# Laboratorio 3: Arduino: PID, GPIO, ADC y Comunicaciones


1nd Ronny Granados Pérez
*Escuela de Ingeniería Eléctrica*
*Universidad de Costa Rica.*
San José, Costa Rica
C03505
ronny.granadosperez@ucr.ac.cr

2nd Marvin Castro Castro
*Escuela de Ingeniería Eléctrica*
*Universidad de Costa Rica.*
San José, Costa Rica
C01884
marvin.castrocastro@ucr.ac.cr


## I. Introducción/Resumen

Para este laboratorio se trabajó, por primera vez en el curso, con el microcontrolador Arduino Uno, una de las placas más populares de la actualidad, para la implementación de un sistema controlado de temperatura que simulara una incubadora que, para una aplicación específica, debía mantenerse en un rango de temperatura de [30, 42]°C la cual es modulada mediante un potenciómetro. La finalidad del laboratorio es lograr la correcta implementación de un controlador PID y hacer uso de los distintos protocolos de comunicación que ofrece Arduino.

## II. Nota Teórica

### *II-A. Arduino Uno*

Este microcontrolador es uno de los más famosos a escala global. Se trata de una placa de desarrollo de hardware de código abierto con un procesador ATMega16U2 con arquitectura AVR de 8 bits. Cuenta con 12 GPIOs digitales, de los cuales 6 pueden ser utilizados como salidas PWM, 6 entradas analógicas, un reloj oscilante de cuarzo de 16 MHz, conexión USB, entrada para alimentación externa y botón de reset. Para programarlo se hace uso del IDE de Arduino desde el cual se puede compilar el código, generar binarios y demás.

2.1 Recommended Operating Conditions

| Symbol | Description | Min | Max |
|---|---|---|---|
| | Conservative thermal limits for the whole board: | -40 °C (-40 °F) | 85 °C ( 185 °F) |

Figura 1. Rangos de temperatura de operación para el Arduino Uno [1].

2.2 Power Consumption

| Symbol | Description | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| VINMax | Maximum input voltage from VIN pad | 6 | - | 20 | V |
| VUSBMax | Maximum input voltage from USB connector | | - | 5.5 | V |
| PMax | Maximum Power Consumption | - | - | xx | mA |

Figura 2. Consumo de potencia y tensión [1].

### *II-B. Pantalla PCD8544*

Se trata de un dispositivo de bajo consumo que se utiliza, en conjunto con el microcontrolador, para controlar una pantalla LCD. No se va a tocar con profundidad su arquitectura o especificaciones, sin embargo, si el cómo se inicializó y cómo se hizo para trabajar con esta desde el Arduino. Existen librerías para poder trabajar con mayor facilidad con esta pantalla y la forma de operara, con conocimientos básicos de programación, es muy intuitivo. Una pantalla PCD8544 real cuenta con más pines, sin embargo, el simulador posee una versión simplificada que se muestra a continuación:

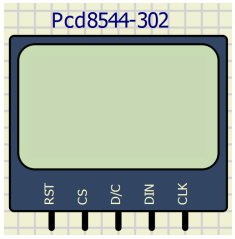


Figura 3. PCD8544 disponible en el simulador SimulIDE.

La forma de poder conectarla al Arduino, según la documentación de la librería utilizada [3], es la siguiente:

Cuadro I
Conexión al Arduino de la pantalla LCD disponible en el simulador.

| LCD Pin | Arduino Pin |
|---|---|
| **RST** | 6 |
| **CS** | 7 |
| **D/C** | 5 |
| **DIN** | 4 |
| **CLK** | 3 |

Para una explicación más detallada de cómo funciona la librería y cómo implementarla en el código se recomienda visitar el repositorio de GitHub de los desarrolladores en donde se podrán encontrar algunos ejemplos [3].

### *II-C. Controlador PID*

En la teoría de control, un controlador PID es un mecanismo/dispositivo que se utiliza para controlar un sistema de

lazo cerrado con el fin de lograr que una variable específica alcance el estado deseado [2]. Para implementar este lazo cerrado, se debe de contar con el propio PID, una entrada o valor de referencia (fijada mediante el potenciómetro) y una planta o proceso, que en este caso se trata de la incubadora. La implementación de la planta puede ser algo confuso con lo que trabajar puesto que se está trabajando en un entorno simulado y no se cuenta con la incubadora como tal, sino se nos fue brindada una función que simulaba el proceso térmico de este dispositivo llamada **simPlanta()**.

Puesto que implementar el lazo cerrado en Arduino es un proceso complejo y el objetivo del curso es distinto, se simplifica mediante una librería llamada **PID**, esta requiere de seis parámetros para funcionar que son: **Setpoint**, **Output**, **Input**, $\mathbf{K_p}$, $\mathbf{K_i}$ y $\mathbf{K_d}$.

En el caso de este laboratorio, se solicitaba llevar la temperatura de la incubadora a cualquiera que sea el valor que se fijara mediante el potenciómetro, sabiendo que, según el enunciado, este debía permitir fijar valores entre 20 y 80 °C.

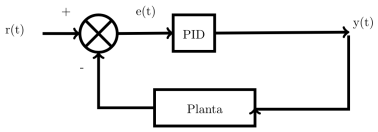


Figura 4. Diagrama de bloques del proceso térmico simulado en este laboratorio.

Sabiendo esto, se puede indicar cómo se definieron los parámetros del PID así como sus constantes proporcional, derivativa e integral, presentadas en la tabla II y III, respectivamente:

Cuadro II
DEFINICIÓN DE SETPOINT, INPUT Y OUTPUT DEL PID DENTRO DEL CÓDIGO.

| PID | Definición |
|---|---|
| **Setpoint** | Temperatura leída del potenciómetro |
| **Input** | Salida de la función **simPlanta()** |
| **Output** | Señal realimentada que entra a la planta |

Cuadro III
DEFINICIÓN DE LOS PARÁMETROS DEL CONTROLADOR.

| Parámetro | Valor |
|---|---|
| $\mathbf{K_p}$ | 15 |
| $\mathbf{K_i}$ | 0.5 |
| $\mathbf{K_d}$ | 1 |

### *II-D. Comunicaciones*

Otro de los objetivos del laboratorio era el poder trabajar con comunicaciones seriales con el protocolo USART para poder conectar, de manera virtual, la placa de Arduino con un script de Python que recopilara los datos de setpoint, entrada y salida del PID con el fin de graficarlos. Para inicializar en el código que se quería establecer una comunicación con un ambiente externo basta con establecer la frecuencia de envío de datos (baud rate) en la función de `setup`:

```
void setup() {
  Serial.begin(9600);
}
```

Esto habilita automáticamente la comunicación mediante el puerto serial del Arduino y también permite, mediante funciones dentro de la función `loop` imprimir el valor de variables en el monitor serial el MCU y por ende, enviarlos al script de Python. Este script hace uso de la librería **Serial**, la cual recibe como parámetros le puerto del cual recibirá los datos y el baud rate, el cual debe coincidir con el establecido en el código de Arduino.

Para poder hacer la conexión virtual desde un ambiente Windows, existen varios métodos pero el utilizado para este laboratorio es un software que se encuentra de manera gratuita en el siguiente link: **(Virtual Serial Port Tools)**). Por medio de este, se pueden definir un "puente local" seleccionando los puertos que se desean utilizar. Es decir, si se desea conectar la placa de Arduino del simulador con un script de Python y creo un puente entre los puertos **COM1** y **COM2**, quiere decir que a **COM1** debo conectar el puerto serial del Arduino y al scrip de Python, indicarle que recibirá datos mediante el puerto **COM2**.

## III. DESARROLLO/ANÁLISIS DE RESULTADOS

### *III-A. Hardware*

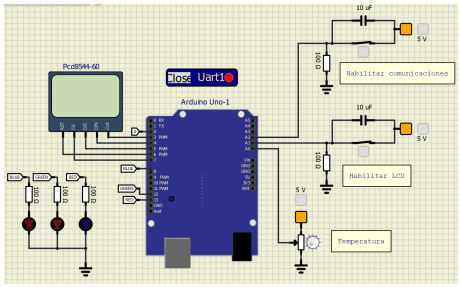


Figura 5. Diseño final de la Incubadora.

El diseño final muestra dos interruptores que se diseñaron específicamente para habilitar o deshabilitar la pantalla LCD y las comunicaciones seriales respectivamente; conexión analógica con el potenciómetro que actua como el regulador de la temperatura de la incubadora; tres LEDs (azul, verde y rojo) como indicadores del rango de temperaturas deseadas para la aplicación, que tal y como se indicó antes es de 30 a 42 °C; y por último, la pantalla LCD cuyas conexiones con la placa ya fueron establecidas en la sección anterior.

A continuación se muestra una tabla de los componentes requerido para este diseño, la cantidad de unidades requeridas y su precio. Este precio es un aproximado y se basó en una búsqueda en **Amazon** para establecerlos:

Cuadro IV
COMPONENTES ELECTRÓNICOS NECESARIOS PARA EL DISEÑO

| Componente | Cantidad | Precio (Unidad) |
|---|---|---|
| Arduino 1 | 1 | $27.60 |
| Resistencia 100 Ω | 5 | $5.99 (100 pcs) |
| LED Rojo | 1 | $6.99 (100 pcs) |
| LED Azul | 1 | $6.99 (100 pcs) |
| LED Verde | 1 | $6.99 (100 pcs) |
| PCD8544 | 1 | $5.90 |
| Potenciómetro 1 kΩ | 1 | $15.99 (2 pcs) |
| Capacitor 0,1 µF | 2 | $5.68 (25 pcs) |
| Switch | 2 | $6.98 (12 pcs) |

Elaborado el diseño en el simulador de cómo estará construida la incubadora, se procede a explicar el funcionamiento de esta según lo realizado en código de Arduino. Primeramente, se trabajó con la implementación del potenciómetro para regular la temperatura de la incubadora por medio de este; se escogió un potenciómetro de 1 kΩ y se conectó este al pin `A0` de la placa. El pin `A0` es una de las entradas analógicas del Arduino y cuenta con un convertidor Analógico/Digital de 10 bits, por lo tanto, hay que asegurarse que cuando el potenciómetro se encuentra en su valor máximo se debe de leer un valor de 1023 de la entrada establecida y 0 cuando se encuentra en el mínimo, sin embargo, hecha esta prueba se encontró que en el punto máximo del potenciómetro se miden 1022, por lo tanto, este rango de valores debe de mapearse a los valores d etemperatura deseados, es decir, de 20 a 80 °C, de la siguiente manera haciendo uso de la función **map()** de Arduino:

```
void loop() {
  int value = analogRead(A0);
  // Incubadora: [20, 80] °C
  float temp = map(value, 0, 1022, 20, 80);
}
```

Tal como se observa, se guarda el valor mapeado en una variable llamada **temp**, que es donde se almacenará el valor de temperatura que se desea para la incubadora.

Una vez hecho esto se debe poder observar la temperatura en la pantalla LCD, por lo que se procede a inicializarla. Para hacer esto se debe de crear una instancia de la pantalla de librería **PCD8544** al inicio del código e inicializarla en la función de **setup()**; en dicha función se puede establecer el tamaño de pantalla con el que se desea trabajar, sin embargo, las dimensiones por defecto son de 84x84 y quedaría lista para recibir caracteres en la función de **loop()**, tal como se muestra a continuación:

```
#include <PCD8544.h>

// Implementación de la pantalla
PCD8544 lcd;

void setup() {
  // Activo el LCD
  lcd.begin(); // default is 84x48
}

void loop(){
  // Mostrar en LCD
  lcd.setCursor(0, 1);
  lcd.print("SP: "); lcd.print(temp);
}
```

Habilitada la pantalla LCD y el potenciómetro como mecanismo para regular la temperatura, se procede a trabajar con el controlador PID. Este controlador, tal como se explicó en la Nota Teórica, es utilizado para regular algún proceso y llevarlo a un punto de operación deseado, en el caso de este proyecto en particular se modela la incubadora como la función **simPlanta(float Q)** la cual recibe un único parámetro y es el valor de temperatura que arroja el controlador PID y devuelve un valor de temperatura en grados celcius (°C) que es la temperatura que se tiene en la incubadora. La forma de implementar esto no requiere de hardware pues todo está empaquetado en la librería **PID** y en la función que modela la planta, por lo tanto, todo esto se traduce a código Arduino inicializando el PID y pasándole como parámetros las variables de entrada, salida y de setpoint, así como los parámetros de control, todo esto discutido en la sección anterior.

Por otra parte, la señal de control de limitó para poder calentar y enfriar la incubadora, por lo que debía alcanzar valores positivos y negativos. Esto se definió en la función de **setup()**:

```
myPID.SetOutputLimits(-125, 125);
```

Con estos valores se asegura que se pueda calentar de una manera no muy lenta y enfriar en igual medida, la forma de elegir estos valores fue arbitraria, se hizo a prueba y error y los valores escogidos fueron los que arrojaron un desempeño óptimo.

De la misma manera que con el potenciómetro, los resultados ahora obtenidos gracias al PID se pueden observar en el LCD y se muestra un ejemplo de esto en la siguiente figura:

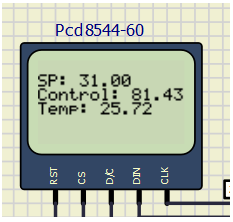


Figura 6. Pantalla LCD mostrando la información solicitada de control.

Se solicitaba también implementar tres LEDs: rojo, verde y azul, para poder monitorear que la incubadora se encontrara en el rango de operación requerido para esta aplicación [30, 42] °C. Si el equipo se encuentra por debajo de ese rango se debe de iluminar el LED azul, si se encuentra dentro se ilumina el LED verde y si es mayor iluminar el LED rojo, tal como se muestra en la figura 7.

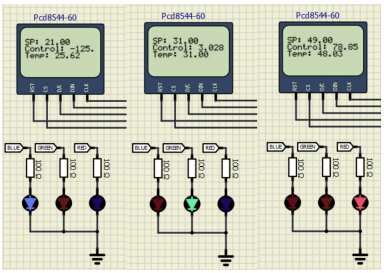


Figura 7. LEDs de aviso encendidos en condiciones distintas de la Incubadora.

Para implementar esto en código basta con definir un bloque de **if** y tomar los valores de temperatura de la incubadora según corresponda.

Por último, se debía de poder interrumpir la comunicación con la pantalla LCD y con el puerto serial para poder enviar los datos a un script de Python. Para la primer tarea se creó una función que tomaba el valor analógico del puerto `A1` y cuando este es cercano a $5\,\mathrm{V}$, se enciende la pantalla y la apaga en el caso contrario:

```
void LCD_state(){
  int switchState = analogRead(A1);
  if (switchState > 700){
    lcd.setPower(1);
  } else{
    lcd.setPower(0);
  }
}
```

Para las comunicaciones del puerto serial se utilizó el mismo mecanismo pero tomando como interruptor la entrada analógica `A2` y se definió que si la señal proveniente de ese puerto es cercana a $5\,\mathrm{V}$ entonces que se envía por medio de **Serial.println()** la información del PID que se mencionó previamente.

Como última parte está la recolección de datos mediante un archivo de Python. Para esto el archivo es bastante simple: toma los datos provenientes desde el puerto **COM2** y los almacena en variables separadas, posteriormente, con un archivo **.csv** abierto comienza a tomar datos cada $0{,}5\,\mathrm{s}$ y una vez que se detenga la comunicación mediante el comando **Ctrl + C** se guardan todos los datos recolectados al script. A continuación se mostrarán las gráficas obtenidas mediante los datos recolectados por este método y adicional este **LINK** de YouTube en donde se puede ver el programa en funcionamiento:

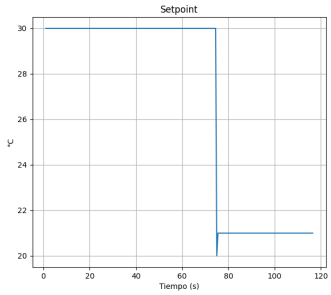


Figura 8. LEDs de aviso encendidos en condiciones distintas de la Incubadora.

La figura 8 muestra como se cambió manualmente el setpoint de la incubadora, iniciando en una temperatura de $30\,^{\circ}\mathrm{C}$ y luego bajando a $21\,^{\circ}\mathrm{C}$. Todo esto fue hecho mediante la manipulación del potenciómetro gracias a toda la configuración mencionada anteriormente.

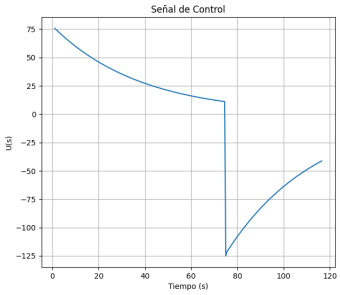


Figura 9. LEDs de aviso encendidos en condiciones distintas de la Incubadora.

Esta figura (figura 9) corresponde a los valores de la señal de control, esta inició desde un valor alto de 75 para poder llevar la temperatura de la planta al valor fijado mediante el setpoint y se estabiliza cerca de cero una vez alcanzado dicho valor. En el instante en que se llevó el setpoint hasta $21\,^{\circ}\mathrm{C}$ entonces ahora la incubadora debe de enfriarse, para esto la señal de control llega instantáneamente hasta su menor valor que es -125 para llevar la temperatura al punto fijado de forma rápida y posteriormente regularse, intentando estabilizarse cerca de cero.

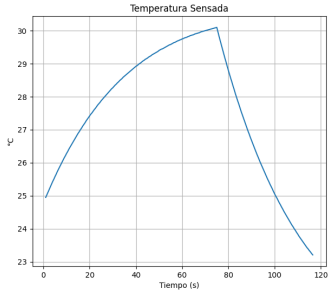


Figura 10. LEDs de aviso encendidos en condiciones distintas de la Incubadora.

Por último se tiene a la temperatura, que iniciando en $25\,^{\circ}\mathrm{C}$, que es la temperatura del ambiente comienza a subir hasta llegar a $30\,^{\circ}\mathrm{C}$ que es el valor fijado mediante el potenciómetro y estabilizarse aquí pero es en este instante cuando se lleva el setpoint a $21\,^{\circ}\mathrm{C}$ y de manera lenta comienza a bajar la temperatura para estabilizarse en el nuevo punto de operación.

Este comportamiento es el esperado con este sistema térmico por lo que se puede afirmar que el programa cumple el comportamiento deseado.

## IV. Conclusiones y Recomendaciones

La implementación de la incubadora, así como todos los requerimientos según el enunciado del laboratorio fueron alcanzados de manera exitosa. Se pudo trabajar con la pantalla LCD, con las comunicaciones seriales, implementar un controlador PID y modelar la incubadora sin ningún tipo de inconveniente.

Se concluye también que el proceso térmico acá trabajado es un proceso lento que toma un tiempo en estabilizarse pero que una vez que lo alcanza, permanece ahí hasta que otro cambio en el setpoint sea generado. Esto puede ser visto con una habitación y un aire acondicionado: una vez que se enciende el aire acondicionado en un día caluroso le demora unos minutos poder enfriar la habitación a cualquiera que sea la temperatura establecida mediante el control remoto, no obstante, sabemos que la habitación, eventualmente, se enfriará. Pasa lo mismo con este modelado de incubadora.

A modo de recomendación se puede mencionar consultar foros de internet como ayuda para Arduino pues es muy utilizado y hay mucha información. También se recomienda ver los ejemplos de las librerías utilizadas para comprender mejor cómo implementarlas en nuestro código.

## Referencias


[1] Arduino. Arduino UNO R3: Product Reference Manual. https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf, 2024.

[2] Pardo, C. Controlador PID. https://www.picuino.com/es/control-pid.html, 2024.

[3] Rodrigues, C. and Puliaiev, Y. and Tuma, O. and Pfleger, M. PCD8544. https://github.com/carlosefr/pcd8544, 2020. GitHub.