https://repository.uaeh.edu.mx/revistas/index.php/investigium/issue/archive

Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca

InvESTigium

ISSN: 2448-4830

Publicación semestral, Vol. 11, No. Especial (2025) 156-164

"Cuarto Congreso Internacional de Investigación e Innovación en Tendencias Globales y Segundo Coloquio de Posgrado en Gestión y Desarrollo de Nuevas Tecnologías"
24 –30 de octubre de 2025

Cronometraje y visualización de datos en carreras de ¼ de milla: un sistema basado en ESP32.

Timing and data visualization in quarter-mile races: an ESP32-based system.

Roberto Carlos García García [a], Uriel Efrain Sánchez Acolt [b], Kristian Freyri Maya Gress [c] Víctor Eduardo Pedraza Vera [d], Juan Carlos González Islas [e]

Abstract:

In this work, the development and implementation of a timing, control, and data visualization system for quarter-mile motorcycle races has been presented. The system architecture is based on an ESP32 microcontroller, which integrates a starting traffic light, a timing interface, and a mobile application for real-time visualization. E3F-5DN1 and E3F-5L photoelectric sensors, along with HC-12 radio frequency modules, are employed to establish wireless communication between the start and finish lines. Field testing has demonstrated the system's capability to accurately detect passing vehicles at high speeds, ensuring reliable times in the millisecond range. This proposal is constituted as a low-cost and easy-to-implement alternative to commercial timing systems, thereby improving the experience of both competitors and spectators.

Keywords:

Drag racing, ESP32 microcontroller, Precision sensors, Radio frequency modules, Mobile app, timing.

Resumen:

Este trabajo presenta el desarrollo e implementación de un sistema de cronometraje, control y visualización de datos para carreras de motociclismo de un cuarto de milla. La arquitectura del sistema se basa en un microcontrolador ESP32 que integra un semáforo de salida, una interfaz de cronometraje y una aplicación móvil para visualización en tiempo real. El sistema emplea sensores fotoeléctricos E3F-5DN1 y E3F-5L, junto con módulos de radiofrecuencia HC-12, para establecer comunicación inalámbrica entre las líneas de salida y meta. Las pruebas de campo demuestran la capacidad del sistema para detectar con precisión el paso de vehículos a alta velocidad, garantizando tiempos fiables en el rango de milisegundos. La propuesta constituye una alternativa económica y de fácil implementación a los sistemas de cronometraje comerciales, mejorando la experiencia tanto de los competidores como de los espectadores.

***Palabras Clave**:*

Aplicación móvil, Carreras de aceleración, Cronometraje, Microcontrolador ESP32, Módulos de radiofrecuencia, Sensores de precisión.

I. Introducción

En los años 50 las carreras ¼ de milla empezaban a tener gran impacto dentro del mundo automovilístico y aunque eran una gran atracción para aquel entonces, las técnicas empleadas para cronometrar y medir dichas carreras no eran las más precisas. Inicialmente la salida se anunciaba con una bandera, empleaban cronómetros manuales para medir el tiempo en el cual los corredores recorrían la pista y el ganador se determinaba por quien cruzaba la meta a simple vista. Años más tarde la Asociación Nacional de *Hot Rods* implementó el primer semáforo conocido como "árbol de navidad" [1], el cual constaba solo de 5 luces color ámbar y una luz color verde. En la temporada de 1986 ya no eran 5 luces ámbar, solo 3, y

[a,b,c,d,e], *Universidad Autónoma del Estado de Hidalgo | Instituto de Ciencias Básicas e Ingeniería | Hidalgo | México,* https://orcid.org/0009-0003-3429-2187, *Email:* ga439422@uaeh.edu.mx; https://orcid.org/0009-0000-4411-6994, *Email:* sa443676@uaeh.edu.mx; https://orcid.org/0000-0001-8425-0960, *Email:* kristian_maya10493@uaeh.edu.mx; https://orcid.org/0009-0009-0610-2597, *Email:* vpedraza@uaeh.edu.mx; https://orcid.org/0000-0002-2190-0660, *Email:* juan_gonzalez7024@uaeh.edu.mx.

Fecha de recepción: 08/09/2025, Fecha de aceptación: 15/10/2025, Fecha de publicación: 12/12/2025

DOI: https://doi.org/10.29057/est.v11iEspecial.15971

presentaban un diseño más moderno con cronómetros analógicos. El siguiente cambio fue hasta inicios de los 2000 cuando se eliminaron las bombillas y se usaron luces LED. Posteriormente, la tecnología ayudó a implementar sensores infrarrojos, de velocidad, GPS, y sistemas de cronometraje computarizados para una alta precisión y confiabilidad.

Actualmente, los sistemas de cronometraje deportivo para competiciones automovilísticas emplean diversas tecnologías en función de las exigencias específicas de cada certamen [2][3]. Entre estas se incluyen sistemas basados en transpondedores [4], tecnologías NFC [5], radiofrecuencia (RFID) [6] y cronómetros de luz [7]. Ante diversas opciones disponibles en el mercado, resulta un desafío para las competencias amateur encontrar un equilibrio entre el rendimiento de los sistemas de medición como la precisión del cronometraje, el tiempo de respuesta de los sensores, la latencia en los protocolos de comunicación y los costos asociados que implican la adquisición, implementación y mantenimiento del sistema.

II. Preliminares

Un estudio previo expone el desarrollo e implementación de un sistema de cronometraje inalámbrico y portátil en carreras de atletismo, el cual fue desarrollado en el entorno Arduino IDE mediante un microcontrolador Arduino UNO e integra módulos de radiofrecuencia NRF24L01+PA+LNA, sensores ópticos que detectan el paso del atleta y módulos Bluetooth HC-05 [8]. La comunicación entre el sistema se realiza por radiofrecuencia para cubrir la distancia de la pista de atletismo, mientras que la comunicación con una aplicación móvil se efectúa a través de Bluetooth. El sistema logró un error de medida máximo de 150 ms y operar con una autonomía superior a 20 horas.

En una publicación aparte se explica el proceso de creación de un sistema de cronometraje asequible para atletas BMX. El sistema consta de dos módulos uno transmisor y un módulo receptor. Ambos módulos compuestos por un Arduino UNO, un sensor fotoeléctrico y un transmisor inalámbrico ASK los cuales en conjunto se encargan de enviar la señal para que el cronómetro se active cada vez que el ciclista atraviesa la barrera láser y lo detiene una vez que el competidor atraviesa la línea láser de meta [9]. Los resultados de dichos trabajos confirmaron el óptimo funcionamiento de sus respectivos sistemas en las largas distancias de las pistas y demostraron que es posible desarrollar un sistema de cronometraje de bajo costo.

Es por ello que en base a los antecedentes del deporte y los trabajos antes mencionados el presente proyecto aspira a constituir una alternativa funcional y de bajo costo para el sector de las carreras ¼ de milla, con el objetivo de satisfacer sus requisitos básicos y mejorar la experiencia de competidores y espectadores tomando en cuenta la facilidad de uso por parte del usuario, portabilidad y el más alto rendimiento posible en distintos entornos.

El desarrollo del sistema se fundamentó principalmente en la programación secuencial en el lenguaje de programación Python en un entorno de desarrollo de alto nivel sobre el microcontrolador ESP32 DEVIKIT V1 [10], sistemas en chip que permiten la integración y comunicación coordinada de los diversos componentes del sistema [11], como los sensores fotoeléctricos E3F-5FL y E3F-5DN1, dispositivos que detectan la presencia o ausencia de un objeto basándose en la interrupción o cambio en la intensidad de un haz de luz infrarroja [12] y el módulo de comunicación inalámbrico HC-12, un transceptor de comunicación serial inalámbrica que opera en la banda de los 433 MHz - 470 MHz [13].

III. Metodología de diseño

Lógica del semáforo. Con el propósito de optimizar la simulación, se empleó el software de diseño electrónico Proteus que integra las librerías requeridas de Arduino, dado que la programación del microcontrolador se desarrolló en el entorno (IDE) de Arduino. [14]. Dichos esquemáticos se muestran en las *Figuras 1 y 2*. La lógica con la que se programó el semáforo de carrera consta de una secuencia de luces que guía a los competidores desde el posicionamiento inicial hasta el final del certamen. Dicha secuencia comienza con dos luces amarillas que indican a los pilotos que se coloquen en posición, cuando los sensores de posicionamiento detectan que las motocicletas están correctamente alineadas se encienden dos luces azules. Una vez cumplida esta condición el operador puede activar el encendido de una secuencia de tres luces anaranjadas en intervalos de un segundo, seguida por una luz verde que indica el inicio de la carrera. En caso de un posicionamiento incorrecto el semáforo recibe la alerta de salida en falso de los sensores y enciende una luz roja en el respectivo carril, una vez que el competidor se reacomode en una posición permitida se puede reiniciar la secuencia.

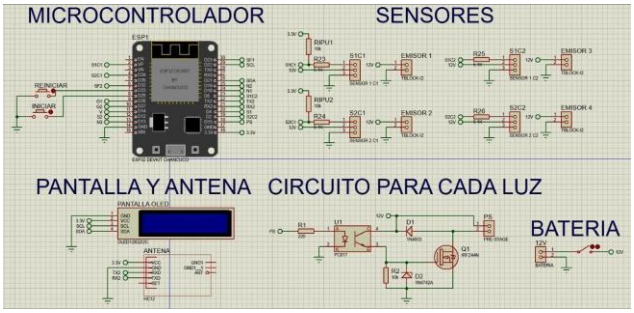

Figura 1. Diagrama esquemático del módulo de línea de arranque.

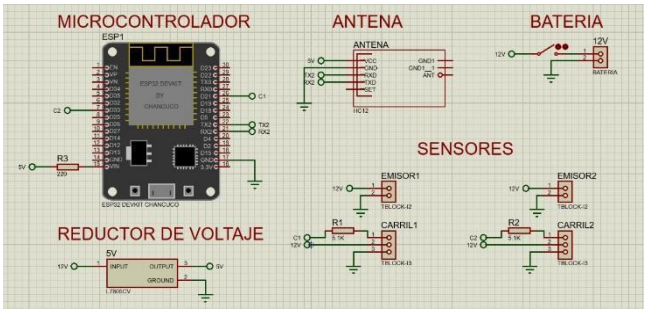

Figura 2. Diagrama esquemático del módulo de línea de meta.

Programación de cronómetro. Los cronómetros se activan cuando el semáforo enciende la luz verde y se detienen una vez que los competidores atraviesan el rayo de luz infrarrojo de los sensores ubicados en la línea de meta para su implementación física en el prototipo, ambos cronómetros fueron programados en el entorno de Arduino IDE y visualizados en una pantalla OLED como se muestra en la *Figura 3*.

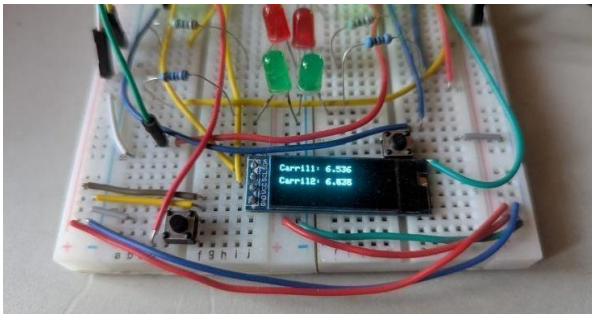

Figura 3. Cronómetros mostrados en el display.

Es preciso recalcar que el sistema se fundamenta en un flujo centralizado a través del ESP32, este actúa como fuente de datos para las interfaces que puede visualizar el usuario, tanto el script de la interfaz de visualización en Python como la aplicación móvil creada en App Inventor extraen y arrojan los datos que son generados por la ESP32 como se ilustra en el diagrama de la arquitectura de software (*Figura 4*). Este diseño garantiza que ambas interfaces muestren de manera consistente los datos de la carrera.

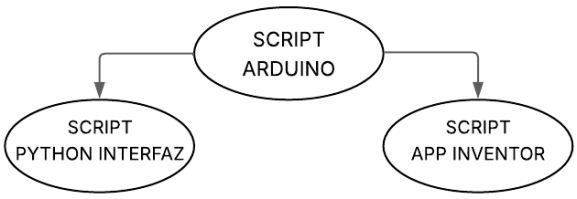

Figura 4. Diagrama de la arquitectura de software.

Interfaz visual para una pantalla grande. En la *Figura 5* se muestra una línea del código de Python para la creación de un script que lee los datos obtenidos del módulo de salida por medio de comunicación serial [15] con la tarjeta ESP32 vía USB, y los muestra en una interfaz junto con las alertas de salida en falso y el ganador del certamen, como se ilustra en la *Figura 6*. Para establecer dicha comunicación es preciso que la ESP32 correspondiente al módulo de salida esté conectado a un equipo de cómputo que ejecute el script [16], sin embargo, el realizar la comunicación de esta manera otorga la posibilidad de proyectar la interfaz a través de los puertos HDMI [17].

```
SERIAL_PORT = "COM4"
BAUD_RATE = 115200

ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
```

Figura 5. Línea de código de Python que establece la comunicación serial con la ESP32.

Figura 6. Vista previa de la interfaz del ordenador.

Comunicación a distancia. Debido a la distancia entre el módulo ubicado en la línea de inicio y el módulo ubicado en la línea de meta se optó por implementar antenas de radiofrecuencia compatibles con los ESP32 para ello, se implementaron 2 transceptores inalámbricos

HC-12 mostrados en la *Figura 7*, los cuales son capaces de comunicarse con protocolos UART y de establecer una comunicación hasta 1000 metros de distancia con la antena omnidireccional de 433Mhz. Para la conexión de ambos módulos se agregó una comunicación serial extra y se declaró en el script de Arduino la velocidad de transmisión (9600 baudios) y en la *Figura 8* se muestra la asignación de los pines TX y RX del ESP32.

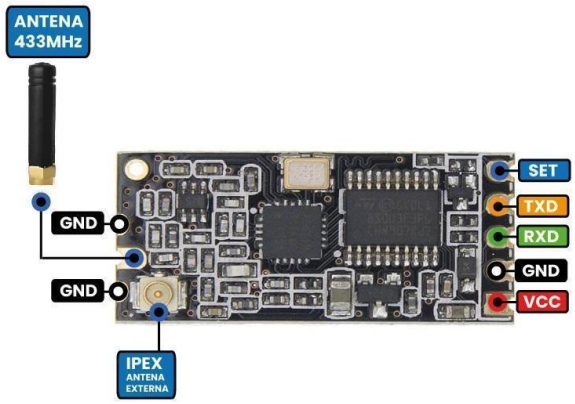

Figura 7. Pinout de módulo de comunicación inalámbrico HC-12.

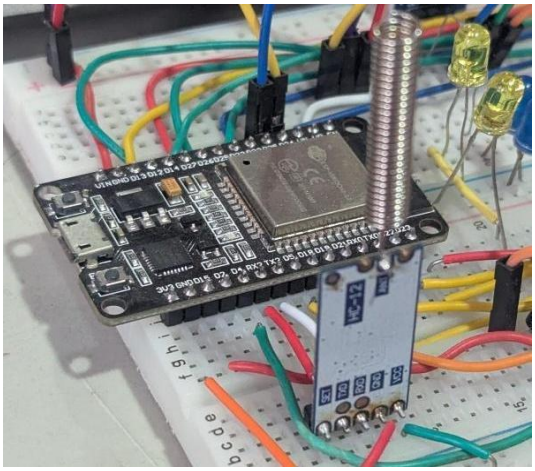

Figura 8. Conexiones entre ESP32 y HC-12 con antena helicoidal de referencia.

Creación del Bróker MQTT. Para fines de este proyecto se empleó MQTT (Message Queuing Telemetry Transport) un protocolo de mensajería ligero que se basa principalmente en un modelo de publicación-suscripción emplea una comunicación desacoplada usando a un nodo central (bróker) que recibe mensajes de los publicadores, y los reenvía a los suscriptores a través de *topics*, los cuales permiten organizar y filtrar los mensajes [18]. El funcionamiento de este protocolo se detalla en el Diagrama del Protocolo MQTT (*Figura 9*).

Considerando la saturación al utilizar un bróker público se optó crear uno privado, para ello se empleó el software *Mosquitto* el cuál es un bróker de código abierto.

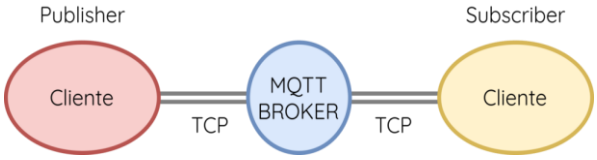

Figura 9. Diagrama de protocolo MQTT.

Primeramente, se instaló el software Eclipse Mosquitto en el sistema operativo Windows para posteriormente realizar la configuración del servicio del bróker para que se inicie automáticamente. Para habilitar la comunicación en red se crearon reglas de entrada y salida en el Firewall (Sistema de seguridad de Windows Defender) para abrir el puerto 1883, utilizado para conexiones MQTT sin cifrar, véase *Figura 10.* Posteriormente el archivo de configuración *mosquitto.conf* fue modificado con la inclusión de los comandos *listener 1883* y *allow_anonymous true*, para permitir que cualquier cliente pueda conectarse a través del puerto previamente configurado, como se muestra en la *Figura 11*. Finalmente, se reinició el servicio para aplicar los cambios y la dirección IP del ordenador se utilizó como la dirección bróker para la comunicación del sistema [19].

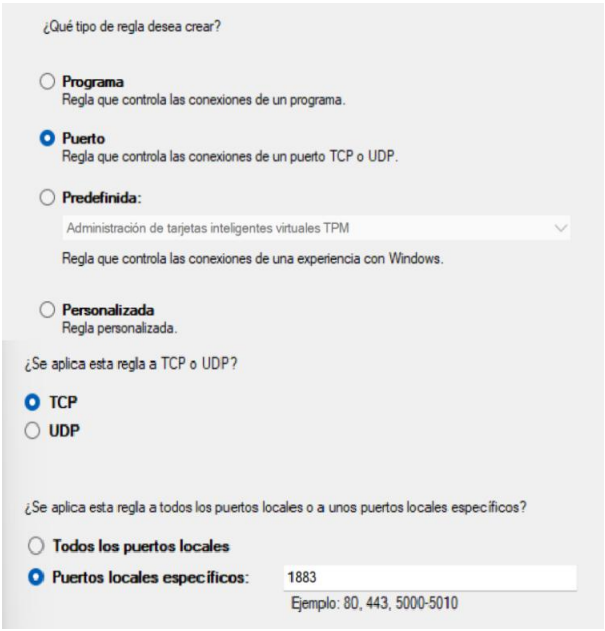

Figura 10. Selección del puerto 1883.

```
# ------------------------------------------------------------------
# PSK based SSL/TLS support
# ------------------------------------------------------------------
# Pre-shared-key encryption provides an alternative to certificate based
# encryption. A bridge can be configured to use PSK with the bridge_identity
# and bridge_psk options. These are the client PSK identity, and pre-shared-key
# in hexadecimal format with no "0x". Only one of certificate and PSK based
# encryption can be used on one
# bridge at once.
#bridge_identity
#bridge_psk
listener 1883
allow_anonymous true

# ##################################################################
# External config files
# ##################################################################
```

Figura 11. Modificación del archivo mosquitto.conf

Aplicación móvil. Para el desarrollo de la interfaz móvil mostrada en la *Figura 15* se utilizó la plataforma MIT App Inventor, una herramienta de programación basada en bloques diseñada para facilitar la creación de aplicaciones prototipo para dispositivos Android [20], la tarea de esta aplicación es mostrar los tiempos de los competidores en tiempo real, así como las notificaciones sobre salidas en falso y ganadores las *Figuras 14* muestran los bloques de programación de la aplicación.

Para transmitir los datos desde el microcontrolador ESP32 a la aplicación móvil se implementó el protocolo de comunicación MQTT debido a la facilidad que brinda al añadir la extensión de este protocolo, véase *Figura 12*. La lógica de la aplicación se basó en el modelo de publicación/suscripción, lo que permite que cualquier persona que se conecte al bróker de MQTT donde se envía la información, pueda visualizar en su dispositivo los mismos datos que se muestran en la interfaz diseñada para ser proyectada.

Figura 12. Extensión para protocolo MQTT

Las variables del código de Arduino destinadas para almacenar los tiempos y mensajes fueron designadas como *topics* publicadas a través de un bróker de MQTT privado como se muestra en la *Figura 13.* La aplicación móvil en función de cliente se suscribe a estos *topics* específicos para recibir y visualizar los datos en tiempo real. Para garantizar una comunicación exitosa y con baja latencia, tanto el ESP3 como los dispositivos móviles debían estar conectados a la misma red Wi-Fi que el ordenador alojaba en el bróker.

```
const char* mqttServer = "172.31.134.152";
const int mqttPort = 1883;
const char* topic_carril1 = "carrera/carril1";
const char* topic_carril2 = "carrera/carril2";
```

Figura 13. Línea de código de Arduino vinculado con MQTT.

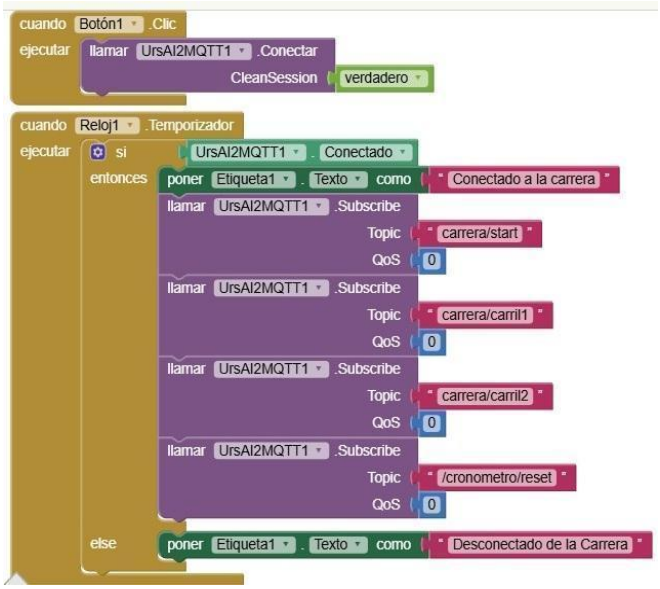

Figura 14. Bloques de programación en App Inventor

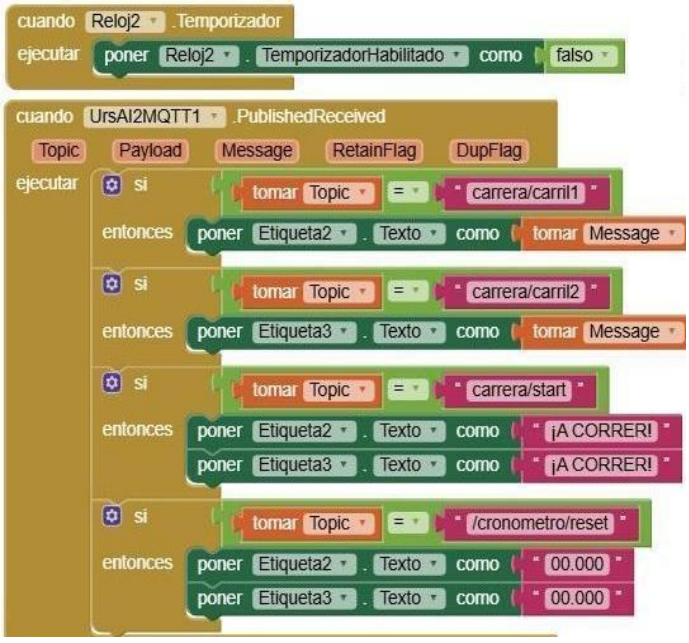

Figura 14.1. Bloques de programación en App Inventor

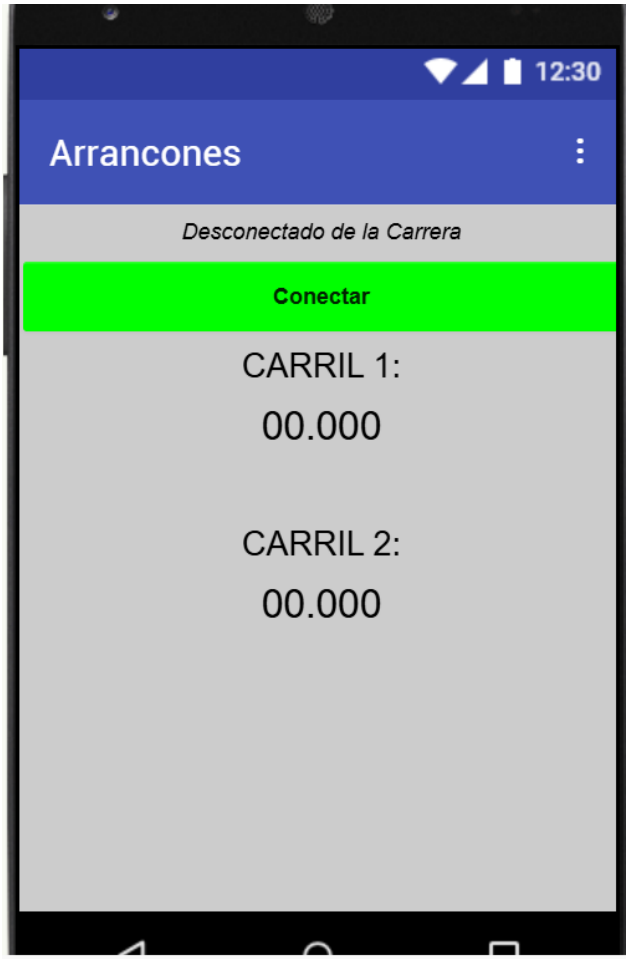

Figura 15. Vista previa de la interfaz en la app.

Estructuras de montaje. Para el diseño de las estructuras de protección, se tomaron en cuenta las dimensiones de los sensores ilustradas en la *Figura 16* con el objetivo de protegerlos de factores ambientales que pudieran dañarlos, cada sensor cuenta con dos tuercas para su ajuste preciso en el orificio de la caja, la base inferior fue diseñada para fungir como soporte con contrapeso para asegurar la estabilidad de los sensores y evitar movimientos inesperados.

Para los módulos de control, se optó por un diseño similar a un sintonizador de frecuencias. Estas estructuras están destinadas para alojar componentes clave como los ESP32, las antenas, la pantalla OLED, botones y baterías, se fabricaron prototipos iniciales para posteriormente construir y estandarizar las estructuras con materiales de mayor calidad y resistencia a esfuerzos mecánicos, los diseños de ambas estructuras se muestran en las *Figuras 17 y 18.*

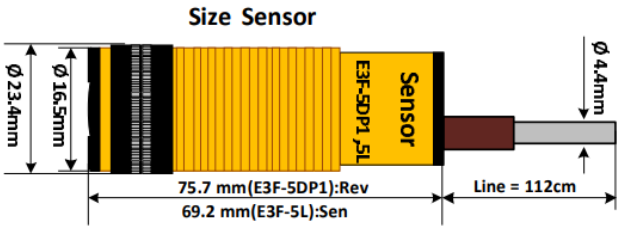

Figura 16. Medidas de los sensores fotoeléctricos.

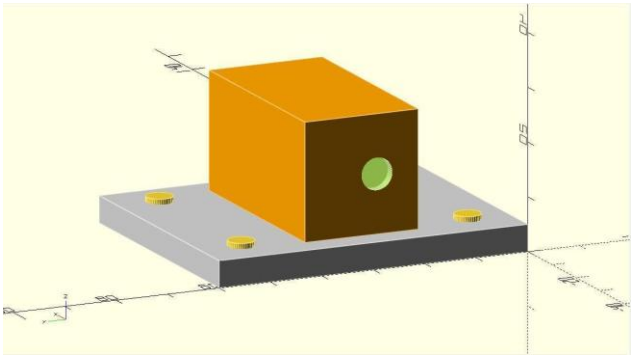

Figura 17. Vista lateral derecha de la base para los sensores.

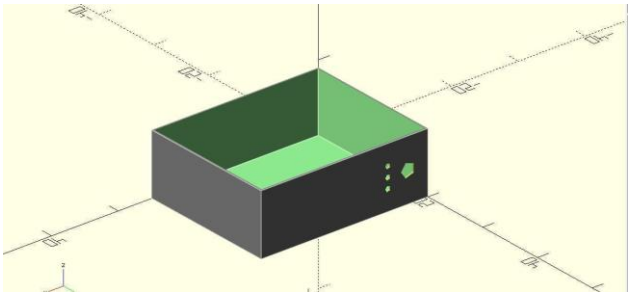

Figura 18. Vista lateral derecha de la estructura para módulos de control.

La estructura del semáforo cuyas medidas y modelado se ilustran en la *Figura 19 y 20* fue fabricada con perfiles tubulares de acero para garantizar su robustez. La secuencia de señalización luminosa se diseñó para guiar a los competidores a través de las diferentes etapas de la carrera: una fila de luces superiores indica el pre-stage, seguida de una fila que confirma el posicionamiento correcto de cada piloto (stage). La secuencia de arranque que se muestra en la *Figura 21* se compone de luces amarillas y una luz verde final que inicia la carrera. Las luces rojas son para indicar si algún piloto tuvo una salida en falso y las luces azules señalan al respectivo ganador.

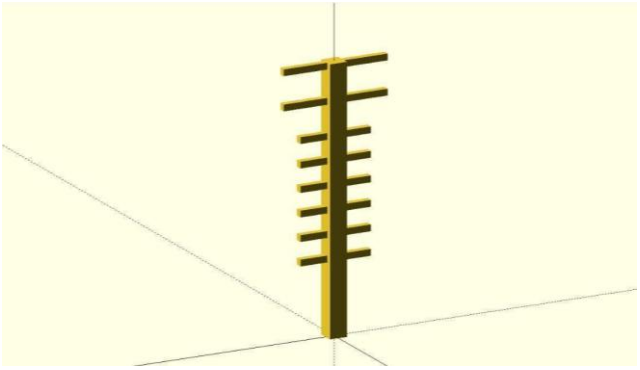

Figura 19. Vista lateral derecha de la estructura del semáforo

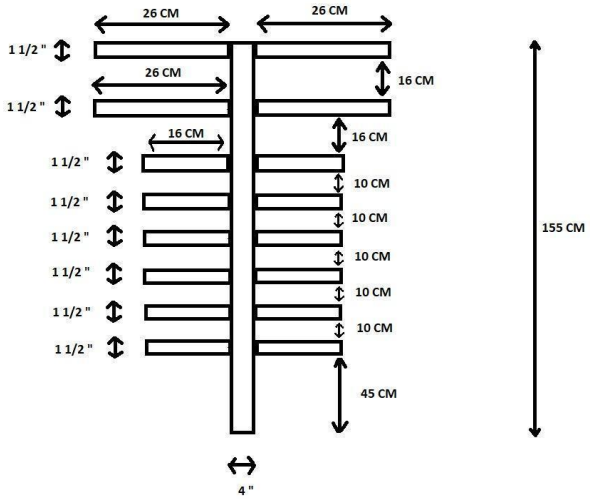

Figura 20. Medidas de la estructura del semáforo

Figura 21. Construcción final del semáforo

Pruebas en campo. Las primeras pruebas de campo se llevaron a cabo con un solo piloto para validar la funcionalidad del sistema, véase *Figura 22*. El objetivo de esta prueba fue verificar la correcta posición inicial por parte de los sensores, el funcionamiento de la lógica programada del semáforo, la capacidad de cubrir la distancia de un cuarto de 402 metros como mínimo y la habilidad de los sensores de meta para detectar el paso de una motocicleta a altas velocidades.

Durante la prueba, se identificaron dos problemas principales: la velocidad de la motocicleta era demasiado alta, impidiendo que los sensores de la meta reaccionaran a tiempo al interrumpir el haz de luz y los módulos RF perdían la comunicación debido a la distancia. Para corregir dichas fallas se realizó un ajuste en la posición y enfoque de los sensores reduciendo la distancia hasta 1 metro de distancia del vehículo maximizando su precisión de detección y se cambió la antena helicoidal del módulo HC-12 por una omnidireccional de ¼ de onda.

Figura 22. Pruebas de campo en condiciones de competición real.

IV. Resultados

Tras las modificaciones, se realizó una nueva serie de pruebas de campo con un piloto en condiciones de competición real. Los resultados demostraron que el sistema, tras los ajustes de posición de los sensores, funcionó sin errores de comunicación RF y detección infrarroja. Se obtuvo una detección precisa del paso de la motocicleta a altas velocidades con un tiempo de respuesta de los sensores de 1 µs. La comunicación de las antenas HC-12, entre la línea de salida y la meta se mantuvo estable con un promedio de 92356.7 µs, ambas mediciones se muestran en las *Figuras 23 y 24* y los tiempos de los competidores mostrados en la *Figura 25* se registraron con una precisión de milisegundos, lo que valida la funcionalidad del sistema para el cronometraje de carreras de aceleración.

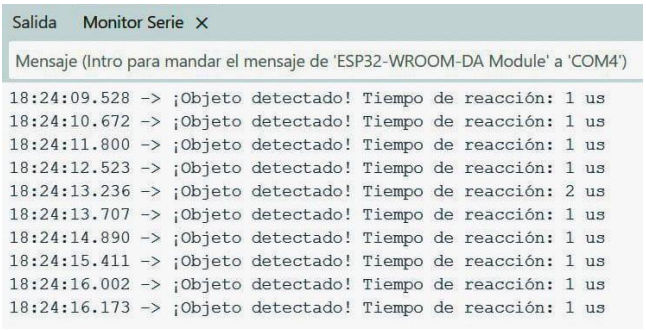

Figura 23. Tiempo de detección de los sensores.

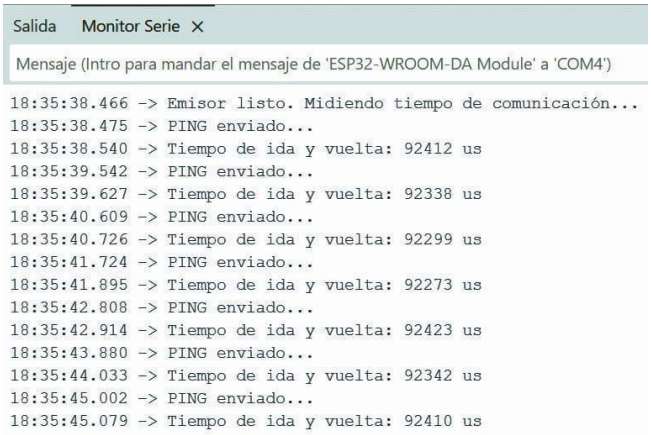

Figura 24. Tiempo de respuesta entre módulos RF.

Figura 25. Tiempos de cronometraje mostrados en la interfaz

Haciendo una comparativa con los resultados obtenidos por una investigación previa que implementó módulos RF ASK en su sistema de cronometraje cuya cobertura máxima fue de 150 metros [9], en el presente trabajo obtuvo una notable mejora debido a la implementación de módulos HC-12 en el sistema, consiguiendo una cobertura máxima de 1000 metros en condiciones de línea de visión directa, cubriendo las distancias de las carreras

De forma análoga se analizaron los resultados de la investigación que utilizó un módulo Bluetooth HC-05 de Arduino para enlazarlo con una aplicación móvil, los cuales arrojaron un retraso de 1.4 segundos, un tiempo muy elevado para la medición de tiempos en pruebas de atletismo [8]. En contraste, en el presente trabajo se optó por desarrollar el sistema en un microcontrolador ESP32 para utilizar protocolos de comunicación más avanzados, como MQTT, que funciona a través de una red WiFi local. Esta elección permitió la conectividad a múltiples dispositivos móviles para visualizar los datos en tiempo real, concluyendo que la conectividad WiFi del ESP32 es una solución más escalable y robusta para este entorno, ya que ofrece una baja latencia en la transmisión de datos a múltiples usuarios.

V. Conclusión

Los resultados obtenidos validan la funcionalidad del sistema propuesto para cronometraje en carreras de 1/4 de milla, demostrando una detección confiable del paso de los vehículos a alta velocidad con una precisión de 1 µs y tiempos registrados con una precisión en el orden de milisegundos. La principal contribución de este proyecto es ofrecer una alternativa de bajo costo que integra sensores fotoeléctricos, microcontroladores ESP32 y comunicación inalámbrica mediante módulos HC-12, permitiendo el registro y visualización en tiempo real de los datos a través de una aplicación móvil e interfaz gráfica. Como trabajo futuro se plantea la incorporación de parámetros adicionales, como la velocidad máxima y la telemetría avanzada, así como la optimización de la precisión del cronometraje e integrar una base de datos con los tiempos cronometrados por cada competidor mediante Python, con el fin de acercarse a los estándares de sistemas profesionales.

Referencias

[1] Burgess, P., & NHRA National Dragster Editor. (2017). A Christmas (Tree) Story. NHRA. https://www.nhra.com/news/2007/christmas-tree-story

[2] Ramos Monja, J. G., & Tigre Seminario, J. Y. (2022). Sistema de procesamiento de señales discretas para el registro de tiempo y velocidad de autos en carreras de aceleración.

[3] Post, R. C. (2001). High performance: the culture and technology of drag racing, 1950-2000. JHU Press.

[4] Dávalos Figueroa, D. S. (2013). Programación de Transponder en sistemas inmovilizadores automotrices de última generación (Bachelor's thesis, Universidad del Azuay).

[5] Laaroussi, S. (2019). Sistema para cronometraje deportivo basado en tecnología NFC (Bachelor's thesis, Universitat Politècnica de Catalunya).

[6] Miranda Tejada, P. J. (2024). Sistema de cronometraje para tramos WRC a escala.

[7] Vargas, D. C. (2017). Sistema de cronometraje para la Barcelona Smart Moto Challenge (Doctoral dissertation, Universitat Politècnica de Catalunya. Escola Tècnica Superior d'Enginyeria Industrial de Barcelona)

[8] Huerta Cristobal, J. J. (2020). *Diseño e implementación de un sistema de medidas de velocidad para pruebas de atletismo* (Doctoral dissertation, Universitat Politècnica de València).

[9] Rodríguez, Y., García, Y., & Galvis, E. (2017). Sistema de cronometraje inalámbrico de bajo costo para deportistas de alto rendimiento en el BMX. *Investigación Formativa en Ingeniería* 107.

[10] Pizarro Pelaez, J.(2020). *Internet de las cosas (IoT) con ESP. Manual práctico*. Ediciones Paraninfo, SA.

[11] *ESP32 Series Datasheet Version 5.1*. (s. f.). https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[12] User Guide of Photoelectric Sensor Switch M18 E3F-5DP1(Receiver) & E3F-5L(Sender). (s. f.). https://www.ett.co.th/productSensor/E3F-5DP1-5L/Manual%20of%20Photo-SensorSW%20M18-E3F.pdf

[13] HC-12 Manual de usuario traducido: (s. f.). https://uelectronics.com/wp-content/uploads/2021/02/HC-12-hoja-de-datos.pdf

[14] Peña, C. (2020). Arduino IDE: Domina la programación y controla la placa. RedUsers.

[15] Galeano, G. (2009). *Programación de sistemas embebidos en C*. Alpha Editorial.

[16] Lozada, J. C. H., Flores, I. T., & Castillo, M. M. (2006). Unidad Básica de Comunicación Serial en un Microcontrolador. Polibits, (33), 3-7.

[17] Rodríguez Vinces, J. A., & Suarez Arreaga, J. D. (2025). *Implementación de una interfaz gráfica de control domótico usando HMI y ESP32* (Bachelor's thesis).

[18] Mahedero Biot, F. (2020). Desarrollo de una aplicación IoT para el envío de imágenes mediante el protocolo MQTT (Doctoral dissertation, Universitat Politècnica de València)

[19] Amaguaya Ramos, R. X. (2020). Análisis comparativo a nivel transaccional de brokers MQTT (Mosquitto, Mosca y Emq) con respecto a la disponibilidad en infraestructuras IoT ante ataques DDoS (Master's thesis, Quito, 2020.).

[20] Munasinghe, T., Patton, E. W., & Seneviratne, O. (2019, December). Iot application development using mit app inventor to collect and analyze sensor data. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 6157-6159). IEEE.