Universidad EAFIT
ID0616 - Lenguajes Técnicos de Programación
Diseño Integrado de Sistemas Técnicos

Reto 1

Stefania Urrego Diaz - 201610038014
Tomas Chamorro Pareja - 201610011014

11 de septiembre de 2020

Resumen

El objetivo del presente trabajo es aplicar los conocimientos adquiridos en las seis semanas académicas previas en la asignatura Lenguajes Técnicos de Programación. Será posible observar aplicación de conceptos como PWM (Modulación por ancho de pulso), comunicación serial, entradas/salidas análogas y digitales y comunicación mediante bus I2C entre componentes y entre un Arduino maestro y un esclavo. Para este reto, el equipo decidió simular un proceso de un horno sobre el cual el cliente tiene control sobre una resistencia y un ventilador. Se utilizará el Software Proteus para simular las condiciones del circuito electrónico pero cabe resaltar que dicho Software presenta limitaciones en cuanto a velocidad de simulación por la gran cantidad de conexiones, multiplexaje de displays, salidas y entradas análogas, entre otros.

Índice

1. Planteamiento

Se debe plantear un circuito de control por medio de comunicación I2C que incluya los siguientes elementos:

- Dos entradas análogas
- Dos salidas PWM
- Dos entradas digitales
- Dos salidas digitales

2. Descripción Circuito Propuesto

Solución: En la figura se puede observar el circuito desarrollado en Proteus.

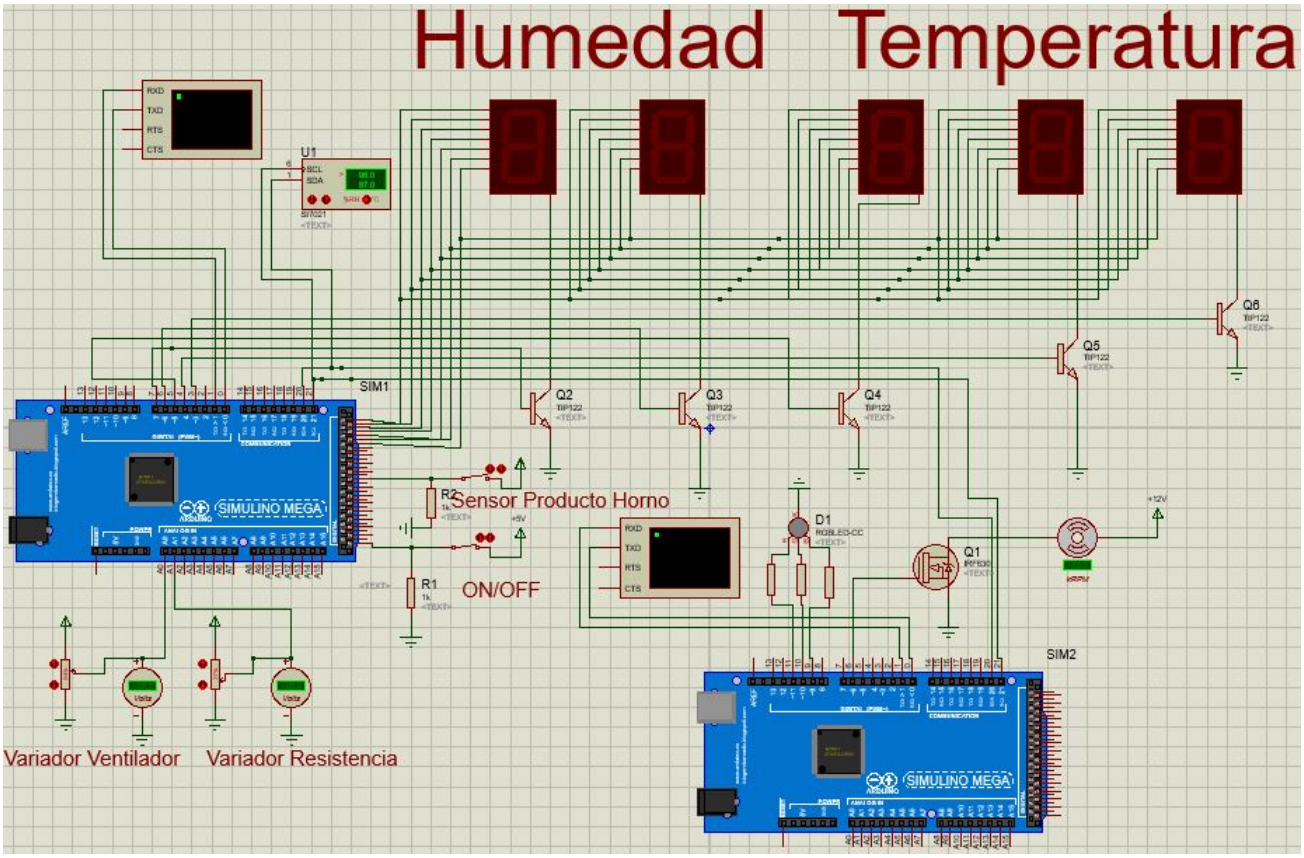

Figura 1: Circuito que simula el proceso al interior de un horno de panadería

Se busca simular el proceso de control en un horno de una panadería. El horno enciende la resistencia una vez se encuentra oprimido el botón de encendido y el sensor de posición notifica que el pan se encuentra dentro del horno. La señal de ON/OFF se simula mediante un suiche de igual manera que el sensor de posición. Todos los sistemas del circuito de control comienzan su ejecución una vez se cumplan la condición de estos dos suiches cerrados (Booleano VERDADERO o Input = HIGH). La resistencia se controla mediante un potenciómetro que corresponde a una entrada de señal análoga al Arduino Maestro y que posteriormente le envía la información al Arduino Esclavo. Asimismo, se cuenta con un motor para simular un ventilador. Este será controlado mediante otra entrada de señal análoga en el Arduino Maestro que también será enviada al Arduino Esclavo. Este ventilador cumple dos funciones: Si se utiliza en conjunto con la resistencia, permitirá aumentar el coeficiente de transferencia de calor y por lo tanto calentar más rápido; si se utiliza solo, permitirá enfriar el compartimiento y el producto.

La resistencia tendrá 3 colores representando la temperatura elegida por el usuario mediante el potenciómetro, como se muestra en la figura a continuación. Cabe resaltar que el Software Proteus no es muy preciso en cuanto al color indicado por la señal modulada por pulso (PWM) por lo que el color en el estado 2 y 3 lucen similares pero corresponden a valor diferentes para el LED RGB.

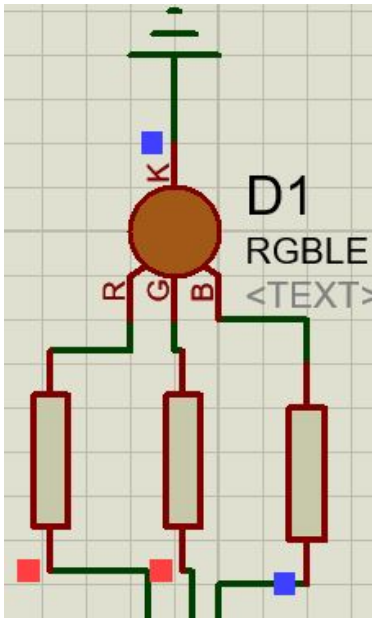

Figura 2: Estado 1 Resistencia @ $R > 33\,\%$ de Potenciómetro. Temperatura baja de resistencia.

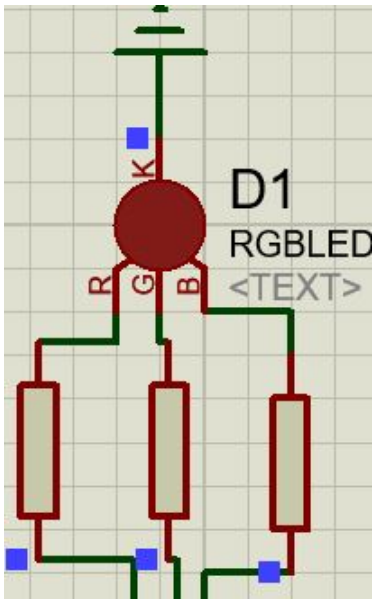

Figura 3: Estado 2 Resistencia @ $33 < R < 66\,\%$ de Potenc. Temperatura media de resistencia.

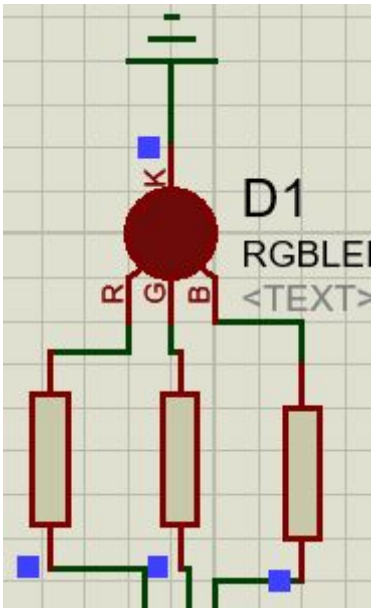

Figura 4: Estado 3 Resistencia @ $R > 66\,\%$ de Potenciómetro. Temperatura alta de la resistencia

La resistencia continua encendida, pudiendo ser modificada por un variador siempre y cuando la temperatura dentro sea inferior a los 90 C, pues se definió en el Código del Arduino Esclavo que si la temperatura medida por el sensor rebasa los 90 grados, se apagará esta resistencia.

Los display siete segmentos muestran el estado del horno mediante las variables humedad y tempe-

ratura al interior. Para la visualización del estado del horno se utilizarán 5 display. Las conexiones corresponden a 7 salidas digitales del Arduino Maestro y cuyas señales deberán ser multiplexadas (encendido y apagado veloz) para su funcionamiento. Para poder multiplexar las señales, se utilizarán 5 transistores que se utilizarán para convertir la señal proveniente del Arduino en un ON o un OFF de cada display. Este display deberá indicar la temperatura marcada con el sensor (que en Proteus cuenta con display propio pero no es así en la vida real) y el usuario podrá tomar decisiones con base en estas variables. En el monitor serial conectado en el Arduino Maestro es posible observar el valor entregados por el sensor y la variable que se visualiza en los display 7 segmentos se puede comprobar mediante dos formas: 1) Viendo este "Serial.println.[en] el monitor serial y 2) Viendo el display del sensor. En la figura 5 se puede observar que el valor entregado por el monitor serial, el mostrado en el display y el digito de decenas de la temperatura coincide. Lamentablemente la velocidad de Proteus no permite multiplexar correctamente y visualizar los 5 datos impresos en los display.

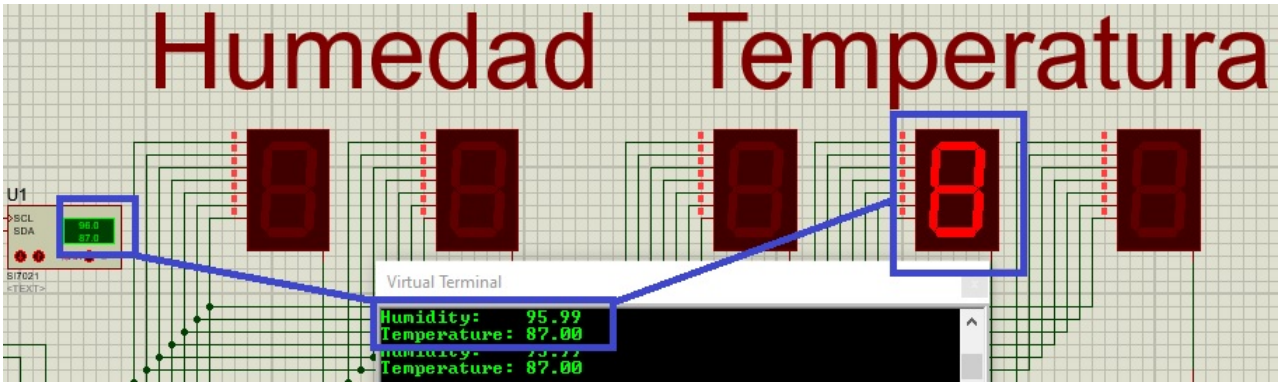

Figura 5: Circuito que simula el proceso al interior de un horno de panadería

Finalmente, el usuario puede controlar el ventilador con el potenciómetro con el rótulo correspondiente. Esta salida proveniente del Arduino Esclavo también corresponde a una salida entre 0 a 255 y permitirá controlar su velocidad. Cabe resaltar que el ventilador girará en sentido anti-horario, por lo que el minidisplay que ofrece Proteus para esta componente imprimirá RPMs negativas.

Sensor Temperatura SI7021 por Comunicación I2C: Se utilizará esta referencia comercial que fue encontrada en la librería de Proteus y se encontró su respectiva librería en GitHub [1]. La información entregada por el sensor será leída mediante la librería mencionada y transmitirá la información al Arduino Maestro de acuerdo al protocolo de comunicación I2C. Esta información será posteriormente descompuesta de la siguiente manera:

- **Temperatura:** Se descompondrá en centenas, decenas y unidades para poder ser visualizado en los display 7 segmentos.
- **Humedad:** La humedad se trabajará únicamente con decenas y unidades y se descompondrá para ser visualizado en las pantallas mencionadas.

Otros datos: Para la conexión del LED RGB, se utilizaron resistencias de 230Ω para cada entrada.

Adicionalmente, tanto el suiche que simula el ON/OFF del sistema como el suiche que define si hay producto dentro del horno utilizan un voltaje de alimentación de 5V y una resistencia de 1k Ω. Para controlar el motor mediante el PWM, se utilizó un MOSFET con referencia IRF630 como suiche y como aislamiento entre circuito control y circuito de potencia, este último alimentado por una fuente de 12V.

3. Descripción Código

Librerías: Como se mencionó anteriormente, se utilizará comunicación I2C para transferir datos tanto desde el sensor al Arduino Maestro como datos desde el Arduino Esclavo al Arduino Maestro. Para ello se incluyó la librería $< Wire.h >$. Se incluyó adicionalmente la librería"Adafruit_Si7021.h"para obtener datos del sensor. Finalmente, se utilizó la librería "dis7seg.h", suministrada por el docente, para controlar el display 7 segmentos de una manera más compacta.

Variables Maestro: Se declararon variables para la descomposición de los números de la humedad y temperatura, conexiones de los transistores para las pantallas, entradas análogas, entradas digitales, salidas digitales a los display (Ver figura 6).

```
int variador_motor;
int variador_rgb;
int rpm_motor;
int cen_temp;
int dec_temp;
int uds_temp;
int dec_hum;
int uds_hum;
int temphorno;
int humhorno;
int tr_dechum = 7;
int tr_udhum = 6;
int tr_centem = 5;
int tr_dectem = 4;
int tr_udstem = 3;
int on_off = 53;
int print1borrar;
int print2borrar;
int sensor_pos = 37;
```

Figura 6: Declaración de variables Arduino Maestro

Variables Esclavo: Para el Esclavo se definieron tres variables (Ver figura 7) que corresponden a las señales análogas (ya escaladas a 255 para poder ser transmitidas) del variador del motor y de la resistencia y la temperatura del horno para establecer la condición de apagado de la misma cuando la temperatura supere los 90ºC.

```
int variador_motor;
int variador_rgb;
int temphorno;
```

Figura 7: Declaración de variables Arduino Esclavo

Funciones Arduino Maestro: En el setup del Arduino del Maestro se incluye unas funciones disponibles para mostrar en el monitor serial que se detectó un sensor I2C y se especifica su dirección. Otras remarcas:

- Se créo una función llamada **"descom_temphum"** la cual devuelve la descomposición en centenas, decenas y unidades de la temperatura leída por el sensor y en decenas y unidades de la lectura de humedad. La función es capaz de determinar si la temperatura contiene centenas o no para evitar errores. La función no descompone valores de temperatura inferiores a 10 ya que no correspondería a un horno (quizás sería ya un refrigerador). Estas variables serán utilizadas para imprimir en los display 7 segmentos.
- **Función "mostrar":** Encargada de tomar la descomposición de los números y hacer el multiplexado de los display mediante los transistores. Es capaz de detectar si hay centenas para hacer la respectiva modificación en el multiplexado (no prender pantalla de centenas).
- **Void Loop del Arduino Maestro:** Descompone la humedad y la temperatura; imprime en el monitor serial los valores DIRECTOS leídos por el sensor; lee los valores de los potenciometros (regulación ventilador y resistencia) y mapea estas señales de los potenciometros (convierte señal de 0 a 1023 en una de 0 a 255) para ser enviados al Arduino Esclavo; comienza la transmisión de datos entre en el Arduino Maestro y el Arduino Esclavo y envía los datos de los potenciómetros y la temperatura del horno; Finalmente, llama la función "mostrar"para hacer el multiplexado e imprimir los valores en los display.

Funciones Arduino Esclavo: En el setup se especifica que se hará transmisión de datos con "1"(A. Maestro) y se llama la función receiveEvent para el procesamiento de los datos provenientes del Arduino Maestro. En la función "receiveEvent"se leen los datos en el siguiente orden: temperatura horno, variador motor y variador rgb. Una vez recibidos estos datos, escribe el valor del variador motor en el pin correspondiente, lee la temperatura y, mientras la temperatura esté por debajo de 90, se hace variar los valores del RGB de acuerdo al valor del potenciometro leido desde el Arduino Maestro.

Referencias

[1] Herrada D. (2020). Librería Arduino adafruit/Adafruit_Si7021. *GitHub*. Recuperado el 9 de septiembre de 2020 de https://github.com/adafruit/Adafruit_Si7021.