**LABORATORIO DE SISTEMAS DE CONTROL EMBEBIDOS**

**PRÁCTICA # 3.0**

**IDENTIFICACIÓN DE LA FUNCIÓN DE TRANSFERENCIA**

1. **Objetivo General**

* Desarrollar habilidades en los estudiantes para adquirir, analizar y modelar el comportamiento de una planta de temperatura utilizando una ESP32.

1. **Objetivos Específicos**

* Recopilar datos en tiempo real de la planta de temperatura. Esto incluirá la configuración y calibración de sensores de temperatura y la adquisición de muestras de datos de la planta.
* Modelar el comportamiento de una planta de temperatura con Funciones de Transferencia, utilizando Matlab para realizar un análisis de los datos y encontrar una función de transferencia que describa el comportamiento térmico de la planta. Esto implicará identificar polos y ceros, así como ajustar el modelo según sea necesario.

1. **Materiales**

* ESP32
* 1 DS18B20
* 2 Leds
* 1 Transistor 2N2222
* 1 Resistencia 470

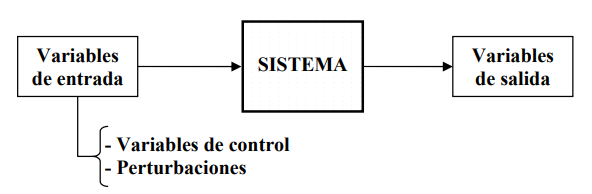
1. **Marco Teórico**

En la presente práctica se revisarán los conceptos básicos ligados a los Sistemas de Control, para facilitar la comprensión de la actividad de laboratorio que está por llevarse a cabo.

* **Concepto**

Un sistema dinámico puede entenderse como una entidad que experimenta influencias externas o variables de entrada, y cuya reacción a estas influencias externas se manifiesta a través de las variables de salida correspondientes.

Es así que, estas influencias externas que afectan al sistema se categorizan en dos conjuntos: las variables de control, que pueden ser ajustadas deliberadamente, y las perturbaciones, sobre las cuales no es factible ejercer ningún control. La Figura 1. ofrece una representación visual de cómo opera el sistema en términos conceptuales.



*Figura 1. Esquema visual de un Sistema de Control.*

En todo caso, el propósito de un sistema de control es alcanzar, a través del ajuste de las variables de control, un control efectivo sobre las variables de salida, asegurando que estas alcancen valores previamente establecidos, conocidos como consignas.

En este sentido, un sistema de control ideal debe satisfacer los siguientes requisitos para alcanzar sus objetivos:

* Asegurar la estabilidad y, especialmente, exhibir robustez ante perturbaciones y posibles errores en los modelos.
* Operar de manera eficiente de acuerdo a un criterio predefinido. Normalmente, este criterio implica que la acción de control en las variables de entrada sea viable, evitando respuestas abruptas e irrealistas.
* Ser fácil de implementar y sencillo de operar en tiempo real con la asistencia de un ordenador.

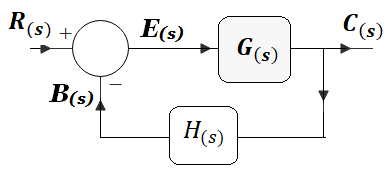
Los componentes fundamentales que integran un sistema de control y posibilitan su ajuste son los siguientes:

* **Sensores.** Permiten conocer los valores de las variables medidas del sistema.
* **Controlador.** Utilizando los valores determinados por los sensores y la consigna impuesta, calcula la acción que debe aplicarse para modificar las
* variables de control en base a cierta estrategia.
* **Actuador.** Es el mecanismo que ejecuta la acción calculada por el controlador y que modifica las variables de control.
* **Función de Transferencia**

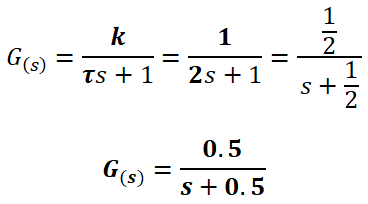
Una función de transferencia es una herramienta matemática utilizada para describir el comportamiento dinámico de un sistema lineal invariante en el tiempo. Representa la relación entre la transformada de Laplace de la salida y la transformada de Laplace de la entrada en un sistema.

* **Representación de la Función de Transferencia**

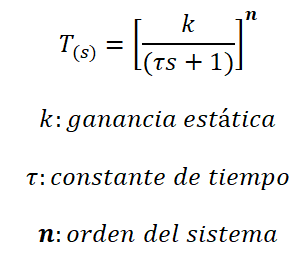
La función de transferencia se representa típicamente como G(s), donde "s" es una variable compleja (frecuencia compleja) y "G" es una función que relaciona la salida con la entrada en el dominio de Laplace.



*Figura 2. Diagrama esquemático de un Sistema de Control.*

**

*Figura 3. Ejemplo de una Función de Transferencia.*

**

*Figura 4. Ecuación para aproximar una Función de Transferencia.*

* **Diseño y Control de Sistemas usando Funciones de Transferencia**

Las funciones de transferencia se utilizan para diseñar controladores y analizar sistemas de control. Esto incluye la selección de polos y ceros para lograr el rendimiento deseado y la estabilidad.

**Tipo de controladores:**

* **Control de Control Proporcional:** Incrementa la velocidad de respuesta del proceso controlado y genera un desplazamiento u offset (excepto en el caso de los integradores puros)
* **Control Integral:** Suprime todo offset, eleva las desviaciones máximas. Además, produce respuestas arrastradas y largas oscilaciones. Por otra parte, el aumento de Kc acelera la respuesta, pero produce más oscilaciones y puede llegar a desestabilizar el sistema.
* **Control Derivativo:** Anticipa el error y actúa en función del error que iría a ocurrir, finalmente estabiliza la respuesta del bucle cerrado.

Si desean profundizar más al respecto, en el siguiente enlace podrán encontrar un documento pdf traducido del noveno capítulo del Libro de “Dinámica y Controles de Proceso” de Peter Woolf et al. De la Universidad de Michigan.

<https://github.com/SistemasEmbebidos2020/PracticaSemana2MACIEmbebidos/blob/3efc3ca4747378dfa7a1f6069f5fdd0b28815836/9.02__Control_de_P%2C_I%2C_D%2C_PI%2C_PD_y_PID.pdf>

1. **Código**

**Explicación General**

Para la presente práctica, se hará uso de Visual Studio Code y de Matlab. En el primero se cargará el código necesario para que nuestra pequeña planta de temperatura caliente el transistor y este pueda ser medido por el sensor de temperatura digital DS18B20. Cabe destacar, que, para una interacción más acorde al curso, se usó un par de Leds que tienen como función consumir corriente eléctrica, de tal manera que hará que a medida que se encienda o apague los Leds el Transistor aumentará o disminuirá su temperatura.

**Código Visual Studio Code**

**Enlace:**<https://github.com/SistemasEmbebidos2020/PracticaSemana2MACIEmbebidos/blob/3efc3ca4747378dfa7a1f6069f5fdd0b28815836/lazoabierto.cpp>

Como la mayoría de los códigos que se han venido trabajando, se empieza con el encabezado (header file) esencial de #include <Arduino.h> para que el código sea compilado y ejecute correctamente. Este archivo contiene definiciones y declaraciones fundamentales para el funcionamiento de los programas escritos para placas Arduino.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ahora bien, en vista de que se está usando un sensor digital, específicamente el DS18B20. Este sensor, además de ofrecer una alta precisión y contar con una variante sumergible, opera mediante un sistema de conexión particular llamado "OneWire". Para lo cual debemos importar su librería, este sistema en sí permite la transmisión y recepción de datos a través de un solo cable, siendo una alternativa valiosa a considerar cuando los recursos de conexión son limitados en un proyecto, ya que nos permite reducir la necesidad de utilizar un cable por cada sensor.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

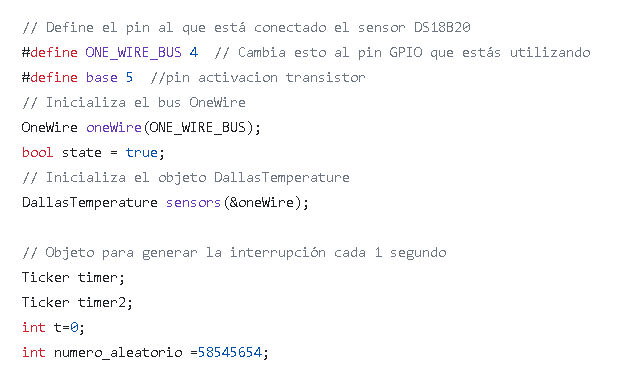
Por otra parte, la librería <DallasTemperature.h> nos permite interpretar los datos enviados y recibidos por nuestro bus. Además, se cuenta con una última librería <Ticker.h> que permite realizar interrupciones de tiempo con la esp32.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aquí se define el pin al que está conectado el sensor DS18B20 y el pin para activar un transistor.

* #define ONE\_WIRE\_BUS 4 🡪Define el pin al que está conectado el sensor DS18B20
* #define base 5 🡪Define el pin de activación del transistor



Se inicializan objetos para trabajar con el bus OneWire y los sensores DS18B20.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Se declara una variable booleana llamada "state" que se usa para controlar el estado del transistor.

Se declaran objetos Ticker para gestionar temporizadores y variables para almacenar valores y números aleatorios.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Continuando con el código, la función readTemperature se ejecuta cuando se genera una interrupción de temporizador cada segundo y realiza lo siguiente:

* Solicita y lee la temperatura del sensor DS18B20.
* Realiza un mapeo del valor de temperatura entre 0 y 50 grados Celsius a un rango de 0 a 1024.
* Imprime la temperatura, el tiempo transcurrido y el estado del transistor en el monitor serial.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Por otra parte, la función chanceout se ejecuta cuando se genera una interrupción de temporizador (timer2) en función del valor de numero\_aleatorio y realiza lo siguiente:

* Cambia el estado de la variable state, que controla el transistor.
* Genera un nuevo número aleatorio entre 1 y 7.
* Programa una nueva interrupción con el nuevo número aleatorio como intervalo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Asimismo, la función setup se ejecuta una vez en el programa y por su parte:

* Configura el pin "base" como una salida y lo inicializa en el estado actual de "state".
* Inicializa la comunicación serial a una velocidad de 115200 baudios.
* Inicializa el sensor DS18B20.
* Genera un número aleatorio y configura el temporizador "timer" para ejecutar la función readTemperature cada 1 segundo. También programa una primera interrupción "timer2" con un intervalo basado en un número aleatorio.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Finalmente, el bucle principal está vacío ya que la lógica principal se ejecuta en las interrupciones configuradas.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En todo caso, este código controla un sensor de temperatura DS18B20, muestra la temperatura en el monitor serial y cambia el estado de un transistor en función de interrupciones de temporizador que también generan números aleatorios.

**Código Matlab**

**Enlace:**<https://github.com/SistemasEmbebidos2020/PracticaSemana2MACIEmbebidos/blob/3efc3ca4747378dfa7a1f6069f5fdd0b28815836/dataadquisition.m>

Como primer paso para configurar la comunicación en serie, se debe borrar todo el texto de la ventana de comandos, además de borrar todas las variables que hubiera en el workspace, para evitar posibles errores. Se lo realiza como buena práctica de programación, a continuación, se detalla lo previamente mencionado:

* close all 🡪 Cierra todas las ventanas de gráficos previamente abiertas en MATLAB.
* clear all 🡪 Borra todas las variables del espacio de trabajo actual.
* delete (instrfind({'Port'},{'COM3'})) 🡪 Intenta cerrar cualquier conexión serie existente en el puerto COM3 antes de abrir una nueva.



De igual manera, con la siguiente línea de código se pretende eliminar (borrar) cualquier objeto de instrumentación (instrument object) que se haya creado previamente y que esté asociado al puerto COM3, para evitar problemas de conexión cuando se trabaje con dispositivos en ese puerto específico.

**\*Importante:** No se olvide de reemplazar el nombre del puerto serie 'COMx' utilizado por su ESP32. Al final de la guía se detalla como puede observar que puerto se encuentra utilizando.

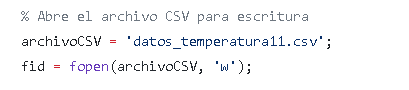


Por otra parte, la siguiente línea de código abre una conexión en serie en el puerto especificado a una velocidad de baudios de 115200. Así que se crea un objeto s que se utiliza para la comunicación.



En las dos siguientes líneas, en cambio es la parte con la que se abrirá el archivo CSV, cabe destacar que no solo se puede abrir en dicho formato, sin embargo para efectos prácticos de este laboratorio se decidió así.

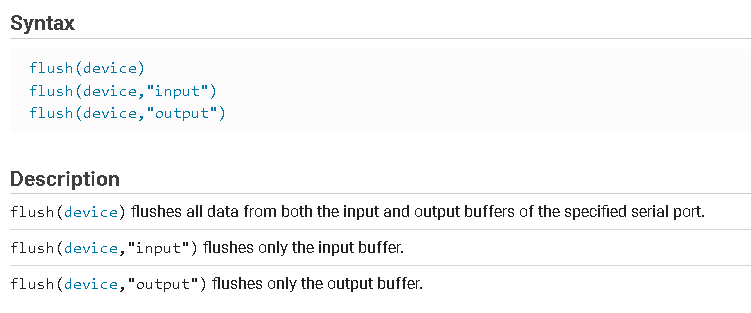
* archivoCSV = 'datos\_temperatura11.csv'; 🡪 Define el nombre del archivo CSV en el que se guardarán los datos.
* fid = fopen(archivoCSV, 'w'); 🡪 Abre el archivo CSV en modo escritura ('w'). Esto crea un identificador de archivo fid para el archivo.



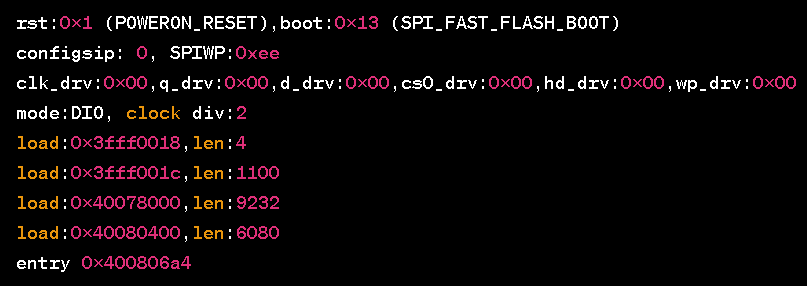
Ahora, se limpia el búfer de salida del puerto serie. Esto se hace para asegurarse de que no haya datos residuales en el puerto antes de comenzar la comunicación.



**\*Importante:** Se muestra la funcionalidad de dicha función en Matlab.



Asimismo, se utiliza un bucle for para leer 9 líneas de datos desde el ESP32, pero no se almacenan ni procesan estos datos. Esto se hace con la finalidad de cuando se tome los datos de la ESP32, no se tome las líneas que salen en el cuándo se cargue un código en la placa. Usualmente las primeras líneas proporcionan información sobre la inicialización del sistema y pueden incluir detalles sobre la configuración del hardware, la versión del firmware, la configuración de pines y otros datos relevantes.



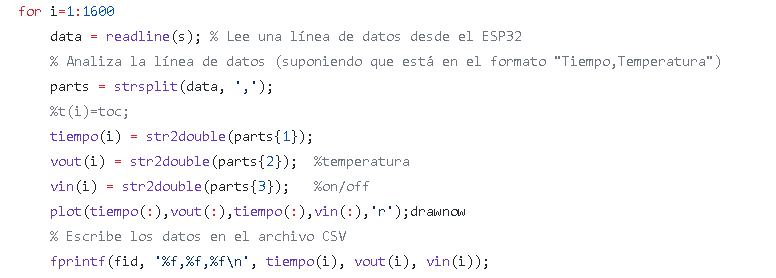
**\*Importante:** Se muestra un ejemplo típico de las primeras líneas que verías al cargar un programa en una ESP32

Siguiendo con el código, se comienza un temporizador para medir el tiempo de ejecución.



Finalmente, se llegó a la parte más importante del código en donde se realizará la lectura, procesamiento y escritura de datos, que se explica seguidamente:

* Se utiliza un bucle for para leer 1600 líneas de datos desde el ESP32.
* readline(s): Lee una línea de datos desde el ESP32.
* Los datos leídos se dividen en partes utilizando strsplit asumiendo que los datos están en el formato "Tiempo,Temperatura".
* Los valores de tiempo, temperatura (vout) y otro valor (vin) se almacenan en las matrices correspondientes.
* Se utiliza plot para trazar los valores de tiempo, temperatura (en azul) y el otro valor (en rojo) a medida que se leen los datos.
* Los datos se escriben en el archivo CSV utilizando fprintf.



El código termina con un mensaje en la ventana de comandos para indicar que se han leído y guardado todos los datos.

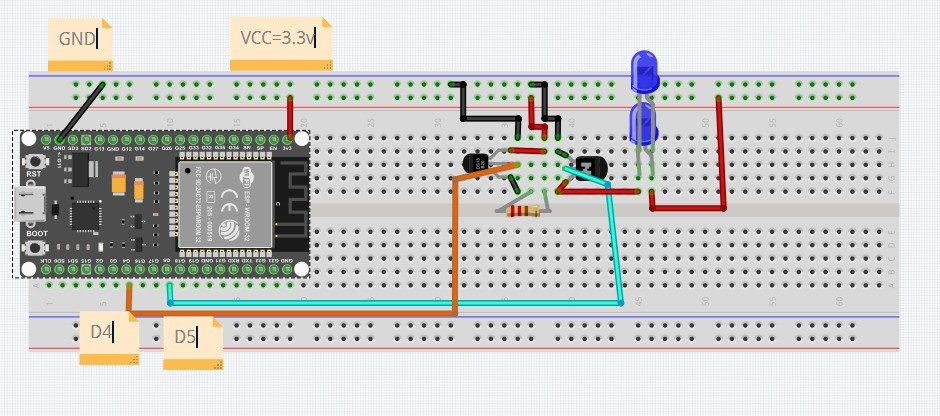


Además de cerrar la comunicación en serie y el archivo CSV.



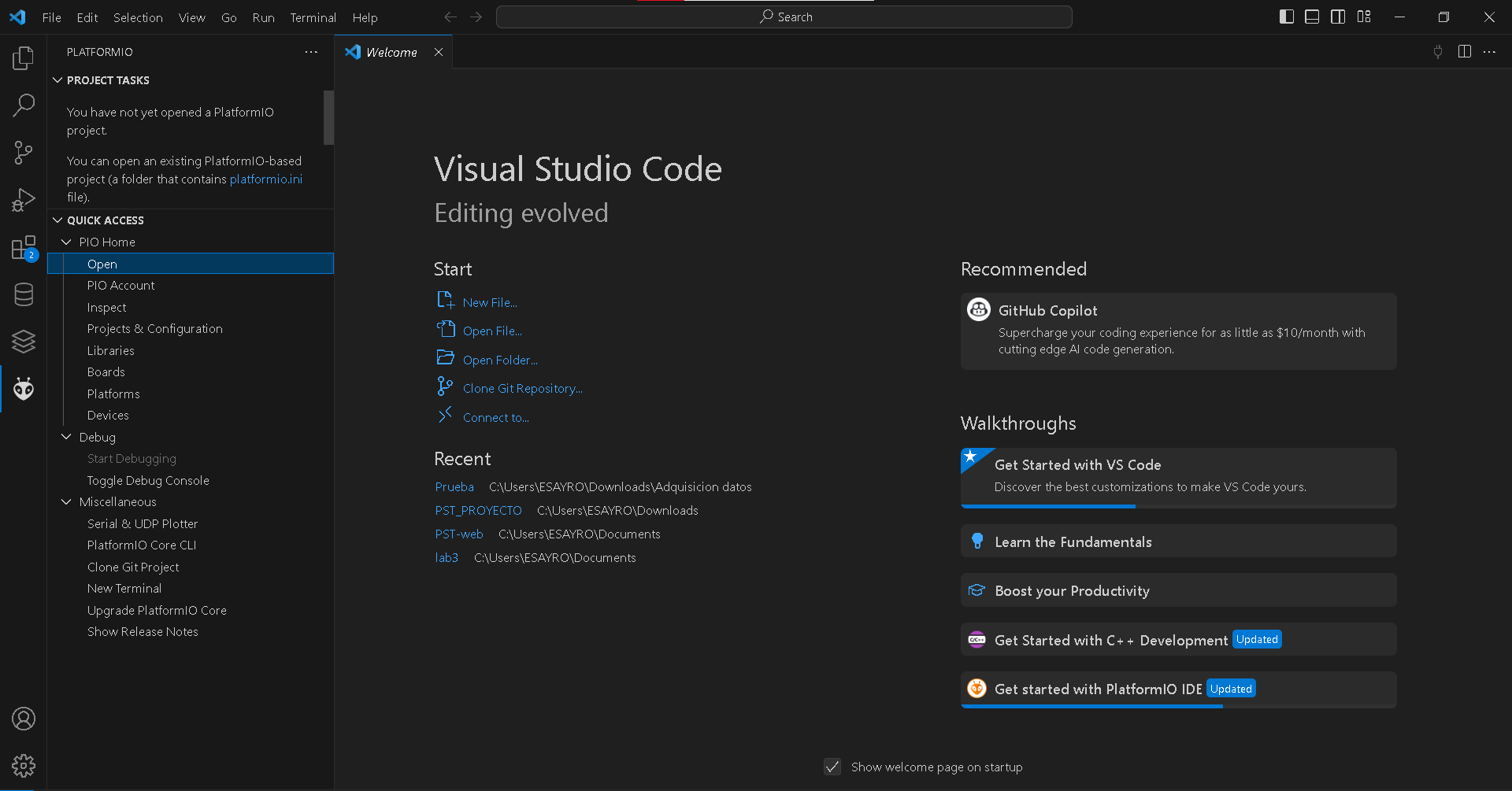
1. **Procedimiento**

En base en el siguiente diagrama esquemático, arme su planta de temperatura.



Ahora en base al Tutorial Previo que tuvo en VS Code, cargue su código en la ESP32 y proceda a cargar del código de Matlab en su respectiva aplicación, para ejecutarlo.

De todas maneras, se les hará una pequeña guía de cada uno de los programas. Primero empezamos con el VS Code, procedemos a abrirlo y nos saldrá la pantalla de inicio, tal como se muestra en la siguiente imagen:



Así que le damos clic al botón de PlatformIO, que tiene forma de Alien y le damos Clic en “Open” de ser necesario, en todo caso tiene que salirle la siguiente ventana.

Captura de pantalla de un celular

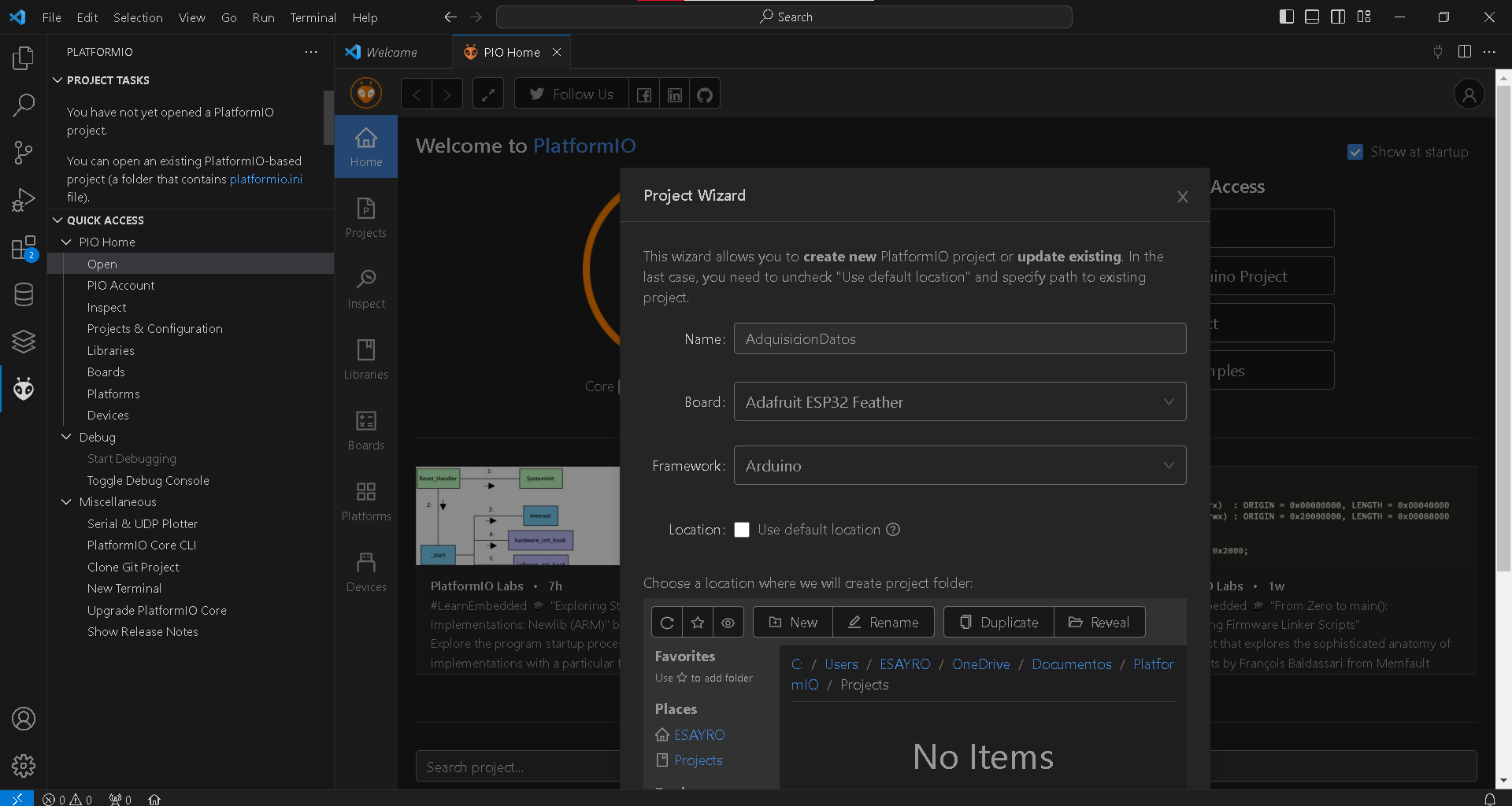
Descripción generada automáticamente

Así que le damos clic a “New Project” y procedemos a llenar la información, nombre del proyecto en Name, la placa de desarrollo en Board (se busca la ESP32 Feather), se mantiene la opción por defecto de Framework (Arduino).

Captura de pantalla de computadora

Descripción generada automáticamente

Ahora bien, dependiendo de su elección puede elegir que se guarde en la ubicación por defecto o en algún directorio de su preferencia. De ser el segundo caso, deselecciona el check de la casilla y elige la carpeta donde desea guardar.



Le damos “Finish” y esperamos que se cree el Proyecto.

Captura de pantalla de computadora

Descripción generada automáticamente

Nos dirigimos a los archivos del proyecto, específicamente a la carpeta src, para abrir el main.cpp en donde borraremos el código que se muestra y lo reemplazamos con el código proporcionado en Github.

Texto

Descripción generada automáticamente

Ahora bien, como pueden observar tienen un delineado de color rojo debajo de la librería OneWire y DallasTemperature, así que debemos anexarlas a nuestro proyecto, eso se realiza de la siguiente manera:

1. Le damos clic al ícono de PlatformIO del costado izquierdo en la parte inferior.
2. Le damos clic en “Open”, y nos centramos en la siguiente ventana.

Captura de pantalla de computadora

Descripción generada automáticamente

1. En dicha ventana, nos dirigimos a Libraries (Librerias) y le damos clic.
2. En el buscador vamos a ingresar el nombre de la librería que deseamos adjuntar al proyecto, en este caso primero vamos a agregar la librería OneWire así que lo escribimos. Nos arrojará toda una lista de resultados, pero nuestra librería tendría que salir como primera así que le damos clic.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. Eso no abrirá toda una descripción de la librería, así que solo procedemos a darle clic a la opción “Add to Project”.

Captura de pantalla de computadora

Descripción generada automáticamente

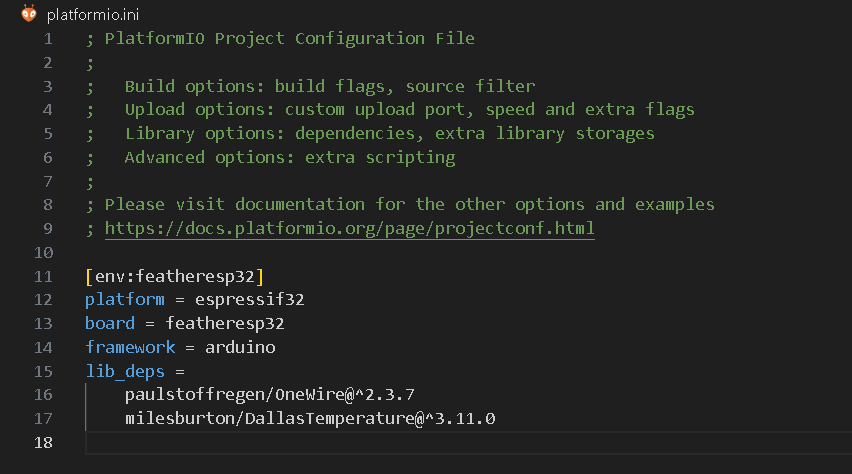
1. Para lo cual nos abrirá otra ventana emergente, en donde lo importante es que seleccionemos el proyecto en el que estamos trabajando, para que en ese se cargue las librerías, por último le damos en “Add”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Este mismo procedimiento lo repetimos, pero buscamos la librería “DallasTemperature”.

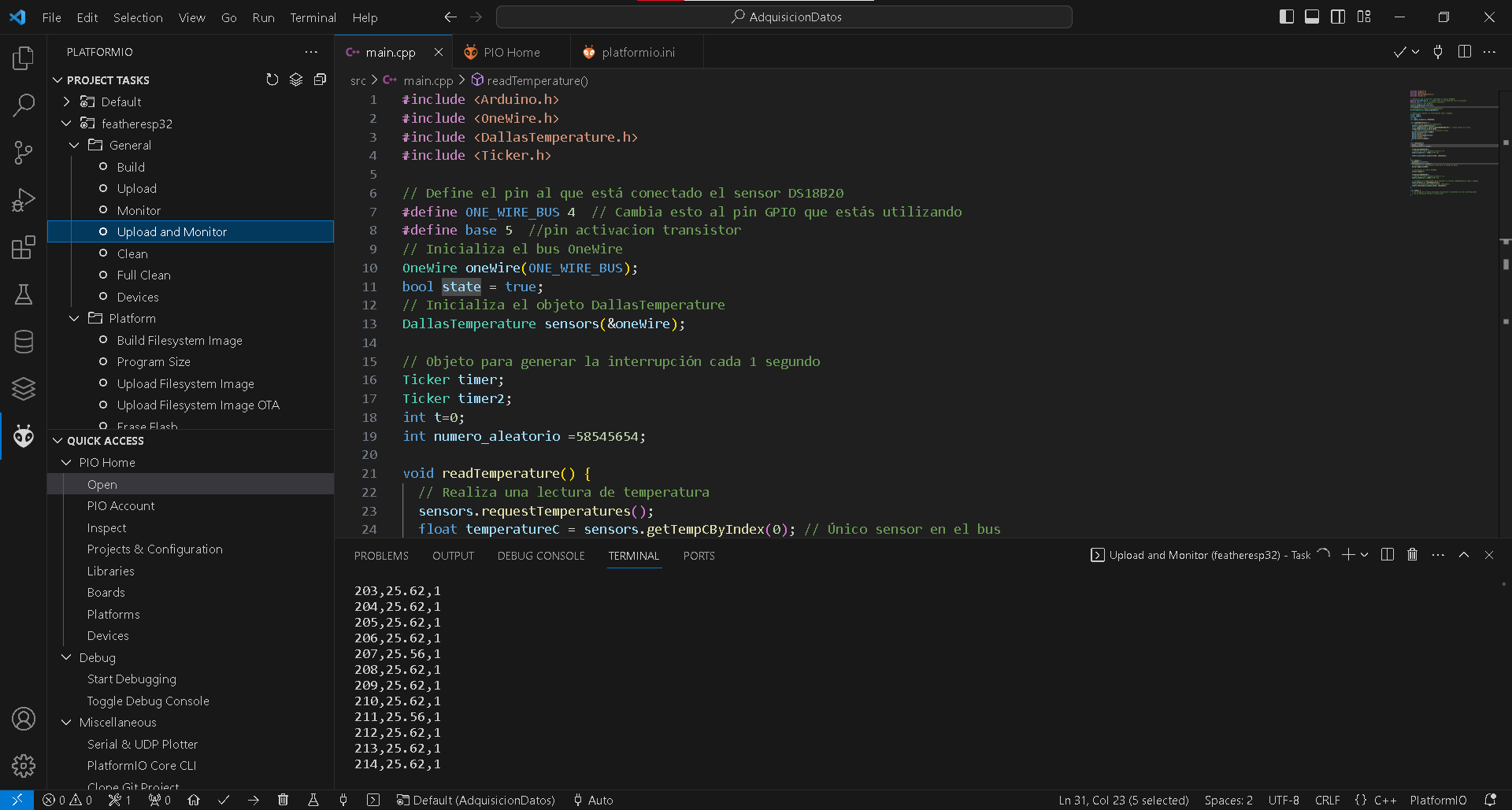
Cuando ya hayamos agregado las dos librerías, verificamos en el platform.ini que se encuentren las librerías que se supone adjuntamos. En este caso vemos que en el parámetro lib\_deps si se encuentran ambas librerías.



Aprovechando que estamos en esa ventana, agregamos un parámetro adicional, el monitor\_speed==115200, que es para una correcta lectura del sensor por lo que se ajusta a los baudios acorde a lo escrito en el serial.begin() del código.

Texto

Descripción generada automáticamente



Finalmente le damos clic en “Upload and Monitor”, esperamos que cargue el código en la ESP32 e inmediatamente verán los datos del sensor.

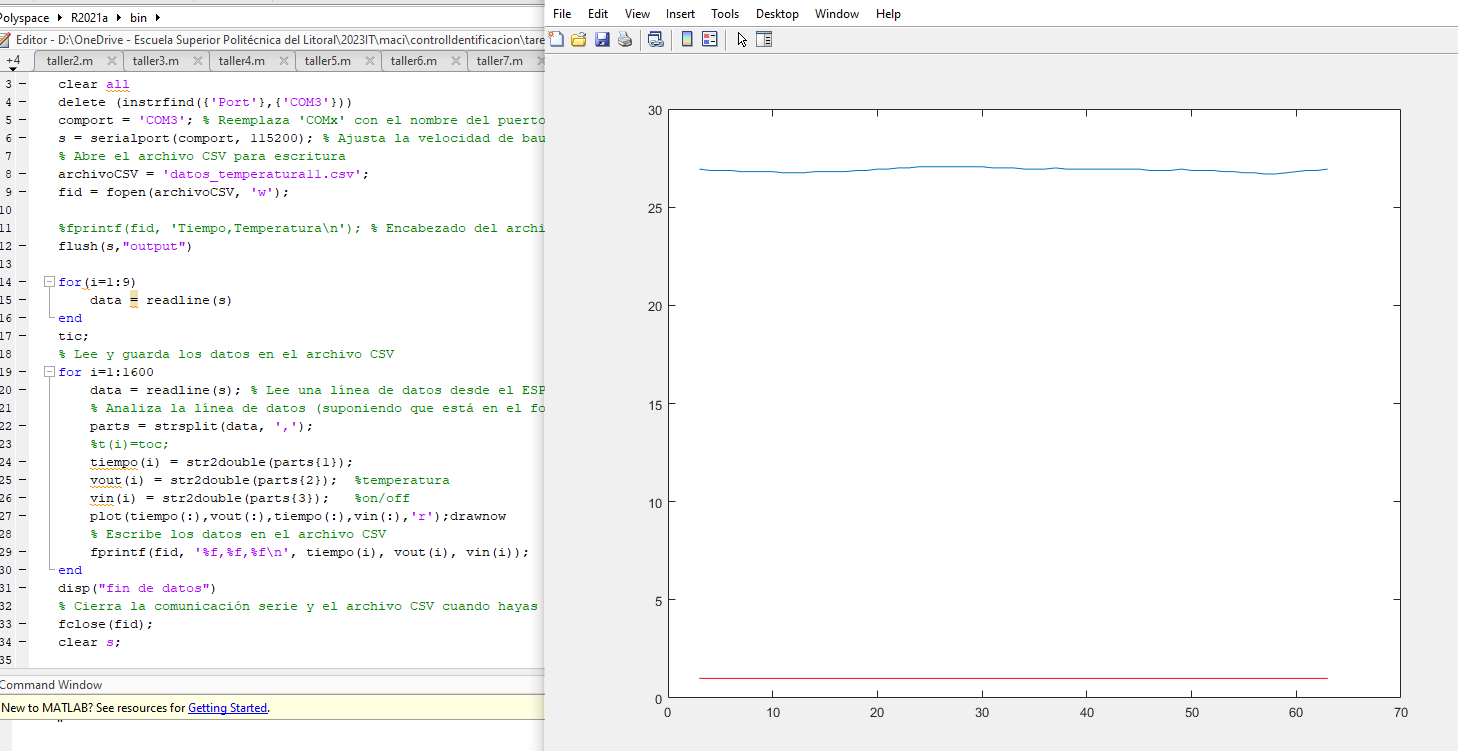
Ahora en cuanto **Matlab**, ingresamos al aplicativo y en su barra de herramientas seleccionamos “New Script”.



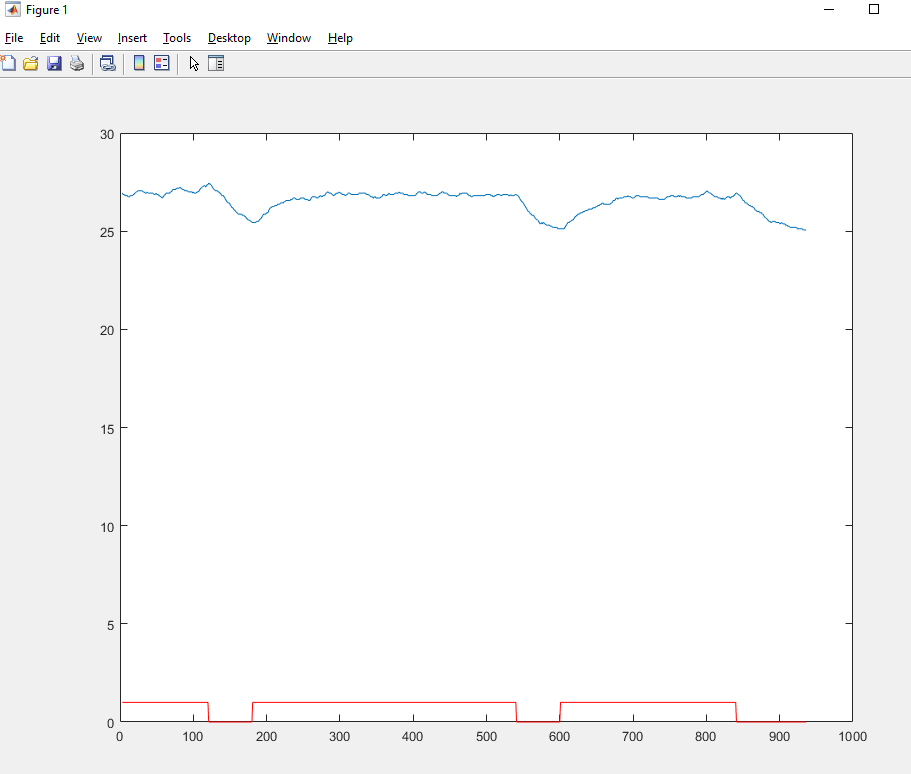
Diagrama

Descripción generada automáticamente con confianza media

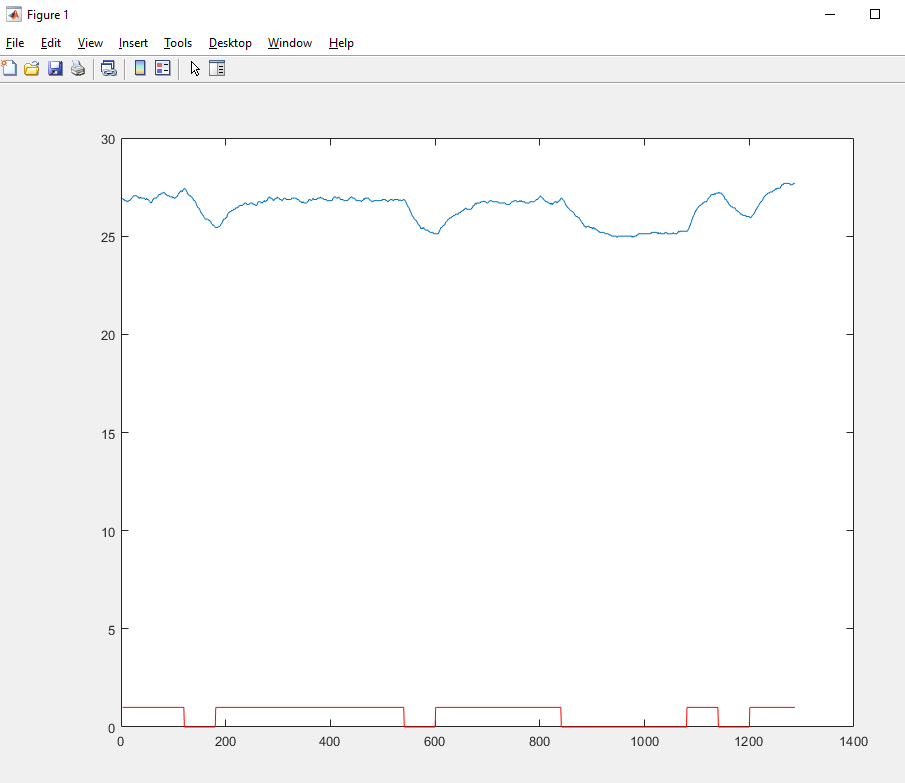
Cargamos el respectivo código que se encuentra en Github y corremos con “Run”. Por lo pronto verán que saldrá una gráfica que se irá actualizando con los datos recopilados del sensor de temperatura digital con la ESP32.

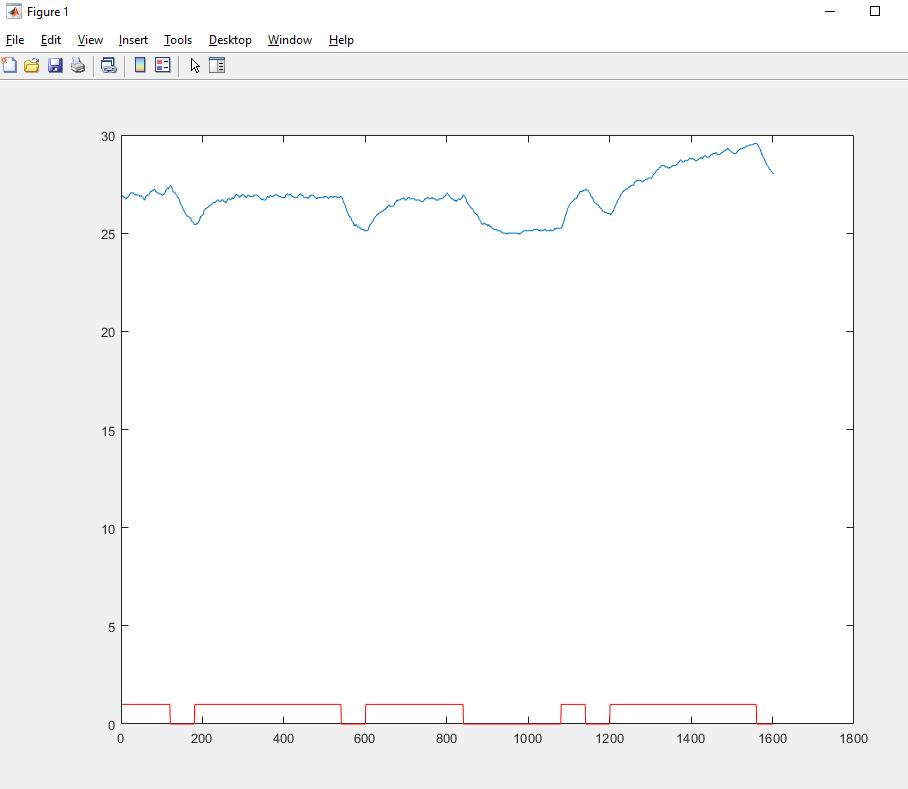


En la que la línea roja representa los pulsos 1 o 0, mientras que la azul la variación de temperatura.

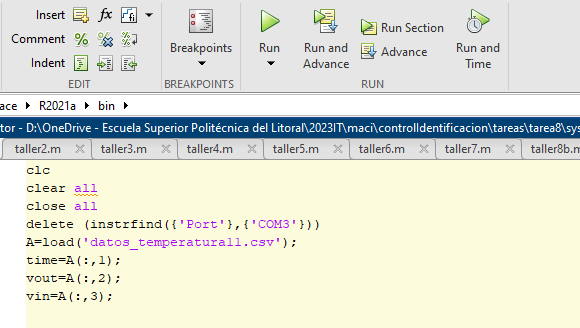


Verán como la curva irá cambiando acorde a los pulsos.



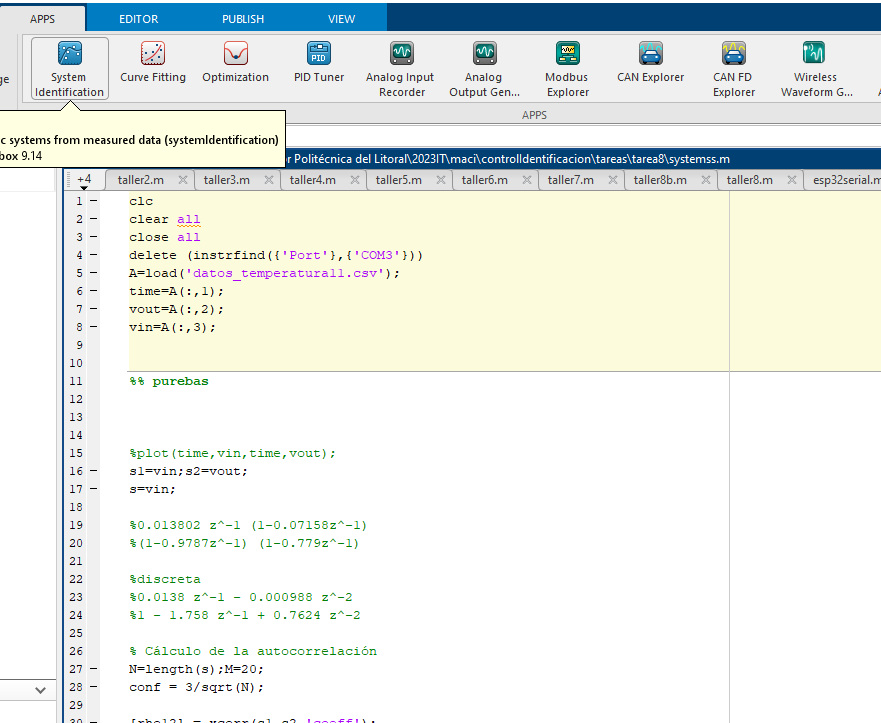


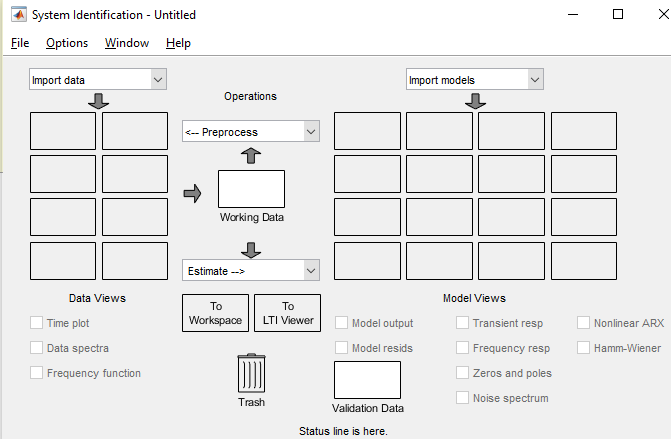
Ahora, se reescribe en el script lo siguiente y se le da clic en “Run section”. Cabe destacar que se debe seleccionar dicha porción mostrada a continuación.



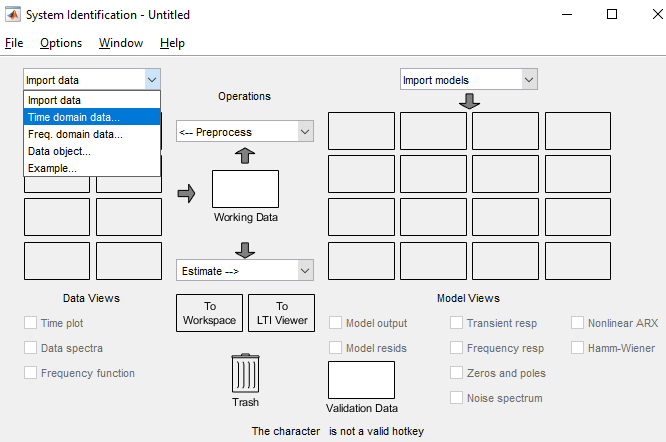
Todo esto con la finalidad de guardar los datos en las variables que sean de nuestro interés.

A continuación, abrimos system identification, que se encuentra dentro de una opción de la parte de Apps (la hallan en la parte superior).

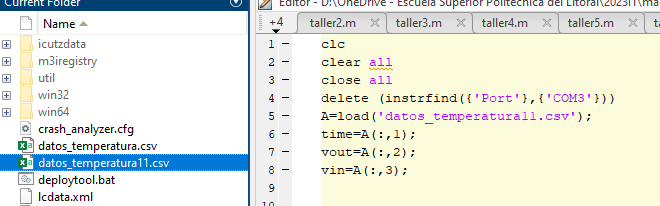


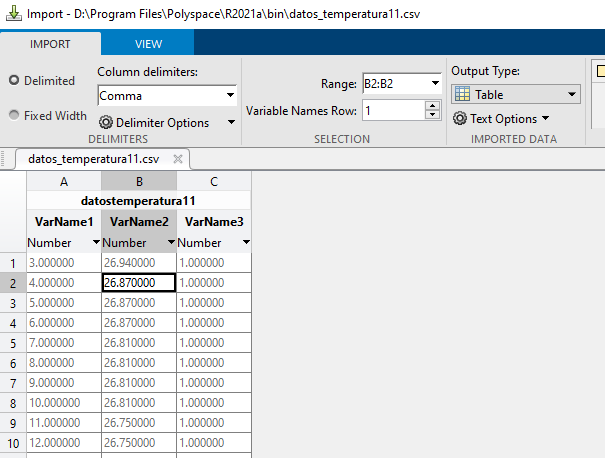


Seleccionar importa data, time domain data



Abrir el csv creado y verificar desde que tiempo se tomaron los datos



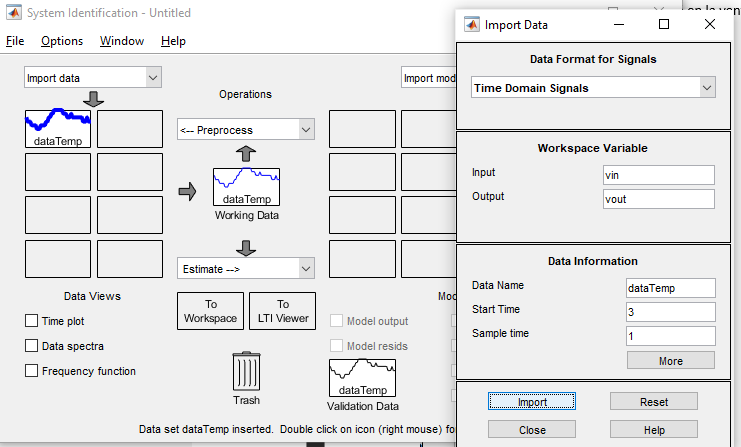


En mi caso desde time 3 y es lo que colocaré en la ventana de import data time domain, y debe quedar de la siguiente manera y clic en import

Interfaz de usuario gráfica

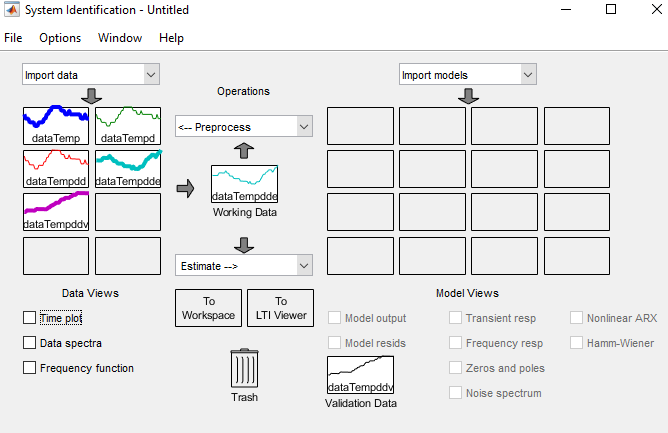
Descripción generada automáticamente

Se visualizará en la ventana principal que los datos están importados

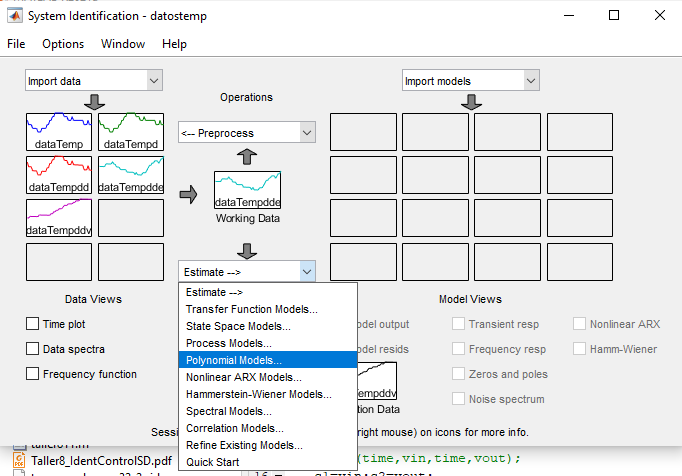


A continuación realizaremos preprocesamiento de datos, quitando las media y quitando tendencias, además se hará la selección de rango del 25% de datos para la validación y 75% para la estimación.

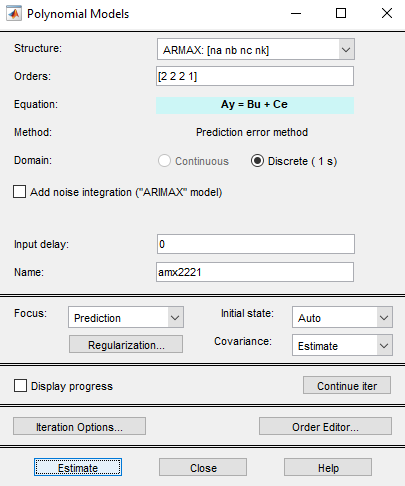
Quedando de la siguiente manera:



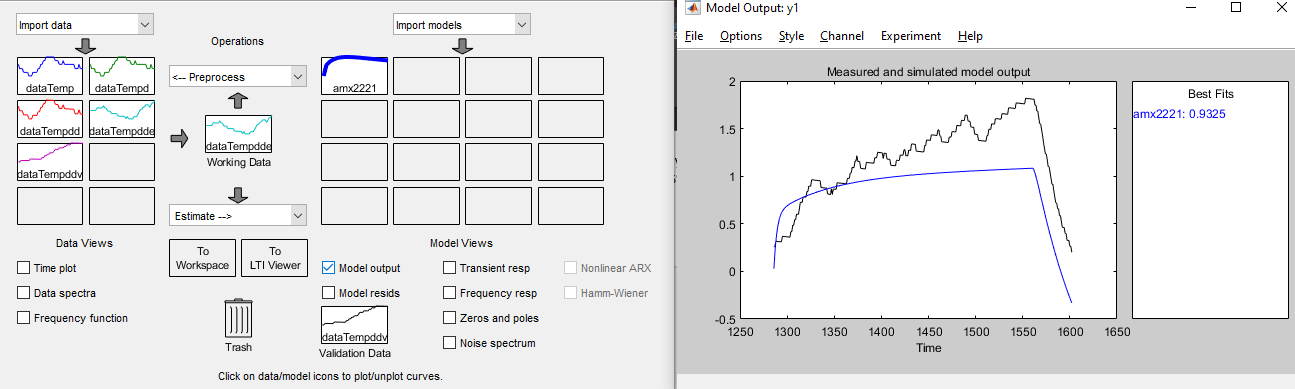
Ahora en estimate seleccionar polynomial model



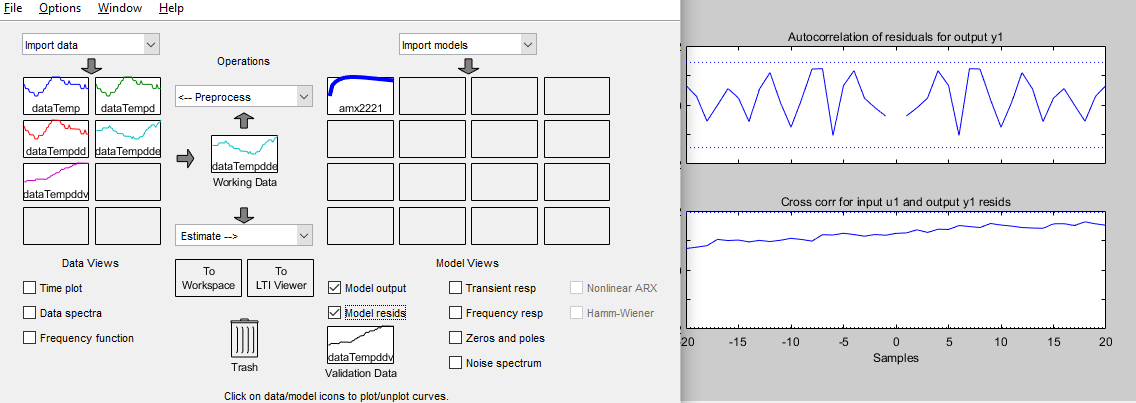
Seleccionar armax, o cualquier otro modelo que usted considere apropiado para el sistema.



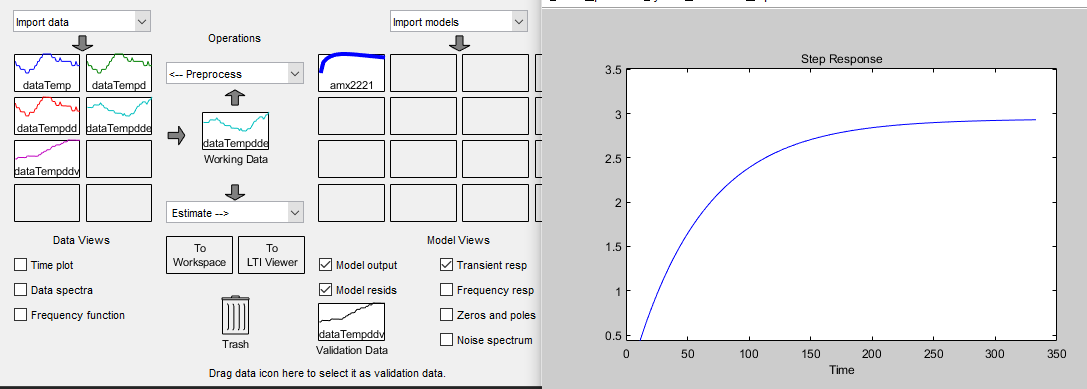
Y clic en estimate y se mostrará en el sistema lo siguiente:



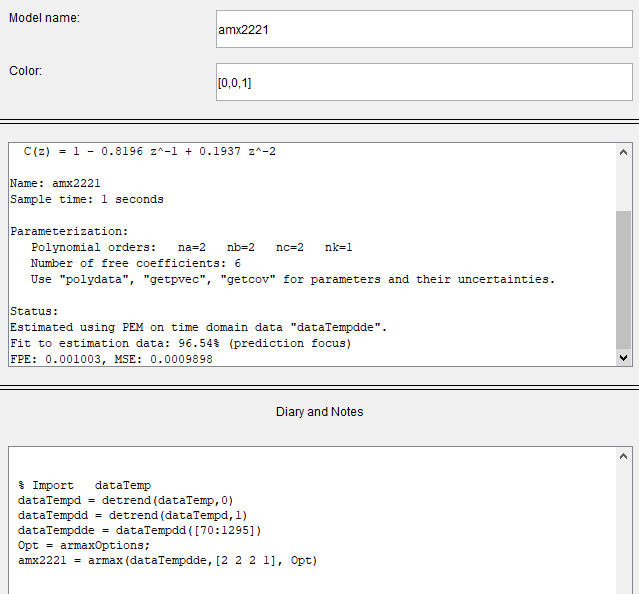
Se verifica los valores residuales:



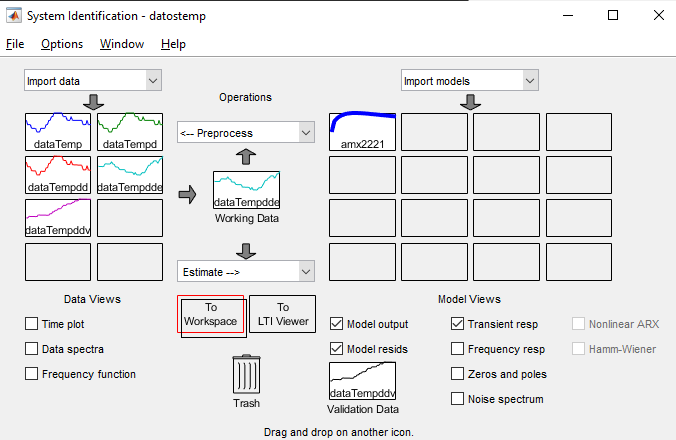
Y por último la respuesta transiente:



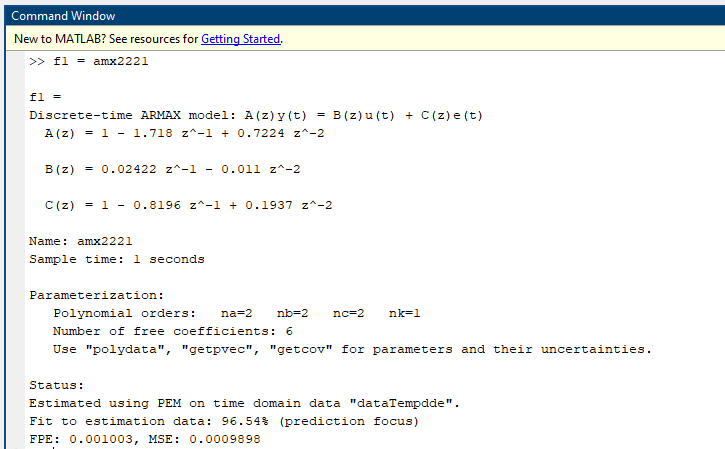
Al hacer doble clic en amx2221 podremos verificar el % de fit:



Con esto procederemos a exportar el modelo, para ello arrastramos el armx2221 al recuadro de To Workspace.



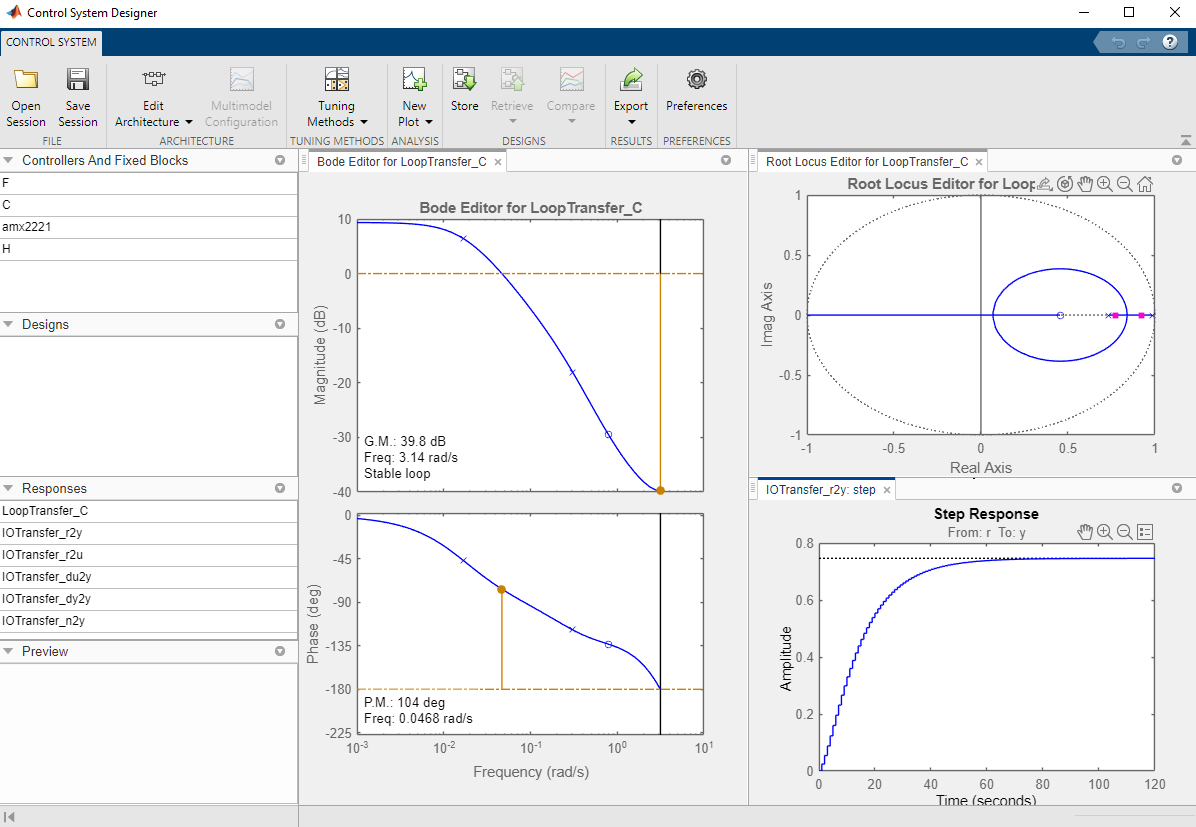
Y en el command window de Matlab, procedemos a escribir amx2221, y guardarlo en una variable por ejemplo f1 = amx2221

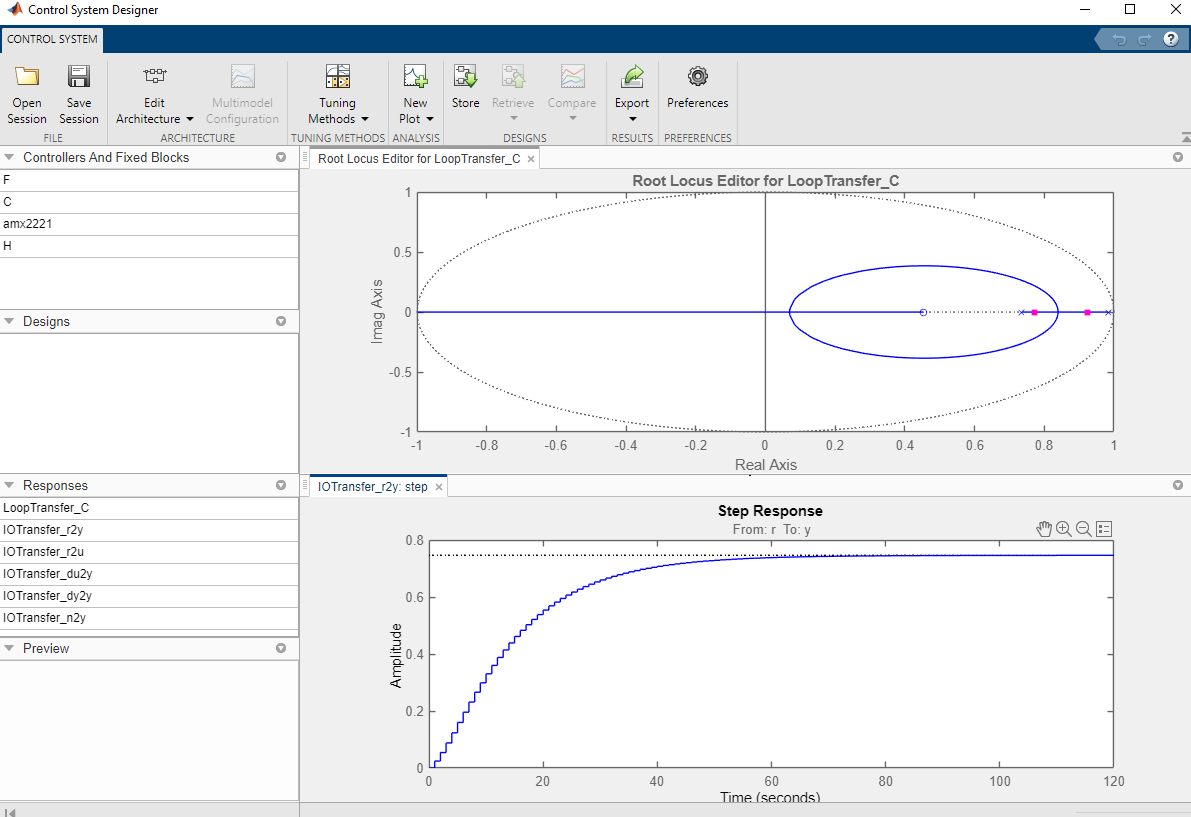


Y acontinuación ejecutamos sisotool para encontrar los valores pid



Se abrirá la siguiente ventana donde cerraremos la de la izquierda y solo nos quedaremos con la de la derecha.





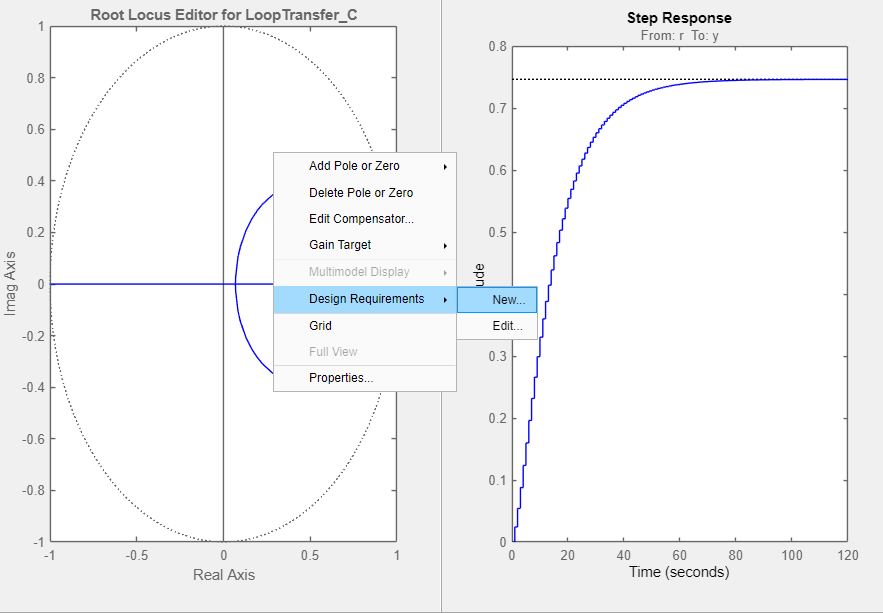
Y moveremos acorde nos sea mejor la visualización

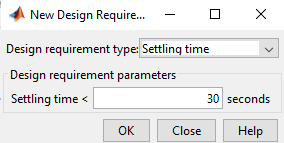
Gráfico, Diagrama

Descripción generada automáticamente

Aquí podremos poner las condiciones que el profesor indique por ejemplo, máximo 5% de overshoot, y un tiempo de respuesta máximo de 30s

Haciendo clic derecho en la gráfica de la izquierda

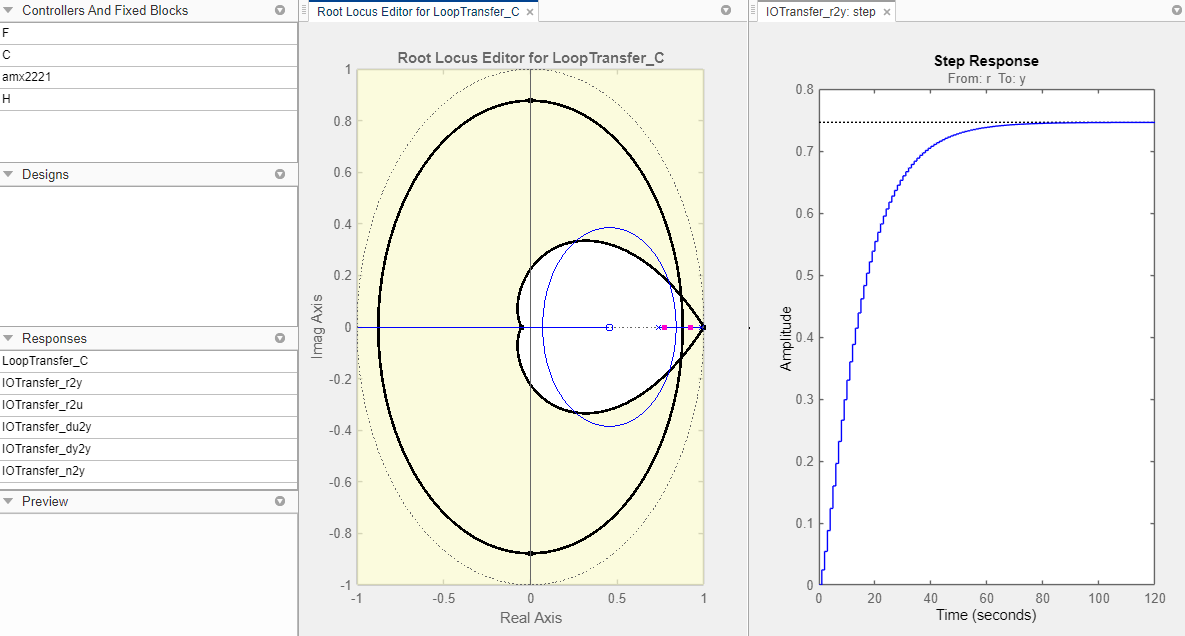




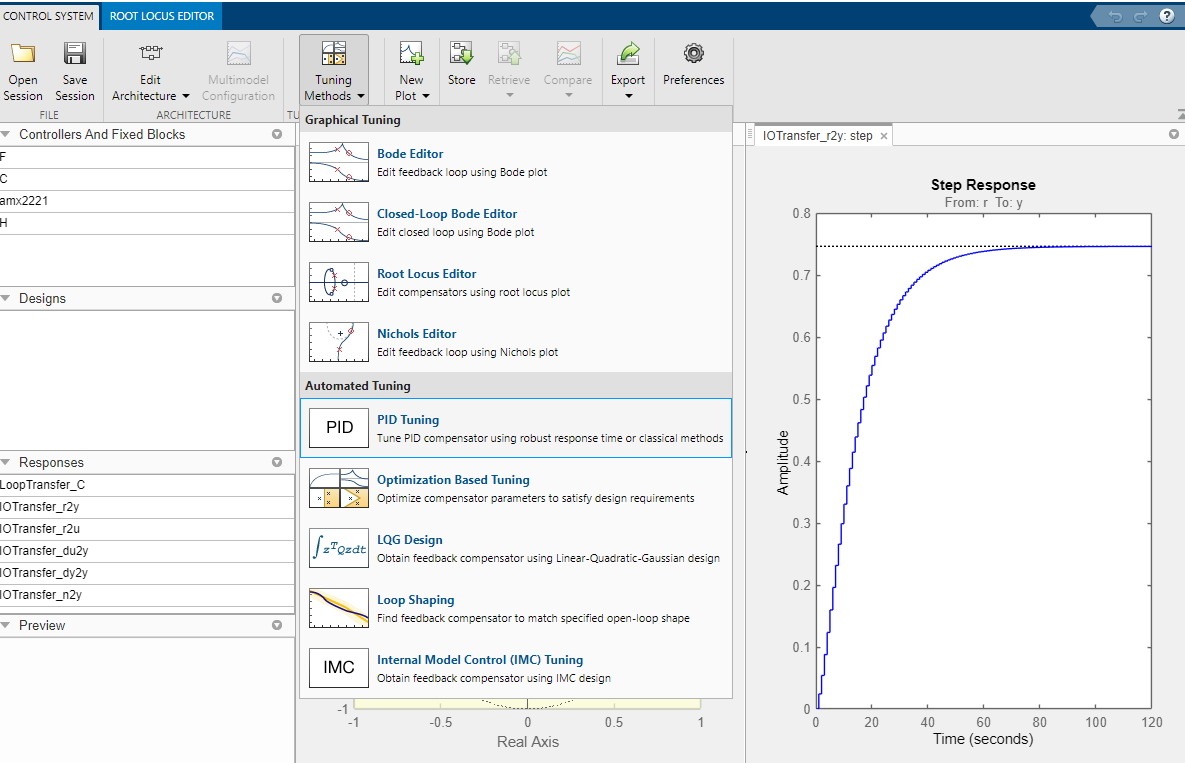
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

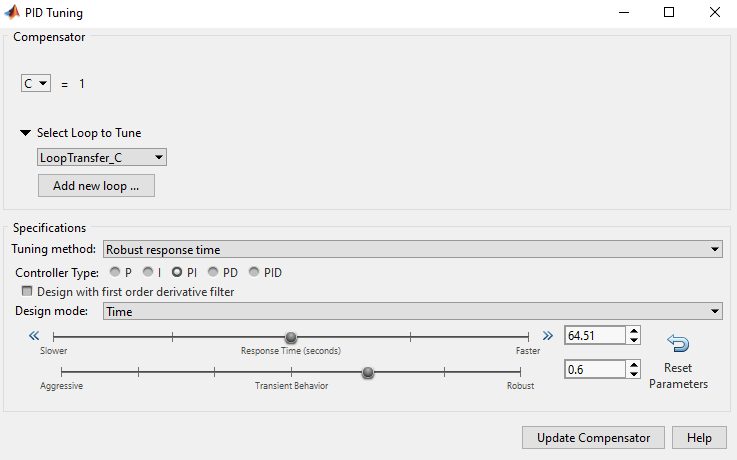
Quedando de la siguiente forma:



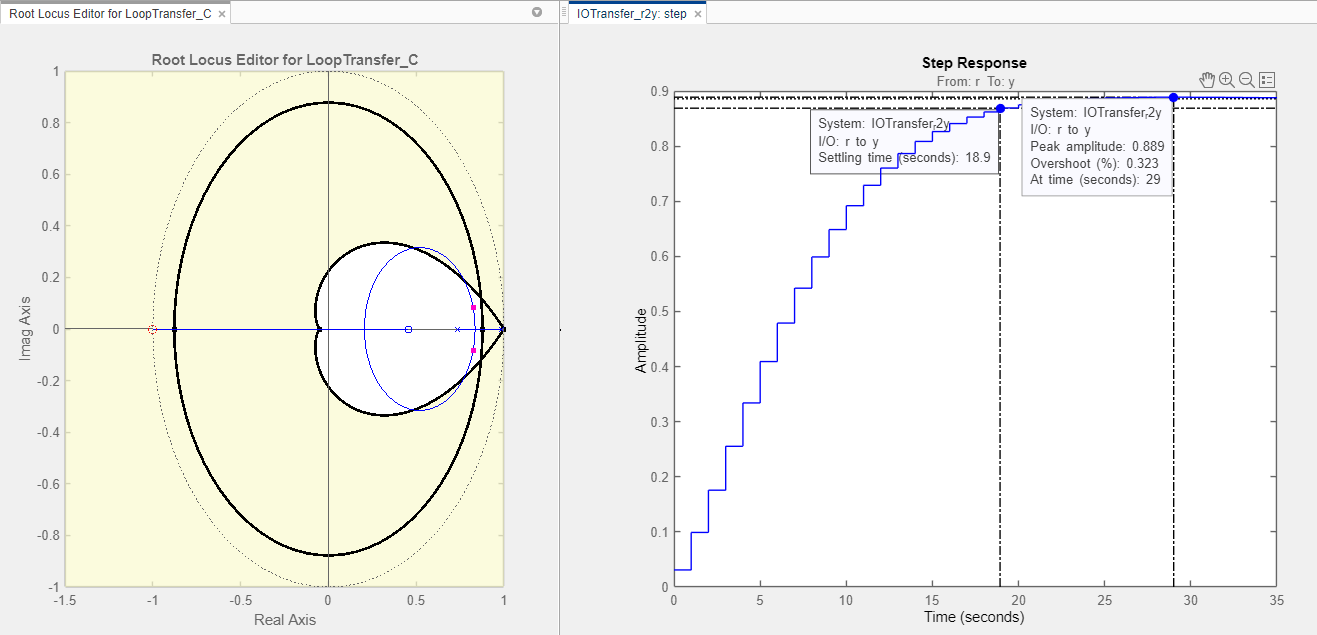
A continuación en control system seleccionar tunning methods y pid tunning



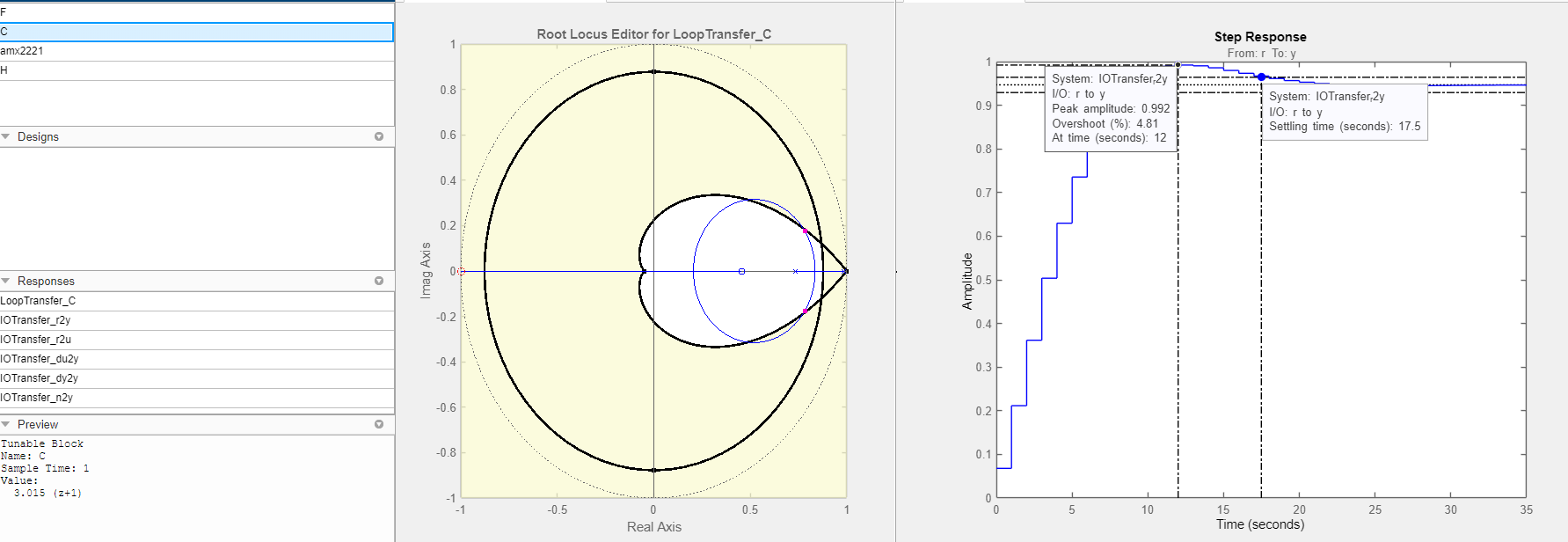
Aquí seleccionar PID, o PD, o PI, acorde a lo que se requiera, en mi caso colocaré PD y probaré la respuesta del sistema



Obteniendo esta respuesta:



Esetá dentro de los valores requeridos sin embargo se pueden modificar los polos para acercarse más al OS requerido



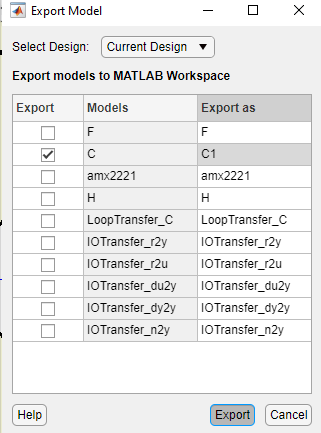
Aquí se obtuvo una mejor respuesta y OS según lo requerido

A continuación dirigirse a export, export tuned blocks

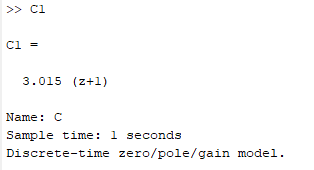
Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Dar un nombre al parámetro C y poner exportar



En command window colocar la variable que se colocó al exportar y se obtendrá



Y por último hacer un PID para obtener los valores de PD, y poder usarlo más adelante

Texto, Carta

Descripción generada automáticamente