Optimización del Control de Temperatura en Máquina de Sellado

Erick Marín Rojas

Monday 12th February, 2024

1 Resumen

El control preciso de la temperatura es crucial para garantizar la calidad y consistencia del proceso de sellado por herramienta caliente. Sin embargo, la ubicación incorrecta de la termocupla K en la máquina de sellado actual conduce a mediciones inexactas que afectan el rendimiento del controlador PID. Este informe analiza dos enfoques para mejorar el control de temperatura: la reubicación de la termocupla K y el estudio del comportamiento en la resistencia, en la termocupla y en la superficie de contacto.

2 Introducción

El proceso de sellado por herramienta caliente se utiliza ampliamente en diversas industrias para unir materiales termoplásticos. Un control preciso de la temperatura es esencial para lograr sellados uniformes, fuertes y duraderos. La máquina de sellado analizada actualmente presenta errores en el control de temperatura debido a la ubicación inadecuada de la termocupla K. Esto provoca lecturas inexactas que comprometen la calidad del sello y pueden generar sobrecalentamiento, subcalentamiento, desperdicio de material y posibles daños.

Este informe propone dos estrategias para abordar este problema y optimizar el control de temperatura:

2.1 Reubicación de la termocupla K

Involucra mover la termocupla a una ubicación en la superficie de contacto que refleje con mayor precisión la temperatura experimentada por el material durante el sellado. Para esto reubicaremos la resistencia por encima de la termocupla, dejándo a esta entre la superficie caliente y la resistencia.

2.2 Evaluación del controlador

Este enfoque involucra el desarrollo de un sistema para de captura de datos evaluar los parámetros del controlador PID autoafinados.

2.2.1 Configuración del sistema de medición con Arduino

Hardware:

- Placa Arduino: Se seleccionará una placa compatible con los sensores elegidos, con suficientes entradas analógicas y digitales. No se priorizará una con WiFi o Bluetooth para comunicación inalámbrica pues estas medidas sólo se toman cuando es necesario evaluar el controlador, no de forma paulatina, entonces una laptop conectada al arduino basta.

Sensores:

- Sensores de temperatura: Varios, utilizados cerca de la termocupla, superficie de contacto y resistencia.
- Sensor de voltaje del relé (para monitorear la potencia aplicada a la resistencia): Con este sensor mediremos la acción aplicada.

Software:

- Leer señales de los sensores.
- Procesar y almacenar datos en tiempo real con filtrado.
- Visualizar los datos procesados en una interfaz gráfica simple (monitor serie en la laptop). Usar la interfaz para ingresar los actuales parámetros del PID
- Tomar los datos procesados y parámetros actuales para producir una evaluación del rendimiento de los actuales parámetros, de forma en la que multiples tomas de datos queden catalogadas (rendimiento, parámetros, series de tiempo con los resultados)

3 Arquitectura del Software

3.1 Módulo de Adquisición de Datos

- Lee los datos de los sensores de temperatura, voltaje del relé y otros sensores relevantes.
- Implementa un filtro digital para eliminar el ruido de las señales, a través del muestreo.
- Almacena los datos en un pequeño caché de instantes que es contrantemente transmitido por el puerto serial.
- Los datos capturados por el arduino y transmitidos al computador son guardados en un caché que el resto del pipeline accesa.

3.2 Módulo de Procesamiento de Datos

- Calcula la respuesta del sistema a la acción de control (método de pasos), para los sistemas vistos desde el sensor en la termocupla, sensor en la resistencia y sensor en la superficie.
- Permite al usuario ingresar los parámetros del PID y la temperatura de referencia que se ingresó al controlador a través de una interfaz, por prueba tomada.
- Permite al usuario ver los gráficos de los datos obtenidos para poder descartar los resultados de una prueba o no.

3.3 Módulo de Evaluación del Controlador

- Evalúa el rendimiento del controlador para extraer métricas como: el tiempo de respuesta, el sobreimpulso y el error de estado estable. Además de heurísticas que castiguen la diferencia entre la temperatura en la superficie de contacto de la herramienta caliente y la termocupla, como el error integral absoluto entre las temperaturas.
- Cataloga el rendimiento del controlador autoajustado con sus debidos valores para los parámetros PID, respuestas y heurísticas.

3.4 Interfaz Gráfica de Usuario

- Muestra los datos de los sensores en tiempo real.
- Permite al usuario ingresar los parámetros del controlador PID y temperatura de referencia.
- Visualiza las gráficas de la respuesta del sistema catalogadas.
- Presenta los resultados de la evaluación del controlador, como catálogo.
- Grafica por item del catálogo junto con las heurísticas el lugar en el plano complejo de los polos y ceros de las respuestas encontradas de la acción a los sensores.

3.5 Pipeline

- El Arduino adquiere los datos de los sensores y los transmite al sistema de procesamiento de datos.
- Python recibe los datos del Arduino y realiza el procesamiento y análisis de la acción vista desde los sensores.
- Se presentan los datos procesados para que el usuario los analice y decida si eliminar.

- El sistema de procesamiento de datos evalúa el rendimiento del controlador y presenta los resultados en la interfaz gráfica de usuario.
- El usuario interactúa con la interfaz gráfica para ajustar los parámetros del controlador PID según sea necesario.

3.6 Consideraciones

- La arquitectura del software debe ser modular y escalable para permitir la fácil incorporación de nuevas funcionalidades.
- El sistema debe ser robusto ante los errores en los sensores.
- La interfaz gráfica debe ser simple (de fabricar) y fácil de usar.
- Todos los módulos pueden ser probados de forma individual y verificados a nivel de unidades (usando datos sintéticos).

Cronograma de 11 días:

Al considerar el tiempo disponible se definen once días de trabajo como roadmap flexible con el cual trabajar el desarrollo del software:

[leftmargin=*]**Día 1:**

- – Definir qué se necesita (hardware y software).
  – Comprar lo necesario.
  – Instalar el software y preparar el espacio de trabajo.
  – Empezar a construir el sistema de toma de datos.
- **Día 2:**
  – Terminar de construir el sistema de toma de datos.
  – Escribir el programa en Python para procesar datos.
  – Empezar a calcular la respuesta del sistema.
- **Día 3:**
  – Terminar de escribir el programa en Python.
  – Probar y corregir el programa.
  – Empezar a evaluar el rendimiento del controlador.
- **Día 4:**
  – Terminar de calcular la respuesta del sistema.
  – Diseñar la interfaz gráfica en Python.
  – Mostrar datos en la interfaz gráfica.

- **Día 5:**
  - Probar y corregir la interfaz gráfica.
  - Unir el sistema de toma de datos con el software de procesamiento.
- **Día 6:**
  - Probar y validar el sistema completo.
  - Empezar a recopilar y analizar datos.
- **Día 7:**
  - Continuar recopilando y analizando datos.
  - Empezar a escribir el informe final.
- **Día 8:**
  - Terminar de recopilar y analizar datos.
  - Continuar escribiendo el informe final.
- **Día 9:**
  - Finalizar el informe final.
- **Día 10:**
  - Practicar la presentación del proyecto.
- **Día 11:**
  - Presentar el proyecto.

4 FODA de las Estrategias Propuestas

4.1 Estrategia 1: Reubicación de la Termocupla K

Fortalezas:

- Mejora significativa en la precisión de la medición de la temperatura: Al ubicar la termocupla entre la superficie caliente y la resistencia, se obtiene una medición más cercana a la temperatura real experimentada por el material durante el sellado.
- Solución relativamente simple y económica: Reubicar la termocupla no requiere de una inversión significativa en nuevos equipos o modificaciones complejas a la máquina.
- Fácil de implementar: El proceso de reubicación de la termocupla es sencillo y puede ser realizado por personal con conocimientos básicos en electrónica.

Oportunidades:

- Optimización del proceso de sellado: La medición precisa de la temperatura permite un mejor control del proceso de sellado, lo que puede traducirse en una mayor calidad y consistencia de los sellos.
- Reducción de desperdicio: Un mejor control de la temperatura puede ayudar a reducir el desperdicio de material debido a sellos defectuosos.
- Mejora de la seguridad: La medición precisa de la temperatura puede ayudar a prevenir sobrecalentamientos que podrían ocasionar riesgos de seguridad.

Debilidades:

- Posible impacto en la vida útil de la termocupla: La exposición a temperaturas más elevadas podría reducir la vida útil de la termocupla.
- Calibración necesaria: La reubicación de la termocupla requiere de una nueva calibración del sistema de control de temperatura.

Amenazas:

- Daños a la termocupla durante la reubicación: Se debe tener cuidado durante el proceso de reubicación para evitar daños a la termocupla.
- Incompatibilidad con algunos modelos de máquinas: Es posible que la reubicación de la termocupla no sea compatible con todos los modelos de máquinas de sellado.

4.2 Estrategia 2: Evaluación del Controlador Autoajustado

Fortalezas:

- Análisis completo del rendimiento del controlador: Esta estrategia permite obtener una evaluación completa del comportamiento del controlador PID autoajustado, incluyendo su respuesta a diferentes tipos de perturbaciones.
- Identificación de áreas de mejora: El análisis del rendimiento del controlador puede ayudar a identificar áreas en las que se puede mejorar la performance del control de temperatura.
- Optimización de los parámetros del controlador: La evaluación del controlador puede ayudar a determinar los parámetros PID óptimos para el sistema.

Oportunidades:

- Mejora significativa en la precisión del control de temperatura: La optimización de los parámetros del controlador puede traducirse en una mejora significativa en la precisión del control de temperatura.

- Reducción de costos: Un mejor control de la temperatura puede ayudar a reducir costos de producción al disminuir el desperdicio y el consumo de energía.
- Aumento de la productividad: La optimización del proceso de sellado puede aumentar la productividad de la máquina.

Debilidades:

- Estrategia más compleja: Esta estrategia requiere de conocimientos especializados en control de sistemas y programación.
- Inversión en hardware y software: Se requiere de una inversión en hardware (sensores, Arduino) y software para la implementación del sistema de evaluación.
- Tiempo de implementación: La implementación de esta estrategia puede tomar más tiempo que la reubicación de la termocupla.

Amenazas:

- Errores en la medición de datos: Los errores en la medición de datos pueden afectar la precisión de la evaluación del controlador.
- Dificultad para interpretar los resultados: La interpretación de los resultados de la evaluación del controlador puede ser compleja y requerir de la ayuda de un experto.

5 Conclusiones

Ambas estrategias tienen sus pros y contras. La reubicación de la termocupla es una solución más simple y económica que puede mejorar significativamente la precisión de la medición de la temperatura. La evaluación del controlador autoajustado es una estrategia más compleja, pero puede proporcionar información valiosa para optimizar el rendimiento del control de temperatura.

La mejor estrategia para una empresa en particular dependerá de sus necesidades específicas, recursos disponibles y presupuesto. Se recomienda que la empresa evalúe cuidadosamente ambas opciones antes de tomar una decisión.

6 Apéndice - Pseudocódigo

6.1 Python

- Módulo de Adquisición de Datos

```
def leer_datos_arduino():
    """
    Lee datos del puerto serie del Arduino y devuelve una tupla con las medida
```

```
    Retorna:
      (temperatura_termocupla, temperatura_resistencia, temperatura_superficie
    """
    datos = serial.readline().decode("utf-8").strip().split(",")
    return float(datos[0]), float(datos[1]), float(datos[2]), float(datos[3])

def almacenar_datos(datos):
    """
    Almacena los datos en un archivo CSV.

    Parametros:
      datos: Tupla con las medidas (temperatura_termocupla, temperatura_resist
             temperatura_superficie, voltaje_rele)
    """
    with open("datos.csv", "a") as archivo:
        archivo.write(",".join(str(dato) for dato in datos) + "\n")
```

- Módulo de Procesamiento de Datos

```
def calcular_respuesta_sistema(temperatura_termocupla, temperatura_resistenc
                               temperatura_superficie, voltaje_rele):
    """
    Calcula la respuesta del sistema a la acci n de control para diferentes s

    Parametros:
      temperatura_termocupla: Temperatura medida por la termocupla.
      temperatura_resistencia: Temperatura medida por la resistencia.
      temperatura_superficie: Temperatura medida en la superficie de contacto.
      voltaje_rele: Voltaje aplicado a la resistencia.

    Retorna:
      (respuesta_termocupla, respuesta_resistencia, respuesta_superficie)
    """
    # Implementar el m todo de pasos para calcular la respuesta del sistema
    # para cada sensor.

    return respuesta_termocupla, respuesta_resistencia, respuesta_superficie

def evaluar_parametros_pid(parametros_pid, temperatura_referencia):
    """
    Eval a el rendimiento del controlador PID con los par metros dados.

    Parametros:
      parametros_pid: Tupla con los par metros PID (Kp, Ki, Kd).
      temperatura_referencia: Temperatura de referencia deseada.
```

```
    Retorna:
      (tiempo_respuesta, sobreimpulso, error_estado_estable,
       metricas_adicionales)
    """
    # Implementar el c lculo de las m tricas de rendimiento del controlador
    # (tiempo de respuesta, sobreimpulso, error en estado estable) y otras m
    # que consideren la diferencia entre la temperatura en la superficie de co
    # y la medida por la termocupla.

    return tiempo_respuesta, sobreimpulso, error_estado_estable, metricas_adic
```

- Módulo de Evaluación del Controlador

```
def evaluar_controlador(datos, parametros_pid, temperatura_referencia):
    """
    Eval a el rendimiento del controlador PID con los datos medidos.

    Par metros:
      datos: Lista de tuplas con las medidas (temperatura_termocupla,
             temperatura_resistencia, temperatura_superficie, voltaje_rele).
      parametros_pid: Tupla con los par metros PID (Kp, Ki, Kd).
      temperatura_referencia: Temperatura de referencia deseada.

    Retorna:
      (rendimiento_termocupla, rendimiento_resistencia, rendimiento_superficie
    """
    rendimiento_termocupla = evaluar_parametros_pid(parametros_pid,
                                                    temperatura_referencia,
                                                    datos[:, 0])
    rendimiento_resistencia = evaluar_parametros_pid(parametros_pid,
                                                     temperatura_referencia,
                                                     datos[:, 1])
    rendimiento_superficie = evaluar_parametros_pid(parametros_pid,
                                                    temperatura_referencia,
                                                    datos[:, 2])
    return rendimiento_termocupla, rendimiento_resistencia, rendimiento_super
```

- Interfaz Gráfica de Usuario

```
import tkinter as tk
from matplotlib.backends.tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
import numpy as np

from modulo_adquisicion_datos import leer_datos_arduino
from modulo_procesamiento_datos import calcular_respuesta_sistema, evaluar_p
from modulo_evaluacion_controlador import evaluar_controlador
```

```
# Par metros
frecuencia_muestreo = 10  # Hz
numero_muestras = 10

# Variables
datos = []
tiempo = np.arange(0, numero_muestras / frecuencia_muestreo, 1 / frecuencia_

# Configuraci n de la ventana principal
ventana = tk.Tk()
ventana.geometry("800x600")
ventana.title("Evaluaci n del Control de Temperatura")

# Figura para gr ficos
figura = plt.Figure(figsize=(5, 4))
ax_termocupla = figura.add_subplot(131)
ax_resistencia = figura.add_subplot(132)
ax_superficie = figura.add_subplot(133)

# Canvas para mostrar la figura
canvas = FigureCanvasTkAgg(figura, ventana)
canvas.get_tk_widget().pack(side=tk.TOP)

# Controles para el PID
label_kp = tk.Label(ventana, text="Kp:")
label_kp.pack(side=tk.LEFT)
entry_kp = tk.Entry(ventana)
entry_kp.pack(side=tk.LEFT)

label_ki = tk.Label(ventana, text="Ki:")
label_ki.pack(side=tk.LEFT)
entry_ki = tk.Entry(ventana)
entry_ki.pack(side=tk.LEFT)

label_kd = tk.Label(ventana, text="Kd:")
label_kd.pack(side=tk.LEFT)
entry_kd = tk.Entry(ventana)
entry_kd.pack(side=tk.LEFT)

# Bot n para iniciar la medici n
boton_iniciar = tk.Button(ventana, text="Iniciar", command=iniciar_medicion)
boton_iniciar.pack(side=tk.BOTTOM)

# Funci n para iniciar la medici n
def iniciar_medicion():
```

```
    # Obtener los par metros del PID
    kp = float(entry_kp.get())
    ki = float(entry_ki.get())
    kd = float(entry_kd.get())

    # Iniciar la medici n
    while True:
      # Leer datos del Arduino
      temperatura_termocupla, temperatura_resistencia, temperatura_superficie,

      # Almacenar datos
      datos.append((temperatura_termocupla, temperatura_resistencia, temperat

      # Calcular la respuesta del sistema
      respuesta_termocupla, respuesta_resistencia, respuesta_superficie = calc
          temperatura_termocupla, temperatura_resistencia, temperatura_superfi
      )

      # Evaluar el rendimiento del controlador
      rendimiento_termocupla, rendimiento_resistencia, rendimiento_superficie
          datos, (kp, ki, kd), 100
      )

      # Actualizar la gr fica
      ax_termocupla.clear()
      ax_termocupla.plot(tiempo, respuesta_termocupla)
      ax_resistencia.clear()
      ax_resistencia.plot(tiempo, respuesta_resistencia)
      ax_superficie.clear()
      ax_superficie.plot(tiempo, respuesta_superficie)

      canvas.draw()

      # Esperar un tiempo
      time.sleep(1 / frecuencia_muestreo)

# Iniciar la ventana principal
ventana.mainloop()
```

6.2 Arduino

- Toma de datos y envío a computador

```
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
```

```
// Definici n de pines
#define PIN_TERMOCUPLA A0
#define PIN_RESISTENCIA A1
#define PIN_SUPERFICIE A2
#define PIN_RELE 13

#define PIN_SD_CS 53
#define PIN_SD_MOSI 51
#define PIN_SD_MISO 50
#define PIN_SD_SCK 52

// Variables sensoriales
float temperatura_termocupla;
float temperatura_resistencia;
float temperatura_superficie;
float voltaje_rele;

// Variables de muestreo
int frecuencia_muestreo = 10; // Hz
int tiempo_muestreo = 1000 / frecuencia_muestreo; // ms
int numero_muestras = 10;
float suma_temp_termocupla = 0;
float suma_temp_resistencia = 0;
float suma_temp_superficie = 0;
float suma_voltaje_rele = 0;

// Variables SD
File archivo_datos;

// Configuraci n inicial
void setup() {
  Serial.begin(9600);

  pinMode(PIN_TERMOCUPLA, INPUT);
  pinMode(PIN_RESISTENCIA, INPUT);
  pinMode(PIN_SUPERFICIE, INPUT);
  pinMode(PIN_RELE, OUTPUT);

  if (!SD.begin(PIN_SD_CS)) {
    Serial.println("Tarjeta SD no inicializada");
    while (true);
  }

  archivo_datos = SD.open("datos.csv", FILE_WRITE);
  if (!archivo_datos) {
```

```
    Serial.println("Error al crear archivo de datos");
    while (true);
  }

  archivo_datos << "Temperatura_Termocupla,Temperatura_Resistencia,Temperat
}

// Bucle principal de adquisici n y env o de datos
void loop() {

  // Promediado de muestras
  for (int i = 0; i < numero_muestras; i++) {
    temperatura_termocupla = analogRead(PIN_TERMOCUPLA);
    temperatura_resistencia = analogRead(PIN_RESISTENCIA);
    temperatura_superficie = analogRead(PIN_SUPERFICIE);
    voltaje_rele = analogRead(PIN_RELE);

    suma_temp_termocupla += temperatura_termocupla;
    suma_temp_resistencia += temperatura_resistencia;
    suma_temp_superficie += temperatura_superficie;
    suma_voltaje_rele += voltaje_rele;

    delay(tiempo_muestreo);
  }

  // C lculo de promedios
  float promedio_temp_termocupla = suma_temp_termocupla / numero_muestras;
  float promedio_temp_resistencia = suma_temp_resistencia / numero_muestras;
  float promedio_temp_superficie = suma_temp_superficie / numero_muestras;
  float promedio_voltaje_rele = suma_voltaje_rele / numero_muestras;

  // Env o de datos por serial
  Serial.print(promedio_temp_termocupla);
  Serial.print(",");
  Serial.print(promedio_temp_resistencia);
  Serial.print(",");
  Serial.print(promedio_temp_superficie);
  Serial.print(",");
  Serial.println(promedio_voltaje_rele);

  // Almacenamiento en SD
  archivo_datos << promedio_temp_termocupla << "," << promedio_temp_resisten
    promedio_temp_superficie << "," << promedio_voltaje_rele << "\n";

  // Reinicio de variables para el siguiente ciclo
  suma_temp_termocupla = 0;
```

```
  suma_temp_resistencia = 0;
  suma_temp_superficie = 0;
  suma_voltaje_rele = 0;
}
```