Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

espol Facultad de Ingeniería en Electricidad y Computación

PROYECTOS DE SISTEMAS DE CONTROL EMBEBIDOS

ADQUISICION Y MONITOREO DE TEMPERATURA UTILIZANDO TSC-LAB Y LA PLATAFORMA THINGSPEAK

DESCRIPCION

En nuestro proyecto, se va a aplicar el método SISO que significa una entrada y una salida.

Se realizó el monitoreo de temperatura activando los actuadores 1 y 2 durante 4 horas, se adquirieron dichos datos y se guardaron en un archivo .csv. El tiempo de muestreo fue de 1 segundo. Para esto utilizamos el hardware TSC-LAB implementado con un código IDE de Arduino y por medio del software Cool Term el cual sirve para visualizar y guardar los datos como archivo .txt.

Dicho archivo se llevó a Matlab, para la identificación por medio del System Identification (SI) toolbox. Se escogió el modelo bj22221 y se lo compara la planta real con el modelo identificado por medio del Simulink y la TSC LAB.

Se realiza el diseño del controlador PID en tiempo continuo por medio del PID Tuner de Matlab y se lo comprueba en Matlab. Mediante el código IDE se establece que la lectura leída de los dos sensores de temperatura, será promediada, para manejar un solo dato. Mediante el Simulink se establece en set point y realiza la lectura de la planta mediante puerto serial y se verificará que el controlador intentará alcanzar dicha temperatura.

Luego los datos se los envía por medio del ThinkSpeak para visualizarlo por medio del protocolo http.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

espol Facultad de Ingeniería en Electricidad y Computación

METODOLOGIA

1. Toma de datos

Se realiza la carga del código de la práctica 5 y por medio del comando case_4 se empieza a adquirir datos y a grabarlos por medio del Cool Term. Ver figura 1.

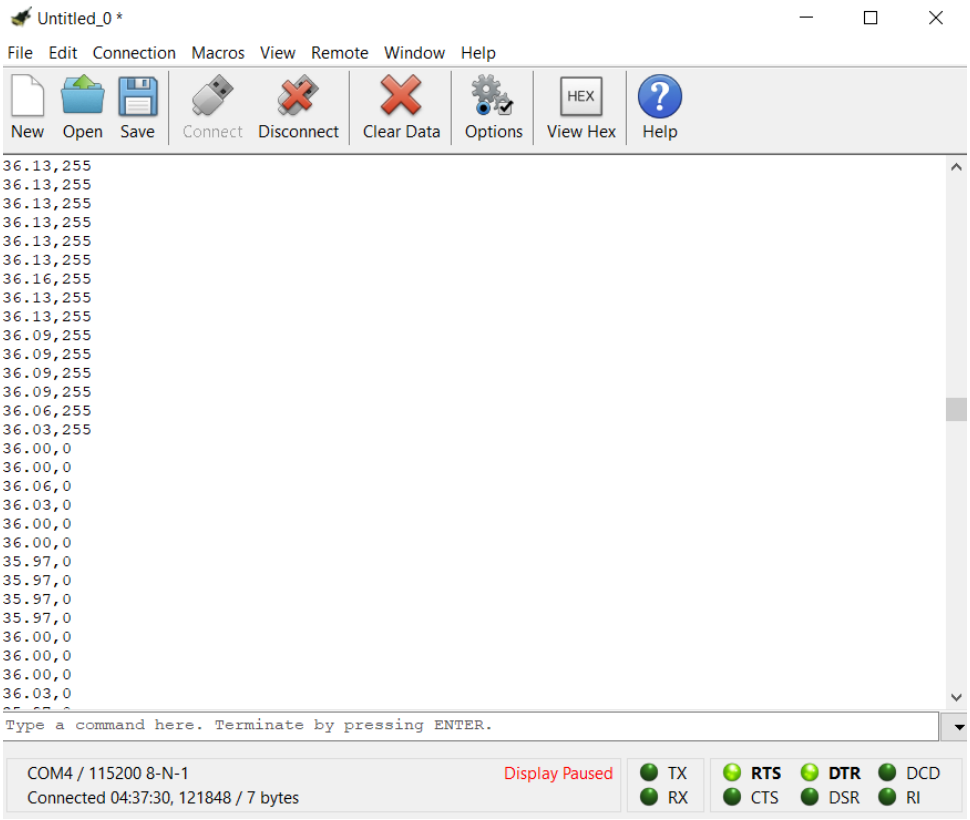

Figura 1. Adquisición y grabación de datos por medio del Cool Term.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

Figura 1. La TSC LAB conectada

Se estableció una variable llamada promedio tipo float, iniciada con valor cero. Ver código.

```
//set parameters
int period=24; //3édium period
int freq=100;  // sampling time
float promedio=0;
```

Como se activan dos actuadores, se establece el valor promedio, para obtener una salida y luego se imprime la variable promedio. Ver código.

```
promedio=(temp1+temp2)/2;
    Serial.print(promedio);
```

Adicional, cada entrada se multiplicó por 255 para tener una entrada escalón entre 255 (nivel alto) y 0 (nivel bajo). Ver código y figura 2. El código completo está en los Anexos.

```
Serial.println(t1*255);
```

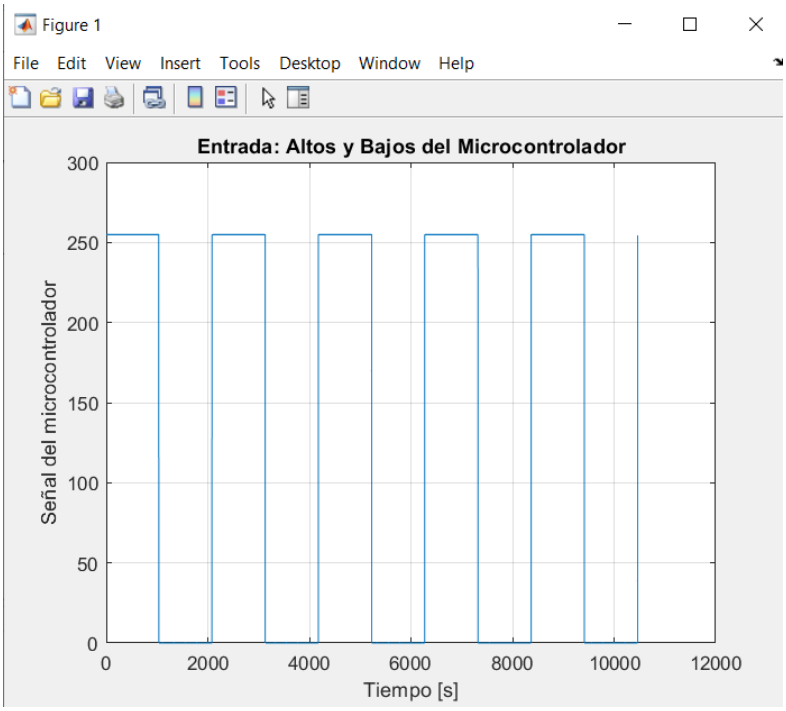

Figura 2. Entrada tipo escalón con valor máximo de 255 (nivel alto).

La toma de datos fue por espacio de 4 horas. Ver figura 3.

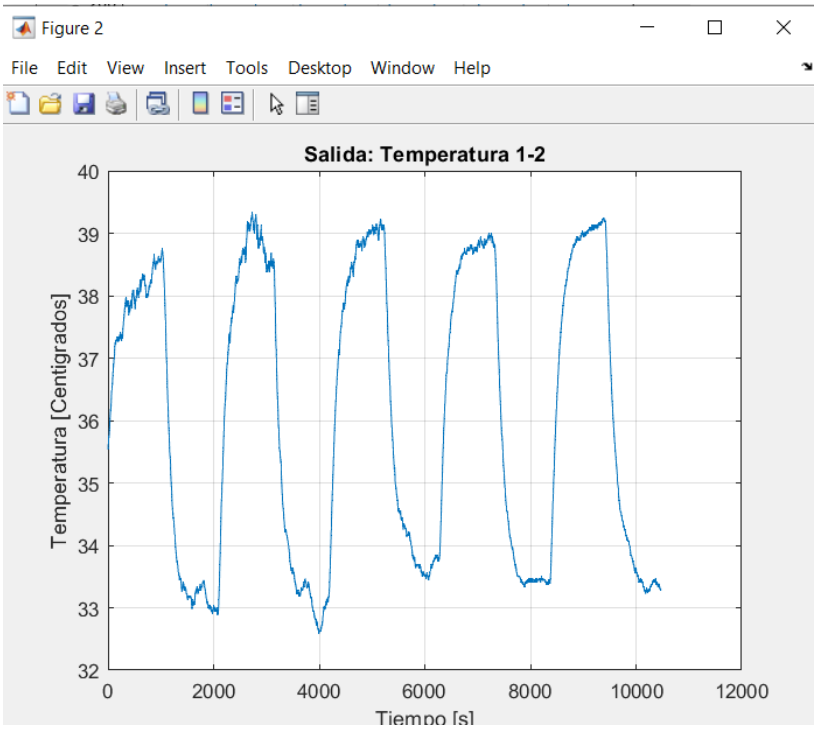

Figura 3. Datos adquiridos por espacio de 4 horas en lazo abierto.

2. Identificación de la planta y comparación con la planta real

Empezamos el proceso de identificación mediante la herramienta SI de Matlab. Para importar lo datos elegimos Time Domain, colocamos IN y OUT en Workspace Variable y en Sample time colocamos uno (1). Ver figura 4. Luego elegimos Import.

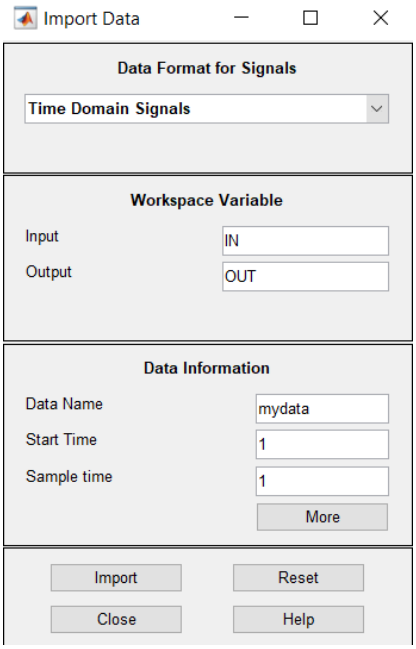

Figura 4. Seteo de las variables de entrada, salida y sample time.

Aparecen los datos en SI como mydata. Ver figura 5.

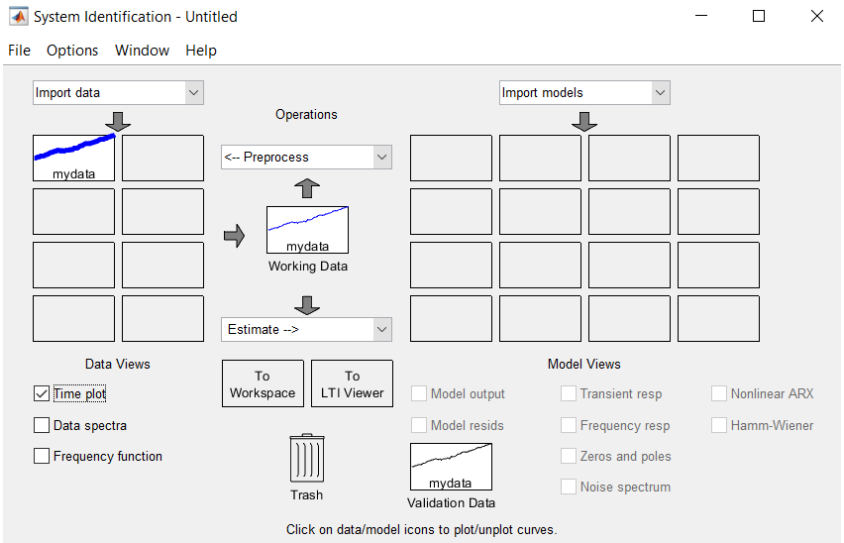

Figura 5. Los datos aparecen como mydata después de elegir Import.

Eligiendo Time plot nos aparecen todos los datos adquiridos y la entrada escalón. Ver figura 6.

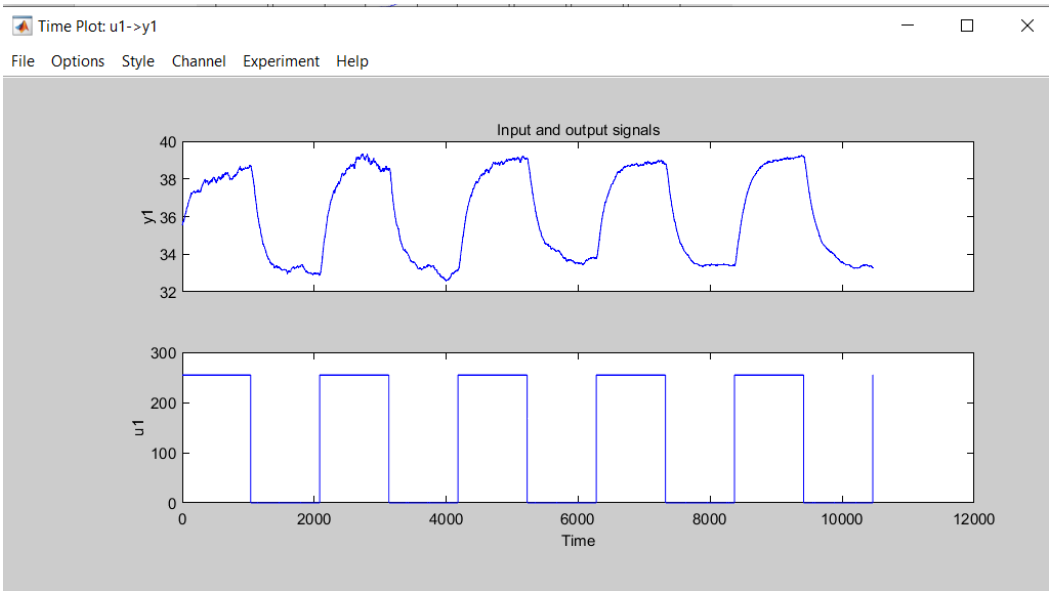

Figura 6. Datos adquiridos como y1 y entrada escalón como u1

En Operations del SI, seleccionamos los rangos de los datos Select Range, 70% para train con el nombre traingacn y 30% para validar con el nombre validargacn. Ver figura 7.

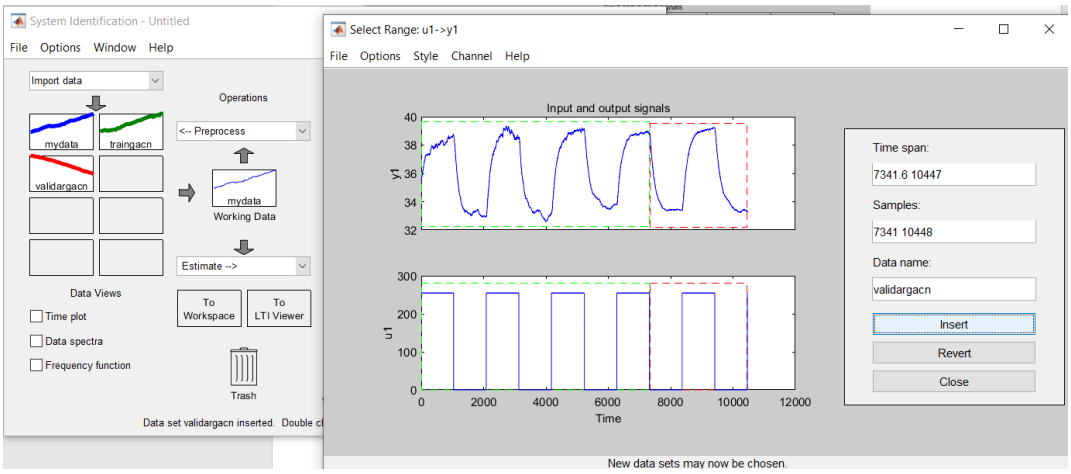

Figura 7. Selección de datos para el train y para validar.

Los datos del 70% y 30%, los trasladamos desde del Data Views hacia el Working Data y Validation Data respectivamente. Ver figura 8.

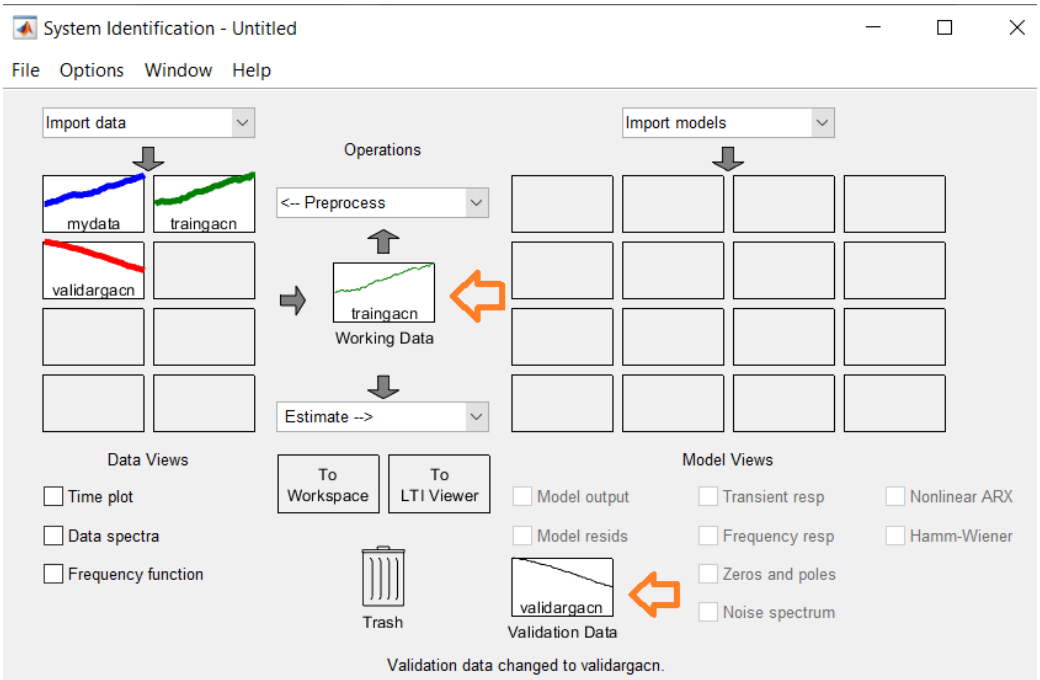

Figura 8. Los datos del 70% y 30% se trasladan a los casilleros Working data y validation data.

Ahora, con los datos en Working data, procedemos a la estimación, elegimos el método Polynomial Models. Se seleccionan los modelos Armax 2221, BJ22221, ARX441 y ARX1016. Ver figura 9.

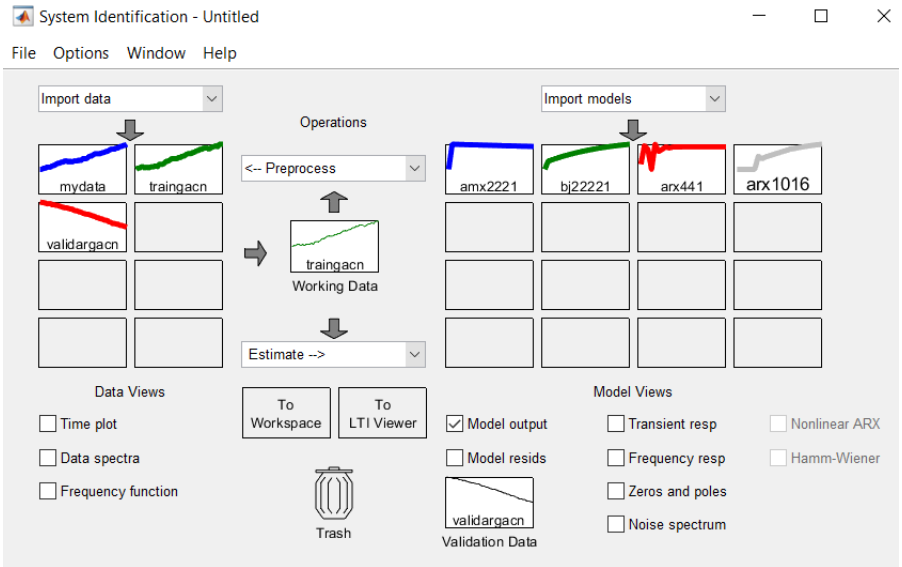

Figura 9. Los cuatro modelos escogidos.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

Cabe indicar que para el modelo ARX1016 se aplicó el ARX y se aplicó Orden Selection, obteniendo la figura 10. Se escogió la barra azul.

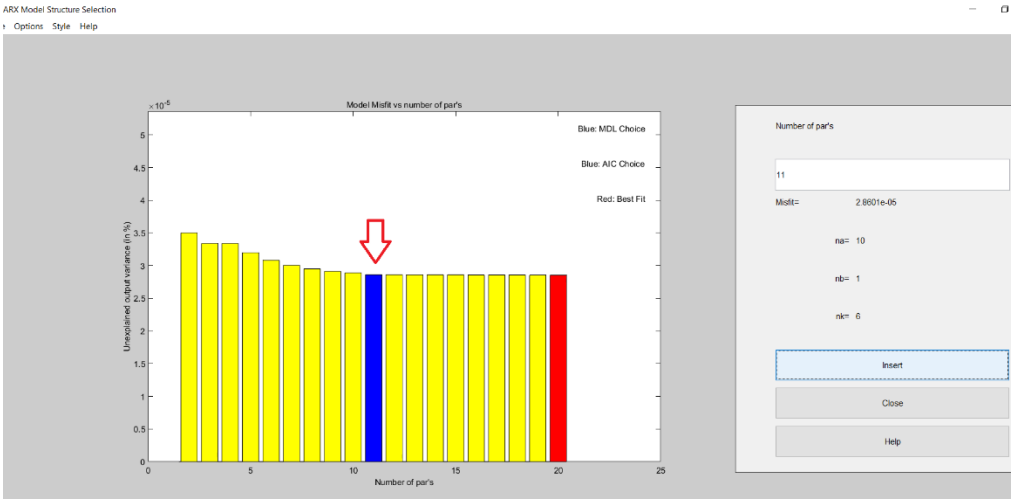

Figura 10. Modelo ARX aplicando Orden Selection.

Aplicando el Model Output, obtenemos los Best Fits de los 4 modelos. Ver figura 11. Observamos que los modelos arx441 y arx1016 tienen valores muy bajos por lo que se descartan. Los mejores modelos son el bj22221 con 92.13 y el armax2221 con 74.83. Entre los dos modelos, se va a escoger el mejor.

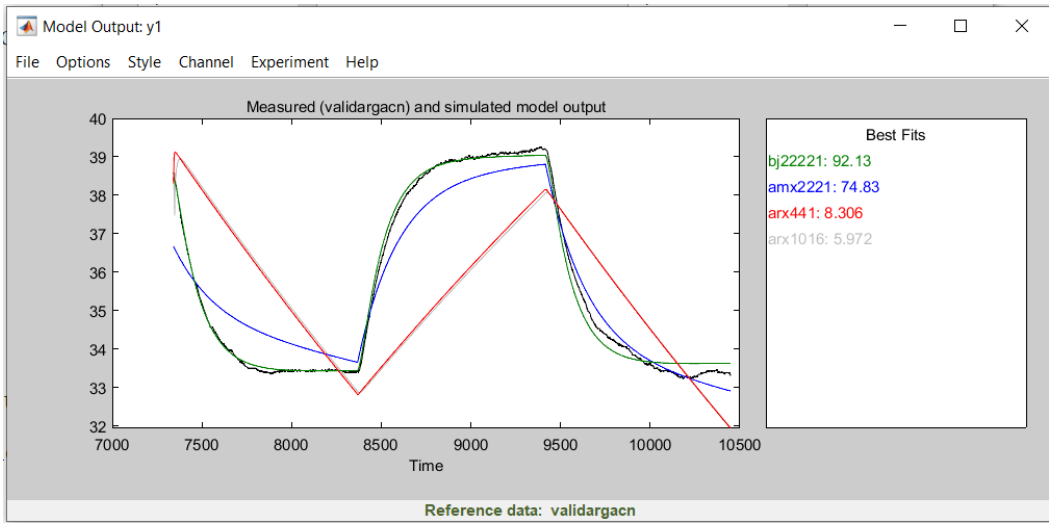

Figura 11. Los cuatros modelos con el Best Fits.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

Llevamos los dos modelos escogidos, bj22221 y amx2221 al Workspace y aplicamos el siguiente código de Matlab para obtener la función de transferencia de cada modelo:

```
G=tf(d2c(bj22221)); %<---Modelo con mejor FIT
num=cell2mat(G.numerator);
den=cell2mat(G.denominator);
FTemp12=tf(num,den)
```

Y aplicamos también, cambiando solo la línea

```
G=tf(d2c(amx2221)); %<---Modelo con mejor FIT
```

Luego obtenemos las siguientes funciones de transferencia:

FTemp12 =

$$\frac{3.064e\text{-}05\ s + 6.743e\text{-}06}{s^2 + 0.04865\ s + 0.0003119}$$

(Del modelo bj22221-Planta 1)

FTemp12 =

$$\frac{8.984e\text{-}05\ s + 2.566e\text{-}08}{s^2 + 0.004405\ s + 1.368e\text{-}07}$$

(Del modelo amx2221-Planta 2)

En la figura 12, se muestran las dos funciones (modelos), obtenidas del Matlab. Se ha obtenido la Planta 1 y Planta 2. Se va a trabajar con la Planta 1.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

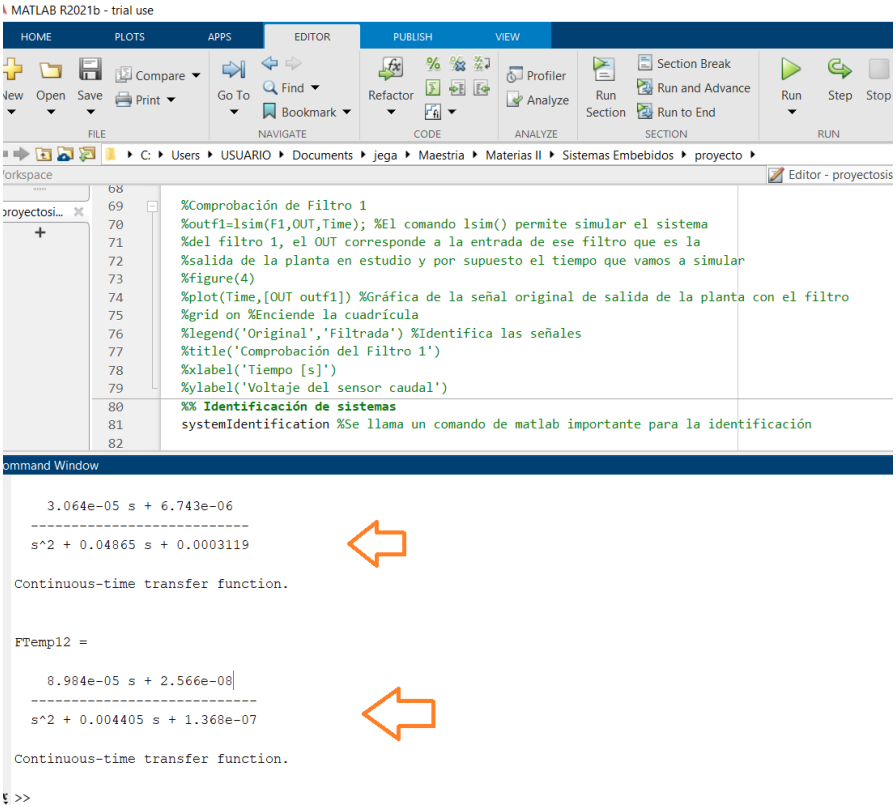

Figura 12. Las dos funciones de transferencia, modelo bj22221 y amx2221.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

Después de identificar la planta con el modelo bj22221, se debe comprobar la similitud entre la planta real con la toma de temperatura en el TSCLAB y el modelo identificado por medio de comunicación serial. Para esto, utilizamos el diagrama de bloques sugerido en Simulink. Ver figura 13. Se eligió el case_4.

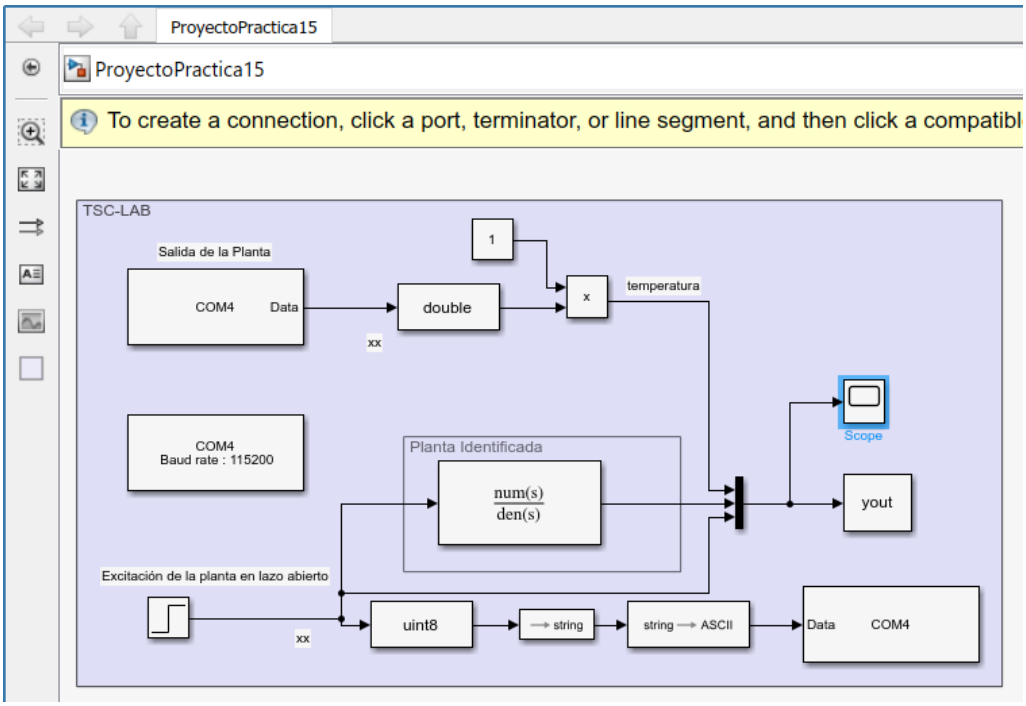

Figura 13. Diagrama de bloques sugerido para comparar la planta y la toma de lectura de temperatura.

En la figura 14, observamos los datos de temperatura que se están leyendo, la función step seteado en 40 y la planta que prácticamente es un línea recta con valor de 2.07.

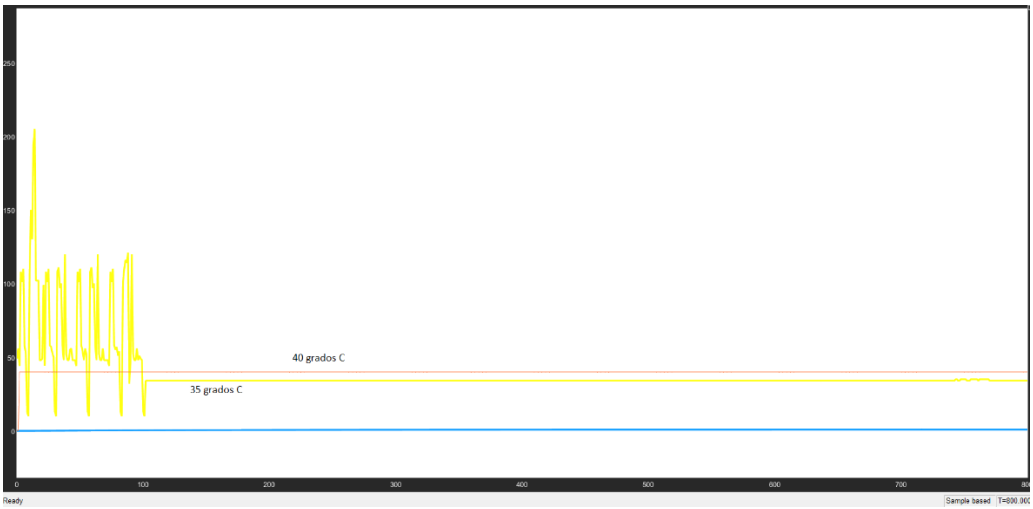

Figura 14. Gráfica de la Planta, la señal pulso continuo y el temperatura tomada con case_4

Se observa que prácticamente en 120 s, la temperatura se estabiliza debido a la planta sugerida. Ver figura 15. En la siguiente figura, vemos teóricamente el Modelo vs Salida y la entrada, se observa un error debido a que está sin controlador.

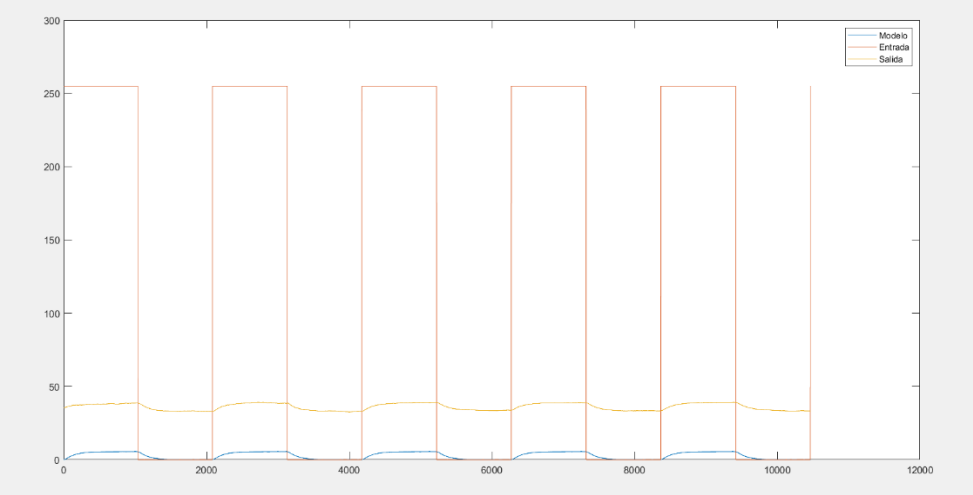

Figura 15. Gráfica de Modelo, la señal pulso continuo y salida

3. Diseño del controlador PID

En APPS de Matlab, elegir PID Tuner. Elegimos Plant, se despliega una ventana para elegir la función de transferencia, en este caso la Planta 1 (bj22221). Ver figura 16.

Se elige Import para importarlo desde el Workspace de Matlab. Automáticamente se diseña un controlador para la nueva planta y muestra la gráfica del sistema en lazo cerrado. Ver figura 17. Sólo nos muestra un controlador Proporcional (P) con un Kp=457.9, a partir de los 128 s se estabiliza pero tiene un error, lo que no es recomendable. Un Overshoot del 14.9 %.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

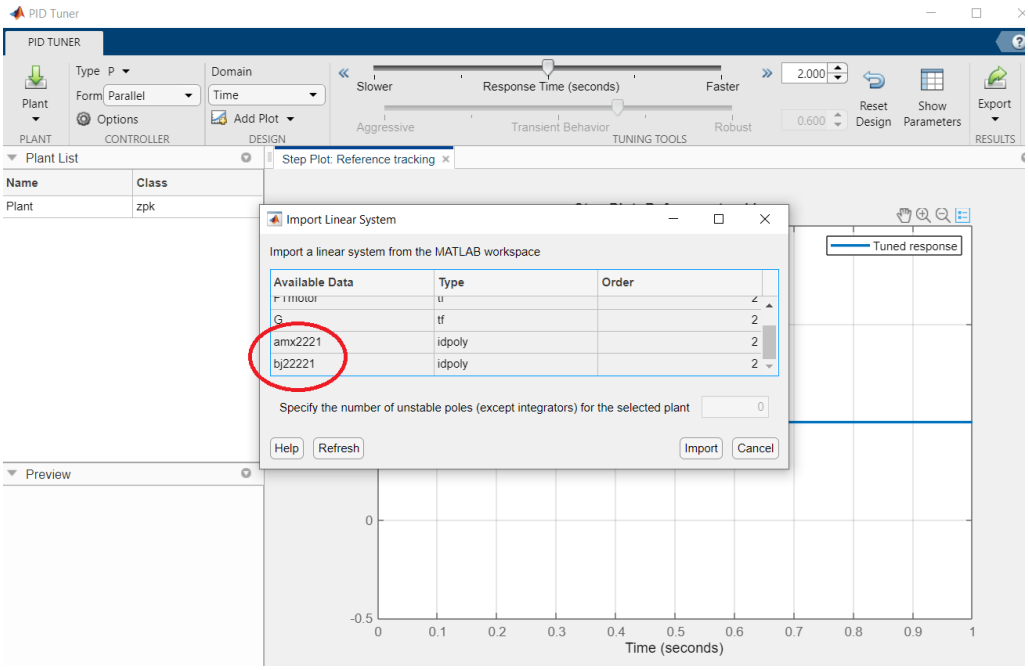

Figura 16. Ingreso de coeficientes del numerador y denominador del modelo bj22221.

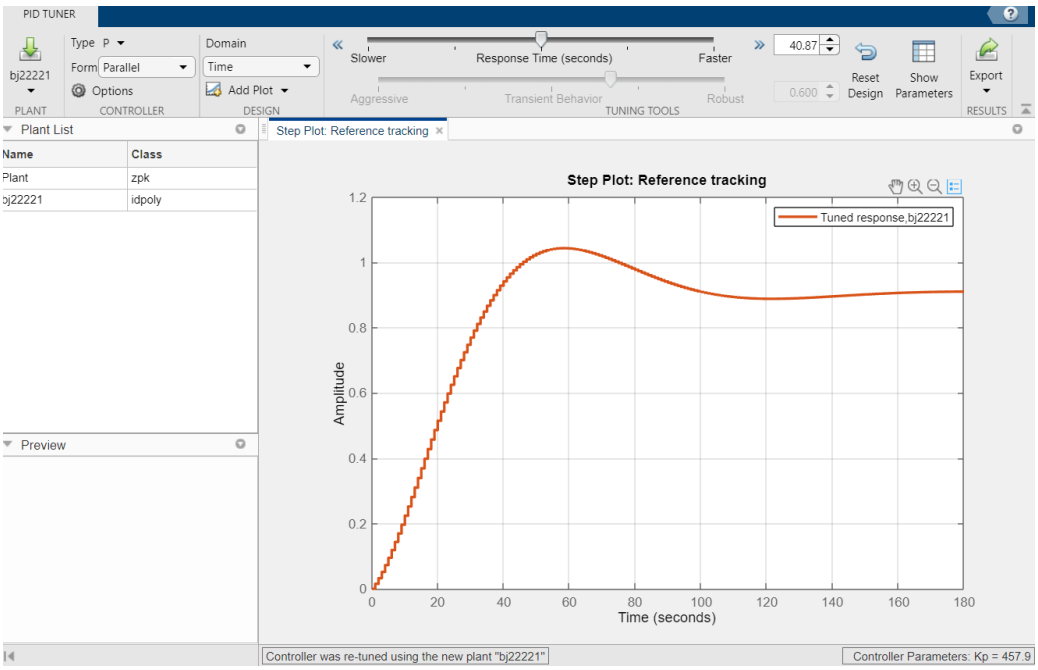

Figura 17. Respuesta en lazo cerrado del modelo bj22221 aplicando PID Tuner, controlador tipo P con sólo ganancia proporcional Kp

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

Volvemos a Type y elegimos el controlador PID y obtenemos los siguientes parámetros:

Kp=461.171 Ki=3.934 Kd=2417.878

Ver figura 18. Observamos que el sistema se estabiliza en 109 s, el rise time es de 32 s y el Overshoot bajó a 11.1%. Se elimina el error, por lo tanto, este control es mucho mejor, se estabiliza más rápido, no hay error y tiene menos Overshoot.

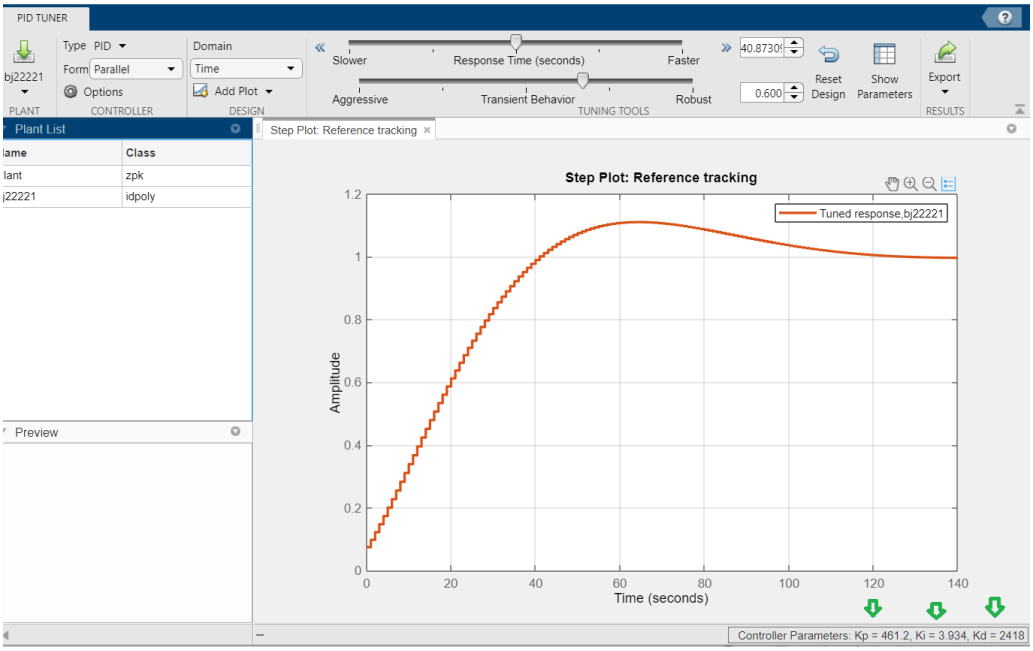

Figura 18. Respuesta en lazo cerrado del modelo bj22221 aplicando PID Tuner con ganancia Kp, Ki, Kd.

Se realizó una prueba con el modelo amx2221 y se obtuvo un controlador con las siguientes características:

Kp = 6.01 Ki = 0.00062 Kd = 3.006
Rise time = 2.61 e+04 s Settling time = 9 e+0.4 s Overshoot = 11%

Se observa que es un controlador muy lento. Ver figura 19.

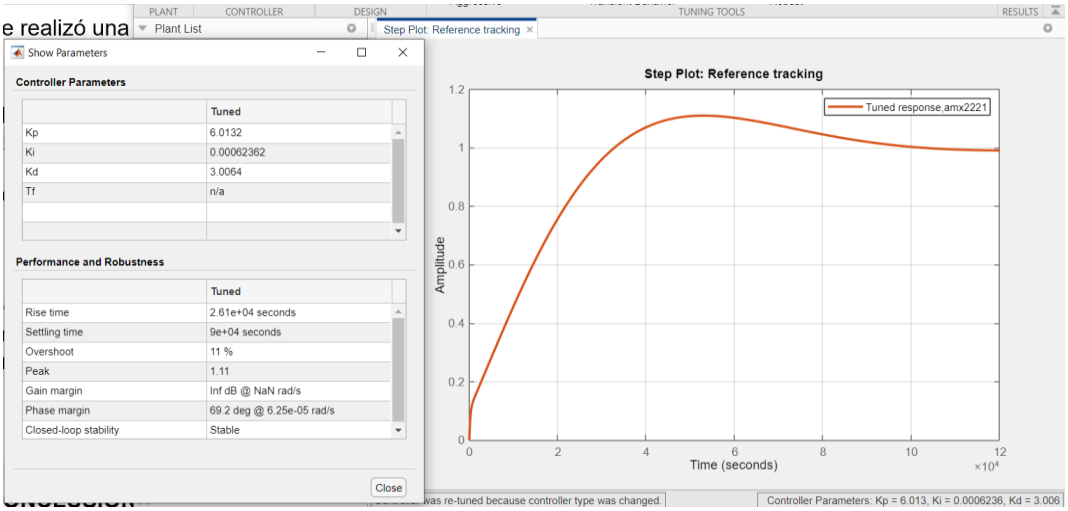

Figura 19. Respuesta en lazo cerrado del modelo amx2221 aplicando PID Tuner con ganancia Kp, Ki, Kd.

El controlador diseñado por medio del modelo bj22221 se lo exporta al Workspace.

Procedemos ahora a realizar una simulación con la ayuda del Simulink. Para esto, realizamos el diagrama de bloques de la figura 20.

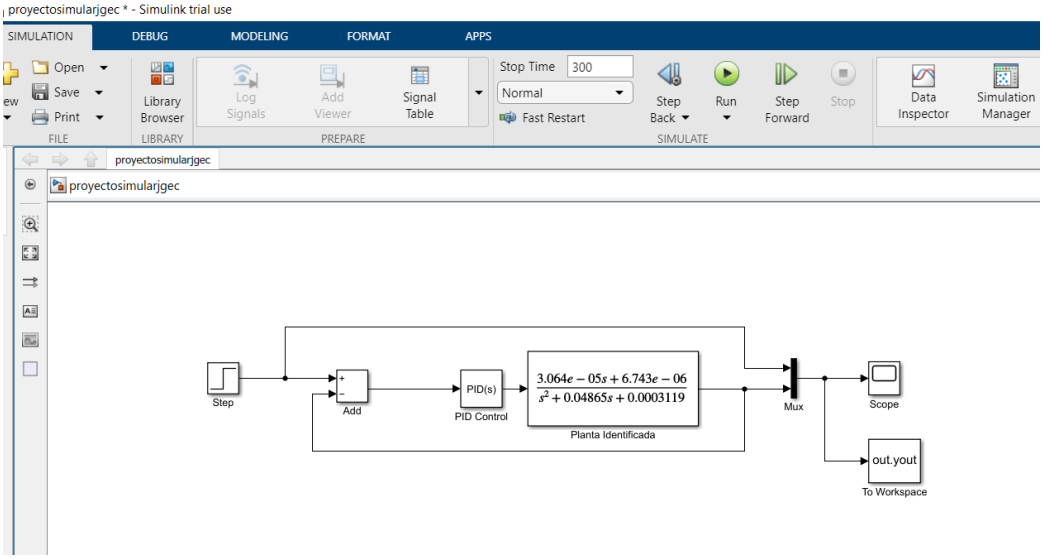

Figura 20. Diagrama de bloques para simular el control PID

Se ingresan los datos del controlador PID según del PID Tuner y los coeficientes de la planta obtenida según el modelo bj22221. Ver figura 21.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

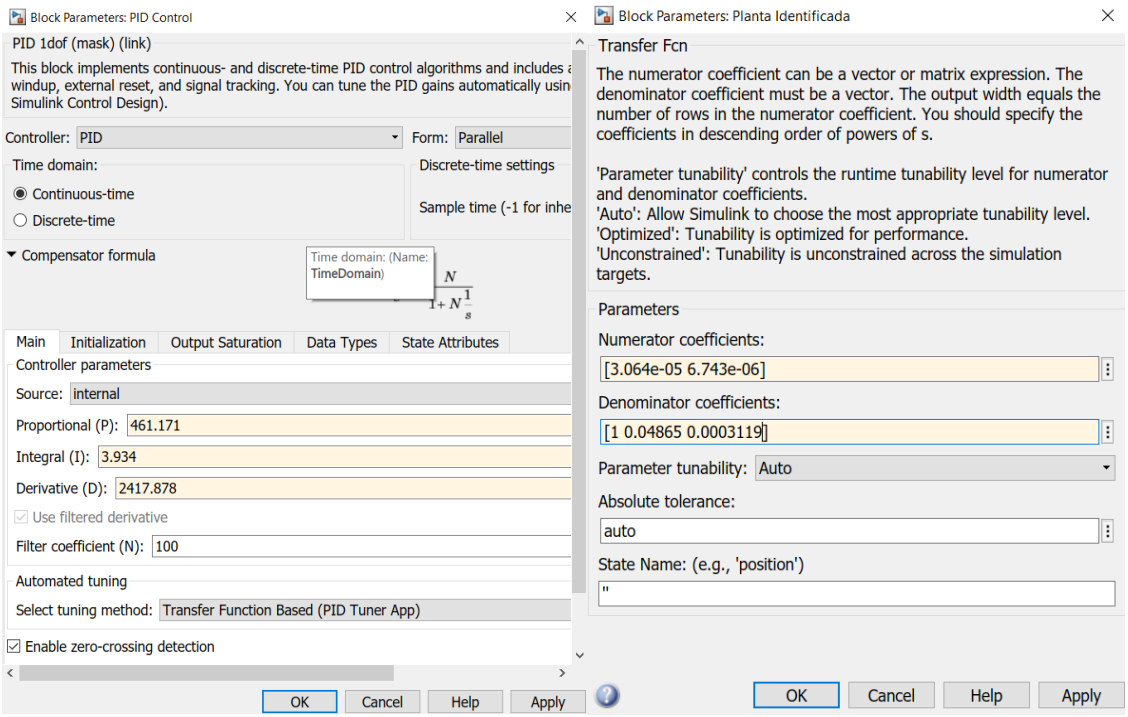

Figura 21. Ingreso de las constantes Kp, Ki, Kd en el bloque PID y de los coeficientes de la planta identificada.

Considerar que la función step, setear tal como viene originalmente, es decir, el Step time es uno (1), Initial value en cero (0), Final value en uno (1).

Procedemos a realizar la exportación de la gráfica del Scope del Simulink al Workspace de Matlab mediante Configuration Properties, elegimos Logging, en Variable name colocamos el nombre que se le quiera asignar, en este caso Proyectojgec y en Save Format elegimos Array. Ver figura 22

Luego de esto, en PREPARE del Simulink, elegir Model Settings, luego Data Import/Export y deshabilitar Single Simulation output. Ver figura 23.

Finalmente obtenemos la gráfica en el ambiente Matlab, luego de aplicar un código, que está en el Anexo. Ver figura 24. Se observa la función de entrada escalón Setpoint y la gráfica del Ajuste PID.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

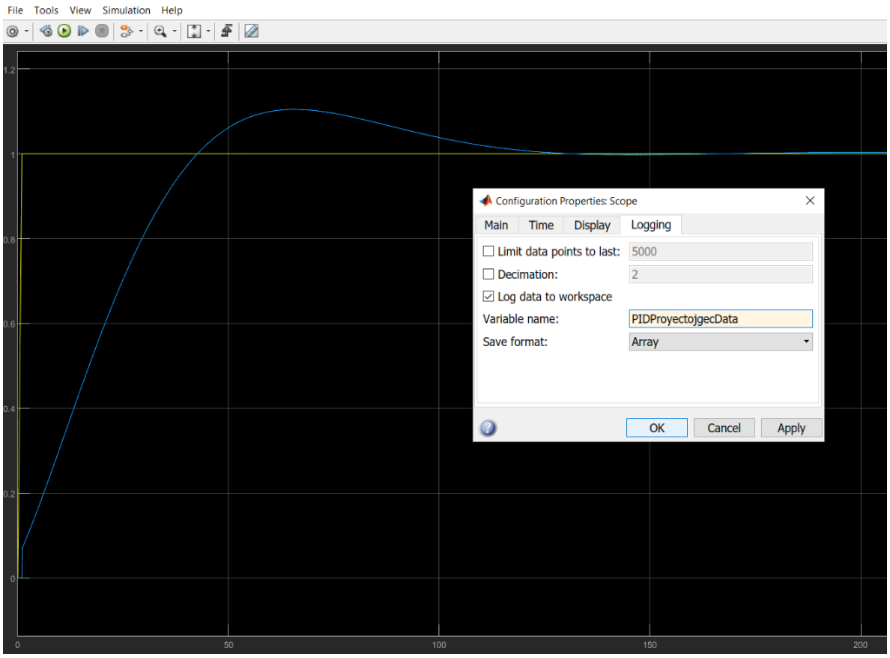

Figura 22. Seteo del Scope para llevar la gráfica al Workspace de Matlab.

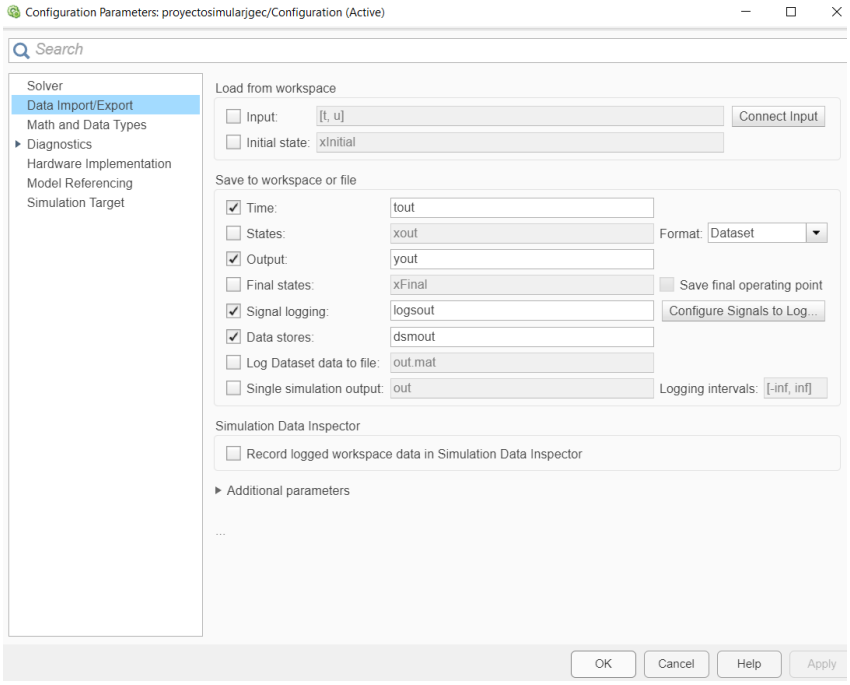

Figura 23. Seteo del Model Settings para exportar la figura del Scope al Matlab.

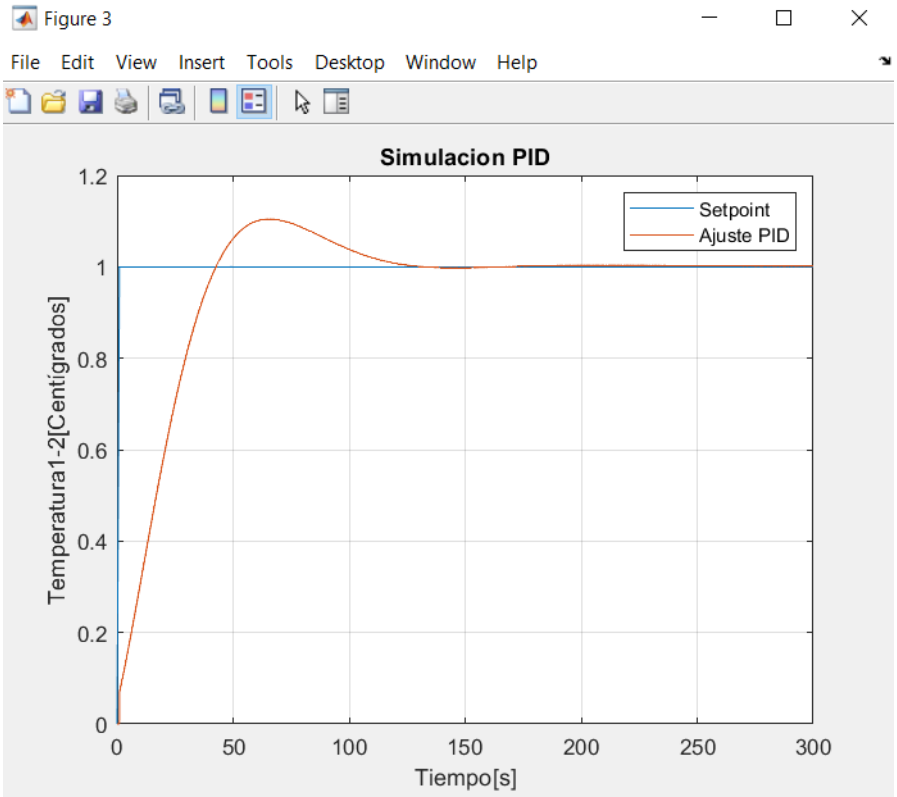

Figura 24. Gráfica del controlador PID del Simulink en Matlab.

4. Comprobación del controlador en Matlab

En esta sección, se va a probar el controlador PID, para esto, aplicaremos el siguiente diagrama de bloque de Simulink (Ver figura 25), el TSC-LAB enviará datos según el case_4, por medio de la comunicación serial, realizaremos la comprobación mediante las gráficas obtenidas en Simulink.

Los siguiente parámetros deben ingresarse al bloque PID:

Kp=461.171 Ki=3.934 Kd=2417.878

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

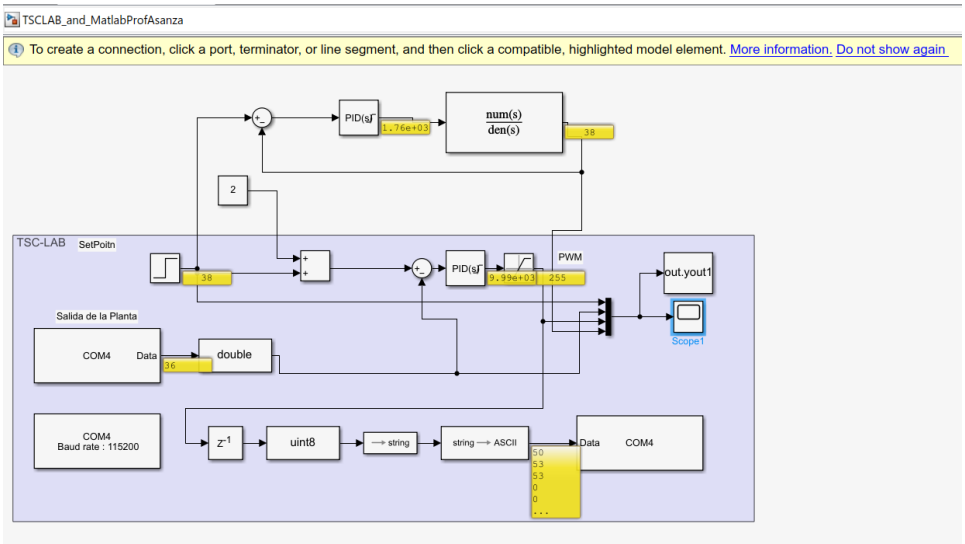

Figura 25. Diagrama de bloque del Simulink para probar el controlador PID.

En el diagrama observamos que los datos de la planta ingresan al Simulink por el puerto serial, se aplica el Set Point, el sumador, el bloque PID hallado en la sección anterior. Utilizamos un MUX de 4 a 1 para verificar las señales del Set Point, la Planta y la planta teórica.

Se realizan varias pruebas, para verificar el control.

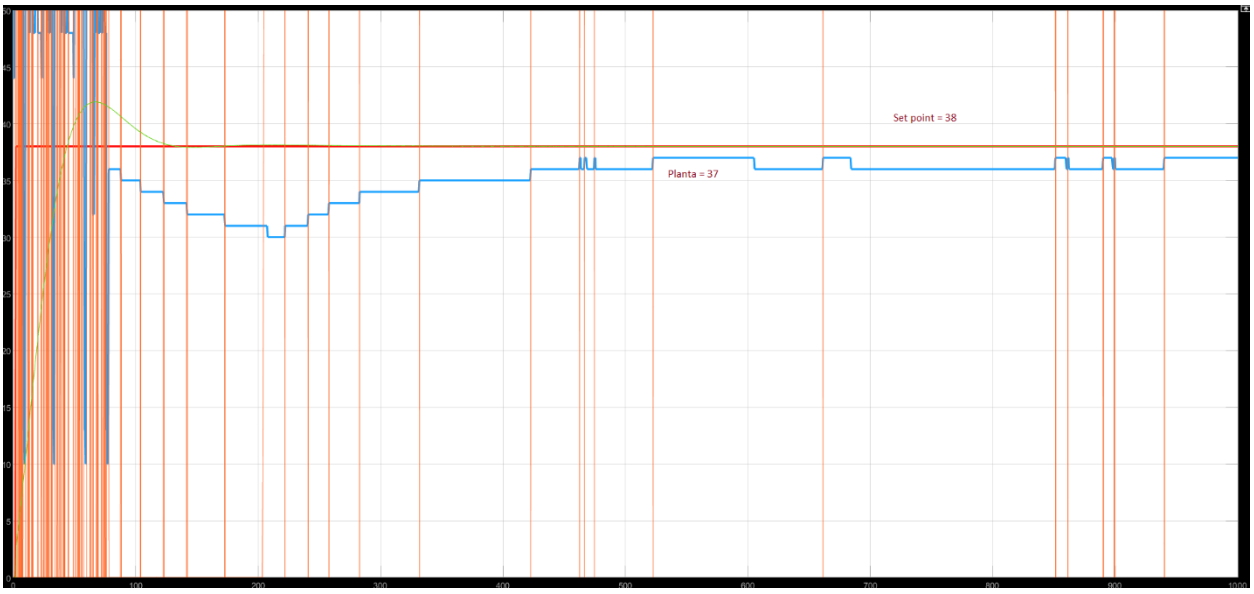

Figura 26. Gráfica de Set point vs datos de la planta.

Tabla 1. Pruebas del PID con diferentes Set point y sumador

Set Point (Grados C)	Planta (grados C)	Sumando (grados C)
38	37	2
38	36	2
38	35	2
40	39	10
40	38	5
40	38	12

Fuente: Propia

En la tabla de observan los datos de varias pruebas, con el Set point indicado y la temperatura de la planta.

5. Envío de datos por la plataforma ThinkSpeak

En esta parte, necesitamos enviar los valores sensados de temperatura y el estado de los transistores. Ver figura 29.

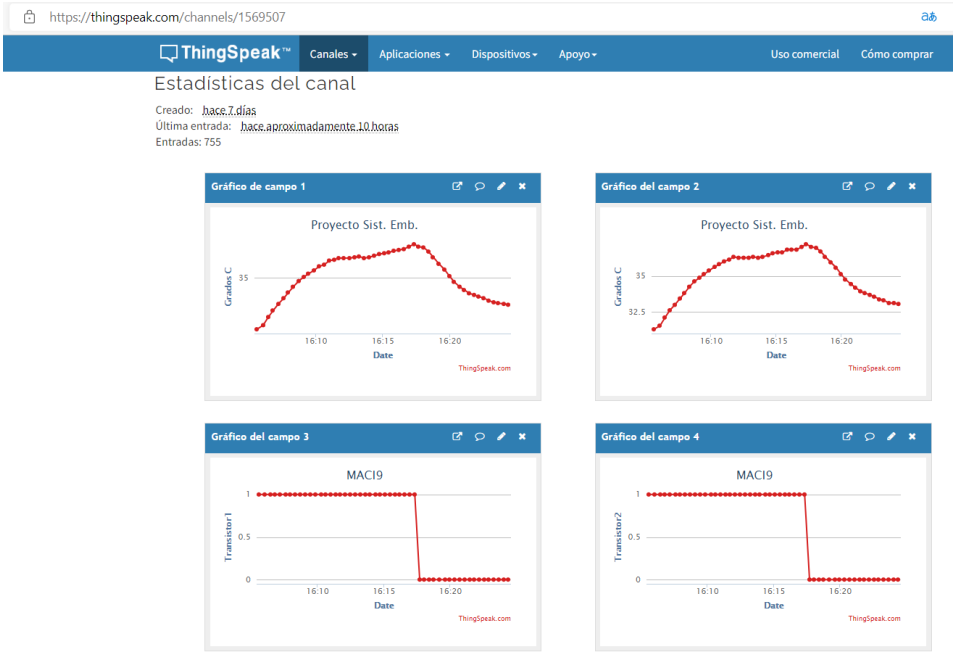

Figura 29. Visualización de datos de temperatura 1 y 2.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

6. RESULTADOS

CONCLUSIONES

1. La conexión serial entre el IDE Arduino y el Simulink y mandar a Run este último, requiere de muchos recursos de máquina y también de tiempo del ploteo. Para un tiempo de 200, se demora aproximadamente 10 minutos en sacar la gráfica en el Scope.
2. Para elegir un buen modelo, se recomienda utilizar la mayor cantidad de datos, en nuestro caso, utilizamos 10462 datos tomados durante cuatro (4) horas.
3. La temperatura en lazo abierto siempre está en 35 grados C pese a que se cambia el seteo de la función escalón. En algunos casos empezó a cambiar la temperatura casi al final del barrido, al parecer se requiere más tiempo de barrido.
4. Se eligió el modelo bj22221 del amx2221 debido a que respondió mejor ante el control PID, fue más rápido, se estabilizaba en 120 s comparado con $12x10^4$ s
5. En el cálculo del controlador, se prefirió el PID sobre el P debido a que este último presentaba un error de 0.1 aproximadamente.
6. Se logra la comunicación entre la TSC LAB y la plataforma ThingSpeak por medio de un código IDE de Arduino. Se realizó la conexión a la red wifi doméstica. Se logró visualizar los datos de temperatura y el estado de los transistores 1 y 2 en forma remota por medio del ID asignado por la plataforma.
7. Con la ayuda del Profesor Víctor Asanza se logró modificar el diagrama de bloques de Simulink y el código de IDE, logrando finalmente la comunicación serial y la recepción y envío de datos de temperatura. Esta fue la parte más difícil del presente proyecto.
8. Se realizaron varias pruebas al control PID y generalmente quedó una diferencia de 1 o 2 grados entre el Set point y la temperatura medida. No se observa que el sumador afecte a los resultados. Colocando 2 o 10 de sumador, igual hay una diferencia de 1 grados. Nunca se obtuvo exactamente el Set Point.
9. Se logra la comunicación ThingSpeak pero el control PID no funciona correctamente. Solamente al inicio del Run del Simulink, la temperatura sube por espacio de 7 minutos aproximadamente, luego de esto, a pesar de que el Simulink sigue en Run, el PID deja de funcionar.
10. Este proyecto fue muy interesante porque se aprendió desde la recolección de datos en forma real hasta realizar la comunicación entre Simulink y la tarjeta TSC LAB con la ayuda del código IDE de Arduino. Se comprobó que el funcionamiento del control PID. Una desventaja sería que actuaba muy lentamente, en algunas pruebas que se realizó, el control actuaba después de 8 o 9 minutos.

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez

Facultad de Ingeniería en Electricidad y Computación

RECOMENDACIONES

1. Para que el Simulink reconozca el puerto serial donde está conectada la TSC LAB, primero conectarla al puerto USB y luego abrir Matlab y Simulink.
2. Cuando se realice la conexión entre la TSC LAB y el Simulink, asegurarse de dejar cerrado el COM utilizado en el IDE Arduino, es decir, cerrar Monitor Serie o Serial Plotter.
3. Mientras se corre el Simulink, no se puede visualizar datos en el ThingSpeak.
4. Para la toma de datos, se recomienda alimentar la TSC LAB por medio de su cargador.
5. Mandar a Run el Simulink si quiere obtener el PID junto al ThingSpeak.

7 ANEXOS

Código IDE para adquirir datos, comunicación serial y comunicación con ThingSpeak

```
#include <OneWire.h>
#include <DallasTemperature.h>

//GPIO pin 0 is set as OneWire bus
OneWire ourWire1(0);
//GPIO pin 4 is set as OneWire bus
OneWire ourWire2(4);

//A variable or object is declared for our sensor 1
DallasTemperature sensors1(&ourWire1);
//A variable or object is declared for our sensor 2
DallasTemperature sensors2(&ourWire2);
int dutyCycle = 0;

//initial setting for data acquisition
int dutyCycleInitial = 255;
int dutyCycleFinish = 0;

// Setting PWM properties
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;

//Status of transistors
int t1 = 0;
int t2 = 0;
int minutos = 10;

//set parameters
int period=10; //medium period
//int freq=1000; // sampling time
int motor1Pin1 = 33;
int motor1Pin2 = 25;
int enable1Pin1 = 16;
int enable1Pin2 = 17;
int conteo = 0;
int tempProm = 0;

// WiFi
#include <WiFi.h>
WiFiClient  client;
const char *ssid = "XTRIM_GOMEZ_ALVARADO"; // Enter your WiFi name
const char *password = "independiente";  // Enter WiFi password
```

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez


```
#include "ThingSpeak.h" // always include thingspeak header file after other header files and custom macros

unsigned long myChannelNumber = 1569507;    //your ID
const char * myWriteAPIKey = "OF9QV5EPP0P11D89"; // your API-KEY

void motor( void *pvParameters );
//void enviar( void *pvParameters );

void setup() {
delay(1000);
Serial.begin(115200);
sensors1.begin(); //Sensor 1 starts
sensors2.begin(); //Sensor 2 starts
// sets the pins as outputs:
pinMode(motor1Pin1, OUTPUT);
pinMode(motor1Pin2, OUTPUT);
//pinMode(enable1Pin, OUTPUT);

// configure LED PWM functionalitites
ledcSetup(pwmChannel, 30000, resolution);

// attach the channel to the GPIO to be controlled
ledcAttachPin(enable1Pin1, pwmChannel);
ledcAttachPin(enable1Pin2, pwmChannel);


//transistor 1
pinMode(16, OUTPUT);
//transitor 2
pinMode(17, OUTPUT);

//wifi
  WiFi.mode(WIFI_STA);
  connect_wifi();
  ThingSpeak.begin(client);  // Initialize ThingSpeak

//Serial.println("Choose any case: ");

xTaskCreatePinnedToCore(
motor
, "MotorDC" // Descriptive name of the function (MAX 8 characters)
, 2048 // Size required in STACK memory
, NULL // INITIAL parameter to receive (void *)
, 1 // Priority, priority = 3 (configMAX_PRIORITIES - 1) is the highest, priority = 0 is the lowest.
, NULL // Variable that points to the task (optional)
, 1); // core 1
}

void loop() {
while (1){
//digitalWrite(16, HIGH);
```

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez


```
//for (int i = 1; i <= minutos*60; i++) { // minute=10
sensors1.requestTemperatures();
float temp1 = sensors1.getTempCByIndex(0);
sensors2.requestTemperatures();
float temp2 = sensors2.getTempCByIndex(0);

//Obtain the temperature in ºC of sensor 1

//print to display the temperature change
 tempProm = (temp1+temp2)/2;
//if (conteo == 60){
//conteo = 0;
//tempProm = tempProm/60;
 //Escribimos en Matlab la temperatura promedio
 Serial.write(tempProm);
  //tempProm = 0;
// Serial.print(",");
//Serial.println(t1);
//}
public_ThingSpeak(temp1,temp2,tempProm,t1,t2);
vTaskDelay(999);
//conteo++;
}
}



void connect_wifi(){
  // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, password);  // Connect to WPA/WPA2 network. Change this line if using open or WEP
network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
}

void motor( void *pvParameters ) {
while (1) {

//digitalWrite(motor1Pin1, HIGH);
//digitalWrite(motor1Pin2, LOW);
if (Serial.available())
{
String string = Serial.readStringUntil('\n');
dutyCycle = string.toInt();
ledcWrite(pwmChannel, dutyCycle);
```

```
}
//vTaskDelay(period);
}
}

void public_ThingSpeak(float temp1,float temp2, float tempProm, int trans1,int trans2){
  // set the fields with the values
  ThingSpeak.setField(1, (temp1));
  ThingSpeak.setField(2, (temp2));
  ThingSpeak.setField(3, (tempProm));
  ThingSpeak.setField(4, (trans1));
  ThingSpeak.setField(5, (trans2));




  // write to the ThingSpeak channel
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }



  //delay(20000); // Wait 20 seconds to update the channel again

}
```

CODIGO MATLAB

```
%% Identificación de Sistemas
clc
clear all
close all
Datos=load('proyectofin1jgavec.csv'); %Cargar los datos del excel y guardarlos en la
variable
OUT=Datos(:,1); %Salida del Sistema
IN=Datos(:,2); %Entrada del Sistema (Verificar que este valor este entre 0 y 255,
caso contrario *255)
Time=(0:1:length(Datos)-1)'; %Vector de tiempo
%% Gráfica de los datos
figure(1)
plot(Time,IN); %Se grafica la entrada del sistema
title('Entrada: Altos y Bajos del Microcontrolador')
ylabel('Señal del microcontrolador')
xlabel('Tiempo [s]')
grid on
```

```
figure(2)
plot(Time,OUT); %Se grafica la salida del sistema
title('Salida: Temperatura 1-2')
ylabel('Temperatura [Centigrados]')
xlabel('Tiempo [s]')
grid on

%% Función de Transferencia obtenida (Metodo 3)
G=tf(d2c(bj22221)); %<---Modelo con mejor FIT
num=cell2mat(G.numerator);
den=cell2mat(G.denominator);
FTemp12=tf(num,den)

%% Parámetros Controlador PID obtenido (FT Metodo 1 y 2)
Kp=461.171
Ki=3.934
Kd=2417.878

%% Obtener la gráfica en el Workspace
sim('proyectosimularjgec') %Simulación del bloque Simulink
figure(3)
plot(Proyectojgec(:,1),Proyectojgec(:,2))
hold on
plot(Proyectojgec(:,1),Proyectojgec(:,3))
hold on
grid on
legend('Setpoint','Ajuste PID')
title('Simulacion PID')
xlabel('Tiempo[s]')
ylabel('Temperatura1-2[Centígrados]')
hold off

%% Gráfica planta vs modelo
%sim('proyectosimularjgec') %Simulación del bloque Simulink
%figure(4)
%plot(plantavsmodelo(:,1),plantavsmodelo(:,2))
%hold on
%plot(plantavsmodelo(:,1),plantavsmodelo(:,3))
%hold on
%plot(plantavsmodelo(:,1),plantavsmodelo(:,4))
%grid on
%legend('Entrada','Planta','Modelo')
%title('Planta vs Modelo')
%xlabel('Tiempo[s]')
%ylabel('Temperatura1-2[Centígrados]')
%hold off

%% Modelo vs Salida
Time=(0:1:length(IN)-1)'; %Vector de tiempo
y = lsim(G,IN,Time); % system response
figure(4)
plot(y);
hold on;
```

Nombres:__Julio Ernesto Gómez Assan
Victor Emilio Cedeño Nuñez


```
plot(IN);
%%plot(OUT);
legend('Modelo','Entrada','Salida');
```