UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster

CURSO 2018/19

DESARROLLO DE HERRAMIENTAS PARA APLICACIONES DE COORDINACIÓN EN ROBÓTICA COLECTIVA

Máster Universitario en Ingeniería Industrial

ALUMNA/O

Daniel López Enseñat

TUTORAS/ES

Abraham Prieto García

Pedro José Trueba Martínez

FECHA

Julio 2019

RESUMEN

Desarrollo de Herramientas para Aplicaciones de Coordinación en Robótica Colectiva.

En este trabajo se ha llevado a cabo el desarrollo de un sistema capaz de realizar un control de acceso y presencia. El entorno elegido para su implementación han sido las oficinas del GII del Campus de Ferrol de la UDC. Para ello, ha sido necesario elaborar, por un lado, una navegación mixta que combina la odometría con una localización mediante *AprilTags.* Por otro lado, se ha desarrollado una estrategia de vigilancia cooperativa con el fin de realizar la tarea de una manera más eficiente, y una centralita donde se muestran los paneles que contienen la información recopilada durante el proceso de control. Tras una validación de cada una de ellas, se ha realizado un ensayo y se han comentado los resultados obtenidos.

RESUMO

Desenvolvemento de Ferramentas para Aplicacións de Coordinación en Robótica Colectiva

Neste traballo levouse a cabo o desenvolvemento dun sistema capaz de facer un control de acceso e presenza. O entorno elixido para ser aplicado foron as oficinas do GII do Campus de Ferrol da UDC. Para isto, foi necesario elaborar, por un lado, unha navegación mixta que combina a odometría cunha localización mediante *AprilTag.* Por outro lado, desenvolveuse unha estratexia de vixilancia cooperativa co fin de realizar a tarefa dunha maneira máis eficiente, e un cadro de control onde pódese acceder os paneis que conteñen a información recollida durante o proceso de control. Despois dunha validación de cada unha de elas, realizouse un ensaio nun entorno real e comentáronse os resultados.

ABSTRACT

Development of Tools for Coordination in Collective Robotics

In this work, the development of a system, which controls the access and the tracking, has been carried out. The chosen environment for its implementation has been the offices of the GII of the Ferrol Campus of the UDC. For this, it has been elaborate, on the one hand, a mixed navigation that combines the odometry with a location using AprilTags. On the other hand, a cooperative surveillance strategy has been developed in order to perform the task in a more efficient way and a switchboard where the panels containing the information collected during the control process are displayed. After a validation of each of them, an essay has been made and the results obtained have been discussed.

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE MÁSTER

CURSO 2018/19

DESARROLLO DE HERRAMIENTAS PARA APLICACIONES DE COORDINACIÓN EN ROBÓTICA COLECTIVA

Máster Universitario en Ingeniería Industrial

Documento 1

MEMORIA

Contenido

Índice de Figuras

Índice de Pseudocódigos

Índice de Pseudocódigos

1 INTRODUCCIÓN

El presente Trabajo Fin de Máster está centrado en la realización de un control de acceso y presencia en las oficinas del GII (Campus de Ferrol de la UDC) con el Robobo, un robot desarrollado y producido por MINT, una spin-off de la UDC, en colaboración con el GII [1].

En primer lugar, para poder realizar el control de presencia ha sido necesario desarrollar una aplicación con la que poder desplazar al Robobo por el entorno de trabajo, es decir, una navegación. En esta navegación se ha combinado una odometría basada en los encoders presentes en las ruedas del Robobo y una localización mediante unas balizas artificiales que serán colocadas en el entorno. Por otro lado, se ha desarrollado un algoritmo que, mediante una librería de detección de objetos basada en OpenCV, es capaz de realizar un control de acceso. Por último, se ha combinado la navegación desarrollada con un algoritmo capaz de ubicar a las personas en función de su posición en la imagen captada por la cámara que ofrece el robot.

Ambos sistemas de control que se han implementado, se llevan a cabo de forma simultánea con dos Robobos diferentes. Para ello, ha sido necesario una arquitectura de comunicación basada en ROS, por donde pueden intercambiar mensajes entre ellos. Permitiendo, de esta manera, variar el comportamiento de uno en función de la información que ha recopilado el otro.

Por todo ello, uno de los grandes campos en los que se enmarca este Trabajo Fin de Máster es la robótica.

1.1 Robótica

La Robótica es una ciencia que estudia el diseño y construcción de máquinas capaces de desempeñar tareas que realizaría un humano o que requieren el uso de inteligencia o lógica. Un robot, según la IFR (*International Federation of Robotics*), es un mecanismo accionado programable con un grado de autonomía, moviéndose dentro de su entorno, para realizar tareas previstas. La autonomía en este contexto significa la capacidad de realizar tareas previstas basadas en el estado actual y la detección, sin intervención humana[2]. Se pueden clasificar en 2 tipos diferentes: Robots industriales y de servicio.

En lo referente a este proyecto, los robots relacionados con el trabajo que se propone son los robots dedicados al servicio. Estos robots se definen como aquellos que realizan tareas útiles para el ser humano, excluyendo todas aquellas relacionadas con automatización industrial. A su vez, existen gran cantidad de clasificaciones según el tipo de servicio que va a realizar (por ejemplo, robots de tareas domésticas, de seguridad, de entrenamiento, sistemas logísticos…)

Figura 1: Ejemplo de robot autónomo

Por otro lado, en este trabajo se van a utilizar varios robots simultáneamente, lo que lo convierte en un ejercicio de robótica colectiva. Este tipo de robótica se refiere a un grupo de robots que interactúan entre sí pero no necesariamente han de colabora explícitamente para realizar una tarea. La solución del problema está ligado al concepto de inteligencia colectiva, es decir, que el problema es resuelto gracias al comportamiento como grupo de los individuos implicados. Esta inteligencia hace que el sistema desarrollado sea más robusto, ya que al existir varios robots pueden repartirse las tareas y conseguir solucionar el problema con una mayor tasa de éxito. Por consiguiente, es más eficiente tener varios robots simples realizando una tarea que un único robot más complejo.

En robótica colectiva, cada individuo puede ser programar de forma diferente, es decir, cada uno tiene su propia inteligencia. Además, tienen capacidad de comunicación y, por tanto, de coordinación y cooperación.

En este caso en particular, los robots van a estar constantemente conectados a un nodo central (un ordenador) por donde se comunicarán mediante la red WiFi del edificio donde se encuentren. Además, el usuario del citado ordenado puede acceder a esta información a tiempo real. Todas estas comunicaciones son posibles gracias al desarrollo que están sufriendo todas las tecnologías del ámbito de las telecomunicaciones. Consecuentemente, en este trabajo se define una arquitectura donde gran cantidad de dispositivos están conectados entre ellos y realizando una tarea común, que es el gran pilar en el que se basa el famoso paradigma del Internet de las Cosas.

1.2 El Internet de las Cosas

Hoy en día, alrededor de dos mil millones de personas se conectan a internet para múltiples funciones diferentes. Mientras este número no deja de crecer, está surgiendo otro gran avance relacionado con el uso de Internet como plataforma global para permitir que las máquinas y los objetos inteligentes se comuniquen, compute y coordinen entre ellos (El Internet de las cosas) Por tanto, el Internet de las Cosas (*IoT*, por sus siglas en inglés) es un concepto que hace referencia a una conexión digital de los objetos cotidianos mediante internet.

Figura 2: Internet de las Cosas

Es previsible que, dentro de la próxima década el contenido y los servicios estarán a nuestro alrededor, siempre disponibles, allanando el camino para el desarrollo de nuevas aplicaciones o funciones, permitiendo nuevas formas de trabajo, nuevas formas de interactuar...En definitiva, una nueva forma de vivir. Estos cambios serán todavía más notables gracias a la aparición de la tecnología 5G, que va a permitir tener una altísima velocidad de descarga y subida de datos sin necesidad de una red WiFi.

Siguiendo esta tendencia, el concepto convencional de Internet como una red que llega hasta los terminales de los usuarios finales desaparecerá con el paso del tiempo, dejando hueco para objetos inteligentes que forman entornos informáticos generalizados [3]. No obstante, la infraestructura de internet no desaparecerá, como es lógico, sino que mantendrá su rol como columna vertebral como medio de comunicación en todo el mundo, interconectando objetos con capacidades de computación y comunicación para realizar una gran cantidad de servicios.

Esta innovación se irá haciendo notable según se vayan sustituyendo los actuales objetos cotidianos, no “inteligentes”, por otros que cumplan su misma función, pero sí sean “inteligentes”. Esto dará lugar a nuevas oportunidades dentro del sector de las telecomunicaciones, facilitando el camino a aplicaciones que requieran de la citada interconexión entre los diferentes objetos de un entorno.

Según [4] y [5], Internet de las Cosas puede hacer referencia a 3 conceptos: Red global resultante que interconecta objetos inteligentes mediante tecnologías de internet extendidas; conjunto de tecnologías de apoyo necesarias para realizar la definición anterior (por ejemplo, sensores y actuadores o dispositivos de comunicación entre máquinas); y el conjunto de aplicaciones y servicios resultantes.

2 OBJETIVOS

El objetivo principal de este Trabajo Fin de Máster es la realización de un control de acceso y presencia simultáneos en las oficinas del GII con Robobo y robótica colaborativa. No obstante, para alcanzar el objetivo principal del proyecto, hemos definido las siguientes etapas intermedias:

- Implementación de librerías desarrolladas en ROS: AprilTags y detección de objetos
- Implementación de una odometría con la que poder estimar la trayectoria del Robobo
- Desarrollo de una navegación con la que poder desplazar el Robobo por un entorno.
- Realizar un control de acceso y presencia en las oficinas del GII
- Implementar una estrategia de colaboración para resolver la tarea de manera distribuida mediante un grupo de robots
- Desarrollo de una centralita con la que poder acceder de manera sencilla y rápida a la información recopilada por los robots
- Validar el correcto funcionamiento, así como calcular el error que se comete en las diferentes aplicaciones desarrolladas
- Realizar un ensayo en un entorno real

3 ANTECEDENTES

En este apartado se va a hablar en primer lugar de la visión artificial y algunas de sus aplicaciones. Posteriormente, se explicarán los diferentes sistemas de localización existentes en la actualidad; y, por último, una pequeña introducción a diferentes métodos de control de acceso y de presencia de personas en un recinto o instalaciones.

3.1 Visión artificial

En el presente trabajo la visión artificial, o también llamada visión por ordenador, tiene un papel fundamental en el cumplimiento de sus objetivos, ya que gracias a ella será posible la detección de las personas mediante una cámara.

La visión artificial es una rama de la ciencia que aporta métodos para adquirir, procesar y analizar las imágenes del mundo real (captadas mediante una cámara) con el fin de producir información que un ordenador pueda tratar. Como en otros campos de la informática, la visión artificial trata de reproducir lo que el ser humano realiza cuando analiza lo que está viendo, es decir, percibir y comprender una imagen o un vídeo y actuar según las circunstancias. La interpretación se consigue gracias a múltiples disciplinas, pero, sobre todo a la geometría, la física y la estadística.

Uno de los campos donde más se ha empleado históricamente la Visión Artificial es en el de la industria, teniendo un papel fundamental en el desarrollo de la que hoy conocemos como industria 4.0. En este contexto, los sistemas de visión son considerados un elemento para la protección de fallos en el proceso de producción y tener la capacidad de detectar anomalías antes de añadir valor al producto y que finalmente sea defectuoso.

Figura 3: Control de calidad mediante Visión Artificial

No obstante, en el mundo real existen gran cantidad de circunstancias que dificultan la compresión de estas imágenes. Además de los ruidos técnicos que puedan aparecer, hay grandes dificultades debido a interferencias causadas por el contexto o entorno de trabajo:

- Obstrucción: puede que el objeto que se quiere detectar se encuentre en segundo plano y oculto parcialmente por otro.
- Escala: en una imagen, un objeto que se encuentre cerca del foco de visión puede parecer más grande que aquel que permanezca lejos, dificultando así el reconocimiento.
- Punto de vista: un mismo objeto puede observarse desde infinitas perspectivas.
- Deformación: Un objeto puede que esté deformado en una imagen, ya sea porque esta haya sido mal capturada o un efecto óptico.
- Fondo desordenado: el fondo. que se encuentra detrás del objeto a encontrar puede ser caótico y dificultar así su búsqueda.
- Diferentes clases de un objeto: un mismo objeto puede tener diferentes formas y colores.

Dentro de la visión artificial existen varias ramas, una de las cuales es la detección de objetos.

3.1.1 Detección de objetos

Object tracking o la detección de objetos, es una rama de la visión artificial encargada de detectar la presencia de objetos en una imagen, o una secuencia de ellas, mediante su apariencia visual. Generalmente, se pueden diferenciar dos partes en el proceso de detección: la obtención de las características de una imagen dada y la búsqueda de ciertos objetos según las características obtenidas anteriormente.

La obtención de las características se basa en modelos matemáticos que compacten el contenido de la imagen para facilitar el aprendizaje del algoritmo que busca los objetos. Estas características suelen ser llamadas descriptores. Existen varios tipos de descriptores que variarán su eficacia según el objeto que se quiera reconocer.

Para el proceso de clasificación se pueden usar diferentes técnicas de aprendizaje máquina. Existen diferentes métodos, desde los más básicos, como la regresión lógica, hasta los más complejos como el SVM o AdaBoost.

Una de las librerías de reconocimiento de objetos más empleadas a nivel mundial es OpenCV, de Intel. Existen otras alternativas como SimpleCV, ImageUltimate o FastCV. En la Figura 4 puede observarse un ejemplo de reconocimiento de objetos con OpenCV.

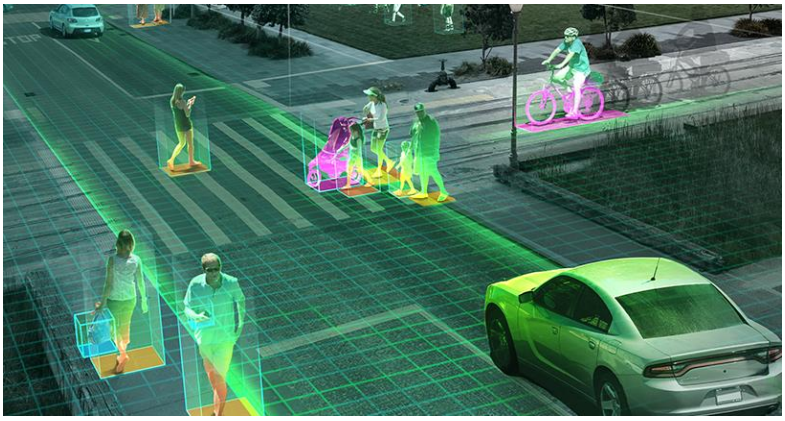


Figura 4: Detección de objetos con OpenCV

Estos tipos de sistemas, gracias al tratamiento de imágenes, permiten una gran variedad de aplicaciones, como, por ejemplo: controlar y monitorizar a tiempo real el tráfico de una carretera, control de acceso a un centro comercial, etc. Además, son capaces de realizar estas tareas sin intervenir prácticamente en el entorno. Sin embargo, por norma general, estos sistemas requieren de una gran cantidad de cálculo para poderse llevar a cabo.

3.1.2 Aprendizaje automático

Las técnicas de aprendizaje automático tienen como finalidad diferenciar automáticamente patrones mediante algoritmos matemáticos. Este método suele ser usado para clasificación imágenes y detección de objetos. El aprendizaje utilizado puede ser tanto supervisado como no supervisado [6]:

- Aprendizaje supervisado: Es el más utilizado y requiere de intervención humana, para la creación de datos que vienen etiquetados con el resultado correcto. Cuanto mayor sea la lista de datos mejor aprenderá la máquina sobre el problema a resolver. Es decir, se utiliza cuándo el usuario conoce algunos resultados que la máquina ha de obtener para ciertos estímulos o entradas. Un ejemplo sería el aprendizaje de una máquina para que sea capaz de identificar los números o letras escritas por una persona
- Aprendizaje no supervisado: En los problemas de aprendizaje no supervisado el algoritmo es entrenado usando un conjunto de datos, con la diferencia de que en este caso no van etiquetados con la respuesta correcta; nunca se le dice al algoritmo lo que representan los datos. La idea es que este pueda encontrar por sí solo patrones que ayuden a entender el conjunto de datos.

Dentro de la Visión Artificial, y más concretamente en el reconocimiento de objetos, el aprendizaje supervisado es ampliamente más utilizado que el no supervisado. Esto se debe que se pueden obtener gran cantidad de casos (imágenes del objeto a identificar) con el que el algoritmo pueda aprender adecuadamente. Cuando el problema a resolver es muy complejo, por ejemplo, la detección de personas, los algoritmos basados en estos aprendizajes son capaces de conseguir tasas de éxito que ningún otro método puede alcanzar.

3.2 Localización

La capacidad de un robot móvil para determinar su ubicación en el espacio es una tarea imprescindible para poder navegar de forma totalmente autónoma [7]. Además de navegar siguiente una ruta planificada o de esquivar ciertas zonas u obstáculos, la localización permite realizar tareas de mapeado o de localización de elementos que sean identificados en un entorno. Sin la capacidad de localización, un robot estaría condenado a interactuar con el entorno a través de comportamientos reactivos, y, consecuentemente, sería imposible planificar acciones fuera de su percepción local [8].

Para cualquier robot, el sistema de localización más fácil y barato de implementar es el odométrico, basado en el movimiento de las ruedas motrices. No obstante, todo sistema basado en la odometría conlleva un error debido a errores intrínsecos del modelo que se acumulan con el paso del tiempo, error conocido como deriva. Consecuentemente, estos sistemas hay que complementarlo con otros que su error se encuentre acotado y siempre sea el mismo, aproximadamente, independientemente de cuando sea calculada la ubicación del robot

A estos métodos se les denomina relocalización y se pueden agrupar en dos grandes grupos, la detección de marcadores externos, tanto naturales como artificiales presentes en el medio, como los *AprilTags*; y la utilización de información suministrada por los sensores del

robot y el mapa con el que se le configura a priori, como en el caso del robot comercial Roomba (Figura 5). Los sensores más utilizados en la actualidad son:

- Sensores de proximidad: Infrarrojos y ultrasonidos.
- Visión artificial [9]
- Sistemas de posicionamiento global (GPS) [10].

Figura 5: Roomba. Ejemplo de un robot autónomo capaz de orientarse mediante sensores de proximidad

La localización que se intenta desarrollar en este trabajo está orientada a un espacio interior. Por tanto, se descarta la opción del GPS. Por otro lado, los únicos sensores de proximidad disponibles en el robot que se va a emplear, el Robobo, son los sensores infrarrojos (se explicará en más detalle en el apartado 4.2). Estos sensores son adecuados para evitar colisiones con objetos, pero no tan eficaces como otros métodos para la localización del robot en el entorno. Por último, a pesar de que el Robobo es una base móvil para un *smartphone* y, por tanto, dispone de una IMU, los sistemas de localización mediante navegación inercial implican una alta complejidad a la hora de implementarlos.

Consecuentemente, el sistema más adecuado es el sistema mediante visión artificial, es decir, mediante marcadores posicionados en el entorno. Como ya se ha comentado, estos marcadores o balizas pueden ser naturales o artificiales [11].

3.2.1 Balizas naturales

Las balizas naturales son cualquier objeto que se encuentre en el entorno, sin necesidad de alterar este último, que pueda servir de ayuda al robot para localizarse. Su gran ventaja respecto a las balizas artificiales es que no es necesario alterar el entorno en el cual el robot va a trabajar.

Por otro lado, el gran problema de las balizas naturales, es ser capaces de detectar y extraer información de objetos que se encuentran originalmente en el entorno por el que se produce la navegación mediante los sensores disponibles en el vehículo autónomo. Además, estos sensores que se requieren para estos sistemas de localización suelen tener un precio bastante elevado.

El estudio de este problema se denomina *SLAM (Simultaneous Localization and Mapping). SLAM* trata de responder si es posible que un robot móvil y autónomo se coloque en una ubicación desconocida en un entorno desconocido y que el robot construya de manera incremental un mapa coherente de ese entorno mientras determina simultáneamente su ubicación dentro del dicho mapa. La resolución de este problema ha sido un suceso crucial para que la robótica autónoma sea como la conocemos actualmente. El proceso del *SLAM*, a grandes rasgos, consiste en cuatro pasos: extracción de características, asociación de datos, estimación del estado y actualización de las características.

Las técnicas estadísticas utilizadas para aproximar las ecuaciones, presentes en [12], proporcionan una estimación de la función de probabilidad para la posición del robot y los parámetros del mapa. Existen métodos que se aproximan a ellas de forma conservadora, como por ejemplo el algoritmo *Covariance Intersection*.

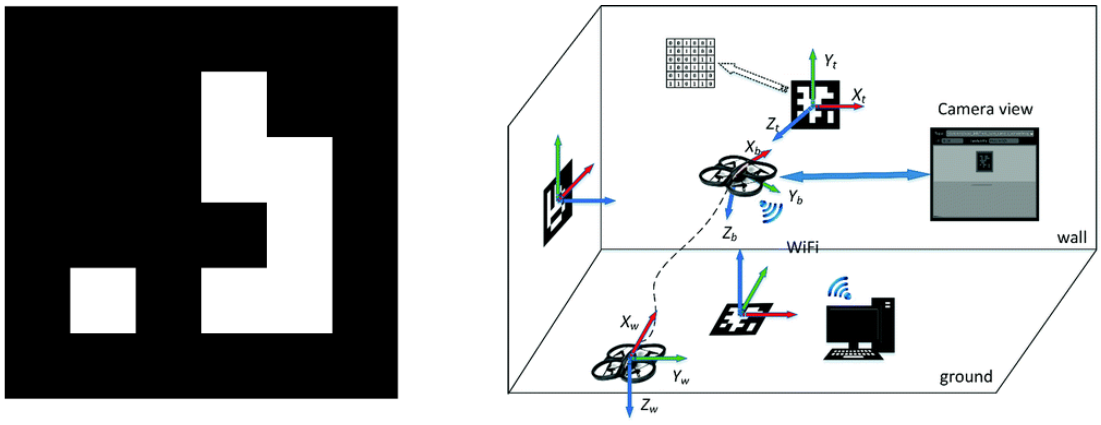
A pesar de ser un método cada vez más optimizado y polivalente, hoy en día siguen teniendo muchas limitaciones. Además, requiere una capacidad de cálculo elevada. Por estas razones, se ha descartado la implementación de un sistema de estas características.

3.2.2 Balizas artificiales

El problema de la localización autónoma del robot mediante balizas se simplifica notablemente cuando estas son artificiales. No obstante, es necesario que en el entorno de trabajo del robot pueda ser modificado y añadir dichas balizas o marcadores. En la actualidad, existen diferentes tipos de *tags* que pueden ser utilizados como balizas artificiales, destacando los códigos QR, los *AprilTags* o los *Aruco*. A pesar de existir una librería nativa ya desarrollada para los códigos QR en Robobo, se ha decido implementar una librería externa para el uso de los *AprilTags* debido a que estos proporcionan una mayor cantidad de información para la localización de los mismos, y a que el rango de posiciones y condiciones de visibilidad en las que son detectables es mayor que para el caso de los códigos QR. Como contrapartida, requieren un mayor coste computacional para ser procesados.

3.2.2.1 AprilTags

AprilTags es un sistema de marcadores de referencia, basado en marcadores o *tags*, que además de ser utilizado para la navegación autónoma de los robots, también es útil para gran variedad de tareas como la calibración de cámaras, la realidad aumentada o la localización de elementos en un entorno.



a) *b)*

Figura 6: ArpilTag id=0 de la familia 16h5 (a); y un ejemplo de navegación de un dron en un entorno virtual mediante AprilTags (b)

A día de hoy, hay disponibles 5 tipos diferentes de *AprilTags,* codificados de la siguiente manera: 16h5, 25h7, 25h9, 36h9, o 36h11. Cuanto más alto sea en valor de la familia a la que pertenece, un marcador puede almacenar más información y, por tanto, dicha familia puede contener un número mayor de identificadores. No obstante, ante un mismo tamaño de *tag*, los cuadrados que lo conforman son más pequeños dificultando así su lectura. Por ello, se ha decidido utilizar la familia 16h5, ya que permiten ser leídos a una gran distancia a pesar de estar trabajando con la cámara interna de un *smartphone,* y los 29 identificadores que nos proporciona son suficientes para el objetivo de este trabajo. En la figura anterior, se muestra un ejemplo de un *tag* de la familia 16h5 y un ejemplo virtual de navegación de un dron utilizando este sistema.

3.3 Control de acceso

El control de acceso engloba el conjunto de técnicas que permiten el seguimiento del número de personas que acceden o abandonan un espacio a través de una de sus zonas de entrada o salida. Este control se lleva a cabo con diversos fines, como por ejemplo para el control de asistencia a eventos, zonas de trabajo o espacios públicos. En este apartado se hablará de las diferentes tecnologías existentes para llevar este control de acceso ya que es uno de los objetivos principales de este trabajo.

3.3.1 Mediante tornos

Uno de los métodos más utilizados a nivel mundial, para el control de acceso de personas en unas instalaciones, son los tornos de acceso. Se pueden ver tanto en los metros y trenes de las grandes ciudades, instalaciones deportivas, etc.

Según un estudio [13], el control de acceso mediante tornos es un sistema rápido y eficaz, con un nivel de confianza muy alto, en torno al 100%. No obstante, a pesar de su gran funcionamiento, estos sistemas no son capaces de conocer cómo están distribuidas las personas en las instalaciones a controlar. Además, este tipo de sistemas conllevan una intrusión significativa en el entorno, produciendo grandes incomodidades en las personas afectadas.

3.3.2 Mediante cámaras de vídeo

Hoy en día, cada vez está más extendido el control de acceso mediante cámaras de vídeo. Igual que en el caso anterior, se desconoce la distribución de la gente en las diferentes zonas del recinto. No obstante, debido a que no implica una molestia para la persona, podrían instalarse cámaras en las diferentes puertas y llevar un control sobre cómo están distribuidas de una manera aproximada.

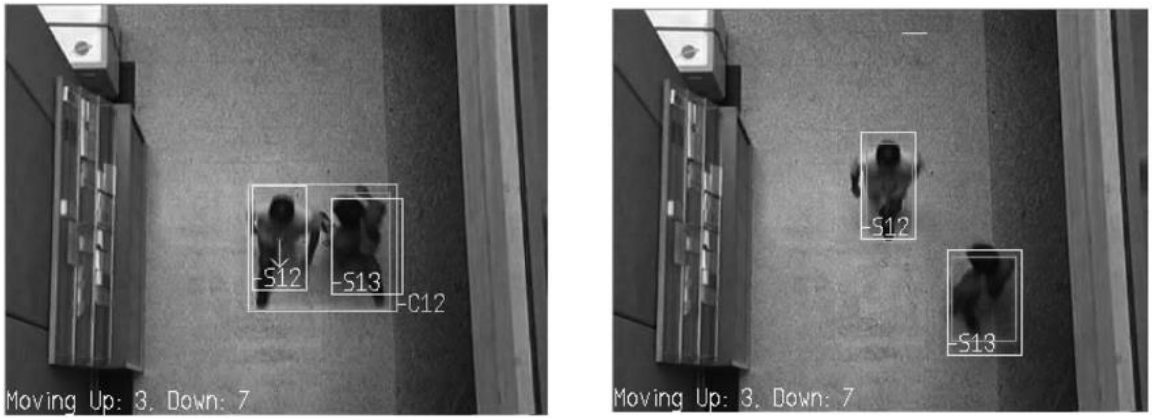


Figura 7: Control de acceso mediante visión artificial [14]

El artículo [14], se desarrolla un algoritmo inteligente capaz de llevar a cabo el control de acceso de personas. Además, nos muestra diferentes configuraciones y sus respectivas tasas de éxito. A pesar de no tener un nivel de confianza tan alto como el control mediante tornos, puede obtener tasas de éxito de entre el 90y el 95%.

Además, con la tecnología adecuada, a estos sistemas se les puede añadir un reconocimiento facial con el que tener un control de intrusos [15].

3.4 Control de presencia

Por otro lado, además del control de acceso, en este proyecto se va a implementar un algoritmo capaz de realizar un control de presencia. Es decir, llevar a cabo un control sobre la localización de las personas a tiempo real en el interior del recinto donde se esté llevando a cabo el control de acceso. Se va a hablar sobre dos sistemas diferentes que existen en la actualidad, pero no existe una gran cantidad de soluciones específicas para este problema: Sistemas basados en fichajes y sistemas basado en visión artificial.

3.4.1 Control mediante fichajes

A día de hoy, uno de los métodos más utilizados es el control de presencia mediante fichajes por sala, es decir, existen terminales de fichaje en cada una de las diferentes salas de una instalación. Este tipo de sistemas, por lo general, ofrece una eficacia que otros sistemas no pueden alcanzar, siempre y cuando sus usuarios lo utilicen de forma adecuada. Por otro lado, igual que en el caso de los tornos en el control de acceso, suele ser necesaria una variación del entorno donde se quiera realizar dicho control. Además, estos sistemas también imponen a las personas a realizar una tarea para poder llevar a cabo el control, es decir, no es un sistema completamente automático.

En el artículo [16], se implementa un sistema de control basado en tarjetas de identificación con el que monitorizar la asistencia de a los estudiantes de una universidad de Hungría. Según el artículo, este sistema cuenta con una muy alta fidelidad, ya que durante el primer año de uso únicamente ha habido una incidencia relacionada con su funcionamiento

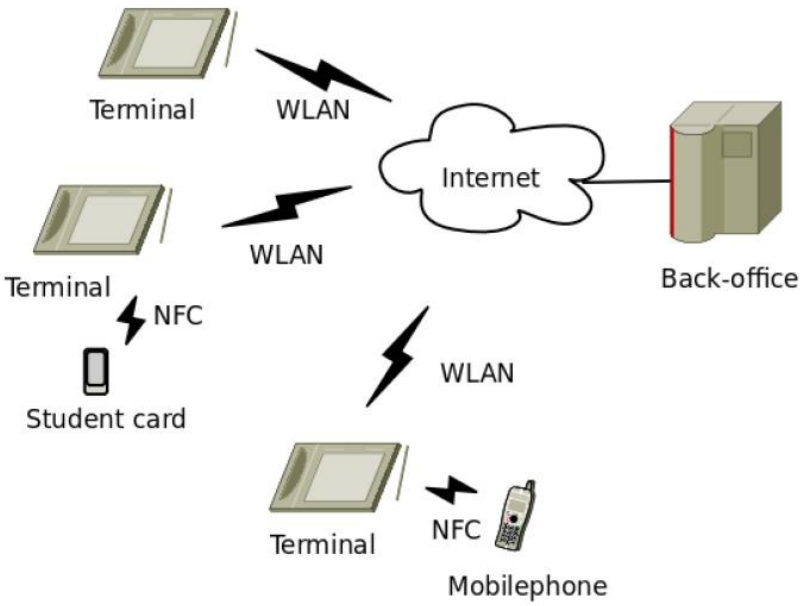


Figura 8: Estructura en la que se basa en sistema de control de asistencia del articulo [16]

Además, existen otros métodos que son cada vez más utilizados, como por ejemplo el fichaje mediante huella dactilar[13]. Este tipo de sistemas, a pesar de ser igual de eficiente que el anterior, implica una mayor inversión económica para poder llevarla a cabo.

3.4.2 Mediante visión artificial

El control de presencia mediante visión artificial utiliza algoritmos de tracking de personas para monitorizar el presencia y localización de estas dentro de un entorno confinado y vigilado mediante una o más cámaras. De nuevo, no existen muchas soluciones tecnológicas o trabajos que estudien este problema en la bibliografía. El trabajo más parecido que se ha encontrado, está descrito en el artículo [17]. Dicho trabajo consiste en un robot móvil y autónomo que es capaz de detectar y seguir a una persona en un entorno real a tiempo real

Además, en el artículo [18], se describe un algoritmo capaz de realizar un seguimiento a las personas captadas a través de una cámara. No obstante, se trata de una cámara fija y no de un robot móvil tal y como se ha implementado en este trabajo.

4 FUNDAMENTOS TECNOLÓGICOS

En este apartado se explicarán los diferentes recursos tecnológicos en los que se han basado este proyecto, tanto a nivel de *hardware* (Robobo) como a nivel de *software* (ROS). En lo referente al *software*, se ha elegido utilizar ROS porque ya hay disponible una librería con la que poder acceder de manera rápida y eficiente a todos los servicios y funcionalidades que el Robobo nos ofrece. De la misma forma, facilita la comunicación entre *scripts.*

4.1 Robot Operating System (ROS)

ROS es un sistema meta-operativo de código abierto pensado para la robótica. Proporciona los mismos servicios que otros sistemas operativos, pero además incluye otros como el control de dispositivos a bajo nivel, implementación de funcionalidades de uso común, comunicación entre diferentes procesos en ejecución, y administración de paquetes. También proporciona herramientas y librerías para programar, escribir y ejecutar código mediante diferentes ordenadores.

La arquitectura de ROS se basa en una red punto a punto (P2P) de procesos, distribuidos en diferentes máquinas, acoplados libremente entre sí mediante la infraestructura de comunicación de ROS. Ros implementa varios tipos diferentes de estilos de comunicación, incluyendo el *synchronous RPC-style* para los servicios, transmisión asíncrona de mensajes para los *topics*, y almacenamiento de datos para el Servidor de Parámetros.

ROS es una estructura distribuida de procesos que permite que los ejecutables sean individualmente diseñados y libremente ejecutados en el tiempo de ejecución. Estos procesos pueden ser agrupados en Paquetes, los cuales se pueden compartir y distribuir de manera fácil y eficaz. ROS también permite distribuir el código en Repositorios.

Estas son unas de las principales características de ROS como sistema operativo orientado a la robótica:

- Está diseñado para ser lo más liviano posible, minimizando los tiempos de respuesta.
- Las librerías están desarrolladas para Python, C++ y Lisp, permitiendo que se pueda implementar con gran facilidad en cualquier programa actual de robótica.

4.1.1 Estructura de ROS a nivel de archivos

A continuación, se muestra como está organizado ROS a la hora de guardar los archivos (ejecutables, código, etc) en el disco:

- Paquetes: Unidad principal para organizar software en ROS. Un paquete puede contener procesos de tiempo de ejecución de ROS (nodos), una biblioteca dependiente de ROS, archivos de configuración, conjuntos de datos, etc. Estos paquetes son el elemento de organización más pequeño dentro de la estructura de ROS, es decir, lo mínimo que puedes compilar y lanzar es un paquete.
- Metapaquetes: Son un tipo especial de paquete que solo sirven para representar un grupo de otros paquetes que están relacionados entre ellos. Normalmente, se utilizan como marcadores de posición.
- Manifiestos de paquetes (*package.xml*): Proporcionan metadatos sobre un paquete, es decir, nombre, versión, descripción, información de licencia, dependencias, etc.
- Repositorios: Un conjunto de paquetes que comparten un sistema VCS. Los paquetes que son compartidos por los sistemas VCS comparten la misma versión y pueden ser lanzados juntos con la herramienta de automatización *catkin bloom*.

- Tipos de mensaje (*msg*): Definen la estructura de datos para los mensajes enviados en ROS
- Tipos de servicio (srv): Definen las estructuras de datos de solicitud y respuesta para los servicios en ROS.

4.1.2 Estructura de ROS a nivel computacional

Como ya se comentó con anterioridad, el gráfico de computación es una red P2P de los procesos de ROS que procesan los datos de forma conjunta. Los conceptos básicos del gráfico de computación son:

- Nodos: son procesos que requieren computación. ROS está diseñado para ser modular y tener una gran cantidad de nodos simultáneos. En un único trabajo puede haber un nodo que controla el motor de las ruedas, otro que realiza la planificación de la trayectoria, otro que publica la imagen de una cámara…
- Maestro: en ROS. El maestro proporciona el registro de nombres y la búsqueda del resto del gráfico de computación. Gracias al maestro los nodos pueden comunicarse y compartir información entre ellos (mediante *topics*).
- Mensajes: los nodos se comunican entre sí mediante mensajes. Estos son simplemente una manera de estructurar los datos. Hay gran cantidad de tipos, los mensajes estándar (enteros, *float*, *boolean*) u otros más específicos para la librería que se vaya a utilizar.
- *Topics*: los mensajes se transmites mediante un sistema de transporte del tipo publicación / suscripción. Un nodo publica un mensaje en un *topic* determinado proporcionándole un nombre a dicho *topic*. Si otro nodo está interesado en acceder al mensaje publicado por el primer nodo, se suscribirá al *topic* (conociendo su nombre). Puede haber múltiples nodos publicando y suscritos a un mismo *topic*. Además, los editores y suscritores no conocen la existencia de los demás nodos implicado en el *topic*, desacoplando de esta manera la producción de la información con su consumo.
- Servicios: el modelo de publicación / suscripción entre múltiples nodos de los *topics* es muy versátil, pero su transporte unidireccional lo convierte en un método inapropiado para interacciones solicitud / repuesta que son tan necesarias en la robótica. Esta comunicación se realiza mediante los servicios. Un nodo proveedor ofrece un servicio con un nombre y un cliente lo utiliza enviando un mensaje de solicitud a dicho servicio.
- *Bags*: son un formato para guardar y reproducir datos de mensajes de ROS.

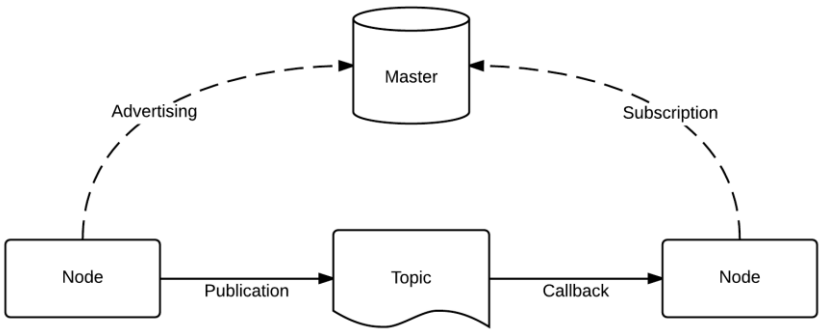


Figura 9: Estructura de comunicación de ROS

Como se muestra en la figura anterior, a pesar de la existencia del *ROS Master* los nodos se conectan entre sí directamente. Por tanto, el maestro únicamente proporciona una

información de búsqueda, para que puedan encontrarse y posibilitar su comunicación. Los nodos se comunican mediante un protocolo de conexión acordado. Generalmente suele ser TCPROS, que utiliza *sockets* TCP/IP estándar. Esta arquitectura permite operaciones desacopladas. Donde los nombres son los medios principales con los que se pueden construir sistemas de gran tamaño y complejidad. Por tanto, los nombres tienen una labor crucial en ROS, ya que los nodos, *topics*, servicios y los parámetros tienen sus respectivos nombres con los cuales pueden ser identificados.

4.2 Robobo

Para la tarea que se va a realizar en el actual trabajo va a ser necesaria la utilización de un robot que pueda moverse, tener una cámara orientable y una mínima capacidad de cálculo. El Robobo, que es un producto desarrollado y producido por MINT, una spin-off de la UDC en colaboración con el GII, cumple estos requisitos mínimos, a pesar de tener ciertas limitaciones ya que no es un robot pensado para su aplicación industrial

En la siguiente figura se muestra el diseño de la base del Robobo. Cabe destacar que no es un robot convencional, sino que es u Un robot que utiliza un *smartphone* como unidad de procesado y que va ubicado en la unidad *pan-tilt (orientable).*

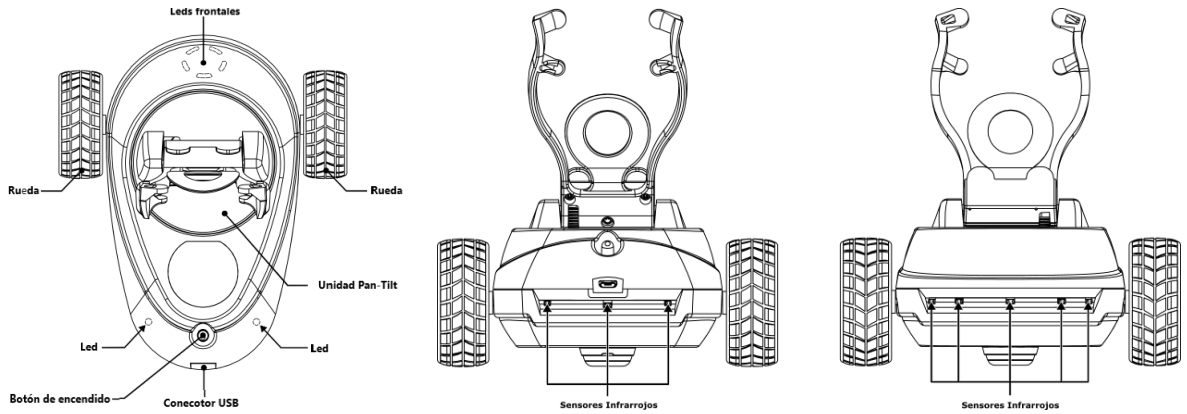


Figura 10: Vista en planta (izquierda), trasera (centro) y frontal (derecha(del Robobo

4.2.1 Partes del Robobo

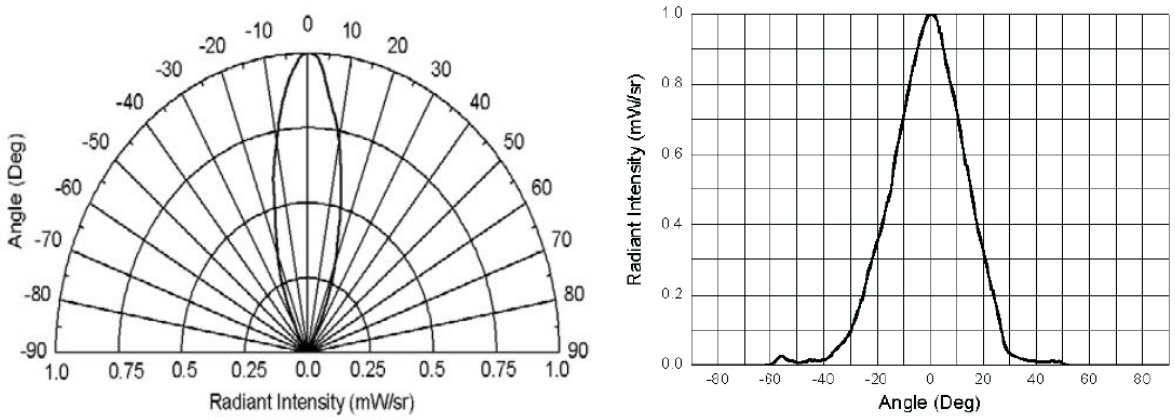
En este apartado se procede a definir las diferentes partes del Robobo que han servido de utilidad en este proyecto (Sensores infrarrojos, motores y unidad pan-tilt, junto a las funciones que aporta el *Smartphone)*

4.2.1.1 Sensores Infrarrojos

Un sensor infrarrojo son unos componentes electrónicos compuestos por un LED infrarrojo y un fototransistor colocados uno a la par del otro. El LED infrarrojo emite una luz infrarroja. Si esta luz choca contra una superficie blanca se reflejará y llegará al fototransistor, si por el contrario golpea en una superficie negra, esta absorberá casi toda la radiación no llegará de vuelta al transistor. Según la radiación captada por el transistor dejará pasar más o menos corriente, de forma que se pueda medir dicha radiación. Por tanto, el LED actúa como emisor y el fototransistor de receptor. El Robobo utiliza 8 sensores VCNL4040 (Figura 11: Sensores infrarrojos VCNL4040), 5 en la parte delantera y 3 en la trasera.

Figura 11: Sensores infrarrojos VCNL4040

Este tipo de sensores varían su medición según la distancia del objeto a medir, hasta una distancia de 200mm, y el ángulo al que se encuentre (Figura 12). Como es lógico, cuanto más cerca esté el objeto mayor radiación va detectar el fototransistor y, por tanto, mayor será la medida del infrarrojo. Por otro lado, cuanto mayor sea el ángulo de detección menor será la medida del sensor. En la siguiente figura se puede ver tanto la variación de la medida con respecto al ángulo (Figura 12b) como un perfil aproximado de su campo de visión (Figura 12a).



a) *b)*

Figura 12: Proyección del campo de visión del sensor(a); y variación de la medida del sensor según el ángulo (b)

4.2.1.2 Motores

El motor que contiene el Robobo es un motor reductor de corriente continua. El modelo es tff-n20va-09220, de TT MOTOR. Este tipo de motores tienen como tensión nominal de trabajo 6V, no obstante, se puede utilizar en un rango de entre 3 y 9V. El movimiento en vacío comienza a partir de los 0.5V, pero no tiene potencia suficiente para mover el Robobo.

Robobo tiene dos motores encargados de su movimiento, uno en cada rueda. Ambos motores tienen una reducción de 150:1 con la que permite tener un mayor par y, por tanto, una mayor aceleración, pero una menor velocidad máxima; un encoder con el que medir el giro de la rueda (en grados); y un sistema de engranajes para la transmisión del movimiento.

El Robobo también cuenta con dos motores, uno para el Pan y otro para el Tilt. En el caso del Pan, es un motor idéntico al de las ruedas. Pero en el caso del Tilt, cuenta con una reducción de 1000:1 para maximizar la precisión a la hora de orientar el *smartphone.*

4.2.1.3 Unidad Pan-Tilt

Como ya se dijo con anterioridad, Robobo es una base móvil para un *smartphone*. La pieza encargada de sujetarlo es el Tilt, que va sujeto encima del Pan. Es una de las piezas claves de este proyecto, ya que, sin una orientación del *smartphone* independiente a la de la base, no sería posible realizarlo. Gracias a los motores mencionados el Pan puede girar entre 11 y 343 grados, y a su vez el Tilt entre 26 y 109. Gracias a estos dos movimientos combinados (Figura 13), el Robobo nos permite mirar casi en cualquier dirección, si necesidad de reorientar la base.

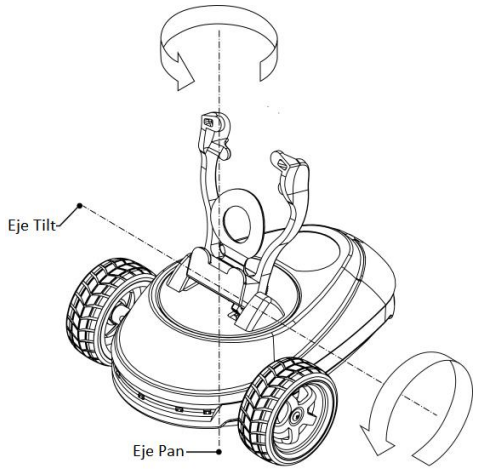


Figura 13: Unidad Pan-Tilt

4.2.2 Robobo en ROS

Para Robobo ya está desarrollada una librería con las cuales se puede solicitar una actividad (Servicios) y recibir la información que publica (*topics*). En primer lugar, se van a mencionar y definir los servicios que se han empleado:

- */robot/moveWheels* (*robobo_msg/MoveWheels*): Ordena mover las ruedas del robot a una velocidad y durante un tiempo especificados en los argumentos. Argumentos:
  - *Lspeed*: Velocidad de la rueda izquierda (entre 0 y 100).
  - *Rspeed*: Velocidad de la rueda derecha (entre 0 y 100).
  - *time*: Duración del movimiento en milisegundos.
  - *Blockid*: Un identificador para notificar el fin del movimiento en el *topic /robot/unlock/move*
- */robot/resetWheels* (*robobo_msg/ResetWheels*): Resetea los valores de los encoders de las ruedas a 0. Sin argumentos
- */robot/movePanTilt* (*robobo_msg/MovePanTillt*): Posiciona tanto el Pan como el Tilt en la posición deseada (ambos en una única solicitud). Argumentos:
  - *panPos:* Posición deseada, en ángulos, del Pan (entre 11 y 343)
  - *panSpeed*: Velocidad a kla que se quiera que gire el Pan (entre 0 y 100)
  - *panUnlockId*: Un identificador para notificar el fin del movimiento en el *topic /robot/unlock/move.* Ha de ser mayor de 0 para que se mueva.
  - *tiltPos:* Posición deseada, en ángulos, del Tilt (entre 11 y 343)
  - *tiltSpeed*: Velocidad a la que se quiera que gire el Tilt (entre 0 y 100)
  - *tiltUnlockId*: Un identificador para notificar el fin del movimiento en el *topic /robot/unlock/move.* Ha de ser mayor de 0 para que se mueva.
- */robot/setSensorFrecuency (robobo_msg/SetSensorFrecuency):* Nos permite cambiar la frecuencia con la que se refrescan los datos publicados por el Robobo.

No obstante, depende tanto de la capacidad de cálculo del *smartphone* utilizado como de la saturación de la red WiFi en la que se esté trabajando. Argumentos:

- *Frecuency*: *String* que representa la frecuencia *('LOW', 'NORMAL', 'HIGH', 'MAX'*)

Por último, se definen los *topics* disponibles en Robobo que han sido utilizados:

- *robot/camera/image/compressed:* Se publica a tiempo real la imagen captada por la cámara interior del *smartphone (*mediante *compressed_image_transport)*
- */robot/pan:* Se publica la posición del pan cuando esté se está moviendo. En grados.
- */robot/tilt:* Se publica la posición del tilt cuando esté se está moviendo. En grados
- */robot/wheels:* Se publica el giro acumulado desde que se encendió o se resetearon los encoders. En grados.

4.3 AprilTags en ROS

La librería empleada para detectar los AprilTag está desarrollada para ROS. Como se puede observar en la Figura 14, la librería tiene como entradas dos *topics*:

- */camera/image_rect (sensor_msgs/Image)*: *Topic* que contiene la imagen a analizar (En este caso, la imagen a tiempo real de la cámara del *Smartphone* acoplado)
- */camera_info (sensor_msgs/CameraInfo)*: *Topic* que contiene la matriz de calibración de la cámara. Única para cada cámara.

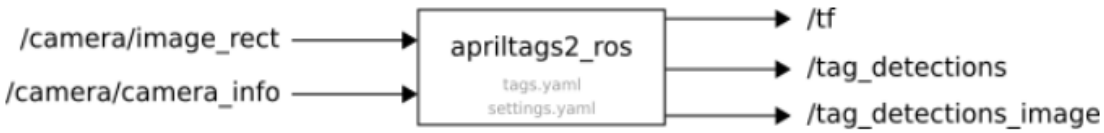


Figura 14: Estructura del funcionamiento de la librería de AprilTags empleada

Por otro lado, tiene como salida 3 *topics:*

- */tf:* Posición relativa entre la cámara y el AprilTag. Solo publica en caso de que un tag haya sido encontrado.
- */tag_detections:* Misma información que */tf* pero con un mensaje más personalizado, donde se muestra los *IDs,* tamaños, etc…Se publica constantemente.
- */tag_detections_image*: Devuelve la misma imagen que recoge del *topic /camera/image,* pero marcando los AprilTags identificados.

4.3.1 Calibración de la cámara

Con el objetivo de utilizar la librería *apriltag_ros*, se realizará una calibración previa de la cámara, la cual permitirá calcular la distancia a la que se encuentra el marcador detectado. Para ello, se ha empleado un programa desarrollado por ROS, donde se mueve un patrón de calibración de dimensiones conocidas por la imagen, intentando barrer todas las posiciones. En la Figura 15, se muestra el procedimiento citado anteriormente. La cámara que se ha utilizado es la interna del *OnePlus 5*:

Figura 15: Calibración cámara interna de un OnePlus 5

Como resultado se han obtenido las siguientes matrices de calibración:

$$A = \begin{bmatrix} 528.129 & 0 & 245.417 \\ 0 & 531.914 & 311.971 \\ 0 & 0 & 0 \end{bmatrix} \quad (1)$$

$$B = [0.0557 \quad -0.0610 \quad 0.0005 \quad 0.0018 \quad 0] \quad (2)$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$D = \begin{bmatrix} 535.045 & 0 & 246.248 & 0 \\ 0 & 539.354 & 312.271 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4)$$

Donde A es la matriz de la cámara, B los coeficientes de distorsión, C la matriz de rectificación y D la matriz de proyección.

Mediante un *script* escrito en *Python*, se publican estos coeficientes en el *topic* del tipo *CameraInfo.* Dicho *topic* será enviado a la librería de los *AprilTags*, que se encargará de realizar los cálculos pertinentes con los coeficientes obtenidos.

4.4 Detección de personas mediante OpenCV y TensorFlow

El paquete que se ha utilizado para la detección de personas mediante una imagen RGB, utiliza fundamentalmente dos librerías: OpenCV y TensorFlow.

OpenCV, como ya se ha dicho en el apartado de antecedentes, es una librería de código libre desarrollada por Intel. Desde su lanzamiento en 1999 ha sido utilizada en una amplia variedad de aplicaciones (desde sistemas de seguridad con detección de movimiento hasta reconocimiento de objetos). Esto se debe gracias a ser de código libre, ya que su publicación se da bajo una licencia BSD, lo que permite que sea usada libremente tanto para tareas comercias como en tareas de investigación, siempre y cuando se cumplan las condiciones definidas en la licencia.

Además, OpenCV es una librería plataforma, puesto que puede ser utilizada en GNU/Linux, Mac OS, Windows y Android. Contiene más de 500 funciones, entre las que destacan la visión robótica, reconocimiento de objetos, reconocimiento facial y visión estéreo, todas ellas desarrolladas en los lenguajes de programación C y C++.

Por otro lado, TensorFlow es una librería de código abierto (desde 2015), desarrollada por Google, para aprendizaje automático a través de un rango de tareas. Consta de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones correlacionados, similares al aprendizaje y lógica usada por el humano. Actualmente es utilizado tanto en la investigación como en los productos del propio Google, sustituyendo a su antecesor de código cerrado, DistBelief.

La librería utilizada, llamada *ros_people_object_detection_tensorflow*, es una herramienta diseñada para ROS con el objetivo de detectar y seguir objetos, entre ellos personas. También permite realizar un reconocimiento facial, pero se necesitaría una cámara con sensores de profundidad que en cuanto los *smartphones* la incluyan, ya podría integrarse en el Robobo. Su funcionamiento se basa en una red neuronal entrenada (de TensorFlow) capaz de reconocer una lista de 90 objetos diferentes con la ayuda de OpenCV.

Por tanto, para el reconocimiento de objetos, en este caso de personas, solo es necesario publicar la imagen de una cámara normal en un *topic*, el cual se ha llamado */Robobo/camera/Image*, del tipo *sensores_msg/Image.* Para ello ha sido necesario la realización de un *script* para cambiar la imagen de la cámara a una no comprimida, puesto que de forma original el Robobo publica con un mensaje del tipo *sensores_msg/CompressedImage.* En la siguiente Figura se puede ver un esquema básico de cómo es el funcionamiento de la librería:

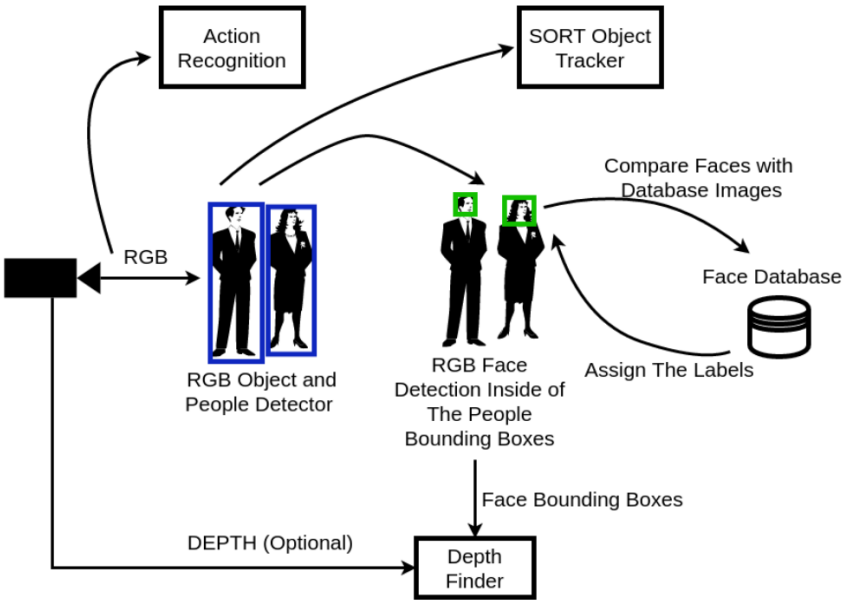


Figura 16: Diagrama del funcionamiento de la librería de detección de personas

Este paquete de ROS tiene múltiples funciones. No obstante, en este trabajo únicamente se van a emplear la detección de objetos y su seguimiento. En relación a los *topics*, hay uno de entrada (imagen de la cámara codificada en código RGB); y tres de salida:

- */object_detection/detections (cob_perception_msgs/DetectionArray):* Se publica en todos los objetos que ha detectado en cada fotograma. Incluye la posición del objeto identificado en cada fotograma, el tipo de objeto y el tamaño de la etiqueta, en píxeles, que recuadra el objeto. También publica un *array* vacío en caso de que no reconozca nada.
- */object_detection/detections_image (sensor_msgs/Image):* Muestra la imagen de la cámara poniendo una etiqueta en los objetos detectados.
- */object_tracker/tracks (cob_perception_msgs/DetectionArray)*: Es capaz de reconocer que objetos son los mismos que en fotograma anterior y ponerles un identificador (ID). Igual en el caso de las detecciones, se publica en el *topic* cada fotograma.

5 DESARROLLO

Este trabajo está formado por 5 componentes básicos. Por un lado, la localización del Robobo (mediante odometría y AprilTags) y una navegación que permita al robot desplazarse por el entorno. Por otro lado, el control de acceso y presencia que se ha llevado a cabo. Y, finalmente, una centralita que permita visualizar la información recopilada por la aplicación a tiempo real.

5.1 Odometría

La odometría es el estudio de la estimación de la posición de vehículos con ruedas a partir de la navegación. Es decir, la odometría tiene como objetivo ser capaz de estimar la posición en la que se encuentra el robot en todo momento. En general, la odometría proporciona una buena precisión a corto plazo, un bajo coste computacional y unas tasas de muestreo muy elevadas. Sin embargo, conlleva una acumulación de errores inevitable, provocando fallos en la estimación de la posición. Este error acumulado aumenta proporcionalmente con la distancia recorrida por el robot y puede darse por múltiples factores, entre los que destacan:

- Diámetros de las ruedas desiguales.
- Alineación inexacta de las ruedas.
- El valor del encoder se refresca de forma discreta.
- Deslizamiento con el suelo, ya sea por suelos resbaladizos, sobre-aceleraciones o giros excesivos.

En este trabajo se va a implementar una estimación de la posición mediante odometría basada en los encoders de los que dispone en Robobo en sus ruedas. Estos encoders nos permite medir el ángulo que han girado las ruedas a tiempo real, aunque de forma discreta.

Con ellos, se puede realizar una estimación de la posición (relativa a la inicial) de una manera bastante precisa. Por tanto, sabiendo que el diámetro de la rueda es de 6.5 cm y la desmultiplicación de la reducción del motor, se puede conocer la distancia recorrida por cada una de las ruedas. En la siguiente Figura se muestra una representación gráfica del movimiento estimado para el caso de que la distancia recorrida por la rueda izquierda sea mayor que la derecha:

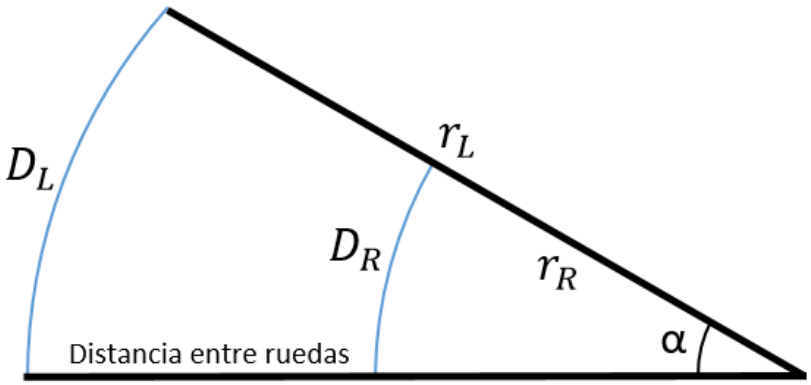


Figura 17: Representación gráfica de la estimación del movimiento

De las variables representadas en la figura, únicamente se conoce las distancias recorridas por ambas ruedas, es decir, D_L y D_R, obtenidas de los encoders. De este manera, es necesario calcular el ángulo α, y los radios de giro r_L y r_R para poder estimar la trayectoria recorrida. Sabiendo que el arco es igual al radio por el ángulo (en radianes) podemos obtener las siguientes ecuaciones:

$$\alpha = \frac{D_R - D_L}{D} \tag{5}$$

Donde D_L y D_R, son las distancias recorridas por las ruedas izquierda y derecha, respectivamente; y D la distancia entre ruedas del Robobo (14.8 cm)

Para el caso en que la rueda derecha tiene un mayor desplazamiento la expresión cambiaría de signo. De la ecuación (5) se obtiene:

$$r_R = \frac{D_R}{\alpha}; \quad r_L = \frac{D_L}{\alpha} \tag{6}$$

Conociendo la dirección que une los centros de ambas ruedas (se ha definido como el vector unitario con origen la rueda izquierda y punto terminal la rueda derecha); la posición inicial de ambas ruedas; y la distancia r_R, se puede calcular el punto sobre el cual rota el Robobo en su trayectoria.

$$P_{Rot} = P_R + r_R \cdot VectorEje \tag{7}$$

Donde P_{Rot} es la posición del punto de rotación, P_R la posición inicial de la rueda derecha, y $VectorEje$ el vector normalizado que une ambas ruedas.

Con este nuevo punto calculado ya se puede estimar las siguientes posiciones de las ruedas:

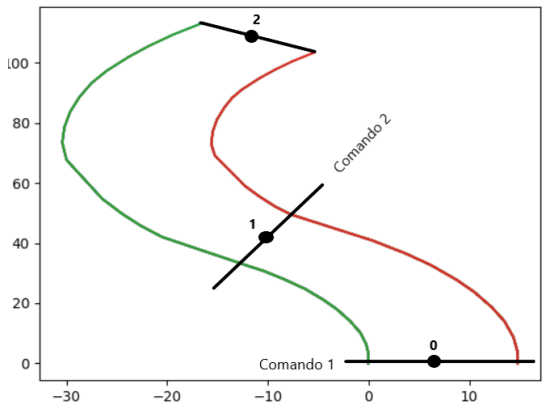
$$P_{L2} = P_{Rot} + r_L[\cos(\alpha_{\mathrm{Rob}} + \alpha), \sin(\alpha_{\mathrm{Rob}} + \alpha)] \tag{8}$$

$$P_{R2} = P_{Rot} + r_r[\cos(\alpha_{\mathrm{Rob}} + \alpha), \sin(\alpha_{\mathrm{Rob}} + \alpha)] \tag{9}$$

Siendo P_{L2} y P_{R2} las nuevas posiciones estimadas de las ruedas y α_{Rob} el ángulo entre el vector que marca la dirección del Robobo y la horizontal.

Por tanto, mediante la información aportada por los encoders, podemos estimar a tiempo real la posición de ambas ruedas respecto a la inicial. Consecuentemente, no solo se puede localizarlo en el entorno, de forma aproximada, sino que se puede estimar su orientación. Además, con las expresiones obtenidas se puede observar que son válidas para cualquier circunstancia en la que nos encontremos, independientemente de que nos encontremos en un giro puro, describiendo una curva o en un movimiento recto.

Se ha desarrolado una función en Python basada en las ecuaciones anteriomente citadas para el cálculo de la trayectoria a tiempo real del Robobo. Para validar su funcionamiento se ha realizado un experimento donde se le solicitado al Robobo, mediante el servicio *MoveWheels,* dos movimientos consecutivos: Un giro hacia la izquierda (más velocidad a la rueda derecha) y otro hacia la derecha (más velocidad a la rueda izquieda); y se ha comprobado si las posiones estimadas coinciden con las reales:



Comando 1	MoveWheels(20,10)
Comando 2	MoveWheels(10,15)

Punto	**Estimada**		**Real**	
0	7.4	0	7.4	0
1	-10.1	44.2	-9.2	45.3
2	-13.2	107.8	-12.1	109.7

Figura 18: Trayectoria estimada mediante la odometría desarrollada para el Robobo

5.2 Localización mediante AprilTags

Cuando un sistema de localización esta basado únicamente en odometría, se va acumulando un error con el paso del tiempo. Con el fin de evitar que el error de localización crezca de manera ilimitada se van a emplear unas balizas artificiales para ubicar el robot con exactitud en el entorno cada cierto tiempo y, por tanto, eliminar el error de deriva debido a la odometría. Concretamente, en este trabajo se han utilizado los *AprilTags* como método de localización.

```
begin /* LeerAPT*/
    Iniciar la librería de AprilTags
    Ejecutar los scripts que publican en los topics CameraInfo e Image.
    If se conoce posicion tag then:
            Se orienta el pan-tilt
            If no se encuentra el tag then:
                    Se busca APT
                    Se orienta pan-tilt
                    Se lee APT
    else:
            Se busca APT
            Se orienta pan-tilt
            Se lee APT
    end while
    CalcularPosicion(DetectionArray)
end
```

Pseudocódigo 1: Funcionamiento simplificado del algoritmo desarrollado para leer un AprilTag

Siguiendo el Pseudocódigo 1, en primer lugar se inicia la librería de los *AprilTags y* los *scripts* que publican la imagen a tiempo real de la cámara del robot y su *CameraInfo* (AprilTags en ROS). Posteriormente, se inicia la lectura del *tag,* donde hay dos posibilidades:

- Se conoce la posición del *AptilTag*: se orienta la unidad pan-tilt hacia él, se lee y se obtiene la posición del *tag* respecto al móvil. En caso de que el *tag* no se encuentre donde se ha orientado la unidad pan-tilt se pasaría al siguiente punto.
- Se desconoce la posición del *AptilTag:* para estos casos se realizará un barrido completo hasta que se detecte un *tag,* donde se parará el pan y se leerá la posición

En último lugar se procede a calcular la posición del Robobo con la información obtenida de los *AprilTags*. Esta información consiste en la posición del *tag* respecto a la cámara en el sistema de referencia móvil de la cámara; y la orientación del *tag* respecto a estos mismos ejes. Sin embargo, para la navegación nos interesa que todas las posiciones estén referenciadas a un sistema de referencia global, es decir, al sistema de referencia de los *AprilTags*. Debido a esto, es necesario realizar un cambio de sistema de referencia.

Para ello, hay que calcular las diferentes proyecciones de los ejes fijos del *tag* en los ejes del móvil, matriz M, según lo valores x, y, z, w obtenidos en el *pose* devuelto por la librería de ROS.

$$M = \begin{bmatrix} X'_X & X'_Y & X'_Z \\ Y'_X & Y'_Y & Y'_Z \\ Z'_X & Z'_y & Z'_z \end{bmatrix} = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + wy \\ 2xy + 2xz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix} \quad (10)$$

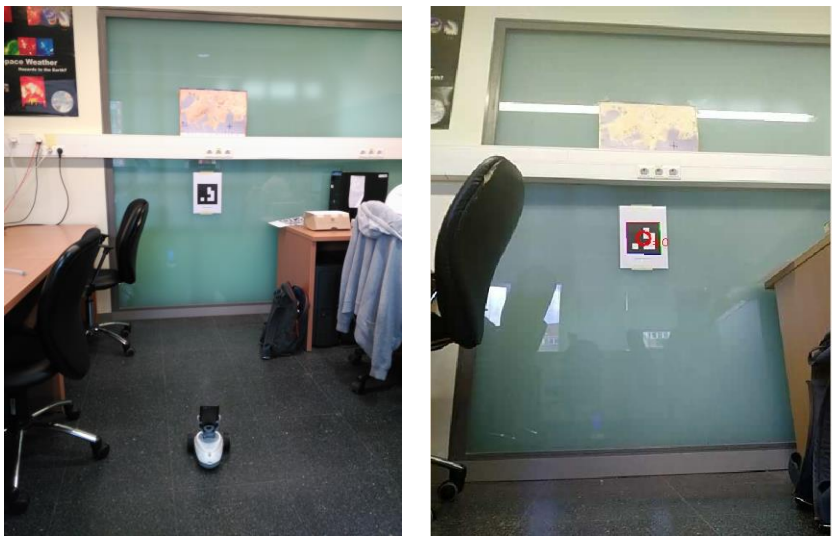
Para calcular el vector posición de la cámara (P), referenciado a los ejes fijos del marcador, se multiplicará dicha matriz M por el vector posición referenciado en los ejes móviles. De esta manera, obtenemos el vector deseado con el cual es posible ubicar el robot en el entorno de una manera más sencilla.

$$P = M \cdot \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (11)$$

Finalmente, la posición actualizada del Robobo (P_2) en coordenadas globales es la suma vectorial de la posición del *tag* y el vector P.

$$P'_{Robobo} = P_{tag} + P \quad (12)$$

Por último, en la Figura 19, se muestra una vista en tercera “persona” del robot y otra en primera mientras se está orientando gracias a los AprilTags.



a) *b)*

Figura 19: Orientación mediante AprilTags: Vista del entorno (a); y vista desde el punto de vista del robot (cámara interna del robot) (b)

5.3 Navegación

En este punto se explicarán los diferentes pasos que se han llevado a cabo hasta alcanzar la navegación que al final ha sido implementada en el Robobo. En primer lugar, se ha desarrollado la navegación mediante la odometría anteriormente explicada. Seguidamente, se ha complementado esta navegación con la localización mediante los AprilTags, obteniendo la denominada Navegación mixta con AprilTags.

5.3.1 Navegación mediante odometría

La principal función que tiene la odometría en este trabajo es ser capaz de crear un algoritmo al cual se le indique la posición y dirección actuales, la posición deseada, y realice las órdenes necesarias para que el robot se desplace a dicha posición objetivo. En el siguiente pseudocódigo se puede observar la lógica que se ha empleado para llevar a cabo esta navegación.

```
begin /* Mover Robot*/
    Registrarse en los servicios de Robobo MoveWheels y ResetWheels
    Se resetean los encoders.
    while distancia > Umbral distancia do:
            Calculo el angulo de
            If abs(Angulo Desvio) > Umbral AnguloGiro then:
                    Giro(). Mientras AnguloDesvio > Umbral AnguloGiro
                    Posición actualizada mediante CalculoTrayectoria
```

```
        If Umbral Angulo  < abs(Angulo Desvio) < Umbral AnguloGiro then:
                Arco(). Mientras AnguloDesvio > Umbral Angulo
                Posición actualizada mediante CalculoTrayectoria

        If distancia > Umbral_distancia then:
                Se desplaza el Robobo hasta llegar a la posición objetivo (se
                comprueba que siga enderezado en todo momento, encase
                de que no, se sale de esta función y volvería a entrar en el giro).
                Posición actualizada mediante CalculoTrayectoria.
    end for
end
```

Pseudocódigo 2: Lógica empleada para el desarrollo de la navegación mediante odometría

Se ha desarrollado la función *CalculoTrayectoria.* En esta función se han implementado todas las operaciones citadas en el apartado anterior. Consecuentemente, tiene como objetivo calcular la nueva posición conociendo la posición inicial y la distancia recorrida por ambas ruedas (última medida de los encoder menos la del paso anterior). Las funciones *Giro, Arco* y *Avance*, utilizarán esta función para actualizar la posición y saber cuándo han de terminar. Devuelve el valor actualizado de la posición y dirección.

El Pseudocódigo 2 contiene la lógica de funcionamiento de la aplicación desarrollada. Esta lógica consiste en lo siguiente:

En primer lugar, se resetea el valor de los encoders. Esto se realiza para evitar que el valor devuelto por los encoders crezca de forma ilimitada durante la navegación.

A continuación, se comienza un bucle que no finaliza hasta que el Robobo está a una distancia menor a la configurada como *Umbral distancia.* Una vez dentro del bucle se calcula la distancia a la que se encuentra de la posición objetivo y el ángulo que distan su orientación con la dirección hacia el objetivo.

Por un lado, se ha implementado una función, llamada *Arco*, que tiene como objetivo variar su orientación al mismo tiempo que se acerca a la posición objetivo. Esto se consigue ordenando al robot una velocidad mayor en una rueda que en la otra. Sin embargo, esta manera obliga a disponer de un gran espacio libre a su alrededor para poder describir el arco. Por ello, se ha realizado otra función con la que intentar disminuir este espacio libre que se requiere, llamada *Giro.*

Esta función consiste en únicamente dar velocidad a una de las ruedas, permitiendo así que el robot varíe su orientación sin apenas variar su posición. Cuando se llega a un cierto ángulo de desvío, definido como *Umbral AnguloGiro*, se sale de esta función y se continua la orientación mediante *Arco*. Ambas funciones utilizan *CalculoTrayectoria* para actualizar la posición y orientación estimada del Robobo y actuar en consecuencia.

Por último, se ha definido la función *Avance*. Esta función es la encargada de realizar un movimiento recto (misma velocidad a ambas ruedas) una vez el Robobo ya se ha orientado hacia la posición objetivo. No obstante, hay que tener en cuenta que es imposible que cuando se ha orientado al Robobo hacia la posición objetivo el ángulo de desvío sea nulo, sino que se ha configurado unos umbrales dentro de los que se considera aceptable. Consecuentemente, según el Robobo avance hacia la posición deseada, el ángulo de desvío se irá haciendo cada vez mayor. Por ello, se comprueba dicho ángulo en cada paso de tiempo y se redirecciona llamando de nuevo a la función *Arco* en caso de que sea necesario. Igual que en los casos anteriores, se actualiza la posición y orientación mediante la odometría (*CalculoTrayectoria).*

Como resultado, se ha obtenido una navegación que describe unos movimientos como los que pueden verse en la siguiente Figura:

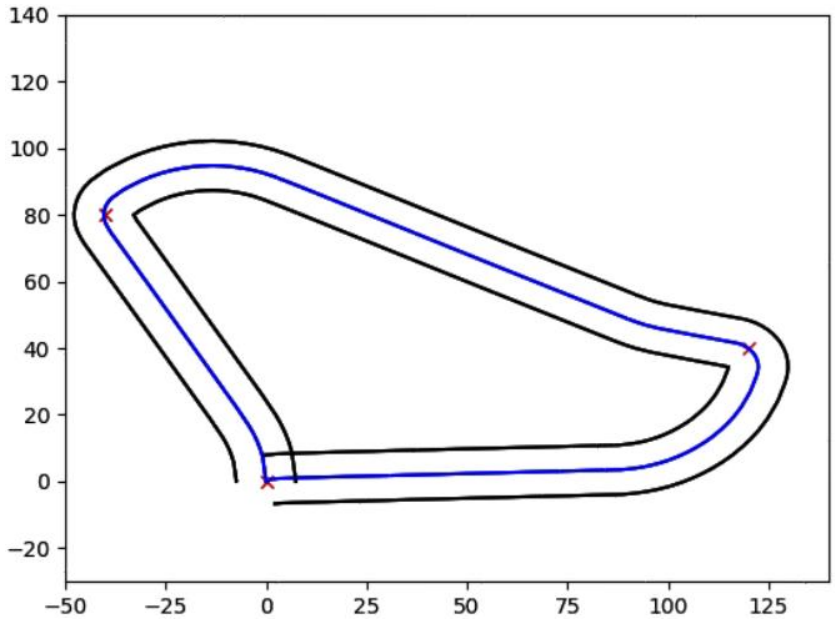


Figura 20: Recorrido que describe la navegación mediante odometría desarrollada

5.3.2 Navegación mixta con AprilTags

La gran desventaja de la Navegación mediante odometría es la ya mencionada deriva. Con el fin de mantener el error de posicionamiento entre unos límites aceptables se ha combinado esta navegación con la localización mediante AprilTags, obteniendo una Navegación Mixta. De esta manera se consigue que la navegación sea estable con el paso del tiempo y pueda utilizarse durante un tiempo limitado.

```
begin /* Navegacion mixta con APT*/
    Registrarse en los servicios de Robobo MoveWheels, ResetWheels y movePanTilt
    Se busca un APT y se lee. Posición inicial del robot
    while True do:
        Se resetean los encoders.
        MoverRobot (Posición, Dirección, PosiciónObjetivo). Mediante odometría
        Se actualiza error esperado.
        If error > Umbral error then:
            Se orienta el móvil hacia la posición del APT.
            Se lee APT
            Se ubica el robot.
    end for
end
```

Pseudocódigo 3: Funcionamiento simplificado de la navegación mixta con AprilTags

Siguiendo el Pseudocódigo 3, en primer lugar se trata de leer un *AprilTag* para obtener la posición en la que comienza la navegación. Por lo general, el robot desconoce la posición del tag y es necesario que realice un barrido.

Una vez calculada su posición inicial, el Robobo comienza la navegación de la misma manera que en el apartado anterior, es decir, únicamente mediante odometría (*MoverRobot*). En cada paso de tiempo se realiza una estimación del error que se ha acumulado hasta el momento (apartado 6.1). Cuando dicho error supera un umbral, a partir del cual se considera inadmisible, el robot se para, orienta su unidad Pan-Tilt hacia el *AprilTag* y se ubica de nuevo, reiniciando así el error cometido debido a la odometría.

Por otro lado, se ha implementado una lógica para que cuando el robot orienta la cámara hacia donde cree que está el marcador y no pueda leerlo vuelva a realizar un barrido. Esto puede deberse a un error demasiado grande en la posición estimada mediante odometría, o un obstáculo inesperado del entorno.

Con la implementación de la localización con *AprilTags* se consigue una navegación estable en el tiempo, algo que con la navegación mediante odometría era imposible. Como puede observarse en la siguiente Figura, durante los instantes iniciales de la navegación la estimación de la odometría se obtienen unos resultados mejores que con los *AprilTags*. Sin embargo, pasado un cierto tiempo los *AprilTags* se vuelven mucho más precisos.

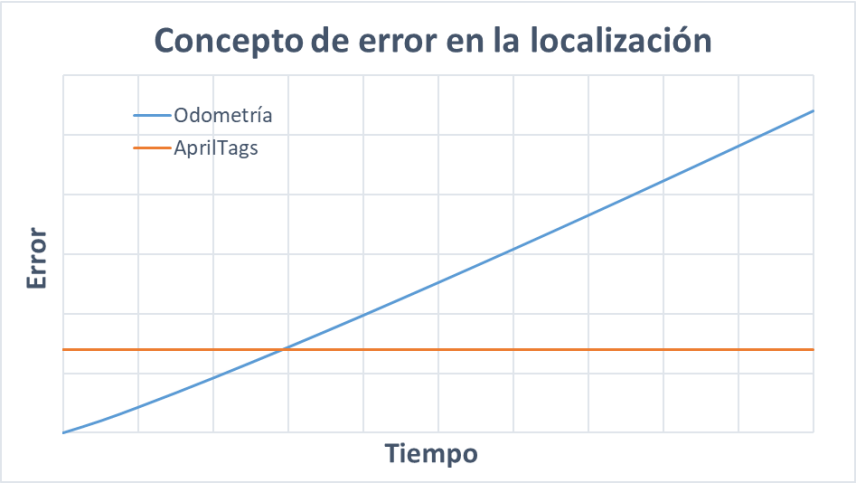


Figura 21: Concepto del error en la localización cometido por la odometría (azul) y los AprilTags (naranja)

Por otro lado, gracias a la utilización de la navegación con odometría la cámara del *smartphone* permanece disponible para otras tareas que se le requiere, además de disminuir significativamente el coste computacional relacionado a la navegación. Por todo ello, entre los dos tipos de navegación se obtiene una navegación resultante con mejores capacidades que cada una de ellas por separado.

5.4 Control de acceso

Como ya se ha comentado en el segundo apartado, uno de los objetivos de este trabajo es realizar una aplicación capaz de llevar a cabo un control de acceso. Con este fin, se ha utilizado la librería de detección de objetos de ROS citada en el apartado 4.3. Gracias a la función *tracker* incorporada, que realiza un seguimiento del objeto detectado y les pone un identificador, se puede realizar un seguimiento de cómo una misma persona avanza sobre la imagen. Tal y como se muestra en la siguiente Figura:

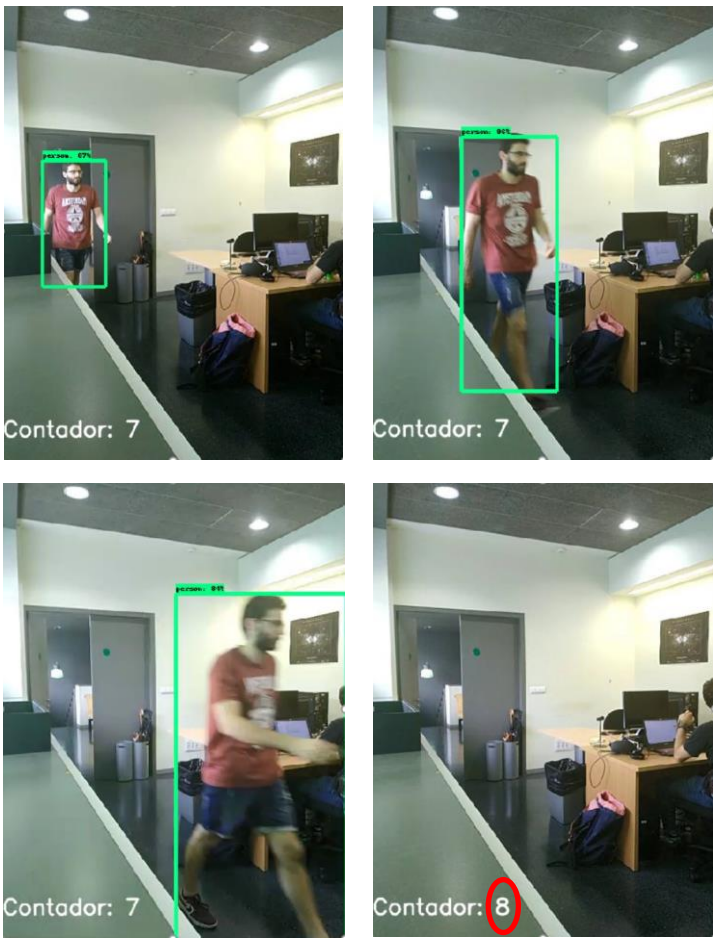


Figura 22: Prueba de control de acceso en el GII

La lógica que se ha empleado es la siguiente (Pseudocódigo 4): Cuando una persona es localizada por primera vez (las distinguimos gracias al identificador proporcionado por el *tracker*) se guarda la posición del centro de su recuadro, en píxeles, y el tiempo actual. Cada vez que esta persona vuelve a ser identificada, se almacena la nueva posición del centro (sobrescribiendo sobre la anterior, pero conservando la inicial) y actualizando el tiempo.

Cuando dicha persona no es localizada durante al menos dos segundos, se compara su posición final e inicial. Con esto, podemos conocer si la persona ha ido de izquierda a derecha, o de derecha a izquierda en la imagen; y, por tanto, si ha salido o entrado en la habitación de estudio. No obstante, en caso de que la posición inicial y final sean similares, se deduce que la persona ha entrado y salido en el campo de visión de la cámara por el mismo lugar. En estos casos, el contador de personas no se verá alterado.

begin */*Control de Acceso*/*
Inicializar librería de detección de objetos
Orientar unidad Pan-Tilt hacia la puerta de acceso
Se inicializa Contador
while *True* ***do****:*
If *Se detecta persona* ***then:***
If *Primera vez que se detecta esta persona* ***then:***
Se guarda su posición inicial
Seguimiento de todas las personas detectadas (Se guarda su posición actual)
Se actualiza el tiempo de última detección.

```
            for Persona in Personas:
                If Tiempo sin detectar > 2.0 then:
                    Se compara posición inicial con la final. Se comprueba
                    si ha salido o entrado.
                    If Sale then:
                        Se resta 1 al contador.
                    If Entra then:
                        Se suma 1 al contador.
                    Se almacenan los datos de la persona detectadas (Centralita).
                    Se borra sus datos en el array Personas.
            end for

    end while
end
```

Pseudocódigo 4: Lógica desarrollada para la implementación del control de acceso

Además, se ha desarrollado un *script* que recoge la imagen publicada en el *topic* de la librería de detección de objetos y le añade el valor de las personas que se han contado hasta al momento, es decir, el número de personas que se encuentran dentro de las oficinas. El valor del mencionado contador lo recibe a través de otro *topic*, en el cual publica el s*cript* donde se implementa la lógica anteriormente explicada.

5.5 Control de presencia

El apartado actual trata sobre el desarrollo de un control de presencia con las herramientas que se tienen a disposición, citadas en el apartado 4.4. Se va a comenzar explicando el funcionamiento general de este control de presencia y se va a continuar explicando la lógica con la que se decide a donde ha de ir el Robobo en cada momento. Por último, se termina con una explicación más en detalle de los pasos en el proceso de localización de las personas.

En el siguiente pseudocódigo se muestra el funcionamiento general del control de presencia realizado en este proyecto. En primer lugar, el Robobo decide en qué posición del entorno ha de ir a buscar personas gracias al *algoritmo de decisión* que se le ha implementado. A continuación, una vez se encuentra posicionado, el robot procederá a realizar un barrido donde tratará de localizar las personas presentes en el entorno. Por tanto, el control de presencia consta de dos partes fundamentales:

- Algoritmo que decide la posición donde ha de ir a buscar el robot.
- Localización de las personas (mediante barrido). A su vez, este proceso contiene 3 pasos:
    - Cálculo del ángulo de la persona respecto al robot.
    - Cálculo de la distancia a la que se encuentra la persona.
    - Identificación de posibles conflictos

```
begin /*Control de Presencia*/
    while True:
        Posicion ←Algoritmo de decisión
        Mover Robobo a Posicion

        for PosicionPan in PosicionesBarrido:
            for Persona in PersonasDetectadas:
                Se guarda el ancho delrecuadro que encuadra a Persona
                Se calcula el ángulo en el que se encuentra según su posición en la imagen.
        end for

        for Persona in PersonasAlmacenadas:
            Se calcula ángulo medio
            Se calcula ancho medio de su recuadro
            Se calcula la distancia a la que se encuentra en función del ancho
            If Persona → Conflicto then:
                Se almacena su ángulo en posiblesConflictos
        end for

        for PosiciónPan in posiblesConflictos:
            Se orienta pan a Posición
            Se vuelve a hace una lectura.
        end for
    end while
end
```

Pseudocódigo 5. Funcionamiento general del algoritmo que lleva a cabo el control de presencia

A continuación, se procede a explicar en más detalle el funcionamiento del sistema de control de presencia. Se comenzará por el algoritmo que decide a donde ha de ir el Robobo:

5.5.1 Navegación para la vigilancia

Dentro de la tarea de control de presencia, se ha desarrollado una lógica con la que se decide a qué posición debe desplazarse el Robobo en función de los datos que se han recogido hasta el momento (personas avistadas). En primer lugar, se han predefinido unas ubicaciones donde el Robobo deber realizar diferentes tareas, como leer lo *AprilTags* o realizar un a detección.

```
begin /*Algoritmo que decide el desplazamiento*/
    for Posición in PosicionesArray:
            Se calcula tiempo desde última visita
            Se revisan personas detectadas en Posición (y sus probabilidades)
            Cálculo de Valoracion en función de los datos recogidos anteriormente
            Valoracion→ ValoracionArray
    end for
    ValoracionArray → Se comprueba que Posición tiene el menor valor →Próximo desplazamiento
end
```

Pseudocódigo 6: Lógica del algoritmo que decide el próximo desplazamiento del Robobo en el control de presencia

Como puede verse en el Pseudocódigo 6, el algoritmo de decisión consiste en la asignación de un coeficiente a cada posición predefinida (denominado valoración) y decidir los siguientes desplazamientos en función de su valor.

De manera general, el procedimiento utilizado aumenta la frecuencia de visita que aquellas ubicaciones donde se haya encontrado una persona la última que se visitó. Además, cuanto menor sea la probabilidad que la persona permanezca donde se detectó, mayores serán las posibilidades de que este punto sea el próximo destino.

Para evitar que determinados puntos donde no se ha encontrado gente no sean visitados nunca, se ha configurado un tiempo de referencia en el cual coeficiente valoración se vuelve 0, sin poder llegar a valores negativos. De esta manera, se consigue que ningún punto esté sin ser visitado un tiempo mayor al tiempo de referencia configurado.

Teniendo en cuenta las prioridades que se han mencionado, se ha implementado una modelo de valoraciones (13) con la que poder elegir el próximo destino del Robobo:

$$\begin{cases} V_i = (t_{ref} - t_{trans}) \prod_{j}^{n} p_j & \forall\ t_{trans} < t_{ref} \\ V_i = 0 & \forall\ t_{trans} > t_{ref} \end{cases} \quad (13)$$

Donde V_i es la valoración del punto i; t_{ref} es el tiempo de referencia predefinido; t_{trans} el tiempo transcurrido desde que la ubicación i fue visitado por el robot por última vez; n el número de personas vistas en la ubicación i; y p_j la probabilidad que tiene la persona j de permanecer en la ubicación donde se localizó.

Por tanto, el punto que será el próximo destino del Robobo, es aquel que tenga menor valor de V. En caso de que dos puntos tengan el mismo valor, se elegirá en función del tiempo desde la última visita.

Por último, se ha buscado una expresión con la que calcular la probabilidad p de que una persona detectada en una cierta ubicación permanezca en allí. Lógicamente, según avance el tiempo dicha probabilidad ha de verse reducida. En este caso, se ha configurado para que dicha reducción sea de aproximadamente un 50% cada 5 minutos, obteniendo la siguiente ecuación:

$$p_{actual} = 0.998^{t_{trans}} \cdot p_{inicial} \quad (14)$$

5.5.2 Localización de personas

Una vez ya se ha posicionado y orientado el robot, se procede a la detección y localización de personas mediante visión artificial. En el siguiente pseudocódigo se muestra la lógica que sigue el algoritmo que lleva a cabo esta tarea.

```
begin /*Algoritmo localización de personas*/
    for PosiciónPan in Barrido:
        for Posición in PosicionesArray:
            Se buscan personas durante 1.5s
            Se calcula y guarda ángulo
            Se guarda ancho del recuadro
        end for
    end for
end
```

Pseudocódigo 7: Lógica que sigue el algoritmo encargado de la localización de las personas

Tal y como se indica en el Pseudocódigo 7, primeramente se realiza un barrido desde 55º hasta 305º. Se ha decidido que las distancia entre posiciones consecutivas del barrido sea de 25º. De esta forma, se eliminan las posibilidades de que una persona no sea detectada por no aparecer completamente en la imagen.

En cada una de estas posiciones del pan se lleva a cabo el siguiente proceso:

- Se permanece 1.5 s en cada posición para que el programa tenga tiempo suficiente para al menos procesar 5 fotogramas.
- Se calcula la dirección y distancia relativa a la cámara en el que se encuentra la persona detectada, si es que la hay.
- Se cuenta el número de veces que las personas han sido detectadas

Por tanto, para poder realizar la localización es necesario calcular tanto la dirección como la distancia a la que se encuentra la persona objetivo de la cámara.

5.5.2.1 Cálculo de la dirección relativa de la persona detectada

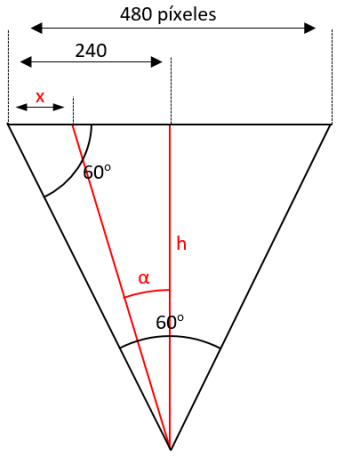


Figura 23: Cálculo del ángulo de la persona según su ubicación en la imagen

Para el cálculo del ángulo de la dirección de la que se encuentra la persona respecto a la cámara, es necesario conocer el centro del recuadro y el ángulo de visión de la cámara, 60º. A continuación, se muestran las ecuaciones con la que se calcula el ángulo α.

$$h = \frac{240}{\tan(30º)} \tag{15}$$

$$\alpha_{camara} = \text{atan}\left(\frac{240 - x}{h}\right) \tag{16}$$

Por tanto, ya se dispones de la dirección relativa a la cámara de la posición de dicha persona. Para conseguir una dirección respecto a los ejes globales hay que sumarle el ángulo del pan y el ángulo del vector que marca la orientación del robot:

$$\alpha_{global} = \alpha_{camara} + \alpha_{pan} + \alpha_{robot} \tag{17}$$

5.5.2.2 Cálculo de la distancia de una persona a la cámara

A continuación, es necesario estimar la distancia a la que la persona se encuentra de la cámara. Con este fin, se ha realizado unas pruebas para comprobar el ancho del recuadro en función de la distancia a la que se encuentre.

No obstante, se ha considerado necesario diferenciar entre aquellas que se encuentren de pie y aquellas que se encuentren sentadas, debido a que cuando la aplicación detecta una persona sentada en una silla también introduce en el recuadro las piernas de la misma. Por tanto, el ancho que se detecta es significativamente mayor. Gracias al ratio altura / ancho es posible la diferenciación entre de pie y sentado.

En las siguientes figuras se muestra las calibraciones que se han llevado a cabo:

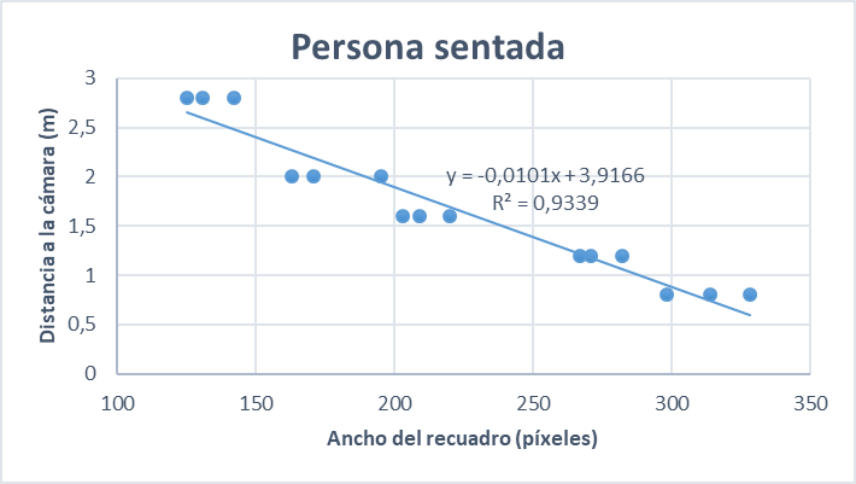


Figura 24: Calibración de la distancia a la cámara según el ancho de recuadro para una persona sentada

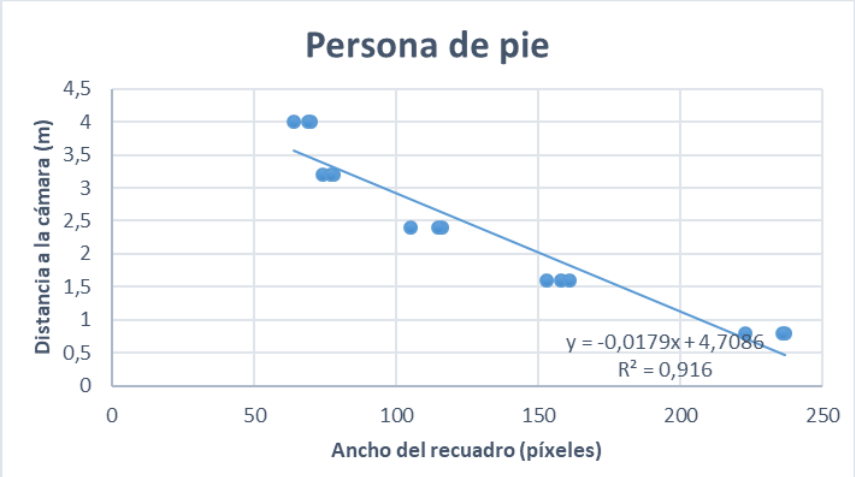


Figura 25: Calibración de la distancia a la cámara según el ancho de recuadro para una persona de pie

Así, obtenemos dos ecuaciones para el cálculo de la distancia. Para una persona sentada:

$$d_{pie} = -0.0082x + 3.3868 \quad (18)$$

Y para la gente de pie:

$$d_{sent} = -0.0179x + 4.7086 \quad (19)$$

Siendo x el ancho del recuadro en píxeles.

A pesar de que es una calibración que funciona de una manera bastante precisa en gran cantidad de ocasiones, puede verse muy afectada por como este orientada la persona. No obstante, tal y como se verá más adelante, los errores que se cometen no son excesivos y se considera una configuración satisfactoria.

Finalmente, conociendo la posición y orientación del robot; y el ángulo y distancia de la gente respecto a este, se puede obtener la posición en coordenadas globales de todas las personas que hayan sido identificadas en el barrido anteriormente realizado.

5.5.2.3 Triangulación de la posición

Debido a que el error cometido en el posicionamiento de las personas era, en gran parte, debido al cálculo de la distancia se ha desarrollado un algoritmo capaz de triangular la posición en función de únicamente el valor de ángulo captado desde dos posiciones diferentes (Figura 26):

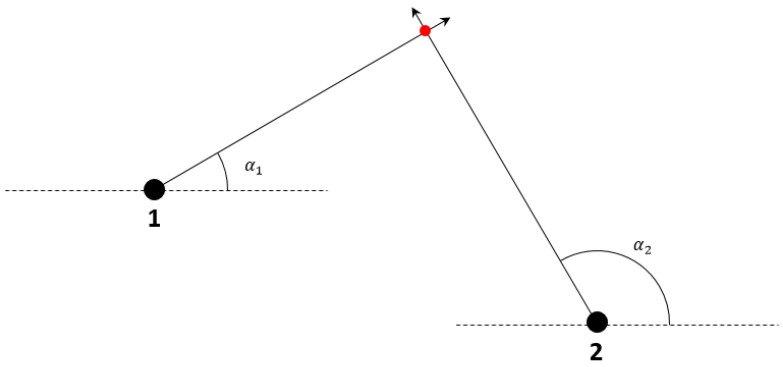


Figura 26: Estimación de la posición mediante triangulación

Sabiendo los dos ángulos, calculados mediante el proceso definido en apartado 5.5.2.1, y las dos posiciones desde donde fue detectada la misma persona se puede conocer la localización de la misma. Para ello, se calcula el punto de corte de las rectas definidas por el punto donde se encuentra el Robobo y el vector que apunta hacia la persona (definidos por α).

5.5.2.4 Puntos de conflicto

Una vez se han calculado lo ángulos medios de todas las personas detectadas, se ha desarrollado un algoritmo para minimizar la posibilidad de que se cometa un error. Dicho algoritmo filtra la información obtenida durante el barrido para obtener puntos conflictivos donde se haya podido cometer un error de lectura.

```
begin /*Algoritmo para evitar errores en deteccion*/
    for Persona1 in PersonasDetectadas:
        for Persona2 in PersonasDetectadas
            If Diferencia entre Angulos Persona 1 y 2 < Umbral then:
                Se considera punto de conflicto. Se guarda Angulos
        end for
        If Veces detectada < Umbral then:
            Se considera punto de conflicto. Se guarda Angulo
    end for
    for Conflicto in ConflictosArray:
        Se orienta el Pan-Tilt
        Se hace una lectura de 3 s
        Se guarda las detecciones
    end for
end
```

Pseudocódigo 8: Funcionamiento del algoritmo encargado de minimizar las posibilidades de que se cometa un error en la localización de las personas

Siguiendo el Pseudocódigo 8, en primer lugar, se comprueba si ha habido dos detecciones muy cercanas entre ellas. Si es el caso, se almacena como una posición de conflicto, ya que puede suceder como en la Figura 27. A continuación, en la sexta línea del pseudocódigo, se comprueba que todas las personas almacenadas hayan sido detectadas un número mínimo de veces. En caso contrario, se almacenan también como punto de posible conflicto.

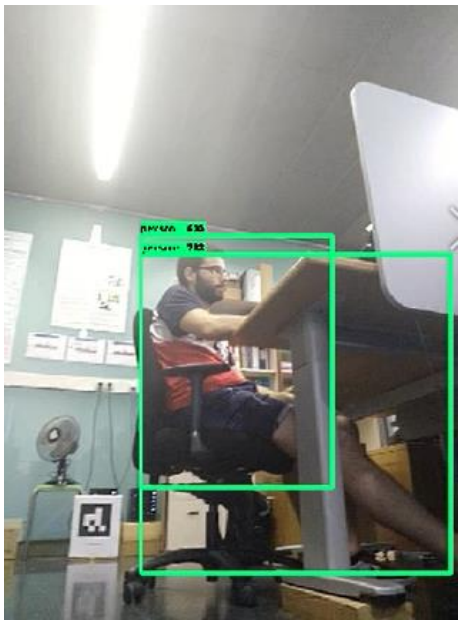


Figura 27: Posible error en la detección de personas

Por último, el Pan se orienta a las posiciones almacenadas como conflictivas y se realiza una lectura más extensa que en el barrido inicial, para darle tiempo a la aplicación a hacer una comprobación más precisa. Sin embargo, las condiciones para que una detección sea almacenada como válida es más restrictiva, ya que permanece más tiempo.

En la Figura 28 se expone un ejemplo de un control de presencia realizado en las oficinas del GII (edificio de talleres del campus de Ferrol de la udc). En ella se muestra un mapa representativo de las posiciones que recorre el robot en las instalaciones (puntos de color verde). Esta figura es actualizada cada vez que el robot cambia de posición, la cual se representa mediante un punto de color azul. En color rojo se muestra las posiciones (con una precisión, representada por el círculo) de las personas que han sido identificadas y ubicadas.

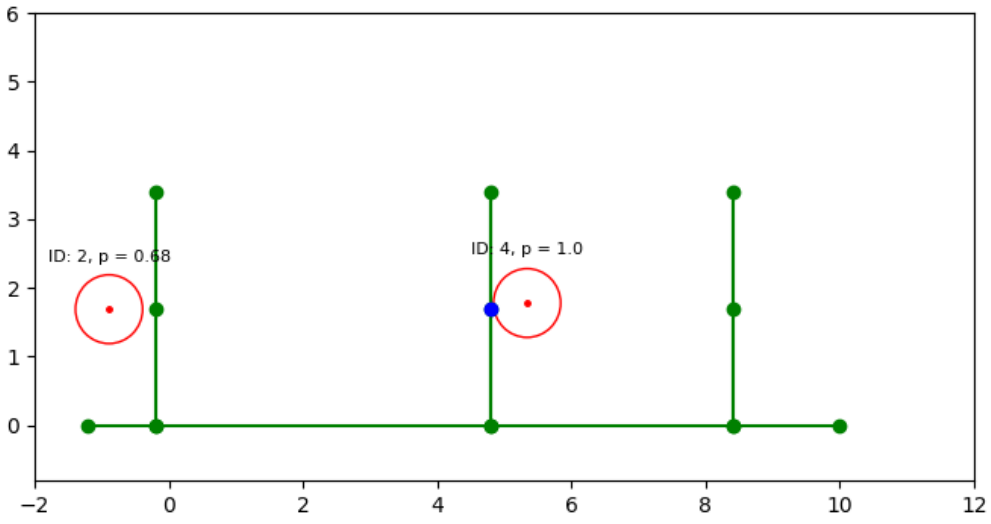


Figura 28: Ejemplo de la representación gráfica del control de presencia

Por último, se muestra una imagen captada por la cámara del *smartphone* del Robobo durante un barrido en busca de personas para ubicarlas en el entorno

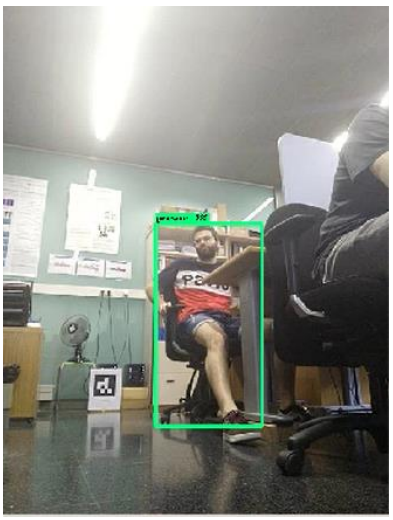


Figura 29: Imagen captada por la cámara durante un control de presencia

5.6 Estrategia de vigilancia coordinada

Partiendo de las aplicaciones que se han explicado anteriormente, navegación, control de acceso y control de presencia se pretende llevar a cabo una vigilancia coordinada donde colaboren los Robobos. Para ello, se ha elaborado una estrategia que permite que los Robobos puedan intercambiarse información a tiempo real y actuar en consecuencia.

Este tipo de estrategias colaborativas permiten alcanzar comportamientos más complejos, como grupo, y una mejor adaptación a la tarea que se quiera resolver. Sin embargo, los algoritmos resultantes de estos comportamientos coordinados suelen ser más complejos de desarrollar. Para este caso en particular, gracias a la coordinación de ambos Robobos (acceso y presencia) se ha podido obtener un comportamiento que permite al Robobo moverse de una manera mucho más eficiente.

```
begin /*Control de Acceso en vigilancia coordinada*/
    Inicializar librería de detección de objetos
    Se inicializa Contador
    while True do:
        If Se detecta persona then:
                If Primera vez que se detecta esta persona then:
                        Se guarda su posición inicial y ancho
            Seguimiento de todas las personas detectadas (Se guarda su posición y ancho actual)
            Se actualiza el tiempo de última detección.

        for Persona in PersonasDetectadas:
            If Tiempo sin detectar > 2.0 then:
                Se compara posición inicial con la final. Se comprueba si ha salido o entrado.
                If Sale then:
```

```
                                    Se resta 1 al contador.
                                    Se comprueba posición inicial → Se calcula pasillo
                            If Entra then:
                                    Se suma 1 al contador.
                                    Se comprueba posición final → Se calcula pasillo
                            Se almacenan los datos de la persona detectadas (Centralita).
                            Se borra sus datos en el array de Personas.
                    end for
            Se publica la información obtenida en un topic
            end while
    end
```

Pseudocódigo 9: Control de acceso en la vigilancia coordinada

En lo que se refiere al control de acceso, el Robobo encargado de esta función ya no está orientado a la puerta de acceso sino que se modifica hacia el interior de las oficinas, tal y como se muestra en la Figura 30 (Robobo 1). Con esto, se busca que este Robobo pueda identificar desde que pasillo han partido las personas que abandonan las oficinas, y a qué pasillo se han dirigido aquellas que han entrado. Por tanto, el algoritmo descrito en el Pseudocódigo 9, cuenta con dos partes fundamentales. La primera consiste en realizar el mismo control de acceso que se ha desarrollado anteriormente en el apartado 5.4. Es decir, contabilizar las personas que se encuentran dentro de las oficinas teniendo en cuenta como se han desplazado en la imagen.

```
begin /*Calculo del pasillo*/
    Inicializar librería de detección de objetos
    Se inicializa Contador
    while True do:
            for Persona in PersonasArray:
                    if Persona sale then:
                            Posicion inicial →Ángulo
                            Ancho recuadro en pos inicial → Distancia
                            Angulo, Distancia → Pasillo origen
                    if Persona entra then:
                            Posicion final →Ángulo
                            Ancho recuadro en pos final → Distancia
                            Angulo, Distancia → Pasillo destino
    end while
end
```

Pseudocódigo 10: Cálculo del pasillo contenido en el Control de Acceso

Por otro lado, una vez se ha identificado si la persona ha entrado o salido se procede a calcular de que pasillo provenía, en caso de aquello que han salido, o a que pasillo se dirigió, en caso de haber entrado. Por tanto, hay tres posibilidades (Pseudocódigo 10):

- Persona que abandona las oficinas: Según el ángulo y la distancia de su posición inicial respecto al robot se puede estimar de que pasillo ha provenido dicha persona.
- Persona que acceden a las oficinas: El procedimiento es idéntico al anterior, pero se realiza el cálculo del pasillo con su posición final.

- Su posición inicial y final en la imagen tienen una posición similar: No se realiza ninguna acción.

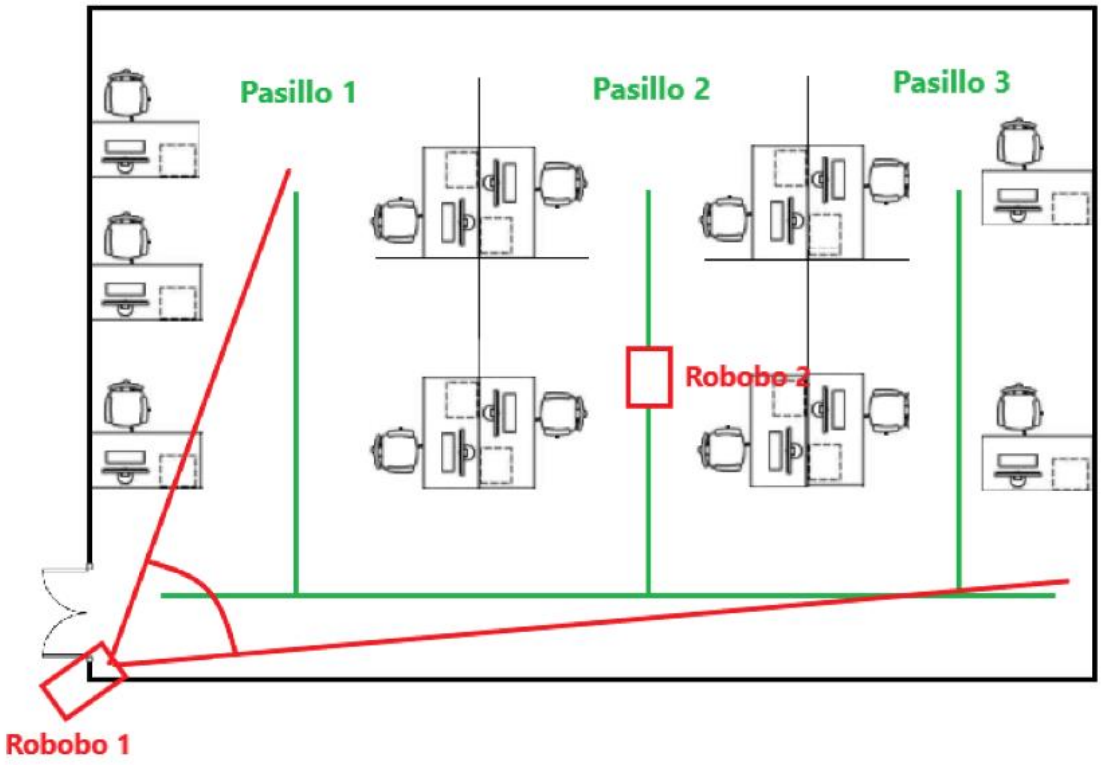


Figura 30: Configuración de los Robobos en la estrategia de vigilancia coordinada

Una vez se ha calculado el pasillo, del que provenía o al que se dirigía según su sentido, se transmite esta información al otro Robobo. Esta comunicación se realiza mediante una publicación a un *topic* al cual el segundo Robobo está suscrito.

A continuación, el Robobo encargado del control de presencia se dirige hacia la posición que le ha ordenado el otro a través de *topic*. Sin embargo, cada pasillo contiene 3 posiciones y ha de decidir cuál de ellas priorizar. La lógica que se ha elaborado está presente en el Pseudocódigo 11 y existen dos únicas posibilidades:

- La persona abandona las oficinas: Se comprueba aquellas ubicaciones donde anteriormente se han detectado personas, priorizándolas en función del número de personas y sus respectivas probabilidades. En caso de no haber detección en ninguna de ellas, se realiza una detección de todo el pasillo. En el momento en el que el Robobo identifica una “desaparición” se detiene y se da por terminado este proceso
- La persona accede a las oficinas: El procedimiento a seguir es el opuesto al anterior, es decir, se priorizan aquellas ubicaciones donde no haya personas. Sin embargo, si no hay ninguna persona asociada a ese pasillo, se realiza una detección de todo el pasillo. En cuanto una nueva persona sea detectada se da por terminado el proceso

```
begin /*Comprobación de las posiciones que ordena el control de acceso*/
    Suscripción al topic donde Robobo 1 publica el pasillo
    Se publica mensaje → Número de pasillo y si ha salido o entrado la persona
    Se desplaza el Robobo al pasillo
    If Persona sale then:
        If Existe persona localizada en pasillo then:
            Se realiza detección en ese punto
            If Se detecta una desaparición then
                Se sale del algoritmo
        else:
            Se realiza detección en todo el pasillo
    If Persona entra then:
        for Ubicación in PuntosSinDetecciones:
            Se realiza detección en Ubicación
            If Nueva detección then:
                Se sale del algoritmo
        If Hay detecciones en todos los puntos or En ninguno then:
            Se realiza detección en todo el pasillo
            If Nueva detección then:
                Se sale del algoritmo

    end while
end
```

Pseudocódigo 11: Lógica que prioriza la posición tras haber recibido una orden del Robobo encargado del control de acceso

Por consiguiente, se ha obtenido como resultado una estrategia donde el Robobo 1, el encargado del control de acceso, actúa como líder y el Robobo 2 como mandado. Con ello, se intenta sacar rendimiento a la perspectiva que dispone el Robobo 1, ya que se encuentra en un punto elevado desde donde alcanza a ver todo el pasillo principal. Sin embargo, no tiene visión de los tres pasillos secundario y, por tanto, requiere de la ayuda del Robobo 2 para llegar a ellos.

5.7 Centralita

Se ha desarrollado un panel de control que recopila información de interés obtenida de los sistemas de control no sé qué llamado centralita. Es un panel de informativo y de control al cual podría tener acceso cualquier persona de interés, como por ejemplo el conserje, servicios de emergencia, o cualquier persona que tenga acceso a la red. Esta centralita resulta de gran interés, puesto que con ella se puede ver en tiempo real la información que han recopilado hasta el momento los Robobo y actuar en consecuencia.

Por otro lado, toda la información que se muestra en los paneles es almacenada en la nube a tiempo real. Con ello se busca que, en caso de necesidad, se pueda acceder a un historial de lo que sucedido en el recinto de estudio (el laboratorio del GII para este caso en particular).

Se han implementado un total de 3 paneles diferentes: Panel sobre los Robobos, panel de estado y panel de acceso.

Panel sobre los Robobos (Figura 31)

En él se muestra la información necesaria (posición, función que está realizando, batería) para facilitar el trabajo de la persona que esté encargada de su mantenimiento. Por ejemplo, en caso de que un Robobo esté bajo de batería, el conserje del edificio podría localizar el robot fácilmente (ya que tenemos su última ubicación) e intercambiarlo por otro cargado.

ID Robot	Sala	Posicion	Funcion	Bateria	Actualizado
1	GII	[-2.0, -1.0]	Control acceso	54	21/06/19 18:04:52
2	GII	[-0.2, 1.7]	Control presencia	62	21/06/19 18:02:23

Figura 31: Panel con la información recogida sobre los Robobos empleados en la prueba

Panel de estado (Figura 32)

Seguidamente, se presenta un ejemplo del panel relativo a las personas que se han ido encontrado durante el control de presencia. Se muestran tanto el identificador interno que se les adjudica (ID), su posición en coordenadas globales (con el radio que indica la precisión de la misma), además de la sala en la que se encuentra; la probabilidad calculada de que permanezca en dicha posición; y el tiempo transcurrido desde la vez que fue detectada. Gracias a esta información recopilada se puede tener un control de las personas de una manera rápida y eficaz.

Acceso Sala	Contador	ID Persona	Sentido	Pasillo	Fecha y Hora
GII	5	2	Sale	2	25/06/19 18:04:52
GII	6	3	Entra	3	25/06/19 18:04:52
GII	5	5	Sale	2	25/06/19 18:04:03
GII	4	8	Sale	3	25/06/19 18:01:01
GII	5	7	Entra	1	25/06/19 17:58:33
GII	6	10	Entra	2	25/06/19 17:58:33

Figura 32: Panel con la información recogida sobre el control de presencia en el GII

Panel de acceso(Figura 33)

Por último, se ha desarrollado un panel relativo al control de acceso que muestra un *log* que guarda todas las personas que han ido entrado o saliendo, además de mostrar a qué hora fue detectada. En caso de que fuese necesario, la persona que esté conectada a dicho panel, como por ejemplo el conserje, puede interactuar con él y reiniciar el valor del contador manualmente mediante el comando ‘Reiniciar’.

Acceso Sala	Contador	ID Persona	Sentido	Fecha y Hora
GII	4	22	Sale	28/06/19 16:25:59
GII	5	25	Entra	28/06/19 16:26:10
GII	4	27	Sale	28/06/19 16:27:12
GII	3	28	Sale	28/06/19 16:27:13
GII	4	30	Entra	28/06/19 16:27:23
GII	5	31	Entra	28/06/19 16:27:25
GII	4	33	Sale	28/06/19 16:30:02
GII	5	36	Entra	28/06/19 16:30:13
GII	6	37	Entra	28/06/19 16:39:23
GII	5	38	Sale	28/06/19 16:40:52
GII	6	43	Entra	28/06/19 16:48:02
GII	7	45	Entra	28/06/19 16:48:02
GII	8	46	Entra	28/06/19 16:51:58

Figura 33: Panel con la información recogida sobre el control de acceso del GII

Por tanto, en este trabajo se ha implementado una arquitectura que implica la interconexión de varios dispositivos diferentes (Robobo, Smartphone y ordenadores) y una interacción entre ellos. Por otro lado, en caso de estar controlando varias salas simultáneamente, el coste computacional puede repartirse entre diferentes ordenadores gracias a la estructura de ROS. No obstante, el nodo central ha de estar en uno de ellos y el resto estar conectados a él.

6 VALIDACIONES

En el punto actual, se ha tratado de cuantificar los errores y tasas de éxitos de las aplicaciones anteriormente desarrolladas con el fin de poder actuar de una manera u otra en función de los resultados obtenidos para intentar minimizarlos. En primer lugar, se estudia el error cometido en la navegación mediante encoders. Se prosigue con el error cometido en la localización mediante AprilTags, y se termina con una validación sobre el control de acceso y presencia.

6.1 Navegación con odometría

Para el cálculo del error de posicionamiento que se comete en la navegación mediante odometría, se han hecho tres estudios con tres maneras diferentes de mover el robot. Mediante movimientos rectos, arcos y una combinación de ambos.

6.1.1 Movimientos rectos

Se comienza con una toma de datos resultantes de una prueba donde se utiliza únicamente las funciones *Giro y Avance* explicadas en el apartado 5.3.1. Dicha prueba consiste en realizar un movimiento rectilíneo hacia delante, de 0.8 metros, parar, dar la vuelta hasta estar orientado hacia el punto inicial y avanzar hasta alcanzar este último. Se anota la posición estimada por los encoders y en la que realmente está situado y se repite el proceso obligando al Robobo a reorientarse sólo hacia le segunda posición. Se realiza 3 veces este proceso, incluyendo cada uno 5 idas y vueltas del robot, es decir, aproximadamente 8 metros recorridos. A continuación, se muestra una tabla con los errores absolutos de una de las pruebas realizadas:

Tabla 1: Error de posición en la estimación mediante odometría. Movimiento recto.

Vuelta	Posición estimada		Posición real		Error absoluto (cm)
1	-0,96	4,71	-2,1	1,1	3,78
2	1,58	4,14	-3,5	0,5	6,25
3	-0,41	4,90	4,9	0,6	6,83
4	1,79	3,65	10,3	3,4	8,52
5	0,87	2,64	11,2	1,7	10,38

Por último, se representan los errores obtenidos en función de la distancia recorrida (Figura 34). Como se esperaba, el error crece de forma prácticamente lineal respecto a la distancia recorrida. Se obtiene un error de poco más de 1.10 cm por metro recorrido.

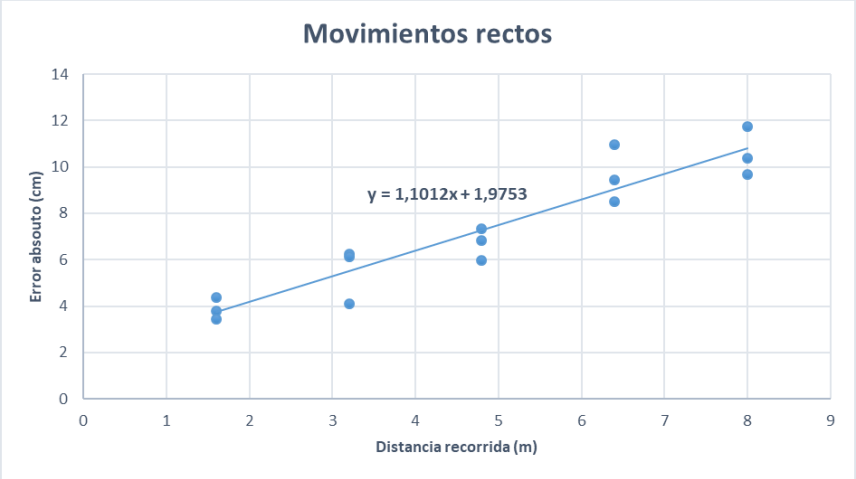


Figura 34: Error absoluto, en cm, en función de la distancia recorrida (m) que comete la estimación de la posición mediante odometría mientras se realizan movimientos rectos.

6.1.2 Movimiento curvo

En segundo lugar, se ha realizado una prueba que consiste en describir un movimiento curvo mediante la función *Arco* (apartado 5.3.1) que termina en el mismo punto donde se comenzó el movimiento, es decir, se realiza un movimiento circular. El radio del círculo descrito es de aproximadamente medio metro, obteniendo así una distancia de más de 3 metros por vuelta recorrida. De la misma manera que en el caso anterior, se calcula el error absoluto cometido entre la estimación y la real (Tabla 2)

Tabla 2: Error de posición en la estimación mediante odometría. Movimiento curvo.

Vuelta	Posición estimada		Posición real		Error absoluto (cm)
1	-1,57	0,30	0,2	-1,6	2,59
2	-1,00	0,48	0,3	-2,2	2,98
3	-1,23	0,33	1,3	-2,9	4,10
4	0,83	0,34	1,2	-3,6	3,96
5	1,01	0,41	1,3	-3,9	4,32

En la Figura 35, se puede observar que se comete un error mucho más pequeño. Aproximadamente de 0.22 cm por metro recorrido. Esta clara diferencia, en favor del movimiento, se debe a las múltiples paradas que se realizan en el movimiento recto. Estas frenadas favorecen que las ruedas deslicen y por tanto se realice un movimiento con el robot que es imposible que sean identificadas por los encoders.

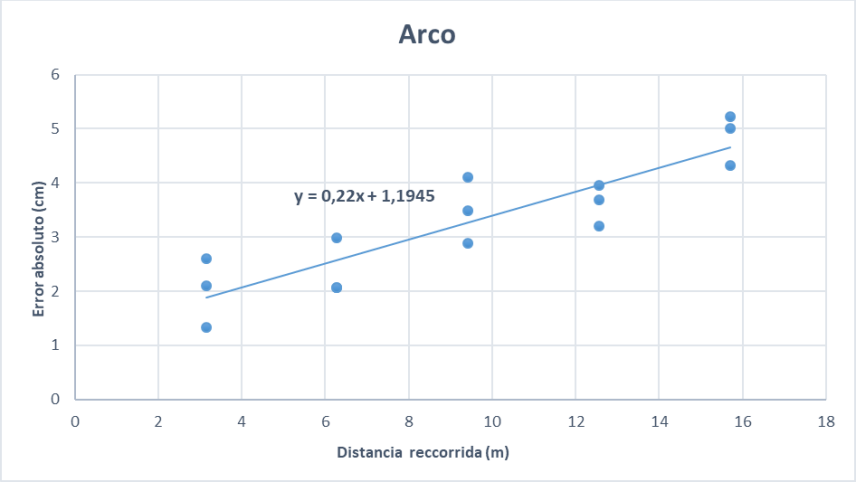


Figura 35: Figura 36: Error absoluto, en cm, en función de la distancia recorrida (m) que comete la estimación de la posición mediante odometría mientras se realizan movimientos circulares.

6.1.3 Combinación de ambos movimientos

Como ya se ha mencionado en el apartado anterior, el error cometido en el desplazamiento mediante movimientos rectos es del orden de 5 veces al error cometido con movimientos circulares. Con el fin de configurar una navegación con el mínimo error posible, se ha implementado una navegación que incluye una combinación los dos movimientos. De esta manera se evitan las frenadas del movimiento recto, pero se reduce el arco de giro que implica el movimiento mediante *Arco.* Igual que en los dos casos anteriores se muestra una tabla con los errores cometidos en una de las pruebas (Tabla 3):

Tabla 3: Error de posición en la estimación mediante odometría. Movimiento combinado.

Vuelta	Posición estimada		Posición real		Error absoluto (cm)
1	1,15	-1,99	1,0	2,0	3,99
2	1,49	0,51	4,6	1,0	3,18
3	-0,62	-0,22	-9,6	3,1	9,58
4	-0,44	0,68	15,6	4,7	16,53
5	0,42	-1,96	19,4	-5,6	19,33

En la gráfica que se encuentra más adelante, se puede ver que los resultados obtenidos han mejorado más de un 25%, de 1.10 cm /m a los 0.80 que se muestran en la Figura 37.

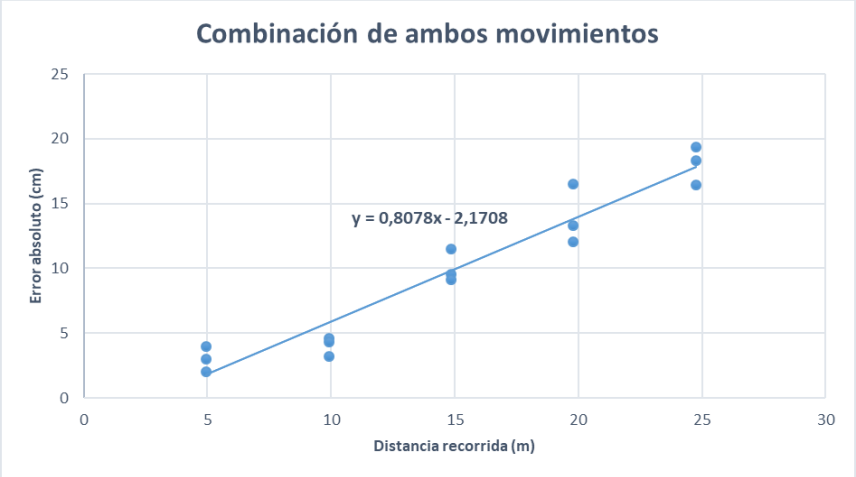
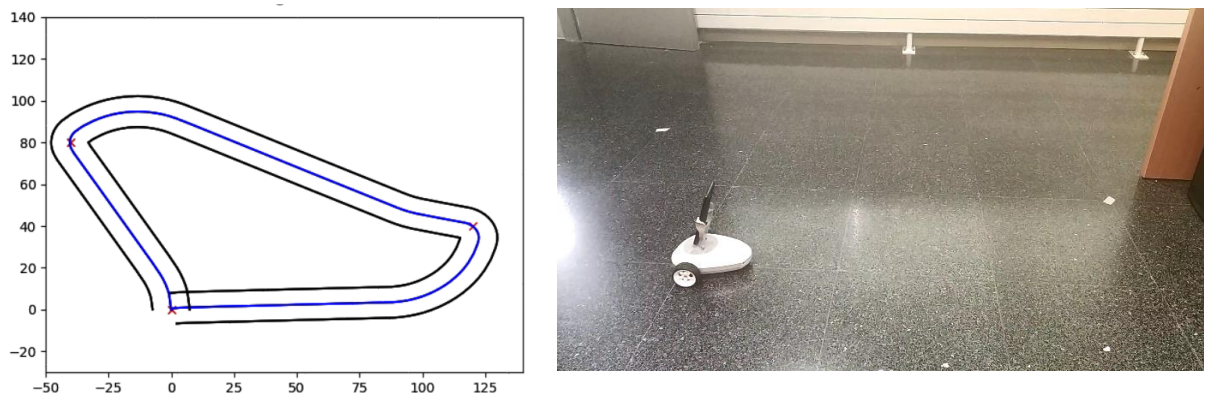


Figura 37: Figura 38: Figura 39: Error absoluto, en cm, en función de la distancia recorrida (m) que comete la estimación de la posición mediante odometría mientras se realizan un desplazamiento combinando ambos movimientos

Para la realización de esta prueba, se ha hecho que el Robobo recorra 5 vueltas de un circuito (Figura 40), de más de 4 metros por vuelta, un total de 3 veces. No obstante, este circuito consta de dos posiciones a mayores de la inicial



a) *b)*

Figura 40: Estimación de la posición del Robobo mediante la odometría basada en los encoders (a) mientras el Robobo se está moviendo por el entorno real (b) durante las pruebas para medir el error absoluto cometido

Cabe destacar que estas pruebas han sido realizadas en un terreno donde el agarre de las ruedas es considerablemente bueno y, por tanto, los errores se ven bastante reducidos ya que el gran problema de esta odometría es el deslizamiento. En otras condiciones más desfavorables estos errores podrían verse aumentados y habría que actuar en consecuencia, es decir, aumentar la frecuencia con la que se recurre a la localización mediante AprilTags.

6.2 Error en la localización mediante AprilTags

En primer lugar, se procede a explicar el procedimiento que se ha llevado a cabo para las mediciones que posteriormente servirán para sacar conclusiones. Dicho procedimiento consta de dos pasos: la toma de medidas en diferentes distancias y oblicuidades; y la corrección del error debido al posicionamiento imperfecto del *tag* y del robot para la prueba. Siendo la oblicuidad el ángulo que distan el vector posición, con origen el centro del AprilTag y terminal el robot, y el vector perpendicular al plano del AprilTag.

En referencia al primer paso, la distancia al tag se ha variado desde los 50cm hasta los 200, con medidas cada 50cm. Además, con el objetivo de variar la oblicuidad, se ha desplazado el robot hacia los lados, por la línea paralela al plano del *tag* (5 y 35cm en las dos distancias más cercanas; 10 y 35cm a metro y medio; y 10 y 45cm a la distancia de dos metros). Todas los datos con los que se trabaja en esta sección (Anexo I) están en centímetros y grados.

Respecto al segundo paso, cuando se colocan los diferentes elementos de la prueba es evidente que es imposible que estos sean colocados de manera ideal. Consecuentemente, es necesario procesar los datos de tal manera que se minimice su influencia imperfección en el cálculo de los errores, ya que un pequeño ángulo de desviación puede significar un gran error a una larga distancia. Por ejemplo, un simple error de 3 grados en el posicionamiento del *tag*, a una distancia de 3 metros, implica un error de más de 10cm.

Como se puede observar en la Figura 41, se toma como referencia el punto leído en la posición central y se calcula donde deberían estar el resto de puntos, teniendo en cuenta que cuando se desplaza el robot debería hacerlo en la dirección perpendicular a la recta que une nuestro punto de referencia y el centro del *tag.* En la siguiente Figura se muestran los puntos que teóricamente debería leer la aplicación (rojo) y los puntos que leería en caso de que todos los elementos de la prueba estuviesen perfectamente alineados (negro). Los puntos estimados se calculan mediante las siguientes ecuaciones:

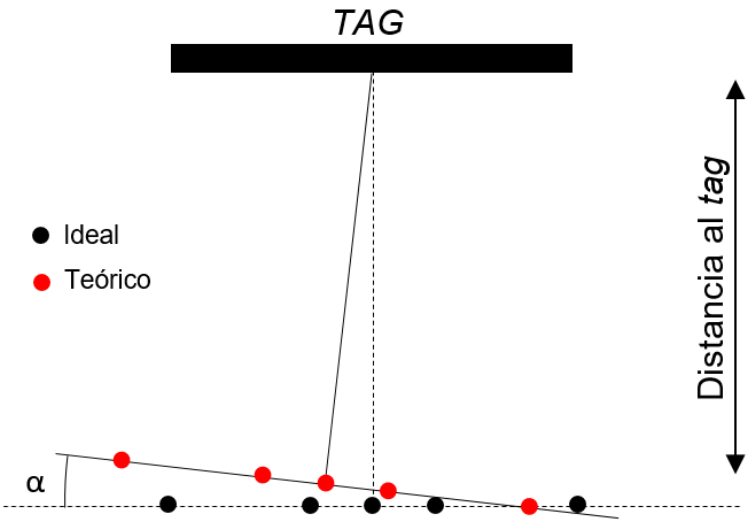


Figura 41: Corrección del error debido al posicionamiento imperfecto del tag y de la cámara.

Posteriormente, se calcula el error absoluto entre las posiciones que debería leer, según los puntos teóricos calculados, y los que realmente lee. Dicho error es la distancia entre ambos puntos. Con estos datos se deducirá que tamaño de AprilTags es conveniente para nuestro trabajo y como tendrá que comportarse el robot para intentar que el error de posicionamiento cometido nunca llegue a ser significativo.

6.2.1 Elección del tamaño del tag

El primer problema que se trata de resolver sobre la precisión de la localización mediante los AprilTags es el tamaño que ha de tener este. Se han barajado dos posibles opciones: de 15.1 cm de lado (impreso en A4) y de 20.2 (impreso en A3).

Por un lado, se espera que en cualquier condición de trabajo (distancia, oblicuidad, iluminación, etc.) se comporte mejor el marcador de tamaño. No, obstante, un tag de mayor tamaño implica una distancia mínima mayor a partir de la cual es legible. Por otro lado, teniendo en cuenta la calidad del sensor óptico empleado es mediocre, una *tag* pequeño puede ser problemático a largas distancias. Por tanto, con el fin de llegar a una solución de compromiso, se han realizado una toma de datos con la que poder medir el error de los *tags* en distintas condiciones.

En la Tabla 4 se muestran los errores absolutos promedio en función de la distancia al marcador y el tamaño del mismo.

Tabla 4: Tabla resumen de los errores mostrados en el Anexo I

Distancia al tag	***Error promedio (cm)***	
	Tag 15.1	***Tag 20,2***
50	*2,24*	*2,76*
100	*3,75*	*4,33*
150	*5,89*	*6,01*
200	*21,67*	*10,33*

Seguidamente, en la Figura 42, se puede observar la representación gráfica de los datos que contiene la tabla anterior:

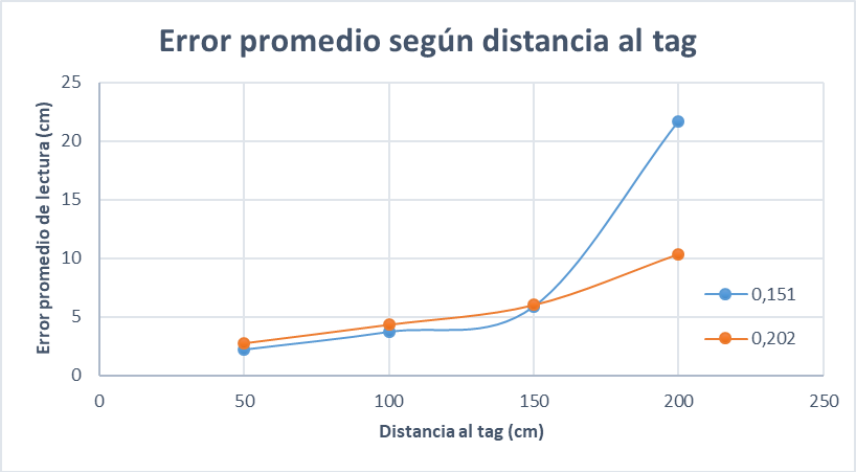


Figura 42. Error promedio de posicionamiento mediante AprilTags según la distancia al mismo

Para distancias cercanas el comportamiento es prácticamente idéntico para ambos tamaños, incluso mejor para el *tag* más pequeño. Esto se debe a las distorsiones que se producen en el perímetro de la imagen de la cámara. No obstante, a partir de metro y medio la precisión del de 15.1 se ve menguada significativamente, mientras que el de 20.2 sigue manteniéndose aceptable hasta los dos metros. Por todo ello, se ha escogido el *tag* de 20.2cm como mejor opción para ser colocados en el entorno de trabajo.

6.2.2 Estrategia en función de los resultados obtenidos

Una vez elegido el tamaño de *tag* que se va a emplear, es necesario desarrollar una estrategia a la hora de localizar el robot para evitar que cometa errores demasiado significativos, lo que podría ser un gran problema para la resolución de la tarea propuesta.

Como se puede ver en la Figura 42, el *tag* de 20.2cm de lado se mantiene en un error aceptable hasta los 2 metros de distancia (menos de 10 cm). Sin embargo, para asegurar que el error no crezca en exceso, o al menos no supere el umbral de los 10 cm, nunca se realizará una lectura de un *AprilTag* a una distancia mayor de un metro y medio, siempre y cuando sea posible.

Por otro lado, se han desglosado los valores obtenidos de error según la distancia y la oblicuidad para poder sacar conclusiones sobre cómo afecta esta última y de esta manera poder minimizar el error que esta pueda conllevar. A continuación, se muestra la gráfica donde se puede ver el error cuantificado:

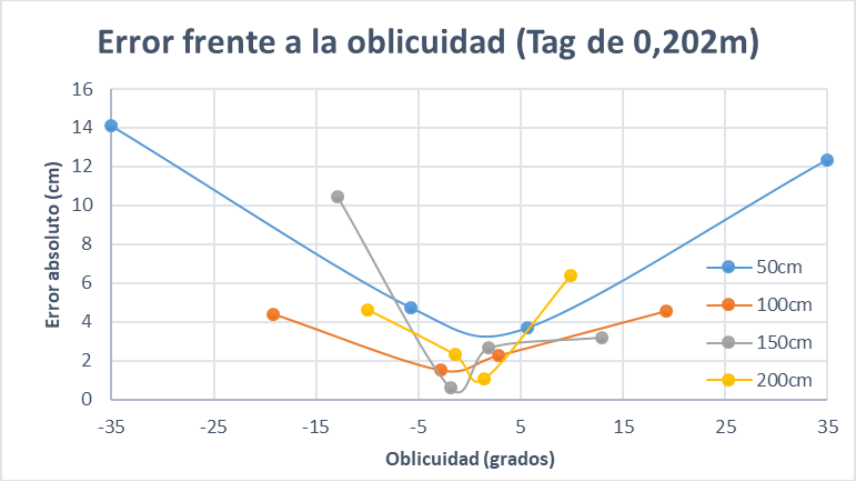


Figura 43: Error absoluto de lectura absoluto en función de la oblicuidad. Para diferentes distancias

Como se muestra en la figura anterior, cuanto más oblicua sea la posición desde donde se lee el *tag* mayor será el error que se está cometiendo en la lectura. Consecuentemente, se ha programado el robot de tal forma que evite realizar una lectura con una oblicuidad superior a 10º, ya que se considera que debajo de este umbral las lecturas son satisfactorias. Además, los *tags* que se han colocado en el entorno se han puesto de tal manera que esto sea posible.

6.3 Control de acceso

En este subapartado se busca estudiar la eficacia que tiene el algoritmo de control de acceso desarrollado. Para comenzar, se va a posicionar al Robobo en un lugar donde pueda tener una buena visión de la entrada de las oficinas del GII. Se realizan un total de 5 pruebas que consisten en contar tanto las personas que entran como las que salen del GII durante una hora y se comparan con las que realmente han entrado y salido.

Tabla 5: Cálculo de la tasa de éxito del control de acceso desarrollado en este trabajo

Prueba	***Detectadas***		***Reales***	
	Entran	**Salen**	**Entran**	**Salen**
1	13	14	13	14
2	18	18	18	18
3	14	11	15	11
4	21	18	21	19
5	17	12	17	12
Total	**83**	**73**	**84**	**74**

Como se muestra en la Tabla anterior, lo resultados que se han obtenido son considerablemente buenos. En 3 de las 5 pruebas que se han llevado a cabo el algoritmo ha detectado exactamente el mismo número de personas que las que realmente han pasado por la entrada de las oficinas. En las dos pruebas restantes únicamente ha dejado de detectar una persona en cada una. Si se agrupan los resultados obtenidos para las 5 pruebas, el algoritmo ha detectado 83 personas que han entrado y 73 que han salido, un total de 156. Por otro lado, realmente han entrado 84 y salido 74, con un total de 158. Por tanto, se puede calcular la tasa de éxito como el cociente de estos dos valores, obteniendo un resultado de más de un 98%. Este valor es un resultado realmente satisfactorio.

Sin embargo, estos resultados se han obtenido en un entorno donde las condiciones de trabajo son bastante habituales. Es decir, la gente no suele ir en grupos de más de 2 personas y a un paso calmado. Cuando se somete al algoritmo a pruebas en las que pasan grupos, de por ejemplo 5 personas, que se solapan entre ellos, dificulta notablemente su detección, pudiendo bajar su tasa de éxito considerablemente. Además, si una persona pasa a una velocidad alta, puede que la aplicación no sea capaz de detectarlo más de una vez mientras se encuentra en su campo de visión. Esto último se debe a la capacidad de cálculo de la que se dispone, ya que la librería de OpenCV únicamente es capaz de publicar en el *topic* del *tracker* una vez cada aproximadamente 0.35 segundos. Esta frecuencia, como ya se ha demostrado, en condiciones normales es más que suficiente, pero insuficiente para otras.

6.4 Control de presencia

En lo relativo al control de presencia, se ha realizado unas pruebas que consiste en ubicar a una persona en unas posiciones conocidas y comprobar cómo responde el sistema, es decir, con qué precisión es capaz de ubicarlas en el espacio. Se van a realizar las mismas pruebas tanto para la ubicación calculada según el ancho del recuadro (apartado 5.5.2.2) como para la calculada mediante triangulación (apartado 5.5.2.3).

Respecto a la posición calculada según la distancia estimada a la cámara, en la Tabla 6 contiene los errores absolutos cometidos en la prueba que peores resultados se han obtenido. Estas pruebas consisten en ubicar a una persona en 6 posiciones diferentes, variando distancia y ángulo tal y como se muestra en la tabla.

Tabla 6: Error cometido en el posicionamiento de personas mediante visión artificial. Cálculo según el ángulo y distancia a la cámara

Medición	***Posición estimada***		***Posición real***		***Error absoluto (cm)***
1	-18,24	135,79	0	100	40,17
2	-52,56	67,29	-50	100	32,81
3	67,17	100,64	50	100	17,18
4	21,65	174,23	0	200	33,66
5	-78,56	166,89	-50	200	43,73
6	65,28	177,45	50	200	27,24
				Promedio	**32,47**

El fin de esta prueba es conseguir un radio alrededor de la posición estimada en el que poder asegurar que la persona detectada se encuentra. Para el método del cálculo de la distancia mediante el ancho del recuadro (Tabla 6) se ha obtenido como resultado de las pruebas un error absoluto medio de 32.47 cm. Sin embargo, ha tenido un máximo de casi 43 cm. Por tanto, se ha decidido que el radio de precisión que será utilizado es de 50cm

A continuación, se realiza las mismas pruebas. Sin embargo, es este caso es necesario ubicar el Robobo en dos posiciones diferentes para poder realizar la triangulación. A diferencia del caso anterior, el error que se comete es debido únicamente al error durante el cálculo de ángulo en el que se encuentra la persona detectar. En la Tabla 7, se puede observar los peores resultados obtenidos de las 3 diferentes pruebas que se han llevado a cabo.

Tabla 7 Error cometido en el posicionamiento de personas mediante visión artificial. Cálculo según triangulación

Medición	***Posición estimada***		***Posición real***		***Error absoluto (cm)***
1	-9,27	96,66	0	100	9,85
2	-49,47	55,88	-50	50	5,90
3	55,58	51,48	50	50	5,77
4	-2,94	187,75	0	200	12,60
5	-63,38	210,09	-50	200	16,76
6	58,82	190,09	50	200	13,27
				Promedio	**10,69**

Para el caso de la triangulación, el error que se comete en el posicionamiento de las personas es mucho menor (10.69 cm). No obstante, se han obtenido un máximo de casi 17 cm. Por todo ello, se ha decido configurar la aplicación con un radio de 25 cm en el caso de que la posición haya sido estimada mediante triangulación.

Finalmente, en la Figura 1 se puede observar la influencia de estos radios en la resolución de la tarea. En esta gráfica se enseña un ejemplo en el cual se han detectado una persona desde una única posición (circunferencia grande) y otra que ha sido detectada desde dos diferentes (circunferencia pequeña)

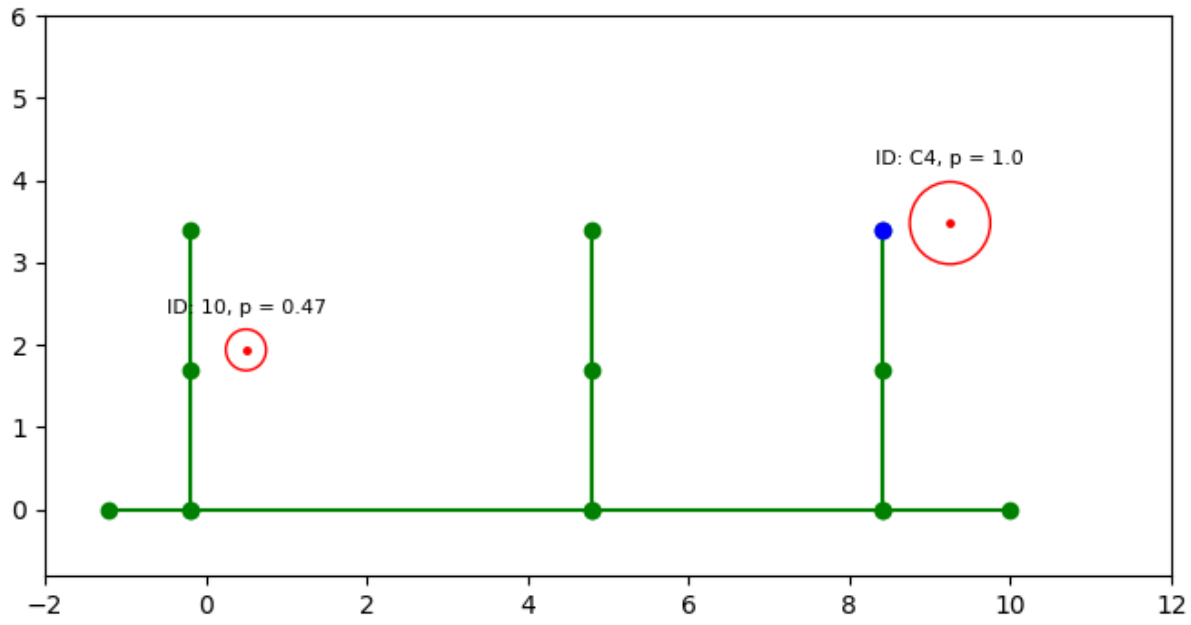


Figura 44:Ejemplo de la diferencia entre la estimación de la posición mediante la distancia a la cámara y la triangulación

7 RESULTADOS

En este último capítulo se presentan los resultados obtenidos tras combinar todas las aplicaciones que se han desarrollado en este trabajo (apartado 5). Es decir, se va a realizar de forma simultánea un centro de acceso y de presencia por las oficinas del GII, utilizando dos Robobos. Primeramente, se va a hacer una descripción del problema a resolver y, finalmente, se mostrarán los resultados obtenidos en omento dado.

7.1 Descripción del problema

Para llevar a cabo la lógica que se ha implementado, primero es necesario conocer el problema en particular que se va a resolver. El lugar donde se ha realizado la prueba final es el mismo donde han hecho todas las validaciones, las oficinas del GII del edificio de talleres del Campus de Ferrol (UDC).

Como puede verse en las Figura 46 , los puestos de trabajo están dispuestos de tal forma que las personas no pueden ser vistas desde el pasillo principal. Además, hay que tener en cuenta que el punto de vista de vista del Robobo se encuentra en todo momento a escasos centímetro por encima del suelo, lo que dificulta notablemente la detección de las personas.

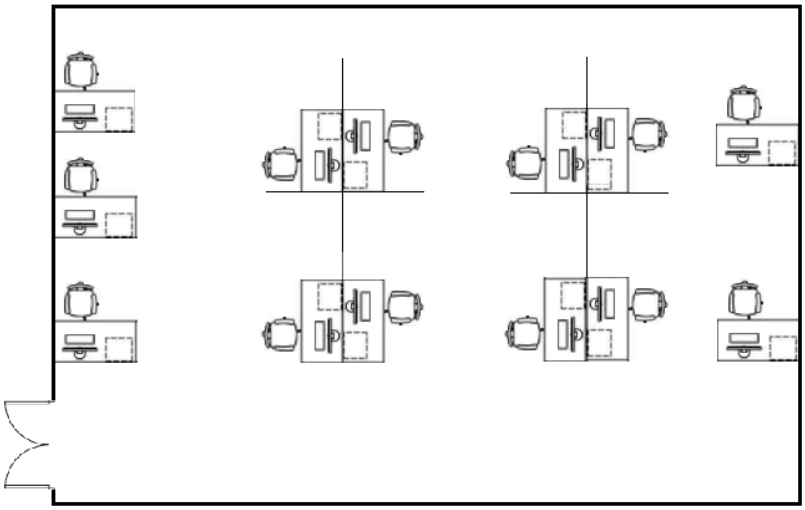


Figura 45: Boceto de la sala de oficinas del GII

Figura 46: Sala de oficinas del GII

Para solucionar este problema, se han configurado un total de 11 posiciones por donde se desplazará el Robobo durante el procedimiento (Figura 47 en color verde). Estas posiciones la aplicación las tiene almacenadas internamente en el código fuente.

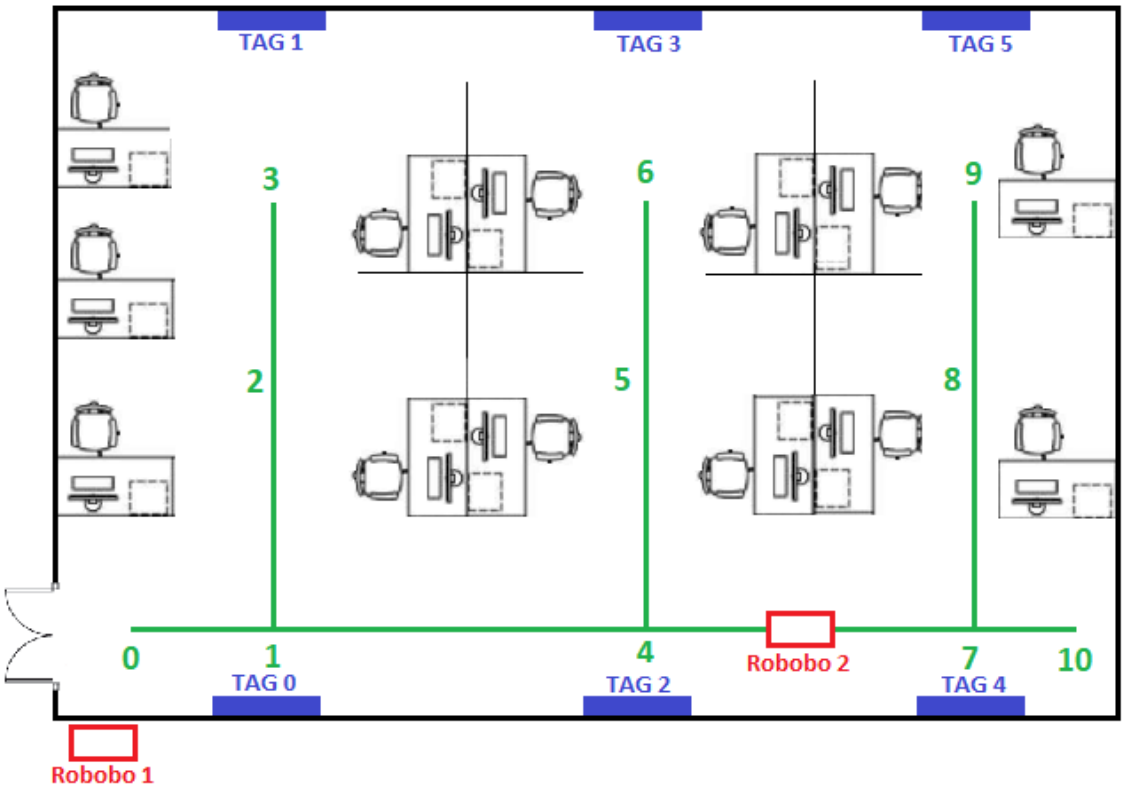


Figura 47: Boceto de la sala de oficinas del GII indicando las posiciones configuradas y la puerta donde se realiza el control de acceso

A continuación, a cada punto de nuestro recorrido se le asigna unas características con las que poder definir el comportamiento del Robobo (Tabla 8).

En primer lugar, se indica al sistema en que posiciones se ha de leer los tags (representados en azul en el boceto de la Figura 47). En estas localizaciones, siempre que el Robobo pase por ellas, se parará, reiniciará el valor de posición mediante el AprilTag y proseguirá su camino. Además, se le indica el ID del *tag* que debe leer en cada una de estas ubicaciones para que se pueda orientar hacia él, ya que también se le indica la posición de cada uno de los *tags*.

Por otro lado, también se indica en qué posiciones se quiere que se realice las detecciones. Esto únicamente se emplea para saber qué puntos han de ser valorados según la expresión desarrollada en el apartado 5.5.1.

Tabla 8: Asignación de las posiciones donde se realizan los

Posición	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Localización	No	Sí	No	Sí	Sí	No	Sí	Sí	No	Sí	No
ID APT	-	0	-	1	2	-	3	4	-	5	-
Detección	No	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	No

7.2 Resultados obtenidos

Con el problema ya definido y configurado, se procede a realizar un ensayo en el que se validará que la combinación de todas las aplicaciones desarrollas en este trabajo funcionan según lo esperado.

Por un lado, se va a comentar los resultados obtenido en el control de presencia y, por el otro lado, los del control de acceso.

7.2.1 Control de presencia

Los resultados que se obtienen del control de presencia pueden diferenciarse 2 tipos: La representación gráfica y la información pensada para completar el panel presente en la centralita.

Respecto a los resultados gráficos (Figura 48), están pensados para que el usuario que esté visualizando la información pueda procesarla de una manera más rápida y cómoda que si estuviesen en forma numérica. Se puede acceder a ella desde cualquier ordenador que se encuentre conectado a la red y conozca la dirección IP del nodo central de la arquitectura implementada en ROS.

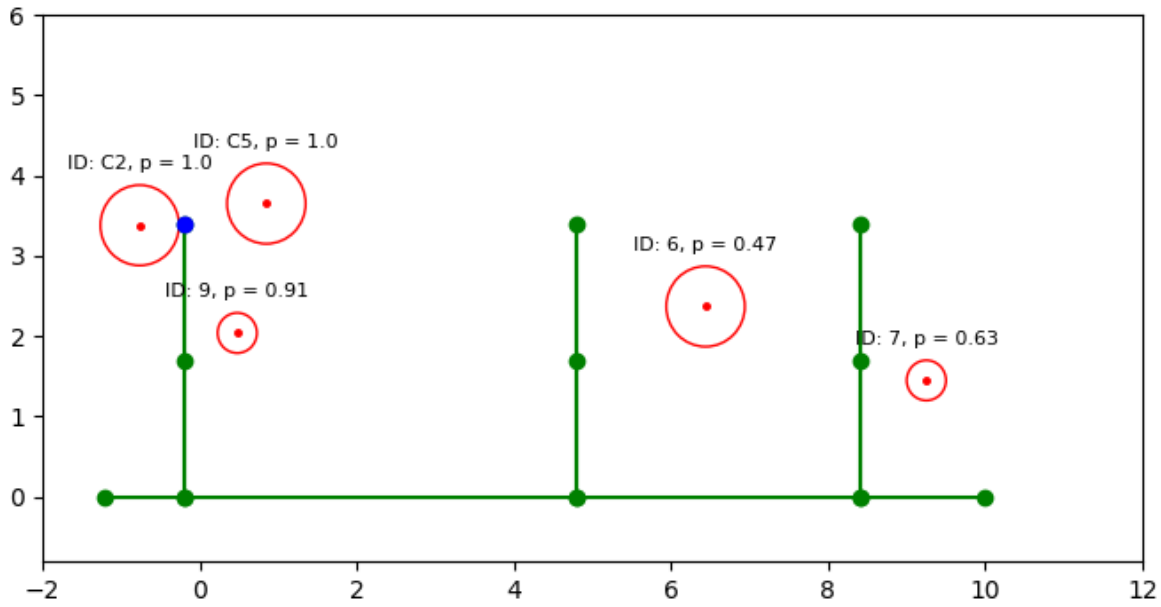


Figura 48: Mapa simplificado de las oficinas del GII durante un control de presencia

La Figura 48 muestra una equivalencia simplificada del boceto de la Figura 47. En ella se representa todas las personas que han sido identificadas, con sus respectivos radios (según las veces que haya sido detectado) y probabilidades estimadas de que permanezcan en el punto marcado en la gráfica. Por último, se muestra en azul la posición en la que estaba ubicado el Robobo la última vez que lo actualizó.

En la siguiente Figura, se puede observar los datos recogidos sobre las 5 personas representadas anteriormente.

ID Persona	Posicion	Probabilidad	Radio	Tiempo Pasado	Sala Avistado	Hora Avistado
C2	[0.85, 3.65]	1.0	0.5	0.01	GII	25/06/19 18:04:52
C5	[-0.76, 3.38]	1.0	0.5	0.01	GII	25/06/19 18:04:52
9	[0.48, 2.04]	0.91	0.25	0.81	GII	25/06/19 18:04:03
7	[9.25, 1.45]	0.63	0.25	3.85	GII	25/06/19 18:01:01
6	[6.44, 2.37]	0.47	0.5	6.31	GII	25/06/19 17:58:33

Figura 49: Panel de estado en el ensayo realizado

7.2.2 Control de acceso

Por otro lado, se encuentra el control de acceso que se lleva a cabo en paralelo. El Robobo se ha colocado en una posición elevada y cercana, como se expone en la

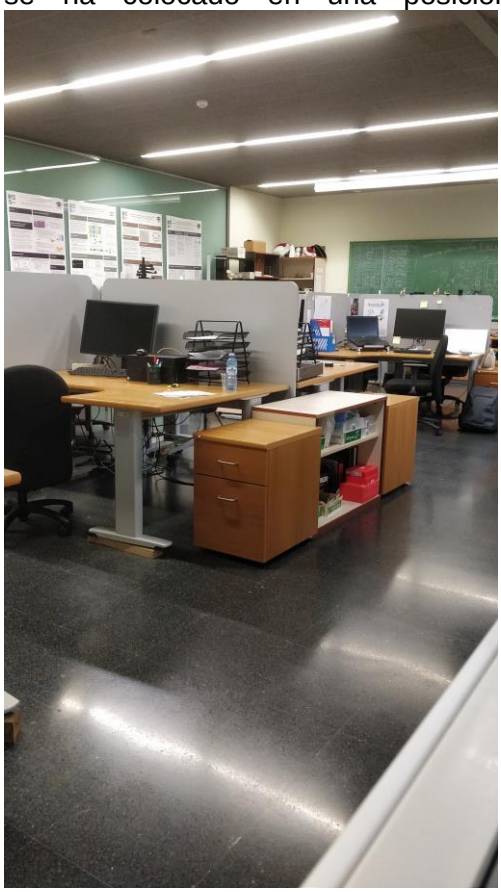


Figura 50. Desde esta posición, el Robobo dispone de una perspectiva desde la cual tiene visión del pasillo principal de las oficinas y poder realizar un seguimiento efectivo de las personas que circulan por él.

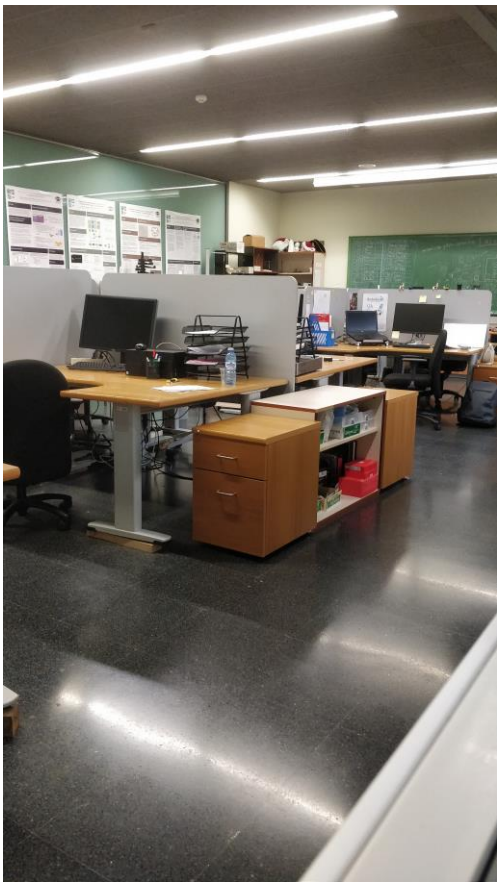


Figura 50: Punto de vista del Robobo encargado que de realizar el control de acceso

Finalmente, se muestra en panel de acceso resultado del ensayo realizado. Por cada una de las personas presentes en el panel el Robobo destinado al control de acceso ha ordenado al otro que compruebe y actualice en función del pasillo donde fue detectado

Acceso Sala	Contador	ID Persona	Sentido	Pasillo	Fecha y Hora
GII	5	2	Sale	2	25/06/19 17:36:06
GII	6	3	Entra	1	25/06/19 17:58:37
GII	5	5	Sale	2	25/06/19 17:59:03
GII	4	8	Sale	3	25/06/19 18:01:01
GII	5	7	Entra	2	25/06/19 18:02:33

Figura 51: Panel de acceso del ensayo realizado

8 CONCLUSIONES

EL objetivo principal de este Trabajo Fin de Máster es la realización de un control de acceso y presencia simultáneos en las oficinas del GII con Robobo y la implementación de una estrategia colaborativa. Para el cumplimiento de este objetivo, ha sido necesario la realización de unos pasos previos.

En primer lugar, se ha desarrollado una navegación que combina una localización mediante *AprilTags* y la estimación de la posición mediante una odometría basada en los encoders que dispone el Robobo en las ruedas.

Por un lado, se ha implementado un sistema capaz de llevar a cabo un control del tráfico de gente en las oficinas GII. En un principio, este sistema únicamente contabilizaba las personas presentes en el interior en función de las que entraba y salían. Sin embargo, posteriormente se ha modificado para que también fuese capaz de detectar la trayectoria que una persona realiza cuando accede o abandona a las oficinas.

Por otro lado, combinando la navegación y la librería de detección de objetos de OpenCV, se ha desarrollado un control de presencia. Este sistema puede tener localizada la gente que está presente en las oficinas.

A continuación, se ha implementado una estrategia de vigilancia coordinada. Con esta estrategia se ha elabora una comunicación entre los Robobos, a través de una arquitectura en ROS, con la que poder desarrollar un comportamiento como grupo más complejo, y obtener mejores resultados en la resolución de la tarea. Además, también se ha desarrollado una centralita desde la que se puede acceder a diferentes paneles que actualizan en tiempo real la información recopilada por los Robobos.

Tras haber desarrollado todas las aplicaciones propuestas en los objetivos, se han validado y se han calculado los errores que se comenten en cada una de ellas. Por último, se ha realizado un ensayo en un entorno real y se han comentado los resultados obtenidos.

9 REFERENCIAS

[1] F. Bellas *et al.*, “The Robobo Project: Bringing Educational Robotics Closer to Real-World Applications,” Springer, Cham, 2018, pp. 226–237.

[2] “International Federation of Robotics.” [Online]. Available: https://ifr.org/. [Accessed: 19-May-2019].

[3] M. Weiser, “The computer for the 21st Century,” *IEEE Pervasive Comput.*, vol. 1, no. 1, pp. 19–25, Jan. 2002.

[4] N. Shahid and S. Aneja, “Internet of Things: Vision, application areas and research challenges,” in *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, 2017.

[5] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[6] D. López Enseñat, “Herramientas de evolución colectiva para el tratamiento de problemas distribuidos,” 2017.

[7] T. Bailey, “Mobile robot localisation and mapping in extensive outdoor environments,” 2002.

[8] C. Fernández Caramés, “Técnicas de navegación para un robot móvil utilizando sistemas de razonamiento espacial,” 2012.

[9] J. Borenstein, H. Everett, L. F.-U. of Michigan, and undefined 1996, “Where am I? Sensors and methods for mobile robot positioning,” *Citeseer*.

[10] M. Agrawal and K. Konolige, “Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS,” in *18th International Conference on Pattern Recognition (ICPR’06)*, 2006, pp. 1063–1068.

[11] M. Parada Pombo, “Desarrollo de librerías de control para aplicaciones de coordinación de robótica colectiva,” 2018.

[12] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[13] B. Fernandez-Saavedra, R. Alonso-Moreno, A. Mendaza-Ormaza, and R. Sanchez-Reillo, “Usability Evaluation of Fingerprint Based Access Control Systems,” in *2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2010, pp. 333–336.

[14] L. Snidaro, … C. M.-I. T. on, and U. 2004, “Video security for ambient intelligence,” *ieeexplore.ieee.org*.

[15] J. N. K. Liu, M. Wang, and B. Feng, “iBotGuard: An internet-based intelligent robot security system using invariant face recognition against intruder,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 35, no. 1, pp. 97–105, Feb. 2005.

[16] B. Benyo, B. Sodor, T. Doktor, and G. Fordos, “Student attendance monitoring at the university using NFC,” in *Wireless Telecommunications Symposium 2012*, 2012, pp. 1–5.

[17] C. Schlegel, J. Illmann, H. Jaberg, M. S.- BMVC, and undefined 1998, “Vision based person tracking with a mobile robot.,” *pdfs.semanticscholar.org*.

[18] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “Pfinder: real-time tracking of the human body,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, Jul. 1997.

10 ANEXO I

10.1 Tablas de errores del t*ag* de 15.1 cm de lado

50 cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	-0,79	-2,79	-0,82	-2,73	0,07	-0,86	-0,89	0,03
0	0,00	0	-1,00	-2,75	-0,82	-2,73	0,19	-1,09	-0,89	0,20
0	0,00	0	-0,65	-2,63	-0,82	-2,73	0,19	-0,71	-0,89	0,18
0	0,00	-10	1,40	-3,72	-0,82	-2,73	2,43	1,50	-0,89	0,61
0	0,00	10	-0,37	-0,64	-0,82	-2,73	2,13	-0,42	-0,89	0,46
-5	-5,00	0	-3,69	-4,07	-5,82	-2,65	2,56	-3,90	-6,30	2,40
-5	-5,00	0	-3,68	-4,07	-5,82	-2,65	2,57	-3,89	-6,30	2,41
-5	-5,00	0	-3,70	-4,07	-5,82	-2,65	2,54	-3,92	-6,30	2,38
-5	-5,00	-10	-0,61	-2,21	-5,82	-2,65	5,22	-0,67	-6,30	5,63
-5	-5,00	10	-5,27	-2,87	-5,82	-2,65	0,59	-5,69	-6,30	0,61
5	5,00	0	6,74	-0,30	4,18	-2,80	3,57	7,63	4,53	3,10
5	5,00	0	6,73	-0,32	4,18	-2,80	3,56	7,62	4,53	3,09
5	5,00	0	6,77	-0,30	4,18	-2,80	3,60	7,67	4,53	3,14
5	5,00	-10	5,44	-0,12	4,18	-2,80	2,96	6,19	4,53	1,66
5	5,00	10	6,99	-1,59	4,18	-2,80	3,06	7,72	4,53	3,19
-35	-35,00	0	-36,34	-2,38	-35,82	-2,19	0,56	-34,76	-34,46	0,30
-35	-35,00	0	-36,32	-2,41	-35,82	-2,19	0,55	-34,72	-34,46	0,26
-35	-35,00	0	-36,32	-2,42	-35,82	-2,19	0,55	-34,72	-34,46	0,25
-35	-35,00	-10	-36,31	-4,29	-35,82	-2,19	2,16	-33,78	-34,46	0,69
-35	-35,00	10	-33,20	-2,93	-35,82	-2,19	2,71	-32,10	-34,46	2,36
35	35,00	0	35,34	-4,47	34,18	-3,27	1,66	32,97	32,69	0,28
35	35,00	0	35,33	-4,47	34,18	-3,27	1,66	32,97	32,69	0,28
35	35,00	0	35,33	-4,47	34,18	-3,27	1,66	32,97	32,69	0,28
35	35,00	-10	36,38	-4,14	34,18	-3,27	2,36	33,90	32,69	1,21
35	35,00	10	33,70	-2,97	34,18	-3,27	0,57	32,46	32,69	0,23

100 cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	-0,97	-4,58	-0,78	-4,66	0,20	-0,53	-0,43	0,10
0	0,00	0	-0,24	-4,75	-0,78	-4,66	0,55	-0,13	-0,43	0,30
0	0,00	0	-1,14	-4,64	-0,78	-4,66	0,36	-0,62	-0,43	0,20
0	0,00	-15	-0,86	0,82	-0,78	-4,66	5,48	-0,50	-0,43	0,07

0	0,00	15	1,38	-4,49	-0,78	-4,66	2,17	0,76	-0,43	0,33
-5	-5,00	0	-1,81	-3,83	-5,78	-4,62	4,05	-1,00	-3,16	2,17
-5	-5,00	0	-1,94	-3,48	-5,78	-4,62	4,01	-1,07	-3,16	2,09
-5	-5,00	0	-2,03	-3,85	-5,78	-4,62	3,83	-1,12	-3,16	2,04
-5	-5,00	-15	-1,15	-2,81	-5,78	-4,62	4,97	-0,64	-3,16	2,52
-5	-5,00	15	-1,89	-2,22	-5,78	-4,62	4,57	-1,06	-3,16	2,10
5	5,00	0	6,77	-5,58	4,22	-4,70	2,70	3,67	2,31	1,36
5	5,00	0	6,41	-5,47	4,22	-4,70	2,33	3,48	2,31	1,17
5	5,00	0	6,51	-5,46	4,22	-4,70	2,42	3,53	2,31	1,23
5	5,00	-15	1,89	-2,43	4,22	-4,70	3,25	1,06	2,31	1,25
5	5,00	15	6,39	-1,40	4,22	-4,70	3,94	3,60	2,31	1,30
-35	-35,00	0	-33,63	-6,19	-35,78	-4,40	2,80	-17,57	-18,92	1,35
-35	-35,00	0	-33,84	-6,15	-35,78	-4,40	2,62	-17,68	-18,92	1,24
-35	-35,00	0	-33,81	-6,20	-35,78	-4,40	2,67	-17,66	-18,92	1,26
-35	-35,00	-15	-29,83	-6,91	-35,78	-4,40	6,46	-15,59	-18,92	3,33
-35	-35,00	15	-33,15	-6,57	-35,78	-4,40	3,41	-17,28	-18,92	1,64
35	35,00	0	36,70	-8,91	34,22	-4,92	4,70	18,62	18,06	0,56
35	35,00	0	36,57	-8,92	34,22	-4,92	4,64	18,56	18,06	0,50
35	35,00	0	36,56	-8,93	34,22	-4,92	4,64	18,55	18,06	0,49
35	35,00	-15	30,31	-8,22	34,22	-4,92	5,11	15,65	18,06	2,42
35	35,00	15	33,43	-6,42	34,22	-4,92	1,69	17,44	18,06	0,62

150 cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	-0,83	-9,62	-3,55	-9,34	2,73	-0,30	-1,28	0,98
0	0,00	0	-2,59	-9,65	-3,55	-9,34	1,01	-0,93	-1,28	0,35
0	0,00	0	-7,22	-8,74	-3,55	-9,34	3,73	-2,61	-1,28	1,33
0	0,00	-15	-3,91	-0,54	-3,55	-9,34	8,80	-1,49	-1,28	0,21
0	0,00	15	1,79	-6,79	-3,55	-9,34	5,92	0,66	-1,28	0,62
-10	-10,00	0	-8,49	-10,21	-13,55	-9,11	5,17	-3,04	-4,87	1,83
-10	-10,00	0	-10,75	-9,49	-13,55	-9,11	2,82	-3,86	-4,87	1,01
-10	-10,00	0	-9,85	-9,08	-13,55	-9,11	3,70	-3,54	-4,87	1,32
-10	-10,00	-15	-13,32	-3,35	-13,55	-9,11	5,77	-4,96	-4,87	0,10
-10	-10,00	15	-3,47	-4,86	-13,55	-9,11	10,94	-1,28	-4,87	3,58
10	10,00	0	6,41	-9,03	6,45	-9,56	0,53	2,31	2,32	0,01
10	10,00	0	7,95	-8,72	6,45	-9,56	1,72	2,87	2,32	0,55
10	10,00	0	9,38	-8,44	6,45	-9,56	3,13	3,39	2,32	1,07
10	10,00	-15	-0,79	-2,92	6,45	-9,56	9,83	-0,30	2,32	2,02
10	10,00	15	8,26	-2,59	6,45	-9,56	7,20	3,10	2,32	0,78

-35	-35,01	0	-35,39	-13,00	-38,55	-8,56	5,45	-12,25	-13,66	1,41
-35	-35,01	0	-35,61	-12,90	-38,55	-8,56	5,24	-12,33	-13,66	1,33
-35	-35,01	0	-35,55	-12,93	-38,55	-8,56	5,30	-12,31	-13,66	1,36
-35	-35,01	-15	-30,13	-8,69	-38,55	-8,56	8,42	-10,75	-13,66	2,91
-35	-35,01	15	-28,26	-11,23	-38,55	-8,56	10,63	-9,94	-13,66	3,72
35	35,01	0	33,27	-6,85	31,45	-10,11	3,74	11,98	11,11	0,86
35	35,01	0	33,08	-7,01	31,45	-10,11	3,50	11,90	11,11	0,78
35	35,01	0	33,30	-6,90	31,45	-10,11	3,71	11,98	11,11	0,87
35	35,01	-15	22,66	-8,50	31,45	-10,11	8,94	8,14	11,11	2,98
35	35,01	15	23,42	-5,98	31,45	-10,11	9,03	8,54	11,11	2,57

200cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	-12,31	-14,81	-11,61	-14,47	0,78	-3,28	-3,10	0,18
0	0,00	0	-12,74	-14,96	-11,61	-14,47	1,23	-3,39	-3,10	0,29
0	0,00	0	-9,79	-13,65	-11,61	-14,47	2,01	-2,62	-3,10	0,48
0	0,00	-15	13,38	-5,42	-11,61	-14,47	26,58	3,73	-3,10	0,63
0	0,00	15	-4,64	-11,10	-11,61	-14,47	7,75	-1,26	-3,10	1,84
-10	-10,01	0	11,42	-6,74	-21,61	-13,93	33,80	3,16	-5,77	2,61
-10	-10,01	0	10,89	-6,49	-21,61	-13,93	33,34	3,02	-5,77	2,75
-10	-10,01	0	10,80	-6,91	-21,61	-13,93	33,16	2,99	-5,77	2,78
-10	-10,01	-15	-10,32	-1,70	-21,61	-13,93	16,65	-2,93	-5,77	2,84
-10	-10,01	15	-10,46	-9,87	-21,61	-13,93	11,87	-2,85	-5,77	2,91
10	10,01	0	17,62	-14,83	-1,61	-15,02	19,23	4,69	-0,43	4,26
10	10,01	0	14,39	-14,58	-1,61	-15,02	16,01	3,84	-0,43	3,41
10	10,01	0	14,45	-15,03	-1,61	-15,02	16,06	3,85	-0,43	3,42
10	10,01	-15	13,79	-10,06	-1,61	-15,02	16,18	3,76	-0,43	3,33
10	10,01	15	13,54	-3,06	-1,61	-15,02	19,30	3,81	-0,43	3,38
-45	-45,07	0	-33,61	-16,32	-56,61	-12,04	23,40	-8,83	-14,95	6,12
-45	-45,07	0	-33,07	-16,41	-56,61	-12,04	23,94	-8,69	-14,95	6,26
-45	-45,07	0	-32,85	-16,43	-56,61	-12,04	24,16	-8,63	-14,95	6,32
-45	-45,07	-15	-29,50	-10,10	-56,61	-12,04	27,18	-7,99	-14,95	6,96
-45	-45,07	15	-24,31	-14,83	-56,61	-12,04	32,42	-6,46	-14,95	8,49
45	45,07	0	55,56	-9,85	33,39	-16,91	23,27	14,83	8,75	6,08
45	45,07	0	54,77	-9,67	33,39	-16,91	22,57	14,64	8,75	5,89
45	45,07	0	60,43	-6,92	33,39	-16,91	28,82	16,28	8,75	7,53
45	45,07	-15	39,74	-14,15	33,39	-16,91	6,93	10,51	8,75	1,76
45	45,07	15	44,59	-8,29	33,39	-16,91	14,14	12,08	8,75	3,33

10.1 Tablas de errores del *tag* de 20.2 cm de lado

50 cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	0,27	0,31	0,27	0,31	0,00	0,31	0,31	0,00
0	0,00	0	0,27	0,31	0,27	0,31	0,00	0,32	0,31	0,01
0	0,00	0	0,26	0,31	0,27	0,31	0,01	0,30	0,31	0,01
0	0,00	-10	2,65	-0,86	0,27	0,31	2,65	2,98	0,31	2,67
0	0,00	10	1,02	1,46	0,27	0,31	1,37	1,20	0,31	0,89
-5	-5,00	0	-2,16	-2,16	-4,73	0,28	3,55	-2,37	-5,43	3,07
-5	-5,00	0	-2,16	-2,16	-4,73	0,28	3,54	-2,37	-5,43	3,06
-5	-5,00	0	-2,19	-2,15	-4,73	0,28	3,52	-2,40	-5,43	3,03
-5	-5,00	-10	0,60	-0,33	-4,73	0,28	5,37	0,68	-5,43	4,75
-5	-5,00	10	-4,20	-1,85	-4,73	0,28	2,20	-4,63	-5,43	0,80
5	5,00	0	7,51	2,03	5,27	0,34	2,81	8,90	6,06	2,85
5	5,00	0	7,51	2,03	5,27	0,34	2,81	8,90	6,06	2,84
5	5,00	0	7,52	2,04	5,27	0,34	2,82	8,92	6,06	2,86
5	5,00	-10	6,29	1,95	5,27	0,34	1,91	7,46	6,06	1,40
5	5,00	10	8,19	0,02	5,27	0,34	2,94	9,31	6,06	3,25
-35	-35,00	0	-34,30	-1,83	-34,73	0,12	2,00	-33,49	-34,85	1,36
-35	-35,00	0	-34,28	-1,83	-34,73	0,12	2,00	-33,48	-34,85	1,37
-35	-35,00	0	-34,27	-1,82	-34,73	0,12	1,99	-33,48	-34,85	1,37
-35	-35,00	-10	-34,51	-3,51	-34,73	0,12	3,63	-32,82	-34,85	2,03
-35	-35,00	10	-31,55	-2,41	-34,73	0,12	4,06	-31,05	-34,85	3,80
35	35,00	0	37,11	-1,38	35,27	0,50	2,63	35,84	35,47	0,37
35	35,00	0	37,01	-1,22	35,27	0,50	2,45	35,85	35,47	0,38
35	35,00	0	37,01	-1,22	35,27	0,50	2,44	35,85	35,47	0,38
35	35,00	-10	38,04	-1,09	35,27	0,50	3,19	36,67	35,47	1,20
35	35,00	10	35,87	0,07	35,27	0,50	0,73	35,69	35,47	0,22

100 cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	1,18	-1,07	1,21	-1,09	0,04	0,67	0,68	0,02
0	0,00	0	1,21	-1,10	1,21	-1,09	0,01	0,69	0,68	0,00
0	0,00	0	1,24	-1,10	1,21	-1,09	0,03	0,70	0,68	0,02
0	0,00	-15	2,60	1,38	1,21	-1,09	2,83	1,51	0,68	0,83
0	0,00	15	2,27	-3,47	1,21	-1,09	2,60	1,26	0,68	0,57
-5	-5,00	0	0,19	-0,02	-3,79	-1,15	4,14	0,11	-2,15	2,04

-5	-5,00	0	0,22	0,04	-3,79	-1,15	4,19	0,13	-2,15	2,02
-5	-5,00	0	0,25	0,00	-3,79	-1,15	4,20	0,14	-2,15	2,01
-5	-5,00	-15	0,57	-0,55	-3,79	-1,15	4,40	0,32	-2,15	1,82
-5	-5,00	15	-4,58	-2,11	-3,79	-1,15	1,24	-2,57	-2,15	0,42
5	5,00	0	8,21	-3,29	6,21	-1,03	3,02	4,54	3,52	1,03
5	5,00	0	8,25	-3,27	6,21	-1,03	3,04	4,57	3,52	1,05
5	5,00	0	8,16	-3,30	6,21	-1,03	2,99	4,51	3,52	1,00
5	5,00	-15	7,12	-3,71	6,21	-1,03	2,83	3,93	3,52	0,41
5	5,00	15	7,36	-1,63	6,21	-1,03	1,29	4,14	3,52	0,62
-35	-35,00	0	-29,61	-5,63	-33,79	-1,51	5,87	-15,66	-18,41	2,75
-35	-35,00	0	-29,60	-5,64	-33,79	-1,51	5,89	-15,65	-18,41	2,76
-35	-35,00	0	-29,62	-5,62	-33,79	-1,51	5,86	-15,67	-18,41	2,75
-35	-35,00	-15	-24,93	-6,94	-33,79	-1,51	10,40	-13,12	-18,41	5,29
-35	-35,00	15	-28,19	-6,42	-33,79	-1,51	7,45	-14,84	-18,41	3,58
35	35,00	0	40,58	-4,05	36,21	-0,67	5,53	21,31	19,78	1,52
35	35,00	0	40,58	-4,06	36,21	-0,67	5,53	21,30	19,78	1,52
35	35,00	0	40,53	-4,06	36,21	-0,67	5,50	21,28	19,78	1,50
35	35,00	-15	34,28	-4,57	36,21	-0,67	4,34	18,15	19,78	1,63
35	35,00	15	37,52	-2,40	36,21	-0,67	2,17	20,12	19,78	0,34

150 cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	2,73	-2,76	2,92	-2,92	0,25	1,02	1,09	0,07
0	0,00	0	3,32	-2,96	2,92	-2,92	0,40	1,24	1,09	0,15
0	0,00	0	2,71	-3,04	2,92	-2,92	0,24	1,01	1,09	0,08
0	0,00	-15	11,20	-2,56	2,92	-2,92	8,28	4,20	1,09	3,10
0	0,00	15	-8,28	-7,93	2,92	-2,92	12,27	-3,00	1,09	1,91
-10	-10,00	0	-0,53	-3,29	-7,08	-3,11	6,56	-0,20	-2,65	2,45
-10	-10,00	0	-1,54	-2,84	-7,08	-3,11	5,54	-0,58	-2,65	2,07
-10	-10,00	0	2,49	-3,22	-7,08	-3,11	9,57	0,93	-2,65	1,72
-10	-10,00	-15	-7,56	1,51	-7,08	-3,11	4,64	-2,92	-2,65	0,27
-10	-10,00	15	-7,65	-4,82	-7,08	-3,11	1,81	-2,83	-2,65	0,18
10	10,00	0	13,14	-4,33	12,92	-2,73	1,61	4,87	4,84	0,03
10	10,00	0	14,90	-4,25	12,92	-2,73	2,50	5,52	4,84	0,68
10	10,00	0	13,86	-4,42	12,92	-2,73	1,93	5,13	4,84	0,29
10	10,00	-15	12,76	-5,86	12,92	-2,73	3,14	4,68	4,84	0,15
10	10,00	15	10,12	1,16	12,92	-2,73	4,79	3,89	4,84	0,95
-35	-35,01	0	-26,99	-9,70	-32,08	-3,59	7,96	-9,59	-11,80	2,21

-35	-35,01	0	-26,99	-9,57	-32,08	-3,59	7,86	-9,60	-11,80	2,20
-35	-35,01	0	-27,17	-9,57	-32,08	-3,59	7,74	-9,66	-11,80	2,13
-35	-35,01	-15	-21,99	-5,16	-32,08	-3,59	10,21	-8,07	-11,80	3,73
-35	-35,01	15	-18,86	-8,52	-32,08	-3,59	14,11	-6,78	-11,80	5,01
35	35,01	0	41,66	-3,27	37,92	-2,25	3,88	15,20	13,99	1,22
35	35,01	0	40,90	-3,52	37,92	-2,25	3,24	14,92	13,99	0,93
35	35,01	0	41,01	-3,49	37,92	-2,25	3,33	14,96	13,99	0,97
35	35,01	-15	35,04	-6,47	37,92	-2,25	5,11	12,62	13,99	1,36
35	35,01	15	32,01	-3,50	37,92	-2,25	6,04	11,78	13,99	2,21

200cm de distancia:

Posición	Proyectado	Angulo Pan	Leída		Teórica		Error	Ob Leida	Ob Teórica	Ob Error
0	0,00	0	9,37	-3,35	9,85	-3,24	0,50	2,64	2,77	0,14
0	0,00	0	10,15	-3,35	9,85	-3,24	0,32	2,86	2,77	0,08
0	0,00	0	10,03	-3,03	9,85	-3,24	0,28	2,83	2,77	0,05
0	0,00	-15	15,28	-8,48	9,85	-3,24	7,54	4,19	2,77	1,42
0	0,00	15	-12,25	-8,93	9,85	-3,24	22,81	-3,35	2,77	0,58
-10	-10,01	0	2,41	-4,12	-0,15	-3,73	2,59	0,68	-0,04	0,63
-10	-10,01	0	3,82	-3,50	-0,15	-3,73	3,98	1,08	-0,04	1,03
-10	-10,01	0	3,07	-3,99	-0,15	-3,73	3,23	0,86	-0,04	0,82
-10	-10,01	-15	16,30	-7,07	-0,15	-3,73	16,79	4,50	-0,04	4,46
-10	-10,01	15	-9,60	-8,01	-0,15	-3,73	10,37	-2,64	-0,04	2,60
10	10,01	0	16,94	-8,36	19,85	-2,76	6,31	4,65	5,59	0,94
10	10,01	0	12,55	-8,19	19,85	-2,76	9,10	3,45	5,59	2,14
10	10,01	0	12,77	-7,54	19,85	-2,76	8,55	3,52	5,59	2,07
10	10,01	-15	13,52	-7,58	19,85	-2,76	7,96	3,73	5,59	1,86
10	10,01	15	13,14	1,71	19,85	-2,76	8,06	3,79	5,59	1,80
-45	-45,05	0	-30,42	-13,09	-35,15	-5,42	9,01	-8,12	-9,71	1,59
-45	-45,05	0	-31,57	-11,72	-35,15	-5,42	7,24	-8,48	-9,71	1,23
-45	-45,05	0	-32,78	-12,47	-35,15	-5,42	7,43	-8,77	-9,71	0,94
-45	-45,05	-15	-23,56	-8,50	-35,15	-5,42	11,99	-6,45	-9,71	3,26
-45	-45,05	15	-18,47	-14,18	-35,15	-5,42	18,84	-4,93	-9,71	4,78
45	45,05	0	49,61	-8,26	54,85	-1,06	8,90	13,40	15,26	1,86
45	45,05	0	50,84	-7,52	54,85	-1,06	7,60	13,77	15,26	1,49
45	45,05	0	50,70	-7,73	54,85	-1,06	7,85	13,72	15,26	1,54
45	45,05	-15	35,76	-10,78	54,85	-1,06	21,42	9,63	15,26	5,63
45	45,05	15	35,37	-4,23	54,85	-1,06	19,73	9,83	15,26	5,43