CONTROL EN TIEMPO CONTINUO

Informe sobre procesos de Control de Temperatura PID

Presentado por:

Jesús Humberto Dorado De La Cruz

Daniel Alejandro Rodríguez López

Juan Camilo Vásquez

Luis Fernando Miranda

Presentado a:

Ing. Hermes Fabián Vargas

Ing. en Automática Industrial

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

2016

Objetivo de la Practica:

Diseñar una planta de temperatura para proporcionarle un Controlador PID por medio de una tarjeta Arduino (Mega 2560 en este caso), y utilizando las herramientas de Matlab, Simulink para hacer la identificación de la planta. El control del sistema una vez identificado y sintonizado el controlador, será atreves de un set point que modificará el valor de la temperatura a la cual queremos llevar la planta, y así suministrar los valores respectivos de PWM para el motor y el generador de calor (en este caso será una bombilla) para compensar la variación del set point; la lectura de la temperatura se hará por un sensor LM35.

Conceptos a tener en cuenta:

- **Arduino:**

  Arduino es una plataforma de creación de prototipos de código abierto basado en fácil de usar hardware y software. Placas Arduino son capaces de leer las entradas- la luz en un sensor, un dedo sobre un botón o un mensaje de Twitter - y lo convierten en una salida - la activación de un motor, encender un LED, publicar algo en línea. Se puede decir que su tablero qué hacer mediante el envío de un conjunto de instrucciones al microcontrolador en el tablero. Para ello se utiliza el lenguaje de programación de Arduino (basado en el cableado), y el software de Arduino (IDE), sobre la base de procesamiento.

  (Arduino, 2016, www.arduino.cc)

- **Arduino Mega 2560:**

  El Mega 2560 es una placa electrónica basada en el Atmega2560. Cuenta con 54
  pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie de hardware), un oscilador de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o la corriente con un adaptador de CA a CC o una batería para empezar. El tablero de 2560 mega es compatible con la mayoría de los shield para el Uno y las anteriores juntas de Duemilanove o Diecimila. El Mega 2560 es una actualización de la Arduino Mega, al que sustituye.

  (Arduino, 2016, www.arduino.cc)

- **Matlab**

  Es una plataforma que está optimizado para la solución de problemas de ingeniería y científicas. El lenguaje MATLAB basado en la matriz es la forma natural en la mayoría del mundo para expresar matemática computacional. Los gráficos integrados hacen que sea fácil de visualizar y obtener información a partir de datos. Cuenta con una vasta biblioteca de cajas de herramientas prediseñados que le permite comenzar de inmediato con algoritmos esenciales a su dominio. El entorno de escritorio invita a la experimentación, la exploración y el descubrimiento. Estas herramientas y capacidades de MATLAB están rigurosamente probados y diseñados para trabajar juntos.

  (MathWorks, 2016, www.mathworks.com)

- **Simulink**

  Simulink es un entorno de diagrama de bloques para la simulación multidominio y diseño basado en modelos. Es compatible con la simulación, generación automática de código, y la prueba continua y verificación de sistemas integrados.
  Simulink ofrece un editor gráfico, bibliotecas de bloques personalizables y solucionadores para el modelado y simulación de sistemas dinámicos. Se integra con MATLAB, lo que le permite incorporar algoritmos de MATLAB en los modelos de simulación y los resultados de exportación a MATLAB para su posterior análisis.

  (MathWorks, 2016, www.mathworks.com)

- **Sensor LM35**

  La serie LM35 son dispositivos de temperatura en circuito integrado de precisión
  con una tensión de salida linealmente proporcional a la temperatura en grados centígrados. El dispositivo LM35 tiene una ventaja sobre los sensores de temperatura lineales calibradas en grados Kelvin, ya que el usuario no está obligado a restar una gran tensión constante de la salida para obtener el escalamiento en centígrados conveniente. El dispositivo LM35 no requiere ninguna calibración externa o recorte para proporcionar precisiones típicas de ± ¼ °C a temperatura ambiente y ± ¾ °C durante un total de entre -55 °C a 150 °C. Tiene una impedancia de salida baja, su salida es lineal, y debido a su calibración precisa el dispositivo LM35 hace que la interfaz de lectura de salida o circuito de control sea fácil. El dispositivo se utiliza con fuentes de alimentación individuales, o con más y menos suministros. El dispositivo LM35 consume sólo 60 µA de la alimentación, tiene muy bajo auto-calentamiento menor de 0.1 °C en el aire inmóvil. El dispositivo LM35 está

calificado para operar en un rango de temperatura de -55 °C a 150 °C, mientras que el dispositivo LM35C está pensado para un rango de -40 °C a 110 °C (-10 °C con una precisión mejorada).

(Texas Instruments, 2016, http://www.ti.com/lit/ds/symlink/lm35.pdf)

- **PWM en Arduino**

  Modulación de ancho de pulso o PWM, es una técnica para obtener resultados
  análogos con medios digitales. Control digital se utiliza para crear una onda cuadrada, una señal de conmutación entre encendido y apagado. Este patrón de encendido y apagado puede simular tensiones en el medio completo en (5 voltios) y desactivación (0 voltios) cambiando la parte de las veces la señal pasa en comparación con el tiempo que la señal pasa fuera. La duración del "tiempo" se llama el ancho de pulso. Para conseguir variando los valores analógicos, se cambia, o modular, que el ancho de pulso. Si repite este patrón de encendido y apagado suficientemente rápido con un LED por ejemplo, el resultado es como si la señal es una tensión constante entre 0 y 5V controlar el brillo del LED".

  (Arduino, 2016, www.arduino.cc)

- **Optoacoplador**

  Es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente opto electrónico, normalmente en forma de fototransistor o fototriac. De este modo se combinan en un solo dispositivo semiconductor, un fotoemisor y un fotoreceptor cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP. Se suelen utilizar para aislar eléctricamente a dispositivos muy sensibles.

  (Wikipedia, 2016)

- **TRIAC**

  Un TRIAC o Tríodo para Corriente Alterna es un dispositivo semiconductor de 4 capas, de la familia de los tiristores. La diferencia con un tiristor convencional es que el TRIAC es bidireccional. De forma coloquial podría decirse que el TRIAC es un interruptor capaz de conmutar la corriente alterna. Su estructura interna se asemeja en cierto modo a la disposición que formarían dos SCR en direcciones opuestas. Posee tres electrodos: A1, A2 (en este caso pierden la denominación de ánodo y cátodo) y puerta (Gate). El disparo del TRIAC se realiza aplicando una corriente al electrodo de Gate/Compuerta. (Wikipedia, 2016)

Desarrollo de la practica:

Elementos y equipos utilizados:

- Caja para montaje en madera (previamente diseñada)
- 1 Arduino Mega 2560.
- 1 sensor de temperatura LM35.
- 1 Fuente de Voltaje de 12V DC
- Cables jumpers.
- 1 display LCD
- 1 Resistencia de 20 Ω.
- 1 Resistencia de 220 Ω.
- 1 Resistencia de 1k Ω.
- 1 Potenciómetro de 5 kΩ.
- 1 Potenciómetro de 10 kΩ.
- 1 TRIAC BT136.
- 1 optoacoplador MOC3022.
- 1 Transistor 2N2222A
- 1 Transistor TIP31C
- 2 Protoboard
- 1 Bombilla incandescente de 70 w.
- 1 Motor DC 12V
- 1 Extensión para fuente alterna domiciliaria de 120V AC
- 1 plafón.
- 1 termómetro
- MATLAB 2014a y librerías de Simulink para Arduino.
- Software de para lenguaje de programación Arduino.

Procedimiento:

1. **Montaje.**

   Procedimos a ensamblar todos los componentes mencionados anteriormente en las protoboard, basados en el siguiente diagrama.
   (Se adjunta el archivo que contiene el diagrama, extensión .fzz el cual se puede ejecutar con el programa Fritzing).

   Imagen 1: modelo real
   Imagen 2: modelo esquemático

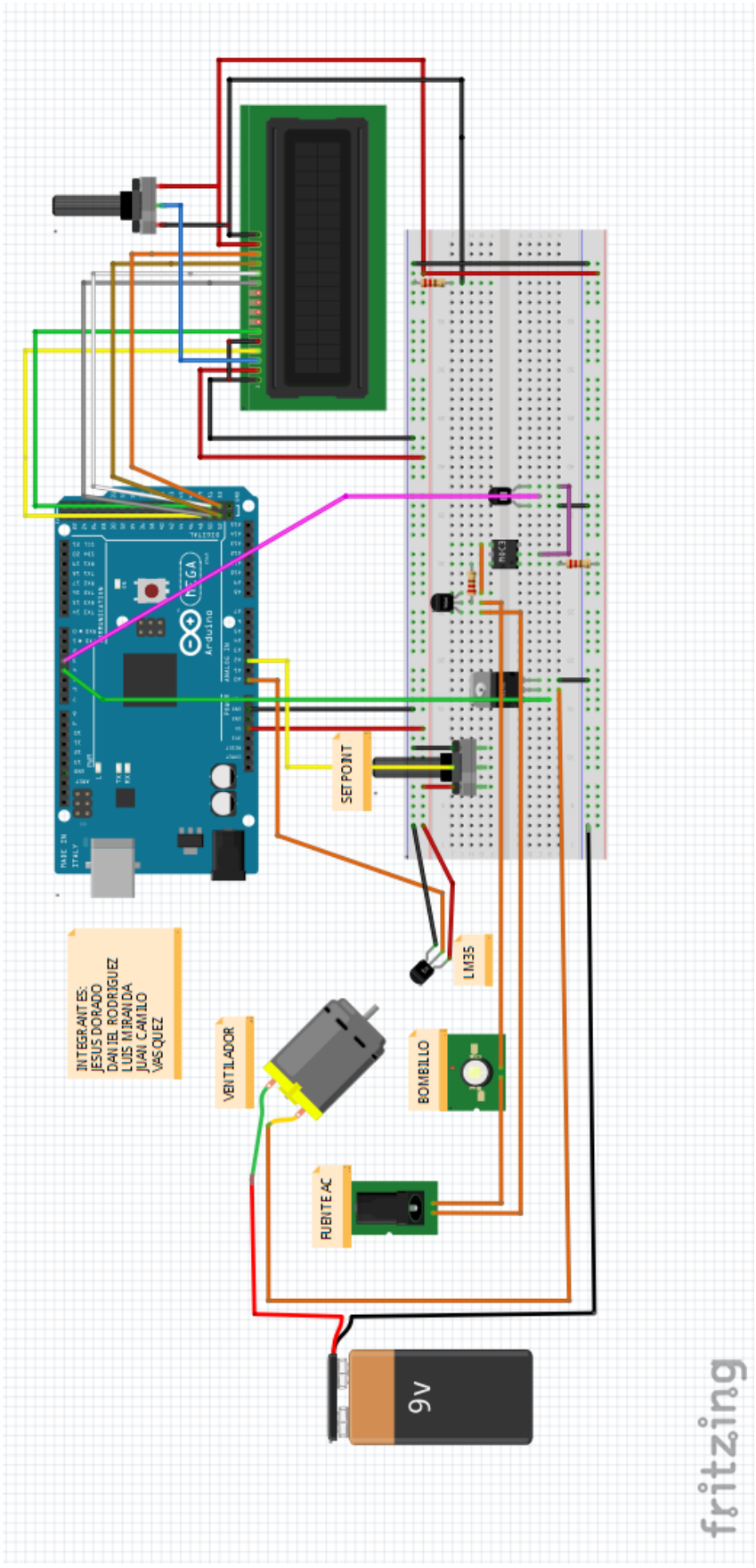


Imagen 1. Diagrama de conexiones con protoboard y Arduino

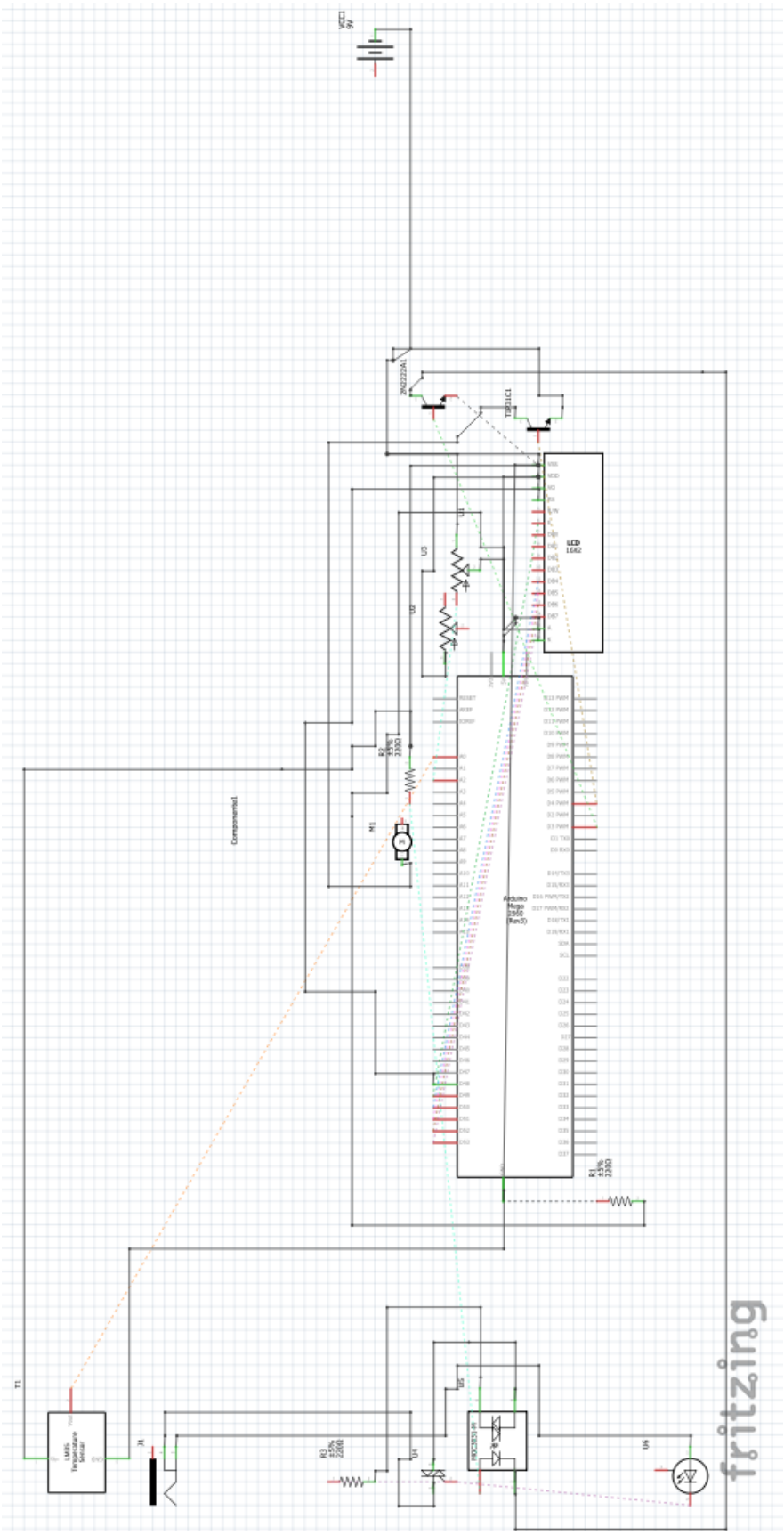


Imagen 2. Diagrama Esquemático

El paso siguiente fue instalar los las protoboard, el plafón, el motor, los potenciómetros, la LCD y la tarjeta Arduino en la caja de madera ensamblada anteriormente. En la siguiente imagen (*Imagen 3*) se puede apreciar un avance en el procedimiento del montaje de la planta.

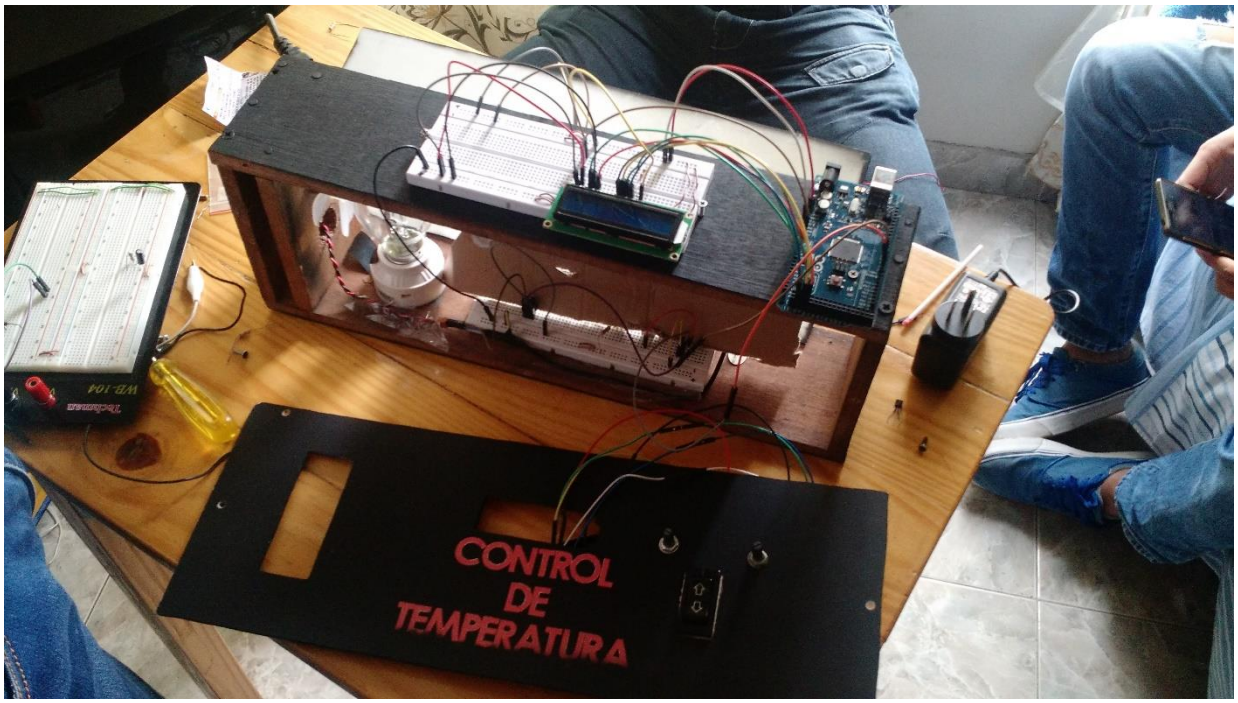


Imagen 3. Montaje parcial de la Planta

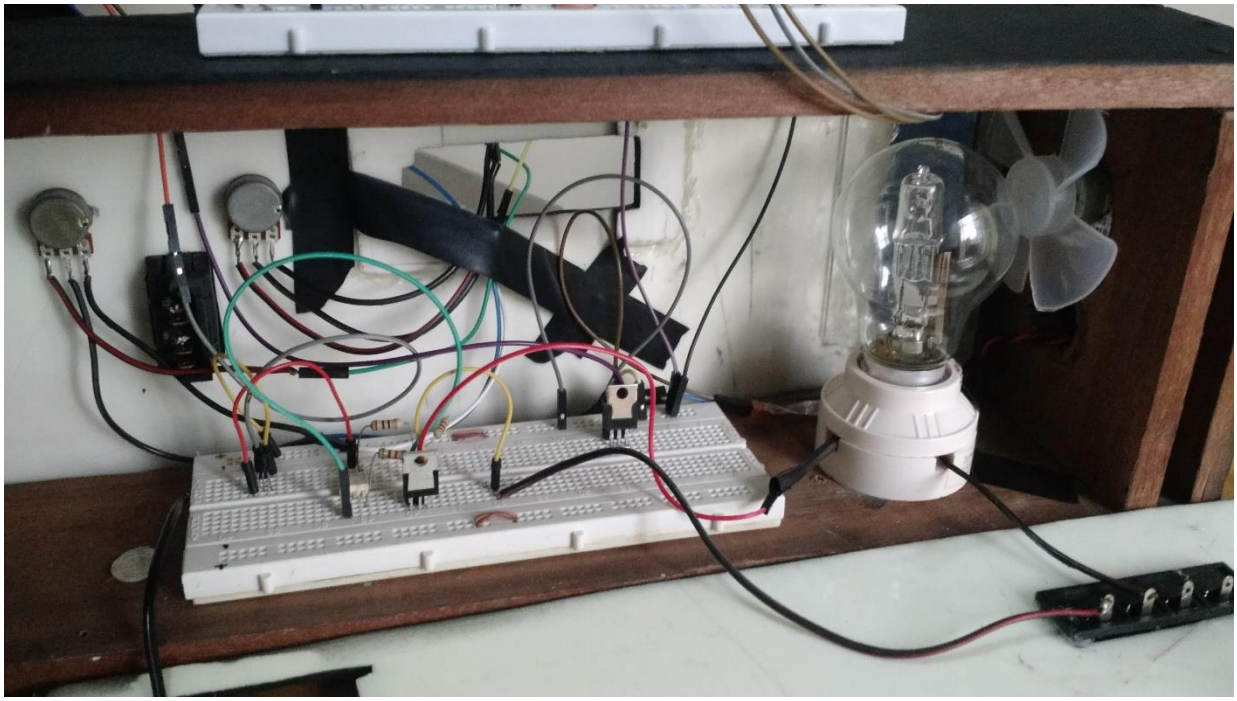


Imagen 4. Circuitería interna, cto de potencia (AC)

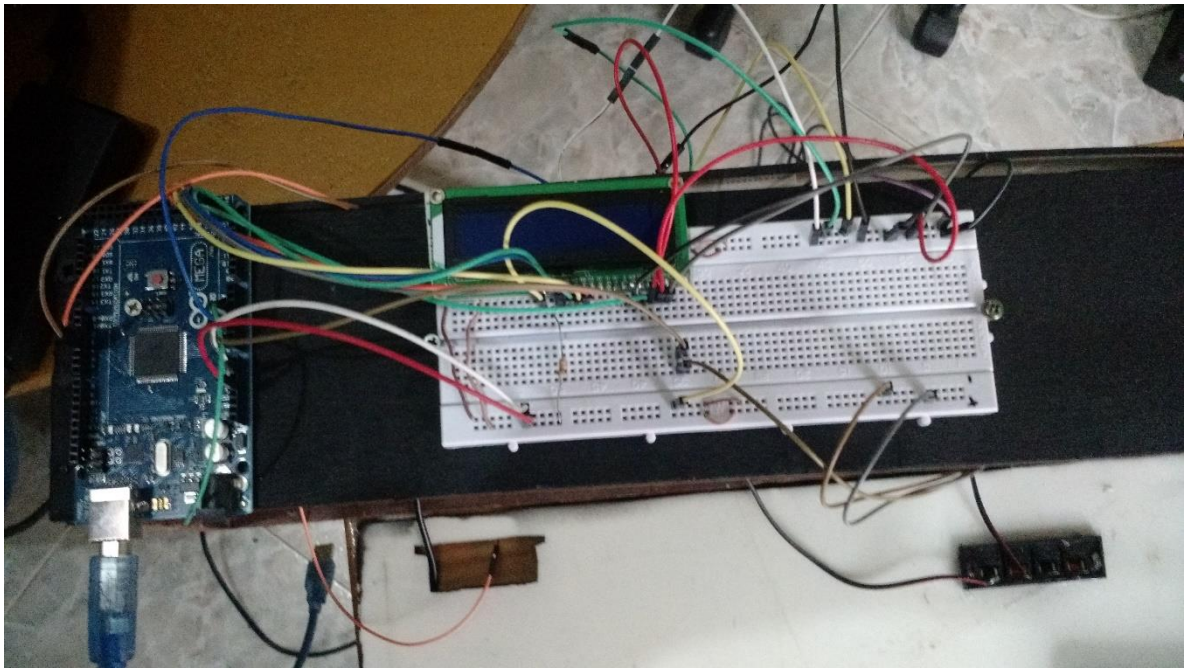
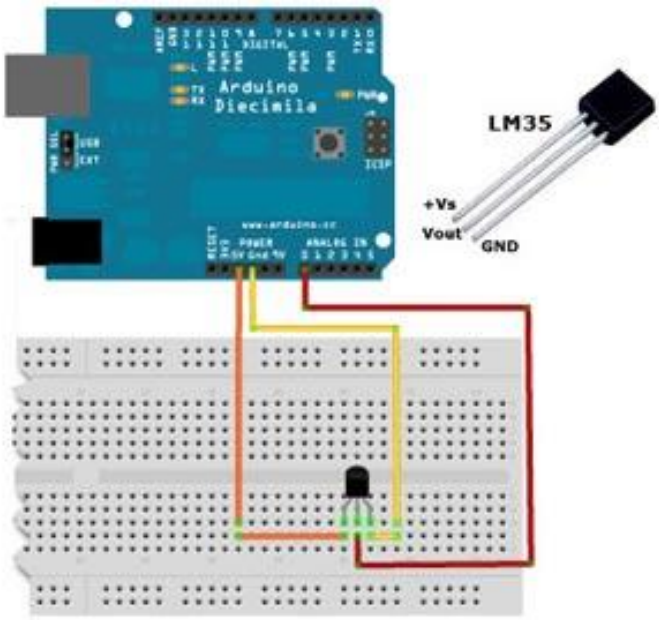


Imagen 5. Circuitería Externa, para LCD, lectura de temperatura por el sensor LM35 y Arduino

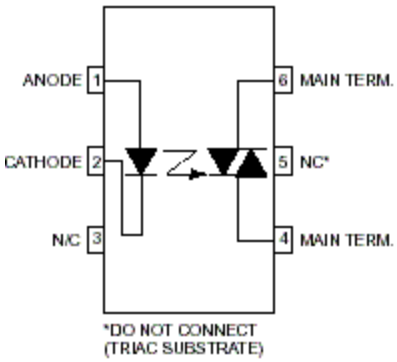
A continuación, se muestra la manera de conectar los elementos como el sensor, el MOC y el Triac.

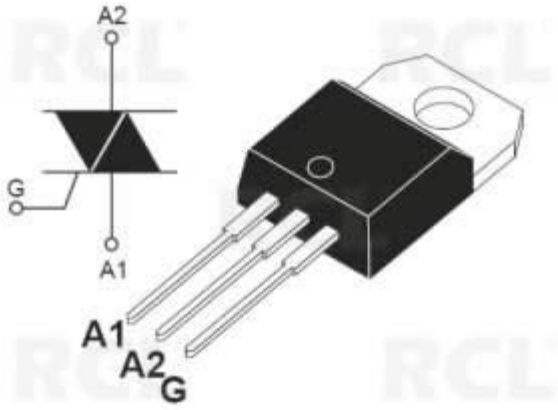
Sensor de temperatura LM35: Para realizar la lectura de los datos de temperatura se utilizó el dispositivo LM35, el cual es un sensor de temperatura que opera con 10 mV / °C. Dicho sensor se implementó en la parte superior de la planta justo debajo de la bombilla incandescente. El método de conexión con la tarjeta Arduino se puede observar en la siguiente imagen.



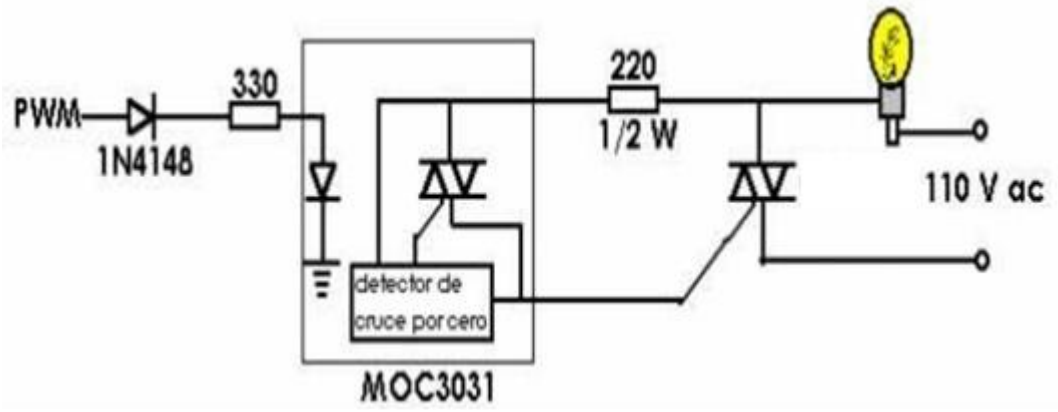
Control de encendido para la bombilla incandescente.

Para poder realizar el control se requiere de una etapa de potencia, con el fin de que la señal de 5 v generada por la salida digital del Arduino por medio de un pulso PWM pueda controlar la lámpara que trabaja con 120 V/AC, para ello se utilizó un arreglo con un Optoacoplador MOC3022 y un TRIAC BT136 como se aprecia en las siguientes imágenes.





La conexión se puede ver en la siguiente imagen, aclarando que la entrada del PWM se realiza por el pin 2 del MOC ya que con el 1 se hace la alimentación de 5 V, además se utiliza un transistor 2N2222 para regular la corriente que entra hacia led del MOC y no se comporte como un Relé, entonces en este caso estaría reemplazando el diodo que se ve en el siguiente diagrama



La señal PWM proveniente del Arduino controla el encendido del LED interno del Optoacoplador y con ello se controla directamente un fototriac, el cual al recibir suficiente luz se dispara y permite controlar la corriente de disparo en la compuerta del TRIAC BT136 y por consecuencia el encendido de la lámpara incandescente que trabaja con 120 VCA.

2. Procedimiento para la Identificación de la Planta.

Como primera medida se generó un impulso al actuador, es decir el que va a generar el calor para elevar la temperatura que en este caso es la bombilla y como realimentación la lectura del sensor LM35 para saber el comportamiento de la planta ante una excitación.
A continuación, se proporcionan las imágenes donde se puede apreciar el código en Simulink que hace posible la lectura de las señales para determinar el modelo de la planta. (*Ilustración 1 y 2*)

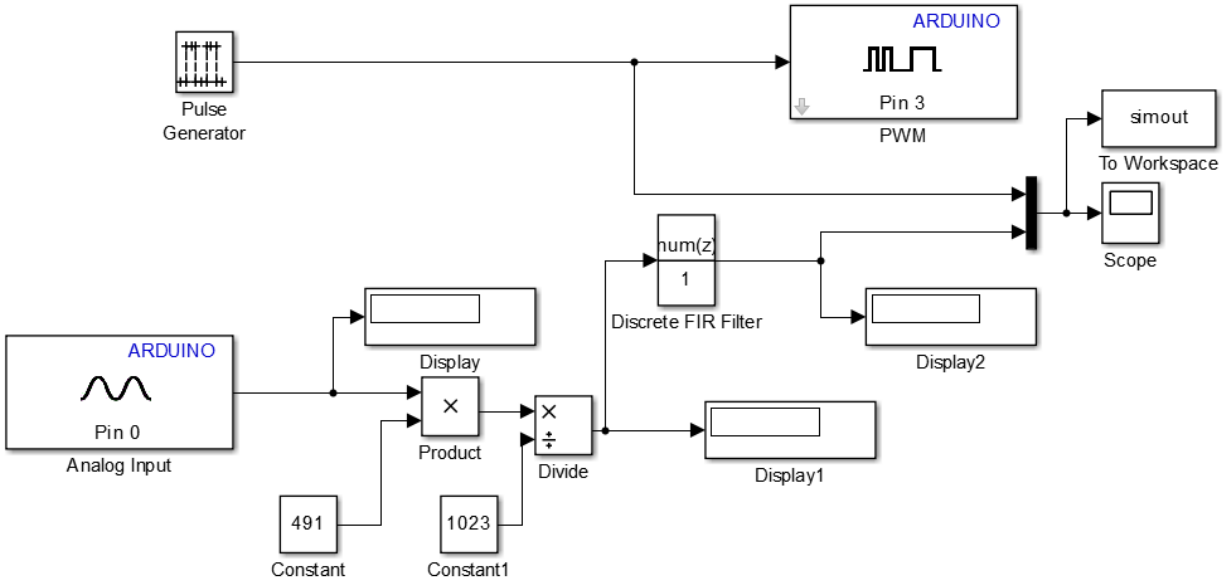


Ilustración 1. Código Simulink para identificación de la planta

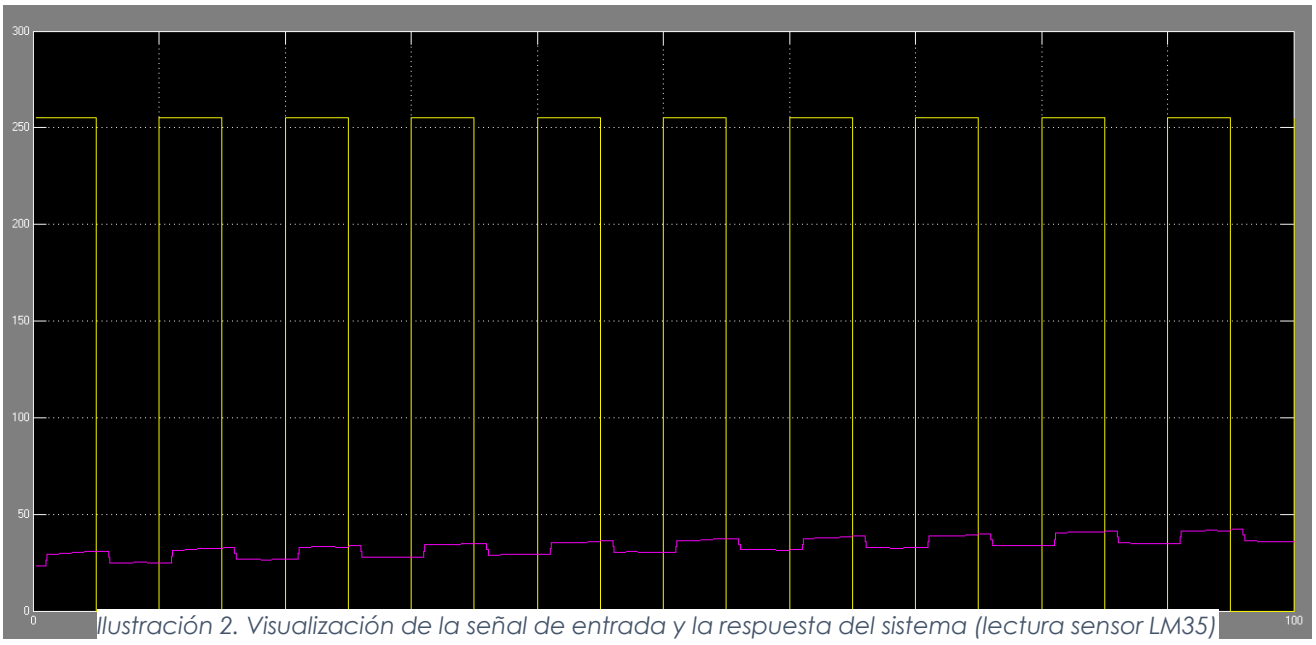


Ilustración 2. Visualización de la señal de entrada y la respuesta del sistema (lectura sensor LM35)

En el procedimiento se hicieron de 0 a 500 muestras en intervalos de 0.1, para obtener un modelo más exacto, con lo cual se escoge las mejores muestras para obtener los valores de I/O que se van a utilizar para hacer la identificación. (*Ilustración 3*)

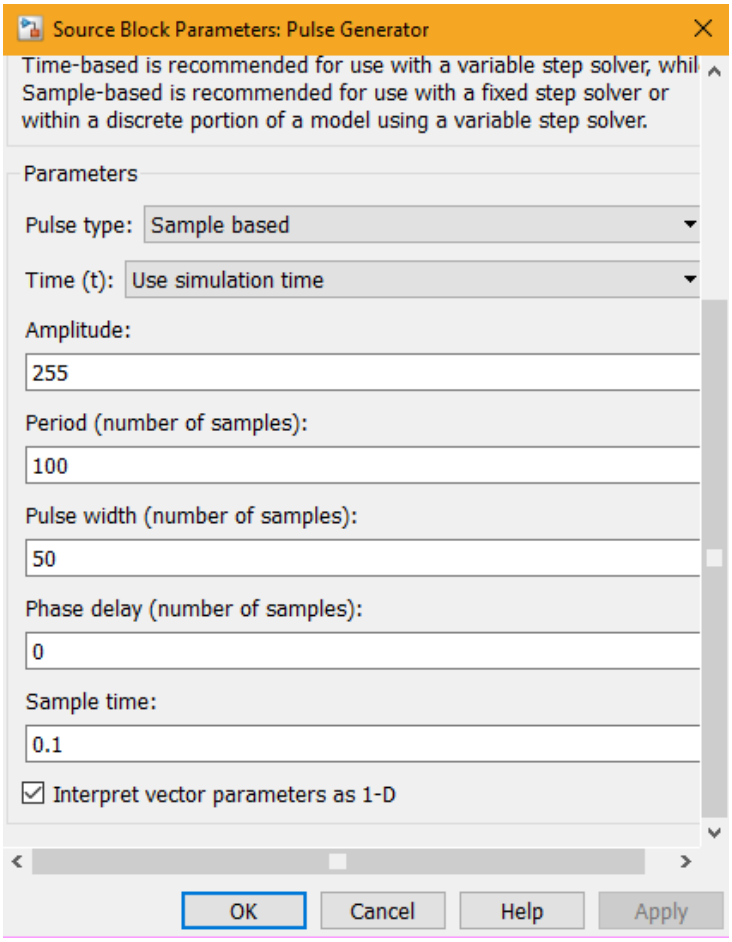


Ilustración 3. Parámetros para el reconocimiento de la Planta

Después se procede a crear dos vectores (para cada señal), con el rango de tiempo y los datos en ese rango para proporcionarlos a Matlab y crear una gráfica de I vs O. Este procedimiento se hace con la función **ident,** la cual estima una función de transferencia.
El paso siguiente es empezar a variar los polos y ceros de la función obtenida para obtener una exactitud como mínimo de 80%. (*Ilustración 4 y 5*)

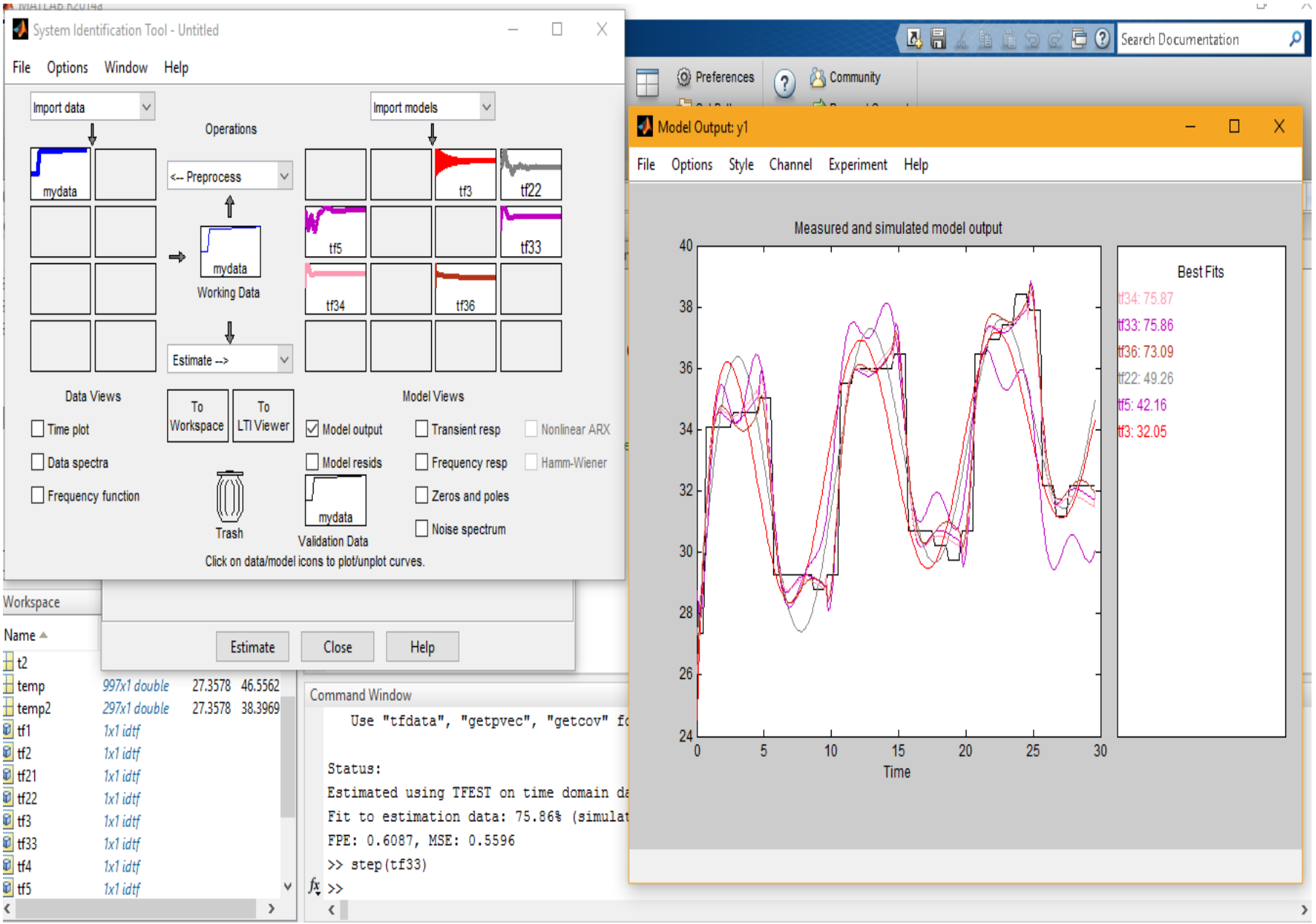


Ilustración 4. Diferentes funciones por el efecto de la variación de los polos y ceros

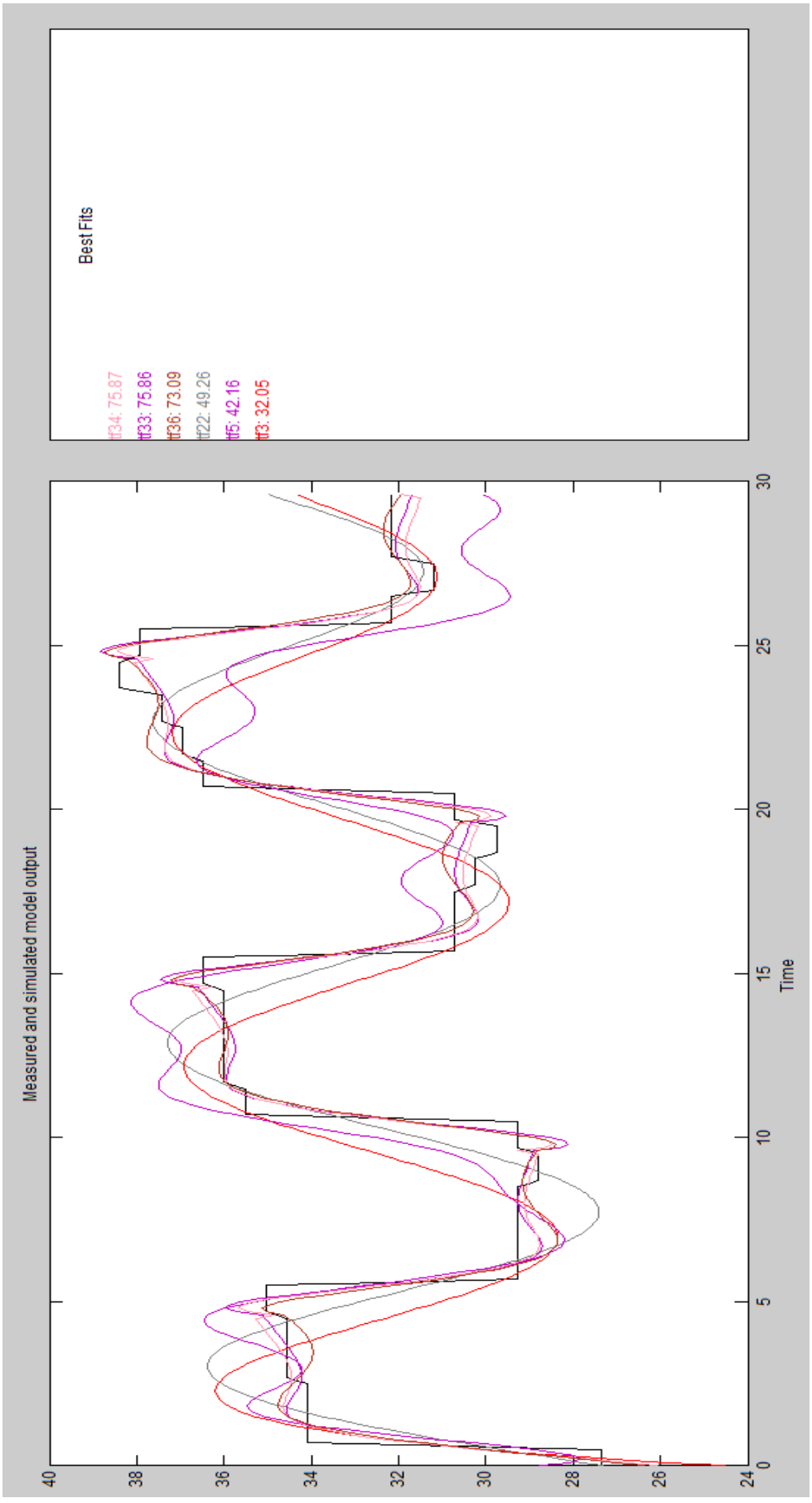


Ilustración 5. Visualización de las diferentes funciones y su aproximación a un modelo más exacto.

Como se puede apreciar en la anterior imagen, en nuestro caso obtener la función de transferencia que cumpliera con el anterior requerimiento no fue posible, ya que el sistema presentaba muchas oscilaciones, pero después de hacer muchas pruebas se obtuvo una función de porcentaje cercano siendo de 75.87%. En la siguiente ilustración se puede observar la función de transferencia estimada. (*Ilustración 6*).

```
tf34 =

  From input "u1" to output "y1":
  0.002001 s^3 - 0.02435 s^2 + 0.1034 s + 0.01344
  -----------------------------------------------
        s^3 + 2.474 s^2 + 4.502 s + 0.03287
```

Ilustración 6. Después de variar los polos y ceros, se encuentra la función de transferencia (tercer orden) que corresponde a casi 76% de aproximación al modelo.

3. Estimación y sintonización del Controlador

Para este procedimiento se hace uso de la herramienta **pidtool,** para hacer la sintonización de la planta, tomando la función de transferencia estimada (en nuestro caso de 3 orden, que proporcionaba casi el 76% de exactitud) para modificar la respuesta del sistema siendo más rápido o más robusto, lo cual modifica las constantes del controlador que estima el **pidtool.**
Una vez obtenido un modelo que cumpla unas condiciones de rapidez, pero estable, se exporta la función de transferencia del controlador, con las variables del mismo.

Como se puede observar en la siguiente imagen, la sintonización se hace para que el sistema cumpla unos requerimientos de rapidez y estabilidad, lo que se hace es hacer una compensación para que el sistema sea eficiente para estas dos características.
(*Ilustración 7*)

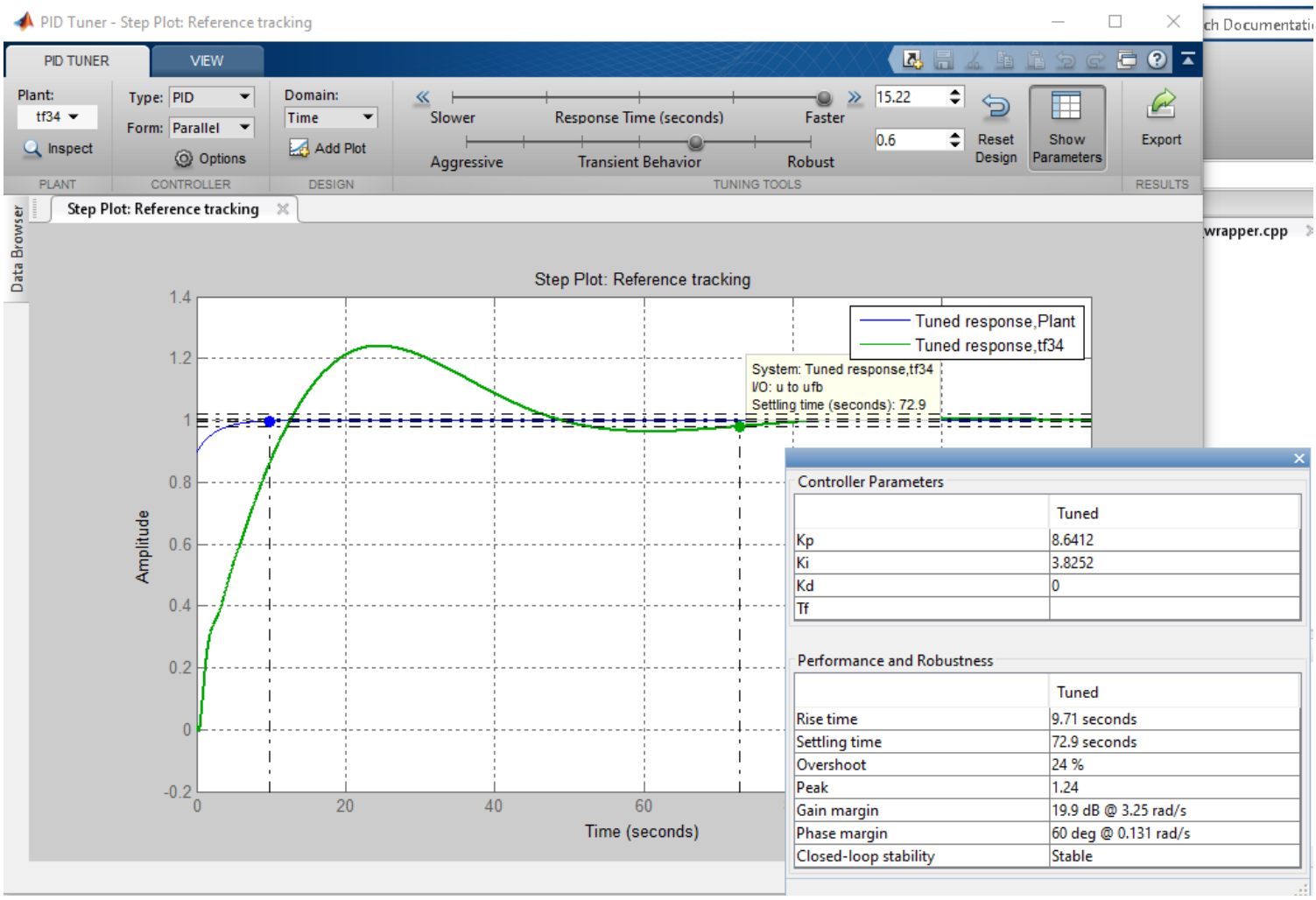


Ilustración 7. Sintonización del PID

Una vez obtenidas las constantes del controlador y la función de transferencia de la planta, se procede a hacer las pruebas tanto en el código de Simulink como en la interfaz de Arduino.

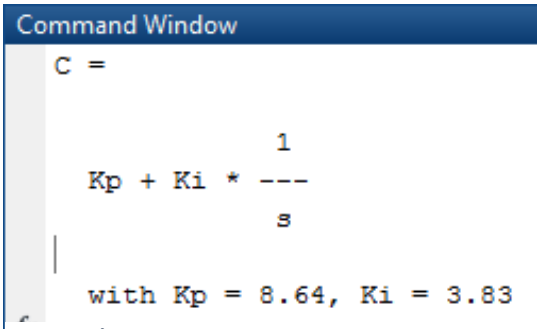


Ilustración 8. Constantes que exporta el pidtool del compensador.

4. Pruebas con Simulink y Arduino

En el caso de Simulink se proporcionan las constantes obtenidas a la función/bloque PID y se procede a activar el sistema. En nuestro caso la planta se comporta de manera muy eficiente y para comprobar que sea el mismo comportamiento de manera teórica, se simula la planta con la función de transferencia estimada y las constantes del controlador para observar gráficamente la respuesta del sistema.

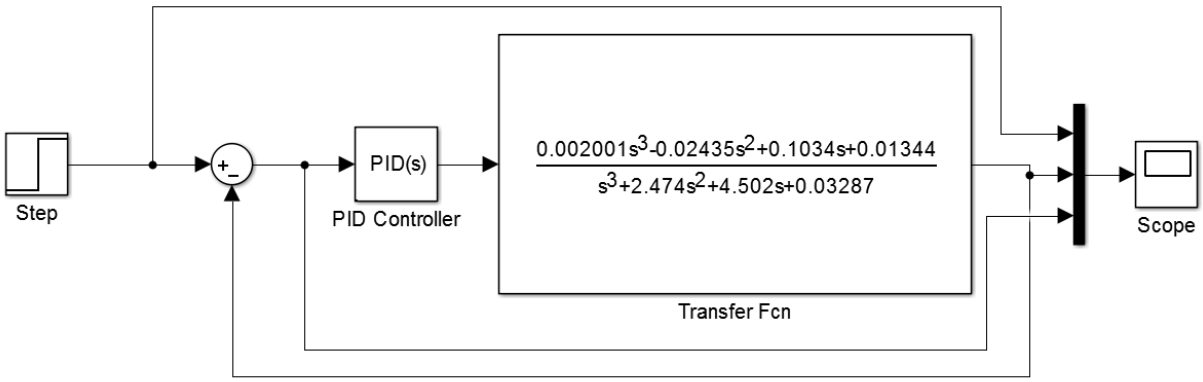


Ilustración 9. Simulación de la planta con su función de transferencia y las constantes obtenidas del PID.

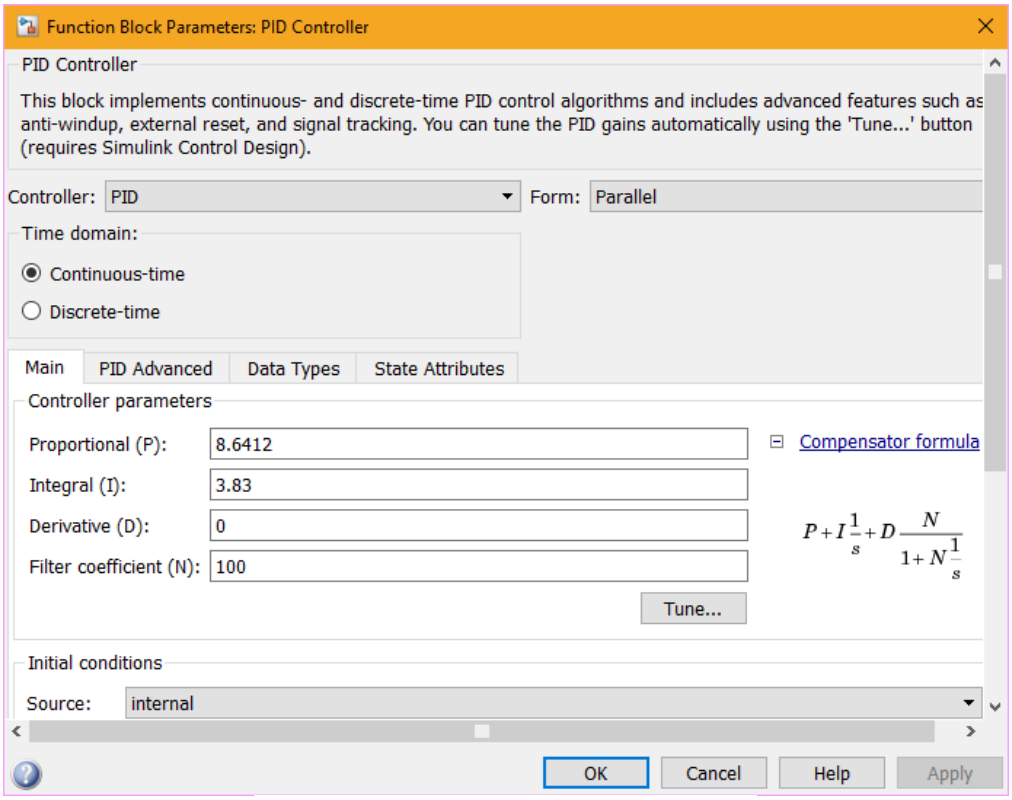


Ilustración 10. Constantes del controlador.

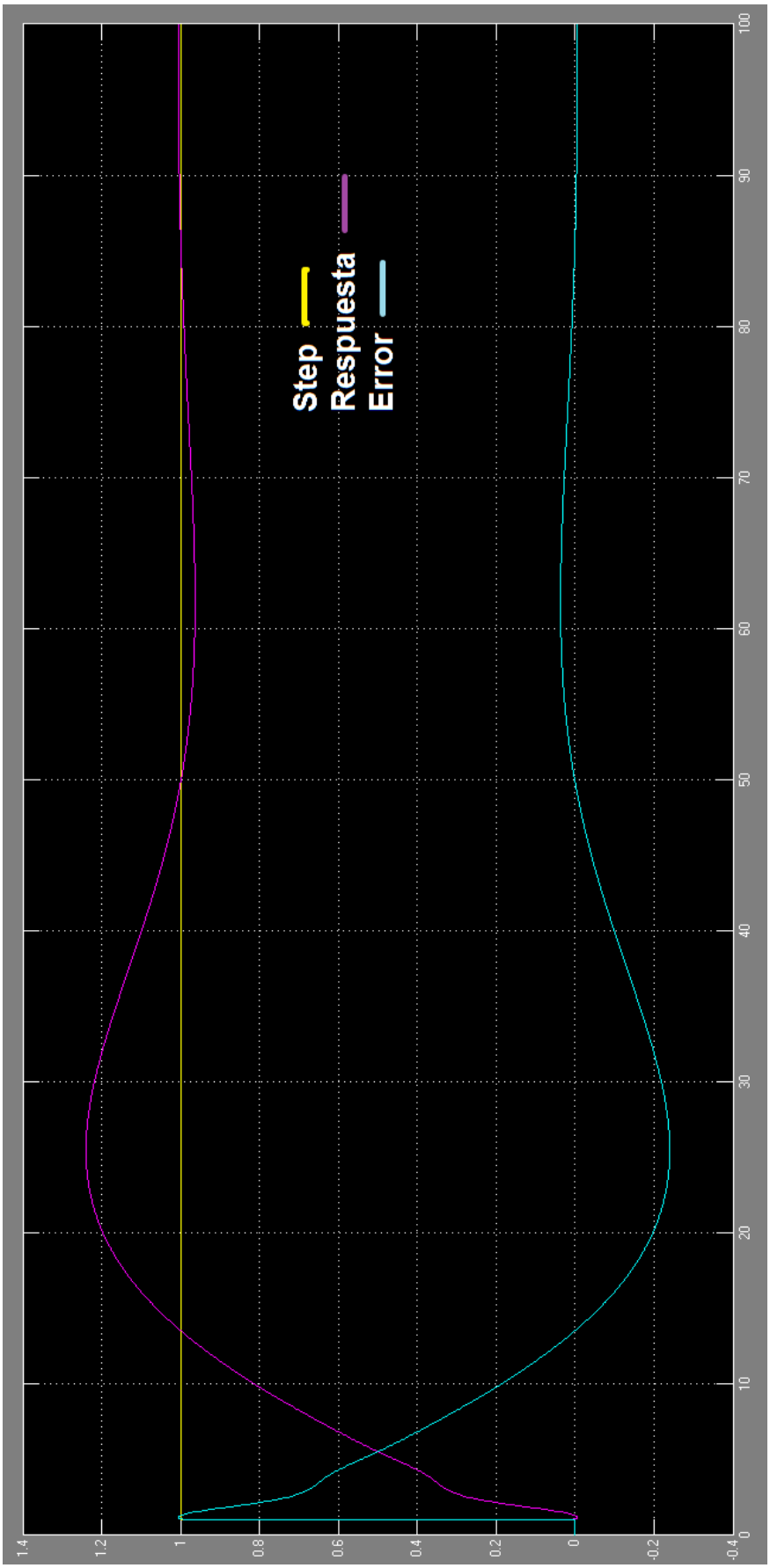


Ilustración 11. Visualización de la planta simulada.

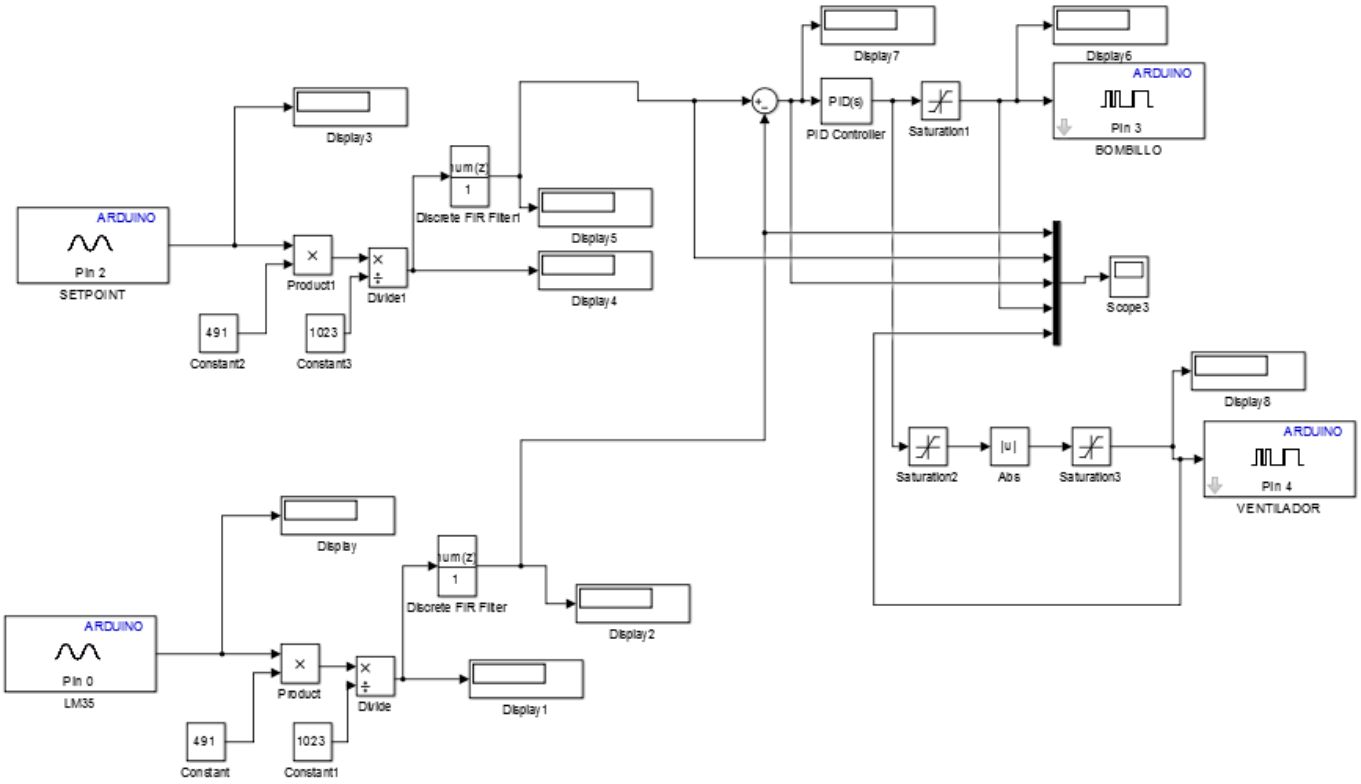


Ilustración 12. Código Simulink de la planta con el controlador PID.

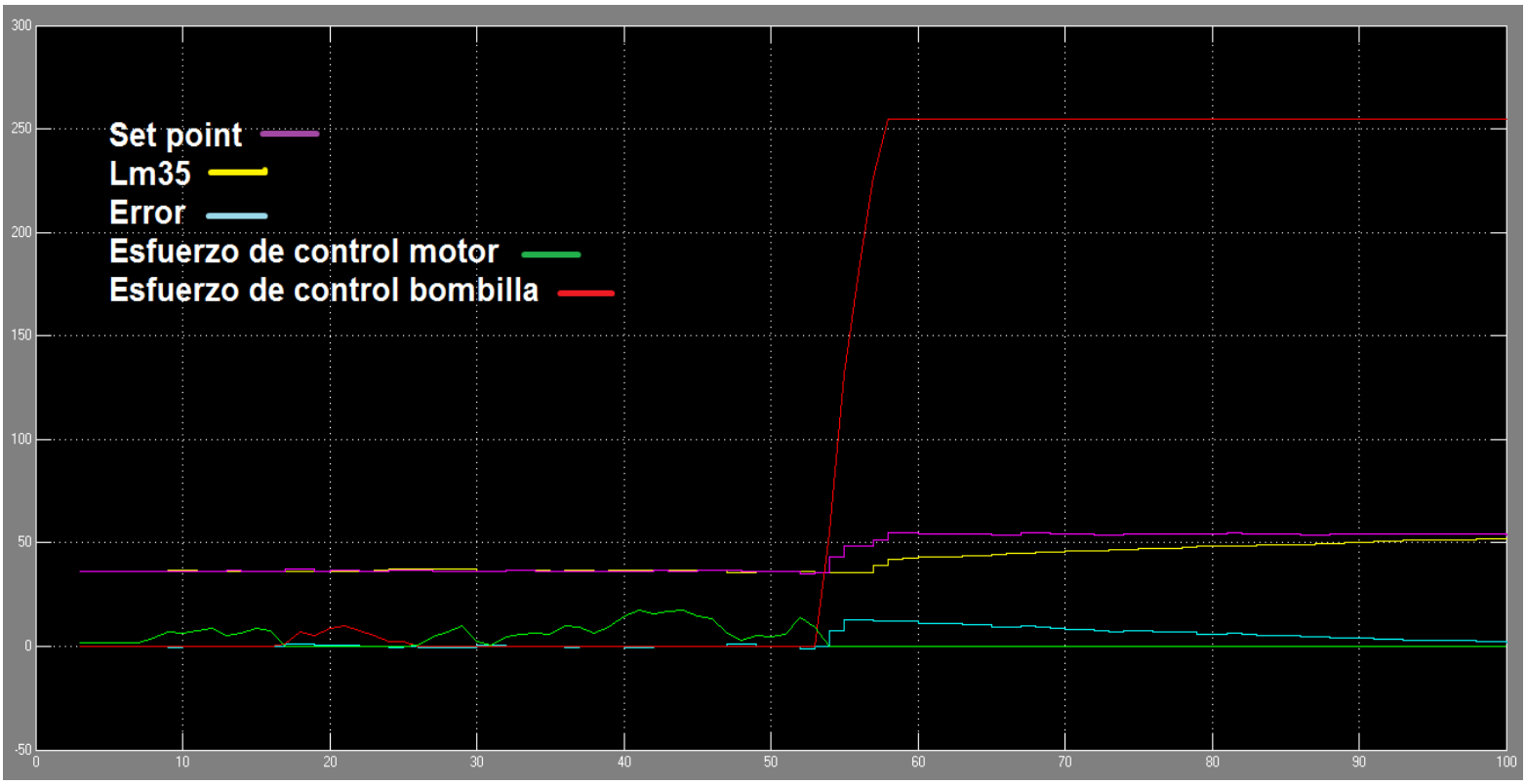


Ilustración 13. Variables de la planta observadas desde Simulink

A continuación, se procede a la creación del bloque para la pantalla LCD 16X2 con **S-FUNTION BUILDER** de la librería de Simulink, se abre su configuración y se realizan los siguientes cambios.

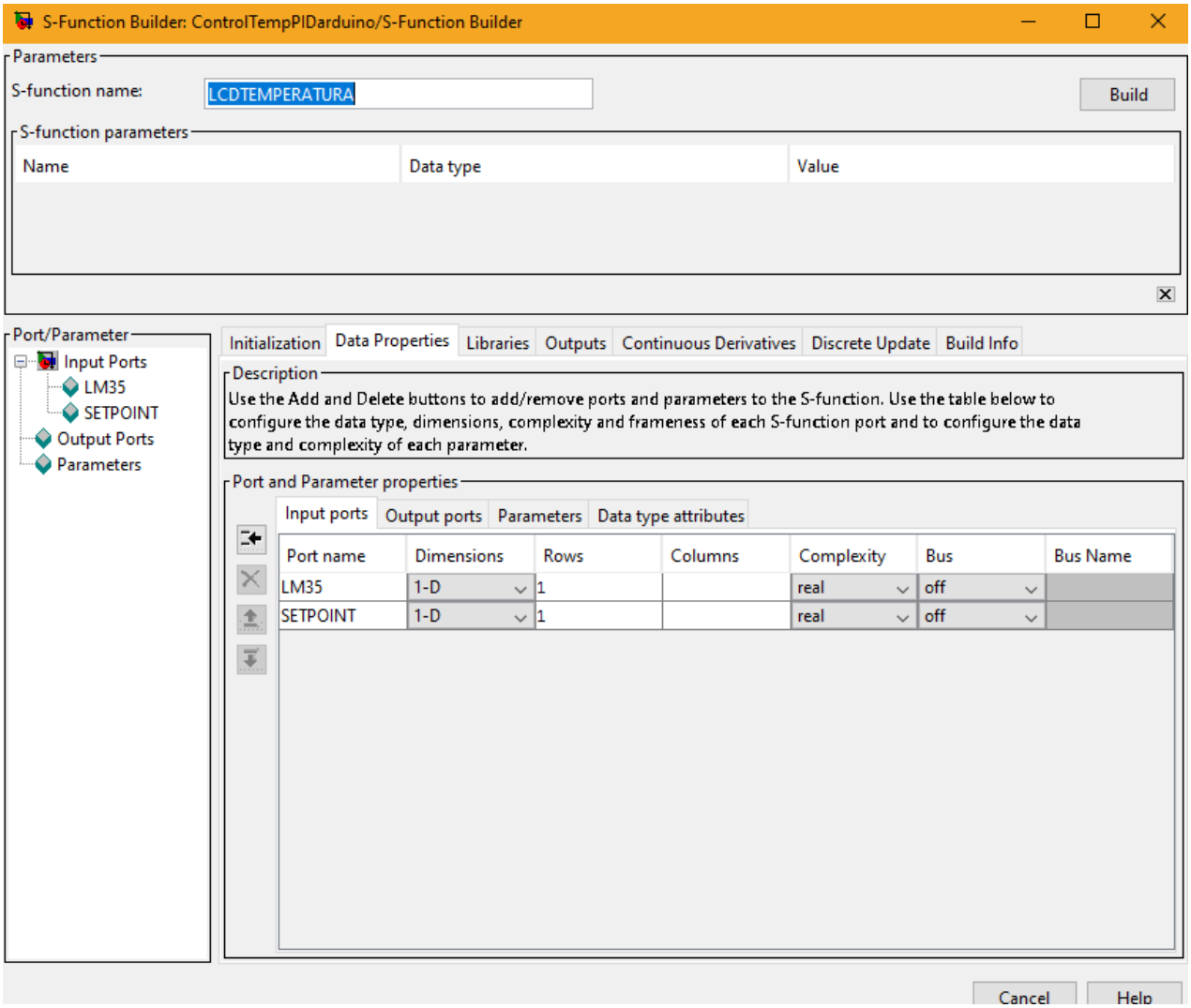


Ilustración 14. Creación de variables hacia la LCD

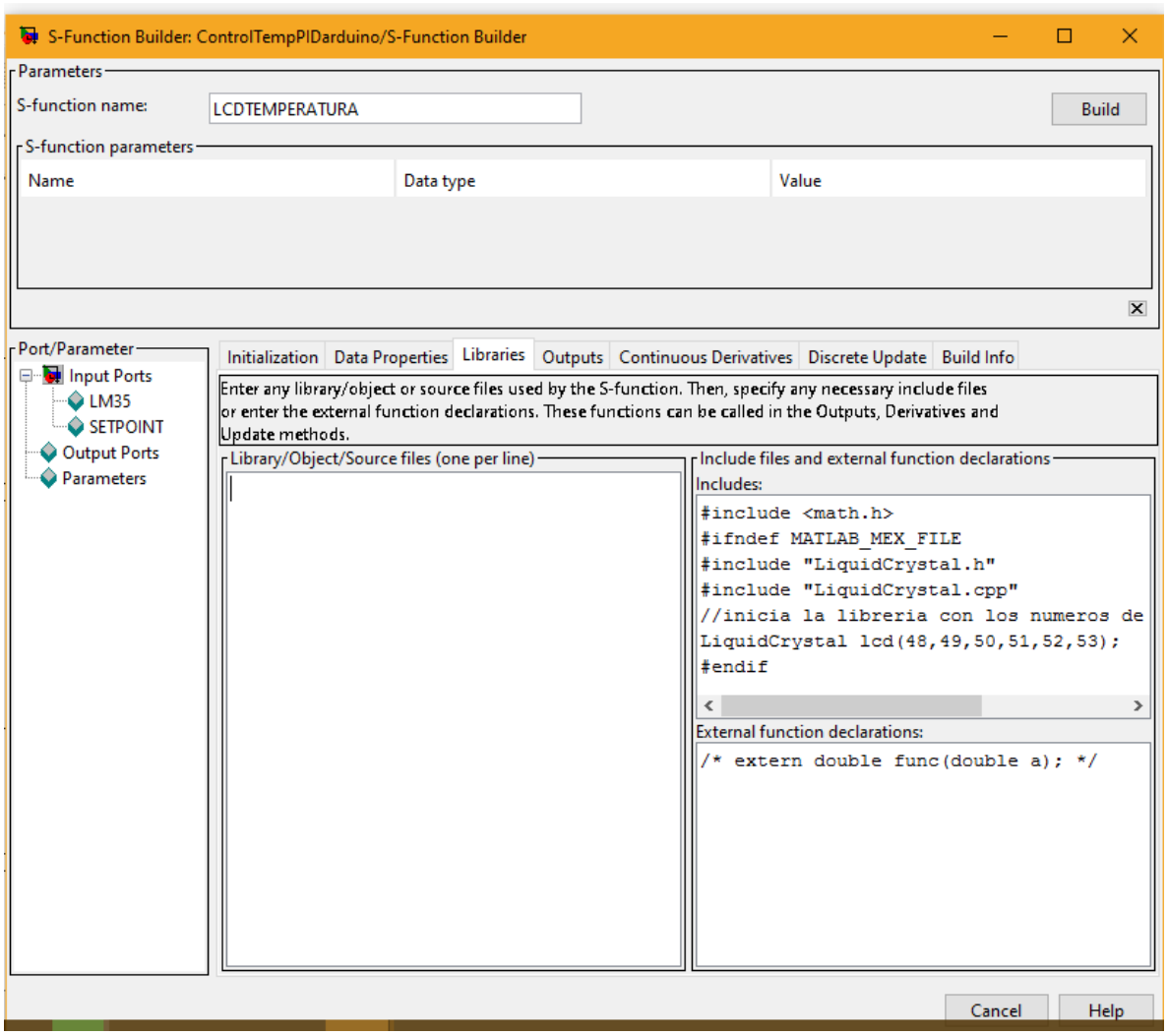


Ilustración 15. Configuración de la librería.

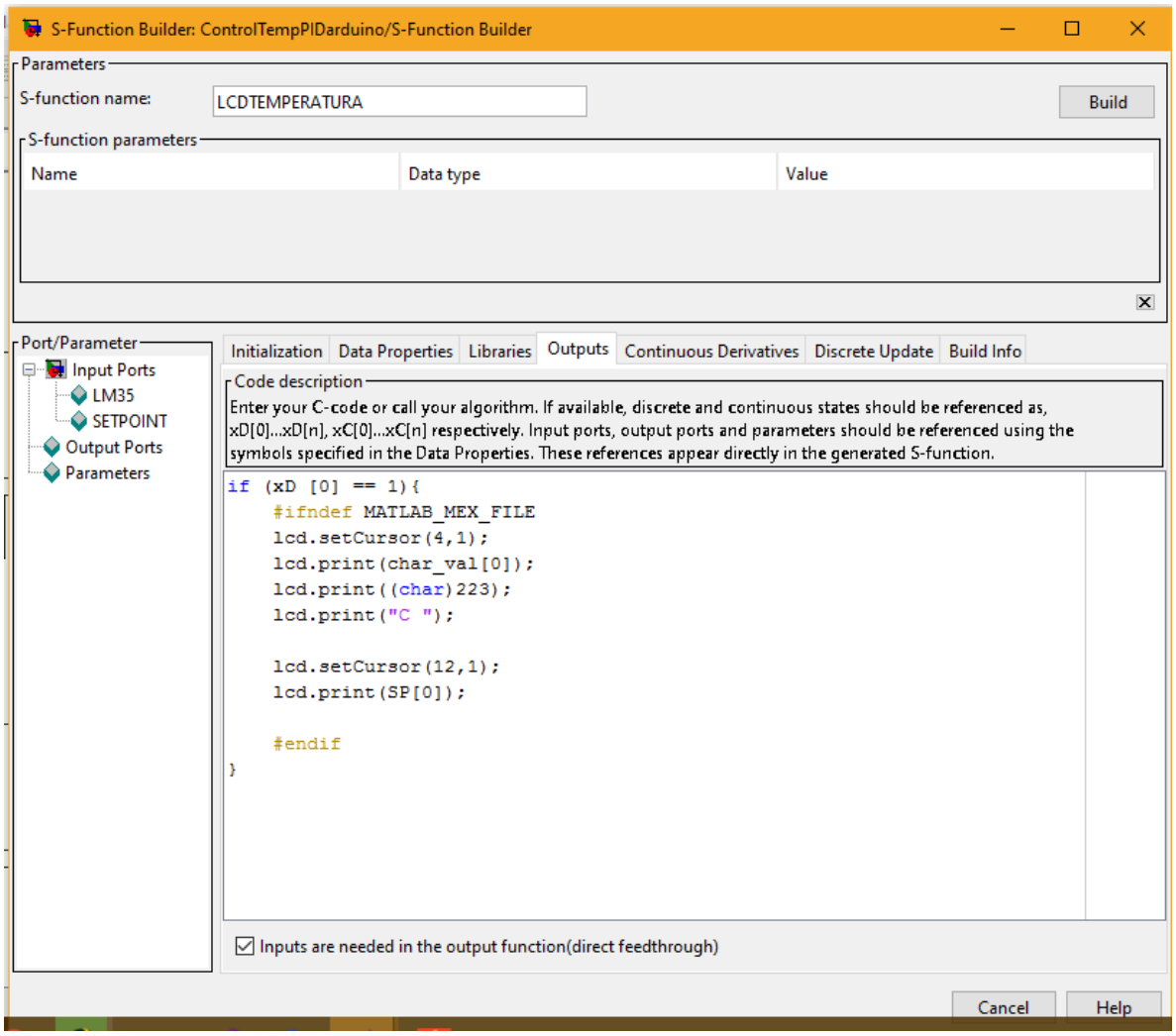


Ilustración 16. Configuración de la salida.

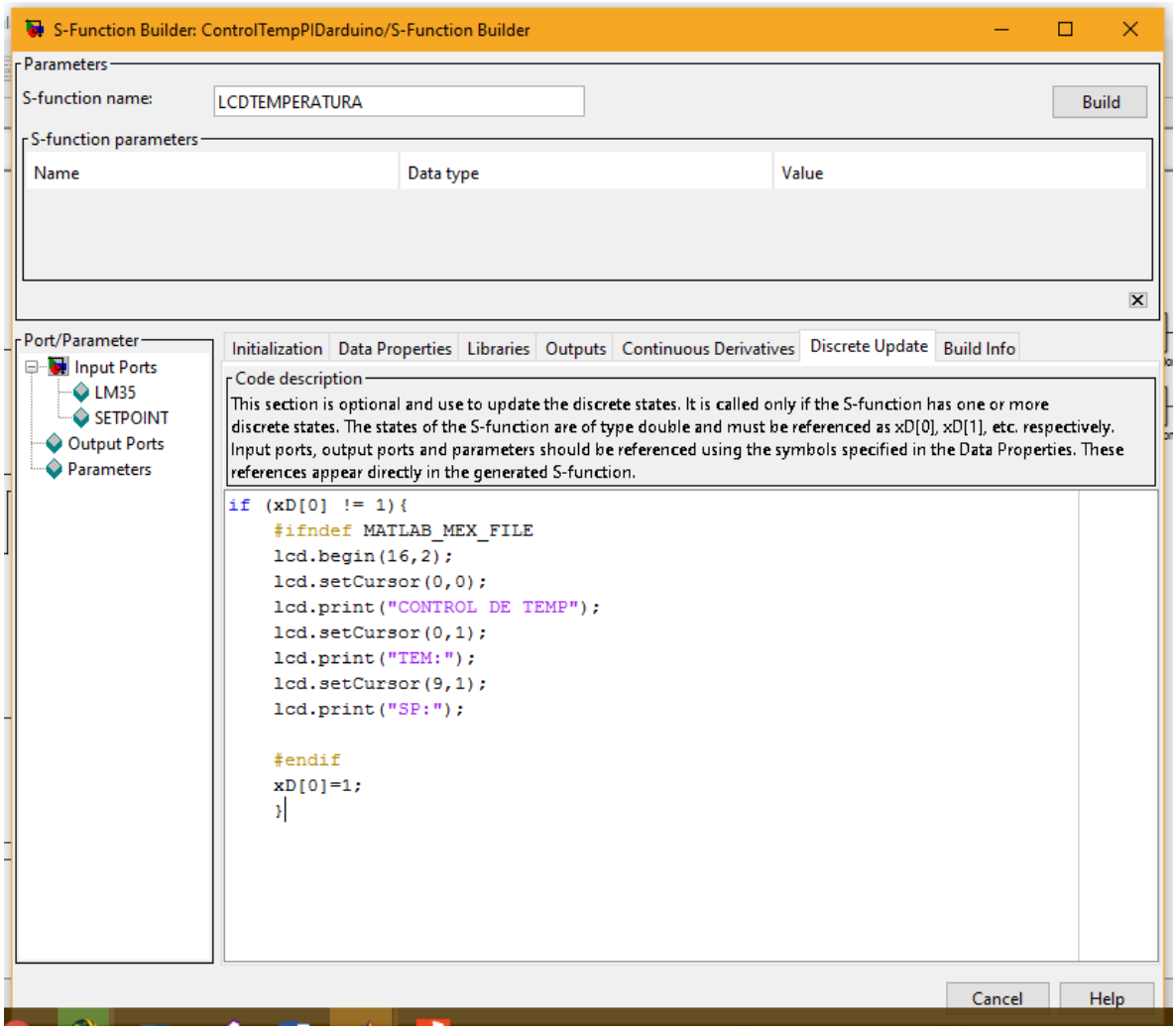


Ilustración 17. Configuración a mostrar en LCD.

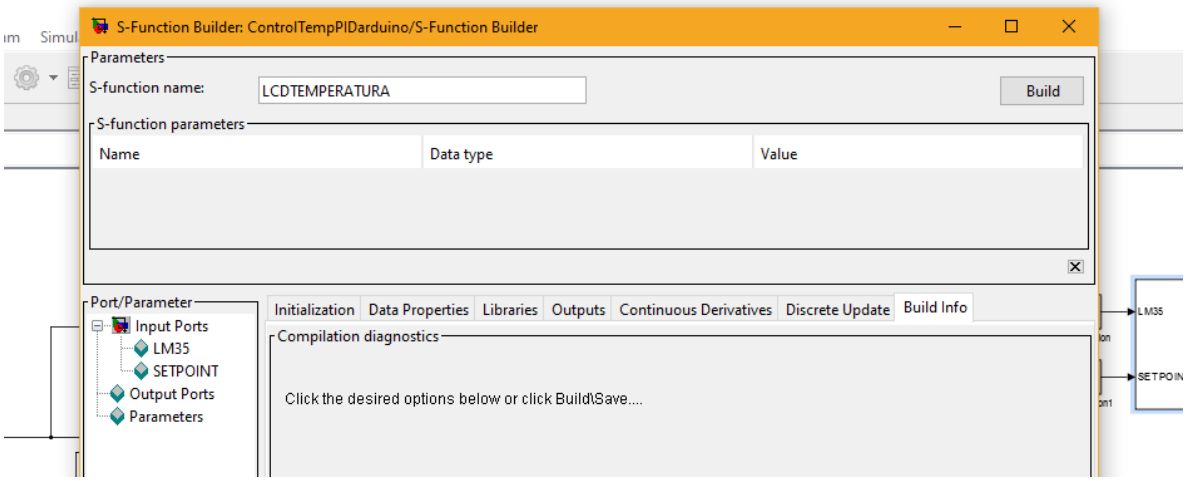


Ilustración 18. Compilación del bloque.

Para la compilación se debe tener instalado Visual Studio 2013.

Ilustración 19. Compilación exitosa.

El archivo “LCDTEMPERATURA.C” se crea en Matlab al finalizar la compilación, lo siguiente que se debe hacer es agregar **extern “C”** renombrar el archivo con extensión **cpp**.

```
 */
extern "C" void LCDTEMPERATURA_Outputs_wrapper(const real_T *char_val,
            const real_T *SP,
            const real_T *xD)
{
/* %%%-SFUNWIZ wrapper Outputs Changes BEGIN --- EDIT HERE TO  END */
```

Ilustración 20. Configuración del codigo generado.

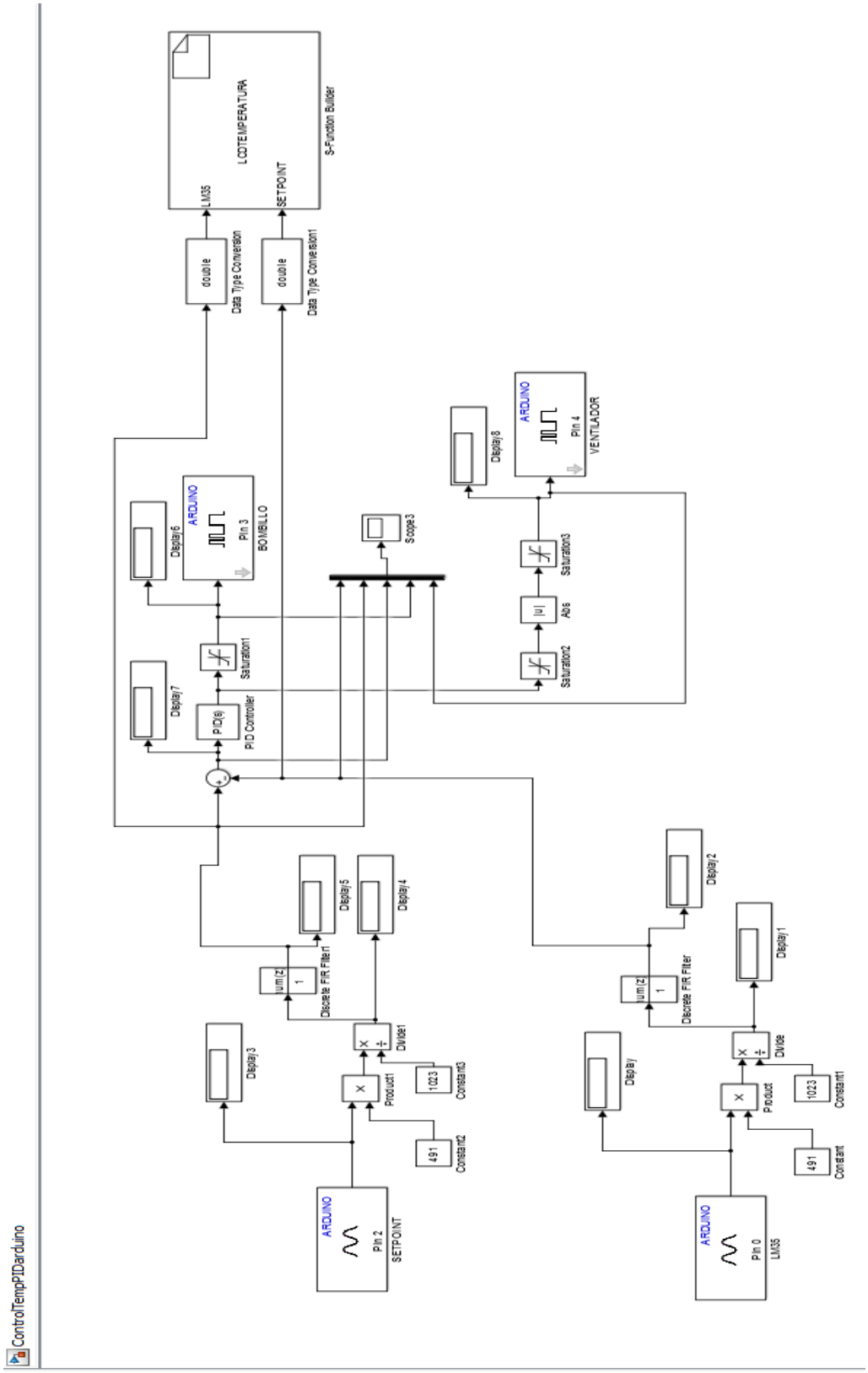


Ilustración 21. Bloque LCD en el programa.

Por otro lado, en el código Arduino con la librería PID, se proporcionan las constantes del controlador y se activa la planta para ver el comportamiento del sistema. Donde se pudo observar la eficiencia de la planta, permitiéndonos concluir que tanto en Simulink como en Arduino funciona de manera correcta y que los tiempos de respuesta del sistema son los esperados tanto en rapidez como en estabilidad.
(*Código adjunto al informe*)

```
%% control PID LM35DZ CONTROL TEMPERATURA
clear all
clc

%planta
num=[0.002001 -0.02435 0.1034 0.01344];
den=[1 2.474 4.502 0.03287];
G=tf(num,den);

%control PID
kp=8.64;
ki=3.83;
kd=0;

num1=[kd kp ki];
den1=[1 0];
Gc=tf(num1,den1);

sysT=series(G,Gc);
sysF=feedback(sysT,1);
step(sysF)

%rltool(G)
rltool(sysT)
```

Anexo. Código de Matlab para simular la planta con la Función de transferencia estimada y las constantes del controlador obtenidas previamente.

Conclusiones.

Como primera consideración, el montaje de la planta se hizo de manera conjunta al interior de la caja, como se pensó en un inicio, pero cuando se empezaron a realizar las pruebas del funcionamiento, se pudo observar que el circuito de potencia generaba interferencia en la lectura del LM35 y las señales de la pantalla LCD, por lo tanto, se modificó el montaje, dividiendo los circuitos de potencia (donde se suministra la corriente alterna AC) y la configuración que permite leer la temperatura con el circuito necesario para la LCD (donde se alimenta con corriente continua DC); de esta manera evitamos compartir la misma protoboard con AC y DC eliminando las posibles interferencias en la circuitería.

Para la identificación del modelo de la planta se necesitó tomar valores con intervalos más cortos ya que un sistema de control de temperatura trabaja con tiempos prolongados y por esto es sistema no se estabiliza rápidamente, por lo tanto, al tomar valores en intervalos más cortos podemos generar una función más precisa para que la planta reaccione de manera más rápida.

Las gráficas que se generan al momento de estimar el modelo de la planta, presentan muchas oscilaciones debido a que los instrumentos no son de precisión, como por ejemplo la variable set point que se modifica a través de un potenciómetro, ante cualquier movimiento genera una señal que se ve como un disturbio en la planta. Por otra parte, el sensor genera una buena lectura, es decir con un error bajo, pero ya que es un elemento muy básico también presenta interferencias ante alguna corriente parasita o algún movimiento.

Durante la implementación del controlador, observamos que el sistema se podría controlar mediante dos tipos de lenguaje; el primero gracias a la librería PID de Arduino con la que se pudo generar un código (anexo al correo) que donde se pueden variar las constantes del controlador PID Kp, Ki y Kd. Por otra parte gracias a la investigación que hicimos en internet pudimos observar que también se podía implementar un código en Simulink, el cual permite hacer la vinculación de la tarjeta Arduino con el ordenador y suministrar dicho código para hacer el control de la planta, al igual que el código en la interfaz de Arduino se modifican las constantes del PID para sintonizar la planta pero con la ventaja de que se puede observar en tiempo real el comportamiento de las variables, Set point, esfuerzo de control, erro, PWM del motor y la bombilla.

A pesar de que la estimación del modelo no alcanzo el 80%, el sistema ya compensado funciona correctamente y con una eficiencia muy buena, respondiendo de manera rápida ante una variación del set point y manteniendo la temperatura, activando la bombilla como el motor para regular de manera eficiente la lectura del sensor.