Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0624 Laboratorio de Microcontroladores

PID,GPIO,ADC, comunicaciones USART y SPI

Ricardo Hidalgo Campos B63464
Wilber Hernández Ruiz B13257

8 de Mayo del 2024

1. Introducción

Este laboratorio trata de la implementación de un microcontrolador Arduino UNO para el control de una incubadora el cual medirá la temperatura y comunicara los datos mediante una pantalla y guardara registro de dichos cambios de temperatura. La pantalla es una LCD PCD8544, esta mostrara la temperatura y la temperatura adecuada, además se utilizaron materiales extra como una resistencia para calentar la incubadora y LED's de aviso para notificar si bajo mucho la temperatura o esta alta. Todo esto viene conectado a consola mediante el protocolo USART el cual crea el archivo de informe. Este laboratorio busca crear el aprendizaje de la manipulación de controladores PID y el uso del protocolo USART, el cual cumple con dichos objetivos.

2. Nota Teórica

El microcontrolador ATmega328P posee varias características que lo hacen ser, algunas de las mas importantes son las siguientes:

- Microcontrolador AVR de 8-bits.
- Arquitectura RISC.
- 32 bytes de memoria flash.
- Dos contadores de 8-bits y uno extra de 16-bits.
- Seis canales PWM.
- USART programable.
- 8 canales de 10-bits ADC.
- interruptores internos y externos.
- Voltaje de operación entre los 2.7V y 5.5V.

A continuación se hará una breve descripción de algunos de los pines mas relevantes:

- Port B: Puerto de 8-bits bidireccional que permite el control de estos. Dependiendo de la configuración del reloj el puerto B6 puede funcionar como entrada para un amplificador de oscilaccion inversa y B7 como su salida.
- Port C: Puerto de 7-bits bidireccional que permite el control de estos.
- PC6/RESET: Este puerto puede ser configurado para ser un puerto PC6 de entrada, pero si no es configurado viene a ser un RESET.
- Port D: Puerto de 8-bits bidireccional que permite el control de estos.
- AREF : es un pin para el A/D convertidor.

Ahora veremos el diagrama de bloque del funcionamiento del ATmega328P

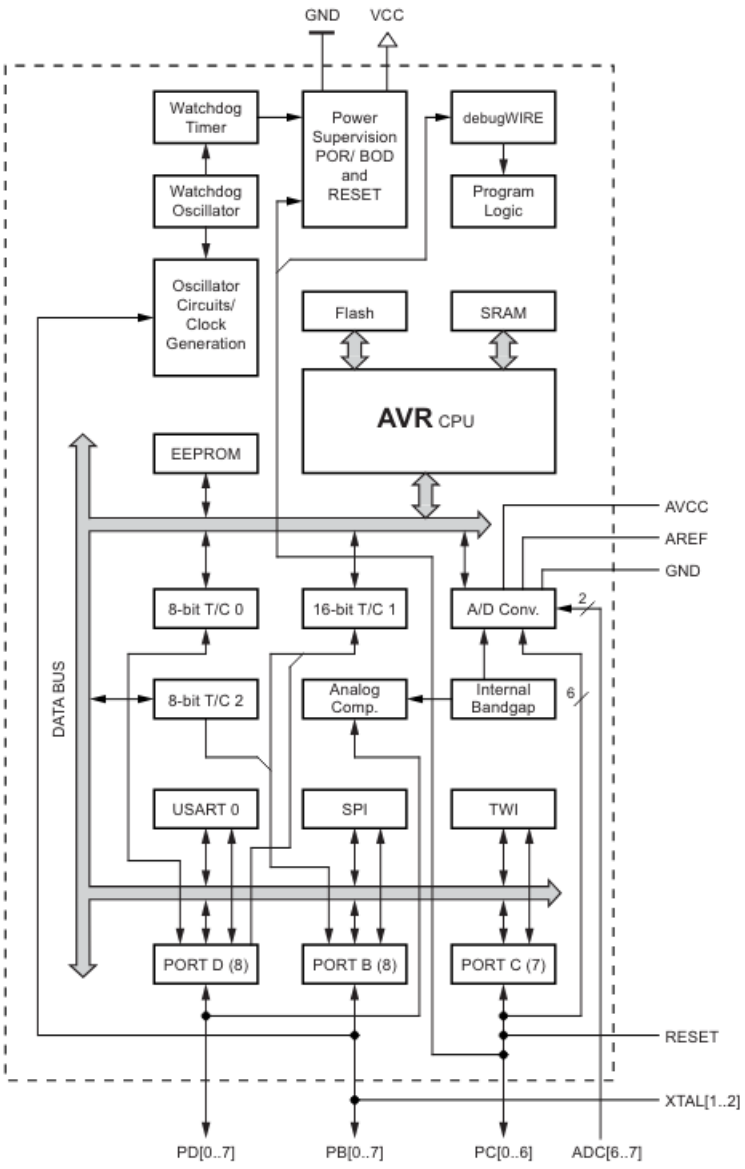

Figura 1: Diagrama de bloques del ATmega328P.

El registro de propósito general tiene las siguientes características:

7	0	Addr.	
R0		0x00	
R1		0x01	
R2		0x02	
...			
R13		0x0D	
R14		0x0E	
R15		0x0F	
R16		0x10	
R17		0x11	
...			
R26		0x1A	X-register Low Byte
R27		0x1B	X-register High Byte
R28		0x1C	Y-register Low Byte
R29		0x1D	Y-register High Byte
R30		0x1E	Z-register Low Byte
R31		0x1F	Z-register High Byte

Figura 2: Registros de uso general del microcontrolador ATmega328P.

Segun la hoja del fabricante, estos son los valores de operacion del microcontrolador:

Parameters	Min.	Typ.	Max.	Unit
Operating temperature	–55		+125	°C
Storage temperature	–65		+150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	–0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to ground	–0.5		+13.0	V
Maximum operating voltage		6.0		V
DC current per I/O pin		40.0		mA
DC current V_{CC} and GND pins		200.0		mA
Injection current at V_{CC} = 0V		±5.0(1)		mA
Injection current at V_{CC} = 5V		±1.0		mA

Note: 1. Maximum current per port = ±30mA

Figura 3: Tabla de operación del microcontrolador ATmega328P.

Este microcontrolador estará unido a una placa Arduino UNO por la cual se conectara a sus puertos de forma indirecta mediante los puertos de alto nivel que posee dicha placa. Esta información es esencial para la operación del microcontrolador y el uso adecuado de este, así como el diseño en la implementación de este.

3. Análisis de resultados

Los componentes utilizados para el laboratorio fueron los siguientes:

Componente	*Cantidad*	*Precio(colones)*
Arduino UNO	1	20000
LED	3	60
Resistencia	3	100
Capacitor	2	200
Potenciometro	1	1000
LCD PCD8544	1	4500

Cuadro 1: Componentes para la realización del laboratorio.

El diseño en el simulador de la construcción de la incubadora con los materiales previamente estipulados se mostrara mas adelante, primero se simulo cuando la temperatura supera los 42 grados donde se debe mostrar una luz LED roja como en la siguiente imagen.

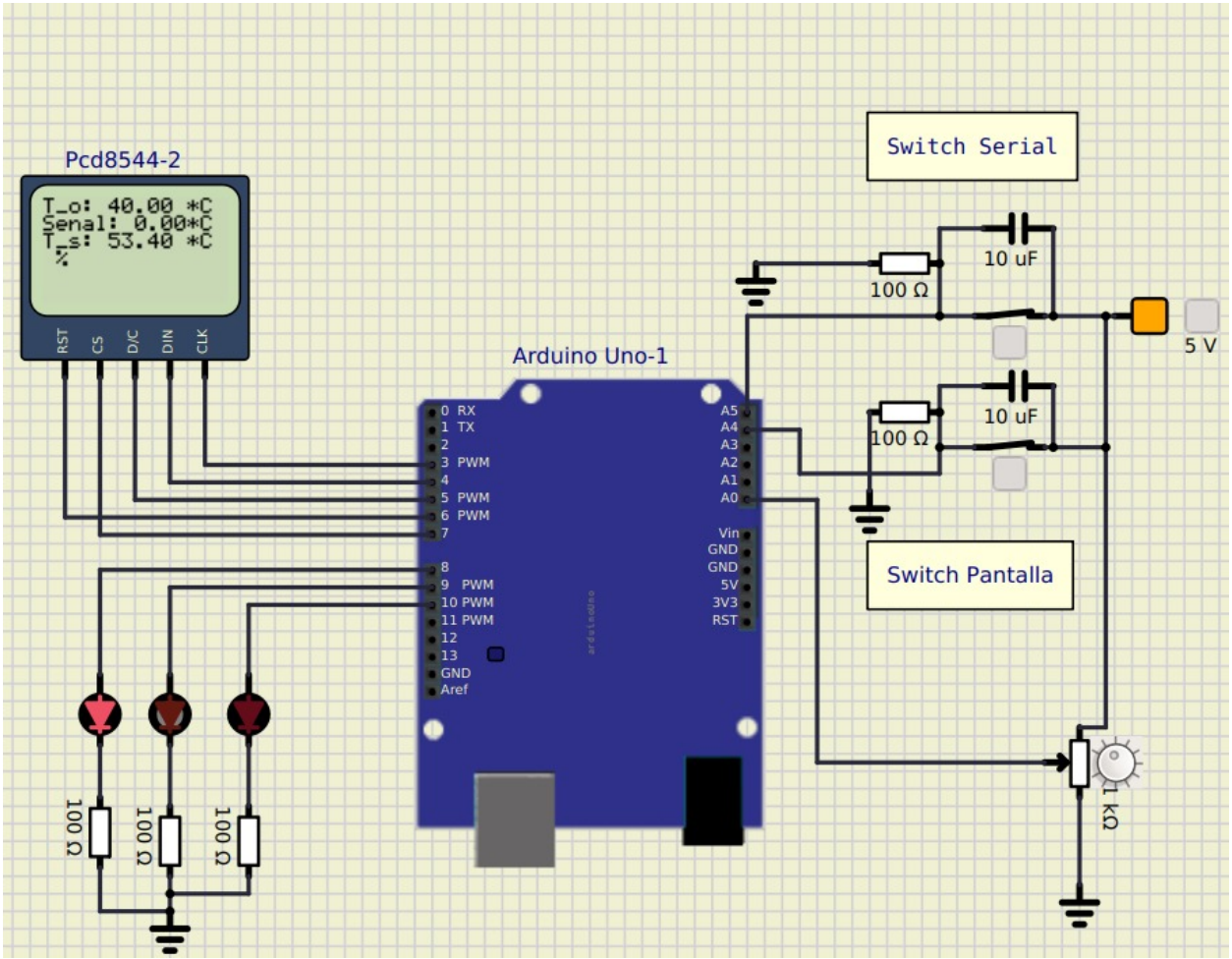

Figura 4: Simulación de de la incubadora mas de 42 grados.

Se observa en la simulación como se detecta la temperatura y como reacciona el controlador para dar la señal de alerta. luego se probo el caso donde la temperatura fue inferior a los 30 grados en la siguiente simulación:

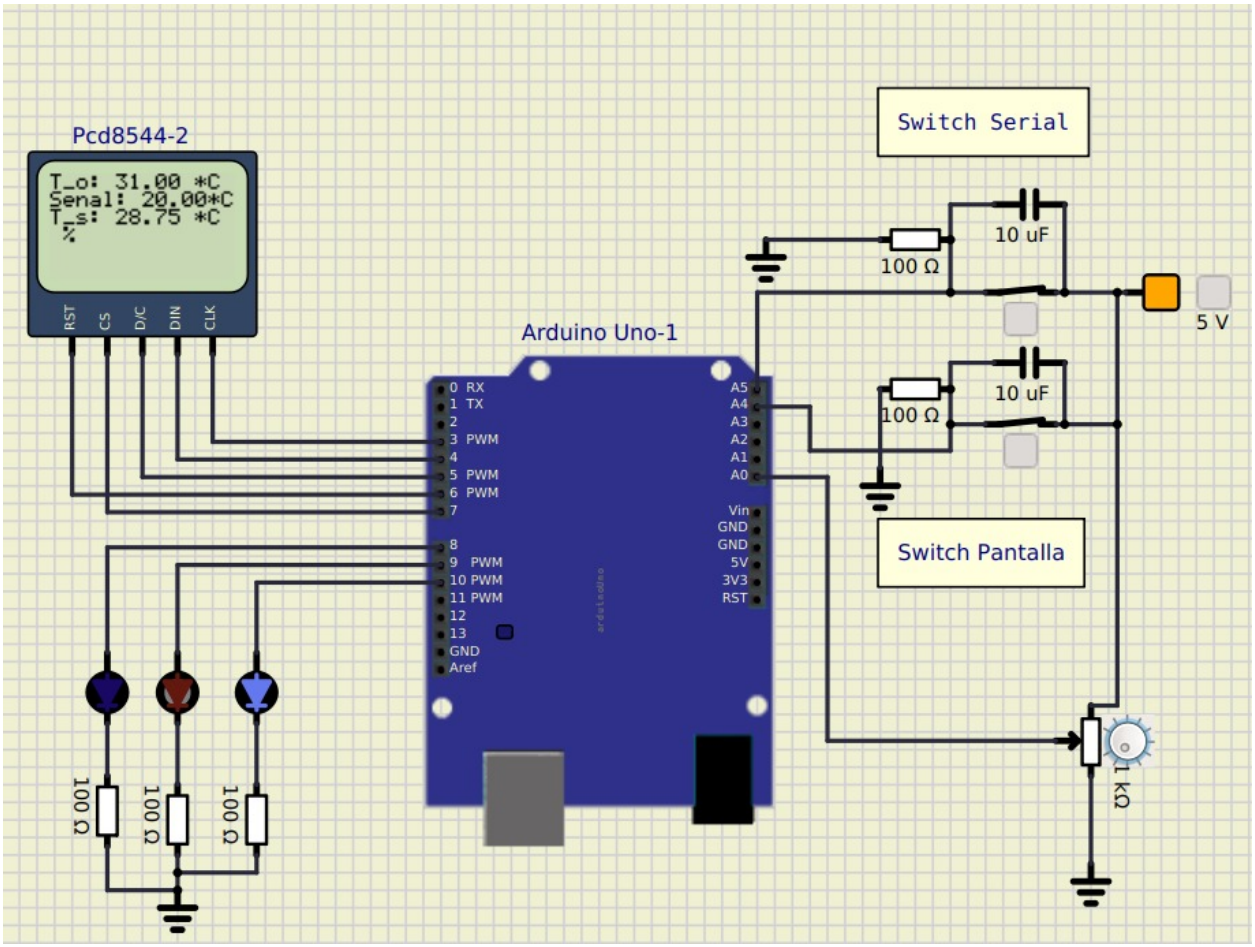

Figura 5: Simulación de incubadora a menos de 30 grados.

Igual aquí, observamos como el comportamiento es el esperado de acuerdo a las especificaciones deseadas y se enciende la luz azul cuando la temperatura baja, demostrando que esta correctamente configurado.

Ahora mostraremos partes del código y como se realizo su implementación.

```
void setup()
{
    pid->Kp = 5;
    pid->Ki = 2;
    pid->Kd = 1;

    pinMode(sclk, OUTPUT);
    pinMode(sdin, OUTPUT);
    pinMode(dc, OUTPUT);
    pinMode(reset, OUTPUT);
    pinMode(sce, OUTPUT);
    pinMode(BLUE, OUTPUT);
    pinMode(RED, OUTPUT);
    pinMode(GREEN, INPUT);

    analogValue1 =0;
    analogValue2 =0;

    PIDControlador_Init(pid);
    pantalla.begin();
    Serial.begin(baudrate);
    setpoint = analogRead(potenciometro)*12/1000 +30;
    temperatura = PIDControlador_Update(pid, setpoint, temperatura_ambiente);
}
```

Figura 6: Código de implementación del setup.

EN esta parte del código, luego de definidas las variables anteriormente, es donde se define el setup, es decir, la inicialización de los pines y variables necesarias. Se puede observar como se inicializa el controlador PID, el cual nos dara el control necesario de la temperatura. SU codigo utilizado es el siguiente:

```
float simPlanta(float Q) {
    // simula un bloque de aluminio de 1x1x2cm con un calentador y con enf
    // float C = 237; // W/mK coeficiente de conductividad termica para el
    float h = 5; // W/m2K coeficiente de conveccion termica para el Alumir
    float Cps = 0.89; // J/gC
    float area = 1e-4; // m2 area por conveccion
    float masa = 10 ; // g
    float Tamb = 25; // Temperatura ambiente en C
    float T = Tamb; // Temperatura en C
    static uint32_t last = 0;
    uint32_t interval = 100; // ms

    if ( millis () - last >= interval) {
        last += interval;
        // 0-transferencia de calor
        T = T + Q * interval / 1000 / masa / Cps - (T - Tamb) * area * h;
    }

    return T;
}
```

Figura 7: Código de implementación del controlador PID.

Luego, donde se implementa la lógica de control para los LED's es en la función loop, en la cual se hacen las configuraciones para encendidos de LED's y configuración en pantalla.

```
void loop()
{
    analogValue1 = analogRead(switchPantalla);
    analogValue2 = analogRead(switchSerial);

    pantalla.clear();

    float TempWatts = senalControl * 20.0 / 255;
    temperatura = simPlanta(TempWatts);
    var_referencia = analogRead(potenciometro)*12/1000 +30;

    if(var_referencia!= setpoint){
        setpoint = var_referencia;
        senalControl = PIDControlador_Update(pid, setpoint, temperatura);
    }



    if(temperatura < 30){
        digitalWrite(BLUE, HIGH);
        digitalWrite(GREEN, LOW);
        digitalWrite(RED, LOW);
    }else{
        if(temperatura > 42){
            digitalWrite(RED, HIGH);
            digitalWrite(BLUE, LOW);
            digitalWrite(GREEN, LOW);
        }else{
            if(temperatura <=42 && temperatura>=30){
                digitalWrite(GREEN, HIGH);
                digitalWrite(BLUE, LOW);
                digitalWrite(RED, LOW);
            }

        }
    }
```

Figura 8: Código de implementación de la lógica para funcionamiento.

Aquí observamos como se utilizan las variables del controlador PID, así como mediante una lógica secuencial se implementa la configuración para los LED's. Luego de esto se programa el uso de Usart.

```
    if(analogValue1 != 0){
        desplegarDatosPantallaLCD(setpoint,senalControl,temperatura);
    }

    if(analogValue2 != 0){
        enviarDatosUSART(setpoint,senalControl,temperatura);
    }



    delay(10);
}
```

Figura 9: Código de implementación del llamado a Usart.

Luego, tenemos la implementación del control PID y como se utilizan sus diferentes funciones para hacer las conversiones y cálculos necesarios a partir de la toma de temperatura.

```
float PIDControlador_Update(PIDControlador *pid, float set_point, float medicion) {


    //Leer entrada
    pid->posicion_actual = medicion;
    //Calcular el error
    pid->error = set_point - pid->posicion_actual ;
    //Calcular la integral
    pid->integral = pid->integral + pid->error ;
    //Calcular la derivada
    pid->derivada = pid->error - pid->ultimo_error ;
    //Calcular variable de control
    pid->var_control = (pid->Kp * pid->error ) + ( pid->Ki * pid->integral ) + (pid->Kd * pid->derivada ) ;
    //Acotar variable de control
    if( pid->var_control > 255){
        pid->var_control = 255;
    }
    else{
        if( pid->var_control < 0){
            pid->var_control = 0;

        }
    }
    pid->ultimo_error = pid->error;
    return pid->var_control;
}
```

Figura 10: Código de implementación de las funciones de PID.

Se observa como el codigo esta estructurado y como se implemento el lazo cerrado, como con el controlador

PID fue posible esto en el simulador y nos permitio obtener los resultados anteriormente mostrados, haciendo que este codigo y la configuracion del microcontrolador mediante la placa de Arduino UNO se implementara de manera correcta, sin inconvenientes.

4. Conclusión

En conclusión, La utilización del microcontrolador ATmega328P implementado mediante una placa Arduino UNO nos permitió lograr un control sobre la temperatura de la incubadora, detectando la temperatura y señalando las diferentes fases de acuerdo a esta. La configuración propuesta en la simulación funciono, respondió a lo deseado, demostrando así que la programación fue la adecuada, aplicando y utilizando correctamente cada pin de la placa, haciendo uso correcto del control PID para tomar la temperatura y aplicar la lógica necesaria para lograr el objetivo.

5. Repositorio

El repositorio de este laboratorio se encuentra en el siguiente enlace:
`https://github.com/kardo05/Laboratorio_Microcontroladores/tree/main`.

Referencias

Atmel. 8 bit AVR microcontroller with 32k bytes in-system programmable flash. `https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf`.