals
http://elinux.org/RPi_BCM2835_GPIOs
```

Note: The GPIO assignment for the raspberry pi is different between revision 1 and 2.

Do not use voltage levels greater than 3.3V, Raspberry pi doesn´t support 5V and doesn't have an over-voltage protection.



There are two different methods to write to or read from peripherals on embedded systems using Linux, the first one is creating a file-type access to the peripheral in the file system and the second is to write/read the base address of the memory allocated to the GPIO or module in the SoC usign pointers. This memory locations can be found in the datasheet for the BCM2835 in the case of the raspberry pi.

Let's start with the first method (File System):

*Before starting startx
Use SuperUser (After every reboot) or use sudo before any command

```
sudo su
```

Connect an LED using a resistor between GPIO11 and GND.

Creating a File access to GPIO using console commands:

If you write to the *./export* file in the */sys/class/gpio/* subdirectory, the system creates a file with a GPIO structure according to the input. In this case we want to create an access to write directly to GPIO11 in order to handle an LED.

Create a GPIO file access:

```
echo 11 > /sys/class/gpio/export
```

Configure the Pin Direction (In/Out):

```
echo out > /sys/class/gpio/gpio11/direction
```

Write a value to turn on the LED using the GPIO11:

```
echo 1 > /sys/class/gpio/gpio11/value
```

Now your led should be ON!!!

Write a value to clear the LED using the GPIO11

```
echo 0 > /sys/class/gpio/gpio11/value
```

Now your led should be OFF!!!

Delete the created GPIO (11)

```
echo 11 > /sys/class/gpio/unexport
```

You are able to access to GPIO using python or any programming language. We wrote a python script to show how. Download

Open a new terminal and execute the script

```
wget https://sites.google.com/site/semilleroadt/raspberry-
pi-tutorials/gpio/ADT_blink.py
python ADT_blink
```

The script prompts the questions to determine which pin and how many times you want it to blink:
Type the number of the GPIO you want to blink : 11
Type the number of times you want it to blink e.g: 10

Now your GPIO 11 is blinking like a christmas tree.

Open the python script with a text editor and study all the lines (try to understand, is a simple script using file access)

Let's start with the Second method (Pointers):

If you want to read/write to GPIOs using pointers the better way is installing the bcm2635 library and understanding how it works, you can read the datasheet of bcm2835 starting with the section 1.2.2

Installing bcm2835 library

```
wget http://www.open.com.au/mikem/bcm2835/bcm2835-
1.15.tar.gz
tar zxvf bcm2835-1.15.tar.gz
./configure
make
make check
make install
```

Now you can easily use this library to access the GPIOs, the library uses pointers to write/read to the GPIOs or to changes the values in registers of the HW, modifying the function of each peripheral (PWM modules, UART, etc)

Download the code example written in C
Open the Terminal and use gcc to compile the code.

```
wget https://sites.google.com/site/semilleroadt/raspberry-
pi-tutorials/gpio/main.c
gcc -o main -l rt main.c -l bcm2835
```

Execute the compiled code

```
./main
```

Now your LED connected in the GPIO11 is Blinking!!! Like the example with the python script.

Open the C code with a text editor and study all the lines.

Now the Question is: What is the best way to access GPIOs on linux?

The Answer is here:
http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/

PWM

The WiringPi project is a library that includes an application for easy GPIO access. For PWM it allows to configure hardware modules for dedicated PWM pins as well as using a software PWM solution on other pins.

Install WiringPi (WiringPi uses git, a source code management system):

```
sudo apt-get install git-core
```

Download or "clone" the WiringPi project and build it:

```
git clone git://git.drogon.net/wiringPi
./build
```

If you have already downloaded it, you can update to the latest version:

```
cd wiringPi
git pull origin
./build
```

In order to use the WiringPi application you need to know the pin assignments related to it, which are explained in this site: https://projects.drogon.net/raspberry-pi/wiringpi/pins/

Using a HW module for PWM. Connect an LED using a resistor between GPIO18 and GND. (Pin 1 for WiringPi)

Refer to the "man page" of the recently installed WiringPi program called "gpio":

```
man gpio
```

Notice that you can configure a pin to be in, out, pwm, up, down or tri.

According to it, configure GPIO18 (WiringPi Pin 1) in HW PWM Mode using the command shell:

```
gpio mode 1 pwm
```

Write a value to the PWM module (from 1 to 1023):

```
gpio pwm 1 500
```

To remove the configuration of the pin, use:

```
gpio unexport 1
```

To remove all configurations:

```
gpio unexportall
```

Feel free to use the gpio program to configure other pins as input or output (PWM is only for special function pins like GPIO18(WiringPi 1), other PWM pins are occupied by the 3.5mm audio connector.

Using wiringPi.h and softPwm.h with C.
This is a code example using wiringPi to configure a Soft-PWM pin.

UART

In order to use the dedicated UART pins on the raspberry pi, first they have to be removed from their default application which is debugging.

To do this edit "/boot/cmdline.txt" and "/etc/inittab".

You can backup this files if you want to return to the default configuration:

```
cp /boot/cmdline.t      Traductor de Google
cp /etc/inittab /etc/inittab.bak
```

Remove "console=ttyAMA0,115200" and "kgdboc=ttyAMA0,115200" configuration parameters from the "/boot/cmdline.txt" configuration file using nano editor.

```
nano /boot/cmdline.txt
```

Comment the last line on the "/etc/inittab" file. Put a '#' before "T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100.

```
nano /etc/inittab
```

Now the RXD (GPIO15) and TXD (GPIO14) pins are available for general UART use.
This is an example application to use the UART pins using Python and the pyserial library:
To install Pyserial download the latest version from http://sourceforge.net/projects/pyserial/files/pyserial/
Install by running the setup.py

```
python setup.py install
```

Open a Python terminal (remember to always be running under SuperUser by using sudo or su):

```
python
```

To test it without the need of another device simply connect raspberry's TXD and RXD pins to each other:
Run the following commands on the python terminal:

```
import serial
ser = serial.Serial("/dev/ttyAMA0")
ser.write("UART the Font")
read = ser.read()
print read
ser.close()
```

The print command should have printed the string sent using the write command.
To confugre the UART port, read the following introduction: http://pyserial.sourceforge.net/shortintro.html

Install "minicom" for terminal emulation:

```
apt-get install minicom
```

You can use minicom using the next line command

```
minicom -b 9600 -o -D /dev/ttyAMA0
```

SABADO CLASE LCD RASPBERRY

```
wget https://sites.google.com/site/semilleroadt/raspberry-pi-tutorials/gpio/lcd_raspi.py
```

sudo crontab -e

@reboot python /home/pi/LCD_raspi/lcd_raspi.py

ADT_blink.py (1k)	Holguer Becerra, Jan 10, 20	v.1	⬇
ADT_wiringPi.c (1k)	Jose Pablo Pinilla Gómez, J	v.1	⬇
lcd_raspi.py (2k)	Holguer Becerra, Sep 28, 20	v.10	⬇
main.c (0k)	Holguer Becerra, Jan 10, 20	v.1	⬇

Comentarios

No tienes permiso para añadir comentarios.