industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERS IDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Desarrollo de un sistema de monitorización de temperaturas en tiempo real para intercambiador de calor de doble tubo con sondas sumergibles de temperatura DS18B20 usando el microcontrolador Arduino

TRABAJO FIN DE GRADO

GRADO EN TECNOLOGÍAS INDUSTRIALES

Autor: **Joaquín Martínez Jimeno**
Director: Andrés Cabrera Lozoya
Codirector: José Ramón Navarro Andreu

Cartagena, 11 de Abril de 2018

Universidad Politécnica de Cartagena

Agradecimientos

A mis padres, José Joaquín Martínez García y María del Carmen Jimeno Yagües, por la oportunidad que me han brindado, apoyándome, tanto mental como económicamente, para finalizar exitosamente la carrera.

Agradecer al resto de mi familia, con especial mención a mi hermana, Laura Martínez Jimeno, por su apoyo incondicional y sus elogios. Sé que en un futuro próximo llegarás a conseguir metas mucho mayores. Mencionar también el apoyo de amigos, siendo comprensibles y dándome apoyo.

A mi pareja, María del Mar García Vega, por su apoyo durante mis estudios, dándome fuerza mental y ayudándome a ser constante en mi trabajo.

A los directores del TFG, por darme la oportunidad de realizar este proyecto. A mi director, Andrés Cabrera Lozoya, siempre atento y disponible en todo momento de la realización del proyecto. Al codirector, José Ramón Navarro Andreu, por su apoyo y entusiasmo en la realización del mismo. De ambos me quedo con la pasión y dedicación por su trabajo.

Mención especial a María Isabel Pérez González, técnico de laboratorio de física aplicada, por su atenta ayuda y dedicación durante el montaje físico de la instalación.

Agradecer a todas las fuentes de información que me han ayudado a la realización de este proyecto. Sin ellas no hubiese sido posible su realización.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Capítulo 1 Introducción, motivaciones y objetivos.

El proyecto consistirá en la instalación de un equipo de monitorización, en tiempo real, de las temperaturas de entrada, medias y de salida, en un intercambiador de calor estándar del laboratorio de Física Aplicada de UPCT, aunque será apto para cualquier otro intercambiador o estación de estudio de temperaturas en tiempo real. Se va a realizar mediante un equipo físico de Arduino y un software de laboratorio.

Esta instalación de laboratorio, nos permite a través de las temperaturas medidas, calcular el calor cedido entre los fluidos que circulen por él, el rendimiento del intercambiador, la diferencia media logarítmica (LMTD) y el coeficiente global de transmisión de calor.

Figura 1.1 - Intercambiador de calor

El intercambiador de calor está formado por tubos concéntricos, por los que circulan ambos fluidos realizando el intercambio de calor por convección, por un caudalímetro para controlar los caudales de entrada, y por un regulador de la temperatura de entrada al flujo caliente.

Figura 1.2 - Regulación de caudal y temperatura

Este equipo nos permitirá realizar ensayos con caudales de entrada fijados, variantes entre rangos de 1000 y 4000 cm^3/min, y ensayos a temperaturas fijas de caudal caliente de entrada entre 40 y 55 ºC. Estos ensayos se realizan tomando medidas hasta la estabilización del sistema.

Esta regulación de temperaturas se consigue con un radiador acoplado en la parte trasera de la instalación, junto a la bomba de retroceso.

Figura 1.3 - Radiador y bomba

La realización del proyecto se ha propuesto como principal motivo debido a los antecedentes. El sistema empleado actualmente cuenta con unos equipos desfasados que, además de estar obsoletos en cuanto a nivel de precisión, están dando muestras de fallo de medición, provocando errores de cálculo, entre ellos la obtención de rendimientos superiores al 100%.

El actual equipo cuenta con:

- Un equipo de termómetros digitales, termómetros de precisión por debajo de los actuales en el mercado con medidas de un decimal. Estos termómetros están dispuestos a las entradas y salidas de ambos flujos, contando cuatro.

Figura 1.4 - Termómetros digitales actuales

- Para la toma de temperaturas en los puntos intermedios del intercambiador, se utilizan dos termómetros de mercurio de laboratorio. Estos termómetros pueden llevar a fallo en la toma de medidas, ya que se realiza a vista del alumno.

Figura 1.5 - Termómetros mercurio actuales

- Los resultados obtenidos no son registrados en ningún sistema informático, ya que los termómetros no tienen disponibilidad de ser conectados a equipos, por lo que las medidas deben ser tomadas manualmente por el equipo de laboratorio que está realizando el ensayo. Esto lleva a poder provocar fallos en la medición, puesto que no se registran simultáneamente las medidas de todos los parámetros a observar. Por otro lado, al no usar equipo informático, no se pueden percibir otros parámetros útiles del estudio, como pueden ser medidas extremas, tiempo de ensayo exacto entre medidas, o gráficas.

Los objetivos, por tanto, del nuevo sistema, será mejorar las anteriores precariedades de la instalación marcando como principales puntos de interés:

- Instalación de unos sensores de temperatura que envíen los datos de las temperaturas tomadas a un equipo informático donde serán almacenadas correctamente, además de ser mostradas en tiempo real en todo momento. Esto nos permitirá una vez acabado el ensayo, exportar los datos obtenidos en un fichero para posterior análisis y estudio de la instalación del intercambiador por parte del equipo humano que realice el ensayo.

- La creación de un software para monitorizar en tiempo real las mediciones que se están realizando, por medio de una interfaz gráfica de fácil comprensión y entendimiento que además sea agradable de observar durante la realización del ensayo, puesto que en la duración del ensayo no se realizará otra tarea que la de observación, por lo que al ser clara y concisa la información mostrada en la interfaz, permitirá al equipo humano sacar unas primeras conclusiones y realizar unas primeras opiniones técnicas de la instalación.

- Elección del equipo físico óptimo, poniendo especial atención en la calidad de los sensores, que se encontrarán en todo momento en contacto con un medio fluido, y por ello, deberán tener un buen encapsulado, para asegurar una gran durabilidad y que no provoque fallos de funcionamiento debido a humedades, lo que dará medidas

ilógicas que llevaran al fallo en cálculos posteriores de estudio, o contacto directo con el fluido, provocando el fallo total del sistema.

Conocidos los objetivos y necesidades marcados para la realización del proyecto la instalación desarrollada será la siguiente:

- Un equipo físico alrededor de Arduino, que nos va a permitir construir un dispositivo digital que mida y monitorice la temperatura. Este dispositivo nos proporciona unas características ajustadas a nuestros objetivos, pues conformará un equipo económico, de poco espacio y digital.

- Arduino es Open-Hardware, de código libre, permitiéndonos mejorarlo fácilmente posteriormente añadiendo nuevos sensores y cambiando el código informático programado fácilmente gracias al extenso campo de sensores que compatibilizan con él, pudiéndose obtenerlo con ayuda de la gran comunidad que tiene detrás, y que, al ser libre, no hay problema de ayudarse de código de otros proyectos sin licencia.

Una interfaz gráfica comprensible y de agradable visualización, que nos muestra los parámetros de la temperatura obtenidos, con la graficación del perfil de temperaturas en tiempo real.

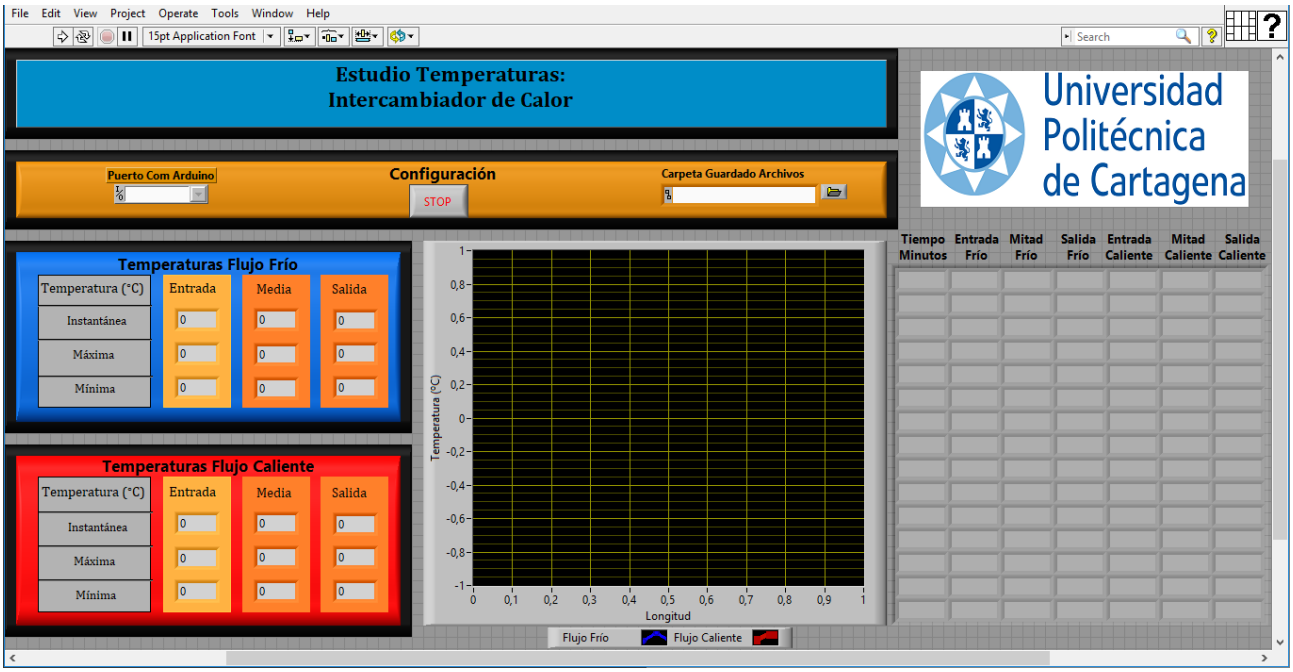

Figura 1.6 - Interfaz Gráfica

- Se ha ganado en precisión y sensibilidad de las medidas, gracias a la actualización de los sensores obsoletos que se poseían en el anterior equipo.

- La monitorización obtenida en cada ensayo se compacta y se da salida una vez finalizado el muestreo, en una hoja de Excel estructurada, que será de fácil extracción por el equipo humano y que les permitirá posteriormente trabajar con ellos.

Capítulo 2 Instrumentación.

La instrumentación del proyecto es una de las partes fundamentales del proyecto a ejecutar. Tanto la elección de los componentes de la parte física como la elección del software se elegirán bajo las condiciones explicadas anteriormente con la finalidad de cumplir los objetivos marcados.

2.1.Componentes físicos

En este apartado se estudiarán los componentes físicos a utilizar, tales como el microcontrolador y los sensores de medición con sus características, elección y mercado.

2.1.1. Microcontrolador Arduino.

El paso más importante de la instalación del equipo es método de conexión entre los componentes de medición y la computadora que se encargará de mostrar y almacenar los resultados obtenidos por esos elementos.

Dentro del mercado existen numerosos métodos de conexión, como son los PLC, DAQ y Arduino.

Los PLC (Programmable Logic Controller), son autómatas programables que se usan en el campo de la automatización para controlar y automatizar procesos de maquinaria y fabricación, por lo que su función está más enfocados a naves industriales y grandes maquinarias.

Figura 2.1 - PLC

Los DAQ (Data Acquisition System) son módulos encargados de generar datos que serán procesados por ordenadores para procesarlas y obtener señales de medida. Este módulo de adquisición de datos es el encargado de transformar las señales físicas convertidas en tensión eléctrica y transformarlas en señales digitales. El principal problema de estos módulos es que vienen compactos y en comparación con Arduino, éste último puede usarse como una DAQ al conectarse a una computadora, pero tiene sus elementos diferenciados y separados y es mucho más flexible.

Figura 2.2 - DAQ

Arduino es un hardware libre basado en una placa formada por un microcontrolador y un entorno de desarrollo de programación que facilita el uso de elementos electrónicos como sensores de todo tipo, actuadores u otro tipo de placas acopladas, pudiendo trabajar en proyectos multidisciplinares.

Los motivos por los que se ha elegido Arduino como herramienta para la adquisición y comunicación es la sencillez y la facilidad de uso de su software de programación y su lenguaje programable, ya que cuenta con su entorno programable en ordenador y en un código sencillo como es C. Por otro lado, es una herramienta económica y de software libre, lo que nos abaratará costes, formando un equipo de bajo coste y muy capacitado. También se tiene en cuenta la flexibilidad que tiene para trabajar multidisciplinarmente, pudiéndose usar en un futuro en otros proyectos del laboratorio o implementando mejoras para el estudio del intercambiador.

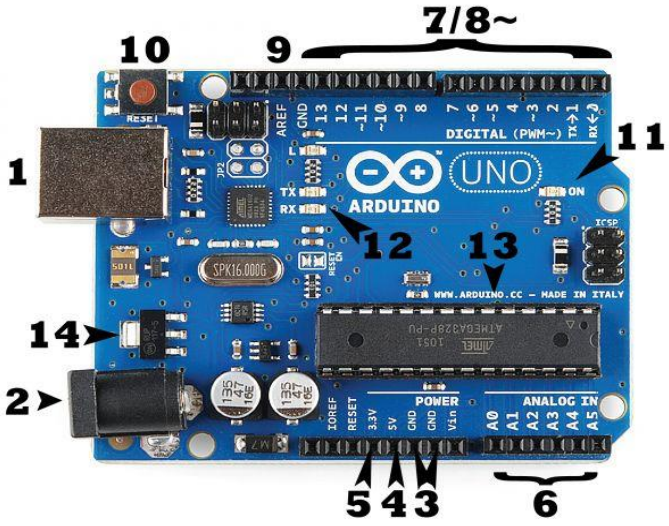

Figura 2.3 - Descripción Arduino

Como se puede observar en la figura, toda placa consta de:

- **Conector USB (1) y conector a alimentación externa:** Toda placa debe estar conectada a una fuente de alimentación externa. Puede ser a través de la conexión USB a una computadora si se mantiene la conexión o a una fuente externa si el equipo está trabajando de forma autónoma. Además, el conector USB es el usado para transferir el código a ejecutar a la memoria flash.

- **Conexión a tierra (3):** Pin para conectar nuestro circuito a tierra.

- **Conexión voltaje (4,5):** Pin que suministra 5 o 3.3 voltios a nuestro circuito.

- **Pines analógicos (6):** Pines para señales analógicas.

- **Pines digitales (7,8):** Pines con señal digital tanto de entrada como de salida. Se diferencia un subgrupo PWM con capacidad para modular su señal por ancho de pulso.

- **Microcontrolador (13):** Circuito integrado encargado de hacer funcionar la placa. Suele ser de la marca Atmel. Está compuesto por una CPU (central processing unit), memorias (RAM y ROM). La CPU es el cerebro central del microprocesador y actúa bajo control del programa almacenado en la memoria. La CPU se ocupa básicamente de traer las instrucciones del programa desde la memoria, interpretarlas y hacer que se ejecuten.

Selección de la placa Arduino

Dentro del campo de Arduino existe una gran variedad de placas con diferentes características, dependiendo de las necesidades físicas y de computación que tenga el proyecto a desarrollar.

Estas diferencias vienen expuestas en la diferencia de pines de entrada, tanto analógicos como digitales, lo que influirá en la cantidad de elementos a conectar a nuestra placa, tantos elementos de medida como elementos de salida a controlar.

Otro elemento a tener en cuenta son las memorias RAM y flash, la memoria flash es la encargada de almacenar el código programado, el cuál ejecutará nuestro procesador y realizará las tareas programadas para nuestro proyecto. Este elemento influirá dependiendo de la complejidad del código programado y si la capacidad de la memoria es suficiente para almacenarlo.

Por otro lado, la memoria RAM es la encargada de cargar los datos durante el funcionamiento del código en tiempo real, afectando a la velocidad de ejecución del código almacenado en la flash. Se tendrá en cuenta ya que las medidas de temperatura se efectuarán cada diez segundos.

Placas de arduino en el mercado:

					
Fabricante	Arduino	Arduino	Arduino	Arduino	Arduino
Modelo	Pro Mini	Nano	Uno	Mega / Mega 2560	Leonardo
Microcontrolador	AVR Atmega 168 ó 328 8bits	AVR ATmega 168 ó 328 8bits	AVR ATmega 328 8bits	AVR ATmega2560 8bits	AVR ATmega 32u4 8bits
Frecuencia	16Mhz	16Mhz	16Mhz	16Mhz	16Mhz
Memoria RAM	2KiB	2KiB	2KiB	8KiB	2.5KiB
Memoria EEPROM	1KiB	1KiB	1KiB	4KiB	1KiB
Memoria FLASH	16 ó 32KiB	16 ó 32KiB	32KiB	128 ó 256KiB	32KiB
Pines digitales entradas/salidas	14/14	14/14	14/14	54/54	20/20
Tensión/corriente pines digitales	3.3v ó 5v 40mA	5v 40mA	5v 40mA	5v 40mA	5v 40mA
Pines analógicos entradas/salidas	6/0	8/0	6/0	16/0	12/0
Tensión/resolución pines analógicos	3.3v ó 5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)
Pines con interrupción externa	2	2	2	6	2
Pines PWM	6	6	6	15	7
Conexiones Serial / UART	1	1	1	4	1
Conexiones I2C / TWI	1	1	1	1	1
Conexiones ISP / ICSP	1	1	1	1	1
Conexión USB	No (necesita adaptador externo)	Si	Si, USB-B	Si, USB-B	Si, Nativa, MicroUSB
Conexión USB de depuración	No	No	No	No	No
Conexión Bluetooth	No	No	No	No	No
Conexión WiFi	No	No	No	No	No
Conexión Ethernet	No	No	No	No	No
Conexión USB Host	No	No	No	No	No
Almacenamiento por SD	No	No	No	No	No
Corriente en el pin de 5v	-	500mA	500~800mA	500~800mA	500~800 mA
Corriente en el pin de 3.3v	-	50mA	50mA	50mA	50mA
Voltaje de alimentación por el USB	3.3v ó 5v (sin usb)	5v	5v	5v	5v
Voltaje de alimentación recomendado por el Jack	3.35 -12 V (modelo 3.3V) ó 5 - 12 V (modelo 5V)	7~12v	7~12v	7~12v	7~12v
Voltaje de alimentación límite por el Jack	-	6~20v	6~20v	6~20v	6~20v
Precio oficial	15+gi	-	20€+gi	40€+gi	18€+gi
Precio BBB	~4€	~9€	~10€	~12€	11€~

Figura 2.4 - Comparativa de microcontroladores

Diferencias más significativas según lo comentado anteriormente:

- Arduino Pro Mini, Nano, Uno y Leonardo:
  - Microcontrolador ATmega 328.
  - RAM de 2 KiB.
  - Memoria flash de 16-32 KiB.
  - Pines digitales entre 14 y 20.
  - Precio aproximado entre 15 y 20 €.
- Arduino Mega 2560:
  - Microcontrolador ATmega 2560.
  - RAM de 8 KiB.
  - Memoria flash de 256 kiB.
  - 54 pines digitales.
  - Precio aproximado de 40€.

Como se puede comprobar, Pro Mini y Nano son placas utilizadas en proyectos muy simples, sin necesidades técnicas importantes. Arduino Uno es una placa destinada a introducirse en el mundo de la electrónica y comenzar proyectos principiantes. Por tanto, y sin que la diferencia económica sea importante, nos decantaremos por una placa Arduino Mega 2560 que nos permita trabajar sin problemas de velocidad de ejecución ni de memoria, y con la capacidad de implementar nuevos proyectos de mejora al laboratorio.

2.1.2. Sensores de temperatura.

El sensor de temperatura va a ser el dispositivo clave de nuestro proyecto, pues va a ser el encargado de tomar y transmitir la medida cuantitativa fundamental para la realización del ensayo que se lleva a cabo en el intercambiador de calor.

Estos dispositivos son capaces de transformar los cambios de temperatura de su entorno, que serán medidos de diferentes formas según el tipo de funcionamiento, y transformarlo en señales eléctricas que podrán mostrarse posteriormente de forma digital a través del mismo dispositivo o enviándose a otro dispositivo o computadora más compleja donde poder trabajar con ellos, visualizarlos, o almacenarlos.

Estos sensores están compuestos generalmente por el elemento sensor capaz de captar la variación de temperatura, la vaina que lo protege y envuelve, esta vaina estará rellena de un material muy conductor de temperatura, que se encargará de transmitir la medición al sensor lo más rápido posible, y del cableado que conectará con el equipo que mostrará la medición.

Dentro de los sensores de temperatura hay tres grandes subgrupos:

- Termistor: Dispositivo basado en la variación de la resistencia de los de óxidos metálicos semiconductores en función de la temperatura.

Existen dos tipos de termistores, los NTC (Negative Temperature Coefficient), compuestos de un material que disminuye su resistencia al aumentar la temperatura, y los PTC (Positive Temperature Coefficient), los cuales al aumentar la temperatura su resistencia también aumenta.

Estos sensores tienen como su principal problema el no ser lineales según la temperatura, por lo que provocará complejidad para determinar la temperatura y son difíciles de calibrar.

Como grandes ventajas sobresale su gran rango de temperaturas de medición y el ser muy conocidos, lo que favorece su compatibilidad con otros dispositivos. También favorece un gran abanico de tamaños y estabilidad.

Como se ha comentado, su gran desventaja es la no linealidad, que como se puede observar sigue una curva logarítmica:

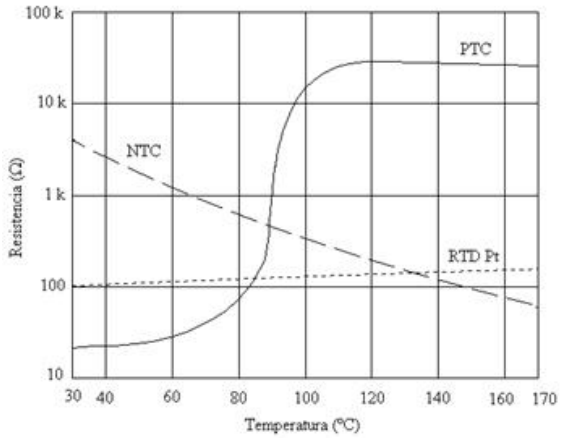

Figura 2.5 - Variación de la resistencia con la temperatura en termistores

- Termopar: Dispositivo formado por dos metales, cuyo principio de funcionamiento es el efecto termoeléctrico. Por medio de este principio, el sensor puede transformar el calor en electricidad directamente, o generar frío al recibir electricidad inversamente.

El funcionamiento termoeléctrico consiste en la generación de un voltaje debido a la diferencia de temperaturas entre los metales, estando uno de ellos a una temperatura de referencia. A partir de ese voltaje generado conoceremos la temperatura.

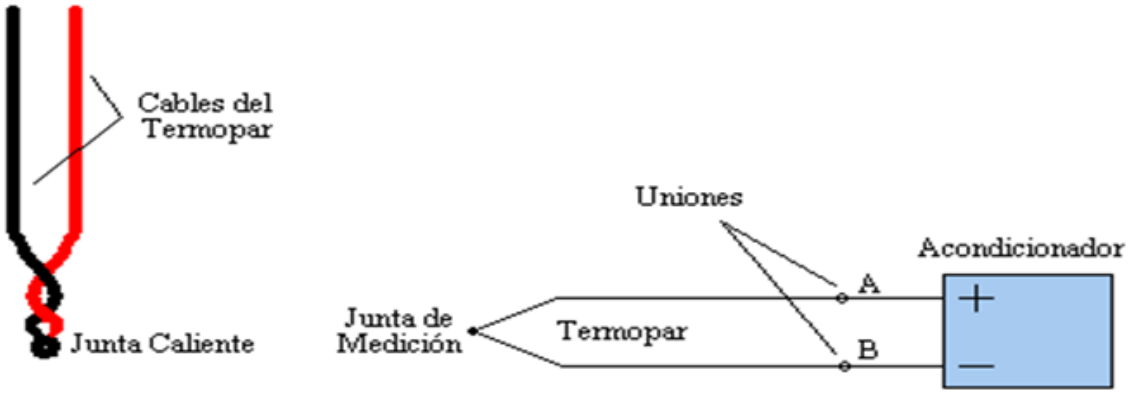

Figura 2.6 - Descripción de un termopar

Son dispositivos económicos, están asentados en el mercado y tienen un amplio rango de medida. Por el contra, tienen poca precisión en comparación con los otros dos subgrupos de sensores.

- Termorresistencia RTD (Resistive Temperature Detector): Se basa en la variación de la resistencia de un conductor con la temperatura. Se suelen emplear metales como platino, cobre y níquel. Los de platino son los de mejor linealidad, rapidez y rango de temperatura.

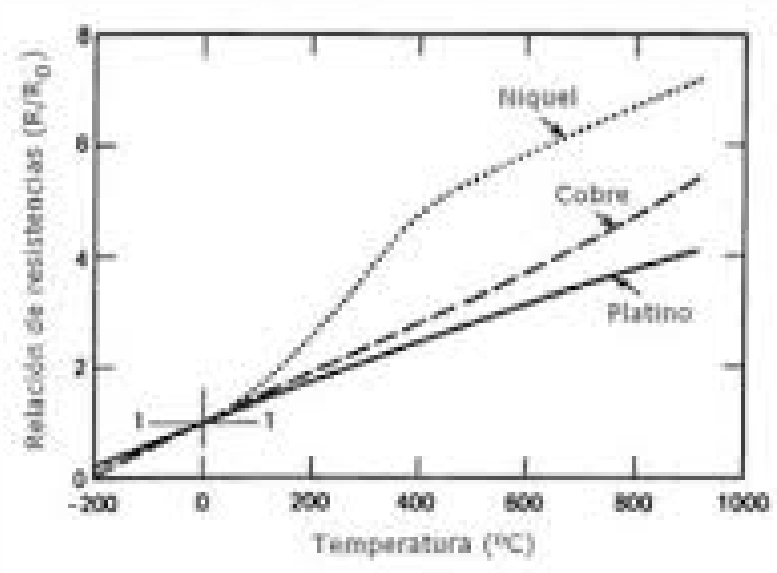

Figura 2.7 - Variación de la resistencia con la temperatura en diferentes metales

Una vez introducidos los tipos de sensores de temperatura y expuestas sus características principales, seleccionaremos el sensor de temperatura DS18B20, puesto que es el sensor que más se adecua a nuestras necesidades, expuestas a continuación:

- Sensor sumergible, ya que nuestra instalación de monitorización de temperaturas es un intercambiador de calor, por lo que el sensor deberá estar en contacto con los fluidos del interior.

- Gran precisión de medida, buscando una mayor calidad de medidas al actualizar el equipo.

-Amplio rango de temperaturas. Para tener la capacidad de medir en ensayos entre 0 y 80 ºC.

- Linealidad de la respuesta de las medidas. Empleo de sensores lineales entre temperatura y dispositivo.

-Dimensiones pequeñas, capaces de ser introducidos en los huecos de medida de los tubos del intercambiador de calor.

- Precio económico, puesto que se han de comprar seis sensores.

Sensor DS18B20

Este sensor tiene dos modelos disponibles, en seco y sumergible.

Figura 2.8 - Sensor DS18B20 para seco y sumergible

Como características principales que le hace el más apto para nuestro equipo, además de tener el modelo sumergible más conocido dentro del mercado, es un sensor muy versátil, su forma alargada nos permite una buena adaptación en el intercambiador de calor y tiene un precio asequible en el mercado.

Posee un gran rango de medidas comprendido entre -55 y 125 grados Celsius como temperaturas más extremas, pero no todo el rango tiene el mismo error debido a factores externos, alteraciones del medio o al circuitado eléctrico. Para el rango intermedio de -10 y

85ºC tiene una tolerancia al error de ±0,5 grados y para los rangos extremos restantes entre esos -55 y 125ºC el error es de ±2ºC.

Otra característica importante es la resolución del sensor, el cual nos permite trabajar en un rango variable de resoluciones, ajustando la precisión, esto es, la variación mínima medible entre dos temperaturas. El rango varía entre 9 y 12 bit. Está resolución podrá ajustarse en el código programable posteriormente.

Resolución	Temperatura (ºC)
9-bit	0,5
10-bit	0,25
11-bit	0,125
12-bit	0,0625

Figura 2.9 - Tabla resoluciones DS18B20

Otra característica específica del sensor es su protocolo de comunicación 1-Wire. Este protocolo nos permite trabajar con múltiples sensores DS18B20 a través de un solo cable conectado a un pin de entrada de Arduino, necesitando, además, una sola resistencia conectada a los múltiples sensores, esto nos ahorrará circuito físico, tanto de cableado como de resistencias, pues si no tendríamos que colocar una resistencia por sensor. Un pequeño inconveniente, por el contrario, es que necesitaremos conocer las direcciones físicas de cada sensor para manipularlo a nuestro interés. Este protocolo es muy interesante y es un punto clave para la elección de nuestro sensor, ya que queremos trabajar con esta tecnología para profundizar en ella.

Está compuesto por el encapsulado impermeable y por el cableado que nos permite conectar al microcontrolador. Este cableado está formado por tres pines de salida:

- DQ: Es el pin que transfiere los datos a nuestro programa a través del protocolo OneWire. Suele ser un cable amarillo.
- GND: El sensor se conectará a la toma de tierra por medio de este pin. Cable negro.
- V_{DD}: La conexión a la tensión de alimentación, haciendo que el sensor funcione al alimentarse a los 5V de salida de la placa. Cable rojo.

Figura 2.10 - Pines de conexión del sensor DS18B20

Protocolo OneWire

Como se ha comentado, este protocolo nos permite la conexión de múltiples sensores de temperatura a través de un único pin de entrada de arduino.

Para realizar este circuito y programarlo, debemos tener en cuenta varios factores:

- Dirección única sensor: Todo sensor DS18B20 incorpora una memoria interna de 8 bytes para almacenar un identificador único. Es de vital importancia conocer esta dirección en cada uno de los sensores a utilizar, ya que, aunque la conexión de estos sensores sea en fila, el orden de recepción de los datos que contienen las medidas no seguirán ese orden, por lo que la primera medida recibida no pertenecerá al primer sensor de la conexión.
  Este código almacenado, además de darnos la opción de identificación, nos puede dar información relativa a características internas del sensor, y nos permite verificar errores de redundancia cíclica (CRC) en los datos.
  En el capítulo de programación de arduino veremos el código para identificar cada sensor y el resultado obtenido.

- Librerías específicas de programación: Es necesario instalar las librerías de programación OneWire y DallasTemperature, dentro del entorno programable de Arduino, para poder programar los sensores. En el capítulo de programación de arduino veremos la instalación de estas librerías y su uso en los comandos del código.

- Conexionado OneWire: Conocido el cableado del sensor, con los tres pines a tierra, alimentación y pin de datos, es necesario conocer la forma de conectar múltiples sensores mediante este protocolo. Existen dos formas válidas de hacerlo, a través del pin V_{DD}, o a través del pin de datos DQ. En ambos métodos se requieren los mismos elementos y cableado, y ambas configuraciones detalladas a continuación se programan igual. La resistencia, que hará la función de pull-up, mantendrá el pin de datos en modo alto, para mantener la tensión cuando los sensores no manden datos, tendrá un valor de 4,7 KΩ.

  - Alimentación por V_{DD}:
    Todos los sensores comparten la misma resistencia pull-up del pin DQ. Por otro lado, una de las patillas irá a tierra y la otra a alimentación.

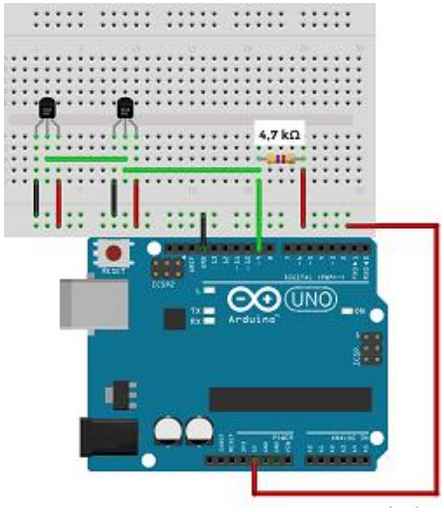

Figura 2.11 - Conexión protocolo OneWire tipo VDD

- Alimentación por pin DQ (modo parásito):

Los sensores están enlazados por sus pines de datos, como en el ejemplo anterior, pero las patillas restantes (GND y V_{DD}), son conectadas a tierra.

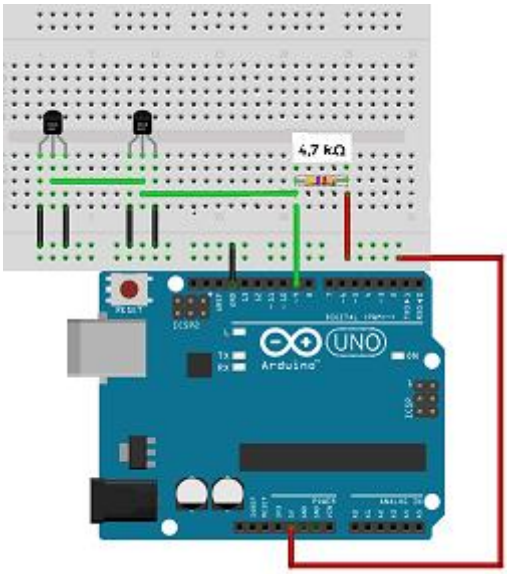

Figura 2.12 - Conexión protocolo OneWire modo parásito

2.2.Conexionado de los componentes físicos

Nuestro equipo cuenta con el microcontrolador arduino mega 2560, un protoboard o placa de conexionado, la resistencia pull-up de 4,7KΩ y seis sensores DS18B20 sumergibles que serán colocados en la entrada, punto medio y salida de los dos tubos del intercambiador de calor.

El circuito será conectado siguiendo el protocolo OneWire visto anteriormente, en el modo parásito o de alimentación por pin DQ.

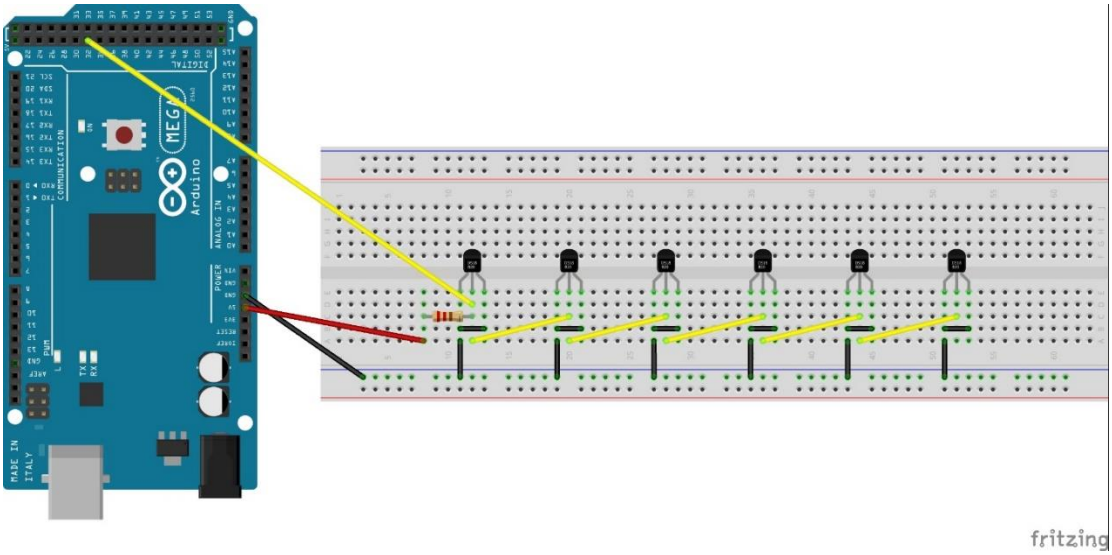

Figura 2.13 - Esquemático de conexión de nuestro circuito

Los componentes son los siguientes:

- Microcontrolador arduino mega 2560:

Figura 2.14 - Microcontrolador Arduino Mega de nuestro circuito

- Resistencia pull-up de 4,7 KΩ:

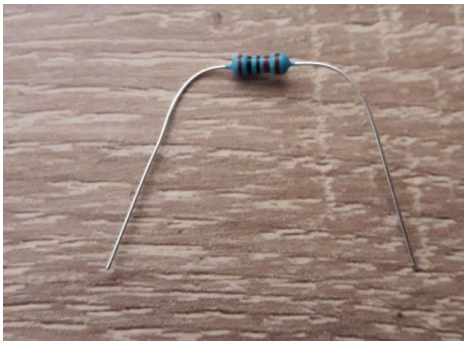

Figura 2.15 - Resistencia Pull up de nuestro circuito

- Sensor sumergible DS18B20:

Figura 2.16 - Sensor DS18B20 de nuestro circuito

Tras ejecutar este esquemático, cuya entrada de datos se refleja al pin 32, aunque es indiferente la entrada siempre y cuando sea una entrada digital, queda como resultado el siguiente montaje.

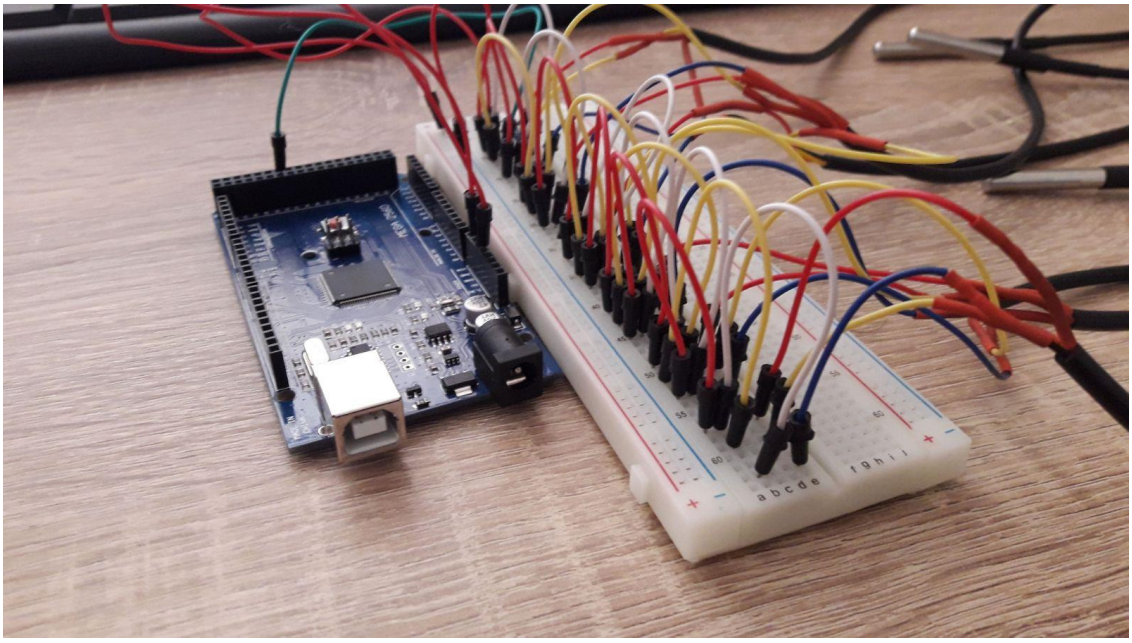

Figura 2.17 - Montaje del circuito

Este montaje básico, puede plantear problemas de conexión, rotura de elementos y dificultad a la hora de manipular el circuito en caso de fallo, se plantea trasladar el montaje a una placa stripboard con los elementos soldados, simulando un circuito impreso, ahorrando todo el cableado planteado en el montaje.

La nueva placa posee la resistencia de 4,7kΩ, y una secuencia de conectores hembra de tipo dupont de tres entradas para cada sensor.

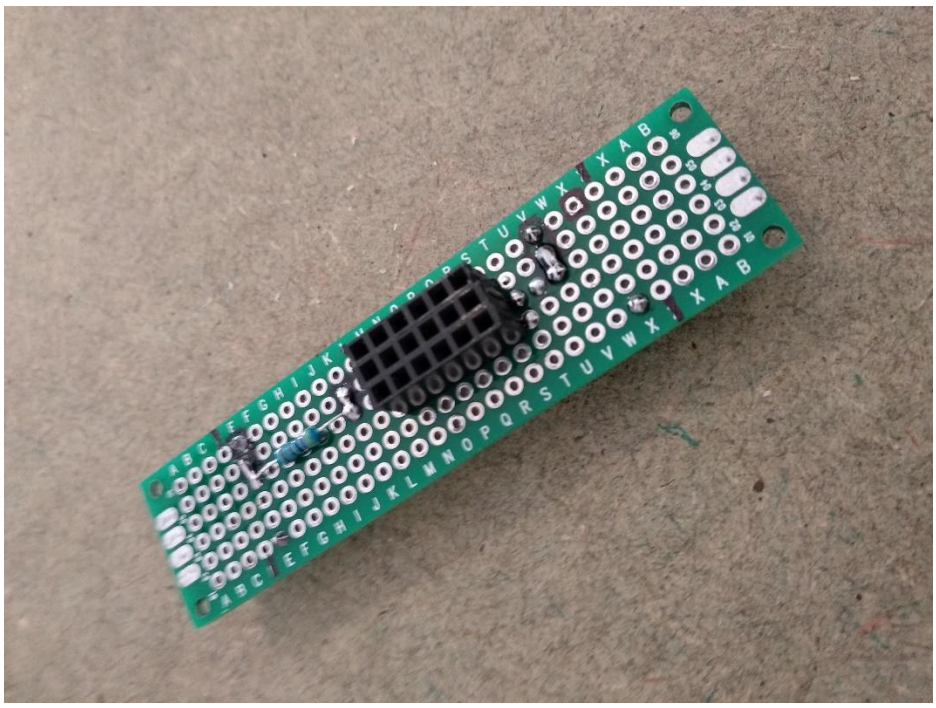

Figura 2.18 - Parte superior de la placa stripboard sin cableado

La parte inferior de la placa contiene las conexiones soldadas de los sensores, creando un cable de datos único central, y dos filas de conexión a los lados que finalmente conecta con el GND. Por último, un pin transfiere la información al pin digital de entrada, y otro se conecta a los 5V.

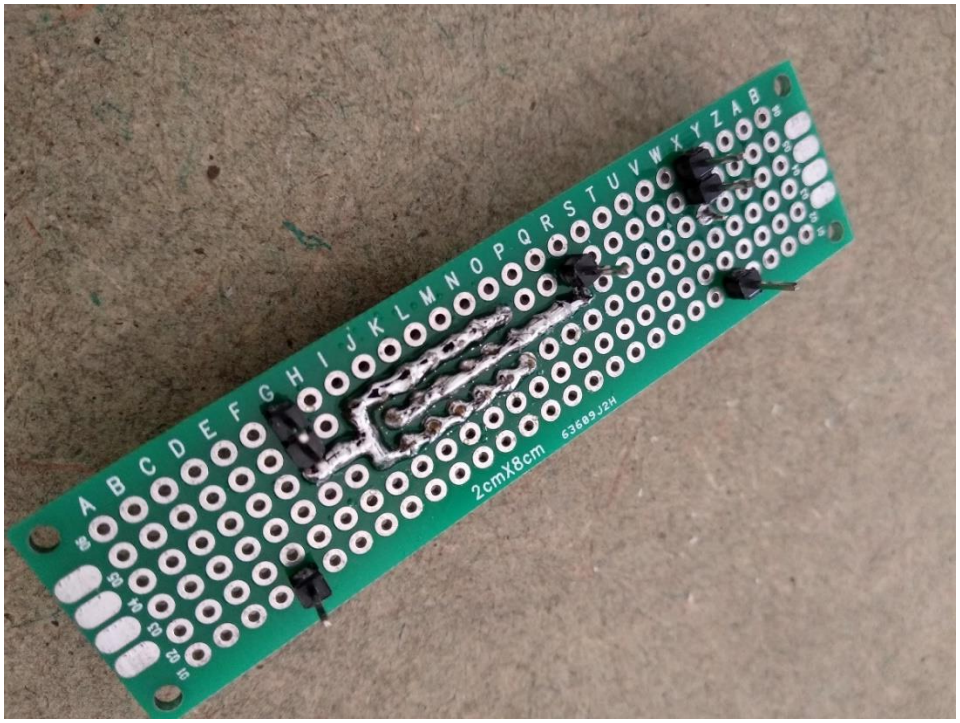

Figura 2.19 - Parte inferior de la placa stripboard sin cableado

El montaje final del equipo Arduino queda compactado, más vistoso y evitando futuros errores de conexión.

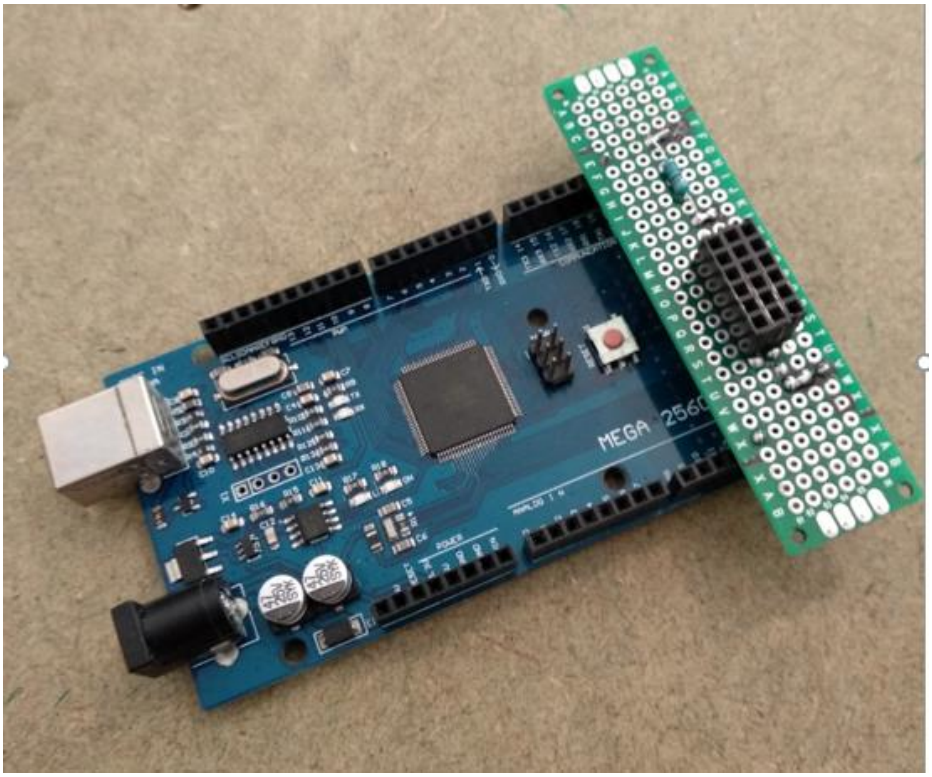

Figura 2.20 - Conexión stripboard con Arduino Mega

2.3.Software como instrumentación del proyecto.

Como parte esencial de la instrumentación en ingeniería encontramos los softwares, o programas capaces de controlar, monitorizar, almacenar o realizar cálculos a partir de las variables creadas, calculadas, o adquiridas a través de sistemas de control automático, mediante dispositivos electrónicos como anteriormente se ha visto, en el mundo físico.

Para nuestro proyecto, será necesario el uso de dos softwares o programas, uno que nos dé acceso a la programación del equipo físico, y sea capaz de manejar y transferir el código programado al microcontrolador, y otro que se usará para la monitorización en tiempo real mediante un entorno gráfico que crearemos bajo nuestras necesidades, de las variables, en nuestro caso las medidas de temperatura, que serán registradas por nuestro equipo electrónico.

Capítulo 3 Arduino como herramienta de adquisición de datos

Arduino realizará la función de DAQ, o herramienta de adquisición de datos para ser procesados y estudiados en tiempo real por el software de monitorización.

Nuestro microcontrolador estará conectado en todo momento con la computadora que mostrará las medidas tomadas en tiempo real a través de su conexión USB, y funcionando en todo momento con el código programado que será almacenado en la memoria flash mencionada anteriormente.

La empresa Open-Source dispone de un software libre, usado como entorno de desarrollo (IDE), basado en Proccessing/Wiring y bootloader (cargador de arranque), que nos facilitará, posterior a la escritura del código, el volcado del mismo en la memoria. Este código ya no necesitará volver a cargarse, excepto si se modifica el código programa a cause de una mejora de la instalación u otros factores.

3.1.Instalación de Arduino

El software IDE de Arduino es de libre, al ser una empresa Open-Source, por lo que se puede descargar fácilmente de la página oficial de Arduino la última versión disponible. Para la realización de este proyecto se ha usado la última versión 1.8.5.

El enlace de descarga es : https://www.arduino.cc/en/Main/Software

Tras la descarga y posterior instalación del software, tenemos el siguiente entorno de desarrollo:

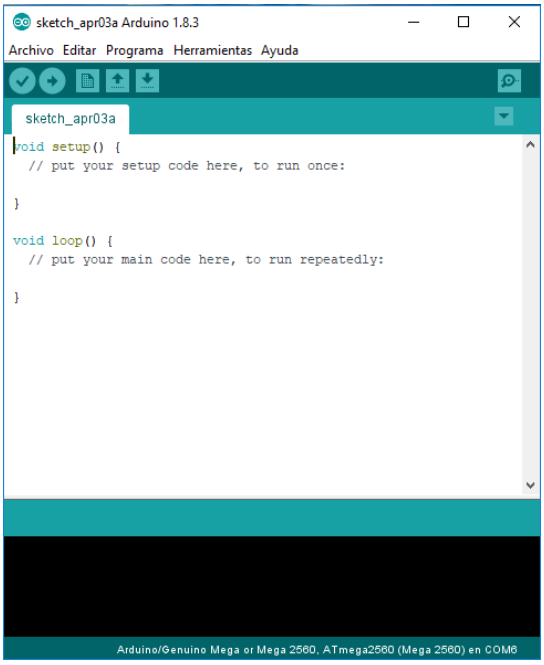

Figura 3.1 - IDE de Arduino

Una vez se tiene el software instalado, se configura para que detecte la tarjeta arduino del equipo:

1. Lo primero que tenemos que hacer es conectar la placa al ordenador por medio del USB.

2. Si los drivers del microcontrolador no se instalan directamente, tendremos que abrir el administrador de dispositivos de Windows dentro de las propiedades de Equipo/Propiedades/Administrador de dispositivos:

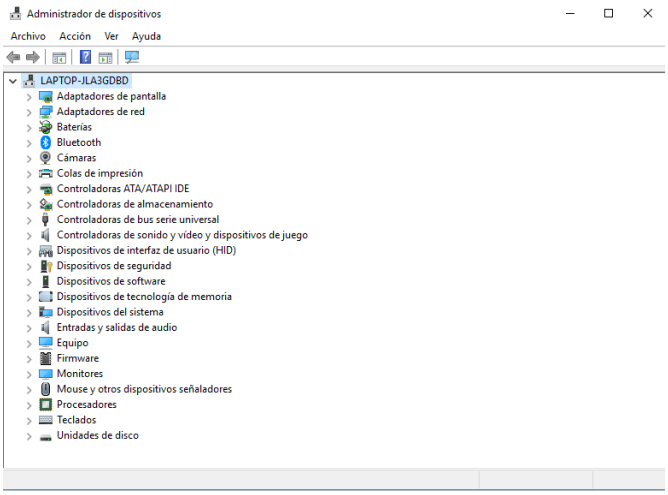

Figura 3.2 - Administrador de dispositivos

En dispositivos desconocidos, aparecería nuestro dispositivo:

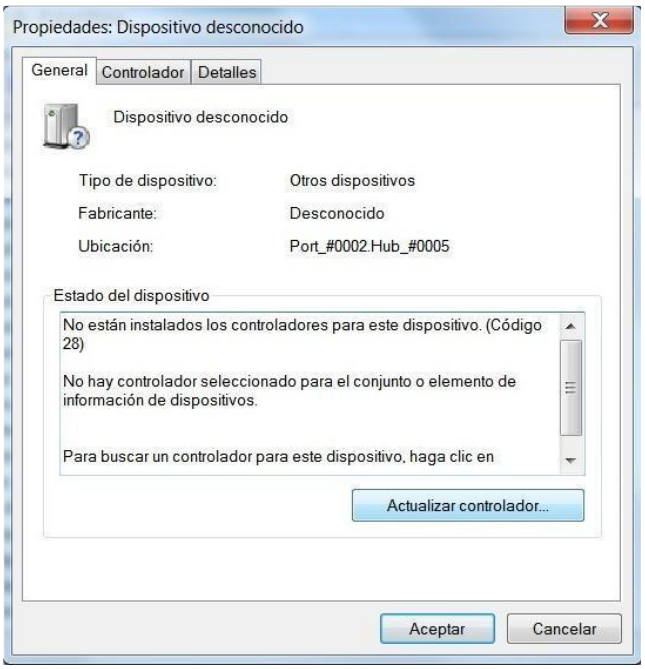

Figura 3.3 - Dispositivo Desconocido

Actualizaríamos el controlador buscando dentro de la carpeta drivers de la carpeta raíz de Arduino:

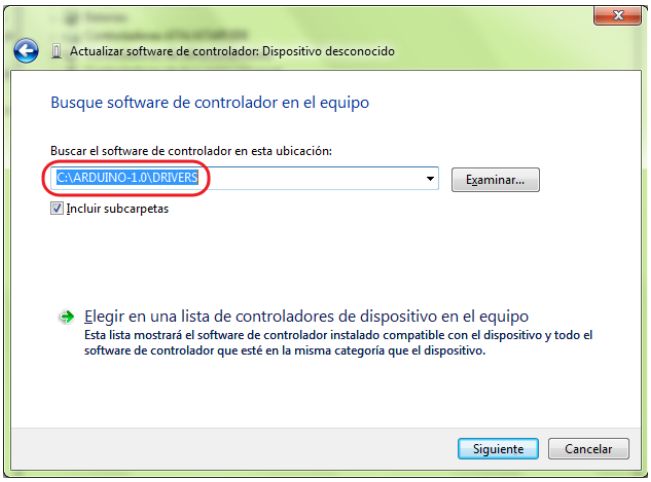

Figura 3.4 - Actualizar drivers

3. Una vez instalado el software y los drivers de la placa, a la que se le ha asignado un puerto COM del ordenador por defecto, debemos configurar dentro de nuestro entorno la placa en la que trabajaremos, dentro de herramientas en las opciones superiores:

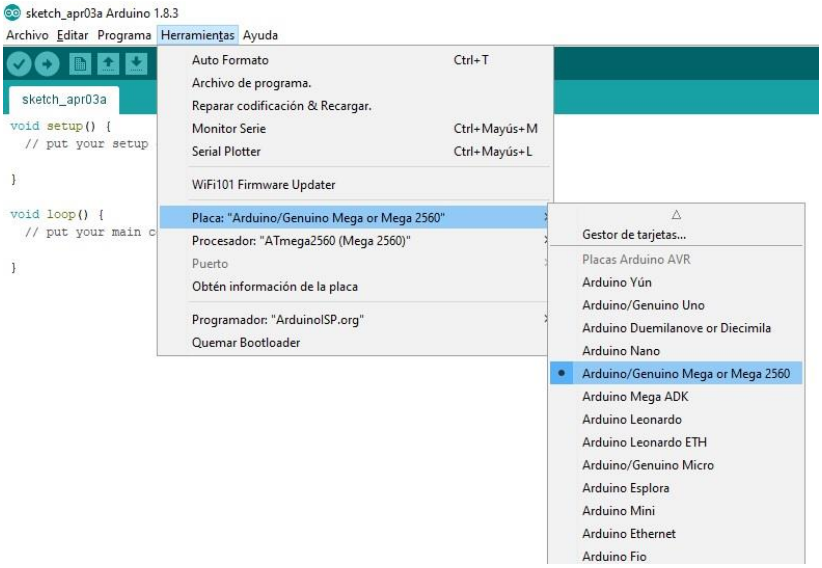

Figura 3.5 - Selección del microcontrolador utilizado

4. Y seleccionamos también el procesador ATmega2560 (Mega 2560), que se encuentra en la opción de debajo de la selección de placa del paso anterior.

5. Por último, debemos seleccionar el puerto COM al que se ha conectado por defecto nuestra placa.

Tras la realización de estos pasos, la instalación del IDE quedaría finalizada, y sólo habría que repetir el último paso en caso de desconectar la placa del ordenador y conectarla posteriormente en otro puerto del ordenador.

3.1.1. Librerías Arduino

Tras la instalación del IDE, antes de programar, es necesario instalar las librerías útiles para el uso del protocolo OneWire en nuestro código.

Las librerías son paquetes de código implementado por terceros, que nos facilitan la programación, haciéndonos más sencillo el trabajo y más entendible. Al ser Arduino Open-Source no hay problemas de derechos de autor y son accesibles.

Las librerías a instalar son las siguientes:

- Librería OneWire: Este paquete implementa el protocolo de conexión. Se descarga desde el enlace siguiente. https://www.arduinolibraries.info/libraries/one-wire

- Librería DallasTemperature: Esta librería contiene comandos que nos facilitan los códigos adecuados para controlar el sensor y obtener la temperatura. Se puede obtener desde el siguiente enlace. https://www.arduinolibraries.info/libraries/dallas-temperature

Las librerías son descargadas en formato comprimido .Zip. Una vez descargas y localizadas en nuestro directorio, se procede a su instalación manual.

Dentro del entorno IDE de Arduino, en la pestaña superior 'programas', tenemos la opción de incluir librería, además de visualizar las que vienen por defecto o instaladas anteriormente.

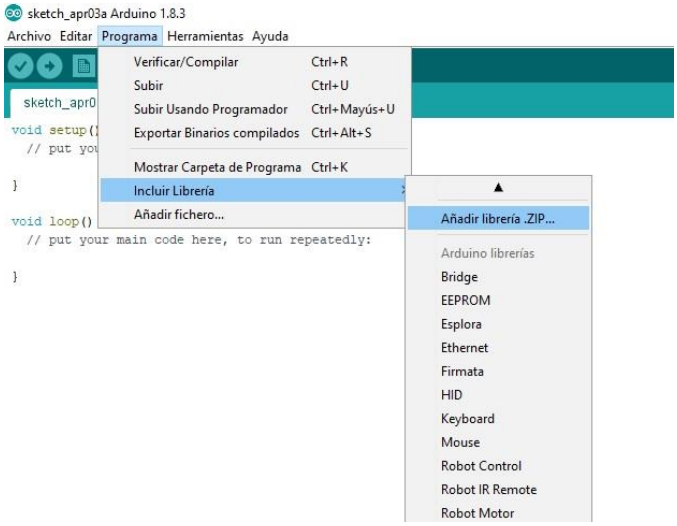

Figura 3.6 - Incluir librerías

Al pinchar en 'añadir librería.ZIP' se nos abre un explorador de archivos, donde buscaremos nuestras librerías descargadas y se instalan automáticamente.

Una vez instalada una librería ya no es necesario volver a realizar los pasos de instalación, tendremos un apartado de ejemplos de códigos para implementar la librería en cuestión, y podremos hacer uso de ella en nuestro código.

3.2.Identificación única de los sensores

Antes de proceder con el desarrollo del código para la obtención del código, debemos ser capaces de identificar y poder controlar la información recibida de cada sensor, ya que todos los datos son transferidos a través del pin de entrada, y el orden de recepción de dichos datos no concuerda con el orden de colocación de los sensores en el circuito.

Como se dijo anterior, cada sensor posee un identificador único integrado en su memoria, que nos concede la posibilidad de hacer una llamada específica a él dentro del código que programemos.

Para la identificación de dicho código, realizamos una conexión individual para cada sensor, ejecutando un código personalizado que nos mostrará su dirección identificativa:

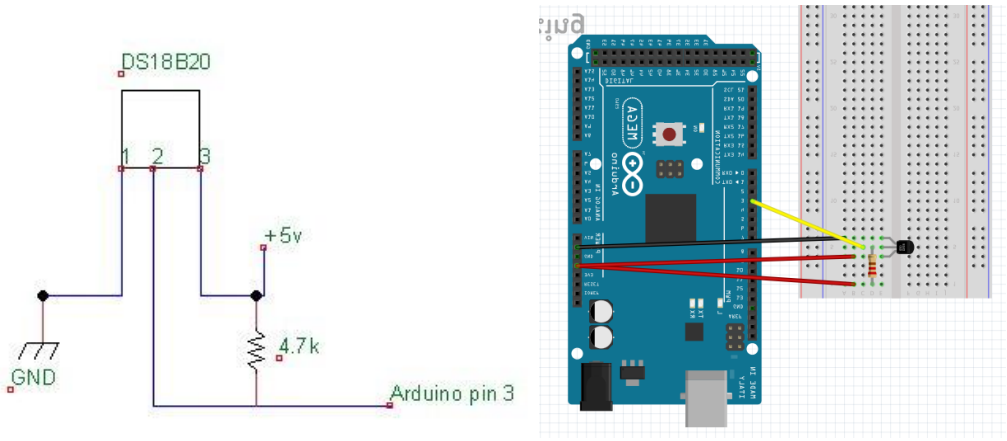

Figura 3.7 - Conexión individual de un sensor DS18B20

Y ejecutamos el siguiente código, percatándonos del pin al que se ha conectado el sensor:

```
#include <OneWire.h>
OneWire  ds(3);  // Referir al pin de conexión
void setup(void) {
  Serial.begin(9600);
  discoverOneWireDevices();
}
void discoverOneWireDevices(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];
  Serial.print("Buscando dispositivos...\n\r");
  while(ds.search(addr)) {
    Serial.print("\n\rEncontrados \'1-Wire\' dispositivos con la
dirección:\n\r");
    for( i = 0; i < 8; i++) {
      Serial.print("0x");
      if (addr[i] < 16) {
        Serial.print('0');
      }
      Serial.print(addr[i], HEX);
      if (i < 7) {
        Serial.print(", ");
      }
    }
    if ( OneWire::crc8( addr, 7) != addr[7]) {
        Serial.print("CRC no es válido!\n");
        return;
    }
  }
  ds.reset_search();
  return;
}
void loop(void) {
  // Nada
}
```

Lo primero que se ha hecho es incluir la librería que contiene el protocolo OneWire. Posteriormente se le informa de que se aplique el protocolo en el pin al que hemos conectado el DQ del sensor, y realiza un ciclo while:

Mientras encuentre dirección, que buscará mediante el comando 'ds.search(addr)', mostrará en pantalla la dirección encontrada, que hemos almacenado a través de un array. Con la variable i=0, y de uno en uno, hasta que sea menor que i<8, mostraremos el valor del array addr[i], retocándolo con el dato 0x, que será necesario para su uso posterior a la hora de definirlo.

El resultado del código mencionado, muestra como salida por pantalla lo siguiente:

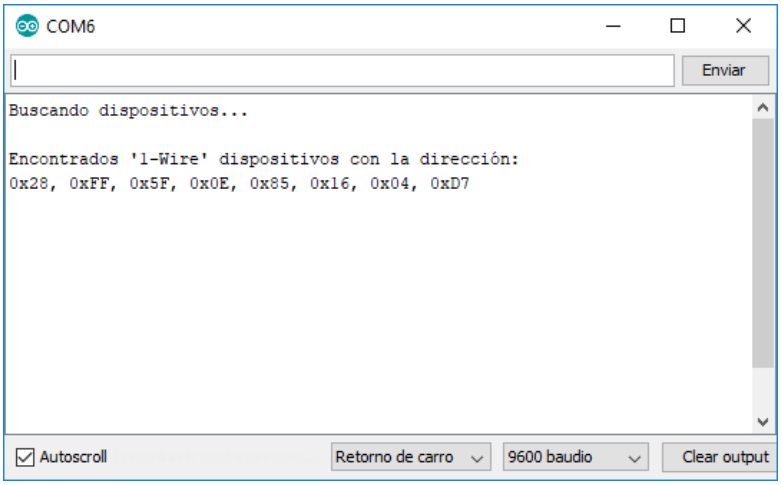

Figura 3.8 - Salida por pantalla del IDE de Arduino con la dirección del dispositivo

Realizando este paso para los seis sensores a utilizar en nuestro proyecto, obtenemos las siguientes direcciones:

Nº externo para la identificación del sensor	Dirección del sensor
Sensor 1	0x28 0xFF 0x5F 0x0E 0x85 0x16 0x04 0xD7
Sensor 2	0x28 0xFF 0x91 0xC5 0x85 0x16 0x05 0xA4
Sensor 3	0x28 0xFF 0x25 0xC5 0x85 0x16 0x05 0xF4
Sensor 4	0x28 0xFF 0xE6 0x09 0x85 0x16 0x04 0xED
Sensor 5	0x28 0xFF 0x2B 0x2C 0x85 0x16 0x04 0xD2
Sensor 6	0x28 0xFF 0xAB 0xDF 0x85 0x16 0x05 0xBE

Figura 3.9 - Tabla de direcciones identificativas de nuestros sensores

Cada sensor se ha identificado externamente, con un nombre clave para identificarlos a la hora del montaje y en el código del sketch de Arduino.

Nº externo para la identificación del sensor	Localización en la instalación
Sensor 1	Entrada flujo frío
Sensor 2	Mitad flujo frío
Sensor 3	Salida flujo frío
Sensor 4	Entrada flujo caliente
Sensor 5	Mitad flujo caliente
Sensor 6	Salida flujo caliente

Figura 3.10 - Localización de los sensores en el intercambiador

Está identificación nos permitirá, independientemente del orden de conexión del circuito, que la programación funcione, y que los datos pertenecientes de las medidas se tomen en el orden correcto para su monitorización en el programa usado para ello.

3.3. Desarrollo de nuestro software de Arduino

Una vez realizada la instalación del software, de las librerías necesarias y de la identificación de los sensores, además de haber visto como realizar el montaje, estamos en plena capacidad de desarrollar el código de programación para los sensores, este código programa la toma de las medidas de temperatura por parte de los sensores, el tratamiento de los datos, y la orden de envío a través del puerto serie, para su posterior monitoreo por parte del programa que se encargará de ello.

El código es continuo, pero se va a ir describiendo por bloques funcionales.

Lo primero a realizar es incluir las librerías instaladas en nuestro código e indicamos a que bus se le ha conectado el pin DQ que transmite la información.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 32
```

El siguiente paso es aplicar las dos librerías al bus que se ha indicado anteriormente.

```
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

Una vez incluidas y aplicadas las librerías donde nos interesa, declaramos las direcciones de los sensores mediante el comando 'DeviceAdress', otorgándole una variable a cada sensor. El nombre de las variables es el definido en el apartado donde se obtenían las direcciones para no incurrir en errores.

```
//Indicamos las direcciones de los sensores
DeviceAddress S1 = {0x28, 0xFF, 0x5F, 0x0E, 0x85, 0x16, 0x04, 0xD7};
DeviceAddress S2 = {0x28, 0xFF, 0x91, 0xC5, 0x85, 0x16, 0x05, 0xA4};
DeviceAddress S3 = {0x28, 0xFF, 0x25, 0xC5, 0x85, 0x16, 0x05, 0xF4};
DeviceAddress S4 = {0x28, 0xFF, 0xE6, 0x09, 0x85, 0x16, 0x04, 0xED};
DeviceAddress S5 = {0x28, 0xFF, 0x2B, 0x2C, 0x85, 0x16, 0x04, 0xD2};
DeviceAddress S6 = {0x28, 0xFF, 0xAB, 0xDF, 0x85, 0x16, 0x05, 0xBE};
```

En el apartado 'setup' del sketch en Arduino, tenemos que colocar las funciones de ejecución única.

Ejecutaremos dentro de este apartado la inicialización de la comunicación con el puerto serie del microcontrolador, indicando entre paréntesis los baudios o velocidad de transmisión de datos, y la inicialización de los sensores.

Por otro lado, definiremos dentro de este apartado la resolución a la que queremos que trabajen cada uno de nuestros sensores. Como se explicó en instrumentación, la resolución en este tipo de sensores es variable. En nuestro caso, le impondremos la máxima resolución posible.

```
void setup()

{

  Serial.begin(115200); //Abrimos la comunicación por serial

  sensors.begin();  //Iniciamos los sensores

//Función que muestra la resolución del sensor de dicha dirección. Las resoluciones posibles pueden ser:

//Resolución a 9 bits 0.50 °C

//Resolución a 10 bits 0.25 °C

//Resolución a 11 bits 0.125 °C

//Resolución a 12 bits 0.0625 °C

  sensors.setResolution(S1, 12);  //Resolución a 12 bits 0.0625 °C

  sensors.setResolution(S2, 12);  //

  sensors.setResolution(S3, 12);  //

  sensors.setResolution(S4, 12);  //

  sensors.setResolution(S5, 12);  //

  sensors.setResolution(S6, 12);  //

} //--(end setup )---
```

El bloque loop es el conjunto de código que se ejecutará en bucle o lazo indefinidamente hasta apagar o reiniciar el microcontrolador. En este apartado se programará el cuerpo del programa, las funciones de obtención y tratamiento de los datos.

Primeramente, mediante la función 'requestTemperatures', preparamos la toma de las medidas de temperatura por parte de los sensores.

Posteriormente, con el comando 'get.TempC' obtenemos la temperatura del sensor que indiquemos dentro de su paréntesis, y lo almacenamos en una variable tipo float, con la misma numeración. Este comando nos devuelve la temperatura en grados Celsius, se puede obtener en las diferentes unidades de medida modificando el comando por 'get.TempF' para grados Fahrenheit.

Tras la operación, el 'delay(100)' hará una pausa de 0,1 segundos, pues viene expresado en milisegundos.

Las variables temp[1-6] con el valor de la temperatura se han guardado como variable tipo float, o coma flotante, útiles para almacenar variables que representen valores numéricos racionales, tanto grandes como pequeños.

```
void loop(void)
{
    sensors.requestTemperatures(); // Comando para tomar las temperaturas.
        float temp1 = (sensors.getTempC(S1));
        delay(100);
        float temp2 = (sensors.getTempC(S2));
        delay(100);
        float temp3 = (sensors.getTempC(S3));
        delay(100);
        float temp4 = (sensors.getTempC(S4));
        delay(100);
        float temp5 = (sensors.getTempC(S5));
        delay(100);
        float temp6 = (sensors.getTempC(S6));
        delay(100);
```

Estas variables tipo float las vamos a multiplicar por mil y almacenar en variables tipo 'long' nombradas con la misma numeración seguida en todo el código. Esta transformación se va a realizar para eliminar la coma decimal de las mediciones, dejando la medida obtenida como número entero, y poder ser leído posteriormente por el programa de monitorización asegurándonos en no caer en un bucle infinito que nos cuelgue el programa indefinidamente.

Las variables tipo `long' almacenan números enteros y están almacenadas en 32bits o 4 bytes de memoria.

Esta transformación quedará explicada más sencillamente con un ejemplo:

> *Sí la temperatura obtenida por el sensor es 5,250 ºC, y la enviamos por el puerto serie para monitorear, el microcontrolador enviará 4 bytes ("5.25"). Por el contrario, si la temperatura medida es 30,250 ºC, el microcontrolador enviará por el puerto COM al ordenador 5 bytes ("30.25"). El problema aparece cuando en el programa de monitorización que se verá más adelante, necesitamos indicar cuantos bytes de información debe esperar el programa en cada ciclo de medida. Si nosotros indicamos que debe recibir 5 bytes y el microcontrolador envía una medida de 4 bytes, el código quedará en espera del byte faltante indefinidamente, colgándonos el programa. Si, por el contrario, indicamos que reciba 4 bytes, la medida será muy poco precisa, algo que no nos interesa.*

Aquí observamos la transformación comentada:

```
// Transformación de las variables.
long temp11;
long temp22;
long temp33;
long temp44;
long temp55;
long temp66;

temp11 = temp1 * 1000;
temp22 = temp2 * 1000;
temp33 = temp3 * 1000;
temp44 = temp4 * 1000;
temp55 = temp5 * 1000;
temp66 = temp6 * 1000;
```

Como último paso, debemos enviar la información de las temperaturas obtenidas al ordenador a través del puerto serie. Esto se realiza por medio del comando 'serial.write' y mandando cada variable que tiene almacenada la medida tras la transformación. Para enviar la medida byte a byte, al constar la variable tipo long con 32 bits, se enviará de 8 bits en 8 bits, dada la equivalencia de 1 byte y 8 bits.

```
// Envío de las variables a través del puerto serie.
Serial.write((temp11 >> 24) & 0xFF);
Serial.write((temp11 >> 16) & 0xFF);
Serial.write((temp11 >>  8) & 0xFF);
Serial.write(temp11 & 0xFF);

Serial.write((temp22 >> 24) & 0xFF);
Serial.write((temp22 >> 16) & 0xFF);
Serial.write((temp22 >>  8) & 0xFF);
Serial.write(temp22 & 0xFF);

Serial.write((temp33 >> 24) & 0xFF);
Serial.write((temp33 >> 16) & 0xFF);
Serial.write((temp33 >>  8) & 0xFF);
Serial.write(temp33 & 0xFF);
```

```
        Serial.write((temp44 >> 24) & 0xFF);
        Serial.write((temp44 >> 16) & 0xFF);
        Serial.write((temp44 >>  8) & 0xFF);
        Serial.write(temp44 & 0xFF);

        Serial.write((temp55 >> 24) & 0xFF);
        Serial.write((temp55 >> 16) & 0xFF);
        Serial.write((temp55 >>  8) & 0xFF);
        Serial.write(temp55 & 0xFF);

        Serial.write((temp66 >> 24) & 0xFF);
        Serial.write((temp66 >> 16) & 0xFF);
        Serial.write((temp66 >>  8) & 0xFF);
        Serial.write(temp66 & 0xFF);

}
```

Una vez realizado el código, sólo nos quedará volcarlo en la memoria del microcontrolador. Esto se realiza mediante el botón del IDE:

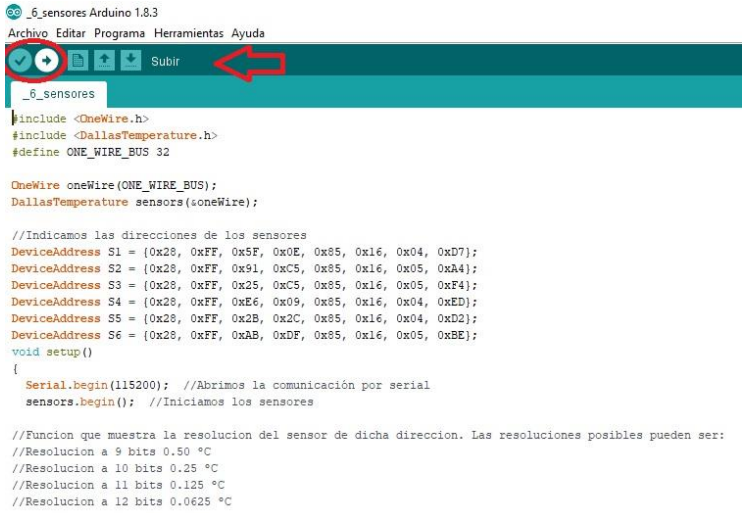

Figura 3.11 - Verificación y vuelco del código en el microcontrolador

Tras el volcado, el código queda almacenado en la memoria flash, y por tanto, no será necesario volver a cargarlo más.

Capítulo 4 LABVIEW como herramienta de monitorización.

4.1.Introducción a los softwares de monitorización.

Tras resolver la programación para el funcionamiento físico de la instalación por medio de Arduino y su IDE, se necesita la aplicación de un programa que sea capaz de procesar, monitorizar en tiempo real y almacenar los datos de las medidas obtenidas por los sensores.

Utilizaremos un software de programación gráfica, que posea la capacidad de crear una interfaz gráfica, y que funcione como un programa esclavo (sólo para monitorización), que nos permita el intercambio de información con el microcontrolador Arduino E/S a través del puerto serie, pero sin modificar el código de funcionamiento de los sensores, esto es, modo esclavo. Esto convierte a Arduino en un DAQ (Data Adquisition System).

Para la elección del software, se tendrá en cuenta que posea un potente gestor de interfaz gráfica, pues buscamos como objetivo que sea ilustrativo durante el periodo de medición, pues se monitoriza en tiempo real para el equipo humano de trabajo, que pueda actualizarse en un futuro, que sea de fácil programación y que estemos en disposición de obtenerlo.

Dentro del mercado encontramos un amplio abanico de posibilidades, como LabWindows CVI, LabView, Pascal, Visual Basic, Agilent-VEE o software libre como MyopenLab, y otros especializados en campos concretos de la ingeniería.

Para nuestro proyecto se ha optado por LabVIEW para el trabajo de monitorización y almacenamiento de las medidas. LabVIEW (Laboratory Virtual Engineering Workbench) es una plataforma y entorno gráfico de desarrollo de sistemas cuyas funcionalidades principales son las de pruebas, control y diseño, de simulación real o embebida, monitorizar, controlar, automatizar y realizar cálculos complejos de señales analógicas y digitales capturadas a través de tarjetas de adquisición de datos, puertos serie y GPIBs (Buses de Intercambio de Propósito General).

Las razones de la elección son:

- Está basado en lenguaje G, un sistema de programación gráfica simple de manejar.
- Programa enfocado a la programación virtual, contando con numerosas herramientas para la implementación de la interfaz gráfica, como pueden ser gráficas, botones, indicadores y controles, los cuales son muy esquemáticos y elegantes.
- Programa con gran versatilidad, contando con librerías especializadas para manejos de DAQ, Redes, comunicaciones, etc.
- Se pueden usar bloques creados anteriormente por otras personas, aprovechándose como subrutinas de nuestro proyecto.
- Se tiene acceso al programa gracias al sistema de programas para la universidad.

4.2. VIs (Instrumentos Virtuales) de LabVIEW

Los programas desarrollados mediante LabVIEW se denominan Instrumentos Virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo, son análogos a las funciones creadas con los lenguajes de programación convencionales.

LabVIEW tiene la característica de descomposición modular ya que cualquier VI que se ha diseñado puede convertirse fácilmente en un módulo que puede ser usado como una subunidad dentro de otro VI. Esta peculiaridad podría compararse a la característica de procedimiento en los lenguajes de programación estructurada, lo que permite crear tareas muy complicadas a partir de módulos o submódulos mucho más sencillos.

Es un sistema abierto, cualquier fabricante de tarjetas de adquisición de datos o instrumentos en general puede proporcionar el driver de su producto en forma de VI dentro del entorno de LabVIEW. También es posible programar módulos para LabVIEW en lenguajes como C y C++, estos módulos son conocidos como Sub- VIs y no se difieren a los VI creados con LabVIEW salvo por la interfaz del lenguaje en el que han sido programados.

4.3.LabVIEW y su entorno de desarrollo

Dentro de todo instrumento virtual en el que nos encontramos trabajando está dividido en dos partes diferenciadas: el Front Panel (Panel Frontal) y el Block Diagram (Diagrama de Bloque). Ambos están enlazados internamente para su correcto funcionamiento y simultaneidad.

Las paletas de LabVIEW son las que proporcionan las herramientas que se requieren para crear y modificar tanto el panel frontal, como el diagrama de bloques.

- **Panel Frontal:**

El panel frontal es la herramienta donde adecuamos la interfaz gráfica que mostrará el programa durante su funcionamiento, la interfaz de usuario. Dentro de esta parte de LabVIEW tendremos una paleta de herramientas específicas de edición para la adecuación del mismo.

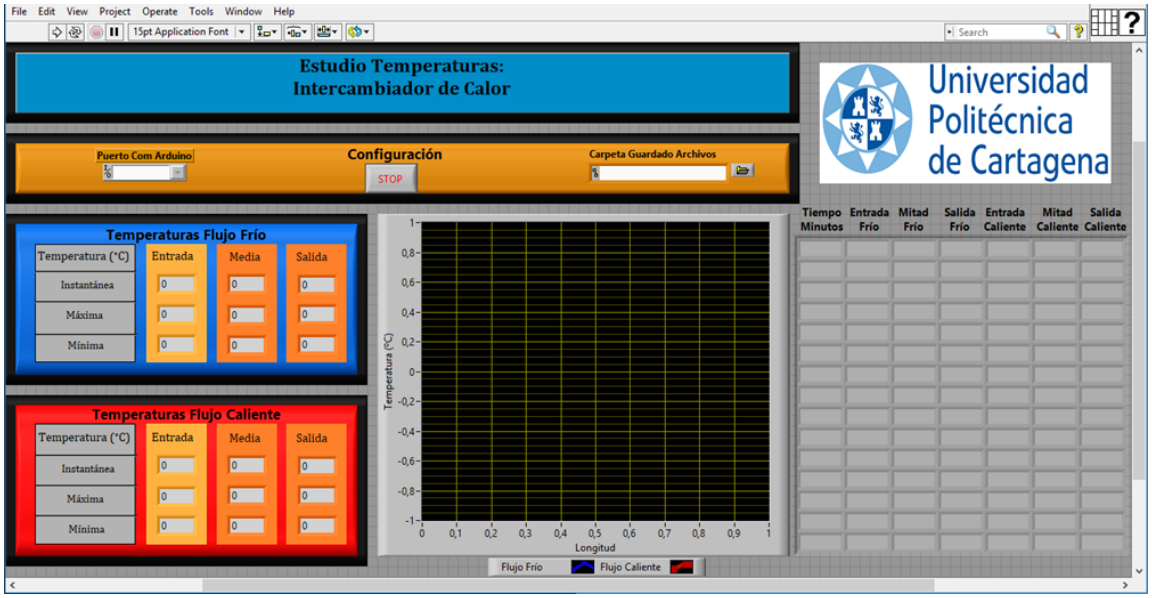

Figura 4.1 - Ejemplo de panel frontal

- **Diagrama de Bloques:**

En el diagrama de bloques está todo el código gráfico programado que define el funcionamiento del instrumento virtual. Dentro del código, además del control de los datos recibidos por el microcontrolador, se encuentran las entradas y salidas expuestas en el panel frontal, las cuales también hay que programar.

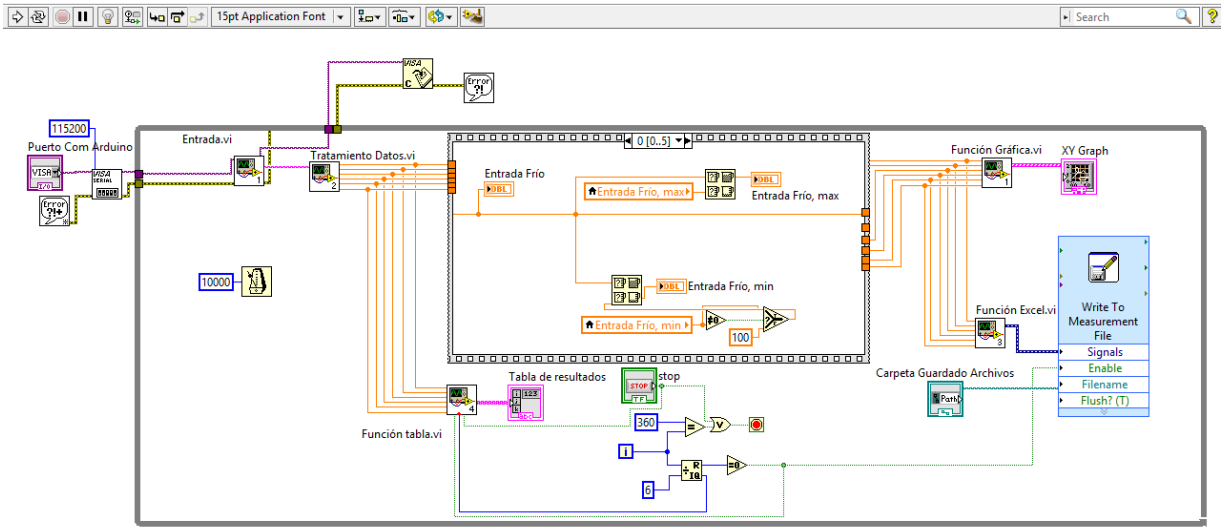

Figura 4.2 - Ejemplo de diagrama de bloques

- **Paletas:**

Dentro de las paletas tenemos las herramientas necesarias para crear y modificar todo nuestro programa.

Por un lado, paletas de controles y de edición para el panel de control, y por otro lado paleta de funciones para el diagrama de bloques.

- La paleta de funciones: Contiene las funciones para la creación del diagrama de bloques, tanto entradas, como operaciones matemáticas, comunicaciones, estructuración de datos, funciones específicas de librerías concretas, etc.

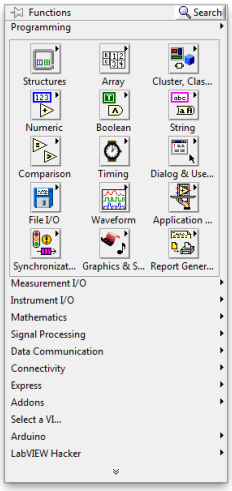

Figura 4.3 - Paleta de funciones

- Paleta de controles: Nos permite incluir todo tipo de controles y estructuras necesarios para la creación de nuestro panel frontal.

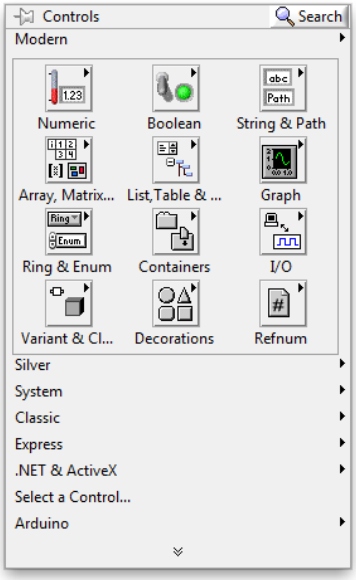

Figura 4.4 - Paleta de controles

- Paleta de herramientas: Es la paleta que contiene las propiedades de edición del panel de control.

Figura 4.5 - Paleta de herramientas

4.4. Instalación de LabVIEW y drivers NI-VISA

El funcionamiento correcto del software de LabVIEW con Arduino requiere de la instalación del software base y de unos drivers específicos.

Puesto que la comunicación se realizará mediante USB, habrá que instalar los drivers que gestionan el protocolo de comunicación USB para LV, en este caso NI-VISA. Tras la instalación de estos drivers, se podrá comunicar Arduino con LabVIEW mediante USB exitosamente.

VISA es un API de alto nivel utilizado para comunicarse con buses de instrumentación. Es independiente de la plataforma, del bus y del entorno. En otras palabras, la misma API se utiliza sin importar si un programa está creado para comunicarse con un dispositivo USB con LabVIEW en una máquina que ejecuta Windows 2000, o con un dispositivo GPIB con C en una máquina que ejecuta Mac OS X. USB es un bus de comunicación basado en mensajes. Esto significa que una PC y un dispositivo USB se comunican enviando comandos y datos a través del bus en forma de texto o datos binarios. Cada dispositivo USB tiene su propio conjunto de comandos. Se pueden utilizar funciones de Lectura y Escritura NI-VISA para enviar estos comandos a un instrumento y leer la respuesta del mismo.

Los pasos a seguir en la instalación son básicos, por tanto:

1. Instalamos LabVIEW 2015. El programa se puede obtener desde el repositorio de la UPCT, pues se cuenta con licencias para uso en prácticas y para alumnos.

2. Descargamos e instalamos los drivers NI-VISA desde la página web de National Instrument.

4.5. Creación de SubVIs

Los SubVIs son sub-bloques de programación que podemos crear para ordenar el código y hacerlo más cómodo y entendible. Podemos empaquetar parte de nuestro código y convertirlo en un bloque más del diagrama de bloques, quedando visible una única caja con las entradas y salidas del código inicial englobado.

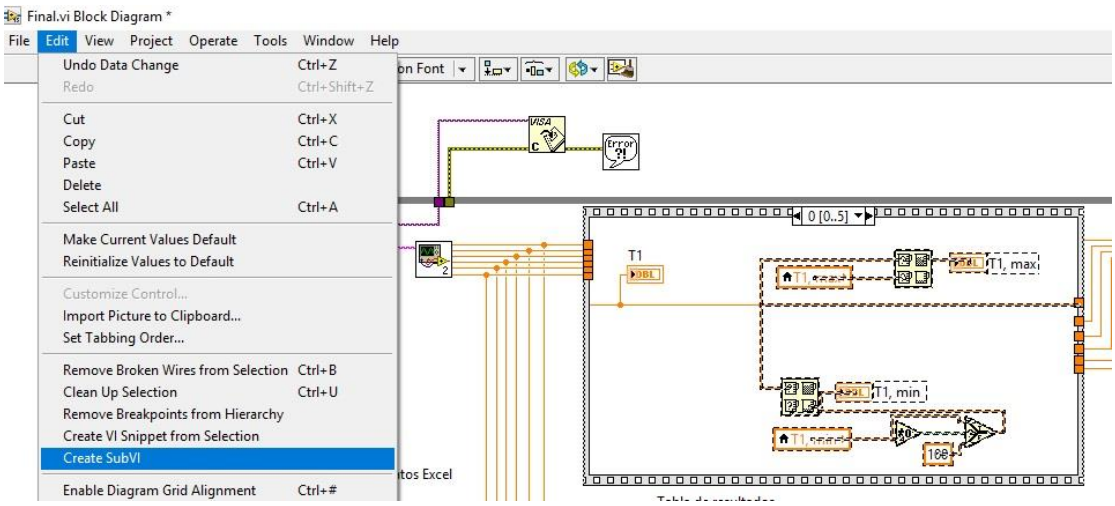

Figura 4.6 - Creación de SubVIs

Esta herramienta se usará más adelante para estructurar nuestro código.

4.6. Panel Frontal de LabVIEW en nuestro proyecto.

La interfaz que está compuesta por los siguientes elementos:

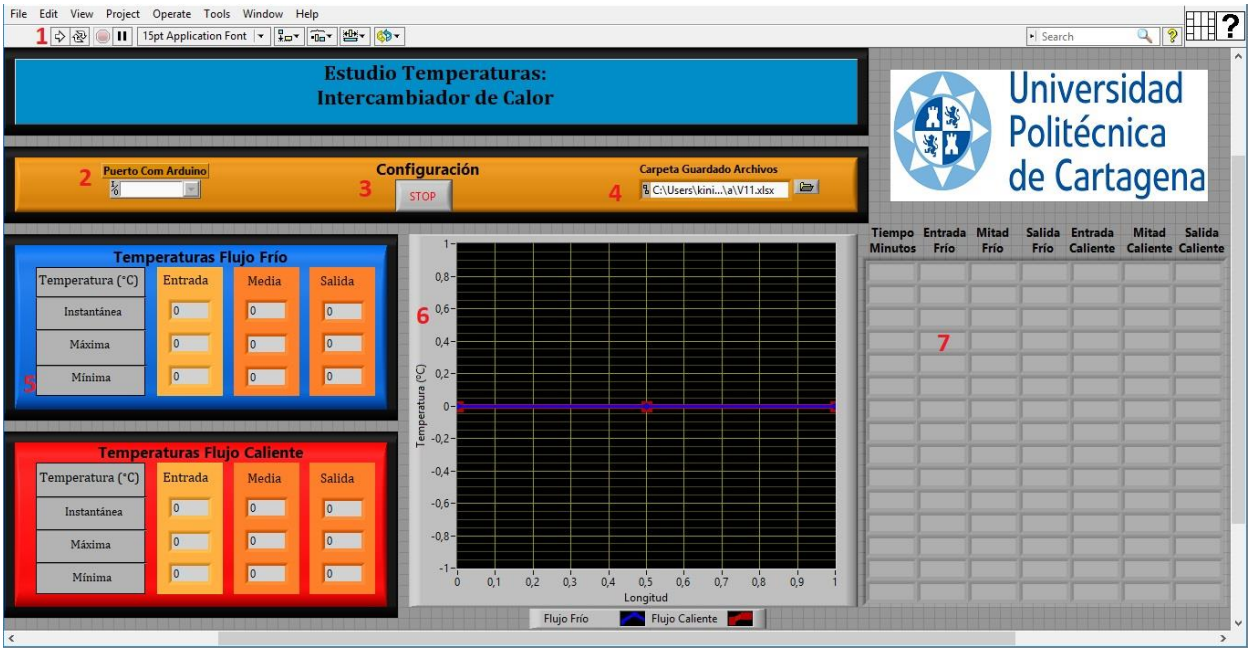

Figura 4.7 - Descripción de nuestra interfaz

1. **Run:** Arranca el programa a partir del botón del propio software de Labview.

2. **Puerto Arduino:** Pestaña para seleccionar el puerto COM al que se ha conectado Arduino. Al desplegar nos saldrán los puertos COM conectados en ese momento. Si se tienen varias posibilidades será necesario conocer cuál pertenece a nuestro proyecto dentro del administrador de dispositivos.

3. **Stop:** Botón que detiene la ejecución del programa y lo reinicia.

4. **Carpeta de guardado de datos:** En esta pestaña se indica donde se almacena y con qué nombre el archivo Excel que contendrá las medidas tomadas. Clicando en el icono de la carpeta nos permite seleccionar la carpeta y el nombre que guste, guardándose automáticamente en formato .xlsx de Excel.

5. **Tabla en tiempo real:** Tablas de temperaturas de ambos flujos, tanto flujo frío como flujo caliente, en el que se muestran los valores de las temperaturas en la entrada, a la mitad del intercambiador y a la salida cada instante de recepción de medidas, programados cada diez segundos. Además, se muestran los valores máximos y mínimos dentro de esa serie de monitorización para ambos flujos y posiciones.

6. **Gráfica:** Graficación en tiempo real del perfil de temperaturas del intercambiador de calor. Es una gráfica muy útil y visual en el campo de los intercambiadores. En el eje de ordenadas se muestran las temperaturas en

grados Celsius y en el eje de abscisas, la longitud del tubo de forma relativa, mostrando el cero como la entrada y el uno como salida de los tubos del intercambiador.

7. **Tabla de datos de salida:** Se irán mostrando las medidas que van siendo almacenadas para esa sesión en nuestro archivo de Excel. Estas medidas son almacenadas cada minuto, y se muestran dentro de la sesión.

Capítulo 5 Diagrama de Bloques en LabVIEW de nuestro proyecto.

El diagrama de bloques mostramos nuestro código propiamente dicho que hace funcionar nuestra interfaz monitorizando y almacenando los datos medidos. Este apartado se tratará como un capítulo aparte del anterior debido a su complejidad.

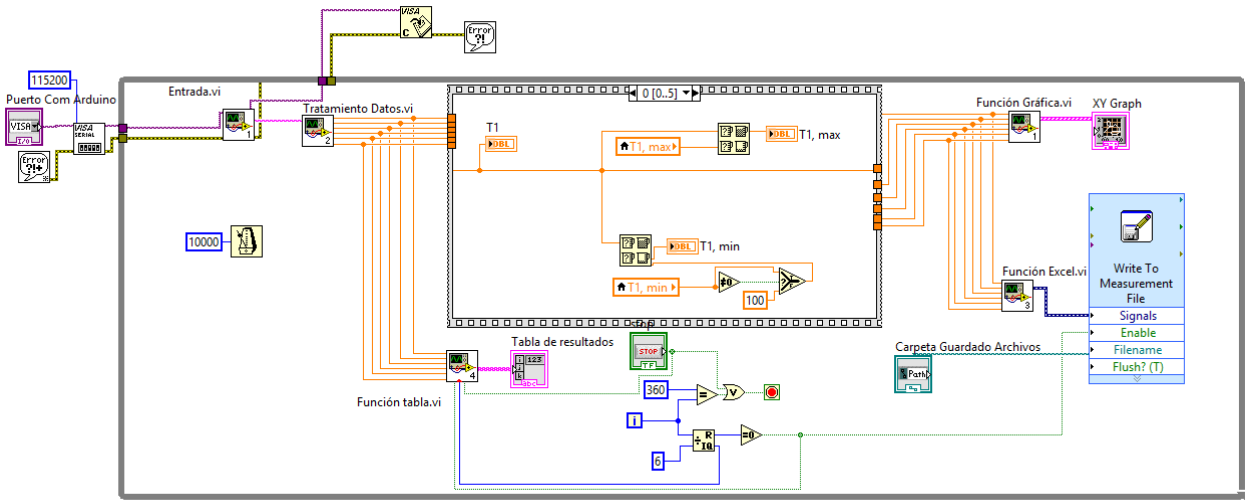

Figura 5.1 - Diagrama de bloques de nuestro proyecto

Este es código principal de nuestro programa, pero como se observa en la figura, los bloques .vi son SubVIs creados para ordenar y minimizar el código general. Cada SubVI contiene parte del código, y están agrupados en formaciones funcionales. Éste código se irá analizando paso a paso.

5.1.Inicio y finalización de la comunicación.

La comunicación con Arduino se producirá a través de la conexión USB, a la cual Arduino enviará las lecturas de temperatura y Labview trabajará con ellas. Por ello se utilizará el recurso VISA que contiene bloques de programación específicos para realizar esta comunicación.

Para inicializar la comunicación utilizaremos el siguiente conjunto:

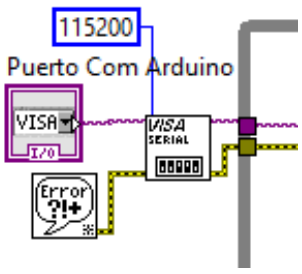

Figura 5.2 - Inicio de comunicación con el puerto serie de Arduino

Este elemento inicializa el puerto serial especificado en la entrada con las configuraciones específicas que nos interesen. En caso de no añadirle ninguna, las configuraciones por defecto aparecen entre paréntesis. Fundamentalmente habrá que añadir un selector del recurso del puerto de entrada, para seleccionar el puerto COM, que nos aparece en el panel frontal, y del recurso de salida.

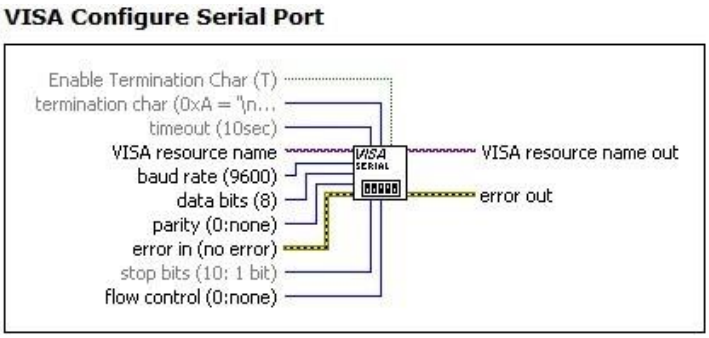

Figura 5.3 - Configuración de VISA Serial

Nuestro VISA Serial se ha personalizado con la salida del recurso de selección del puerto de entrada, y con el baud rate, que se ha puesto constante a 115200, puesto que tiene que trabajar al mismo baudaje que el código programado en Arduino, y se le ha añadido el mensaje general de error, que nos advierte si hay algún problema, con el error específico.

Para cerrar la comunicación con Arduino se utiliza el elemento contrario.

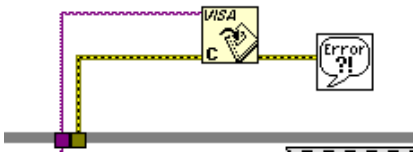

Figura 5.4 - Cierre de comunicación con el puerto serie de Arduino

5.2.Ciclo While Loop

Es el equivalente al bucle While empleado en los lenguajes convencionales de programación. Sirve para hacer que una secuencia de instrucciones se repita una cantidad de veces, siempre y cuando una afirmación sea verdadera. En Labview se ejecutarán las funciones que se encuentren dentro del cuadro de ciclo. Al seleccionar la estructura While Loop y situarla en el diagrama de bloques, un botón de stop aparece en el mismo. La condición que se le puede asignar para que se repita el ciclo puede ser: Stop si es verdad o Continúa si es verdad.

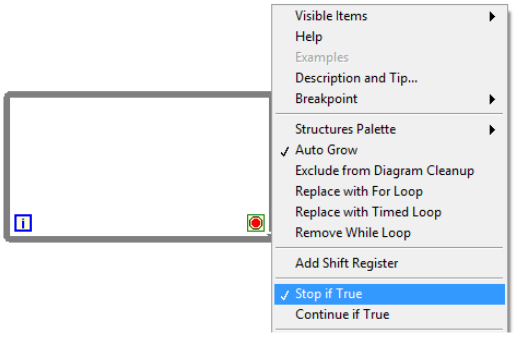

Figura 5.5 - Ciclo While Loop

En nuestro caso, se usará la opción Stop If True, y que parará siempre que se oprima el botón de Stop de la interfaz, o en caso de acabar una sesión completa de medidas fijada en una hora. Esto se ha programado de la siguiente manera:

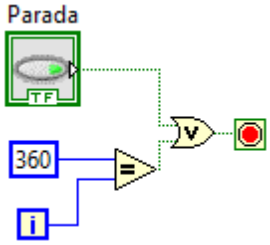

Figura 5.6 - Métodos de parada del programa

Este conjunto nos indica las dos formas de parada, por un lado, pulsando el botón STOP, y, por otro lado, mediante el operador OR, se detendrá cuando el número de ciclos sea igual a la constante 360, esto es, como el ciclo de medidas se repite cada 10 segundos, al cabo de una hora habrá iterado 360 veces, dando por válida la igualdad y parando el proceso.

Dentro del bloque While Loop, se encuentra todo nuestro código del programa, y la contante de repetición del bucle, controlado mediante:

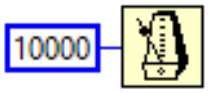

Figura 5.7 - Función periodo de repetición del While Loop

5.3. Recepción y lectura de los datos.

Tras la apertura de la comunicación, por medio del bloque VISA Serial, y ya dentro del ciclo While Loop, comenzamos con el código de recepción y lectura de los datos enviados por parte de Arduino.

Esta parte del código se ha agrupado dentro en un SubVi, que ha sido nombrado como Entrada.VI:

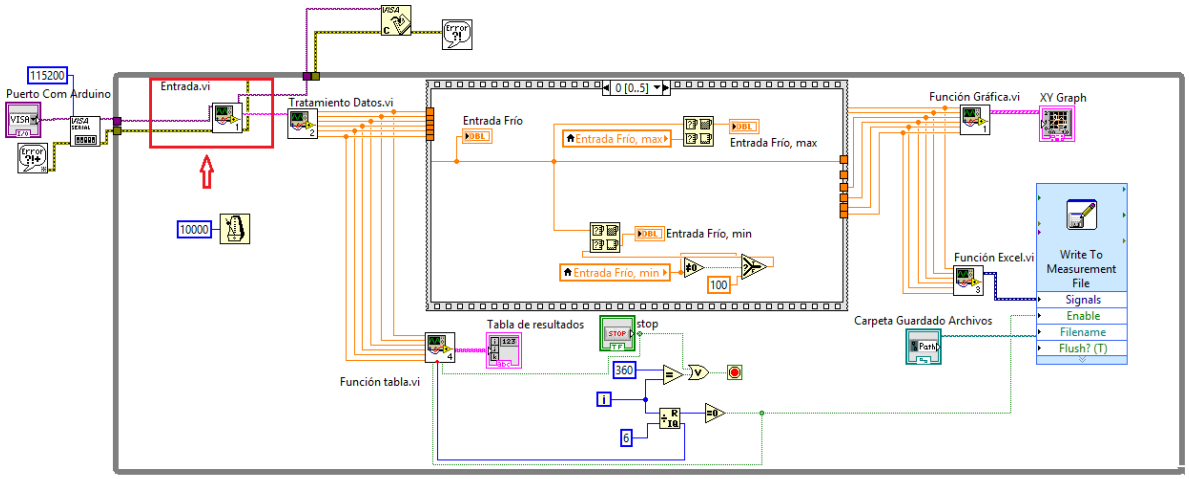

Figura 5.8 - SubVI Entrada.vi

Este SubVI está compuesto en su interior por el conjunto:

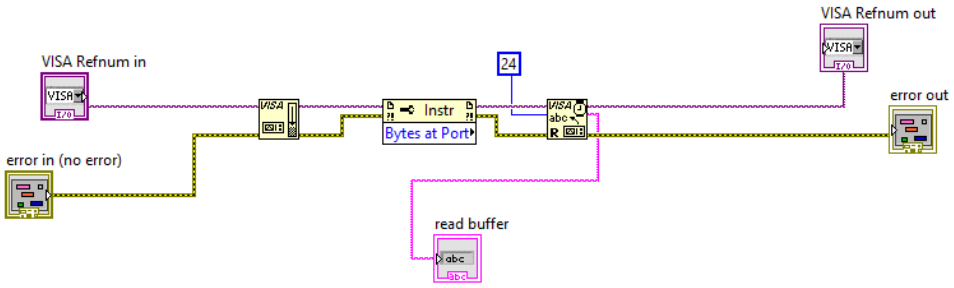

Figura 5.9 - Código interno del SubVI Entrada.vi

Las entradas y salidas de este conjunto Entrada.vi son:

- VISA Refnum in: Recepción de entrada de los datos al SubVI Entrada.vi desde el VISA Serial.
- VISA Refnum out: Salida de la conexión VISA del SubVI Entrada.vi y que se conectará al VISA Close que cierra la conexión con Arduino fuera del While Loop.
- Read buffer: Salida del cable que contiene los datos de las medidas en formato “string” o cadena de caracteres.
-

El SubVi contiene los siguientes bloques de programación:

- VISA Flush: Bloque encargado de vaciar la memoria intermedia, con los datos entrantes, tras su paso por el bloque, para la siguiente iteración.

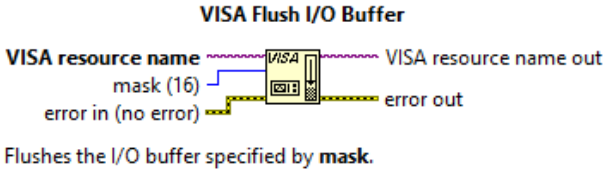

Figura 5.10 - Función VISA Flush

- Property Node: Realiza el conteo de los bytes que son enviados por el puerto serie, no es necesario si especificamos en la entrada del VISA Read cuantos bytes debe leer.

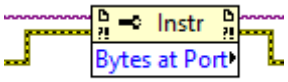

Figura 5.11 - Función Property Node

- VISA Read: Bloque encargado de leer el número de bytes que se le ha fijado, en nuestro caso 24 bytes, 4 bytes por cada uno de los 6 sensores de temperatura. La transformación que se le realizó a los datos en el código de Arduino era para asegurar exactamente que entren estos 24 bytes indicados y que no deje al bloque esperando bytes que no llegarán, colgando el programa. Este bloque devolverá como salida la cadena de caracteres “string” que contiene la información.

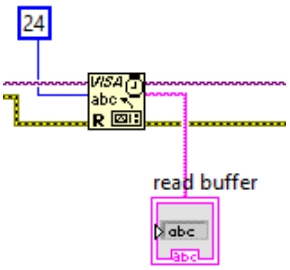

Figura 5.12 - Función VISA Read

Ya disponemos de los datos que contienen las mediciones, en formato “string”, saliendo de Entrada.vi.

5.4.Tratamiento y acondicionamiento de los datos.

Una vez disponemos de la cadena de caracteres que contiene la información, debemos realizar un proceso de transformación de esa cadena a los datos numéricos de las mediciones. Este tratamiento se encuentra compactado en el SubVI Tratamiento Datos.vi, que recibe los datos en formato "string" salientes de Entrada.vi y devuelve el valor numérico de las mediciones de los seis sensores de temperatura.

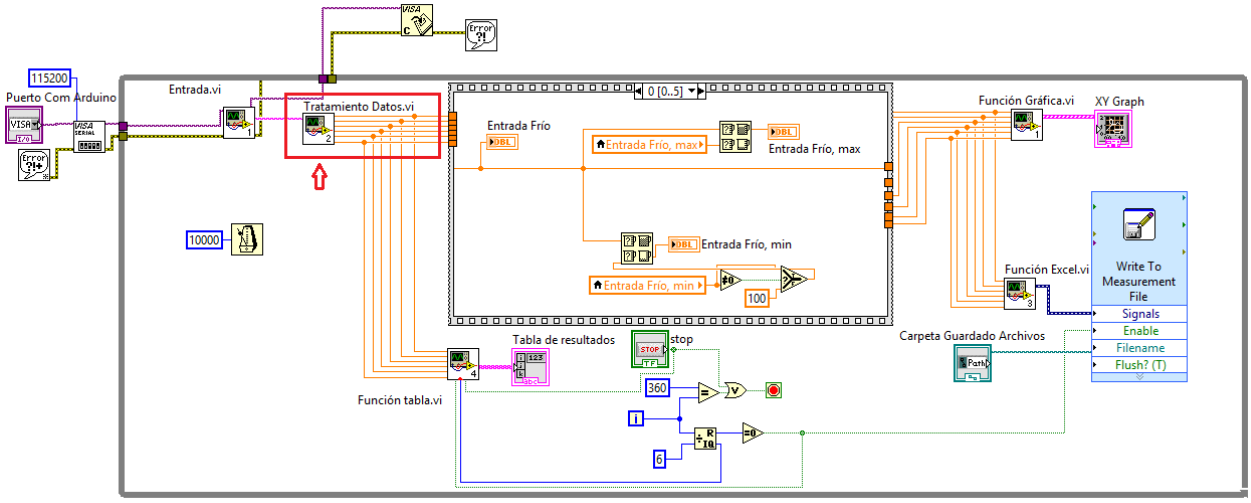

Figura 5.13 - SubVI Tratamiento Datos.vi

Este SubVI contiene en su interior el siguiente código:

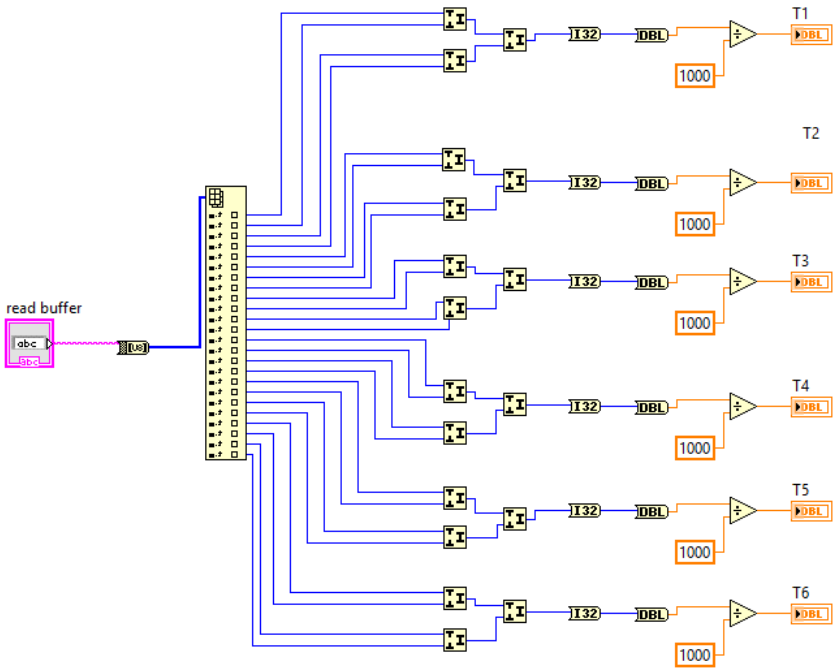

Figura 5.14 - Código interno SubVi Tratamiento Datos.vi

Dentro de este SubVI Tratamiento Datos.vi encontramos los siguientes elementos:

- Read buffer: Es el cable que contiene la cadena de datos "string" proveniente de Entrada.vi.

- String To Byte Array Function: Como su nombre indica, convierte la cadena string de caracteres en un vector formado por los 24 bytes leídos, del tipo [1byte, 2 byte, …,24 byte].

Figura 5.15 - Función String To Byte Array

- Index Array Function: Esta función nos permite clasificar o enlazar cada elemento del array anterior individualmente y trabajar con cada elemento del array.

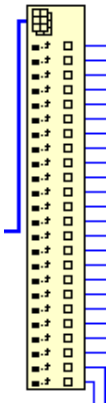

Figura 5.16 - Función Index Array

Como se observa, este bloque nos proporciona cada byte del vector, ordenadamente, como salidas.

- Join Numbers: Con esta función unimos los bytes dos a dos, hasta unificar los 4 bytes recibidos de cada sensor de temperatura.

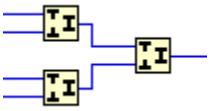

Figura 5.17 - Conjunto Unión de Bytes

- Tras la unión de los bytes, el siguiente paso será indicarle al programa que ese número unificado es un número del tipo "long" de 32 bits y posteriormente lo transformamos en un número float" o coma flotante. Estamos deshaciendo las transformaciones que se le realizan a la medida en el código de Arduino.

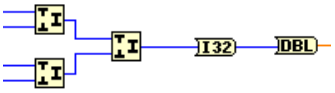

Figura 5.18 - Conversión de Bytes a Float

- Por último, mediante el operador matemático de división, deshacemos la última modificación que se le hizo a la medida entrante.

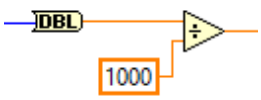

Figura 5.19 - Función de división

Con este último paso, ya tenemos las 6 medidas de temperatura de los sensores, que serán las salidas del SubVI de este apartado.

Figura 5.20 - Salida de temperatura instantánea final

A partir de Index Array, los pasos de unificación y cambio en los bytes agrupados es igual para los 6 sensores.

Temperatura instantánea, máxima y mínima.

Tras la transformación de los datos, y obtener los valores numéricos de las temperaturas, como primer paso, vamos a realizar el código para mostrar los valores instantáneos, y los valores máximos y mínimos. Para ello, y como el código es repetido para los seis sensores, se ha optado por utilizar una estructura Stacked Sequence, para ordenar el código y disminuir el espacio del mismo visualmente.

La estructura Stacked Sequence, que permite ejecutar varios subdiagramas denominados “frames” en un estricto orden, y, donde sólo es visible uno a la vez; permitiendo así la lectura primero de un sensor y después de otro. En los lenguajes de programación convencionales basados en código de líneas no se requiere, y por lo tanto no existe una estructura análoga.

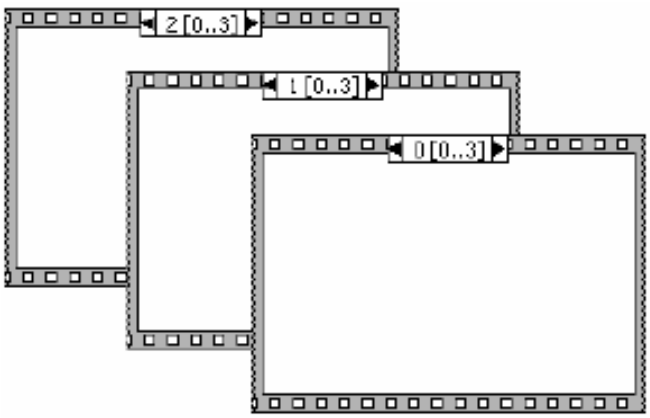

Figura 5.21 - Función Stacked Sequence

En la parte superior del marco de cada estructura se encuentra el identificador de diagrama, que es utilizado para navegar entre frames.

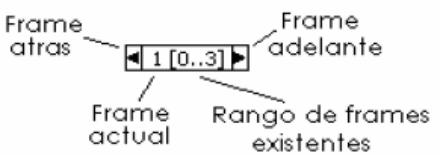

Figura 5.22 - Ventanas de Stacked Sequence

Por defecto la estructura Sequence posee un solo frame y no tiene identificador de diagrama. Para adicionar un frame después del actual debemos de seleccionar la opción Add Frame After

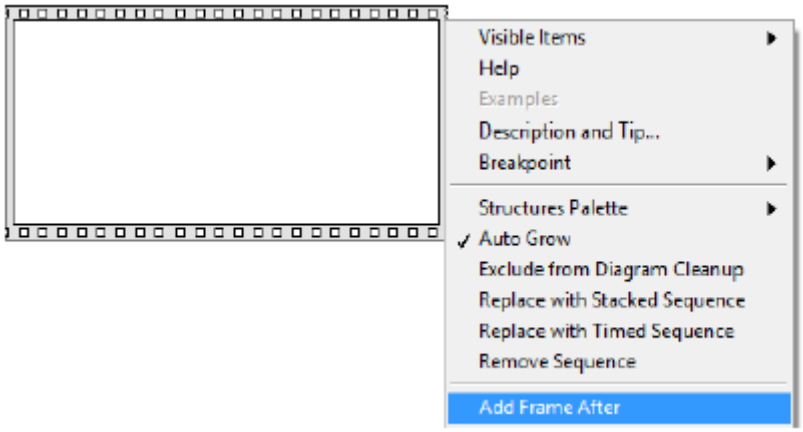

Figura 5.23 - Propiedad de incremento de frames

En nuestro proyecto, como se van a utilizar 6 sensores de temperatura, necesitamos emplear 6 *frames*, es decir, el rango de frames existentes será [0..5]. A continuación, se detalla el VI correspondiente al primer sensor, T1, que corresponde al sensor localizado en la entrada del flujo frío de la instalación.

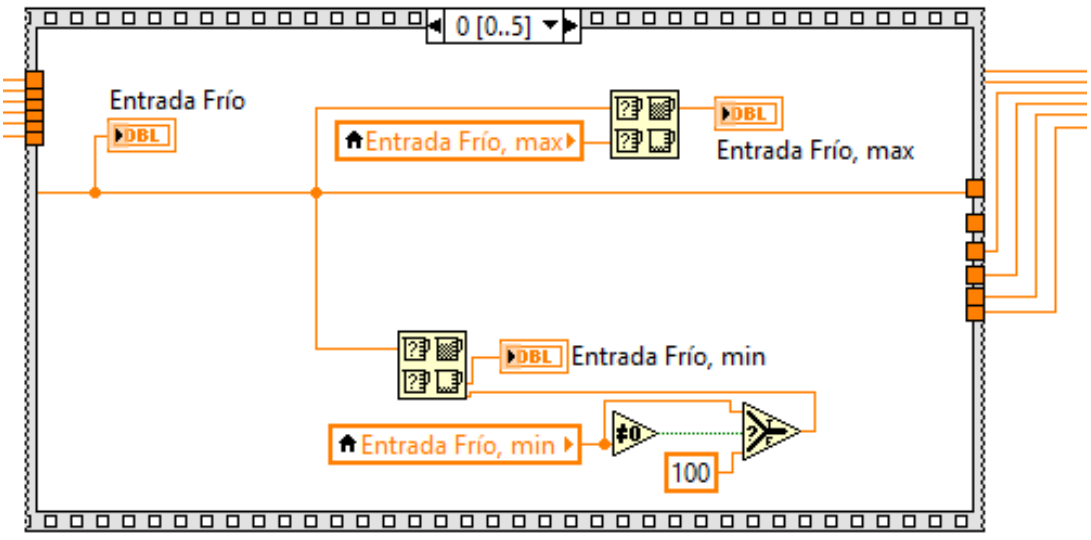

Figura 5.24 - Código de máximos y mínimos

Este bloque tiene una estructura basada en el cable central que porta el valor de la temperatura del sensor, del cual se mostrará el valor a través de la interfaz y se comparará para saber si es un máximo o un mínimo en comparación con los valores ya medidos anteriormente para ese sensor.

- Visualización en la interfaz del valor numérico de la medida de temperatura:

Figura 5.25 - Salida por interfaz de la temperatura instantánea

- Máximo: Se utiliza el bloque de función "Max & Min" para comparar los valores de x e y que entran al bloque, y devuelve el valor máximo o mínimo en diferentes salidas del bloque funcional.

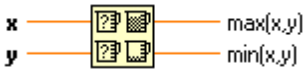

Figura 5.26 - Función Máximos y Mínimos

Para nuestro código, comparará la medida de temperatura de ese ciclo concreto, que será nuestra temperatura instantánea, entrando en 'x', con el valor que se encuentre en la interfaz, almacenado como máximo, entrando en 'y'. El máximo lo seleccionaremos con la salida máx. (x, y) del bloque.

Figura 5.27 - Comparación temperatura máxima

- Mínimo: Por el mismo método realizamos la comparación para obtener el valor mínimo de temperatura medido, pero debido a un problema que nos surge debemos hacer una variación. Al inicializar el programa todos los valores de la interfaz son nulos, por lo que, si se trabaja con temperaturas superiores a cero grados centígrados, al realizar la comparación con el valor almacenado en la interfaz, siempre quedaría el cero como valor mínimo del ensayo, llevando a error.

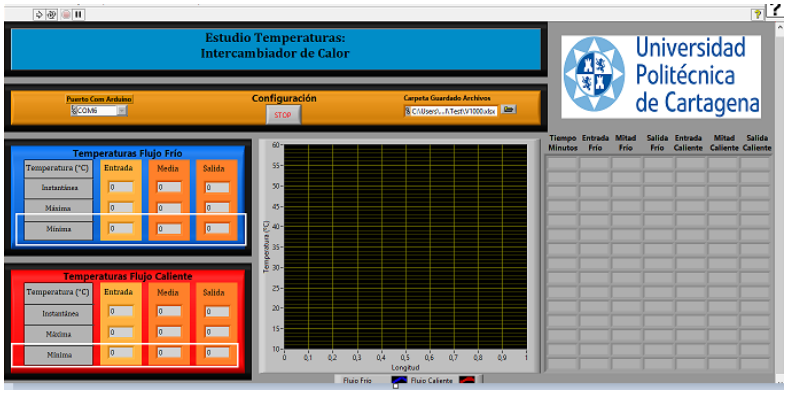

Figura 5.28 - Mínimos inicializados a cero

Para corregir este problema, realizamos el siguiente conjunto de programación:

1. Not Equal to 0?: Devuelve un TRUE si x no es igual a 0, o de otro modo, si x=0 devuelve un FALSE.

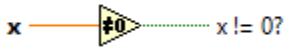

Figura 5.29 - Función Not Equal to 0?

2. Select: Si s es TRUE devuelve a la salida el valor numérico de la entrada t, y en el caso de que s sea FALSE devuelve a la salida el valor contenido en la entrada f.

Figura 5.30 - Función Select

3. El código para el cálculo del valor mínimo queda de la siguiente manera:

Figura 5.31 - Comparación de temperatura mínima

De este modo, existirán dos comportamientos en función de si el valor almacenado en la interfaz es o no, nulo.

- Valor nulo (caso de inicialización): la función Not Equal To 0? devolverá un FALSE lo que conlleva que la función Select muestre a la salida el valor 100. La comparación de dicho valor con la temperatura instantánea (siempre inferior a 100°C) se registrará última como valor mínimo.

- Valor no nulo: la función Not Equal To 0? devolverá un TRUE lo que conlleva que la función Select muestre a la salida el valor almacenado en la interfaz como mínimo. Al comparar dicho valor con la temperatura instantánea se registrará la temperatura más pequeña.

5.5.Graficación en tiempo real del perfil de temperaturas.

Otro de los objetivos del software es mostrar en pantalla la gráfica del perfil de temperaturas del intercambiador de calor. El código para realizar este paso se ha compactado en el SubVI Función Gráfica.vi, y posteriormente la salida de este bloque se comunica con el bloque de la gráfica en sí misma. Por tanto, este SubVI realiza una adecuación de los datos para poder graficarlos.

Una gráfica de perfil de temperaturas relaciona las temperaturas de los fluidos con la longitud del tubo. Esta longitud se suele indicar de forma relativa de 0 a 1, contando el 0,5 como la mitad. Por tanto, en esta gráfica se mostrarán los perfiles de ambos flujos simultáneamente, y

por ello, debemos agrupar la medida del sensor con el punto en el que se localiza el sensor, dependiendo de la entrada, mitad o salida del tubo del intercambiador, formando así los puntos (Tº, Longitud) a representar.

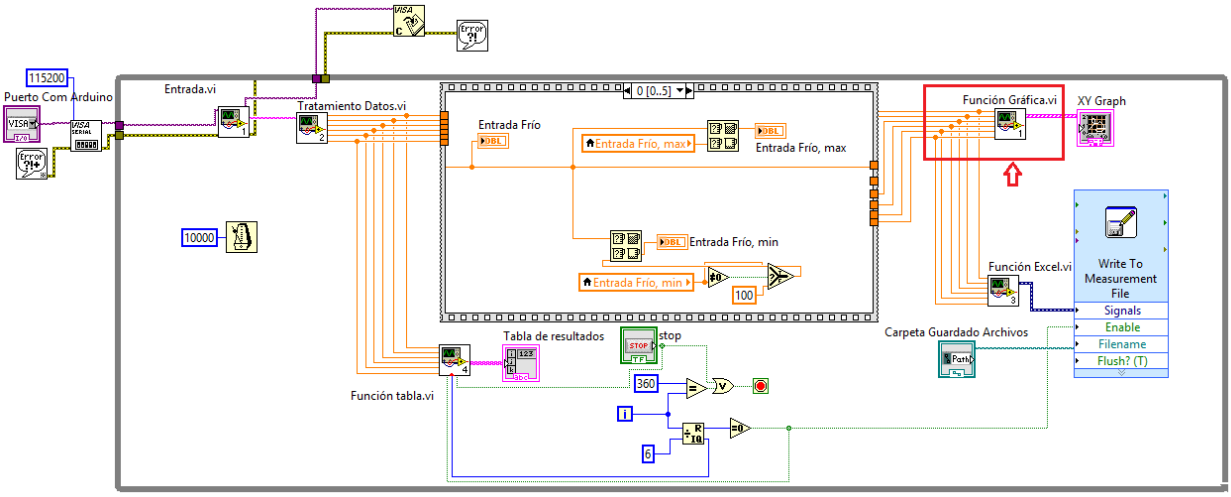

Figura 5.32 - SubVI Función Gráfica.vi

Este SubVI contiene el siguiente código en su interior:

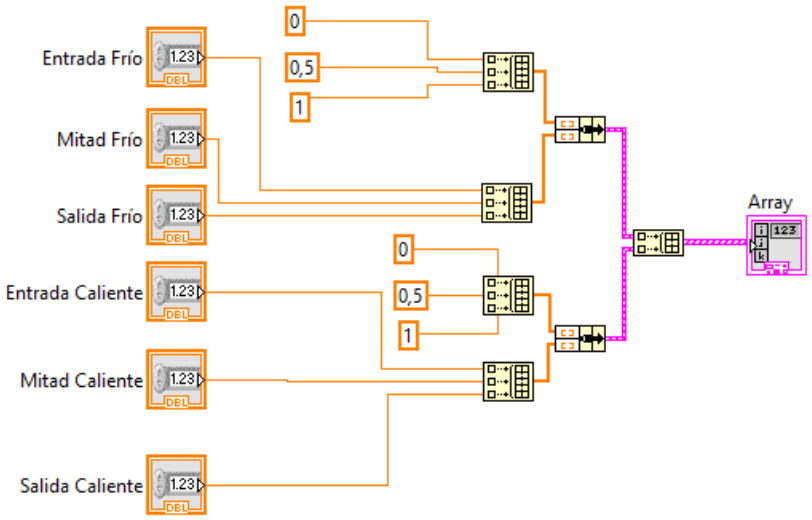

Figura 5.33 - Código interno del SubVI Función Gráfica.vi

- Build Array: Este bloque compacta los datos que entran en él y devuelve un vector formado por ellos en el orden de entrada.

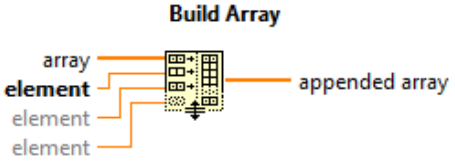

Figura 5.34 - Función Build Array

En nuestro caso, creamos dos vectores de posición [0, 0.5, 1], y con las temperaturas se construyen otros dos arrays, uno para las temperaturas del flujo frío y otro para las temperaturas del flujo caliente.

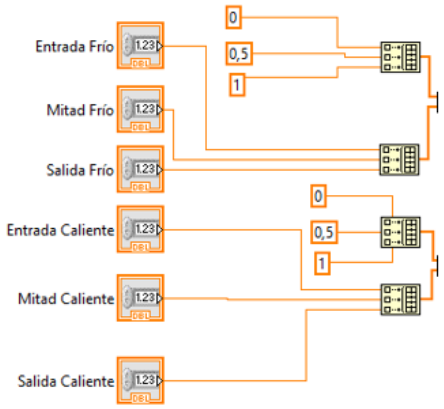

Figura 5.35 - Agrupación para crear las dos gráficas

- Bundle: Este elemento nos permitirá unificar los array de longitud con los array de los flujos de temperatura, enlazando cada elemento del vector longitud con su análogo del vector flujo, formando así una cadena de tipo "string", de tres puntos formados cada uno con sus respectivos valores de longitud y temperatura. Se puede observar que el valor de temperatura del flujo de entrada caliente se enlaza con el valor de longitud cero, esto es debido a que el intercambiador de calor es equicorriente, en caso de un intercambiador de calor contracorriente, se tienen que intercambiar los valores de entrada y salida calientes para que coincidan con sus posiciones reales.

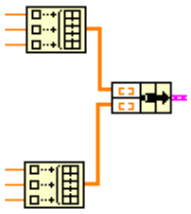

Figura 5.36 - Función Bundle

- Por último, unimos las dos cadenas de puntos en un último array para poder representar ambas gráficas simultáneamente.

5.6.Tabla de datos en tiempo real.

La interfaz va a mostrar las mediciones que se van almacenando en el archivo Excel. Estos valores se almacenan cada minuto, aunque el ciclo While Loop, y por tanto las temperaturas instantáneas se obtengan cada 10 segundos, así se almacenarán las medidas cada 6 muestras.

Los ensayos se realizarán a diferentes caudales de flujo, y otros a temperaturas de entrada fijadas de antemano. Por esto, y aunque el ciclo While Loop esté programado para que duré una hora para un ensayo más prolongado, se podrá ir parando el programa mediante el botón STOP presente en la interfaz, para realizar los ensayos comentados. Necesitaremos, por tanto, que la tabla de datos de la interfaz se vacíe y se resetee en cada paro del programa.

El código para la realización de esta tabla también se encuentra agrupado en un SubVI llamado Tabla de Resultados.vi.

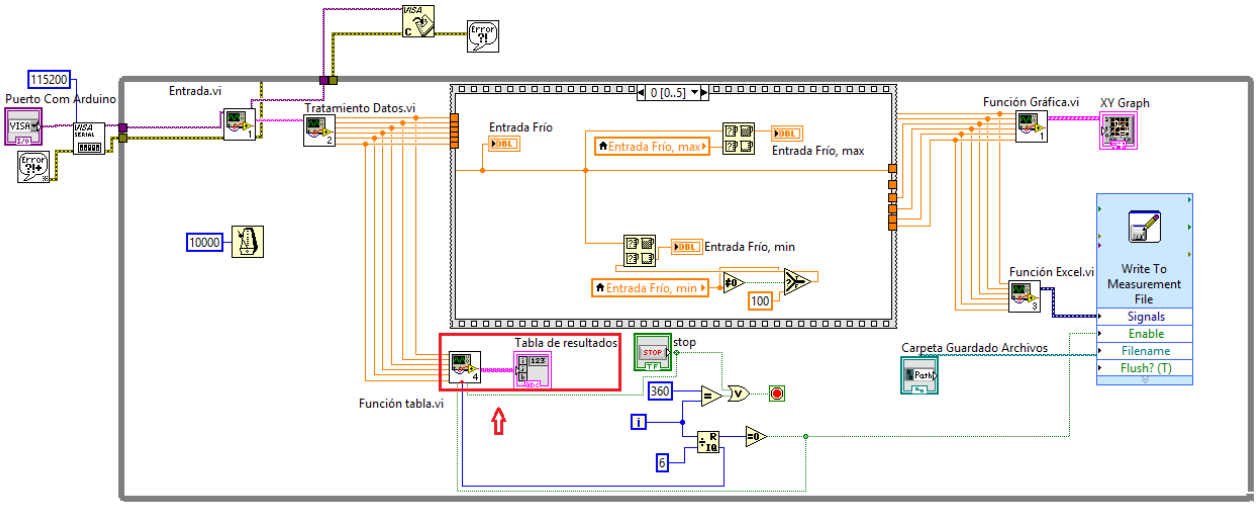

Figura 5.37 - SubVI Función Tabla.vi

Este SubVI tiene las entradas de las 6 temperaturas, de un conjunto de operaciones para almacenar los datos cada minuto, y del botón de STOP para resetear la tabla al accionarlo. También entrará al SubVI el resultado del operador que nos indicará el número de minutos del ensayo para mostrar la duración del ensayo en la tabla.

- Las operaciones que se realizan para indicar que se han de almacenar los datos pues ha pasado un minuto son las siguientes.

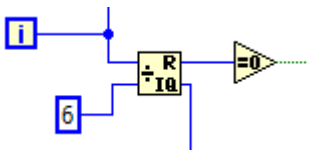

Figura 5.38 - Función Quotient & Remainder

- Quotient & Remainder: Este bloque recibe dos entradas, x e y, y realiza dos operaciones, la salida superior se ha realizado la operación x –y*x/y. La x representará la iteración del ciclo While Loop, y la y es una constante de valor 6. Si esa salida es igual a cero, esto le indicará al programa que la iteración es múltiplo de 6, y tendrá que almacenar las medidas, tanto en la tabla de datos de la interfaz como en el archivo de Excel.
  Como salida inferior nos devuelve el valor de la división, que, cada vez que vayamos a representar los datos en la tabla, ésta salida nos indicará el valor de los minutos durante el ensayo.

Dentro del SubVI encontramos el siguiente código.

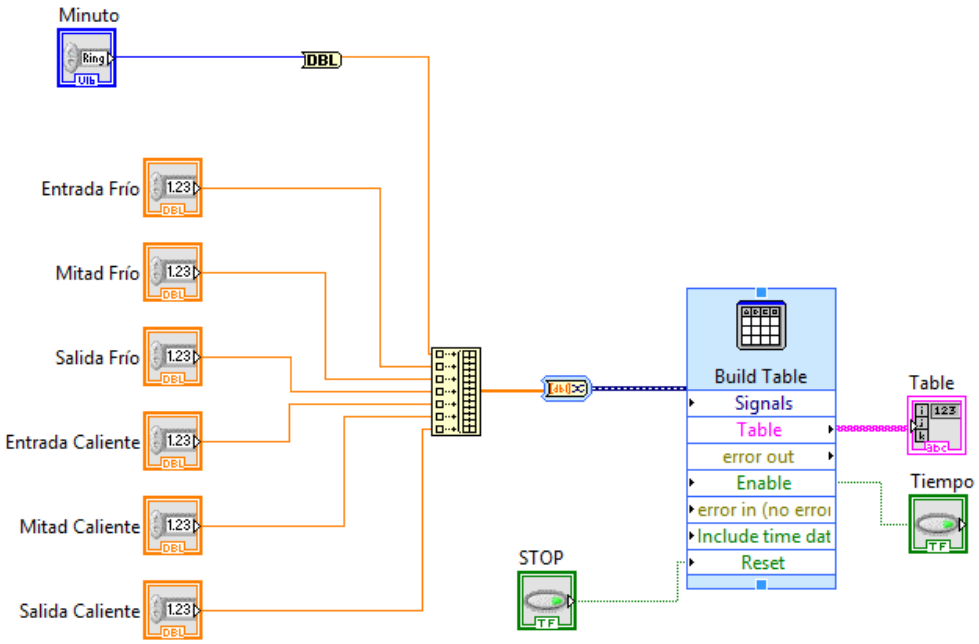

Figura 5.39 - Código interno del SubVI Función Tabla.vi

- El primer elemento es el bloque "Build Array" visto en varias ocasiones anteriormente, su función dentro de este SubVI es la de crear un vector con las 6 temperaturas instantáneas del ciclo. El primer elemento del vector es el valor numérico de la división entre el número de ciclos y la constante 6, operación que se encuentra en el instrumento virtual general. Este número, como se ha comentado, al habilitar la tabla de datos del panel de control cada 6 ciclos, se corresponderá con los minutos transcurridos en el ensayo para el ciclo de muestreo.

- Convert to Dynamic Data: La entrada de datos en los express.vi, que son los bloques de configuración de tablas y gráficas dinámicas, tienen la entrada de datos de tipo dinámico. Este vi convierte multitud de tipos de variables, arrays, booleanos, dobles, etc. En nuestro caso transforma nuestro array de temperaturas.

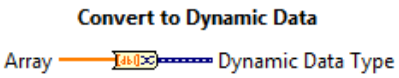

Figura 5.40 - Función Conver to Dynamic Data

- Build Table Express: Para mostrar datos en el panel de control se pueden crear dos tipos de tablas. Una tabla básica, con la entrada de datos en "string", y las tablas express.vi, que contiene el conjunto de Build Table Express y la conexión del bloque de la tabla final. Este Build Table nos permite configurar y controlar la tabla con varias entradas.
  En nuestro caso, tiene como entradas de configuración y control la propia señal de datos convertidos en el paso anterior, y las entradas al SubVI provenientes del botón

de STOP y del conjunto del While Loop que hemos preparado para almacenar las medidas cada minuto.

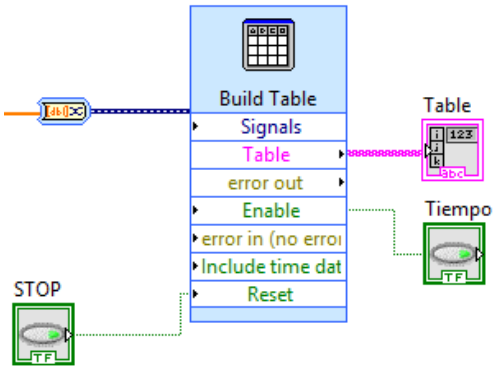

Figura 5.41 - Función Build Table

El botón de STOP se enlaza con la entrada de configuración de Reset. Esto reseteará la tabla a sus valores iniciales, que son vacíos por defecto, cada vez que se pare el ensayo. En el laboratorio se van a ejecutar ensayos a distintos caudales fijados y a distintas temperaturas de entrada de flujo, por lo que nos interesa que la tabla se resetee cada vez que se finaliza el ensayo, si no, la tabla continuaría para las siguientes tandas de pruebas, pues sólo se resetearía en caso de cerrar el programa, lo que sería una molestia. La entrada nombrada como Tiempo, se conecta a la entrada Enable, activando la tabla cada minuto.

Para ver con más claridad estas entradas, se muestra el código general que interviene en las entradas al SubVI, que se conectan a la Build Table.

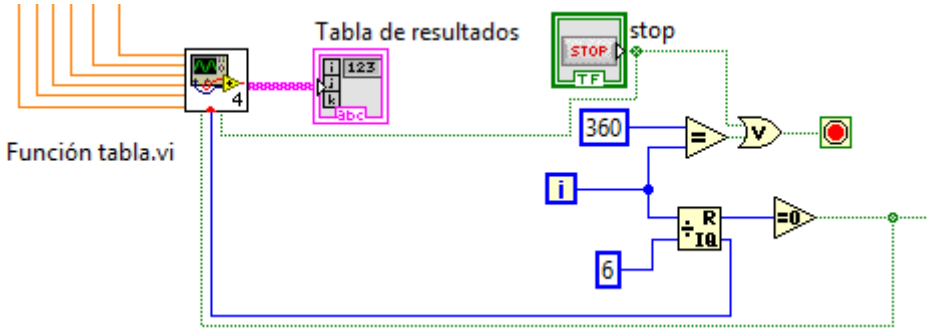

Figura 5.42 - Entradas al SubVI Función Tabla.vi

5.7. Exportación de datos a Excel.

Una vez realizada la monitorización y graficación en tiempo real de los ensayos, queda, como último paso, generar un archivo Excel donde se vayan almacenando las temperaturas instantáneas de los sensores. Las medidas serán almacenadas cada minuto. La tabla debe estar perfectamente ordenada, indicando con cabeceras, a que localización corresponde cada columna de temperaturas, además de incluir una primera columna con la fecha y hora realizada.

El bloque completo que realiza este almacenado de datos es el siguiente.

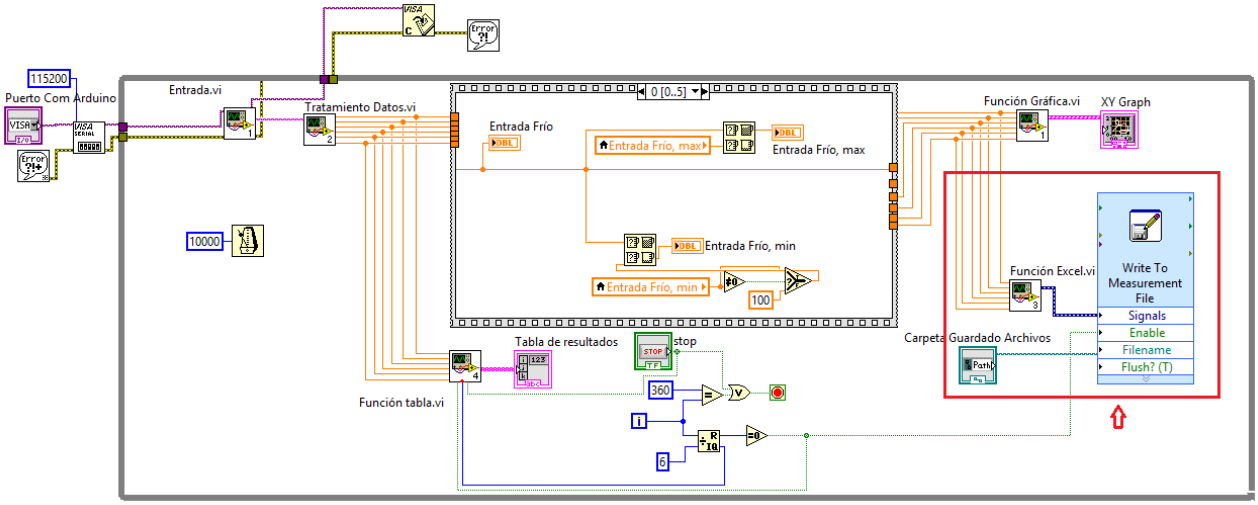

Figura 5.43 - SubVI Función Excel

Y está compuesto por dos partes diferenciadas. El SubVI Función Excel.vi que adecua los datos para su correcto envío para formar la tabla de Excel, junto con sus cabeceras, y la función "Write To Measurement File" para la exportación.

- Write To Measurement File: Esta función, localizada en el código general, realiza la función de almacenado de datos en archivos externos a LabVIEW, permitiéndonos modificar algunos parámetros.

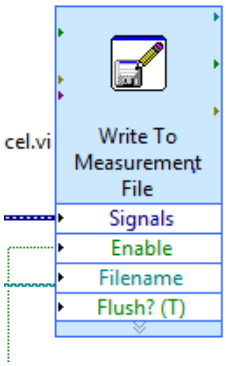

Figura 5.44 - Función Write to Measurement File

A está función se le conecta la señal con los datos acondicionados salientes del SubVI.

Para efectuar la toma de datos cada minuto de forma periódica, de forma análoga a como se realizó para la tabla de datos de la interfaz, tendrá lugar siempre que el número de iteraciones realizadas sea múltiplo de seis, es decir, cuando el resto resultante de dividir el número de iteraciones, i, entre 6, sea igual a cero. Este paso se resuelve conectando la salida del bloque "Quotient & Reminder visto, a la entrada Enable de la función.

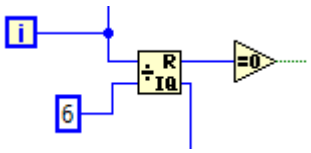

Figura 5.45 - Quotient & Remainder cuando son múltiplos

- Para una mayor comodidad, la interfaz del programa nos permite elegir la carpeta donde se almacena el Excel y ponerle nombre a este archivo del ensayo. Como se mostró en la descripción del panel frontal, la opción Carpeta Guardado de Archivos nos permite esta elección. Al clicar en el botón de exploración que posee se nos muestra la siguiente ventana del explorador de archivos, como se puede observar, se guarda el archivo en formato .xlsx de Excel.

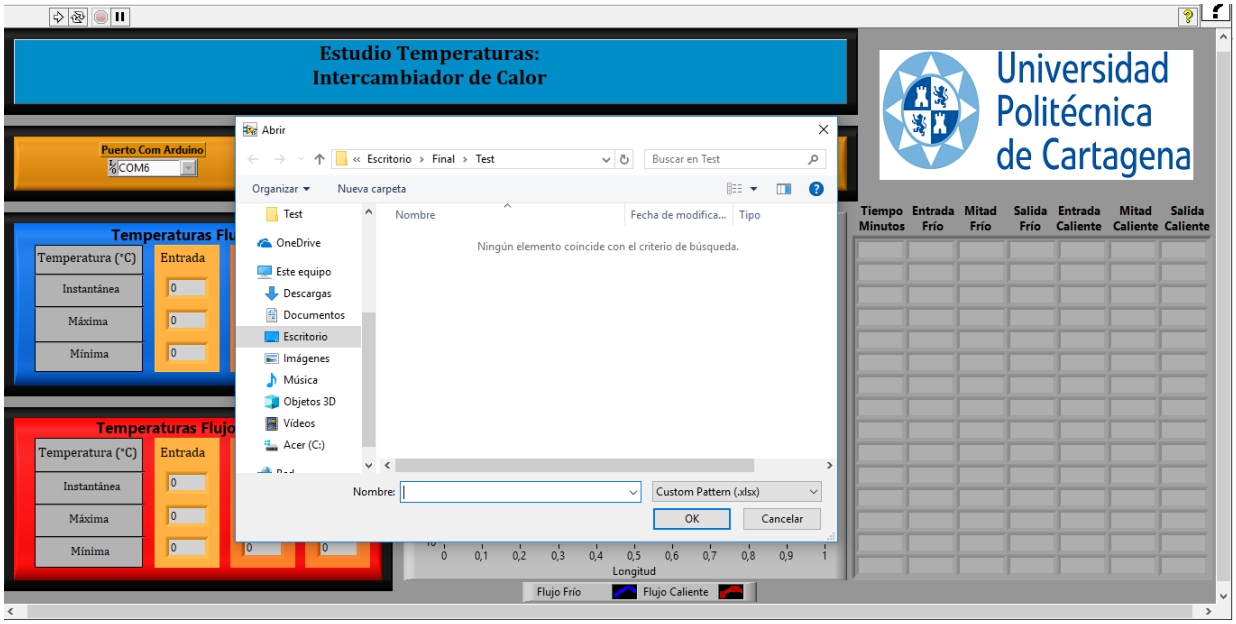

Figura 5.46 - Explorador de archivos para guardar el archivo Excel

Para crear este explorador de archivos, clicando con el botón derecho sobre la opción "Filename" y seleccionamos "create,Control".

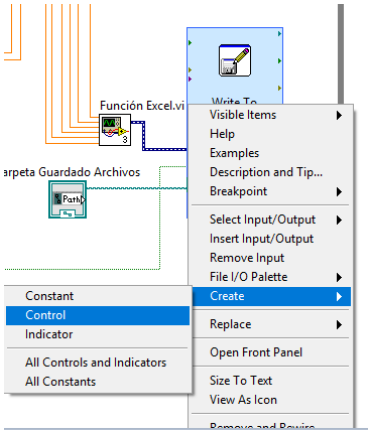

Figura 5.47 - Creación del explorador de archivos

Para que el programa guarde automáticamente los archivos en un formato de Excel, haciendo clic derecho en el elemento del panel frontal y clicando en propiedades, en la pestaña de "Browse Options" dentro de propiedades, en el apartado pattern escribimos ".xlsx" que será el formato del fichero a crear que queremos, y en "Selection Mode" marcamos las opciones "files" y "New or existing" como opciones de la carpeta a guardar.

Figura 5.48 - Configuración del explorador de archivos

Y, por último, clicando con el botón derecho sobre la función "Write To Measurement File", elegimos las opciones que nos interesan a la hora de exportar el archivo. Para nuestro trabajo se seleccionarán las siguientes opciones, que nos creará nuevos nombres continuados si el fichero ya existe con ese nombre, y creará una columna en el archivo exportado con la fecha y la hora.

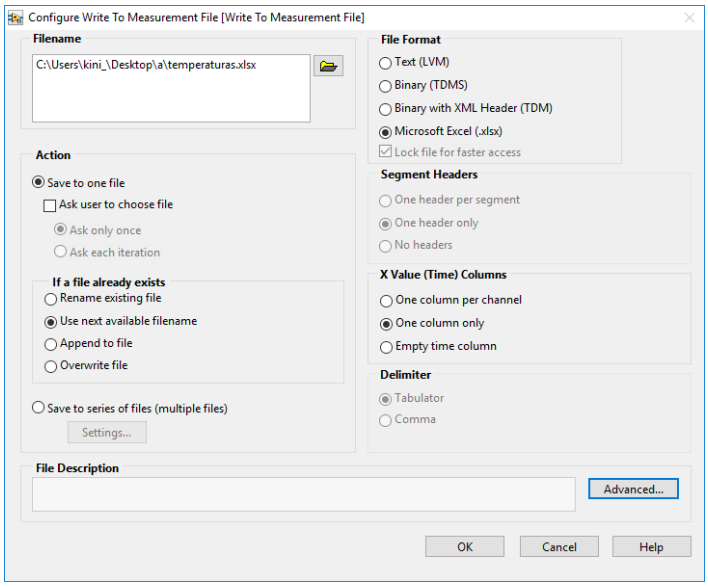

Figura 5.49 - Configuración de la función Write to Measurement File

El SubVi "Función Excel", contiene el código para la adecuación de los datos para ser exportados.

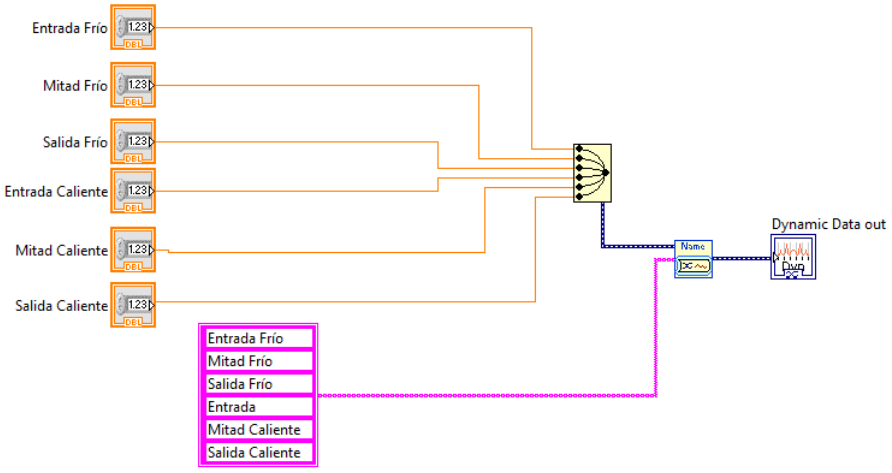

Figura 5.50 - Código interno del SubVI Función Excel.vi

Como se observa en la imagen superior, las 6 temperaturas instantáneas se agrupan con el elemento "Merge Signals", que reestructura todo tipo de variables en una única señal de salida de datos dinámicos, lista para conectarse en la entrada de señales de la función de exportación a Excel.

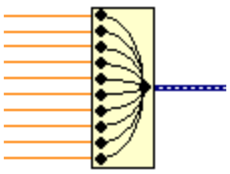

Figura 5.51 - Función Merge Signals

La función “Name” es un SubVI específico para exportación, que unifica la señal de datos con las cadenas de texto que compondrán las cabeceras del archivo.

Figura 5.52 - Función nombre de las señales

Capítulo 6 Realización física del proyecto.

6.1. Ejecución y resultados.

Debido a que el momento de la redacción de esta memoria aún no se encuentra instalado el sistema en su lugar de operación, el laboratorio de transmisión de calor en Física Aplicada, la entrada de lecturas por parte de los sensores de temperatura se realizará a temperatura ambiente como muestra del funcionamiento del presente proyecto. De este modo se podrá observar el resultado final del programa. El programa trabajará del mismo modo cuando el equipo se encuentre conectado al intercambiador, variando únicamente el valor de las temperaturas.

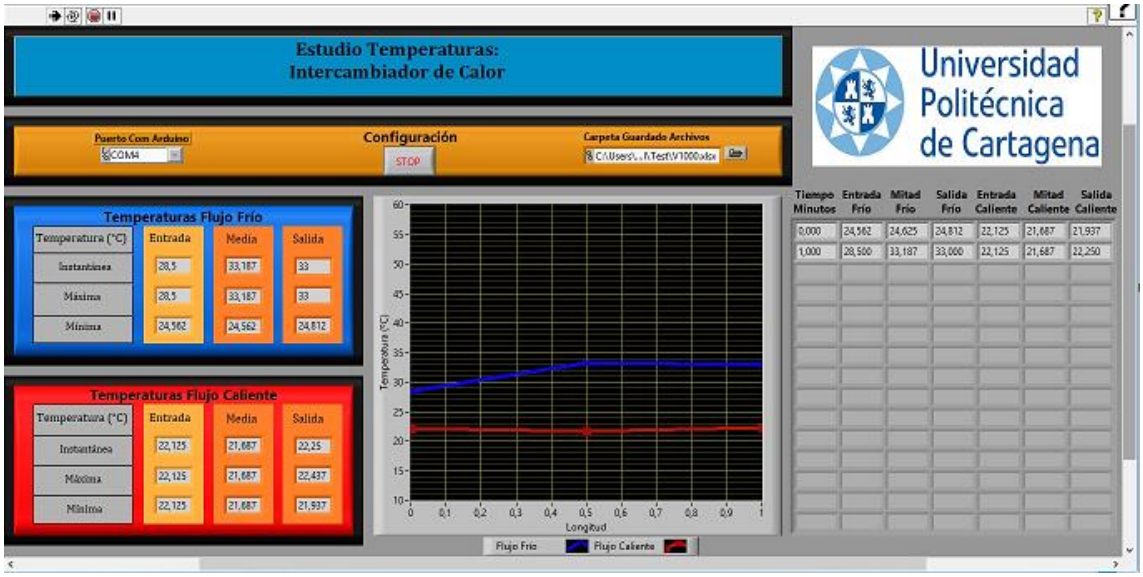

Figura 6.1 – Ejemplo funcionamiento

Una vez realizada la monitorización y, por ende, el ensayo, se nos habrá generado el archivo Excel en la localización y con el nombre indicados, mostrando los datos medidos de la siguiente forma:

	A	B	C	D	E	F	G
1	Time	Entrada Frío	Mitad Frío	Salida Frío	Entrada Caliente	Mitad Caliente	Salida Caliente
2	16/04/2018 16:40:50	22,25	21,75	21,937	22	21,562	21,812
3	16/04/2018 16:41:42	27,375	21,75	21,937	22,062	21,625	21,812
4	16/04/2018 16:42:43	26,25	21,812	22	22,062	21,625	21,875
5	16/04/2018 16:43:43	25,062	21,875	22	22,125	21,687	21,875
6	16/04/2018 16:44:43	24,25	21,875	22,062	22,187	21,75	21,875
7	16/04/2018 16:45:43	23,625	21,875	22,062	22,187	21,75	21,937
8	16/04/2018 16:46:44	23,187	21,875	22,062	22,187	21,75	21,937

Figura 6.2 – Ejemplo de ensayo de salida en Excel

6.2.Presupuesto.

El proyecto se ha basado en la monitorización mediante un equipo electrónico y software de instrumentación. Una de los objetivos planteados consistía en montar una instalación de calidad lo más económica posible. La lista de precios final se adjunta a continuación, obviando los softwares de trabajo, ya que el IDE de Arduino es libre, y, por tanto, gratuito, y la UPCT posee LabVIEW con licencia gratuita para estudiantes y laboratorios.

Materiales necesarios para este proyecto			
Material	**Cantidad**	**Precio Unitario(€/unidad)**	**Importe final (€)**
Arduino Mega 2560 + USB B016	1	9,49	9,49
Protoboard MB-102 (830 puntos)	1	1,78	1,78
50Resistencia 4,7KΩ 0,25W	50	-	1
Sensor de temperatura DS18B20 sumergible	6	1,79	10,74
Cable JUMPER	65	-	1,72
		Importe Total	29,73 €

Figura 6.3 - Tabla de presupuesto

Capítulo 7 Bibliografía.

El proyecto está basado en electrónica y software de programación, siendo práctico en su mayoría, la ayuda de la comunidad en línea y los ensayos han resultado de gran ayuda para su realización.

1)Documentación oficial de Arduino: https://www.arduino.cc/en/Guide/HomePage

2)Datasheets sensor DS18B20: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

3)Manual básico de programación en Arduino: http://foros.giltesa.com/otros/arduino/fc/docs/02-arduino_manual_de_programacion.pdf

4)Tutorial básico sensor DS18B20: http://www.naylampmechatronics.com/blog/46_Tutorial-sensor-de-temperatura-DS18B20.html

5)Floyd. Thomas L. Fundamentos de Electrónica Digital 9ª Edición, Pearson-Prentice Hall.2009.

6)Arduino thermometer with DS18B20: https://create.arduino.cc/projecthub/TheGadgetBoy/ds18b20-digital-temperature-sensor-and-arduino-9cc806

7) Sensores DS18B20 en serie, instalación y programación: http://cetroniconline.blogspot.com.es/2014/07/tutorial-arduino-iv-sensor-de.html

8) Foro: Arduino and Dallas DS1820 (one-wire): https://forums.ni.com/t5/LabVIEW-Interface-for-Arduino/Arduino-and-Dallas-DS1820-one-wire/td-p/3446528?start=0&tstart=0

9) LAJARA VIZCAÍNO, José Rafael. LabVIEW entorno gráfico de programación. Barcelona: Marcombo, 2010. 2ª edición. 477p, ISBN: 9789788426714