

UNIVERSIDAD AUTÓNOMA DE MADRID

Escuela Politécnica Superior



Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

REALIDAD VIRTUAL PARA PERSONAS QUE TIENE
DISCAPACIDAD MOTORA

Cristina Kasner Tourné

Tutor: Francisco de Borja Ortiz

Junio 2016

REALIDAD VIRTUAL PARA PERSONAS QUE TIENE DISCAPACIDAD MOTORA

Autor: Cristina Kasner Tourné
Tutor: Francisco de Borja Ortiz

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2016

RESUMEN

Resumen La fusión de la Realidad Virtual con la robótica supone una apertura a infinitas posibilidades.

Ambas tecnologías están en continuo desarrollo, de hecho la realidad virtual empezó a darse a conocer muy recientemente, a pesar de que su nacimiento data sobre (año).

En los últimos años estamos viendo cómo la Realidad Virtual va haciéndose un hueco en las tecnologías que usamos habitualmente. Quizá el uso más comercial que se le está dando es en lo referente al mundo de los videojuegos. Sin embargo se puede aplicar a muchos otros campos , en concreto en el campo de la salud , donde se están obteniendo muy buenos resultados (en cirugía, rehabilitación, etc).

Este trabajo pretende explorar el uso de la Realidad Virtual y la robótica como herramienta enfocada a personas con movilidad reducida , ayudarles a ser más independientes y dándoles la posibilidad de transportarse de forma virtual por entornos reales.

Para conseguir este objetivo se ha construido un sistema que consta de: un robot y las Oculus Rift (gafas de realidad virtual). El robot se controla desde las Oculus Rift, de forma que si el usuario lleva puestas las gafas, podrá controlar el robot con movimientos de cabeza. El usuario verá a través de las gafas todo lo que vea el robot, actuando éste último como extensión de la vista del usuario.

Al utilizar las Oculus Rift dejamos abierta la posibilidad de ver, no solo el entorno en el que se mueve el robot, sino un entorno virtual creado por el propio usuario. Esta parte se deja como tema de estudio para trabajos futuros.

Además hemos diseñado el proyecto de forma que la conexión entre el robot y las gafas sea a través de red inalámbrica, lo que le da al robot una libertad de movimiento fundamental para el objetivo que se persigue.

A lo largo del desarrollo del trabajo han surgido varias complicaciones, la gran mayoría referentes a la utilización de los sensores de las Oculus y de la librería

de las mismas, que aún no está perfectamente adaptada a la versión DK2 de las gafas.

Esto se ha ido solucionando calibrando el movimiento de los servo motores mediante pruebas experimentales.

A pesar de esto el proyecto se ha terminado con éxito, dejando abierta una línea de investigación y mejora sobre la que trabajar.

Palabras clave Driver de red, Altas prestaciones, Captura de tráfico 40Gbps

ABSTRACT

Abstract

Keywords Network drivers, High Performance, 40 Gbps traffic capture

ÍNDICE GENERAL

Índice general	IV
Índice de tablas	VI
Índice de figuras	VII
1 Introducción	1
1.1 Motivación del proyecto	1
1.2 Objetivos	2
1.3 Estructura del documento	3
2 Estado del arte	5
2.1 Introducción	5
2.2 Realidad virtual y salud	6
2.3 Robótica y realidad virtual.	6
3 Análisis, diseño y desarrollo	7
3.1 Análisis	7
3.1.1 Requisitos Funcionales	8
Construcción del Robot	8
Streaming	8
Control del Robot	9
3.1.2 Requisitos no funcionales	9
3.2 Diseño	10
3.2.1 Diseño del robot	10
Raspberry Pi 3 model B	10
Servo motores	10
Cámara	12
3.2.2 Router	12
3.2.3 Oculus Rift	13
Detección de la orientación	13
3.2.4 Diseño de conexiones	16
3.3 Desarrollo e implementación	18
3.3.1 Construcción del Robot	18
Soporte de la cámara	18

3.3.2	Streaming	20
	Raspberry Pi3	20
	Ordenador	21
3.3.3	Control del Robot	22
	Control de los motores de las ruedas del robot	26
	Control del motor de la cámara del robot	26
	Instrucciones de mando	27
4	Pruebas y Resultados	29
4.1	Latencia de recepción de comandos	29
4.2	Feedback de los usuarios	30
5	Conclusiones y trabajo futuro	33
5.1	Conclusiones	33
	5.1.1 Tecnologías aprendidas	34
5.2	Trabajo futuro	34
	Bibliografía	35

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

3.1	Raspberry Pi 3 model B	11
3.2	Esquema conexiones servos , cámara, Raspberry Pi 3	12
3.3	Tabla DHCP del router con las parejas MAC-IP fijas	13
3.4	Archivo de configuración de red Raspberry Pi 3	13
3.5	Ejes de giro	14
3.6	Corrección del tilt error	16
3.7	Esquema conexiones	17
3.8	Diseño 3D de la pinza de agarre en Blender	19
3.9	Imagen de la configuración del Cura para enviar la Pieza de agarre a la impresora 3D	19
3.10	Uso de CPU del programa Mjpeg-Streamer por el que se hace el streaming de video	20
3.11	Ejecución del comando para iniciar el streaming	21
3.12	Página html donde se recibe el streaming a tiempo real desde la Raspberry	21
3.13	Configuración y resultado de la reproducción de video en VLC	22
3.14	Posición de las Oculus respecto al eje X una vez procesados los datos de HeadPose.ThePose.Orientation	24
3.15	Posibles movimientos del robot según el movimiento de las Oculus Rift	27
4.1	Diferencia de tiempos entre envío y recepción de comandos a 7,50 metros	30
4.2	Diferencia de tiempos entre envío y recepción de comandos a 7,50 metros	30

INTRODUCCIÓN

1.1 Motivación del proyecto

La motivación de este proyecto es explorar las posibilidades que nos ofrece combinar la robótica con la realidad virtual.

El campo de la robótica lleva muchos años en desarrollo y mejora exponencialmente a lo largo del tiempo. Está inmerso en nuestro día a día y no paramos de sorprendernos con nuevos avances como las impresoras 3D o los drones.

La realidad virtual es algo que, aunque lleva mucho tiempo estudiándose (En 1968 Ivan Sutherlan creó el primer casco de realidad virtual), hasta hace bien poco, para la gran mayoría de gente era solo ciencia ficción.

Ahora nos encontramos en un momento en el que ya podemos empezar a disfrutar de la realidad virtual. Tenemos las CardBoard , las Oculus Rift... a nuestro alcance. El uso más generalizado que se está dando a estas herramientas es en el mundo de los videojuegos.

Este trabajo pretende aplicar estas dos tecnologías para hacer más fácil el día a día de las personas, en concreto nos hemos centrado en las personas con discapacidad motora.

Hemos construido un robot que lleva incorporada una cámara y que se comunica con las oculus rift.

El usuario puede controlar el movimiento del robot y de la cámara solo con el movimiento de la cabeza, aunque no sería difícil adaptar este control para personas que les sea más fácil controlarlo a través de mandos.

El robot transmite a las oculus en tiempo real un video del entorno. Al ser las oculus gafas de realidad virtual, esto nos permite modificar el entorno según las necesidades del usuario . P.ej: podríamos controlar una casa domotizada.

De esta forma damos al usuario más independencia y le abrimos un mundo al que hasta ahora, tenía difícil acceso.

1.2 Objetivos

El objetivo de este proyecto es investigar las oportunidades que nos ofrece juntar la robótica y la realidad virtual para facilitar la vida de personas con discapacidad motora.

Para ello se quiere construir un robot que se pueda controlar a través de la Oculus Rift, las gafas de realidad virtual, tan solo con el movimiento de cabeza del usuario.

El robot por su parte transmitirá a tiempo real un vídeo del entorno, que el usuario verá en las Oculus Rift.

Con este propósito hemos dividido el trabajo en pequeños objetivos:

- Construir un robot con una cámara integrada y con capacidad de movimiento en todas las direcciones.
- Conseguir streaming a tiempo real desde el robot hasta las Oculus Rift, con la mínima latencia posible.
- Aplicar al video el formato SBS (Side By Side) para poder verlo correctamente desde las Oculus Rift y así dar al usuario mayor sensación de inmersión.
- Sincronizar el movimiento del robot con el movimiento de las Oculus Rift.
Para ello debemos:
 - Capturar la señal de los sensores de posición de las Oculus Rift (giroscopio, magnetómetro, acelerómetro)
 - Enviar con la mínima latencia posible la información de posición y orientación de las Oculus Rift al robot.
 - Traducir la información de posición y orientación de las Oculus a comandos para enviarlos a los motores del robot.

1.3 Estructura del documento

El objetivo de este documento es explicar el trabajo realizado y los resultados obtenidos.

En el siguiente capítulo se explica el estado del arte, hablando brevemente de la realidad virtual y de algunos proyectos previos que utilizaban esta tecnología. Como se verá más adelante el capítulo se centra especialmente en proyectos relativos a salud y robótica.

Luego se verá el análisis, diseño y desarrollo del proyecto.

En el análisis se presentan los requisitos que debe cumplir nuestro sistema. Posteriormente se hace un estudio de diseño de los componentes del trabajo (robot, conexiones entre dispositivos...) y por último se explicará como se ha desarrollado el proyecto.

Los resultados y las pruebas realizadas para ver si el proyecto cumplía con el objetivo se verán en el capítulo "Pruebas y resultados" de este documento.

Para finalizar expondremos brevemente las conclusiones y el trabajo futuro que se puede realizar en base a este proyecto.

ESTADO DEL ARTE

2.1 Introducción

La realidad virtual es un concepto que empezó siendo ciencia ficción y poco a poco está ganando terreno en la vida cotidiana, ya sea en el mundo de los videojuegos, la medicina o la aviación, entre otros campos.

Uno de los acontecimientos con los que empezó la realidad virtual es la creación de "The Sword of Damocles", un casco de realidad virtual (HMD- head-mounted display) creado por Ivan Sutherland en 1968. Ivan Sutherland considera la realidad virtual como una herramienta para familiarizarnos con una realidad que, hasta ahora no estaba a nuestro alcance.

Al mismo tiempo Thomas A. Furness III empezó a introducir herramientas de realidad virtual en entrenamientos de vuelo, de forma que estos fueran mucho más seguros para los pilotos. Hoy en día se sigue utilizando y mejorando esta tecnología ya que, no solo es más segura, sino que además es más barata.

Desde entonces , y con el desarrollo exponencial que ha vivido la tecnología las últimas décadas, la realidad virtual ha ido haciendo un hueco en la industria tecnológica. La cara más visible de la realidad virtual es el uso que se le da en los videojuegos. ¿cómo se consigue la sensación de inmersión que dan las gafas de realidad virtual?

Hay dos puntos clave para conseguirlo:

1. Visión 3D
2. Visión 360°

La técnica utilizada para la visión 3D fue descubierta cerca de 1840, consiste en separar la imagen en dos, una para el ojo derecho y otra para el izquierdo, dando a cada imagen una inclinación distinta, para simular como ven nuestros ojos y así hacer que el cerebro lo interprete tal y como interpreta el mundo real que vemos.

Otro sector muy interesante donde se aplican técnicas de Realidad Virtual es en el campo de la salud, hacia el que está enfocado este trabajo.

2.2 Realidad virtual y salud

El uso de realidad virtual está siendo muy útil en medicina, ya que permite estudiar comportamientos en diferentes situaciones de forma bastante realista.

Un ejemplo de esto es la evaluación del deterioro cognitivo en personas que sufren esclerosis múltiple. Para dicha evaluación se requiere una gran cantidad de datos sobre la velocidad de procesamiento de la información, atención, etc... Esto se conseguía haciendo múltiples test que no siempre eran fieles a la realidad.

Ahora se pueden crear entornos virtuales y estudiar el comportamiento de los usuarios en diversas situaciones de forma que el estudio es más realista y eficiente. [ref]

También se está estudiando el uso de programas de rehabilitación para que los pacientes no necesiten trasladarse hasta el hospital.

Uno de los estudios se hizo a personas con hemiparesia, con un software que mostraba por pantalla el movimiento de los pies y les mandaba ejercicios para mejorar su capacidad motora. Los resultados de este estudio mostraron que la herramienta de realidad virtual era efectiva para la rehabilitación y muy útil para los pacientes con dificultades para trasladarse hasta el hospital[ref]

2.3 Robótica y realidad virtual.

La unión de robótica y realidad virtual es algo que se lleva haciendo desde hace bastante tiempo.

En el campo de la salud una de las aplicaciones son entrenamientos de cirugía. La realidad virtual permite crear un entorno semejante al quirófano. El estudiante hace uso de herramientas que simulan los utensilios quirúrgicos y puede hacer una "operación virtual" sin riesgo para ningún paciente y sin necesidad de gastar recursos o un equipo quirúrgico de apoyo . [ref]

En educación hay varios programas de realidad virtual que hacen el aprendizaje más sencillo y completo a los estudiantes. Un ejemplo de esto es la visualización de brazos robóticos para diseñar correctamente el movimiento. Es relativamente complejo traducir cada movimiento de las articulaciones del brazo robótico en la posición final de éste. Si estas pruebas se hicieran directamente sobre el robot, sería muy sencillo romperlo. Con los programas de realidad virtual se simulan todos los movimientos sin peligro de malgastar recursos. [ref]

Otra aplicación es el control remoto de robots, tanto con software que crea una interfaz virtual para que el usuario obtenga de forma clara toda la información del robot (batería, almacenamiento de datos, posición...) [ref NASA xD], como interfaces BCIs.

ANÁLISIS, DISEÑO Y DESARROLLO

En este capítulo se explicará cómo se ha enfocado el desarrollo del proyecto.

La primera parte trata del análisis del problema. Se explican las diferentes partes en las que he dividido el trabajo para su posterior implementación. También se presenta de forma muy general qué requisitos básicos debe cumplir cada parte.

La sección de diseño explica las herramientas que se han utilizado para implementar cada parte y cómo funcionan.

Por último se explica de forma más detallada cómo se ha desarrollado cada parte del problema, qué dificultades han ido apareciendo y cómo se han solucionado. También explica cómo se ha realizado la integración para conseguir el proyecto final.

3.1 Análisis

El objetivo de este trabajo es conseguir un robot con una cámara que transmita video a tiempo real. Este video le llegará a unas gafas de realidad virtual. El robot se controlará con el movimiento de cabeza del usuario.

DIBUJO CHULI DE TODO JUNTO

El proyecto por lo tanto se divide en 3 problemas:

1. Construcción del robot → para ello necesitaremos una placa base, una cámara y la estructura que le permita desplazarse.
2. Streaming → Debemos conseguir transmitir a tiempo real todo lo que ve el robot a la Oculus
3. Control del robot → Queremos que el robot sea una extensión de los sentidos del usuario, por lo que debemos idear movimientos intuitivos para que el usuario mande comandos de movimiento al robot.

Vamos a desarrollar cada uno de ellos, identificando qué funcionalidades básicas debe cumplir.

3.1.1 Requisitos Funcionales

Construcción del Robot

El robot debe ser un dispositivo fácil de controlar , que a su vez actúe como una extensión de los ojos del usuario, dándole a éste la sensación de inmersión en un espacio real o virtual.

Por tanto los requisitos que debe cumplir el robot son los siguientes:

- **RF-CR1**

Permitir grabar video y transmitirlo a tiempo real.

- **RF-CR2**

Ser capaz de desplazarse en todas las direcciones

- **RF-CR3**

Tener la capacidad de recibir comandos de control a través de una red inalámbrica

- **RF-CR4**

Ser capaz de que la cámara apunte en la misma dirección que la cabeza del usuario

DIBUJO AMPLIADO ROBOT

Streaming

Se quiere conseguir la máxima sensación de inmersión para la persona que controla el robot.

Para ello los requisitos principales que debe cumplir son:

- **RF-S1**

La transmisión de video debe ser a tiempo real, con la mínima latencia posible.

- **RF-S2**

El usuario verá el video con las Oculus Rift, por lo que la imagen debe estar desdoblada (formato SBS, que se explicará en la sección de desarrollo).

- **RF-S3**

La transmisión debe ser inalámbrica ya que queremos que el robot tenga total libertad de movimiento, siendo éste independiente del ordenador.

Dado que la placa que utilizamos en el el robot es la Raspberry Pi 3 tenemos dos opciones para la transmisión inalámbrica: Bluetooth o red Wifi.

Control del Robot

Como hemos dicho anteriormente, el robot pretende ser una extensión de los ojos de usuario, por lo que es lógico que se controle con el movimiento de cabeza de la persona que esté recibiendo el video.

Dentro del control del robot vamos a diferenciar entre control del movimiento del robot y control de la dirección de la cámara:

- Movimiento del robot :

RF-CTRL1

Debemos diseñar un sistema intuitivo para mover el robot en todas las direcciones con movimientos de la cabeza.

- Movimiento de la cámara:

RF-CTRL2

La cámara debe estar apuntando siempre en la misma dirección que la cabeza.

Si el usuario mueve la cabeza hacia la derecha o hacia la izquierda, el robot se moverá en esa dirección, por lo que no hay que preocuparse por la cámara.

Para controlar los movimientos verticales vamos a incorporar un servo unido a la cámara, de forma que ésta se mueva en función de la posición de la Oculus.

3.1.2 Requisitos no funcionales

- **RNF1 - Usabilidad**

El sistema para controlar el movimiento del robot y de la cámara debe ser intuitivo, sin requerir movimientos de cabeza inusuales por parte del usuario.

- **RNF2 - Documentación**

El usuario debe disponer de un manual de uso sencillo para poder utilizar correctamente el sistema .

3.2 Diseño

Tras analizar las diferentes funcionalidades del trabajo vamos a ver cómo está diseñado cada componente.

Los principales dispositivos que componen el proyecto son

- Robot
- Router
- Oculus Rift

También se tratará el **diseño de conexiones** entre los 3 componentes.

3.2.1 Diseño del robot

El robot que se utiliza en este proyecto está basado en un diseño anterior hecho por (Nombre del alumno que diseñó el robot)

Consta de:

- Una placa base → Raspberry Pi 3
- Dos ruedas conectadas con dos servos que permiten al robot desplazarse.
- Una cámara Logitech, conectada a la placa por USB
- Un servo conectado a la cámara

FOTO ROBOT

Raspberry Pi 3 model B

La Raspberry Pi 3 model B es un ordenador de placa reducida que lleva un sistema operativo Linux.

Su procesador es un ARM Cortex A53 de cuatro núcleos a 1.2GHz de 64 bits.

Tiene 1 GB de memoria RAM , 4 puerto USB, 40 pins GPIO, puerto HDMI , Ethernet y entrada para MicroSD.

Además tiene Wifi 802.11n integrado y bluetooth 4.1.

Servo motores

Para el movimiento del robot y de la cámara utilizamos servo motores

Un servo motor es un motor eléctrico que se puede controlar su velocidad y su posición (dentro del rango de posición permitido).

Los servos constan de:

- Un motor de corriente continua
- Una caja reductora

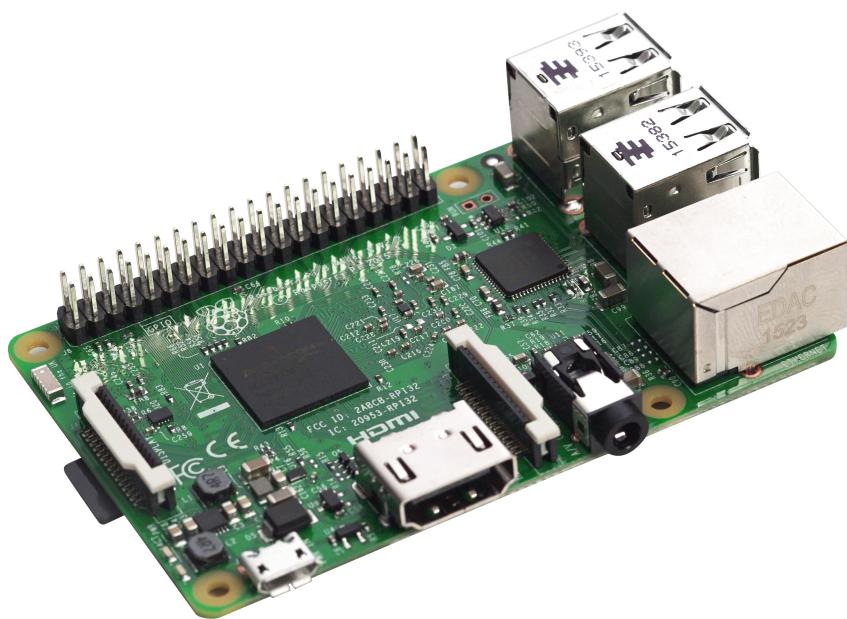
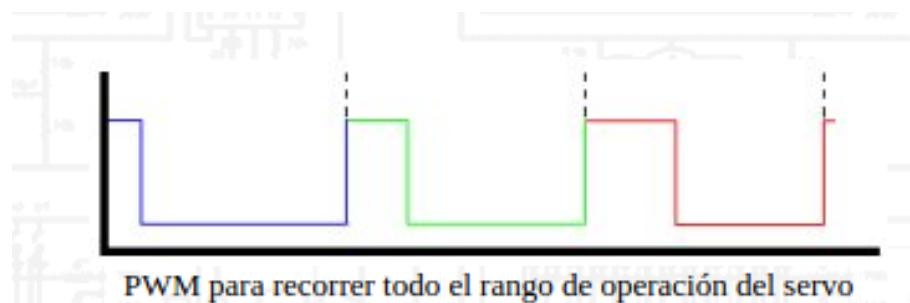


Figura 3.1: Raspberry Pi 3 model B

- Un circuito de control

El sistema que utilizamos para controlar la velocidad y la posición de los servos es la modulación por anchura de pulso (PWM).

PWM Este sistema consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo período (normalmente), con el objetivo de modificar la posición del servo según se desee.



Para que un servo se mantenga en la misma posición durante un cierto tiempo, es necesario enviarle continuamente el pulso correspondiente.

La otra función del PWM hemos dicho que era el control de velocidad.

Esto lo hace alimentando el motor con una señal de pulsos con la frecuencia suficiente para que el motor no note las variaciones y haga un giro constante, ya que variando el porcentaje de tiempo de la señal rectangular en alta, y en baja, variaremos la potencia que le entregamos al motor, con lo que controlamos la velocidad de giro con mucha precisión.

En el robot utilizamos 3 servomotores. Dos de ellos se utilizan para el movimiento de las ruedas y el tercero moverá la cámara permitiendo al usuario mirar hacia arriba y hacia abajo.

Los tres servos están controlados por el movimiento de la cabeza del usuario, que obtenemos gracias a las Oculus Rift.

Cámara

La cámara utilizada es una webcam USB. El proyecto también se podría haber hecho con una RaspiCam.

A continuación se muestra un esquema de la conexión entre los componentes que acabamos de explicar.

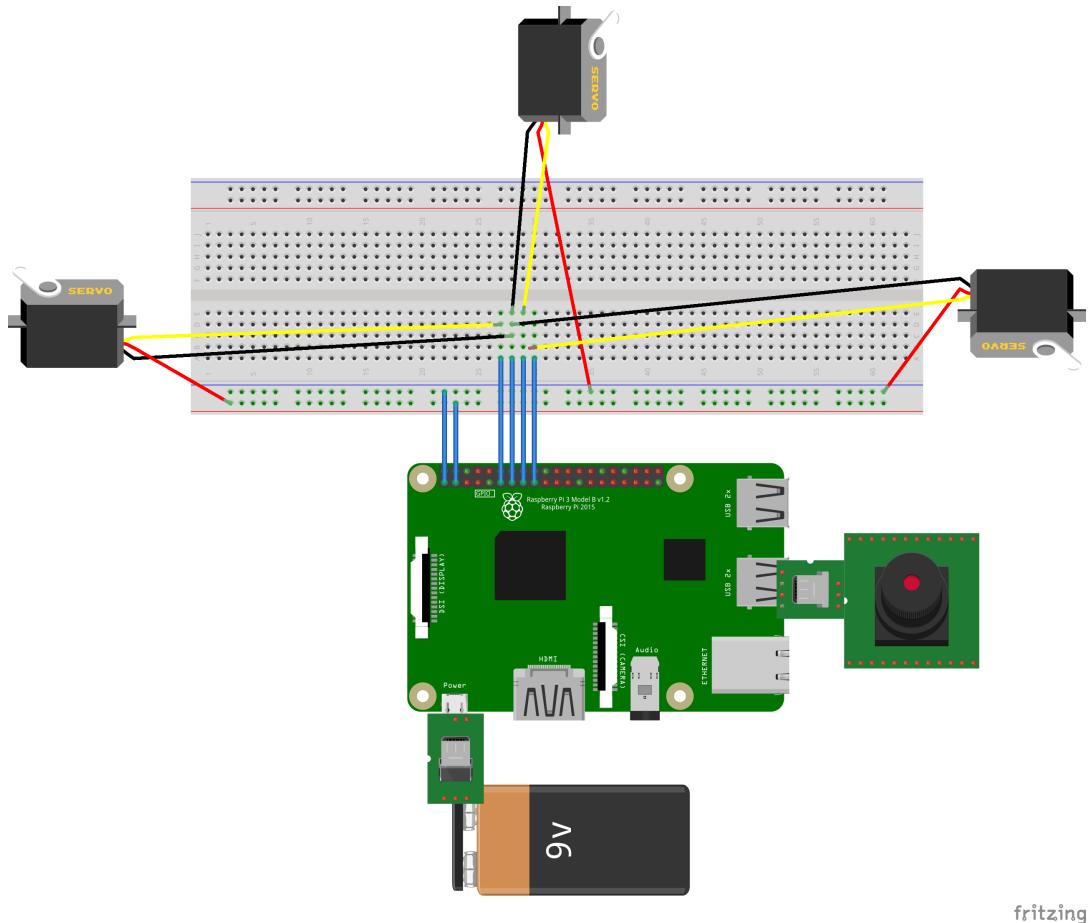


Figura 3.2: Esquema conexiones servos , cámara, Raspberry Pi 3

3.2.2 Router

Para la comunicación entre el robot y el ordenador se necesita una red local, para este trabajo la hemos creado haciendo uso de un router inalámbrico Xavi7968.

El router crea una red Wifi con el nombre de WifiRaspi3 y sin acceso a internet.

Para una mayor facilidad de conexión se ha configurado la tabla DHCP del router para asignar siempre la misma IP a la Raspberry y al equipo que utilizamos para conectarnos.

The screenshot shows the 'DHCP Server Configuration' page from a web-based router interface. On the left, a sidebar menu includes 'Home', 'Overview', 'Troubleshooting', 'Configuration' (selected), 'Quick Setup', 'Wireless Network', 'Internet Connection', 'Local Network (LAN)', 'DHCP Server' (selected), 'Vlan Config', 'Port-PVC', 'Security', 'Services', 'Port Statistics', and 'Admin'. The main content area has two sections: 'Existing DHCP server subnets' and 'Existing DHCP fixed IP/MAC mappings'. The subnet section shows one entry: Subnet Value 192.168.1.0, Subnet Mask 255.255.255.0, Use local host address as DNS server false, Use local host address as default gateway true, Assign Auto Domain Name true, with 'Edit' and 'Delete' buttons. The fixed IP section shows three entries:

IP Address	Mac Address	Max Lease Time	Default Lease Time
192.168.1.35	74:da:38:2e:03:38	86400	43200
192.168.1.33	b8:27:eb:70:7b:14	86400	43200
192.168.1.34	f8:16:54:c7:f1:ff	86400	43200

With 'Edit' and 'Delete' buttons for each row. A 'Server Settings' sidebar on the right provides information about DHCP and the server settings page.

Figura 3.3: Tabla DHCP del router con las parejas MAC-IP fijas

La Raspberry Pi 3 está configurada para conectarse automáticamente al router. Para ello se ha modificado el archivo `/etc/wpa_supplicant/wpa_supplicant.conf` donde se especifica el nombre y la clave de la red local.

```
pi@raspberrypi: /etc/wpa_supplicant
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.2.6          File: wpa_supplicant.conf

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
    ssid="Wifi_RaspPi3_2"
    key_mgmt=NONE
}
```

Figura 3.4: Archivo de configuración de red Raspberry Pi 3

3.2.3 Oculus Rift

Las Oculus Rift son unas gafas de Realidad Virtual [parrafo definiendo las Oculus]

De las gafas nos interesan especialmente:

- Detección de la orientación de las Oculus
- Recepción de video

Detección de la orientación

Las Oculus Rift tienen integradas un giroscopio, un acelerómetro y un magnetómetro que manda constantemente información al ordenador, de forma que éste sabe en todo momento la orientación de las gafas.

La técnica para interpretar la señal de estos sensores se llama **fusión de sensores**

A continuación se explica en qué consiste esta técnica.

Fusión de Sensores Como ya hemos mencionado, en las Oculus tenemos un giroscopio, un acelerómetro y un magnetómetro.

Los dos primeros dan información acerca de la orientación respecto a los ejes X y Z, mientras que el magnetómetro mide la orientación respect al eje Y.

Vamos a ver cómo funciona cada uno.

El **giroscopio** de las Oculus mide el cambio de orientación de la cabeza a una velocidad de 1KHz.

Una forma simplificada de calcular la orientación actual es:

$$\text{Orientación actual} = \text{Orientación anterior} + \text{Diferencia horaria} \cdot \text{Velocidad observada (giroscopio)}$$

El problema está en que la cabeza puede rotar alrededor de tres ejes.

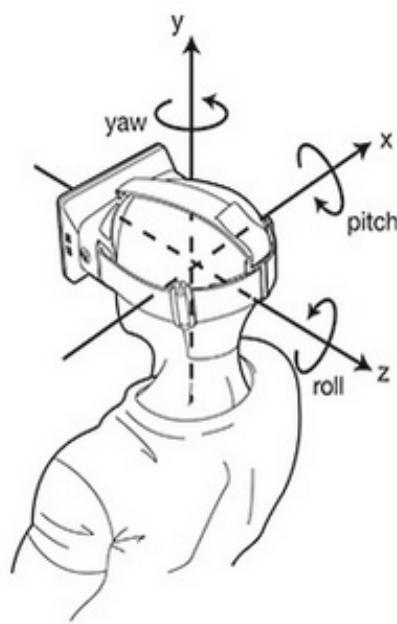


Figura 3.5: Ejes de giro

Vamos a suponer que solo rotase alrededor del eje Y y que el sensor captase una velocidad angular w por segundo.

Si tuviéramos 1000 sensores la fórmula de la orientación sería:

$$\text{Orientación actual} = \text{Orientación anterior} + 0,001 \cdot w$$

Pero en el caso de las Oculus, al estar midiendo un objeto que se mueve en 3 ejes, el giroscopio da la velocidad angular respecto de los tres ejes, devolviendo un vector 3D (w_1, w_2, w_3)

Al basar nuestro cálculo en el cálculo anterior, el error irá creciendo con el tiempo.

Vamos a llamar *tilt error* al error en la medición de los ángulos sobre los ejes XZ. El error sobre el eje Y se llamará *yaw error*.

El *tilt error* se corresponde con nuestra sensación de lo que "está arriba", percepción que se basa en la gravedad.

La gravedad se expresa con un vector de aceleración , por lo que utilizamos un **acelerómetro** para medirla.

Nos encontramos con el inconveniente de que el acelerómetro no solo mide la gravedad. Para asegurarnos de que el momento en el que tomamos los datos de referencia solo estamos midiendo la gravedad, esperamos a que se den 2 condiciones:

1. El acelerómetro nos devuelve una medida cercana a 9,8
2. El giroscopio nos devuelve una velocidad angular muy lenta (es decir, no estamos girando).

En este momento sabemos que nos encontramos en una posición vertical y podemos corregir el error.

Cómo corregimos el error: Supongamos que se dan las dos condiciones previas, y la posición que nuestro cálculo de orientación nos devuelve un vector \vec{a} , tal y como vemos en el dibujo.

Tenemos el ángulo θ entre el eje Y y el vector \vec{a} pero ¿cómo calculamos el eje de rotación para rotar \vec{a} y corregir el error?

Dicho vector debe ser perpendicular a \vec{a} y al eje Y, y apoyarse en el plano XZ.

Para encontrar el vector basta con hacer la proyección de \vec{a} en el plano XZ, obteniendo así $(a_x, 0, a_z)$. Haciendo un vector perpendicular a éste obtenemos $(-a_z, 0, a_x)$. Y ya tenemos el eje de rotación que necesitábamos para corregir el error.

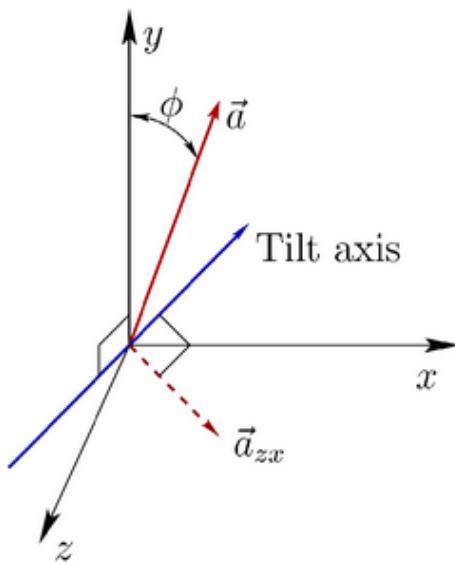


Figura 3.6: Corrección del tilt error

Error sobre el eje Y (yaw error)

Este error se basa en nuestra percepción de dónde está el norte.

Para esto utilizamos el magnetómetro.

El procesamiento de la señal del magnetómetro y la corrección del error es similar al tratamiento de la señal del giroscopio.

3.2.4 Diseño de conexiones

La arquitectura del proyecto es muy sencilla, como podemos observar en la siguiente imagen:

Como podemos ver en el esquema, las Oculus Rift están conectadas al ordenador, el ordenador y la Raspberry se mandan información a través de un router que crea una red local.

La raspberry recibe en streaming de la cámara y lo va mandando a una IP disponible dentro de la red local, la (192.168.1.35). A su vez el ordenador accede a esa IP, recoge el streaming utiliza VLC para dividir la imagen (SBS) y reproducir el video y con la herramienta Virtual Desktop podemos ver el video desde las Oculus.

Por otra parte las Oculus tienen integrado un giroscopio , un acelerómetro y un magnetómetro.

Combinando la información de estos sensores a través de un proceso conocido como fusión de sensores se determina la orientación de la cabeza del usuario en el mundo real , y se sincroniza con la perspectiva virtual del usuario en tiempo real.

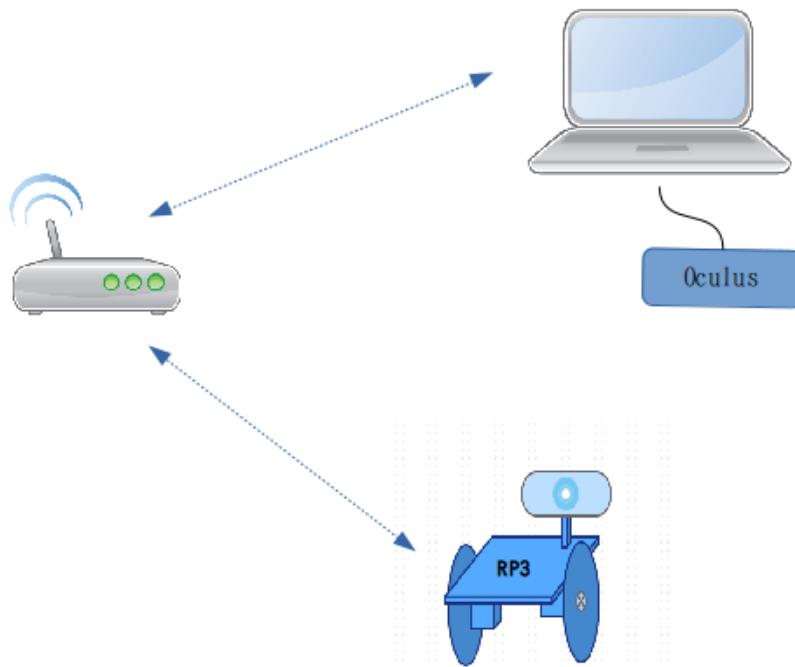


Figura 3.7: Esquema conexiones

Se ha implementado un programa en python que recoge esa información y la traduce en comandos sencillos para mandárselos a la Raspberry. Para obtener esa información hacemos uso de la librería de las Oculus (ovr).

La Raspberry lee los comandos del ordenador y mueve los servos según lo recibido, de esta forma controlamos el movimiento del robot y de a cámara.

3.3 Desarrollo e implementación

En esta sección se explicará cómo hemos desarrollado las distintas funcionalidades del proyecto. Para ello vamos a diferenciar entre:

- Construcción del Robot
- Streaming
- Control del Robot

Para conseguir **transmitir a las Oculus el video en tiempo real** es necesario que:

1. La Raspberry recoja el video de la cámara y lo transmita al ordenador.
2. El ordenador recoja el video, lo transforme a formato SBS y lo envíe a las Oculus Rift.

Para el **control del robot** se deberán implementar las siguientes tareas:

1. Leer la señal de los sensores de las Oculus Rift (giroscopio, acelerómetro y magnetómetro) para saber la orientación de la cabeza del usuario.
2. Transmitir la información obtenida al robot.
3. Traducir dicha información en comandos para mover los servo motores.

3.3.1 Construcción del Robot

Como se ha dicho en la sección de diseño, el robot está construido basándose en el diseño de un robot de un trabajo anterior.

Para este proyecto se han añadido algunos componentes:

- Soporte de la cámara
- Batería portátil

Soporte de la cámara

Para integrar la cámara en el robot han sido necesarias dos piezas:

- Soporte vertical → En esta pieza se coloca el servo motor que controla el movimiento de la cámara.
- Pinza de agarre → Se utiliza para unir la cámara con el servo motor que la mueve.

Ambas piezas han sido impresas con la impresora 3D del Club de Robótica de la Universidad Autónoma de Madrid.

Para realizar el diseño 3D de la pinza de agarre se ha utilizado el programa Blender.

Para el soporte vertical se ha reciclado el diseño de un soporte de un robot construido en el Club de Robótica.

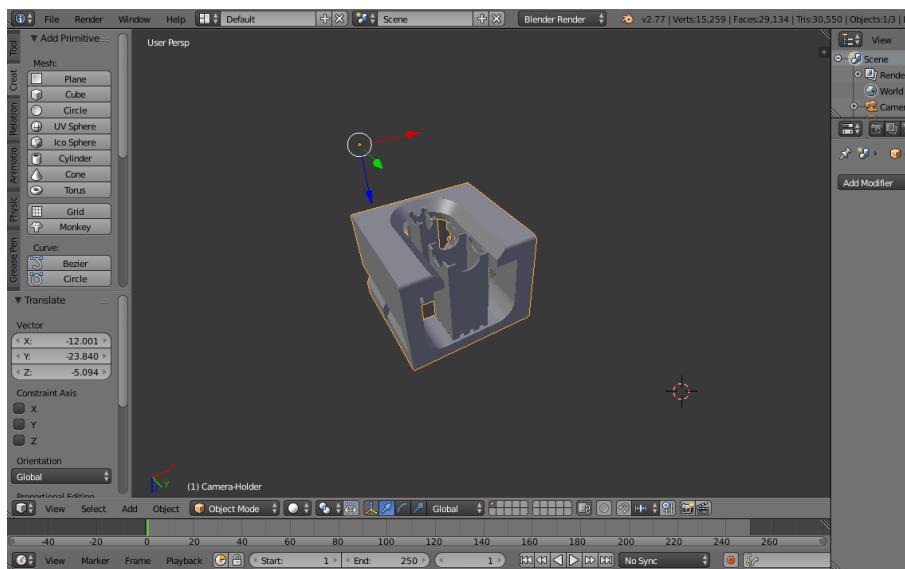


Figura 3.8: Diseño 3D de la pinza de agarre en Blender

Una vez diseñadas las piezas se ha utilizado Cura, un programa que genera archivos .gcode a partir de diseños 3D. Este tipo de archivos son los que se le envían a la impresora 3D.

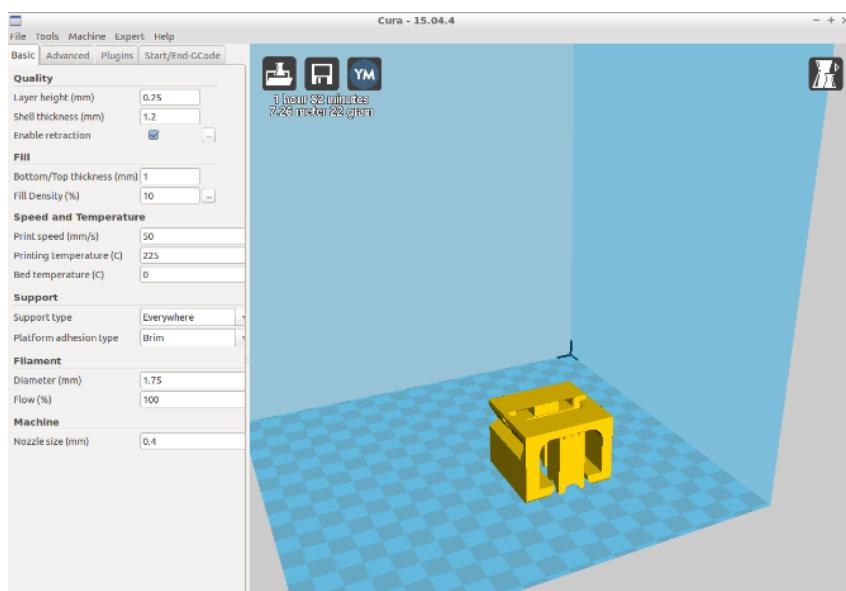


Figura 3.9: Imagen de la configuración del Cura para enviar la Pieza de agarre a la impresora 3D

3.3.2 Streaming

Raspberry Pi3

El primer elemento en el streaming de video es la Raspberry Pi, que tiene conectada por usb una cámara.

Para recoger el video y mandarlo en tiempo real utilizamos la librería mjpeg-streamer, disponible en github.

Mjpeg-streamer es una aplicación en linea de comandos que permite crear un servidor, para retransmitir imágenes JPG sobre una red basada en IP, desde la cámara hasta un navegador convencional.

Soporta la compresión por hardware (GPU, Unidad de Procesamiento Gráfico) de la cámara, en nuestro caso H.264 Advanced Video Coding (AVC) Standard, que es el compreso de la Webcam Logitech.

Esto permite reducir drásticamente el uso de la CPU de este servidor, haciendo esta aplicación un servicio ligero.

El puerto que emplea es el 8080

```

pi@raspberrypi: ~
top - 20:34:28 up 17 min, 4 users, load average: 0.54, 0.32, 0.15
Tasks: 120 total, 1 running, 119 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 2.7 sy, 0.0 ni, 95.0 id, 0.0 wa, 0.0 hi, 2.1 si, 0.0 st
KiB Mem: 883052 total, 216956 used, 666096 free, 26448 buffers
KiB Swap: 102396 total, 0 used, 102396 free. 108000 cached Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  976 root      20   0       0       0      0 S 6.9  0.0  0:08.66 kworker/u8+
  210 root     -2   0       0       0      0 S 5.0  0.0  0:13.70 ksdiiorqd/+
  6 root      20   0       0       0      0 S 2.0  0.0  0:12.72 kworker/u8+
1020 pi      20   0  51816  2756   2524 S 1.7  0.3  0:03.99 mjpg_strea+
  3 root      20   0       0       0      0 S 1.3  0.0  0:03.71 ksoftirqd/0
1072 pi      20   0  5112  2460  2092 R  0.7  0.3  0:00.61 top
  15 root     20   0       0       0      0 S 0.3  0.0  0:00.35 ksoftirqd/2
  19 root     20   0       0       0      0 S 0.3  0.0  0:00.54 ksoftirqd/3
  1 root      20   0 23872  3904  2732 S 0.0  0.4  0:05.00 systemd
  2 root      20   0       0       0      0 S 0.0  0.0  0:00.00 kthreadd
  5 root      0 -20      0       0      0 S 0.0  0.0  0:00.00 kworker/0:+
  7 root      20   0       0       0      0 S 0.0  0.0  0:00.11 rcu_sched
  8 root      20   0       0       0      0 S 0.0  0.0  0:00.00 rcu_bh
  9 root      rt  0       0       0      0 S 0.0  0.0  0:00.01 migration/0
 10 root     rt  0       0       0      0 S 0.0  0.0  0:00.00 migration/1
 11 root     20   0       0       0      0 S 0.0  0.0  0:00.44 ksoftirqd/1
 13 root     0 -20      0       0      0 S 0.0  0.0  0:00.00 kworker/1:+
 14 root     rt  0       0       0      0 S 0.0  0.0  0:00.00 migration/2
 17 root     0 -20      0       0      0 S 0.0  0.0  0:00.00 kworker/2:+
 18 root     rt  0       0       0      0 S 0.0  0.0  0:00.00 migration/3
 21 root     0 -20      0       0      0 S 0.0  0.0  0:00.00 kworker/3:+

```

Figura 3.10: Uso de CPU del programa Mjpeg-Streamer por el que se hace el streaming de video

Para iniciar el streaming debemos ejecutar un script Streaming que manda el siguiente comando: `./mjpg_streamer -i "./input_uvc.so" -d "/dev/video0" -o "./output_http.so" -w "./www"`.

En este momento la Raspberry empezará a coger el video de la cámara y transmitirlo en tiempo real a la siguiente url: `http://192.168.1.33:8080/stream.html`

```
p@raspberrypi:~/mjpgStreamer/mjpg-streamer-experimental$ ./mjpg_streamer -l "./input_uvc.so -d /dev/video0" -o "./output_http.so -w ./www"
MJPEG Streamer Version.: 2.0
l: Using UVC2 device.: /dev/video0
l: Desired Resolution: 640 x 480
l: Framerate Per Second.: 30
l: Format.: JPEG
l: TV-Norm.: DEFAULT
o: www-folder-path...: /www/
o: HTTP/TCP port...: 8088
o: username:password.: disabled
o: commands.....: enabled
```

Figura 3.11: Ejecución del comando para iniciar el streaming

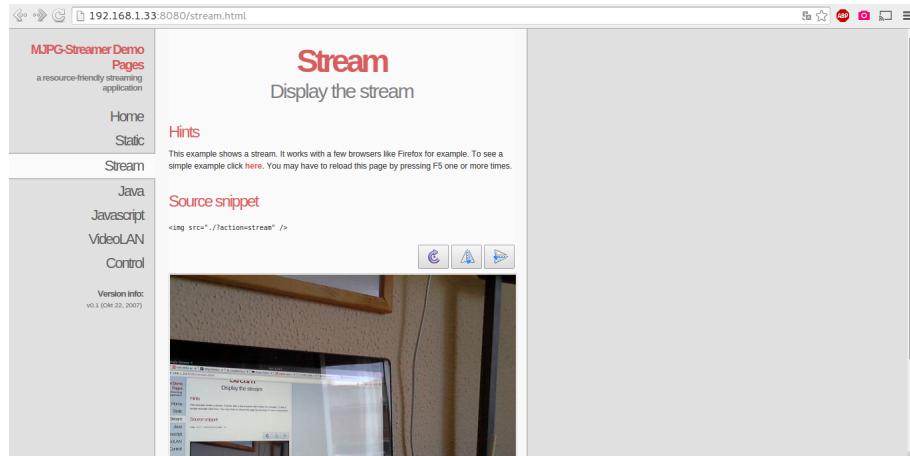


Figura 3.12: Página html donde se recibe el streaming a tiempo real desde la Raspberry

Ordenador

El ordenador accede a la IP donde mjpeg-streamer está retransmitiendo el video y lo recoge a través del reproductor de video VLC.

Para ello especificamos en VLC que el medio de reproducción será una ubicación de red.

Posteriormente dividimos el video en dos pantallas exactamente iguales para poder verlo desde las Oculus en modo SBS (Side by Side).

El ordenador tiene instalado un programa, Virtual Desktop.

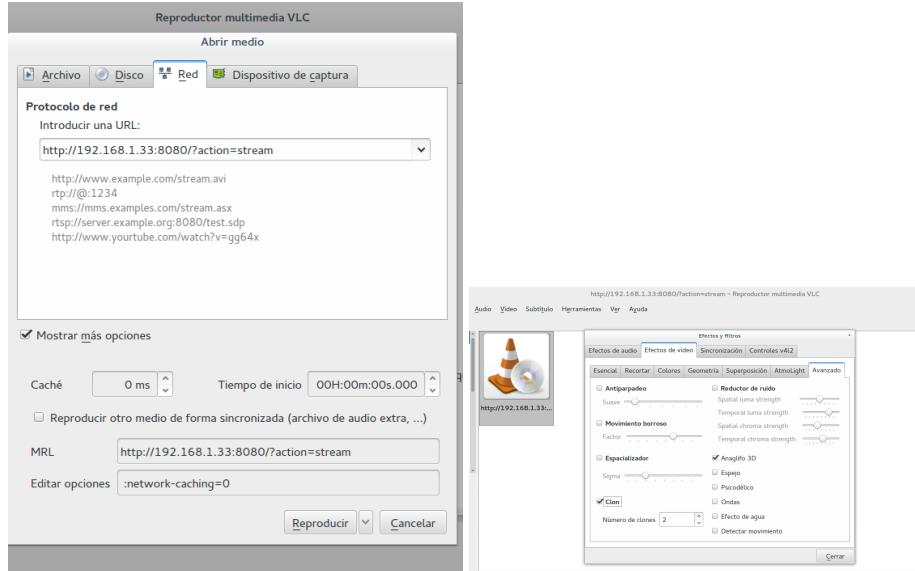
Virtual Desktop es una aplicación desarrollada por Oculus Rift y HTC Vive para utilizar las Oculus Rift en Windows.

Permite ver el escritorio desde las Oculus, pudiendo configurar efectos como ver las imágenes SBS o cambiar el entorno en el que te encuentras (el espacio, una sala de cine...)

También te permite ver videos descargados en el ordenador y reproducir videos 360.

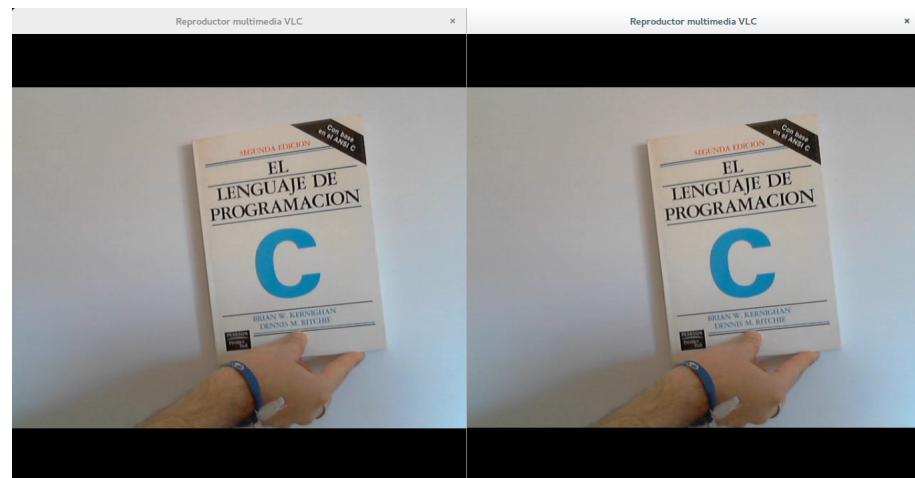
Nosotros utilizamos la funcionalidad de ver la pantalla del ordenador desde las Oculus con SBS, para visualizar el video que se está reproduciendo en VLC a tiempo real.

3. ANÁLISIS, DISEÑO Y DESARROLLO



(a) Medio de reproducción-Ubicación de red

(b) División de imágenes



(c) Reproducción SBS

Figura 3.13: Configuración y resultado de la reproducción de video en VLC

3.3.3 Control del Robot

El control del robot es el control del movimiento de los tres servos que lo componen.

Este movimiento se basará en el movimiento de las Oculus, que llevará puestas el usuario.

Por lo tanto el control del robot se divide en tres acciones:

- Recogida y procesamiento de la información que mandan las Oculus al ordenador → Esto se hará mediante la utilización de la librería ovr.
- Envío de la información procesada a la Raspberry Pi 3 → Para ello el ordenador abrirá dos sockets a través de los cuales mandará los datos al robot

- Movimiento de los servo motores

Recogida y procesamiento de la información que mandan las Oculus al ordenador

Librería OVR. Para recoger desde el ordenador la información que mandan las Oculus utilizamos la librería de Oculus (ovr), escrita en python.

La función *ovr.getTrackingState* nos devuelve una estructura (*TrackingState*), que contiene el campo:

Pose Statef HeadPose

- **Posef** ThePose $\begin{cases} \text{Quatf Orientation} \\ \text{Vector3f Position} \end{cases}$

Para obtener la orientación de la cabeza se lee el valor de *HeadPose.ThePose.Orientation.x/y* lo procesa para obtener los grados de orientación de la cabeza y se lo manda a la raspberry a través de sockets.

La información obtenida sobre la orientación respecto al eje *y* servirá para mover los servos que hay en las ruedas del robot, mientras que la información de *x* servirá para mover la cámara.

El ordenador crea un servidor y abre dos sockets distintos, uno para la información de *x* y otro para la de *y*.

De esta forma manejamos los dos movimientos de forma independiente.

Procesamiento de HeadPose.ThePose.Orientation. Como se ha dicho en el apartado anterior, la información proveniente de las Oculus nos llega a través de la estructura HeadPose, donde podemos leer el campo Orientation que nos manda la orientación actual de las Oculus Rift en forma de cuaternión.

Para averiguar en qué dirección apuntan las Oculus respecto al eje X guardamos los datos relativos a la posición y la velocidad angular.

Estos datos son procesados y se le aplican unos parámetros que se han obtenido mediante la realización de pruebas para obtener una correcta calibración.

De este modo se ha podido realizar una conversión de los datos obtenidos por las Oculus a unidades de grados que nos facilitan el control posterior de la posición de los servomotores.

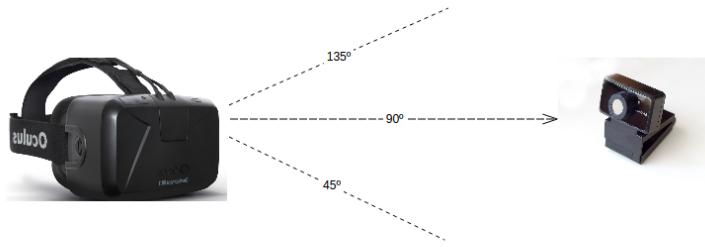


Figura 3.14: Posición de las Oculus respecto al eje X una vez procesados los datos de HeadPose.ThePose.Orientation

Envío de la información procesada a la Raspberry Pi 3 Para el control del movimiento del robot es necesario que el ordenador le mande los datos referentes a la posición de las Oculus de forma automática, inmediatamente después de procesarlos.

Con este fin, desde el mismo programa donde se recibe y procesan los datos de las Oculus, el ordenador abre dos sockets, y se queda a la espera de que el robot se conecte.

Recordemos que el ordenador y la Raspberry Pi 3 están dentro de la misma red local (WifiRaspi3), y que ambos tenían IP fija.

La apertura de sockets por parte del ordenador se hace de la siguiente forma:

```

host = "192.168.1.35"      #IP asignada al ordenador
port = 4446                #movimiento de la cámara
port2 = 4447                #movimiento de las ruedas

s = socket(AF_INET, SOCK_STREAM)
s2 = socket(AF_INET, SOCK_STREAM)

s.bind((host, port))
s2.bind((host, port2))

s.listen(5)
s2.listen(5)

q,addr = s.accept()
q2,addr2 = s2.accept()

```

De esta forma, cada vez que se recibe y procesan datos referentes a la posición de las Oculus, se dividen en información para la cámara e información para las ruedas y se ejecutan las instrucciones q2.send(data) y q.send(data) para mandar los datos.

Recepción de los datos en el robot. La Raspberry tiene dos programas (uno para las ruedas y otro para la cámara) que están escuchando continuamente lo que manda el ordenador y en cuanto recibe el comando lo manda a los servos.

Para ello, al iniciarse cada programa, se conectan al socket correspondiente

```
host = "192.168.1.35"
port=4447
s=socket(AF_INET, SOCK_STREAM)
s.connect((host, port))
```

Y se ponen a escuchar (`msg=s.recv(1024)`) hasta que les llegan los datos.

Una vez que tiene los datos los transforma en comandos válidos para los servomotores, tal y como se explica en el siguiente apartado, les manda la instrucción y vuelve a escuchar.

Movimiento de los servo motores Como ya se indicó en la sección (sección de los servos, PWM, etc..) el movimiento de los servos se controla con una señal PWM (Pulse With Modulation).

Por ello utilizamos la librería RPi.GPIO, que ofrece funciones para PWM.

Hay dos parámetros principales para determinar el pulso que se envía al motor:

- **Frecuencia**: veces por segundo que se genera el pulso.
- **Duty cycle**: es el porcentaje de tiempo que el pulso está arriba. (recordemos que la señal es una señal cuadrada)

Veamos un ejemplo para clarificar esto:

PWM- Frecuencia:50Hz , DutyCycle 50 %. Si fijamos una frecuencia de 50Hz estaremos mandando una señal de 50 pulsos por segundo, es decir, un pulso cada 0.02 segundos.

Al poner un DutyCycle al 50 % estamos estableciendo que durante esos 0.02 segundos el pulso estará 0.01 segundos en posición "High" y el resto del tiempo en posición "Low".

PWM- Frecuencia:50Hz , DutyCycle 80 %. Si fijamos una frecuencia de 50Hz, como la del ejercicio anterior pero por el contrario ponemos un DutyCycle del 80 % tendremos:

- 1 pulso cada 0.02 segundos
- El pulso estará en posición "high" el 80 % del tiempo (0.016 segundos) y en posición "low" los otros 0.004 segundos restantes.

La Raspberry Pi recibe a través de los sockets el ángulo de orientación de la cabeza sobre el eje y o x.

Esta información se procesa de forma distinta según si queremos mover el servo de la cámara o las ruedas del robot.

Esto es porque los servos de las ruedas son de rotación continua, a diferencia que el de la cámara.

Control de los motores de las ruedas del robot

Los motores que mueven las ruedas del robot son dos servo motores que están modificados para tener rotación continua. El funcionamiento de los servo motores de rotación continua ya se ha explicado en la sección de análisis.

Recordamos muy brevemente que en los servo motores de rotación continua, la pulsación que le mandas al motor no indica la posición en la que debe colocarse si no la velocidad a la que debe rotar.

Para controlar nuestro robot se ha fijado una velocidad constante de giro en base a pruebas experimentales de la velocidad de movimiento de cabeza medio en una persona.

Por lo tanto , si queremos rotar nuestro robot hasta una posición concreta la variable no es la velocidad si no el tiempo durante el que rotan los motores.

Para calcular dicho tiempo se midió experimentalmente cuanto tardaba el robot en girar 90°. El resultado fueron 0.5 segundos por lo que

$$90/0.5 = 0,005555556 \text{ segundos en girar un grado}$$

De esta forma, cuando el robot recibe una posición en grados desde el ordenador ejecuta las siguientes instrucciones:

$\left\{ \begin{array}{l} \text{Grados recibidos del ordenador} = g \\ dif = |posActual - g| \rightarrow \text{Se calcula el número de grados que tiene que moverse} \\ \text{Rotar motores derecho e izquierdo} \\ time.sleep((dif * 0.005555) \rightarrow \text{Rotamos el tiempo necesario alcanzar la posición} \\ \text{Parar motores derecho e izquierdo} \end{array} \right.$

Control del motor de la cámara del robot

La cámara se mueve gracias a un servo motor cuyo rango de movimiento va de 0 a 180 grados.

Este motor se mueve por pulsos, tal y como se explica en la sección de Análisis.

La Raspberry recibe un ángulo que indica hacia donde están orientadas las Oculus en ese momento.

Para transformar este ángulo el pulso(PWM) correspondiente debemos tener en cuenta que un servo requiere una señal PWM con un periodo de 20 ms y un ancho de pulso entre 0.9 y 2.1 ms.

Ya que el servo tiene un rango de ángulos de 0 a 180 grados, es fácil ver que el ancho de pulso de 0.9 ms se corresponde con 0° y 2.1 ms con 180°

Así podemos convertir los ángulos en pulsos PWM y mover la cámara hacia la dirección deseada.

Control del Robot por parte del usuario En las anteriores secciones se ha explicado el funcionamiento interno del control de movimientos del robot.

En este nuevo apartado se presenta cómo debe utilizar el usuario este producto, qué rango de movimiento tiene y cómo puede hacer avanzar el robot.

Instrucciones de mando

Como se especifica al inicio de este trabajo, el robot estará controlado exclusivamente por el movimiento de cabeza del usuario.

Para ello se debía idear una forma cómoda e intuitiva de mandar instrucciones al robot.

Diferenciamos dos estados de movimiento del robot:

- Exploración → En este estado el robot no avanza, solo explora lo que tiene a su alrededor. Esto lo hace moviendo la cámara o rotando sobre sí mismo.
- Transporte → El robot avanza hacia delante.

Finalmente se ha implementado de la siguiente forma:

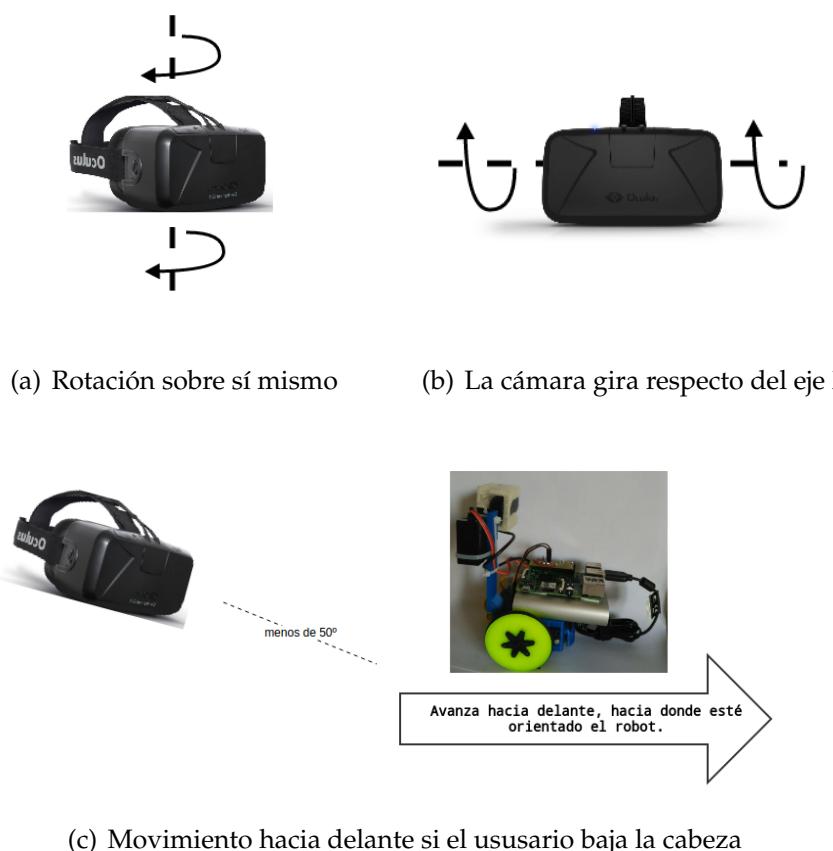


Figura 3.15: Posibles movimientos del robot según el movimiento de las Oculus Rift

Como viene explicado en la imagen, si el usuario mira hacia los lados el robot rotará sobre sí mismo sin moverse del sitio, hacia la dirección que señalen las Oculus Rift.

Si el usuario mueve la cabeza hacia arriba o hacia abajo sin llegar a menos de 50°, la cámara se moverá como si fuera la cabeza del usuario.

3. ANÁLISIS, DISEÑO Y DESARROLLO

En el caso de que el usuario oriente la cabeza hacia abajo, con un ángulo menor de 50° respecto al eje Y, la cámara se orientará horizontalmente (90° respecto al eje Y) y el robot avanzará hasta que el usuario cambie la posición de la cabeza

PRUEBAS Y RESULTADOS

A continuación se explican las pruebas y los resultados obtenidos tras desarrollar el proyecto.

4.1 Latencia de recepción de comandos

Para probar el sistema primero hemos hecho pruebas del funcionamiento de la red local, ya que en este trabajo es esencial tener poca latencia.

Para ello se ha probado el alcance de la red inalámbrica, mandando comandos a los servo motores desde el ordeador.

A continuación se ven los resultados obtenidos tras mandar comandos a una distancia de 7,5 metros y mandar comando estando el robot y el ordenador en habitaciones separadas

Vemos que la latencia entre la medición en las Oculus y la recepción en la Raspberry es muy pequeña.

En la gráfica que nos muestra la diferencia de tiempos a 7,5 metros vemos que de media el comando tarda en transmitirse menos de 0.5 segundos.

En las pruebas entre habitaciones separadas, vemos que la diferencia no pasa de los 0,5 segundos por lo que creemos que es una latencia asumible ya que el robot no va a grandes velocidades.

4. PRUEBAS Y RESULTADOS

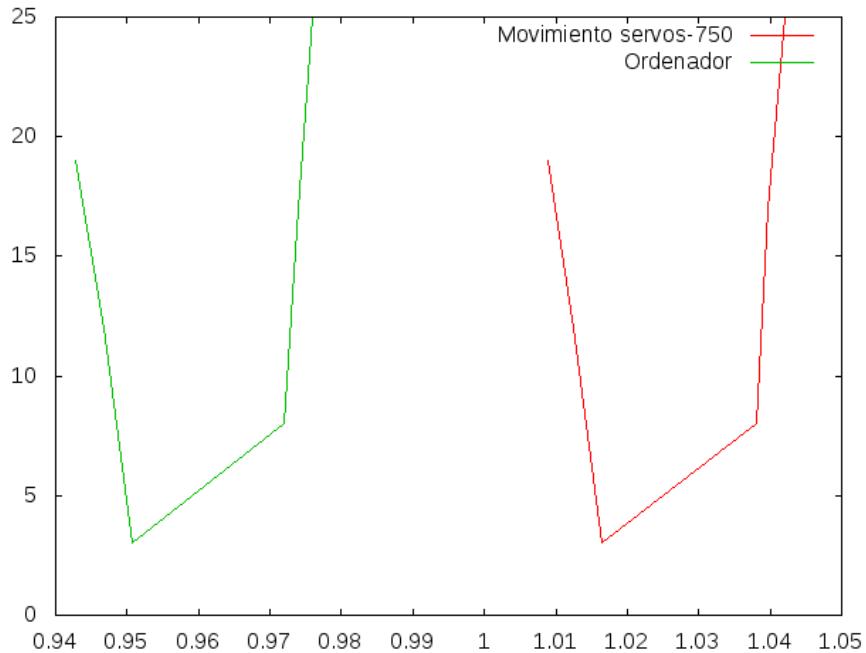


Figura 4.1: Diferencia de tiempos entre envío y recepción de comandos a 7,50 metros

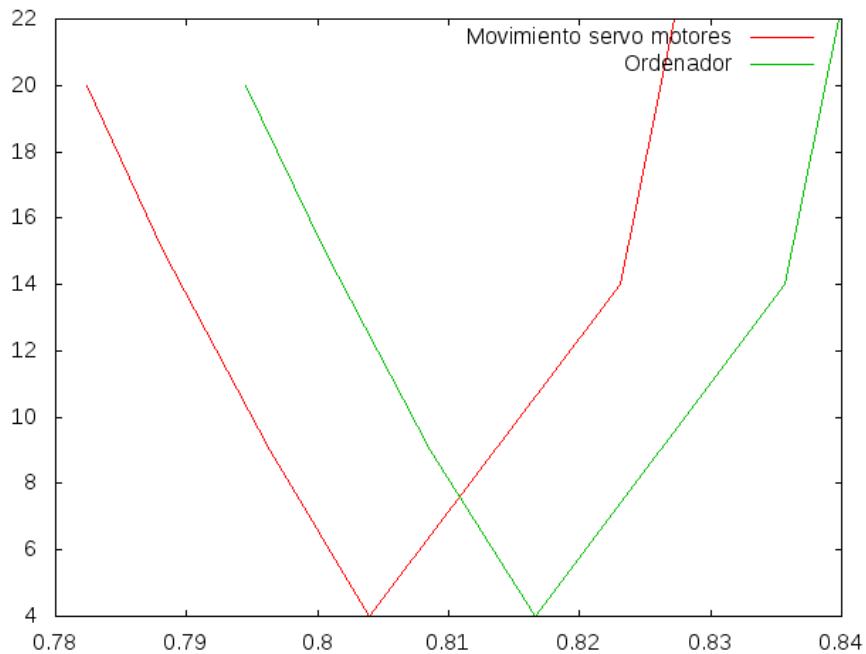


Figura 4.2: Diferencia de tiempos entre envío y recepción de comandos a 7,50 metros

4.2 Feedback de los usuarios

Se han hecho pruebas de control a usuarios para comprobar su usabilidad.

Las primeras pruebas han sido de calibración. Un usuario ha aprendido a controlar el robot y ha pedido cambios para que el manejo sea más intuitivos.

Tras hacer los cambios pedidos por el primer usuario se ha pedido a distintas personas que hagan diversas acciones con el robot y que puntúen la facilidad de realizar dichas acciones.

Cambios realizados en la fase de calibración

- **Se ha reducido la velocidad de giro de los motores de las ruedas.**
- **Se ha modificado la forma de girar.** En las primeras pruebas el robot giraba de forma que siempre apuntaba al mismo sitio que la cabeza del usuario.

Esto era bastante incómodo porque para dar la vuelta el usuario debía girar sobre sí mismo.

Además para avanzar hacia la derecha o hacia la izquierda el usuario debía estar mirando en esa dirección.

Esto se ha modificado de forma que si la cabeza del usuario está orientada en un ángulo mayor de 110 grados (siendo 90° la posición frontal natural) el robot girará continuamente hacia la derecha hasta que el usuario vuelva a mirar al centro.

El giro hacia la izquierda se hace de la misma forma.

- **Se ha modificado la forma de parar.** Cuando el usuario mira hacia abajo el robot avanza.

En las primeras pruebas el usuario debía girar hacia la derecha o hacia la izquierda para dejar de avanzar.

Esto se ha modificado de forma que ahora para parar el usuario deberá colocar la cabeza en la posición central natural (90° respecto al eje Y)

Valoración de los usuarios Se ha pedido a los usuarios que realicen las siguientes acciones:

- Girar a la derecha
- Girar a la izquierda
- Mirar hacia arriba
- Estar 30 segundos aproximadamente girando el robot sobre sí mismo hasta tener controlado el giro.
- Andar hacia delante
- Parar
- Estar cerca de un minuto paseando para familiarizarse con los movimientos.
- Fijar un objetivo y alcanzarlo.

4. PRUEBAS Y RESULTADOS

La siguiente tabla muestra la valoración de los usuarios tras realizar las pruebas.

	U1	U2	U3	U4	U5	U6	U7
Giro derecha	9	6	6	4	5	5	6
Giro izquierda	7.5	4	6	4	5	5	7
Mirar hacia arriba	9	10	9	10	9	9	9
Andar	10	8	10	8	8	9	9
Parar	9	10	9	7	7	7	8
Alcanzar un objetivo	8.5	6	8	7	5	5	8
Sensación de Control	7.5	6	9	4	5	4	7
Facilidad de aprendizaje	9	8	10	10	9	8	10

Comentarios tras las pruebas Los comentarios más usuales tras utilizar el robot han sido:

- El robot gira demasiado deprisa.
- Sería bueno incluir un micrófono en el robot ya que marea un poco el mirar un espacio desde el robot y estar oyendo los ruidos del espacio en el que我真的 estoy.
- Es muy intuitivo la forma de controlar el robot.
- Cuanto más lo uso más fácil me es controlarlo.

Las conclusiones tras analizar los resultados se exponen en el siguiente capítulo.

CONCLUSIONES Y TRABAJO FUTURO

5.1 Conclusiones

En este apartado analizaremos los resultados obtenidos, hablaremos de posibles mejores y se expondrán las conclusiones del proyecto.

Puntos positivos y a mejorar Tras las pruebas se ve que el mayor defecto del proyecto es la **velocidad de giro** de los motores de las ruedas.

La calibración de servo motores es una tarea que se resuelve de forma experimental. La dificultad que nos encontramos es que la velocidad de giro cambia según en la superficie en la que se encuentre el robot por lo que habría que realizar las pruebas en diferentes entornos y hacer una calibración media.

Los usuarios han estado cada uno menos de 7 minutos con el robot y **todos menos uno lograron alcanzar su objetivo** en menos de un minuto, por lo que es una **aplicación fácil de manejar y de aprender**, como se ha visto en las valoraciones.

Otro problema es el **alcance de red de la Raspberry Pi 3**. Si el robot se alejaba mucho del router los servos seguían recibiendo correctamente los comandos pero el streaming tenía algo más de latencia.

Además la **Raspberry Pi 3 necesita de más amperaje que la Raspberry Pi 2** por lo que se concluye que por ahora, y hasta que se mejore la Raspberry Pi 3, es mejor utilizar Raspberry Pi 2 con un pincho Wifi para poder conectarse a la red local.

De esta forma se consigue mayor alcance de red y se necesita menos Amperios para alientar la placa.

A pesar de todos estos puntos a mejorar el resultado es muy bueno, teniendo en cuenta que es una primera aproximación a lo que pretende ser un proyecto futuro.

Además es un proyecto con una **muy buena acogida**, a todos los usuarios les ha resultado ameno hacer las pruebas y ver cómo iban mejorando.

5. CONCLUSIONES Y TRABAJO FUTURO

También es fácil de desarrollar con las herramientas de las que disponemos y con **mucho potencial**, ya que incluye dos tecnologías que están en pleno desarrollo y que aún tienen mucho que avanzar.

Otra ventaja es que hace uso de dispositivos de **fácil acceso** por lo que no resultaría demasiado caro de fabricar.

5.1.1 Tecnologías aprendidas

Durante la realización de este proyecto he utilizado muchas tecnologías, alguna de ellas completamente nuevas para mí.

El ejemplo más claro de esto son las gafas de realidad virtual Oculus Rift. Ha sido una suerte tener acceso a ellas y poder aprender a utilizar algunas de las librerías que hay disponibles para los desarrolladores.

También ha sido una sorpresa descubrir la gran variedad de posibilidades a desarrollar que permiten las Oculus, como es este proyecto.

He afianzado los conocimientos de robótica que ya tenía. Todo lo que sabía de robótica lo había aprendido en el Club de Robótica de esta universidad y gracias a este trabajo he podido aplicarlos y aumentarlos.

Todo el proyecto está desarrollado en python, lenguaje de programación que no he utilizado en ninguna asignatura de la carrera y que solo había visto muy por encima para proyectos sencillos. Durante este proyecto he aprendido a utilizar las librerías de python disponibles para las Oculus Rift y para el control de servomotores.

La placa base del robot es una Raspberry Pi 3, que tampoco había tenido la oportunidad de utilizarla nunca. De todas formas esto no ha sido novedoso para mí porque sí había participado en algún proyecto sencillo con Raspberry.

En conclusión este trabajo ha sido muy instructivo y ha ampliado de forma significativa mis conocimientos sobre robótica y realidad virtual.

5.2 Trabajo futuro

Este trabajo es solo el inicio de lo que podría ser un proyecto muy interesante.

El robot que hemos construido es una extensión del sentido de la vista, pero se podría construir un robot controlado por una persona y que fuera la extensión de todos sus sentidos, permitiendo así conocer y sentir el mundo sin necesidad de moverse de su casa.

Además la realidad virtual es una herramienta muy potente, que no solo te permite modificar tu mundo a placer sino que te premita crear entornos totalmente distintos. Se podría estudiar cómo interactuar con ese mundo virtual no solo a través de la vista, sino con todos los sentidos.

Otro camino por el que se puede investigar es la unión de este proyecto con la domótica. Creo que en un futuro no muy lejano, las casas serán casas domotizadas, que igual que se podrán controlar desde el móvil podremos controlarlas con las gafas de realidad virtual.

BIBLIOGRAFÍA

