

Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Unidad de Formación:

TE2003B **Diseño de Sistemas en Chip**

Entregable de Reto Final

Profesor: Enrique González Guerrero

Raúl Peña Ortega

Grupo: 601

Equipo y porcentaje de participación:

Brenda Cruz Arango **A00836915** 100%

Ximena Lizeth Trejo Lavín **A01198557** 100%

Valeria Aranza Cerda **A01236733** 100%

Confirmamos que somos las autoras de este trabajo y que es la versión final. Se citaron debidamente las palabras o ideas de otras personas, expresando estas de forma escrita, oral o visual.

Monterrey, Nuevo León, México, **13 de junio del 2024**

Índice

Sección	Página(s)
<i>Introducción</i>	3-4
- <i>¿Por qué es importante tener un sistema inteligente de conducción?</i>	3
- <i>¿Qué tipo de soluciones actuales existen para los sistemas de conducción?</i>	3-4
<i>Marco teórico</i>	5-7
- <i>Sistemas embebidos</i>	5-6
<i>¿Qué es un sistema en chip?</i>	5
<i>¿Qué es un microcontrolador?</i>	5
<i>¿Cómo facilita un microcontrolador a la creación de nuevos productos?</i>	5-6
- <i>Sistemas operativos en tiempo real para sistemas embebidos</i>	6
<i>¿Qué es un RTOS?</i>	6
<i>Ventajas y desventajas de usar un sistema operativo en tiempo real sobre el diseño de una aplicación Bare Metal.</i>	6
- <i>Linux Embebido</i>	7
<i>¿Qué es Linux embebido?</i>	7
<i>Justificación del problema</i>	8
<i>Problemática</i>	8
<i>Metodología</i>	9-20
- <i>Descripción</i>	9-11
<i>Identificación del problema</i>	9

<i>Conceptualización</i>	9
<i>Prototipado</i>	9-10
<i>Implementación final y validación</i>	10-11
- <i>Hardware utilizado</i>	11-12
- <i>Software utilizado</i>	12
- <i>Recursos utilizados del STM32 y Raspberry Pi</i>	12
- <i>Diagrama esquemático</i>	12-13
- <i>Diagrama de interacción de las tareas (RTOS)</i>	13-14
- <i>Diagramas de flujo por tarea</i>	14-17
- <i>Scheduling de Tareas (RTOS)</i>	18
- <i>Acciones de mejora llevadas a cabo</i>	18-19
- <i>Interfaz gráfica desarrollada en Python</i>	19-20
<i>Conclusiones y reflexiones individuales</i>	21
- <i>Ximena</i>	21
- <i>Valeria</i>	21
- <i>Brenda</i>	21
<i>Trabajo a futuro</i>	22
<i>Referencias bibliográficas</i>	23

Introducción

¿Por qué es importante tener un sistema inteligente de conducción?

Los sistemas inteligentes son una parte importante de todos los autos modernos. Las funciones de los vehículos se concentran en una interfaz y estos nos brindan información que son utilizadas para indicarnos recomendaciones de conducción, la navegación, entre más. Hablando desde el punto de vista empresarial para nuestro socio formador, un sistema inteligente de conducción puede ser de mucho valor. Su valor radica en qué tan beneficioso es para la empresa, que se tenga un costo eficiente y que este mismo sistema sea sostenible a lo largo del tiempo. Es importante tener en cuenta esto, ya que esto también permite que la integración de todas las tecnologías sean seguras y rentables, lo que garantiza un éxito para una empresa como John Deere.

En un contexto más específico, podemos mencionar que un sistema inteligente nos permite tener mayor seguridad. En cuanto a datos, esto mejora mucho en su transmisión, teniendo una conexión en tiempo real, y así mismo se puede compartir esta información con dispositivos externos.

¿Qué tipo de soluciones actuales existen para los sistemas de conducción?

Actualmente en la industria de los sistemas de conducción existen soluciones para diferentes problemas. Por ejemplo, se ha incrementado mucho la implementación de asistentes inteligentes de velocidad. Con esta mejora se puede respetar el límite de velocidad y adaptar el vehículo a las señales de la pista donde circule. También el detector de marcha atrás, con el que se puede detectar si hay alguna persona u objeto detrás de nuestro vehículo. El último que también es muy importante, es un ESS (sistema de frenado de emergencia). Con él se reducen los efectos de colisiones ante frenadas muy inesperadas. A continuación, presentamos una gráfica que nos ayuda a comprender cómo los microcontroladores aplicados en la industria automotriz nos permiten mejorar en los sistemas inteligentes de conducción.

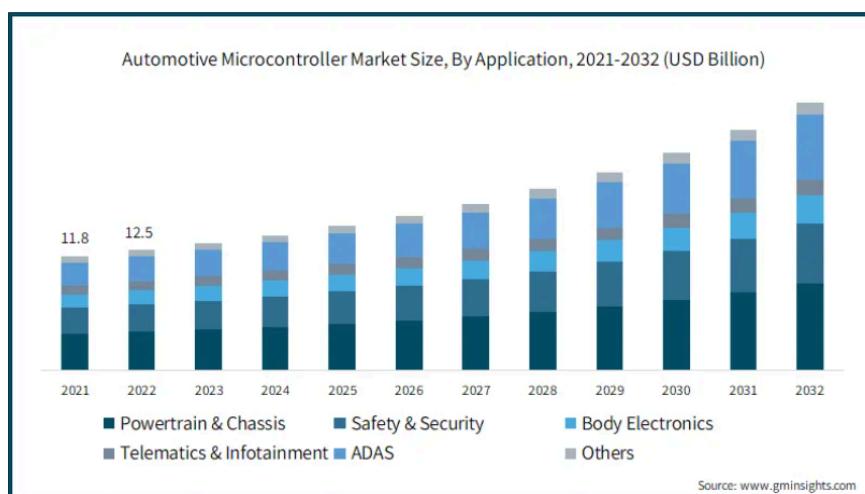


Figura 1. Tamaño de distintas industrias de sistemas automotrices.

Como vemos, las ADAS forman parte de estas aplicaciones, y es por ello que a lo largo de este reporte se encontrará un análisis muy específico de nuestro prototipo desarrollado para John Deere. Podemos crear diseños estables en tracción y seguridad, donde cada vez un automóvil exime más capacidad de procesamiento y rendimiento. Para el caso de John Deere, la implementación en la gestión de comunicación de las herramientas de sus prototipos con todas las seguridades que conlleva un tractor.

Marco Teórico

Sistemas embebidos

¿Qué es un sistema en chip?

A lo largo del curso y en el desarrollo de este proyecto aprendimos distintos conceptos como lo que es un sistema en chip, la parte de microcontroladores, comprendimos su uso y aplicaciones. En este caso, por medio de una Raspberry Pi (RPi) proporcionamos una solución a nuestro socio formador en la cuestión de transmisión de datos, y un modelo proporcionado, incluyendo una STM32 NUCLEO-F103RB. Cuando hablamos de sistemas en chip hacemos referencia a un circuito integrado que contiene muchos componentes de una computadora. Tienen muchas ventajas, como la parte de costos al poder fabricarse en grande escala, y también tienen un mayor rendimiento a beneficio de un menor retrato de propagación en cuestión de cables internos.

En cuanto al mercado de los sistemas en chip, podemos mencionar que este ha crecido de manera exponencial en los últimos años, y es por justamente como se mencionaba en clase, son implementados en muchos de los dispositivos electrónicos que tenemos en nuestros hogares. Cada vez se adaptan más estos dispositivos a redes 5G, lo que involucra internet de las cosas y dispositivos que se pueden aplicar en área médica.

¿Qué es un microcontrolador?

Un microcontrolador realiza una tarea específica y tiene distintos componentes, por ejemplo el procesador central y periféricos. Estuvimos trabajando con una STM32, es una placa muy accesible que nos ayudó a poder recibir datos de un potenciómetro y de un modelo, y a su vez transmitirlo a la RPi, con esto podemos visualizarlo en nuestra lcd, que más adelante se especifica. Una de las ventajas que encontramos al estar utilizando esta placa es la comunidad que la implementa en sus proyectos u otros fines, nos permitió poder encontrar información relevante que nos fuera de ayuda a nuestro proyecto, además de toda la documentación que ya se incluye en la datasheet.

¿Cómo facilita un microcontrolador a la creación de nuevos productos?

Un microcontrolador es de mucha utilidad en la creación de nuevos productos para distintos sectores. Gracias a esto podemos realizar tareas específicas, el mismo permite que exploremos nuestra creatividad para aplicarlo a una gran o pequeña escala. En cuanto a ventajas también podemos mencionar que son muy confiables y muy precisos a nivel industrial, además de ser muy prácticos y poderosos en su implementación. Para el caso de nuestro microcontrolador, la programación es en C/C++, y se requiere de conocimientos avanzados para poder utilizarlo. En cuanto a las desventajas, podemos encontrar la estructura compleja, que la programación requiere de un buen conocimiento en registros, la declaración de pines, módulos de comunicación y más.

Algo que es muy interesante y que involucra a los microcontroladores son los circuitos integrados de los vehículos, que controlan funciones relacionadas con el motor y la seguridad de todo el sistema. Creando nuevos productos brindan muchos beneficios, pero no solo es tomar en cuenta las ventajas a nivel *hardware* y *software*, si no que también existen normas que se deben seguir para la seguridad y calidad de cada empresa. Esto hace que los costos puedan subir, limitando ciertos mercados.

Sistemas operativos en tiempo real para sistemas embebidos

¿Qué es un RTOS?

Una parte esencial en nuestro proyecto fue el uso de un Real-Time Operating System. Fue la última parte que se agregó a nuestro proyecto. Esto sin mencionar las mejoras que aplicamos en cuanto a la parte física, es decir, los circuitos. Además de cómo fue que desarrollamos el Bare Metal, sin duda alguna esto definió mucho la trayectoria de todo el proyecto. Al utilizar Bare Metal pudimos tener un control total de todo el hardware.

En esta implementación pudimos darnos cuenta de cómo mediante este entorno se pueden ejecutar muchas tareas, que para nuestro proyecto definimos cuatro esenciales, además de subprocessos al mismo tiempo. Esta implementación hace que se gestione de manera adecuada todos los recursos del hardware, además de optimizar la respuesta en tiempo real. Es decir, mediante esta parte pudimos mejorar la cuestión de la API y la gestión de todas las tareas.

Ventajas y desventajas de usar un sistema operativo en tiempo real sobre el diseño de una aplicación Bare Metal

Reflexionar acerca de la importancia que tienen las tareas en los sistemas integrados como el caso de nuestro proyecto es fundamental para llevar un seguimiento adecuado. Aquí podemos tomar en cuenta las implicaciones del consumo de memoria. En RTOS se requiere memoria para las estructuras de datos como los *stacks*, y las llamadas a API generalmente pueden ser lentas en comparación con la interacción directa del hardware.

En cuanto a la seguridad de nuestro proyecto, que es un punto relevante, aunque con Bare Metal se tiene un control de casi todo, existen ciertas limitantes como la posibilidad de cometer errores, y la administración. En conclusión, a pesar de que se tiene un acceso directo al hardware y que esto nos ayuda a adaptarlo a cuestiones específicas, la gestión de interrupciones, si no se manejan correctamente, puede provocar retrasos.

RTOS utiliza la programación *preemptive*, que hace que se ejecute la tarea con mayor prioridad lo antes posible y que esto no retrase todo con las tareas que no tienen tanta prioridad. Lo único en desventaja que podríamos mencionar es la parte de gastos adicionales, que pueden afectar la velocidad de respuesta que se espera.

Linux Embebido

¿Qué es Linux embebido?

Linux embebido es un sistema operativo escalable (OS), no requiere una tarifa de uso para los usuarios que lo usan. De hecho es de código abierto, y una de las ventajas de este sistema operativo es su eficiencia. En cuanto a las desventajas de Linux embebido, no es tan ideal para aplicaciones en tiempo real, así como puede funcionar bien pero es lento con el procesamiento de señales analógicas. Pudimos notar esto debido a su implementación con los gráficos en tiempo real, donde mejorar el desfase fue algo tedioso.

Otro punto importante que tiene que ver con el Linux embebido es la programación con Bash scripts, la cual utilizamos para poder configurar la Raspberry Pi a través de los distintos comandos.

También utilizamos Python, que tiene muchas ventajas. La primera y más identifiable es la parte de la sintaxis, lo intuitivo que es poder utilizar este lenguaje. A través de la Raspberry Pi lo utilizamos para mandar datos a un archivo CSV, y también con ella creamos una API para mostrar los datos mediante gráficas en tiempo real. Gracias a que Python tiene una gran variedad de librerías fue más sencillo desarrollar código.

Queda por mencionar que para el caso de Linux utilizamos Thonny. Por último, resaltar que es un lenguaje muy versátil que cuenta con mucha documentación, además que es un lenguaje que gestiona en automático la memoria. Claramente tiene sus desventajas como el rendimiento (otros lenguajes tales como C y C++ son más rápidos al ser compilados), y también poder gestionar el uso de memoria de manera adecuada puede ser una limitante para los desarrolladores.

Justificación del problema

Probleática

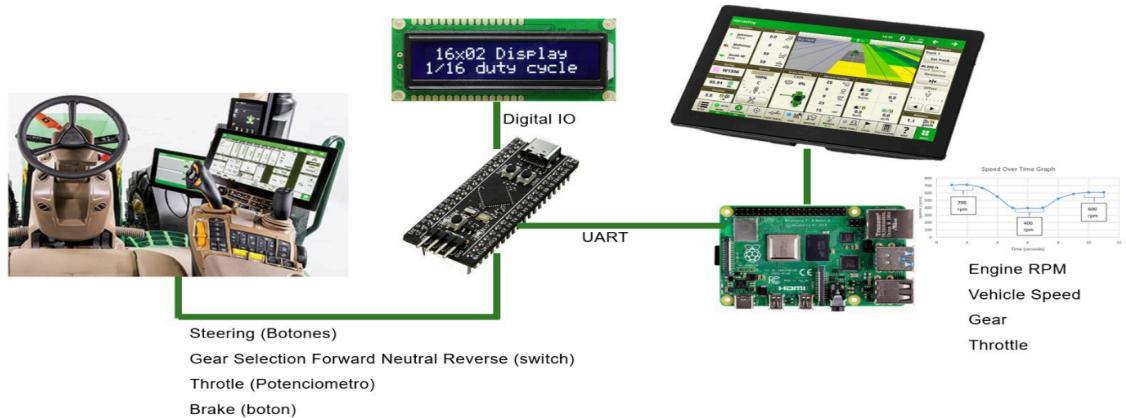


Figura 2. Layout del Reto en términos del socio formador.

En nuestro proyecto se busca un análisis de datos obtenidos mediante un modelo. Y como ya se mencionó, la implementación de un teclado matricial para manipular el freno y volante, un potenciómetro para controlar el acelerador, LEDs indicadores y un buzzer de direccionales.

Otro punto importante, es que nuestro socio formador John Deere necesita de precisión en sus tractores, y esto beneficia notablemente en su productividad. Además hablando de una mayor escala, tener lo más óptimo posible este aspecto reduce muchos costos que para una empresa es bastante significativo. Con esto se logra que John Deere siga siendo una empresa líder en cuanto a la tecnología agrícola. Por ejemplo, sus tractores incluyen sistemas de aceleración, control de bolsas de aire, control de frenos, control de inyección de combustible, control de emisiones, interfaz de instrumentos, entre muchos otros.

A mediano y largo plazo, se espera que la mayoría de los autos se manejen de forma autónoma, incorporando cada vez más sistemas inteligentes. Estos avances permitirán mejorar la calidad de vida mediante la reducción de accidentes y fatalidades. En el ámbito agrícola, facilitarán sistemas de precisión en el sembrado. Los sistemas inteligentes son una parte crucial de los automóviles modernos. Estos dispositivos concentran en una sola interfaz la mayoría de las funciones del vehículo, proporcionando información, navegación, recomendaciones de conducción, entre otros servicios.

Metodología

Descripción

Identificación del problema

Lo primero que realizamos fue la parte de poner en análisis todos los requerimientos del problema a resolver. Se nos pidió un sistema inteligente de conducción enfocado en la graficación de datos en tiempo real. Determinamos cómo sería la manera más adecuada de poder resolver la problemática. Conforme avanzamos en el contenido de clase, se añadieron mejoras. Realmente en esta primera parte comprendimos el contexto y situación de nuestro socio formador.

Para el primer entregable solo fue graficar en Python datos aleatorios y almacenarlos en un CSV. Lo que obtuvimos de aquí fue la base para lo que sería la graficación en tiempo real con los datos obtenidos del potenciómetro y el teclado matricial para las demás variables. Utilizamos *matplotlib* para este caso, y de hecho agregamos threads, que nos sirven para compartir recursos con la memoria y las variables globales.

Conceptualización

En esta segunda etapa se hizo la funcionalidad de todos los requerimientos del Reto, por lo cual agregamos las gráficas en tiempo real. Mediante un potenciómetro se cambiaba la aceleración. Con los botones del *keypad* pudimos hacer que con el tres giremos a la derecha, con el uno a la izquierda y con el dos usar el freno.

Prototipado

El prototipado conllevó tomar en cuenta que se tuvo que pensar en el rediseñado de los circuitos para todos los componentes, así como en un medio para manipularlos de manera fácil. Además de esto, se incluyó un modelado en 3D de una caja que simularía el panel de un tractor de John Deere. El software que se utilizó fue Shapr3D:

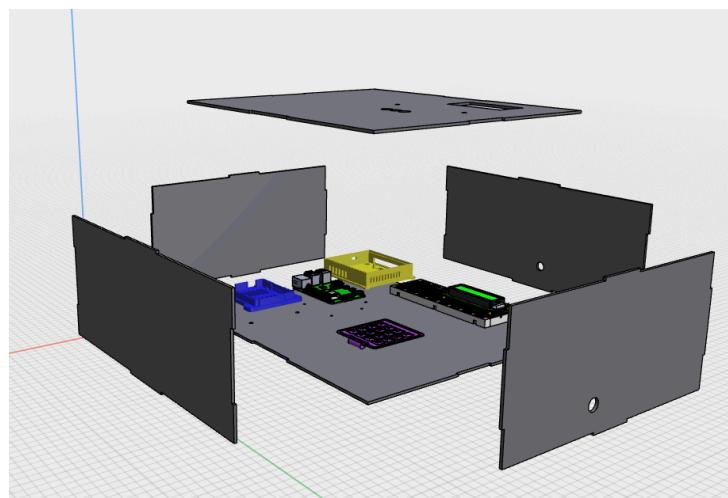


Figura 3. Modelado 3D de bocetos para corte láser del prototipo.

Por medio de esto, se pudo tener una manera de distribuir todo de mejor manera, lo cual significó un punto bueno en relación a la dedicación del equipo. Además de esto, se soldaron algunos elementos como los potenciómetros, ya que era mejor tenerlos fijos en alguna pared que tenerlos en la protoboard, de donde muy probablemente se desconectarían.

Implementación final y validación

Después de haber cortado en láser, ensamblado y acomodado todo para tener el producto final, se obtuvo una visualización de los datos en tiempo real del modelo de un tractor de John Deere de manera fluida. Por medio de esto, se pudieron manipular elementos como el acelerador, las direccionales y el freno de forma física (potenciómetro y keypad, respectivamente), así como de manera digital (a través de una interfaz GUI en Python desde la Raspberry Pi).

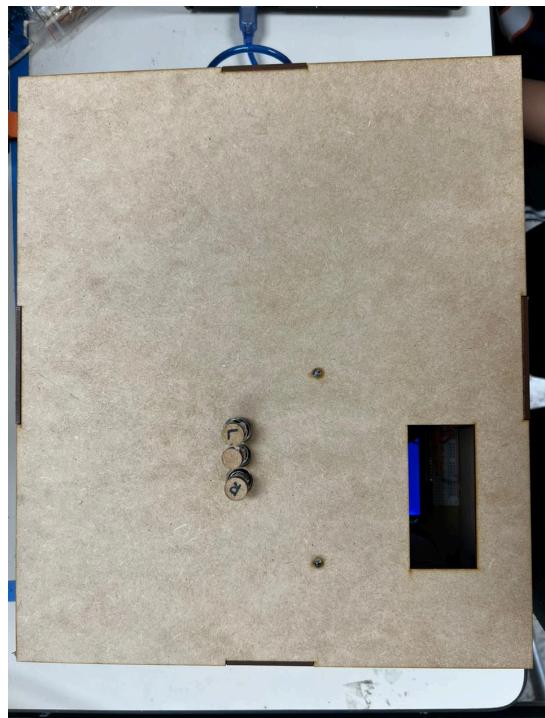


Figura 4. Vista superior del prototipo

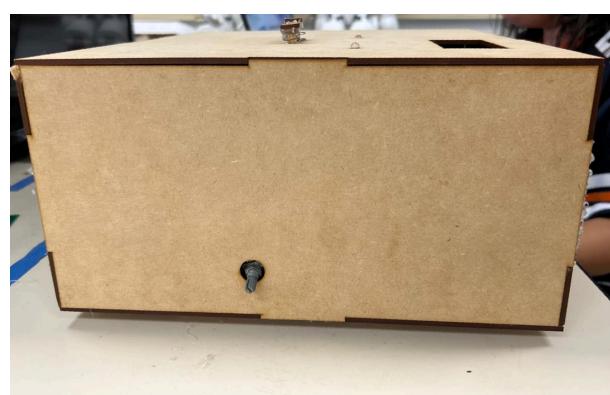


Figura 5. Vista lateral derecha del prototipo. Se ve el potenciómetro correspondiente al acelerador.

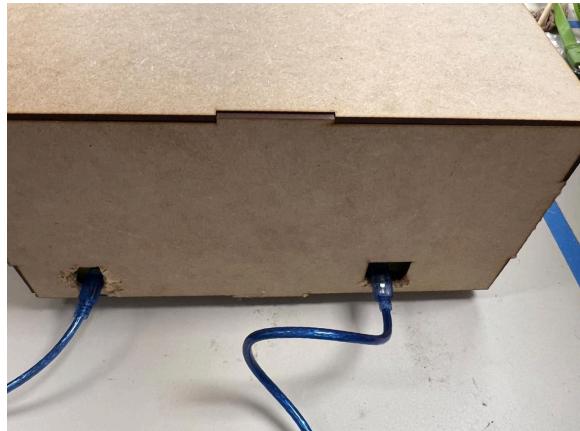


Figura 6. Vista lateral izquierda del prototipo. Se aprecian las entradas de los cables de alimentación.

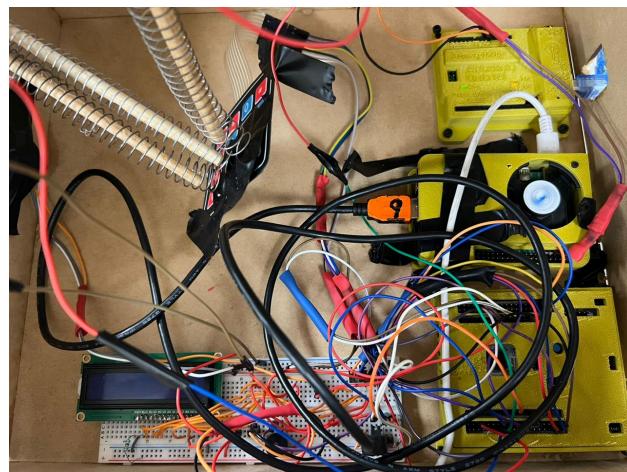


Figura 7. Vista interior del prototipo. Se aprecian los circuitos y los componentes utilizados.

Hardware utilizado

Dentro del hardware que utilizamos se encuentra la STM32 NUCLEO-F103RB y la Raspberry Pi B4 como tarjetas de desarrollo. Nuestro prototipo también cuenta con numerosos componentes electrónicos. Utilizamos dos potenciómetros, uno para el control de la aceleración y el otro para el contraste de nuestra LCD. Un *keypad* que nos indica las direcciones y el freno, y también un buzzer conectado para que suene como direccionales. Lo siguiente fue una pantalla LCD 16x2 para la impresión de datos como la velocidad del motor, velocidad del vehículo, marcha y velocidad del motor. También utilizamos un FTDI para la comunicación serial a través de la UART. En la última etapa también agregamos dos LEDS, donde se prende uno para la derecha y otro para la izquierda.



Figura 8. Hardware utilizado.

Software utilizado

El software con el que configuramos la STM32 es justamente STM32 CubeIDE. El desarrollo es en C/C++, y con esto pudimos crear el proyecto presentado. Ofrece configuración de periféricos, y es una plataforma que requiere de un guía para poder comprenderla. Otro de los software utilizados fue Thonny para el caso de la Raspberry Pi. El desarrollo es en Python, un lenguaje de programación muy intuitivo. Por último, agregar que también estuvimos trabajando con el sistema operativo Linux, que ya se especificó.



Figura 9. Software utilizado.

Recursos utilizados del STM32 y Raspberry Pi

Dentro de los periféricos de entrada se tuvieron componentes como potenciómetros y el keypad. Un periférico de salida fue la LCD, así como actuadores como el buzzer y los LEDs. Dentro del código de la STM se inicializó cada una de las funcionalidades. Para la comunicación serial se utilizó el módulo FTDI, que sirve como un convertidor de voltaje entre la Raspberry Pi y la STM32.

Diagrama esquemático

A continuación, se muestra la manera en que se conectaron los distintos componentes utilizados en el Reto. Es importante aclarar que no se conectó algo a los GPIOs de la Raspberry Pi, ya que la conexión fue por medio del puerto USB al FTDI.

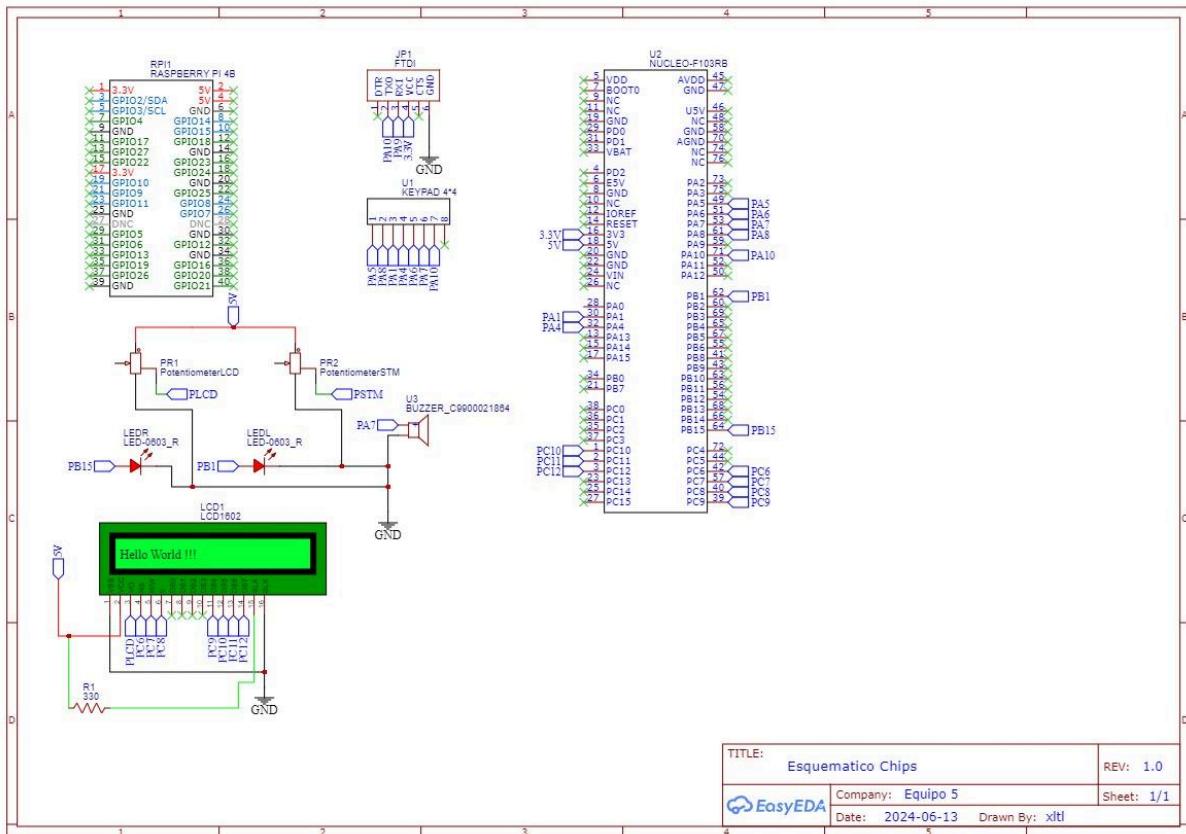


Figura 10. Diagrama esquemático de las conexiones realizadas en el Reto.

Diagrama de interacción de las tareas (RTOS)

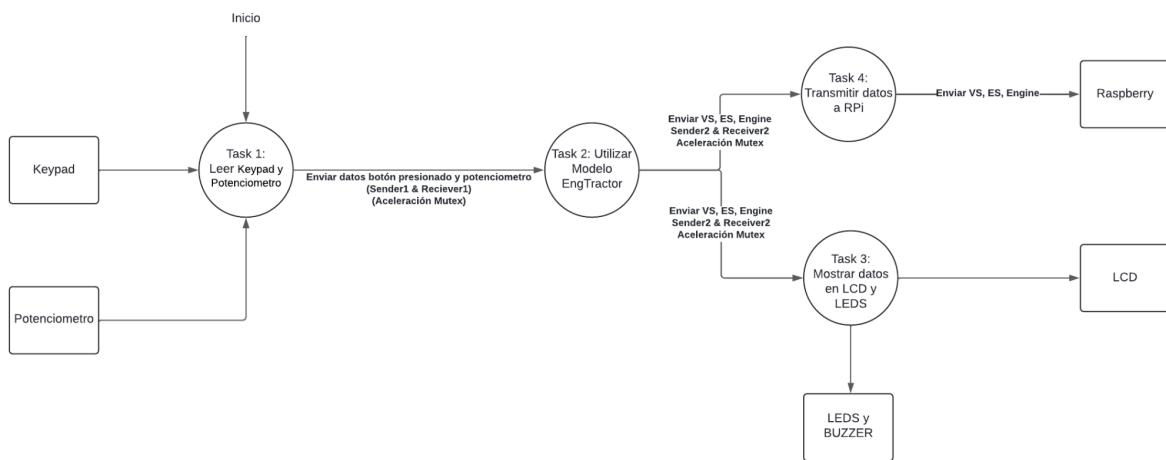


Figura 11. Diagrama de interacción de las tareas (RTOS).

En el diagrama anterior se muestra la interacción entre las 4 tareas que definimos. Como vemos, tenemos los datos de entrada que se obtienen del *keypad* y del *potenciómetro*, por lo que nuestra tarea 1 es la lectura de estos datos. La tarea 1 se conecta con la tarea 2, que es usar el modelo que nos proporcionaron. Se especifica un *sender* y *receiver*. En cuanto a la aceleración implementamos un *mutex*, ya que nos

dimos cuenta que es un recurso compartido entre las tareas. La tarea 2 se conecta a la tarea 4, que transmite los datos a Raspberry para poder ser graficados, y la tarea 3 que muestra los datos en la LCD y los LEDs.

Diagramas de flujo por tarea

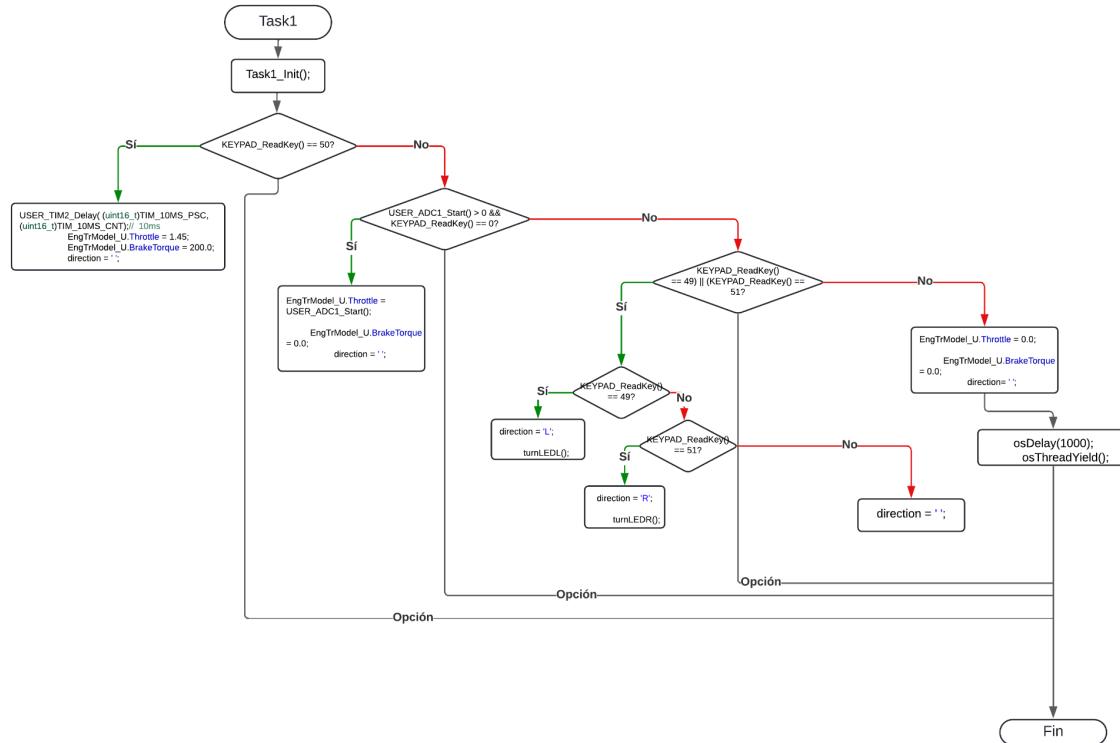


Figura 12. Diagrama de flujo de la Tarea 1.

La primera tarea recibe datos con el *keypad*. Es un bucle que está evaluando si se presiona un determinado botón y qué acción hacer. Por ejemplo, si se presiona el botón 2 (50 en ASCII), se realiza cierta acción del modelo que se nos proporcionó. Si se presiona el botón 1 (correspondiente a 49 en ASCII) manda la indicación que la direccional a la izquierda fue alertada. Si se presiona el botón 3 (correspondiente a 51 en ASCII) se manda la direccional derecha. El *osDelay* y *osThreadYield* salen de los condicionales, por lo que se siguen ejecutando.

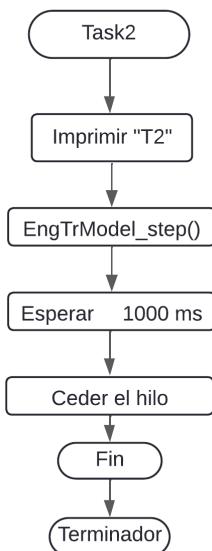


Figura 13. Diagrama de flujo de la Tarea 2.

La Tarea 2 llama al modelo que se nos dió, Engine Model, e imprime primero un mensaje. Esto se hizo para verificar en la consola de la STM que realmente se estuviera ejecutando la tarea con la prioridad que le habíamos definido. Posteriormente, también hay un delay, y la función `osThread()`, termina la función.

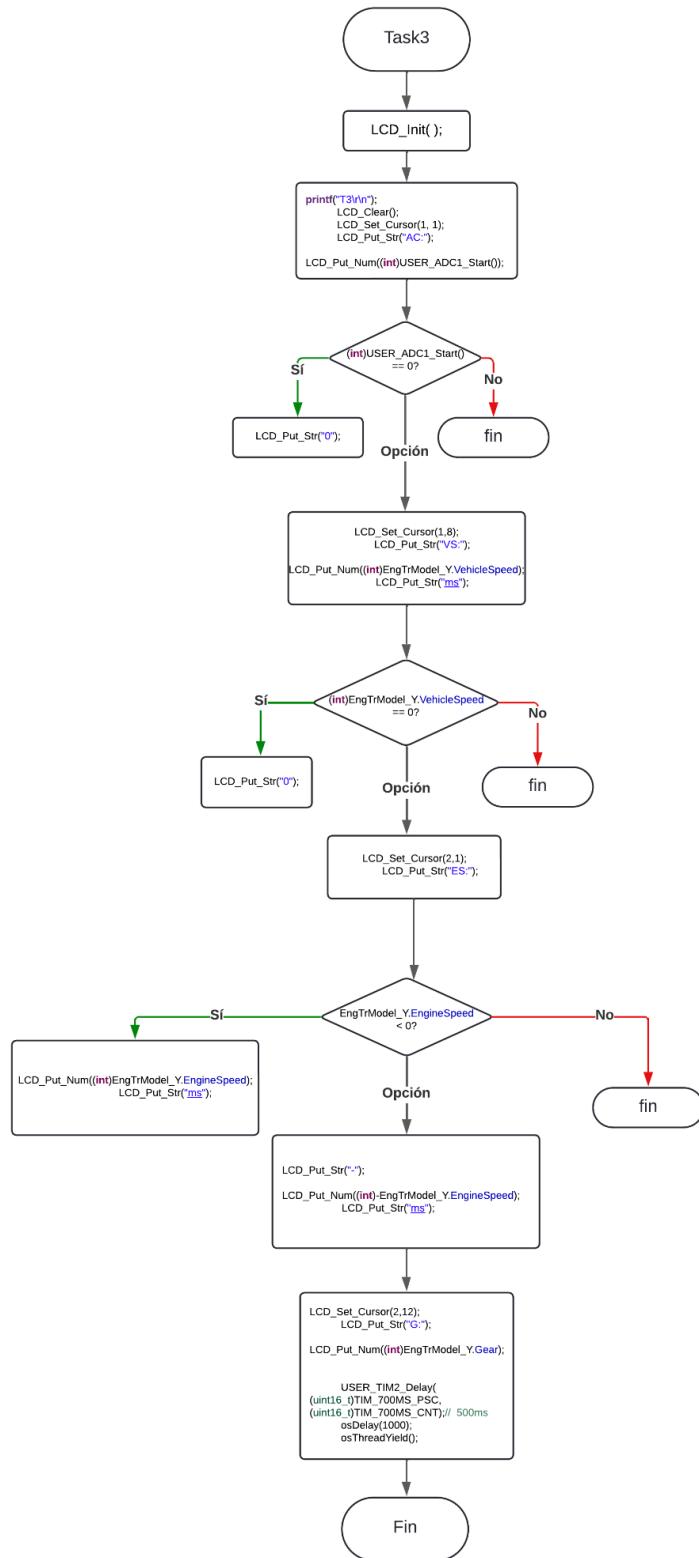


Figura 14. Diagrama de flujo de la Tarea 3.

La Tarea 3 muestra los datos en la LCD y en los LEDs. Se inicializa la LCD. Posteriormente se limpia la pantalla, se muestra el valor del acelerador, y luego se muestra el valor numérico de la función `USER_ADC1_Start()`, y si el valor es 0. Se incluye

posteriormente la parte del modelo en cuanto a la velocidad del vehículo y del motor. La marcha también se muestra, y se introduce un *osDelay()* y un *osThreadYield()*.

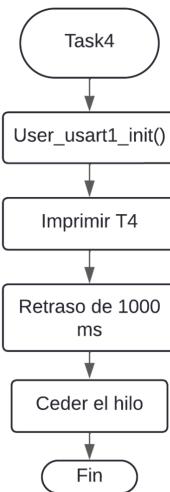


Figura 15. Diagrama de flujo de la Tarea 4.

Esta cuarta tarea transmite los datos de la STM a la Raspberry Pi. Entonces se inicializa la UART, se imprime un comentario para comprobar el orden en que se ejecutan las tareas, se retrasa y llama a la función *osThread()*.

Scheduling de Tareas(RTOS)

Características de las tareas RMS

Task	Period (ms)	Execution time (ms)	Priority
T1	200	50	1
T2	2000	300	3
T3	6000	1055	4
T4	500	50	2

Timeline de la calendarización de las tareas del sistema

T39 ₀	T35 ₀	T45 ₀	T39 ₁	T39 ₂	T35 ₁	T44 ₀	T39 ₃	T35 ₂	T39 ₁₀	T35 ₃	T45 ₁	T39 ₁₁	T35 ₄	T39 ₁₃	
0	50	100	399	400	450	500	550	1605	1955	2005	2055	2155	2455	2555	2605 2654

Características de las tareas EDF

Task	Period (ms)	Execution time (ms)
T1	360	50
T2	1800	300
T3	1800	1055
T4	600	50

Timeline de la calendarización de las tareas del sistema

T28 ₀	T25 ₀	T23 ₀	T28 ₁	T28 ₂	T25 ₁	T28 ₃	T26 ₀	T25 ₂	T28 ₄	
0	50	100	1155	1205	1255	1305	1355	1655	1705	1754 1755

Acciones de mejora llevadas a cabo

Antes de finalizar con el último entregable, realizamos ciertas acciones de mejora. Revisamos distintas variables que sentimos podrían afectar la parte del despliegue de datos en tiempo real. Por ejemplo, comprobamos que el *baud rate* no nos diera algún problema. Intentamos modificar los delays, pero esto no ayudó, de hecho se alentó más. Entonces, lo que decidimos hacer fue una investigación y búsqueda en foros de internet que nos pudieran ser de utilidad. Encontramos que se podía modificar la latencia del FTDI. También hicimos mejoras en cuanto al código de la STM, porque aunque sí se reducía el desfase, aún era bastante notorio. Entonces, tuvimos que

obtener la frecuencia con que la STM enviaba los datos, y teniendo esto modificamos el delay. Los hilos y la librería pyQT que utilizamos fue una gran mejora, para la graficación e interfaz.

Es muy importante el monitoreo en tiempo real de todos los datos, y en el contexto de nuestro proyecto se ve reflejada la importancia en la interpretación de estos datos, ya que las gráficas que se muestran a través de la Raspberry Pi deben de ser lo más precisas posibles. Estos datos pueden ser utilizados para la toma de decisiones en ciertos procesos de automatización en el caso de nuestro tractor, y así mismo podemos medir la eficiencia de nuestro sistema.

Por lo que, si obtenemos datos precisos del comportamiento del tractor, podemos realizar un análisis preciso (en este caso la graficación es parte de este análisis), que nos permita obtener resultados y conclusiones precisas para nuestras soluciones. Si queremos un sistema inteligente debemos de aprender a registrar patrones para identificar anomalías a tiempo y prevenir accidentes. Nosotras como estudiantes de ingeniería en Robótica estamos comprometidas en la mejora de estos procesos, explorando más a detalle las estrategias y metodologías necesarias que mejoren nuestro sistema.

De los principales problemas que encontramos fue el desfase de los datos que se estaban enviando y los que estábamos recibiendo, lo cual se pudo arreglar con el ajuste de delays, y la aplicación de otras librerías para graficación. Esto significó una mejora en la interacción con la interfaz de datos, lo cual fue un cambio para mejor. Aplicando las estrategias de mejora, se pudo tener una interfaz de datos más eficiente. A través de la implementación correcta de esto, se pudo corregir uno de los problemas más persistentes dentro del desarrollo del proyecto. Una vez se pudo resolver esto, las demás tareas empezaron a trabajar de mejor manera.

Interfaz gráfica desarrollada en Python

En la implementación final, se utilizó la librería *PyQtGraph* para poder graficar los datos en tiempo real. Esta librería es usada por modelos altamente exigentes para ámbitos como la ingeniería. La interfaz gráfica consiste de dos ventanas: una con cuatro gráficas (velocidad del motor, velocidad del vehículo, marcha y acelerador), así como otra que tenía botones para manipular las direccionales y el freno.

Caso de uso: manipulación de acelerador

Al momento de girar el potenciómetro en un sentido, este mismo cambia el valor de la gráfica del acelerador. Se puede ver una curva por este mismo movimiento. Este caso de prueba se narra en el video de evidencia.

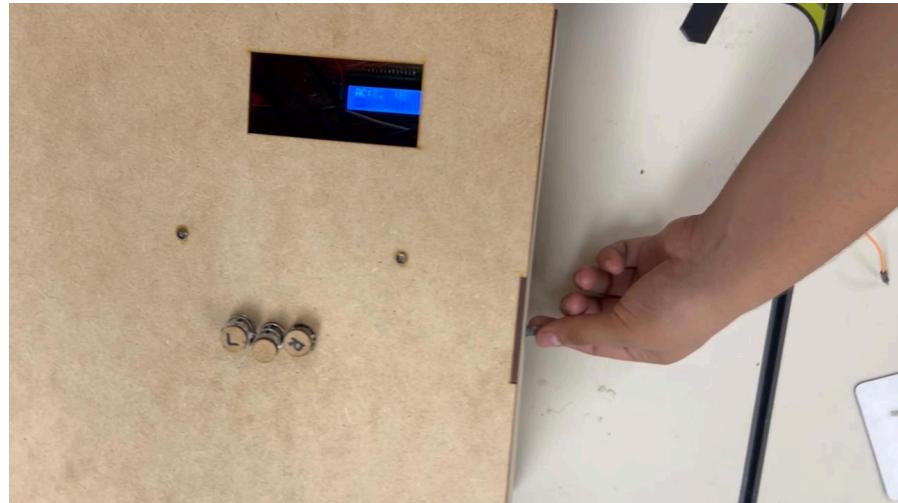


Figura 16. Giro de potenciómetro.

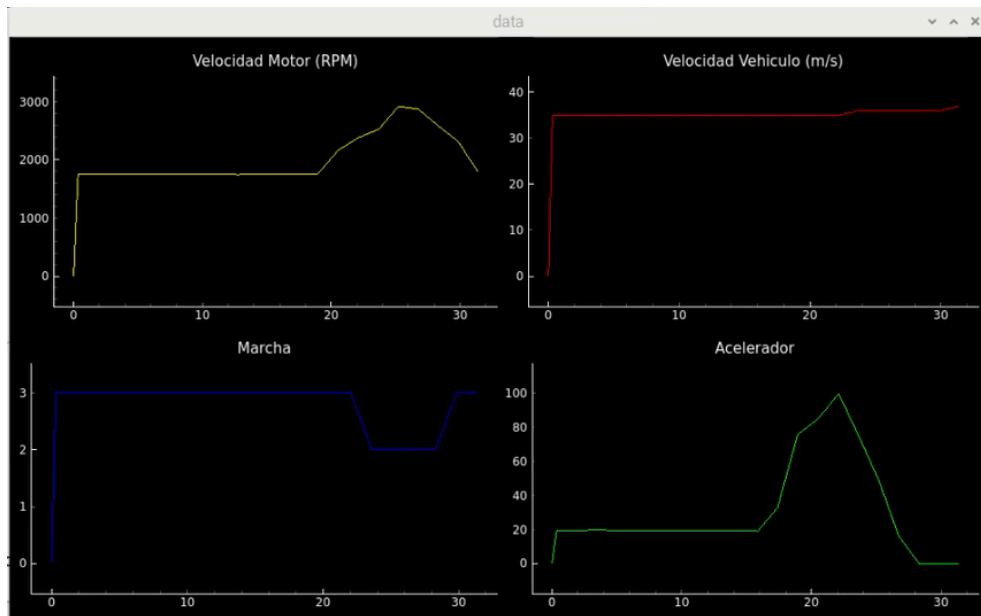


Figura 17. Gráficas de datos. El acelerador tiene un cambio en su valor, lo que se refleja en la gráfica.

Conclusiones y reflexiones individuales

- Ximena

La interfaz de un microcontrolador y un microprocesador ayuda a tener un mejor panorama sobre cómo funciona un sistema como lo fue el de este Reto. Se pudo implementar un protocolo de comunicación, se vieron datos generados por componentes físicos y manipulación digital, entre otros beneficios. Los distintos desafíos que se presentaron a lo largo del desarrollo de este proyecto contribuyeron no solamente a saber más de los temas, sino también a tener mejores habilidades de resolución de problemas. Todo esto llevó a un mejor entendimiento de los temas, así como a nuevos conocimientos adquiridos.

- Valeria

Durante este reto aprendimos sobre la relevancia que tienen los tiempos y períodos de las tareas. Un incorrecto periodo u orden de ejecución puede crear un sistema lento o en el que algunas Tasks no se corran, por ello las herramientas de calendarización son clave. Así mismo el manejo de registros y timers que nos permitieron programar la STM32. El realizar este sistema inteligente que nos permite analizar comportamientos de un tractor tiene un gran impacto en la vida de las personas y la innovación en la tecnología, por ello aprender a desarrollar y eficientizar lo mencionado anteriormente es muy importante.

- Brenda

Con la realización de este proyecto me di cuenta de la relevancia que tiene usar microcontroladores y sistemas en chip para la realización de un prototipo. Tuve muchos aprendizajes. No tenía idea de lo que era un RTOS. Me gustó el tema, pero aplicarlo al proyecto fue un poco complicado. También averiguar cómo mandar datos de la Raspberry a la STM. Me encantó el poder implementar un prototipo con un sistema mecánico, aunque la modelación y la realización de corte en láser requirió de bastante tiempo, fue muy satisfactorio aplicar esos conocimientos. Sin duda es un proyecto de valor que me ayudará en los próximos proyectos de la carrera y la vida profesional.

Trabajo a futuro

Con el desarrollo de nuestro prototipo pudimos identificar áreas de mejora. Más adelante se pueden implementar estrategias que mejoren distintas partes de nuestro sistema. Por ejemplo, la parte del código y la parte mecánica, que en sí fue un punto adicional que añadimos para tener una mejor presentación. Se puede hacer todo el prototipo en impresión 3D, pero para esta ocasión no lo hicimos de esta manera, ya que esto conlleva mayor tiempo. Esto fue una de las cuestiones que nos generó problema, ya que tuvimos que ser bastante organizadas en el sentido de buscar los horarios disponibles para poder imprimir y cortar a láser.

También en el *hardware* se pueden agregar más sensores, lo que implicaría cambiar nuestro circuito y las conexiones a los pines de nuestra STM32, es decir, se tendría que volver a organizar el cableado del circuito.

Se puede explorar más las librerías de Python para la cuestión de los gráficos en tiempo real y la interfaz. Por ejemplo, librerías que incluyan C como *Cython*. También en Python (que involucra a la Raspberry Pi) se puede eficientar más el procesamiento de datos. Es decir, que nos pueda dar estadísticas que nos sean de valor para poder revisar las partes correctas y las que no.

Un punto que pudimos analizar mediante nuestra exposición y la retroalimentación de los demás equipos es la cuestión del modelo que nos proporcionó John Deere. Como equipo no lo modificamos. Es decir, implementamos el modelo sin alteraciones, por lo que no teníamos un alcance límite de RPM. Sin embargo, sí puede ser muy positivo tener un límite, ya que en un caso real el rango que se debe tener es entre 1500 y 2000 RPM.

Con el RTOS en la STM32 se podría ver la manera de mejorar los periodos de las tareas. Es muy cierto que un menor periodo nos permite tener una deadline más cercana, pero no necesariamente tener un periodo más corto es mejor (eso dependerá de cada tarea), por ejemplo, las tareas que contienen el modelo. El periodo debe ser más corto ya que no hay tanto que procesar, por el contrario en las otras tareas, donde si se realizan operaciones el periodo es mayor.

Ahora como una idea innovadora y con tecnología de vanguardia, nos dimos a la tarea de investigar acerca de la inteligencia artificial, los modelos predictivos, la parte de algoritmos, y una mejora que se puede añadir con mayor conocimiento sobre estos temas puede ser la parte de redes neuronales para lograr que nuestro prototipo sea entrenado y los datos nos sean aún más serviciales.

Referencias bibliográficas

PR Newswire Europe. (6 de marzo del 2013). *Automóviles, aeronáutica, defensa, salud. Altran integra los sistemas inteligentes en todo el mundo.* Spain. NewsBanck. Recuperado de

<https://infoweb.newsbank.com/apps/news/document-view?p=AWNB&docref=news/144DE2D9C0DB2D60>

NewsBank. (n.d.). *Tamaño de mercado de sistemas de casas inteligentes global, participación, análisis de inversión de la industria y pronóstico para 2025 con mejores aplicaciones.* Recuperado de

<https://infoweb.newsbank.com/apps/news/document-view?p=AWNB&docref=news/1723FAF862E895A0>

Marina Baranova. (07 julio 2022). *Así funcionan los 8 sistemas de seguridad que llevarán los coches nuevos desde el 6 de julio.* Neomotor. Recuperado de <https://neomotor.epe.es/conduccion/asi-funcionan-los-8-sistemas-de-seguridad-que-llevaran-los-coches-nuevos-desde-el-6-de-julio-YJNM8908>

Manuel J. Bellido Díaz. (Febrero de 2017). *Introducción al diseño de SoC.* Recuperado de <https://www.dte.us.es/docencia/master/micr/soc-basados-en-sistemas-abiertos-socbsa/temas/SoCIntro>

Janos Magasrevy. (18 September 2023). *RTOS vs. Bare Metal: Navigating Performance, Complexity, and Efficiency.* Weston. Recuperado de <https://weston-embedded.com/support/media-articles/119-rtos-vs-bare-metal-navigating-performance-complexity-and-efficiency>

PhoenixNAP. (Junio 21, 2023). *¿Qué es un sistema en un chip?* PhoenixNAP Recuperado de <https://phoenixnap.mx/glosario/sistema-en-un-chip>

DercoCenter. (8 de junio 2022). *¿Qué son las revoluciones por minuto de un motor?* . DercoCenter Recuperado de <https://www.dercocenter.cl/noticias/que-son-las-revoluciones-por-minuto-de-un-motor>

DercoCenter. (13 de junio 2023). *La industria de chips mundial y el futuro de los semiconductores..* Arrow Recuperado de <https://www.arrow.com/es-mx/research-and-events/articles/future-of-the-global-microchip-industry>