

“Proceso de instrumentación basado en una compostadora con LabView”

Adrian Evaraldo
Escuela de Producción, Tecnología y Medio Ambiente
Universidad Nacional de Río Negro

16/11/2015

Índice general

1.	Introducción	1
2.	Hardware	2
2.1.	Hardware utilizado	2
2.2.	Círculo de detección de temperatura	2
2.3.	Características de las entradas y salidas del Arduino UNO R3	3
2.4.	Conexionado entre el Arduino y los instrumentos	4
3.	Software	5
3.1.	Comunicación entre planta y la computadora	5
3.2.	Arduino	5
3.3.	LabView	8
4.	Histórico	11
4.1.	Hardware	11
4.2.	Software	13
5.	Conclusiones	15
6.	Bibliografía	15

“Proceso de instrumentación basado en una compostadora con LabView”

1. Introducción

Se desea hacer el proceso de instrumentación de una compostadora por medio de una computadora con el programa LabView 2013 de 64 bits y un Arduino UNO. Las interacciones entre el Arduino y la planta serán las siguientes:

- medir la temperatura interna de la compostadora
- accionar una resistencia calefactora
- saber si la compuerta se encuentra abierta o cerrada
- accionar una electroválvula para el escape de gases

Modelo de compostadora

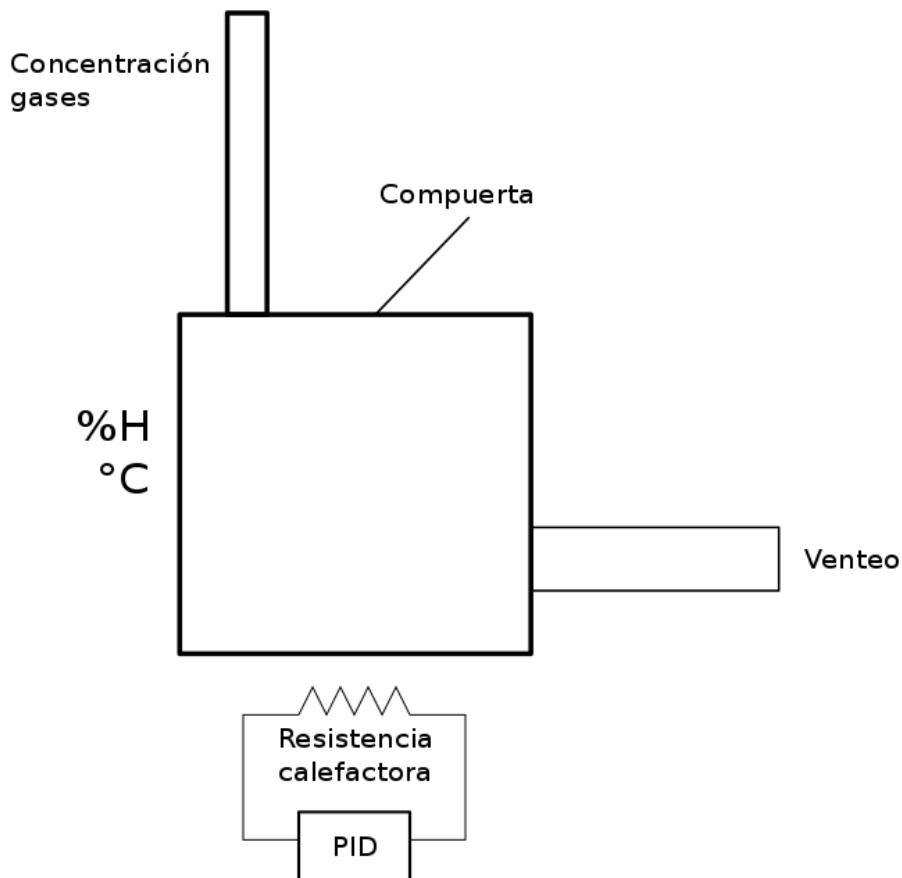


Figura 1.1: diagrama de una compostadora

2. Hardware

Debido a que adquirir una RTD y una resistencia calefactora se excede del presupuesto, se optó por realizar simulaciones de éstos elementos.

La RTD, se simulará a través de un potenciómetro. La medición del valor resistivo, se hace a través del método de 4 hilos. Véase la sección 2.2 en la página 2

En el caso de la resistencia calefactora, se utilizará un led comandado por una salida PWM (Pulse Width Modulation)

Para el accionamiento de la electroválvula se comanda un rele. Y, por último, la retroalimentación del estado de la compuerta será por medio de un switch de inclinación.

Para mayor información sobre las entradas y salidas del Arduino, véase la sección 2.3 en la página 3

2.1. Hardware utilizado

- 1 Arduino UNO R3
- 1 LM317
- 1 resistencia de 560Ω 1 %
- 1 potenciómetro de 470Ω
- 2 resistencias de 100Ω 5 %
- 1 sensor de inclinación
- 1 relé de 5V
- 1 resistencia 150Ω 5 %
- 1 led verde

2.2. Circuito de detección de temperatura

Según la ley de Ohm, podemos calcular el valor de una resistencia si conocemos el valor de la corriente que la atraviesa, y de la caída de tensión que ocurre en ella. Debido a que las entradas analógicas miden tensión, se debe hacer circular una corriente conocida a la resistencia que se desea medir.

Para ésto, se utilizó un LM 317 en configuración de fuente de corriente. El integrado utilizado, genera una diferencia de potencial entre los pines Vout y Ref típica de 1.25V. Colocando una resistencia conocida entre ambos pines, resulta en una fuente de corriente de valor conocido.

El potencial entre Vout y Ref puede variar si se cambia el IC, por lo que debe medirse las tensiones en cada uno de los pines.

“Proceso de instrumentación basado en una compostadora con LabView”

Utilizando la tensión de referencia de 1.25V y una resistencia de 560Ω , ésta configuración, según la ley de ohm, entrega una corriente aproximada de 2.23mA.

Para asegurarse de no quemar ningún componente, se colocaron resistencias en paralelo de 100Ω .

Para conocer el valor de la resistencia del potenciómetro, se usan las leyes de Kirchhoff.

$$IR + I50\Omega = A1 \quad (2.1)$$

$$IR = A1 - I50\Omega$$

$$R = \frac{A1 - I50}{I} \quad (2.2)$$

Los valores de $A1$ están comprendidos entre 0 y 1023 y no representan una tensión, sino que es el número en codificación decimal del resultado del ADC de la entrada analógica. Para llevar $A1$ a un valor de tensión, se lo debe dividir por el número de cuentas y multiplicar por el valor de tensión de referencia. Como el Arduino tiene un voltaje de referencia de 5V y un ADC de resolución de 10 bits entonces, la cantidad de cuentas es $2^{10} = 1024$.

Por lo tanto la ecuación 2.2 resulta en:

$$R = \frac{\frac{5}{1024}A1 - I50}{I} \quad (2.3)$$

2.3. Características de las entradas y salidas del Arduino UNO R3

El Arduino UNO R3, cuenta con:

- 14 pines GPIO (General Purpose input/output) digitales, de los cuales 6 pueden funcionar como salidas PWM (Pulse Width Modulation) con resolución de 8 bits
- 6 pines que actúan como entradas analógicas con resolución de 10 bits
- salida regulada de 5V, 3.3V y GND (masa)

Los pines que pueden funcionar como salidas PWM son 3, 5, 6, 9, 10 y 11. Éstos funcionan a una frecuencia aproximada de 490Hz, salvo por los pines 5 y 6, los cuales andan a 960Hz

“Proceso de instrumentación basado en una compostadora con LabView”

2.4. Conexión entre el Arduino y los instrumentos

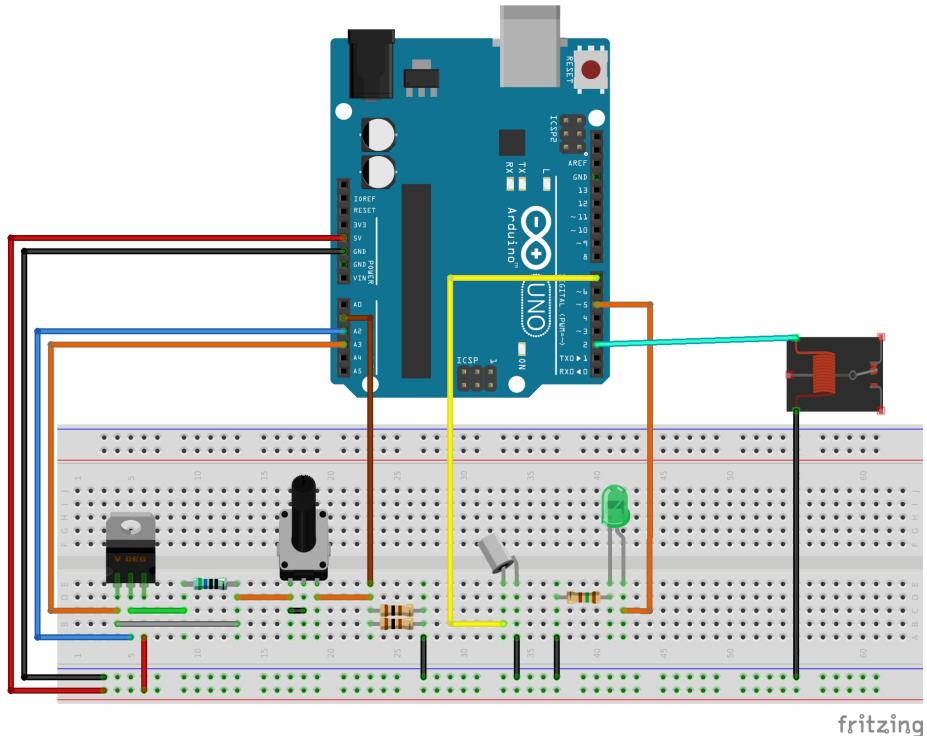


Figura 2.1: circuito protoboard

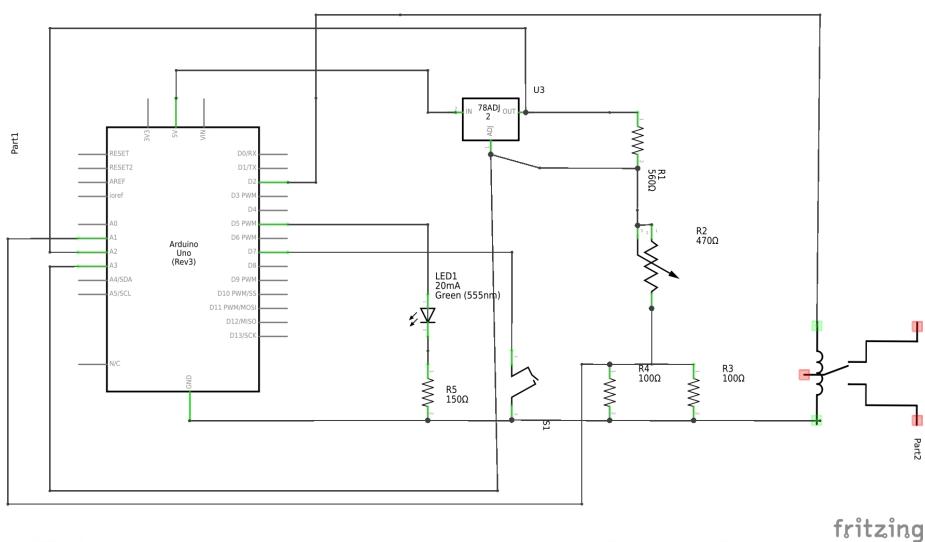


Figura 2.2: esquemático

3. Software

En los siguientes apartados, se detallarán el código implementado tanto en el Arduino como en el LabView. También la forma de comunicación entre ambos elementos.

Se debe aclarar que a causa de que la comunicación entre el Arduino y el LabView es lenta, se decidió hacer que el elemento de control de toda la planta sea el Arduino. El software de la computadora, pasa a ser sólamente un HMI (Human Machine Interface).

3.1. Comunicación entre planta y la computadora

En nuestro caso, el Arduino se encuentra en la planta. La comunicación con la computadora, se realiza a través del cable USB que viene de fábrica con el dispositivo que se encuentra en la planta y se lo conecta un puerto USB de la PC.

En el LabView, se instaló el paquete VISA v5.0.3. Éste permite la comunicación entre el programa y el Arduino

3.2. Arduino

Para realizar el programa y subirlo, se utilizó el Arduino Software IDE v1.6.5. Para poder lograr realizar el control PID, desde el mismo IDE, se consiguió la librería PID v1.1.1 creada por Brett Beauregard.

link de la librería: <https://github.com/br3ttb/Arduino-PID-Library/>

“Proceso de instrumentación basado en una compostadora con LabView”

Código implementado

```
1 #include <PID_v1.h>
2
3 int pin_resistencia = 1;
4 int puerta = 7;
5 int bin;
6 int adc_temp;
7 int pwmVar = 0;
8 int pwm = 5;
9 int rele = 2;
10 int releVar = 0;
11 int analogVout = 2;
12 int analogRef = 3;
13 int auxrel;
14 int adc_vout, adc_ref, refaux;
15 int i, adc_sum = 0, adc;
16 double Iref317 ,Vref317;
17 double aux1, aux2, aux3, resistencia;
18 double auxt1, temp, auxt2 = 100.0 * 0.003850;
19 double pid_out, setpoint = 25.0;
20
21 //Specify the links and initial tuning parameters
22 PID myPID(&temp, &pid_out, &setpoint, 2, 5, 1, DIRECT);
23
24 /*
25 * PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)
26 *Parameters
27 *Input: The variable we're trying to control (double)
28 *Output: The variable that will be adjusted by the pid
29 *(double)
30 *Setpoint: The value we want to Input to maintain (double)
31 *Kp, Ki, Kd: Tuning Parameters. these affect how the pid
32 *will chage the output. (double>=0)
33 *Direction: Either DIRECT or REVERSE. determines which
34 *direction the output will move when
35 *faced with a given error. DIRECT is most common
36 */
37
38 void serialEvent()
39 {
40     //—————
41     /*
42     * leo los valores que me llegan desde el labview y los
43     * paso de ASCII a enteros
44     */
45     //—————
46
47     pwmVar = Serial.parseInt();
48     setpoint = pwmVar;
49     releVar = Serial.parseInt();
50     if(releVar == 0)
51     {
52         digitalWrite(rele, LOW);
53         //Serial.println(" ejecute low");
54     }
55     else
56     {
57         digitalWrite(rele, HIGH);
58         //Serial.println(" ejecute high");
59     }
```

“Proceso de instrumentación basado en una compostadora con LabView”

```
60     }
61
62     void setup() {
63         // put your setup code here, to run once:
64
65         //abro la comunicacion serie
66         Serial.begin(9600);
67
68         //declaro inputs y outputs
69         pinMode(puerta, INPUT_PULLUP);
70         pinMode(pwm, OUTPUT);
71         pinMode(rele, OUTPUT);
72         digitalWrite(rele, LOW);
73
74         //inicio el PID
75         myPID.SetMode(AUTOMATIC);
76     }
77
78     void loop() {
79         // put your main code here, to run repeatedly:
80
81         //-----
82         //leo entradas
83         //-----
84
85         //leo el potenciómetro
86         //filtro para evitar ruidos, leyendo 100 veces y tomo el
87         //promedio
88         for(i = 0; i < 100; i++){
89             {
90                 adc = analogRead(pin_resistencia);
91                 adc_sum += adc;
92             }
93
94             adc_temp = adc_sum / 100;
95
96             //reinicializo el sumador
97             adc_sum = 0;
98
99             //leo el valor de tensión en Vout y REF del 317
100            adc_vout =analogRead(analogVout);
101            adc_ref = analogRead(analogRef);
102
103            //me fijo si la compuerta esta abierta. open= 1, close=0
104            bin = digitalRead(puerta);
105
106
107         //-----
108         /*
109         * proceso los valores analógicos para poder conocer la
110         * temperatura supongo que utilizo una PT100
111         */
112         //-----
113
114         //calculo el valor de referencia para poder calcular la
115         //cantidad de corriente que entrega el 317
116         //refaux es el valor en formato de 10 bits
117         refaux = adc_vout - adc_ref;
118
119         //lo paso a valor de tensión
120         Vref317 = refaux *(5.0/1024.0);
121         Iref317 = Vref317 / 560.0;
```

“Proceso de instrumentación basado en una compostadora con LabView”

```
122 // calculo el valor de la resistencia
123 //R = ( 5 * ((lectura / 1024) - (Vref/56)) )/(Vref/560)
124 aux1 = adc_temp / 1024.0;
125 aux2 = Vref317 / 56.0;
126 aux3 = 5.0 * (aux1 - aux2);
127 resistencia = aux3 / Iref317;
128
129 // calculo la temperatura en base a la aproximacion de
130 // Callendar – Van Dusen donde Rt = R0 * (1 + alpha*t)
131 // con alpha=0.003850, despejando la temperatura
132 // queda t = (Rt – R0)/(R0 * alpha)
133
134 aux1 = resistencia - 100.0;
135 temp = aux1 / aux2;
136 //auxt2 esta hardcodeado en la declaracion de variables
137 //para hacer mas rapido el loop
138
139 //—————
140 //llamo al PID para que tome una accion
141 //—————
142
143 myPID.Compute();
144
145 //—————
146 //mando los valores al labview
147 //—————
148
149
150 // valor de la lectura de temperatura
151 Serial.print(temp);
152 Serial.print(",");
153
154 // valor de la salida del pid
155 Serial.print(pid_out);
156 Serial.print(",");
157
158 // valor de la compuerta abierta = 0 ; cerrada = 1
159 Serial.println(bin);
160
161 // comando la salida pwm
162 analogWrite(pwm, pid_out);
163
164 delay(5);
165 }
166 }
```

3.3. LabView

El programa implementado en LabView, tiene como objetivo recibir los datos recibidos desde el Arduino (temperatura, salida del PID y estado de la compuerta), mostrarlos al usuario y permitirle a él que modifique el setpoint de temperatura del proceso.

Los datos que se reciben llegan como un solo string. Para determinar que parte de éste corresponde a qué variable, se siguen los siguientes pasos:

- Del string recibido, se detecta la primera coma (”, ”).

“Proceso de instrumentación basado en una compostadora con LabView”

- Lo que está antes de la coma, resulta ser una variable. Se la separa del string y se lo convierte a un numero con formato doble o entero.
- A lo que queda del string, se le quita la coma.
- Se lo manda al bloque que sigue para sacarle el próximo dato.

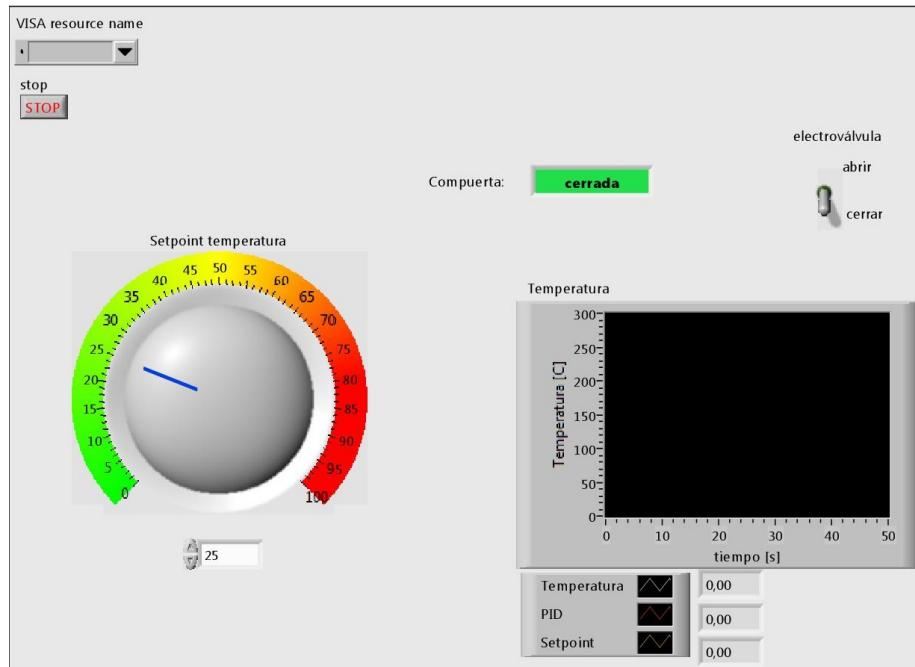


Figura 3.1: Interfaz con el usuario

“Proceso de instrumentación basado en una compostadora con LabView”

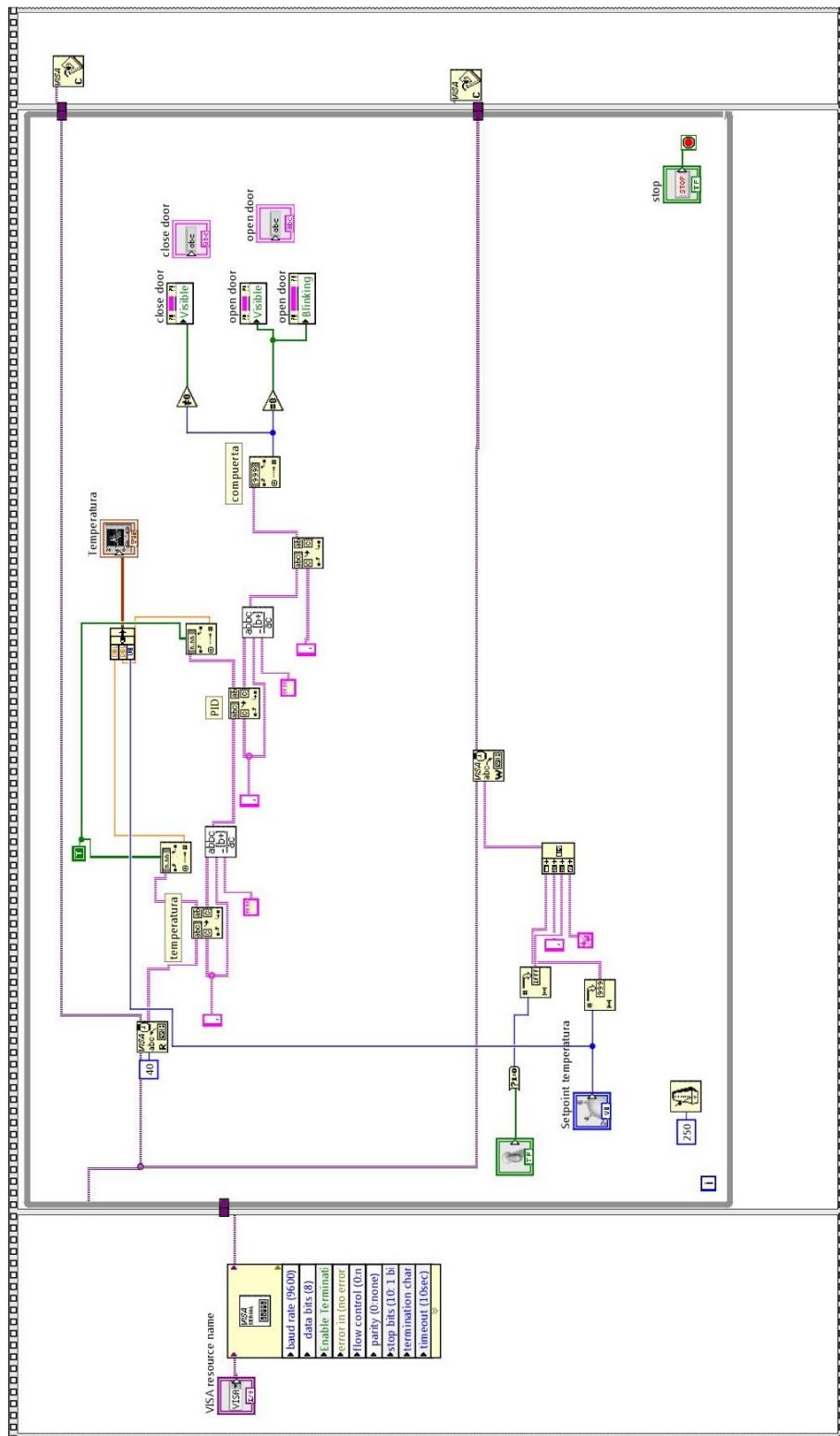


Figura 3.2: Diagrama de bloques del LabView

4. Histórico

4.1. Hardware

En un principio, se intentó medir la RTD con el siguiente circuito:

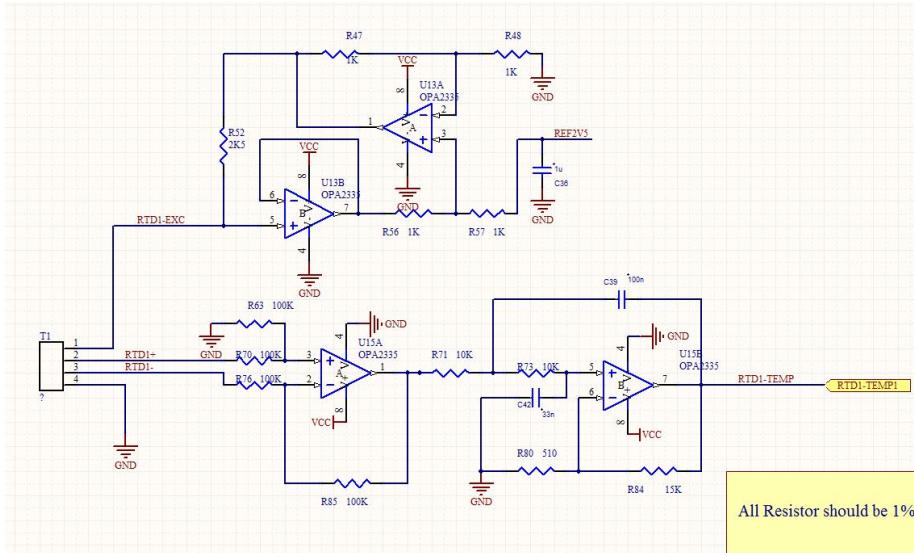


Figura 4.1: primer circuito para medir RTD

Por una cuestión de disponibilidad, se cambió el operacional por un TL074. Éste, tiene en un mismo encapsulado los cuatro amplificadores necesarios en vez de necesitar los dos OPA2335 del circuito original.

Al carecer de una RTD de cuatro hilos y utilizar un potenciómetro, se conectó los pines 1 y 2 del conector a las patas puenteadas de la resistencia variable. Y los pines 3 y 4 a la pata restante.

Luego de varias pruebas, por algún motivo todavía desconocido, el circuito nunca funcionó. En la salida del U15B, la tensión se mantenía constante a pesar de cambiar el valor del potenciómetro en toda su escala. Por éste motivo, se decidió simplificar el circuito, implementando la fuente de corriente que quedó en la versión final.

Circuito final

El circuito final, se realizó primero en un protoboard (figura 4.2). Pero debido a conexiones defectuosas que inducían ruidos en la señal, se tuvo que hacer el mismo esquemático, pero en una placa. Para simplificar la fabricación y optimizar tiempos, se emplearon dos placas microperforadas de 5x5, colocando en una de ellas el rele y el interruptor de inclinación (figura 4.3), y en la otra, el potenciómetro y el led (figura 4.4).

“Proceso de instrumentación basado en una compostadora con LabView”



Figura 4.2: circuito en protoboard

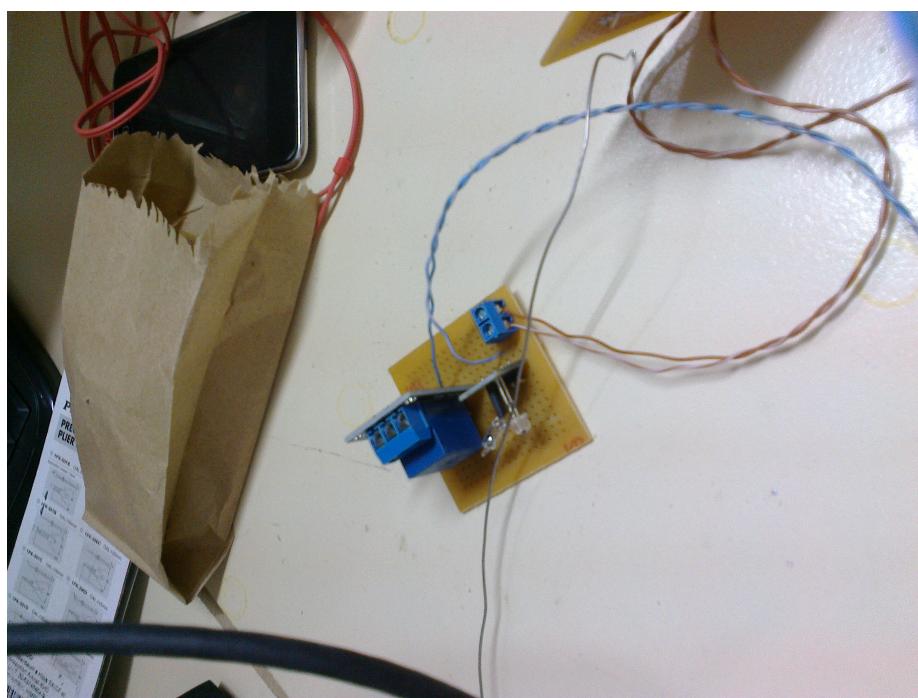


Figura 4.3: placa microperforada con rele y sensor de inclinación

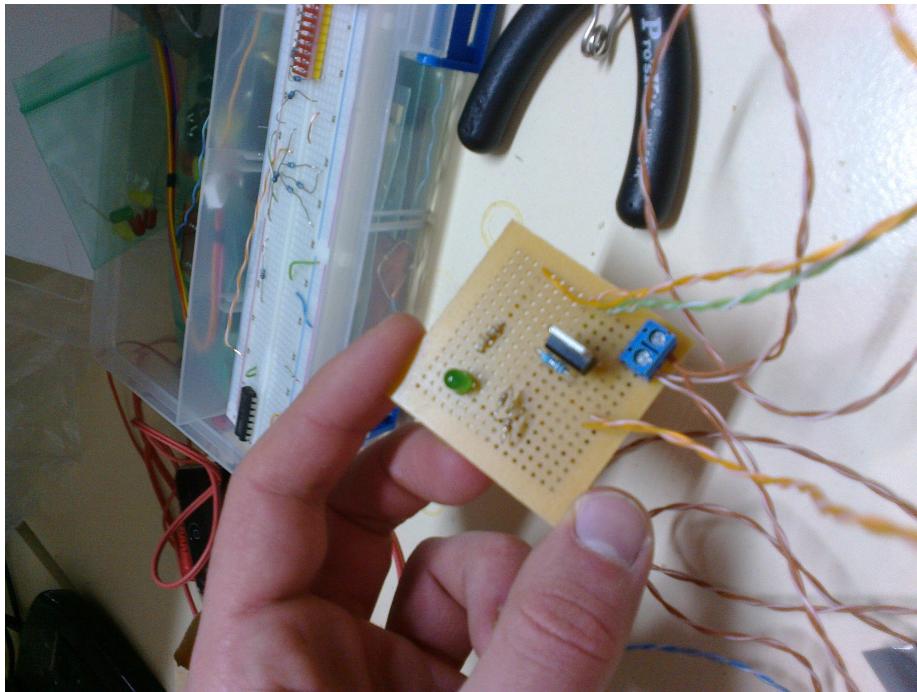


Figura 4.4: placa microperforada con medición RTD y led

4.2. Software

En un principio, se decidió utilizar la librería LIFA (LabView Interface For Arduino). Al momento de realizar las primeras pruebas, se hizo evidente de que con éste paquete se necesita hacer muchas operaciones para hacer una simple acción. No sólo eso, sino que muchas de ellas, se terminan repitiendo. Éste es el motivo por que se optó por utilizar el paquete VISA.

En la figura 4.5 que se muestra a continuación, se puede observar el programa realizado con LIFA para encender y apagar el led del pin 13 del Arduino. En las iteraciones del programa pares, el led se apaga. Cabe aclarar, que con éste código no se observó un delay importante de comunicación como es el que se percibe con VISA en la versión final del programa.

“Proceso de instrumentación basado en una compostadora con LabView”

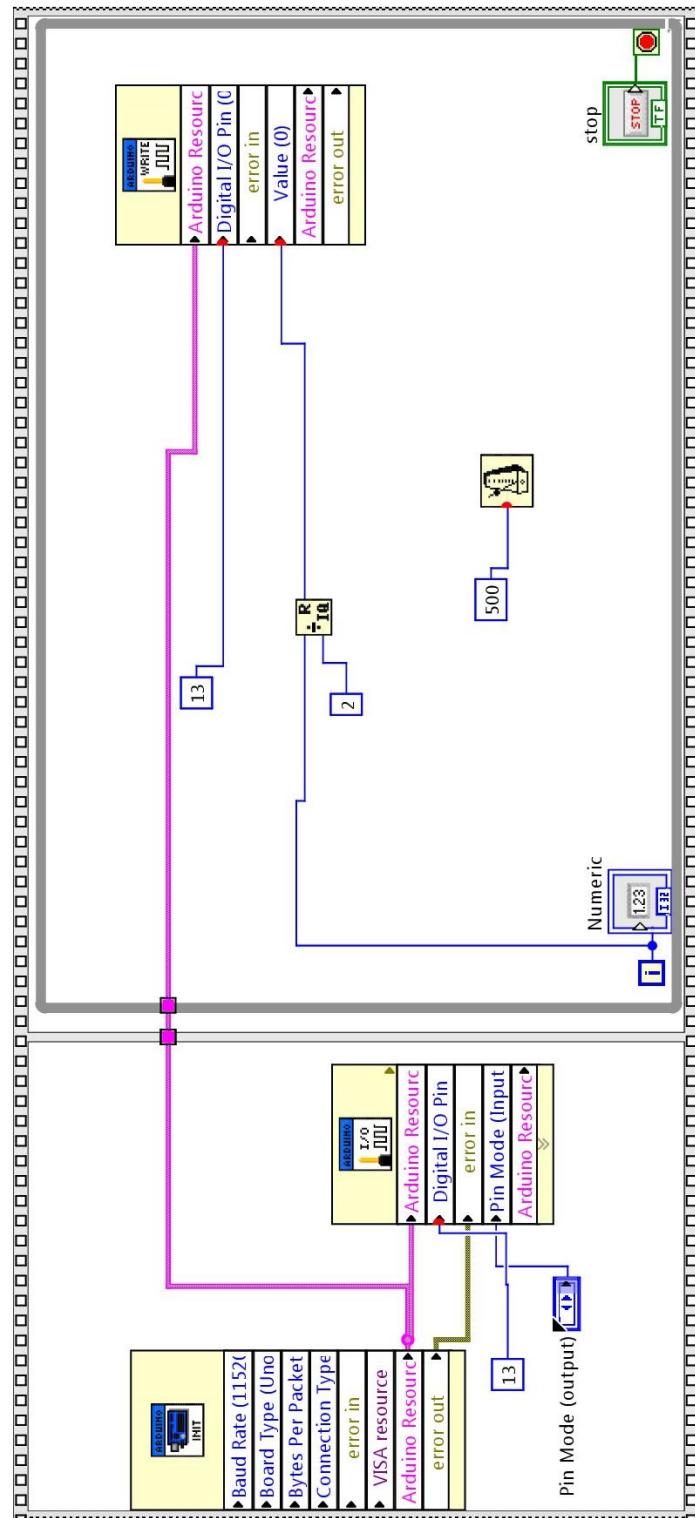


Figura 4.5: diagrama en bloques de la implementación con LIFA

5. Conclusiones

Si bien el proyecto con el que se trabajó, no tuvo una inclusión de medición de humedad o de concentración de gases, Es factible el agregado de éstos componentes.

6. Bibliografía

- <https://www.arduino.cc/en/Main/ArduinoBoardUno>
- <https://www.arduino.cc/en/Reference/AnalogWrite>
- <http://playground.arduino.cc/Code/PIDLibrary>
- <https://github.com/br3ttb/Arduino-PID-Library/>
- <http://www.ti.com/lit/ds/symlink/lm117.pdf>