Laboratorio 3:
Arduino: PID,GPIO,ADC y comunicaciones

Jafet Soto Arrieta, B77543
Universidad de Costa Rica, Escuela de Ingeniería Eléctrica
IE-0623 Laboratorio de Microcontroladores
08 de mayo del 2024
jafet.soto@ucr.ac.cr

***Resumen*—**

El presente reporte describe el desarrollo de una incubadora de huevos automática controlada por un Arduino UNO. El sistema está diseñado para regular y monitorear la temperatura en un rango operativo de 30°C a 42°C.

I. Introducción

Para el desarrollo de la incubadora de huevos automática, se ha seleccionado un microcontrolador Arduino UNO, el cual se encarga de regular y monitorear la temperatura dentro de un rango operativo establecido entre 30°C y 42°C. El sistema cuenta con una interfaz de usuario a través de una pantalla LCD PCD8544, la cual muestra en tiempo real la temperatura junto con otros parámetros cruciales del proceso.

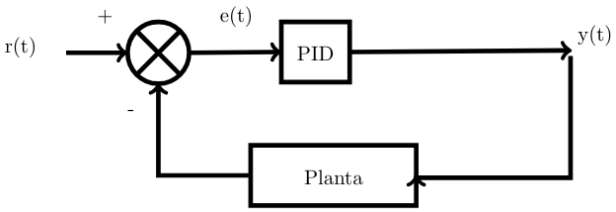

Figura 1: Diagrama del sistema de control para la incubadora automática [2].

El ajuste de la temperatura deseada se realiza manualmente utilizando un potenciómetro. Para asegurar la estabilidad térmica dentro del rango seleccionado, se implementa un controlador de tipo PID. Los parámetros de este controlador han sido ajustados de manera heurística, considerando la capacidad del sistema para enfriar por debajo y calentar por encima de la temperatura ambiente.

El sistema de monitoreo y seguridad de la incubadora incluye varias indicaciones visuales:

- Un LED azul que se activa cuando la temperatura desciende por debajo de los 30°C.
- Un LED rojo que se enciende al superar los 42°C.
- Un LED verde que indica que la temperatura se encuentra dentro del rango de operación seguro.

Adicionalmente, se ha incorporado un interruptor que permite habilitar o deshabilitar la comunicación con una computadora personal. Esta funcionalidad facilita el envío de los datos de temperatura a través del puerto serial, utilizando el protocolo USART. Se ha desarrollado una aplicación en Python capaz de recibir, almacenar y visualizar los datos en formato CSV. Esta aplicación también proporciona gráficos de la temperatura, la señal de control del PID y la temperatura real medida.

II. Nota teórica

II-A. Características generales del Arduino UNO

El Arduino UNO es una plataforma de microcontrolador de código abierto basada en el microchip ATmega328P; es uno de los modelos más populares y accesibles de la familia Arduino, ampliamente reconocido por su facilidad de uso y su versatilidad.

Esta placa cuenta con un conjunto robusto de entradas y salidas digitales y analógicas, que se pueden conectar a diversos sensores, actuadores y otros circuitos para el desarrollo de proyectos avanzados. El Arduino UNO se programa a través del entorno de desarrollo de Arduino (IDE), que simplifica la implementación de código en C o C++.

II-B. Diagrama de pines

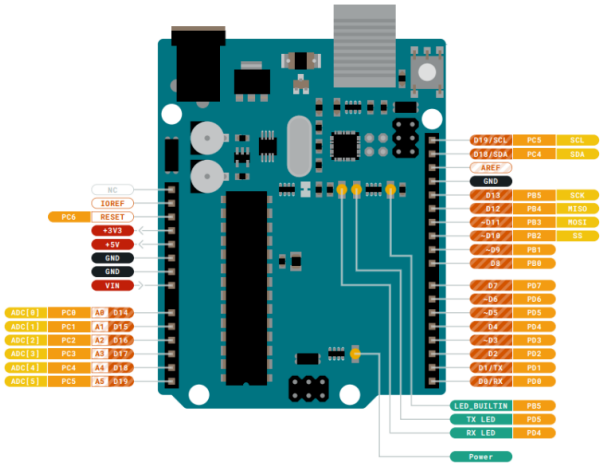

Figura 2: Diagrama de pinout del arduino UNO [1].

A continuación se muestra la descripción de los pines en configuración analógica y digital para el arduino UNO:

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

Figura 3: JANALOG [1].

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

Figura 4: JDIGITAL [1].

II-C. La pantalla PCD8544

Es un pequeño display LCD gráfico que es popular por ser utilizada en los teléfonos Nokia 5110 y 3310. Este módulo es especialmente apreciado en proyectos de electrónica DIY debido a su facilidad de uso con plataformas como Arduino, además de su bajo costo. La pantalla ofrece una resolución de 84x48 píxeles, lo que permite mostrar texto, gráficos simples y otros elementos visuales con una claridad razonable. Operando con un voltaje de 3.3V, la PCD8544 se comunica a través de una interfaz SPI.

II-D. Controlador PID

Un controlador PID es un componente esencial en sistemas de control para ajustar una variable a un estado específico mediante un proceso de lazo cerrado. En nuestro proyecto, el sistema que define el PID se compone de un potenciómetro para establecer la temperatura deseada y una simulación del proceso termal mediante la función simPlanta(), que juntos controlan la temperatura de una incubadora virtual.

La implementación del lazo cerrado en Arduino, dada su complejidad, se simplifica usando una biblioteca PID que maneja seis parámetros: Setpoint, Output, Input, Kp, Ki y Kd. Aquí, el Setpoint se ajusta mediante el potenciómetro, permitiendo seleccionar temperaturas entre 20 y 80 °C. El Input corresponde a la salida de simPlanta(), y el Output es la señal de retroalimentación al sistema. Los valores de los parámetros del PID se han establecido en Kp=15, Ki=0.5 y Kd=1, optimizando el control para mantener la temperatura en el rango requerido.

II-E. Comunicación serial

La comunicación serial se implemenat utilizando el protocolo USART para conectar virtualmente la placa Arduino con un script de Python. Este script se encarga de recopilar los datos del setpoint, entrada y salida del PID para su posterior visualización gráfica. La configuración de la comunicación se inicia en el Arduino mediante la función Serial.begin(9600); dentro del setup(), lo que establece la tasa de baudios a 9600 para la transmisión de datos.

Esta configuración permite que el Arduino envíe datos automáticamente a través de su puerto serial, que pueden ser capturados por el script de Python.

III. Desarrollo y análisis.

Una vez definimos el funcionamiento de las diferentes partes de nuestro sistema podemos establecer el diseño del esquema de simulación en SimulIDE:

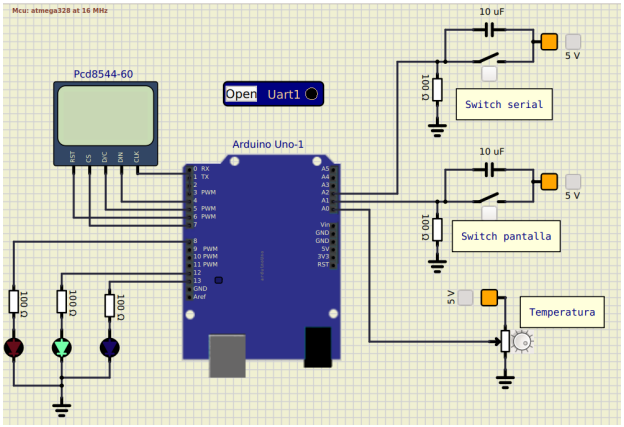

Figura 5: Esquema del laboratorio.

Como en laboratorio pasador utilizamos el sistema sencillo de debouncer que nos permite eliminar el rudo de los interruptores, para que la detección de la señal de cambio de estado se pueda detectar de forma eficiente.

El primer pulsador se utiliza para activar la comunicación serial, se conecta al pin A2, la detección de una entrada cercana a los 5V activa el protocolo diseñado para imprimir los resultados. Los resultados se imprimen en la terminar, se pueden apreciar en **Open Serial Monitor**, en las opciones del MCU. Con los datos generados se elabora un archivo .csv que se analiza con el script Python.

El segundo pulsador se encarga de detectar el momento en el que se debe encender la pantalla, este pulsador se conecta al pin A1.

La conexión de la resistencia variable se realiza por la entrada A0, esta entrada analógica cuenta con un convertidor ADC de 10 bits, lo que implica que tenemos un rango de uso de 1024, es decir de 0 a 1023. La apertura mínima del potenciómetro se representa como 0 y su valor máximo como 1022; note que el valor máximo no es 1023, debido a que se en las pruebas este valor no se alcanza. Estos valores máximos y mínimos representan el rango de temperaturas de 20 a 80 °C.

Esta configuración la logramos con la función **map()** de arduino:

```
float tempWatts= map(value, 0, 1022, 20, 80);
```

Este valor representa la temperatura de la incubadora.

Para activar la pantalla utilizamos la función **setup()** de la biblioteca **PCD8544**, primero definimos la pantlla y luego configuramos la función **setup()**

```
// Objeto para la pantalla LCD:
PCD8544 lcd;

//
// ----------------------Setup:----------------------------
void setup() {
    Serial.begin(9600);

    // Configuración de pines para LEDs:
    pinMode(PIN_BLUE, OUTPUT);
    pinMode(PIN_GREEN, OUTPUT);
    pinMode(PIN_RED, OUTPUT);

    // Inicialización del LCD:
    lcd.begin(); // Resolución por defecto: 84x48

    // Configuración del PID:
    myPID.SetOutputLimits(-125, 125);
    myPID.SetMode(AUTOMATIC);
}
```

Figura 6: Código para la pantalla.

Note que también estamos configurando el PID, este se crea dentro del código y no tiene una parte física en el esquemático. El PID no excede los limites de -125 a 125 delimitado como máximo y mínimo.

Finalmente se agrega la etapa de salida de las señales de estado por medio de LEDs de colores; con la siguiente estructura de estados:

```
if (Input < 30) {
    digitalWrite(PIN_BLUE, HIGH);
    digitalWrite(PIN_RED, LOW);
    digitalWrite(PIN_GREEN, LOW);
} else if (Input > 42) {
    digitalWrite(PIN_RED, HIGH);
    digitalWrite(PIN_GREEN, LOW);
    digitalWrite(PIN_BLUE, LOW);
} else {
    digitalWrite(PIN_GREEN, HIGH);
    digitalWrite(PIN_RED, LOW);
    digitalWrite(PIN_BLUE, LOW);
}
```

Figura 7: Código para LEDs.

Se completa el programa principal con las asignaciones de GPIOs y se definen los loops que nos permiten completar el simulador de incubadora, a continuación se muestra los 2 estados de los LEDs.

El LED azul indica que las temperaturas son menores a 30°:

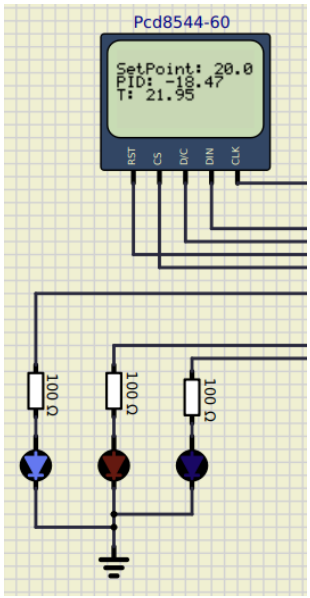

Figura 8: Temperatura inferior al valor permitido.

El LED verde indica que la temperatura está en el rengo permitido:

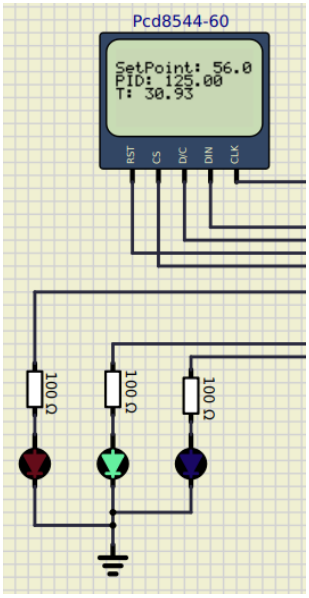

Figura 9: Temperatura ideal.

El LED rojo indica que se sobrepasó la temperatura limite superior:

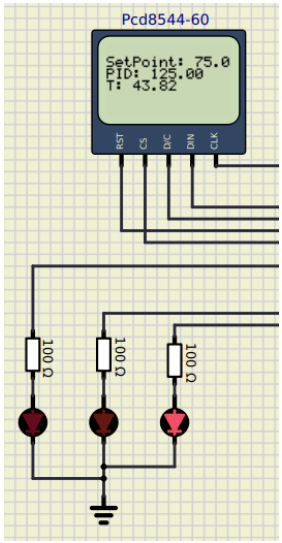

Figura 10: Temperatura sobrepasa el valor permitido.

Una vez que comprobamos el funcionamiento del sistema, podemos generar el archivo con los valores en formato **.csv** para que sean manipulados por Python.

Empleamos una típica función de impresión de gráficos de la biblioteca **pandas** y **matplotlib**, para manipular los datos y visualizar las relaciones entre el setpoint de la temperatura de operación, la salida de PID y la salida de la planta. Este enfoque nos permite observar claramente las dinámicas y el comportamiento del sistema de control a lo largo del tiempo.

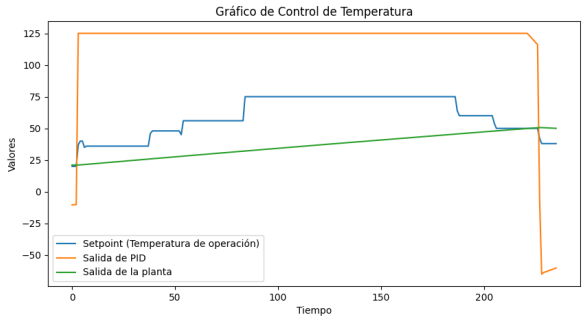

Figura 11: Gráfico de señales de control.

IV. Conclusiones y recomendaciones

Finalmente se implementa una simulación exitosa para una incubadora, demostrando la efectividad del sistema de control basado en Arduino para gestionar y estabilizar la temperatura dentro de un rango operativo específico. La integración de un controlador PID junto con una interfaz permite una interacción fácil y eficaz, garantizando que las condiciones dentro de la incubadora se mantengan dentro de los límites seguros para el proceso.

El desarrollo y la implementación de la simulación y el sistema de monitoreo mediante LEDs proporcionan señales visuales claras sobre el estado del sistema, lo que mejora la capacidad de respuesta y la interacción del usuario con el dispositivo. La comunicación serial extendida para exportar los datos a un sistema de análisis externo demuestra la flexibilidad del sistema para ser adaptado y monitorizado en diferentes plataformas, aumentando su aplicabilidad en entornos reales.

Es importante repasar el código Python para ser mejorar la implementación ya que no fue posible conectar en tiempo real la comunicación del codigo Python y la simulación.

Referencias

[1] *Arduino UNO R3 Product Reference Manual. (2024, 16 de abril).* Arduino. SKU: A000066. Recuperado de https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[2] *Laboratorio #3: Arduino: PID, GPIO, ADC y comunicaciones. (2022, II Ciclo).* MSc. Marco Villalta Fallas. IE-0624 Laboratorio de Microcontroladores, Escuela de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de Costa Rica.