

SISTEMAS COMPUTACIONAIS AVANÇADOS (SISTCA)

ADVANCED COMPUTING SYSTEMS

LICENCIATURA EM ENGENHARIA ELETROTÉCNICA E DE
COMPUTADORES

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
POLITÉCNICO DO PORTO

Lab Classes Script:

The world of the Raspberry pi

Version Control

Version number	Date issued	Authors	Update information
V1.0	March 2022	Clara Gonçalves (1190481) Tiago Nazário (1191092) Rui Teixeira (1191039) Diogo Moreira (1190528) Supervision: Prof. Paula Viana	Original version of the script.

Table of Content

1.	Introduction.....	1
1.1	Scientific/Technological Context	1
1.2	Motivation for this topic/lab script.....	1
1.3	Objectives (of this lab script)	1
1.4	Structure (of this lab script).....	1
	The rest of the lab script is organized as follows:	1
2.	Theoretical (scientific/technological background)	2
2.1	Main theoretical concepts/terminology	Erro! Marcador não definido.
2.2	State-of-The-Art	Erro! Marcador não definido.
3.	Outlook of the technology.....	5
3.1	Equipment/Material	5
3.2	Setup/Installation (SW/HW)	6
3.3	First steps	11
4.	Tutorial/Functionality	12
4.1	Hardware and GPIOs	12
4.1.1	WiringPi – GPIO Interface library for Raspberry Pi	14
4.1.2	Turning a LED ON and OFF	18
4.1.3	Controlling the brightness of a LED with PWM	19
4.1.4	Receive a logic input using a button.....	20
4.2	Server using SAMBA	23
5.	Exercise(s)	26
5.1	Problem A	26
5.1.1	Material	26
5.1.2	Circuit and problem	26
5.1.3	(re)solution A	27
5.2	Problem B	27
5.2.1	(re)solution B	28

6.	Challenge	Erro! Marcador não definido.
7.	References	29
	Annexes	Erro! Marcador não definido.

1. Introduction

1.1 Scientific/Technological Context

This Lab Script aims to show the student a very wide used microcomputer as well as what it can do, in this report the student will be able to work with the technology, learn of its hardware and one of the many possibilities that it has like creating a file server that can be access from other computers by a folder. The student will be using some electronic components to see how they can interact with them with the raspberry.

1.2 Motivation for this topic/lab script

In LEEC we don't talk about microcomputers, and has it been used in a lot of industries, we thought it would be important to talk about them as they are used in a big variety of things. They are cheaper than some dedicated machines and consume less energy. We also show some programs that can be useful and give some tips on how to use them like how to make a bootable drive or connect with SSH with VS code.

1.3 Objectives (of this lab script)

We aim to showcase Raspberry to the student and show how it can interact with electronics, we also want to show a possible project that anyone can implement at their home, this is one of the many lists of possibilities that anyone can do with this technology, the only limit being creativity.

1.4 Structure (of this lab script)

The rest of the lab script is organized as follows:

- A theoretical background on the history of the microcomputers and the raspberry pi
- A comparison between raspberry and Arduino
- State-of-The-Art
- An outlook on how to set up the raspberry pi and the first steps in how to use it
- Overlook on GPIO with circuit examples and implement server using samba
- Two exercises using the raspberry pi potential on the first focusing on hardware and the second one focusing on software

Theoretical (scientific/technological background)

Nowadays Raspberry Pi is an affordable and versatile technology that allows beginners and advanced users explore the world of microcomputers. It's used in schools and college to do some projects, but it's used also in companies and businesses as servers or equipment monitoring.

2.

2.1 Microcomputer

What is a microcomputer?

The concept of a microcomputer changed with time, in the beginning it started with the ENIAC in 1946, the first ever programming computer in history.

With time, computers which size allowed them to be used in workplaces, being nicknamed microcomputers due to their reduced size compared to the ENIAC.

After around the 70s, the definition changed, microcomputers are computers that use only one processor to process data, the first one was the MITS ALTAIR 8800 in 1974, being also created by IBM PC, the Apple Macintosh, between others and the most recent Raspberry Pi.

ENIAC

The ENIAC (Electrical Numerical Integrator and Computer) with its 30 tons, 5.50 meters height and 25 meters of width was the first ever programmable computer. Financed by the North American army it was built in Pennsylvania University, it was projected by John Mauchly and J. Presper Eckert. It was presented to the world as the first ever machine that would help the army in their ballistic area. [1]

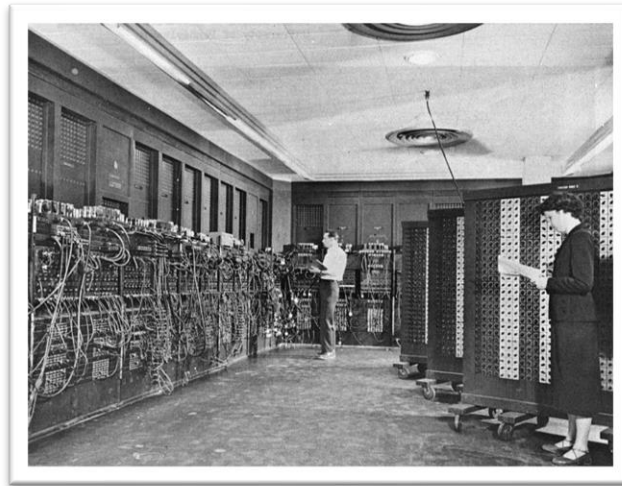


Figure 1 - ENIAC

2.2 Raspberry pi

- **What is a raspberry Pi?**

A raspberry pi is credit card sized computer where you can add peripherals just like everyday computers.

- **Models**

- Pi 1 Model B (2012)
- Pi 1 Model A (2013)
- Pi 1 Model B+ (2014)
- Pi 1 Model A+ (2014)
- Pi 2 Model B (2015)
- Pi Zero (2015)
- Pi 3 Model B (2016)
- Pi Zero W (2017)
- Pi 3 Model B+ (2018)
- Pi 3 Model A+ (2019)
- Pi 4 Model B (2020)
- Pi 400 (2021)

- **Specifications of Raspberry Pi 2 Model B**

Raspberry Pi 2 Model B has a 900MHz quad-core ARM Cortex-A7 CPU, 1 GB RAM, 4 USB and 1 Ethernet ports, 1 full-size HDMI output, 40 GPIO pins, micro SD card slot, camera interface (CSI), display interface (DSI), VideoCore IV 3D graphics core and combined 3.5 mm audio jack and composite video.

- **Peripherals of Raspberry Pi 2 Model B**

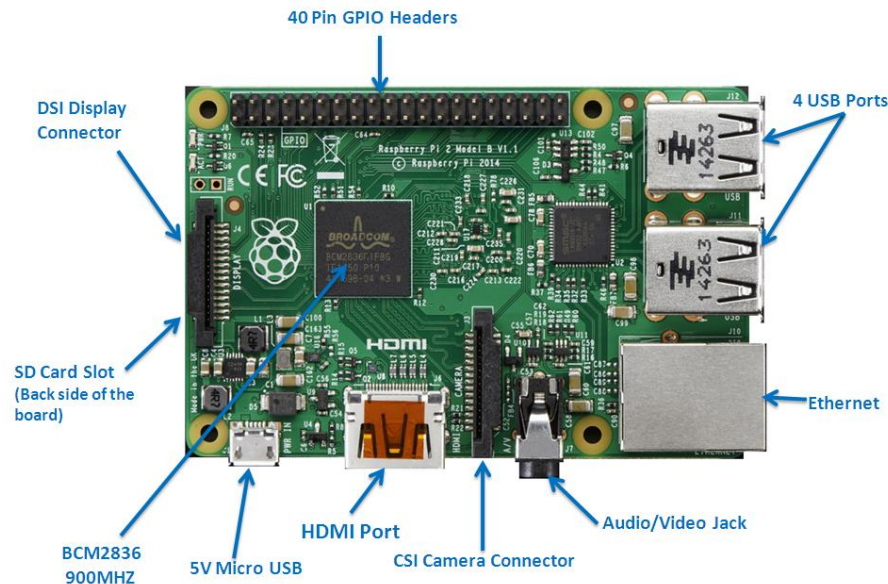


Figure 2 - Raspberry Pi 2 Model B

- **Areas of Interest**

Nowadays Raspberry Pi is an affordable and versatile technology that allows beginners and advanced users explore the world of microcomputers. It's used in schools and college to do some projects, but it's used also in companies and businesses as servers or equipment monitoring. [2]

2.3 Arduino vs Raspberry Pi

Tabela 1 - Arduino and Raspberry Pi [3] [4]

Arduino	Raspberry Pi
Microcontroller	Microcomputer
You can programm in C and C++ using Arduino IDE	You can programm in C,C++,Python, Java,...

Used for running simple tasks repeatedly	Can run multiple tasks
CPU architecture: 8 bit	CPU architecture: 64 bit
The price varies between \$22 and \$46	The price varies between \$7 and \$75
Processing Speed: 16 MHz	Processing Speed: 1.2 GHz
Open-source hardware and software	Closed-source hardware and software
There is no wireless connectivity on board	There is wireless connectivity on board
Arduino doesn't need an Operating System to run	Raspberry Pi needs an Operating System to run
Better for interfacing sensors and controlling LEDs and Motors	Better for developing software applications

3. Outlook of the technology

3.1 Equipment/Material

The raspberry pi is very easy to use and setup. It will take about 10 min for you to have it on full capacity. For that you will need some accessories:

- USB-B Micro power supply (5V power Supply and 3 amps(3A))
- MicroSD card
- Keyboard and mouse
- HDMI cable
- 2x Network cable (one for your computer and the other for the raspberry)
- Micro SD card reader
- Computer Monitor

NOTE: Raspberry Pi can be used without a case, but you need to be careful because a contact with a metal of some sort can cause a short circuit and damage permanently the Raspberry Pi.

3.2 Setup/Installation (SW/HW)

Step 1: Install balenaEtcher

This program flashes the MicroSD with an operative system for the raspberry pi. For both Linux and Windows, the process is similar.

- Install the program from the official website <https://www.balena.io/etcher/>, then select the option that corresponds to your operative system.
- After installation you need to have the application like figure 2. [5]

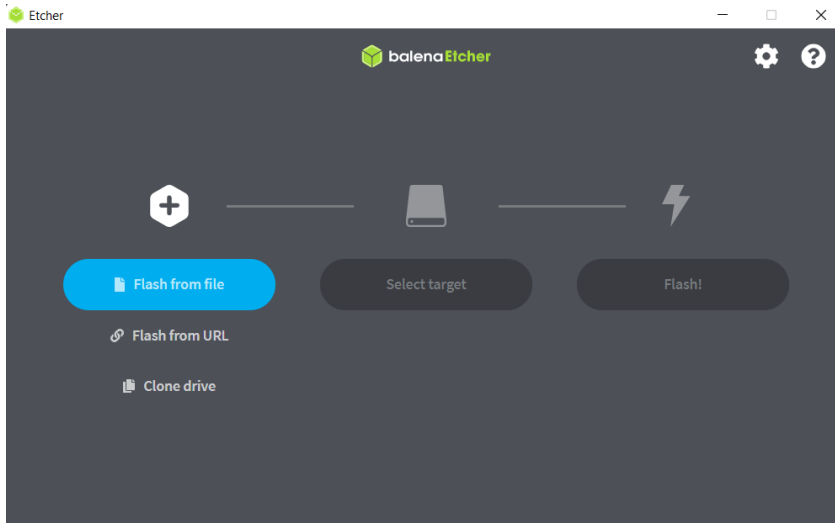


Figure 3 - balenaEtcher Program

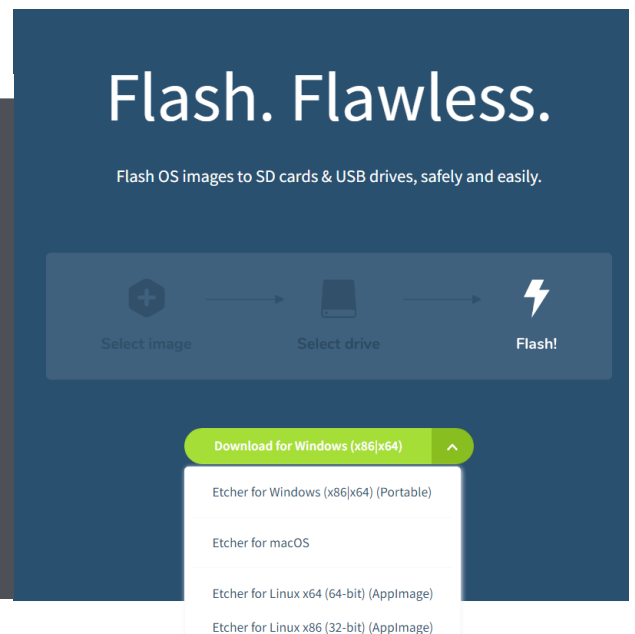


Figure 4-balenaEtcher website

Step 2: Download Raspberry pi OS

- The Raspberry Pi needs an operating system in order to work (Raspberry Pi OS). Like Step 1 you can just download it from the official website <https://www.raspberrypi.com/software/> and select the option that corresponds to your operative system.

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. [Watch our 45-second video](#) to learn how to install an operating system using Raspberry Pi Imager.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Windows](#)

[Download for macOS](#)

[Download for Ubuntu for x86](#)

To install on **Raspberry Pi OS**, type
`sudo apt install rpi-imager`
in a Terminal window



Figure 5 - raspberry pi software Website

NOTE: this is the full download of the operating system with the graphic environment and all the features. On the website you can manually install an operating system image with different aspects, this is relevant because and if you need to use the rasp to process a long code, by installing the full version you will be taking space from the microSD with programs and aspects that you will not use just occupying space needed for your code.

Step 3: Flash the operative system

- After you have the software and the program to flash into the micro-SD, for that you need to place the micro-SD in a microSD card USB adapter.
- Launch BalenaEtcher. Click on 'Flash from file' then select the file that your previous download in Step2, then click on 'Select Target' and select the device that you are using with the micro-SD.
- Finally, click the on the 'Flash!' button and wait this will take some time. When is completed, you can remove the microSD card and insert it into your Raspberry Pi.

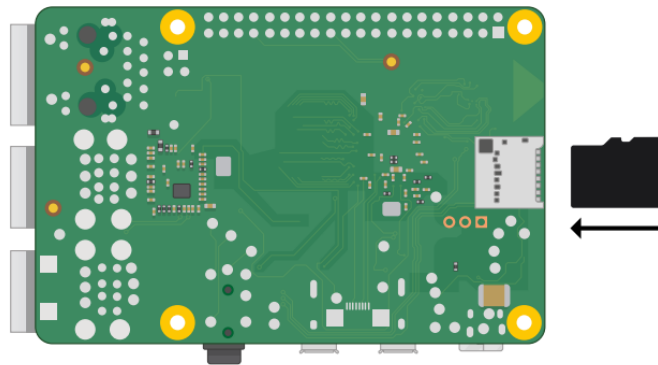


Figure 6 - insert microSD

To connect the micro-SD just gently pull the micro-SD.

Step 4: Connections

This is the final part of the setup. You will need to do the following connections:

- Keyboard to USB
- Mouse to USB
- Computer Monitor to HDMI
- USB-B Micro power supply to power supply
- And network cable(optional)

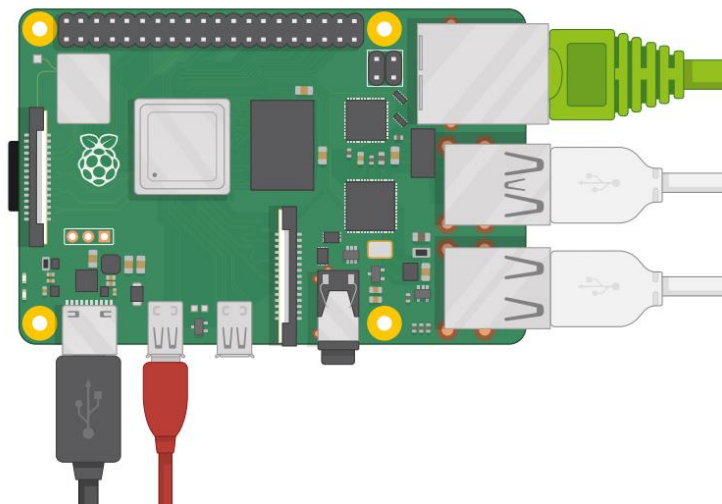


Figure 7 - Connections

Your raspberry pie connections should look something similar to the picture above.

When you connect you raspberry pi for the first time to a screen you may have to wait a couple of minutes for the system to start. Then you will see in the screen something similar to Figure 6.

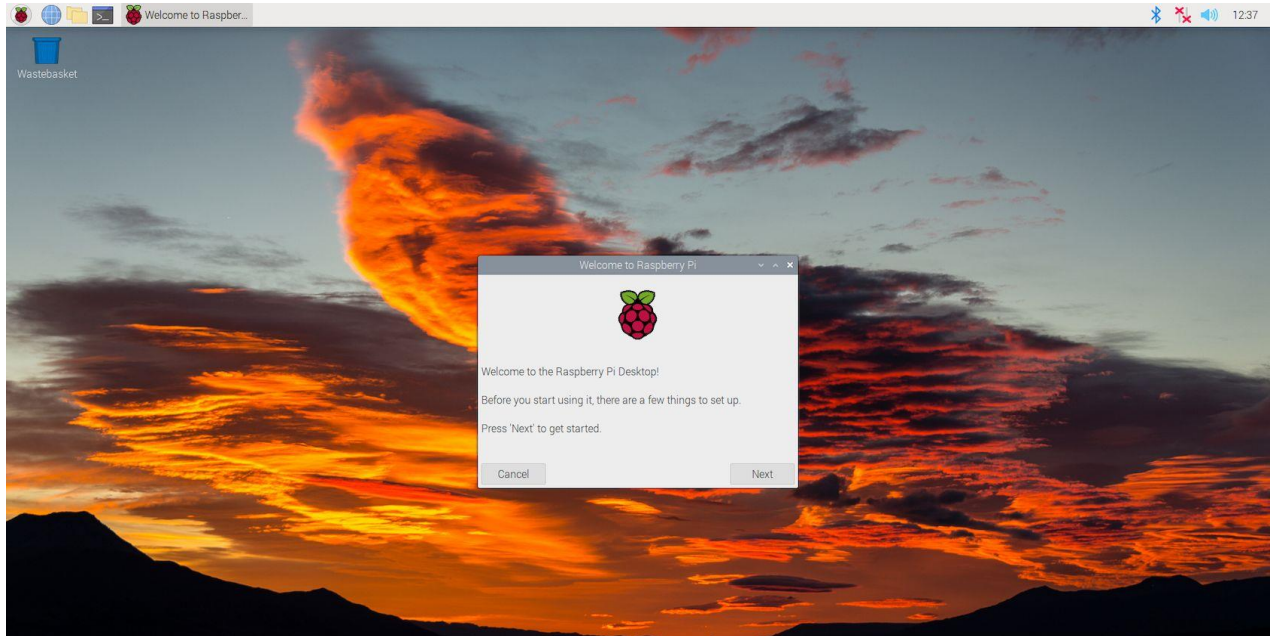


Figure 8 - Raspberry pi GUI

Select next, then select your country, time zone, and change the password (the default password is *Raspberry Pi*). When the setup is ready just restart the Raspberry.

Congrats 🎉 !! Your Raspberry Pi is now fully operative. [6]

Step 5: Get you Raspberry Pi IP address

To get your Raspberry Pi IP is an easy process just open the terminal and use the following command:

```
$ ifconfig
```

The IP address will be in the second line.

Step 6: Connect your pc to the raspberry pi

To be able to program from your computer directly to the raspberry pi or manipulate folders and files you can use Visual Studio Code. To do that go to the extensions page in vs code and install the extension "Remote - SSH" this will add an icon at the bottom left of the page.

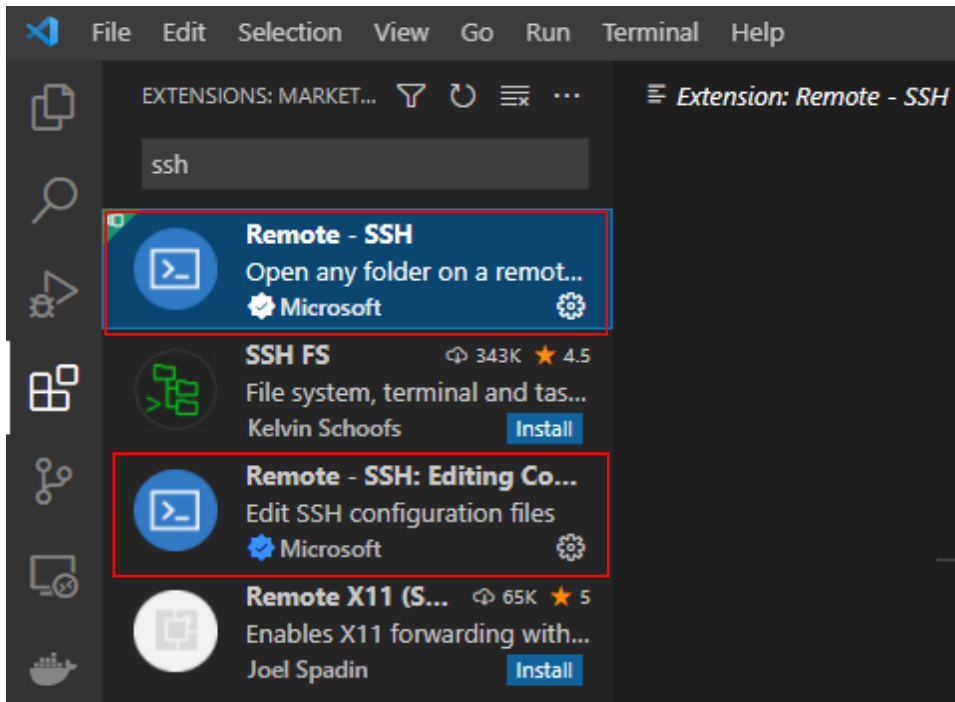


Figure 9 - SSH Extensions

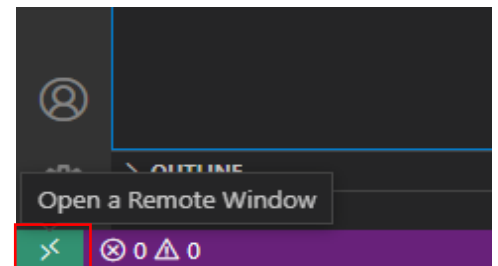


Figure 10 - Icon to connect with SSH

Then you can select "connect to host", click on "Add New SSH Host" and then type "ssh root@RaspberryIP"

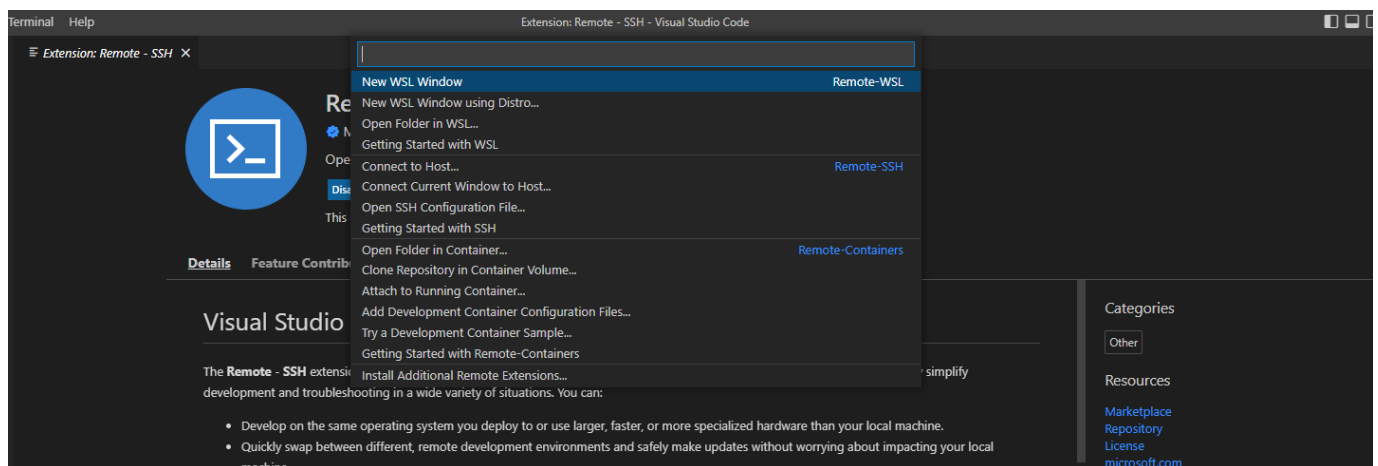


Figure 11 - Connect to host with SSH

The bottom icon will change, and it will display the Raspberry IP.

After that you can go and open a folder and a terminal, and you will be ready to work with SSH!

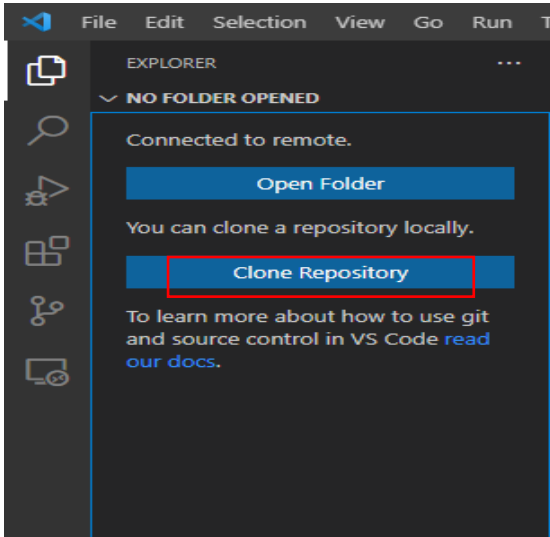


Figure 12 - Open a folder in the Raspberry

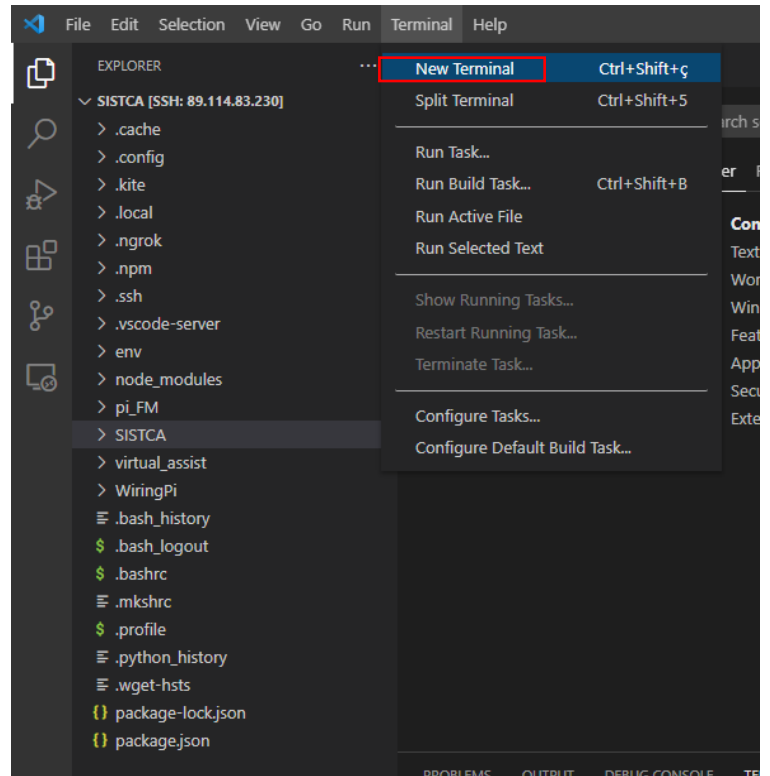


Figure 13 - Open a Terminal

3.3 First steps

Here you are going to run a simple code in C on the Raspberry.

1. Create and open a file on raspberry pi:

```
sistca@raspberrypi:~ $ sudo nano hello.c
```

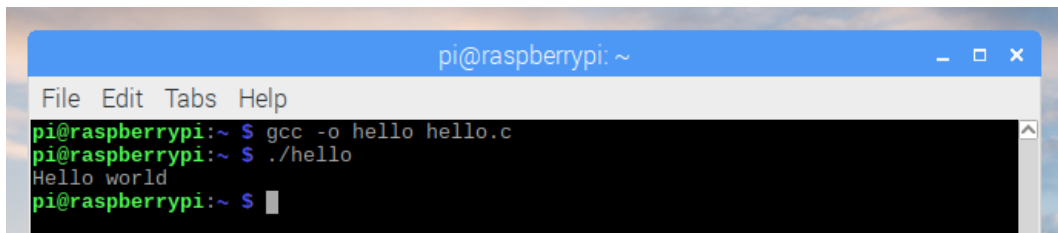
2. Write the following code:

```
1.include <stdio.h>
2.
3.int main(){
4.
5.printf("Hello world!!\n");
6.
7.return 0;
8.}
```

3. Compile and run the code:

```
sistca@raspberrypi:~ $ gcc -o hello hello.c
sistca@raspberrypi:~ $ ./hello
```

It should look something like Figure 18:

A terminal window titled 'pi@raspberrypi: ~' with a menu bar containing 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the following commands and output:

```
pi@raspberrypi:~ $ gcc -o hello hello.c
pi@raspberrypi:~ $ ./hello
Hello world
pi@raspberrypi:~ $
```

Figure 14 - Hello World!

4. Tutorial/Functionality

4.1 Hardware and GPIOs

Similarly to Arduino, the Raspberry Pi is equipped with a physical interface that allows for the prototyping of electrical circuits. These pins are referred to as GPIO, which stands for *General Purpose Input/Output* and there is a total of 40 pins in the model used in this tutorial.

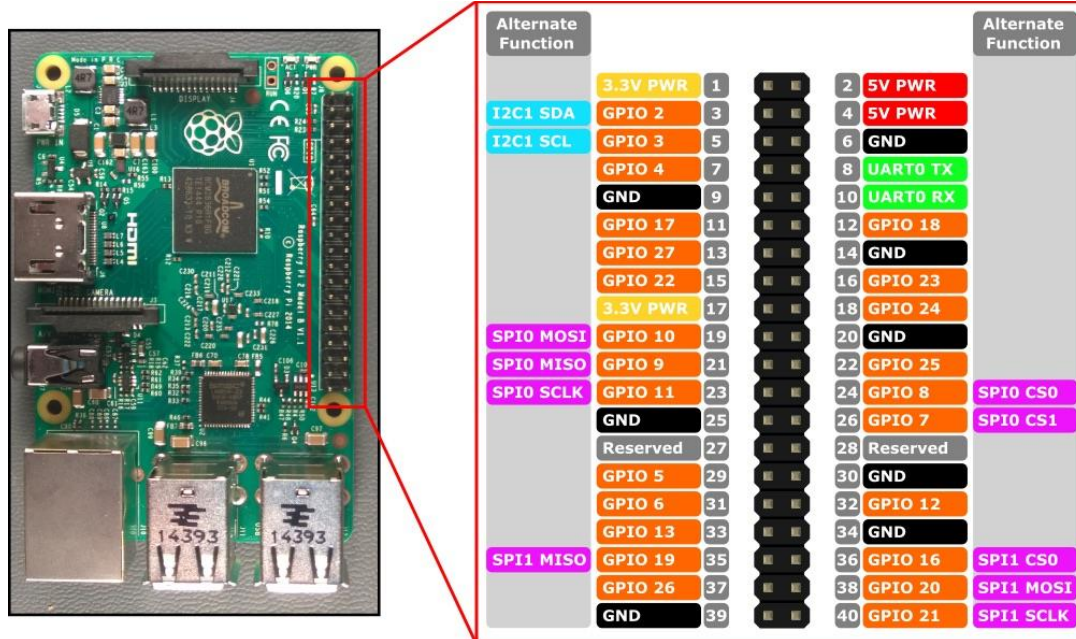


Figure 15 - Raspberry Pi 2 Model B GPIOs

The GPIO pins allow the Raspberry Pi to control and monitor the outside world by being connected to electronic circuits. The Pi can control LEDs, turning them on or off, run motors, and many other things. It is also able to detect whether a switch has been pressed, the temperature, and light. We refer to this as physical computing. [7]

In the following steps, we will explain how to control the GPIOs using commands on Raspberry's terminal as well as how to create a simple program in C to control a circuit.

Usually, Raspbian comes by default with a handy command line utility called *gpio*. This command can export pins, set directions, set and read logic states and some other advanced features like PWM. If *gpio* does not work, install it using:

```
sistca@raspberrypi:~ $ sudo apt-get install wiringpi
```

Note: The official distribution of WiringPi has been deprecated by the author, therefore, there is no official way to install WiringPi on recent versions of Raspbian. If you are using a any Raspberry above Raspberry Pi 2B+, or any version of Raspbian above Bullseye, look for an alternative to WiringPi or download an unofficial version of WiringPi. An unofficial version at the current writing of this guide can be found here: <https://github.com/WiringPi/> [8] [9]

Installation:

```
sistca@raspberrypi:~ $sudo git clone
https://github.com/WiringPi/WiringPi.git
sistca@raspberrypi:~ $ cd WiringPi
sistca@raspberrypi:~ $ ./build
```

4.1.1 WiringPi – GPIO Interface library for Raspberry Pi

WiringPi is a PIN based GPIO access library written in C and designed for use with C and RTB (BASIC) on Raspberry Pi. WiringPi includes a command-line utility – *gpio* – which can be used to program and setup the GPIO pins directly from terminal, allowing for the read and write of the pins or for using in shell scrips.

WiringPi supports analog reading and writing, and while there is no native analog hardware on a Pi by default, modules are provided to support the Gertboards analog chips and other A/D and D/A devices can be implemented relatively easily. [10]

The GPIO Utility

WiringPi comes with a separate program to help manage the on-board GPIO interface as well as additional modules, as mentioned previously. This program, called *gpio*, although not the most efficient, can control the GPIO pins directly from the Terminal or from a shell script. It's also possible to call it using *system()* function in C/C++.

See the man page for the GPIO program to see what all the features are by typing:

```
sistca@raspberrypi:~ $ man gpio
```

A few important commands are:

- **gpio [-g | -1] ...**

The optional flag **-g** causes the pin numbers to be interpreted as BCM pin numbers, while **-1** causes the pins to be interpreted as hardware pin numbers. If no flag is used, *gpio* will interpret the pin numbers using WiringPi's pin numbers by default.

- **gpio mode <pin> [in | out | pwm | clock | up | down | tri]**

This sets the mode of a pin as Input, Output, PWM or Clock mode. Additionally, it can set the internal pull-up/down resistors to pull-up, pull-down, or none.

- **gpio pwm <pin> <value>**

Sets the pin to a PWM value (0 – 1023).

- **gpio read <pin>**

Reads the logic value of a pin and print it (0 – LOW and 1 – HIGH).

- **gpio readall**

This reads all the normally accessible pins and prints a table of their numbers (WiringPi, BCM_GPIO and physical board pin numbers), along with their current modes and values. This command will detect the version/model of the used Raspberry and print the appropriate diagram. [11]

Setup and Core Functions

When writing a program, one of the four available setup functions must be called at the start, otherwise the program will fail to work correctly. If one of the setup functions fails, it would be considered a fatal program fault and the program execution will be terminated with an error message printed on the Terminal.

There are four ways to initialize WiringPi:

- **int wiringPiSetup(void);**
- **int wiringPiSetupGpio(void);**
- **int wiringPiSetupPhys(void);**
- **int wiringPiSetupSys(void);**

The main difference in each of the former functions is the pin numbering scheme that will be adopted by the program. The first one will use WiringPi's numbering scheme, while the second one will use the Broadcom GPIO pin numbers directly with no re-mapping. Both functions will need root privileges. The third setup function, identically to the previous two, uses the physical board pin numbers, however it only works on the P1 connector.

The fourth function initializes WiringPi but uses the `/sys/class/gpio` interface rather than accessing the hardware directly. This can be called as a non-root user provided the GPIO pins have been exported before-hand using `gpio` program. The pin numbering in this mode is the same as the second function, that is, the native Broadcom GPIO numbers.

NOTE: When using **int wiringPiSetupSys(void)** you can only use the pins which have been exported via the `/sys/class/gpio` interface before you run your program. You can do this in a separate shell-script, or by using the `system()` function from inside your program to call the `gpio` program.

Also note that some functions have no effect when using this mode as they're not currently possible to action unless called with root privileges. [12]

The following functions work directly on the RaspberryPi and also with external GPIO modules, such as GPIO expanders, although not all modules support all functions. Raspberry Pi has no on-board analog hardware, so none of the analog functions will work directly on Pi.

- **void pinMode (int pin, int mode);**

This sets the mode of a pin to either INPUT, OUTPUT, PWM_OUTPUT or GPIO_CLOCK. Note that only a set of pins support PWM and Clock modes (This function has no effect when in Sys mode, to change pin modes you must use the *gpio* program).

- **void pullUpDnControl (int pin, int pud);**

This sets the pull-up/down resistor on the specified pin, which should be set as INPUT. Unlike Arduino, Raspberry has both pull-up and down resistors. The parameter *pud* should be: **PUD_OFF** (no pull-up/down), **PUD_DOWN** (pull to Ground) or **PUD_UP** (pull to 3,3V). The internal resistors have a value of approximately 50kΩ on the Raspberry Pi.

- **void digitalWrite (int pin, int value);**

Writes the value **HIGH** or **LOW** (1 or 0) to the given pin which must have been previously set as an output.

WiringPi treats any non-zero value as HIGH, however 0 is the only representation of LOW.

- **void pwmWrite (int pin, int value);**

Writes the value to the PWM register for the given pin on the range of 0 to 1023.

- **int digitalRead (int pin);**

Returns the logic value read on a given pin.

- **analogRead (int pin);**

This returns the value supplied on an analog input pin. This alone does not work directly on Raspberry Pi.

- **analogWrite (int pin, int value);**

This writes the given value to the supplied analog pin. This alone does not work directly on Raspberry Pi. [13]

Timing functions

While Linux provides a multitude of system calls and functions that allow for various timing and sleeping functions, WiringPi comes with a set of timing functions made to resemble those available on the Arduino platform, making for easier portability between these two platforms.

- **void delay (unsigned int howLong)**

This causes program execution to pause for **howLong** milliseconds. Note that the maximum delay is an unsigned 32-bit integer milliseconds, or approximately 49 days.

- **void delayMicroseconds (unsigned int howLong)**

This causes program execution to pause for **howLong** microseconds. Note that the maximum delay is an unsigned 32-bit integer microseconds, or approximately 71 minutes.

NOTE: Even if you are not using any of the input/output functions, WiringPi still needs to be initialized by calling on of the wiringPiSetup functions. Just use **wiringPiSetupSys()** if you don't need root access to your program and remember to include `<wiringPi.h>`. [14]

Interrupt Routines

With a newer kernel patched with the GPIO interrupt handling code, you can now wait for an interrupt in your program. This frees up the processor to do other tasks while you're waiting for that interrupt. The GPIO can be set to interrupt on a rising, falling or both edges of the incoming signal.

- **int wiringPiISR (int pin, int edgeType, void (*function)(void));**

This function registers a function to received interrupts on the specified pin. The **edgeType** parameter is either **INT_EDGE_FALLING**, **INT_EDGE_RISING** or **INT_EDGE_BOTH**. This function will work in any mode and does not need root privileges to work. [15]

4.1.2 Turning a LED ON and OFF

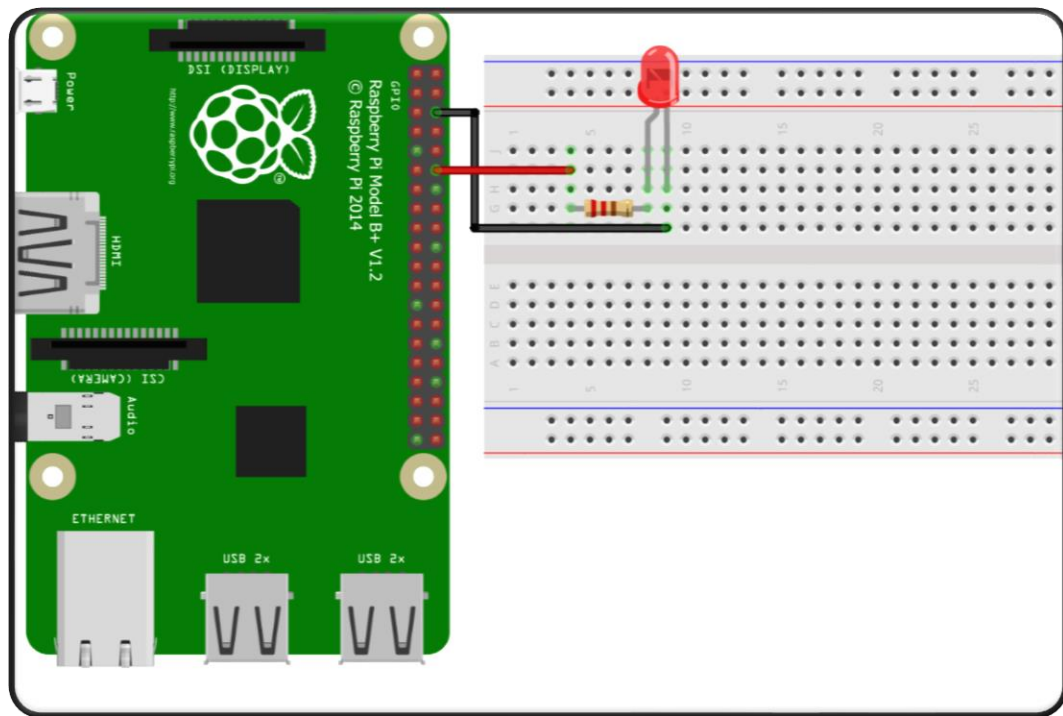


Figure 16 – LED Circuit

Start by assembling the circuit shown above. Connect the cathode of the LED through a 220Ω resistor to GPIO18 (Board Pin 12). Next, connect the anode of the LED to any Ground pin of the Raspberry, like Pin 6.

On Raspberry's terminal, write the following command:

```
sistca@raspberrypi:~ $ gpio -g mode 18 out
```

This will set GPIO 18 as an output. Now, we need to switch the pin between HIGH and LOW modes. To do so, we use the following commands:

```
//Set GPIO2 to HIGH  
sistca@raspberrypi:~ $ gpio -g write 18 1  
  
//Set GPIO2 to LOW  
sistca@raspberrypi:~ $ gpio -g write 18 0
```

We use the -g option to tell the command to use the BCM pins instead of the WiringPi pins.

4.1.3 Controlling the brightness of a LED with PWM

Using the same circuit configuration as in the previous example, execute now the following commands:

```
//Set GPIO17 to PWM mode
sistca@raspberrypi:~ $ gpio -g mode 18 pwm

//Regulate de Duty Cycle of the PWM
sistca@raspberrypi:~ $ gpio -g pwm 18 [0 - 1023]
```

Choose a value in the interval of 0 and 1023 to vary the duty cycle of the PWM and regulate the LED's brightness (with 0 = 0V and 1023 = 3,3V). The Raspberry Pi only has 4 PWM pins, but each pair is sharing one PWM resource, which means there are only 2 unique/controlled PWM pins on Raspberry.

You can check all pin modes in your Raspberry using:

```
sistca@raspberrypi:~ $ gpio readall
```

Something like this should be displayed in your terminal: [7]

```
sistca@raspberrypi:~ $ gpio readall
```

Pi 2											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	IN	1	3	4		5v			
3	9	SCL.1	IN	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	1	ALT0	TxD	15	14
		0v			9	10	1	ALT0	RxD	16	15
17	0	GPIO. 0	OUT	0	11	12	0	ALT5	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21

Figure 17 - GPIO readall output

If you pay attention to BCM Pin number 18, you can see that the mode is called ALT5. This means that the pin is in Alternative Mode 5, in this case, PWM Mode. To learn more about what each pin does and their alternative modes, you can check this link: <https://pinout.xyz/>. [16]

4.1.4 Receive a logic input using a button

We will now use a button to read a logic value and determine if the button is or is not pressed. With that info, we will tell the Raspberry to turn on or off the LED, respectively.

To the previous circuit, add a push button. Connect one terminal to GPIO17 (Board Pin 11) and the other terminal to ground, as follows:

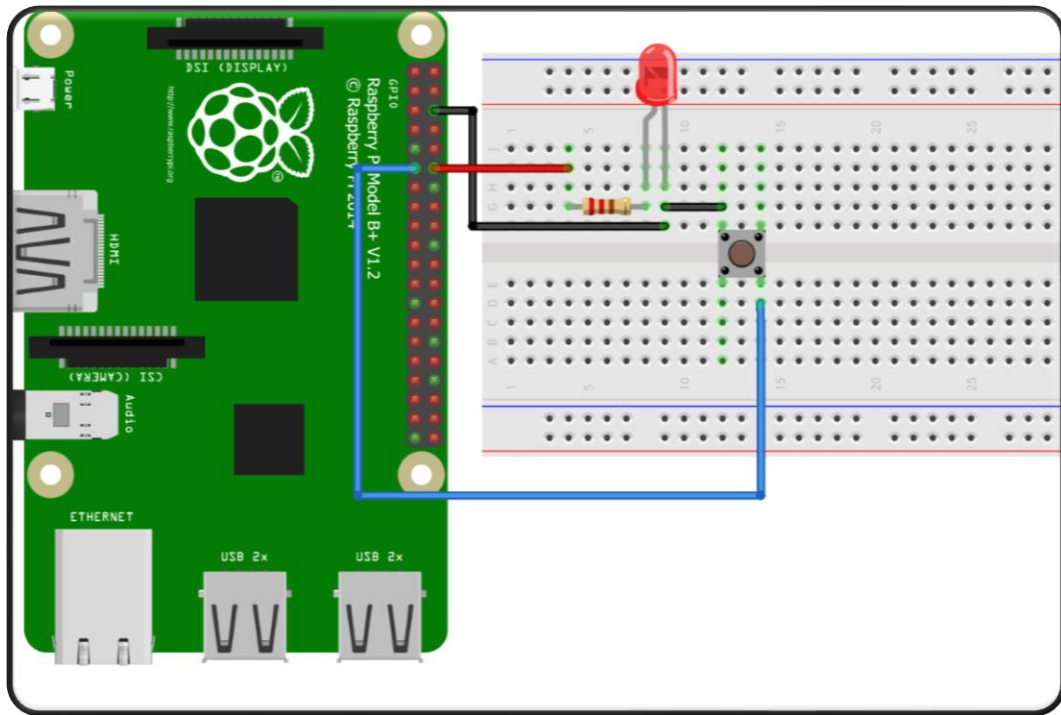


Figure 18 - LED and Button Circuit

This application will require more than just a few lines on Raspberry's Terminal, so let's create a simple C program to run our code. Start by creating a new directory and `cd` into it. Inside it, create a new C file and edit with your preferred text editor or IDE. [17]

```
sistca@raspberrypi:~ $ mkdir SISTCA
sistca@raspberrypi:~ $ cd SISTCA
sistca@raspberrypi:~ $ touch Button.c
```

To control GPIOs using a C program, we need to include WiringPi's library, `wiringPi.h`. We also need to initialize WiringPi inside our main function using `"wiringPiSetupGpio();"` or `"wiringPiSetup();"`. Both functions work, however, the former defines the GPIO Pins using the Broadcom (BCM) numbers and the latter defines them using WiringPi's numbers.

You can see below an example of a C program to turn on an LED on GPIO18 using a button on GPIO17:

0.....

```

1. #include <wiringPi.h> //Include WiringPi library
2. #include <stdio.h>

//Define GPIOs for easier readability

3. #define Button 17
4. #define LED 18

5. int main(void)
6. {
7.   wiringPiSetupGpio();    //Initializes WiringPi using Broadcom (BCM)
pin numbers

8.   pinMode(LED, OUTPUT);   //Set GPIO18 as OUTPUT
9.   pinMode(Button, INPUT); //Set GPIO17 as INPUT

10.  pullUpDnControl(Button, PUD_UP);    //Set a Pull Up Resistor on
GPIO17

11.  printf("Program Started:\n");

12.  while(1)
13.  {
14.    if(digitalRead(Button))    //If Button is HIGH
15.    {
16.      digitalWrite(LED, LOW);
17.    }
18.    else                        //If Button is LOW
19.    {
20.      digitalWrite(LED, HIGH);
21.      printf("Button pressed!\n");
22.    }
23.  }
24.  return 0;
25. }

```

Now you need to compile the program using, for example, *gcc*, and run the program, as follows:

```

sistca@raspberrypi:~ $ gcc -o Button Button.c -l wiringPi
sistca@raspberrypi:~ $ ./Button

```

A successful compilation will not produce any error messages. The `-l wiringPi` part is important as it loads the WiringPi library. [7]

4.2 Server using SAMBA

In this part of the script, we will create a NAS (Network-attached storage) [18], that allows the student to store files on the raspberry if both are in the same network. This is one of the biggest advantages of the raspberry, because it is relatively cheap if you want a home server. A low-level NAS server will cost over 160€ and it will make a lot of noise and consume a lot of electricity.

In order to do that we will install SAMBA which is a service that allows us to turn the raspberry into a file server. In this script this will work on the same network only (and on networks connected with it by visual studio code as it makes a tunnel that allows you to connect to the server on a browser with localhost) and we will link a folder on the computers to the server.

Before we install the service, we will need to add disk space to the raspberry. For most of the persons the SD card is not enough so it is usually configured with a SATA drive which can be a hard drive or a solid-state drive or a USB Hard drive, there are also SD cards available which can go up to 1TB of space, but it is usually cheaper to get a SATA drive or USB drive.

After choosing which driver the student will use (we recommend using a USB hard drive because it is something everyone usually has with them or laying around in the house) we choose which folder will be used to store the files. To do that we will first have to see the device name of the USB, mount the driver if we haven't already and link a folder to it, so first will check what is the USB drive name and then create the folder and mount it to that folder.

To check all the devices connected to the raspberry we will type the following command

```
sistca@raspberrypi:~ $ sudo fdisk -l
```

This will print an extensive list, but the USB device is located last on the list with an available space of 57.8GB. [19]

```

Disk /dev/mmcblk0: 14.64 GiB, 15720251392 bytes, 30703616 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0efb5331

Device            Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk0p1          8192   532479    524288   256M  c W95 FAT32 (LBA)
/dev/mmcblk0p2     532480 30703615 30171136  14.4G  83  Linux

Disk /dev/sda: 57.8 GiB, 62058921984 bytes, 121208832 sectors
Disk model: DataTraveler 3.0
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 61FB6CB8-EB9D-4C5E-BFC9-40F81ACD095D

Device      Start      End  Sectors  Size Type
/dev/sda1    2048 121206676 121204629  57.8G Microsoft basic data

```

Figure 19 - fdisk -l output

After locating the driver, you can create the folder where the files will be stored and then link the files to that folder, you can use the following commands to do that:

```

sistca@raspberrypi:~ $ sudo mkdir /media/usb
sistca@raspberrypi:~ $ sudo mount /dev/sda1 /media/usb

```

Now that the folder is linked, we can install SAMBA, but before that we will update and upgrade the system.

```

sistca@raspberrypi:~ $ sudo apt-get update
sistca@raspberrypi:~ $ sudo apt-get upgrade -y
sistca@raspberrypi:~ $ sudo apt-get install samba -y

```

Now that samba is installed, we want to send files to raspberry, but before we do that, we have to configure it to allow guest share, this will allow users in the same network to send files to the raspberry without using any credentials.

```

sistca@raspberrypi:~ $ sudo chmod 777 /media/usb

```

We also want to change the configuration file of samba, you can open the file and then add the following lines at the end of it, we recommend using nano, which is a Linux text editor, but you can also use VI or even VS Code. The following lines allow everyone to write and read the files on the configured folder (/media/usb).

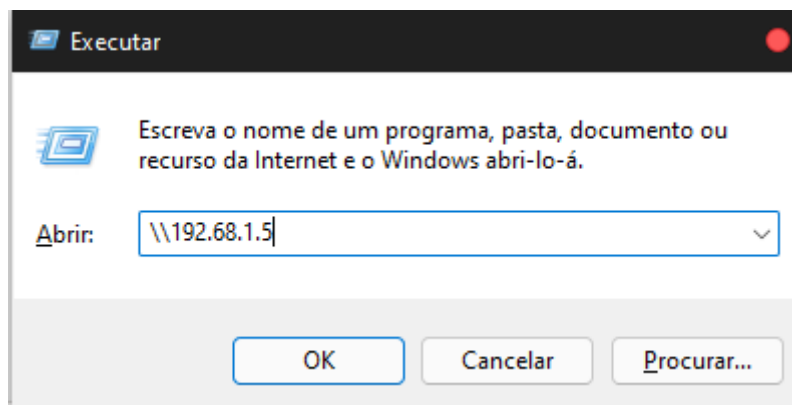
```
sistca@raspberrypi:~ $ sudo nano /etc/samba/smb.conf
```

```
[SISTCA]
comment = RaspberryPi
public = yes
writeable = yes
browsable = yes
path = /media/usb
create mask = 0777
directory mask = 0777
```

You can leave nano with CTRL + X, y and then press enter to save. After saving you should restart SAMBA.

```
sistca@raspberrypi:~ $ sudo service smb restart
```

After doing this SAMBA will be working and we can access it, in windows you can press WINDOWS+R and type \\RaspberryIP.



Now we have access to the server and can manipulate all the files through windows.

If you can't write to the folder your driver might be in FAT32 or NTFS, and you should format it to EXT4, to do that you should unmount the drive and then format it[20].

```
sistca@raspberrypi:~ $ sudo umount /dev/sda1
sistca@raspberrypi:~ $ sudo mkfs.ext4 /dev/sda1
```

Now you have your own server working with your raspberry. First you should install *npm* (Node Package Manager) which is a package manager [19].

5. Exercise(s)

5.1 Problem A

5.1.1 Material

To assemble the circuit in this exercise, you will require the following:

- 1x RGB LED
- 2x Push buttons
- 3x 1k Ω Resistors
- 6x Male to Female jumper wires
- 3x Male to Male jumper wires

5.1.2 Circuit and problem

Now that you have learned how to control the GPIO pins using WiringPi's library assemble the circuit below:

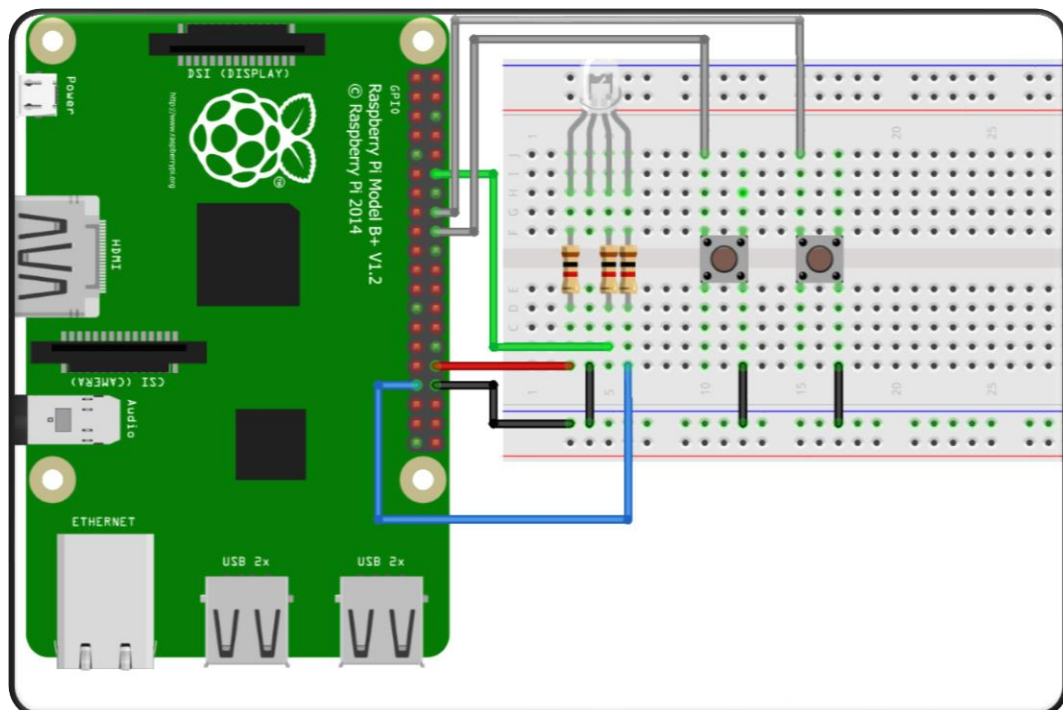


Figure 21 - Circuit problem

Using an RGB LED, connect each cathode to GPIOs 12, 13 and 18 through 1k Ω resistors. Connect one terminal of the push buttons to GPIOs 23 and 24, ground the other terminal.

Using this circuit as base, create a C program that displays a menu to the user with the following options:

- LED
- Button
- PWM Light Follower

Inside the LED menu, create a submenu that allows the user to choose what colour or combination of colours they want to light the LED with, or the option to turn it off.

In the second menu option, the button one, create a cycle in which every time the button is pressed, a message will be printed in the console saying how many times it has been pressed. On the first press, the LED should turn red for half a second, on the second press, green, and blue on the third press, and return to the main menu.

The third menu will make use of the PWM functionalities to create a cycle of lights. Create a cycle in which the LED will increase its brightness until maximum, where it should turn off and switch to the next colour and repeat the brightening. Don't forget that GPIO12 and GPIO18 share the same PWM channel, thus controlling one also changes the value on the other.

Use the second button to create an interrupt to turn the LED off and return to the main menu.

5.1.3 (re)solution A

A code with a possible solution is available at:
<https://github.com/ruitails/SISTCA.git>

You can simply clone the repository, compile, and run the program. The exercise is separated in 3 functions for each main menu option and a main function.

5.2Problem B

The way the server is configured right now everyone can access it if they have the IP address, make it so when someone wants to connect is asks for an ID and a password. You should use the command "smbpasswd".

5.2.1 (re)solution B

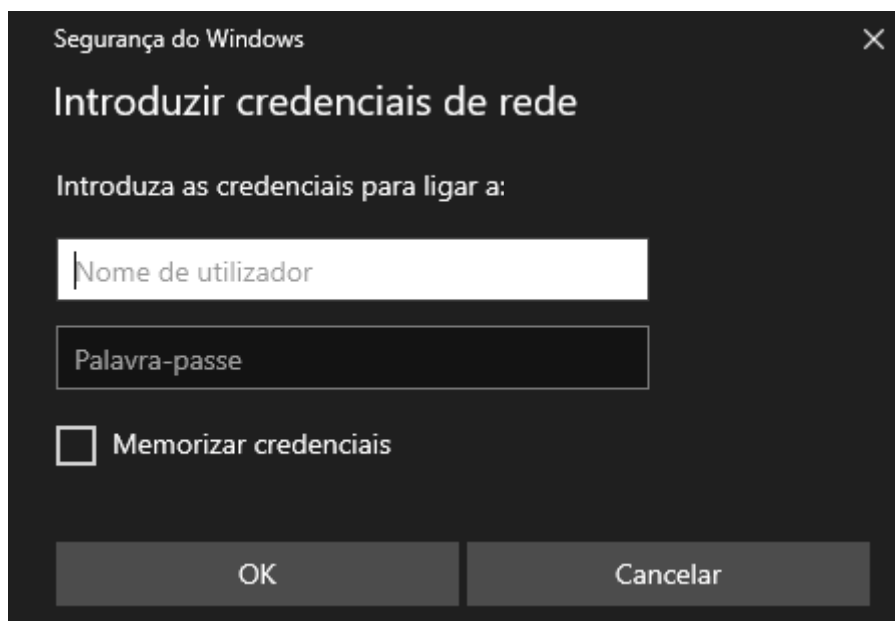
To request a password when connecting change, at the end of the SAMBA configuration file the public variable so that the server is no longer public:

```
[SISTCA]
comment = RaspberryPi
public = no
writeable = yes
browsable = yes
path = /media/usb
create mask = 0777
directory mask = 0777
```

Now you should tell SAMBA that the user is allowed to connect with:

```
sistca@raspberrypi:~ $ sudo adduser student
sistca@raspberrypi:~ $ sudo smbpasswd -a student
sistca@raspberrypi:~ $ sudo service smbd restart
```

Now when you try to connect it should prompt you to enter the log in credentials:



Now you can log in in the server and store all your files.

6. References

- [8] <https://www.raspberrypi.com/products/raspberry-pi-2-model-b/>
- [13] <https://www.digiteum.com/raspberry-pi-for-small-business/>
- [1] "ENIAC, o cérebro gigante, faz 70 anos - Tecnologias - Jornal de Negócios." https://www.jornaldenegocios.pt/empresas/tecnologias/detalhe/eniac_o_cerebro_gigante_faz_70_anos (accessed May 16, 2022).
- [2] "What is a Raspberry Pi?" <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/> (accessed May 16, 2022).
- [3] "Difference between Arduino and Raspberry Pi - GeeksforGeeks." <https://www.geeksforgeeks.org/difference-between-arduino-and-raspberry-pi/> (accessed May 16, 2022).
- [4] "What are the differences between Raspberry Pi and Arduino?" <https://www.electronicshub.org/raspberry-pi-vs-arduino/> (accessed May 16, 2022).
- [5] S. Shovon, "Install Etcher on Linux." https://linuxhint.com/install_etcher_linux/ (accessed Apr. 05, 2022).
- [6] G. Halfacree, *Raspberry Pi Beginners Guide*. 2020.
- [7] "Raspberry GPIO - learn.sparkfun.com." <https://learn.sparkfun.com/tutorials/raspberry-gpio/all> (accessed May 14, 2022).
- [8] G. Henderson, "Raspberry Pi | Wiring | Download & Install | Wiring Pi." <http://wiringpi.com/download-and-install/> (accessed May 14, 2022).
- [9] "WiringPi/WiringPi: Gordon's Arduino wiring-like WiringPi Library for the Raspberry Pi (Unofficial Mirror for WiringPi bindings)." <https://github.com/WiringPi/WiringPi> (accessed May 15, 2022).
- [10] G. Henderson, "WiringPi." <http://wiringpi.com/> (accessed May 14, 2022).
- [11] G. Henderson, "The GPIO utility | Wiring Pi." <http://wiringpi.com/the-gpio-utility/> (accessed May 14, 2022).
- [12] G. Henderson, "Setup | Wiring Pi." <http://wiringpi.com/reference/setup/> (accessed May 15, 2022).
- [13] G. Henderson, "Core Functions | Wiring Pi." <http://wiringpi.com/reference/core-functions/> (accessed May 15, 2022).
- [14] G. Henderson, "Timing | Wiring Pi." <http://wiringpi.com/reference/timing/> (accessed May 15, 2022).
- [15] G. Henderson, "Raspberry Pi | WiringPi | Functions | Priorities, Interrupts and Threads." <http://wiringpi.com/reference/priority-interrupts-and-threads/> (accessed May 15, 2022).

- [16] "Raspberry Pi GPIO Pinout." <https://pinout.xyz/> (accessed May 12, 2022).
- [17] J. Tranter, "How to Control GPIO Hardware from C or C++ | ICS." <https://www.ics.com/blog/how-control-gpio-hardware-c-or-c> (accessed May 22, 2022).
- [18] "What is Network-Attached Storage (NAS) and How Does it Work?" <https://www.techtarget.com/searchstorage/definition/network-attached-storage> (accessed May 19, 2022).
- [19] "How to turn a Raspberry Pi into a file server? – RaspberryTips." <https://raspberrytips.com/raspberry-pi-file-server/> (accessed May 19, 2022).
- [20] "How to format and mount a USB drive on Raspberry Pi? – RaspberryTips." <https://raspberrytips.com/format-mount-usb-drive/> (accessed May 19, 2022).