Universidad Practica Territorial Aragua
Maestría en Automatización, Control y Robótica

Taller 2

Estudio Modelo de Temperatura de Horno

Diego Nieto

Gustavo Mujica

Con los datos obtenidos experimentalmente, de las medidas de temperatura y tiempo de la maqueta:
- Explique el procedimiento para la obtención de los datos, especificando el uso del PLX-DAT, tiempo de muestreo, señal de entrada, unidades de las señales
- Obtener la curva de reacción del proceso
- Verificar comportamiento en Simulink
- Determinar el modelo matemático de POMTM del proceso

- Determine los parámetros de
- Implementar un control PID-ISA por medio de Ziegler-Nichols y verificar comportamiento en Simulink
- Analizar los resultados

1. El procedimiento de adquision de datos se basa en una comunicación serial entre el arduino Uno y la hoja de datos Excel configurada como macro con la capacidad de recibir buffer de datos y procesarlo de acuerdo a la cabecera de los datos

   El sensor utilizado en arduino es un LM35

   El Arduino Uno tiene la capacidad de trabajar a una velocidad 16Mhz

   El PLX DAQ lo limita la velocidad de comunicación serial y la capacidad de procesamiento de la hoja Excel, por esta razón se le coloca un retardo al arduino para poder procesar la información sin que se cuelgue la pagina

2. Obtener la curva de reacción del proceso
   Se utiliza el método Smitch para determinar las constantes.

   La constante $K = \Delta Y \div \Delta X$

   Donde K = (59-30)/(13.4-0)

   K=2.164

   Obtención de los datos a partir de la curva

   $t_{63} = 214{,}3\ seg$

   $t_{28} = 123{,}1\ seg$

Obtencion de tao

$tao = 1.5(\ t_{63} - t_{28})$

$tao = 1.5(\ 214{,}3\ - 123{,}1)$

$tao = 136{,}8\ seg$

$tao = 2{,}28\ min$

Obtención del tiempo muerto

Tm = c*t1 + d*t2

Tm = 1.5* 214.3 – 0.5*136.5

Tm = 31.05 seg

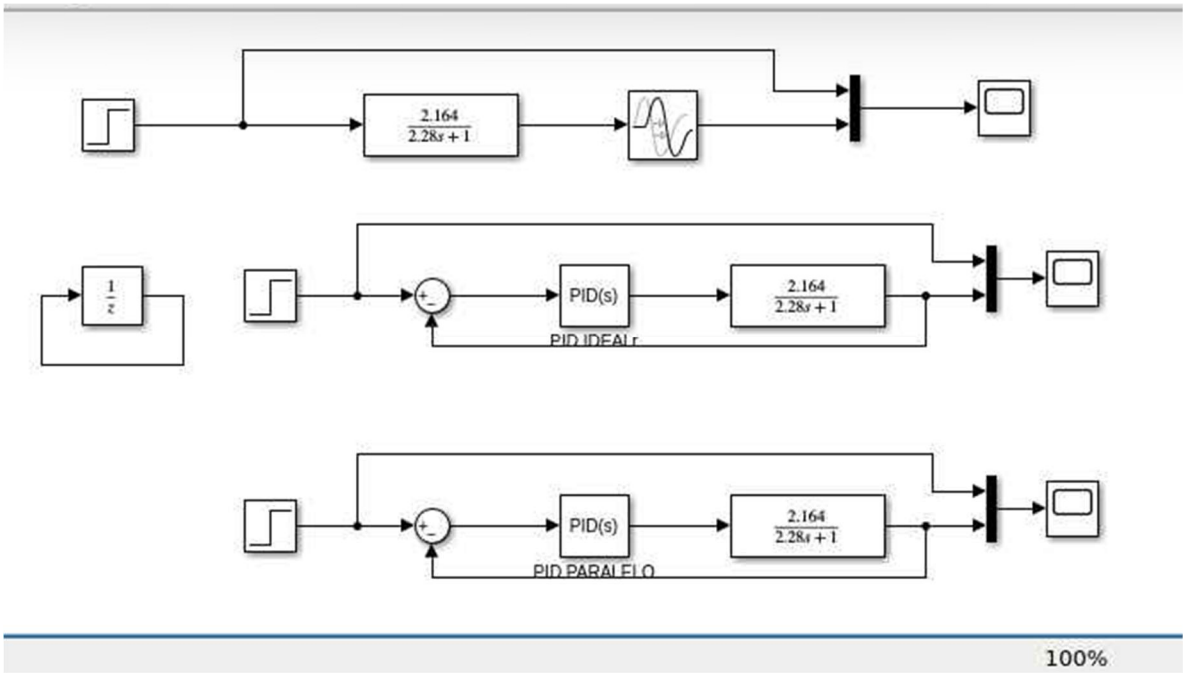
Tm = 0.5175 min

Curva de reacción

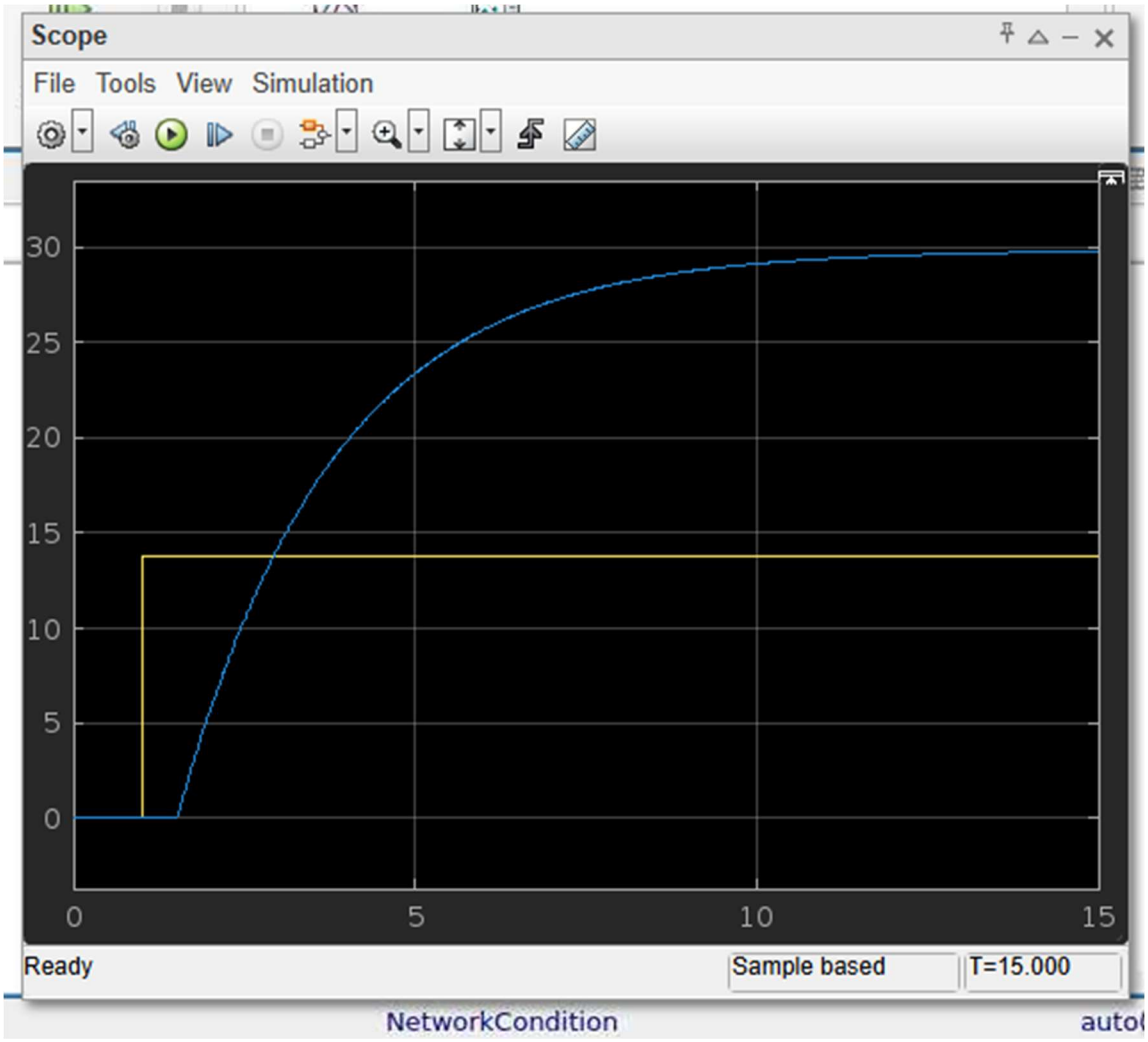
$$G(s) = \frac{2.164}{2.28 * S + 1} * e^{-0.5175 * S}$$

3. Verificar comportamiento en simulink

Se construye el modelo



Para un escalon de 13.8 voltios se tiene la curva de reacción, siendo el resultado esperado



4. Verificando el modelo del proceso

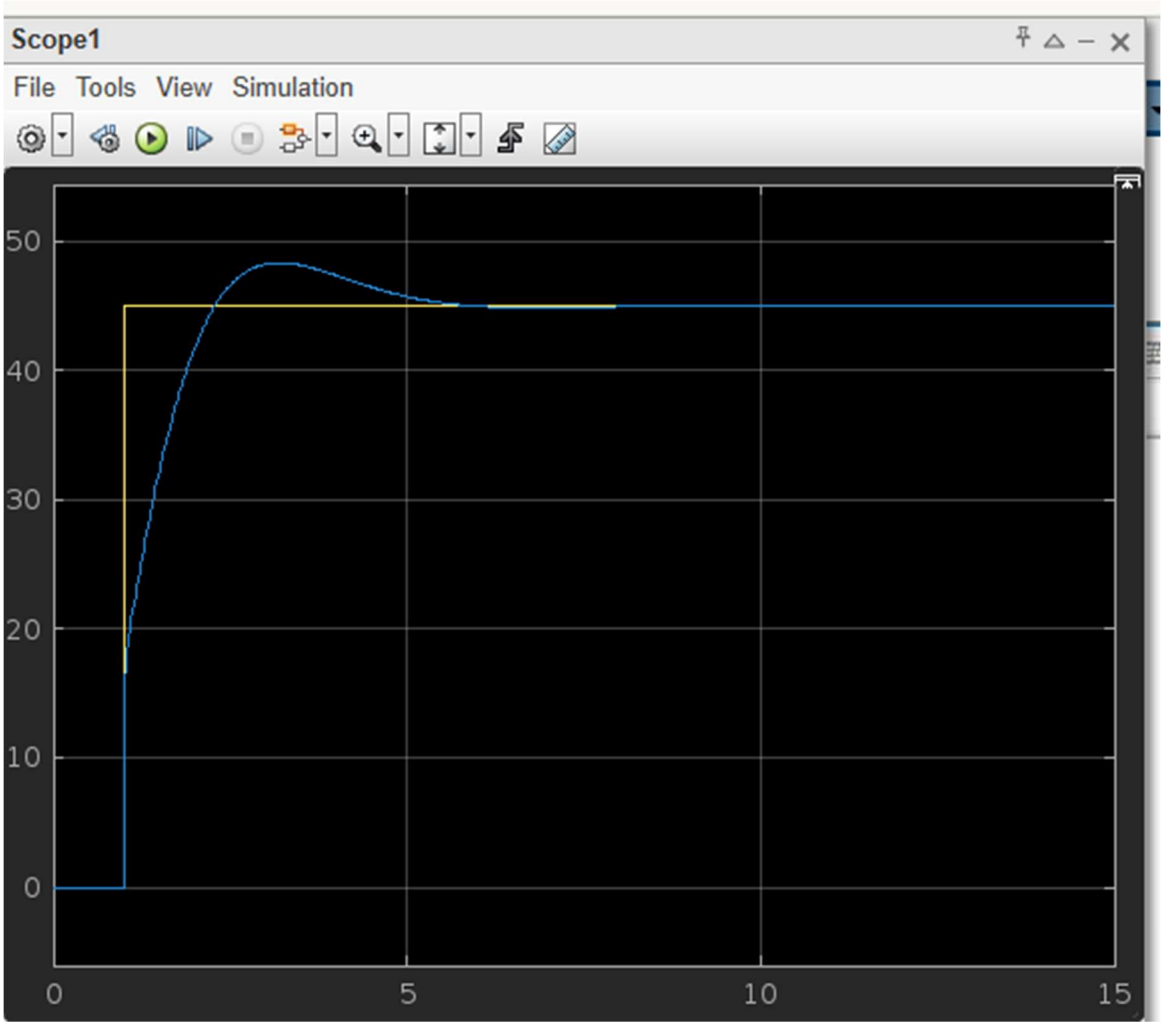
   Aplicando la sintonización por Ziegler –Nichols

Controlador	K_p	τ_i	τ_d
P	$\frac{\tau}{KL}$	∞	0
PI	$0.9\frac{\tau}{KL}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{\tau}{KL}$	$2L$	$0.5L$

Tabla de Ziegler Nichols – Método 1

Valores obtenidos para PID ISA

Kp=2.443
Ti=1.035
Td=0.258



Valores Obtenidos para PID Paralelo

Proporcional

Kp = K

Integral

Ki = Kp / ti

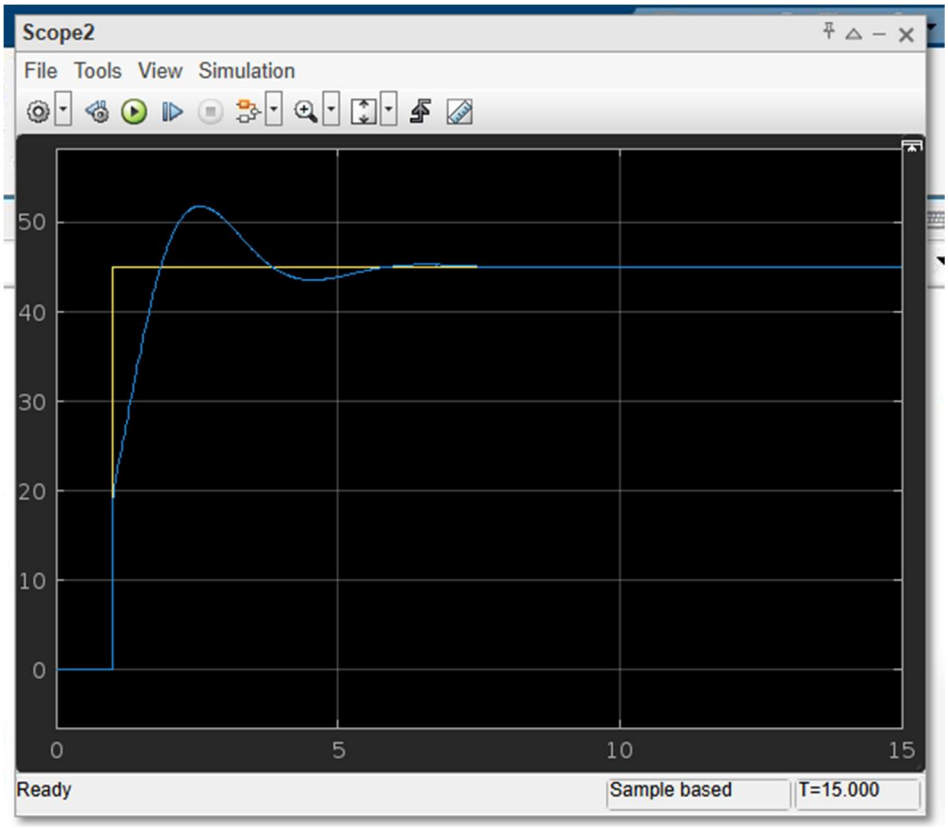
Derivativo

Kd = td * tm * Kp

Kp=2.443

Ki=2.36

Kd= 0.326



Aplicando el PID Paralelo al dispositivo Arduino, se obtuvo las siguientes conclusiones

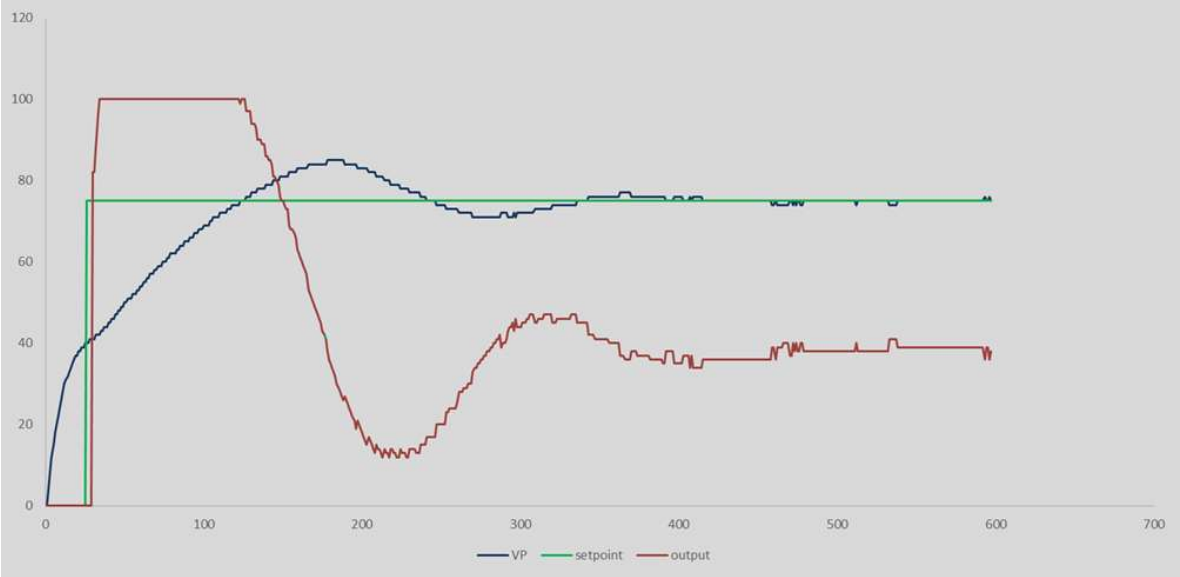
- Los resultados obtenidos son aproximaciones a los valores reales
- Se debe realizar ajustes finos a los valores de las constantes
- Cuando se utilizan los valores obtenidos de las sintonización , el sistema oscila
- Los valores más ajustados al proceso y que permiten estabilizarlo son los siguientes

Kp = 2.443

Ki = 0.12

Kd = 0.01

Resultados reales en el dispositivo Arduino, para un setpoint de 75 Grados



5. Programa del controlador Arduino

```
#include <TimerOne.h>           // libreria especializada en interrpciones
del timer1
volatile int i=0;               // variable contador
volatile boolean cruce_cero=0;  // variable cruce cero
#define triac 3                 // salida del moc 3021
#define led 13
#define AD A0                   // entrada analogica
int dim=0;                      // on= 0 , 83=off
int T_int = 100;                 // tiempo para interrupciones en useg es
100, se coloca 10 para poder simularlo
//int T_int = 100;              // en una placa real utilizar este valor
int POT=0;                      // variable para A/D
int MOT=0;
float lm35=0;
int temperatura=0;
float filteredPOT=0; // filtro de suavizado

String dato;
int pos;
String label;
String valor;


int j = 0;   // contador de la aplicacion de excel

// pid
float Kp = 0;   // Ganancia Proporcional
float Ki = 0;    // Ganancia Integral
float Kd = 0;    // Ganancia Derivativa
float minOutput = 0;  // Límite inferior de salida
float maxOutput = 100;   // Límite superior de salida
float output=0;

// ==========================================
// Variables de estado del PID
// ==========================================
float integral = 0;
float prevError = 0;
unsigned long lastTime = 0;
float setpoint=0;
```

```
float input;
float error;
float P=0;
float D=0;
float KP=0;
float KD=0;
float KI=0;
float PID=0;



void setup() {
Serial.begin(9600);
pinMode(triac, OUTPUT);                              // configura como salida
pinMode(led, OUTPUT);
pinMode(AD, INPUT);                                  // configura como entrada
attachInterrupt(0,deteccion_cruce_cero, RISING);  // detecta la interrupcion
por cambio de estado flanco de subida INT0 es el mismo pin PD2
Timer1.initialize(T_int);                            // inicia la libreria con
el tiempo
Timer1.attachInterrupt(Dimer, T_int);                // en cada interrupcion
ejecuta el codigo del dimer cada 100 useg



Serial.println("CLEARSHEET"); // clears sheet starting at row 1
Serial.println("LABEL,Date,Time,Timer,j,vp,setpoint,output,error,Kp,Ki,Kd,PI
D,millis");


}

void deteccion_cruce_cero()    // si existe cruce por cero entoces ejecuto
el codigo
{                              // resenteando el valor de i y apago el triac
cruce_cero=true;               // actica la bandera
i=0;
digitalWrite(triac,LOW);
digitalWrite(led,LOW);
}

void Dimer()
{
  if (cruce_cero==true)  // si la bandera esta activa
```

```
  {
    if (i>=dim)                 // si i es menor que el valor
setpoint  mantiene el triac apagado y aumenta el contador i
    {                           // si i supera al set point dim activa el
triac hasta el final del bucle y desactiva la bandera
      digitalWrite(triac,HIGH);
      digitalWrite(led,HIGH);
      i=0;
      cruce_cero=false;
    }
    else
    {
      i++;
    }
  }
}

void actuador()
{
dato.trim();
dato.toLowerCase();
pos= dato.indexOf(':');
label= dato.substring(0,pos);
valor= dato.substring(pos+1);
if (label.equals("mot"))
  {
    if (MOT != valor.toInt())
    {
      MOT=valor.toInt();
      //mot.write(POT);
      setpoint=MOT;
    }
  }
if (label.equals("kp"))
  {
    if (KP != valor.toFloat())
    {
      KP=valor.toFloat();
    }
  }
if (label.equals("ki"))
  {
    if (KI != valor.toFloat())
    {
```

```
      KI=valor.toFloat();
    }
  }
if (label.equals("kd"))
  {
    if (KD != valor.toFloat())
    {
      KD=valor.toFloat();
    }
  }
if (label.equals("pid"))
  {
    if (PID != valor.toInt())
    {
      PID=valor.toInt();
    }
  }
}



void loop()
{
  if (Serial.available())
  {
    dato=Serial.readString();
  }

actuador();

float Kp = KP;   // Ganancia Proporcional
float Ki = KI;    // Ganancia Integral
float Kd = KD;    // Ganancia Derivativa

POT= analogRead(AD);
lm35= POT*0.488;   //  (5000 mv / 1023)  /  10

filteredPOT= (0.9*filteredPOT + 0.1*lm35);      // filtro de suavizado

temperatura = filteredPOT; // Leer temperatura (°C)
//delay(2000);
input = int(temperatura); // Leer sensor

if (PID==0)
```

```
{
  // 1. Calcular tiempo transcurrido
  unsigned long now = millis();
  float deltaTime = (now - lastTime) / 1000.0; // en segundos
  lastTime = now;
  // 2. Calcular error
  error = setpoint - input;
  // 3. Término Proporcional
  P = Kp * error;
  // 4. Término Integral (con anti-windup)
  integral += Ki * error * deltaTime;
  if(integral > maxOutput) integral = maxOutput;
  if(integral < minOutput) integral = minOutput;
  // 5. Término Derivativo (evita "derivative kick")
  D = Kd * (error - prevError) / deltaTime;
  prevError = error;
  // 6. Calcular salida total
  output = P + integral + D;
  // 7. Aplicar límites
  if(output > maxOutput) output = maxOutput;
  if(output < minOutput) output = minOutput;
}

if (PID==1)
{output=setpoint;}

dim = map(output,0,100,83,0);  // mapea el valor a un rango de 0 a 83
Serial.println( (String) "DATA,DATE,TIME,TIMER," + j++ + "," +
String(int(temperatura)) + "," + String(int(setpoint)) + "," +
String(int(output)) + "," + String(int(error)) + "," + String(KP) + "," +
String(KI) + "," + String(KD) + "," + String(int(PID)) + "," + millis() + ""
);
delay(1000); //  retardo para poder enviar los datos a excel
}
```

Nota: la función actuador() permite recibir la información desde la hoja de datos Excel

El pid utilizado es un tipo PARALELO

Se controla la potencia cortando la onda sinusoidal de acuerdo al valor de la salida del controlador PID