**Controlador de temperatura**

**Objetivo:** calentar una muestra a una temperatura deseada. Para eso se calienta un porta-muestra con 2 calentadores y se mide la temperatura con un sensor en contacto con el porta-muestra. Para el control de temperatura se diseña un lazo cerrado ON/OFF y de realimentación PID.

Rango de temperatura deseado: ambiente a 80 °C.

Cuidados en su futura implementación: evitar la disipación del porta-muestra con el ambiente, así se utiliza la menor energía necesaria para calentar. Para eso se aísla el porta-muestra de materiales que disipen más que la muestra. Como el porta-muestra estará contacto con una mesa metálica, se aísla el porta-muestra con papel Kapton.

**Sensor temperatura:**

* LM35:

Conexión con el arduino: patita del medio en canal A0, otra patita a 5 V, la otra a tierra. Importante ver en la hoja de datos en qué sentido se conecta, te das cuenta que está al revés porque se sobrecalienta y se puede quemar.

**Calentadores:**

* 2 calentadores MP825, 20.0 ohm, 1 % R:

<https://www.mouser.mx/ProductDetail/Caddock/MP825200-1?qs=tucQmhgEO3rgL8GhB7bH%252Bw%3D%3D>

Necesitan 5 Watt cada una para aumentar 1 °C.

**Control temperatura:**

* 1 POWER MOSFET:

Se alimenta con una fuente continua externa, una señal on/off del Arduino 5V o PWM para hacer PID con el duty cycle. Ver de la hoja de datos en qué pata se conecta cada cosa.

Se calientan, por hay es necesario ponerle el disipador, dependiendo de la corriente que circule.

opciones:

Mejor opción:

**1j26ah - rfp50n06:**

rsd(on) = 0.022 ohm

Préstamo de Damián, hay que devolver:

irf224n - p352d - mk 8.0:

irf840 - y97k - gu:

**Conexiones:**

Alimentar a los 2 calentadores y el Power MOSFET (**1j26ah - rfp50n06)** con fuente continua, entre 10 y 22 Volt, dependiendo del máximo de set point de temperatura deseada. Chequear en clase: conectar el negativo de la fuente con la tierra del Arduino.

opción 1: Los 2 calentadores en paralelo entre sí, y en serie con el Power mosfet. Esta opción es mejor si se desea calentar más.

opción 2: Los 2 calentadores en serie, y en serie con el Power mosfet.

opción de prueba: Conectar un solo calentador y el mosfet en serie para probar la programación con 10 Volt.

Discutir: agregar resistencia de seguridad al circuito?

**Programación:**

Orden: primero compilar el arduino file ***Temperature***, luego el script de python. Probar con que funcione el sensor LM35 con el script ***read\_temperature.py***.

Cuando python establece la conexión con Arduino, imposibilita el manejo desde el sketch.

* 1 Arduino file:

***Temperature***: es el archivo que hará todo, lee temperatura y la controla con on/off. La idea es que este archivo funcione, independiente de si existen los de python. Tendrá sus propios valores de set point, error, P, I, D, y a lo sumo python se los cambiara.

Estado actual: Solo lee temperatura en A0. Hay que programar todo lo de on/off, PID del power mosfet.

* 4 scripts de python: 1 de conexión (***instrument\_arduino.py***), 2 independientes (***read\_temperature.py***, ***control\_temperature.py***) y 1 principial (***PyTemperature.py***) para juntarlos.

Los siguientes scripts tiene 2 clases (Frontend y Backend) y se comunican entre sí a través de pyqtSignals y pyqtSlots. En el Frontend está toda la programación de interfaz usuario. En Backend todo lo que es comunicarse con el instrumento, medir, tratar la data de ser necesario.

***read\_temperature.py*** : sirve para ver la data de arduino cada cierto tiempo. Es una especie de Plot Monitor del Arduino, con la opción de guardar la data que lee. Cuidados: no se puede pedir la data de Arduino a tiempos muy cortos (menos de 0.1segundos) pq crashea, puede pq la comunicación entre python y Arduino es lenta. Tiene que haber cierta magnitud entre la frecuencia con que lee data el Arduino y la frecuencia con la que python le pide esa data. Estado: completo, funciona bien si se corre solo con TemperatureSensor.

***control\_temperature.py***: sirve para setear los valores de temperatura deseados (set point), y su error (que el sistema oscile en tal delta grados), valores de P,I,D. Estado: en construcción, falta la comunicación con Arduino.

***PyTemperature.py***: junta los 2 scripts anteriores en un Frontend donde se instancia los frontend de read y control, y un Backend donde se instancia los backends. Tiene la opción de crear la carpeta del día y avisarle a read\_temperature.py donde se puede guardar la data de temperatura medida.

Estado: A veces tira error por como estoy pasando al instrumento “arduino”, en realidad tengo que ver como pasarselo como argumento a los Backends de read\_temperature y control\_temperature, pero pareciera andar bien.