



Diseño de Sistemas en Chip

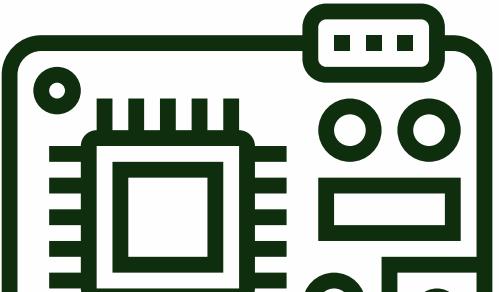
SISTEMA

INTELIGENTE

de conducción

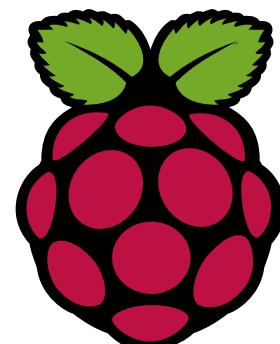
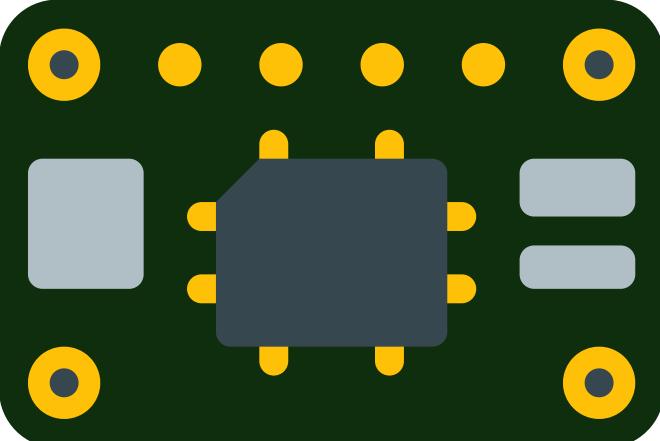
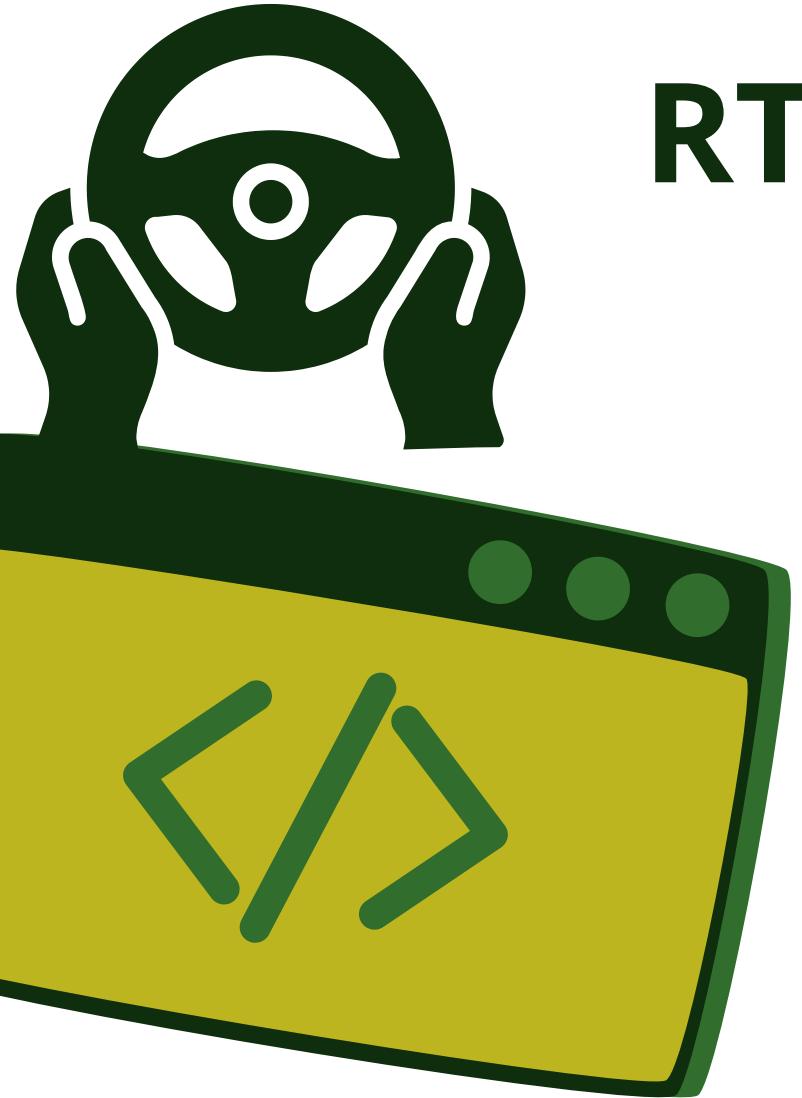


Proyecto realizado por:
Ximena Trejo
Valeria Aranza
Brenda Cruz



ÍNDICE

RTOS



01. Introducción

02. Justificación del Problema

03. Metodología

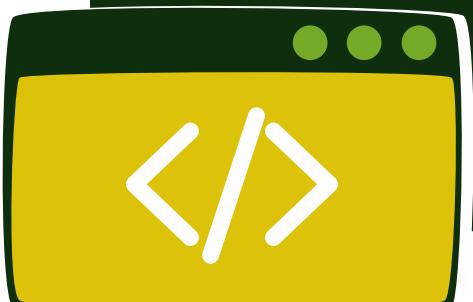
04. Acciones de mejora

05. Conclusiones y Reflexiones

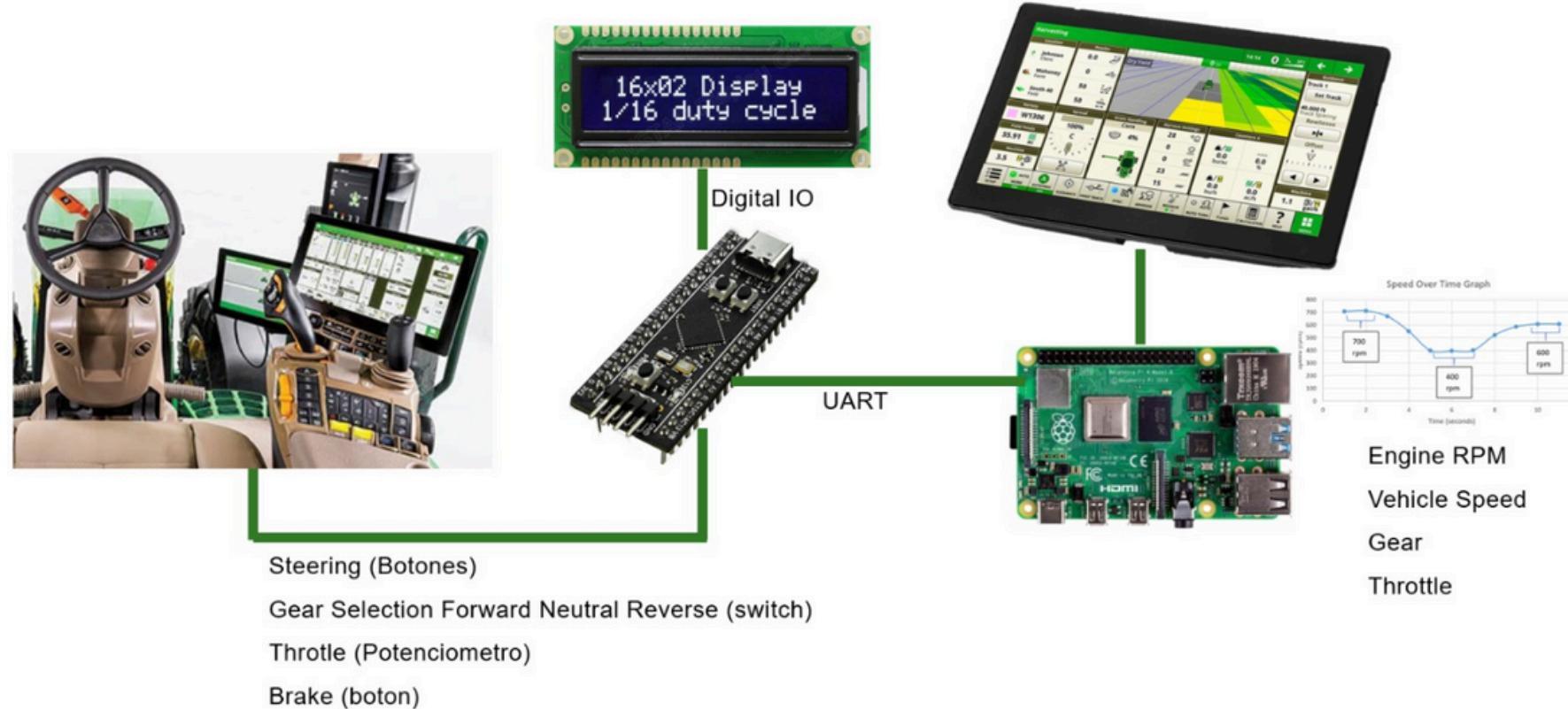
06. Trabajo a futuro

Introducción

En colaboración con John Deere, diseñamos un sistema inteligente de conducción, para la elaboración de este diseño se utilizó principalmente una STM32 F103 RB y una Raspberry Pi 4. Un sistema de conducción puede ser de mucho valor, es importante la seguridad, la mejora en transmisión de datos y conexión en tiempo real.



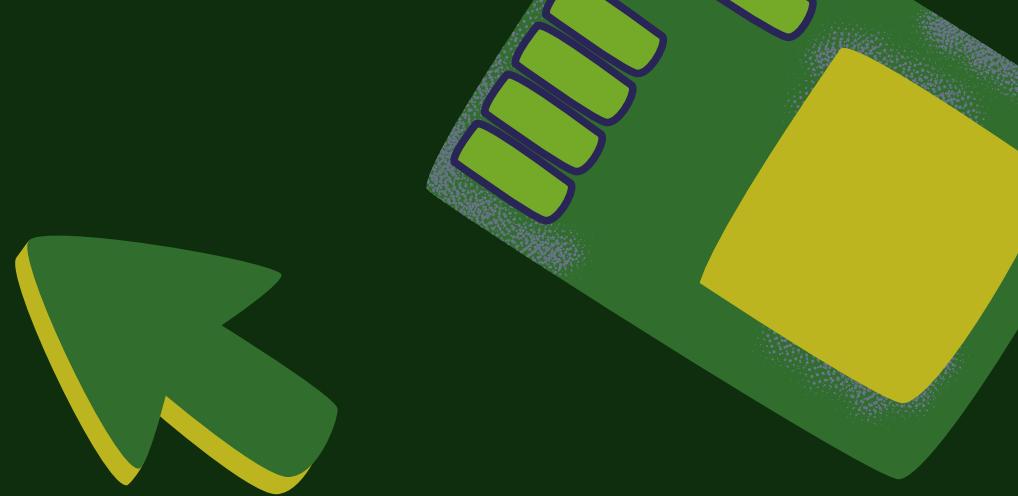
Justificación del problema



El reto consiste en diseñar un sistema inteligente de conducción de un tractor de campo agrícola. Utilizamos ciertos datos como la aceleración, dato de entrada para el modelo Engine Tractor Model proporcionado por el socio formador John Deere. Es fundamental el monitoreo en tiempo real de los datos, las gráficas deben ser lo más precisas posibles, esto permite que John Deere tenga muchos beneficios en cuanto a productividad y costos.

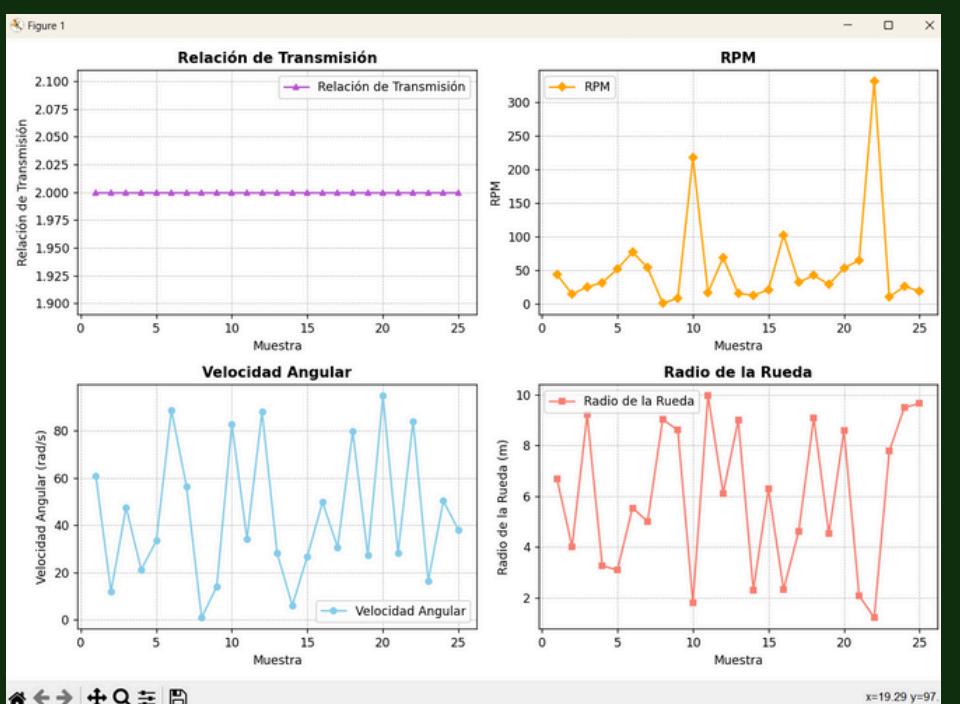


METODOLOGÍA



Etapa 1

En la primera etapa analizamos la problemática expuesta por el socio formador, después realizamos el primer entregable que consistió en desarrollar código en Python para poder graficar ciertos datos y guardarlos en un CSV, fue la base para nuestro desarrollo del proyecto en python dentro de linux.



The figure shows a dual-pane interface. The left pane displays the code for a C program named `main.c`, which includes functions for reading sensors and controlling actuators. The right pane shows a breadboard setup connected to a microcontroller board, with various components like resistors, capacitors, and a blue LCD screen.

```

// main.c
32 #include "main.h"
33 #include "USER_TMR1.h"
34 #include "USER_ADC1.h"
35 #include "USER_BNO055.h"
36 #include "LCD.h"
37 #include "KEYPAD.h"
38
39 void main()
40 {
41     USER_TMR1_Start();
42     USER_ADC1_Start();
43     USER_BNO055_Start();
44     LCD_Init();
45     KEYPAD_Init();
46
47     EngTModel_U_Throttle = 0.0;
48     EngTModel_U_BrakeTorque = 0.0;
49     direction = '+';
50
51     EngTModel_U_Gear = 1;
52     EngTModel_U_VehicleSpeed = 0.0;
53     EngTModel_U_BrakeTorque = 0.0;
54     direction = '+';
55
56     EngTModel_Stop();
57     EngTModel_Delay((uint16_t)TMR_500MS_PSC, (uint16_t)TMR_500MS_CNT); // 500ms
58     //EngTModel_SetSpeed(40000, (int)EngTModel_U_VehicleSpeed);
59     //EngTModel_SetGear((int)EngTModel_U_Gear);
60     //EngTModel_SetBrake((int)EngTModel_U_Brake);
61     //EngTModel_SetAccel((int)EngTModel_U_Accelerator);
62
63     printf("%d,%d,%d,%d", (int)EngTModel_U_VehicleSpeed, (int)EngTModel_U_Gear, (int)USER_ADC1_Start());
64 }

```

Build Analyzer | Static Stack Analyzer | Cyclomatic Complexity | Disassembly

Reto.elf - /ReTo - May 14, 2024, 5:27:15 PM

Memory Regions | Memory Details

Right pane details:

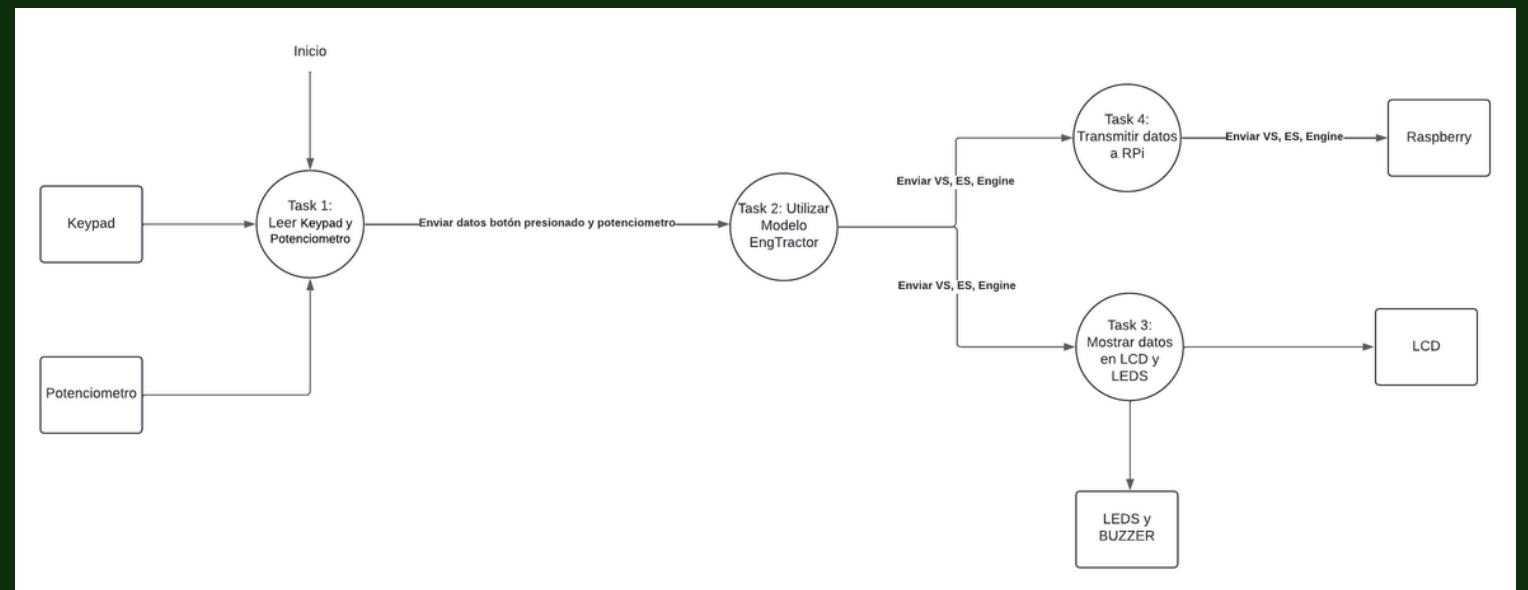
- User GPIO
- User USART
- User TMR2
- User ADC1
- User ADC2
- User KEV1
- User TMR1
- User direction
- User BNO055
- User I2C2
- User GPIO
- User TMR2
- User ADC1
- User ADC2

Etapa 2

Mediante el conocimiento adquirido de las clases pudimos generar un modelo de interacción por UART entre la RPI y la STM32, además de eso incluimos la representación de los datos en tiempo real. Se implementó teclado matricial, LCD y modelo de controlador de transmisión automática.



METODOLOGÍA



Etapa 3

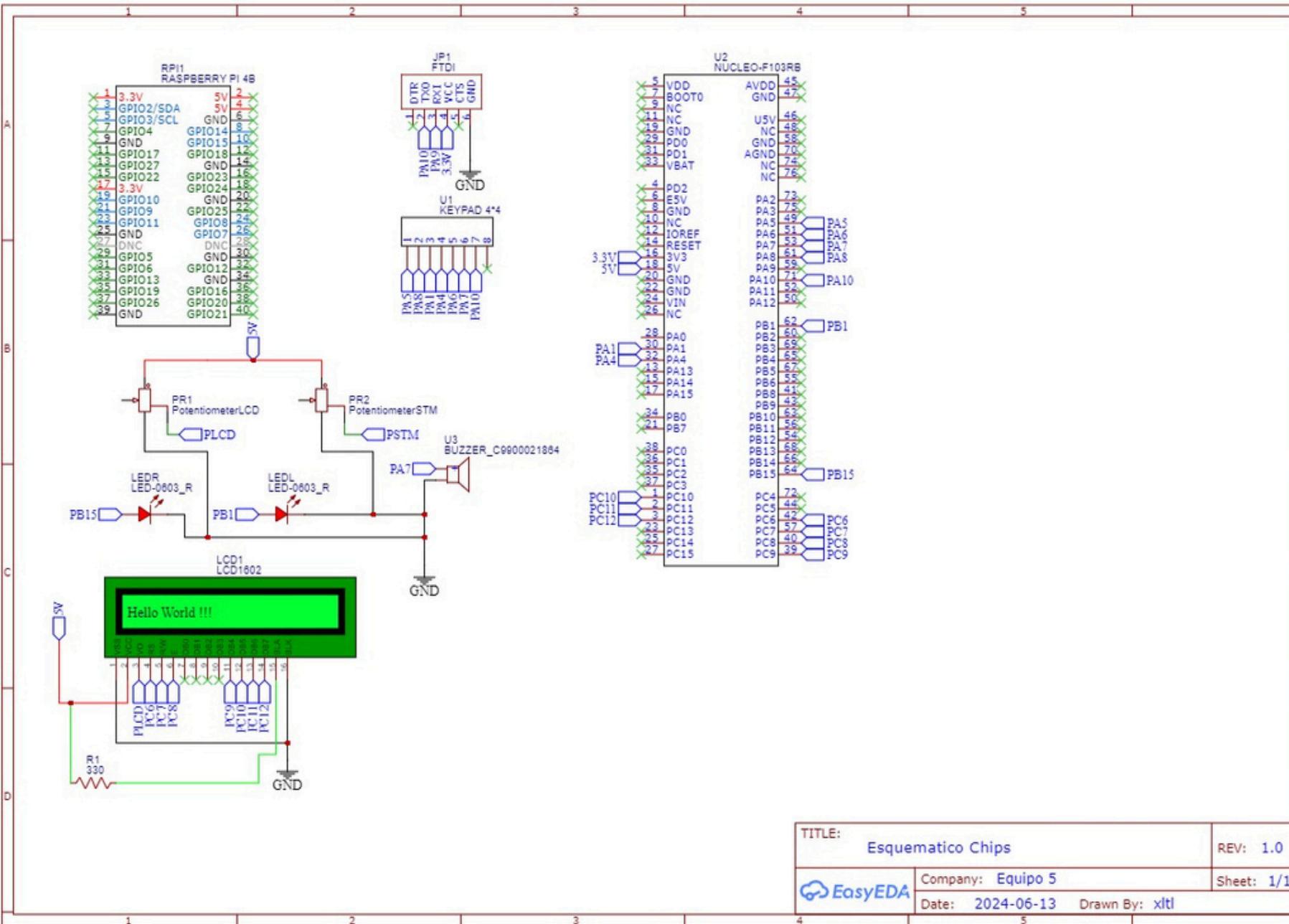
Se agregaron mejoras a la graficación en tiempo real, nos dimos cuenta de que influía en que hubiera desfase de los datos con respecto a lo que se recibía y se mostraba. Buscamos en foros de discusión para poder checar y configurar la latencia del FTDI



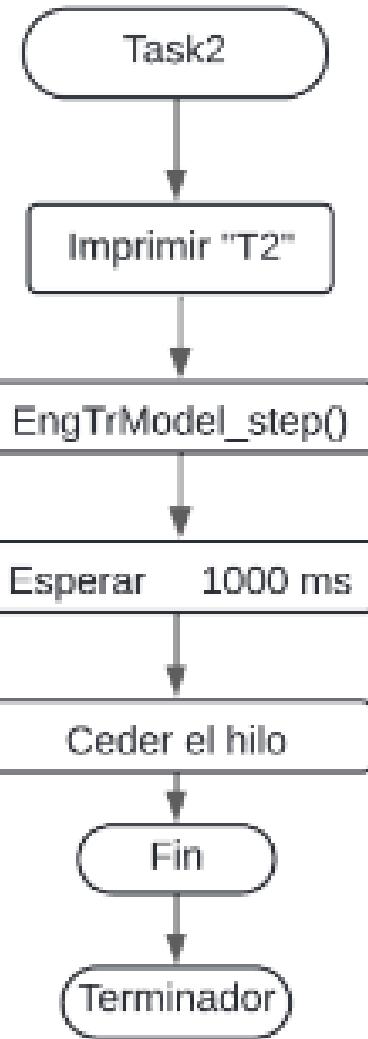
Etapa 4

Se realizó un diagrama de las tareas involucradas para la implementación de RTOS, calendarizamos las tareas, definimos recurso compartido. En cuanto a la funcionalidad del Keypad, ahora la dirección se ve reflejada con dos LEDs.

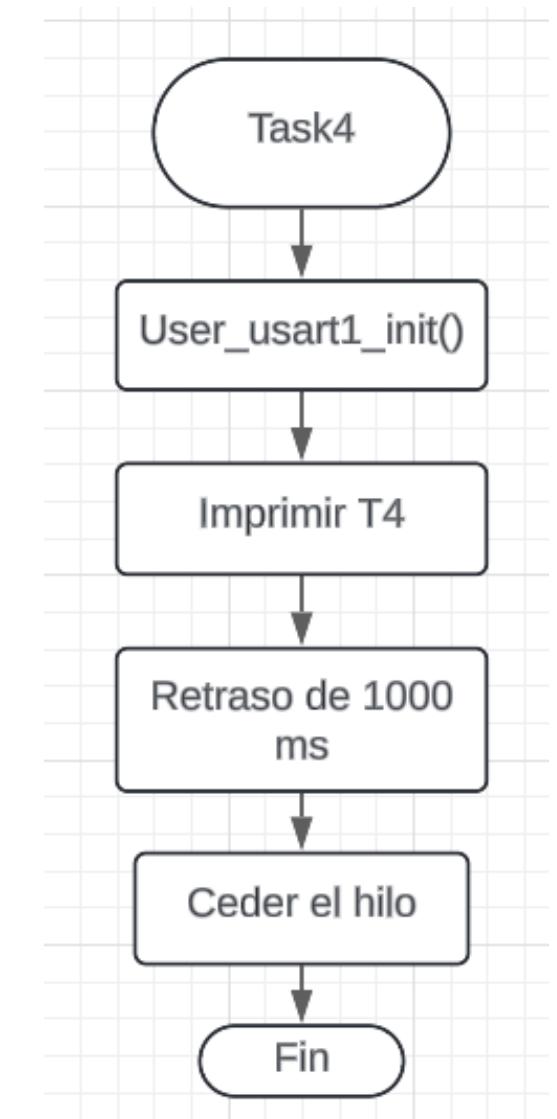
Diagramas



Esquemático

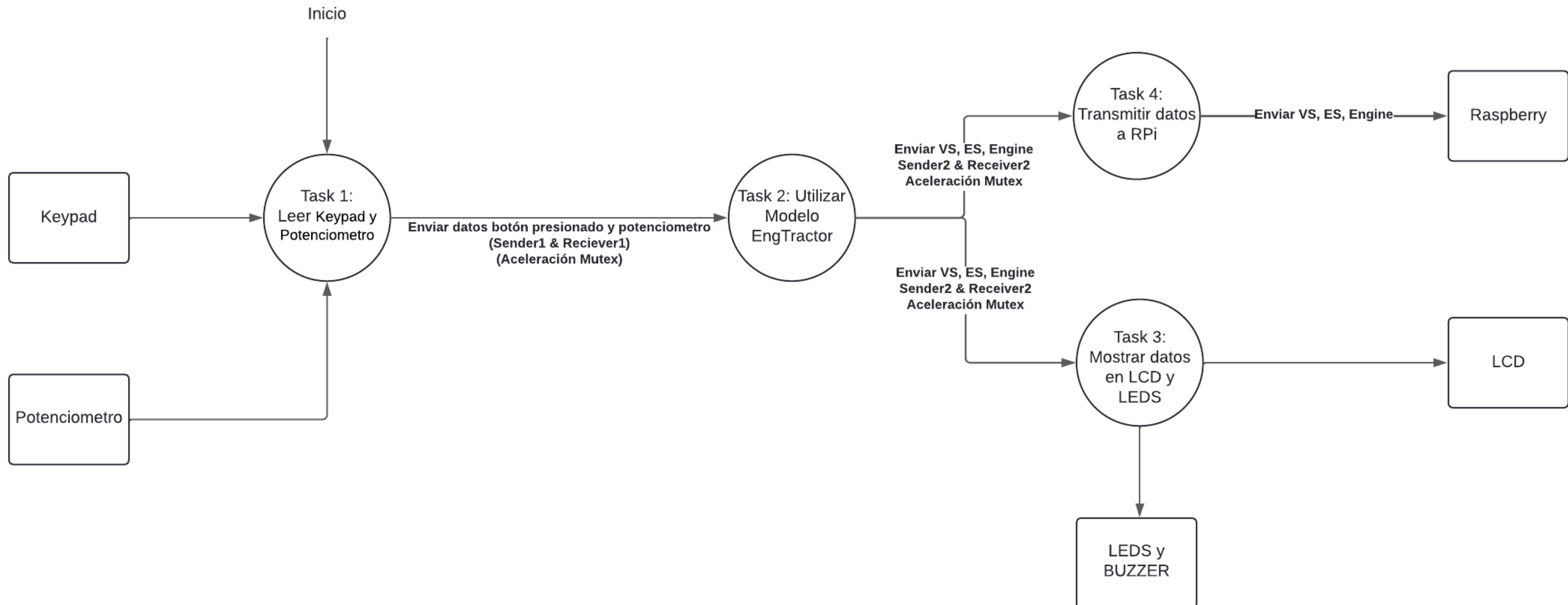


De flujo



Diagramas

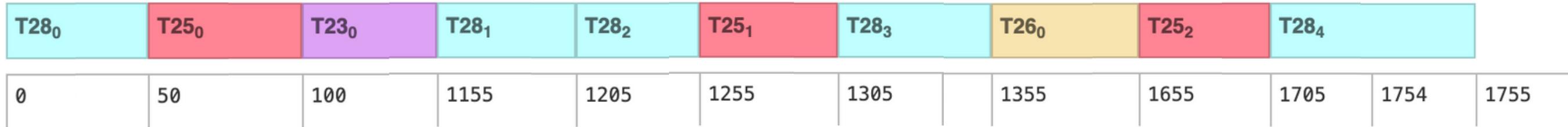
Diagrama de interacción de las Tareas (RTOS)



Scheduling de Tareas (RTOS)

TIMELINE EDF

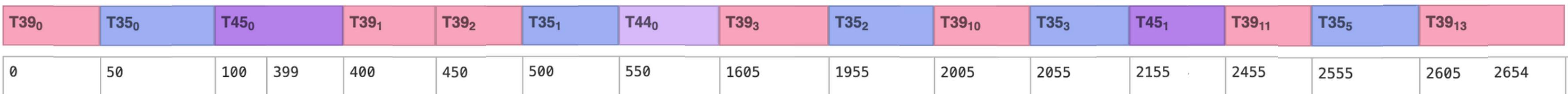
H = 1800



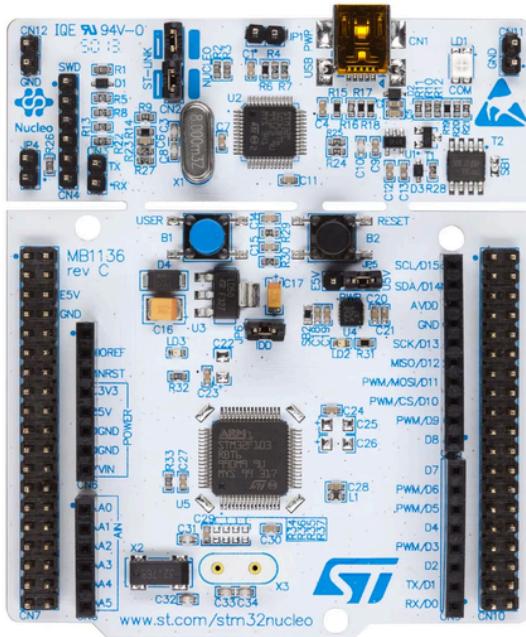
Task	Period (ms)	Execution time (ms)	Priority
T1	360	50	1
T2	1800	300	3
T3	1800	1055	4
T4	600	50	2

TIMELINE RMS

H = 6000



Hardware utilizado



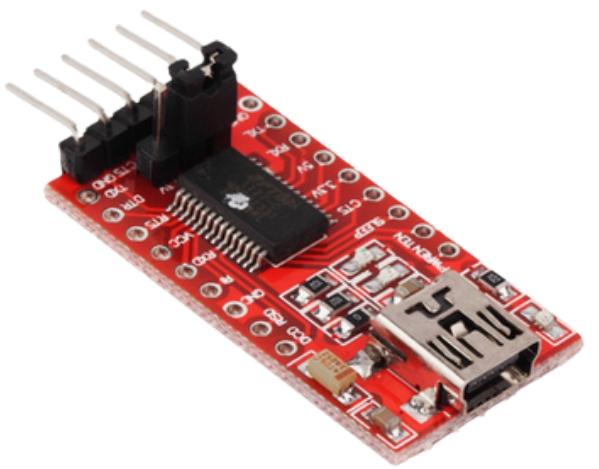
STM32 F103 RB



Raspberry PI 4



Potenciómetro



FTDI



Leds



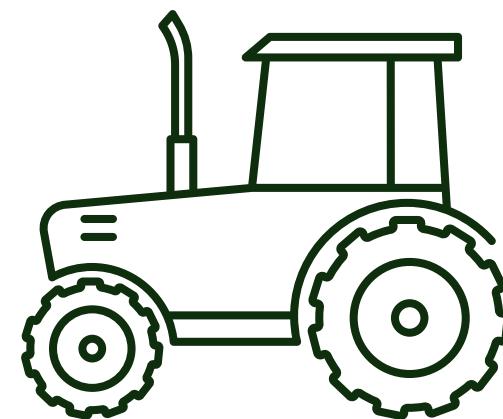
Buzzer



LCD



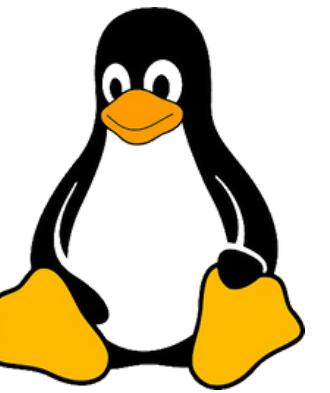
Keypad



Software utilizzato



STM32 CubeIDE



Sistema operativo
linux



Thonny

ACCIONES DE MEJORA RPI

- Modificamos latencia de la FTDI
- Se utilizó otra libreria para la interfaz gráfica, antes utilizábamos matplotlib pyqtgraph.
- Disminuimos lo más que pudimos los delay
- Se agregaron hilos para los botones y poder transmitir de la RPI.





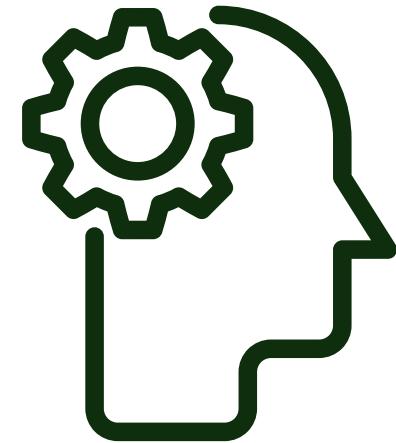
Trabajo a futuro

Se pueden realizar más mejoras, ya hemos mencionado algunas, sin embargo, la parte mecánica que fue algo adicional a lo que se nos pidió también podría mejorar, el diseño 3D y la organización de los circuitos, se puede implementar algún otro sensor para visualizar ciertos datos como la velocidad. En cuanto a la graficación de datos se pueden explorar más librerías de Python como Cython.



- Cuidar lo que pueda ciclar nuestro proyecto (Deadlocks, Priority Inversion)
- En python poder eficientizar mucho más los datos, que nos dé estadísticas para determinar que está bien y que mal.
- Mejorar en cuanto a los periodos de las tareas.
- La implementación de alguna tecnología inteligente, por ejemplo, la idea de redes neuronales como parte de la inteligencia artificial para lograr que nuestro modelo sea entrenado y los datos nos sean de mayor valor.

Conclusiones y reflexiones



XIMENA

BRENDA

VALERIA

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. Todo esto llevó a un mejor entendimiento de los temas, así como a nuevos conocimientos adquiridos.

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 muy complicado, también averiguar como mandar datos de la raspberry a la STM fue complicado, sin duda son conocimientos de valor.

Durante este reto aprendimos sobre la relevancia que tienen los tiempos y periodos 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.

MUCHAS GRACIAS