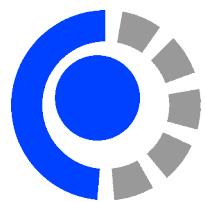




UNIVERSIDAD NACIONAL DEL COMAHUE  
FACULTAD DE INFORMÁTICA



TESIS DE LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

**Diseño e implementación de un sistema embebido de  
localización para robots móviles de bajo costo en  
ambientes interiores**

Alejandro Jersson Mora Vasquez

Director: Rafael Ignacio Zurita

NEUQUÉN

ARGENTINA

2022

## PREFACIO

Esta tesis es presentada como parte de los requisitos finales para optar al grado académico de *Licenciado en Ciencias de la Computación*, otorgado por la Universidad Nacional del Comahue, y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otras. La misma es el resultado de la investigación llevada a cabo en el Departamento Ingeniería de Computadoras, de la Facultad de Informática, en el período comprendido entre octubre de 2019 y septiembre de 2022, bajo la dirección de Rafael Ignacio Zurita.

Alejandro Jersson Mora Vasquez  
FACULTAD DE INFORMÁTICA  
UNIVERSIDAD NACIONAL DEL COMAHUE  
*Neuquén, 22 de Septiembre de 2022.*



UNIVERSIDAD NACIONAL DEL COMAHUE

Facultad de Informática

La presente tesis ha sido aprobada el día ..... , mereciendo la  
calificación de .....

## DEDICATORIAS

A mis padres, Ruth Marianela Vasquez y Moises Eduardo Mora.

## AGRADECIMIENTOS

A mi familia, por siempre alentarme y apoyarme con mis estudios.

A Rafael, por la constante ayuda y paciencia.

A compañeros y amigos, por acompañarme y darme ese empujón anímico que siempre hace falta para continuar.

## RESUMEN

La localización es un requisito de la navegación con mapas de robots móviles. Usualmente se implementa con múltiples sensores y gran poder de cómputo, por lo que no suele ser una capacidad de los robot móviles de bajo costo. Se propone el diseño e implementación de un sistema prototípico que reporta su localización utilizando un hardware de mínimas prestaciones y marcas artificiales en el terreno. Se realizaron pruebas en un ambiente de interior, obteniendo desde el prototípico repetidas localizaciones, para evaluar la precisión y estimar la exactitud. En base a los resultados se concluye que el prototípico propuesto es capaz de determinar ubicaciones con una precisión menor a 10cm, y a una frecuencia de 2Hz; lo que permite plantear la posibilidad, en trabajos futuros, de incluir navegación con mapas a una clase de robots que hasta hoy no suele utilizar esta característica.

## ABSTRACT

Localization is a prior needed condition for land navigation of mobile robots. It is usually implemented with multiple sensors and high performance computers, so it is not usually a low-cost mobile robot capability. In this paper we present the architecture and design of a prototype embedded system which reports its location using a minimum-performance hardware and artificial landmarks. In order to evaluate accuracy, experimental tests were performed in an indoor environment, obtaining repeated locations from the prototype system. Based on the results, it is concluded that the proposed system is able to localize itself, with +/-10cm of accuracy, at a frequency of 2Hz. Conclusions raise the possibility, in future works, for the inclusion of land navigation using maps into low-cost robots, that until today do not usually present this characteristic.

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Localización . . . . .	1
1.2. Robótica Educativa . . . . .	2
1.3. Robótica Educativa en la Facultad de Informática . . . . .	3
1.4. Objetivos . . . . .	4
1.5. Organización de la Tesis . . . . .	4
<b>2. Técnicas para localización de robots móviles en interiores</b>	<b>7</b>
2.1. Técnicas que requieren modificación del ambiente . . . . .	8
2.1.1. Requiere infraestructura dedicada . . . . .	8
2.1.2. Utiliza infraestructura del ambiente . . . . .	10
2.1.3. Marcas artificiales en el terreno . . . . .	11
2.2. Técnicas que no alteran el ambiente . . . . .	12
2.2.1. Navegación por estima . . . . .	12
2.2.2. Marcas naturales en el terreno . . . . .	13
2.3. Comparativa entre las distintas técnicas de localización . . . . .	14
2.4. Conclusiones . . . . .	15
<b>3. Localización utilizando marcas de referencias visuales</b>	<b>17</b>
3.1. Sistemas de marcas fiduciales . . . . .	17
3.2. ARToolkit . . . . .	18
3.3. ARTag . . . . .	19
3.4. AprilTag . . . . .	22
3.5. Conclusiones . . . . .	24
<b>4. Arquitectura del prototipo propuesto</b>	<b>27</b>
4.1. Componentes de Hardware . . . . .	27
4.2. Arquitectura de Hardware . . . . .	27
4.2.1. Hardware de procesamiento embebido . . . . .	28
4.2.2. Dispositivo de cámara rotativa . . . . .	28
4.2.3. Conexión entre componentes . . . . .	30
4.2.4. Diseño del gabinete . . . . .	31
4.3. Arquitectura de Software . . . . .	32
4.3.1. Firmware Embebido . . . . .	32
4.3.2. Sistema Linux . . . . .	34
4.4. Conclusiones . . . . .	36
<b>5. Experimentos y Resultados</b>	<b>39</b>
5.1. Tiempo de ejecución . . . . .	39
5.2. Precisión . . . . .	39
5.2.1. Primer Experimento . . . . .	39
5.2.2. Segundo Experimento . . . . .	43

5.3. Conclusiones . . . . .	44
<b>6. Conclusiones y trabajos futuros</b>	<b>47</b>
6.1. Conclusiones . . . . .	47
6.2. Trabajos futuros . . . . .	47
6.3. Publicaciones e Impacto . . . . .	48
<b>Bibliografía</b>	<b>49</b>

# Índice de figuras

1.1. Arquitectura de un sistema de navegación de un robot autónomo . . . . .	2
1.2. Frankestito (2012 - 2013) . . . . .	3
1.3. Frankestito (2018) . . . . .	4
2.1. Tecnologías para localización de robots móviles en interiores . . . . .	7
2.2. Ejemplo de localización utilizando ultrasonido . . . . .	9
2.3. Ejemplo de localización utilizando RFID . . . . .	10
2.4. Ejemplos de marcadores de ArtoolKit, ArTag y AprilTag . . . . .	12
2.5. Vehículo autónomo utilizando la tecnología LIDAR . . . . .	14
3.1. Proceso de detección de marcadores ARToolKit. (a) Imagen original capturada. (b) Conversión a imagen blanco y negro. . . . .	18
3.2. Ejemplo de marcador de referencia ARToolkit . . . . .	19
3.3. Ejemplo de marcador de referencia ARTag de 36 bits . . . . .	20
3.4. Proceso de detección de marcadores ARTag. (a) Imagen original. (b) Segmentos de línea encontrados en la imagen. (c) Segmentos de líneas agrupados en cuadriláte- ros, (d) Marcadores ARTag encontrados a partir de los cuadriláteros cuyo interior contienen un código ARTag válido. . . . .	21
3.5. Ejemplo de marcador de referencia AprilTag . . . . .	22
3.6. Proceso de detección de marcadores AprilTag. (a) Imagen de entrada. (b) Algo- ritmo de detección calcula la magnitud del gradiente de cada píxel. (c) Algoritmo de detección calcula la dirección del gradiente de cada píxel. (d) Píxeles con di- recciones de gradiente y magnitudes similares agrupados. (e) Segmentos de línea ajustados a los píxeles de cada componente. (f) Cuadrantes detectados. . . . .	24
4.1. Router TP-Link MR3020 . . . . .	28
4.2. Camara USB genérica . . . . .	28
4.3. Motor paso a paso con placa controladora . . . . .	29
4.4. Sensor óptico de fin de carrera . . . . .	29
4.5. Microcontrolador con chip ATMEGA328 . . . . .	29
4.6. Diseño del dispositivo de cámara rotativa. . . . .	30
4.7. Diagrama de conexiones físicas entre los distintos componentes de hardware . .	30
4.8. Software OpenScad para diseño de modelos 3D . . . . .	31
4.9. Diseño 3D de gabinete soporte cámara-motor . . . . .	31
4.10. Prototipo terminado y ensamblado . . . . .	32
4.11. Diagrama de la Arquitectura de Software . . . . .	33
4.12. Etapas del proceso de localización. . . . .	35
4.13. Transformación del sistema de coordenadas de la cámara al sistema de coordenadas del TAG. (a) posición de la marca de referencia en el sistema de coordenadas de la cámara. (b) posición de la cámara en el sistema de coordenadas de la marca de referencia. . . . .	36

5.1. Escenario real donde se realizaron las mediciones del primer experimento . . . . .	40
5.2. Gráfico de dispersión de las localizaciones reportadas por el dispositivo prototipo en cada intersección de la cuadrícula . . . . .	41
5.3. Gráfico de dispersión de localizaciones promediadas reportadas por el dispositivo prototipo en cada intersección de la cuadrícula . . . . .	41
5.4. Histograma en X de las distancias en centímetros de las localizaciones con respecto al valor real . . . . .	42
5.5. Histograma en Y de las distancias en centímetros de las localizaciones con respecto al valor real . . . . .	42
5.6. Diagrama del escenario real del segundo experimento . . . . .	43
5.7. Ubicación de las localizaciones reportadas por el dispositivo prototipo en el segundo experimento . . . . .	44

# Índice de tablas

2.1. Tabla comparativa entre las distintas técnicas de localización . . . . .	15
5.1. Tiempos promedios de E/S y CPU en el dispositivo prototipo en 415 localizaciones.	39
5.2. Estadística de las distancias de localizaciones con respecto a su valor estimativo real	43



# Capítulo 1

## Introducción

En esta tesis se detalla el diseño e implementación de un sistema embebido prototipo de bajo costo, que reporta la ubicación de un robot educativo en un ambiente real de interiores. El sistema de hardware y software es de bajo consumo, y puede ser incorporado a cualquier clase de robot educativo de manera sencilla. Su utilización amplía las funcionalidades de estos tipos de robots, posibilitando a futuro la navegación con mapas, característica no disponible actualmente en esta clase de robots.

### 1.1. Localización

La localización es uno de los componentes requeridos para la navegación con mapas de los robots móviles, y requiere que el robot sea capaz de determinar su ubicación y orientación exacta en el ambiente en el cual se encuentre. Con esta información y un marco de referencia virtual que represente el ambiente real, el robot puede planificar una ruta y desplazarse para cumplir con sus objetivos [34]. En la Figura 1.1 se puede observar un diagrama de bloques de alto nivel que representa la arquitectura de un sistema de navegación con mapas. Como se observa, tanto el componente de *Razonamiento* como el de *Planificación (planificador de ruta)* dependen de la localización.

En la actualidad, los robots móviles capaces de realizar navegación en interiores son diseñados y construidos para tareas complejas. Suelen trabajar en ambientes industriales, para desplazar grandes cargas dentro de galpones o edificaciones. También en ambientes públicos, por ejemplo en hoteles, donde los robots suelen navegar desde la recepción a diferentes habitaciones, realizando la entrega de suministros solicitados por huéspedes [39, 8]. En esta clase de robots el costo y/o consumo energético usualmente no es una restricción primordial, por lo que suelen contar con sensores láser (LIDAR), visuales (cámaras) y de orientación y velocidad (unidades de medición inercial) para recolectar datos del ambiente; y con computadoras capaces de procesar grandes cantidades de información en tiempo real, lo que en conjunto permite conocer la ubicación del robot y navegar en sus ambientes de trabajo [17].

Por el contrario, los robots móviles de bajo costo utilizados en interiores, están diseñados para realizar tareas sencillas, tales como aspirar el polvo en un living (domésticos), o realizar movimientos simples (adelante, izquierda, etc) al ejecutar programas desarrollados por estudiantes (robótica educativa). Para ello utilizan un microcontrolador o microprocesador de bajas prestaciones como sistema central de control, y sensores que permiten una interacción básica con el entorno, por ejemplo, para evitar el choque con obstáculos o moverse dentro de un cerco invisible. Los sensores en esta clase de robots pueden detectar el objeto a evitar cuando se está a una distancia muy corta, devolviendo información básica que puede ser discretizada para indicar si existe o no un objeto frente al sensor. Por tal motivo estos robots realizan únicamente navegación reactiva (sin utilización de mapas), ya que no son capaces de determinar su ubicación [9]. Siendo que no pueden localizarse, tampoco pueden planificar trayectorias, y llevarlas a

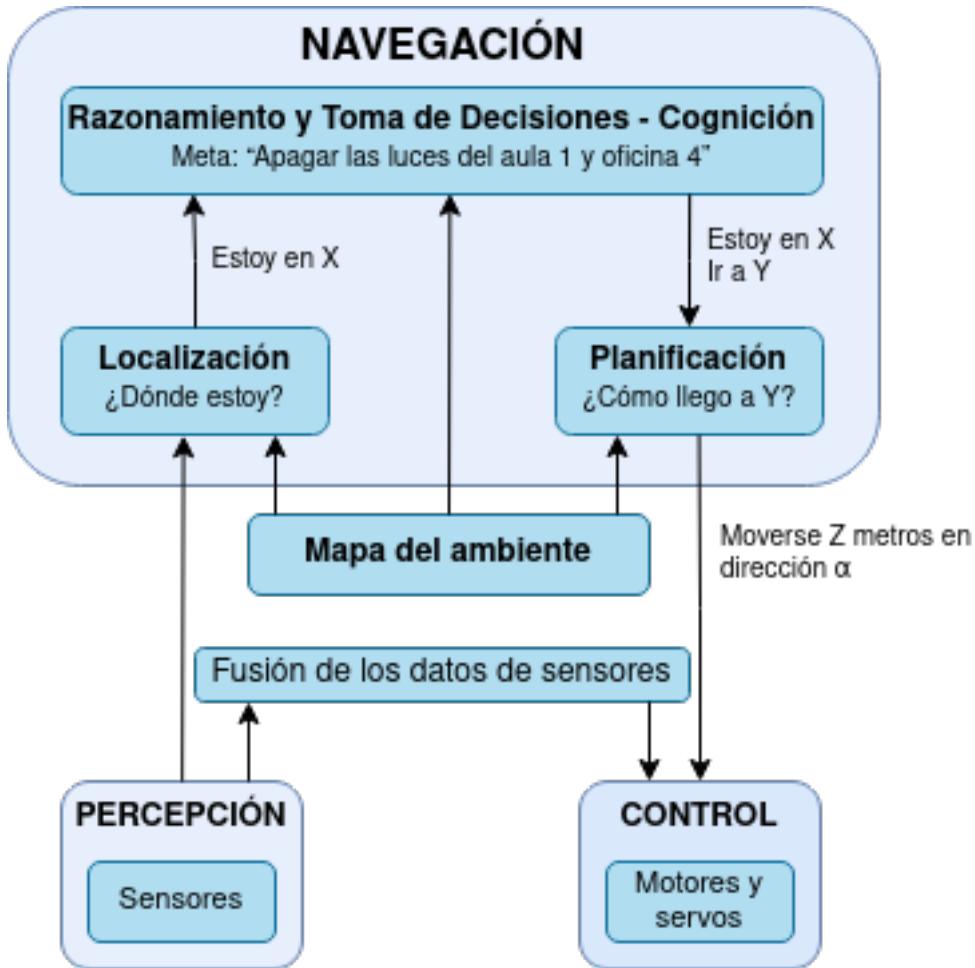


Figura 1.1: Arquitectura de un sistema de navegación de un robot autónomo

cabo. Ejemplos de esta clase de robots son las aspiradoras Roomba, de iRobot<sup>1</sup>; y los robots Frankestito-N6 desarrollados y utilizados por nuestra Facultad de Informática principalmente en robótica educativa [42].

## 1.2. Robótica Educativa

La robótica educativa es una subdisciplina de la robótica aplicada en el ámbito de la enseñanza, y se utiliza como herramienta de aprendizaje en los diferentes niveles del sistema educativo, fortaleciendo destrezas del pensamiento y permitiendo alcanzar un mayor grado de capacidades y competencias. Se pretende, mediante esta herramienta, impulsar el pensamiento crítico y creativo, y generar interés sobre diversas disciplinas de las Ciencias de la Computación [5, 7, 33].

Se considera entonces, a la Robótica Educativa, como un dispositivo pedagógico que estimula el desarrollo de habilidades en los estudiantes, a través de la concepción, creación, planificación, ensamblaje y utilización de robots. De esta manera la Robótica Educativa recurre a la robótica como un recurso didáctico-tecnológico para favorecer experiencias científicas específicas que posibilitan la integración de los diferentes campos del conocimiento. Un potencial importante de la robótica educativa es hacer visibles conceptos totalmente abstractos de la computación. De esta manera se propicia el desarrollo de técnicas directamente relacionadas con el pensamiento computacional, como la resolución de problemas y el trabajo colaborativo [21].

<sup>1</sup><https://www.irobot.lat/>

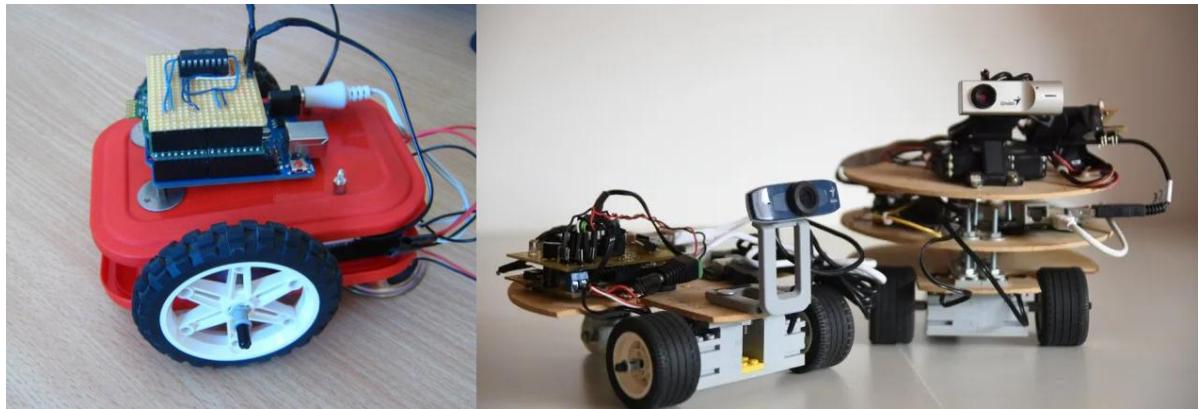


Figura 1.2: Frankestito (2012 - 2013)

En esta modalidad del aprendizaje los participantes diseñan diferentes soluciones lógicas (software) para manipular un robot (hardware) y concretar un objetivo determinado. Dependiendo de la plataforma educativa a veces se cuenta con un simulador que permite a los participantes ensayar a priori las soluciones lógicas para luego ejecutarlas directamente sobre el robot. De este modo, la utilización de lenguajes de programación de alto nivel textual o por bloques para controlar dispositivos físicos, se transforma en un recurso satisfactorio para alcanzar la comprensión de conceptos fundamentales de la computación, que en muchas ocasiones presentan una gran dificultad en su enseñanza [32].

### 1.3. Robótica Educativa en la Facultad de Informática

La Facultad de Informática, de la Universidad Nacional del Comahue, tiene una trayectoria en el área de la robótica que se expresa en actividades concretadas en el ámbito de la investigación, la extensión universitaria y el desarrollo académico. En 2003 se desarrollan las primeras actividades relacionadas a la robótica educativa en el ámbito de la Extensión Universitaria. En 2005 la línea fue de investigación y desarrollo, destinada al estudio y producción de Laboratorios Remotos de Robótica Educativa, y en 2008 se realizó la Competencia Argentina de Fútbol de Robots. Desde la perspectiva académica la temática de robótica inteligente, robótica educativa y robots de servicio es objeto de estudio de numerosas tesis de grado en la Facultad.

El 26 de Octubre de 2012, nuestro entrañable profesor Eduardo Grosclaude, realizaba la primera actividad académica de robótica educativa de la Facultad de Informática con el robot Frankestito. Fue en una clase de Introducción a la Programación, en la Sede Chos Malal de UNCOMA. El robot utilizado era un prototipo frágil, al que cada ciertos minutos había que apagar, ajustar sus cables (para ver si algún cable que perdió conexión se conectaba nuevamente) y volver a encender para corroborar si funcionaba. De hecho, este prototipo inicial construido en la Facultad, estaba empotrado en una caja plástica<sup>2</sup>, y se lo bautizó Frankestito (ver Figura 1.2), porque se reutilizaron motores y electrónica recuperada de aparatos considerados basura electrónica. Eduardo Grosclaude, en esos días en Chos Malal, implementó una sencilla conexión notebook-robot, lo que fue clave para la adopción del prototipo, y futuros prototipos de robots educativos, en actividades de la Facultad.

Si bien el robot prototipo inicial era frágil, tenía entre sus características, visión por computadora y conectividad WiFi. Además, se puede programar en varios lenguajes de programación de alto nivel, y seguir su ejecución remotamente. En nuestro entendimiento, estos prototipos fueron los primeros robots educativos construidos en Argentina con tales características. En la

<sup>2</sup>Según rumores, su nombre pudo haber sido Tuppertito, pero no contamos con referencias



Figura 1.3: Frankestito (2018)

Figura 1.3 se puede ver el estado evolutivo actual de Frankestito.

De cualquier manera, existe aún nuevas características que podrían potenciar aún más su uso como recurso pedagógico-tecnológico, como es la localización, introducida en la sección 1.1.

## 1.4. Objetivos

El objetivo general de este trabajo es diseñar e implementar un sistema embebido de localización para robots móviles de bajo costo, utilizando marcas de referencia artificiales en ambientes interiores. Dicho sistema, y trabajos futuros, permitirán la inclusión de navegación con mapas a una clase de robots que no suele utilizar esta característica, particularmente a los robots físicos construidos y utilizados por la facultad. Los objetivos específicos se enumeran a continuación.

- Definir y diseñar un módulo embebido de localización para robots móviles de bajo costo.
- Implementar un prototipo basado en el diseño referenciado anteriormente.
- Evaluar el sistema prototipo experimentando con el módulo en un robot móvil y verificar la precisión de sus mediciones contra un sistema de localización validado.

## 1.5. Organización de la Tesis

A continuación se describe brevemente el contenido de los capítulos restantes:

### **Capítulo 2: Técnicas para Localización en interiores**

Se describen distintas técnicas de localización en interiores, al final del capítulo se justifica la elección de la técnica utilizada en este trabajo.

### **Capítulo 3: Localización utilizando marcas de referencias visuales**

Se profundiza en la técnica que se basa en marcas artificiales en el terreno. Se estudian herramientas existentes, analizando ventajas y desventajas de cada una, y finalmente, se selecciona la herramienta a utilizar.

**Capítulo 4: Arquitectura del prototipo propuesto**

Se enumeran los componentes de hardware necesarios para la construcción del prototipo, se describe la arquitectura de hardware y software, y se explica el proceso de localización.

**Capítulo 5: Experimentos y resultados**

En este capítulo se realizan experimentos en un escenario real para validar el prototipo construido. Se evalúa el tiempo de ejecución, la precisión, la exactitud y el error de las localizaciones reportadas por el sistema propuesto.

**Capítulo 6: Conclusiones**

Finalmente se presentan las conclusiones en base a los experimentos realizados y resultados obtenidos. Se proponen posibles mejoras y trabajos futuros.



## Capítulo 2

# Técnicas para localización de robots móviles en interiores

En este capítulo se desarrolla un análisis introductorio de las técnicas más utilizadas para la localización de robots móviles en interiores. En el diagrama de la Figura 2.1 se puede observar el nombre de estas técnicas, y se las clasifica en dos categorías. Una categoría requiere una adaptación o modificación del ambiente, mientras que la otra utiliza el ambiente sin alteración alguna. A continuación se describe el funcionamiento de estas técnicas de manera general, y, finalmente, se concluye con la elección de la técnica seleccionada para este trabajo.

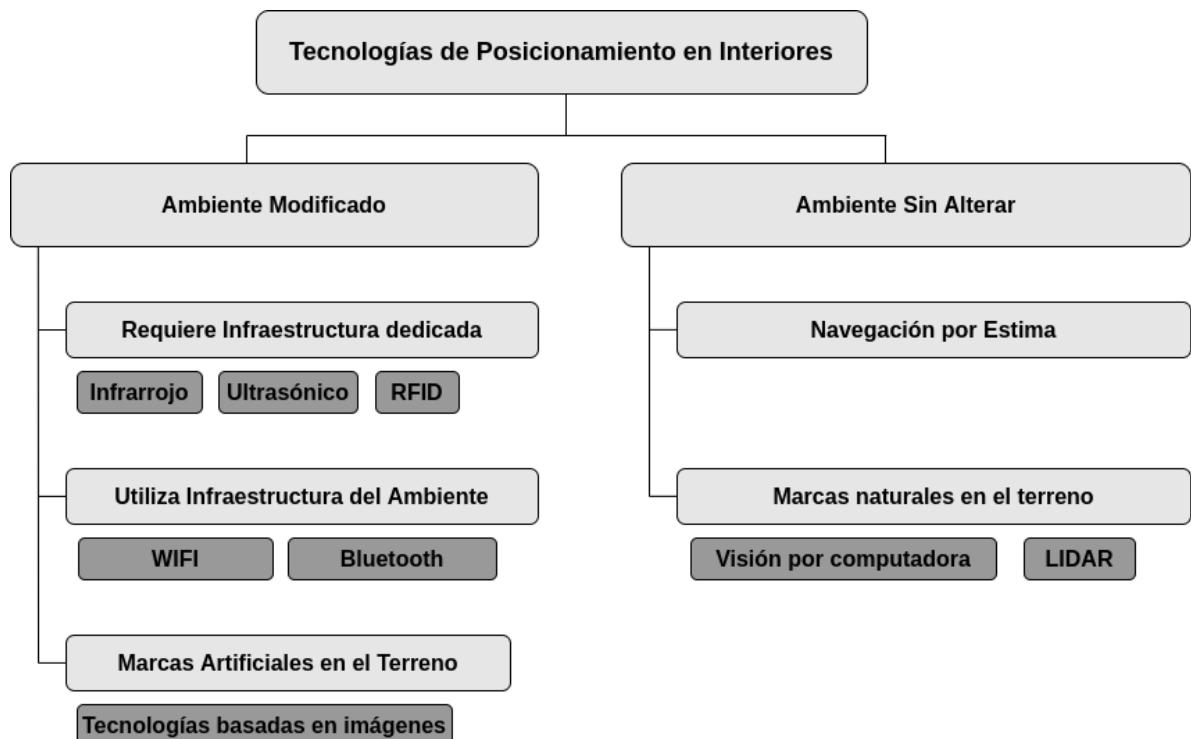


Figura 2.1: Tecnologías para localización de robots móviles en interiores

## 2.1. Técnicas que requieren modificación del ambiente

### 2.1.1. Requiere infraestructura dedicada

Una tecnología para posicionamiento requiere una infraestructura dedicada si es necesario agregar equipamiento extra al ambiente donde se la utilizará. Por ejemplo, la mayoría de los edificios actuales contienen WiFi, mientras que casi ninguno contiene equipamiento para identificación por radiofrecuencia. Por lo tanto, esta última, es una técnica que requiere infraestructura dedicada. Otras tecnologías utilizadas para posicionamiento en interiores que requieren una infraestructura dedicada son: Luz infrarroja y Ultrasonido.

#### Luz infrarroja

La luz infrarroja o radiación infrarroja (IR) es un tipo de radiación electromagnética. Su longitud de onda es mayor que la de la luz visible y menor que la de las microondas. La IR es invisible para el ojo humano en la mayoría de las condiciones, lo que hace, a esta tecnología, menos intrusiva en comparación con el posicionamiento en interiores basado en luz visible. Básicamente, la técnica consiste en tener una baliza emisora de IR y un receptor capaz de detectar la luz emitida por dicha baliza. En [24] se describe un ejemplo de localización que utiliza esta técnica, la cual consiste en una serie de balizas de IR ubicadas en puntos fijos dentro ambiente de trabajo y un dispositivo receptor giratorio montado sobre el robot. Este dispositivo es capaz de detectar la IR emitida por las balizas. Conociendo la velocidad de giro del receptor el sistema determina los ángulos entre balizas consecutivas a partir del tiempo que transcurre entre las detecciones de éstas. La posición del robot se estima a partir de estos ángulos mediante relaciones trigonométricas.

#### Ultrasonido

El sonido es una onda mecánica, una oscilación de la presión transmitida a través de un medio. Tiene la propiedad de propagarse a través de la materia sólida, líquida o gaseosa. El ultrasonido tiene iguales propiedades que el sonido, excepto que los humanos no podemos oírlas. Como la velocidad de propagación es constante, si una señal es emitida desde un punto  $A = (x_1, y_1)$  hasta un punto  $B = (x_2, y_2)$ , se debe simplemente conocer el momento de emisión y recepción para poder calcular la distancia desde A hasta B. Para conocer la posición de un robot móvil en un ambiente, se requiere de, al menos, tres receptores en posiciones fijas en el ambiente. Los tres receptores reciben la señal, calculan su distancia al robot móvil, y por medio de la técnica de trilateración se puede calcular la posición del robot.

Un ejemplo comercial de posicionamiento a través de ultrasonido es utilizando el sistema de *Marvelmind Robotics* [31]. En este sistema, el robot móvil tiene un emisor de ultra sonido y existen tres nodos fijos en el ambiente, R1, R2 y R3, los cuales son receptores. En la Figura 2.2 (a) se puede observar una gráfica de este escenario. Al menos un receptor debe estar localizado en la posición  $(x, y) = (0, 0)$  del mapa del ambiente, en este caso es R1. El robot (nodo móvil) puede navegar libremente dentro del ambiente delimitado por el área rectangular y posee un emisor ultrasónico que dispara una señal a 360 grados. Los receptores deben conocer el momento exacto en el cual se dispara dicha señal, para lo cual, los nodos están sincronizados con el robot. Cuando la señal es recibida por los receptores, cada uno de ellos calcula su respectiva distancia al robot. Individualmente, los receptores estiman que el robot se encuentra en algún punto de la circunferencia que se forma con el radio distancia calculado por cada uno ellos ( $d_1$ ,  $d_2$  y  $d_3$ ), como se observa la Figura 2.2 (a). Al realizar la intersección entre las tres circunferencias (trilateración) se obtiene la posición absoluta del robot en el ambiente, la cual es  $(x_1, y_1)$  en la Figura 2.2 (b).

En [40] se describe otra implementación de un sistema de posicionamiento global interior multi-robot que utiliza la técnica de ultrasonido.

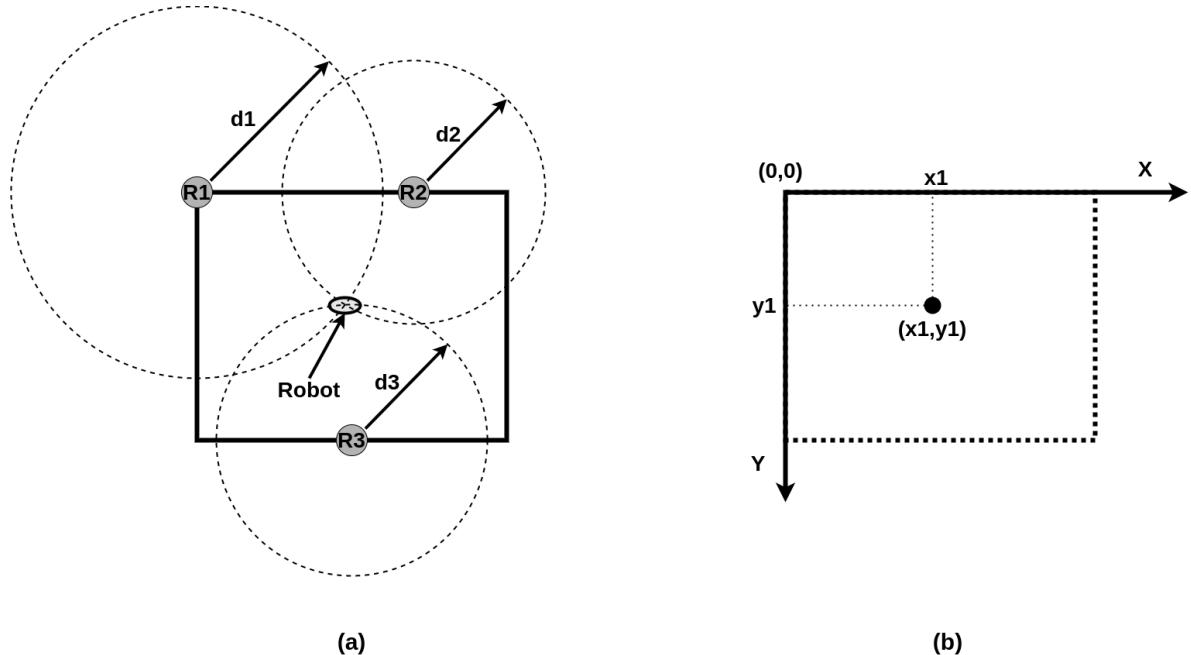


Figura 2.2: Ejemplo de localización utilizando ultrasonido

## RFID

Un sistema de identificación por radiofrecuencia (RFID) consiste en un lector que posee una antena. Este lector interroga transceptores activos cercanos o etiquetas pasivas (ambas llamadas simplemente *etiquetas*). Con la tecnología RFID, los lectores pueden obtener datos de las etiquetas a través de ondas de radio.

Las etiquetas RFID pasivas funcionan sin batería. Se utilizan principalmente para reemplazar la tecnología de código de barras tradicional y son mucho más ligeras, de menor volumen y costo que las etiquetas activas. Reflejan la señal de radio frecuencia (RF) que les transmite un lector y añaden información modulando la señal reflejada. Sin embargo, sus rangos son muy limitados. El rango de lectura típico es de 1 a 2 metros. Los sistemas RFID pasivos suelen utilizar cuatro bandas de frecuencia: LF (baja frecuencia), HF (alta frecuencia), UHF (ultra alta frecuencia) y frecuencia de microondas.

Las etiquetas RFID activas son pequeños transceptores que pueden transmitir activamente su ID, además de otros datos adicionales, en respuesta a una interrogación. Los rangos de frecuencia utilizados son similares a los del caso RFID pasivo. Las ventajas de la RFID activa son las antenas más pequeñas y con mayor alcance (decenas de metros) que las RFID pasivas, aunque su desventaja es que requieren energía para funcionar.

En [27] se presenta *Landmarc*, un sistema de localización prototípico experimental que utiliza la tecnología RFID. En su primer intento instalan varios lectores como se ve en la Figura 2.3. Cada lector tiene un nivel de potencia predeterminado, definiendo así un cierto rango en el que puede detectar etiquetas RFID. Al colocar correctamente los lectores en ubicaciones conocidas, toda la región se puede dividir en una serie de subregiones, donde cada subregión se puede identificar de manera única por el subconjunto de lectores que cubren esa subregión. Dada una etiqueta RFID, según el subconjunto de lectores que pueden detectarla, se puede asociar esa etiqueta con una subregión conocida. La precisión de este enfoque se determina luego por la cantidad de lectores requeridos, la ubicación de estos lectores y el nivel de potencia de cada lector. Hay muchos factores que afectarán el alcance, incluidas las obstrucciones estáticas y el movimiento humano dinámico. Debido a estas interferencias dinámicas, incluso un objeto estático podría informarse en diferentes subregiones de vez en cuando. En el ejemplo presentado en la Figura

2.3 un robot móvil podría transportar una etiqueta, y la localización del robot será la subregión o intersección de subregiones de los lectores que la detecten.

La tecnología RFID es de muy bajo costo, en comparación con WiFi, o Bluetooth, aunque se requiere en general poblar todo el ambiente, donde un robot móvil podría navegar, con decenas o centenas de lectores o etiquetas.

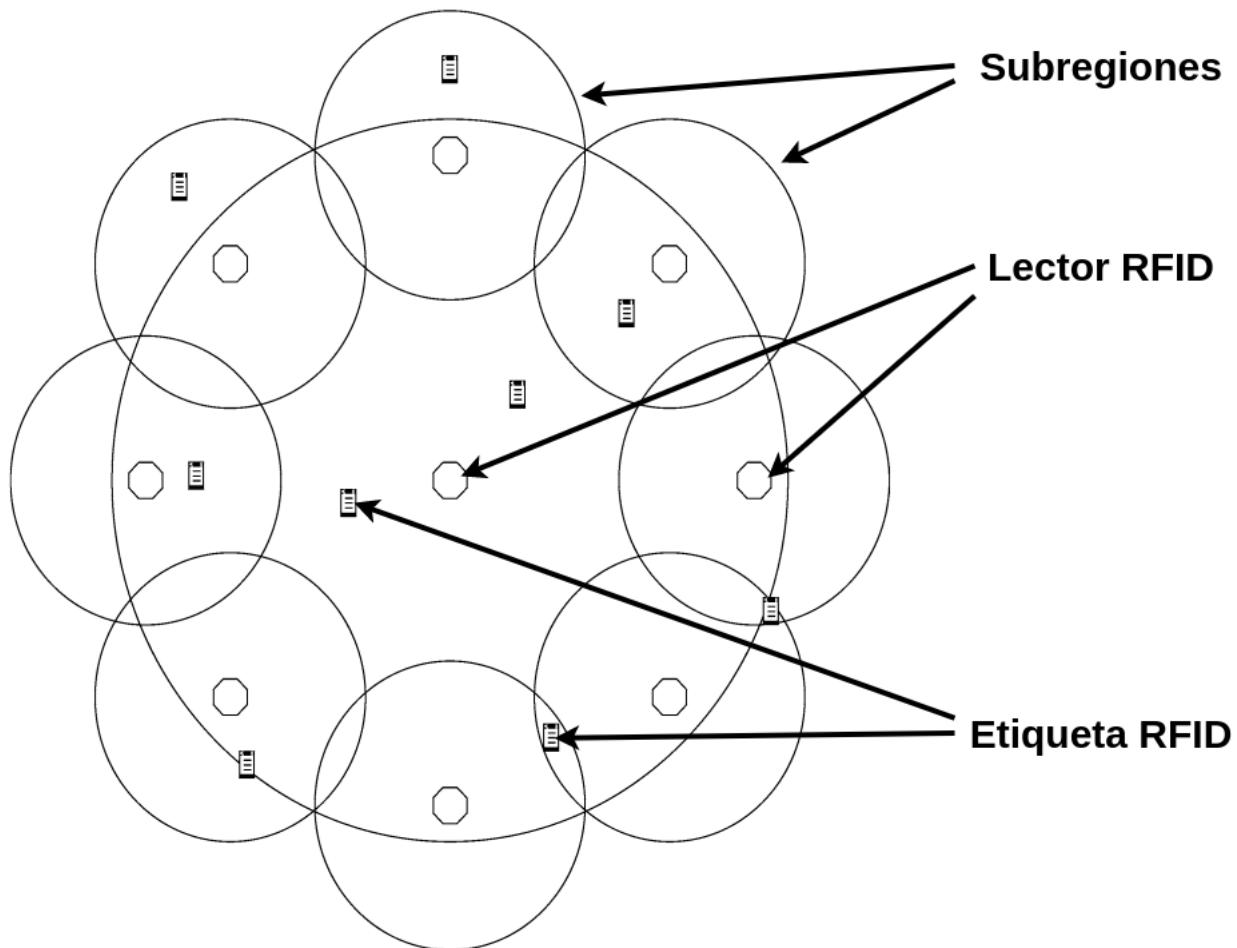


Figura 2.3: Ejemplo de localización utilizando RFID

### 2.1.2. Utiliza infraestructura del ambiente

Las tecnologías de posicionamiento en interiores que utilizan la infraestructura del ambiente son: WiFi y Bluetooth.

#### WiFi

WLAN (Red de Área Local Inalámbrica) es una de las formas de comunicación de datos más populares que existen en la actualidad. Básicamente, se trata de un sistema que permite que diferentes dispositivos electrónicos se conecten a las redes de comunicación a través de un punto de acceso de red inalámbrico. WiFi es una marca de la compañía *Wi-Fi Alliance* que está a cargo de certificar que los equipos cumplen con la normativa vigente (estándar IEEE 802.11). El uso de WiFi en sistemas de navegación y posicionamiento en interiores depende de conocer una lista de puntos de acceso inalámbricos que están distribuidos en el área en la cual opera el sistema. El método de posicionamiento de WLAN más popular es a través del uso de la intensidad de la

señal recibida (RSS), la cual se puede extraer sin mayores inconvenientes en redes 802.11 [23]. La intensidad de la señal es siempre la misma en la ubicación desde donde se la mide, sin importar el momento del día. Igualmente en días posteriores, si la ubicación de medición es la misma, la intensidad obtenida de cada punto de acceso es la misma.

La técnica que se utiliza para localizar un robot en un ambiente de interior utilizando la intensidad de la señal de los puntos de acceso se denomina fingerprint<sup>1</sup>. Consiste en, primeramente, realizar una base de datos del mapa del ambiente. Se coloca al robot en todas las posiciones posibles del ambiente. En cada posición, se obtienen las intensidades de señales de todos los puntos de acceso al que el robot, en esa ubicación, tenga acceso. Se registran todas esas mediciones en la base de datos, junto a la ubicación. Una vez completada la base de datos, el robot puede navegar en el ambiente. Por cada posición en donde navegue, el robot obtiene las intensidades de señales de los puntos de acceso WiFi, y un algoritmo contrasta estas intensidades y selecciona de la base de datos la ubicación mas probable.

Utilizando el método RSS, la precisión de los sistemas de posicionamiento WLAN pueden ser de alrededor de 3 a 30 m [20].

## **Bluetooth**

Bluetooth opera en la banda de 2,4 GHz. En comparación con WLAN, la tasa de bits bruta es menor (1 Mbps) y el rango es más limitado (10 a 15 m). La tecnología Bluetooth está integrada en la mayoría de los teléfonos móviles, asistentes digitales personales (PDA), etc. Las etiquetas Bluetooth son transceptores de pequeño tamaño y tienen una identificación única (ID). Este ID se puede utilizar para localizar las etiquetas [30].

Antti et al. presentan un posible diseño e implementación de una Aplicación de Posicionamiento Local Bluetooth (BLPA) [19], que puede ser utilizada para localizar un robot móvil. En un escenario ubican tres dispositivos Bluetooth en ubicaciones fijas conocidas. El robot móvil lleva un cuarto. En cada posición el robot móvil obtiene los niveles de potencia de cada Bluetooth fijo, y convierte esos niveles en estimaciones de distancia según un modelo de propagación simple. Utilizando un filtro de Kalman extendido y un sistema de trilateración como el explicado en la sección 2.1.1 calculan la posición del móvil. Se informa que la precisión de BLPA es de 3,76 m. Un trabajo similar ha sido realizado por Hallberg et al. [14].

### **2.1.3. Marcas artificiales en el terreno**

#### **Tecnologías basadas en imágenes**

Las técnicas basadas en imágenes consisten en colocar marcas artificiales en el terreno (etiquetas o TAGs), de manera que, estas puedan ser únicamente identificadas de manera fácil y precisa; usando simplemente una cámara y una computadora (visión por computadora). Por ejemplo, un código QR impreso y colocado en el ambiente. Si bien con estos métodos se puede evitar el uso de varios de los sensores que se comentaron en secciones anteriores, aún requiere visión por computadora para reconocer la ubicación y orientación de la marca artificial. Los algoritmos de visión por computadora para la detección de estas marcas artificiales suelen implementarse y verificarse en computadoras con recursos suficientes para procesar un gran número de imágenes por segundo, y permitir el uso de un amplio rango de bibliotecas y aplicaciones al mismo tiempo. Por ejemplo, OpenCV<sup>2</sup> para la captura y filtrado de imágenes, y el entorno MATLAB para programación.

Existen algunos pocos sistemas de marcas artificiales en el terreno basada en códigos en imágenes que tengan la característica de fácil de identificar. Entre los mas populares están: Artoolkit [36], ArTag [13] y AprilTag [28]. Como punto en común todos ellos utilizan un cámara

---

<sup>1</sup>La técnica fingerprint no tiene traducción al español aceptada. La traducción general es huella dactilar.

<sup>2</sup><https://opencv.org/>

digital para la detección de la marca de referencia, y mediante algoritmos que analizan la imagen obtenida pueden detectar el TAG y calcular la posición del mismo respecto a la cámara. La diferencia recae en los tipos de marcadores sobre la imagen que utiliza cada sistema, y la forma en la cual implementan el algoritmo de detección y localización. En la Figura 2.4 se puede observar ejemplos de TAGs de cada sistema. Los trabajos que documentan estos sistemas concluyen que es posible lograr una precisión en la localización en el rango de centímetros.

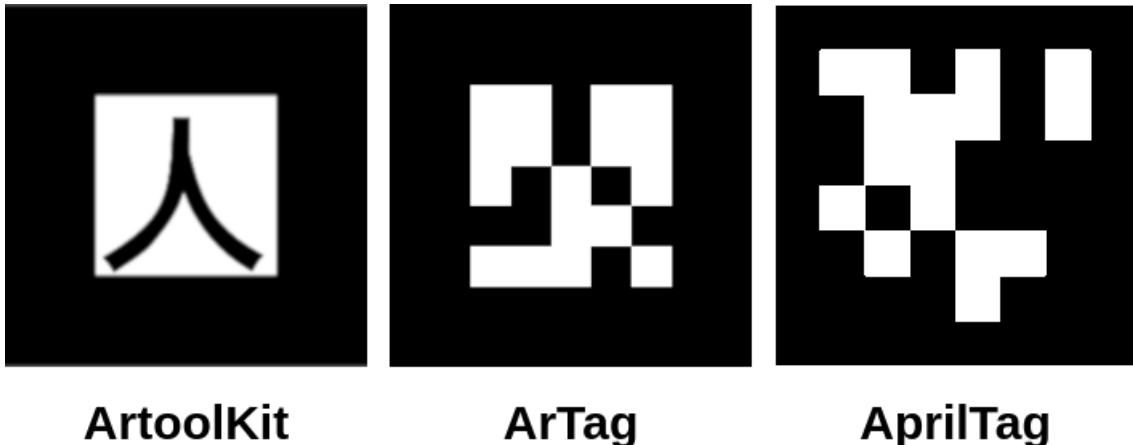


Figura 2.4: Ejemplos de marcadores de ArtoolKit, ArTag y AprilTag

En [41] se presenta el diseño de un sistema para localización en interiores utilizando AprilTag. La experimentación se llevó a cabo mediante un robot móvil Pioneer 3-DX, que contó con una notebook empotrada con procesador Intel Core 2 Duo 2.0 GHz y 2G de RAM. El sistema toma imágenes con la cámara, detecta el tag, y calcula la distancia y el ángulo a la misma. A partir de estos datos, se puede localizar el robot en el ambiente.

## 2.2. Técnicas que no alteran el ambiente

Los mecanismos de localización mas utilizados, que no requieren modificar el ambiente, son la navegación a estima y las tecnologías basadas en imágenes.

### 2.2.1. Navegación por estima

La mayoría de los vehículos autónomos terrestres utilizan ruedas para moverse en el ambiente de operación. El método más directo para seguir su posición es medir la velocidad angular de las ruedas y la orientación del mecanismo de dirección. Con esta información se puede estimar la siguiente posición del vehículo, y esta predicción se denomina navegación por estima. La idea general es sencilla, y se puede describir como "contar pasos al caminar", o "contar la cantidad de giros de la rueda", etc. El término proviene de la navegación antigua, donde el capitán del barco a vela calculaba su siguiente posición utilizando la velocidad de viaje (en base al movimiento del barco sobre el agua y el viento, etc) y su orientación inicial (en base al sol, las estrellas, etc). El conjunto de sensores que miden la dinámica de los estados de un vehículo, velocidad, aceleración o variaciones angulares, se denominaron sensores internos o proprioceptivos. En este tipo de sensores se incluyen sensores inerciales tales como los acelerómetros y giroscopios, codificadores incrementales o *encoders* en inglés (ópticos, inductivos, capacitivos, magnéticos), tacómetros, sincros, resolutores, transductores de desplazamiento lineal variable (LVDT), compases y potenciómetros. En [4] se utiliza un sistema embebido de navegación por estima para la localización de robots móviles en ambientes de interiores.

Los sensores internos proveen información de movimiento en forma incremental a lo largo de una trayectoria y la posición es normalmente obtenida a través de la integración temporal de la secuencia de medidas. Consecuentemente los errores de medición son también integrados lo que implica que estos crezcan sin límite. Se hace necesario, por lo tanto, modelar cuidadosamente este error para disminuir sus efectos adversos en la predicción de trayectorias cubriendo distancias largas. En general los errores crecerán con el tiempo y por lo tanto hacen que el sistema de navegación no pueda basarse sólo en este tipo de sensores.

### 2.2.2. Marcas naturales en el terreno

#### Visión por computadora

Una de las técnicas para localización más estudiadas y complejas es utilizar una cámara (o un grupo de cámaras), para capturar y procesar fotos de manera constante, con el fin de detectar marcas o formas llamativas existentes naturalmente en el ambiente, y de esta manera, estimar la ubicación de la cámara dentro del ambiente. Esta técnica se ha implementado de diferentes maneras, pero un mecanismo ampliamente utilizado, consiste en detectar, de las fotos capturadas, características, por ejemplo bordes, esquinas, sombras, cambios de colores bruscos, etc. Estas características, o marcas naturales en el terreno, se almacenan en una base de datos y se las intenta detectar nuevamente en subsiguientes fotos capturadas. En base a estas re-detecciones que van sucediendo, el sistema estima una ubicación en un espacio 3D de cada característica, utilizando en esta información de la cámara, y estimando la velocidad de movimiento de la misma. Actualmente, esta técnica se utiliza para armar el mapa y estimar la localización de la cámara en el mapa en simultáneo, y se la denomina mapeo y localización en simultáneo (*SLAM* en inglés). El uso de la cámara como herramienta de captura de información del ambiente conlleva a que el sistema sea denominado *visual SLAM* [10].

Como se puede intuir, preliminarmente, esta técnica requiere un sensor de baja complejidad (la cámara), y computadoras con un alto poder de procesamiento, para que sea capaz de procesar en tiempo real una gran cantidad de fotos por segundo (usualmente 30 fotos por segundo)[25, 26].

## LIDAR

El LIDAR<sup>3</sup> (*Light Detection and Ranging*) es un tipo de sensor activo de rango que provee directamente información de distancia y ángulo de los obstáculos, y es insensible a condiciones de iluminación externa que perjudica a sensores pasivos como las cámaras de vídeo. El sensor opera con el principio de la medida del tiempo de reflexión de la luz emitiendo una serie corta de pulsos al mismo tiempo que se activa un circuito detector. Si la luz encuentra un objeto en su camino, es reflejada hacia el sensor. Con el tiempo entre la emisión y la recepción el sensor calcula la distancia al objeto. Los láseres mas comunes tienen un espejo rotante que desvía los pulsos de luz de manera de barrer un área semicircular. Determinando el ángulo del espejo, se determina en qué dirección está el objeto.

La tecnología lidar tiene diversas aplicaciones como en geología, sismología y física de la atmósfera. Pero también se utiliza en vehículos autónomos, ya que, resulta de gran utilidad debido a la precisión con la que esta tecnología es capaz de medir distancias de todo su entorno circundante (Figura 2.5). La localización de un robot móvil de interior se puede realizar con un LIDAR y un sistema que contiene un mapa del ambiente previamente digitalizado. A medida que el robot se mueve el LIDAR puede detectar todo su entorno, y un algoritmo mapea ese calculo del entorno dentro del mapa digitalizado, para encontrar la posición del robot en el ambiente.

---

<sup>3</sup><https://proyectoidis.org/laser-lidar/>

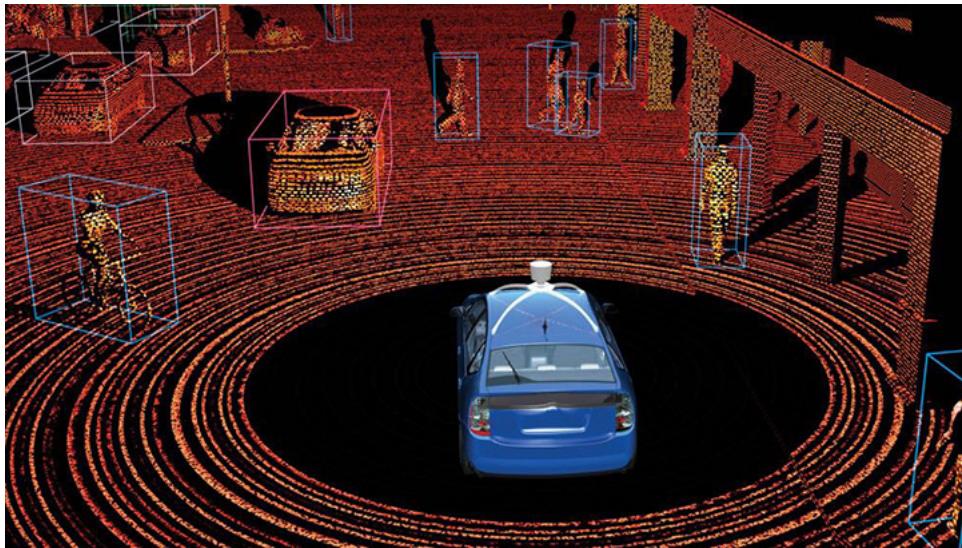


Figura 2.5: Vehículo autónomo utilizando la tecnología LIDAR

### 2.3. Comparativa entre las distintas técnicas de localización

Si bien todas las técnicas descritas pueden ser utilizadas para localización de robots en interiores, estas son difíciles de comparar directamente entre sí, principalmente porque utilizan tecnologías diferentes. Por ejemplo, la técnica utilizando marcadores RFID es muy distinta a la localización basada en señales de WiFi. Los marcadores RFID pueden devolver una señal con un identificador, a partir de una señal recibida. Es una técnica más bien binaria: o se detecta un marcador o no se detecta. Si se detecta, simplemente se conoce que se está a una cierta distancia cercana del mismo. En cambio, la técnica de WiFi utiliza la potencia de la señal proveniente de un access point. A partir del valor que describe la potencia de la señal, se puede estimar una distancia desde el access point. Como estas señales pueden alcanzar decenas de metros, esta potencia puede ser utilizada como un indicador de la posición en regiones amplias, con valores más bien continuos de distancias estimadas, en contraste con la técnica binaria RFID, que sólo permite conocer si se está al lado o no de un marcador.

Por tal motivo, para poder realizar una elección de la técnica a utilizar en este trabajo, se definieron cuatro características deseables del sistema final, las cuales, son características mayormente cualitativas. Estas características deseables se utilizarán como parámetros de comparación entre las diferentes técnicas descriptas.

La siguiente lista presenta estas características deseables.

- A) *Sencillo de configurar y reubicar*: la característica sencillo de configurar y reubicable se define en este trabajo como la capacidad de poder utilizar el sistema rápidamente. Que no se requiera grandes tiempos de instalación de los componentes de hardware en el escenario, y que no se requiera mucho tiempo de calibración, realizando por ejemplo, muchas mediciones previas a su uso. Además, que todo el sistema no esté atado a un espacio físico fijo y sea fácilmente reubicable, ya que, los robots educativos se utilizan en diversos eventos y lugares.
- B) *De bajo costo*: la característica de bajo costo estará definida por el valor del hardware principalmente. En este trabajo definimos bajo costo cuando el valor no supera al costo del robot educativo.
- C) *Utilizable en escenarios acotados*: la característica utilizable en escenarios acotados se define, en este trabajo, en función de la utilización que se le da a los robots educativos, cuyo ambiente de trabajo es de aproximadamente algunos pocos metros cuadrados.

- D) *Precisión de centímetros*: la característica precisión de centímetros viene acompañada de la característica anterior, se requiere de un sistema que permita localizar con utilidad al robot educativo dentro de ese escenario. Y también tiene relación directa con el tamaño de los robot educativo Frankestito-N6 usado por la Facultad. Debido a su pequeño tamaño, se requiere que el sistema de localización pueda reportar movimientos que pueden ser de unos pocos centímetros de trayectoria.

En la Tabla 2.1, se presenta una comparativa de las técnicas descriptas, evaluando para cada técnica si cumple o no con cada característica deseable del sistema de localización que se requiere para el presente trabajo.

Comparativa de técnicas de localización en interiores				
Técnica de localización	A	B	C	D
<b>Ambiente modificado</b>				
Luz infrarroja			X	
Ultrasonido		X	X	X
RFID			X	
WiFi	X	X	X	
Bluetooth		X	X	
Basada en imágenes	X	X	X	X
<b>Ambiente sin modificar</b>				
Navegación por estima	X	X	X	
Visión por computadora			X	X
Lidar	X		X	X

Tabla 2.1: Tabla comparativa entre las distintas técnicas de localización

## 2.4. Conclusiones

Comenzamos analizando las técnicas que no necesitan modificar el ambiente. La navegación por estima presenta el inconveniente de que con el paso del tiempo, el error acumulado suprime la característica *precisión de centímetros*. Las técnicas basadas en imágenes (visión por computadora) que utilizan marcas naturales en el terreno presentan la desventaja de no ser *de bajo costo*, ya que necesitan contar con cámaras de vídeo de alta resolución y computadoras con gran poder de cómputo. El sensor de láser lidar tiene un costo demasiado elevado en comparación con los Frankestitos-N6.

Continuamos con las técnicas que necesitan modificar el ambiente. Las técnicas de Luz Infraarroja, Ultrasonido y RFID requieren una infraestructura dedicada, montada en un ambiente fijo, lo que las hace difícil configurar y reubicar. La técnica de ultrasonido por ejemplo requiere que se instale en el ambiente una infraestructura compleja, ya que todos los nodos (fijos y móviles) requieren de una fuente de energía, además, la sincronización en estos sistema es un requerimiento critico. Dentro de las técnicas que necesitan modificar el ambiente también están las que utilizan la infraestructura del mismo, como las técnicas WiFi y Bluetooth. Estas presentan como desventaja de que están condicionadas a un ambiente fijo (por la ubicación de las antenas y puntos de acceso, etc.), y la precisión que se logra no está en el rango deseado, es decir, no presentan la característica de *precisión de centímetros*. Aunque las técnicas de WiFi y Bluetooth podrían ser construidas en otro ambiente distinto, esto no es tan *sencillo de configurar y reubicar*.

Del análisis previo concluimos que la técnica más conveniente para el presente trabajo es la basada en imágenes utilizando marcas de referencias artificiales en el terreno. Estas son *sencillas de configurar y reubicables* porque en el escenario solo hay que ubicar los TAGs, los cuales se

## 16 CAPÍTULO 2. TÉCNICAS PARA LOCALIZACIÓN DE ROBOTS MÓVILES EN INTERIORES

puede instalar y extraer fácilmente. La calibración de la cámara respecto al tamaño del TAG se puede configurar con anterioridad de manera que al momento de querer utilizarlo, no se requiera calibrar. Son *de bajo costo*, ya que la cámara de vídeo requerida es simplemente una cámara genérica de baja resolución, y los TAGs pueden fabricarse fácilmente con impresora y papel común. Son *utilizables en escenarios acotados* y se puede definir el tamaño de los mismos de manera flexible. Además, en buenas condiciones de iluminación y calibración de la cámara se puede lograr una *precisión de centímetros* en la localización. El desafío restante es implementar el sistema en una computadora de limitada prestaciones como la equipada en los Frankestitos-N6.

En el siguiente capítulo se realiza un análisis de las técnicas más comunes basadas en imágenes, utilizando marcas de referencias artificiales en el terreno, con el fin de decidir la más conveniente para el diseño e implementación del prototipo de localización de este trabajo.

## Capítulo 3

# Localización utilizando marcas de referencias visuales

Los marcadores fiduciales visuales son puntos de referencia artificiales diseñados para ser fáciles de reconocer y distinguir unos de otros. Aunque están relacionados con otros sistemas de códigos de barras 2D como los códigos QR, tienen objetivos y aplicaciones diferentes. En un código QR, es necesario alinear la cámara con la etiqueta y tener una imagen de alta resolución, donde se pueden obtener cientos de bytes, como una dirección web. Por el contrario, un marcador fiducial tiene una pequeña carga útil de información, pero está diseñado para ser detectado y localizado con mayor facilidad, incluso cuando la resolución de la cámara es baja, la iluminación es defectuosa, se encuentra rotado, o está localizado parcialmente en la esquina de una imagen. Por este motivo, los marcadores fiduciales son útiles para Realidad Aumentada (AR), navegación de robots y aplicaciones generales donde se requiere la pose relativa entre una cámara y un objeto.

En este capítulo, se presentan detalles de los sistemas de marcas fiduciales y se describen las principales implementaciones existentes en la actualidad. En particular, se explica el mecanismo de detección de cada sistema fiducial, se realiza una comparativa de sus características y, por último, se selecciona el sistema fiducial más adecuado para utilizar en este trabajo.

### 3.1. Sistemas de marcas fiduciales

Los sistemas de marcadores fiduciales consisten en patrones que se montan en el entorno de trabajo donde se requiere localizar la pose relativa de la cámara con un objeto. Utilizando un algoritmo que analiza una imagen obtenida desde la cámara digital se los pueden detectar. Localizando la cámara con respecto a un objeto ofrece la posibilidad de calcular la ubicación de la cámara en un ambiente (si el objeto con la marca fiducial está fijo en el ambiente), o la ubicación del objeto en sí (y su posición) con respecto a la cámara.

Existen varios algoritmos similares que implementan los sistemas de marcadores fiduciales, y debido a que todos ellos se basan en el análisis de imágenes digitales, su rendimiento o eficacia dependen de los parámetros descriptos a continuación:

- La *tasa de falsos positivos*: es la tasa de informar erróneamente la presencia de un marcador cuando no hay ninguno presente.
- La *tasa de confusión entre marcadores*: es la tasa de cuándo el sistema detecta un marcador, pero se lo identificó incorrectamente, es decir, un marcador se confundió con otro.
- La *tasa de falsos negativos*: es la tasa de informar erróneamente que no existe un marcador en la imagen, cuando en realidad si lo hay.
- El *tamaño mínimo del marcador*: es el tamaño en píxeles requerido para una detección confiable. Cuanto más pequeño sea el marcador en la imagen, mayor será el rango utilizable

desde la cámara del sistema de marcadores, es decir, podrá detectar varios marcadores en simultáneo y a mayores distancias.

- La *robustez a condiciones de iluminación*: este parámetro es una señal analógica del mundo real difícil de modelar (la luz es la parte de la radiación electromagnética que puede ser percibida) debido a que presenta valores infinitos. En la práctica, se deben realizar ensayos y análisis bajo estrictas condiciones de experimentación para poder evaluar diferentes sistemas con respecto a las condiciones de luz.
- El *jitter de detección*: se denomina jitter a un cambio indeseado y abrupto de la propiedad de una señal y suele considerarse como una señal de ruido no deseada. Es un factor importante en los sistemas de Realidad Aumentada. La estimación actual de la pose por lo general se basa sólo en el conjunto de características visible. La cantidad de jitters aumenta cuando el conjunto de características cambia significativamente entre los frames.

Las tasas de falsos positivos y falsos negativos están relacionadas entre sí, y representan una compensación entre perder un marcador y ver uno inexistente. Se tiende a bajar la tasa de falsos positivos aunque esto deriva en que aumente la tasa de falsos negativos, ya que, es preferible pasar por alto un marcador existente en la imagen, que detectar uno que no existe.

### 3.2. ARToolkit

ARToolkit [1] es una de las primeras implementaciones en trabajar con este sistema (1999), y sirvió como base para muchas implementaciones posteriores [18]. ARToolKit es una biblioteca de software en lenguaje de programación C que permite a los programadores desarrollar fácilmente aplicaciones de Realidad Aumentada. Utiliza técnicas de visión por computadora para calcular el punto de vista de la cámara real en relación con un marcador del mundo real.

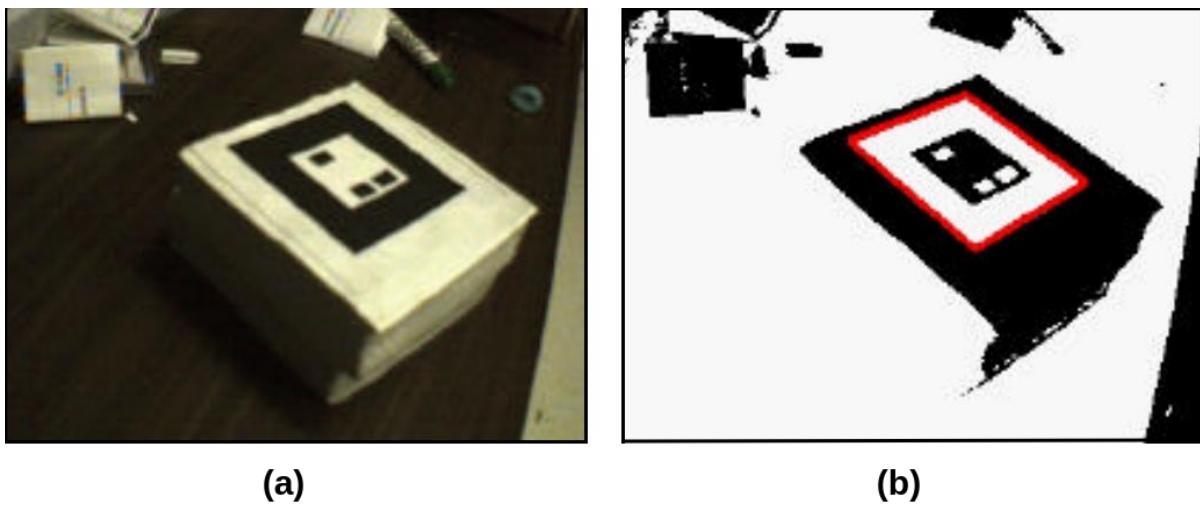


Figura 3.1: Proceso de detección de marcadores ARToolKit. (a) Imagen original capturada. (b) Conversión a imagen blanco y negro.

#### Detector

El detector de ARToolKit convierte la imagen capturada por la cámara, Figura 3.1 (a), en una imagen en blanco y negro, basada en un valor umbral de iluminación para decidir si el pixel es blanco o negro, Figura 3.1 (b). En esta imagen se buscan regiones cuadradas, ARToolKit

encuentra todos los cuadrados en la imagen binaria. Luego, para cada región cuadrada detectada, se obtiene el patrón dentro de la misma y se lo compara con un conjunto de plantillas de patrones conocidas que ARToolKit tiene en una base de datos. En la Figura 3.2 se puede observar un ejemplo de un marcador de referencia ARToolKit. Si hay una coincidencia, entonces se ha encontrado uno de los marcadores de referencia. Luego se usa el tamaño del cuadrado y la orientación del patrón para calcular la posición de la cámara de vídeo en relación con el marcador de referencia [29].

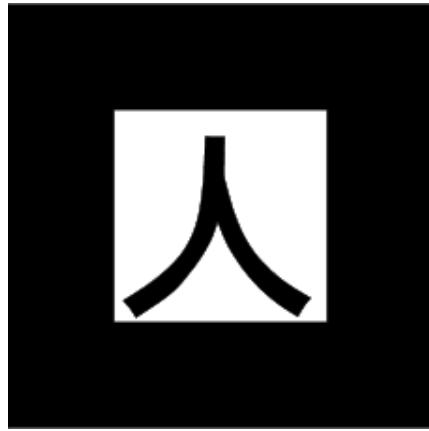


Figura 3.2: Ejemplo de marcador de referencia ARToolkit

### 3.3. ARTag

ARTag [13, 12] es un sistema de marcadores fiducial compuesto de 2002 marcas, las cuales consisten en un contorno cuadrado como los de ARToolkit. La detección de ese contorno se basa en la detección de la línea que limita su superficie, y el procesamiento del patrón interno de la marca se realiza con un algoritmo digital que reconoce los 2002 patrones, sin utilizar una correlación con una base de datos de imágenes, como se realiza en ARToolkit. Al utilizar patrones digitales representando un código binario, ARTag mejora la tasa de falsos positivos, y de confusión entre marcadores, que presenta ARToolkit. El algoritmo asociado para la detección localiza primero cuadriláteros que pueden ser vistos en perspectiva del borde del marcador, luego el interior se muestrea en 36 símbolos binarios '1' o '0'. El procesamiento adicional se realiza en el dominio digital y proporciona una respuesta no lineal.

La característica de ARToolkit que más contribuye a su funcionalidad es el uso de solo blanco y negro para el borde. El uso de solo dos extremos de reflectancia en un marcador permite evitar muchos problemas de captura de imágenes y no linealidad en escala de grises. Esta decisión binaria se extiende en ARTag desde la simple definición del borde hasta la definición del patrón interno. ARTag fue diseñado para contener los elementos exitosos de ARToolkit y Datamatrix<sup>1</sup>, y tomar lo mejor de ambos y hacer un sistema mínimo pero robusto para Realidad Aumentada.

En la Figura 3.3 se muestra un ejemplo de marcador ARTag. Las características principales son un borde cuadrado de polaridad (blanco sobre negro o negro sobre blanco) y una cuadrícula cuadrada de 6 x 6 que divide el interior. El marcador completo es de 10 x 10 unidades, con un borde de grosor de 2 unidades que deja 36 celdas en el interior para transportar información, cada celda transporta un bit de datos digitales.

---

<sup>1</sup>[https://es.wikipedia.org/wiki/Matriz\\_de\\_datos](https://es.wikipedia.org/wiki/Matriz_de_datos)

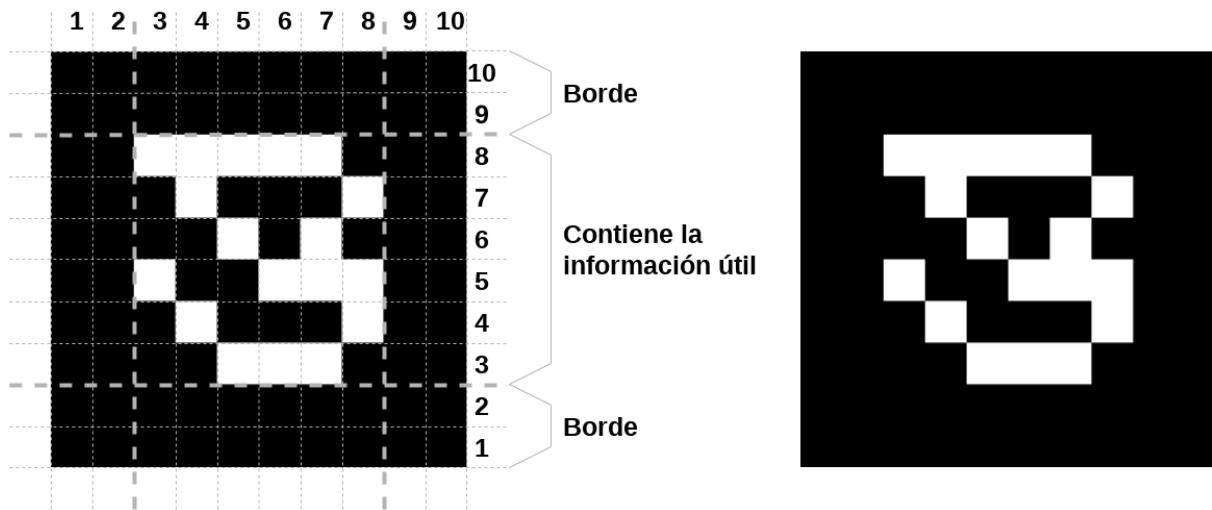


Figura 3.3: Ejemplo de marcador de referencia ARTag de 36 bits

## Detector

Se buscan los contornos cuadriláteros en la imagen que pueden pertenecer al borde exterior de un marcador. Se utiliza un método basado en bordes, los píxeles de bordes se agrupan y se unen en segmentos, que a su vez se agrupan en cuadriláteros. Las cuatro esquinas del límite del cuadrilátero se utilizan para crear un mapeo de homografía para muestrear el interior del marcador. La Figura 3.4 muestra la captura de una imagen, los segmentos de línea encontrados en la misma, los segmentos de linea agrupados en cuadriláteros, y los cuadriláteros en los que se encontraron códigos de marcador ARTag en el interior. Notar que el marcador ARTag con ID #2 tiene su borde de cuadrilátero ubicado, pero falla la validación interior debido a que él bolígrafo oculta demasiados bits de datos. El enfoque basado en el borde ofrece una mejora en el rendimiento sobre el enfoque de umbral de la región en escala de grises del sistemas ARToolkit. En ARToolkit, se encuentran grupos de píxeles conectados por debajo de un umbral específico, y aquellos que poseen un límite cuadrilátero se usan como marcadores potenciales. Tener una derivada espacial del umbral de intensidad de escala de grises en lugar de un simple umbral de intensidad de escala de grises permite encontrar marcadores en condiciones de iluminación menos controladas. De hecho, el nivel "blanco" del borde de un marcador puede ser más oscuro que el nivel "negro" del otro lado y el marcador aún se puede detectar. Otra ventaja del enfoque basado en bordes de ARTag es la capacidad de detectar contornos de marcadores en presencia de oclusión. En la Figura 3.4, dos de los marcadores (ID #103, #104) están parcialmente tapados, pero aún así son detectados por la heurística de segmentos de línea.

Para reducir aún más la tasa de falsos negativos, ARTag busca cuadrilateros en tres escalas. La extracción del segmento de línea y la agrupación en cuadrilateros se realiza en el tamaño de la imagen original, en una versión muestreada de la mitad del ancho y la altura, y en la de un cuarto de tamaño. Esto permite la detección de bordes que pueden ser borrosos y no tener una derivada espacial por encima del umbral.

## Procesamiento digital

Una vez que se han ubicado los contornos del borde cuadrilátero, la región interna se muestrea con una cuadrícula de 6 x 6 y se asignan los símbolos digitales '0' o '1'. El umbral aplicado se deriva de las intensidades encontradas alrededor del borde cuádruple. Luego se obtienen cuatro secuencias binarias de 36 bits del conjunto de símbolos de la cuadrícula de 6 x 6, una para cada

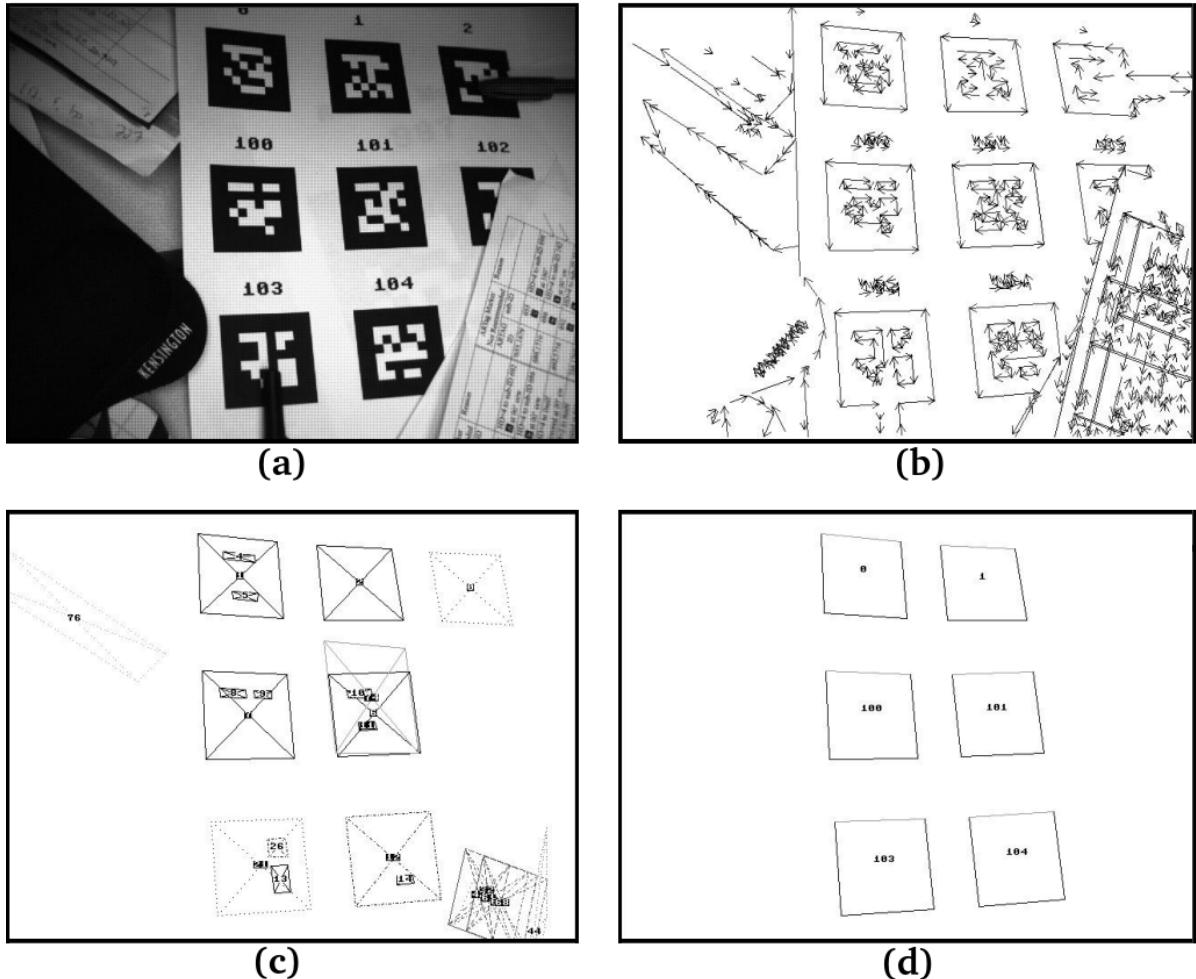


Figura 3.4: Proceso de detección de marcadores ARTag. (a) Imagen original. (b) Segmentos de linea encontrados en la imagen. (c) Segmentos de lineas agrupados en cuadrilateros, (d) Marcadores ARTag encontrados a partir de los cuadriláteros cuyo interior contienen un código ARTag válido.

una de las cuatro posibles posiciones de rotación. Solo una de las cuatro secuencias puede terminar siendo validada en el proceso de decodificación. La secuencia binaria de 36 bits codificada en el marcador encapsula un identificador único de 10 bits. Los 26 bits restantes proporcionan redundancia para bajar la tasa de confusión de falsos positivos, y para proporcionar unicidad sobre las cuatro rotaciones posibles. Se utiliza verificación de redundancia cíclica (CRC) y corrección de errores de reenvío (FEC) para identificar si el código de 36 bits es parte del conjunto de marcadores ARTag, y para extraer su ID. CRC es un código de detección de errores para identificar cambios accidentales en los datos y FEC es un tipo de mecanismo de corrección de errores que permite la corrección en el receptor sin retransmitir de la información original.

Dos números de identificación de ARTag están ausentes de la biblioteca de marcadores, lo que reduce el tamaño de esta a 2046. Se trata de los IDs #682 y #1706, la razón de esto es que los IDs mencionados son un caso particular que conduce a un código de 36 bits que contiene todos '0'. Esto se traduce en un interior totalmente negro o blanco, que con frecuencia se detectará falsamente en el entorno. La biblioteca se reduce aún más, a 2002 identificadores, mediante la eliminación de otros 44 identificadores para mejorar la tasa de confusión entre marcadores como se describe en [12].

### Singularidad de los marcadores

La tasa de confusión entre marcadores para ARTag puede analizarse considerando la probabilidad de confundir un código de símbolos digitales con otro. Una medida de la facilidad con que dos códigos binarios se pueden confundir entre sí es calcular la distancia de Hamming [15], que es simplemente la suma de las diferencias entre dos secuencias digitales. Por ejemplo, la distancia de Hamming entre las secuencias 01001 y 00011 es 2. La probabilidad de un evento de confusión entre marcadores se puede calcular utilizando el conocimiento de las distancias de Hamming dentro de un conjunto de marcadores entre todos los marcadores, teniendo en cuenta la rotación y, opcionalmente, el reflejo. Como puede observarse con mayor detalle en [12].

### 3.4. AprilTag

El diseño de AprilTag [28] se basa en los anteriores ARToolkit y ARTag. AprilTag introduce un método mejorado para generar cargas binarias útiles, garantizando una distancia mínima de Hamming entre las etiquetas bajo todas las rotaciones posibles, haciéndolas más robustas que los diseños anteriores. Además, AprilTag proporciona una implementación open source de los detectores, y sus algoritmos e implementación están bien documentados, lo que alentó su adopción por parte de la comunidad académica <sup>2</sup>.

Las características del marcador de referencia de AprilTag son muy similares a las de ARTag, un borde cuadrado de polaridad (blanco sobre negro o negro sobre blanco) y una cuadrícula cuadrada de 6 x 6 que divide el interior. La diferencia es que el borde en AprilTag tiene un grosor de 1 unidad, mientras que en ARTag es de 2 unidades. El marcador completo es de 8 x 8 unidades, con 36 celdas en el interior para transportar información, Figura 3.5.

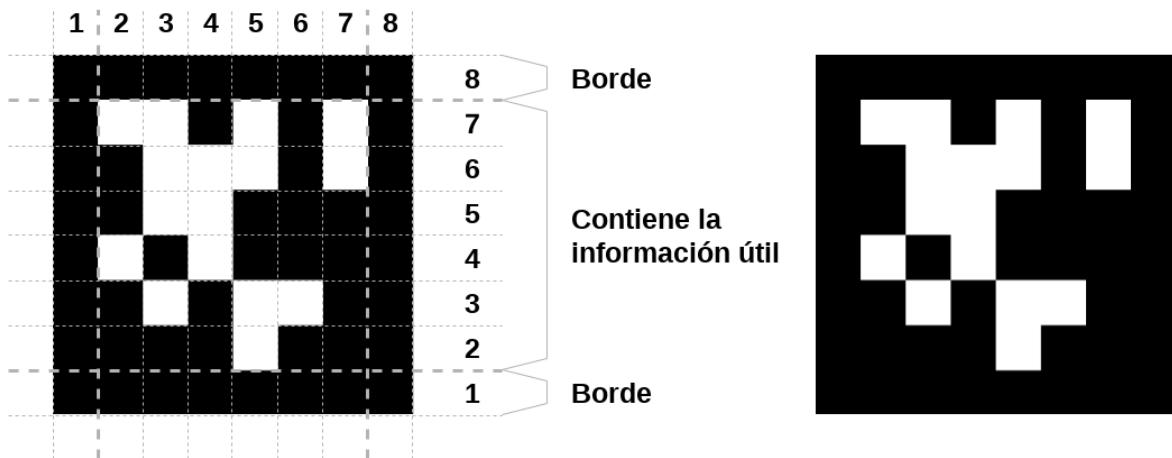


Figura 3.5: Ejemplo de marcador de referencia AprilTag

El sistema se compone de dos componentes principales: el detector de etiquetas y el sistema de codificación. En términos generales, el detector intenta encontrar regiones de cuatro lados (*quad*) que tienen un interior más oscuro que su exterior. Las etiquetas tienen bordes en blanco y negro para facilitar esto. El detector cuádruple está diseñado para tener una tasa de falsos negativos muy baja y, en consecuencia, tiene una tasa de falsos positivos alta. El sistema de codificación reduce esta tasa de falsos positivos a niveles útiles.

<sup>2</sup><https://april.eecs.umich.edu/software/apriltag.html>

## Detector

Se comienza detectando líneas en la imagen. Se calcula la dirección y la magnitud del gradiente en cada píxel y se agrupan de forma aglomerada los píxeles en componentes con direcciones y magnitudes de gradiente similares Figura 3.6 (a,b,c,d). El algoritmo de agrupamiento utilizado es similar al método basado en gráficos de Felzenszwalb [11], se crea un gráfico en el que cada nodo representa un píxel. Los bordes se agregan entre píxeles adyacentes con un peso de borde igual a la diferencia de los píxeles en la dirección del degradado. Estos bordes se clasifican y procesan en términos de peso de borde creciente. Este método de agrupamiento basado en gradientes es sensible al ruido en la imagen, incluso cantidades modestas de ruido harán que las direcciones locales de los gradientes varíen, inhibiendo el crecimiento de los componentes. La solución a este problema es filtrar con paso bajo la imagen [11, 22]. A diferencia de otros dominios donde este filtrado puede desenfocar información útil en la imagen, los bordes de una etiqueta son características intrínsecamente de gran escala por lo que este filtrado no causa pérdida de información. Una vez que se completa la operación de agrupamiento, los segmentos de línea se ajustan a cada componente conectado mediante un procedimiento tradicional de mínimos cuadrados, ponderando cada punto por su gradiente de magnitud. Se ajusta cada segmento de línea para que el lado oscuro de la línea esté a su izquierda, y el lado claro esté a su derecha.

La siguiente tarea es encontrar secuencias de segmentos de línea que forman un cuadrante Figura 3.6 (e). El enfoque se basa en una búsqueda recursiva de profundidad con una profundidad de cuatro, cada nivel del árbol de búsqueda agrega una ventaja al cuadrante. En la profundidad uno se consideran todos los segmentos de línea. En las profundidades dos a cuatro, se consideran todos los segmentos de línea que comienzan "lo suficientemente cerca" de donde terminó el segmento de línea anterior y que obedecen a un orden de enrollamiento en sentido contrario a las agujas del reloj. Una vez que se han encontrado cuatro líneas, se crea una detección de *quad* candidato. Las esquinas de este *quad* son las intersecciones de las líneas que lo componen.

Luego se calcula la matriz de homografía que proyecta puntos 2D en coordenadas homogéneas desde el sistema de coordenadas del marcador de referencia hacia el sistema de coordenadas de la imagen 2D. La homografía se calcula utilizando el algoritmo de transformación lineal directa (DLT) [16]. Para el cálculo de la posición y orientación del marcador de referencia se requiere información adicional, la distancia focal de la cámara y el tamaño físico del marcador. Se cuenta con dicha información al seleccionar la cámara y confeccionar las marcas de referencia.

La tarea final es leer los bits del campo de carga útil. Para esto se realiza el cálculo de las coordenadas relativas a la etiqueta de cada campo de bits, transformándolas en coordenadas de imagen utilizando la homografía y el umbral de los píxeles resultantes. Para ser robustos a la iluminación se utiliza un umbral que varía espacialmente. Se construyen dos modelos, uno que varía espacialmente de la intensidad de los píxeles "negros", y otro para la intensidad de los modelos "blancos". En la Figura 3.6 (f) se observa la detección de cuadrantes y toma de muestras. Las dos detecciones extrañas se descartan porque su carga útil no es válida. Los puntos blancos corresponden a las muestras alrededor del borde de las etiquetas que se utilizan para ajustar un modelo lineal de intensidad de píxeles "blancos"; un modelo es igualmente adecuado para los píxeles negros. Estos dos modelos se utilizan para umbralizar los bits de carga útil de datos, mostrados como puntos amarillos.

## Sistema de codificación

Una vez que la carga útil de datos se decodifica de un cuadrante, el trabajo del sistema de codificación es determinar si es válido o no. El sistema de codificación está basado en lexicodes modificados [35]. Los lexicodes clásicos están parametrizados por dos valores, el número de bits en cada palabra de código y la distancia mínima de Hamming entre cualquier par de palabras de código. El sistema de rotación debe ser robusto a la rotación (cuando el marcador se encuentre a 90, 180 o 270 grados). Algunas palabras de código, a pesar de satisfacer la restricción de la

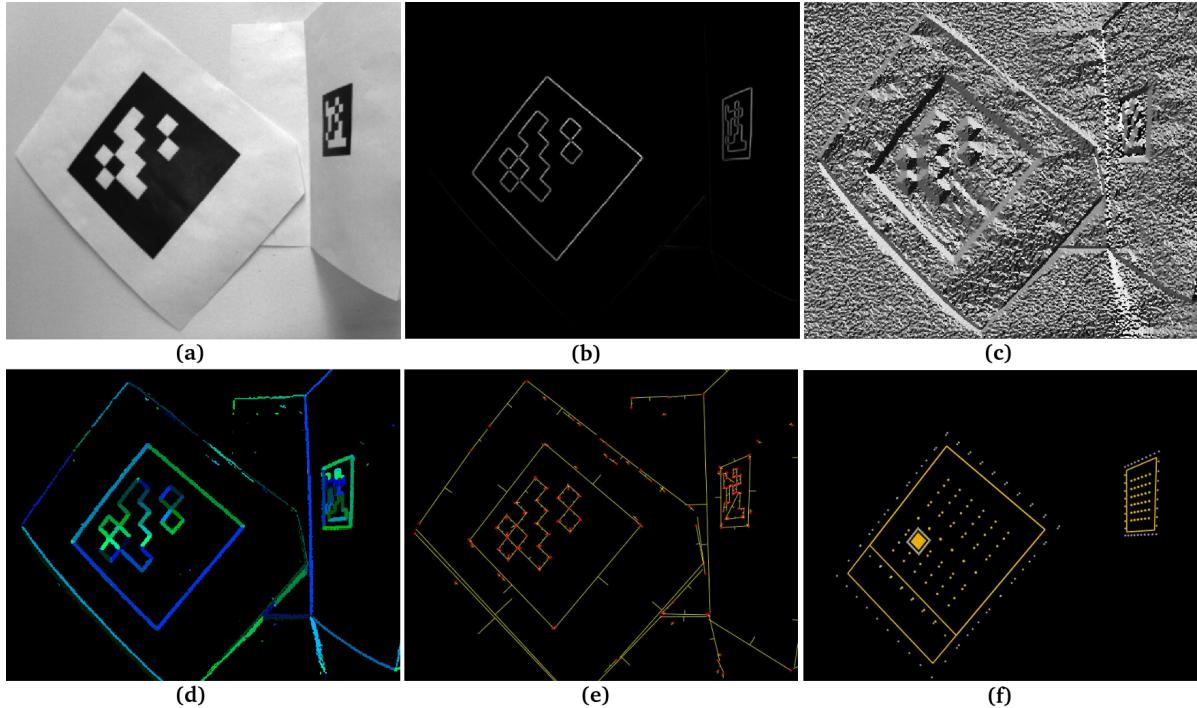


Figura 3.6: Proceso de detección de marcadores AprilTag. (a) Imagen de entrada. (b) Algoritmo de detección calcula la magnitud del gradiente de cada píxel. (c) Algoritmo de detección calcula la dirección del gradiente de cada píxel. (d) Píxeles con direcciones de gradiente y magnitudes similares agrupados. (e) Segmentos de linea ajustados a los píxeles de cada componente. (f) Cuadrantes detectados.

distancia mínima de Hamming no son buenas elecciones, como por ejemplo la palabra de código que consta de todos los ceros da como resultado una etiqueta que parece un solo cuadrado negro. Estos patrones geométricos simples ocurren comúnmente en escenas naturales y darían como resultado falsos positivos. Se modifica el algoritmo de generación de lexicodes para rechazar palabras de código candidatas que dan como resultado patrones geométricos simples. Estableciendo una distancia de mínima de Hamming igual a 10, se pueden codificar 2221 palabras distinguibles.

### 3.5. Conclusiones

ARToolkit es útil para muchas aplicaciones, pero al utilizar correlación para verificar e identificar marcadores, causa altas tasas falsos positivos y de confusión entre marcadores. A menudo puede detectar falsamente marcadores donde no los hay, y con frecuencia confundirlos. Otra desventaja de ARToolkit es que el esquema de detección se basa en una simple binarización de la imagen de entrada basada en un umbral especificado por el usuario. Si bien este esquema lo hace muy rápido, no es robusto a los cambios de iluminación. En general, las detecciones de ARToolkit no pueden manejar occlusiones parciales del borde del marcador.

ARTag proporciona esquemas mejorados de detección y codificación. El mecanismo de detección se basa en el gradiente de la imagen, lo que lo hace robusto a los cambios en la iluminación. Si bien los detalles del algoritmo del detector no son públicos, el mecanismo de detección de ARTag es capaz de detectar marcadores cuyo borde está parcialmente oculto. ARTag también proporcionó el primer sistema de codificación basado en la corrección de errores hacia adelante, lo que hace que los marcadores sean más fáciles de generar, más rápidos de correlacionar y proporciona una mayor ortogonalidad entre los marcadores. El rendimiento de ARTag inspiró

varias mejoras en ARToolkit, que posteriormente evolucionó a ARToolkitPlus [37]. Estas versiones introdujeron cargas útiles codificadas digitalmente como las utilizadas en ARTag. ARTag puede codificar hasta 2002 identificadores únicos diferentes sin necesidad de almacenar los patrones dentro del sistema. El inconveniente con ARTag es que se eliminó del sitio de descarga del NRC (National Research Council Canada), ya que el contrato con el investigador principal (Mark Fiala) expiró. La gerencia del NRC ha detenido la investigación de realidad aumentada y visión por computadora [2].

AprilTag está basado en ARToolkit y ARTag, presenta un bajo porcentaje de falsas detecciones, aún con cambios en la iluminación del ambiente. El sistema de codificación de AprilTag logra una mayor distancia mínima de Hamming entre todos los pares de palabras de código mientras codifica un numero mayor de identificadores que ARTag. AprilTag puede codificar hasta 2221 identificadores únicos diferentes sin necesidad de almacenar los patrones dentro del sistema. Al igual que ARToolkit, AprilTag también evolucionó. En AprilTag 2 [38] se mejora el detector logrando un algoritmo de decodificación mas rápido y una tasa de falsos positivos mas baja. Estas mejoras se lograron gracias a la retroalimentación de los usuario sobre el uso de casos comunes, por ejemplo, en la mayoría de las implementaciones la detección de etiquetas parcialmente ocluidas no es de gran utilidad (la mayoría de los usuarios deshabilitan esta función). Actualmente, se encuentra en el repositorio oficial la versión mas reciente de AprilTag, AprilTag 3 [3], que incluye un detector aún más rápido, una tasa de detección mejorada en etiquetas pequeñas y diseños de etiquetas flexibles. Además, consta de una biblioteca en lenguaje C con dependencias mínimas.

En base al análisis anterior se seleccionó el sistema fiducial AprilTag para ser utilizado en el diseño e implementación del prototipo desarrollado en este trabajo, principalmente porque existe gran documentación de sus algoritmos en la literatura académica, y los autores originales han publicado la implementación de los algoritmos como bibliotecas con licencia de código abierto (open source), de fácil reuso.



## Capítulo 4

# Arquitectura del prototipo propuesto

En el presente capítulo, se enumeran los componentes de hardware y la disposición de los mismos para la fabricación del prototipo propuesto. Posteriormente, se presenta la Arquitectura de Software, se explica la función de cada módulo de software y cómo estos interactúan entre sí. Finalmente, se describen las etapas del proceso de localización del prototipo, lo cual permite al usuario del robot educativo conocer la ubicación del mismo en un escenario real pre-establecido, y así poder tomar una decisión, por ejemplo, al navegar por dicho escenario.

### 4.1. Componentes de Hardware

Los parámetros priorizados para la selección de los componentes de hardware fueron la disponibilidad y el bajo costo, teniendo en cuenta que el valor del prototipo implementado no supere el valor del robot Frankestito-N6. A continuación se enumeran dichos componentes.

- *Computadora MIPS*: Se seleccionó el router TP-Link MR3020. Este presenta características similares al utilizado por los frankestitos (TP-Link MR3040), pero con la ventaja de poseer menores dimensiones físicas. Ver Figura 4.1.
- *Cámara USB*: Cámara genérica para tomar fotos del ambiente y buscar puntos de referencias. Ver Figura 4.2.
- *Motor Paso a Paso con Placa controladora*: Necesario para ubicar la cámara en distintas posiciones. Ver Figura 4.3.
- *Sensor óptico de fin de carrera*: Necesario para calibrar la posición inicial del motor, ángulo grado 0. Ver Figura 4.4.
- *Microcontrolador con chip ATMEGA328*: Microcontrolador encargado de controlar los movimientos de la cámara y mantener la comunicación con la computadora MIPS. Ver Figura 4.5.

### 4.2. Arquitectura de Hardware

La arquitectura de hardware del prototipo presenta dos componentes principales:

- El hardware de procesamiento embebido.
- El dispositivo de cámara rotativa;

#### 4.2.1. Hardware de procesamiento embebido

El Hardware de procesamiento embebido es la computadora MIPS donde se ejecuta el sistema de localización. Este contiene un System On Chip (SOC) Atheros AR9331, con una CPU MIPS de 400Mhz. La memoria principal (RAM) es de 32MB y la memoria Flash es de 4MB. También cuenta con puertos de E/S (GPIO), un puerto USB 2.0, una interfaz serial (UART) y una interfaz Wireless WiFi 802.11n/g/b (Figura 4.1).

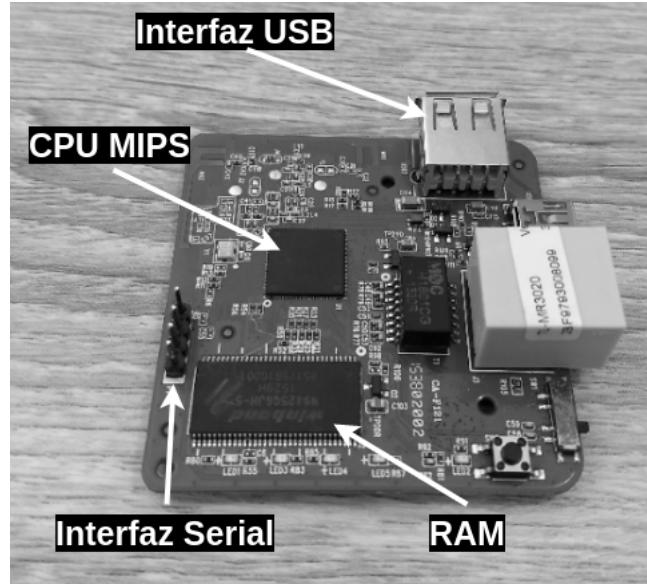


Figura 4.1: Router TP-Link MR3020

#### 4.2.2. Dispositivo de cámara rotativa

El dispositivo de cámara rotativa consta de una cámara USB genérica, montada sobre el motor paso a paso, el cual permite rotar a la cámara de 0 a 360°, con una resolución de giro de un grado. Los movimientos del motor paso a paso son controlados por el microcontrolador ATMEGA328 a través de puertos de E/S (GPIO). Se utilizan los pines 8,9,10 y 11 del microcontrolador (Figura 4.5) para la conexión con la placa controladora del motor. La cámara es un dispositivo de clase UVC (USB Video Class), y se conecta a la computadora MIPS a través la interfaz USB. La computadora MIPS se conecta al microcontrolador a través de la interfaz serial (UART). Se utilizan los pines TX y RX del microcontrolador (Figura 4.5). Además, el microcontrolador recibe la alimentación desde la computadora MIPS (5 VDC), utilizando los pines VCC y GND. El sensor óptico de fin de carrera se conecta un puerto de E/S del microcontrolador (pin 12).



Figura 4.2: Camara USB genérica



Figura 4.3: Motor paso a paso con placa controladora

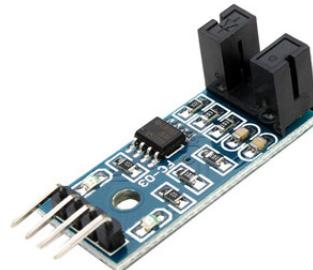


Figura 4.4: Sensor óptico de fin de carrera

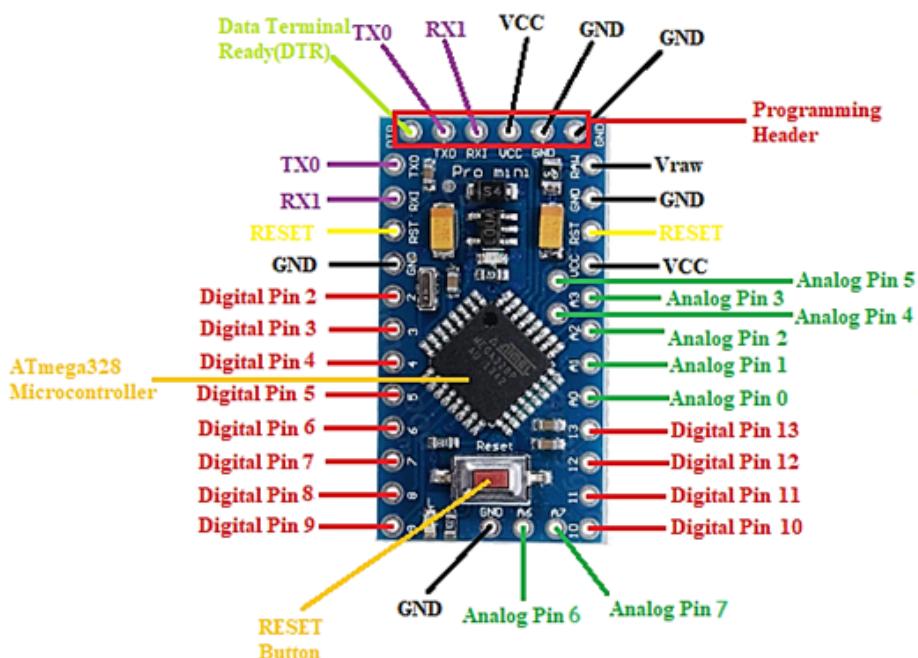


Figura 4.5: Mircrocontrolador con chip ATMEGA328

En la Figura 4.6 se puede observar el diseño del dispositivo de cámara rotativa. La cámara se encuentra montada sobre el motor paso a paso a través de un soporte plástico diseñado a medida. El soporte tiene una pequeña perforación que deja pasar la luz emitida por el led del sensor óptico fin de carrera, esto es utilizado para calibrar la posición inicial de la cámara.

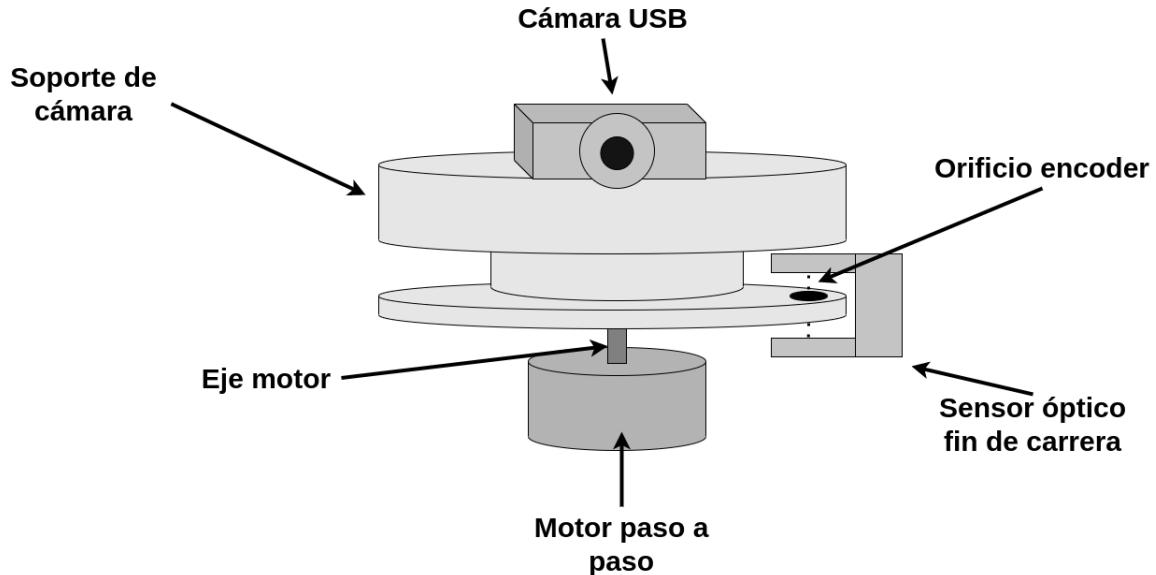


Figura 4.6: Diseño del dispositivo de cámara rotativa.

#### 4.2.3. Conexión entre componentes

Las conexiones físicas entre los distintos componentes de hardware pueden observarse en la Figura 4.7. La comunicación entre el microcontrolador y la computadora MIPS es a través de la interfaz serial. La cámara se conecta al puerto USB de la computadora MIPS. El sensor óptico fin de carrera se conecta al pin 12 del microcontrolador y la placa controladora (driver) del motor paso a paso a los pines 8,9,10 y 11 del microcontrolador.

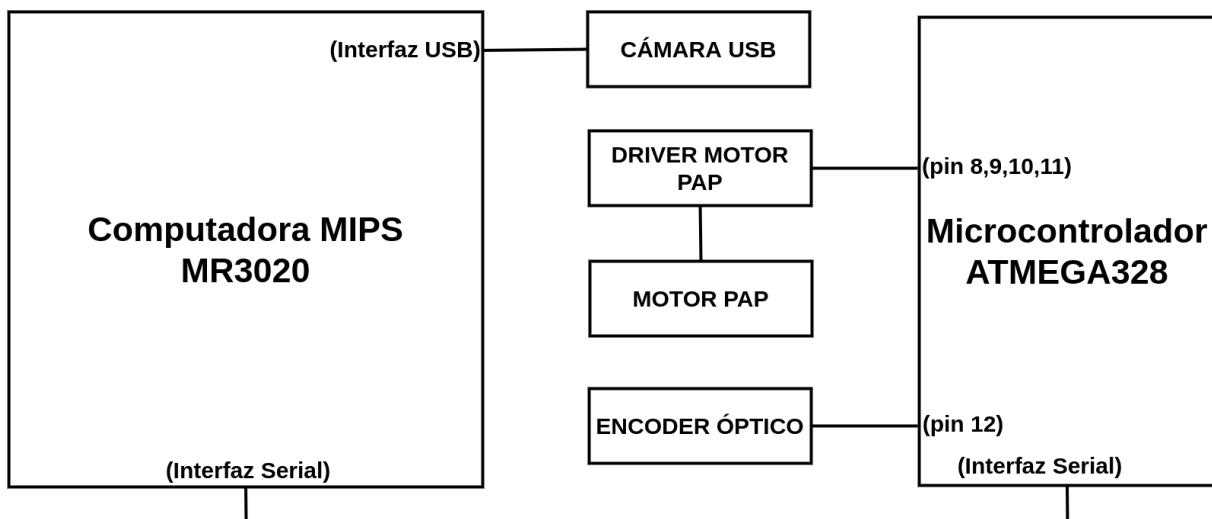


Figura 4.7: Diagrama de conexiones físicas entre los distintos componentes de hardware

#### 4.2.4. Diseño del gabinete

Se utilizó la aplicación libre OpenScad<sup>1</sup> para el diseño 3D del gabinete y del soporte cámara-motor (Figura 4.8). Se puede observar el diseño terminado tanto del gabinete (Figura 4.9 (a,b)) como del soporte de la cámara (Figura 4.9 (c)). El tamaño total del prototipo construido es de aproximadamente 7 cm de ancho x 9 cm de largo x 9 cm de alto, y tiene un peso total aproximado de 175g. El prototipo terminado y ensamblado puede observarse en la Figura 4.10.

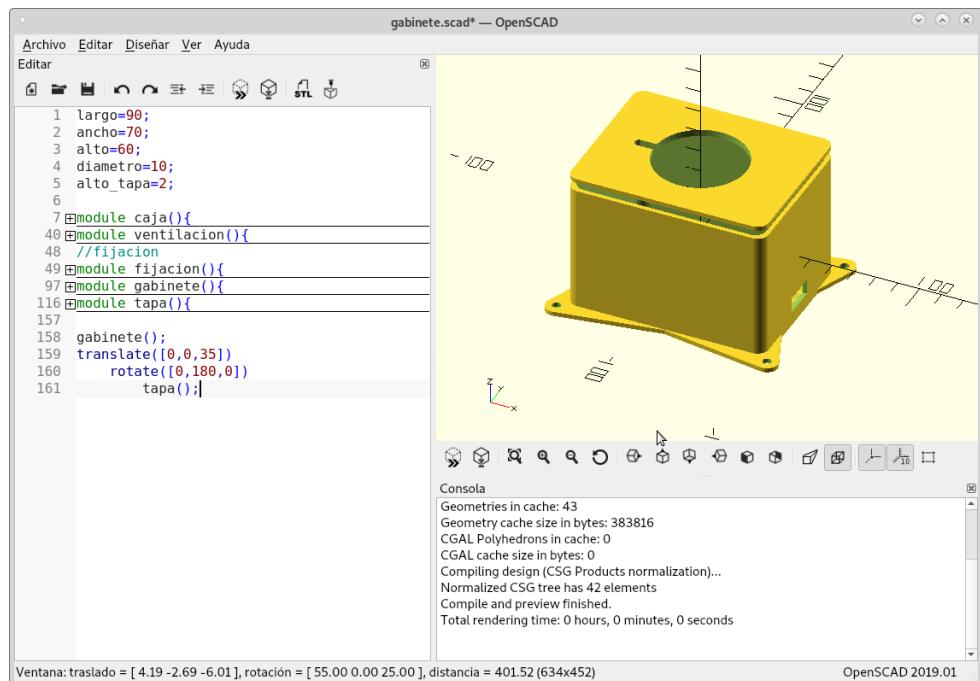


Figura 4.8: Software OpenScad para diseño de modelos 3D

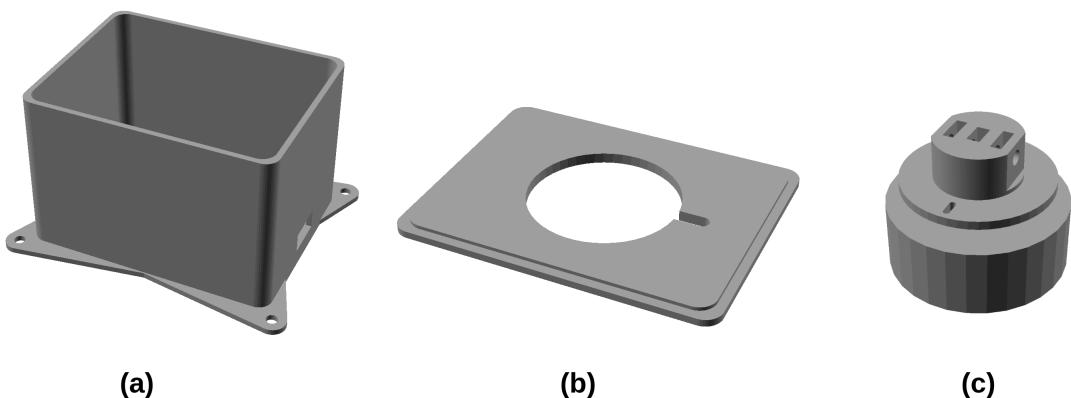


Figura 4.9: Diseño 3D de gabinete soporte cámara-motor

<sup>1</sup><https://openscad.org/>

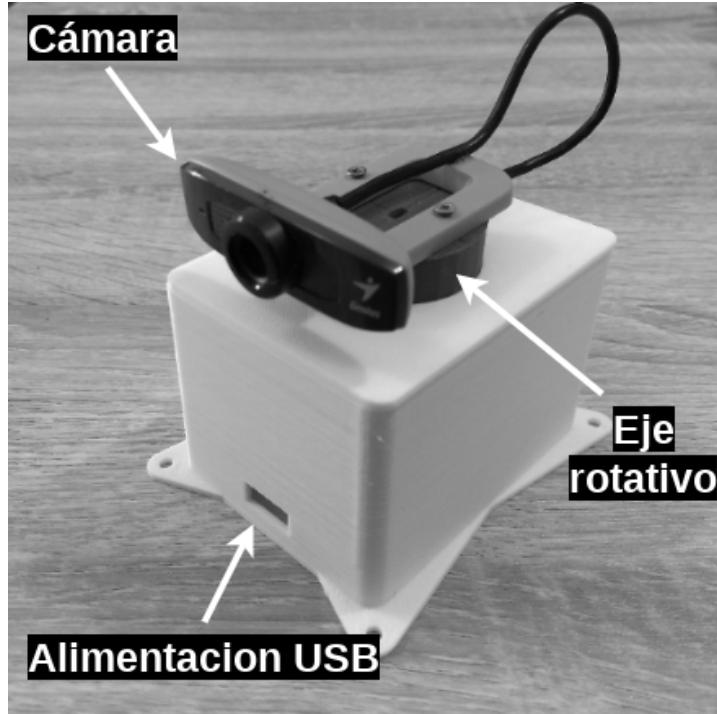


Figura 4.10: Prototipo terminado y ensamblado

### 4.3. Arquitectura de Software

La arquitectura de software del sistema de localización se puede separar en 2 módulos principales (Figura 4.11). Un módulo controla el dispositivo de cámara rotativa (Firmware Embebido) y el otro controla el hardware de procesamiento embebido (Sistema Linux), que es donde se ejecuta la aplicación de localización.

#### 4.3.1. Firmware Embebido

El software embebido en el Firmware contiene un pequeño gestor de tareas (OS embebido) diseñado para microcontroladores. Este se ejecuta en el microcontrolador ATMEGA328 (AVR) y maneja 2 tareas independientes.

Una tarea controla el envío y recepción de mensajes a través de la interfaz UART, utilizando el *driver serial*. Esta interfaz es utilizada para la comunicación entre ambos módulos de software (el del firmware embebido y el sistema Linux). La segunda tarea controla la *lógica general del servo*, es decir, se encarga de mover el motor para posicionar la cámara en el ángulo requerido. El motor paso a paso no tiene posiciones absolutas, por lo tanto es necesario implementar un sistema para poder controlar las posiciones del eje del motor con un mecanismo externo. Para esto, se utiliza un fin de carrera, el cual se encarga de informar al software cuando el motor alcanzó la vuelta completa. La cámara está montada sobre una estructura que va ajustada sobre el eje del motor. Dicha estructura tiene un disco con una perforación, la cual se utiliza para definir la posición inicial mediante el sensor óptico del fin de carrera. Entonces, cuando el sistema se inicia, el motor gira en sentido anti-horario hasta llegar al fin de carrera, y esta posición será establecida como la inicial, ángulo 0°.

En el *driver fin de carrera* se inicializa un puerto de entrada (GPIO) del microcontrolador AVR, para obtener las lecturas del sensor óptico. El uso de este sensor es necesario para mover el eje motor a la posición inicial (ángulo 0°).

En el *driver del motor paso a paso* se implementaron las siguientes funciones para controlar

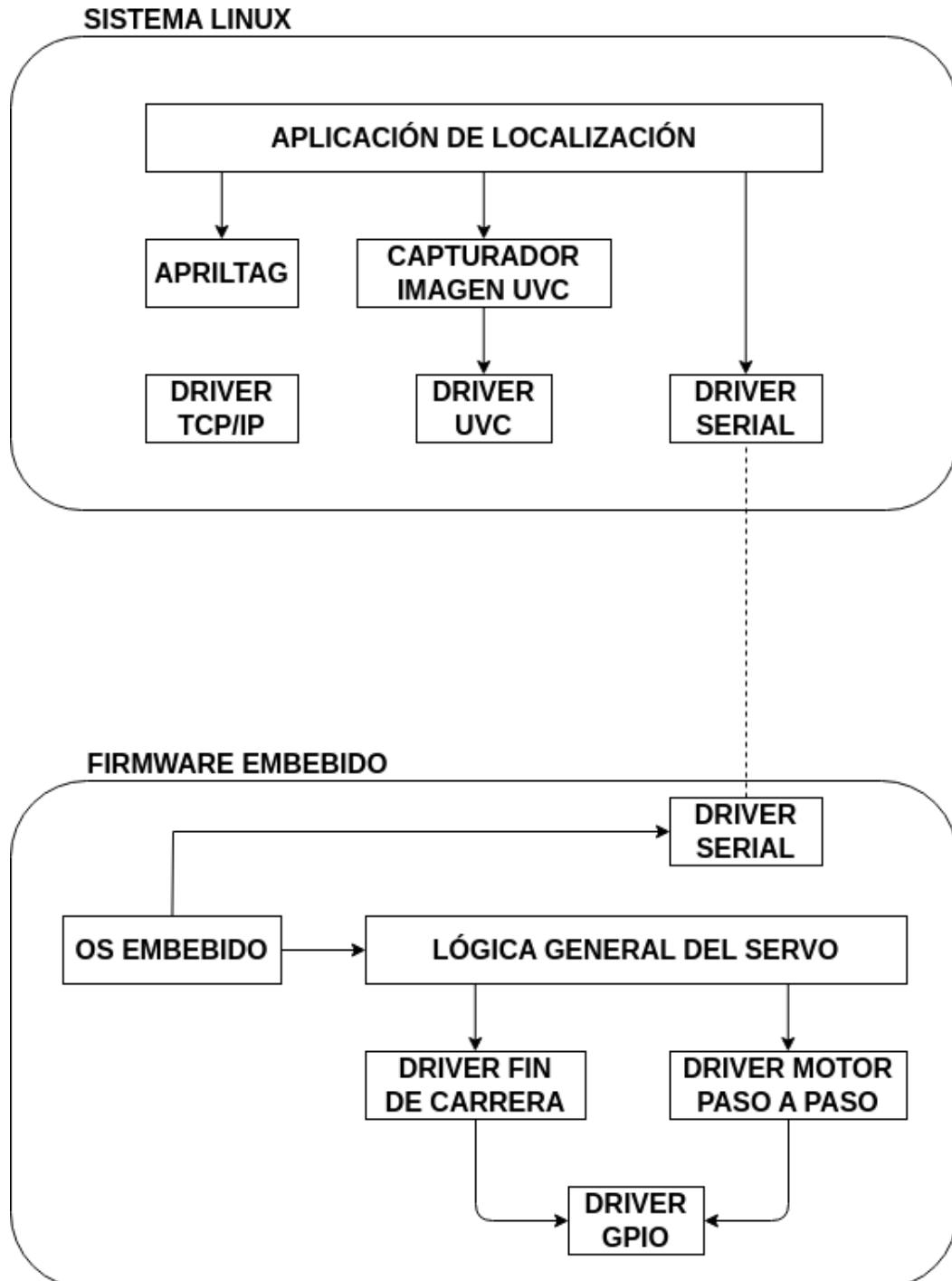


Figura 4.11: Diagrama de la Arquitectura de Software

los movimientos del mismo:

- *inicializar()*: Inicializa los puertos de salida (GPIO) del microcontrolador que controla los movimientos del motor paso a paso.
- *avanzar\_un\_paso()*: Mueve el eje del motor un paso en sentido horario.
- *avanzar\_n\_pasos(N)*: Mueve el eje del motor N pasos en sentido horario.
- *retroceder\_un\_paso()*: Mueve el eje del motor un paso en sentido antihorario.

- *retroceder\_n\_pasos(N)*: Mueve el eje del motor N pasos en sentido antihorario.

La *lógica general del servo* tiene la función de controlar los movimientos del motor para posicionar la cámara en un ángulo específico, haciendo uso de los drivers mencionados anteriormente. A continuación se muestra la sección del código fuente correspondiente a la rutina de inicialización, la cual mueve el motor en sentido antihorario hasta llegar al fin de carrera, para finalmente establecer esa posición como posición inicial, ángulo 0 grados.

```

1 #include <avr/io.h>
2 #include <util/delay.h>
3
4
5 #define PIN_8 0b00000001 // PB0 -> pin 8 avr
6 #define PIN_9 0b00000010 // PB1 -> pin 9 avr
7 #define PIN_10 0b00000100 // PB2 -> pin 10 avr
8 #define PIN_11 0b00001000 // PB3 -> pin 11 avr
9 #define TOPE 0b00010000 // PB4 -> pin 12 avr
10 #define LED 0b00100000 // PB5 -> pin 13 avr
11
12 volatile unsigned char * puerto_b = (unsigned char *) 0x25;
13 volatile unsigned char * ddr_b = (unsigned char *) 0x24;
14 volatile unsigned char * pin_b = (unsigned char *) 0x23;
15
16 int paso_actual, grado_actual;
17
18 void inicializar(){
19     volatile unsigned char entrada; /* sensor de fin de carrera */
20     *puerto_b = 0;
21     *puerto_b = *puerto_b | TOPE; /* pin 12 se usa como entrada */
22     *(ddr_b) = *(ddr_b) | 0b00101111;
23
24     entrada = *pin_b & TOPE;
25     blink_led(1); /* enciende el led hasta que el motor se encuentre en la
26     posicion inicial */
27     while (entrada == TOPE){
28         retroceder_un_paso(); /* mueve el eje del motor un paso en sentido anti-
29         horario */
30         entrada = *pin_b & TOPE;
31     };
32     paso_actual=0;
33     grado_actual=0;
34 }
```

La cantidad de pasos necesarios para que un motor pase a paso haga un giro completo es un dato proporcionado por el fabricante, es decir, es un valor conocido. Al dividir este valor por 360 (grados), se obtiene el numero de pasos necesarios que debe realizar el motor para mover el eje 1 grado. En todo momento se conoce la posición en grados del eje del motor.

El *driver serial* implementado en el *Firmware Embebido* permite la comunicación con el módulo *Sistema Linux*. Las peticiones provienen siempre del *Sistema Linux* (detallado en la siguiente sección) hacia el *Firmware Embebido*. Estas pueden ser: mover el motor a un ángulo específico, detener el motor, y obtener la posición actual del motor (grados). El Firmware Embebido se encarga de procesar esos mensajes y actuar en consecuencia.

#### 4.3.2. Sistema Linux

El *hardware de procesamiento embebido (computadora MIPS)* está controlado por un sistema operativo Linux construído con buildroot, destinado a sistemas embebidos. Los controladores de hardware que se utilizan en este dispositivo son: el driver UVC (Universal Video Class), para

la captura de imágenes desde la cámara; el driver serial (UART), para la comunicación con el dispositivo de cámara rotativa; y el driver TCP/IP, para la conexión remota vía WiFi con un sistema remoto (por ejemplo, con una notebook o laptop de desarrollo). Sobre este sistema Linux se ejecuta la aplicación principal (aplicación de localización) desarrollada en este trabajo, la cual consiste de cuatro etapas, como se observa en la Figura 4.12.

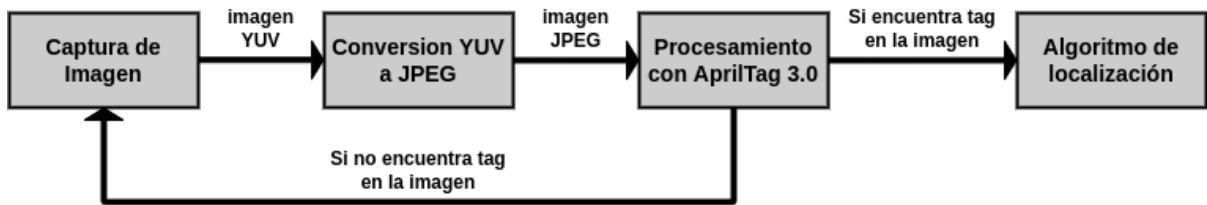


Figura 4.12: Etapas del proceso de localización.

1. En la primera etapa del proceso de localización la aplicación embebida captura una imagen desde la cámara, utilizando la interfaz en espacio de usuario del driver UVC del sistema Linux.
2. En la segunda etapa se convierte la imagen obtenida de formato YUYV al formato JPEG, que es el formato esperado por AprilTag. La imagen tiene una resolución de 320x240 píxeles.
3. En la tercera etapa el sistema procesa la señal obtenida (imagen JPEG) con el software AprilTag. En esta etapa AprilTag analiza la imagen y pueden suceder dos casos:
  - a) La marca de referencia no fué detectada en la imagen. En este caso la aplicación de localización envía un mensaje a través del driver serial al dispositivo de cámara rotativa para que la cámara gire 30° en sentido horario. Una vez recibido el mensaje de confirmación por parte del dispositivo de cámara rotativa, se retorna nuevamente a la primer etapa.
  - b) La marca de referencia fué detectada en la imagen. En este caso AprilTag reporta la posición y orientación de las etiquetas identificadas. Como en este trabajo se experimenta con una única marca de referencia en el ambiente, se obtiene como salida una única matriz de rotación y un vector de traslación de la marca detectada, ubicados en el sistema de coordenadas de la cámara, con origen el centro de la lente focal. Se prosigue con la siguiente etapa.
4. En la última etapa el Algoritmo de Localización realiza el cálculo de la posición de la cámara en el sistema de coordenadas del mundo real (sistema de coordenadas de la marca de referencia o TAG) y consta de tres sub-etapas:
  - a) Se obtiene el ángulo euler pitch a partir de la matriz de rotación obtenida desde AprilTag. Para su cálculo se utilizan funciones de la biblioteca eigen, las cuales se incorporaron como parte del código de la aplicación. En la Figura 4.13 (a) (1), se puede observar que el ángulo euler pitch obtenido ( $\alpha$ ), es el ángulo formado por el eje  $X_{cam}$  del sistema de coordenadas de la cámara y el eje  $X_{tag}$  del sistema de coordenadas del tag.
  - b) Luego, a partir del punto  $(X_c, Y_c)$  provisto por AprilTag, Figura 4.13 (a), se obtiene el punto  $X'_c$  e  $Y'_c$  en el sistema de coordenadas del mundo real, Figura 4.13 (b) (2), donde  $X'_c = X_c$  y  $Y'_c = Y_c$  (sus valores algebraicos son equivalentes).

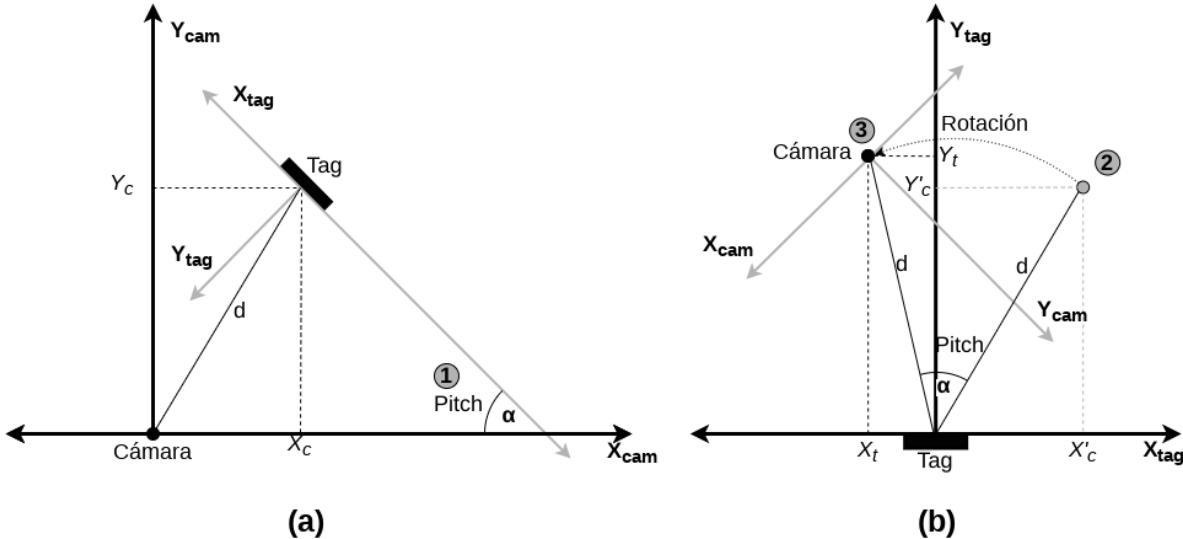


Figura 4.13: Transformacion del sistema de coordenadas de la cámara al sistema de coordenadas del TAG. (a) posición de la marca de referencia en el sistema de coordenadas de la cámara. (b) posición de la cámara en el sistema de coordenadas de la marca de referencia.

- c) Finalmente, se realiza una rotación del punto  $(X'_c, Y'_c)$ , la cantidad de grados  $\alpha$  obtenido en (a). Para esto se define un vector columna a partir de  $X'_c$  y  $Y'_c$ , y se multiplica por una matriz de rotación calculada a partir del ángulo  $\alpha$ , como se observa en la siguiente ecuación.

$$\begin{bmatrix} X_t \\ Y_t \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} X'_c \\ Y'_c \end{bmatrix}$$

Luego se obtiene el punto  $(X_t, Y_t)$  a partir de la siguiente ecuación:

$$\begin{aligned} X_t &= \cos(\alpha) \cdot X'_c - \sin(\alpha) \cdot Y'_c \\ Y_t &= \sin(\alpha) \cdot X'_c + \cos(\alpha) \cdot Y'_c \end{aligned}$$

El punto  $(X_t, Y_t)$ , Figura 4.13 (b) (3), indica la posición de la cámara en el sistema de coordenadas de la marca de referencia, es decir, la posición del dispositivo portátil en el escenario o mundo real.

#### 4.4. Conclusiones

En este capítulo se presentó la arquitectura de hardware y software de un sistema de localización diseñado para ser utilizado sobre los robots educativos Frankestito-N6. El software principal de localización utiliza las bibliotecas de AprilTag. AprilTag no está diseñado para funcionar nativamente en la arquitectura MIPS, por lo que se debieron realizar algunas adaptaciones. El hardware de la computadora MIPS es limitado, por ejemplo, cuenta con sólo 4MB de memoria flash para almacenar el sistema operativo junto con la aplicación. En etapas tempranas de desarrollo, al compilar el software AprilTag 3 desde el repositorio oficial [3], se generó un binario ejecutable de 4.7MB (que no cabe en la memoria flash mencionada). Por tal motivo, una de las modificaciones que se realizó fue obtener la foto directamente desde el controlador de la cámara de Linux (`/dev/video0`) y luego convertirla en RAM a formato JPEG para que pueda ser procesada por AprilTag (sin almacenar la imagen). Además, se eliminaron dependencias externas

innecesarias para nuestro objetivo, por ejemplo OpenCV, y se limitó la localización a un ambiente 2D. El código fuente fue escrito casi en su totalidad en lenguaje C y se crearon nuevos *Makefiles* para la compilación de todo el desarrollo para la arquitectura MIPS [43].

A partir de lo anterior se logró diseñar e implementar un prototipo funcional, se integraron todos los componentes de hardware y software, y se logró un programa ejecutable que ocupa 0.8MB de la memoria flash.

En el siguiente capítulo se detallan los experimentos realizados con los sistemas de localización, para evaluar la precisión del mismo contra un sistema de localización validado. También se evalúa mediante experimentos el tiempo de ejecución de todo el sistema para conocer la frecuencia posible de uso.



# Capítulo 5

# Experimentos y Resultados

Para validar el prototipo se evaluó el tiempo de ejecución, la precisión, la exactitud y el error de las localizaciones reportadas por el sistema propuesto. La precisión se refiere a la repetibilidad de una medición usando un instrumento dado y la exactitud se refiere a cuán cerca está una medición de su valor verdadero.

## 5.1. Tiempo de ejecución

En la etapa de obtención de requerimientos de este trabajo se tomó como referencia los robots móviles de bajo costo utilizados por la Facultad de Informática en proyectos de robótica educativa [42]. El robot Frankestito-N6 se desplaza a una velocidad máxima de 12cm/seg, y se estimó que, para futuras funcionalidades de navegación, el robot educativo debería indicar su ubicación aproximadamente cada 20 centímetros. Esto requiere una localización cada 1,6 segundos en este robot, por lo que una frecuencia de 1Hz sería suficiente si el error de la localización es +/- 12cm.

Se utilizó la función de la biblioteca *libc gettimeofday()* para evaluar el tiempo transcurrido de cada etapa del sistema propuesto (tiempo de E/S y tiempo de ejecución) y corroborar que el tiempo de localización está dentro de los requerimientos. En una prueba de 415 mediciones utilizando el hardware propuesto se obtuvo (como se observa en la Tabla 5.1) una frecuencia general de localizaciones de 2Hz, y una frecuencia para el caso más desfavorable de 1Hz.

Etapa	Tipo	Tiempo Medio (media aritmética)	Tiempo Max. (peor caso)
Captura	Tiempo de E/S	74 ms	95 ms
Detección con AprilTag	Tiempo de ejecución	292 ms	750 ms
Algoritmo de Localización	Tiempo de ejecución	41 ms	46 ms
TOTAL	Tiempo de E/S + CPU	407 ms	891 ms

Tabla 5.1: Tiempos promedios de E/S y CPU en el dispositivo prototipo en 415 localizaciones.

## 5.2. Precisión

### 5.2.1. Primer Experimento

Para las pruebas se delimitó un escenario real de navegación de 2mts x 2mts, con una cuadrícula en el terreno compuesta de mosaicos de 33cm x 33cm. Se colocó una marca de referencia (TAG en la terminología de AprilTag) en este ambiente. El dispositivo prototipo fue entonces

montado sobre un robot móvil de bajo costo, y se calibró la altura de la lente de la cámara para que se encuentre a la misma altura que el centro del TAG, como se observa en la Figura 5.1. Para la comunicación entre la CPU del prototipo y el microcontrolador del robot móvil se interconectó la interfaz serial UART de ambos.

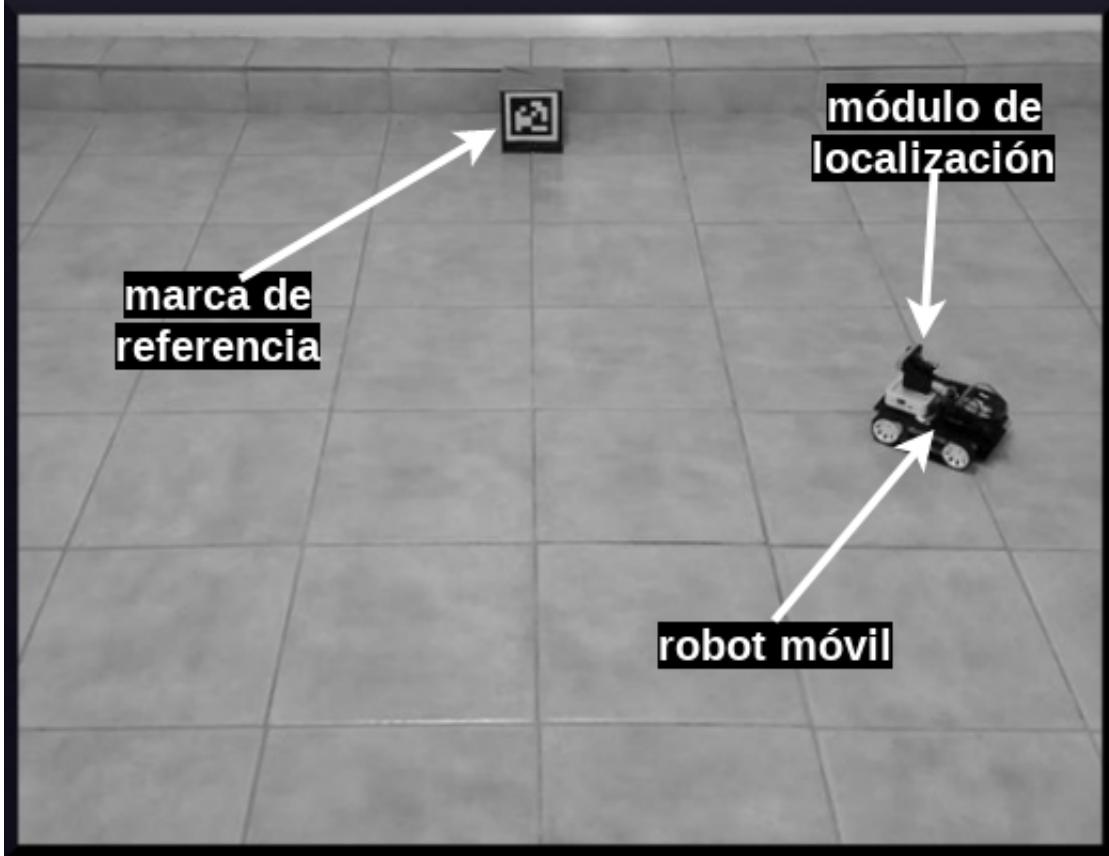


Figura 5.1: Escenario real donde se realizaron las mediciones del primer experimento

Durante el experimento se utilizó una marca de referencia de 20cm x 20cm, y se fue situando al robot móvil en cada esquina de la cuadrícula del ambiente, cuyas localizaciones exactas son conocidas a priori. En cada ubicación se realizaron aproximadamente 10 mediciones desde el dispositivo prototipo y se almacenaron los resultados para su análisis. El conjunto de datos fue representado por un gráfico de dispersión (Figura 5.2). La marca de referencia se ubicó en la posición X=99 e Y=0 de esta representación. Cada punto representa una localización reportada por el prototipo.

Como se observa la Figura 5.2, hasta una distancia en el eje Y de 125cm el robot reporta localizaciones muy cercanas al valor real de la ubicación. El valor exacto en cada medición (la ubicación real) está representada en la figura por cada intersección de la cuadrícula. Hasta ese rango, las mediciones son también muy precisas (las 10 mediciones en cada ubicación están concentradas). Luego, al superar en el eje Y la distancia de 175cm la precisión decae en el eje X mucho más que en el eje Y. Esto puede notarse en la Figura 5.2 ya que, los puntos de cada medición tienen una dispersión mayormente horizontal. En esta zona del escenario la distancia máxima (diferencia) entre mediciones de una misma ubicación es de hasta aproximadamente 20cm en el eje X, y de 10cm en el eje Y. Finalmente, a 2 m de distancia, las mediciones ya no se concentran alrededor de la ubicación real, por lo que estas mediciones sólo podrían utilizarse como un cálculo estimativo de la zona donde se encuentre el robot, y no como una localización exacta.

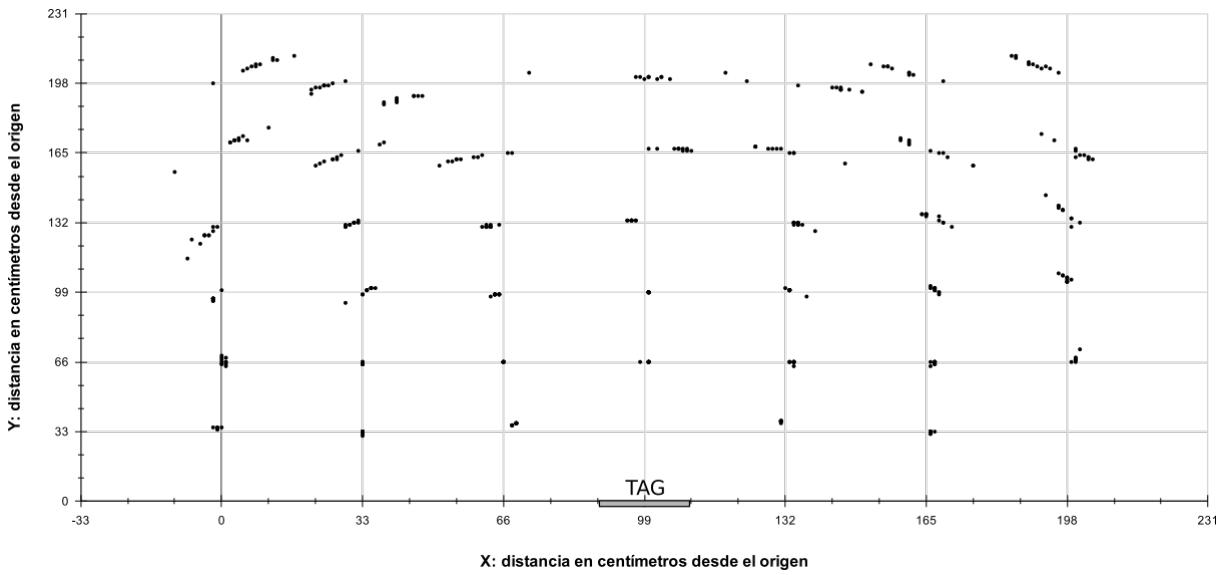


Figura 5.2: Gráfico de dispersión de las localizaciones reportadas por el dispositivo prototipo en cada intersección de la cuadrícula

En el gráfico de dispersión de la Figura 5.3 se promediaron las mediciones de las localizaciones en cada intersección de la cuadricula. En esta figura se puede apreciar de mejor manera que a partir de los 125cm en el eje Y, se comienza a perder exactitud en las mediciones.

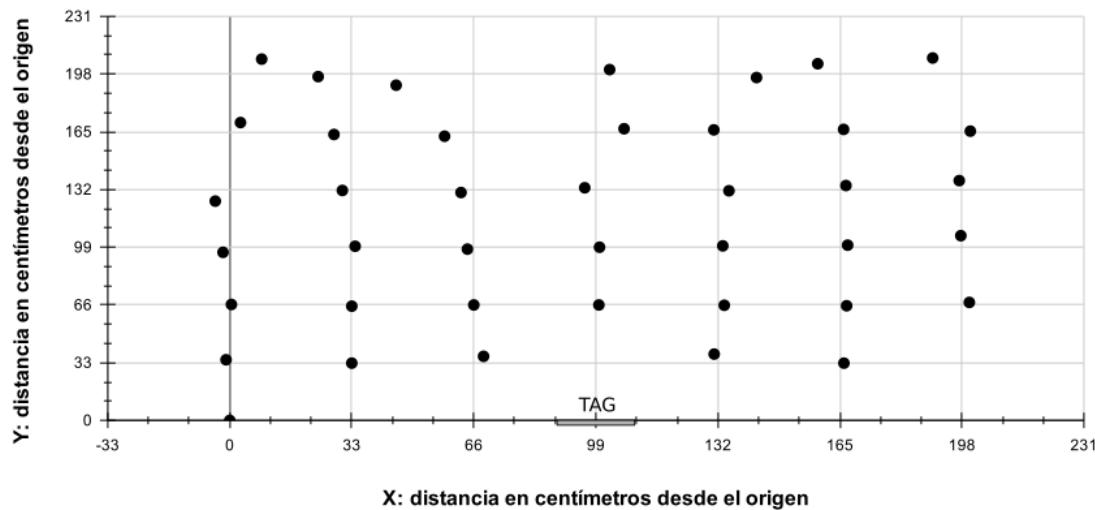


Figura 5.3: Gráfico de dispersión de localizaciones promediadas reportadas por el dispositivo prototipo en cada intersección de la cuadrícula

Cada localización es una ubicación en un plano 2D con ejes ( $X, Y$ ). Los histogramas de las distancias de las localizaciones con respecto al valor de la ubicación real para los ejes  $X$  e  $Y$  se pueden observar en la Figura 5.4 y en la Figura 5.5, respectivamente.

El valor  $X = 0$  e  $Y = 0$ , son la distancia mínima con respecto valor estimativo real (ubicaciones conocidas previo al experimento), y representan las mediciones más exactas. Se observa que al menos unas 100 mediciones coincidieron con la ubicación real (de un total de 415), y también, que un gran número de mediciones están comprendidas en el rango  $[-5, 5]$  cm. Estos datos se

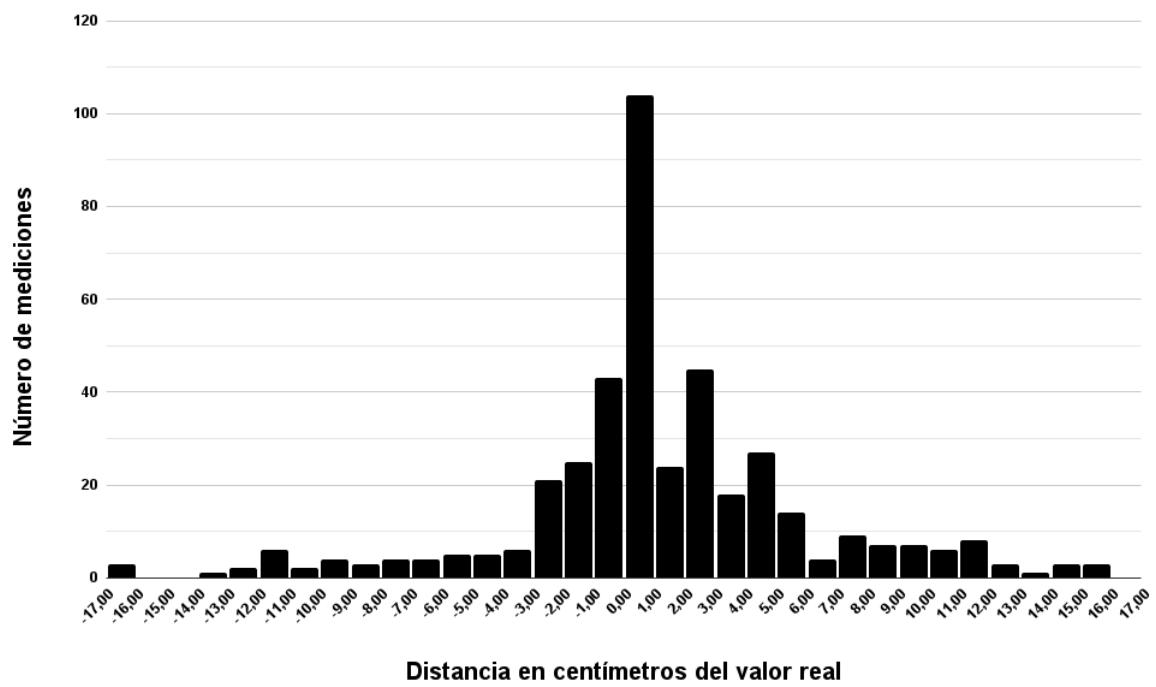


Figura 5.4: Histograma en X de las distancias en centímetros de las localizaciones con respecto al valor real

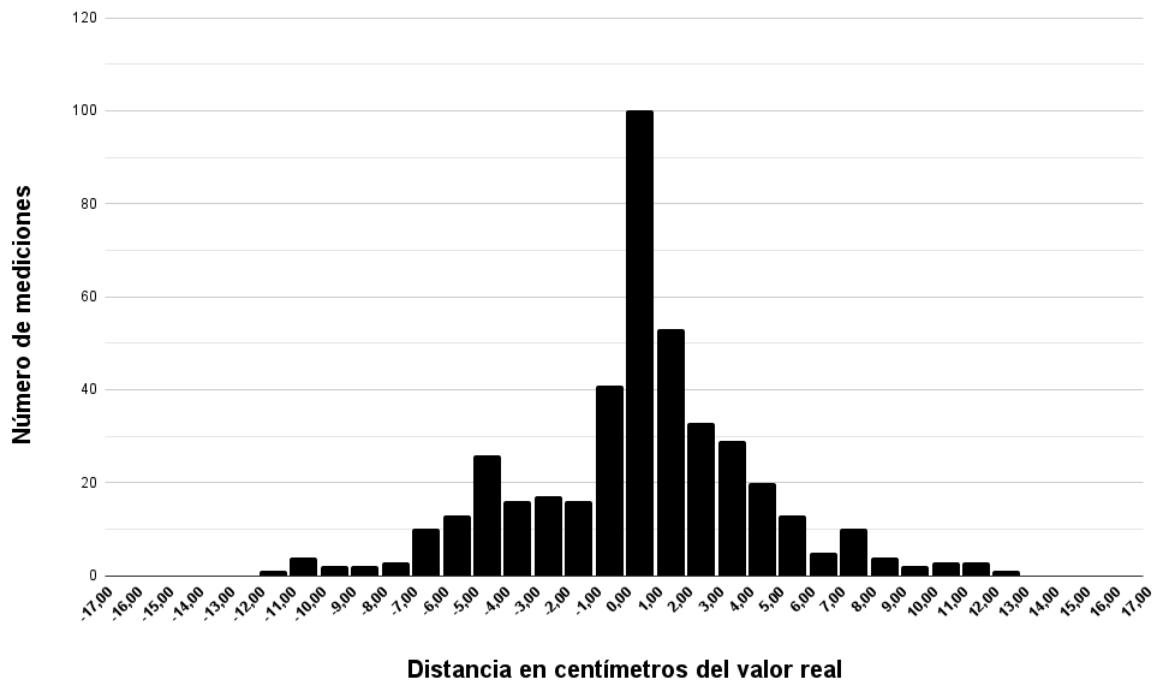


Figura 5.5: Histograma en Y de las distancias en centímetros de las localizaciones con respecto al valor real

pueden observar en las estadísticas de la Tabla 5.2.

Estadística	Valor
Cantidad de mediciones	415
Media del error	4,971cm
Varianza	28,985
Desvío estándar	0,264
Error estándar del promedio	0,118
Resultado de las mediciones	(4,971 +/- 0,118) cm
Porcentaje de mediciones dentro del desvío estándar	65 %

Tabla 5.2: Estadística de las distancias de localizaciones con respecto a su valor estimativo real

### 5.2.2. Segundo Experimento

Para este experimento se estableció un sistema de visión global, el cual fue calibrado como se explica en [6]. En este sistema, se monta una cámara sobre la plataforma de navegación del robot y se conecta a una computadora donde se ejecuta el sistema de visión. El sistema detecta la posición y orientación del robot e informa toda esta información al sistema que controla el experimento. Para este escenario en particular, ambos sistemas (visión global y control de experimentos) funcionaron en la misma computadora. Para que el sistema de visión global identifique al robot, se adjuntaron unos parches de colores en la parte superior del robot. El sistema de visión global detecta esos parches y los utiliza para identificar y calcular la ubicación del robot dentro del entorno. Dado que el sistema de visión global, evaluado en [6], alcanza una precisión superior a 15 mm, sus medidas se utilizaron en este experimento como la verdad en el terreno.

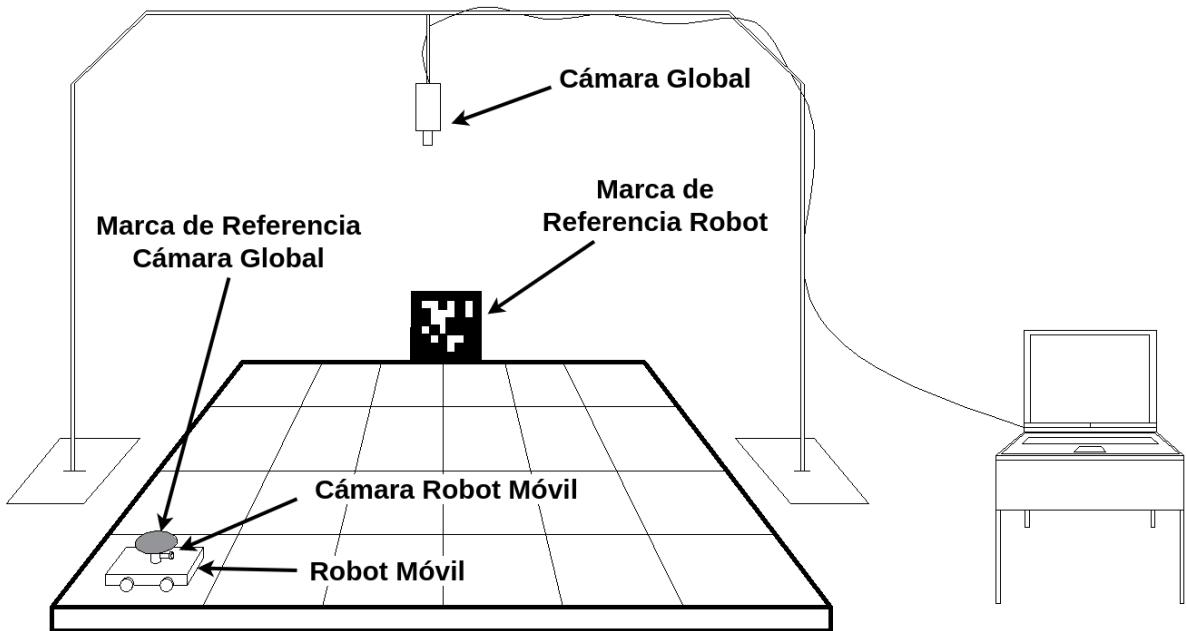


Figura 5.6: Diagrama del escenario real del segundo experimento

Entonces, se ubicó un TAG de 10cm x 10cm en una de las paredes del perímetro del ambiente.

Se encendió el robot, y este navegó reactivamente de manera aleatoria por el escenario, encontrando automáticamente el TAG con la cámara rotativa y reportando su localización. Cuando el prototipo encontraba la marca de referencia y se disponía a calcular su localización realizaba las siguientes cuatro acciones:

1. Enviaba una señal al robot y éste detenía los motores;
2. Reportaba su localización calculada;
3. Enviaba una señal al sistema de visión global y éste reportaba la localización real del robot;
4. Enviaba una señal al robot y éste comenzaba nuevamente a navegar de manera aleatoria.

Se obtuvieron 119 mediciones por parte del robot, y también de la cámara global. Los resultados obtenidos en 2) y 3) se representaron gráficamente para estimar la exactitud de las mediciones (Figura 5.7). El gráfico representa sistema de coordenadas del mundo real, y los círculos y asteriscos son las ubicaciones en el escenario reportadas por el sistema prototipo montado en el robot. Los círculos llenos indican una diferencia de [0-5]cm con respecto a la ubicación real reportada por la cámara global calibrada. Los círculos sin rellenos indican un error de entre [5-12]cm, y los asteriscos un error de [13-20]cm.

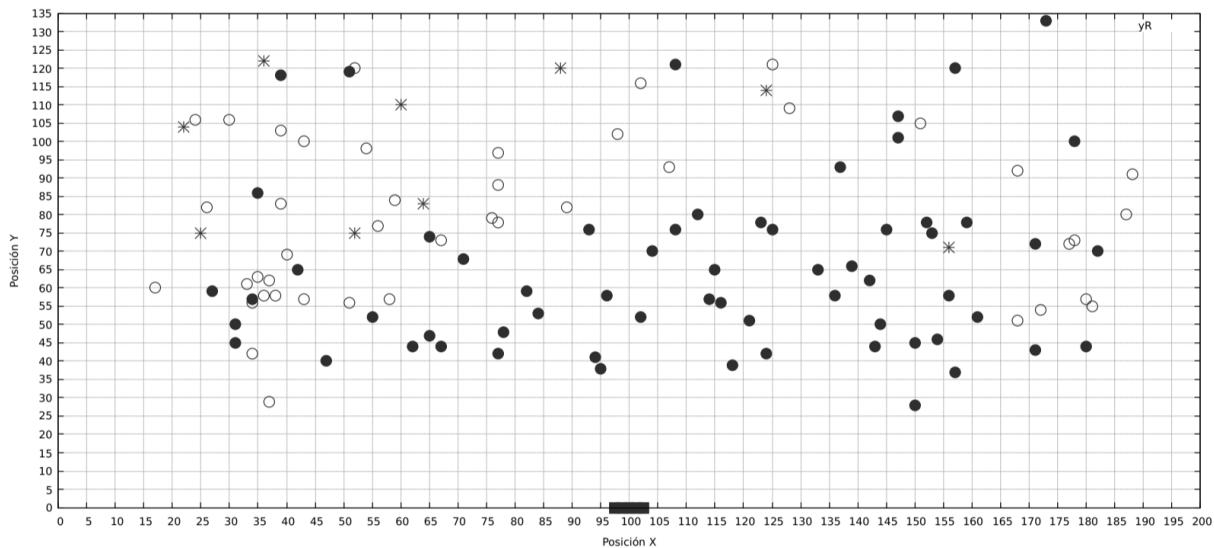


Figura 5.7: Ubicación de las localizaciones reportadas por el dispositivo prototipo en el segundo experimento

Se puede apreciar que con este tamaño de marca de referencia en el terreno (10cm x 10cm) y estando el robot a menos de 66cm de distancia de esta marca el sistema reporta, en la mayoría de los casos, ubicaciones con hasta 5cm de error con respecto al valor real. Luego, a mayores distancias la exactitud decrece, llegando hasta 12cm de error (círculos vacíos en la Figura 5.7), y en algunos casos se observan localizaciones con hasta 20cm de error respecto del valor real.

### 5.3. Conclusiones

En este capítulo se detallaron tres experimentos, y sus resultados, para evaluar la frecuencia alcanzada, y la precisión y error de las localizaciones obtenidas por el prototipo desarrollado en este trabajo.

En base a los resultados se concluye que el prototipo es capaz de realizar localizaciones con una frecuencia de 2Hz, y de 1Hz en el caso mas desfavorable. En cuanto a la precisión el sistema reporta mediciones dentro de un rango de (4,971 +/- 0,264) cm en el 65 % de los casos, si el robot navega en un escenario de 2mts x 2mts, y con una marca de referencia (TAG) de 20 cm de tamaño. El error esperable es de 5cm hasta una distancia de 66cm del TAG, y de hasta 12cm de error si el robot se encuentra a mas de 1 metro de distancia del TAG.



# Capítulo 6

## Conclusiones y trabajos futuros

### 6.1. Conclusiones

En este trabajo se detalló la arquitectura e implementación de un dispositivo prototípico para reportar localizaciones de robots móviles de bajo costo en ambientes interiores, utilizando marcas de referencia artificiales en el terreno. El tamaño, consumo e interfaz de programación está planteado para ser incorporado, en el futuro, a los robots educativos Frankestito-N6. Se experimentó para medir la frecuencia, precisión y exactitud, cuando se lo utiliza en un hardware de mínimas prestaciones.

A continuación se presenta un resumen de las tareas realizadas en esta tesis.

- Se estudiaron las técnicas generales de localización de robots móviles en interiores, haciendo hincapié en las tecnologías involucradas.
- Se realizó una revisión bibliográfica de las técnicas de detección de marcas de referencias artificiales, y se seleccionó una implementación adecuada para la localización de robots educativos.
- Se analizaron las computadoras embebidas disponibles para el trabajo, y se definió una arquitectura para todo el sistema.
- Se construyó, implementó, y evaluó el prototípico de hardware y software embebido. El hardware final logrado es una cámara rotativa, compuesta de un motor paso a paso, un microcontrolador, y una cámara USB. El software controla la cámara rotativa y a través de un port y ampliación del software AprilTag a la computadora embebida seleccionada (MIPS), es posible capturar fotos desde la cámara rotativa, realizar el cálculo de localización del sistema con respecto a la marca artificial, y reportar la localización del robot como resultado.
- Se evaluó el sistema realizando dos experimentos, que incluyeron la toma de mediciones por parte del prototípico y de un sistema de referencia. Se contrastaron los resultados y se caracterizó al sistema.

En base a los resultados se concluye que el hardware y software propuesto es capaz de localizar un robot de bajo costo a 2hz, y con una precisión de  $(4,971 \pm 0,264)$  cm el 65 % de los casos, si el robot navega en un escenario de 2mts x 2mts, y con una marca de referencia (TAG) de 20 cm de tamaño.

### 6.2. Trabajos futuros

Se enumera a continuación, y como trabajo futuro, posibles líneas de extensión y continuación de esta tesis.

- Evaluar el sistema con escenarios y marcas artificiales mas grandes.
- Continuar con el desarrollo o adaptación de los componentes restantes para lograr un sistema de navegación con mapas en robots móviles de bajo costo.
- Mejorar el rendimiento del prototipo actual, aumentando la frecuencia del sistema, sin modificar el hardware. Una posibilidad es modificando toda la aritmética de punto flotante por punto fijo, si ese cambio es factible, y evaluando la mejora de rendimiento. También se propone incrementar el rendimiento del módulo de localización, realizando mejoras en el software (por ejemplo, solapando la E/S con el tiempo de CPU).

### 6.3. Publicaciones e Impacto

Algunos de los resultados obtenidos en este trabajo han formado parte de las siguientes publicaciones:

- Rafael Ignacio Zurita, Alejandro Mora, Candelaria Alvarez, Miriam Lechner. Sistema de localización para robots móviles de bajo costo utilizando marcas de referencia artificiales en ambientes de interiores. XXV Congreso Argentino de Ciencias de la Computación (CACIC). 2019.
- Rafael Ignacio Zurita, Alejandro Mora, Candelaria Alvarez, Miriam Lechner. Localization System Using Artificial Landmarks for Indoor Low-Cost Mobile Robots. Computer Science - CACIC 2019. Springer International Publishing.

## Bibliografía

- [1] Artoolkit documentation. <http://www.hitl.washington.edu/ar toolkit/documentation>. [Online; accessed 2 October 2022].
- [2] ARTag Archive Web Site. <https://web.archive.org/web/20120814082107/http://www.artag.net/>, 2009. [Online; accessed 2 October 2022].
- [3] AprilTag 3 Repository. <https://github.com/AprilRobotics/apriltag>, 2019. [Online; accessed 2 October 2022].
- [4] Candelaria Álvarez. *Diseño e implementación de un sistema embebido de navegación por estimación para la localización de robots móviles en ambientes de interiores*. PhD thesis, Facultad de Informática. Universidad Nacional del Comahue, Marzo 2022.
- [5] Soumela Atmatzidou and Stavros Demetriadis. Advancing students' computational thinking skills through educational robotics. *Robot. Auton. Syst.*, 75(PB):661–670, January 2016.
- [6] David Ball, Wyeth Gordon, and Nuske Stephen. A global vision system for a robot soccer team. In *Australasian Conference on Robotics and Automation*, 2004.
- [7] Samuel Blanchard, Viktor Freiman, and Nicole Lirrete-Pitre. Strategies used by elementary schoolchildren solving robotics-based complex tasks: Innovative potential of technology. *Procedia - Social and Behavioral Sciences*, 2:2851–2857, 12 2010.
- [8] Michael Jae-Yoon Chung, Justin Huang, Leila Takayama, Tessa Lau, and Maya Cakmak. Iterative design of a system for programming socially interactive service robots. 2016.
- [9] Peter Corke and Oussama Khatib. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.
- [10] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.
- [11] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.
- [12] Mark Fiala. ARTag Revision 1, A Fiducial Marker System Using Digital Techniques. pages 11–42, 11 2004.
- [13] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.
- [14] J. Hallberg, M. Nilsson, and K. Synnes. Positioning with bluetooth. In *10th International Conference on Telecommunications, 2003. ICT 2003.*, volume 2, pages 954–958 vol.2, 2003.
- [15] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.
- [16] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.
- [17] Luc JAULIN. *Mobile Robotics*. Elsevier, 2015.

- [18] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94, 1999.
- [19] A. Kotanen, M. Hannikainen, H. Leppakoski, and T.D. Hamalainen. Experiments on local positioning with bluetooth. In *Proceedings ITCC 2003. International Conference on Information Technology: Coding and Computing*, pages 297–303, 2003.
- [20] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.
- [21] Nelson Barrera Lombana. Uso de la robótica educativa como estrategia didáctica en el aula. *Praxis Saber*, 6:215–234, 2015.
- [22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
- [23] Rainer Mautz. Indoor positioning technologies. 2012.
- [24] C. D. McGillem and T. S. Rappaport. Infra-red location system for navigation of autonomous vehicles. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1236–1238 vol.2, 1988.
- [25] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.
- [26] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgbd cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [27] L. M. Ni, Yunhao Liu, Cho Lau Yiu, and A. P. Patil. Landmarc: indoor location sensing using active rfid. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003)*, pages 407–415, 2003.
- [28] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. pages 3400 – 3407, 06 2011.
- [29] I. Poupyrev, H. Kato, and M. Billinghurst. *ARToolkit User Manual*. Human Interface Technology Lab, University of Washington, 2000.
- [30] P. Prasithsangaree, P. Krishnamurthy, and P. Chrysanthis. On indoor position location with wireless lans. In *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 720–724 vol.2, 2002.
- [31] Marvelmind Robotics. Precise Indoor Positioning System for autonomous robots, drones, vehicles and humans. <https://marvelmind.com/>, 2022. [Online; accessed 2 October 2022].
- [32] Jorge Rodriguez, Guillermo Grosso, Rafael Zurita, and Laura Cecchi. Intervención de la Facultad de Informática en la enseñanza de Ciencias de la Computación en la Escuela Media basada en Robótica Educativa. In *XI Congreso de Tecnología en Educación y Educación en Tecnología*, 2016.
- [33] Theodosios Sapounidis and Stavros Demetriadis. Educational robots driven by tangible programming languages: A review on the field. In *International Conference EduRobotics 2016*, pages 205–214. Springer, 2016.
- [34] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

- [35] A. Trachtenberg, E. Vardy, and C. L. Liu. Computational methods in coding theory. *Tech. Rep.*, 1996.
- [36] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '08*, pages 125–134, Washington, DC, USA, 2008. IEEE Computer Society.
- [37] Daniel Wagner and Dieter Schmalstieg. Artoolkitplus for pose tracking on mobile devices. 01 2007.
- [38] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.
- [39] Melonee Wise, Michael Ferguson, Daniel King, Eric Diehr, and David Dymesich. Fetch & freight : Standard platforms for service robot applications. 2016.
- [40] Hong'an Yang, Xuefeng Bao, Shaohua Zhang, and Xu Wang. A multi-robot formation platform based on an indoor global positioning system. *Applied Sciences*, 9:1165, 03 2019.
- [41] Xu Zhong, Yu Zhou, and Hanyu Liu. Design and recognition of artificial landmarks for reliable indoor self-localization of mobile robots. *International Journal of Advanced Robotic Systems*, 14(1):1729881417693489, 2017.
- [42] Rafael Zurita, Juan de la Fuente, Martín Bucarey, Daiana Bonet, Rodolfo del Castillo, Guillermo Grossó, Laura Cecchi, and Jorge Rodríguez. Mejorando las posibilidades de aprender a programar: ampliación del robot educativo multiplo n6 max a frankestito. 06 2017.
- [43] Rafael Zurita, Alejandro Mora, and Candelaria Álvarez. Cam Localization for MIPS. [https://github.com/zrafa/cam\\_localization\\_for\\_mips](https://github.com/zrafa/cam_localization_for_mips), 2019. [Online; accessed 2 October 2022].