Embedded Systems: Architecture and Programming

Lab Session 4 & 6 – Raspberry Pi, Web and GPIO

Project Title	Lab Session 4 & 6 – Raspberry Pi, Web and GPIO
Team Number	1
Team Member	BENKHOUDA Kamel BERGEN Frédéric GOUET Nathan

Table of contents

Introduction

Today, the objective is to create and manage a web server where sensors data can be found on a web page.

First, share the Wifi connection of your computer with your Ethernet port, launch the DHCP server on your laptop. Then, connect the Raspberry Pi and open a SSH connection. We start by playing with the GPIO with command line, then, you will have to create a program to do the same thing. In the last part, you will install a web server on the Raspberry and display a page with various information.

The following is based on the great tutorial given here: http://falsinsoft.blogspot.fr/2012/11/access-gpiofrom-linux-user-space.html.
You may consider reading it carefully, even if you have a short version in this lab subject.
You also need to remember:

- We have Raspberry Pi 2 and Raspberry Pi B+ models.
- This page explain how GPIO are working: https://www.raspberrypi.org/documentation/usage/gpio-plusand-raspi2/
- You are not allowed to use library to control your GPIO, we ask you to use the basic functions explained in the following sections. (So, forget WiringPi, RPi.GPIO etc, you will use them when you will really understand how it works.)
- The pin numbering of Raspberry Pi board is a mess. You have different numbers for physical pins, GPIO. Even libraries (WiringPi) are using different numbers... This page gives a clear explanation on each GPIO of the Raspberry : http://pi.gadgetoid.com/pinout
  On this page, you can find the link between the number of the pin and the number of the GPIO (noted "BCM" on the webpage). For example, GPIO/BCM 24 is on the physical pin 18 of the board.

A GPIO (for General Purpose Input/Output) can be used on a Raspberry Pi board, just like on an Arduino board. But, notice one thing: there is no ADC on a Raspberry board so you can "only" use digital input/output. No worries, you can use I²C and other communication protocols to communicate with advanced sensors or... Arduino boards.

Questions

5.1 Using GPIO with a code

1. Create a program named GPIOBlink (C or C++) to blink a LED connected to the GPIO 24 (pin 18).

```
C:\WINDOWS\system32>arp -a

Interface : 192.168.137.1 --- 0x5
  Adresse Internet      Adresse physique      Type
  169.254.25.148        b8-27-eb-2b-9a-3f     dynamique
  192.168.137.30        b8-27-eb-2b-9a-3f     statique
  192.168.137.255       ff-ff-ff-ff-ff-ff     statique
  224.0.0.22            01-00-5e-00-00-16     statique
  224.0.0.251           01-00-5e-00-00-fb     statique
  224.0.0.252           01-00-5e-00-00-fc     statique
  239.255.255.250       01-00-5e-7f-ff-fa     statique
  255.255.255.255       ff-ff-ff-ff-ff-ff     statique

Interface : 172.21.42.20 --- 0xf
  Adresse Internet      Adresse physique      Type
  172.21.0.39           00-1a-8c-f0-c6-e9     dynamique
  172.21.63.255         ff-ff-ff-ff-ff-ff     statique
  224.0.0.2             01-00-5e-00-00-02     statique
  224.0.0.22            01-00-5e-00-00-16     statique
  224.0.0.251           01-00-5e-00-00-fb     statique
  224.0.0.252           01-00-5e-00-00-fc     statique
  239.255.255.250       01-00-5e-7f-ff-fa     statique
  255.255.255.255       ff-ff-ff-ff-ff-ff     statique
```

Raspberry Pi's adress : 192.168.137.30
In the Raspberry Pi, we create the GPIOBlink.c file with the code below :

GPIOBlink.c :

```c
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#define MAX_BUF 10000

int main ()
{
int fd;
char buf[MAX_BUF];
int gpio = 24; //connection to the pin 18

int time = 1;

fd = open("/sys/class/gpio/export", O_WRONLY); //reserve the GPIO
printf("%d",fd);
sprintf(buf, "%d", gpio); //we save the export path of the gpio in a buf
write(fd, buf, strlen(buf));

sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);//Set the direction in the GPIO folder just created

close(fd);

fd = open(buf, O_WRONLY);

// Set out direction
write(fd, "out", 3);

close(fd);
//We are in a case of out direction, we mus set the value of GPIO
sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);

fd = open(buf, O_WRONLY);

int count = 0;

for (count; count < 20 ; count++)
{
    //Set GPIO high status
    write(fd, "1", 1);
    sleep(time);
    //Set GPIO low status
    write(fd, "0", 1);
      sleep(time);
}

close(fd);
//Once finished, we free (unexport) the GPIO
fd = open("/sys/class/gpio/unexport", O_WRONLY);

sprintf(buf, "%d", gpio);

write(fd, buf, strlen(buf));

close(fd);

}
```

2. Now, we must modify this code so we can give the number of the GPIO using a paramater (for example : ./GPIOBlink 23 will blink a LED connected to the GPIO 23).

GPIOBlink.c :

```c
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#define MAX_BUF 10000


int main (int argc, char* argv[])
{
int fd;
char buf[MAX_BUF];
int gpio = atoi(argv[1]); //the parameter will be used as the int gpio

int time = 1;


fd = open("/sys/class/gpio/export", O_WRONLY); //reserve the GPIO
printf("%d",fd);
sprintf(buf, "%d", gpio); //we save the export path of the gpio in a buf
write(fd, buf, strlen(buf));

sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);//Set the direction in the GPIO folder just created

close(fd);

fd = open(buf, O_WRONLY);

// Set out direction
write(fd, "out", 3);


close(fd);
//We are in a case of out direction, we mus set the value of GPIO
sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);

fd = open(buf, O_WRONLY);

int count = 0;


for (count; count < 10 ; count++)
{
    //Set GPIO high status
    write(fd, "1", 1);
    sleep(time);
    //Set GPIO low status
    write(fd, "0", 1);
    sleep(time);
}

close(fd);
//Once finished, we free (unexport) the GPIO
fd = open("/sys/class/gpio/unexport", O_WRONLY);

sprintf(buf, "%d", gpio);

write(fd, buf, strlen(buf));

close(fd);
}
```

3. Now, we will create a new program named GPIOLed that switches on or off a LED on a specific GPIO, taking two parameters:
   - the GPIO number
   - the value :
     0 to export the GPIO, set the direction to "OUTPUT" and switch off the LED
     1 to export the GPIO, set the direction to "OUTPUT" and switch on the LED
     3 to unexport the GPIO

GPIOLed.c :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <time.h>
#include <string.h>

#define MAX_BUF 10000

int main (int argc, char* argv[])
{
    int fd;
    char buf[MAX_BUF];
    int gpio = atoi(argv[1]); //the first parameter will be used as the int gpio
    int value = atoi(argv[2]); //the second parameter will be used as the int value

    if (value == 1 || value == 0)//if we want to export the GPIO and switch on or off the LED
    {
        fd = open("/sys/class/gpio/export", O_WRONLY);//reserve the GPIO

        sprintf(buf, "%d", gpio);// we save the gpio number in the buffer
        write(fd, buf, strlen(buf));
        close(fd);

        sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);//Set the direction in the GPIO folder just created at the buffer

        fd = open(buf, O_WRONLY);

        // Set out direction
        write(fd, "out", 3);
        close(fd);

        sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);//Set the direction in the GPIO folder just created

        fd = open(buf, O_WRONLY);

        sprintf(buf, "%d", value);
        write(fd, buf, 1);

        close(fd);
    }
    else if (value == 3) //if we want to unexport the GPIO
    {

        fd = open("/sys/class/gpio/unexport", O_WRONLY);// reserve unexport gpio file


        sprintf(buf, "%d", gpio);// we save the gpio number in the buffer

        write(fd, buf, strlen(buf));
        close(fd);
        printf("GPIO %d is unexported\n", gpio);
    }
    else
    {
        printf("Enter value only as 0 = switch off the LED, 1 = switch on the LED, 3 = free the GPIO");
    }
}
```

4. Finally, we will create a program GPIOButton that displays the value on a specific GPIO, by taking 2 parameters :
   - the GPIO number
   - the value : 0 to export the GPIO, set the direction to "INPUT" and display on the screen the value on the GPIO
     3 to unexport the GPIO

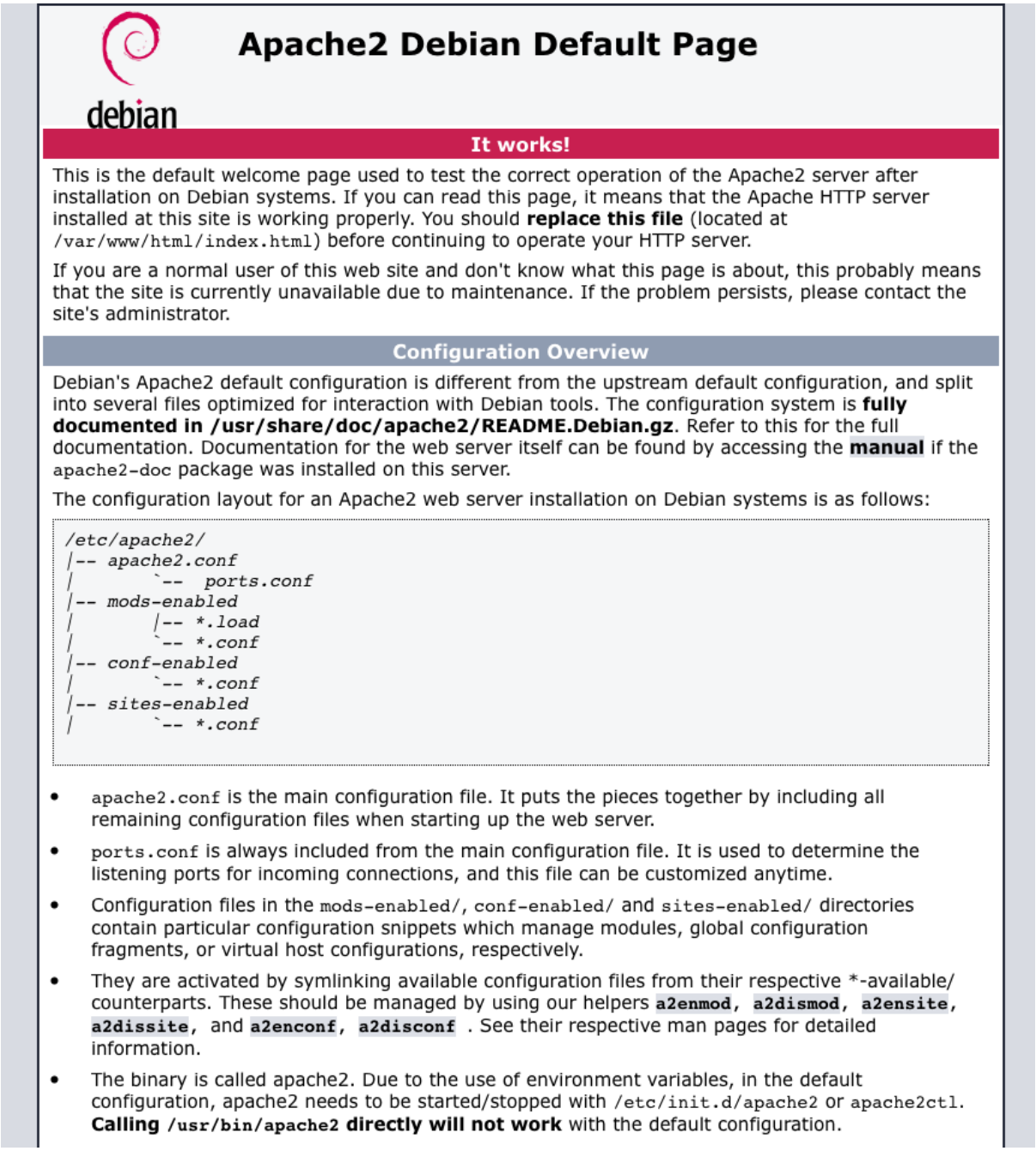
GPIOButton.c :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <time.h>
#include <string.h>

#define MAX_BUF 10000

int main (int argc, char* argv[])
{
    int fd;
    char buf[MAX_BUF];
    int gpio = atoi(argv[1]);
    int value = atoi(argv[2]);

    if (value == 0) //if we want to to export the GPIO, set the direction to "INPUT" and display on the screen the value on the GPIO
    {
        fd = open("/sys/class/gpio/export", O_WRONLY);//reserve the GPIO

        sprintf(buf, "%d", gpio);// we save the gpio number in the buffer

        write(fd, buf, strlen(buf));
        close(fd);

        sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);//Set the direction in the GPIO folder just created at the buffer

        // open the file written at the buffer
        fd = open(buf, O_WRONLY);

        //Set in direction
        write(fd, "in", 2);
        close(fd);
        printf("GPIO value: %d", gpio); // display on the screen the value on the GPIO
    }
    else if (value == 3) //if we want to unexport the GPIO
    {
        fd = open("/sys/class/gpio/unexport", O_WRONLY);// reserve unexport gpio file

        sprintf(buf, "%d", value);// we save the gpio number in the buffer

        write(fd, buf, strlen(buf));
        close(fd);
        printf("GPIO %d is unexported\n", gpio);
    }
    else
    {
        printf("Enter value only as 0 = export the GPIO, set the direction to INPUT and display on the screen the GPIO value\n");
        printf("3 = free the GPIO\n");
    }
}
```

5.2 Web Server

1. We have downloaded apache2 and php as it's requested in the question.

2. After that, we must open a web browser on our laptop and type the IP address of our Raspberry board. We observe this page appearing :



Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers **`a2enmod`**, **`a2dismod`**, **`a2ensite`**, **`a2dissite`**, and **`a2enconf`**, **`a2disconf`** . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

5.3 Control GPIO from a web page

1. We will create a PHP web page which can receive two parameters, gpio and value for example and the page will display those :

   page.php :

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Réception de paramètres dans l'URL</title>
    </head>
    <body>

    <p> GPIO Value : <?php echo 'GPIO Value : ' . htmlspecialchars($_GET["
        gpio"]) . '!'; ?> !</p>
    <p> Value : <?php echo 'Value (0 or 1) : ' . htmlspecialchars($_GET["
        value"]) . '!'; ?> !</p>


    </body>
</html>
```

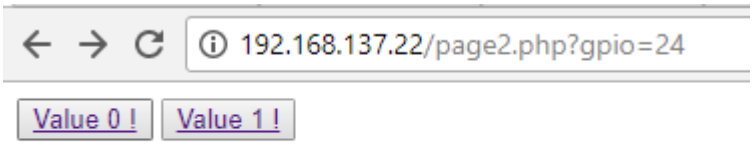
GPIO Value : GPIO Value : 24! !

Value : Value (0 or 1) : ! !

2. We will create a web page with two buttons :
   a) One button will call the previous PHP web page with parameters gpio=24 and value=0
   b) The other will call the previous PHP web page with parameters gpio=24 and value=1

   index.php :

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Envoi de paramètres dans l'URL</title>
    </head>
    <body>

    <button><a href="page.php?gpio=24&amp;value=0">Value 0 !</a></button>

    <button><a href="page.php?gpio=24&amp;value=1">Value 1 !</a></button>


    </body>
</html>
```



We push the button "Value 0 ! " :