2021-09

DETECCIÓN DE LA DISPOSICIÓN DE LA VÍA PARA UNA PLATAFORMA EXPERIMENTAL DE AGENTES AUTÓNOMOS BASADO EN INTERPRETACIÓN DE SEÑAL ÉTICAS DE CÓDIGO QR

GARRIDO VALENZUELA, CRISTOBAL ANTONIO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE

"Detección de la disposición de la vía para una plataforma experimental de agentes autónomos basado en interpretación de señaléticas de código QR"

CRISTÓBAL ANTONIO GARRIDO VALENZUELA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO, MENCIÓN COMPUTADORES

PROFESOR GUIA: **FRANCISCO VARGAS**

PROFESOR CORREFERENTE: **ANDRÉS PETERS**

PROFESOR CONSULTOR: **JORGE VERGARA**

A mis padres por siempre apoyarme
en todas las decisiones que he tomado
en mi vida y por siempre confiar en mí

Agradecimientos

Quiero agradecer a todas las personas que me han acompañado durante esta larga etapa en mi vida y que hoy en día me acompañan al momento de cerrarla. Especialmente a mis padres que siempre han confiado en mí y en mis capacidades. Además, quiero agradecer a mi hermana por siempre preocuparse por mi y apoyarme. También quiero darle las gracias a mi novia por ser un pilar fundamental en mi vida y por ayudarme a crecer como persona.

A mis amigos que me acompañan desde el colegio y a los que conocí a lo largo de este camino, quienes me han dado fuerzas para seguir adelante y nunca rendirme.

Además, agradecer a los profesores Andrés, Francisco y Jorge por dedicarme gran parte de su tiempo y apoyo en este proyecto.

Finalmente, agradecer al Centro Avanzado de Ingeniería Eléctrica y Electrónica (AC3E) y a los colaboradores correspondientes por darme la oportunidad de desarrollar este trabajo basado en su proyecto.

Resumen

Esta memoria presenta el desarrollo de un sistema de detección de la señalética de la vía de una plataforma experimental de agentes autónomos que se desplazan sobre rieles. Este sistema está basado en el procesamiento y análisis de imágenes para detectar e interpretar señaléticas que están codificadas mediante códigos QR, las que indican un cambio en la disposición de las vías.

La motivación de este trabajo surge para resolver algunos problemas en el desarrollo de una plataforma experimental de agentes a escala que navega de forma autónoma, la que busca observar, medir y corregir problemas generados en control multi-agente. El problema a resolver de este proyecto basal surge cuando los agentes se ven enfrentados a curvas pronunciadas, pues los sensores que mantienen la distancia de un agente con respecto a otro tienen un ángulo limitado de visión. Si la cadena de agentes pasa por una curva, estos pierden dicha medición, lo que produce grandes errores en el movimiento de la flota pudiendo hasta generar que los agentes colisionen.

La propuesta planteada en este trabajo es añadir el microcontrolador ESP32-CAM, el que cuenta con una cámara capaz de capturar imágenes que posteriormente son procesadas e interpretadas mediante un algoritmo desarrollado para este microcontrolador.

Los resultados obtenidos muestran que la implementación de la solución propuesta logra cubrir la falencia presentada en el proyecto basal principalmente para velocidades bajas.

Dentro de las conclusiones más relevantes se tiene que los tiempos de procesamiento ocupados en cada imagen son los limitantes de la solución, los que a su vez dependen de la capacidad de procesamiento del microcontrolador.

Palabras Clave:

ESP32-CAM, plataforma de agentes autónomos, control multi-agente, Código QR.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y Objetivos

1.1. Motivación

En los últimos años el avance tecnológico y su impacto en nuestro día a día es cada vez más fuerte. Un ejemplo es la tecnología utilizada en la industria automovilística, ya que su continuo avance ha permitido mejorar la experiencia de sus conductores y el funcionamiento de los vehículos. En consecuencia, uno de los grandes desafíos actuales de la industria automotriz es implementar sistemas de vehículos autónomos capaces de realizar labores de manera más eficiente y eficaz que al ser conducidos por un ser humano[1].

Cada vez existen más modelos de vehículos que cuentan con grados de asistencia definidos en una escala de seis niveles. Donde el nivel cero indica que no existe ningún tipo de asistencia en el manejo mientras que el nivel cinco indica que existe una completa autonomía del vehículo, es decir, que no depende de un conductor en ninguna circunstancia para su funcionamiento.

El objetivo final el desarrollo de vehículos autónomos es llegar a contar con una automatización completa (nivel 5) y de esta forma evitar errores humanos en las carreteras, aumentando la seguridad de las personas que se desplazan por estas. Además, este tipo de automatización al ser más eficiente disminuirá los recursos malgastados por los errores humanos al momento de la conducción, tales como insumos de reparación y de mantención.

Otro objetivo de esta automatización es lograr generar flotas de vehículos para trabajos dentro de faenas, para aumentar su productividad gracias a que las flotas autónomas requieren menos pausas disminuyendo consigo los tiempos muertos entre cambios de turnos. Además, se disminuyen

los gastos asociados a combustible y desgaste de neumáticos, debido a que los camiones autónomos suelen ir a velocidades más constantes durante el movimiento de cargas. El factor de accidentes laborales también disminuye gracias a la seguridad con la que cuentan los sistemas autónomos, lo que genera una faena más segura para todos los trabajadores.

Como se mencionó, estos tipos de vehículos aún se encuentran en desarrollo, por lo que cada día se consiguen pequeños avances en su teoría o aplicabilidad. Un problema que se desprende al momento de estudiar y desarrollar estos sistemas, es que se utiliza tecnología muy cara, ya que los sistemas necesitan tener una gran fiabilidad en sus componentes al momento del funcionamiento. Es por esto que no todos los laboratorios o investigadores tienen la capacidad económica para estudiar y hacer pruebas con respecto a estas áreas.

1.2. Formulación del problema

Dentro del contexto mencionado en la sección 1.1, se desarrolla el proyecto base del cual se desprende este trabajo. Se sabe que las flotas autónomas de vehículos están en continuo desarrollo [2] , por lo que se necesita investigar y realizar pruebas para poder determinar las mejores condiciones en su operación o verificar la teoría detrás de estos sistemas multi-agente. Por tal razón, los colaboradores del proyecto base desarrollan una plataforma experimental a pequeña escala y de bajo costo para poder verificar las propiedades de escalabilidad de estos sistemas. Ellos tienen como objetivo crear una plataforma capaz de replicar sistemas multi-agente donde se puedan ver los problemas o características de distintos métodos de control en el movimiento de la flota, ya que para estudiar estos tipos de sistemas, no todos los laboratorios pueden contar con camiones o vehículos de alta tecnología para realizar pruebas, ya sea por recursos económicos o de infraestructura.

Los agentes que son parte de esta plataforma, solo se mueven hacia adelante o hacia atrás, debido a que su movimiento está restringido por las vías tipo tren por las que se desplazan.

Además, dentro de la flota es importante qué agente es el que se encuentra en la primera posición, debido a que éste tiene configuraciones distintas al resto de agentes. Es por esto que este agente se considera el **Agente Líder** y así será mencionado a la largo de este proyecto. El objetivo de cada agente distinto al Agente Líder es avanzar manteniendo una distancia establecida con respecto al agente que lo antecede en la flota. Mientras que el Agente Líder se encarga de mantener una velocidad constante establecida y de entregar información al resto de la flota.

El modelo actual de la flota, permite medir su velocidad y la distancia con respecto al agente anterior. En la plataforma, la velocidad del Agente líder es enviada mediante protocolos de comunicación al resto de los agentes para que estos utilicen dicha información y así mejorar el comportamiento de la flota. En la Figura 1.1 se muestra un esquema básico del funcionamiento de esta flota.

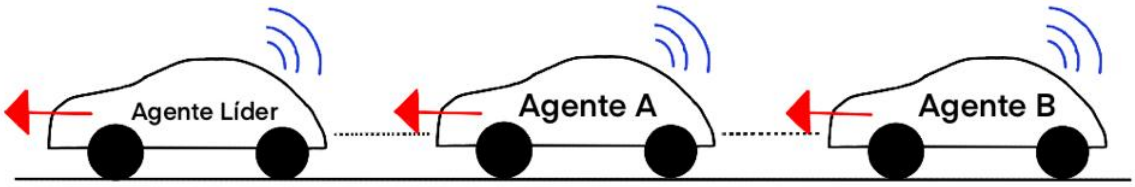

Figura 1.1: Esquema básico de funcionamiento de la flota de agentes autónomos

En este esquema de la Figura 1.1 se muestran tres agentes moviéndose hacia la izquierda. Se puede observar que solo los agentes A y B miden la distancia con respecto al agente anterior. Además, todos los agentes conocen su velocidad y cuentan con comunicación inalámbrica para interactuar entre ellos. Una vez que cada agente mide con sus sensores y recibe la velocidad del Agente Líder, calcula una actuación mediante un método de control programado en el microcontrolador de este agente.

La direccionalidad del sensor de distancia presente en cada uno de los agentes (excepto en el Agente Líder) provoca un potencial problema en curvas, donde los agentes predecesores aparentan desaparecer del ángulo del sensor al virar, especialmente al mantener mucha distancia entre agentes como se observa en la Figura 1.2.

En la Figura 1.2 se observa que el Agente A ya recorrió la curva, mientras que el Agente B aún no lo hace, por lo que el primer agente salió del ángulo de visión del sensor de distancia de este segundo agente. Esta pérdida de visión genera que el Agente B no realice un cálculo correcto con respecto a la actuación que debe generar en el motor, lo que a su vez produce error en su movimiento y posibles colisiones con los otros agentes.

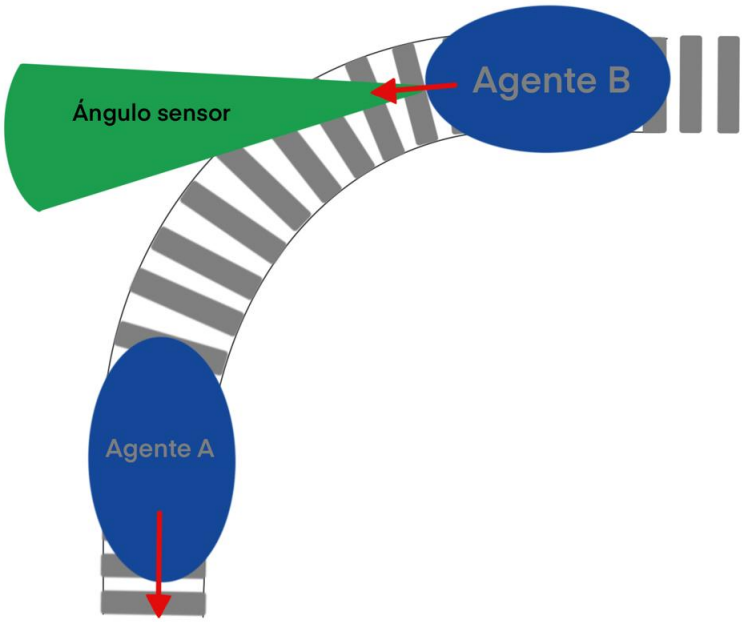

Figura 1.2: Esquema de pérdida de visión de sensor de distancia. Agente B no logra medir distancia hacia el Agente A, debido a que este último se encuentra en una curva.

El presente proyecto tiene como objetivo mejorar el comportamiento de la flota en estas situaciones donde se presenta una curva en la trayectoria de los agentes. Actualmente los agentes no tienen un mecanismo de medición que logre indicar la presencia de un cambio en la disposición de las vías que transitan, por lo que los agentes no son capaces de corregir los errores generados por estas curvas.

La solución desarrollada en este trabajo consiste en posicionar señaléticas con un código QR a un costado de las vías por las que se desplazan los agentes. Estas señaléticas son capturadas en imágenes y luego procesadas para decodificar dichos códigos QR. La captura de imágenes y procesamiento de éstas está a cargo del microcontrolador ESP32-CAM, el que se posicionará en la zona delantera del Agente Líder para tener visibilidad de las señaléticas.

Al finalizar esta memoria, se logró acoplar la solución desarrollada al funcionamiento basal de la flota de agentes. De esta forma se puede utilizar la información con respecto al cambio en la disposición de la vía medido mediante las señaléticas con códigos QR y gracias a esto no existe errores al momento de seguir un circuito que no es recto a velocidades bajas. Un correcto procesa-

miento de imagen logra ser eficiente y eficaz a la hora de determinar la indicación correspondiente en la señalética, lo que otorga al sistema un tiempo adecuado para cambiar el tipo de control anticipándose a lo que viene en el camino.

1.3. Estado del Arte

El objetivo de este proyecto es complementar las funcionalidades que tiene un proyecto basal. Por esta razón se comenzará con el trabajo de C. Andrade y col [3] para introducir esta memoria con respecto a la flota de agentes autónomos con microcontroladores ESP32. En C. Andrade y col [3] se presenta una plataforma de bajo costo para la observación y análisis de sistemas multiagente. Gracias a este trabajo, se logra comprender las capacidades y límites que tiene el sistema con el que se trabajará. Además, especifica todos los componentes asociados a dicha plataforma. Cabe destacar que hoy en día el proyecto ha seguido avanzando y ya cuenta con capacidades como almacenamiento automático de datos y gráfico en tiempo real de variables internas y de las mediciones de los sensores de todos los agentes. Esta característica ayudará al momento de realizar las pruebas para almacenar, comparar y concluir con respecto a los resultados obtenidos.

Ahora bien, como se explicó en la sección 1.2 este proyecto basal necesita la información necesaria con respecto a la existencia de una curva en la trayectoria, por lo que frente a esta problemática se encuentran tres posibles tecnologías existentes hoy en día que son capaces de solucionar lo planteado.

La primera tecnología encontrada corresponde a los sensores LIDAR, los que son capaces de escanear grandes áreas y mapear el entorno en el que se encuentran. Esto se puede encontrar en el trabajo de Ghorpade y col [4] donde el sensor es utilizado para encontrar obstáculos en el recorrido de un vehículo que se mueve en un plano 2D. Esta misma tecnología se puede aplicar a esta plataforma y se pueden establecer tipos de figuras geométricas para indicar un cambio en la disposición de las vías. Cabe destacar que se plantean las figuras geométricas, porque el sensor no tiene la suficiente capacidad de escaneo como para ir en movimiento detectando las vías de los agentes debido al tamaño de estas. Además, se debe tener en consideración que esta tecnología aún es muy cara, por lo que adquirir este sensor elevaría el costo total de la plataforma.

La segunda opción planteada es la localización mediante sensores Ultra Wide Band (UWB),

la cual es una tecnología de radio de corto alcance con gran capacidad de transmisión de datos (480 Mbits/s). Este tipo de comunicación se utiliza para un sistema de localización basado en la triangulación de la posición de un objeto. Esto se puede aplicar instalando varias anclas alrededor de las vías donde se realizará el experimento y estableciendo el Agente Líder como el objeto a localizar. Con esta configuración se puede predecir la disposición de las vías por las que pasará el agente. Al igual que los sensores LIDAR, aplicar esta opción es muy cara, ya que no son tecnologías comunes en el día a día.

La tercera opción planteada, corresponde al procesamiento de imágenes para detectar señaléticas que indiquen un cambio en la disposición de las vías. Hoy en día se conocen muchos usos para el procesamiento digital de imágenes, debido a su versatilidad y facilidad de integrarse a diferentes sistemas y aplicaciones. Desde un detector de incendios forestales basado en el color y forma de las llamas [5], hasta sistemas de seguridad basado en el movimiento o reconocimiento de objetos o personas [6]. En el ámbito del control automático, las cámaras y sus algoritmos de procesamiento de imágenes también han sido utilizadas por ejemplo: para verificar si el entorno donde se está trabajando es seguro, que no ha ocurrido ningún problema en los procesos o utilizando una imagen como referencia para el control de una señal. Es por esto que la combinación del control automático junto con el procesamiento digital de imágenes cada día es más fuerte, ya que permite autorregular o verificar el estado de muchos sistemas de control.

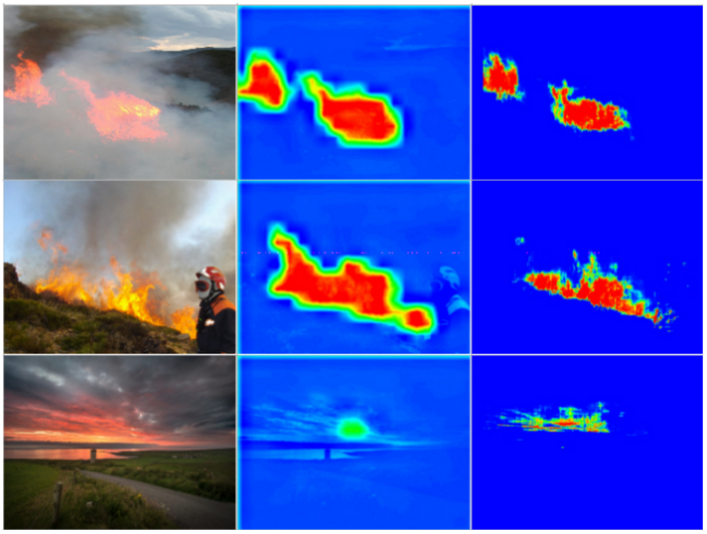

Figura 1.3: Ejemplo de procesamiento de imágenes para detectar un incendio forestal.

Teniendo en cuenta el uso del procesamiento de imágenes, se logra el procesamiento de un código QR, es por esto que primero se debe tener completa claridad sobre el funcionamiento, capacidades y limitaciones de los códigos QR. En la publicación de Tan Jin Soon [7] se detallan las capacidades de los códigos QR y cómo establecer las decodificaciones en un procesamiento de imagen. Por esta razón se utilizará esta información para programar las funciones de decodificación del código QR y las de detección de estos. Los códigos QR son aplicados en la plataforma en calidad de señaléticas, por lo que se necesita de un dispositivo capaz de capturar imágenes y procesarlas para la correcta entrega de la información codificada.

En este sentido, el dispositivo escogido es el microcontrolador ESP32-CAM, el que cuenta con un procesador similar a la placa ESP32. A raíz de ello se utilizará el trabajo [8] para aprender a programar dichos microcontroladores. Con respecto a la ESP32-CAM, es muy poco el trabajo que hoy en día se tiene sobre éste. Por este motivo la mayor información se obtendrá a partir de [9] donde se utiliza un algoritmo de reconocimiento de gestos para la ayuda en una sala de clases. En este se utiliza una ESP32-CAM junto con una Raspberry Pi, por lo que de ser necesario para el procesamiento, también se puede aplicar dicha combinación de hardware para garantizar una correcta y rápida interpretación del código QR. Esto sin olvidar que el objetivo principal es lograr el procesamiento solo con la ESP32-CAM, así se mantiene el bajo costo que tiene la plataforma de experimentos. También, se tiene [10] donde se presenta el uso de la ESP32-CAM para obtener imágenes de una casa la que ha sido automatizada utilizando las placas ESP32 y ESP32-CAM. Esto corrobora que la cámara que se quiere utilizar en la aplicación de este proyecto es acoplable a un sistema como lo es la flota de agentes autónomos.

Finalmente, para el correcto acoplamiento del sistema y los resultados esperados, se tiene el trabajo [11] de multi-agentes, donde se presenta los posibles resultados que se espera tener en cuanto al comportamiento de los agentes. Esto basado en la topología de la flota, el tipo de control y el tipo de sensado que tiene cada uno de los agentes y en las características generales de la flota. Gracias a esto se corroborará si se cumple el objetivo del proyecto que consiste en mejorar el seguimiento de la flota, de manera que se espera menor error en el seguimiento gracias a la detección de la disposición de la vía.

1.4. Objetivos

A continuación se establecen los objetivos para cumplir con los requerimientos básicos de una correcta solución. Estos objetivos son:

- Añadir señaléticas con códigos QR a un lado de las vías para indicar un cambio en la disposición de éstas. Además, determinar la mejor ubicación y ángulo de dichas señaléticas.
- Añadir al diseño actual del primer agente de la flota un dispositivo ESP32-CAM, determinando el método de interconexión necesaria con las componentes existentes.
- Elaborar código para la ESP32-CAM capaz de detectar e interpretar los códigos QR de las señaléticas.
- Determinar limitaciones y capacidades del proceso de detección con el hardware y software utilizado.
- Programar la flota para recibir la señal de cambio en la disposición de las vías y ajustar su sistema de seguimiento entre agentes.

Capítulo 2

Antecedentes

En este capítulo se entregará una breve información con respecto a hardware, software y conocimientos utilizados a lo largo del desarrollo de este proyecto.

Primero se abarcará el respectivo hardware utilizado en la plataforma base y en la solución desarrollada. Seguido de esto, se mencionará el software Arduino IDE y la librería Quirc lo que es necesario para la programación del hardware recién descrito.

Posteriormente, se describirán los códigos QR para conocer la base de su funcionamiento y el objetivo e impacto que tienen en la sociedad actualmente.

Finalmente, se mencionará la base del procesamiento y análisis de imágenes que se requiere saber para comprender el funcionamiento del algoritmo que procesa las imágenes de la solución desarrollada.

2.1. Microcontrolador ESP32

ESP32 es un microcontrolador de bajo consumo calificado como sistema en chip (SoC) el que cuenta con características de comunicación inalámbrica tales como WiFi o Bluetooth. Este microcontrolador es creado y desarrollado por la compañía china Espressif Systems como un sucesor de su antiguo y muy versátil microcontrolador ESP8266.

A continuación se detallarán las características más importantes de este microcontrolador que se pueden encontrar en su datasheet [12]:

- Microprocesador de 32-bit Xtensa LX6 de doble núcleo
- 520 KiB SRAM
- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR y BLE
- 34 Puertos de Salida/Entrada
- 4 Puertos para SPI
- 2 interfaces I^2C
- 2 interfaces I^2S
- 3 Puertos para UART

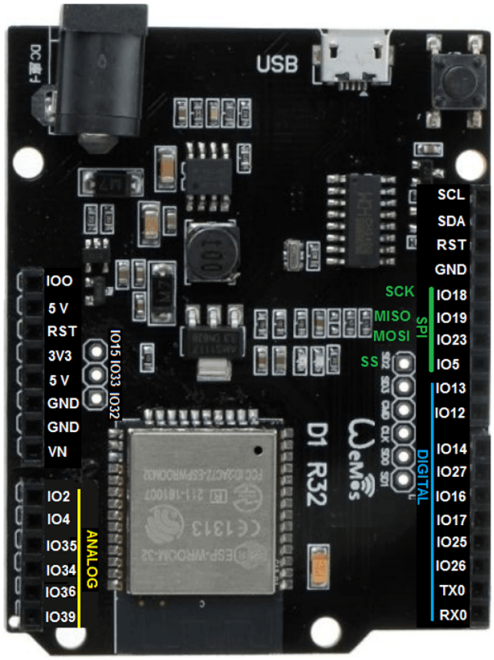

Figura 2.1: Imagen de un microcontrolador ESP32

Estas cualidades convirtieron esta placa en la opción ideal para reemplazar al Arduino en este proyecto, tal como se menciona en la sección 2.8.

Esta placa junto con la ESP32-CAM (descrita en la sección 2.2) son de la misma familia de microcontroladores, por lo que comparten ciertas características, entre ella se encuentra la capacidad

de programar estas placas mediante la herramienta ESP-IDF o el entorno de trabajo de Arduino. Este último se utiliza actualmente para desarrollar los algoritmos que realiza el procesador en cada agente perteneciente a la plataforma. Es por esto que se debió revisar estos códigos y librerías usadas para entender el funcionamiento individual de cada agente y así crear una solución acorde a lo planteado como objetivo, el que consiste en acoplar la solución desarrollada a la plataforma base. Además, gracias a esto se logra crear posteriormente un código capaz de utilizar la información entregada por la ESP32-CAM.

2.2. Descripción de ESP32-CAM

ESP32-CAM es una placa de desarrollo basada en ESP32 de bajo costo con una cámara integrada, pero de tamaño pequeño. Es una solución ideal para aplicaciones IoT, construcciones de prototipos y proyectos de bricolaje. Un imagen de la ESP32-CAM se muestra en la Figura 2.2.

La placa integra WiFi, Bluetooth tradicional y BLE de bajo consumo, con 2 CPU alto rendimiento LX6 de 32 bits. Adopta una arquitectura de tubería de 7 etapas, sensor en chip, sensor de pasillo, sensor de temperatura, etc., y sus principales rangos de ajuste de frecuencia de 80MHz a 240MHz.

Totalmente compatible con los estándares WiFi 802.11b/g/n/e/i y Bluetooth 4.2, se puede utilizar como un nodo maestro para construir un controlador de red independiente, o como esclavo de otro host MCU para agregar capacidades de red a dispositivos existentes.

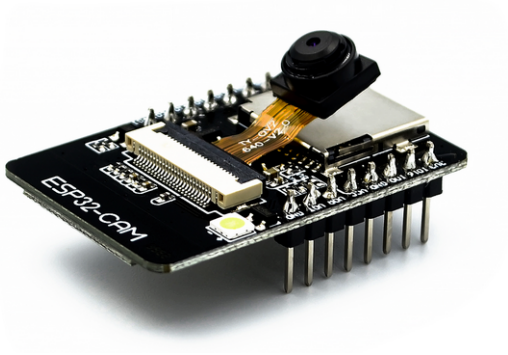

Figura 2.2: Placa ESP32-CAM

A continuación, se presentan algunas de las características más importantes extraídas del datasheet [13] de este microcontrolador.

- Velocidad de reloj hasta 160MHz
- 520KB de SRAM interna
- Soporta UART/SPI/I2C/PWM/ADC/DAC
- Soporta FreeRTOS
- SPI Flash de 32Mbit
- Bluethooth 4.2
- Wifi 802.11b/g/n/e/i
- 9 Puertos de Salida/Entrada
- Puerto Serial por defecto de 115200 bps
- Imagen de salida en formatos: JPEG,BMP y GRAYSCALE
- Rango de espectro: 2412-2484 MHz
- Antena en integrada en la PCB
- Consumo: 180mA en 5V con flash apagado y 310mA en 5V con flash encendido al máximo
- Alimentación: 5V
- 10g de peso

Además de estas funcionalidades, es importante mencionar que para la programación de esta placa se utiliza principalmente el entorno de trabajo de desarrollo de la marca Espressif: ESP-IDF. Este entorno viene con un SDK incluido para el desarrollo de las aplicaciones.

Finalmente, este tipo de placas también se puede programar con el IDE de Arduino. El ejemplo de Código QR en el que se basará el proyecto, está desarrollado con Arduino IDE, por lo que se debe seguir la misma línea de trabajo para lograr una correcta implementación dentro de la plataforma.

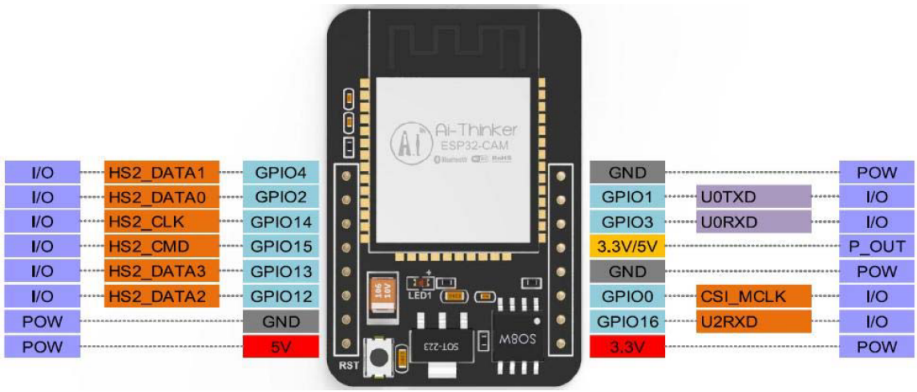

Figura 2.3: Esquema de pines de microcontrolador ESP32-CAM

2.3. Cámara de ESP32-CAM

En la placa ESP32-CAM viene incluida una cámara modelo OV2640, la que es la encargada de capturar las imágenes que posteriormente serán procesadas por el algoritmo. Esta cámara cuenta una capacidad de capturar imágenes de hasta resolución UXGA (1600x1200 píxeles), por lo que las imágenes se consideran de alta calidad para el costo que tiene dicha cámara.

Figura 2.4: Cámara OV2640

Si bien esta cámara puede capturar en resoluciones muy altas, el procesador no es capaz de procesarlas mediante el algoritmo de reconocimiento y decodificación de QR de manera rápida, por lo que al momento de escoger la resolución a utilizar, se debe tener en cuenta el tiempo de respuesta máximo que se requiere. Esto último será abordado de manera más específica en la sección 3.3.3 de este trabajo.

A continuación, se detallarán otras características de la cámara presentadas en su datasheet [14] a tener en consideración:

- Resolución UXGA, SVGA y menores
- Rango Dinámico 50dB
- Sensibilidad 0.6V/Lux-sec
- Tasa de transferencia máxima 1600x1200 a 15 frames por segundo
- Relación señal/ruido 40dB
- Formato de salida YUV/RGB/Raw RGB Data

2.4. Arduino IDE

Arduino IDE es un entorno de programación para los microcontroladores Arduino, el que es conocido a través del mundo gracias a su simpleza y alta funcionalidad en el mundo de la robótica. Este entorno es una aplicación capaz de correr en distintas plataformas como Linux, macOS y Windows. Además, soporta lenguaje C y C++ con ciertas restricciones en su estructura.

Si bien se utiliza como entorno para desarrollar y cargar software en placas Arduino, esta también cuenta con la opción de trabajar con distintas placas de desarrollo de otros fabricantes. Entre estas placas soportadas, se encuentra ESP32 y ESP32-CAM. Esto lo hace mediante un administrador de librerías, dentro del cual se encuentra además una base de librerías comúnmente usada por los desarrolladores.

Si bien durante el comienzo del proyecto se utilizaba la versión actual del Arduino IDE, los últimos meses se optó por la nueva versión beta de este entorno de desarrollo, ya que esta nueva versión cuenta con mejores opciones al momento de desarrollar un código.

Entre las mejoras se destaca la capacidad de buscar la librería y sección de código a la que pertenece una variable. Esto ayudó al entendimiento del código de reconocimiento y de la librería quirc, ya que permitía un mayor seguimiento de las variables y estructuras utilizadas.

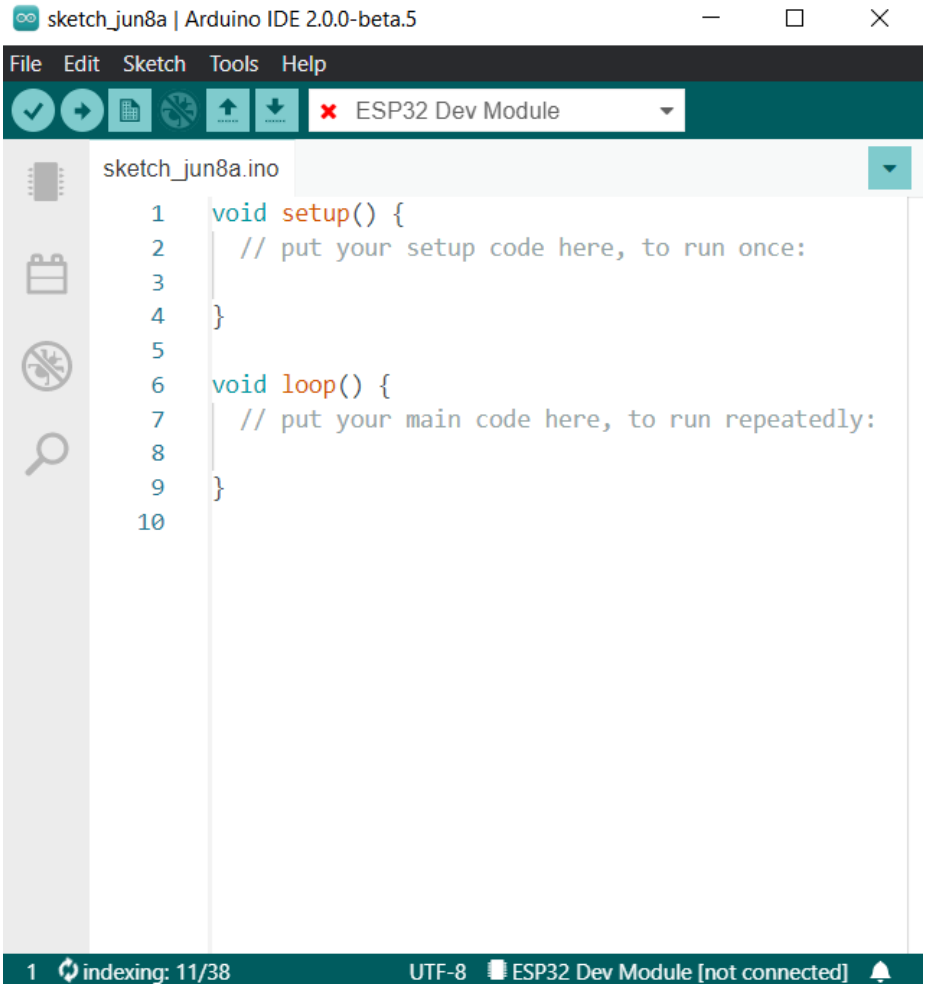

Figura 2.5: Arduino IDE

2.5. Código QR

El código QR es un tipo de código de barras creado en Japón en el año 1994 por la compañía Denso Wave. El objetivo es poder almacenar mayor información que los códigos de barra de ese entonces, donde solo se podían codificar símbolos alfanuméricos.

El código QR permitió almacenar mayor información gracias a su codificación en 2D. Entre esto, se encontraba la capacidad de almacenar kanjis, lo que ayudó mucho a los avances tecnológicos que estaba llevando acabo Japón en ese entonces. Esta información se extrae a partir del sitio oficial **https://www.qrcode.com/en/**

Un ejemplo de código QR se observa en la Figura 2.6.

Figura 2.6: Ejemplo de código QR

Este es un tipo de código de barra con una matriz bidimensional el que consta de dos colores que se contrasten entre ellos (comúnmente blanco y negro) para mejorar posteriormente la eficiencia de la decodificación. Cuenta con distintas versiones, desde la 1 a la 10, donde la versión 1 consiste en un código de 21 x 21 módulos y la versión 10 contiene 177 x 177 módulos.

Los módulos mencionados corresponden a la cantidad de subdivisiones dentro del código, entre estos módulos se encuentran los distintos factores que convierten esta cuadrícula en una correcta estructura de código QR, entre estos factores se encuentran: Patrones de máscara, nivel de corrección de errores, formato de la corrección de errores, posición, alineamiento y sincronización. Todo esto se muestra en la Figura 2.7

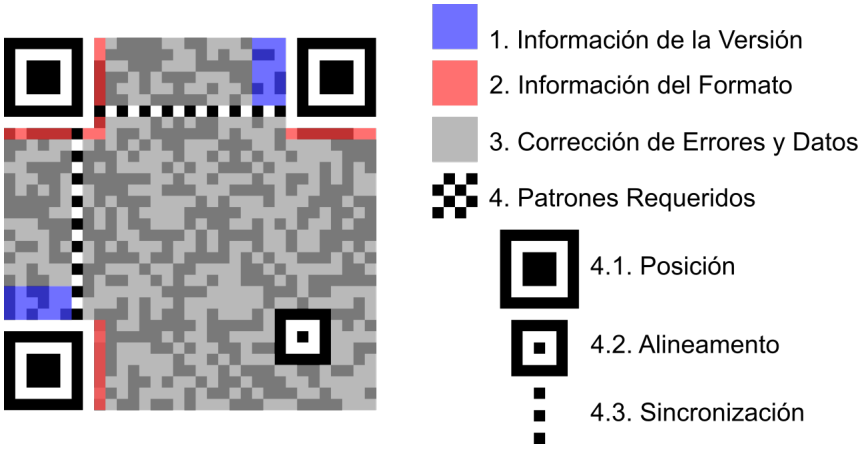

Figura 2.7: Estructura de código QR

Si bien en un comienzo esto se utilizaba para identificar productos en fábricas o tiendas, hoy en día es utilizado para entregar información o redireccionar a distintas páginas webs solo con analizar el código QR con la cámara de un smartphone. Es por esto que si bien existen códigos QR de versión 10 con 177x177 módulos, lo normal es encontrar códigos QR con solo 25x25 o 29x29, debido a que estos tienen la capacidad suficiente de datos para las aplicaciones utilizadas en el día a día de las personas. Gracias a su simpleza de funcionamiento y a su fácil reconocimiento de patrón, cada vez se utiliza más en distintas aplicaciones cotidianas, lo que ha generado que esté fuertemente presente en la cultura popular actual.

2.6. Librería Quirc

Quirc es una librería para extraer y decodificar códigos QR de imágenes. Esta librería está en lenguaje C, lo que permite ser usada en el entorno Arduino IDE. Tiene varias características que lo convierten en una buena opción para el propósito de este trabajo, entre ellas:

- Es lo suficientemente rápido para aplicaciones de video en tiempo real.
- Cuenta con un algoritmo de reconocimiento robusto y tolerante.
- Puede detectar códigos rotados y reconoce múltiples códigos en una misma imagen.
- Es pequeño y fácilmente integrable en proyectos.
- Se puede utilizar sin problemas en aplicaciones que utilizan multihilos.
- Cuenta con licencia BSD, casi sin restricciones de uso y/o modificación,
- Está diseñada y optimizada para ser utilizada en sistemas embebidos.

Dentro de esta librería, se encuentran las funciones básicas para lograr un correcto descifrado de códigos QR dentro de una imagen, pero para lograr usarlas adecuadamente, se debe tener en cuenta que esta función no es la única que cumple el procesador, por lo que se requiere de un correcto manejo de semáforos para que no existan errores al momento de trabajar con los datos de las imágenes. Basado en esto es que no se utiliza esta librería directamente, si no que se optó por una librería llamada ESPIno32CAM. Esta última cuenta con funciones de detección y descifrado

de códigos QR en imágenes, ya que utiliza la librería Quirc como base, pero además administra la memoria y los procesos de la ESP32-CAM para adaptar correctamente todas las funcionalidades de Quirc y el resto de procesos del microcontrolador sin generar conflictos entre ellos.

A continuación se mencionarán las principales funciones utilizadas de esta librería dentro del desarrollo de este proyecto:

- $quirc\_new$(void): Construye un nuevo reconocedor de códigos QR y retorna NULL si no cuenta con la suficiente memoria disponible.

- $quirc\_destroy$(struct quirc *q): Libera la memoria utilizada en quirc destruyendo el reconocedor de códigos QR. El argumento es la dirección de memoria donde se creó anteriormente el quirc que ahora se desea destruir.

- $quirc\_resize$(struct quirc *q, int w, int h): Se encarga de modificar el tamaño de la memoria del reconocedor dependiendo de la resolución de la imagen que utiliza. Los argumentos corresponden respectivamente: (1) dirección de memoria donde se encuentra el quirc utilizado, (2) ancho en píxeles de la imagen que se procesará y (3) alto en píxeles de la imagen que se procesará.

- $*quirc\_begin$(struct quirc *q, int *w, int *h): Entrega el acceso al buffer de quirc para posteriormente ingresar la imagen que será procesada. Los argumentos corresponden respectivamente: (1) dirección de memoria donde se encuentra el quirc utilizado, (2) ancho en píxeles de la imagen que se procesará y (3) alto en píxeles de la imagen que se procesará.

- $quirc\_end$(struct quirc *q): Indica que ya se terminó de copiar la imagen al buffer, por lo que se comienza a procesar la imagen. El argumento es la dirección de memoria donde se encuentra el quirc utilizado.

- $quirc\_extract$(const struct quirc *q, int index, struct $quirc\_code$ *code): Extrae un código específico de la imagen en caso de existir más de uno en la misma imagen. Los argumentos corresponden respectivamente: (1) dirección de memoria donde se encuentra el quirc utilizado, (2) índice del código QR que se desea extraer desde la imagen y (3) estructura donde se almacenará el código QR extraido de la imagen.

- $quirc\_decode$(const struct $quirc\_code$ *code, struct $quirc\_data$ *data): Decodifica el código QR y entrega la información almacenada en éste. Esta función recibe como argumentos: (1) el código QR que se desea decodificar (extraido con $quirc\_extract$) y (2) estructura donde almacenará la información decodificada del código QR ingresado.

2.7. Procesamiento y análisis de imágenes

Desde la creación de la imágenes digitales, se ha trabajado con ellas para poder obtener información o modificarlas a través de procesos computacionales. Un breve ejemplo de esto es mejorar la calidad de la imagen gracias a filtros aplicados posteriormente a la captura de la imagen.

Dentro de este ámbito se ve la imagen como una matriz multidimensional a la cual se le pueden aplicar ciertos algoritmos. La base de esto es un plano de dos dimensiones catalogado como píxeles, donde cada píxel puede contener distinta información según el formato de imagen con el cual se esté trabajando.

Existen las imágenes en escala de grises y las imágenes a color. Dentro de los formatos a color más conocidos se encuentran HSI, HSV, YUV, YIQ y RGB. Siendo este último el más popular hoy en día.

El modelo RGB cuenta con tres canales donde cada uno de estos contiene 8 bits de información, por lo que cada píxel cuenta finalmente con 24 bits de información cada uno. Cada uno de los canales mencionados representa la intensidad de un color primario de este modelo, los que son rojo, verde y azul, ya que con estos tres colores principales se puede formar una amplia gama de colores secundarios mezclando los tres básicos en distintas proporciones.

Una vez que se tiene claridad del formato de píxel con el que se trabaja, se establece la resolución de la imagen, la que indica la cantidad de columnas y de filas de píxeles que existen dentro del plano.

Las resoluciones de imágenes vistas en este proyecto van desde las QQVGA (160x120 píxeles) hasta SVGA (800x600 píxeles) pero en la actualidad existen resoluciones de hasta 4096x3072 píxeles solo que para trabajar con dichas imágenes, se necesitan cámaras capaces de capturarlas y de poder computacional suficiente para procesarlas y analizarlas.

Al considerar estas imágenes como matrices, éstas pueden ser operadas fácilmente con otras matrices dentro de los algoritmos desarrollados, lo que facilita el procesamiento y análisis mediante

poder computacional. Gracias a esto se desarrollan procesos como filtros (Figura 2.8), operaciones morfológicas, geometría proyectiva, segmentación, descriptores, etc. De los últimos temas mencionados solo se utiliza la geometría proyectiva en la subsección 3.2.1.

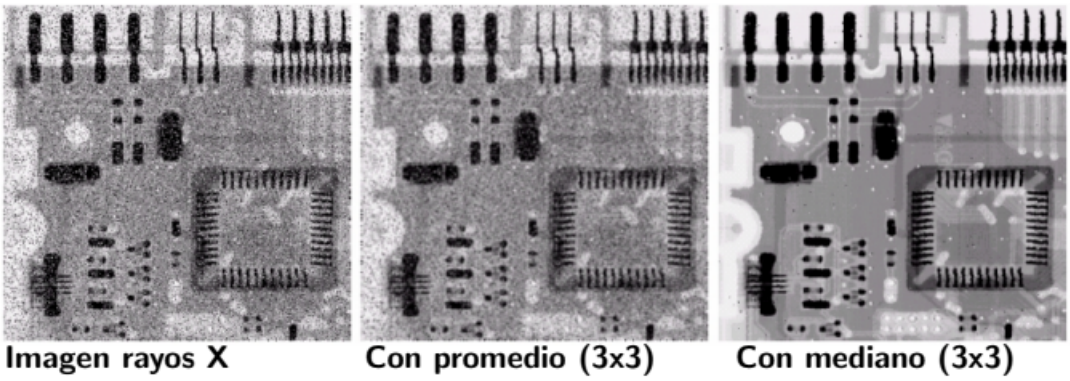

Figura 2.8: Ejemplo de distintos filtros aplicados sobre una imagen

Si bien el procesamiento y análisis de imágenes es un tema que sigue en constante evolución, para lograr desarrollar este proyecto se utilizaron los conceptos básicos recién mencionados para entender las distintas funciones utilizadas para el procesamiento de las imágenes y posterior análisis de los códigos QR.

Hoy en día este tipo de tecnología es muy utilizado en distintos ámbitos de la vida cotidiana. Un ejemplo de esto son los vehículos capaces de conducir sin intervención de un humano, estos cuentan con una gran cantidad de sensores y cámaras que aportan información para luego ser procesada y obtener como resultado un viaje seguro y confiable.

Dentro de esta misma área es que se utiliza esta tecnología en este proyecto, ya que se espera lograr el objetivo de predecir el cambio en la disposición de las vías mediante el procesamiento continuo de las imágenes capturadas por la cámara de la ESP32-CAM, la que va en movimiento y necesita de la información de su entorno (señaléticas) para un correcto funcionamiento.

2.8. Plataforma Experimental de Agentes Autónomos a escala

Esta plataforma experimental de bajo costo tiene sus inicios en el año 2018 cuando el estudiante Sebastián Oyanadel creó la primera versión para su memoria de título[15]. En ese entonces la plataforma solo contaba con sensores de distancia ultrasónicos y con Arduinos encargados del procesamiento.

Los componentes de cada agente en esta etapa son:

- Sensor de proximidad ultrasónico HC-SR04
- Motor de tren marca LEGO
- Circuito integrado puente H L293D
- Arduino UNO R3
- Batería de 9[V]

El funcionamiento de cada agente consistía en medir la distancia con respecto al agente antecesor y mediante una distancia de referencia y a un controlador PID programado en el Arduino, se lograba calcular una actuación para mantener dicha distancia de referencia entre los agentes. En la Figura 2.9 se muestra la estructura de estos primeros agentes.

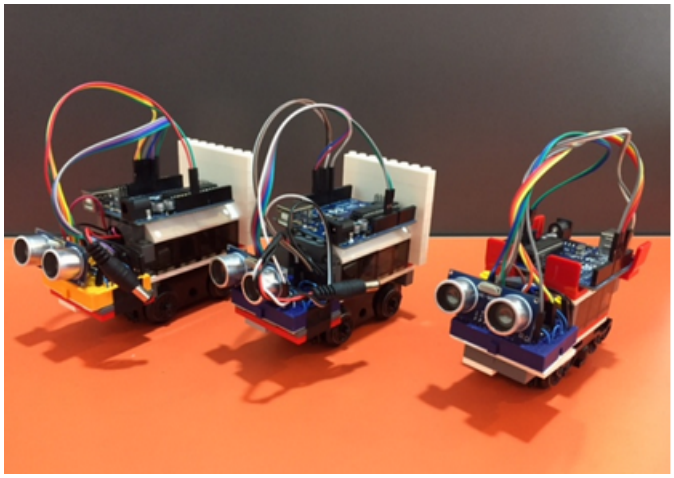

Figura 2.9: Estructura de los agentes en la primera fase de desarrollo

En los años posteriores, el AC3E siguió con el desarrollo de la plataforma agregando comunicación entre los agentes y nuevos sensores. De esta forma se buscaba mejorar el comportamiento

de la flota en movimiento y el tiempo de respuesta de los agentes.

Para llevar a cabo este objetivos, entre los primeros cambios que se realiza está el nuevo microcontrolador, el que pasó de ser un Arduino a una placa ESP32, ya que esta cuenta con mejor capacidad de procesamiento y características de comunicación inalámbrica, las que serán descritas con mayor detalle en la sección 2.1.

Además del nuevo microcontrolador, se agregó un nuevo sensor de distancia con mejor precisión y tiempo de respuesta. También, se agregó una cámara capaz de medir la velocidad del agente. Junto con esos dos sensores se logra obtener dos datos muy importantes, los que son velocidad del agente y distancia al agente que lo antecede.

La antigua estructura del agente estaba conformada por bloques de LEGO, por lo que se reemplazó por una estructura impresa en 3D, quedando el agente como se muestra en la Figura 2.10 y además se puede ver todos los componentes detallados en la Figura 2.11. Como se observa, esta nueva estructura cuenta con dos partes conectadas por un acople, esto debido a que el largo de este modelo es superior al anterior y esto dificulta que el agente vire correctamente en las trayectorias curvilíneas.

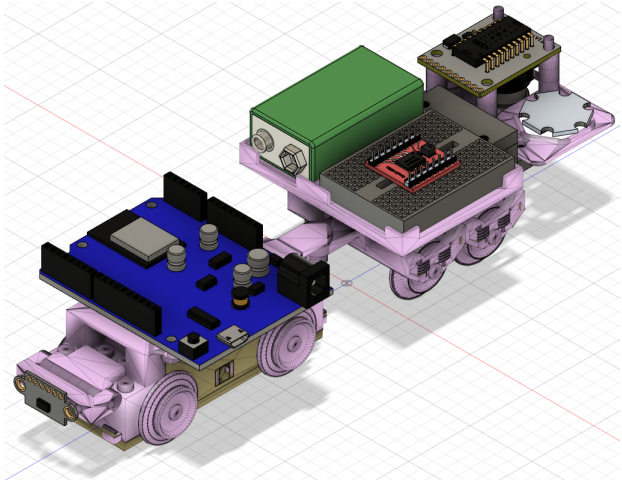

Figura 2.10: Modelo 3D de un agente

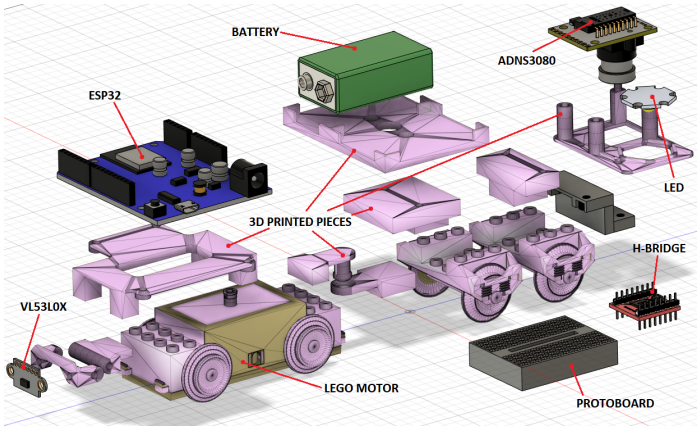

Figura 2.11: Modelo 3D de un agente con sus componentes rotulados

En esta nueva etapa existe traspaso de información entre los agentes y un broker mediante el protocolo MQTT (como se muestra en la Figura 2.12). Gracias a esto los agentes son capaces de enviar información como su velocidad o distancia con el agente antecesor, lo que permite no solo generar una respuesta por parte de cada agente con respecto al movimiento basado en toda la información de la flota, sino que además, cuenta con la capacidad de almacenar y procesar posteriormente esta información.

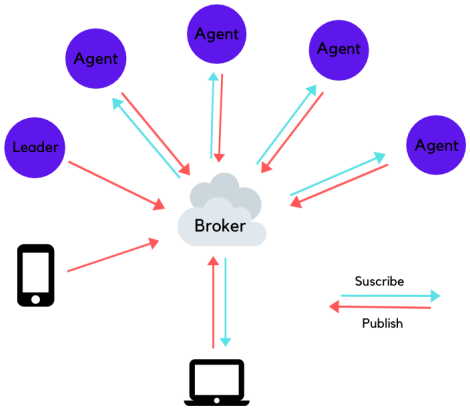

Figura 2.12: Esquema de traspaso de información en la plataforma mediante protocolo MQTT

Con todos estos cambios implementados, se pueden establecer nuevos modelos de control multi-agente sobre la plataforma. De esta forma se pueden observan más efectos de control y probar la influencia de la información al momento de generar una actuación.

Capítulo 3

Materiales y métodos

3.1. Línea de trabajo

En este capítulo se describirá el método de trabajo utilizado para lograr desarrollar la solución al problema planteado. También, se explicarán los principales pasos a seguir para que la solución logre cumplir los objetivos del proyecto.

Además, se presentarán las ventajas y desventajas de esta solución con respecto al método de trabajo.

La actual flota cuenta con microcontroladores ESP32 para su funcionamiento. Esta placa es de la empresa Espressif, que también produce las placas ESP32-CAM (Figura 2.2). Esta última cuenta con un procesador similar al utilizado en los agentes actualmente, pero además tiene integrada una cámara y un módulo de lectura y escritura para memorias SD.

Esta nueva placa se añade en la parte delantera del Agente Líder, de esta forma tiene visión sobre las vías que está por recorrer el agente.

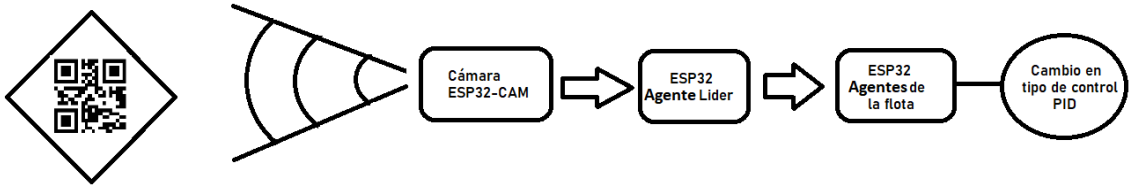

Figura 3.1: Esquema de aplicación de ESP32-CAM a la flota de agentes.

Teniendo esto en mente, se plantea establecer señaléticas en código QR al borde de las vías, las que indiquen un cambio en la disposición de estas. El proceso a seguir se muestra en el esquema de la Figura 3.1 donde los pasos de detección y traspaso de información son:

1. Detección de código QR por parte de la placa ESP32-CAM, la cual cuenta con un algoritmo de reconocimiento de códigos QR y de traducción de estos.

2. La placa ESP32-CAM envía señal mediante un pin digital al Agente Líder para indicar un cambio próximo en la disposición de las vías.

3. El Agente Líder recibe esta señal y a su vez envía un señal vía broadcast por ESP-NOW al resto de los agentes de la flota para indicarles que cambien el algoritmo de control.

4. Los agentes de la flota reciben esta señal y desactivan el factor del sensor de distancia en el cálculo de la actuación. De esta forma solo mantienen seguimiento de acuerdo a la velocidad del Agente Líder, lo que ayuda a un seguimiento parejo aunque se pierda visibilidad del agente que lo antecede en la cadena. El seguimiento en base a la distancia se reactivará una vez que los sensores vuelvan a detectar a sus predecesores, volviendo a la funcionalidad del esquema mostrado en la Figura 1.1.

Cabe destacar que esta nueva placa se añadiría en la zona delantera del Agente Líder, específicamente a la altura del sensor de distancia, quedando como se indica en la Figura 3.2, ya que así tiene visión sobre las vías que está por recorrer el agente y de las posibles señaléticas que indicarían un cambio en la disposición de las vías.

Figura 3.2: Modelo parte delantera del Agente Líder con ESP32-CAM integrada

Ventajas y Desventajas de esta solución

Según lo mencionado como posible solución, se encontraron las siguientes características a resaltar:

- La placa ESP32-CAM tiene un valor menor a los 10 USD$ por lo que es ideal para la plataforma que tiene como objetivo ser de bajo costo.
- El tamaño y peso de este microcontrolador, es reducido, por lo que no afecta al peso del agente ni a su trayectoria.
- Esta nueva placa se debe programar mediante la herramienta ESP-IDF. Esta herramienta es proporcionada por la empresa Espressif para una programación a más bajo nivel que simplemente el IDE de Arduino, por lo que ayuda a tener un mayor control sobre los recursos de la placa. Este tipo de herramienta requiere tiempo aprender a utilizarla y a instalarla, por lo que se debe estudiar con mayor profundidad en caso de seleccionar esta alternativa.
- Además de la nueva placa, los cambios en los códigos del resto de agentes es poco, ya que estos ya cuentan con sistemas de comunicación entre ellos. Por lo que agregar este nuevo módulo no afecta al actual funcionamiento de los agentes.
- Esta alternativa proporciona una predicción, por lo que los agentes tienen tiempo para prepararse para el nuevo tipo de control antes de entrar en el nuevo tipo de disposición de vías.

- Como ESP32-CAM y ESP32 son de la misma empresa, comparten el protocolo ESP-NOW para la comunicación, lo que facilita el traspaso de información entre ellos en caso de ser necesario.
- Finalmente, al agregar esta nueva placa, esta se encarga de todo el procesamiento de imágenes y de envío de la señal hacia el Agente Líder, es por esto que no afecta los tiempos de procesamiento que ya tiene la ESP32 en los agentes.

Para lograr desarrollar correctamente este proyecto, se establecen ciertos hitos importantes que se alcanzan durante el trabajo realizado, los cuales a su vez tienen relación con los objetivos planteados para este proyecto. Estos hitos son:

- Lograr función de reconocimiento y traducción de código QR con alto porcentaje de aciertos.
- Optimizar el tiempo de procesamiento para obtener una rápida respuesta.
- Lograr reconocimiento y traducción de código QR en movimiento.
- Utilizar la información de un código QR para obtener un efecto en la flota.

3.2. Desarrollo de código para ESP32-CAM y ESP32

A lo largo de este capítulo se describirán las distintas etapas de desarrollo por las cuales avanzó el proyecto.

Comenzando por la principal funcionalidad de reconocimiento y detección de códigos QR, la que es la base de la solución planteada.

Una vez que se logra el código base, se procede a agregar funcionalidades extra que favorecen las pruebas a realizar para verificar la efectividad de la solución. Entre estas funcionalidades se encuentran: (1) Almacenamiento de imágenes en microSD y (2) estimación de distancia hacia el código QR.

3.2.1. Desarrollo e implementación de código para ESP32-CAM

Se utilizó la librería instalada en el capítulo Configuración IDE Arduino como la base del código para posteriomente comenzar a programar las funciones necesarias para llevar a cabo el propósito final del código.

Las funciones de esta librería requieren establecer ciertos valores iniciales para configurar el funcionamiento de estas. Es por esto que los primeros valores definidos en la función Setup del código, son: Imagen de tamaño QVGA (320x240 píxeles), Control de Exposición desactivado y Nivel de exposición 300. Además, se crean los constructores base de esta misma librería, los que son: ESPino32CAM el encargado de obtener las imágenes de la cámara y ESPino32QRCode el encargado de detectar y descifrar las imágenes.

El objetivo de este desarrollo es lograr un algoritmo como se ejemplifica en la Figura 3.3

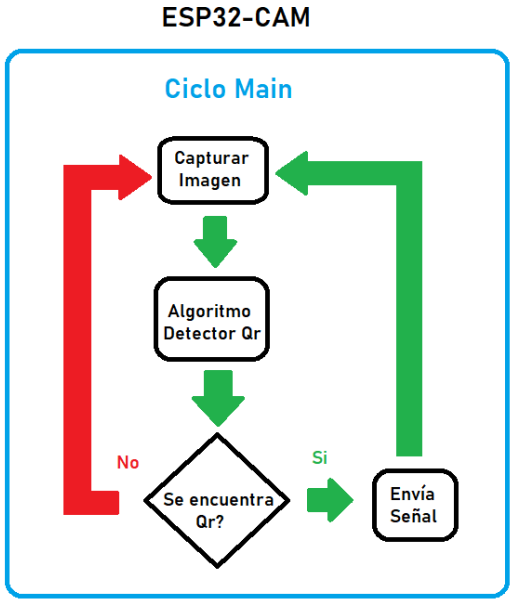

Figura 3.3: Diagrama de flujo de la ESP32-CAM

Desarrollo de función de reconocimiento y traducción de código QR

Una vez que se tiene la base de las configuraciones del código, se procede a crear la función Loop, la que es la función principal que se lleva a cabo continuamente durante el funcionamiento del microcontrolador. Aquí es donde se utiliza gran parte del código ejemplo $QR\_RECOGNIZE$ encontrado en `https://github.com/theeraphong0/ESP32Cam_QRCode_Scanner`. Esto debido a que este ejemplo cuenta con la funcionalidad de capturar imágenes, procesarlas continuamente en busca de códigos QR y además descifra los códigos QR detectados.

El código base genera varias salidas por puerto serial que logran entregar información con respecto a la imagen procesada. Esta información no es necesaria para la funcionalidad requerida para la flota, es por esto que se eliminan dichas lineas de código, así como también la comunicación serial para disminuir la carga en el procesador y aumentar su eficiencia.

Además, se configura el código para mostrar información mediante un led rojo y el flash de la cámara. De esta forma se encenderá el led rojo cada vez que el código procese una imagen y en esta imagen no se encuentre ningún código QR o si se detecta uno, pero la imagen no cuenta con una calidad mínima para descifrarlo. En cambio, el flash de la cámara se encenderá para indicar que se encontró un código QR en la imagen y se logró procesar y descifrar correctamente.

Desarrollo de función para almacenar imágenes en microSD

El código actual solo entrega como salida si se ha descifrado o no un código QR, es por esto que se ve la necesidad de buscar una forma para lograr ver las imágenes que el código procesa, así se puede determinar si estas imágenes están en buena calidad, tienen la correcta iluminación o hasta saber si realmente la cámara tiene completa visión del código. De este modo se puede ver como afectan ciertas configuraciones como el control de exposición o el nivel de exposición al momento de modificar estos parámetros y además, se puede analizar los resultados de pruebas en base a las imágenes capturadas.

Con esto en mente, se encuentra un código capaz de almacenar imágenes capturadas, pero este código requería reiniciar el microcontrolador cada vez que se almacenaba una imagen, ya que utilizar la microSD generaba conflictos con el funcionamiento de la cámara. Es por esto que se modificó dicho código para que al momento de almacenar una imagen, este libere la memoria utilizada, así se evita los conflictos con la cámara y permite seguir capturando imágenes.

Cabe destacar que esta función solo se aplica en una copia del código principal, ya que esto agrega tiempo de procesamiento. La finalidad de este código es ser utilizado para comprobar cambios en las imágenes, ver como afectan las configuraciones en las estas y obtener un registro de lo procesado en alguna prueba.

Desarrollo de función para estimar distancia hacia el código QR

Dentro de la librería Quirc, se encontró que la función encargada de detectar códigos QR, trabajaba con las esquinas del código, por lo que estos valores podían ser utilizados. Esto sumado a conocimientos previos en cuanto a geometría proyectiva en procesamiento de imágenes, se plantea la opción de estimar la distancia entre la cámara y el código QR al momento de capturar la imagen.

Si bien la librería Quirc trabaja con las coordenadas de las esquinas, la librería principal ESPIno32CAM no entrega acceso a estos datos. Es por ello que se debe modificar esta librería para que la estructura que retornan las funciones principales contengan dichas coordenadas. Estas coordenadas junto con la Ecuación 3.1, permiten estimar la distancia a la que se encuentra el código con un rango de aproximadamente 5 centímetros de error.

$$\text{Distancia al objeto}(mm) = \frac{f(mm) \times \text{altura real}(mm) \times \text{Altura de imagen (píxeles)}}{\text{altura del objeto (píxeles)} \times \text{altura del sensor}(mm)} \tag{3.1}$$

Como se observa en la fórmula, se necesitan datos intrínsecos de cámara y datos base de la plataforma. Entre estos últimos se encuentran la resolución de la imagen, el tamaño del código QR y la distancia focal de la cámara.

Durante el desarrollo de esta función se determina que la orientación del código QR afecta al momento de obtener sus coordenadas, lo que a su vez afecta al cálculo de la estimación de distancia. Es por esto que se generan 4 de casos de cálculo según la orientación del código QR, la que se obtiene comparando el orden en que las funciones de la librería quirc entrega las coordenadas de las esquinas del código.

Esta función así como la de almacenamiento en microSD, solo se aplica en el segundo código, ya que son funcionalidades extra que no son necesarias para la solución planteada, pero que pueden aportar información al momento de realizar pruebas específicas.

3.2.2. Conexión entre ESP32-CAM y Agente Líder

El microcontrolador ESP32 que se encuentra en el Agente Líder y la ESP32-CAM cuentan con las mismas funcionalidades en cuanto a conectividad inalámbrica, siendo estas: WiFi y Bluetooth. Es por esto que para establecer comunicación entre los dos microcontroladores y así entregar la información de un cambio en la disposición de las vías se cuenta con distintas opciones, ya que dentro de estas características físicas existen distintos protocolos de comunicación soportados.

Si bien las funcionalidades inalámbricas permiten transmitir mayor información, éstas agregan mayor tiempo de procesamiento, por lo que sabiendo que ambas placas se encuentran a escasos centímetros de distancia, se escoge como medio de comunicación un cable físico que conecta la ESP32-CAM con la ESP32. De esta forma se puede obtener de manera rápida y sin gasto de procesamiento, una señal indicadora de un cambio en la disposición de las vías.

Esta configuración logra entregar una señal desde un pin digital al momento de descifrar un código QR, por lo que, para lograr recibir esta señal, se propone un pseudocódigo de la ESP32, el que destine un pin digital como entrada. Una vez que se detecte un cambio en este pin, la ESP32 se debe encargar de enviar la señal al resto de la flota, ya que esta se comunica con los otros agentes mediante MQTT o ESP-NOW.

A continuación, se presenta un pseudocódigo con los cambios agregados al código base del Agente Lider para detenerse al momento de detectar una señal desde la ESP32-CAM.

```
void setup(){
        Serial.begin(9600);
        *Setup Pines*
        pinMode(17, INPUT); //Receptor señal desde ESP32-CAM
        *Setup WiFi*
}

void loop(){
        *Rutina normal del Agente*
        if (digitalRead(17)){
            u = 0;
            Serial.println("QR Detectado, deteniendo agente!");
        }
}
```

3.3. Pruebas a realizar

En este capítulo se describirán las pruebas planteadas a realizar para corroborar que se alcanzan los hitos mencionados anteriormente en la sección 3.1. Se establecen estas pruebas para demostrar el avance durante el proyecto y además asegurar un correcto desarrollo de la solución.

Estas pruebas a realizar consisten en verificar el porcentaje de efectividad de la solución en distintos escenarios. Además, se variaran distintas características de la cámara para encontrar el punto óptimo entre tiempo de procesamiento y porcentaje de efectividad.

3.3.1. Pruebas de algoritmo de detección de código QR con cámara estática

Una vez que se termina de desarrollar el código principal, se procede a realizar pruebas con este para lograr determinar la eficacia de dicho algoritmo frente a distintas circunstancias y de esta forma corroborar que se cumpla el primer hito establecido.

Para realizar esta prueba, se utilizan los siguientes componentes: ESP32-CAM, señalética con código QR y una regla. Estos se ordenan como se observa en la Figura 3.4.

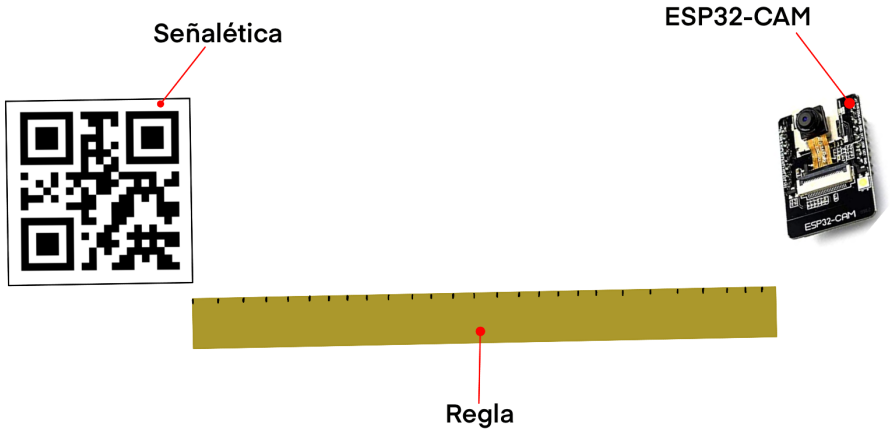

Figura 3.4: Esquema de prueba con ESP32-CAM estática a distintas distancias

Además, se establece una posición fija para la señalética con código QR mientras que la ESP32-CAM se mantiene estático a una distancia específica guiada por la regla. Esta distancia varía en 5 centímetros para cada prueba. De esta forma se puede determinar el mejor rango de funcionamiento del algoritmo.

Una vez que la ESP32-CAM y el código QR están en su posición, se capturan y procesan 10 imágenes, de las cuales se obtiene un porcentaje de acierto.

Posteriormente, cuando ya se tienen todos los porcentajes de acierto en todas las distancias planteadas entre en código y el microcontrolador, se procede a variar el tamaño del código QR. De esta forma se obtienen las tablas de resultados donde se puede apreciar el porcentaje de efectividad del código, basado en el tamaño del código QR y la distancia a la que se encuentra la ESP32-CAM con la señalética.

3.3.2. Pruebas de Nivel de Exposición

Dentro de las capacidades de la cámara y gracias a la librería ESP32Ino es posible modificar el nivel de exposición que se desea en una imagen. Si bien normalmente esta configuración viene en modo automático, como se desea tener una plataforma estable donde las condiciones (como la luz ambiente) siempre serán las mismas, se decide realizar una prueba para verificar el efecto de utilizar un nivel de exposición fijo.

Es por esto que primero se realiza una prueba para determinar si el nivel de exposición afecta al tiempo de procesamiento de la imagen. Por lo que se capturarán 10 imágenes con niveles de exposición Automático, 100 y 1000.

Luego se procede a determinar si el mejor nivel de exposición es 300 o 500, ya que estos valores son los que arrojan mejor claridad en las imágenes según las condiciones de la plataforma.

Para realizar esta segunda prueba se registrará el porcentaje de acierto con estos dos niveles de exposición variando la distancia hacia el código, pero manteniendo el tamaño del código en 16 cm de lado.

Esta prueba se realiza con los mismo elementos utilizados en la prueba anterior, por lo que se sigue el mismo diagrama de la Figura 3.4 para la medición de distancias entre la señalética y la ESP32-CAM.

3.3.3. Pruebas de impacto de la resolución de la imagen en el algoritmo

Durante el desarrollo del código principal y el estudio de la librería ESPIno32CAM, se encuentra la función capaz de configurar la resolución de la imagen obtenida desde la cámara.

Dentro las posibles resoluciones que soporta la ESP32-CAM se encuentran: QQVGA, QVGA, VGA, CIF y SVGA. Debido a la gran cantidad de resoluciones que son soportadas tanto por la cámara como por el algoritmo decodificador, se requiere revisar el impacto de la resolución tanto en el tiempo de procesamiento como en el porcentaje de aciertos del algoritmo.

Con lo anterior en mente, se realiza un experimento donde el código muestra el tiempo de procesamiento, la cámara se mantiene estática frente a un código QR y se modifica solo la resolución de la imagen en la configuración del código.

Con esta prueba se espera determinar la mejor resolución que minimice el procesamiento sin sacrificar el porcentaje de detección, de esta forma aporta al primer y segundo hito planteado.

3.3.4. Prueba de algoritmo de detección de código QR con cámara en movimiento

Como el objetivo del proyecto es que la solución implementada funcione en movimiento, la prueba más importante es esta donde se medirá el desempeño del algoritmo con el agente en movimiento a distintas velocidades. Es por esto que ahora se realizará esta prueba en la plataforma real, es decir, con el agente en las vías y con el código QR a un costado de las vías para indicar una curva.

Las velocidades se basarán en la actuación del agente, la que puede tomar valores desde 0 hasta 800, pero debido a la zona muerta del agente, se requiere un mínimo de 300 para poder avanzar. Esta velocidad fija se le asignará al agente mediante el smartphone utilizando el protocolo MQTT previamente configurado. En la Figura 3.5 se observa la interfaz de la aplicación utilizada para este propósito

Figura 3.5: Interfaz de aplicación IoTOnOff utilizada para enviar actuación a Agente Líder vía MQTT

La prueba consistes en 10 iteraciones con distintos tamaños de códigos QR para confirmar que la decisión tomada a partir de la prueba de cámara estática sea la correcta.

Tras completar 10 iteraciones, se aumentará en 100 la actuación para repetir la prueba y así sucesivamente hasta llegar a una actuación de 800.

La posición de cada elemento de esta prueba se muestra en un foto extraída de una de las pruebas realizadas. Esta foto se encuentra rotulada en la Figura 3.6. Donde además, se observa que el Agente Líder recorre la vía mientras que la señalética con código QR se posiciona en la curva. Por último, esta señalética se encuentra iluminada directamente con una fuente de luz externa para aumentar la calidad de la imágenes capturadas.

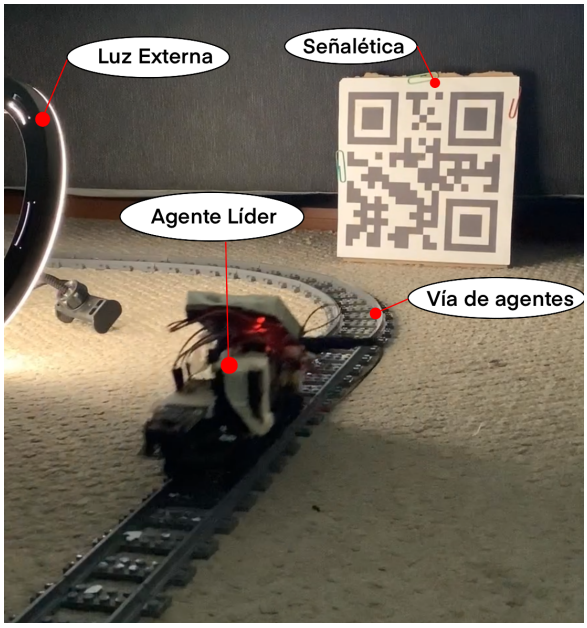

Figura 3.6: Esquema de prueba con Agente Líder en movimiento.

3.3.5. Prueba de correcta comunicación entre ESP32-CAM y Agente Líder

Como se mencionó anteriormente, para la comunicación entre el Agente Líder y la ESP32-CAM se escogió un cable que une un puerto de cada uno de los microcontroladores. Como ya se configuró la conexión en la sección 3.2.2, se comprueba que esta comunicación es correcta y que además la solución planteada funciona como se planteó desde un comienzo. Para esto se crea esta prueba donde el Agente Líder debe detenerse cada vez que la ESP32-CAM descifre un código QR.

Además, esto se probará con distintas actuaciones, desde 300 a 800. De esta forma se podrá apreciar los porcentajes de aciertos obtenidos en la prueba anterior, pero esta vez observados como un efecto en la flota.

Para la posición de cada elemento de esta prueba, se utiliza la misma configuración espacial mostrada en la Figura 3.6.

Finalmente, además de determinar el funcionamiento cuantitativamente, es decir, obteniendo un porcentaje de acierto, se obtendrá una apreciación cualitativa con respecto a la distancia hacia el código QR al momento en que se detiene el agente y a la reacción del sistema frente a la nueva señal.

3.3.6. Pruebas menores

Captura y almacenamiento de imágenes

Para comprobar que el código de almacenamiento de imágenes funciona, se procede a insertar una microSD recién formateada en la ESP32-CAM. Luego, se inicia una prueba con las mismas condiciones que las pruebas con cámara estática. Finalmente, luego de realizar 20 iteraciones, se comprueba que en la microSD estén almacenadas las imágenes capturadas y la categoría a la que pertenece cada una.

Tiempo que demora en almacenar imagen

Como se mencionó anteriormente, existen dos códigos utilizados durante el desarrollo del proyecto, donde uno es la copia del otro pero con características adicionales. Es por esto que se ve la necesidad de medir el tiempo de procesamiento que añaden estas capacidades adicionales.

Con esto en mente se genera una variable para medir el tiempo entre cada iteración del código, obteniendo de esta forma la diferencia entre ambos algoritmos.

Finalmente, para cada código se obtienen 10 iteraciones con sus tiempos de procesamiento, los cuales se promedian para obtener un resultado final.

Escala de grises

La librería ESPIno32CAM cuenta con una función que convierte una imagen RGB a escala de grises para luego esta ser procesada por la librería quirc. Pero desde hace unos años que existe un debate dentro del área del procesamiento de imágenes con respecto a la transformación de una imagen a escala de grises, ya que muchas veces si no se hace de manera correcta, se pierde detalles de la imagen.

Es por esto que al revisar dicha función en la librería se encontró que estaba utilizando un algoritmo de promedio con respecto al menor y al mayor valor de los canales de la imagen.

Se decide poner a prueba este algoritmo en comparación con una fórmula de ponderación de canales (Ecuación 3.2). Para esto se capturan 10 imágenes con la cámara estática con cada algoritmo para determinar cuál detecta con mejor efectividad los códigos QR.

$$\text{Canal Gris} = 0{,}299 * \text{Rojo} + 0{,}587 * \text{Verde} + 0{,}144 * \text{ Azul} \tag{3.2}$$

3.4. Configuraciones Realizadas

En este capítulo se entregarán las indicaciones necesarias para configurar y conectar el hardware utilizado durante el desarrollo de este proyecto. De esta forma, es posible replicar su funcionamiento en caso de ser necesario.

Además, se detalla como ensamblar la solución desarrollada en la plataforma base. Esto consiste en posicionar las señaléticas con código QR y como ubicar la ESP32-CAM en el Agente Líder.

3.4.1. Configuración IDE Arduino

Para lograr trabajar con el código de la ESP32-CAM a partir de la interfaz de programación de Arduino, se requirió realizar ciertos ajustes en dicha interfaz. Entre estos cambios se encuentran: Velocidad de carga, frecuencia de Flash, tamaño de Flash, esquema de partición y habilitar PSRAM.

Además, se requirió instalar la librería ESPIno32CAM, la que cuenta con las funciones necesarias para capturar imágenes, identificar códigos QR y decodificar códigos QR. Esta librería a su vez cuenta con la librería QUIRC como base en su funcionamiento.

A continuación se detallarán las opciones de Arduino IDE que se deben tener seleccionadas para lograr cargar códigos correctamente en la ESP32-CAM, las cuales se cambian en la pestaña $Tools$ en la barra de opciones.

- Board : ESP32 Dev Module
- CPU Frequency : 240 MHz
- Core Debug Level : None

- Flash Frequency : 80 MHz
- Flash Mode : QIO
- Flash Size : 4 MB
- PSRAM : Enabled
- Partition Scheme : Huge APP
- Upload Speed : 115200

En la Figura 3.7 se muestra una sección del menú **Herramientas** donde se aprecia como debería verse dicha sección luego de aplicar las configuraciones recién mencionadas.

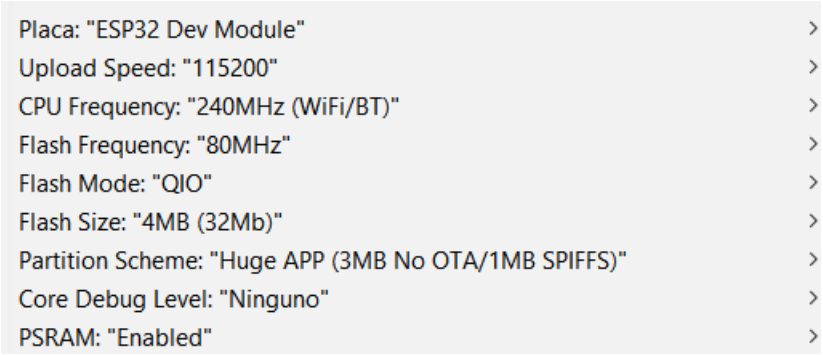

Figura 3.7: Sección de menú Herramientas luego de aplicar las configuraciones necesarias para el funcionamiento con ESP32-CAM

3.4.2. Conexiones de ESP32-CAM

Para poder programar la ESP32-CAM se requiere un Arduino, el que cumple la función de intermediario para la carga de código. Esto debido a que la ESP32-CAM no cuenta con entrada USB, por lo que se debe conectar mediante los cables TX y RX al Arduino y este a su vez se conecta al computador, tal como se muestra en la Figura 3.8.

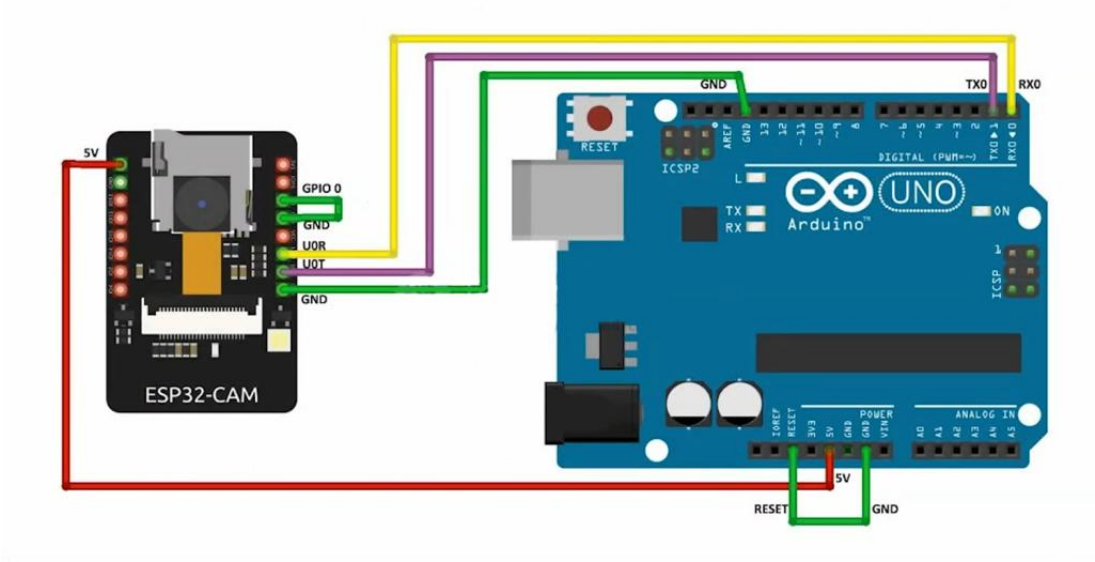

Figura 3.8: Diagrama de conexión para flashear la ESP32-CAM con un Arduino UNO

Además, para cargar el código correctamente, la ESP32-CAM debe estar en modo FLASH, lo cual se consigue conectando el pin 0 a tierra, tal como se muestra en la Figura 3.8-

3.4.3. Comunicación Inalámbrica

Si bien la ESP32-CAM cuenta con conectividad inalámbrica tipo WiFi y Bluetooth, estas no serán ocupadas durante el desarrollo del proyecto. Es por esto que las comunicaciones inalámbricas que se deben configurar son las de las placas ESP32 que poseen los agentes de la flota.

Esto se debe a que para poder realizar las pruebas con el agente en movimiento, el agente debe estar conectado vía WiFi para poder utilizar el protocolo MQTT tal como se indica en la Figura 2.12.

Para poder configurar esta conectividad se deben seguir los siguientes pasos:

- Instalar e inicializar Broker Mosquitto en computador o Raspberry. Este Broker debe tener como configuración el puerto 9001 y permitido el acceso mediante WebSocket.
- Instalar la aplicación IoT OnOff en nuestro smartphone
- Conectar la ESP32, el dispositivo del Broker y el smartphone a la misma red WiFi.
- Configurar los tópicos con los cual se enviará y recibirá información con respecto a la actuación del agente.

Siguiendo estos pasos, el usuario debería ser capaz de generar una velocidad en el Agente Líder enviando la actuación deseada desde el smartphone.

3.4.4. Crear señaléticas con códigos QR a utilizar

Para lograr resolver el primer objetivo planteado con respecto a la creación e implementación de señaléticas, se establecen dos posibles soluciones, las cuales se probaron durante el desarrollo de este trabajo. En ambas soluciones se genera el código QR a partir de página web `https://www.codigos-qr.com/generador-de-codigos-qr/`. La diferencia se encuentra en que una opción se basa en mostrar este código directo desde la pantalla de un iPad, de esta forma se puede ajustar la luz de la pantalla durante las pruebas. La otra solución corresponde a imprimir estos códigos QR e iluminarlos directamente con una fuente de luz externa (Como se muestra en la Figura 3.9).

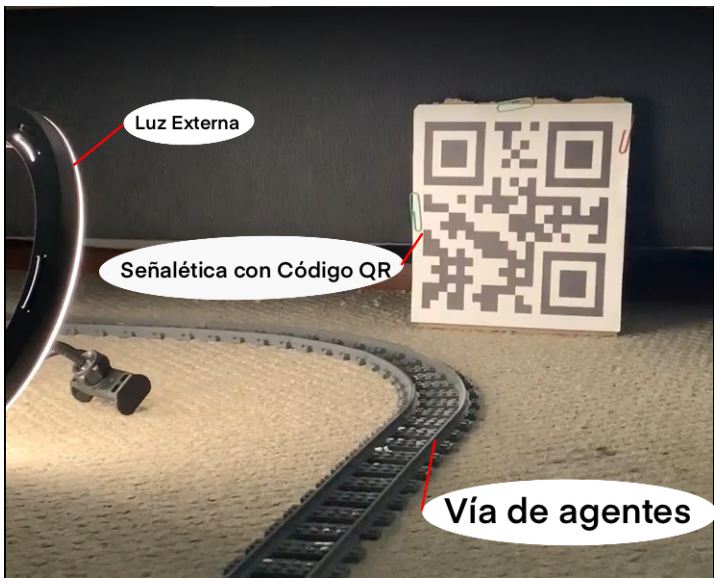

Figura 3.9: Ejemplo de posición de señalética con código QR utilizando una luz externa.

Si bien los códigos QR se pueden decodificar en cualquier orientación gracias a sus indicadores de posición codificados como información dentro de este. Al momento de agregar esta señalética en la plataforma experimental, siempre un lado del código QR quedará paralelo al piso, es decir, el código QR solo se posicionará con 0, 90, 180 o 270 grados de rotación.

3.4.5. Ensamblaje de ESP32-CAM en el Agente Líder

Desde que se inician las primeras pruebas, la ESP32-CAM se posiciona en la parte delantera del agente, ya que de esta forma se realizarían las pruebas con el ángulo de visión real al que sería sometido finalmente. Para llevar a cabo esto, se debió modificar el agente, ya que anteriormente todos los agentes de la flota eran iguales, los que cuentan con un sensor de distancia en su parte delantera, pero para facilitar el posicionamiento de la ESP32-CAM, este sensor se retiró del Agente Líder.

Además, para poder energizar la ESP32-CAM, se agregó un banco de 3 pilas AA y una batería de 9V. El banco de pilas se encarga de alimentar la ESP32-CAM, mientras que la nueva batería de 9V se encarga de alimentar el motor. Anteriormente una sola batería alimentaba a la ESP32 y al motor, pero durante las pruebas se determinó que esta batería no era suficientemente estable alimentando ambos módulos, por lo que finalmente el Agente Líder cuenta con 2 Baterías de 9V y un banco de 3 pilas AA.

Si bien durante el desarrollo de las pruebas la ESP32-CAM se encuentra montada en una miniprotoboard y esta a su vez ensamblada en el agente mediante un hilo, la posición es lo suficientemente estable como para realizar las pruebas y tener un buen ángulo de visión, por lo que, si se desease montar de manera fija en este punto, se recomienda una pieza de impresión 3D capaz de mantener la ESP32-CAM estable.

Finalmente, al Agente Líder se le logra acoplar la ESP32-CAM sin que esta impida o dificulte el movimiento del agente, ya que el microcontrolador se posiciona en la parte delantera como se observa en la Figura 3.10.

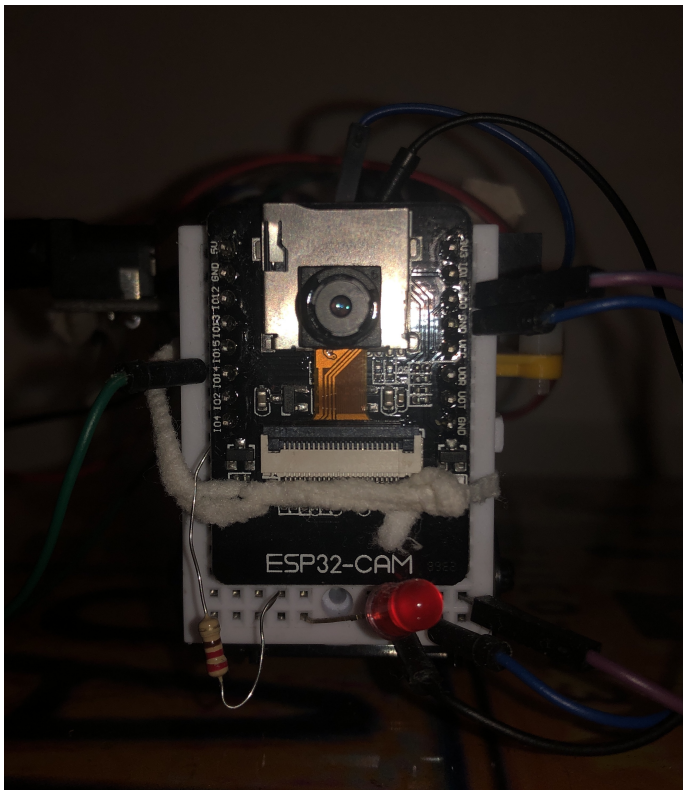

Figura 3.10: Agente Líder con ESP32-CAM acoplada en la zona delantera.

Capítulo 4

Resultados

A lo largo de este capítulo se expondrán los resultados obtenidos en todas las pruebas expuestas en la sección 3.3. Además de estos resultados, en cada una de las pruebas se detalla como afecta lo obtenido en la solución final, es decir, qué aporta esto en las configuraciones finalmente escogidas.

También se revisará como afecta el movimiento del agente en la captura de imágenes, ya que las pruebas permiten obtener la eficacia de la solución en movimiento y con la cámara fija, lo que permite contrastar estos resultados.

Por último, se revisa la eficacia de las funciones adicionales como el almacenamiento de imágenes en la microSD y la estimación de distancia entre la cámara y la señalética. Esto para determinar la utilidad que pueden tener dentro de esta solución o en proyectos futuros.

4.1. Resultado de pruebas de algoritmo de detección de código QR con cámara estática

Las primeras pruebas de este tipo se realizan con el código QR mostrado en una pantalla de un iPad. Luego de ciertas iteraciones, se concluye que es mejor hacerlo de forma que el código QR esté impreso en papel e iluminado con una fuente externa de luz. Esto debido a que no se generaba el contraste necesario debido a la iluminación que tiene la pantalla del dispositivo utilizado.

Gracias a los resultados que se observan en las Tablas A.1 y A.2 en la sección A del apéndice, se determina que el código QR se puede detectar y descifrar desde mayor distancia mientras mayor sea el tamaño de este. Pero a su vez, se pierde rango a corta distancia, ya que la cámara no logra capturar el código completo cuando se encuentra a escasos centímetros. Con esto en mente, se escoge como tamaño final para los códigos QR, un tamaño de 16 cm de lado, ya que esto permite un rango de trabajo entre 30 y 80 centímetros. Este rango se escoge debido a que es preferible que el agente detecte el código QR a mayor distancia, de esta forma tiene mayor tiempo para realizar cambios en la configuración de la flota para afrontar lo descrito en la señalética.

En la tabla 4.1 se muestran los porcentajes de acierto para un código QR de 16 centímetros de lado, el cual fue escogido como el óptimo para este proyecto. El resto de resultados de esta prueba se puede revisar en las Tablas A.1, A.2 y A.3 en la sección A del apéndice.

Tamaño lado QR(Cm)	Distancia(Cm)	Porcentaje Eficiencia(%)
16	25	0
16	30	100
16	35	100
16	40	100
16	45	100
16	50	100
16	55	100
16	60	100
16	65	70
16	70	60
16	75	30
16	80	20

Tabla 4.1: Resultados de Prueba estática con código QR impreso de 16 cm de lado

Se observa que cuando la cámara está estática, tiene muy buen rango de trabajo, ya que desde los 30 centímetros hasta los 60 cuenta con una efectividad del 100 % y si bien desde los 60 centímetros hasta los 80 centímetros decae considerablemente la eficacia, aun tiene más del 60 % de eficacia hasta los 70 centímetros, por lo que sigue siendo un buen resultado.

Además, se determina que la cota inferior del rango, los 30 centímetros, generan un rango de tiempo de reacción hasta el momento en que el agente logre llegar hasta el código QR, por lo que es el mínimo de distancia que tendrá el agente para efectuar un cambio en la configuración de la flota. Es por esto que si el agente detecta un código QR, ya se sabe que el código QR no va a estar a menos de 30 centímetros, lo cual dependiendo de su velocidad, genera un tiempo donde el agente

puede aplicar los cambios de configuración. Un ejemplo de esto: si se detecta un código QR y el agente se mueve a 10 centímetros por segundo, el mínimo de tiempo que tiene el agente son 3 segundos, donde puede enviarle las señales correspondientes al resto de agentes de la flota.

4.2. Resultados de prueba de Nivel de Exposición

A partir de la primera prueba con respecto al nivel de exposición, se obtiene la Tabla 4.2. Aquí se puede observar que una diferencia de nivel de exposición no afecta al tiempo de procesamiento, por lo que solo afecta a la calidad de la imagen.

Tipo de Exposición	Tiempo sin Detectar (ms)	Tiempo Descifrando (ms)
Automática	237	1763
100	241	1761
1000	238	1753

Tabla 4.2: Tiempos de procesamiento del algoritmo según el nivel de exposición

Esto se tiene con consideración al momento de escoger un nivel de exposición final, ya que si el tiempo de captura de imagen fuese dependiente del nivel de exposición, se debería escoger un punto donde no se sacrifique tiempo de procesamiento y exista una buena calidad de imagen, ya que se recuerda que se busca una rápida respuesta de la solución. Pero debido a los resultados recién mostrados, no es necesario esto, si no que solo buscar el mejor nivel de exposición que logre una calidad de imagen óptima.

Basándose en los resultados recién mostrados, se procede a la segunda prueba donde a partir de un análisis de las imágenes procesadas, se establece que los mejores niveles de exposición son 300 y 500, ya que son los que generan una mayor cantidad de casos descifrados correctamente.

Aquí se obtienen los resultados a distintas distancias comparando los niveles 300 y 500, donde se capturan imágenes y se almacenan en la microSD, esto permite saber la cantidad de imágenes que se descifran, las que solo se detecta el código QR y las imágenes en las que no se logra procesar nada. A partir de esto se obtienen los resultados reflejados en la tabla 4.3.

Nivel de Exposición	Tamaño lado QR	Distancia	Porcentaje Eficiencia
300	16	25	0
300	16	30	0
300	16	35	100
300	16	40	100
300	16	45	100
300	16	50	100
300	16	55	90
300	16	60	90
300	16	65	100
300	16	70	100
300	16	75	100
300	16	80	100
300	16	85	70
300	16	90	100
500	16	25	0
500	16	30	0
500	16	35	100
500	16	40	100
500	16	45	100
500	16	50	100
500	16	55	80
500	16	60	100
500	16	65	90
500	16	70	100
500	16	75	70
500	16	80	100
500	16	85	70
500	16	90	60

Tabla 4.3: Resultados de prueba con cámara estática variando el nivel de exposición

Esta prueba solo se realizó con un tamaño de código QR, ya que anteriormente en la subsección 4.1 se establece que el tamaño 16 centímetros es el que genera el mejor rango de trabajo.

Basándose en los resultados de estas pruebas es que finalmente se decide establecer el nivel de exposición en 300, ya que este no afecta el tiempo de procesamiento del algoritmo y además ayuda a mejorar la calidad de la imagen, lo que por consecuencia mejora el porcentaje de aciertos del algoritmo desarrollado.

Cabe destacar que este nivel escogido solo es válido para el entorno donde se realizaron las pruebas, ya que se tenía un nivel de luz ambiental constante. Es por esto que en caso de utilizar el algoritmo en un ambiente distinto, se debe volver a realizar un ajuste con respecto a la cantidad de

luz ambiental en el nuevo sitio de pruebas.

4.3. Resultados de pruebas de impacto de la resolución de la imagen en el algoritmo

Esta prueba se plantea debido a que mientras se realizaban las primeras pruebas del algoritmo, se probaron distintas resoluciones, donde se apreciaba cualitativamente un retardo en la obtención de un resultado cuando la resolución de la imagen era mayor. Es por esto que se diseñó esta prueba para detectar el tiempo exacto que aumentaba debido a este factor.

Lo esperado antes de iniciar la prueba era que los tiempos aumentaran proporcionalmente a la resolución, en vista de que una mayor resolución indica una mayor cantidad de píxeles, los que luego deben ser procesados por el algoritmo.

Con esta base de pensamiento, se realizó este experimento y se obtuvo que la mejor resolución para trabajar es la QVGA. La mejor competencia que tiene con respecto a tiempo de procesamiento es la resolución QQVGA, pero tiene un porcentaje de aciertos muy bajo, mientras que QVGA cuenta con 100 % de acierto en lograr descifrar en todas las distancias probadas con un código QR de lado 14 centímetros. Esto se puede ver en la Tabla 4.4 que compara ambas resoluciones a distintas distancias con el mismo tamaño de código QR. (Los datos completos se encuentran en las Tablas A.4 y A.5 en el apéndice A)

Resolución	Tamaño lado QR	Distancia	Porcentaje Eficiencia
QVGA	14	15	0
QVGA	14	20	0
QVGA	14	25	100
QVGA	14	30	100
QVGA	14	35	100
QVGA	14	40	100
QVGA	14	45	100
QVGA	14	50	100

Tabla 4.4: Extracto de resultados de Prueba con cámara estática con resolución QVGA

Se puede apreciar que esta prueba se realizó con un código QR de lado 14 centímetros, esto en virtud de que esta fue una de las primeras pruebas que se realizó, ya que se consideraba importante establecer una resolución base para la que se realizaran el resto de pruebas debido al impacto que

tiene esta en los tiempos de procesamiento del algoritmo.

Además de esto, se obtuvo un tiempo de procesamiento promedio para cada resolución, donde se consideraron los tiempos donde no se detecta ningún código QR y también los casos contrarios donde si se detecta y existe mayor procesamiento. Obteniendo los siguientes resultados:

Resolución	Tiempo sin Descifrar (ms)	Tiempo Descifrando (ms)
QQVGA	80	1651
QVGA	470	2025
CIF	750	2390
VGA	1890	3540
SVGA	3295	4800

Tabla 4.5: Resultados de prueba de tiempo de procesamiento según resolución

Se observa que los tiempo donde no se descifra un código QR son bastante bajos, por lo que permite capturar imágenes seguidas. Aquí es donde se destacan las resoluciones QQVGA y QVGA, ya que permiten capturar más de 2 imágenes por segundo, lo que es bastante bueno para la solución, porque el agente se encuentra en movimiento, por lo que necesita la mayor captura de imágenes posibles por segundo.

Pese a esto, se nota que los tiempo cuando sí existe descifrado del código QR en ningún caso baja de 1 segundo, por lo que esto implicará que al momento de capturar una imagen que contenga un código QR, el agente no recibirá la señal en menos de 1 segundo, lo que genera un tiempo muerto donde no se están capturando imágenes y además el agente aún no recibe la señal con respecto al código QR. Este tiempo muerto donde el agente seguirá avanzando, está considerado al momento de elegir el rango de trabajo del algoritmo, porque se recuerda que el mínimo de distancia entre el agente y el código QR al momento de capturar la imagen es de 30 centímetros, donde el agente se pude mover mientras espera la señal con respecto a la información descifrada del código QR.

Como se mencionó, este tiempo de procesamiento, al ser de casi 2 segundos cuando se detecta un código QR, trae errores al momento de aplicar la solución, debido a que mientras ocurre el procesamiento, no se pueden capturar más imágenes. Un ejemplo de esto es si el agente se mueve a 10 centímetros por segundo y detecta un código QR a 40 centímetros de distancia. Al momento de intentar descifrarlo, se detecta un error en la imagen, por ende no se logra descifrar el código QR. Esto significa que los casi 2 segundos de procesamiento se perdieron y cuando se captura la

siguiente foto, el agente ya avanzó 20 centímetros, por ese motivo la señalética ahora se encuentra fuera del rango de trabajo.

Finalmente, considerando lo mencionado y ambos resultados, se escoge la resolución QVGA como óptima, ya que minimiza el tiempo de procesamiento dejándolo en aproximadamente 2 segundos, pero a su vez no disminuye el porcentaje de aciertos con respecto al resto de resoluciones. Se espera que estos dos segundos generen errores al momento de necesitar capturar imágenes seguidas luego de un intento de descifrado, pero a su vez la capacidad de capturar 2 imágenes por segundo ayudará a detectar los códigos QR con mayor facilidad.

4.4. Resultados de prueba con cámara en movimiento

Si bien en los resultados expuesto en 4.1 ya se escoge el tamaño de los códigos QR, en este caso se prueba con el resto de opciones debido a que al necesitarse una captura de imagen en movimiento, la forma y tamaño puede afectar la calidad de las imágenes y por consecuencia, al porcentaje de aciertos.

Luego de realizar las 10 iteraciones para cada tamaño de código QR y para cada velocidad, se obtiene que la mejor tasa de acierto se consigue con un código QR de 16 centímetros de lado, ya que a una actuación de 300 es capaz de conseguir hasta un 90 % de códigos QR descifrados exitosos tal como se muestra en el extracto de los resultado en la Tabla 4.6. (Los datos completos se encuentran en la Tabla A.6 en el apéndice A)

Tamaño de lado del QR (cm)	Actuación	Porcentaje de aciertos (%)
16	300	90
16	400	70
16	500	40
16	600	30
16	700	0
16	800	0

Tabla 4.6: Extracto de los resultados de prueba con agente en movimiento

Se puede observar que a grandes actuaciones el código no es capaz de descifrar los códigos QR, por lo que una velocidad óptima para el trabajo de la plataforma y un correcto funcionamiento del algoritmo, sería una actuación entre 300 y 400. Uno de los motivos de esto se debe al tiempo de procesamiento que requiere una imagen al usar la resolución QVGA, porque se tarda 2 segundos

en procesar una imagen donde se detecta un código QR. Esto genera que a grandes velocidades, solo pueda capturar una imagen donde se intente descifrar un código QR, pero si esa imagen no cuenta con una correcta calidad, el código QR no se puede descifrar y al momento de capturar otra imagen, debido a la gran velocidad que lleva el agente, este se sale del rango de trabajo del algoritmo. Lo que finalmente resulta en que no se entrega la información de la señalética.

Durante esta prueba, el agente debía volver a su posición inicial, por lo que recorría el mismo trayecto pero en sentido contrario y al pasar a grandes velocidades en sentido contrario, en ciertos casos detectaba el código QR, porque el algoritmo no se detenía en ningún momento. Esto se considerará como una opción a revisar como trabajo a futuro, ya que se notó cualitativamente una mayor cantidad de aciertos a velocidades altas en reversa que en sentido correcto.

4.5. Resultados de prueba de correcta comunicación entre ESP32-CAM y Agente Líder

El fin de esta prueba es comprobar que la solución logra entregar información desde la ESP32-CAM hacia la ESP32 del Agente Líder, es por esto que el objetivo no es verificar el porcentaje de aciertos en la comunicación, debido a que al algoritmo programado en el Agente Líder, estará siempre pendiente de la entrada digital a la cual se conecta la ESP32-CAM.

Durante la realización de esta prueba, se grabaron vídeos donde se observan los resultados. En estos vídeos se aprecian distintas actuaciones transmitidas a través del smartphone hacia el Agente Líder y donde además el agente se detiene a distintas distancias de la señalética. Este tipo de prueba es cualitativo, porque se determina que la solución cumple cuando se logra detener antes de que el agente vire completamente en la curva, ya que el problema de la plataforma base ocurre cuando el siguiente agente de la flota pierde al Agente Líder.

Los vídeos de todas las pruebas realizadas se pueden encontrar en en el siguiente link: `https://drive.google.com/drive/folders/1e1SSi4u2JKrBQXL0S9cTxPH0UqzZ3E5u?usp=sharing`. En dicho link, se encuentran dos videos donde la actuación del Agente Líder es 300 y éste detecta correctamente el código QR. Además, hay dos videos donde la actuación es 400, pero en uno de los casos sí se descifra el código QR y en el otro el agente no lo logra detectar.

Gracias a esta prueba se puede observar que hay situaciones donde la solución detecta el cambio en la disposición de las vías y logra frenar hasta 30 centímetros antes de que este suceda, pero

también hay casos donde se detiene a pocos centímetros pasado el código QR, lo que ya se considera que el Agente Líder está girando, pero aún está a tiempo de enviar la información al resto de la flota para que no sufra los efectos de la pérdida de visión.

Figura 4.1: Código QR para ver video de demostración de la solución implementada.

El código QR de la Figura 4.1 redirige a un video de esta prueba, donde se puede observar que el agente se detiene al momento de detectar un código QR, lo que cumple con lo establecido como objetivo desde un comienzo, por lo que ese caso se considera completamente exitoso.

Si bien los resultados de estas pruebas no son todos exitosos, se puede apreciar claramente que hay ocasiones donde si logra identificar la señalética, decodificarla y luego enviar la señal al Agente Líder, por lo que si esta solución se aplica en la plataforma mejorará el comportamiento de esta aunque no sea en todos los casos, por lo que la solución elaborada logra aportar con cierto porcentaje de efectividad.

4.6. Resultados de pruebas menores

Captura y almacenamiento de imágenes

En estas pruebas se obtienen imágenes como las de las Figuras 4.2, 4.3 y 4.4, donde 4.2 es una imagen en la que el algoritmo no detecta ningún código QR. En 4.3 el programa es capaz de detectar que existe un código QR en la imagen, pero no es posible decodificarlo debido a la calidad de éste. Finalmente, en 4.4 se logra identificar y descifrar el código QR correctamente.

Figura 4.2: Imagen donde no se detecta ningún código QR

Figura 4.3: Imagen donde se detecta código QR, pero no se descifra

Figura 4.4: Imagen donde se logra descifrar el código QR

Estos ejemplos mostrados en las figuras se utilizaron para observar la calidad de captura de las imágenes en la prueba de nivel de exposición, ya que además de poder observar las imágenes que procesaba el algoritmo, era capaz de entregar la categoría de la imagen mediante el nombre de la foto. Estas categorías son: imagen con código QR descifrado, imagen solo con código QR detectado e imagen donde nada ha sido detectado. Por ello simplemente se realiza la prueba y luego se revisa la microSD para verificar la cantidad de imágenes correspondiente a cada categoría.

Este tipo de imágenes también se utiliza en las pruebas en movimiento cuando se detectaba algún error en el algoritmo, ya que permitía observar las imágenes que capturaba la cámara mientras el agente se movía y así entender el posible fallo en la prueba.

Finalmente, tener la posibilidad de almacenar las imágenes en la microSD puede aportar a futuros trabajos, porque se pueden procesar en paralelo con otro microcontrolador o enviar vía inalámbrica a un computador el cual se encargue del procesamiento. Además, estas fotos quedan como registro de lo que logra capturar la cámara a lo largo del recorrido establecido, por lo que se pueden analizar posteriormente para establecer conclusiones en base a lo observado.

Tiempo que demora en almacenar imagen

A partir de esta pequeña prueba se obtiene que el segundo código (capaz de almacenar imágenes) cuenta con un tiempo extra de aproximadamente 100 ms, por lo que se concluye este tiempo es ocupado en el almacenamiento.

Si bien el tiempo que se obtiene como tiempo promedio, es aproximadamente un 20 % de lo que se demora normalmente en capturar una imagen con resolución QVGA. Este tiempo solo se usará cuando se deseen hacer pruebas de nuevos tipos de algoritmos o de distintas configuraciones de la cámara, por lo que no será un problema para la solución final. Esto ayudará al momento del desarrollo para lograr observar lo que la cámara captura con los cambios realizados y de esta forma determinar si estos cambios son beneficiosos para la solución.

Escala de grises

Para esta prueba también se utilizó el almacenado de imágenes en la microSD, ya que esto permite ver cualitativamente la calidad de las imágenes y además categoriza si estas contienen o no un código QR y si éste logró ser descifrado.

Una vez que se probó cada una de las fórmulas establecidas, se determinó que la ecuación ponderada que se planteó (3.2) logra un 90 % de efectividad a 60 centímetros de distancia hacia el código QR en cuanto a la detección y descifrado de estos, ya que se define mejor el contraste entre negro y blanco en la imagen. Mientras que el promedio ponderado de los canales que se utilizaba anteriormente en el algoritmo, solo genera un 70 % de efectividad en las mismas condiciones de la prueba.

Debido a que este tipo de transformación ocurre dentro de la librería quirc, no se puede extraer la imagen con el cambio a escala de grises aplicado para verificar la calidad de la transformación, pero se puede aplicar estos dos tipos de transformaciones a la una imagen de ejemplo como se muestra en la Figura 4.5.

Figura 4.5: Imagen de ejemplo original

A esta imagen se le aplica el cambio de RGB a escala de grises utilizando la ecuación ponderada y la ecuación de promedio, obteniendo los resultados de las Figuras 4.6 y 4.7 respectivamente. Se puede observar que la imagen procesada con la ecuación de los canales ponderados, logra una mejor calidad y contraste, mientras que en la imagen donde solo se ponderan los canales, no se logra un buen contraste entre blancos y negros, por lo que en el algoritmo final se dificultaría el descifrado del código QR.

Es por esto que para la solución final se decide establecer la Ecuación 3.2 como la indicada para la correcta transformación de las imágenes de escala RGB a escala de grises. Este paso es muy importante en la detección y decodificación, ya que es el paso inicial para la posterior binarización del código QR, lo que permite finalmente diferenciar los módulos del código QR al momento del descifrado.

Figura 4.6: Imagen de ejemplo con conversión a escala de grises a través de ecuación ponderada.

Figura 4.7: Imagen de ejemplo con conversión a escala de grises a través de ecuación promediada.

Capítulo 5

Conclusiones y Trabajo a Futuro

En este capítulo se presentarán las conclusiones extraídas a partir de todo el trabajo realizado durante este proyecto. Además, se contrastan los resultados del capítulo 3.4 con los objetivos planteados al comienzo de esta memoria.

Finalmente, se entregan sugerencias de trabajos que se pueden realizar a partir de lo desarrollado como solución en este trabajo. Las sugerencias van desde nuevas configuraciones en la cámara hasta nuevos métodos que aprovechen de mejor manera la cantidad de información que entrega actualmente el algoritmo desarrollado.

5.1. Conclusiones y resultados contrastando con los objetivos planteados

A partir de todos los resultados obtenidos y del desarrollo realizado a lo largo del proyecto, se pueden establecer ciertas conjeturas con respecto a la solución planteada e implementada.

Empezando por que tanto la impresión 3D planteada, como la solución momentánea implementada durante las pruebas, generarían el mismo ángulo de visión de la cámara y esto a su vez no afectaría en el procesamiento o descifrado del código QR, se considera este objetivo cumplido, ya que lo importante es la ubicación de la ESP32-CAM dentro del modelo del agente y que este posicionamiento no interrumpa o dificulte la funcionalidad del agente.

Durante el desarrollo de las pruebas, se probó con distintos tamaños de códigos QR y con distintas formas de exhibirlos (desde pantalla de iPad o impreso en papel). Según los últimos re-

sultados obtenidos, se concluye que lo mejor es utilizar un código QR de tamaño 16 centímetros y éste debe estar impreso, ya que de esta forma se puede iluminar directamente con una fuente de luz externa para lograr una imagen uniformemente iluminada.

Gracias a las pruebas realizadas en movimiento, donde se logra hasta un 90 % de efectividad en la decodificación de los códigos QR a velocidades bajas, el objetivo de lograr desarrollar un código capaz de descifrar dichos códigos QR se considera cumplido, ya que, si bien no se tiene un 100 % de efectividad, se debe tener en consideración las limitaciones que presenta trabajar con un microcontrolador que, además de capturar las imágenes, es el encargado del procesamiento de éstas.

El objetivo general del proyecto es lograr diseñar e implementar una forma de determinar anticipadamente un cambio en la disposición de las vías, es por esto que el objetivo se cumple una vez que se programa la flota para detenerse al descifrar un código QR. Esto se considera como logrado gracias a la últimas pruebas de comunicación entre ESP32-CAM y ESP32 del Agente Líder, por lo que el resultado del proyecto es positivo.

Por último, se considera como una ventaja extra las funcionalidades añadidas de almacenamiento de imágenes y de estimación de distancia al código QR, las que pueden ser utilizadas para futuras pruebas o continuaciones de este trabajo.

5.2. Trabajo a Futuro

Si bien este proyecto se da por completado, durante el desarrollo de éste se encontraron ciertos cambios que se pueden aplicar a futuro que por falta de tiempo o colaboradores no se pudo llevar a cabo. Estas posibles mejoras a la solución pueden aportar más información a la flota e incluso mejorar su comportamiento si se logra implementar correctamente.

Estas posibles mejoras o posibles soluciones extra son:

- Cambiar más parámetros de la cámara de modo automático a estático, para mejorar la calidad de imagen según ciertas condiciones buscadas.
- Durante las pruebas se determina que, si el agente se mueve en dirección contraria al QR, éste lo detecta aún en velocidades altas, por lo que se recomienda probar apuntando la cámara hacia atrás. Es decir, que lo códigos QR estén en dirección contraria a donde se mueve el

agente.

- Actualmente se puede estimar la distancia al código QR, por lo que, si esta distancia se pudiese enviar al Agente Líder, éste podría estimar el tiempo que falta para llegar a la curva, ya que éste conoce su propia velocidad.

- Como la ESP32-CAM cuenta con conectividad inalámbrica y ésta no se usa durante la solución, se puede plantear una variante del algoritmo, donde el procesado de la imagen no ocurra en la misma ESP32-CAM, si no que se puede enviar a otro microcontrolador o a un computador para su procesado. De esta forma la ESP32-CAM del Agente Líder, solo se preocuparía de capturar las imágenes y enviarlas.

- Debido a que la actual solución no cuenta con un 100 % de efectividad, se pueden plantear poner una mayor cantidad de señaléticas, ya que de esta forma multiplicaría las posibilidades de ser capturadas y descifradas por la ESP32-CAM, pero considerando que no deben obstruirse la visión entre ellas.

- Si bien el nivel de exposición está estático, se deben realizar pruebas para establecer dicho nivel dependiendo del ambiente donde se trabajará, por lo que se sugiere crear una rutina donde se ajuste automáticamente dicho parámetro basándose en el porcentaje de efectividad de las imágenes capturadas. Esta rutina debe ser de inicialización, por lo que se sugiere que ocurra antes de realizar cualquier prueba.

Bibliografía

[1] "Seguridad vial en vehículos altamente automatizados," *Revista Centro Zaragoza*, 2018. [Online]. Available: https://revistacentrozaragoza.com/seguridad-vial-en-vehiculos-altamente-automatizados/

[2] "Self-driving trucks likely to hit the roads before passenger cars," *CNBC*, 2019. [Online]. Available: https://www.cnbc.com/2019/11/22/self-driving-trucks-likely-to-hit-the-roads-before-passenger-cars.html

[3] C. Andrade, C. Garrido, A. Peters, and F. Vargas, "A low cost experimental platform for the study of scalability issues in multi-agent systems," in *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 2019, pp. 1–6.

[4] D. Ghorpade, A. D. Thakare, and S. Doiphode, "Obstacle detection and avoidance algorithm for autonomous mobile robot using 2d lidar," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 2017, pp. 1–6.

[5] A. Gonzalez, M. D. Zuniga, C. Nikulin, G. Carvajal, D. G. Cardenas, M. A. Pedraza, C. A. Fernandez, R. I. Munoz, N. A. Castro, B. F. Rosales *et al.*, "Accurate fire detection through fully convolutional network," 2017.

[6] M. Owayjan, A. Dergham, G. Haber, N. Fakih, A. Hamoush, and E. Abdo, "Face recognition security system," in *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering*. Springer, 2015, pp. 343–348.

[7] T. J. Soon, "Qr code," *Synthesis Journal*, vol. 2008, pp. 59–78, 2008.

[8] N. Kolban, “Kolban’s book on esp32,” *USA: Leanpub*, 2017.

[9] M. N. Babu and P. M. Krishna, “Hand gesture based camera monitoring system using raspberry pi,” 2019.

[10] M. F. Wicaksono and M. D. Rahmatya, “Implementasi arduino dan esp32 cam untuk smart home,” *Jurnal Teknologi dan Informasi (JATI)*, vol. 10, no. 1, pp. 40–51, 2020.

[11] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” *Automatica*, vol. 53, pp. 424–440, 2015.

[12] *ESP32 Series Datasheet*, Espressif Systems, 2019, ver. 3.1. [Online]. Available: https://www.espressif.com/en/products/hardware/esp32/overview

[13] *ESP32-CAM Module*, Shenzhen Ai-Thinker, 2017. [Online]. Available: https://docs.ai-thinker.com/_media/esp32/docs/esp32-cam_product_specification_zh.pdf

[14] *OV2640 Advanced Information Preliminary Datasheet*, OmniVision, 2017. [Online]. Available: http://www.uctronics.com/download/OV2640_DS.pdf

[15] S. O. Fernandez, “Sistema automatizado de trenes para estudiar el control de la formación de vehículos,” 2018.

Apéndice A

Tablas Extras

En esta sección se entregarán los resultados completos de las distintas pruebas realizadas a lo largo del proyecto. Estos datos se presentan en forma de tablas, debido a la gran cantidad de iteraciones que se realizó para cada prueba.

Resultados de pruebas con cámara estática

Se presentan las tres tablas de resultados relacionadas a las pruebas donde la ESP32-CAM está estática tal como es descrita en la sección 3.3.1. Las primeras dos tablas son resultados con los códigos QR mostrados en la pantalla de un iPad, mientras que los resultados de la tercera tabla corresponden al código QR impreso e iluminado con una fuente de luz externa. Cabe destacar que la resolución utilizada para estas pruebas es QVGA.

Tamaño lado QR (cm)	Distancia (cm)	Porcentaje Eficiencia (%)
5	5	0
5	10	60
5	15	100
5	20	90
5	25	60
5	30	30
5	35	60
5	40	50
5	45	0
5	50	0
8	5	0
8	10	0
8	15	100
8	20	100
8	25	100
8	30	100
8	35	100
8	40	80
8	45	40
8	50	0
10	5	0
10	10	0
10	15	0
10	20	100
10	25	100
10	30	90
10	35	100
10	40	100
10	45	80
10	50	70

Tabla A.1: Parte 1 de los Resultados de Prueba estática con código QR en iPad

Tamaño lado QR (cm)	Distancia (cm)	Porcentaje Eficiencia (%)
12	5	0
12	10	0
12	15	0
12	20	0
12	25	100
12	30	100
12	35	100
12	40	100
12	45	100
12	50	90
14	5	0
14	10	0
14	15	0
14	20	0
14	25	100
14	30	100
14	35	100
14	40	100
14	45	100
14	50	100

Tabla A.2: Parte 2 de los Resultados de Prueba estática con código QR en iPad

Tamaño lado QR (cm)	Distancia (cm)	Porcentaje Eficiencia (%)
16	25	0
16	30	100
16	35	100
16	40	100
16	45	100
16	50	100
16	55	100
16	60	100
16	65	70
16	70	60
16	75	30
16	80	20
19	25	0
19	30	0
19	35	100
19	40	100
19	45	100
19	50	100
19	55	60
19	60	70
19	65	50
19	70	60
19	75	40
19	80	20

Tabla A.3: Resultados de Prueba estática con código QR impreso

Resultados de pruebas de nivel de exposición

En esta sección se presentan las tablas correspondientes a las dos pruebas relacionadas con el nivel de exposición configurado en la ESP32-CAM. En la primera tabla se entregan los resultados de la efectividad del algoritmo a distintas distancias y distintos tamaños de códigos QR, utilizando la resolución QQVGA en las imágenes. En la segunda tabla, se presentan los resultados del experimento con las mismas características al recién descrito, pero la resolución utilizada en las imágenes corresponde a QVGA. Como se mencionó en la sección 3.3.2 esta prueba cuenta con las mismas condiciones espaciales que la prueba de cámara estática y los porcentajes de aciertos se basan en la cantidad de códigos descifrados en 10 iteraciones.

Resolución	Tamaño lado QR (cm)	Distancia (cm)	Porcentaje Eficiencia (%)
QQVGA	8	15	100
QQVGA	8	20	60
QQVGA	8	25	0
QQVGA	8	30	0
QQVGA	8	35	0
QQVGA	8	40	0
QQVGA	8	45	0
QQVGA	8	50	0
QQVGA	10	15	40
QQVGA	10	20	100
QQVGA	10	25	40
QQVGA	10	30	30
QQVGA	10	35	0
QQVGA	10	40	0
QQVGA	10	45	0
QQVGA	10	50	0
QQVGA	12	15	0
QQVGA	12	20	40
QQVGA	12	25	40
QQVGA	12	30	90
QQVGA	12	35	60
QQVGA	12	40	0
QQVGA	12	45	0
QQVGA	12	50	0
QQVGA	14	15	0
QQVGA	14	20	0
QQVGA	14	25	90
QQVGA	14	30	90
QQVGA	14	35	60
QQVGA	14	40	10
QQVGA	14	45	0
QQVGA	14	50	0

Tabla A.4: Resultados de Prueba con cámara estática con resolución QQVGA

Resolución	Tamaño lado QR (cm)	Distancia (cm)	Porcentaje Eficiencia (%)
QVGA	8	15	100
QVGA	8	20	100
QVGA	8	25	100
QVGA	8	30	80
QVGA	8	35	90
QVGA	8	40	10
QVGA	8	45	30
QVGA	8	50	0
QVGA	10	15	40
QVGA	10	20	100
QVGA	10	25	100
QVGA	10	30	100
QVGA	10	35	90
QVGA	10	40	60
QVGA	10	45	100
QVGA	10	50	0
QVGA	12	15	0
QVGA	12	20	100
QVGA	12	25	100
QVGA	12	30	100
QVGA	12	35	100
QVGA	12	40	100
QVGA	12	45	60
QVGA	12	50	50
QVGA	14	15	0
QVGA	14	20	0
QVGA	14	25	100
QVGA	14	30	100
QVGA	14	35	100
QVGA	14	40	100
QVGA	14	45	100
QVGA	14	50	100

Tabla A.5: Resultados de Prueba con cámara estática con resolución QVGA

Resultados de pruebas con cámara en movimiento

Estos resultados corresponden a la prueba de efectividad del algoritmo mientras la ESP32-CAM está ensamblada al Agente Líder, el que va en movimiento según la actuación solicitada. Se entregan los resultados de efectividad del algoritmo correspondiente a cada tamaño de código QR y a cada velocidad de actuación probada.

Tamaño lado QR (cm)	Actuación	Porcentaje Eficiencia (%)
8	300	0
8	400	30
8	500	0
8	600	0
8	700	0
8	800	0
10	300	20
10	400	20
10	500	0
10	600	0
10	700	0
10	800	0
12	300	10
12	400	20
12	500	10
12	600	0
12	700	0
12	800	0
14	300	70
14	400	30
14	500	20
14	600	0
14	700	0
14	800	0
16	300	90
16	400	70
16	500	40
16	600	30
16	700	0
16	800	0

Tabla A.6: Resultados de prueba con agente en movimiento