Universitat Autònoma
de Barcelona

Sistema de videovigilancia a través de una Raspberry Pi

Informe de Progreso I

Aitor Domene Sánchez
1332008

Grau en Enginyeria Informàtica

Escola d'Enginyeria

Curso 2016-17

Tabla de contenidos

Tabla de figuras

1. Introducción

En este documento se encuentran los avances efectuados en el desarrollo del proyecto de fin de grado *Sistema de videovigilancia a través de una Raspberry Pi*. Se presentan los ajustes realizados y metodologías del proyecto, así como una explicación de estas últimas.

También, se hace un resumen a modo de conclusiones del proyecto hasta ahora y se presentan las fuentes bibliográficas y de información que han sido necesarias para elaborar este informe.

Al final del documento se detallan diferentes partes que hacen referencia al proyecto como, la arquitectura de comunicación entre la Raspberry Pi y la app Telegram o el funcionamiento del sistema a día de hoy, entre otros.

2. Palabras claves

Keyboard: menú de peticiones definidas en el chat de la app Telegram para facilitar el uso del sistema al usuario.

Sprint: subconjuntos de requerimientos para ser ejecutados durante un periodo de tiempo

OTP (One-Time Password): contraseña solo para una autenticación implementada en la petición *Video Streaming* solicitada por usuario.

UML (Unified Modeling Language): lenguaje de modelado de sistemas de software

Thread: secuencia de tareas ejecutadas por un sistema operativo

3. Objetivos del proyecto

Los objetivos del desarrollo del proyecto que se describieron en el informe inicial, son los siguientes:

1. Integrar una comunicación entre el sistema y el usuario

2. Facilitar al usuario el uso del sistema, así como poder estar siempre informados de lo que suceda en todo momento

3. Gestionar los diferentes componentes conectados a la Raspberry Pi para proporcionar un sistema de videovigilancia con capacidades avanzadas.

4. Integrar una plataforma web con un historial de imágenes/videos proporcionadas por el sistema.

Para conseguir que los objetivos descritos sean finalizados de forma satisfactoria se introdujeron tareas que se van a llevar a cabo durante todo el proyecto.

A continuación se pueden ver los objetivos establecidos y su grado de prioridad:

Nº	Objetivo	Prioridad
1	Comunicar la Raspberry Pi con la app Telegram.	CRITICO
2	Implementar petición 'captura instantánea'	CRITICO
3	Implementar petición 'video streaming'	CRITICO
4	Implementar módulo HC-SR501 (Sensor PIR)	CRITICO
5	Implementar módulo Buzzer	PRIORITARIO
6	Implementar nuevos componentes	CRITICO
7	Diseño de la plataforma web	PRIORITARIO
8	Comunicar plataforma web con Raspberry Pi	CRITICO
9	Test de las funcionalidades del sistema	PRIORITARIO

Tabla 1. Objetivos del proyecto

4. Metodología a utilizar

El desarrollo del proyecto ha ido avanzando satisfactoriamente conforme a lo que se especificó en la planificación que se hizo presente en el Informe Inicial. Las tareas se han ido realizando secuencialmente para poder conseguir una visión global y más precisa de lo que se va implementando.

En los *sprint* iniciales (Sprint 0 – Sprint 1), aparte de configurar correctamente la Raspberry Pi para poder utilizar la API de Telegram, se recopilaron diferentes propuestas para establecer un modo intuitivo en el que el usuario se comunicará con el sistema mediante *keyboards*. El diseño definitivo de lo que sería el menú para el usuario y así facilitar la interactuación con el sistema, se encuentra en el ANEXO A de este documento.

Al finalizar estos dos primeros Sprints, se empezó a trabajar con el Sprint 2 que consistía en implementar dos de las opciones definidas en el keyboard: *Captura instantánea* y *Video Streaming*. La opción *Captura instantánea* consiste en recibir una imagen de lo que capture en ese instante la PiCamara del sistema, en cambio, *Video Streaming* le permite al usuario poder visualizar en tiempo real lo que la PiCamera capte.
Para poder entender cómo se comunica la app Telegram al solicitar estas peticiones, entre otras, a la Raspberry Pi se detalla en el ANEXO B la arquitectura de comunicación entre estos dos sujetos.

Seguidamente se desarrollaron las tareas asignadas al Sprints 3, Sprint 4 y Sprint 5. En dichos Sprint se debían de implementar diferentes módulos electrónicos:

- Sprint 3: se implementó el Sensor PIR. Este módulo nos permite detectar cualquier presencia. Esta opción nos permitía detectar un intruso y seguidamente realizar una captura de imagen y video, y enviarla al usuario para notificarle de dicha presencia.

- Sprint 4: se implementó el Buzzer. Este módulo ejerce de sistema de alarma. En cuanto el Sensor PIR detecta una presencia, el Buzzer emite unos pitidos para alertar dicha presencia.

- Sprint 5: se realizó un análisis para posibles implementaciones de módulos extras. Entre ellos, se encuentran:

    - IR Remote: es un receptor de infrarrojas para implementar el control de un mando a distancia. Con este dispositivo de distancia nos permite activar el sistema (a una cierta distancia) sin necesidad de tener el dispositivo móvil en ese momento.

    - Servo Moto: modulo que nos permite ejercer una pequeña fuerza giratoria poder rotar la cámara cuando se le solicita dicha operación.

Para poder detallar la conexión de estos módulos mencionados anteriormente en la Raspberry Pi se encuentra esquema del circuito a en el ANEXO C de este documento. A demás, en el ANEXO D, ANEXO E y ANEXO F se puede observar el funcionamiento del sistema, el diagrama de flujo y la arquitectura de software, respectivamente, a día de hoy.

5. Planificación del proyecto

A continuación se presentan las tareas asociadas a los 6 primeros sprints del proyecto – Sprint 0, Sprint 1, Sprint 2, Sprint 3, Sprint 4, Sprint 5:

Nº	Nombre tarea	% Completado
1	Definición y diseño de los objetivos del proyecto	**100%**
2	Instalación y configuración de la Raspberry Pi	**100%**
3	Instalación y configuración del Bot Telegram	**100%**
4	Comunicación Raspberry Pi – App Telegram	**100%**
5	Implementar *keyboard* en app Telegram	**100%**
6	Implementar petición ‘captura instantánea’	**100%**
7	Implementar petición ‘video streaming’	**100%**
8	Test de cada una de las peticiones principales	**100%**
9	Implementar módulo HC-SR501	**100%**
10	Test de módulo HC-SR501	**100%**
11	Implementar módulo Buzzer	**100%**
12	Análisis de nuevos módulos	**100%**
13	Implementación de nuevos módulos	**100%**

Tabla 2. Tareas de los primeros sprints por % Completado

Como se ha podido comprobar, las tareas asociadas a los 6 primeros sprints han sido llevadas a cabo de forma satisfactoria y sin cambios en la planificación.

Por lo que respecta al sprint 5, análisis e implementaciones de nuevos módulos, la introducción de dichos nuevos módulos provoca el reajuste para calcular el coste del proyecto, como se muestra en la siguiente tabla:

Componente	Horas	Cantidad	Coste unitario	Coste total
Ingeniero de Requisitos	5 h	1	25€	125€
Diseñador	12 h	1	15€	180€
Programador	126 h	1	20€	2520€
Tester	37 h	1	20€	740 €
Raspberry Pi	0 h	1	40 €	40 €
Sensor HC-SR501	0 h	1	5 €	5 €
Buzzer	0 h	1	5 €	5 €
Pi Camera	0 h	1	21€	21€
Cables	0 h	1	7€	7€
Servo Motor	0 h	1	8€	8€
IR Remote	0 h	1	12€	12€
Sublime Text	0 h	1	0 €	0 €
phpMyAdmin	0 h	1	0 €	0 €
Xampp	0 h	1	0 €	0 €
Dropbox	0 h	1	0 €	0 €
Microsoft Office	0 h	1	0 €	0 €
		Sub Total		3.663,00€
		15% Imprevistos		550,00€
		25 % Ganancias		916,00€
		Total		**5.129,00€**

Tabla 3. Costes del proyecto actualizados

Todos los datos expuestos en este apartado pueden analizarse mejor gracias a la Imagen 1 que se encuentra en la siguiente página.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Trabajo	Nombres de los recursos	% trabajo completadc	Costo	Costo real
1		**Sistema de videovigilancia a través de una Raspberry Pi**	**79 días**	**mar 24/01/17**	**vie 12/05/17**	**180 horas**		**66%**	**4.125,00 €**	**2.722,50 €**
2		**Sprint 0**	**9 días**	**mar 24/01/17**	**vie 03/02/17**	**17 horas**		**100%**	**365,00 €**	**365,00 €**
3		Definición y diseño de los objetivos del proyecto	3 días	mar 24/01/17	jue 26/01/17	5 horas	Ingeniero de Requisitos	100%	125,00 €	125,00 €
4		Instalación y configuración de la Raspberry Pi	1 día	vie 27/01/17	vie 27/01/17	2 horas	Programador	100%	40,00 €	40,00 €
5		Recopilación de fuentes bibliograficas	5 días	lun 30/01/17	vie 03/02/17	10 horas	Programador	100%	200,00 €	200,00 €
6		**Sprint 1**	**5 días**	**lun 06/02/17**	**vie 10/02/17**	**11 horas**		**100%**	**220,00 €**	**220,00 €**
7		Instalacion y configuracion del Bot Telegram	1 día	lun 06/02/17	lun 06/02/17	1 hora	Programador	100%	20,00 €	20,00 €
8		Configurar conexión Raspberry Pi - Bot Telegram	2 días	lun 06/02/17	mar 07/02/17	4 horas	Programador	100%	80,00 €	80,00 €
9		Introducir keyboard en la aplicación Telegram	2 días	mié 08/02/17	jue 09/02/17	6 horas	Programador	100%	120,00 €	120,00 €
10		**Sprint 2**	**15 días**	**lun 13/02/17**	**vie 03/03/17**	**32 horas**		**100%**	**1.200,00 €**	**1.200,00 €**
11		Implementar petición "captura instantanea"	2 días	lun 13/02/17	mar 14/02/17	2 horas	Programador	100%	400,00 €	400,00 €
12		Implementar petición "video streaming" en "local"	5 días	mié 15/02/17	mar 21/02/17	20 horas	Programador	100%	400,00 €	400,00 €
13		Implementar petición "video streaming" en "remoto"	3 días	mié 22/02/17	vie 24/02/17	10 horas	Programador	100%	200,00 €	200,00 €
14		Pruebas y resolución de problemas	5 días	lun 27/02/17	vie 03/03/17	10 horas	Tester	100%	200,00 €	200,00 €
15		**Sprint 3**	**5 días**	**lun 06/03/17**	**vie 10/03/17**	**12 horas**		**100%**	**240,00 €**	**240,00 €**
16		Implementar modulo HC-SR501 (Sensor PIR)	3 días	lun 06/03/17	mié 08/03/17	7 horas	Programador	100%	140,00 €	140,00 €
17		Pruebas y resolución de problemas	2 días	jue 09/03/17	vie 10/03/17	5 horas	Tester	100%	140,00 €	140,00 €
18		**Sprint 4**	**5 días**	**lun 13/03/17**	**vie 17/03/17**	**12 horas**		**100%**	**240,00 €**	240,00 €
19		Implementar modulo Buzzer	3 días	lun 13/03/17	mié 15/03/17	7 horas	Programador	100%	140,00 €	140,00 €
20		Pruebas y resolución de problemas	2 días	jue 16/03/17	vie 17/03/17	5 horas	Tester	100%	100,00 €	100,00 €
21		**Sprint 5**	**15 días**	**lun 20/03/17**	**vie 07/04/17**	**36 horas**		**100%**	**720,00 €**	**720,00 €**
22		Analaisi de nuevos componenetes	5 días	lun 20/03/17	vie 24/03/17	12 horas	Programador	100%	240,00 €	240,00 €
23		Implementación de nuevos componenetes	10 días	lun 27/03/17	vie 07/04/17	24 horas	Programador	100%	480,00 €	480,00 €
24		**Sprint 6**	**5 días**	**lun 10/04/17**	**vie 14/04/17**	**12 horas**		**0%**	**180,00 €**	**0,00 €**
25		Diseño de la plataforma web	3 días	lun 10/04/17	mié 12/04/17	7 horas	Diseñador	0%	105,00 €	0,00 €
26		Diseño base de datos	2 días	jue 13/04/17	vie 14/04/17	5 horas	Diseñador	0%	75,00 €	0,00 €
27		**Sprint 7**	**10 días**	**lun 17/04/17**	**vie 28/04/17**	**24 horas**		**0%**	**480,00 €**	**0,00 €**
28		Implementación Página Inicial	3 días	lun 17/04/17	mié 19/04/17	7 horas	Programador	0%	140,00 €	0,00 €
29		Implementación Loggin	2 días	jue 20/04/17	vie 21/04/17	5 horas	Programador	0%	100,00 €	0,00 €
30		Implementación Página Historial	3 días	lun 24/04/17	mié 26/04/17	7 horas	Programador	0%	140,00 €	0,00 €
31		Pruebas y resolución de problemas	2 días	jue 27/04/17	vie 28/04/17	5 horas	Tester	0%	100,00 €	0,00 €
32		**Sprint 8**	**10 días**	**lun 01/05/17**	**vie 12/05/17**	**24 horas**		**0%**	**480,00 €**	**0,00 €**
33		Implementar comunicación	5 días	lun 01/05/17	vie 05/05/17	12 horas	Programador	0%	240,00 €	0,00 €
34		Pruebas y resolución de problemas	5 días	lun 08/05/17	vie 12/05/17	12 horas	Tester	0%	240,00 €	0,00 €

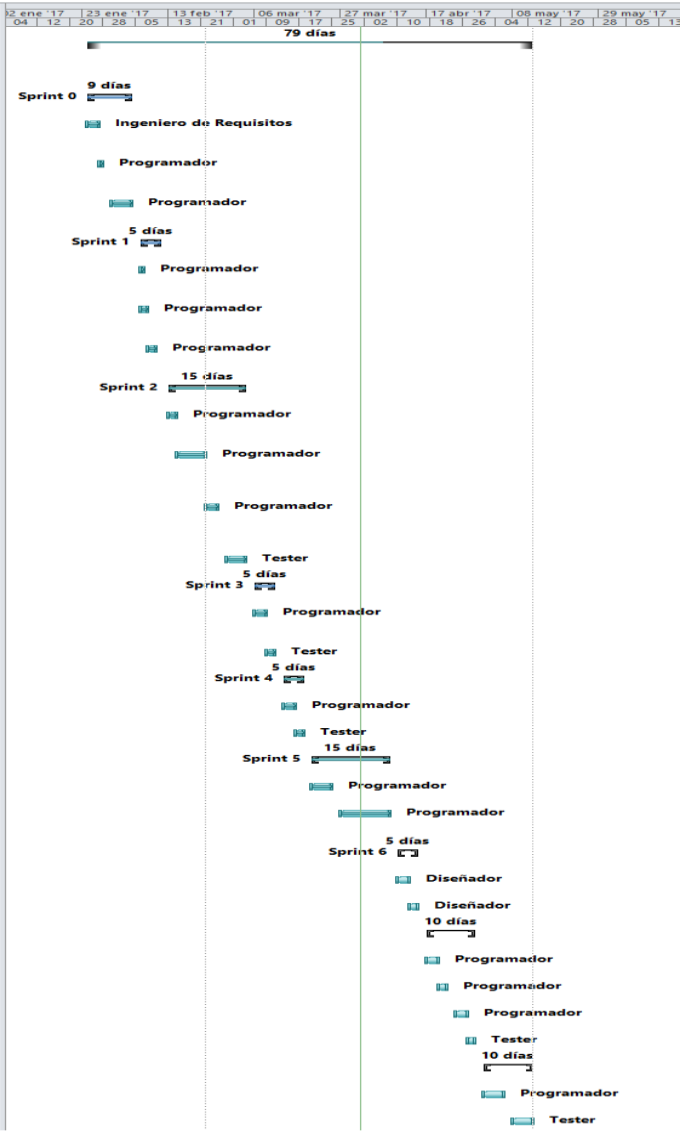


Imagen 1. Planificación del proyecto (Microsoft Project)

6. Conclusiones

El desarrollo del proyecto hasta hoy ha sido satisfactorio ya que el trabajo planificado se ha ajustado al cien por cien de lo que se planificó en un principio.

Para los últimos sprints se pretende realizar una plataforma web en el que se almacenen todas las imágenes y videos capturados por el sistema con el objetivo de tener un pequeño historial de todo lo que vaya sucediendo y acceder en cualquier momento.

7. Bibliografía y fuentes de información

Anaya Multimedia. (2007). Gestión de proyectos con MS Project

Gestió de Projecte, UAB (2016). Plantilla informe del Projecte

Gestió de Projecte, UAB (2016). Planificació del Projecte

Rumbaugh, J. (2000). El lenguaje unificado de modelado.

[1] "Bots: An introduction for developers". Accedido en marzo de 2017. Disponible: https://core.telegram.org/bots#keyboards

[2] "IR Receiver Modules for Remote Control Systems". Accedido en marzo de 2017. Disponible: http://www.vishay.com/docs/82489/tsop322.pdf

[3] "Buzzers". Accedido en marzo de 2017. Disponible: https://product.tdk.com/info/en/catalog/datasheets/ec211_sd.pdf

[4] "PIR Motion Sensor". Accedido en marzo de 2017. Disponible: https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf

[5] "SG90 Micro Servo". Accedido en marzo de 2017. Disponible: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

ANEXO A

Diseño del Keyboard

La principal comunicación mediante un servicio de mensajería es mediante el envío de mensajes. Este era el primer sistema que había implementado para poder comunicar el sistema de videovigilancia con la app Telegram como se puede observar en la Imagen 2.

Después de una serie de pruebas, se comprobó que era un sistema de comunicación algo incómodo por parte del usuario ya que debía de escribir la petición que quería solicitar. Para poder mejorar este sistema de interactuación se optó por realizar un diseño intuitivo en el que contenía un pequeño menú con una serie de opciones como en la Imagen 3. [1]

Con este nuevo sistema se evitaba la necesidad de que el usuario escribiera la petición que necesitase. Con esta mejora, el usuario solo deberá seleccionar la opción que desee con un clic y automáticamente el mensaje ser enviado y recepcionado por parte del sistema de videovigilancia.

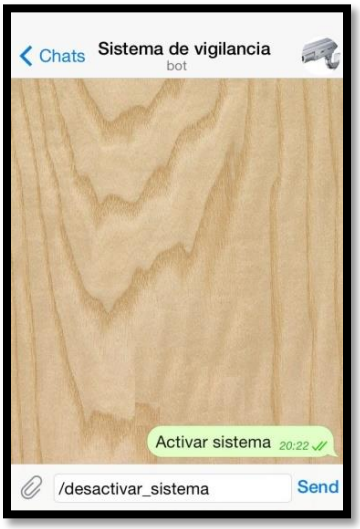


Imagen 2. Chat Telegram sin menú

Imagen 3. Chat Telegram con menú

ANEXO B

Arquitectura de comunicación

La arquitectura de comunicación entre la App Telegram y la Raspberry Pi contiene 3 sujetos fundamentales:

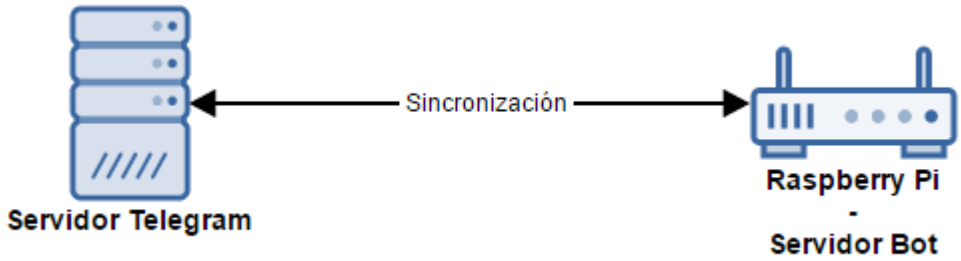
- El dispositivo móvil, el cual contendrá la app Telegram con su respectivo chat Bot, será el encargado de enviar las peticiones para que el sistema de videovigilancia realice.

- El servidor telegram es el sujeto que hace de puente entre el dispositivo móvil y el sistema de videvigilancia (Raspberry Pi).

- La Raspberry Pi, el cual se encargará de ejecutar el servidor Bot para gestionar las peticiones enviadas por el usuario.

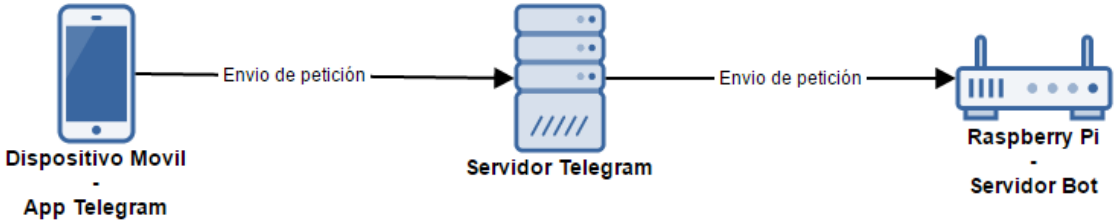
A continuación se detalla a modo general el proceso de comunicación:

En el momento en que la Raspberry Pi es puesta en marcha se inicia el servidor Bot dentro de ella. Al iniciarse el servidor Bot se sincroniza con los servidores de Telegram identificándose mediante un TOKEN.



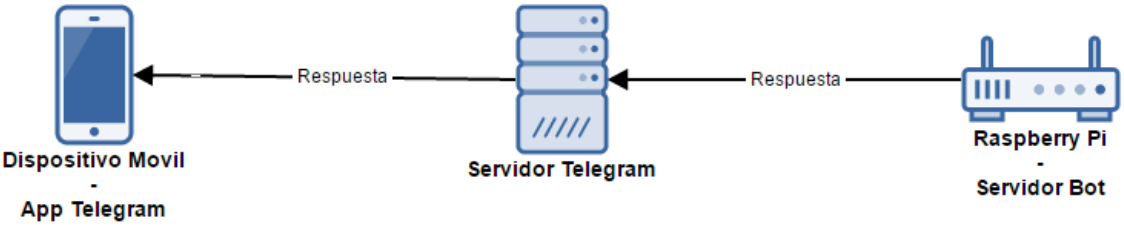
Una vez realizada la sincronización, el servidor Bot se pone a la espera de recibir peticiones por parte del chat Bot en la app Telegram. El chat del Bot, en el cual le solicitamos las peticiones, tiene definido el mismo TOKEN (como el servidor Bot) como sistema de autentificación para poder comunicarse con su respectivo servidor Bot y no con otro cualquiera.

Asi pues, cuando se envíe una petición desde el chat del Bot, esta petición será recibida por el servidor Bot.



Estas peticiones enviadas desde el chat del Bot son enviadas a los servidores de la app Telegram, el cual redirige el mensaje hacia el servidor Bot.

En el momento en el que el servidor Bot recibe la petición, procesa la operación a realizar y envía la respuesta a través de un identificador del chat Bot (chat_id).



Cada usuario de la app Telegram tiene su propio identificador. En este caso, el servidor Bot sabe a quién ha de enviar la respuesta ya que en el servidor Bot se le define el chat_id para identificar al respectivo usuario.

La secuencia de mensajes entre los 3 sujetos una vez la Raspberry Pi es puesta en marcha es la siguiente:

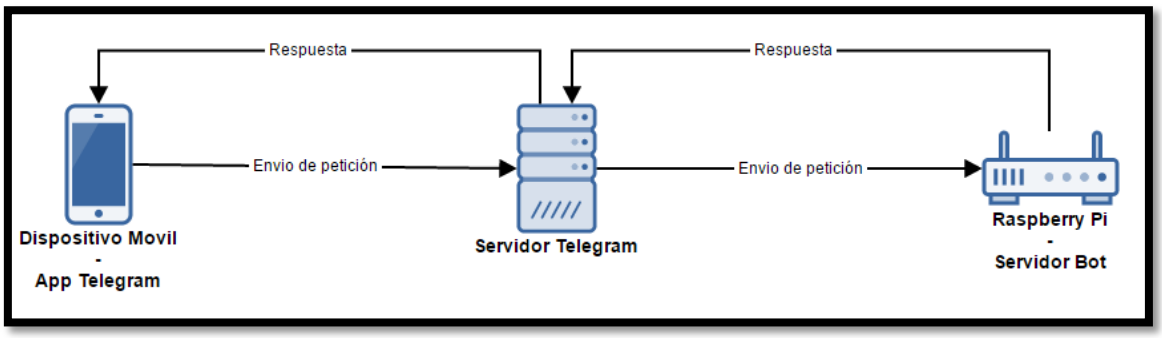


Imagen 4. Arquitectura de comunicación

ANEXO C

Circuito Esquemático

A continuación se adjunta el esquema electrónico de la Raspberry Pi con sus respectivos módulos conectados:

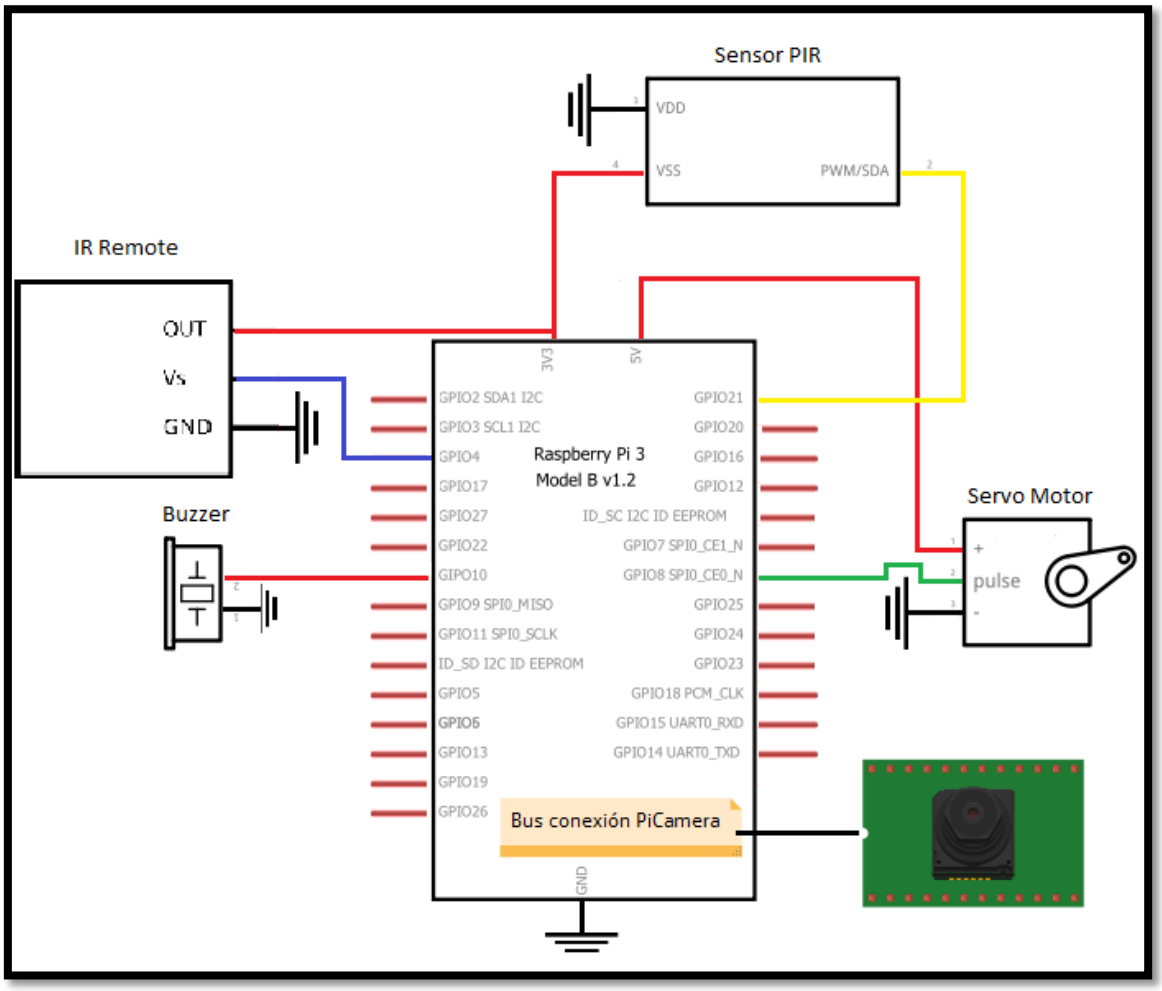


Imagen 5. Esquema electrónico Raspberry Pi

Hay un total de 5 componentes conectados a la Raspberry Pi:

IR Remote: este receptor de infrarrojos tiene tres señales. La señal de alimentación debe alimentarse a 3,3V por motivos de seguridad, ya que un voltaje mayor podría dañar el componente o incluso la Raspberry Pi. El pin 4 de la GPIO se encargará de controlar las señales infrarrojas, y por último, la señal GND que ira conectada a uno de los pin de GND de la Raspberry Pi. [2]

Buzzer: este componente se compone de dos señales. Para poder alimentarlo (que es cuando emitira los zumbidos) lo controlaremos a partir del pin 10 de la Raspberry. Es decir, cuando el pin 10 se active, dejando pasar una corriente

de 3,3V, el Buzzer efectuara un zumbido. La señal de tierra se conecta a uno de los pin GND de la Raspberry Pi. [3]

Sensor PIR: este sensor de detección de movimiento tiene tres señales. Al igual que el IR Remote, para evitar daños se ha de alimentar a 3,3V. El pin 21 de la Raspberry Pi se encargará de detectar si el sensor detecta o no una presencia y alertará al sistema. Para la señal de tierra se debe de conectar a un pin GND de la Raspberry Pi. [4]

Servo Motor: este componente, que permite control la posición del eje para poder rotar la posición de la cámara, contiene tres señales. El Servo Motor necesita ser alimentado por 5V, por lo tanto, esta señal ira conectada al pin de la Raspberry Pi que nos proporciona 5V. La señal que recibirá los pulsos y hará que el Servo Motor cambie la posición del eje estará conectada al pin 18 de la Raspberry Pi. La señal GND del componente, al igual que el resto de módulos, estará conectado a un pin GND de la Raspberry Pi. [5]

PiCamara: este componente, que ejerce de papel de cámara, está conectado mediante un bus de cinta al conector especial que hay junto al conector Ethernet de la Raspberry Pi. [6]

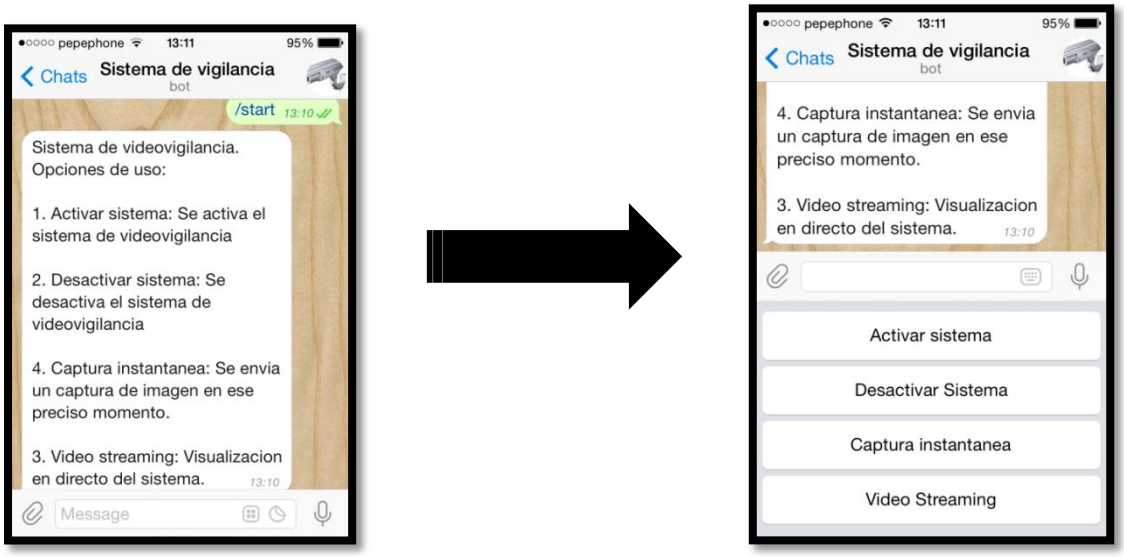
ANEXO D

Funcionamiento del sistema de videovigilancia

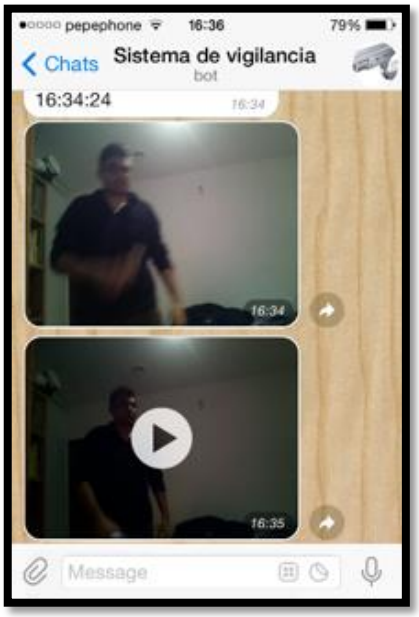
En esta sección se presenta la implementación correspondiente del proyecto hasta el sprint 5.

Una vez conectada la Raspberry Pi a la fuente de alimentación, el servidor Bot empezará a ejecutarse lo cual el usuario podrá realizar las peticiones que desee. El usuario se encontrará el chat del Bot vacío, ya que nos encontramos en la situación en la que el usuario utiliza el sistema por primera vez.

En el momento en el que el usuario de clic a la opción *start* se iniciará el sistema enviando un mensaje de inicio en el que se detalla las definiciones de cada petición que el usuario puede realizar. Automáticamente se le abrirá el menú de peticiones.



Como primera opción nos encontramos la de *Activar sistema*. Esta opción pone en alerta al sistema para cualquier presencia que detecte. En el momento en el que el sistema detecte una presencia, se le notificará al usuario detallando la hora exacta, incluyendo una captura y un video de periodo corto para poder visualizar lo que ha sucedido. Además, se emitirá un zumbido constante alertando de la presencia para intimidar al intruso.



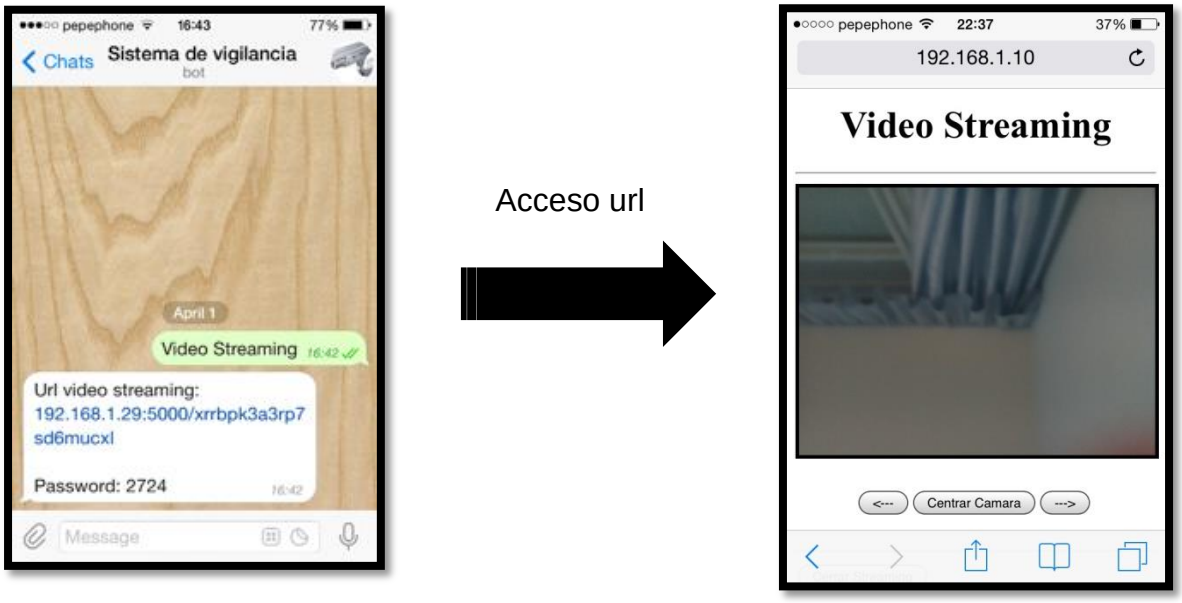
Una vez detectada la presencia el sistema retoma su tarea de detección al cabo de unos segundos.

La segunda opción le permite al usuario desactivar el sistema. Esto quiere decir, que la opción *Desactivar sistema* no detectará e informará de ninguna presencia al usuario.

Como tercera opción, el usuario solicita al sistema una captura de imagen. El sistema le enviará dicha captura:

A pesar de que el sistema de videovigilancia esté en marcha, la opción *Captura instantánea* se podrá realizar.

Como última opción, *Video Streaming* permite al usuario visualizar en tiempo real lo que esté sucediendo. Para ello, el sistema le enviará una url con la dirección para acceder al servidor que permite visualizar lo que este captando la PiCamara. Esta url será dinámica, es decir, la url siempre será distinta en cada petición a la opción *Video Streaming* para evitar que cualquier atacante descubra la url y tenga acceso en todo momento. Además, para darle una capa más de seguridad, se le asociará una password para poder acceder. Esta password será un *OTP*



En *Video Streaming* el usuario podrá rotar la cámara horizontalmente a través de los botones definidos en la interfaz para poder ver visualizar otro punto de perspectiva

Tanto la opción *Activar sistema* como la de *Desactivar sistema* pueden ser activadas desde un mando a distancia específico para este proyecto.

La posibilidad de activar/desactivar estas opciones del sistema con este dispositivo permite al usuario evitar la necesidad de tener su dispositivo móvil en ese preciso momento.

ANEXO E

Diagrama de flujo

Para comprender mejor el funcionamiento interno del sistema de videovigilancia, se mostrara a continuación el diagrama de flujo. El diagrama de flujo corresponde al funcionamiento interno de la opción *Activar Sistema* en el chat Bot. Esta opción, a diferencia del resto, es la que utiliza paralelamente todos los módulos implementados y es necesario comprender sus funcionalidades.

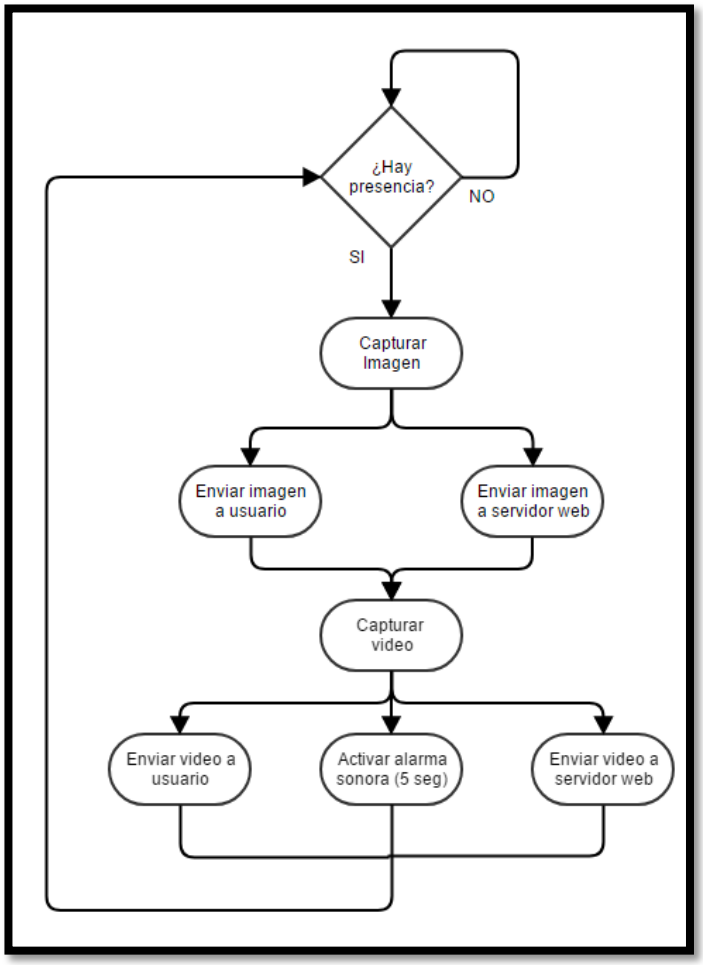


Imagen 6. Diagrama de flujo

Tal y como se puede observar en el diagrama de flujo, una vez que el usuario selecciona la opción *Activar Sistema*, el sistema empieza a detectar si existe alguna presencia. El sistema entrara en bucle a la espera de detectar una presencia hasta que la encuentre. Una vez el sistema haya detectado una presencia, se realiza una captura de imagen en ese preciso instante. Seguidamente, se ejecuta en paralelo el envió de dicha imagen tanto al servidor web (que es donde tendremos el historial de todo lo que haya captado el sistema) como a la app Telegram del usuario.

Con la ayuda de los *threads* nos permite, a pesar del que el sistema no haya enviado todavía la imagen, continuar con la siguiente acción, capturar en formato video durante 5 segundos. Al igual que en el caso de capturar la imagen, se ejecutaran tres procesos en paralelo. Dos de esos procesos corresponden al envió del video al servidor web y a la app Telegram. El tercer proceso en ejecución se trata del módulo Buzzer, que realiza una serie de zumbidos para alertar de intrusión.

Una vez que el sistema haya finalizado, retoma la función de detectar presencia.

Como se ha comentado anteriormente, los threads le permiten al sistema poder trabajar de manera paralela ahorrando tiempo de ejecución, ya que, en el caso de que el intruso fuese detectado, permite capturar imagen y video, y notificar al usuario lo antes posible.

ANEXO F

Arquitectura de Software

Para poder entender a nivel de software como está estructurado el sistema de videovigilancia a día de hoy se detalla un diagrama *UML*:

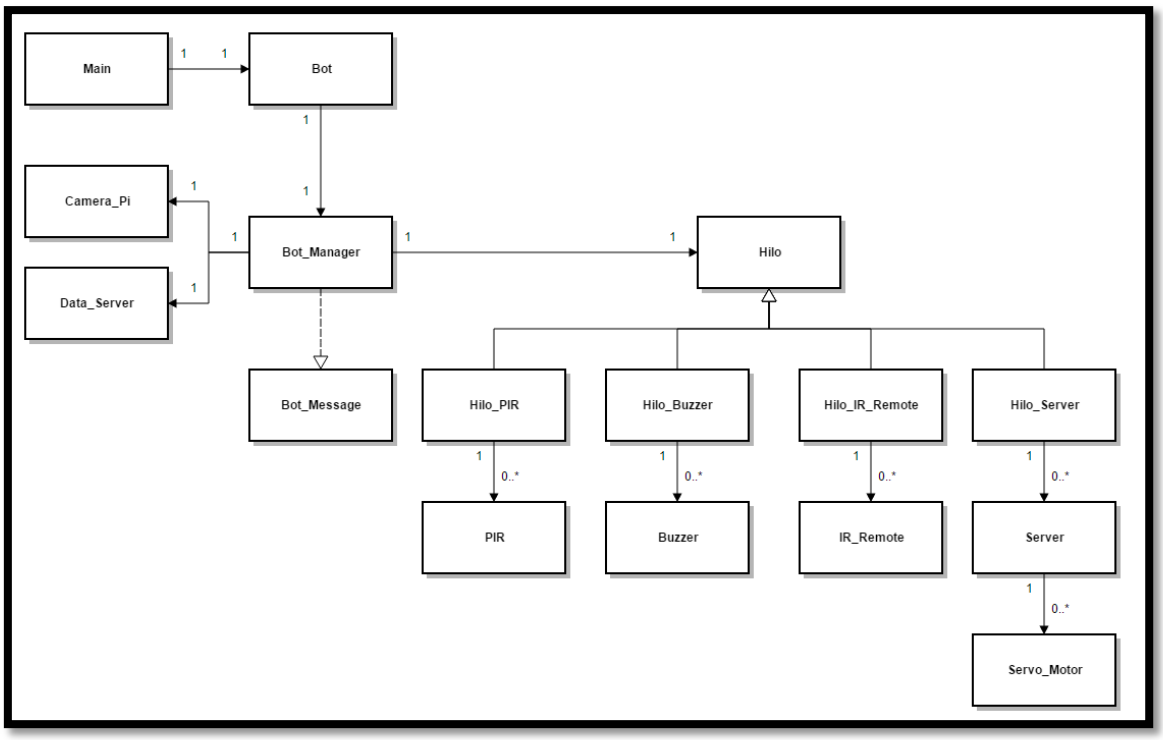


Imagen 7. Diagrama UML del sistema

Esta arquitectura de software esta basa en el paradigma de programación *Programación orientada a objetos (POO)* el cual manipula los datos de entrada con el objetivo de obtener unos datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

Para esta arquitectura de software del sistema de videovigilancia cada clase tiene sus definiciones de las propiedades y comportamiento que logran cumplir dicha funcionalidad especial.

Para poner en marcha el sistema, la **clase Main** es la encargada de que se ejecute. Es el punto de entrada que especifica dónde debe de comenzar la ejecución del programa.

Como se ha comentado en el ANEXO B, hay un sujeto que da vida al Bot, el Servidor Bot. Este servidor ha de sincronizarse con los servidores del Telegram. Para ello, la **clase Bot** es quien gestiona todo esto, dando vida al Bot mediante el funcionamiento del servidor Bot y su respectiva sincronización.

La gestión de las peticiones se encarga la **clase Bot_Manager.** Esta clase es quien recoge y gestiona las peticiones enviadas por parte del usuario. Según la

petición del usuario, la clase Bot_Manager la gestionara llamando a otras clases para obtener su objetivo, cumplir la petición del usuario.

Esta clase es el cerebro del programa ya que también gestionara el módulo PiCamera (clase Camera_Pi) y los datos del server (clase Data_Server).

La **clase Camera_Pi** es la clase que gestiona las funcionalidades principales de la cámara: capturar una imagen o captar en tiempo real a modo de video.

Por otro lado, la **clase Data_Server** es la que realiza la capa de seguridad de programa. Dentro de ella gestionará por cada petición de la opción *Video Streaming* una url y password aleatoria. Esto es importante de cara a evitar el acceso de cualquier persona que escuche la red.

Para la gestión de la respuesta por parte del Bot, como el envió de la captura de imagen, la url para el video en streaming o las alertas de presencia, entre otras, la gestionará la **clase Bot_Message**. Se encargará de enviar la respuesta al usuario.

La ejecución de diferentes módulos a la vez hace la necesidad de tener una forma de ejecutar varios procesos simultáneamente en el programa. Para ello surge la necesidad de utilizar el concepto *Thread* o *Hilo*. La **clase Hilo** es la que ayudará a poder realizar ejecuciones simultáneas. Para ello, la clase Hilo (Padre) tiene cuatro clases (Hijos) en los que harán posible la ejecución de cada módulo por separado: PIR, Buzzer, IR_Remote y Server.

La **clase Hilo_Pir** es quien ejecuta en paralelo el módulo PIR. Para ello, hace uso de la **clase PIR** que es la que se encarga de gestionar de la detección de alguna presencia.

La **clase Hilo_Buzzer** es quien ejecuta en paralelo el módulo Buzzer. La **clase Buzzer** permite gestionara el zumbido que ha de emitir si se lo ordenan.

La **clase Hilo_IR_Remote** realiza el subproceso del módulo IR_Remote. La **clase IR_Remote** es la que se encargará de dar sentido a las ondas infrarrojas recibidas por el modulo IR Remote, y realizar una acción u otra.

La **clase Hilo_Server** gestiona el proceso de ejecutar el servidor para poder visualizar en streaming lo que capté la PiCamera. Para ello es necesario **clase Server** que hará uso de un framework para la creación de una pequeña aplicación web para poder acceder y visualizar en tiempo real lo que perciba la PiCamera.

Por último, la **clase Servo_Motor** es la que se encarga de ejercer una fuerza para poder rotar la PiCamera. La clase Server contendrá un objeto de la clase Servo_Motor para poder gestionarlo en la aplicación web ya que el usuario será quien ordene la rotación.