

# Streaming de video y control remoto

Tecnologías Multimedia

Vitor Hugo Augusto, Jorge Vicente, Jonathan López, Antonia Rubio, Agustín Casado, Andrés Rueda y Germán Ruano

# ÍNDICE

[1. Introducción](#)

[2. Materiales](#)

[3. Planificación](#)

[4. Desarrollo](#)

[4.1 Instalación del escritorio remoto](#)

[4.2 Instalación servidor web LIGHTTPD](#)

[4.3 Instalación librería WiringPI](#)

[4.4 Instalación librería mjpg-stream](#)

[4.5 Instalación openCV](#)

[4.6 Estructura proyecto](#)

[4.7 Servidor, movimiento y cámara](#)

[4.8 Reconocimiento Facial](#)

[5. Montaje](#)

[5.1 Montaje Puente H](#)

[6. Anexo montaje impresora](#)

[6.2 Materiales](#)

[6.3 Montaje](#)

[6.4 Programación y calibración](#)

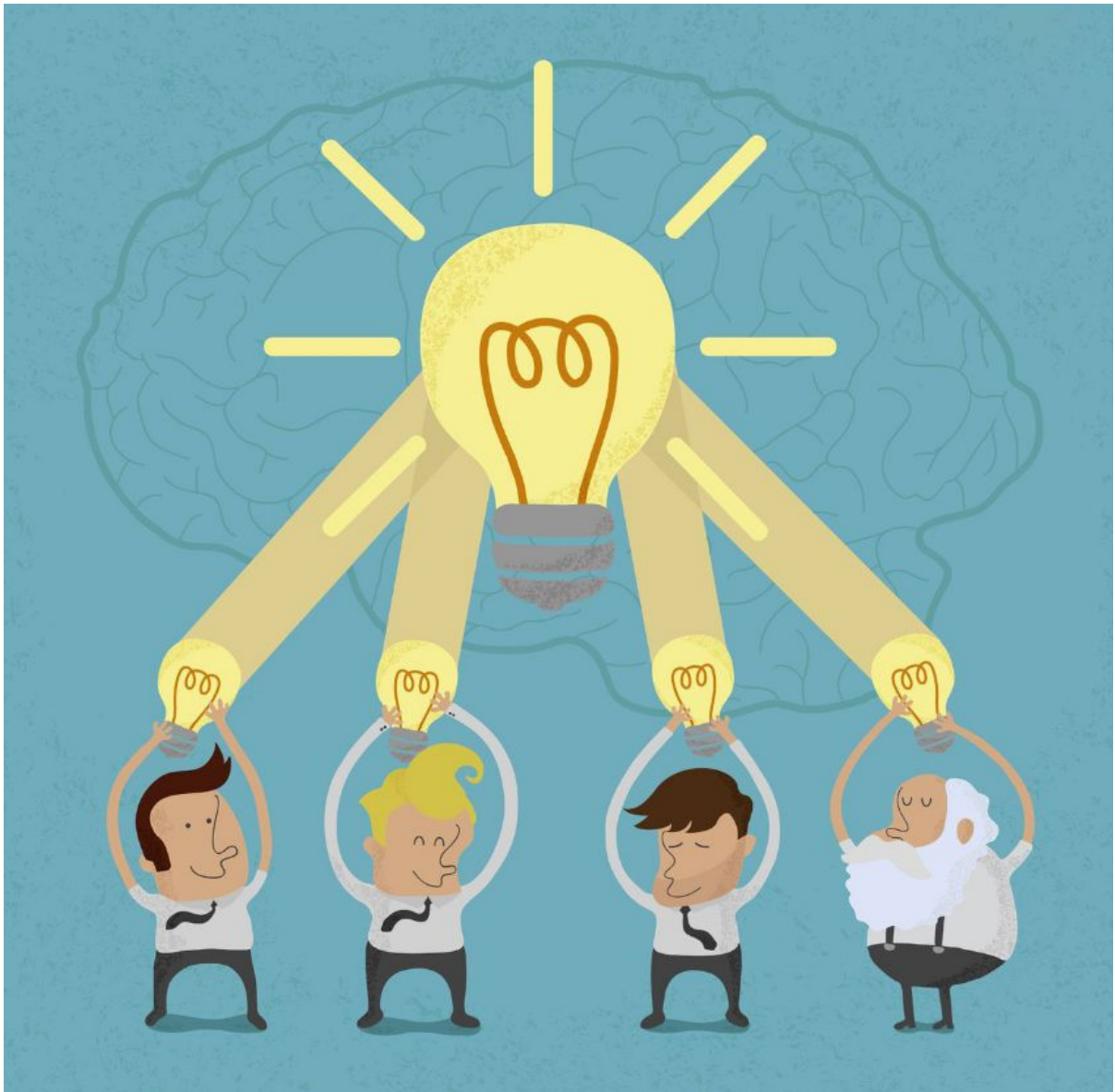
[7. Enlace proyecto GitHub](#)

[8. Bibliografía](#)

# 1. Introducción

En este proyecto vamos a realizar un tanque usando Raspberry pi 3 con las siguientes funcionalidades:

- Controlado a distancia vía wifi mediante interfaz web.
- Reconocimiento facial.
- Emisión de video streaming.



## 2. Materiales

### Modelo B, Raspberry Pi 3



Procesador	GPU	RAM	Conectividad
<ul style="list-style-type: none"> <li>- Chipset Broadcom BCM2387</li> <li>- 1.2 GHz de cuatro núcleos ARM Cortex-A53</li> </ul>	<ul style="list-style-type: none"> <li>- Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.</li> <li>- Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA</li> </ul>	<ul style="list-style-type: none"> <li>- 1 GB LPDDR2</li> </ul>	<ul style="list-style-type: none"> <li>- Ethernet socket Ethernet 10/100 BaseT</li> <li>- 802.11 b / g / n LAN inalámbrica y Bluetooth 4.1 (Classic Bluetooth y LE)</li> <li>- Salida de vídeo: <ul style="list-style-type: none"> <li>+ HDMI rev 1.3 y 1.4</li> <li>+ RCA</li> </ul> </li> <li>- Salida de audio: <ul style="list-style-type: none"> <li>+ jack de 3,5 mm de salida de audio, HDMI</li> <li>+ USB 4 x Conector USB 2.0</li> </ul> </li> <li>- Conector GPIO <ul style="list-style-type: none"> <li>+ 40-clavijas de 2,54 mm</li> <li>+ Proporcionar 27 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro</li> </ul> </li> <li>- Cámara de 15 pines cámara MIPI interfaz en serie (CSI-2)</li> <li>- Ranura de tarjeta de memoria Micro SDIO</li> </ul>

## Bateria externa

Especificaciones técnicas:

- Batería de Ion-Litio
- 2600 mAh de capacidad
- Permite la recarga de modo simultáneo a la carga del dispositivo

Puertos:

- Salida: USB 5V/1A
- Entrada: micro USB 5V/1A

Cables/conectores incluidos:

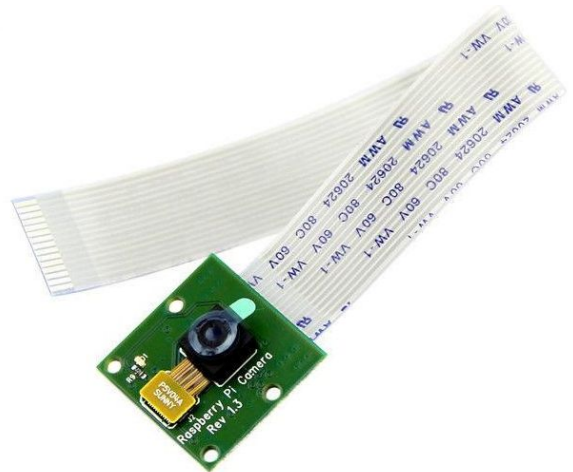
- Micro USB



## Cámara pi revision 3

Dimensiones:

- profundidad: 20 mm
- altura: 10 mm



## DC Motors (2x)

Características:

- Reducción: 1:120
- Velocidad sin carga(3V): 50RPM
- Velocidad sin carga(5V): 83RPM
- Par(3V): 0.8Kgcm
- Par(5V): 1.0Kgcm
- Peso: 30g



## Micro SD

- Capacidad 16GB
- Dimensiones: 11mm x 15mm x 1.0mm
- Op. Voltaje: 2.7V ~ 3.6V

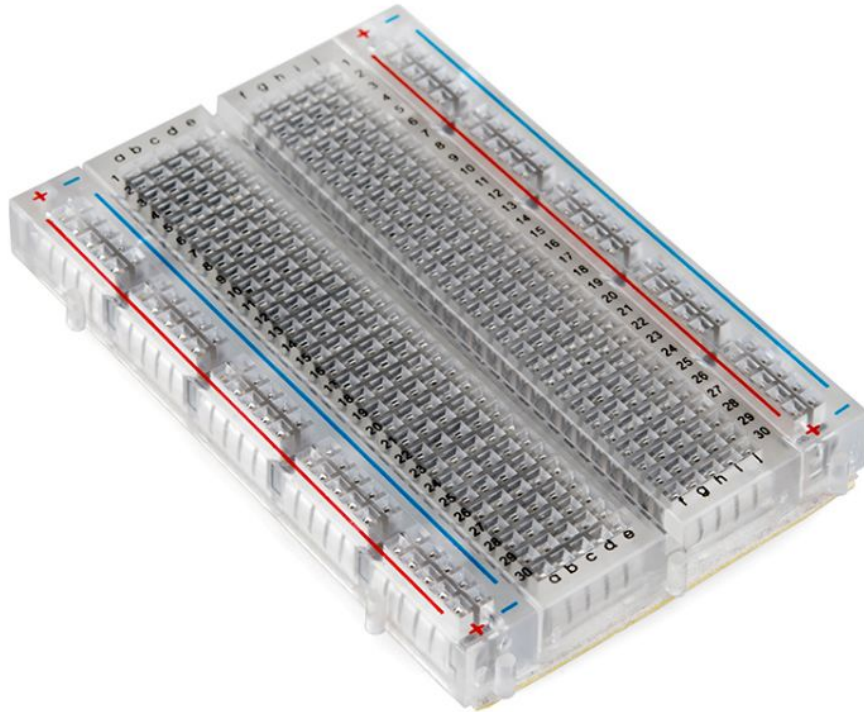


## L293-D H Bridge

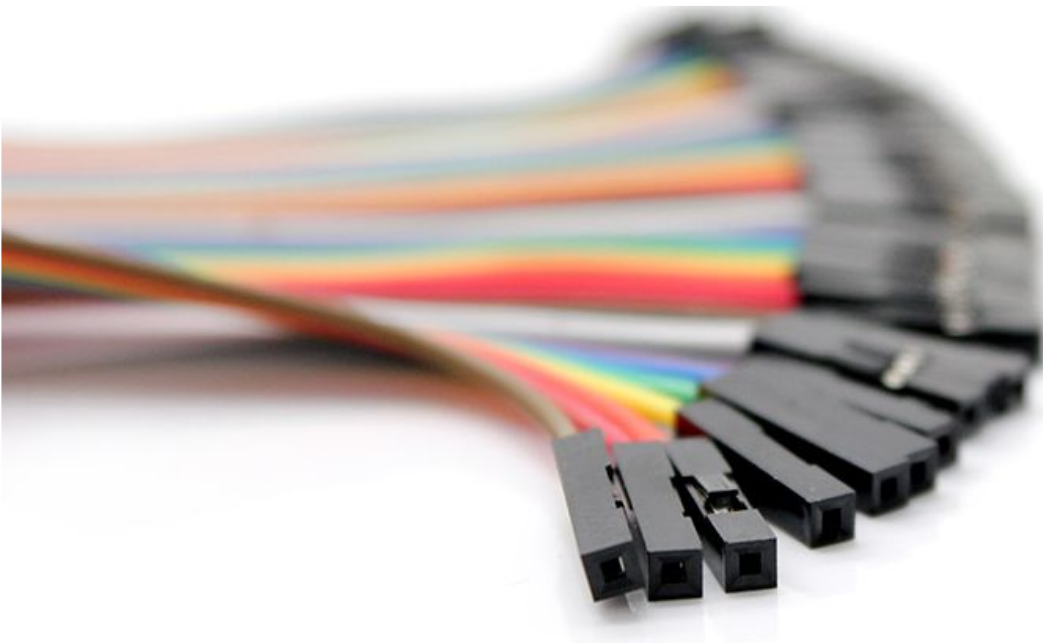




## Breadboard



## Cables



## Faltan materiales impresos

### 3. Planificación

Esta es nuestra planificación inicial de la realización del proyecto, la cual hemos modificado al repartimos el trabajo entre nuestros componentes, quitando así una semana de trabajo, lo que nos llevó por lo tanto a volver a realizar la planificación de nuevo, pero esta vez metiendo los tiempos reales.

	Semana1	Semana2	Semana3	Semana4	Semana5	Semana6	Semana7	Semana8	Semana9	Semana10	Semana11	Semana12	Semana13	Semana14	Semana15
Memoria Inicio	X														
Iniciando Pi		X	X												
Vídeo Stream				X	X										
Interfaz Web						X	X								
Face Tracking								X	X	X					
Creacion de Tanque			X	X	X	X									
Uniendo Componentes											X				
Testeo final												X	X		
Memoria Final														X	X



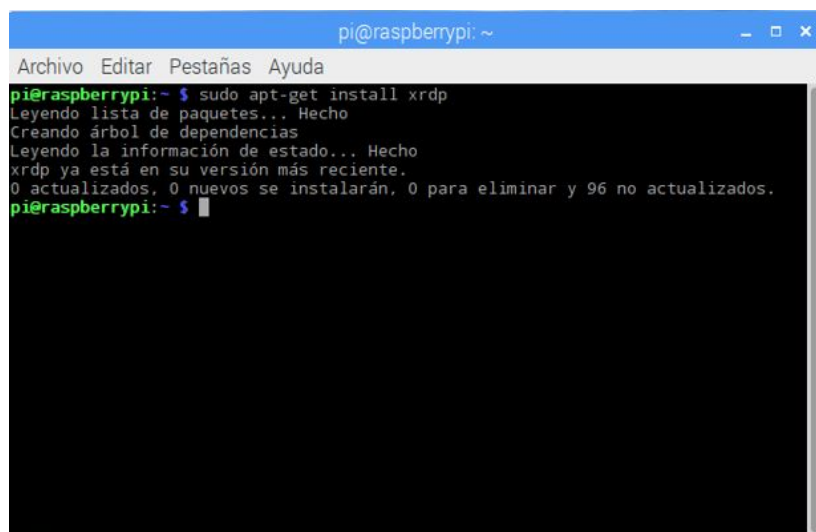
## 4. Desarrollo

Para crear nuestro proyecto lo hemos dividido en varios apartados

### 4.1 Instalación del escritorio remoto

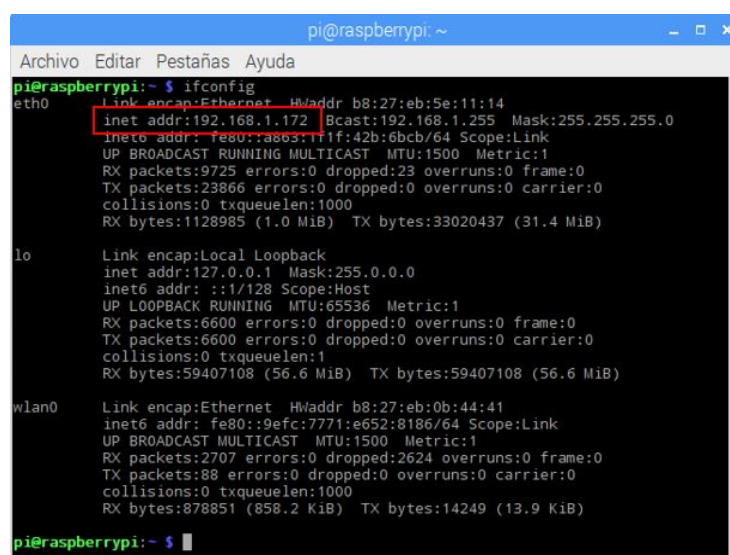
En primer lugar, debemos realizar un update a nuestra raspberry con el comando, apt-get update.

Para el escritorio remoto utilizaremos el paquete xrdp, que instalaremos de la siguiente forma, con el comando: apt-get install xrdp.



```
pi@raspberrypi: ~  
Archivo Editar Pestañas Ayuda  
pi@raspberrypi:~$ sudo apt-get install xrdp  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
xrdp ya está en su versión más reciente.  
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 96 no actualizados.  
pi@raspberrypi:~$
```

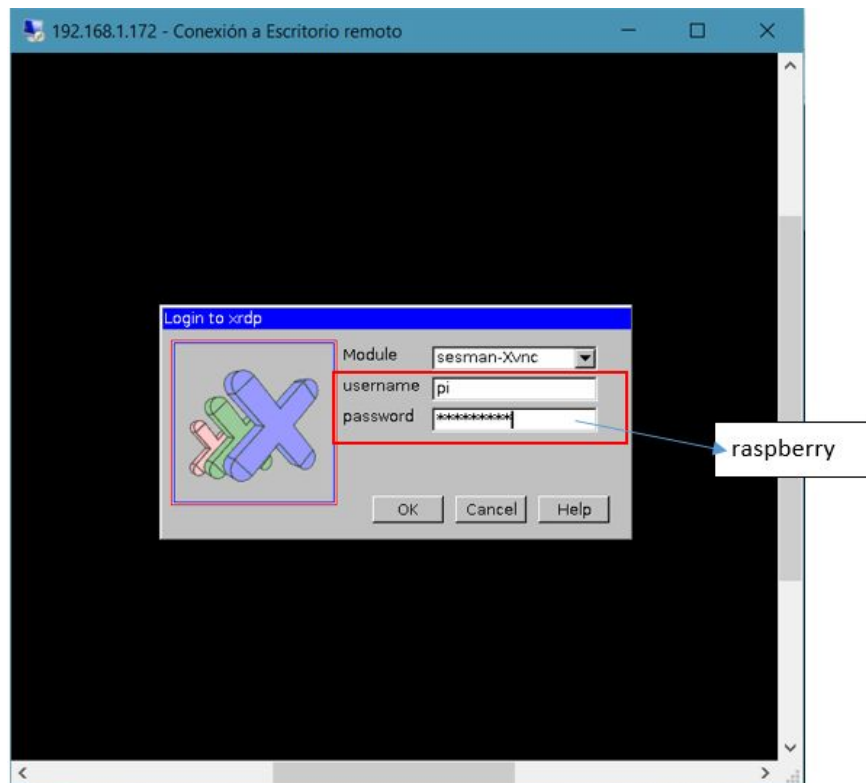
Conectamos nuestra raspberry por cable o por wifi a internet y comprobamos la ip asignada.



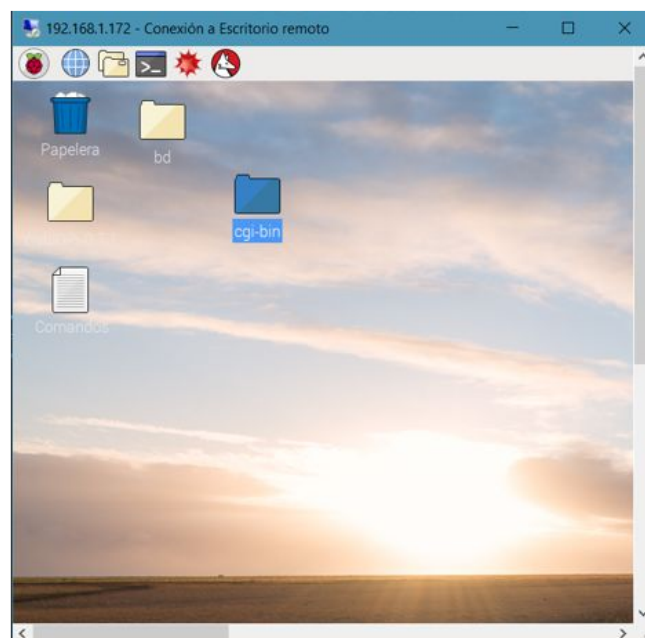
```
pi@raspberrypi: ~  
Archivo Editar Pestañas Ayuda  
pi@raspberrypi:~$ ifconfig  
eth0: Link encap:Ethernet HWaddr b8:27:eb:5e:11:14  
      inet addr:192.168.1.172 Bcast:192.168.1.255 Mask:255.255.255.0  
      inet6 addr: fe80::a863:111f:42b:6bcb/64 Scope:Link  
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
      RX packets:9725 errors:0 dropped:23 overruns:0 frame:0  
      TX packets:23866 errors:0 dropped:0 overruns:0 carrier:0  
      collisions:0 txqueuelen:1000  
      RX bytes:1128985 (1.0 MiB) TX bytes:33020437 (31.4 MiB)  
  
lo:    Link encap:Local Loopback  
      inet addr:127.0.0.1 Mask:255.0.0.0  
      inet6 addr: ::1/128 Scope:Host  
      UP LOOPBACK RUNNING MTU:65536 Metric:1  
      RX packets:6600 errors:0 dropped:0 overruns:0 frame:0  
      TX packets:6600 errors:0 dropped:0 overruns:0 carrier:0  
      collisions:0 txqueuelen:1  
      RX bytes:59407108 (56.6 MiB) TX bytes:59407108 (56.6 MiB)  
  
wlan0: Link encap:Ethernet HWaddr b8:27:eb:0b:44:41  
      inet6 addr: fe80::9efc:7771:e652:8186/64 Scope:Link  
      UP BROADCAST MULTICAST MTU:1500 Metric:1  
      RX packets:2707 errors:0 dropped:2624 overruns:0 frame:0  
      TX packets:88 errors:0 dropped:0 overruns:0 carrier:0  
      collisions:0 txqueuelen:1000  
      RX bytes:878851 (858.2 KiB) TX bytes:14249 (13.9 KiB)  
  
pi@raspberrypi:~$
```

En nuestro caso realizaremos la conexión desde un equipo Windows por lo que podremos utilizar la aplicación escritorio remoto por defecto de Windows, en la

cual usaremos la ip asignada anterior, al conectarnos nos saldrá la siguiente ventana, donde deberemos autenticarnos con el usuario pi y la contraseña raspberry.



Y con esto ya tendríamos la conexión por escritorio remoto con nuestra raspberry.

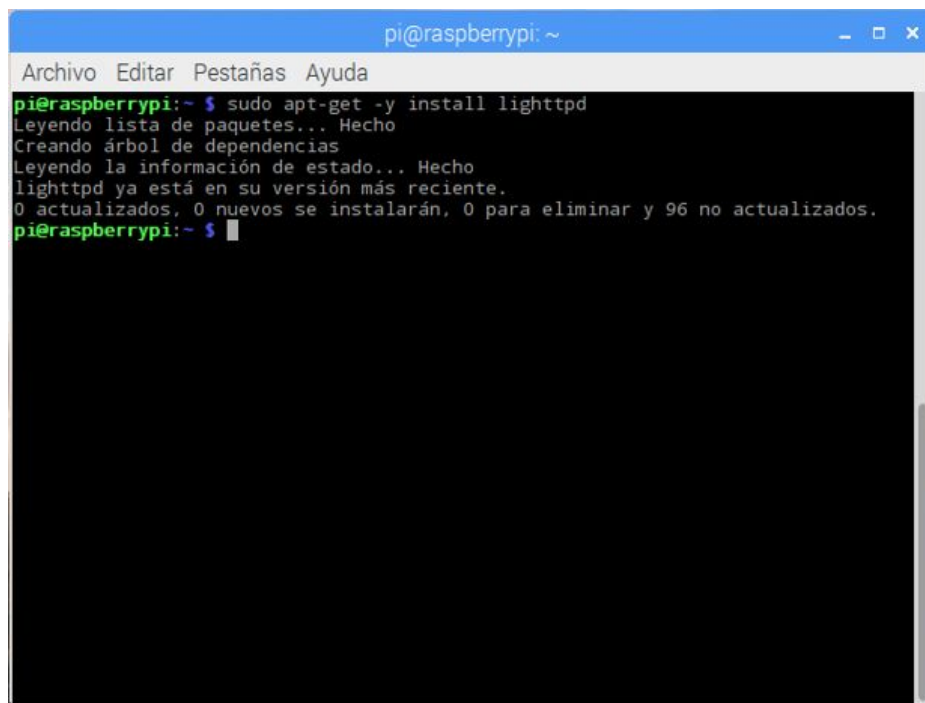


## 4.2 Instalación servidor web LIGHTTPD

Para el servidor web nos hemos decidido por lighttpd ya que es un servidor bastante ligero y rápido, aunque también podemos utilizar cualquier otro como por ejemplo apache.

Para ello utilizaremos el siguiente comando para su instalación: `apt-get -y install lighttpd`.

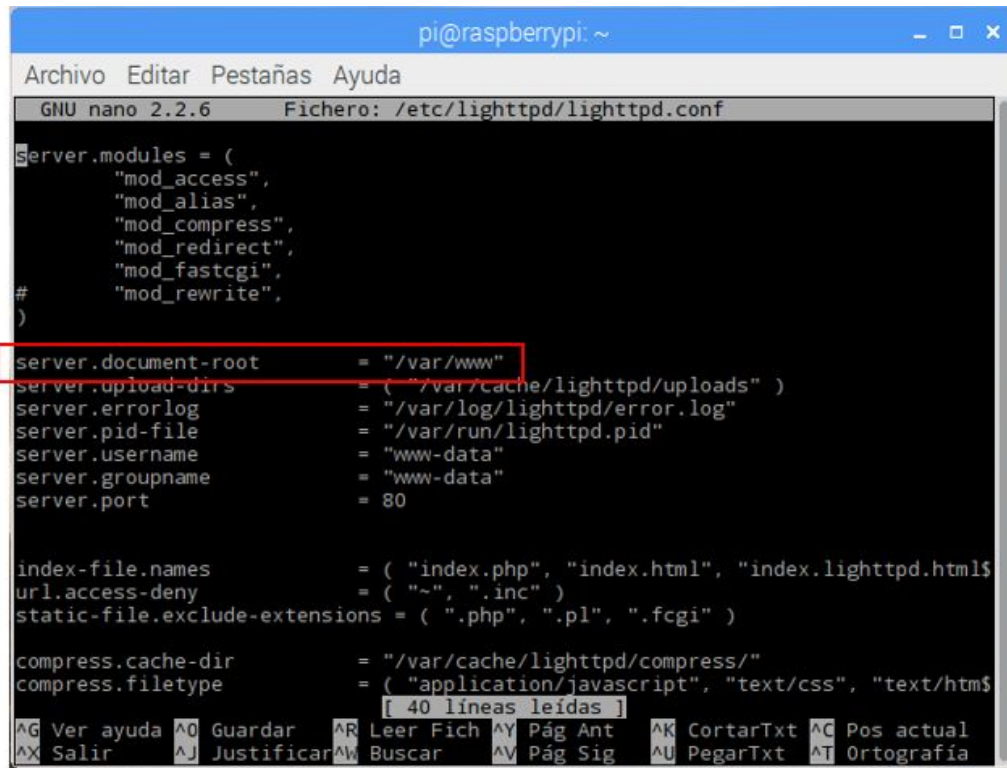
Además, activaremos el modo cgi para poder utilizar archivos .cgi que veremos en el siguiente punto. Para ello utilizaremos los siguientes comandos: `lighttpd-enable-mod cgi` y `lighttpd-enable-mod fastcgi`.



```
pi@raspberrypi: ~  
Archivo Editar Pestañas Ayuda  
pi@raspberrypi:~ $ sudo apt-get -y install lighttpd  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
lighttpd ya está en su versión más reciente.  
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 96 no actualizados.  
pi@raspberrypi:~ $
```

Modificaremos también la ruta predeterminada del servidor, para ello abrimos el archivo `lighttpd.conf` con el siguiente comando: `sudo nano /etc/lighttpd/lighttpd.conf`.

Moveremos nuestro directorio por defecto a la carpeta `/var/www`.



```
pi@raspberrypi: ~
GNU nano 2.2.6 Fichero: /etc/lighttpd/lighttpd.conf

Server.modules = (
    "mod_access",
    "mod_alias",
    "mod_compress",
    "mod_redirect",
    "mod_fastcgi",
    #
    "mod_rewrite",
)

server.document-root = "/var/www"
server.upload-dirs = ( "/var/cache/lighttpd/uploads" )
server.errorlog = "/var/log/lighttpd/error.log"
server.pid-file = "/var/run/lighttpd.pid"
server.username = "www-data"
server.groupname = "www-data"
server.port = 80

index-file.names = ( "index.php", "index.html", "index.lighttpd.html" )
url.access-deny = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )

compress.cache-dir = "/var/cache/lighttpd/compress/"
compress.filetype = ( "application/javascript", "text/css", "text/html" )

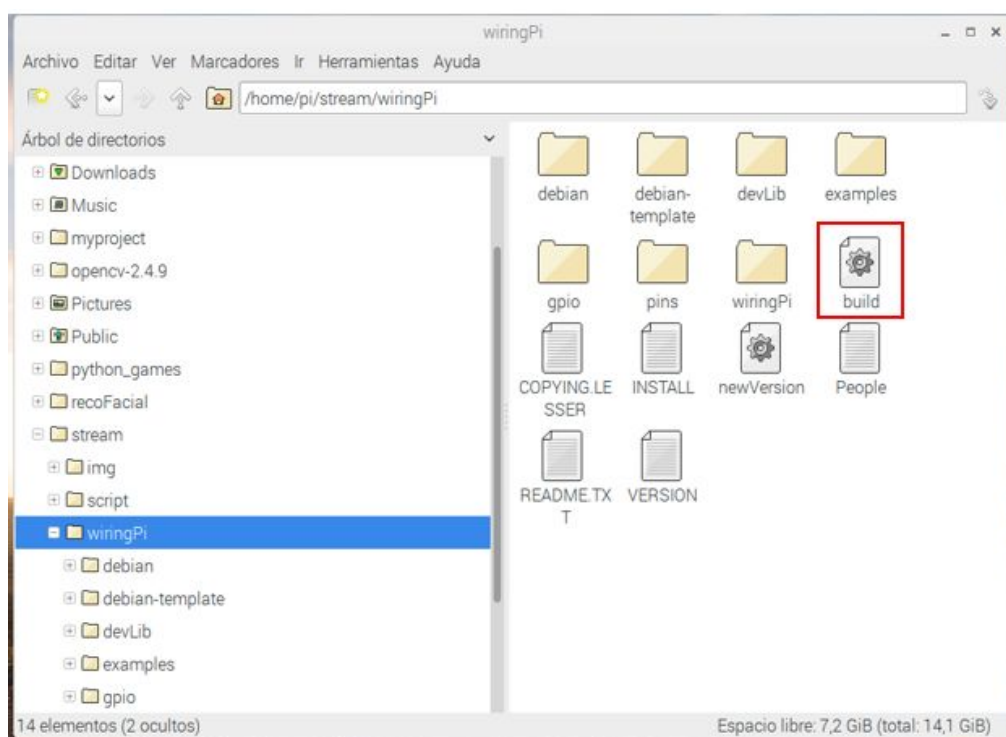
[ 40 líneas leídas ]
^G Ver ayuda ^O Guardar ^R Leer Fich ^Y Pág Ant ^K CortarTxt ^C Pos actual
^X Salir ^J Justificar ^W Buscar ^V Pág Sig ^U PegarTxt ^T Ortografía
```

## 4.3 Instalación librería WiringPi

Para poder tener acceso a los GPIO de las raspberry utilizaremos la librería WiringPi, con la que podremos modificar de manera sencilla los pines, con el comando gpio.

Para instalar dicha librería deberemos descargarnos el siguiente repositorio: `git clone git://git.drogon.net/wiringPi`.

Una vez descargado nos movemos a la carpeta wiringPi y tendremos lo siguiente.



Lo único que tendremos que hacer es ejecutar el archivo build y ya tendremos nuestra librería instalada.

Para comprobar la versión podemos utilizar el comando `gpio -v`.

```

pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
pi@raspberrypi:~$ gpio -v
gpio version: 2.32
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
Type: Pi 3, Revision: 02, Memory: 1024MB, Maker: Sony
* Device tree is enabled.
* This Raspberry Pi supports user-level GPIO access.
-> See the man-page for more details
-> ie. export WIRINGPI GPIOMEM=1
pi@raspberrypi:~$ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | 8 | 3.3v | | | 1 | 2 | | | 5v | | |
| 3 | 9 | SDA.1 | ALT0 | 1 | 3 | 4 | | | 5V | | |
| 4 | 7 | SCL.1 | IN | 1 | 5 | 6 | | | 0v | | |
| 17 | 0 | GPIO. 7 | IN | 0 | 7 | 8 | 1 | ALT5 | TxD | 15 | 14 |
| 27 | 2 | GPIO. 0 | IN | 0 | 9 | 10 | 1 | ALT5 | RxD | 16 | 15 |
| 22 | 3 | GPIO. 2 | IN | 0 | 11 | 12 | 0 | ALT5 | GPIO. 1 | 1 | 18 |
| 10 | 12 | GPIO. 3 | IN | 0 | 13 | 14 | 0 | IN | 0v | 4 | 23 |
| 9 | 13 | 3.3v | | | 15 | 16 | 0 | IN | GPIO. 4 | 5 | 24 |
| 11 | 14 | MOSI | OUT | 0 | 17 | 18 | 0 | IN | GPIO. 5 | 6 | 25 |
| 13 | 23 | MISO | ALT0 | 0 | 19 | 20 | 0 | IN | 0v | 10 | 8 |
| 26 | 25 | SCLK | ALT0 | 0 | 21 | 22 | 0 | IN | CE0 | 11 | 7 |
| 0 | 30 | SDA.0 | IN | 1 | 23 | 24 | 1 | OUT | SCL.0 | 31 | 1 |
| 5 | 21 | GPIO.21 | OUT | 0 | 25 | 26 | 1 | IN | 0v | 26 | 12 |
| 6 | 22 | GPIO.22 | OUT | 0 | 27 | 28 | 1 | IN | 0v | 27 | 16 |
| 13 | 23 | GPIO.23 | OUT | 0 | 29 | 30 | 0 | IN | GPIO.26 | 28 | 20 |
| 19 | 24 | GPIO.24 | OUT | 0 | 31 | 32 | 0 | IN | GPIO.27 | 29 | 21 |
| 26 | 25 | GPIO.25 | IN | 0 | 33 | 34 | 0 | IN | | |
| | | 0v | | | 35 | 36 | 0 | IN | | |
| | | | | | 37 | 38 | 0 | IN | | |
| | | | | | 39 | 40 | 0 | IN | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

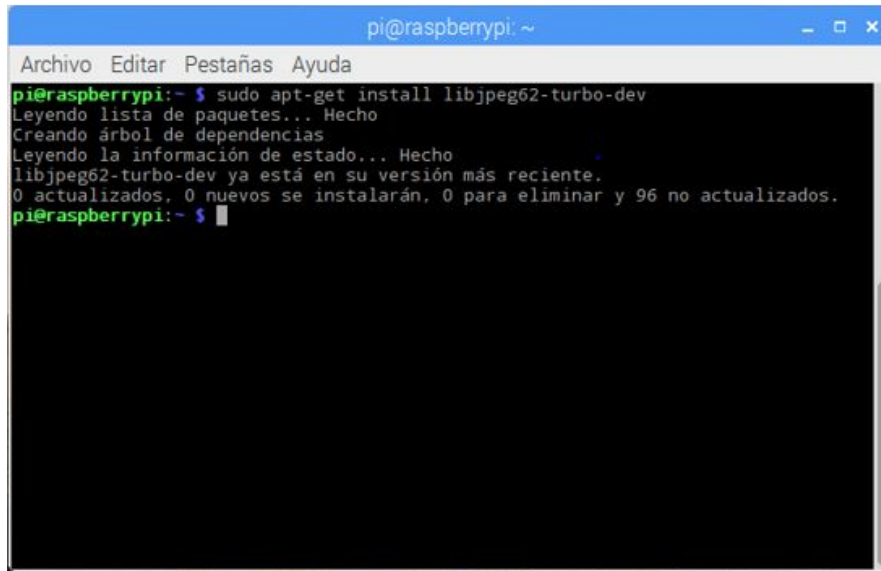
```

Otros dos comandos importantes que utilizaremos serán `gpio readall` para comprobar el estado de nuestros pines y `gpio unexportall` para borrar las configuraciones anteriores.



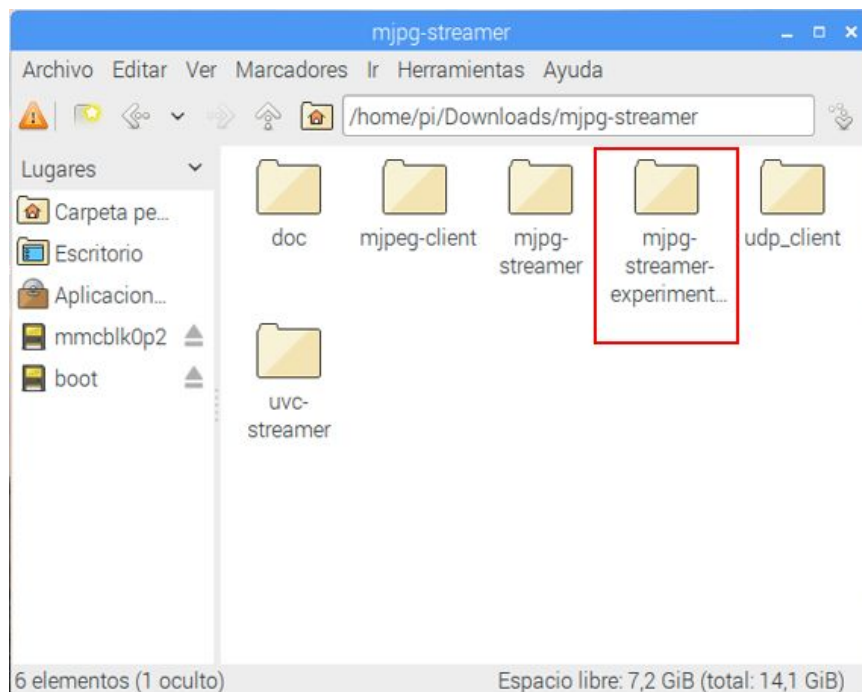
## 4.4 Instalación librería mjpg-stream

Primero instalaremos la librería libjpeg con el siguiente comando: `apt-get install libjpeg62-turbo-dev`.



```
pi@raspberrypi: ~  
Archivo Editar Pestañas Ayuda  
pi@raspberrypi:~$ sudo apt-get install libjpeg62-turbo-dev  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
libjpeg62-turbo-dev ya está en su versión más reciente.  
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 96 no actualizados.  
pi@raspberrypi:~$
```

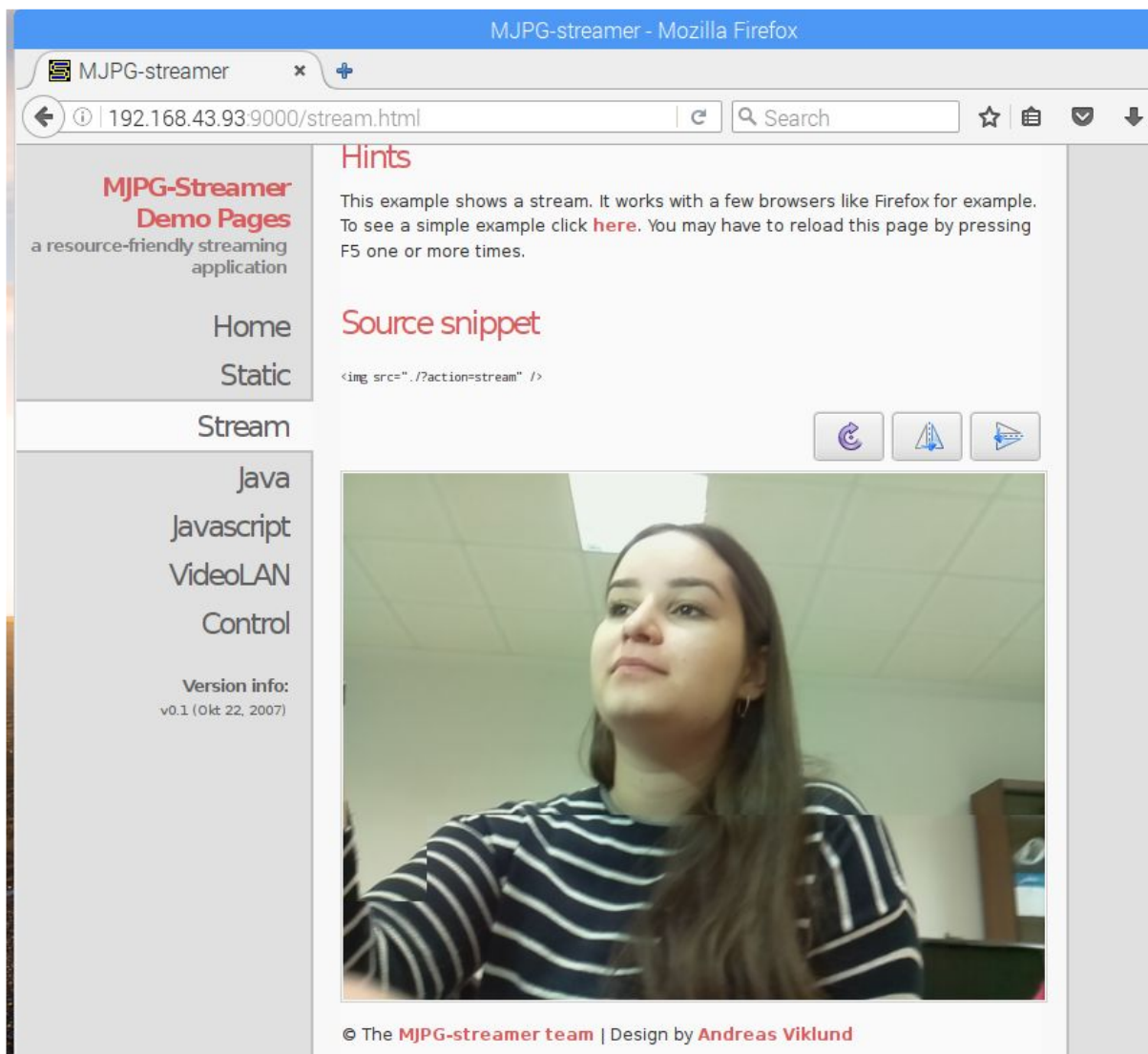
Descargamos el repositorio que contiene mjpg-streamer.  
`git clone https://github.com/jacksonliam/mjpg-streamer.git ~/mjpg-streamer`



Y nos movemos a la carpeta mjpg-streamer-experiemental y compilamos con make clean all.

Para activar el stream haremos uso de lo siguiente:

```
LD_LIBRARY_PATH=/opt/mjpg-streamer//opt/mjpg-streamer/mjpg_s  
treamer-i "input_raspicam.so -fps 15 -q 50 -x 640 -y 480" -o  
"output_http.so -p 9000 -w /opt/mjpg-streamer/www" &
```



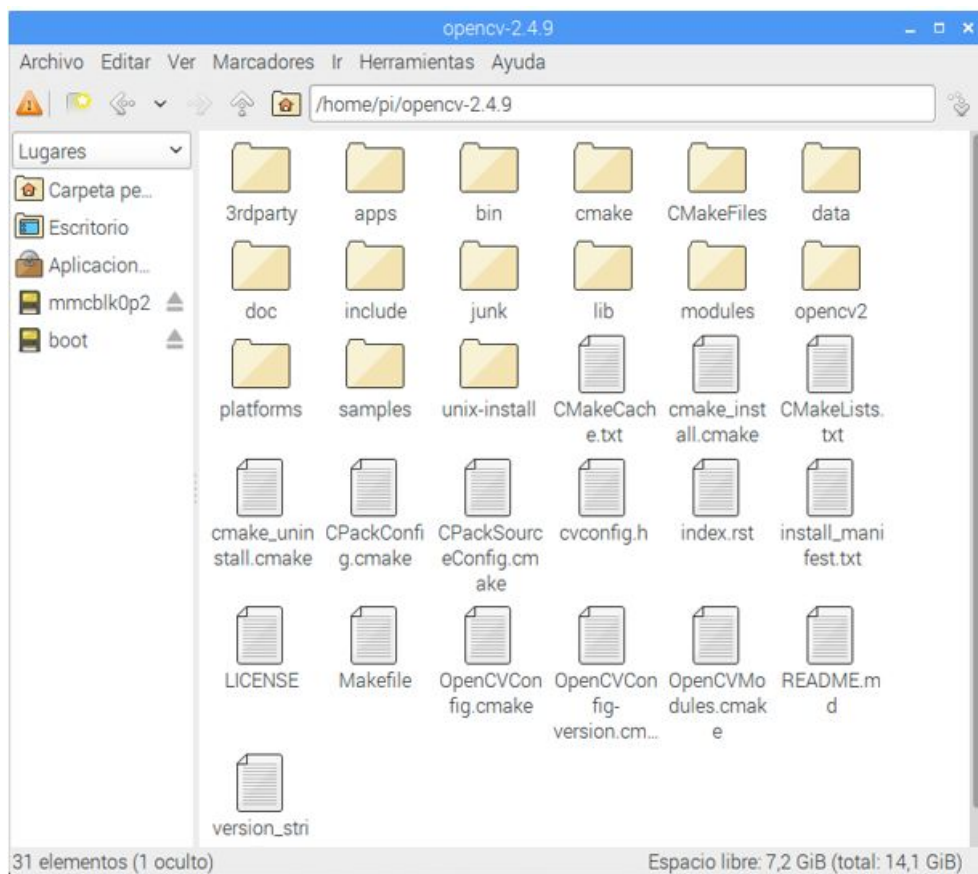
## 4.5 Instalación openCV

Primero deberemos instalar diversos repositorios para que no tengamos posibles errores a la hora de compilar openCV.

```
sudo apt-get install build-essential cmake pkg-config python-dev  
libgtk2.0-dev libgtk2.0 zlib1g-dev libpng-dev libjpeg-dev libtiff-dev  
libjasper-dev libavcodec-dev swig unzip  
sudo apt-get install python-numpy python-opencv  
sudo apt-get install python-pip  
sudo apt-get install python-dev  
sudo pip install picamera  
sudo pip install rpio
```

Ahora descargamos openCV con el siguiente comando, wget <http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.9/opencv-2.4.9.zip>

Una vez descargado y descomprimido tendremos la siguiente carpeta.



Compilamos openCV con el siguiente comando:

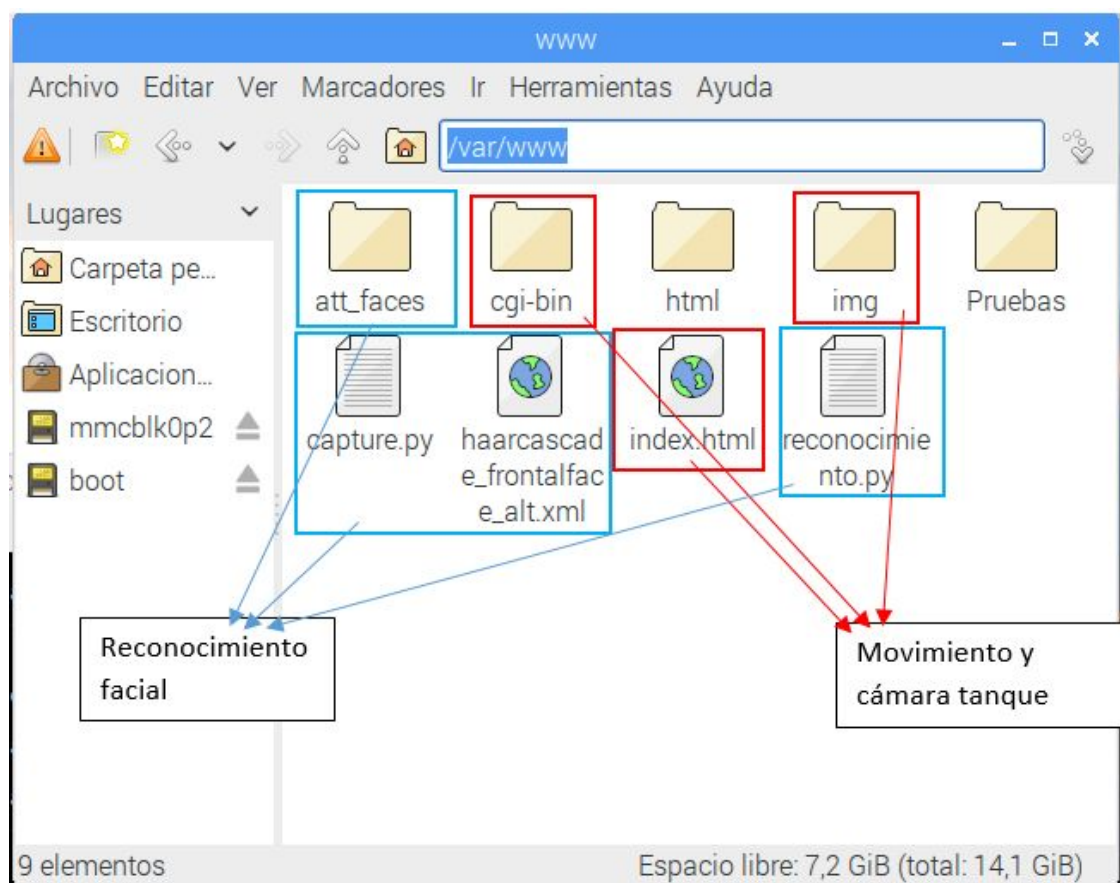
```
cmake-DCMAKE_BUILD_TYPE=RELEASE  
-DCMAKE_INSTALL_PREFIX=/usr/local -DBUILD_PERF_TESTS=OFF  
-DBUILD_opencv_gpu=OFF -DBUILD_opencv_ocl=OFF.
```

Y una vez acabe lo instalamos con el comando: `sudo make install`.

## 4.6 Estructura proyecto

Para estructurar nuestro proyecto de una manera clara lo dividiremos en dos partes, por un lado, servidor web para controlar el tanque y la cámara y por otro lado el reconocimiento fácil.

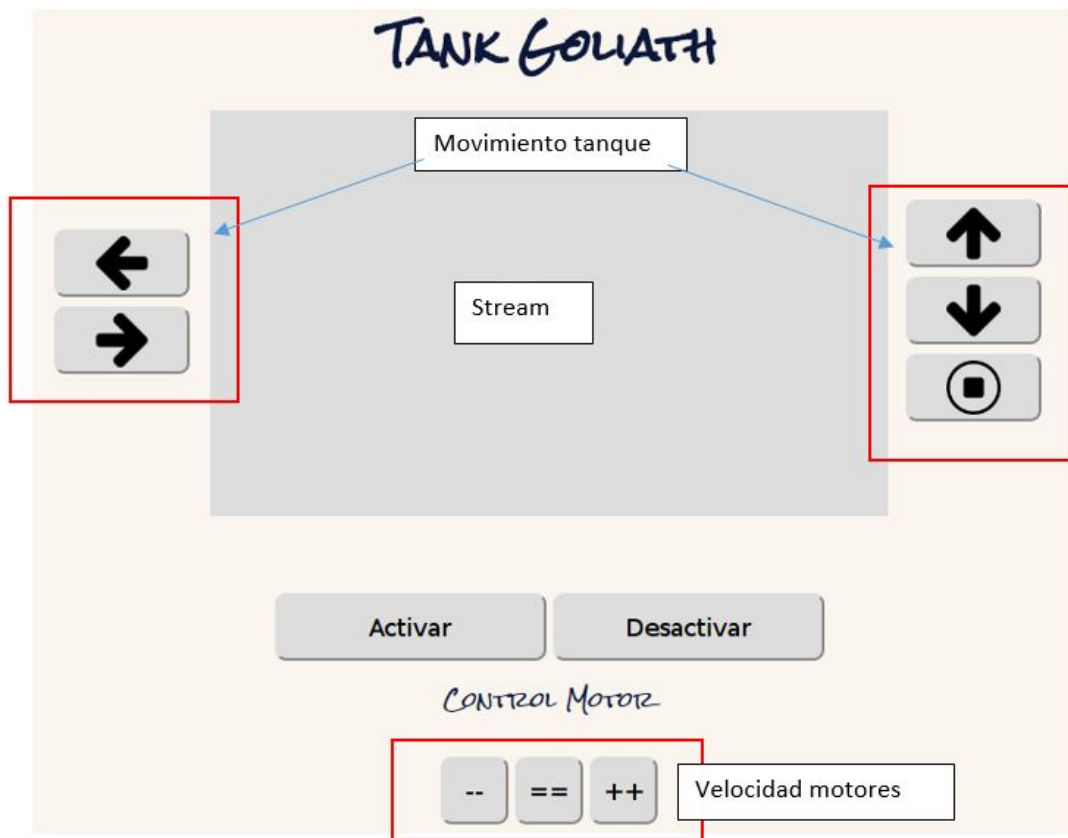
Trabajaremos sobre la carpeta `/var/www` donde está situado nuestro servidor.



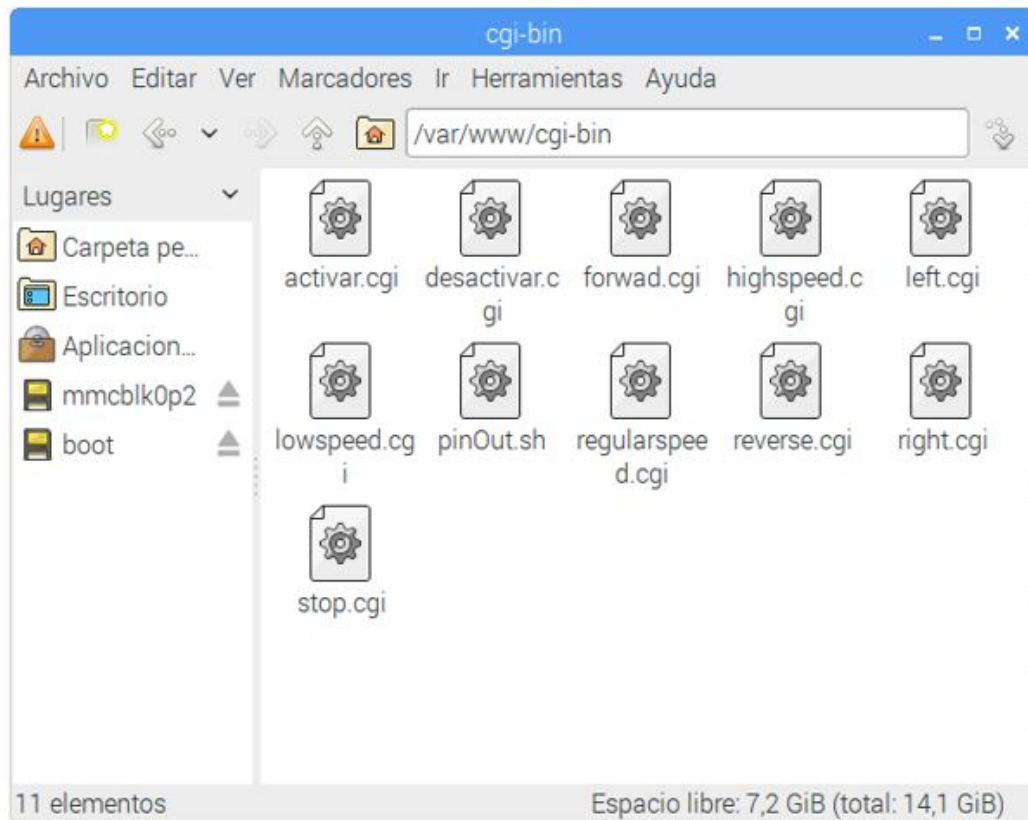
## 4.7 Servidor, movimiento y cámara

Antes de nada, diseñaremos una pequeña interfaz gráfica para poder controlar nuestro tanque desde cualquier dispositivo sin utilizar una conexión por escritorio remoto.

La interfaz será renombrada como index.html para que podamos acceder a ella directamente desde nuestro navegador al acceder a nuestro servidor.



Para realizar dichos controles haremos uso de la librería WiringPi y de los archivos .cgi que crearemos a continuación. Los archivos estarán situados en la carpeta cgi-bin.



Antes de crear los archivos debemos tener claro qué pines utilizaremos en nuestro caso utilizaremos los siguientes pines: 5,6,13,19,1.

Todos nuestros archivos serán iguales solo que alternando entre unos y ceros dependiendo del efecto que queremos, además todos ellos deben tener los siguientes permisos (755 o 777) los cambiaremos con el comando **chmod 777 index.html**.



Ahora pasaremos a trabajar con los pines, antes de nada, debemos poner los pines que vamos a utilizar en modo out, esto lo haremos gracias al comando gpio que nos proporciona wiringPi.

```

Archivo  Editar  Buscar  Opciones
#!/bin/sh
gpio -g mode 5 out
gpio -g mode 6 out
gpio -g mode 13 out
gpio -g mode 19 out
gpio -g mode 10 out

exit 0

```

Para explicar un archivo .cgi cogemos por ejemplo el siguiente, donde basta utilizar el comando write para escribir unos o ceros en los pines deseados.

```

Archivo  Editar  Buscar  Opciones  Ayuda
#!/bin/bash
gpio -g write 5 1
gpio -g write 6 0
gpio -g write 13 1
gpio -g write 19 0

```

Ejecutamos y vemos los cambios con el comando readall.

```

pi@raspberrypi: /var/www/cgi-bin
Archivo  Editar  Pestañas  Ayuda
pi@raspberrypi: /var/www/cgi-bin $ sudo bash ./forwad.cgi
pi@raspberrypi: /var/www/cgi-bin $ readall
bash: readall: no se encontró la orden
pi@raspberrypi: /var/www/cgi-bin $ gpio readall
+-----Pi 3-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | 8 | SDA.1 | ALTO | 1 | 3 | 4 | | | 5v | | |
| 3 | 9 | SCL.1 | IN | 1 | 5 | 6 | | | 5v | | |
| 4 | 7 | GPIO. 7 | IN | 0 | 7 | 8 | 1 | ALT5 | TxD | 15 | 14 |
| | | | | | | | | | | | |
| 17 | 0 | GPIO. 0 | IN | 0 | 11 | 12 | 0 | ALT5 | RxD | 16 | 15 |
| 27 | 2 | GPIO. 2 | IN | 0 | 13 | 14 | | | 0v | | |
| 22 | 3 | GPIO. 3 | IN | 0 | 15 | 16 | 0 | IN | GPIO. 4 | 4 | 23 |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| 10 | 12 | MOSI | OUT | 0 | 19 | 20 | 0 | IN | GPIO. 5 | 5 | 24 |
| 9 | 13 | MISO | ALTO | 0 | 21 | 22 | 0 | IN | 0v | | |
| 11 | 14 | SCLK | ALTO | 0 | 23 | 24 | 1 | OUT | GPIO. 6 | 6 | 25 |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| 0 | 30 | SDA.0 | IN | 1 | 27 | 28 | 1 | IN | CE0 | 10 | 8 |
| | | | | | | | | | | | |
| 5 | 21 | GPIO.21 | OUT | 1 | 29 | 30 | 0 | IN | CE1 | 11 | 7 |
| 6 | 22 | GPIO.22 | OUT | 0 | 31 | 32 | 0 | IN | SCL.0 | 31 | 1 |
| 13 | 23 | GPIO.23 | OUT | 1 | 33 | 34 | | | 0v | | |
| 19 | 24 | GPIO.24 | OUT | 0 | 35 | 36 | 0 | IN | GPIO.26 | 26 | 12 |
| 26 | 25 | GPIO.25 | IN | 0 | 37 | 38 | 0 | IN | 0v | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
pi@raspberrypi: /var/www/cgi-bin $

```

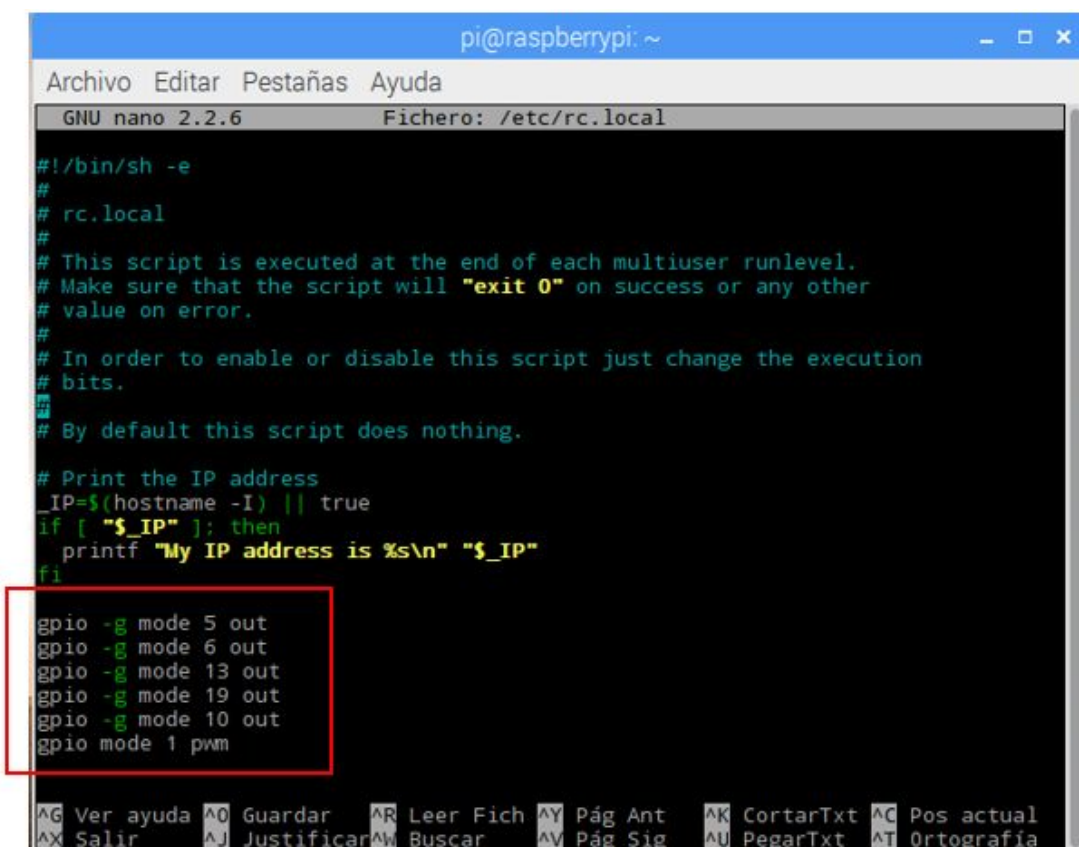
Para la velocidad de los motores será una configuración muy parecida, aunque en este caso utilizaremos el comando `pwm`, con el que nos permite controlar la potencia enviada a nuestro circuito.

```
Archivo  Editar  Buscar
#!/bin/bash

gpio pwm 1 1023
```

Estas configuraciones cada vez que reiniciamos la raspberry se eliminarán por lo que es mejor hacer que la configuración inicial de nuestros pines se use desde el inicio de la raspberry.

Para ello añadiremos lo siguiente al archivo **/etc/rc.local**.



```
pi@raspberrypi: ~
Archivo  Editar  Pestañas  Ayuda
GNU nano 2.2.6      Fichero: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

gpio -g mode 5 out
gpio -g mode 6 out
gpio -g mode 13 out
gpio -g mode 19 out
gpio -g mode 10 out
gpio mode 1 pwm

^G Ver ayuda  ^O Guardar    ^R Leer Fich  ^Y Pág Ant   ^K CortarTxt  ^C Pos actual
^X Salir      ^J Justificar ^W Buscar    ^V Pág Sig   ^U PegarTxt   ^T Ortografía
```

Para vincular nuestros scripts a la interfaz usaremos funciones de javascript y las llamaremos con la función **onclick()** de html.  
Para las funciones en javascript utilizaremos la función de **XMLHttpRequest()**, que nos permitirá interactuar con los archivos .cgi.

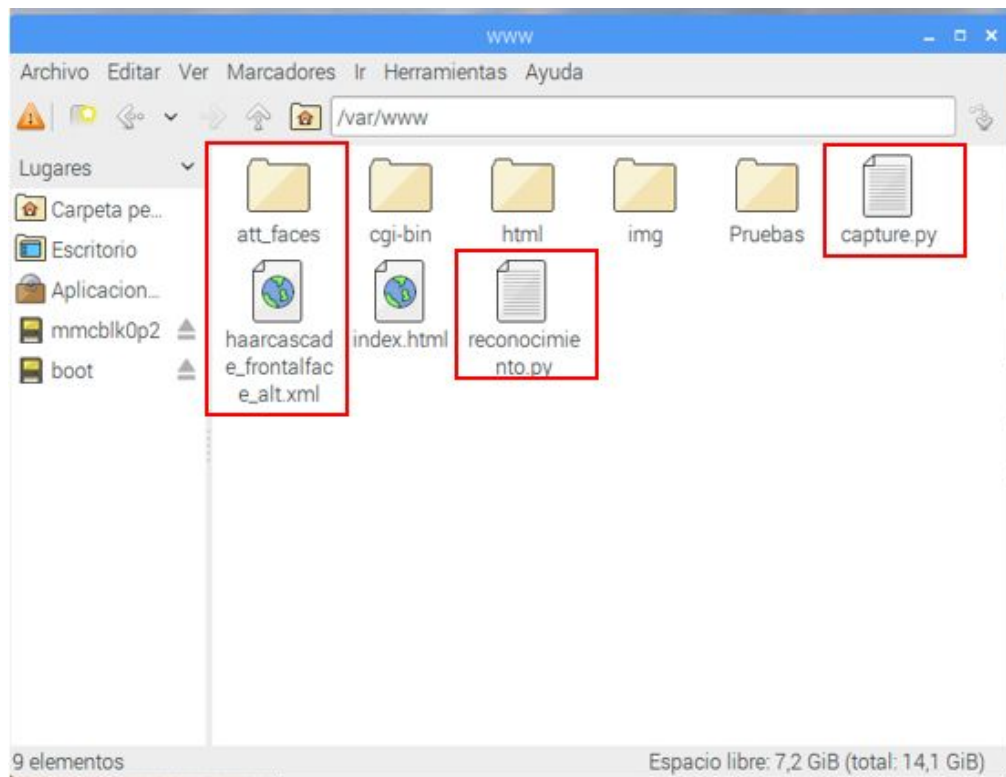
```
<script type="text/javascript">
    var xml;
    xml = new XMLHttpRequest();
    function left() {
        xml.open("GET","cgi-bin/left.cgi",true);
        xml.send();
    }
    function right() {
        xml.open("GET","cgi-bin/right.cgi",true);
        xml.send();
    }
    function down() {
        xml.open("GET","cgi-bin/reverse.cgi",true);
        xml.send();
    }
    function up() {
        xml.open("GET","cgi-bin/forwad.cgi",true);
        xml.send();
    }
    function stop() {
        xml.open("GET","cgi-bin/stop.cgi",true);
        xml.send();
    }
}
```

Y a través de la función **onclick()** como hemos dicho antes vinculamos dichas funciones a los botones.

```
<div id="botonesD" style="height: 65.0px;">
    <button style="height: 50.0px;width: 100.0px;" onclick="left()"><
    <button style="height: 50.0px;width: 100.0px;" onclick="right()">
<div id="botonesU">
    <button style="height: 50.0px;width: 100.0px;" onclick="up()"><
    <button style="height: 50.0px;width: 100.0px;" onclick="down()">
```

## 4.8 Reconocimiento Facial

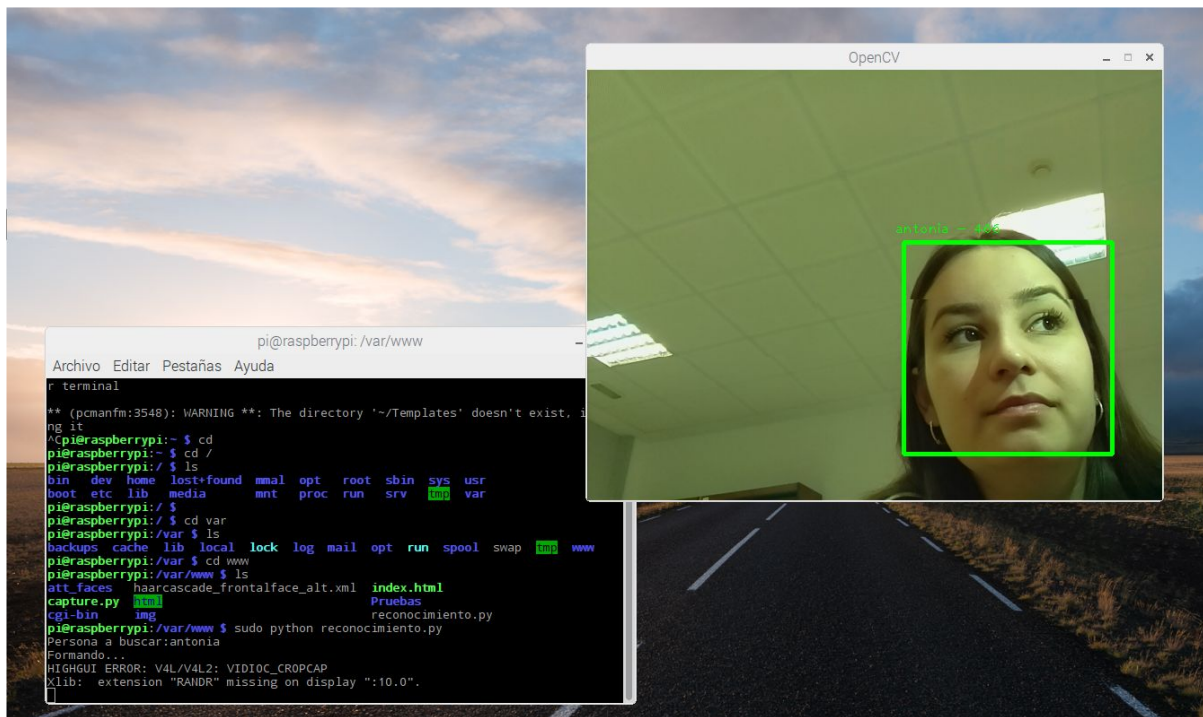
Para el reconocimiento facial utilizaremos openCV y la siguiente estructura donde tendremos una carpeta con las imágenes tomadas que serán nuestra base de datos la cual utilizaremos para a comparación de caras.



La carpeta `att_faces` será nuestra base de datos donde tendremos una carpeta por persona identificada, y esta tendrá un total de 50 fotos. Después tendremos el xml haarcascade que utilizaremos para la detección de la cara en el stream, dicho xml se puede encontrar en <http://docs.opencv.org/>.

Para el reconocimiento trabajaremos con dos scripts, `capture.py` y `reconocimiento.py`, con `capture.py` realizaremos un total de 50 fotos a la persona elegida para su identificación. Para ejecutar dicho script tendremos que ejecutar el siguiente comando: **Python capture.py nombrePersona**, con esto generará una carpeta en la base de datos con ese nombre y las 50 fotos obtenidas.

Ahora ejecutando el script `reconocimiento.py` con el comando, **Python reconocimiento.py**, podremos identificar las personas que estén en nuestra base de datos en caso de no estar en ella, mostrará un mensaje de desconocido.





## 5. Montaje

Una vez tenemos todos los materiales listos y las librerías instaladas podemos empezar a montar el tanque.

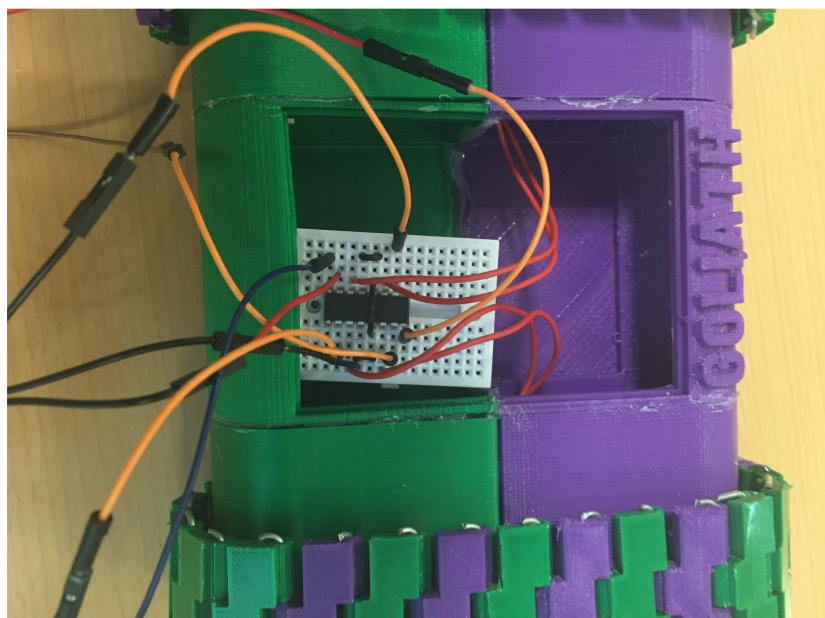
Para empezar procedemos a conectar el módulo de la cámara en la raspberry



Con la cámara conectada ya podremos realizar capturas o visualizar en streaming la visión de la cámara.

El siguiente paso será conectar los motores y la fuente de alimentación del tanque, en nuestro caso será una batería externa.

Para habilitar ahora el movimiento del tanque hacia delante y hacia atrás debemos conectar el puente H, que se explicará a continuación.

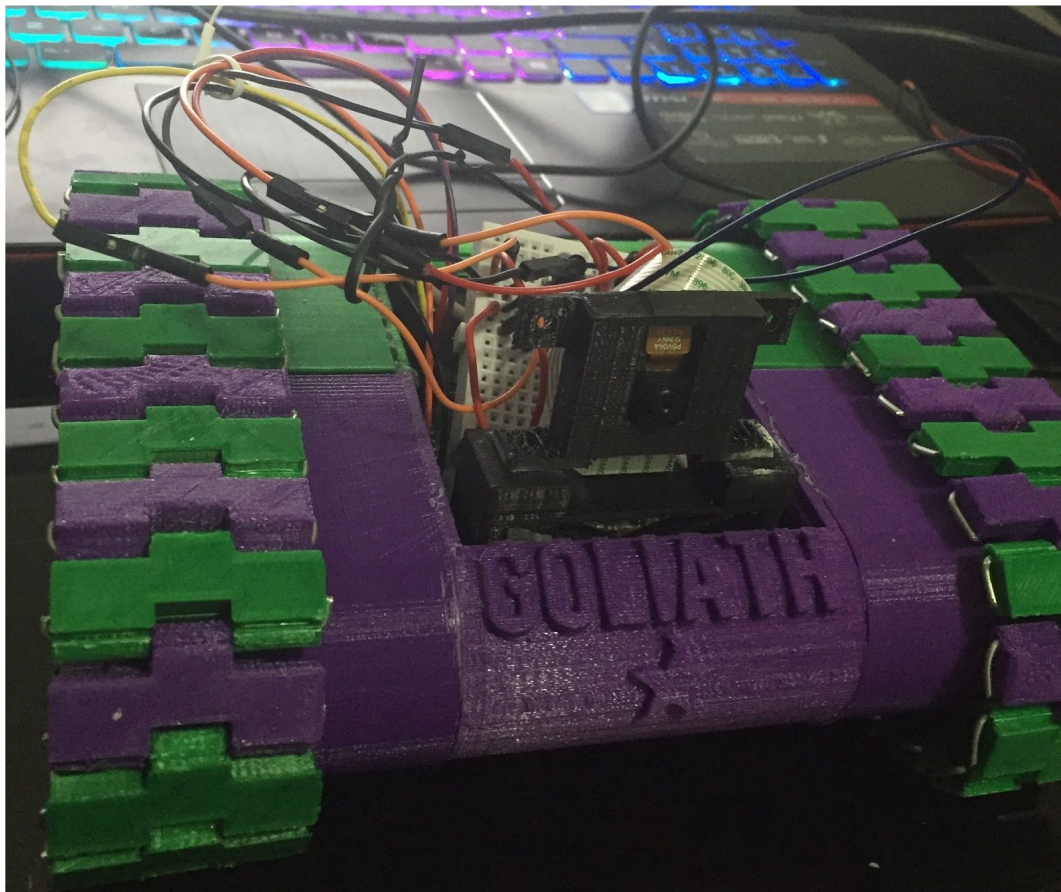




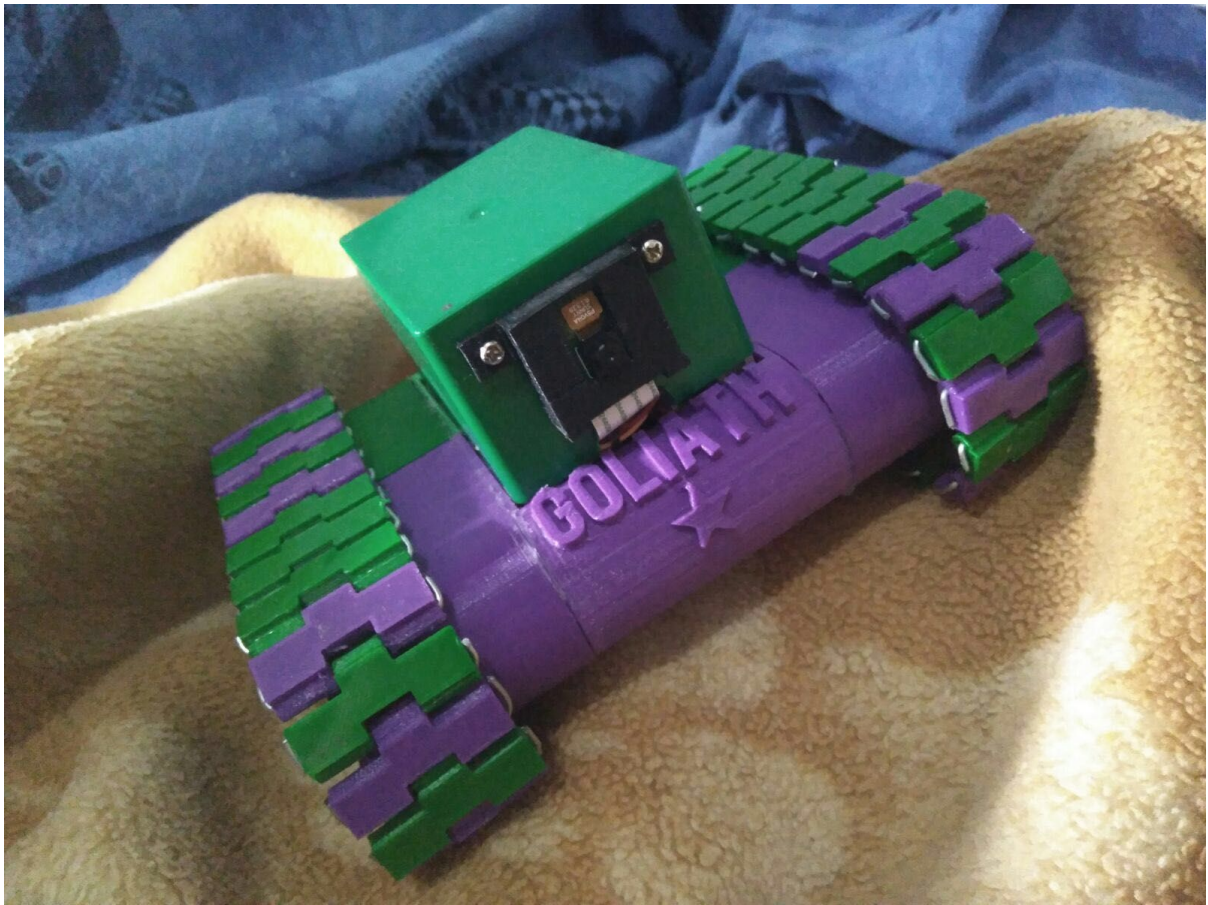
Ahora ya tendremos montado el esqueleto funcional del tanque, ahora procederemos a montar la parte externa. Lo primero de todo será colocar las ruedas.

Dado que nuestro tanque tendrá tracción (delantera/trasera) conectaremos las ruedas (delanteras/traseras) al eje de giro de los motores, y conectaremos las otras dos ruedas en la parte (delantera/trasera) del tanque.

Para transmitir la energía de las batería a las ruedas (delanteras/traseras) conectaremos las ruedas de cada lado del tanque con una cadena de transmisión.



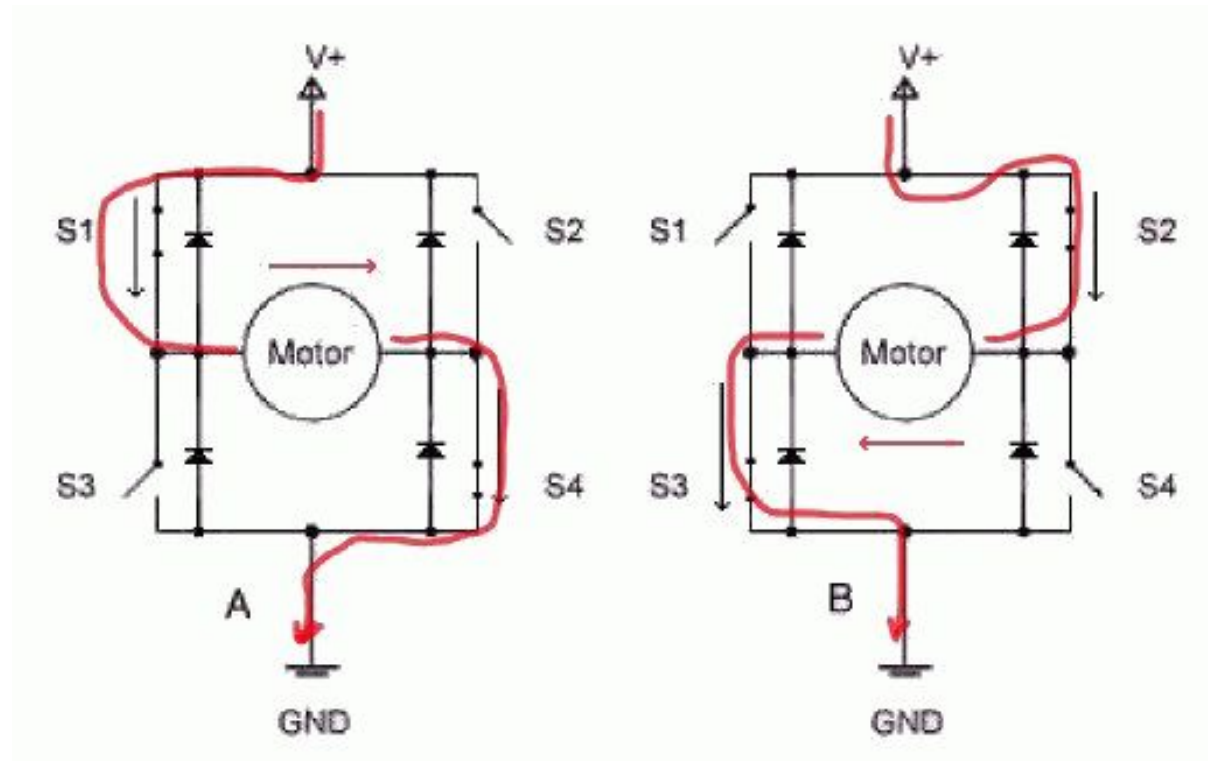
En este punto ya tendremos un tanque móvil pero deberemos cubrir la raspberry y las conexiones que permiten su funcionamiento, para ello colocaremos una carcasa protectora en la parte superior de la siguiente manera



## 5.1 Montaje Puente H

El puente h es un circuito muy común en electrónica que se encarga de permitir el giro de un motor en sus dos sentidos.

Debe su nombre a la similitud que tiene con la letra 'H', como vemos en la siguiente imagen.



Como podemos ver en la imagen anterior, si cerramos los switches 1 y 4 la corriente de la batería (9v) atraviesa el motor y lo hace girar en un sentido. Si por el contrario cerramos los switches 2 y 3 la corriente atraviesa el motor en el sentido contrario y por tanto lo hace girar en el otro sentido.

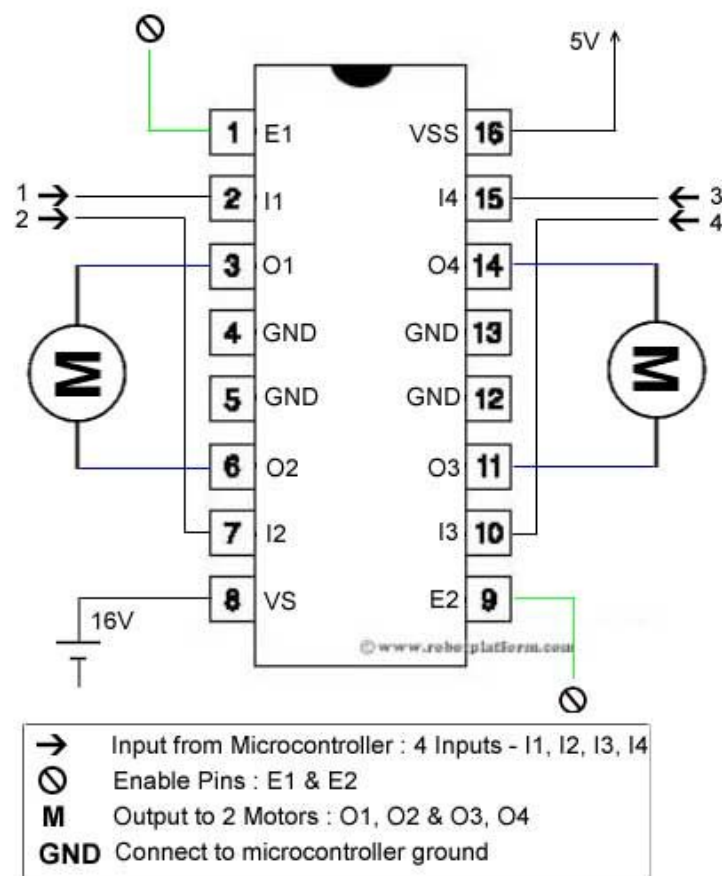
En la práctica los switches serán transistores, que se activarán aplicándoles una pequeña corriente en su base traída desde los pines de salida de nuestra raspberry pi (5v, no confundir con la corriente de alimentación del motor que es de 9v), es decir, si aplicamos una corriente a la base de los transistores 1 y 4 podremos hacer que la corriente colector- emisor del transistor circule entre los transistores 1 y 4 y atraviese el motor en un sentido, si dejamos de aplicar dicha corriente a la base de los transistores 1 y 4, el transistor se abre y no circulará corriente en la unión colector-emisor, de forma análoga para los otros transistores.

Dicho de otra forma, cuando aplicamos 5v a la base del transistor, hacemos que este permita la corriente de la batería a través de él, basta con activar los transistores justos para permitir que la corriente mueva el motor en uno u otro sentido.

Este circuito dispone de 4 diodos que se ponen en el circuito a modo de seguridad, debido a que el motor está (como todos los motores eléctricos) formado por una bobina que lo hace girar por el fenómeno denominado 'inducción electromagnética', sabemos que toda bobina en el instante exacto en el que se le aplica una corriente eléctrica (o se le quita) responde de forma

natural con un pulso de sentido contrario, ese pulso de respuesta natural de cualquier bobina de duración ínfima puede romper los transistores, es decir, en el instante en el que la corriente comienza a atravesar la bobina, esta se 'opone' al paso de dicha corriente con un pulso de corriente instantáneo en el sentido contrario, y lo mismo en el caso de la desconexión, así pues dichos diodos están colocados de forma que los pulsos que genera dicha bobina no puedan atravesar el transistor en el sentido contrario (emisor-colector), y lo hagan únicamente en el sentido correcto (el sentido colector-emisor).

En la práctica para simplificar nuestro proyecto usaremos un circuito integrado llamado D293D que incorpora el puente H en su interior. Este es su diagrama de pines.



<b>Pin Número</b>	<b>Abreviatura</b>	<b>Nombre</b>	<b>Conectado A</b>
1	E1	Enable 1	No conectado
2	I1	Input 1	Raspberry gpio X
3	O1	Output 1	Motor Izq
4	GND	Ground	Raspberry ground
5	GND	Ground	Raspberry gpio ground
6	O2	Output 2	Motor Izq
7	I2	Input 2	Raspberry gpio X
8	Vs	Voltaje Motor	Pila 9V
9	E2	Enable 2	No conectado
10	I3	Input 3	Raspberry gpio X
11	O3	Output 3	Motor der
12	GND	Ground	Raspberry ground
13	GND	Ground	Raspberry ground
14	O4	Output 4	Motor der
15	I4	Input 4	Raspberry gpio X
16	Vss	Voltaje circuito	Raspberry 5v



## 6. Anexo montaje impresora



### 6.1 Introducción

Para la realización del proyecto "Goliath" ha sido necesario realizar un subproyecto paralelo que consiste en la creación de una impresora 3D que nos permitiese fabricar las piezas necesarias.

El modelo elaborado de impresora, consiste en una versión mejorada de la conocida "Prusa I3", la cual se conoce como "P3steel". Esta varía del modelo con diferencia al modelo convencional, posee una estructura metálica. Dicha estructura mejora consistentemente las vibraciones en el momento de la impresión, a la vez que posee una mayor robustez y resistencia cualquier golpe.



## 6.2 Materiales

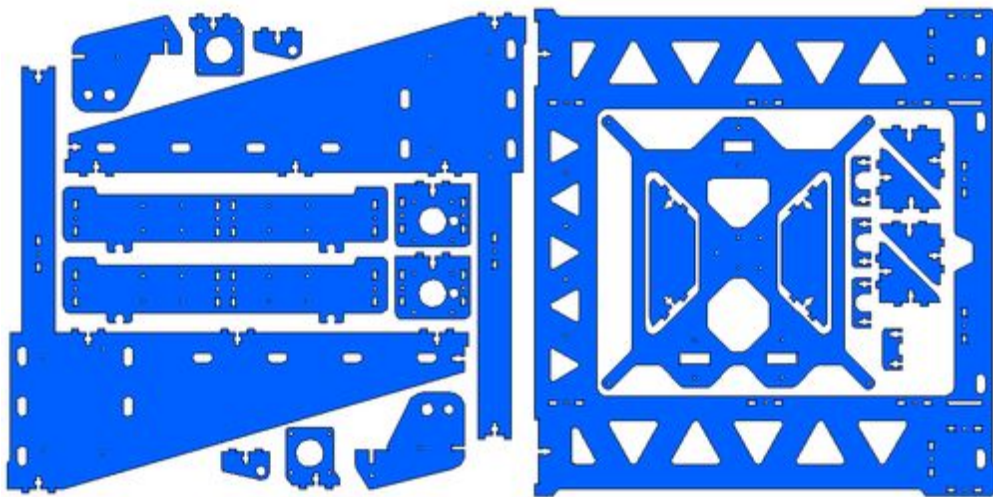
La lista de materiales utilizados para la impresora son los siguientes:

- Fuente alimentación PC 500w.
- Set de piezas impresas para "P3Steel".
- Kit de estructura "P3Steel".
- 5x Motores paso a paso modelo Nema-17.
- 3x Varillas de acero de 8mm.
- 1x Varilla roscada de 5 mm.
- Controlador Arduino Mega.
- Shield Ramps 1.4.
- 5x Driver drv8825.
- 3x Final de carrera.
- 2x Poleas GT2.
- Correa GT2 1m.
- Hotend v6.
- Kit de tornillería para "P3Steel".
- 11x Rodamiento Lineal 8mm.
- Tablero de madera.
- Panel de vidrio de 22x22cm.

## 6.3 Montaje

Este proceso es bastante simple aunque un poco tedioso. El diseño de las piezas de la estructura viene preparado para que sea bastante simple de montar, por lo que no se requiere de ningún tipo de manual.

En la siguiente imagen podemos ver el plano de las piezas, para obtener las piezas habría que ir a una carpintería metálica que realicen corte laser o hidráulico. En nuestro caso optamos por comprar un kit.



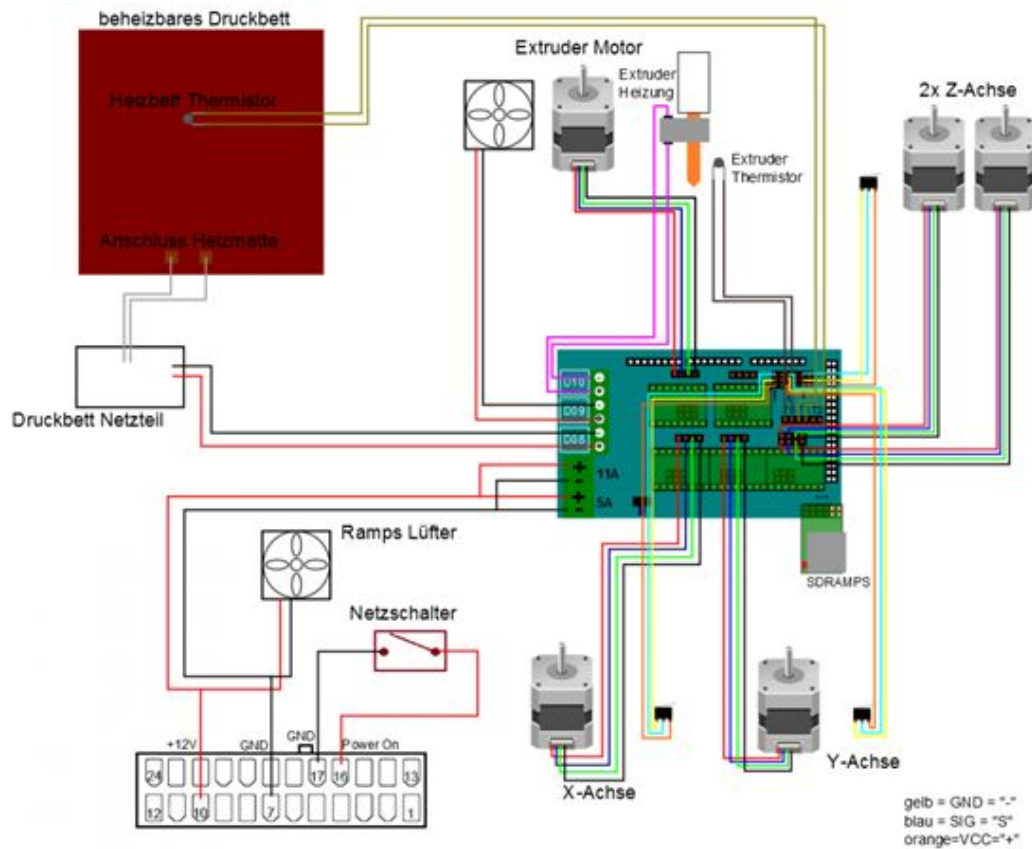
Una vez realizado el montaje, obtendremos una estructura similar a la siguiente.



Una vez tenemos la estructura montada, procedemos al montaje del resto de piezas mecánicas. En el que debemos obtener algo similar a la siguiente imagen.



Terminado el montaje de lo que denominamos parte mecánica, toca conectar toda la electrónica para que cumpla su función.  
Para el montaje de esta parte, habrá que seguir el siguiente esquema.



En este paso debemos prestar atención a los colores de los cables de los motores, ya que en función de cómo los conectemos giraran en un sentido u otro.

## 6.4 Programación y calibración

En este paso se explica cómo hacer que la impresora “cobre vida” mediante la programación del módulo arduino y su calibración para que imprima con la exactitud que le indiquemos.

Comenzamos descargando “Marlin”, que es un firmware Open Source muy común en el mundo de la impresión 3d. Lo podemos descargar desde aquí <https://github.com/MarlinFirmware/Marlin>.

Una vez tenemos descargado el archivo, abriremos con el entorno de desarrollo de arduino un archivo llamado marlin.ino. A abrir este archivo se abrirán un montón de pestañas, entre las cuales tendremos que buscar “configuration.h”.

El primer paso es seleccionar que placa de desarrollo estamos utilizando, en nuestro caso es la 33, ya que es una ramps 1.4 y el número de extrusores, que también es 1



También tendremos que configurar la temperatura máxima y del “hotend” y la cama caliente.



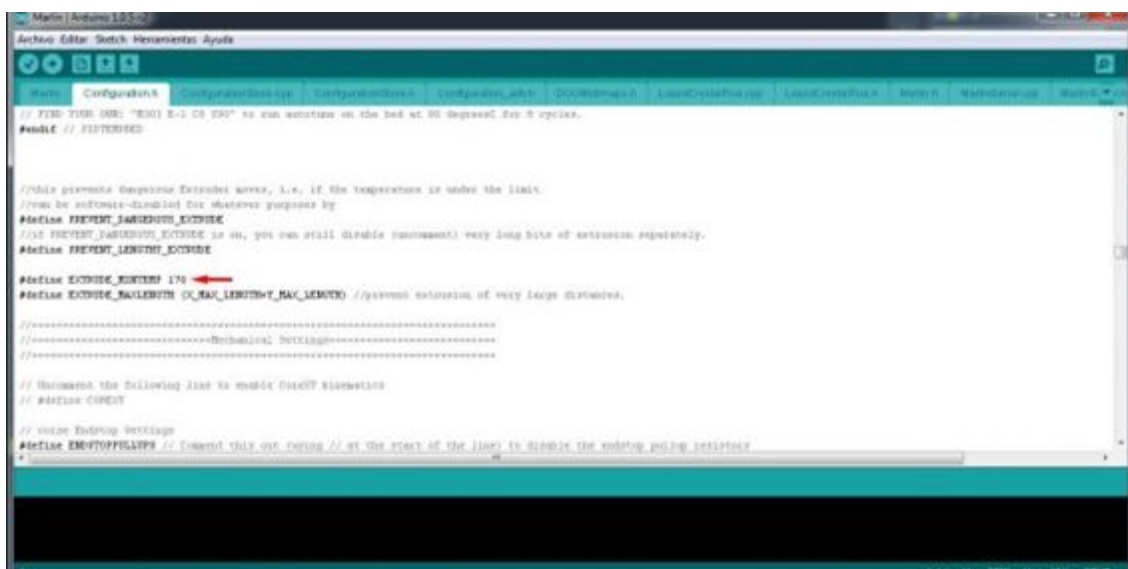
```
// When temperature sensors are temp, your heater will be switched off.
// This feature exists to protect your hotend from overheating accidentally, but *NOT* from thermistor short/circuits!
// You should use MINTEMP for thermistor short/circuit protection.
#define HEATER_0_MAXTEMP 300
#define HEATER_1_MAXTEMP 275
#define HEATER_2_MAXTEMP 275
#define BED_MAXTEMP 130

// If your bed has low resistance e.g. .6 ohm and throws the fuse you can duty cycle it to reduce the
// average current. The value should be an integer and the heat bed will be turned on for 1 interval of
// HEATER_BED_DUTY_CYCLE_DIVIDER intervals.
// #define HEATER_BED_DUTY_CYCLE_DIVIDER 5

// PID settings
// Uncomment the following line to disable PID and enable bang-bang.
#define BANG_BANG
#define BANG_MAX 255 // limits current to 255 while in bang-bang mode; 255=full current
#define BANG_MIN 255 // limits current to 255 while PID is active (see PID_FUNCTIONAL_RANGE below); 255=full current
// #define BANG_BANG

// #define PID_PERIODIC // Sends output data to the serial port.
// #define PID_PERIODIC 1 // Puts PID in open loop. BANG_BANG sets the output power from 0 to PID_MAX
// #define PID_FUNCTIONAL_RANGE 10 // If the temperature difference between the target temperature and the actual temperature
```

Además, como medida de seguridad, tendremos que poner una temperatura mínima para poder extruir plástico. Según la información encontrada por la red lo recomendable es poner 170



```
// PID: FIRM: "MKS 2.1.0 0.0" to run extruder on the bed at 90 degrees for 5 cycles.
// #define PID_PERIODIC

// This prevents dangerous Extruder moves, i.e. if the temperature is under the limit.
// You can be safe and disable for whatever purpose by
// #define PREVENT_DANGEROUS_EXTRUDE
// If PREVENT_DANGEROUS_EXTRUDE is on, you can still disable movement very long time of extrusion segments.
// #define PREVENT_LENGTHY_EXTRUDE

// #define EXTRUDE_MINTEMP 170
// #define EXTRUDE_MAXLENGTH (X_MAX_LENGTH+Y_MAX_LENGTH) // prevent extrusion of very large distances.

// ===== Mechanical Settings =====
// =====

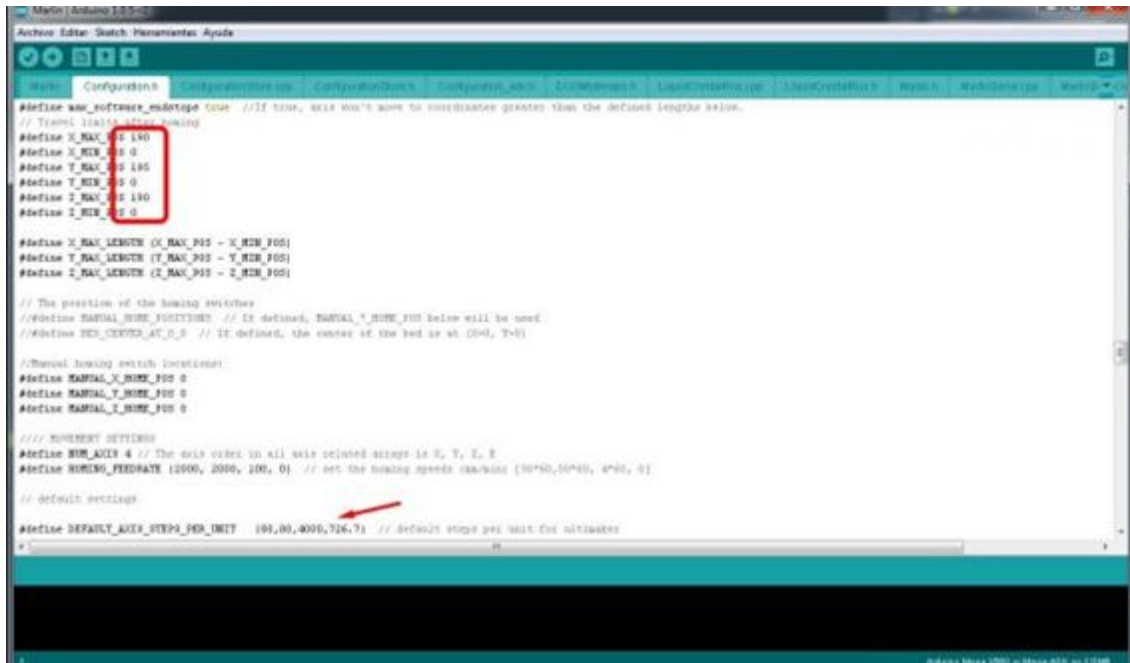
// Uncomment the following line to enable safety shutoff
// #define SAFETY

// ===== Endstop Settings =====
// #define ENDSTOPPULLUPS // Comment this out (uncomment if at the start of the line) to disable the endstop pullup resistors
```



A continuación se explica uno de los pasos más importantes, la superficie de impresión y los pasos por milímetro que dará cada motor, es decir, cuántos pasos tiene que dar para que cuando le ordenemos que se mueva 1cm, se mueva exactamente 1 cm.

Vemos que hay 4 números, equivalen respectivamente al eje X, Y, Z y al extrusor.



```
#define MAX Software endstop time // If time, this won't move to coordinates greater than the defined lengths below.
// Time to limit the endstop
#define X_MAX_POS 150
#define X_MIN_POS 0
#define Y_MAX_POS 150
#define Y_MIN_POS 0
#define Z_MAX_POS 150
#define Z_MIN_POS 0

#define X_MAX_LENGTH (X_MAX_POS - X_MIN_POS)
#define Y_MAX_LENGTH (Y_MAX_POS - Y_MIN_POS)
#define Z_MAX_LENGTH (Z_MAX_POS - Z_MIN_POS)

// The position of the homing switches
// Define MANUAL_HOME_POSITIONS // If defined, MANUAL_*_HOME_POS below will be used
// Define BED_CENTER_AT_0_0 // If defined, the center of the bed is at (0,0), (X=0, Y=0)

// Manual homing switch locations:
#define MANUAL_X_HOME_POS 0
#define MANUAL_Y_HOME_POS 0
#define MANUAL_Z_HOME_POS 0

//===== SETTINGS =====
#define MIN_AXIS 4 // The only value in all sets should be X, Y, Z, E
#define HOMING_FEEDRATE (2000, 2000, 100, 0) // set the homing speeds (X*Y*Z*E) (30*30, 30*30, 4*30, 0)

// default settings:
#define DEFAULT_AXIS_STEPS_PER_UNIT {100,100,4000,126.7} // default steps per unit for ultimaker
```

Para obtener una calibración perfecta, nos hemos basado en ciertos números que hemos encontrado en la web dependiendo de nuestros componentes, pero esto solo nos da una estimación aproximada de por dónde empezar.

Para conseguir la precisión, hay que hacer pruebas de desplazamiento y de impresión, hasta dar con el número exacto.

Una vez realizado todo esto, hemos obtenido una impresora 3D perfectamente funcional

## 7. Enlace proyecto GitHub

<https://github.com/vhar95/ProyectoTank>

## 8. Bibliografía

<http://opencv.org/>

<https://www.lighttpd.net/>

<https://github.com/jacksonliam/mjpg-streamer>

<http://www.xrdp.org/>

<https://www.python.org/>

<http://wiringpi.com/>