Simulación control de temperatura para incubadora de huevos Arduino - Proteus

Cesar Alejandro Roa Palomino
Luisa Maria Correa Mosquera

Circuitos Digitales II

Universidad del cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Introducción a las Redes de Telecomunicaciones

Marco teórico

Los microcontroladores son circuitos integrados programables, que son capaces de ejecutar órdenes que están almacenadas en su memoria, contiene dentro de sí mismo una unidad de procesamiento central, memoria y periféricos de entrada y salida.

Muchos microcontroladores son muy adecuados para aplicaciones de batería de larga duración, ya que al encontrarse en estado de reposo en espera de una interrupción o interacción, el consumo de energía puede ser de nanowatts. También hay otros microcontroladores que sirven para roles de rendimiento crítico, donde actúan más como un procesador digital de señales, con velocidades de reloj y mayor consumo de energía.

El arduino es una placa que tiene todos los elementos necesarios para conectar sensores,variables eléctricas,dispositivos como motores,etc a las entradas y salidas de un microcontrolador. Es decir, es una placa impresa con los componentes básicos para que funcione el microcontrolador y se comunica con un ordenador a través de la comunicación serial.

Tiene una interfaz de entrada que se encarga de trasladar información y procesarla dependiendo de los requisitos del proyecto dependiendo del tipo de placa de arduino. Dado que tiene una gran variedad de fabricantes y versiones.

También tiene una interfaz de salida la cual se encarga de llevar esta información procesada a los periféricos o dispositivos correspondientes de estos datos, o puede estar conectada a otra placa en algunos casos para que esta información sea transformada o tratada en otro tipo totalmente distinta.O al final es una pantalla que muestra los datos originales de otra manera.

Arduino se basa en C++, que es un lenguaje de programación de alto procesamiento muy conocido. Es un lenguaje de programación orientado a objetos muy potente que procede del lenguaje c o evolucionó de él, hoy en día este lenguaje aún se sigue usando para realizar programación estructurada de alto nivel y rendimiento.

Tiene un estándar ISO, conocido como ANSI-C++ y la última versión estandarizada fue la del 2011, desde ahí, se comenzó a desarrollar nuevas versiones en ciclos de 3 años. Actualmente está vigente la versión C++20 y se está trabajando en la C++23.

INTRODUCCIÓN

Existen varios animales ovíparos, de los cuales nos puede interesar asegurar su reproducción, como lo son, algunos en peligro de extinción. Para este posible caso, hemos pensado en crear una incubadora para huevos, que tenga un control de temperatura, manteniéndola estable en una temperatura o un rango de temperatura predeterminado que sea propicio para la incubación y correcta eclosión de los huevos.

Un ejemplo de incubación es para los huevos de gallina dado que estos para una incubación óptima se necesita que estén en un rango de temperatura entre 37 y 41 grados celsius. Estos huevos deben estar en la incubadora entre 19-24 días antes de eclosionar.

El sistema de control de temperatura se programa en Arduino, con una tarjeta Arduino Mega, y se simula en Proteus.

DESARROLLO DEL PROYECTO

Modelo o esquemático del proyecto:

En el desarrollo del proyecto hemos planteado el funcionamiento de la forma que se explica en el siguiente gráfico de diagrama de bloques.

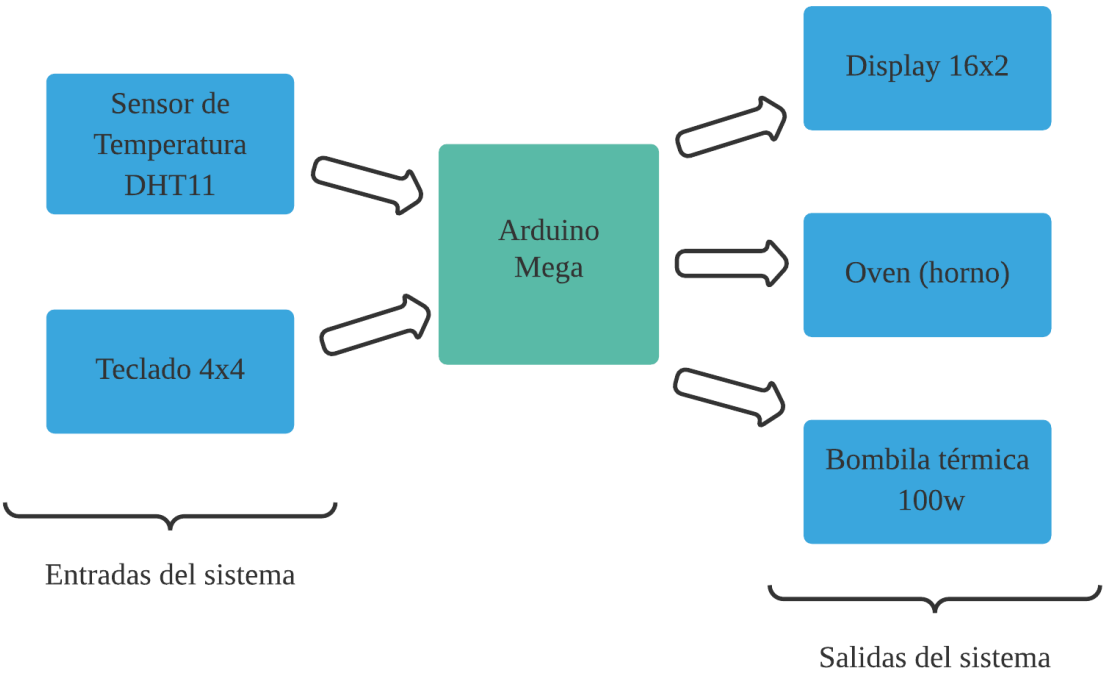

Fig.1 diagrama de bloques

Tenemos 5 bloques, de los cuales dos son de entrada y tres de salida. En los de entrada tenemos un teclado 4x4, con el cual el usuario fijará la temperatura deseada. También está el sensor de temperatura y humedad DTH11, que va a medir la temperatura del ambiente y con esta información se decide la activación del calentamiento.

En los bloques de salida tenemos un display de 16x2, que se usará para interactuar con el usuario y donde se mostrará los datos de temperatura actuales y los ingresados desde el teclado. Luego tenemos el oven(horno) que se usa para calentar la temperatura del ambiente, seguido de una bombilla térmica de 100w, que tendrá el mismo objetivo.

Arduinos - Proteus:

Nuestro diseño tiene un botón que permite decidir qué tipo de funcionamiento queremos que tenga. El primer funcionamiento sucede con el botón en HIGH, sin presionar, y en este caso se activa el oven, con una temperatura predeterminada en el código de 38°C, que es la temperatura de incubación de los huevos de gallina. Este se controla con PID y con configuración del elemento dentro de Proteus, (por esto el usuario no podrá fijar una temperatura deseada). El segundo funcionamiento sucede con el botón en LOW, presionado, y en este caso se le pide al usuario que fije una temperatura deseada, con la cual se definen límites mínimos y máximos. Luego se leería la temperatura desde el DHT11, y dependiendo de donde se encuentre se enciende o se apaga la bombilla térmica (que en esta simulación se representa con un led rojo).

Cabe aclarar que si se cambia el estado del botón, mientras ya está funcionando, no se podrá cambiar el tipo de funcionamiento, ya que el botón se lee solo al momento de encendido.

Ahora presentamos el esquema eléctrico del sistema, en Proteus:

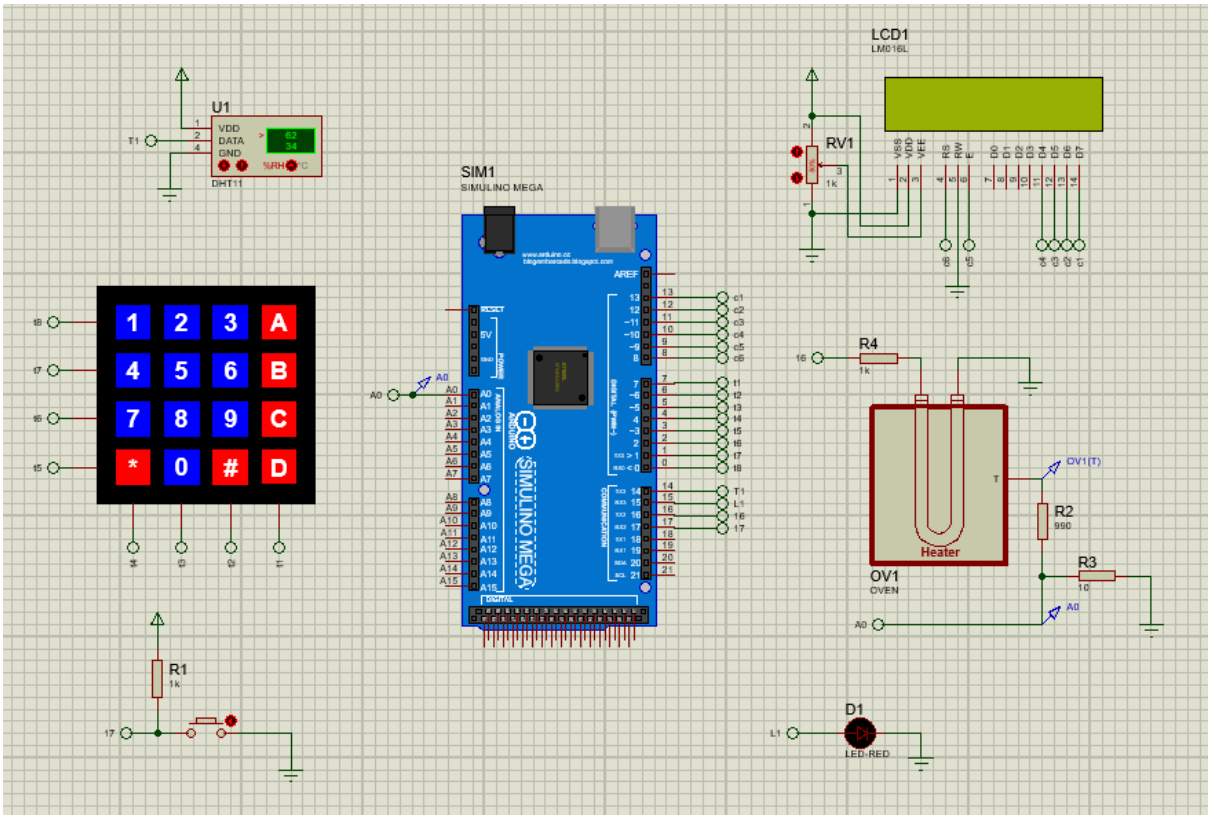

Fig.2 simulación del prototipo en proteus

En la izquierda del arduino, tenemos las entradas del sistema, que son, desde arriba hacia abajo, el sensor DHT11, el teclado 4x4, y el botón. Y en la derecha, tenemos las salidas del sistema, está el LCD 16x2, el oven(horno) y la bombilla térmica.

A continuación, el código final de Arduino:

```
//librerias
#include <DHT.h>
#include <Key.h>
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <PID_v2.h>

//pines del lcd
LiquidCrystal lcd(8,9,10,11,12,13);
```

Fig.3 librerías de los periféricos,el sensor y la función PID

Para comenzar, incluímos las librerías que necesitamos. Primero DHT.h para el sensor DHT11. El Key.h y Keypad.h para el teclado 4x4. El LiquidCrystal.h para el LCD 16x2. Y el PID_v2.h para el control PID.

Luego configuramos el LCD en los pines digitales, 8,9,10,11,12,13.

```
//pines y configuracion del keypad
const byte FILAS = 4;
const byte COLUMNAS = 4;
char TECLAS[FILAS][COLUMNAS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte PinFilas[FILAS] = {0,1,2,3};
byte PinColumnas[COLUMNAS] = {4,5,6,7};
Keypad keypad = Keypad(makeKeymap(TECLAS), PinFilas, PinColumnas, FILAS, COLUMNAS);
```

Fig.4 creación del teclado 4x4

Ahora se configuran el teclado y sus pines. También se crea el mapa, con filas x columnas.

```
//pin y configuracion del sensor DHT11
int sensor=14;
DHT dht (sensor, DHT11);

//pines y constantes para el control del oven
#define PIN_INPUT 0
#define PIN_OUTPUT 16
double Setpoint, Input, Output;
double Kp = 10, Ki = 1, Kd = 0;
float tempC = 0;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
```

Fig.5 creación del control PID temperatura

Se configura el sensor en el pin 14, y el oven usa el pin 16 de salida y el pin A0 de entrada. Se crean las constantes y el control PID.

```
//pin boton
const int pinbutton = 17;

//inicializacion de variables
int temperaturaIngresada[2] = {0};
int temperatura = 0;
int digito = 0;
int temp = 0;
int cont = 0;
int t1 = 0;
int t2 = 0;
int i = 0;
int button;
```

Fig.6 declaración e inicialización de distintas variables

El botón utiliza el pin 17. Inicializamos las variables que vamos a utilizar durante todo el código.

```
void setup() {
  dht.begin();
  lcd.begin(16,2); //inicializa el display de 16x2
  pinMode(pinbutton, INPUT); //configuro el boton como entrada

  //codigo de una sola vez para el oven
  tempC = analogRead(PIN_INPUT) * 0.0048828125 * 100;
  Input = tempC;
  Setpoint = 38;
  myPID.SetMode(AUTOMATIC);

  button=digitalRead(pinbutton); //lectura del boton
  if(button == HIGH){
    lcd.setCursor(0, 0);
    lcd.print("Botón sin");
    lcd.setCursor(0, 1);
    lcd.print("presionar");
    fijar_temp(); //fijacion de temperatura para el primer tipo de funcionamiento
  }
    else{
      lcd.setCursor(0, 0);
      lcd.print("Botón");
      lcd.setCursor(0, 1);
      lcd.print("presionado");
    }

}
```

Fig.7 programación en el void setup

En el setup iniciamos el sensor, el botón, y configuramos el botón como entrada. Luego está el código de encendido del oven, donde le damos los parámetros de temperatura y PID que necesita para funcionar.

Por último, en el condicional, leemos el botón por primera vez, se imprime el estado del botón en el LCD y se decide si se necesita fijar una temperatura o no.

```
void loop() {
  lcd.clear();
  lcd.setCursor(0,0);
  //funcion condicional para decidir el tipo de funcionamiento
  if(button == HIGH){
    //llamado a la funcion que controla el led y la lectura del DHT11
    tem_act_led();

  }
    else {
      //llamado a la funcion que controla la temperatura en el horno
      temp_oven();
    }

}
```

Fig.8 Programación en el void loop

Para el loop primero limpiamos el LCD y fijamos el cursor en el inicio de este. Luego leemos el botón por segunda vez y decidimos el funcionamiento que se usará. Para el botón en HIGH, llamamos la función de control de encendido y apagado de la bombilla térmica. Y para el botón en LOW, se llama la función de control del oven con PID.

```
//funcion para fijar la temperatura deseada en el teclado
void fijar_temp(){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Fijar temp: ");
  do{
     lcd.setCursor(0, 1);
     digito = keypad.waitForKey(); //se esperra a que se ingrese un valor
      if (digito == '*'){  //si el valor ingresado no es un numero se imprime vacio
        lcd.print("                ");
        lcd.setCursor(0,1);
        cont = 0;
      }
      //Cuando se ingresa un digito, se resta 48 (ver código ASCII) y se almacena
        else{
          digito -= 48;
          temperaturaIngresada[cont] = digito;
          lcd.print(digito);

          cont++;
            if( cont == 2 ){
              //se consigue la temperatura ingresada como un solo valor
              temperatura=(temperaturaIngresada[0]*10)+temperaturaIngresada[1];
              t1=temperatura-2;  //temperatura minima
              t2=temperatura+2;  //temperatura maxima

            }

        }
    i++;
  }while(i<2);

}
```

Fig.9 programación de la función Fijar temperatura

La primera función que tenemos es la de fijación de temperatura que se usa para el primer estado del botón. Esta función se llama en el condicional del setup si se lee que el botón está en HIGH.

Primero se le pide al usuario, a través del LCD, que fije una temperatura. El sistema espera a que se ingrese un valor en el teclado, se hace una comprobación del valor ingresado en el condicional y si es un número se transforma a entero basados en el código ASCII y se almacena en un vector. Luego de que se reciban 2 dígitos, se convierte en un solo valor para luego determinar los límites mínimo y máximo, y de está forma generar el rango de temperatura con el que se va a controlar el encendido y apagado de la bombilla térmica.

```
//funcion para leer la temperatura desde el dht11 y variar el led
void tem_act_led(){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp deseada: ");
  lcd.print(temperatura);
  delay (1000);
  temp=dht.readTemperature();  //se lee la temperatura desde el dht11
  lcd.setCursor(0, 1);
  lcd.print("Temp actual: ");
  lcd.print(temp);

  delay(500);

  //Si la temperatura es menor o igual que el min de temp
  if(temp<=t1){
    digitalWrite(15,HIGH); //la bombilla se enciende
  }
  //Si la temperatura es mayor o igual que el max de temp
  if(temp>=t2){
    digitalWrite(15,LOW); //la bombilla se apaga
  }
}
```

Fig.10 programación de la activación o el apagado del led

La siguiente función imprime en el LCD, la temperatura deseada, que fue la ingresada, lee la temperatura desde el DHT11 y la imprime como la temperatura actual, y finalmente con el condicional dependiendo de la temperatura se decide si se enciende o se apaga la bombilla.

```
//funcion de control del temperatura e impresion con el horno
void temp_oven(){
  //correccion temperatura en oven
  tempC = analogRead(PIN_INPUT) * 0.0048828125 * 100;
  Input = tempC;
  myPID.Compute();
  analogWrite(PIN_OUTPUT, Output);

  //impresion de la temperatura actual en el horno
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp actual: ");
  lcd.print(tempC);
  lcd.print("C");

  delay(1000);
}
```

Fig.11 programación de la impresión de la temperatura en el LCD

La última función que tenemos es el control de temperatura en el horno. Primero se hace una corrección de la temperatura y luego se imprime en el LCD la temperatura actual, que será directamente la del horno.

Apartado:
Adjunto los archivos de proteus, arduinos y este informe en un repositorio de github a continuación el enlace:
https://github.com/ceasaropa/Proyecto_dig_2

Conclusiones
Arduino es un programa con muchas posibilidades dado a que utiliza un lenguaje como c++ por ende observamos que hay muchas posibilidades a la hora de crear distintas funciones con las cuales podemos satisfacer una necesidad o una solución que se adapte a los requerimientos que se hayan diseñado para el dispositivo que se esté creando.

Proteus fue una herramienta muy buena a la hora de simular un arduino pero pudimos concluir que a este le faltaban periféricos que satisfagan nuestras necesidades que tengamos las cuales se plantean en un diseño previo al prototipo.

Para terminar de concluir podemos decir que aprendimos a usar de forma apropiada el arduino y sus distintas herramientas,no en su totalidad dado que hay muchos tipos de arduinos pero si el estándar y el mega arduino.También se aprendió cómo podemos publicar nuestros trabajos mediante la aplicación con sus repositorios en los cuales guardamos y publicamos nuestros archivos.

Bibliografía

- Bejob. (2017, 14 febrero). *Qué es la programación con arduino y para qué sirve*. https://www.bejob.com/que-es-la-programacion-con-arduino-y-para-que-sirve/
- *El lenguaje de programación C++*. (2020, 8 septiembre). Lenguajes de programación. https://lenguajesdeprogramacion.net/cpp/