_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CONTROL DE SISTEMA DE REALIDAD AUMENTADA ACCIONADO MEDIANTE UN GUANTE

Jorge Iranzo Rodrigo

Tutor: José Manuel Mossi García

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2016-17

Valencia, 30 de junio de 2017

Resumen

En este documento se abordan los procedimientos y conocimientos necesarios para la creación de un sistema de control para una aplicación de realidad aumentada. Este control es llevado a cabo por medio de un guante, con sensores de presión de tipo membrana en los laterales de los dedos, y un lector de tags NFC en la muñeca, de manera que el control no necesite de ningún mando o instrumento que pueda entorpecer las labores del usuario, si no que éste se integre en el equipamiento estándar del usuario. Esta lectura es llevada a cabo por un microcontrolador Adafruit feather HUZZAH, compatible con el IDE y las librerías de Arduino, e incluye wifi para la transmisión de los datos obtenidos, gracias a su chip ESP8266 integrado. Estos datos son enviados mediante UDP a una Raspberry Pi. Por otro lado, se ha desarrollado una interfaz gráfica de control que se superpone con la aplicación de realidad aumentada a controlar, ésta recibe estos datos y realiza en consecuencia una navegación por un menú. Esta aplicación, resultado de la superposición de la interfaz de control y la aplicación a controlar, es procesada en la Raspberry Pi (o cualquier computador que use una distribución basada en Linux) y posteriormente enviada a las gafas de realidad aumentada EPSON Moverio BT-200 para su visualización, utilizando el protocolo de transmisión VNC. Adicionalmente, también se han desarrollado mejoras en el sistema de captura y visualización de imágenes térmicas del proyecto "Sistema de visualización para emergencia en incendios", integrando esta aplicación junto con la interfaz de control.

Resum

En aquest document s'aborden els procediments i coneixements necessaris per a la creació d'un sistema de control per a una aplicació de realitat augmentada. Aquest control és dut a terme per mitjà d'un guant, amb sensors de pressió de tipus membrana en els laterals dels dits, i un lector de tags NFC en la nina, de manera que el control no necessiti de cap comandament o instrument que pugui entorpir les labors de l'usuari, si no que aquest s'integri en l'equipament estàndard de l'usuari. Aquesta lectura és duta a terme per un microcontrolador Adafruit feather HUZZAH, compatible amb el IDE i les llibreries de Arduino, i inclou wifi per a la transmissió de les dades obtingudes, gràcies al seu xip ESP8266 integrat. Aquestes dades són enviades mitjançant UDP a una Raspberry Pi. D'altra banda, s'ha desenvolupat una interfície gràfica de control que se superposa amb l'aplicació de realitat augmentada a controlar, aquesta rep aquestes dades i realitza en conseqüència una navegació per un menú. Aquesta aplicació, resultat de la superposició de la interfície de control i l'aplicació a controlar, és processada en la Raspberry Pi (o qualsevol computador que usi una distribució basada en Linux) i posteriorment enviada a les ulleres de realitat augmentada EPSON Moverio BT-200 per a la seva visualització, utilitzant el protocol de transmissió VNC. Addicionalment, també s'han desenvolupat millores en el sistema de captura i visualització d'imatges tèrmiques del projecte "Sistema de visualización para emergencia en incendios", integrant aquesta aplicació juntament amb la interfície de control.

Abstract

This document describes the procedures and knowledges needed for the development of a system for the control of an augmented reality application. This control is based on a glove, with membrane pressure sensors on the side of the fingers, and a NFC tag reader on the wrist. Becouse of that, there is no need of any remote control or tool which could interfere with the operator's labours, as the controller is integrated on the normal equipment of the user. This sensor's read is made by the micro-controller Adafruit feather HUZZAH, compatible with the IDE and the Arduino's libraries. Moreover, it includes WIFI for the received data's transmission, thanks to its integrated chip ESP8266. This data is send through UDP to a Raspbarry Pi. Besides that, has been developed a control graphic interface which is overlapping the application wanted to be controlled. This graphic interface receives the data and navigates through a menu. This application, result of the overlapping of the control graphic interface and the augmented reality application wanted to be controlled, is processed on the Raspberry Pi (or any other PC which uses any distribution based on Linux), and then send to the augmented reality glasses EPSON Moverio BT-200 for the viewing, using the VNC protocol. In addition, improvements have been developed on the system for capturing and viewing thermal images of the project "Sistema de visualización para emergencia en incendios", integrating this application along with the control graphic interface.

Índice

Capítulo 1. Introducción

En este documento se pretende realizar un sistema de control para una aplicación de realidad aumentada. Este sistema de control ha sido orientado para un uso en una aplicación de visualización para emergencias en caso de incendio, por lo que se ha realizado el diseño de manera que entorpezca lo mínimo posible las labores principales del agente.

Podemos diferenciar dos grandes partes en el sistema de control: por un lado, tendríamos el hardware de recepción y transmisión de los estímulos (4 botones tipo membrana, uno por dedo, y un tag NFC controlados por una placa Adafruit feather HUZZAH), y por otro lado el software ubicado en la Raspberry Pi, basado en C++, donde se ejecutará la aplicación a controlar junto a nuestro sistema de control.

Ésta última parte, a su vez, la podemos diferenciar en tres secciones: por un lado, la encargada de la recepción de estos datos de control, por otro lado, la encargada de mostrar de manera superpuesta a la aplicación a controlar una interfaz gráfica de control que proporciona un menú, y por último la encargada de transmitir la aplicación resultante de combinar la aplicación a controlar y la interfaz de control a las gafas de realidad aumentada.

Para el control de la aplicación por parte del agente, se ha elegido el uso de una placa Adafruit feather HUZZAH, de reducido tamaño, capaz de trabajar con las librerías de Arduino e incluso de ser programada a través de su IDE. Esta placa tiene la ventaja de integrar wifi mediante el uso de un chip ESP8266, por lo que el envío de los datos no necesita de ningún elemento externo. Otra ventaja de esta placa es la integración de una entrada de alimentación basada en baterías, lo que nos proporciona independencia de una alimentación externa adicional.

El sistema de control se basa en cuatro botones (uno por dedo exceptuando el pulgar), de tipo membrana, que se asocian cada uno a una acción dentro del menú diseñado. Adicionalmente se integra un lector NFC, que sirve para ocultar o mostrar la aplicación a voluntad. Para el ejemplo elegido, se coloca el tag NFC sobre el pecho del agente, de manera que para ocultar y mostrar la interfaz de control éste debe llevarse la muñeca al pecho, un gesto que no es demasiado complicado ni engorroso de hacer pero que es difícil que el agente lo realice sin desearlo.

Se ha diseñado el método de control para que esté oculto al iniciar la aplicación, así como el guante desactivado. Cuando el agente realiza el gesto de mostrar la interfaz por primera vez se guarda el tag NFC leído y se muestra la aplicación. A partir de ese momento el sistema de control no reaccionará ante la lectura de un tag NFC distinto al leído en un primer momento hasta que se desconecte su alimentación.

El método de transmisión de estos datos es UDP, a través de la red wifi.

Para la creación de la interfaz de control se utilizarán las bibliotecas GTK+ 3 y gtkmm 3. Estas bibliotecas permitirán crear una interfaz gráfica superpuesta a la aplicación a controlar, pudiendo redimensionar y ajustar los elementos de la misma a gusto del desarrollador.

Para una mayor comodidad del desarrollador, se instalarán los paquetes y librerías necesarios en el sistema operativo Ubuntu ejecutándose en una máquina virtual de VirtualBox instalada en Windows. Una vez desarrollada satisfactoriamente una versión del código, éste es compilado y ejecutado sobre Rapsbian.

Los objetivos que se abordan en este trabajo son:

- Creación de aplicaciones gráficas con superposiciones con GTK+ 3 y gtkmm 3.
- Instalación de programas y librerías necesarias para la transmisión VNC.
- Creación de un sistema de procesado mediante hilos para la ejecución de funciones en paralelo.
- Desarrollo de un hardware no intrusivo de detección de eventos.
- Desarrollo de un software de envío de los eventos detectados mediante UDP
- Desarrollo de un software de recepción de datos mediante UDP ejecutado sobre un hilo
- Realización de las conexiones de eventos y llamadas desde los hilos al programa principal
- Mejora del algoritmo de recepción y visualización de imágenes provenientes de la cámara térmica mediante el uso de un hilo de ejecución

No son objeto de estudio en este proyecto el desarrollo de la aplicación utilizada para la captura de imágenes desde la cámara térmica Seek Thermal, utilizada como ejemplo de aplicación de realidad aumentada a controlar. Tampoco es objeto de investigación la implementación de ninguna de las funciones contempladas en el menú mostrado.

Capítulo 2. Hardware de detección de eventos

2.1 Introducción

Para este proyecto se ha optado por un sistema de control mediante 4 pulsadores y la lectura de un tag NFC. Los pulsadores elegidos son de tipo membrana, como se puede observar en la siguiente figura. Utilizan configuraciones pull-down y pull-up por restricciones de la placa elegida.

Figura 2.1 – Pulsadores de membrana

La lectura del tag NFC se realizará al principio de la utilización del hardware de detección de eventos, para inicializar el control. Esta lectura se realizará con una placa SPI MRC522 como la mostrada en la siguiente figura.

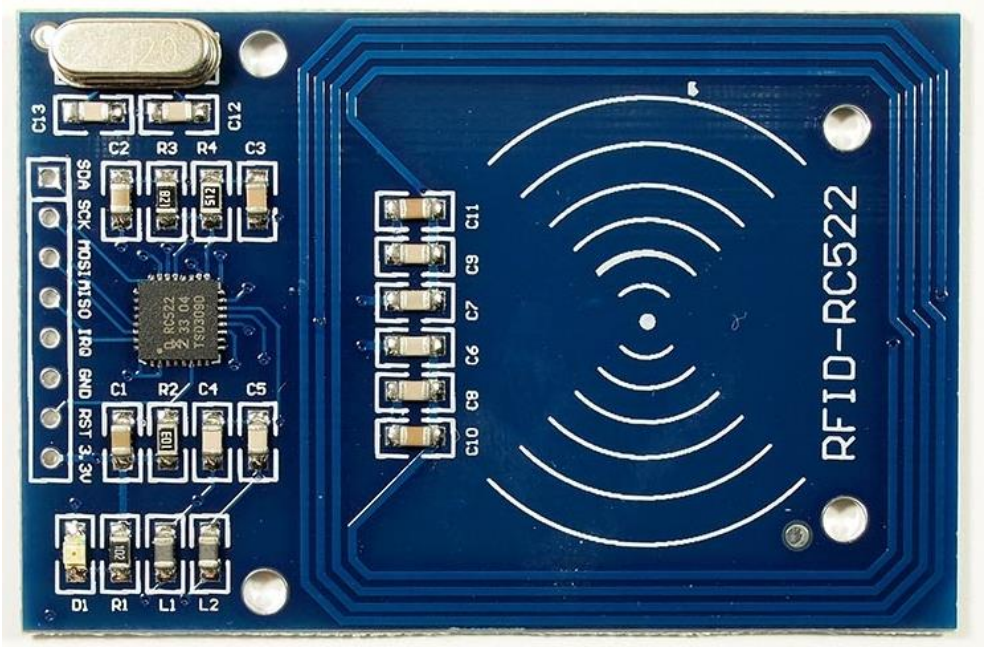


Figura 2.2 – Módulo SPI MRC5266 utilizado

Para el procesado de estas señales y el envío de la notificación de evento a la Raspberry Pi, se ha optado por una placa Adafruit Feather HUZZAH, que integra 9 pines GPIO (5 para el MRC522 y 4 para los botones de membrana). Así mismo, esta placa también integra un chip ESP8266 y una antena Wifi, por lo que es capaz de realizar conexiones con redes wifi, así como enviar y recibir paquetes TCP y UDP sin necesidad de módulos adicionales. También decir que esta placa es compatible con la IDE de Arduino, programa con el que se ha trabajado en este proyecto.

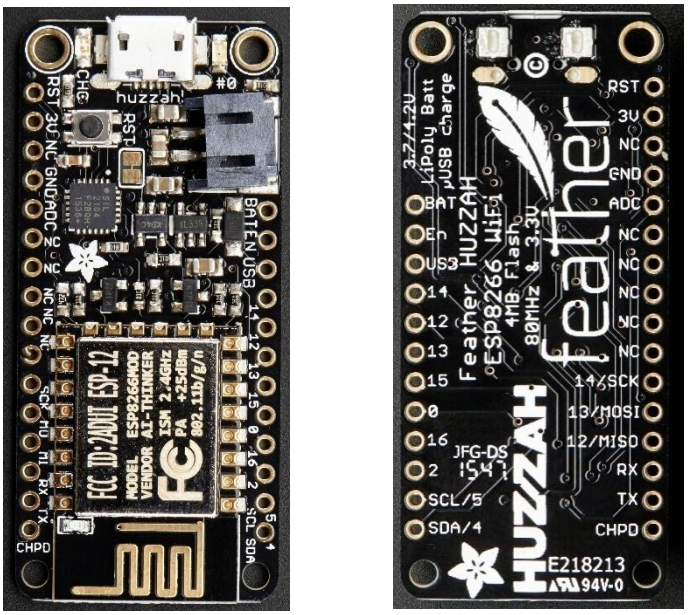


Figura 2.3 – Placa Adafruit feather HUZZAH utilizada

2.2 Conexión de los componentes

2.2.1 MRC522 (NFC)

Para la conexión SPI del MRC522 utilizaremos los pines 12(MISI), 13(MOSI), 14(SCK), 4(SDA) y el 5(SCL, aunque lo utilizaremos como pin de salida para el reset del MRC522). Realizaremos también la conexión de la alimentación del MRC522 con la proporcionada por la placa Adafruit feather HUZZAH. La conexión sigue el siguiente esquema:

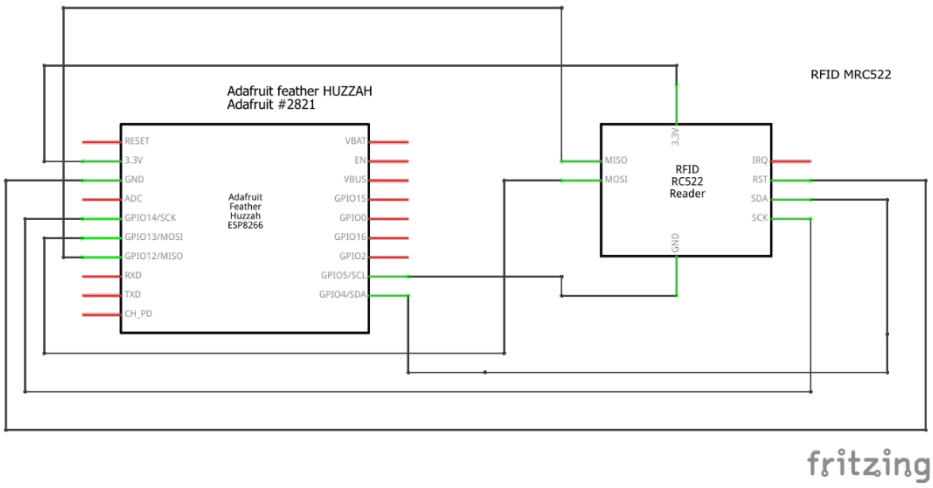


Figura 2.4 – Esquemático de conexión con el módulo MRC5266

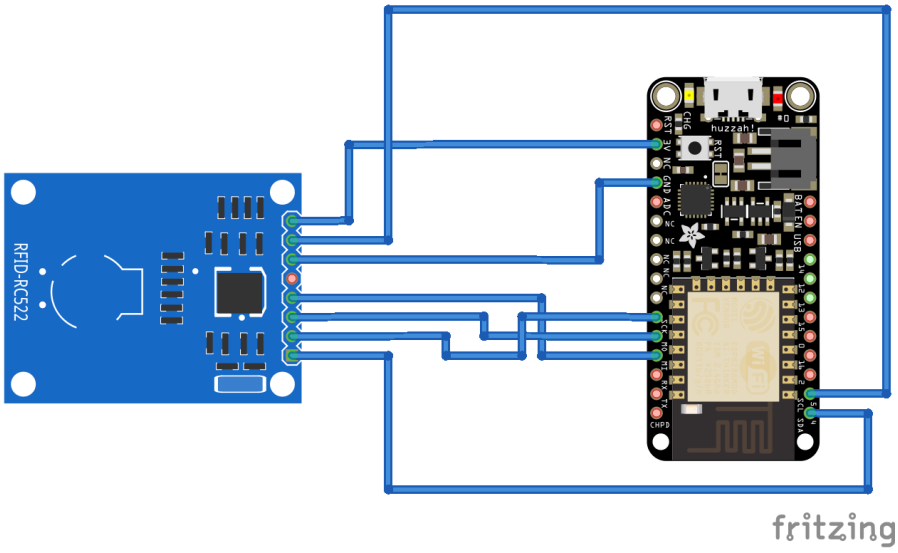


Figura 2.5 – Conexión con el módulo MRC5266

2.2.2 Botones

Por otro lado, la conexión con los botones de membrana ha sido un poco más complicada, al tener ciertas restricciones sobre los pines de la placa Adafruit feather HUZZAH. Estas limitaciones son:

- **GPIO #0,** No tiene un pull-up interno y está conectado al led rojo integrado. Este pin es usado por el ESP8266 para determinar cuándo ha de ser iniciado en modo Bootloader. Si el pin está a cero durante el arranque la placa iniciará en modo Bootloader y el programa introducido no arrancará.
- **GPIO #2**, También es usado para detectar el modo de arranque. Tiene el led azul integrado conectado a él. Tiene un pull-up interno conectado a él.
- **GPIO #15**, También es usado para detectar el modo de arranque. Tiene un pull-down interno conectado a él, no puede estar a 1 durante el arranque.
- **GPIO #16** Sirve para sacar a la placa del modo SLEEP, para ello ha de ser conectado al pin de RESET.

Al necesitar cumplir todas estas restricciones, vemos que es imposible determinar un solo modo de funcionamiento para todos los botones, puesto que siempre violaríamos alguna de estas restricciones. Se ha elegido por tanto un modelo de diseño donde dos de los botones funcionarán en modo pull-up, y otros dos funcionarán en modo pull-down.
Se han elegido resistencias de 10kΩ.

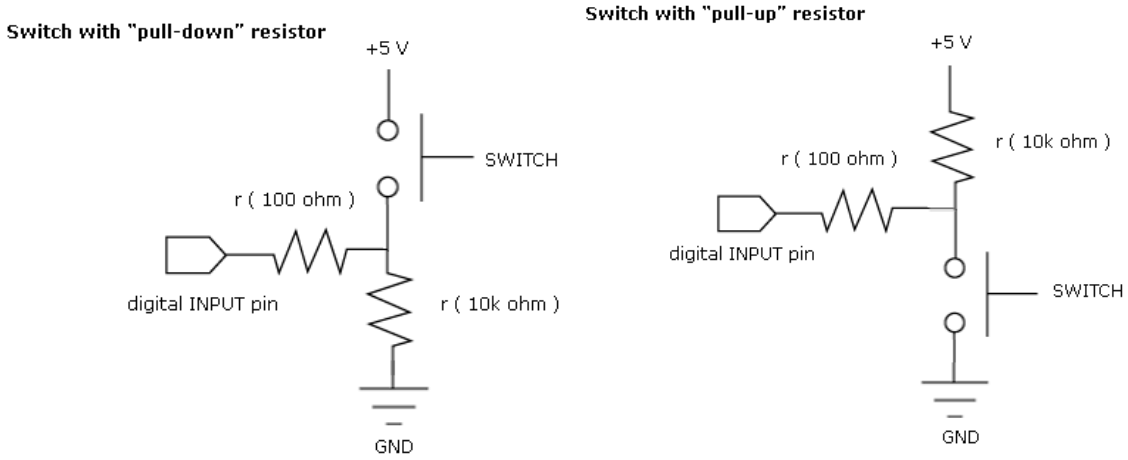


Figura 2.6 – Configuraciones Pull-Up y Pull-Down utilizadas

La conexión realizada es por tanto la siguiente:

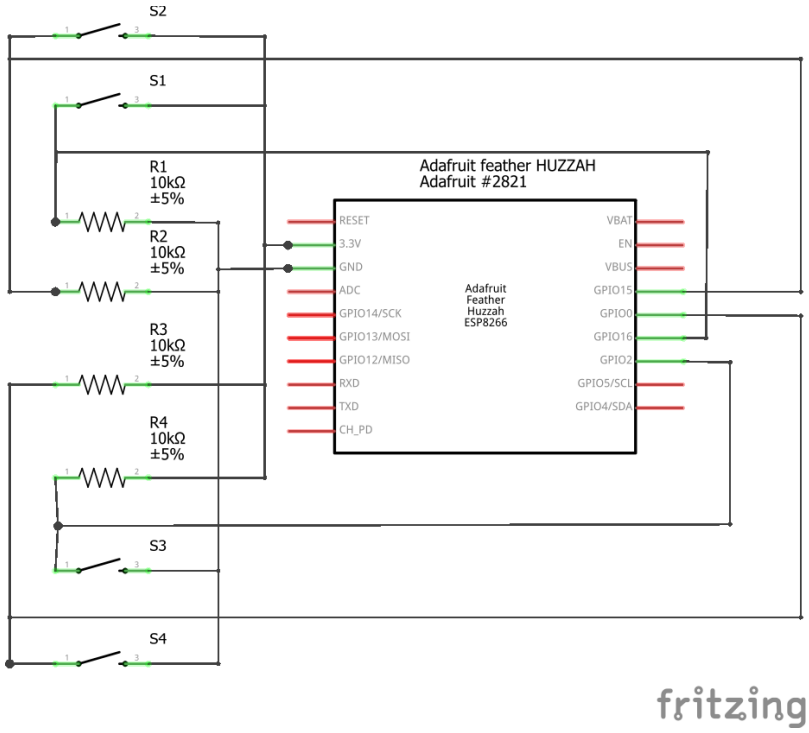


Figura 2.7 – Esquemático de conexión con los botones de membrana

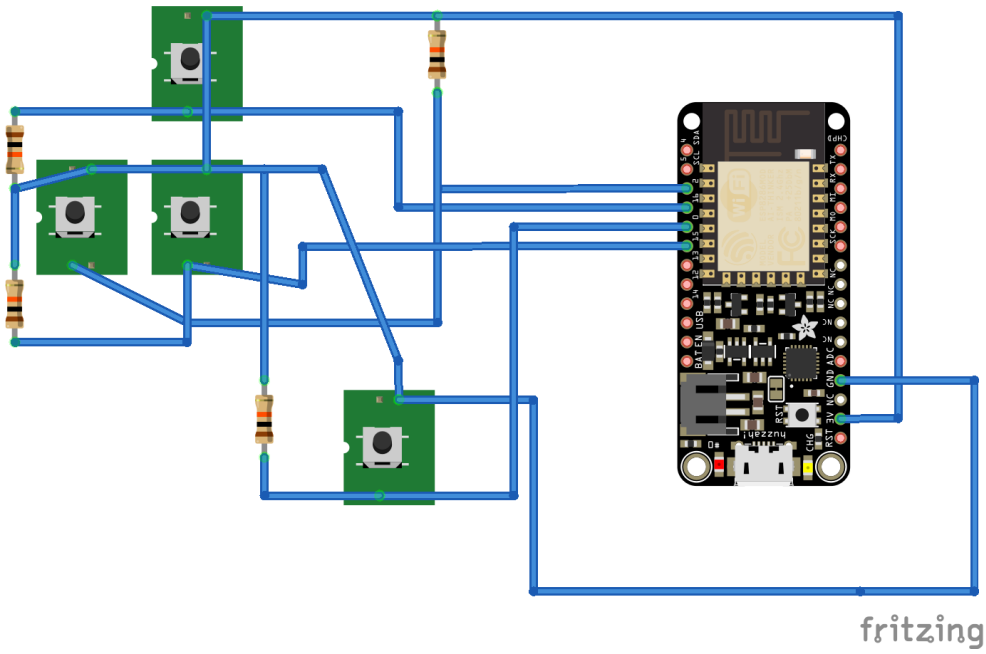


Figura 2.7 – Conexión con los botones de membrana

2.3 Eventos y estados para el envío

Se ha determinado un sistema de estados para el envío del dato. Este sistema de estados es el siguiente:

- Estado 0: No se ha detectado ningún evento
- Estado 1: Se ha detectado la pulsación del botón 1
- Estado 2: Se ha detectado la pulsación del botón 2
- Estado 3: Se ha detectado la pulsación del botón 3
- Estado 4: Se ha detectado la pulsación del botón 4
- Estado 5: Se ha detectado una etiqueta NFC (y su UID coincide con la almacenada en caso de no estar en modo aprendizaje)

Adicionalmente se ha establecido un "modo aprendizaje", que se inicia al comienzo de la ejecución.
Durante este modo, el estado no cambiará al pulsar ningún botón, sólo cambiará al detectar un tag NFC cualquiera, momento en el que estado se establecerá a 5. Cuando esto ocurra, el tag será almacenado y el modo aprendizaje será desactivado, de manera que a partir de ese momento el estado sólo se establecerá en el valor 5 si el UID del tag NFC leído coincide con el almacenado.
De esta manera, al iniciar el sistema de control, éste estará desactivado, y habrá que acercar el lector a la etiqueta NFC para iniciarlo.

2.4 Envío del evento

El evento detectado cambia la variable estado en cada iteración del bucle, pero este estado no es enviado en cada iteración del bucle para evitar un gasto innecesario de energía y ciclos del microprocesador. El evento sólo es enviado si éste cambia de una iteración a la siguiente y es distinto de 0. De esta manera el evento sólo es enviado cuando se pulsa un botón o cuando se lee la etiqueta NFC, evitando así envíos innecesarios.

El envío se realiza sin ninguna codificación, pues el estado ya identifica al evento. Simplemente es enviado el estado.

El envío se realiza mediante UDP. Para ello antes el chip ESP8266 integrado en nuestra placa Adafruit feather HUZZAH debe establecer una conexión con la red wifi.

Los parámetros demandados son introducidos al principio del código, siendo éstos el SSID de la red, la contraseña correspondiente, la IP a la que enviar los datos y el puerto de recepción de dicha IP.

2.5 Construcción del dispositivo de control

El dispositivo de control elegido es un guante, donde se han integrado todos los elementos descritos anteriormente. Los botones han sido recortados y cosidos a los laterales de los dedos, mientras que su conexión se ha ido cosiendo desde su ubicación hasta la zona de la muñeca, dejando el cable con la suficiente holgura para permitir un movimiento natural de la mano. Estas conexiones se han llevado hasta la zona de la muñeca, lugar donde se ha habilitado una extensión del guante, consistente en una muñequera unida a él, donde se han integrado todas las interconexiones pertinentes, se ha hospedado tanto la placa Adafruit feather HUZZAH como la MRC522, y se ha instalado la batería que alimenta al conjunto, integrando también un interruptor accesible que la activa o desactiva. La batería puede cargarse a través de la propia placa Adafruit feather HUZZAH, simplemente alimentándola por USB mientras se encuentra conectada. En las siguientes figuras se ilustra el dispositivo de control diseñado.

Figura 2.8 – Guante de control en uso

Figura 2.9 – Diseño del guante de control

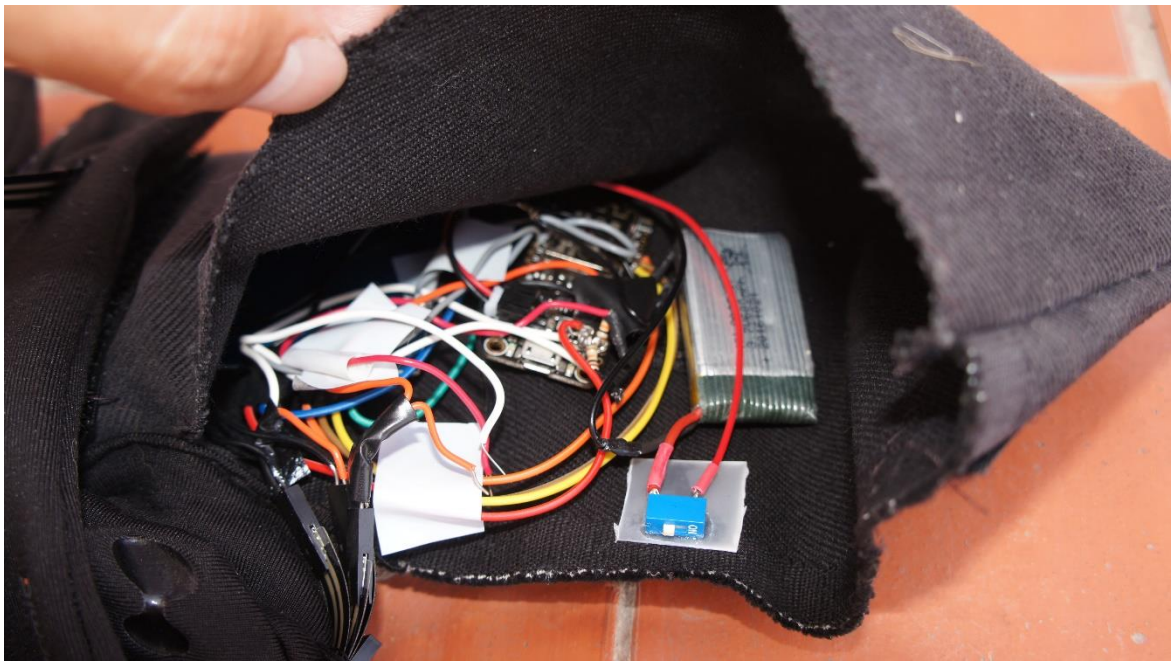


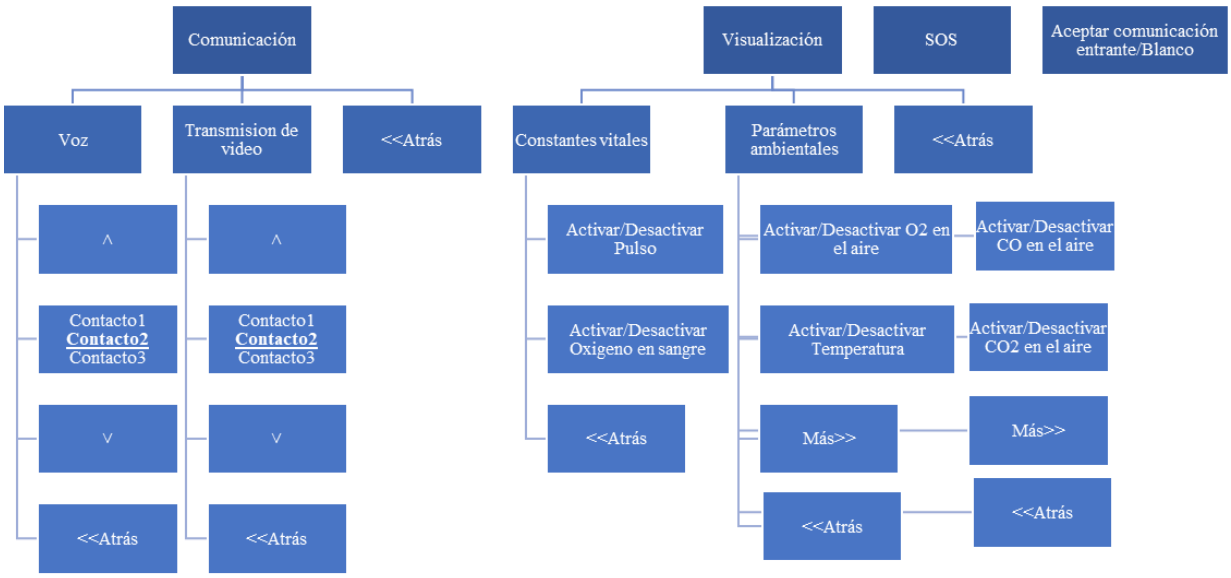
Figura 2.10 – Conexiones del guante de control

Capítulo 3. Desarrollo de la aplicación de control

3.1 Introducción

Nuestra aplicación a desarrollar es una aplicación de control de un sistema de realidad aumentada. En este caso se ha utilizado esta aplicación de control sobre una aplicación de visualización para emergencia en incendios, consistente en una visión termográfica del entorno.

Del mismo modo, se ha añadido un menú no invasivo y superpuesto a la aplicación a controlar, donde el agente que va a entrar en acción en una situación de emergencia puede navegar de manera rápida y eficaz para llevar a cabo distintas acciones. Aclarar de nuevo que dichas acciones sirven como ejemplo, y no son objeto de estudio en este documento, ya que el objetivo era representar una interfaz de control para una aplicación de realidad aumentada. El menú que se ha elegido como ejemplo se puede observar en la siguiente figura:

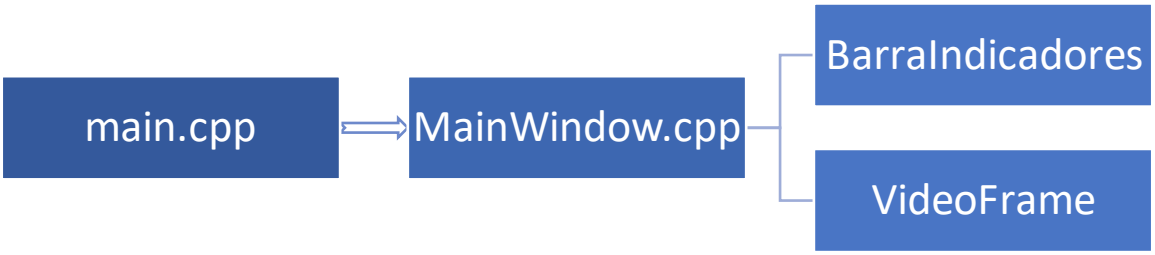


Esquema 3.1 – Estructura del menú

Este menú deberá adoptar una forma intuitiva para el usuario, desplegándose superpuesto a la aplicación a controlar, de manera semitransparente y en una esquina para no interferir en ella, y debe poder ser habilitada y deshabilitada de manera rápida mediante el gesto correspondiente al NFC.

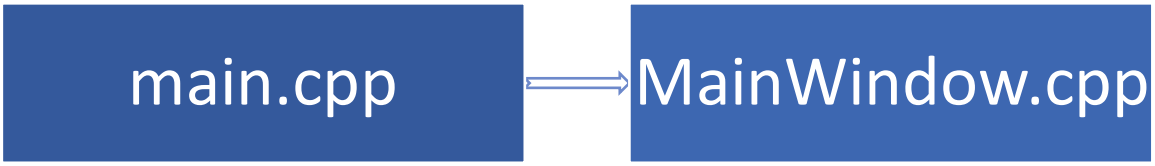
3.2 Estructura de la aplicación

3.2.1 Estructura general



Esquema 3.2 – Estructura general de la aplicación

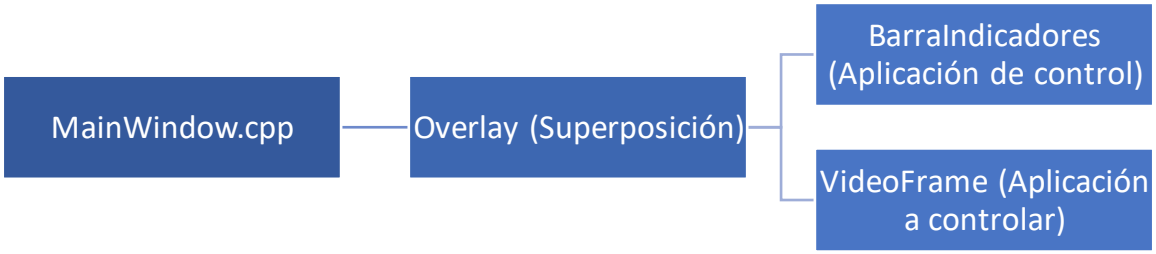
3.2.2 main.cpp



Esquema 3.3 – Estructura main.cpp

La función main.cpp es la encargada de crear la ventana principal de ejecución y desde donde se crean y ejecutan el resto de funciones. Esta función es la principal y contiene el resto de clases a través de la clase MainWindow. En esta función main.cpp se crea un objeto de la clase MainWindow, que contiene el resto de clases que constituyen la interfaz gráfica. A través de la llamada a la función run(MainWindow) se pone en marcha la aplicación creada.

3.2.3 MainWindow.cpp



Esquema 3.4 – Estructura MainWindow.cpp

La clase MainWindow se compone de un archivo fuente (MainWindow.cpp) y un archivo cabecera (MainWindow.h).

En MainWindow.h se declaran 3 componentes de MainWindow.cpp, que serán los que conformen nuestra aplicación.

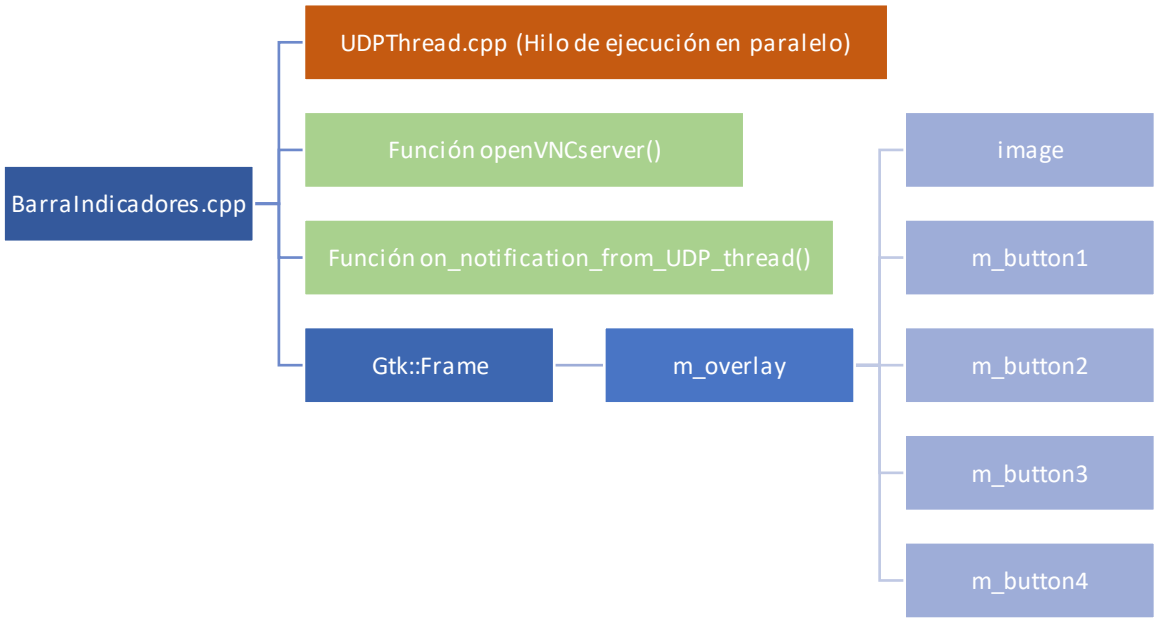
- Overlay: Se crea un objeto de la clase Gtk::Overlay. Este es un objeto que puede tener infinitos hijos contenidos en él. Todos sus objetos hijos se superponen en el mismo espacio, lo que nos permite mostrar el menú y la aplicación de manera superpuesta.

- VideoFrame: Se crea un objeto de la clase VideoFrame. Esta es la aplicación a controlar. En este caso se trata de una visualización de una imagen térmica, por lo que se declara de tipo VideoFrame para tal efecto.

- BarraIndicadores: Se crea un objeto de la clase BarraIndicadores. Esta es nuestra aplicación de control. Esta aplicación es en la que se centra el estudio en este documento.

Gracias a esta estructura, la aplicación de control desarrollada se puede integrar en cualquier aplicación de manera sencilla, sólo hace falta sustituir la aplicación VideoFrame por cualquier otra para que sirva la misma estructura desarrollada.

3.2.4 BarraIndicadores.cpp



Esquema 3.5 – Estructura BarraIndicadores.cpp

La clase BarraIndicadores también se divide en dos archivos, el que contiene el código fuente (BarraIndicadores.cpp), y el que contiene las cabeceras (BarraIndicadores.h)

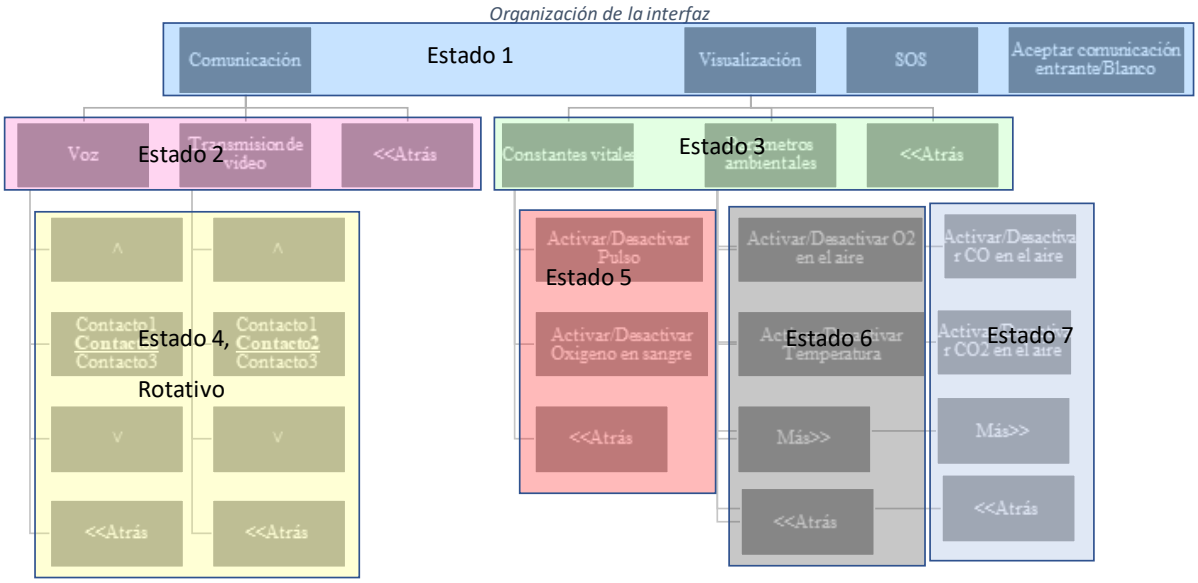
El archivo BarraIndicadores.h declara los objetos que se utilizan en BarraIndicadores.cpp. Estos objetos son:

- Gtk::Frame: El contenido de BarraIndicadores se engloba dentro de un objeto Frame. Este objeto contiene al resto de la aplicación BarraIndicadores
    - m_overlay: objeto de tipo Gtk::Overlay que superpone los elementos del menú para que las opciones queden superpuestas a la imagen.
        - image: objeto de tipo Gtk::Image que contiene la imagen de una mano, que sirve al usuario para saber con mayor facilidad qué hacen los controles del guante.
        - m_button1-4: Estos cuatro objetos son objetos de tipo Gtk::Label, e indican al usuario qué acción realizan los controles del guante en cada momento. El texto que muestran cambia al navegar por el menú.

- openVNCserver(): Método de BarraIndicadores.cpp que se ejecuta al comienzo e inicia el servidor VNC en la Raspberry Pi para poder visualizar la aplicación en las gafas de realidad aumentada.
- on_notification_from_UDP_thread(): Método de BarraIndicadores.cpp que se conecta a un objeto tipo Glib::Dispatcher. Se encarga de recibir las notificaciones desde el thread de recepción de datos UDP (Más adelante se profundizará en este thread), y poner en marcha el mecanismo de decisión de estado de la máquina de estados del programa BarraIndicadores.cpp.
- UDPThread.cpp: Aplicación encargada de recibir datagramas UDP, leer su contenido y notificar a través del Dispatcher a la aplicación BarraIndicadores, proporcionándole el dato recibido. Se profundizará en este thread más adelante.

La aplicación BarraIndicadores se encarga de conectar y mostrar todos los elementos que contiene. Al objeto m_overlay se le ha aplicado un margen superior e izquierdo de manera que éste sólo se muestre en la esquina inferior derecha. Al objeto image se le ha ajustado para que éste se alinee a la derecha y se mantenga en el centro en el eje vertical, mientras que a los botones se les ha ajustado una alineación vertical dependiendo de a qué altura se quiere mostrar, y un margen a la derecha acorde con el tamaño de la imagen.

El sistema de funcionamiento interno de la aplicación se basa en un sistema de máquina de estados, donde cada acción, menú o submenú es representado por un estado. Cada estado ejecuta un método diferenciado, de manera que añadir nuevos menús o acciones sólo requiere añadir el método con la acción o menú a desarrollar, y ajustar el mecanismo de la máquina de estados para ubicarlo en el sitio deseado dentro del organigrama de acciones, menús y submenús. La organización para el menú elegido como ejemplo es la que se muestra en la siguiente figura.



Esquema 3.5 – Estructura menú dividido en estados

La decisión de un nuevo estado de la máquina de estados es un método del programa, que es llamado cada vez que éste recibe un dato.

Capítulo 4. Hilo de ejecución de recepción de datos por UDP

4.1 Introducción

Un hilo de ejecución (o thread en inglés), es un proceso que puede ejecutarse en paralelo a otro programa u otro hilo. Los sistemas operativos como Linux, Windows o Mac OS, incorporan de manera nativa un sistema de ejecución por hilos que les permite ofrecer multitarea y realizar funciones que necesitan de ejecuciones paralelas. De esta manera, RaspBian, el sistema operativo utilizado en Raspberry Pi para este proyecto, es capaz de realizar ejecuciones en paralelo.

En este proyecto vamos a hacer uso de esta capacidad para la lectura de datos de un puerto de la interfaz de red. Al ser la operación de lectura de los datos una operación bloqueante (Es decir, no se puede ejecutar ninguna otra operación mientras el puerto está a la escucha), no se puede compatibilizar con la ejecución de un programa gtk, ya que éste no podría refrescar la imagen mostrada hasta la recepción de un dato.

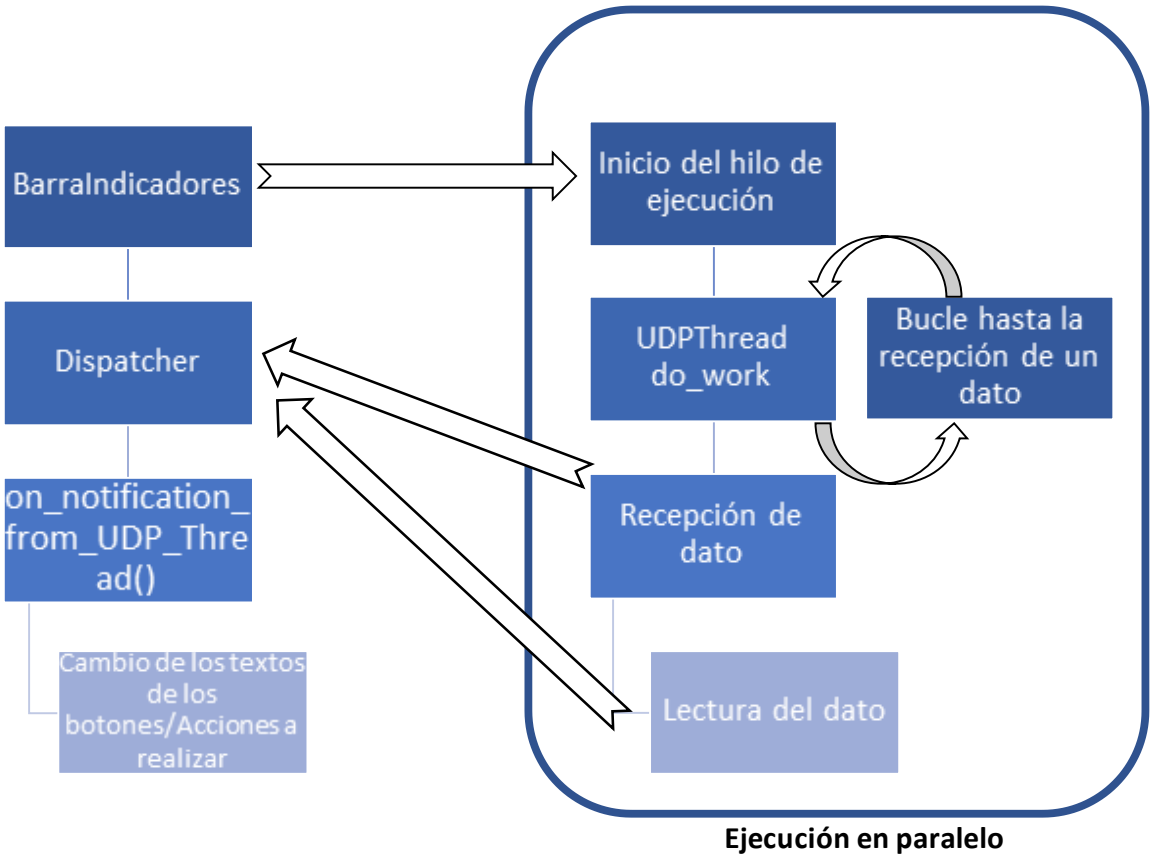
Para ello vamos a lanzar desde nuestro programa BarraIndicadores.cpp, un hilo de ejecución con un servidor UDP, que se mantendrá a la escucha ante los paquetes UDP enviados a el puerto indicado. En caso de que este hilo reciba un paquete, éste es leído y notificado al programa principal, proporcionándole el dato recibido.

4.2 Dispatcher

Un Dispatcher es un objeto de tipo Glib::Dispatcher, que conecta a un hilo de ejecución con otros hilos o programas. Éste objeto es declarado en BarraIndicadores.cpp, el programa que llama a la ejecución del hilo de recepción UDP.

Este objeto es conectado a un método on_notification_from_UDP_thread(), que será el que se ejecute cuando se notifique al objeto Dispatcher.

4.3 Estructura del hilo



Esquema 4.1 – Esquema de funcionamiento del hilo de recepción de datos por UDP

Capítulo 5. Transmisión de la aplicación por VNC

5.1 Introducción

VNC es un software de escritorio remoto, que permite el envío de la imagen mostrada en la pantalla del terminal servidor a un terminal cliente, así como el control del terminal servidor por parte del terminal cliente.

Se ha hecho uso de esta tecnología para transmitir la imagen de la aplicación resultante de superponer la aplicación de realidad aumentada a la aplicación de interfaz gráfica de control a unas gafas de realidad aumentada EPSON Moverio BT-200, gobernadas por el sistema operativo Android 4.0.4.

En este caso no vamos a realizar control del terminal servidor desde el terminal cliente, que se limitará a visualizar la aplicación, ya que el control de la misma se realizará mediante el hardware de control.

VNC también permite establecer una contraseña para proteger la conexión, pero en este caso no se utiliza dicha funcionalidad puesto que la seguridad de la comunicación ya viene dada por la red Wifi, que es una red protegida por una contraseña WPA2 que sólo compartirán las gafas Epson Moverio BT-200, el hardware de control y la propia Raspberry Pi.

5.2 Método openVNCserver de BarraIndicadores.cpp

Este método, ejecutado al inicio de la aplicación, es el encargado de habilitar la Raspberry Pi como servidor VNC, así como de servir la aplicación deseada a través de este servicio. Esto lo realiza mediante dos programas externos que deberemos haber instalado previamente, xdotool y x11vnc.

Mediante el comando popen, ejecutamos los comandos deseados para la ejecución de estas dos aplicaciones desde nuestra aplicación BarraIndicadores.cpp.

5.2.1 xdotool

Mediante esta herramienta externa, podemos realizar un compendio de funciones Xlib, entre ellas la que nos interesa, getwindowfocus.

Esta instrucción nos devuelve el ID de la ventana en primer plano en el momento de la ejecución del comando, necesario para que el servidor VNC sea iniciado mostrando sólo dicha ventana, y no todo el escritorio.

El comando utilizado es:

xdotool getwindowfocus

5.2.2 x11vnc

Este programa es una distribución libre de servidor VNC, y, al igual que anteriormente se hizo con xdotool, es llamado a través de la función popen.

Para la ejecución de este comando, utilizamos varios argumentos que ajustan el comportamiento de nuestro servidor VNC a nuestras necesidades. Estos argumentos utilizados son:

- -viewonly: Limita la conexión VNC a sólo visualización por parte del cliente. De esta manera se impide que el cliente pueda manejar el servidor remotamente.
- -scale 0.7x0.7: Reduce la resolución de la aplicación en un 70%, de esta manera la interfaz puede ser mostrada sin problemas en las gafas de realidad aumentada Epson Moverio BT-200, y además al reducir la resolución del envío ganamos velocidad de transmisión, lo que se traduce

en un retardo aún menor para nuestra visualización en las gafas, a costa de una ligera reducción en la calidad de la imagen.

- -forever: Indica al programa x11vnc que no debe cerrar el puerto tras la desconexión de un cliente, si no que el servicio debe seguir ejecutándose, de manera que tras una desconexión del cliente que visualiza la aplicación, no es necesario reiniciar la misma, el cliente se puede reconectar de nuevo como hizo en una primera instancia.
- -wait 0.25: Establece el tiempo entre refrescos de la pantalla en el servidor. Se ajusta a un valor muy bajo para eliminar retrasos en la transmisión.
- -defer 0.25: Establece el tiempo entre envíos de la imagen al cliente. Se ajusta a un valor muy bajo para eliminar retrasos en la transmisión.
- -q: Ajusta el programa a su modo silencioso, de manera que no satura la línea de comandos con información sobre el estado de la conexión
- -bg: Ajusta el programa a su modo de ejecución en segundo plano, dejando a la aplicación principal funcionar en primer plano.
- -id + [ID]: Este argumento seguido de la id de una ventana en ejecución en el servidor, le indica al programa que la imagen a enviar por VNC se corresponde sólo con la correspondiente a la mostrada en la ventana con el id dado. De esta manera se limita así que se transmita únicamente la aplicación, y no todo el escritorio.

El comando utilizado es:

x11vnc -viewonly -scale 0.7x0.7 -forever -wait 0.25 -defer 0.25 -q -bg -id (+ id de la ventana obtenida anteriormente)

Capítulo 6. Visualización en las gafas de realidad aumentada

6.1 EPSON Moverio BT-200

Las gafas elegidas para la visualización de la aplicación son las EPSON Moverio BT-200, unas gafas de realidad aumentada con buenas prestaciones y asequibles.

Estas gafas poseen dos pequeñas pantallas TFT de 0.42 pulgadas con resolución 960x540p 60Hz, cuyo contenido se superpone de manera semitransparente con la visión normal del usuario, gracias a que estas pantallas son translúcidas. El efecto para el usuario es el de estar viendo una pantalla de 80 pulgadas a 5 metros de distancia.

Estas gafas están impulsadas por un procesador TI OMAP 4460 1.2Ghz Dual Core, junto a 1Gb de memoria Ram y 8Gb de memoria Rom. También posee wifi, lector de tarjetas microSD y Bluetooth 3.0.

El sistema operativo de las gafas es Android 4.0.4, el cual se puede controlar mediante un panel táctil ubicado en un módulo conectado a las gafas mediante un cable. En dicho módulo también se hospeda el procesador, la memoria Ram, la memoria Rom, así como el resto de componentes de las gafas exceptuando las pantallas. En este módulo también se incluye una batería de 2720mAh que ofrece un tiempo de uso de las gafas de hasta 6h.

Figura 6.1 – Gafas de realidad aumentada EPSON Moverio T-200

6.2 VNC Viewer

Mediante la instalación en las gafas de la aplicación VNC Viewer a través de su instalador APK, podemos visualizar el contenido mostrado por cualquier servidor VNC.

Ahora solo queda conectar las gafas al wifi deseado, y ejecutar la aplicación VNC Viewer utilizando la IP de la Raspberry Pi y el puerto 5900.

Capítulo 7. Mejora de la aplicación de visualización térmica

7.1 Introducción

Para el desarrollo de este trabajo se ha orientado la interfaz de control a una aplicación concreta: un sistema de visualización para emergencia en incendios, basada principalmente en la visualización de una imagen térmica del entorno. Durante el desarrollo de este proyecto, se ha intentado llevar esta aplicación a un entorno en realidad aumentada, ofreciendo además una plataforma y una interfaz de control para futuras acciones implementadas en esta aplicación conjunta.

Además de simplemente realizar una integración de este sistema de control, y proporcionar también un método de visualización de toda la aplicación en unas gafas de realidad aumentada, se ha realizado mejoras en la aplicación de visualización para emergencia en incendios.

El programa de visualización para emergencia en incendios del que se partía en el inicio de este proyecto tenía una limitación de rendimiento en su versión orientada a la visualización mediante una ventana gtkmm, obteniendo una tasa de fotogramas por segundo inferior a un fotograma por segundo, una tasa insuficiente para su uso. Por ello se han realizado mejoras en el mismo con el objetivo de aumentar esa tasa de fotogramas por segundo a los que realmente es capaz de ofrecer la cámara.

7.2 Método de visualización para emergencia en incendios. Cámara Seek Thermal.

El método de visualización para emergencia en incendios se basa en la utilización de una cámara Seek Thermal Compact de la compañía Seek Themal, Inc. Esta cámara tiene como ventaja ante otras alternativas presentes en el mercado su alta resolución, su calibrado automático, y su tamaño compacto. Está originalmente diseñada para ser utilizada en dispositivos móviles, por lo que utiliza una conexión microusb, orientado a su conexión a un dispositivo Android.

Cabe destacar que esta cámara tiene una tasa de tan sólo 8 fotogramas por segundo, puesto que entra en la categoría de cámaras térmicas orientadas a un uso exclusivamente civil. Esto es así porque Seek Themal, Inc., la empresa fabricante, es una empresa ubicada en Santa Bárbara, California, EEUU.

La legislación de EEUU especifica que las cámaras térmicas que superen los 9 fotogramas por segundo son consideradas como cámaras de uso civil o militar, mientras que las que no superan dicha tasa de fotogramas por segundo son consideradas cámara de uso exclusivamente civil. De igual manera, la legislación vigente americana establece que las cámaras de uso civil o militar no puedan ser exportadas sin una autorización del departamento de comercio de EEUU. A pesar de esa aparente falta de velocidad en la captura, el resultado de la visualización es lo suficientemente fluido como para ofrecer una experiencia satisfactoria, cumpliendo el objetivo planteado.

La cámara se puede observar en la siguiente figura.

Figura 7.1 – Cámara Seek Thermal Compact

7.3 Esquema del algoritmo mejorado de captura de imágenes

Para la mejora del algoritmo de captura de imágenes, se ha optado por utilizar un sistema basado en el uso de threads. El problema del algoritmo de visualización de imágenes original residía en que el bucle infinito encargado de recibir los fotogramas de la cámara era el mismo que el encargado de refrescar la imagen mostrada en pantalla, es decir, de manera lineal, el programa primero recibía la imagen de la cámara, y hasta que ésta no había sido recibida y procesada por completo, no se iniciaba el método encargado de refrescar la imagen mostrada en pantalla.

Para solucionar este problema, la mejora desarrollada ha consistido en la separación del programa de recepción y visualización de la imagen en dos threads distintos, el principal, la aplicación gtkmm, encargada de mostrar la imagen y de refrescar continuamente la pantalla, y el thread de recepción de fotogramas desde la cámara, encargado de recibir y procesar de manera independiente y paralela los fotogramas provenientes de la cámara.

A continuación, se explicará con detalle la labor realizada por cada una de las partes del programa.

7.3.1 VideoFrame.cpp

Esquema 7.1 – Estructura VideoFrame.cpp

De la misma manera que el resto de programas desarrollados, el programa VideoFrame consta de un archivo VideoFrame.h donde se declaran los objetos a utilizar, mientras que en el archivo VideoFrame.cpp se establece las relaciones entre dichos archivos y se lleva a cabo la visualización de los mismos.

La estructura consta de los siguientes apartados:

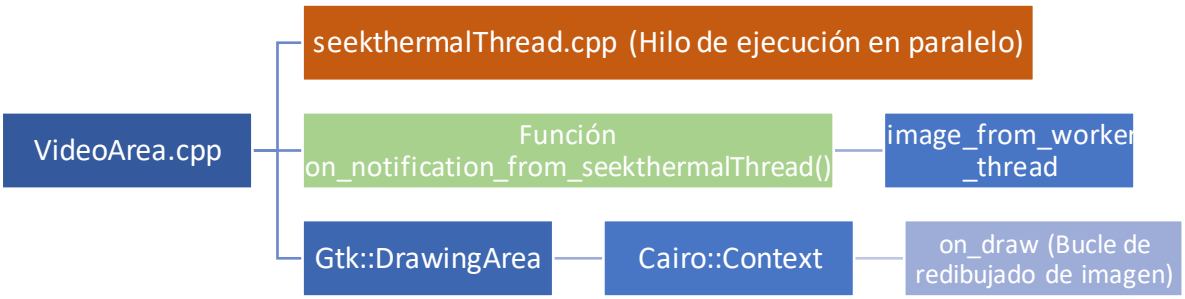
- Frame: Es un objeto de tipo Gtk::Frame, encargado de contener al objeto VideoArea, un objeto que se explicará en el siguiente apartado.
- VideoArea: Objeto encargado de mostrar las imágenes recibidas y de inicializar el thread de captura de las mismas. En el siguiente apartado se explicará el mismo con más detalle.

El archivo VideoFrame.cpp también se encarga de establecer las propiedades del objeto Frame que contiene el objeto VideoArea, estableciendo un título al mismo. Se ha establecido un título acorde con la finalidad del programa, "Seek Thermal Capture". También establece un tamaño de la ventana acorde con el tamaño deseado para la correcta visualización de las imágenes.

Por último se ejecuta la función show_all(), que muestra el objeto Frame y aquello que contiene en la interfaz.

7.3.2 VideoArea.cpp

De la misma manera que el resto de programas desarrollados, el programa VideoArea consta de un archivo VideoArea.h donde se declaran los objetos a utilizar, mientras que en el archivo VideoFrame.cpp se establece las relaciones entre dichos archivos y se lleva a cabo la visualización de los mismos. La peculiaridad de esta parte del programa es la inicialización de un hilo de ejecución en paralelo.



Esquema 7.2 – Estructura VideoArea.cpp

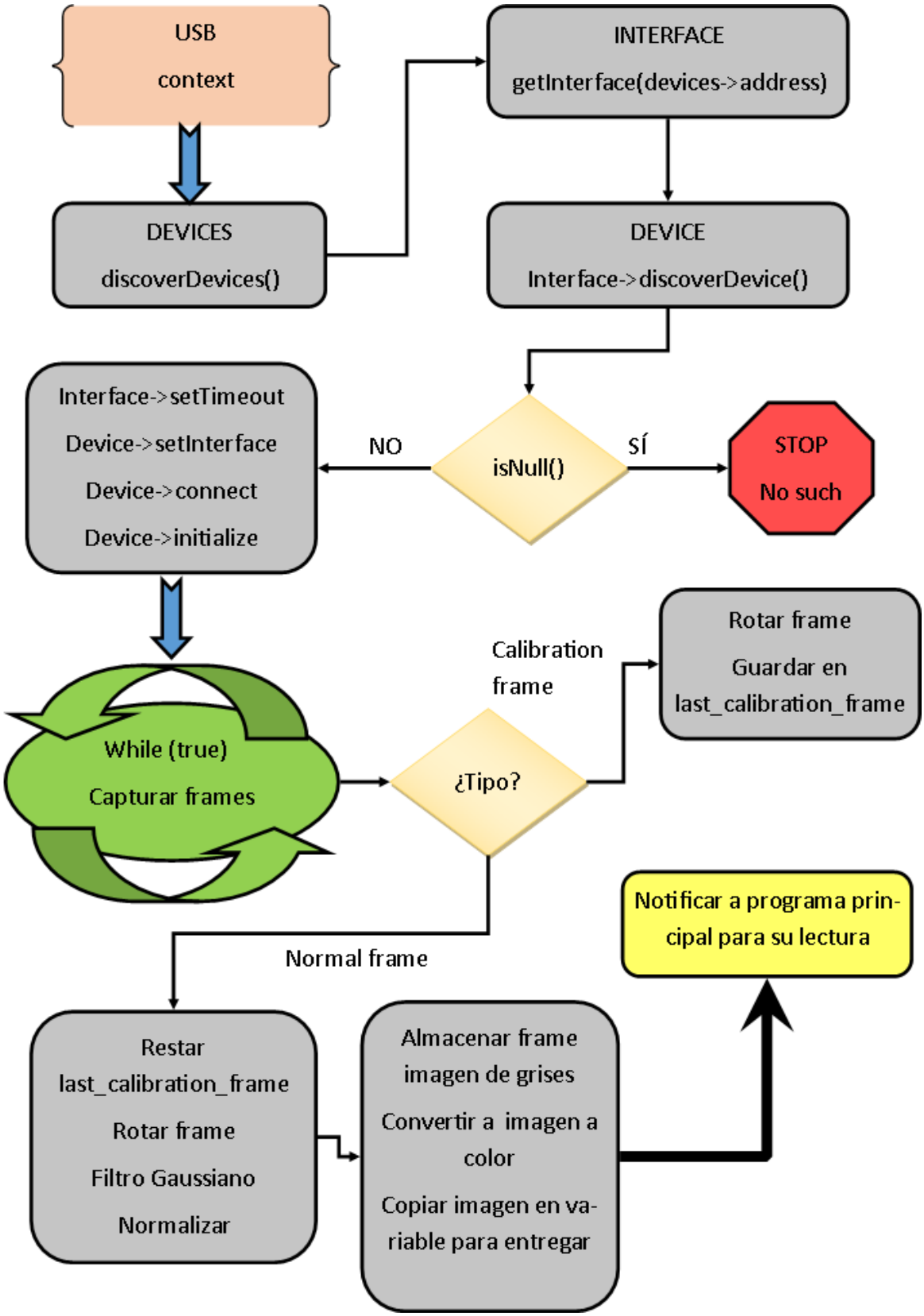
La estructura consta de los siguientes apartados:

- seekthermalThread.cpp: Hilo de ejecución encargado de leer y procesar los fotogramas desde la cámara Seek Thermal Compact. Es lanzado al principio de la ejecución de VideoArea.cpp y se encarga de notificar al programa VideoArea.cpp cuándo ha terminado de procesar un fotograma.
- on_notification_from_seekthermalThread(): Es el método ejecutado al ser notificado el programa VideoArea.cpp desde el hilo de ejecución seekthermalThread.cpp. Se encarga de recibir el dato desde este hilo de ejecución y de almacenarlo en la variable

image_from_worker_thread (que contiene la imagen a mostrar). Tras realizar esta acción el método redibuja un rectángulo en la pantalla, de manera que obliga a la ventana gtkmm a ejecutar la función on_draw.

- Gtk::DrawingArea: Es el objeto principal, creado por VideoArea.cpp, contiene la imagen a mostrar y el contexto de la misma.
- Cairo::Context: Es el contexto inherente a los objetos Gtk::DrawingArea, es el espacio de memoria desde la que es leída la imagen a mostrar en el objeto padre Gtk::DrawingArea.
- on_draw: método que es ejecutado cuando es dibujado algún objeto en la pantalla. Al ser ejecutado este método, éste refresca el contexto Cairo::Context con la imagen almacenada en image_from_worker_thread, lo que hace que ésta se muestre. Este método, al ejecutarse al principio del arranque de la ejecución, también es el encargado de lanzar el hilo de ejecución en el primer inicio.

7.3.3 seekthermalThread



Esquema 7.3 – Estructura seekthermalThread.cpp

El programa seekthermalThread es el programa encargado de leer los fotogramas desde la cámara Seek Thermal Compact a través de su interfaz USB, así como de procesarlas y finalmente notificar al programa VideoArea.cpp.

El algoritmo de lectura y procesado de datos se puede dividir en tres partes:

- Conexión con la cámara
- Captura y procesado de fotogramas
- Guardado y notificación al programa VideoArea.cpp

Empecemos por el primer apartado, correspondiente al algoritmo de conexión con la cámara.

- En primer lugar, se encuentran los dispositivos Seek Thermal que se encuantren conectados al equipo. Para ello se utiliza la función discoverDevices(), que almacena en un listado de punteros de tipo device, los dispositivos Seek Thermal encontrados.
- En segundo lugar, se declara un contexto USB de Seek Thermal. De esta manera, se accede al interface correspondiente de la lista generada por discoverDevices() mediante la función getInterface(address)
- A continuación, se crea el objeto device a partir del interface, mediante la función discoverDevice().
- Seguidamente, se comprueba que se haya creado dicho objeto, mediante la función isNull(), que devuelve un valor booleano true en caso de no encontrar el dispositivo especificado, en cuyo caso se notifica al usuario mediante un texto en la línea de comandos de "No such device". En caso de sí haber encontrado el dispositivo, el valor booleano devuelto es false, y continuamos ejecutando los siguientes pasos del algoritmo.
- Se establece un timeout mínimo para la interfaz mediante la función setTimeout(1000). Si no realizáramos este paso, el programa mostraría el mensaje "USB error: Operation timed out", y no continuaría con la ejecución. Este tiempo de timeout tan prolongado se debe a que en la Raspberry utilizada al establecer un tiempo de timeout corto puede darse el error "USB error: Operation timed out" de igual manera al estar utilizando la aplicación.
- Por último, se establece el interfaz con setInterface(interface), y se conecta e inicializa con las funciones connect() y initialize().

Ahora el dispositivo ya se encuentra listo para transmitir las imágenes que captura a nuestra aplicación. Las imágenes que transmite esta cámara térmica constan de 32448 palabras de 16 bits. Esto se corresponde con una resolución de 206x152 píxeles. Cada palabra se transmite con 14 de los 16 bits, y utilizando el sistema little-endian. Los fotogramas son enviados a una tasa de 8 fotogramas por segundo. El fotograma no consta de ninguna cabecera, en lugar de ello, los metadatos se almacenan en lugares especiales del píxel. Tanto para la función de captura y procesado de los fotogramas como para la de guardado y notificación al programa VideoArea.cpp, se utiliza un bucle infinito que engloba ambas funciones.

A continuación, se explicará la segunda parte del algoritmo, la correspondiente con la captura y procesado de los fotogramas:

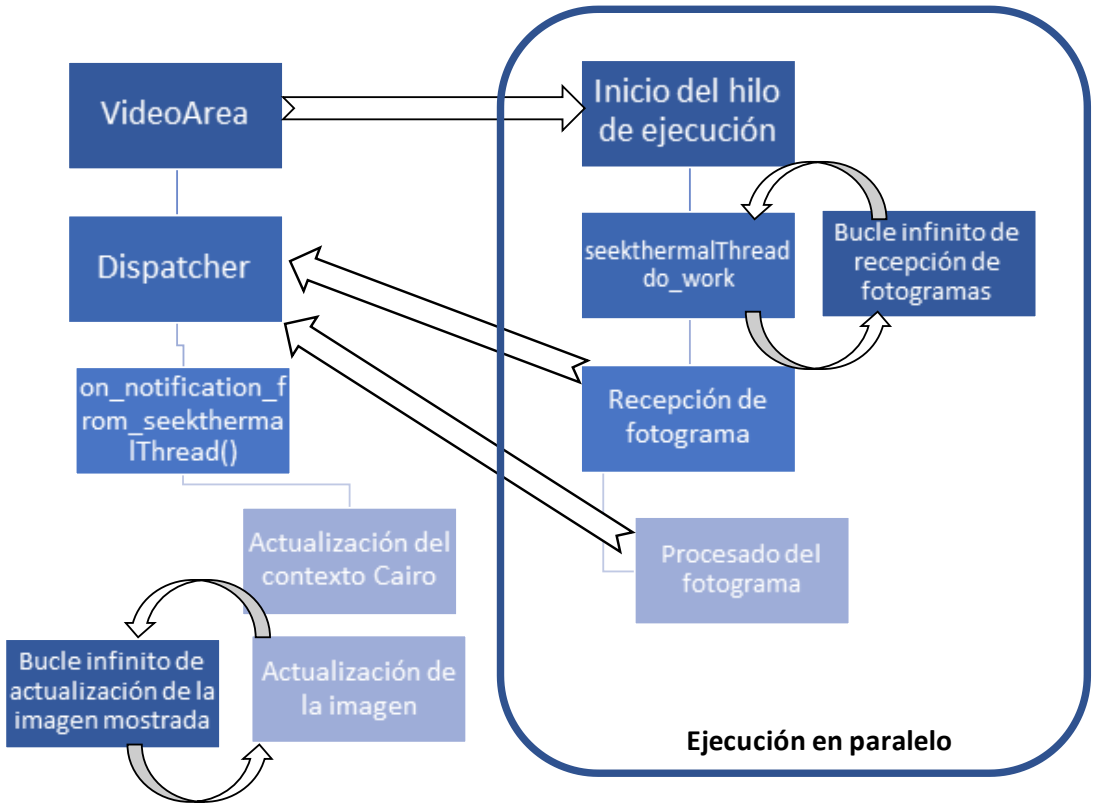
- En primer lugar, se crea un puntero de tipo fotograma que almacenará los datos capturados. Para capturar los datos desde la cámara se utiliza la función capture(*frame).
- En segundo lugar, se ha de discernir si el fotograma capturado es de tipo inválido, normal, calibración, precalibración o desconocido. Esto se realiza con la función getType().
- Sea como fuere el fotograma, se aplica una rotación en el sentido de las agujas del reloj mediante la función rotateClockwise().
- Si el fotograma es de tipo calibración, éste se almacena en la variable last_calibration_frame.
- Si el fotograma es de tipo normal, se continúa con el algoritmo.

- En caso de haber continuado con el algoritmo, se suaviza el fotograma mediante la aplicación de un filtro gaussiano, aplicado mediante la función gaussianBlur().
- Se almacenan la altura y anchura del fotograma en variables tipo size_t, mediante las funciones getWidth() y getHeight().
- El fotograma ha de ser normalizado entre 0 y 1, de manera que obtenemos el valor mínimo y máximo del fotograma mediante las funciones getMinimumValue() y getMaximumValue(). La normalización se realiza mediante la función normalize(min, max).
- Ya normalizado, se crea una variable de OpenCV tipo Mat de 8 bit sin signo de un canal (en escala de grises). Mediante un doble bucle for, asignamos a cada píxel el valor correspondiente de multiplicar el valor de dicho píxel por 255, de manera que obtenemos los valores correspondientes a una escala de grises.
- Se aplica por último una paleta de color, que ofrece una mejor visualización al usuario. Para ello, creamos una nueva variable de OpenCV tipo Mat de 8 bit sin signo de tres canales (R, G y B). Se asigna un valor a cada uno de los canales en función del valor del píxel original en la imagen en escala de grises.

Por último, se explicará la tercera parte del algoritmo, encargada del guardado y de la notificación al programa VideoArea.cpp

- Se copia la imagen final en la variable m_image, preparada para entregar al programa VideoArea.cpp
- Mediante el uso del Dispatcher, se notifica al programa VideoArea.cpp que el fotograma ha sido leído.

El funcionamiento de este mecanismo se puede apreciar en la siguiente figura:



Esquema 7.4 – Esquema de funcionamiento del hilo de recepción y procesado de imágenes térmicas

Capítulo 8. Instalación de paquetes y librerías necesarias. Compilación y ejecución.

8.1 Introducción

En el desarrollo de este documento se ha detallado la creación de un sistema de control de un sistema de realidad aumentada accionado mediante un guante, enfocado a un control sobre una aplicación genérica que se quiera llevar a un sistema de realidad aumentada.

Más tarde se ha aplicado este sistema de control sobre una aplicación concreta, un sistema de visualización para emergencia en caso de incendios, mejorando el algoritmo de este último.

Por este motivo, las instrucciones de paquetes y librerías a instalar, así como la compilación y la ejecución de los archivos fuentes, se dividirán en dos posibilidades:

- Instalación de los paquetes y librerías, compilación y ejecución sólo de la interfaz de control con una aplicación cualquiera programada sobre el archivo VideoArea.cpp.
- Instalación de los paquetes y librerías, compilación y ejecución de la interfaz de control conjuntamente a la aplicación "sistema de visualización para emergencia en caso de incendios". De ahora en adelante, este apartado será identificable por la etiqueta "Seek Thermal".

8.2 Instalación de paquetes

Para compilar y ejecutar el programa creado hace falta tener instalados una serie de paquetes instalados. Esto pueden ser instalados mediante los siguientes comandos:

sudo apt-get update

sudo apt-get install xdotool x11vnc libgtk-3-dev libgtkmm-3.0-dev

8.2.1 Paquetes adicionales sólo para Seek Thermal

Para poder realizar la instalación de las librerías necesarias para ejecutar el programa de control junto con la aplicación Seek Thermal integrada, es necesario instalar antes otros paquetes adicionales. Éstos pueden ser instalados mediante la ejecución de las siguientes instrucciones:

sudo apt-get update

sudo apt-get install doxygen pkg-config debhelper cmake groff libboost-chrono-dev libboost-dev libusb-1.0 build-essential cmake pkg-config libmp3lame-dev libvorbis-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev libqt4-dev libjpeg-dev libtiff-dev libxml2-dev libboost-dev liblapack-dev libblas-dev libeigen3-dev

8.3 Instalación de librerías (Solo Seek Thermal)

8.3.1 OpenCV

Se descarga el archivo con el código fuente desde el siguiente link:

https://github.com/opencv/opencv/archive/3.2.0.zip

Este archivo se ha de descomprimir, y a continuación se ha de abrir un terminal de Linux situado en la carpeta donde se ha descomprimido el archivo.

Para la realización de los siguientes pasos, en el caso de realizarlos en la Raspberry Pi, se recomienda desactivar el tiempo de apagado automático, de lo contrario ésta se puede apagar al realizar la instalación, puesto que es un proceso que demora mucho tiempo.

mkdir build

cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..

sudo make install

cd

sudo nano ~/.bashrc

Dentro del archivo .bashrc abierto, se añade la siguiente línea al final del mismo, para que se puedan enlazar las librerías de OpenCV en ejecución.

export LD_LIBRARY_PATH=/usr/local/lib

El archivo se guarda usando el comando Control+O para guardar, pulsando Enter para confirmar, y Control+X para salir.

8.3.2 ReMake

Esta librería es necesaria para poder instalar la librería libseekthermal. Para descargarla e instalarla ejecutamos en el terminal los siguientes comandos.

git clone https://github.com/kralf/remake

cd remake

mkdir build

cd build

cmake ..

sudo make install

8.3.3 libseekthermal

Esta librería es la que nos permitirá comunicarnos con la cámara Seek Thermal Compact. Para descargarla e instalarla ejecutamos en el terminal los siguientes comandos.

git clone https://github.com/ethz-asl/libseekthermal

cd libseekthermal

mkdir build

cd build

cmake ..

sudo make install

8.4 Compilación y ejecución

8.4.1 Compilación de la aplicación de control

Para compilar el programa final se utiliza:

cd /path hasta la carpeta donde se encuentren los códigos fuente

g++ -std=c++11 main.cpp BarraIndicadores.cpp MainWindow.cpp VideoFrame.cpp -o guiRun `pkg-config --cflags --libs gtkmm-3.0`

8.4.2 Compilación de la aplicación de control sobre aplicación Seek Thermal

Si se quiere compilar la aplicación de control junto a esta aplicación ejemplo, teniendo las librerías correctamente instaladas, se debe hacer con el siguiente comando:

cd /path hasta la carpeta donde se encuentren los códigos fuente

g++ -std=c++11 main.cpp BarraIndicadores.cpp MainWindow.cpp VideoArea.cpp VideoFrame.cpp -o guiRun `pkg-config --cflags --libs opencv` `pkg-config --cflags --libs libseekthermal` `pkg-config --cflags --libs gtkmm-3.0`

Mediante este comando se compilan los archivos con el código de la aplicación y se crea un ejecutable llamado guiRun.

8.4.3 Ejecución del programa compilado de la aplicación de control

Estando en la misma carpeta que el ejecutable creado, se puede abrir el mismo utilizando el siguiente comando:

./guiRun

8.4.4 Ejecución del programa compilado de la aplicación Seek Thermal

Antes de ejecutar el programa compilado, es necesario otorgar permisos de superusuario a la interfaz USB de la cámara Seek Thermal Compact. De no hacerlo, el programa devolverá un error "USB error: Access denied". Para ello realizamos los siguientes pasos:

En primer lugar, se ha de determinar en qué interfaz se encuentra conectada la cámara Seek Thermal Compact. Para ello ejecutamos:

ls –l /dev/bus/usb/00*

Esta instrucción devuelve un listado con todos los dispositivos USB conectados. Una vez identificado el interfaz al que queremos otorgar permisos de superusuario, realizamos la siguiente instrucción.

sudo chmod a+w /dev/bus/usb/00x/00x

Las x han de ser sustituidas por el bus y el dispositivo USB identificado, como se puede apreciar en la siguiente figura de ejemplificación.

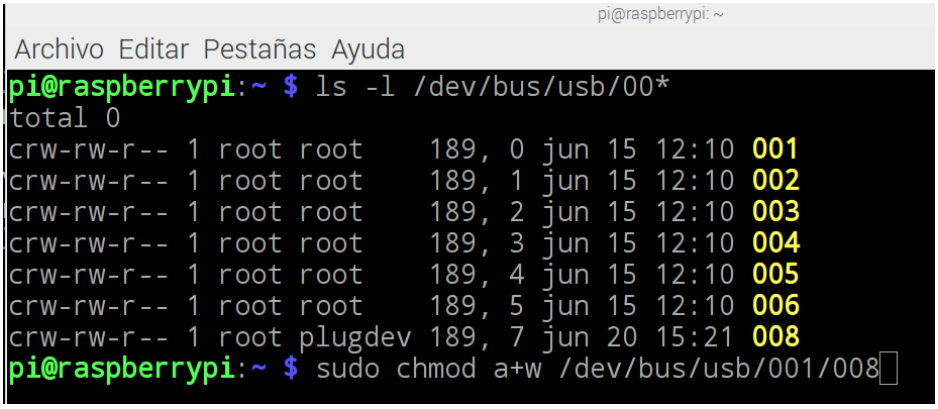


Figura 8.1 – Concesión de permisos de superusuario al interfaz USB

Una vez realizado este paso, con la consola situada en el directorio que contiene los códigos fuente y el programa guiRun compilado en el paso anterior, ejecutamos el mismo con el siguiente comando:

./guiRun

8.5 Resultado de la ejecución y layout del programa de control

8.5.1 Interfaz de control

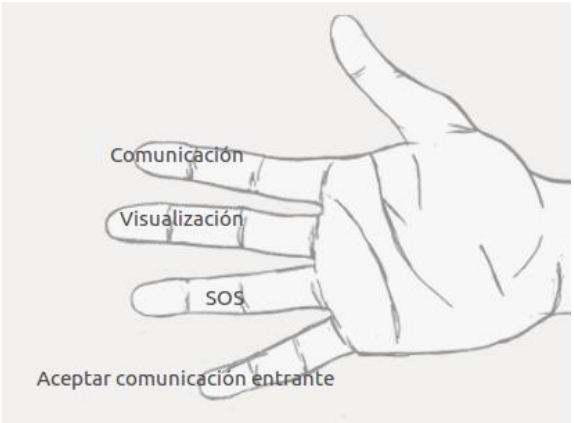


Figura 8.2 – Interfaz de control diseñada

8.5.2 Interfaz superpuesta

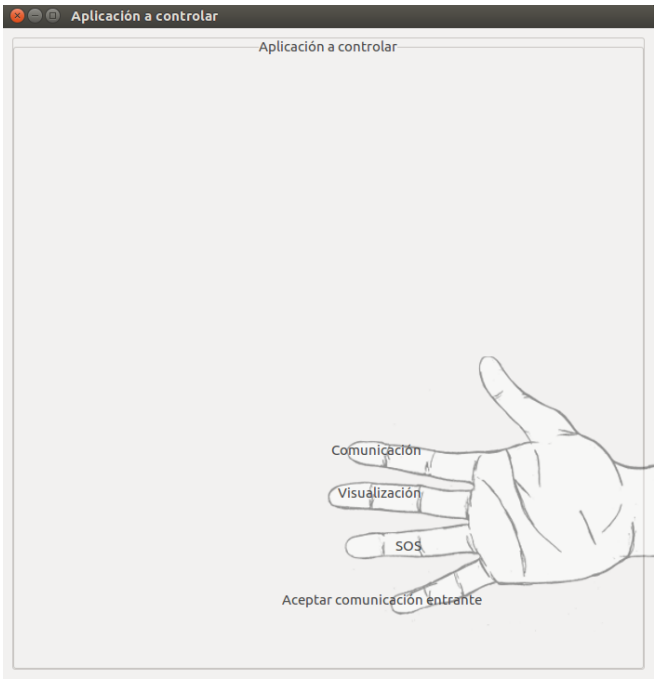


Figura 8.2 – Interfaz de control superpuesta sobre aplicación vacía

8.5.3 Interfaz superpuesta a aplicación Seek Thermal

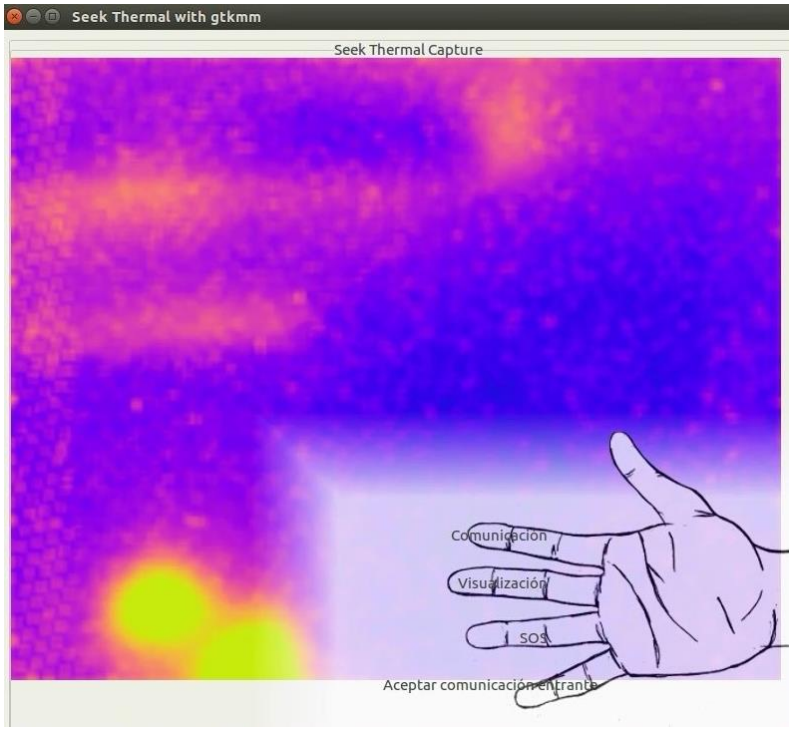


Figura 8.3 – Interfaz de control superpuesta sobre aplicación Seek Thermal

Capítulo 9. Conclusión y propuesta de trabajo futuro

Se ha conseguido de manera satisfactoria que el proyecto desarrollado suponga una aplicación útil para el control y la visualización de una imagen térmica en un sistema de realidad aumentada, enfocado a su uso en situaciones de emergencias en caso de incendio.

Las propuestas de líneas futuras de trabajo para avanzar al siguiente paso de desarrollo del sistema son dos:

La primera es ir incorporando a los menús de este proyecto los sistemas que proporcionan los datos. Por ejemplo, en la parte de visualización hemos diseñado espacio para visualizar la cantidad de oxígeno remanente en las botellas para la respiración del agente. Cuando se termine el sistema que proporciona este dato, habrá que integrarlo con el código desarrollado aquí y que realiza la visualización. Y así con el resto de funcionalidades.

En segundo lugar, se debe mejorar el dispositivo de procesado de la aplicación principal. La Raspberry Pi utilizada no tiene la potencia necesaria para mover con soltura el programa diseñado, ya que utiliza 4 hilos de ejecución en paralelo simultáneamente:

- Hilo de ejecución de la aplicación principal, muestra la interfaz y recarga la imagen térmica
- Hilo de ejecución del servidor UDP, recibe los datos UDP a través de un puerto
- Hilo de ejecución del servidor VNC, se encarga del envío de la imagen a las gafas
- Hilo de ejecución de recepción y procesado de imágenes térmicas

Cuando se ejecuta el programa principal en la Raspberry Pi utilizada, éste no se ejecuta a la velocidad necesaria para ser utilizado con comodidad por el usuario, resultando en una interfaz lenta.

Adicionalmente, tras unos minutos de uso, y debido a que el sistema operativo necesita el procesador para las tareas de gestión básica del sistema a la vez que se ejecutan las aplicaciones de usuario, la interfaz USB deja de responder y el programa aborta su ejecución o bien deja de refrescar la imagen térmica. Esto es debido a la carga computacional excesiva que sufre la Raspberry Pi. El programa desarrollado se ejecuta sin problemas en una máquina virtual Ubuntu ejecutada sobre Windows 10, sin presentar ninguno de estos problemas.

También se ha estudiado el comportamiento de la aplicación sobre la Raspberry Pi al desactivar partes de su código que suponen un mayor coste computacional. En estos casos la aplicación se ejecutaba sin presentar los problemas mencionados, confirmando la hipótesis de que el problema reside en la falta de potencia para el procesado de toda la aplicación en su conjunto de manera simultánea.

El modelo utilizado para el desarrollo de esta aplicación es una Raspberry Pi 2 Model B, que utiliza un procesador Broadcom BCM2837 ARMv7 de cuatro núcleos Cortex-A7 a 900 MHz e integra 1 Gb de memoria RAM.

La solución a este problema pasa por intentar aligerar el coste computacional del código ejecutado, y si a pesar de ello el problema persistiera, considerar el uso de otro computador de placa simple (SBC) con más capacidad de procesado.

Una alternativa podría ser la SBC ODROID-XU4, con el procesador ARM Samsung Exynos5422 con tecnología de fabricación de 28 nanómetros (8 núcleos en arquitectura big.LITTLE, 4 núcleos A-15 a 2 GHz y otros 4 a 1.5 GHz) y 2 Gb de memoria RAM LPDDR3, a la venta por 59$.

Otra posibilidad a contemplar es la placa SBC UP, actualmente en la plataforma KickStarter, que utiliza un procesador x86 Intel x5-Z8300 con tecnología de fabricación de 14 nanómetros (4 núcleos con una frecuencia de hasta 1.84 GHz) y 1 o 2 Gb de memoria RAM. Se puede reservar por 89€ en su versión con 1 Gb de memoria RAM o por 109€ en su versión con 2 Gb de memoria RAM.

Capítulo 10. Bibliografía

[1] Adafruit, “Adafruit Feather HUZZAH ESP8266”, https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/overview [Online].

[2] Miguel Balboa, “rfid”, https://github.com/ethz-asl/libseekthermal [Online].

[3] ESP8266 Community Forum, “esp8266”, https://github.com/esp8266 [Online].

[4] Ralf Kaestner, “libseekthermal”, https://github.com/miguelbalboa/rfid [Online].

[5] The GNOME Project, “GTK + Platform Support”, https://developer.gnome.org/gtk3/stable/platform-support.html [Online].

[6] The GNOME Project, “Programming with gtkmm 3”, https://developer.gnome.org/gtkmm-tutorial/stable/index.html.en [Online]

[7] itseez, “OpenCV Reference Manual”, http://docs.opencv.org/master/index.html#gsc.tab=0 [Online]

[8] Raspberry Pi Foundation, “Raspberry Pi Documentation”, https://www.Raspberrypi.org/documentation/ [Online]