UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

TESIS

"DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE MONITOREO REMOTO PARA LA SIMULACIÓN Y EVALUACIÓN DE UN TUBO DE CALOR PARA CALENTAMIENTO DEL AIRE"

PARA OBTENER EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS CON MENCIÓN EN FÍSICA

ELABORADO POR:

CIRO ANTONIO CARHUANCHO LUCEN

ASESOR:

DR. ABEL AURELIO GUTARRA ESPINOZA

LIMA – PERÚ

2020

DEDICATORIA

A MI HIJO CHRISTIAN Y

A MI ESPOSA DOMNINA.

AGRADECIMIENTOS

Agradezco en primer lugar a mi hijo Christian y a mi esposa Domina que en todo momento me dieron su apoyo y ánimos para terminar esta tesis.

Agradezco a mi asesor y amigo Dr. Abel Gutarra, a Mag. Oswaldo Rojas por el apoyo en el uso del Laboratorio de Ingeniería Física.

Agradezco a todas las personas docentes, empleados y alumnos que contribuyeron en alguna forma a concluir esta tesis.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

Se diseñó y construyó un sistema electrónico para almacenamiento de datos provenientes de múltiples sensores y con capacidad de transmisión remota. El diseño se basó en una estructura modular que permite su adaptación a diversas condiciones experimentales como rangos de medición, valores críticos, y secuencias variables de medida.

Como aplicación particular del sistema diseñado, se eligió monitorear un equipo de prueba que consiste en el uso de tubos de calor (Heat Pipe) para generar aire caliente a partir de la captación de la radiación solar.

El análisis se basa en tomar las señales de temperatura del tubo de calor y procesarlos usando una tarjeta ARDUINO, para luego ser enviados remotamente usando una tarjeta Raspberry pi 3. El programa se encarga de seleccionar la señal, leer el dato del módulo de interfaz y grabarlo en una memoria microSD. También se puede acceder remotamente por medio del INTERNET para descargar los datos obtenidos en cualquier instante.

Las curvas obtenidas del tubo de calor por medio del sistema modular son analizadas para evaluar la eficiencia del tubo en el proceso de calefacción del aire.

Esta tesis está organizada en seis capítulos. En el primero se dan la descripción de un tubo de calor, la operación de un tubo de calor y la aplicación en un colector de terma solar para calentamiento del aire. En el segundo se detalla el diseño del monitoreo remoto. En el tercero la construcción física de los mismos (incluido el programa de adquisición de datos y de monitoreo remoto) y, en el capítulo cuarto la utilización del sistema de monitoreo para evaluar un tubo de calor aplicado al calentamiento del aire.

La discusión de los resultados obtenidos se tiene en el capítulo quinto y finalmente en el capítulo seis se tienen las conclusiones y recomendaciones para trabajos futuros en el área de investigación de la Instrumentación Científica Moderna.

CAPÍTULO 1. INTRODUCCIÓN

La separación entre el conocimiento abstracto y la realidad en las medidas es el instrumento. El avance científico está supeditado a la correcta toma de datos, a la transmisión y al procesamiento de los mismos. Gran parte de los logros alcanzados por las ciencias entre los siglos XIX al XXI de deben directamente a los instrumentos disponibles para explorar el mundo y cuantificarlo. El alcance de la investigación que permite la instrumentación se ha ampliado considerablemente, abarcando no solo el mundo natural (físico y biológico) sino también muchas facetas de la sociedad y el comportamiento humanos como la neuropsicología. La instrumentación científica ha sido con frecuencia un factor de estimulación para la investigación, dando resultados objetivos que muchas veces produjeron el abandono de teorías establecidas o la construcción de nuevas teorías.

En una investigación en física, la colección de datos es primordial, los mismos que por su naturaleza estadística, se deben muestrear por tiempos prolongados, lo cual hace que el investigador permanezca “atado” de su deber de coleccionar datos. La Instrumentación Científica en su área de automatización alivia esta necesidad, sabiendo que generalmente, los datos son abundantes y necesitan una gran cantidad de cálculos. Actualmente existen sistemas modulares que alivian este trabajo que, por ser de carácter rutinario, pueden ser ejecutados por un sistema de monitoreo adecuadamente construida.

Los conceptos de Instrumentación Científica Moderna tales como adquisición y tratamientos de datos se dan ahora en forma modular tanto para sensores como tarjetas de adquisición de datos mediante programas, se busca soluciones innovadoras para concebir y fabricar instrumentos científicos básicos con aplicaciones en física, química y biología. Esto presume el uso de una plataforma electrónica Arduino, la microcomputadora Raspberry pi 3 y módulos de sensores como de relés. También se monitorean y controlan en forma remota mediante el INTERNET usando módulos WIFI o Bluetooth. Estos sistemas son cada vez más económicos y al alcance de un gran número de Laboratorios o Centros de Investigación.

En la presente tesis se diseña y construye un sistema modular a un tubo de calor (llamada Interfaz remota) para monitoreo en forma remota, teniendo como aplicación la evaluación del tubo de calor.

1.1 Objetivo General

- Utilizar la Instrumentación Científica para desarrollar una interfaz modular capaz de coleccionar los datos de cualquier equipo mediante módulos de sensores y una plataforma electrónica Arduino y enviarlos mediante una microcomputadora Raspberry pi 3 B+ para la adquisición y procesamiento remoto en cualquier computadora conectada a INTERNET.

1.2 Objetivos específicos

- Construir una interfaz para para evaluar y caracterizar un tubo de calor (Heat Pipe), coleccionando y procesando los datos obtenidos en forma remota.
- Desarrollar un programa en lenguaje C++ para la adquisición de datos mediante la tarjeta Arduino y en lenguaje Python para la microcomputadora Raspberry pi 3 B+ para el monitoreo remoto.

CAPÍTULO 2. EL TUBO DE CALOR PARA CAPTACIÓN DE ENERGIA SOLAR

La utilización de sistemas termosolares es conocido como un método efectivo para enfrentar la alta demanda de energía y sus consecuentes desafíos, tiene un amplio potencial de desarrollo. El calentamiento solar del agua es la primera y más común aplicación de los sistemas solares térmicos. En nuestro medio existen muchos proyectos de investigación orientados a la captación de energía solar para zonas altoandinas, donde el promedio anual de temperatura está por debajo de los 10 ^{o}C. El estudio de sistemas de captación solar requiere toma de datos por tiempos prolongados, típicamente varios meses, debido a las variaciones estacionales.

Los desafíos relacionados con la recolección y el almacenamiento efectivos de la energía solar plantean limitaciones significativas para su desarrollo. Esto explica por qué el colector solar es el componente más crítico de cada sistema de calefacción solar.

2.1 Descripción de un tubo de calor

El tubo de calor es esencialmente un dispositivo de transferencia de calor pasivo con una conductividad térmica efectiva extremadamente alta. Es un dispositivo simple de circuito cerrado que puede transferir rápidamente calor de un lugar a otro utilizando un esquema de flujo de dos fases (Wang, Diao, Zhao, Liu y Wei, 2017, p. 230) [1]. También es un dispositivo de alta conductancia térmica que transfiere calor por circulación de fluido en dos fases que involucran un evaporador, un condensador y, a menudo, una sección adiabática (sección de vapor en la Figura 1) [2]. El intervalo de temperatura de funcionamiento de un tubo de calor está determinado por el tipo de fluido de trabajo utilizado y su envoltura de diseño óptimo.

2.2 Principio de operación de un tubo de calor

Los tres componentes básicos de un tubo de calor son:

- El contenedor
- El fluido de trabajo
- La mecha o estructura capilar

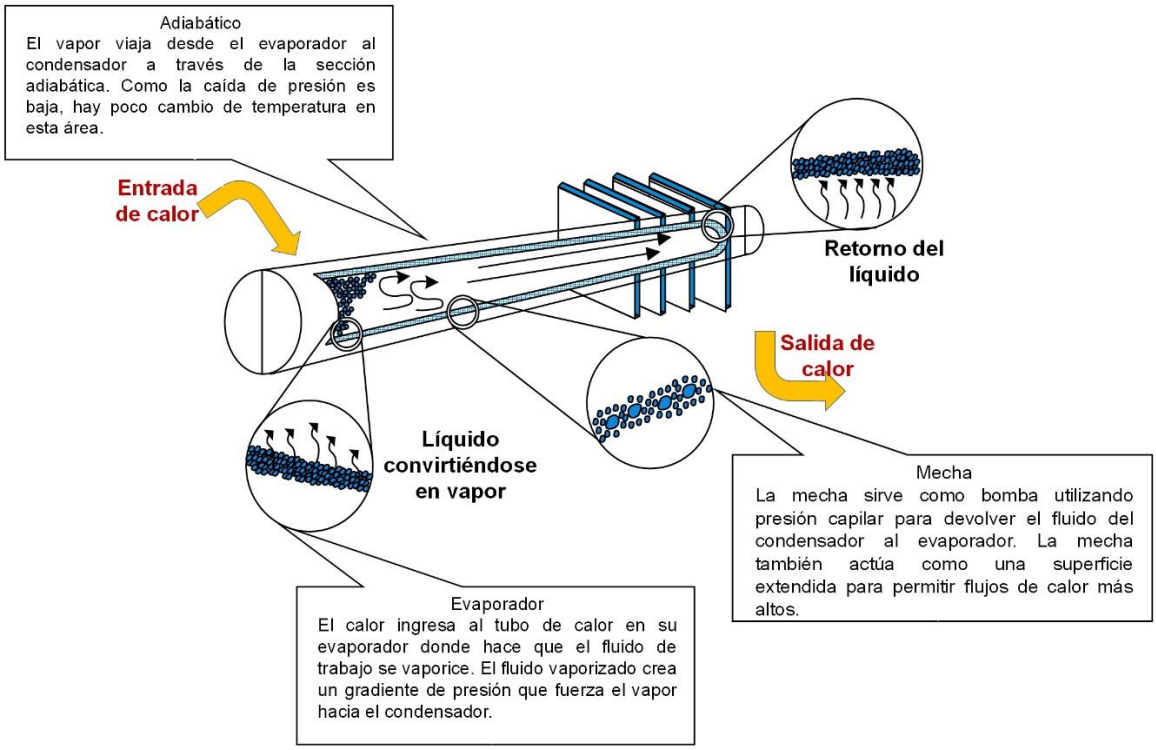


Figura 1. Diagrama básico de un tubo de calor. Fuente: Electronics-cooling, Scott D. Garner (1996), Heat Pipes for Electronics Cooling Applications, https://www.electronics-cooling.com/1996/09/heat-pipes-for-electronics-cooling-applications/

2.2.1 El contenedor

El contenedor tiene como función aislar el fluido de trabajo del ambiente exterior. Por lo tanto, debe ser a prueba de fugas, mantener el diferencial de presión a través de sus paredes y permitir el flujo de calor desde y hacia el fluido de trabajo [3].

La selección del material del contenedor depende de muchos factores, se observa en la Tabla 1.

Tabla 1. Factores para seleccionar el material del contenedor.

• Compatibilidad (tanto con el fluido de trabajo como con el entorno externo)
• Resistencia a la relación de peso
• Conductividad térmica
• Facilidad de fabricación, incluyendo soldadura, maquinabilidad y ductilidad.
• Porosidad
• Humectabilidad

El material no debe ser poroso para evitar la difusión del vapor. Una alta conductividad térmica asegura una caída de temperatura mínima entre la fuente de calor y la mecha.

2.2.2 El fluido de trabajo

Una primera consideración en la identificación de un fluido de trabajo adecuado es el rango de temperatura de funcionamiento del vapor. Dentro de la banda de temperatura aproximada, pueden existir varios fluidos de trabajo posibles, y se deben examinar una variedad de características para determinar el más aceptable de estos fluidos para la aplicación considerada. En la Tabla 2 se muestra los requisitos para seleccionar el fluido.

Tabla 2. Factores para seleccionar el fluido de trabajo.

• Compatibilidad con materiales de mecha y pared.
• Buena estabilidad térmica
• Humectabilidad de materiales de mecha y paredes.
• Presión de vapor no demasiado alta o baja sobre el rango de temperatura de funcionamiento.
• Alto calor latente
• Alta conductividad térmica
• Baja viscosidad de líquidos y vapores
• Alta tensión superficial
• Congelación aceptable o punto de fluidez

La selección del fluido de trabajo también debe basarse en consideraciones termodinámicas relacionadas con las diversas limitaciones al flujo de calor que se producen

dentro del tubo de calor, como los niveles de ebullición viscosa, sónica, capilar, de arrastre y nucleada.

En el diseño del tubo de calor, es deseable un alto valor de tensión superficial para permitir que el tubo de calor opere contra la gravedad y genere una alta fuerza motriz capilar. Además de la alta tensión superficial, es necesario que el fluido de trabajo humedezca la mecha y el material del recipiente, es decir, el ángulo de contacto debe ser cero o muy pequeño.

La presión de vapor sobre el rango de temperatura de funcionamiento debe ser lo suficientemente grande como para evitar altas velocidades de vapor, que tienden a establecer un gradiente de temperatura grande y causar inestabilidades de flujo [4].

2.2.3 La mecha o estructura capilar

Tiene una estructura porosa hecha de materiales como acero, aluminio, níquel o cobre en varios rangos de tamaños de poro. Se fabrican con espumas metálicas y, más particularmente, fieltros (lanas industriales), siendo estos últimos utilizados con mayor frecuencia. Al variar la presión sobre el fieltro durante el ensamblaje, se pueden producir varios tamaños de poro. Al incorporar mandriles (tipo especial de prensa usada para sujetar un objeto) metálicos extraíbles, también se puede moldear una estructura arterial en el fieltro.

Los materiales fibrosos, como la cerámica, también se han utilizado ampliamente. Comúnmente tienen poros más pequeños. La principal desventaja de las fibras cerámicas es que tienen poca rigidez y generalmente requieren un soporte continuo de una malla metálica. Por lo tanto, aunque la fibra en sí misma puede ser químicamente compatible con los fluidos de trabajo, los materiales de soporte pueden causar problemas. Más recientemente, el interés se ha convertido en fibras de carbono como material de mecha. Los filamentos de fibra de carbono tienen muchos surcos longitudinales finos en su superficie, tienen altas presiones capilares y son químicamente estables. Varias tuberías de calor que se han construido con éxito utilizando mechas de fibra de carbono parecen mostrar una mayor capacidad de transporte de calor.

Cuando el calor se transfiere a la sección del evaporador, el fluido hierve. Luego, el vapor sube o circula al extremo del condensador del tubo de calor, donde se pierde calor debido a la temperatura más baja en este lugar, y el vapor vuelve a su estado líquido. El líquido es conducido de regreso a la sección del evaporador por gravedad (en un tubo de calor asistido por gravedad), inercia (en un tubo de calor giratorio) o por acción capilar a través de una estructura de absorción (en tubos de absorción de calor). En una tubería de calor asistida por gravedad (una tubería de calor orientada en ángulo), el extremo inferior es el evaporador y el extremo superior es el condensador. Cuando el extremo del evaporador está más caliente que el condensador, se logra una alta conductividad térmica mediante la circulación.

Debido a su innovador diseño, los tubos de calor consiguen un rendimiento mayor que los colectores de placa plana convencionales prácticamente sin disminución energética y permite la obtención de energía en días nublados o a temperaturas bajo cero lo que lo hace también excelente para climas fríos y lluviosos.

2.2.4 Orientación con respecto a la gravedad

Para obtener el mejor rendimiento, la aplicación debe tener a la gravedad trabajando con el sistema; es decir, la sección del evaporador (calentada) debe ser más baja, con respecto a la gravedad, que la sección del condensador (enfriamiento). En otras orientaciones donde la gravedad no está ayudando al retorno del líquido condensado, el rendimiento general se degradará. La degradación del rendimiento depende de varios factores, incluida la estructura de la mecha, la longitud y el fluido de trabajo del tubo de calor junto con el flujo de calor de la aplicación. Un diseño cuidadoso puede minimizar la pérdida de rendimiento y permitir una predicción precisa del rendimiento [5].

2.3 Aplicación en un colector de terma solar

Para determinar los parámetros de eficiencia de los colectores solares térmicos, se pueden utilizar dos procedimientos diferentes: el método de prueba de estado estable y el método de prueba cuasi-dinámico.

Mientras la prueba es estado estable, todas las condiciones límite, como la irradiación solar, la temperatura ambiente y la temperatura de entrada del colector, se mantendrán invariables. Después de registrar el punto de datos en un rango representativo de condiciones de operación, la curva de eficiencia del colector se puede determinar mediante técnicas de regresión multilineales (Xu, Wang, Yuan, Li y Ruan, 2012, p. 1224) [6].

Durante la prueba cuasi dinámica, las condiciones de frontera se dejan libres para variar. En base de una serie de mediciones experimentales, también se determinan los parámetros específicos del colector. Con el método de prueba cuasi-dinámico, se pueden determinar los parámetros adicionales tales como la capacidad calorífica del colector y el coeficiente modificador del ángulo incidente además de la curva de eficiencia.

En ambas técnicas, el concepto básico es exponer el colector a la radiación solar y medir las temperaturas de entrada (T_{ENT}, en °C) y salida (T_{SAL}, en °C) del fluido de trabajo que fluye con un caudal conocido. La transferencia de calor Q (en W) obtenida por el fluido viene dado por,

$$Q = \dot{m} c_P (T_{SAL} - T_{ENT}) . \qquad (2.1)$$

La fuente de calor en un colector solar es la energía solar y la potencia de entrada suele ser la irradiación, G (en W/m^2), recibido en la superficie del colector, absorbido y luego transferido al fluido de trabajo. Al dividir la potencia de salida neta por la potencia de entrada, se puede definir una eficiencia general. La eficiencia η generalmente se considera como eficiencia instantánea porque es una función de los parámetros de operación instantáneas, incluidos las condiciones climáticas locales como la temperatura ambiente, la velocidad del viento, etc.,

$$\eta = \left[\dot{m} c_P (T_{SAL} - T_{ENT}) \right] / A_C G , \qquad (2.2)$$

donde A_C es la superficie del colector (en m^2).

La salida de potencia neta se puede escribir en términos de cantidades que representan el mecanismo de transferencia de calor, o la entrada de calor menos las pérdidas de calor [7],

$$Q = A_C F_R \left[G - U_L \left(T_m - T_a \right) \right], \qquad (2.3)$$

donde F_R es el factor de eliminación de calor del colector, T_a es la temperatura ambiente y U_L es el coeficiente global de pérdida de calor. T_m es la temperatura media del fluido de trabajo que fluye dentro del colector. La combinación del flujo de calor dada por la ecuación (2.3) con la definición de la eficiencia, y observando que U_L es generalmente una función de la temperatura, conduce a la siguiente expresión,

$$\eta = \eta_0 - a \frac{\left(T_m - T_a \right)}{G} - b \left(\frac{T_m - T_a}{G} \right)^2, \qquad (2.4)$$

donde η_0, a y b son constantes para ser evaluados analítica o experimentalmente.

El tubo de calor Heat Pipe utilizado en el colector solar, se compone de 2 tubos concéntricos de vidrio de borosilicato, habiendo entre estos tubos una presión inferior a 0.001 atmósferas, y existiendo además una capa absorbedora en el tubo interior que captura los rayos solares aportando así mucho calor al sistema (Figura 2).

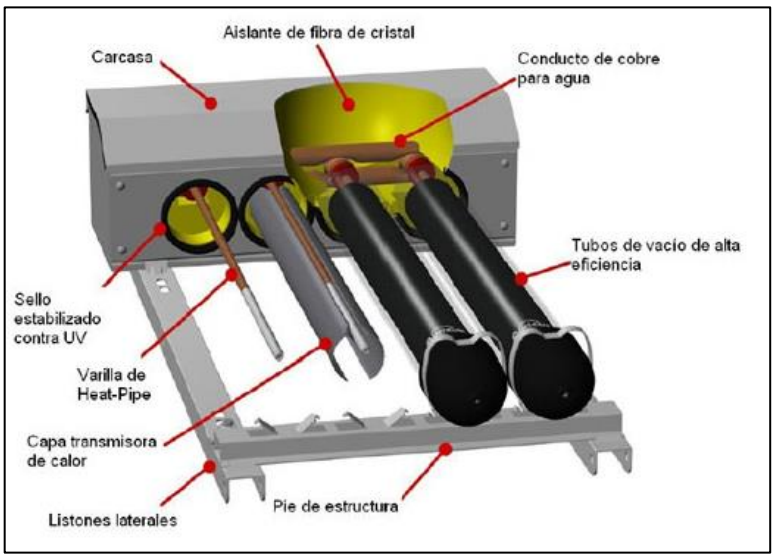


Figura 2. Partes de un colector solar. Fuente: Blog spot Eficiencia energética y utopía, Juan Francisco Diaz (2015), Conceptos de energía solar térmica, https://juanfrancisco207.wordpress.com/tag/colectores-solares/

CAPÍTULO 3. DISEÑO DE LA INTERFAZ REMOTA AL TUBO DE CALOR

En este capítulo se describe cada una de las partes que conforman el diseño del sistema: módulo de termocupla tipo K, módulo DS18B20, tarjeta Arduino Mega, tarjeta Raspberry pi 3, módulo DS3231, módulo display, módulo Relé.

3.1 Diseño del sistema modular de monitoreo remoto

Mediante la Instrumentación científica se busca contribuir con un sistema de monitoreo remoto económico y seguro para equipos de investigación científica, integrando la tecnología modular de tarjetas electrónicas y sensores a una interfaz accesible que permita de una manera simple monitorear y captar los valores obtenidos por los sensores, ya sea en forma local o remota mediante el INTERNET.

Para el diseño de un sistema de monitoreo remoto necesitamos (Figura 3):

- Una minicomputadora de placa reducida, en este caso la placa Raspberry pi 3, que tiene conexión con INTERNET mediante Ethernet y Wifi (también tiene conexión Bluetooth) [8].

- Una placa programable con un microcontrolador que controla entradas y salidas digitales, también entradas analógicas. En este caso la placa Arduino Mega cumple con los requisitos. Esta placa se conecta a los sensores, actuadores y periféricos.

- Módulos de sensores, un sensor es un artefacto capaz de captar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser, por ejemplo: temperatura, inclinación, desplazamiento, presión, aceleración, fuerza, torsión, humedad, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica, una tensión eléctrica, una capacidad eléctrica, una corriente eléctrica, etc. Los sensores se pueden clasificar en función de los datos de salida en: digitales, analógicos y comunicación por Bus.
- Módulo RTC, reloj en Tiempo Real es la solución ideal cuando necesitamos integrar mediciones de tiempo a nuestros proyectos. Los RTC son de muy bajo consumo por

lo que pueden ser alimentados por baterías y de esa forma no pierden la sincronización.

- Modulo MicroSD, esta tarjeta de interfaz está diseñada para acceder a la memoria microSD en modo SPI, se utiliza para almacenar los datos obtenidos y luego procesarlos.

- Módulo actuador, un actuador es un artefacto capaz de transformar energía eléctrica, hidráulica o neumática en la activación de un proceso con el objetivo de generar un efecto sobre el elemento externo. Los actuadores van conectados a las salidas de la placa Arduino.

- Módulos periféricos, es la denominación común para designar al aparato o artefacto auxiliar y autónomo (pantalla LCD, teclados, etc.) conectado a la unidad central de procesamiento o en este caso a la placa Arduino.

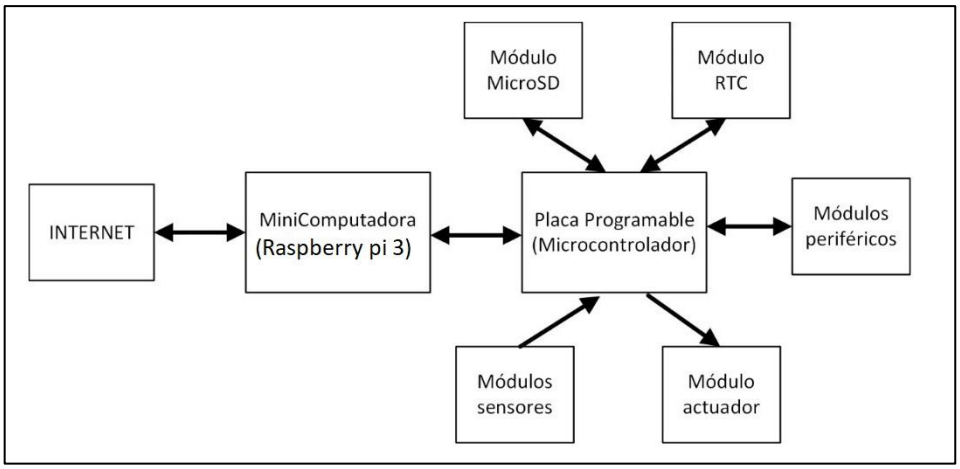


Figura 3. Diseño de un sistema modular remoto.

3.2 Sensores de temperatura

Los parámetros que debemos fijarnos a la hora de elegir un sensor son:

- **Sensibilidad**: podemos definir como la cantidad mínima que el sensor será capaz de medir y por lo tanto, modificará la salida. Si tomamos el ejemplo de un sensor de temperatura, la sensibilidad será cuantos grados es capaz de captar para que modifique la señal de la salida en voltios.
- **Rango de valores**: son los valores máximo y mínimo que es capaz de medir el sensor. En nuestro caso tendremos una temperatura mínima y una temperatura máxima. Dependerá de las condiciones físicas del propio sensor.
- **Precisión**: en términos coloquiales podemos decir que es el error que se produce entre el valor real y el valor obtenido. Por ejemplo, si tenemos la certeza de que la temperatura es de 25º C y medimos con el sensor, la desviación obtenida con el sensor nos dará la precisión ± Xº C.
- **Resolución**: si ya hemos visto la sensibilidad que nos indica la capacidad de detectar un cambio en la entrada, la resolución es igual, pero en la salida. Será el cambio mínimo detectable en la señal de salida. En nuestro caso, dependerá de la resolución de la entrada al microcontrolador en sensores analógicos y del propio sensor en sensores digitales.
- **Tiempo de respuesta**: los sensores no modifican su estado de salida inmediatamente. Para que cambie la salida con respecto a una entrada debe pasar un tiempo y a este tiempo se le llama el tiempo de respuesta. Por lo tanto, será el tiempo necesario para que cuando se produzca un cambio en la entrada este produzca un cambio en la salida.
- **Offset**: es un factor de corrección que debemos de tener en cuenta a la hora de hacer nuestros cálculos. Se puede resumir como el valor de salida que tenemos cuando debería ser cero [9].

Para la interfaz remota aplicado al tubo de calor se han elegido dos tipos de sensores: termocupla tipo K (para medir la temperatura del metal de cobre, que supera los 100°C, Figura 4) y el sensor de circuito integrado DS18B20 (para medir la temperatura del aire que

sale, que a lo más llega a 60°C, Figura 5). Para el colector de tubos de calor se necesita medir cuatro temperaturas, se muestra en la Tabla 1 los sensores y los rangos de temperaturas.

Tabla 3. Sensores de temperatura de la Interfaz

DISPOSITIVO	VARIABLE	RANGO INTERFAZ (°C)
Termocupla K: MAX6675	TH	0 - 120
DS18B20 (1)	T1	0 - 80
DS18B20 (2)	T2	0 - 60
DS18B20 (2)	T3	0 - 60

.

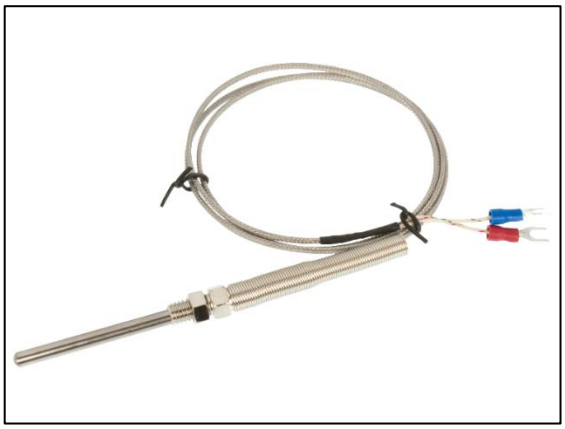


Figura 4. Termocupla tipo K.

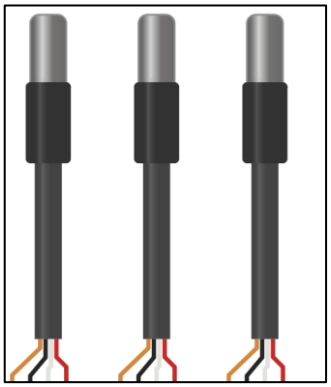


Figura 5. Sensores DS18B20.

3.2.1 Módulo de termocupla tipo K

Se utiliza un módulo en base al circuito integrado MAX6675, está diseñado para termocuplas tipo K que incluye una compensación de junta fría, tiene un conversor análogo a digital con una resolución de 12 bits en un rango de 0° hasta 1023°C. El protocolo de comunicación es SPI, por lo que puede comunicarse con una plataforma Arduino o Raspberry. Las especificaciones técnicas se muestran en la Tabla 4.

Tabla 4. Especificaciones técnicas del Módulo Termocupla tipo K.

• Voltaje de Operación: 5V • Corriente de trabajo: 50mA • Rango de Temperaturas Termocupla K: -20019:54°C hasta 1300°C • Resolución Transmisor MAX6675: 12 bits (0°C → 1023°C) • Rango de temperatura MAX6675: 0° → 1023°C • Resolución de temperatura: 0.25°C • Protocolo: SPI (Serial Peripheral Interface)

Se muestra en la figura la conexión de la placa Arduino Mega al módulo MAX6675.

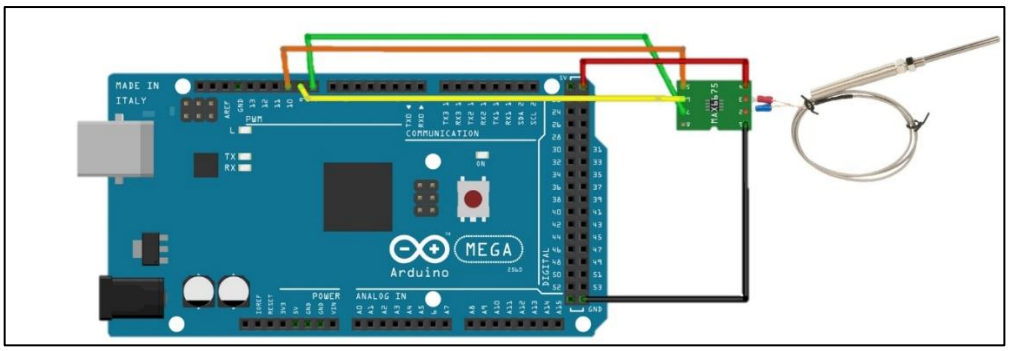


Figura 6. Conexión Arduino Mega y módulo MAX6675.

Conexión para Arduino Mega (ver Figura 6):

- D0 → pin 8
- CS → pin 9
- CLK → pin 10
- VCC → 5V
- GND → GND

3.2.2 Módulo DS18B20

El DS18B20 es un sensor digital de temperatura que se comunica mediante el protocolo 1-Wire, este protocolo necesita solo un pin de datos para comunicarse y tiene la ventaja de conectar más de un sensor en el mismo bus. Cada sensor trabaja con un pin diferente y necesita su propia resistencia Pull-up (resistencias que establecen un estado lógico en un pin o entrada de un circuito lógico cuando se encuentra en estado reposo) de 4.7 kΩ. Las especificaciones técnicas se muestran en la Tabla 5.

Tabla 5. Especificaciones técnicas del Módulo DS18B20.

• Voltaje de Operación: 5V • Corriente de trabajo: 4 mA • Resolución: de 9 y 12 bits. • Rango de temperaturas: -50° → 125°. • Resolución: 0.5°C. • Protocolo: OneWire (protocolo de comunicaciones de un alambre).

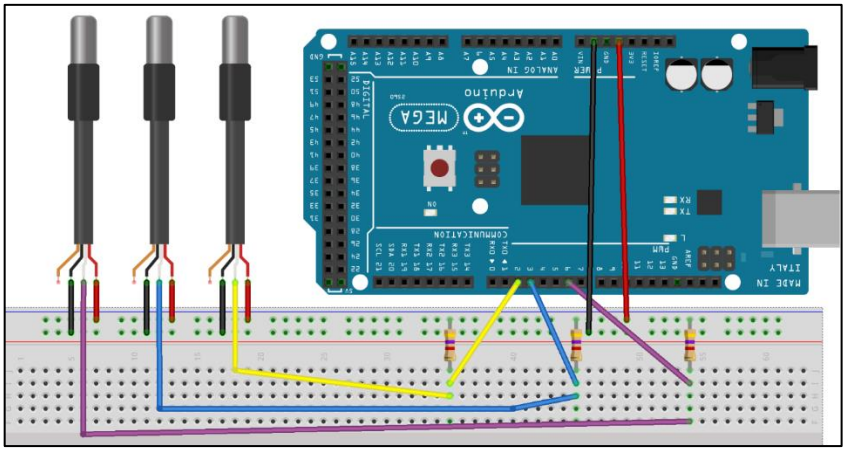


Figura 7. Conexión Arduino Mega al Sensor DS18B20.

Conexión para Arduino Mega (ver Figura 7):

- DQ1 → pin 2
- DQ2 → pin 3
- DQ3 → pin 6
- VCC → 5V
- GND → GND

3.3 Sistema modular de la interfaz

Los shields (módulos de tarjetas como sensores o actuadores) son placas de circuitos modulares que se conectan unas con otras para dar funcionalidad extra a una placa Arduino. Los shields tienen varios componentes integrados que evitan hacer cableados, permiten conectar y usarlos con la placa Arduino sin tener conocimientos especializados de electrónica [10].

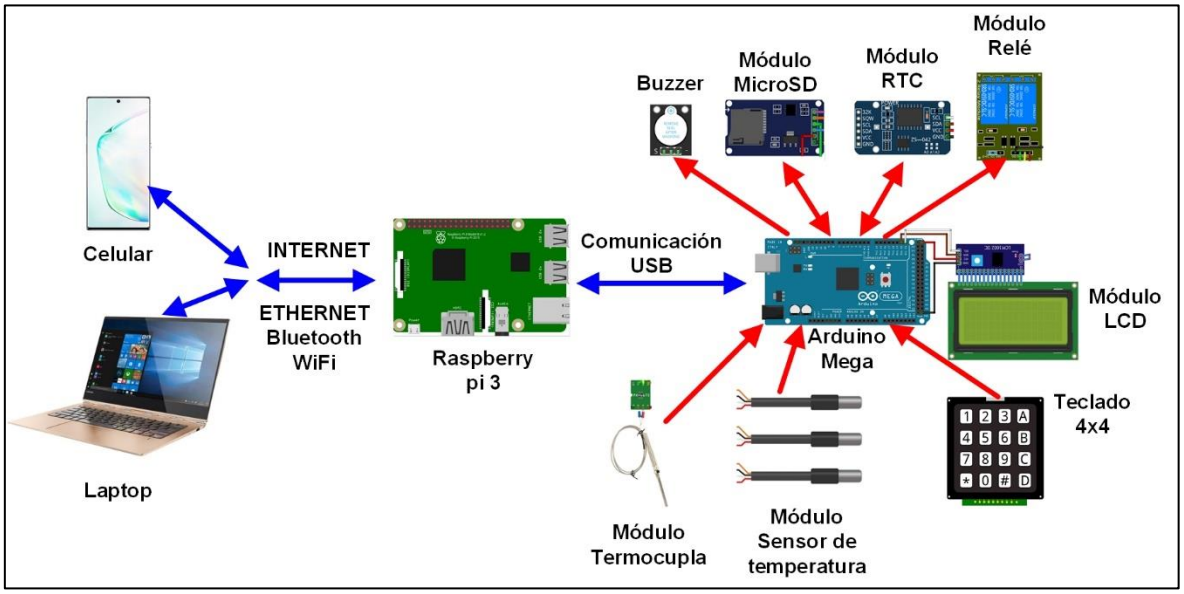


Figura 8. Sistema Modular de Interfaz remota.

Los módulos utilizados en la interfaz de los tubos de calor (Figura 8) son:

- Módulo de termocupla tipo K
- Módulo sensores DS18B20
- Módulo RTC DS3231
- Módulo MicroSD
- Módulo Relé
- Módulo pantalla LCD

3.3.1 Tarjeta Arduino Mega

La tarjeta Arduino Mega 2560 (Figura 9) es una placa con un microcontrolador construido en base al ATmega2560. Tiene 54 pines de entrada / salida digital (de los cuales 15 se pueden programar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un cabezal ICSP, y un botón de reinicio. Contiene todo lo necesario para el funcionamiento del microcontrolador; simplemente se conecta a una computadora con un cable USB o alimentándolo con un adaptador de CA a CC o una batería. Las especificaciones técnicas se muestran en la Tabla 6.

Tabla 6. Especificaciones técnicas de la Tarjeta Arduino Mega.

- Voltaje de Operación, 5V
- Voltaje de alimentación (recomendado), 7 → 12V
- Voltaje de alimentación (límite), 6 → 20V
- Número de pines digitales de E / S, 54 (de los cuales 15 proporcionan salida PWM)
- Número de pines de entrada analógica, 16
- Corriente (CC) por pin de E / S, 20 mA
- Corriente (CC) para 3.3V por pin, 50 mA
- Memoria flash, 256 KB de los cuales están ocupados 8 KB (para el gestor de arranque)
- SRAM: 8 KB, EEPROM, 4 KB
- Velocidad de reloj, 16 MHz

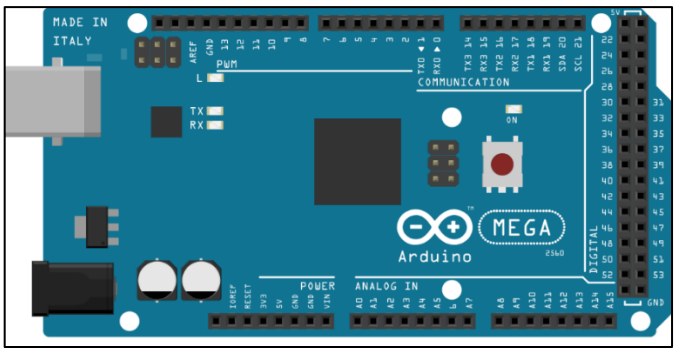


Figura 9. Tarjeta Arduino Mega 2560.

3.3.2 Tarjeta Raspberry pi 3 B+

La tarjeta Raspberry Pi 3 B+ (Figura 10) tiene un procesador Broadcom BCM2837B0 de 1.4GHz de 64-bits con 1GB de memoria RAM, al mismo tiempo cuenta con conectividad Wifi 802.1n y Bluetooth 4.2. Estas características lo convierten en una mini PC ideal para proyectos IoT (internet de las cosas) pero dado que su procesador es de 64-bits de 4 núcleos, también es muy capaz en cuanto a potencia de programación de alto nivel. Las especificaciones técnicas se muestran en la Tabla 7.

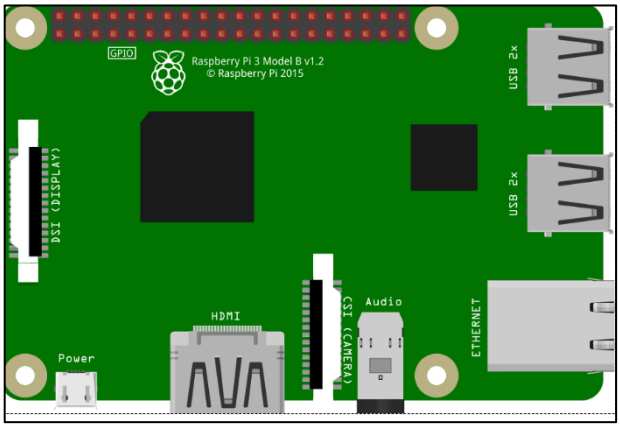


Figura 10. Tarjeta Raspberry pi 3 B+.

Tabla 7. Especificaciones técnicas de la Tarjeta Raspberry pi 3 B+.

- Voltaje de Operación, 5V. Corriente de trabajo máxima, 2,5 A
- Procesador Broadcom BCM2837B0 64-bit ARM Cortex-A53 Quad Core Processor SoC @ 1.4GHz
- Memoria RAM, 1GB LPDDR2 SDRAM
- 4 puertos USB2.0 con salida hasta 1.2A
- Tiene salida de Video/Audio (conector jack 3.5mm), HDMI, cámara CSI
- Zocalo: microSD
- Tiene red "Gigabit" Ethernet sobre USB 2.0 (max. 300Mbps)
- Sostiene redes Wifi 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac wireless LAN y Bluetooth 4.2, BLE
- Decodificación H.264, MPEG-4 (1080p30); H.264 encoding (1080p30); OpenGL ES 1.1, 2.0
- Periféricos integrados: 27 pines GPIO, UART, I2C, SPI.

3.3.3 Módulo DS3231

El módulo está basado en el circuito integrado RTC DS3231 de MAXIM y la EEPROM AT24C32 de ATMEL (Figura 12), tienen el mismo bus comunicación con el protocolo I2C. su precisión es muy alta debido a un oscilador interno compensado por temperatura. La memoria EEPROM AT24C32 almacena hasta 32Kbits (4K Bytes) de datos de manera permanente [11]. Las especificaciones técnicas se muestran en la Tabla8.

Tabla 8. Especificaciones técnicas del Módulo DS3231.

• Voltaje de operación, 3.3V o 5V • RTC de alta precisión DS3231 con oscilador interno. Exactitud Reloj, 2ppm • La dirección I2C del DS3132, Leer(11010001), Escribir(11010000) • Tiene memoria EEPROM AT24C32 (4K * 8bit = 32Kbit = 4KByte) • La comunicación I2C (protocolo bus maestro-esclavo), solo usa 2 cables. • La batería puede mantener al RTC funcionando por 10 años.

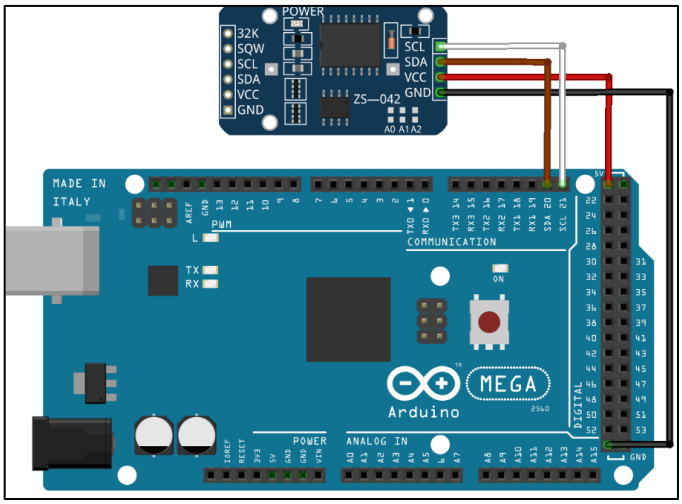


Figura 11. Conexión Arduino Mega y módulo DS3231.

Conexión para Arduino Mega:

- SCL → pin 21
- SDA → pin 20
- VCC → 5V
- GND → GND

3.3.4 Módulo MicroSD

Esta tarjeta de interfaz está diseñada para acceder a la memoria microSD en modo SPI, por lo que las señales de control se etiquetan claramente con los nombres de las señales en dicho bus de comunicaciones. Soporta tarjetas microSD y micro SDHC hasta 32GB. Tiene un circuito de conversión de voltaje para comunicarse a 3.3V o 5V. Puede ser alimentado hasta con 5V debido a su regulador de voltaje incluido. Se puede usar con la placa Arduino y en general con cualquier microcontrolador o tarjeta de desarrollo [12]. Las especificaciones técnicas se muestran en la tabla 9.

Tabla 9. Especificaciones técnicas del Módulo MicroSD.

• Voltaje de Alimentación: 3.3V o 5V. • Bus: SPI. • Cuenta con todos los pines SPI de la tarjeta SD: MOSI, MISO, SCK, CS. • Permite almacenar grandes cantidades de datos en memorias SD hasta 32 GB utilizando Arduino o PIC.

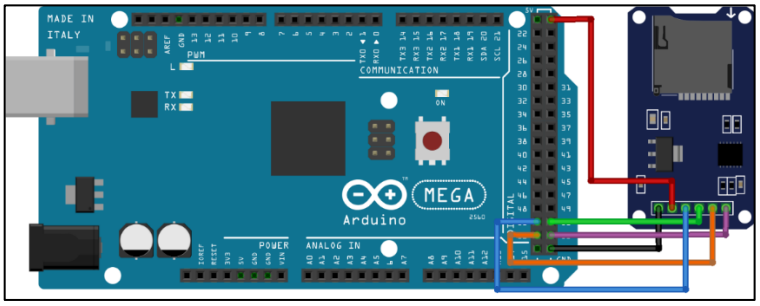


Figura 12. Módulo MicroSD.

Conexión para Arduino Mega (Figura 12):

- MISO → pin 50
- MOSI → pin 51
- CS → pin 53
- SCK → pin 52
- VCC → 5V
- GND → GND

3.3.5 Módulo pantalla LCD

El Módulo pantalla LCD contiene un adaptador LCD a I2C que está basado en el controlador I2C PCF8574, el cual es un expansor de entradas y salidas digitales controlado por I2C. Por el diseño del circuito impreso, este módulo se usa especialmente para controlar un LCD alfanumérico de 16 o 20 caracteres. El adaptador LCD a I2C se conecta directamente al LCD, esto lo podemos hacer a través de un protoboard o soldando directamente al LCD.. Las especificaciones técnicas se muestran en la Tabla 10.

Tabla 10. Especificaciones técnicas del Módulo pantalla LCD.

• Voltaje de alimentación: 5V. • Controlador: PCF8574. • Dirección I2C: 0x3F (en algunos modelos es 0x27). • Compatible con el protocolo I2C. • Jumper para Luz de fondo. • Potenciómetro para ajuste de contraste.

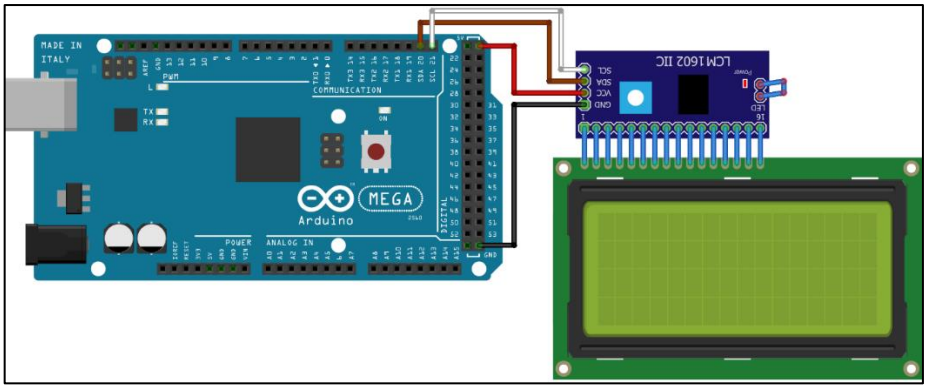


Figura 13. Conexión Arduino Mega y Display.

Conexión para Arduino Mega (figura 13):

- SCL – pin 21
- SDA – pin 20
- VCC - 5V
- GND – GND

3.3.6 Teclado 4x4

Un teclado matricial es un artefacto que reúne varios pulsadores y permite controlarlos empleando un número de conductores menor al que necesitaríamos al usarlos de forma individual. Estos artefactos reúnen los pulsadores en filas y columnas formando una matriz, la disposición que da lugar a su nombre. El teclado utilizado en la interfaz remota es 4x4 en circuito impreso. Las especificaciones técnicas se muestran en la Tabla 11.

Tabla 11. Especificaciones técnicas del Teclado 4x4.

• Tiempo de rebote (Bounce time): ≤5 ms • Máximo voltaje de alimentación: 24 V DC. Máxima corriente en operación: 30 mA • Resistencia de aislamiento: 100 MΩ (100 V) • Voltaje que soporta el dieléctrico: 250 VRMS (60Hz, por 1 min) • Expectativa de vida: 1.000.000 de operaciones

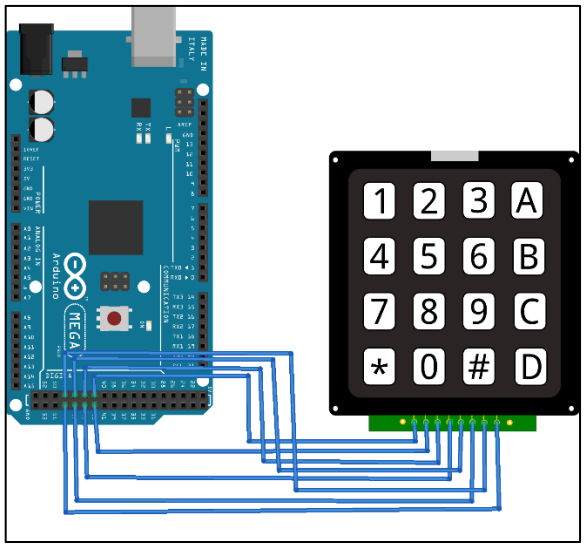


Figura 14. Conexión Arduino Mega y teclado 4x4.

Conexión para Arduino Mega (Figura 14):

- F1 → pin 42, C1 → pin 46
- F2 → pin 43, C2 → pin 47
- F3 → pin 44. C3 → pin 48
- F4 → pin 45, C4 → pin 49

3.3.7 Módulo Relé

Esta placa puede ser controlada directamente desde cualquier controlador, solo se requiere que el microcontrolador proporcione 5 Voltios y 20 mA por cada uno de los dos pines y fácilmente estará comandando artefactos eléctricos de hasta 10 Amperes y 220 voltios en corriente alterna o artefactos de 10 Amperes y 30 Voltios en corriente continua. Las especificaciones técnicas se muestran en la Tabla 12.

Tabla 12. Especificaciones técnicas del Módulo Relé.

• Voltaje de operación de 5 Vcc. • Corriente de activación de 15 a 20 mA. • 2 canales independientes, protegidos con opto acopladores. • Voltaje máximo de carga en los relés de 250VCA x 10A o 30VCD x 10A. • Modo de funcionamiento enclave. • Los 2 relés cuentan, cada uno con salidas NC y NA. • Distancia de alcance de 15 a 30 metros.

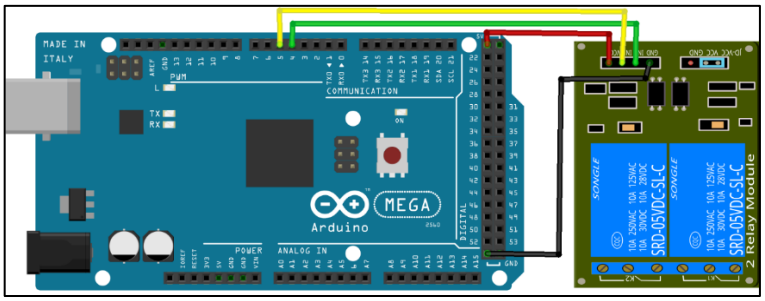


Figura 15. Conexión Arduino Mega y Relés.

Conexión para Arduino Mega (Figura 15):

- Relé 1 → pin 4
- Relé 2 → pin 5
- VCC → 5V
- GND → GND

3.4 Mapeo de los dispositivos electrónicos

Para conectar la tarjeta Arduino Mega con los diferentes módulos se requiere un mapeo (ver Tabla 2) para evitar errores en la conexión.

Tabla 13. Mapeo de los dispositivos electrónicos.

DISPOSITIVO			ARDUINO MEGA		
	Función	Pines	Cable	Pines	Cable
Termocupla : MAX6675	VCC	1	Rojo	VCC	Rojo
	GND	2	Negro	GND	Negro
	SCK	3	Naranja	10	naranja
	CS	4	Amarillo	9	amarillo
	SO	5	Verde	8	verde
DS18B20 (1)	VCC	1	Rojo	VCC	Rojo
	GND	2	Negro	GND	Negro
	OUT	3	Amarillo	2	Amarillo
DS18B20 (2)	VCC	1	Rojo	VCC	Rojo
	GND	2	Negro	GND	Negro
	OUT	3	Azul	3	Azul
DS18B20 (3)	VCC	1	Rojo	VCC	Rojo
	GND	2	Negro	GND	Negro
	OUT	3	Verde	6	Verde
RELAY	VCC	4	Rojo	VCC	Rojo
	GND	1	Negro	GND	Negro
	IN1	2	Verde	4	Verde
	IN2	3	Amarillo	5	Amarillo
DISPLAY LCD	VCC	3	Rojo	VCC	Rojo
	GND	4	Negro	GND	Negro
	SDA	2	Marron	20	Marron
	SCL	1	Blanco	21	Blanco
MicroSD card	VCC	2	Rojo	VCC	Rojo
	GND	1	Negro	GND	Negro
	CS	6	Violeta	53	Violeta
	SCK	5	Naranja	52	Naranja
	MOSI	3	Verde	51	Verde
	MISO	4	Azul	50	Azul
RTC DS3231	VCC	1	Rojo	VCC	Rojo
	GND	2	Negro	GND	Negro
	SDA	3	Marron	20	Marron
	SCL	4	Blanco	21	Blanco
BUZZER	VCC	1	Rojo	VCC	Rojo
	GND	3	Negro	GND	Negro
	SO	2	Amarillo	6	Amarillo
Teclado 4x4	F1	1	Violeta	42	Violeta
	F2	2	Violeta	43	Violeta
	F3	3	Violeta	44	Violeta
	F4	4	Violeta	45	Violeta
	C1	5	Azul	46	Azul
	C2	6	Azul	47	Azul
	C3	7	Azul	48	Azul
	C4	8	Azul	49	Azul

3.5 Diseño del software

Existen muchas metodologías para diseñar programas, aplicaremos una metodología sencilla, que es adecuada para la construcción del software para placas Arduino y Raspberry pi , y que se puede resumir en los siguientes pasos (Figura 16) [13]:

- **Analizar el problema.** comprender **cuál** es el problema que se trata de resolver, abarcando el contexto en el cual se utilizará.
- **Especificar la solución.** éste es el momento en el cual se describe **qué** debe hacer el programa, sin importar el cómo. Aquí deberemos decidir cuáles son los datos de entrada que se nos suministran, cuáles son las salidas que debemos elaborar, y cuál es la relación entre todos ellos.
- **Diseñar la solución.** éste es el lugar en el cuál abordamos el **cómo** vamos a resolver el problema, cuáles son los algoritmos y las estructuras de datos que utilizaremos. Analizamos posibles variantes, y las elecciones las tomamos usando como dato de la realidad el contexto en el que se aplicará la solución, y los costos relacionados a cada diseño.
- **Implementar el diseño.** Trasladar a un lenguaje de programación (en nuestro caso, C++ para Arduino y Python para Raspberry pi) el diseño que elegimos en el punto anterior.
- **Probar el software.** Planear un conjunto de ensayos para probar cada una de sus partes por separado, y también la correcta integración entre ellas.
- **Mantener el programa.** Realizar los cambios en respuesta a nuevas demandas del sistema.

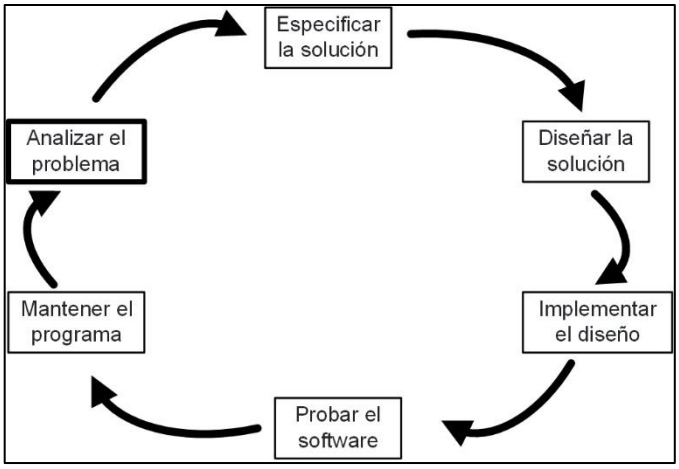


Figura 16. Diseño del software.

3.5.1 Lenguaje C para Arduino

El lenguaje de programación de Arduino está basado en C++, también es posible usar comandos estándar de C++ en la programación de Arduino. Las características del lenguaje C son:

- Es un lenguaje de programación asociado al sistema operativo UNIX.
- Es un lenguaje de medio nivel. Maneja objetos básicos como caracteres, números, etc., también con bits y direcciones de memoria.
- Se puede ejecutar en diversas plataformas (x86, IA64, amd64, etc.)
- Se emplea para la programación de sistemas: compiladores, construcción de intérpretes, editores de texto, etc.

El software se descarga de la página web: **https://www.arduino.cc/en/Main/Software** donde podemos descargar la última versión del software de desarrollo de Arduino para cualquier SO, así como las versiones anteriores, código fuente, programa beta de las próximas versiones y otro software relacionado [14].

El entorno de desarrollo integrado (IDE) de Arduino (Figura 17) es una adaptación multiplataforma (para Windows, macOS, Linux) que está desarrollada en el lenguaje de programación Java. Se utiliza para escribir y cargar programas en placas compatibles con Arduino, pero también, con la ayuda de núcleos de terceros, se puede usar con placas de desarrollo de otros proveedores.

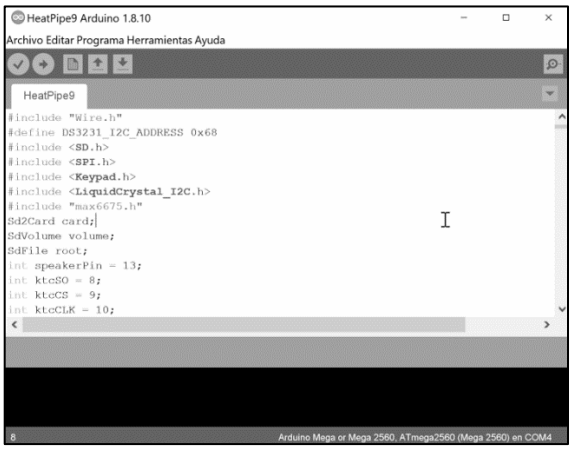


Figura 17. Entorno Arduino IDE.

3.5.2 Lenguaje Python para Raspberry pi 3

El sistema operativo Raspbian (Debian Jessie) es una distribución desarrollada por Linux, utilizada para las minicomputadoras de placa reducida como Raspberry pi 3. El objetivo de esta distribución es el acercamiento de más desarrolladores de software al mundo que hay detrás de las siglas GNU/Linux, conocida como UNIX.

En el sistema operativo Raspbian, Thonny (Figura 18) es un nuevo entorno de desarrollo (IDE) para poder desarrollar en lenguaje Python sobre nuestra Raspberry. Tiene las funcionalidades de un IDE profesional como la inspección de código, la ejecución paso a paso, los coloreados, etc., pero sigue manteniendo la simplicidad de un editor de código como puede ser Notepad en Windows [15].

Figura 18. Thonny entorno de desarrollo de Python.

Python es un lenguaje de programación de alto nivel, interactivo e interpretado, es de código abierto, multi-plataforma y se adecua a diversos paradigmas de programación (programación orientada a objetos, programación imperativa y programación funcional, etc.). Python 3 es una versión revisada del lenguaje, la cual fue publicada en el 2009 y que incluye modificaciones y mejoras que lo hacen incompatible con código de versiones previas; mientras que Python 2 es una versión que es compatible con código antiguo.

CAPÍTULO 4. CONSTRUCCIÓN DE LA INTERFAZ REMOTA

En este capítulo se describe la construcción física de la interfaz remota, el ensamblaje de las placas Arduino mega y Raspberry pi 3 B+ con los módulos (shields). También los programas de adquisición de datos para la placa Arduino, comunicación remota con la placa Raspberry mediante INTERNET.

4 .1 Construcción del sistema de monitoreo remoto

La interfaz remota al colector de los tubos de calor (Figura 19) consiste en un sistema de tres etapas:

- La fuente de alimentación.
- La placa Arduino y sus módulos (shields).
- La placa Raspberry (minicomputador).

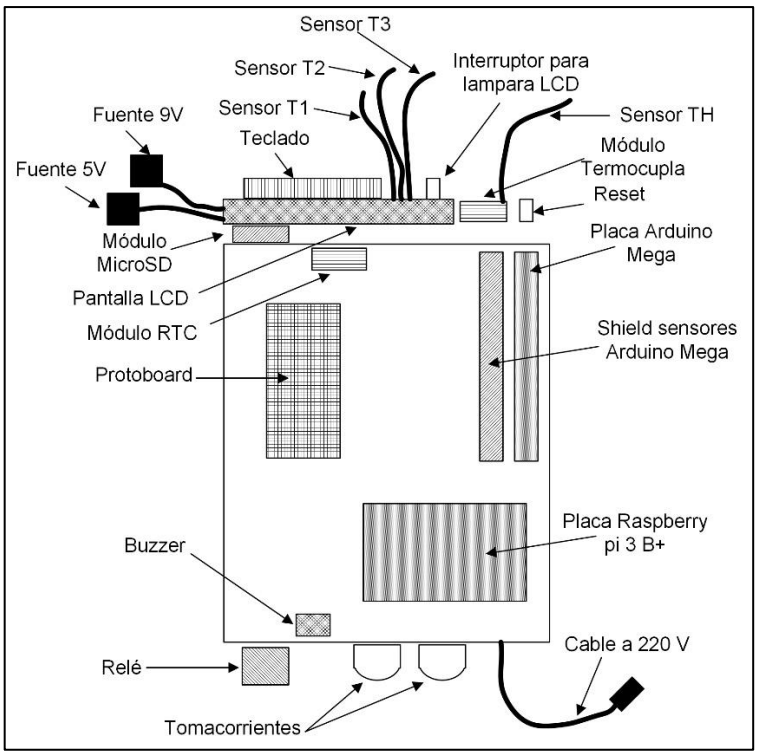


Figura 19. Construcción de la interfaz remota (vista de planta).

4.2 Características técnicas del Sistema interfaz remota

Las características técnicas de la Interfaz remota (Tabla 3) corresponden a la placa Raspberry pi 3 B+, placa Arduino Mega, sensores, módulos, periféricos y fuente de alimentación.

Tabla 14. Características técnicas de la Interfaz remota.

Tipo de equipo	Interfaz remota
Sensores	Termocupla : MAX6675 (TH), 12-Bit, resolución 0.25°C, tiempo de conversión 0,22 s
	DS18B20 (T1), 12-Bit, resolución 0,5°C, tiempo de conversión 0,75 s
	DS18B20 (T2), 12-Bit, resolución 0,5°C, tiempo de conversión 0,75 s
	DS18B20 (T3), 12-Bit, resolución 0,5°C, tiempo de conversión 0,75 s
Relés	2 puertos C.A. 220 V
Procesador y memoria RAM	Broadcom BCM2837B0 de 1.4GHz de 64-bits con 1GB de memoria RAM
Almacenamiento externo	MicroSD hasta 32GB
Reloj RTC	Placa Arduino
Sistema operativo	Raspbian
Redes y conectividad	Ethernet, WiFi, Bluetooth
Teclado	Menbrana 4x4
Pantalla	LCD 20x4
Diseño	Lata en color blanco
Programación remota	Python y Arduino IDE
Peso neto	1,27 kg
Dimensiones	altura 15 cm, largo 20 cm, ancho 15 cm
Consumo total de potencia	aproximadamente 2,25 W
Fuente de alimentación	2,5 A max.- 5 V (Raspberry pi 3)
	1,5 A max. - 9 V (Arduino)
Costo prototipo	S/. 920.00

4.3 Programa de adquisición de datos (Arduino)

El software provisto con la Interfaz remota es un software abierto y flexible con posibilidades de interacción con el usuario y que pueda ser modificado remotamente para aplicaciones especiales tanto por el usuario como por personal de apoyo informático del equipo de investigación.

El programa de Arduino se denomina sketch o proyecto y tiene la extensión "**.ino**". Hay que tener en cuenta: para que funcione el sketch, el nombre del fichero debe estar en una carpeta con el mismo nombre que el sketch.

La estructura básica de un sketch de Arduino es bastante simple y se compone de al menos dos partes. Las siguientes partes son obligatorias y encierran bloques que contienen declaraciones, estamentos o instrucciones:

- **setup()** es la parte que contiene la configuración de los dispositivos.
- **loop()** es la parte que contiene el programa que se ejecuta cíclicamente (de ahí el término loop –bucle-).

Además se puede incluir una introducción con los comentarios que describen el programa, la declaración de las variables y llamadas a librerías.
En el programa de adquisición de datos se utilizan las siguientes librerías:

- **OneWire.h**, accede a sensores de temperatura de 1 cable, memoria y otros dispositivos.
- **DallasTemperature.h**, biblioteca Arduino para circuitos integrados de temperatura del fabricante Dallas.
- **SD.h**, la biblioteca SD permite leer y escribir en tarjetas SD
- **SPI.h**, esta biblioteca permite comunicarse con dispositivos SPI, con Arduino como dispositivo maestro.
- **Keypad.h**, el Keypad es una biblioteca para usar teclados de estilo matricial con Arduino. A partir de la versión 3.0, ahora admite múltiples pulsaciones de teclas.
- **LiquidCrystal_I2C.h**, la biblioteca permite controlar pantallas I2C con funciones extremadamente similares a la biblioteca LiquidCrystal.

Una función es un segmento de código que está separado del código principal del programa, que realiza una tarea particular y regresa a la zona del programa donde fue

llamada, la función se llama con **void <u>nombre de la función</u>**. En el programa de adquisición de datos tenemos las funciones:

- **void setDS3231time**: actualiza la hora, minutos, segundos y fecha en el módulo RTC.
- **void readDS3231time**: lee la hora, minutos, segundos y fecha del módulo RTC.
- **void displayTime()**: muestra los datos del reloj en tiempo real.
- **void tomadedatos():** función de adquisición de datos.
- **void tecladatos()**: función para leer dato del teclado.
- **void beep1()**: función alarma.

El programa de adquisición de datos Arduino (Figura 21) de la Interfaz remota toma datos cada minuto y envía los datos a la placa Raspberry (temperaturas T1, T2, T3, TH y tiempo), el programa tiene dos partes (Figura 20):

- Simulación, se va a simular la radiación solar sobre un tubo Heat Pipe. Se tiene dos formas

  a) Periódica, para esto se enciende y apaga (mediante el relé1) los focos Dicroico 50W luz cálida a intervalos de 0,5h, 1h, 2h, 3h o 4h.

  b) Continua, se enciende (mediante el relé1) los focos Dicroicos durante 5h continuas.

- Luz solar, se realiza mediciones continuas durante 7 días.

Figura 20. A la izquierda se muestra el equipo para simulación y a la derecha el equipo para luz solar.

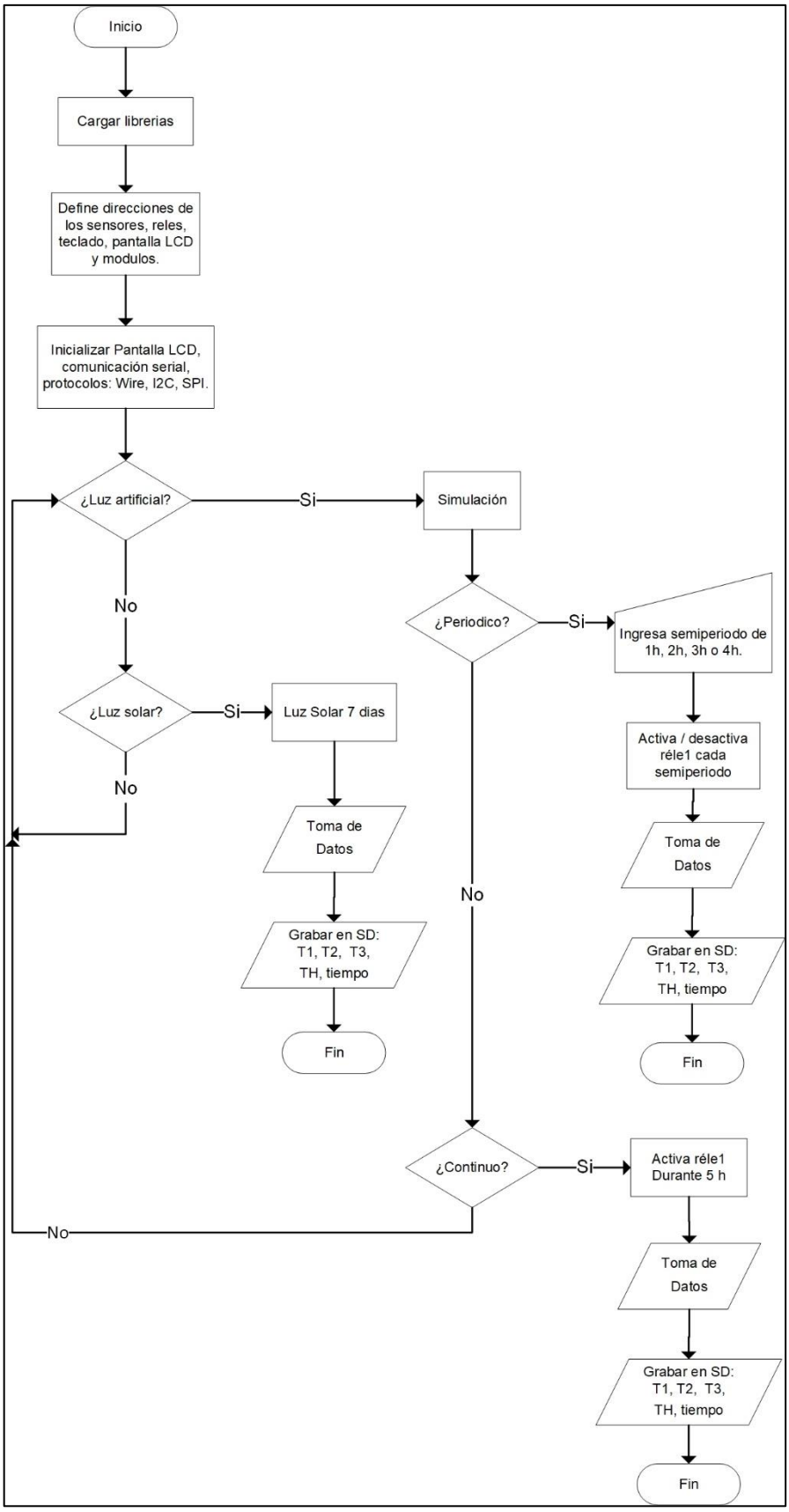


Figura 21. Programa de adquisición de datos Interfaz remota.

4.4 Programa de comunicaciones (Raspberry pi 3)

El programa de Python se llama script y deberá guardarse con una extensión "**.py**". Script es un archivo de texto sin formato que contiene código Python que está destinado a ser ejecutado directamente por el usuario. Por otro lado, un archivo de texto sin formato, que contiene código Python que está estructurado para ser importado y utilizado desde otro archivo Python, se llama módulo [16].

En el programa Python de comunicaciones de datos se utilizan los siguientes módulos:

- **Serial**, este módulo encapsula el acceso para el puerto serie.
- **Time**, este módulo proporciona varias funciones relacionadas con el tiempo.
- **Os**, este módulo permite acceder a la funcionalidad del sistema operativo, como por ejemplo si se desea leer o escribir un archivo.

La estructura básica de un script de Python es bastante simple y se compone de al menos dos partes:

- Definición de constantes, funciones, módulos.
- Bloque principal, entrada de datos, procesamiento entrega de resultados (salida).

El programa de comunicación de datos Python (Figura 22) de la Interfaz remota Heat Pipe espera los datos enviados por la placa Arduino, el programa tiene dos partes:

- Recibir datos de luz solar, cuando la placa Arduino envía la selección luz solar (crea el archivo SOLAR con la fecha) a la placa Raspberry espera recibir el inicio de la toma de datos para grabar temperaturas T1, T2, T3, TH y tiempo. Cuando recibe el Fin de la toma de datos se cierra el archivo.
- Recibir datos de Simulación, cuando la placa Arduino envía la selección simulación (crea el archivo SIMUL con la fecha y tipo de simulación) a la placa Raspberry espera recibir el inicio de la toma de datos para grabar temperaturas T1, T2, T3, TH y tiempo. Cuando recibe el Fin de la toma de datos se cierra el archivo.

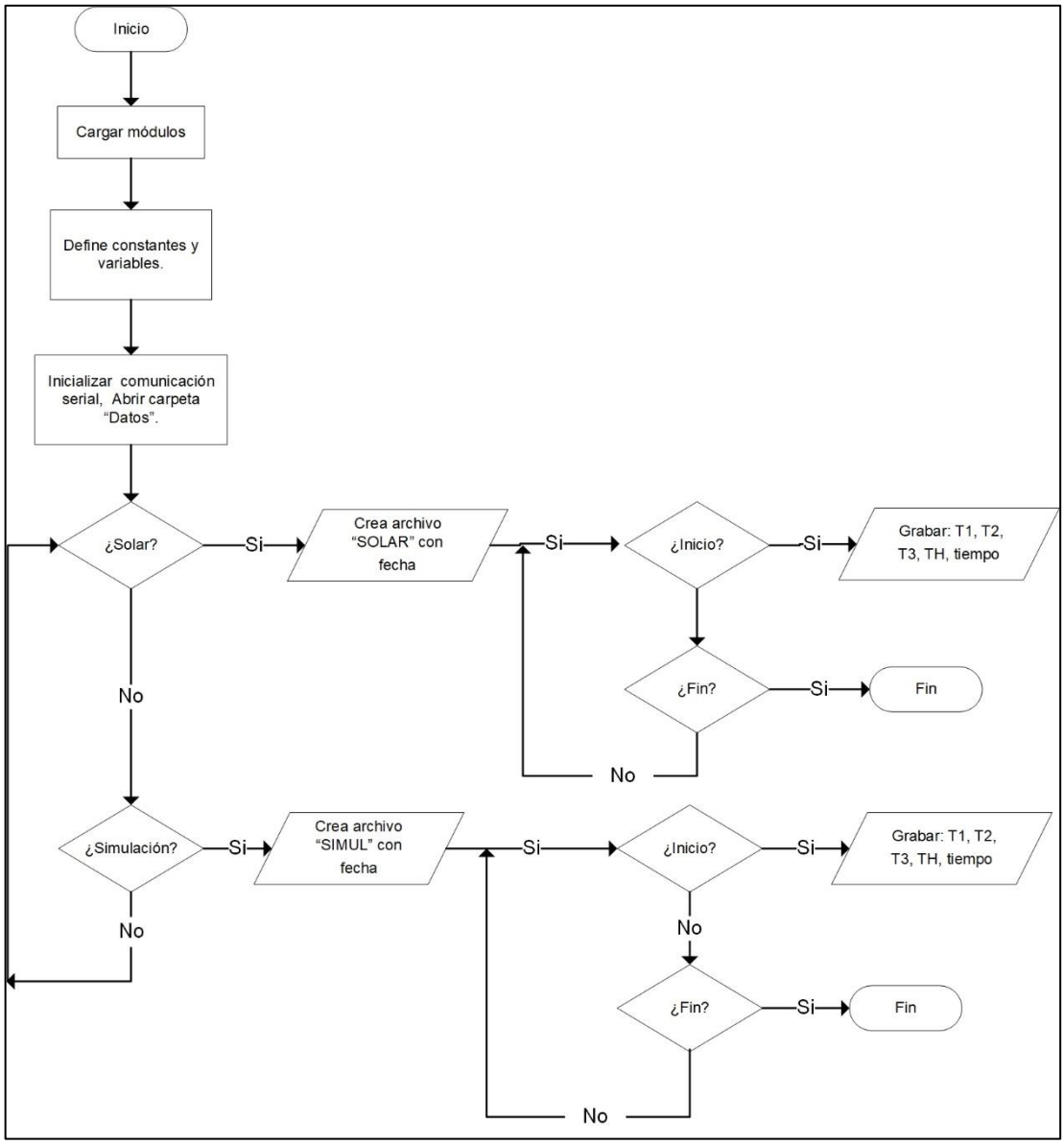


Figura 22. Programa de comunicación de datos de la Interfaz remota.

4.5 Conexión remota

Existen varias herramientas y protocolos que nos permiten conectarnos de forma remota a una computadora, algunos de ellos de forma insegura, al no estar cifrados, como es Telnet, y otros pensados para un uso concreto, como FTP, para poder acceder a los archivos. Para conectarnos de forma remota a cualquier servidor Linux y a placas minicomputadoras como el Raspberry Pi de forma segura y, además, hacer uso ilimitado de la placa y de todos sus recursos, utilizando su interfaz gráfica entonces debemos recurrir a otras aplicaciones que nos permiten utilizar en modo gráfico nuestra computadora, como es el caso de VNC y TeamViewer.

TeamViewer ofrece una de las soluciones de control remoto Raspberry Pi más eficaces y eficientes. Tiene varias funciones como: compartir y controlar escritorios, videoconferencias, reuniones en línea y transferencia de archivos entre computadoras. Fácil de descargar y de usar, permite acceder al dispositivo Raspberry Pi de forma remota, desde cualquier parte del mundo [17]. Utilizamos la aplicación TeamViewer para conectarnos desde una computadora o teléfono Smart al equipo Interfaz remota.

Para acceder desde una computadora o teléfono Smart a la Interfaz remota, antes debemos seguir los siguientes pasos:

- Descargar e instalar el programa TeamViewer Host en la placa Raspberry desde la página web, https://www.teamviewer.com/es/descarga/linux/
- Descargar e instalar Arduino Linux Arm en la placa Raspberry de la página web, https://www.arduino.cc/en/Main/Software.
- Crear en el directorio pi la carpeta Datos.
- Desde la placa Raspberry, utilizando el entorno de desarrollo **Thonny** ejecutamos el programa heatpipe2.py, luego podemos ver remotamente los datos que envía la placa Arduino a la placa Raspberry.
- Los datos obtenidos se pueden copiar a una computadora o teléfono Smart y procesarlos mediante una aplicación (Excel, Origin Pro, etc.).
- La ventaja con este sistema de monitoreo es que los datos no se perderán si se desconecta el INTERNET porque se almacenan en la carpeta “Datos” de la placa Raspberry.

CAPÍTULO 5. RESULTADOS Y DISCUSIONES

Para verificar el funcionamiento de la Interfaz remota al tubo de calor (Heat Pipe) y su versatilidad en la adquisición de datos y transmisión remota, lo aplicaremos a tres casos:

(a) Respuesta de un tubo de calor, con radiación artificial intermitente.

(b) Respuesta de un tubo de calor, con radiación artificial, para diferentes flujos de aire de entrada.

(c) Monitoreo de temperaturas de un colector solar de 12 tubos de calor para generación de aire caliente.

5.1 Respuesta de un tubo de calor, con radiación artificial intermitente.

Esta prueba puede ser utilizada para evaluar la eficiencia de conversión, radiación visible $\rightarrow$ calor, de un tubo de calor. El montaje consiste de un tubo de calor inclinado 12^{o} respecto a la horizontal donde la parte saliente del metal de cobre, llamado condensador, se encerró dentro de una cámara aislante con dos aberturas: una para la entrada de aire con un cierto flujo y el otro para la salida de aire caliente, como se muestra en la Figura 23. En este primer diseño, se iluminó perpendicularmente al tubo de calor, por medio de 11 focos dicroicos de 50W, los focos se espaciaron uniformemente de tal forma que la luz incidió perpendicularmente a lo largo del tubo.

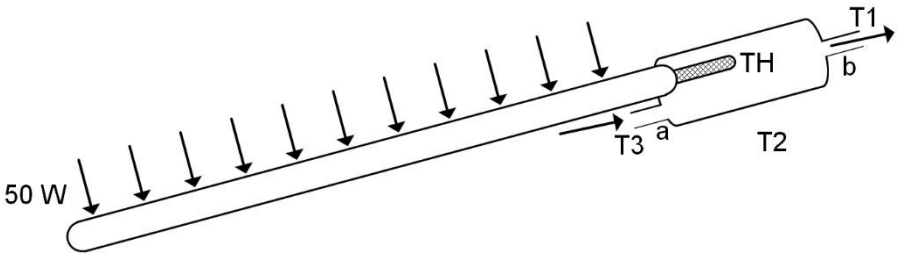


Figura 23. Simulación de la radiación solar en un tubo de calor.

Como se muestra en la Figura 23, el aire se introduce (por medio de una compresora) por **a** con un flujo $\dot{V}$, con temperatura T3 y sale por **b** con una temperatura mayor T1. Lo que se busca en esta parte es conocer la respuesta dinámica de los procesos de transferencia de calor desde los focos radiantes hasta el aire caliente. Con este fin, la Interfaz remota, se

programó en modo periódico con intervalos de encendido y apagado de 30 minutos para un flujo de aire de 10 l/min, y con intervalo de toma de datos de 3 minutos. Los resultados de estas mediciones se muestran en la Figura 24.

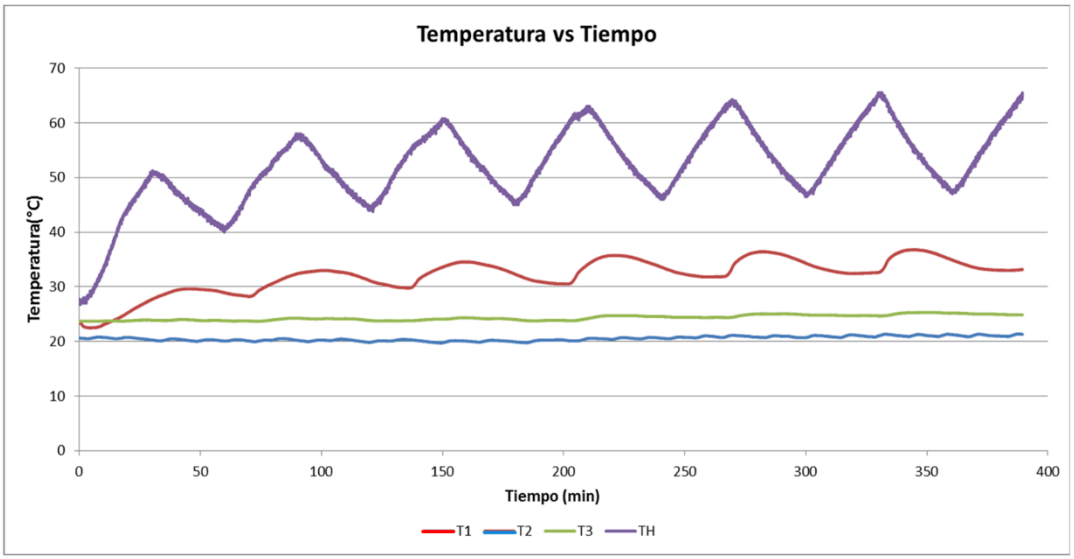


Figura 24. Grafica de temperaturas vs. tiempo en el tubo de calor en modo periódico. T1 es la temperatura de salida del aire, T2 es la temperatura de ambiente, T3 es la temperatura de entrada del aire y TH es la temperatura del condensador de cobre.

5.2 Respuesta de un tubo de calor, con radiación artificial, para diferentes flujos de aire de entrada.

En esta prueba se utilizó el mismo montaje del tubo de calor, mostrado en la Figura 23, pero al condensador se le adosó un disipador anular de aluminio, cuya forma y dimensiones se pueden observar en la Figura 25. La interfaz remota se programó en modo continuo 110 minutos. Se aplicaron tres diferentes flujos de entrada de aire: 10 l/min (líneas verdes), 14 l/min (líneas rojas) y 25 l/min (líneas negras). Para menor flujo de aire, tanto el disipador como el aire aumentan su temperatura [18].

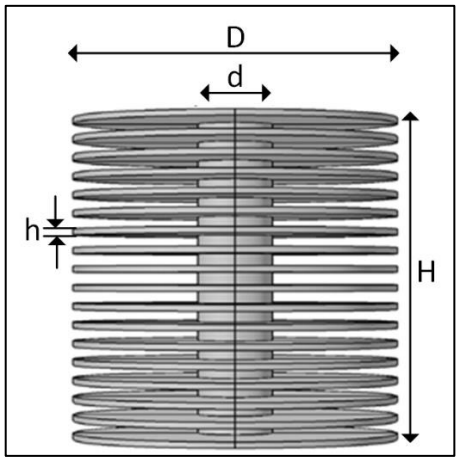


Figura 25. Disipador de aluminio tipo anular. Sus dimensiones son: h =2 mm, H = 105 mm, d = 24 mm y D =105 mm.

En la Figura 26 muestran las temperaturas del disipador anular (TH) y las temperaturas del aire calentado por el disipador (T3).

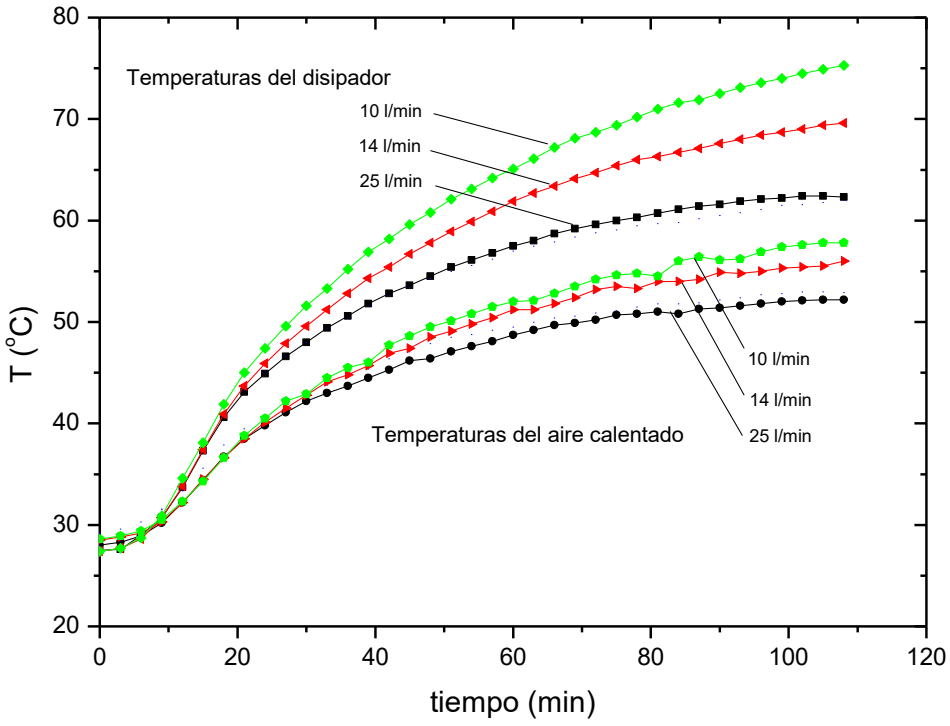


Figura 26. Gráfico temperaturas del disipador tipo aleta (TH) y las temperaturas del aire calentado (T3) vs. Tiempo para diferentes flujos de aire.

5.3 Monitoreo de temperaturas de un colector solar de 12 tubos de calor para generación de aire caliente.

En esta parte se utilizó un prototipo que fue construido como parte de una investigación en sistemas de generación de aire caliente solar (Facultad de Ciencias-UNI). El prototipo consta de 12 tubos de calor, cada uno con un disipador anular de aluminio, como el que se muestra en la Figura 27. Los disipadores se encierran en una caja aislante. Por el lado izquierdo se tiene una abertura circular donde se coloca un ventilador (cuyas características se muestran en el apéndice G) alimentado por una fuente de voltaje [19].

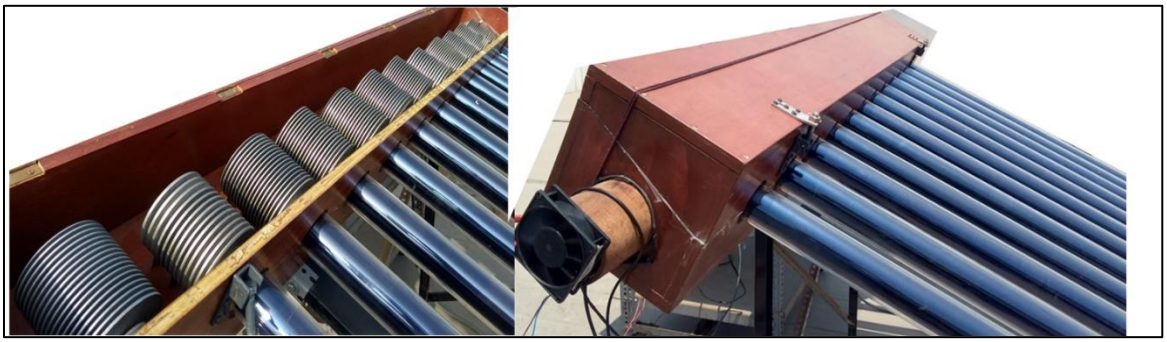


Figura 27. Disposición de los disipadores anulares de aluminio y el ventilador.

La disposición del equipo fue de tal forma que los tubos de calor formen un ángulo de inclinación (entre 12° y 30°) mirando hacia el Norte ya que es la forma más eficiente de captar la radiación solar para nuestra latitud sur.

Como se observa en el esquema de la Figura 28, se mide la temperatura T3 de la entrada del aire al que se le conecto una fuente de alimentación variable con el fin de cambiar el flujo del aire. El aire fluye a través de todos los disipadores y sale con una temperatura T1 (temperatura de salida). T2 es la temperatura del ambiente y TH, la temperatura superficial del disipador del tubo de calor No. 11.

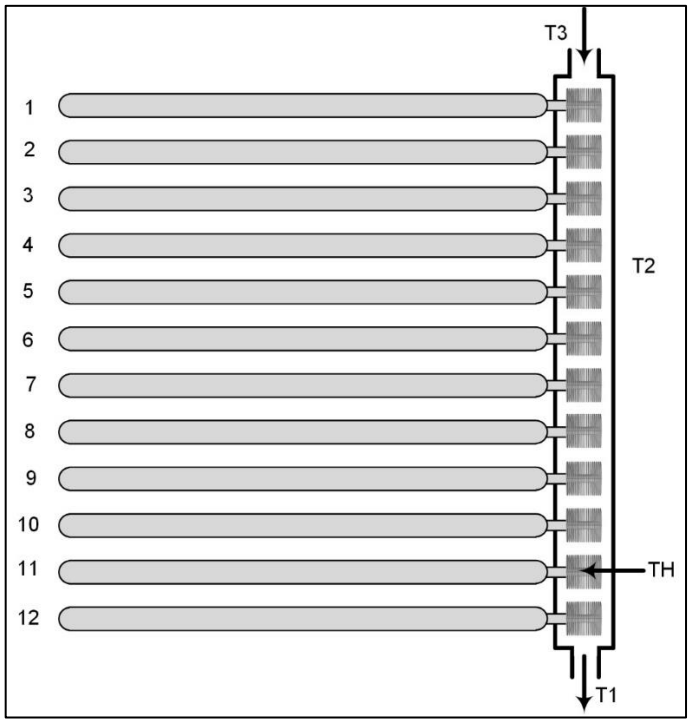


Figura 28. Colector solar de 12 tubos de calor con disipadores.

En el proceso de la adquisición de datos se utiliza la Interfaz remota con la finalidad de obtener datos de manera automática cada cierto tiempo, estos datos se obtuvieron usando sensores de temperatura DS18B20 para las temperaturas T1 (salida), T2 (ambiente) y T3 (salida). Se utilizó una termocupla tipo K para medir la temperatura TH en la superficie del disipador número 11. La Interfaz remota almacena todos los datos en una memoria microSD (Arduino) y en la carpeta “Datos” (Raspberry), después terminar la toma de datos, la memoria microSD puede ser extraída para así poder graficar y analizar los datos obtenidos si se hubiera perdido la conexión de INTERNET, en caso contrario se puede graficar y analizar usando los datos almacenado en la placa Raspberry.

Utilizando la Interfaz remota en modo Luz Solar se empezó la toma de datos el día 19/03/2019 a las 10:49 h hasta el día 21/03/2019 a las 11:55h. Para este caso se alimentó al ventilador con un voltaje de 11V que corresponde a un flujo de aire $\dot{V}$ = 0.0165 m^3/s. Las temperaturas medidas en función del tiempo se muestran en la Figura 29. En la misma Figura 29, se muestra la irradiancia solar.

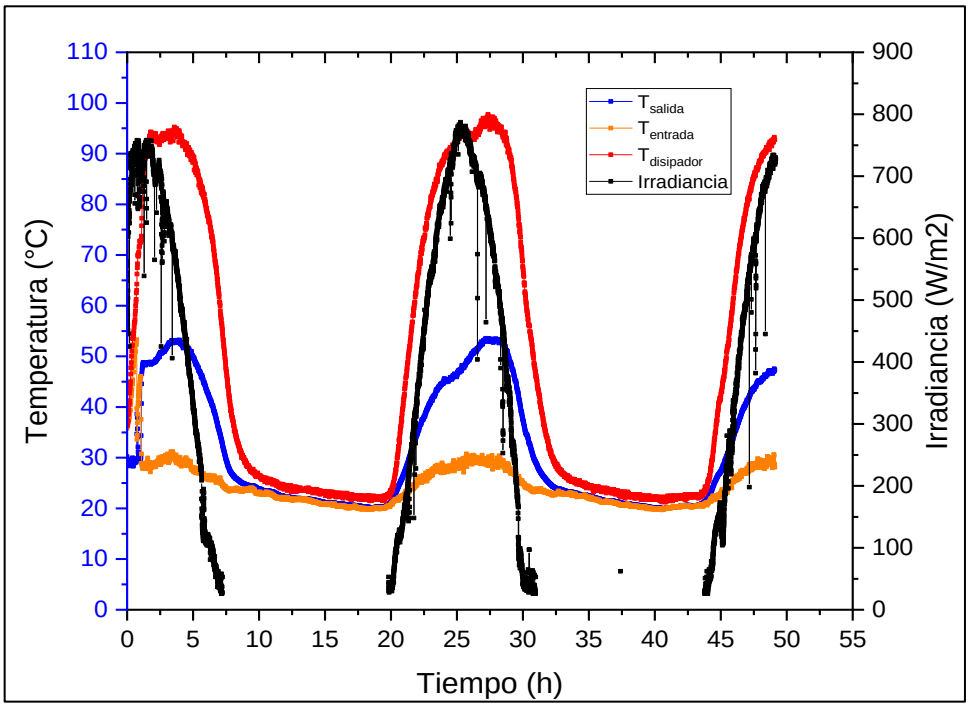


Figura 29. Temperaturas de entrada y salida del aire, $T_{entrada}$ y T_{salida}. $T_{disipador}$ es la temperatura del disipador de aluminio conectado al tubo número 11. Los puntos de color negro, corresponden a la irradiancia solar.

5.3.1 Cálculo de la eficiencia energética, η_E de un colector solar.

Los datos obtenidos con la Interfaz remota, permite evaluar la eficiencia energética del colector. Como un ejemplo tomaremos los datos presentados en la Figura 29 para evaluar la eficiencia del colector.

Para ello se define la eficiencia η_E como la relación entre la energía térmica ganada por el flujo de aire caliente y la energía solar irradiada sobre el colector durante todo un día, esto es,

$$\eta_E = \frac{Energia\ termica\ ganada\ (Eg)}{Energia\ solar\ recibida\ (Es)}. \qquad (5.1)$$

La energía térmica ganada E_g por el aire caliente se calcula a partir de la ecuación,

$$E_g = m \bullet ce \bullet \Delta T\ , \qquad (5.2)$$

derivando con respecto al tiempo entonces obtenemos,

$$\dot{E}_g = \dot{m} \bullet ce \bullet \Delta T\ , \qquad (5.3)$$

para finalmente obtener,

$$\dot{E}_g = \rho \bullet ce \bullet \Delta T \bullet \dot{V} \qquad (5.4)$$

donde,

m: masa del fluido (aire)

ρ : densidad del aire

$\dot{V}$: flujo del aire (Φ)

ce: calor especifico del aire

ΔT: variación de temperatura

Como el fluido es aire, entonces tenemos los siguientes datos: ρ = 1.23 kg/m^3, ce = 1010 J/kg.K. Teniendo en cuenta que la variación de temperatura ΔT es $\Delta T = T_{salida} - T_{entrada}$. Luego reemplazando estos datos en la ecuación (5.4) obtenemos la gráfica de $\dot{E}_g$ representada en color negro en la Figura 30.

En términos prácticos, el valor de la energía del aire caliente E_g se obtiene del área de la curva de color negro en la Figura 30, en el periodo escogido,

$$E_g = \int_{t_0}^{t} \dot{E}_g \bullet dt \qquad (5.5)$$

La energía solar recibida $\dot{E}_s$ se obtiene a partir de la irradiancia I por,

$$\dot{E}_s = I \bullet A\,, \qquad (5.6)$$

donde:

I: Irradiancia (W/m^2)

A: Área de absorción del tubo Heat Pipe (m^2).

Como el diámetro del tubo Heat Pipe es 5.5 cm, la longitud del tubo es 172 cm, entonces el área de absorción por tubo es: $0.0946 m^2$. Por lo tanto, el área de absorción para los 12 tubos es A = 1.1352 m^2.

A partir de los datos de irradiancia mostrados en la Figura 29, y conociendo el área del colector, se obtiene la curva de energía mostrada en color rojo en la Figura 30.

Para calcular la energía radiante solar, de calcula el área bajo la curva $\dot{E}_s$ de la Figura 30 [20],

$$E_s = \int_{t_0}^{t} \dot{E}_s \bullet dt\,. \qquad (5.7)$$

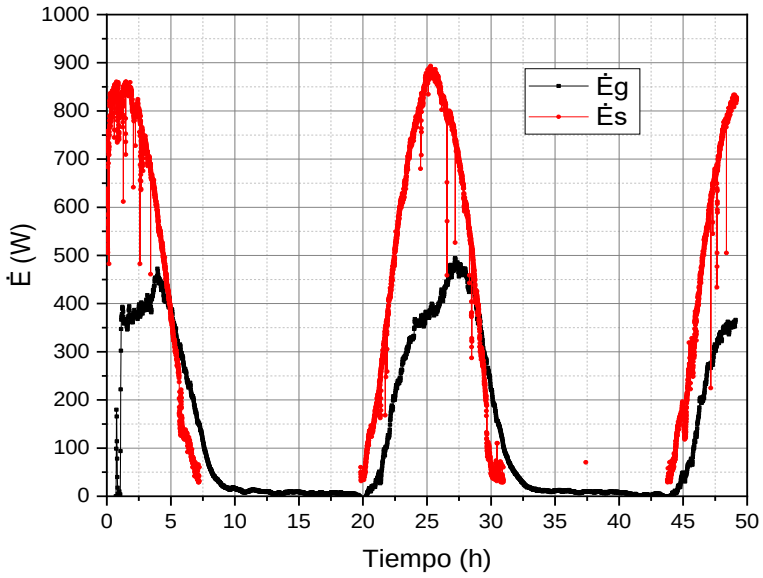


Figura 30. Graficas del comportamiento de la energía por unidad de tiempo vs tiempo.

Para el cálculo de la eficiencia del colector se eligió el intervalo de 12 h entre las 20h y 32h. Para este intervalo de tiempo se obtienen los valores, E_s = 5.531kWh y E_g = 3.309 kWh. Reemplazando los datos en la ecuación (5.1) la eficiencia del sistema es: η_E = 59.82%.

En la Figura 31 se observa el colector de 12 tubos HP usado como un ejemplo de aplicación del uso de la Interfaz remota.

Figura 31. Disposición del equipo para la toma de datos. Se observa la posición del radiómetro, la Interfaz remota Heat Pipe y la salida de aire caliente.

CAPÍTULO 6. CONCLUSIONES Y RECOMENDACIONES

En esta parte se dan las conclusiones y recomendaciones para trabajos futuros en el mejoramiento y la utilización de la Interfaz remota.

CONCLUSIONES

- Se diseñó y construyó una interfaz con la capacidad de leer los datos de cualquier equipo mediante módulos de sensores y una plataforma electrónica Arduino. Además la interfaz puede enviar los datos mediante una microcomputadora Raspberry pi 3 B+ para la adquisición y procesamiento remoto en cualquier computadora conectada a INTERNET.

- La interfaz construida se aplicó en la evaluación de diferentes configuraciones que utilizan tubos de calor,

  (a) Respuesta de un tubo de calor, con radiación artificial intermitente.
  (b) Respuesta de un tubo de calor, con radiación artificial, para diferentes flujos de aire de entrada.
  (c) Monitoreo de temperaturas de un colector solar de 12 tubos de calor para generación de aire caliente.

RECOMENDACIONES

- Desarrollar un programa en lenguaje C++ para la adquisición de datos mediante la tarjeta Arduino y en lenguaje Python para la microcomputadora Raspberry pi 3 B+ para el monitoreo remoto.

- Para las mediciones de radiación solar se puede conectar en el futuro un radiómetro a la Interfaz remota Heat Pipe mediante el módulo ADS1115 (convertidor analógico digital ADC de 16 bits) a través del protocolo I2C.

- En vez de tener dos fuentes de alimentación de corriente continua de 9V (Arduino) y 5V (Raspberry), se puede tener una sola fuente de 9V (Arduino) y utilizar el convertidor de voltaje DC-DC Step-Down 3A LM2596 para regular a 5V (raspberry). También se puede adicionar otro regulador LM2596 para una conexión externa de batería 12V alimentada con paneles solares.

- Se puede cambiar a una pantalla LCD Gráfico 128x64 (funciona con el controlador interno ST7920), esta pantalla puede mostrar tanto gráficos como texto permitiendo diseñar interfaces hombre-máquina (HMI) de mayor complejidad.

- Conectando al puerto USB de la placa Raspberry una cámara (webcam) y ejecutando el programa de video VLC Player, podemos ver remotamente el funcionamiento del equipo y la Interfaz remota Heat Pipe desde cualquier parte del mundo en tiempo real.

- El siguiente paso en el avance de la Instrumentación Científica es desarrollar la aplicación de la Inteligencia Artificial en los Grupos de Investigación en Física. La mayoría de las aplicaciones de la Inteligencia Artificial en física caen en tres categorías principales: análisis de datos, modelado y análisis de modelos.
    - Análisis de los datos es la aplicación más conocida de aprendizaje automático (machine learning). Las redes neuronales se pueden entrenar para reconocer patrones específicos y también pueden aprender a encontrar nuevos patrones por su cuenta.
    - El aprendizaje automático (machine learning) ayuda a modelar sistemas físicos tanto al acelerar los cálculos como al habilitar nuevos tipos de cálculos.
    - Para análisis de modelo, el aprendizaje automático (machine learning) se aplica para comprender mejor las propiedades de las teorías ya conocidas que no pueden extraerse mediante otros métodos matemáticos, o para acelerar la computación.

- Para preparar la Interfaz remota Heat Pipe como una interfaz inteligente se debe cambiar la placa Raspberry pi 3 B+ a Raspberry pi 4 B que tiene grandes recursos, agregarle un

disco duro SSD (estado sólido) de 480 GB, módulo de cámara HD 4K, parlante bluetooth y desarrollar mediante Python (TensorFlow y Machine Learning) la programación de inteligencia artificial.

- La interfaz remota es una muestra del desarrollo tecnológico nacional y de la investigación en el área de la Instrumentación Científica. Espero que esta tesis sea de inspiración para futuras tesis en esta área.

REFERENCIAS

[1] Wang, T., Diao, Y., Zhu, T., Zhao, Y., Liu, J. y Wei, X. (2017). Thermal performance of solar air collection-storage system with phase change material based on flat micro-heat pipe arrays. Energy Conversion and Management, (142), 230–243.

[2] Hayek, M., Assaf, J. y Lteif, W. (2011), Experimental Investigation of the Performance of Evacuated-Tube Solar Collectors under Eastern Mediterranean Climatic Conditions, Energy Procedia Review, (6), 618-626.

[3] Tiwari, G., Tiwari, A. y Shyam (2016). Handbook of Solar Energy. Theory, Analysis and Applications. Singapore: Springer Science+Business Media, 317-322.

[4] Sabiha, M., Saidur, R., Mekhilef, S. y Mahian, O. (2015). Progress and latest developments of evacuated tube solar collectors. Renewable and Sustainable Energy Reviews, (51), 1038–1054.

[5] Tang, R., Yang, Y. y Gao, W. (2011). Comparative studies on thermal performance ofwater-in-glass evacuated tube solar water heaters with different collector tilt-angles. Solar Energy, (85), 1381–1389.

[6] Xu, L., Wang,Z., Yuan, G., Li X., y Ruan Y. (2012). A new dynamic test method for thermal performance of all-glass evacuated solar air collectors. Solar Energy, (86), 1222–1231.

[7] Duffie, J., Beckman, W., (2006). Solar engineering of thermal processes, 3rd ed., J. Wiley & Sons.

[8] Gay, W. (2018). Advanced Raspberry Pi. Raspbian Linux and GPIO Integration. Ontario: Apress.

[9] Cameron, N. (2019). Arduino Applied. Comprehensive Projects for Everyday Electronics. Edinburgh: Apress.

[10] Ziemann, V. (2018). A Hands-On Course in Sensors Using the Arduino and Raspberry Pi. Florida, USA: CRC Press Taylor & Francis Group.

[11] Beddows, P. y Mallon, E. (2018). Cave Pearl Data Logger: A Flexible Arduino-Based Logging Platform for Long-Term Monitoring in Harsh Environments. Sensors, (18), 530-556.

[12] Bell, C. (2013). Benning Sensor Networks with Arduino and Raspberry Pi. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co.

[13] Donat, W. (2018). Learn Raspberry Pi. Programming with Python. Palm dale, California: Apress.

[14] Strickland, J. (2018). Raspberry Pi for Arduino Users. Highlands Ranch, Colorado: Apress.

[15] Hart-Davis, G. (2017). Deploying Raspberry Pi in the Classroom. County Durham, U K: Apress.Gay, W. (2017).

[16] Custom Raspberry Pi Interfaces Design and build hardware interfaces for the Raspberry Pi. Ontario: Apress.

[17] Osmer, Ch. (2018). Defending IoT Infrastructures with the Raspberry Pi. Longs, South Carolina: Appress.

[18] Kumar, A., Kumar, P. y Sharma, H. (2014). Experimental Analysis on Solar Air Dryer. International Journal of Engineering, Management & Sciences, 1(12), 2348 –3733.

[19] Shafieian, A., Khiadani, M. y Nosrati, A. (2019). Thermal performance of an evacuated tube heat pipe solar water heating system in cold season. Applied Thermal Engineering, (149), 644–657.

[20] BhagwatV., Patil, V., Bhosale, K. y Kambale, S. (2017). Experimental Analysis of a Solar Air Dryer with Thermal Energy Storage Unit (PCM). International Advanced Research Journal in Science, Engineering and Technology, 4(1), 174-179.

APÉNDICES

A. Pines de la placa Arduino Mega

Tabla 15. Tabla de Mapas de Pines de la placa Arduino Mega. Fuente: Ardubasic (2014), Arduino Mega 2560 revisión 3, https://ardubasic.wordpress.com/2014/05/04/arduino-mega-2560-revision-3/

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (	
5	PE3 (	Digital pin 5 (PWM)
6	PE4 (	Digital pin 2 (PWM)
7	PE5 (	Digital pin 3 (PWM)
8	PE6 (T3/INT6	
9	PE7 (	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (	Digital pin 53 (SS)
20	PB1 (	Digital pin 52 (SCK)
21	PB2 (	Digital pin 51
22	PB3 (	Digital pin 50
23	PB4 (	Digital pin 10 (PWM)
24	PB5 (	Digital pin 11 (PWM)
25	PB6 (	Digital pin 12 (PWM)
26	PB7 (	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (	Digital pin 21 (SCL)
44	PD1 (	Digital pin 20 (SDA)
45	PD2 (	Digital pin 19 (RX1)
46	PD3 (	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38

Pin Number	Pin Name	Mapped Pin Name
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (	Digital pin 15 (RX3)
64	PJ1 (	Digital pin 14 (TX3)
65	PJ2 (	
66	PJ3 (PCINT12	
67	PJ4 (PCINT13	
68	PJ5 (PCINT14	
69	PJ6 (PCINT	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (	Analog pin 15
83	PK6 (	Analog pin 14
84	PK5 (	Analog pin 13
85	PK4 (	Analog pin 12
86	PK3 (	Analog pin 11
87	PK2 (	Analog pin 10
88	PK1 (	Analog pin 9
89	PK0 (	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (	Analog pin 5
93	PF4 (	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

B. Pines de la placa Raspberry pi 3 B+.

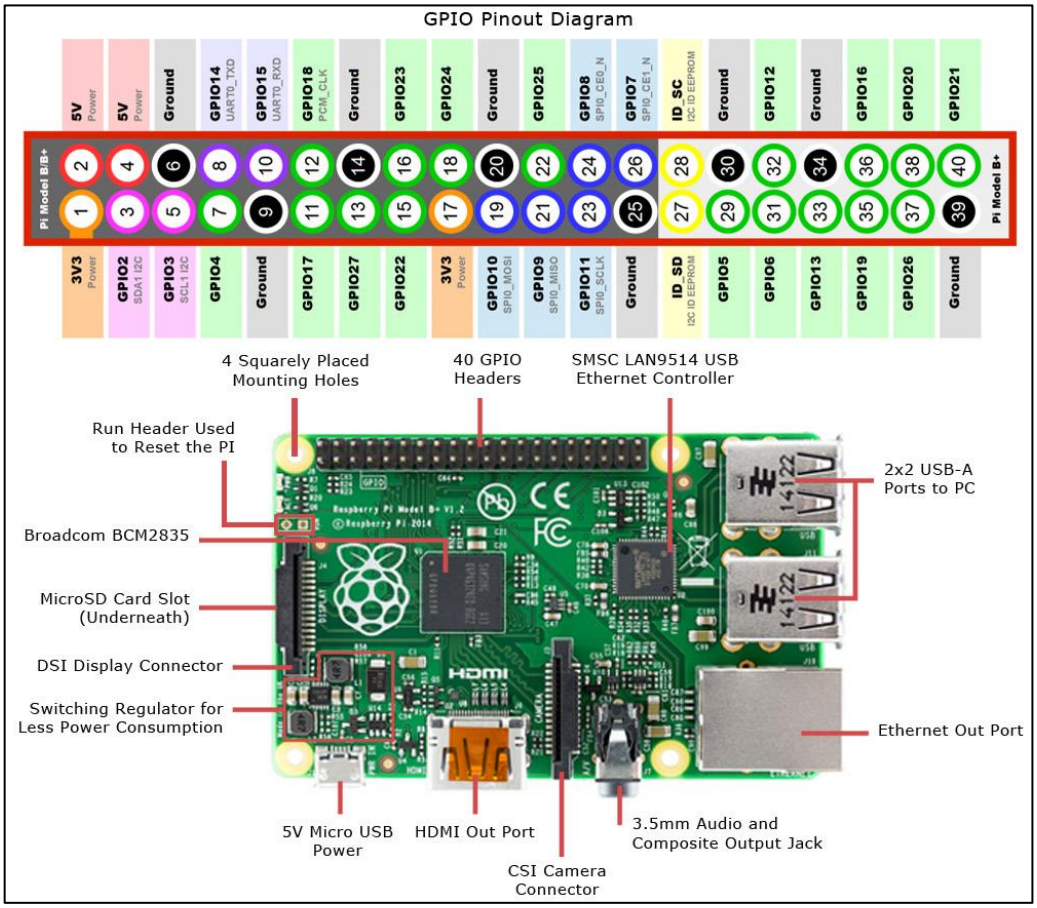


Figura 32. Raspberry pi 3 B+. Fuente: Blog spot Electrónica y ciencia (2016), Conexión GPIO de Raspberry Pi 3, http://electronicayciencia.blogspot.com/2016/11/conexion-gpio-de-raspberry-pi-3.html.

C. Fotografías del sistema de monitoreo remoto Heat Pipe

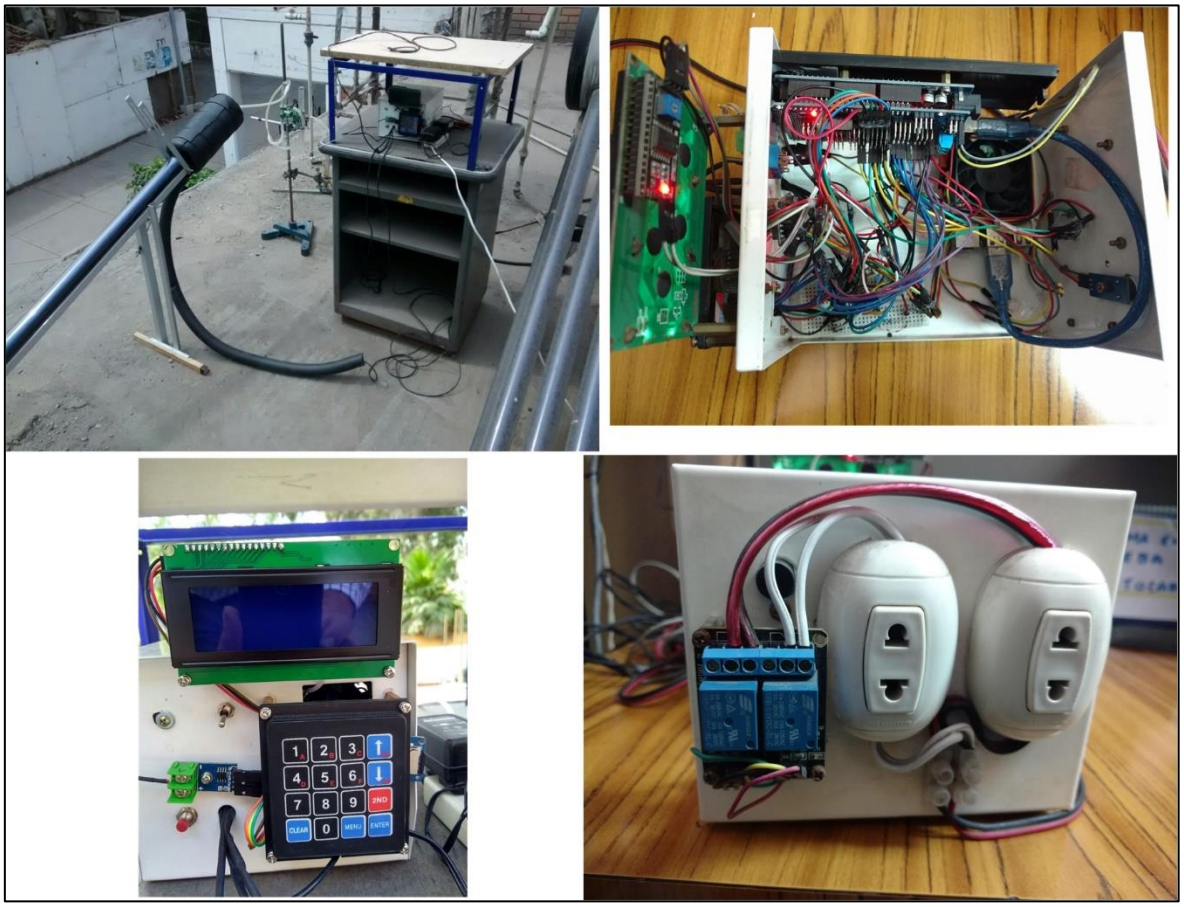


Figura 33. Fotografías del equipo Interfaz remota.

D. Costo Interfaz remota

Los costos se hicieron respecto al mes de diciembre del año 2019.

Tabla 16. Costo Interfaz remota.

DISPOSITIVO	CANTIDAD	COSTO TOTAL (S/.)
Kit termocupla : MAX6675	1	55.00
DS18B20	3	45.00
RELAY	1	20.00
Kit DISPLAY LCD	1	45.00
MicroSD card	1	10.00
RTC DS3231	1	25.00
BUZZER	1	10.00
Teclado 4x4	1	25.00
Raspberry pi 3 B+	1	285.00
Disipadores y ventilador para Raspberry pi	1	35.00
Memoria MicroSD 32 GB clase 10	1	50.00
Arduino Mega	1	65.00
Shield sensores Arduino Mega	1	30.00
Cables Dupont Arduino 40 pines	4	40.00
Fuente raspberry pi	1	30.00
Fuente Arduino	1	20.00
Case lata	1	45.00
Tomacorriente simple oval	2	30.00
Cable arduino USB	1	10.00
Cables mellizos rojo-negro 2m	1	5.00
Interruptor de palanca	2	10.00
Resistencias, condensadores, etc.	1	30.00

TOTAL	920.00

E. Programa Arduino para la Interfaz remota

El programa Arduino usado es **HeatPipe9.ino**:

```
#include "Wire.h"
#define DS3231_I2C_ADDRESS 0x68
#include <SD.h>
#include <SPI.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>
#include "max6675.h"
Sd2Card card;
SdVolume volume;
SdFile root;
int speakerPin = 13;
int ktcSO = 8;
int ktcCS = 9;
int ktcCLK = 10;
MAX6675 ktc(ktcCLK, ktcCS, ktcSO);
char filename[] = "HP0000.txt"; // nombre seis caracteres
String archivo1;
File datafile;
#include <OneWire.h>
#include <DallasTemperature.h>
unsigned long  periodo;  // tiempo que esta el relay alto y bajo
int const RelayControl1 = 4;    // Arduino Pin digital usado para controlar el rele1
int const RelayControl2 = 5;    // Arduino Pin digital usado para controlar el rele2
OneWire ourWire1(2);                //Se establece el pin 2  como bus OneWire
OneWire ourWire2(3);                //Se establece el pin 3  como bus OneWire
OneWire ourWire3(6);                //Se establece el pin 6  como bus OneWire
DallasTemperature sensors1(&ourWire1); //Se declara una variable u objeto para nuestro
sensor1
DallasTemperature sensors2(&ourWire2); //Se declara una variable u objeto para nuestro
sensor2
```

```
DallasTemperature sensors3(&ourWire3); //Se declara una variable u objeto para nuestro sensor2
char pulsacion;
char pulsacion2;
String intervalo;
long semiperiodo;
float tiempo;
int ciclo1;
int segundo;
int segundo1;
int minuto;
int minuto1;
int hora;
int hora1;
int dia;
int dia1;
String numero[4];
const byte Filas = 4;     //Cuatro filas
const byte Cols = 4;    //Cuatro columnas
byte flechaArriba[8] =  {B00000, B00100, B01110, B11111,
                         B00100, B00100, B00100, B00000
                        };  // Flecha Arriba
byte flechaAbajo[8] =   {B00000, B00100, B00100, B00100,
                         B11111, B01110, B00100, B00000
                        };  // Flecha Abajo
byte Pins_Filas[] = {42, 43, 44, 45};  //Pines Arduino a los que contamos las filas.
byte Pins_Cols[] = { 46, 47, 48, 49};   // Pines Arduino a los que contamos las columnas.
char Teclas [ Filas ][ Cols ] =
{
  {'1', '2', '3', 'U'},
  {'4', '5', '6', 'D'},
  {'7', '8', '9', 'B'},
  {'C', '0', 'M', 'E'}
```

```
};
Keypad Teclado1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);
// Se establece la dirección LCD en 0x3F para una pantalla de 16 caracteres y 2 líneas
LiquidCrystal_I2C lcd(0x3F, 20, 4);
// Se incializa la pantalla LCD
// se enciende la lampara e imprime un mensaje.
int relayState = LOW;             // relayState usado para inicializar el rele
unsigned long previousMillis = 0;
unsigned long currentMillis;
unsigned long tiempo1;
unsigned long tiempo2;
long intervalOn;          // tiempo en segundos ON
long intervalOff;         // tiempo en segundos OFF
// Convierte números decimales a decimales codificados binarios
byte decToBcd(byte val)
{
  return( (val/10*16) + (val%10) );
}
// Convierte decimal codificado en binario a números decimales
byte bcdToDec(byte val)
{
  return( (val/16*10) + (val%16) );
}
void setup()
{
  Wire.begin();
  Serial.begin(9600);
  // establezca la hora inicial aquí:
  // DS3231 segundos, minutos, horas, día, fecha, mes, año
  //setDS3231time(00,16,11,7,4,2,18);
  if (!SD.begin(53)) {    //arduino Mega pin 53, UNO pin 4
    Serial.println("No se pudo inicializar");
    lcd.print("No se pudo inicializar");
```

```
  return;
}
sensors1.begin();   //Se inicia el sensor 1
sensors2.begin();   //Se inicia el sensor 2
sensors3.begin();   //Se inicia el sensor 3
pinMode(RelayControl1, OUTPUT);
digitalWrite(RelayControl1, HIGH); // NO1 y COM1 desconectados (LED apagado)
lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
lcd.print("RELAY1 OFF ");
pinMode(RelayControl2, OUTPUT);
digitalWrite(RelayControl2, HIGH); // NO2 y COM2 desconectados (LED apagado)
pinMode (speakerPin, OUTPUT);
lcd.begin();
lcd.backlight();
lcd.createChar(1, flechaArriba);   // Caracter personalizado "Flecha arriba"
lcd.createChar(2, flechaAbajo);    // Caracter personalizado "Flecha abajo"
lcd.setCursor ( 0, 0 );
lcd.print("Heat Pipe Smart V1.1");
Serial.println("HPS");
lcd.setCursor ( 0, 1 );
lcd.print("Ciro Carhuancho");
delay(4000);
lcd.clear();
lcd.setCursor ( 0, 0);//cursor en columna 1 y fila 1
lcd.print("Elegir una opcion:");
Serial.println("Elegir");
lcd.setCursor ( 0, 1);//cursor en columna 1 y fila 2
lcd.write(1); // imprime flecha arriba
lcd.print(" Luz artificial");
lcd.setCursor ( 0, 2);//cursor en columna 1 y fila 3
lcd.write(2); // imprime flecha abajo
lcd.print(" Luz solar");
pinMode (speakerPin, OUTPUT);
```

```
  analogWrite (speakerPin, 255);
}
void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte
dayOfMonth, byte month, byte year)
{
  // sets time and date data to DS3231
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // establecer la siguiente entrada para comenzar en el registro de segundos
  Wire.write(decToBcd(second)); // establecer segundos
  Wire.write(decToBcd(minute)); // establecer minutos
  Wire.write(decToBcd(hour)); // establecer horas
  Wire.write(decToBcd(dayOfWeek)); // establecer el día de la semana (1 = domingo, 7 =
sábado)
  Wire.write(decToBcd(dayOfMonth)); // fijar fecha (1 a 31)
  Wire.write(decToBcd(month)); // establecer mes
  Wire.write(decToBcd(year)); // año establecido (0 a 99)
  Wire.endTransmission();
}
void readDS3231time(byte *second,
byte *minute,
byte *hour,
byte *dayOfWeek,
byte *dayOfMonth,
byte *month,
byte *year)
{
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // establecer el puntero de registro DS3231 a 00h
  Wire.endTransmission();
  Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
  // solicita siete bytes de datos desde DS3231 a partir del registro 00h
  *second = bcdToDec(Wire.read() & 0x7f);
  *minute = bcdToDec(Wire.read());
```

```
  *hour = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month = bcdToDec(Wire.read());
  *year = bcdToDec(Wire.read());
}
void displayTime()
{
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  // recuperar datos de DS3231
  readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,
  &year);
  // envía al monitor serial
  Serial.print(hour, DEC);
  hora=hour;
  datafile.print(hour, DEC);
  // convertir la variable de byte a un número decimal cuando se muestra
  Serial.print(":");
  datafile.print(":");
  if (minute<10)
  {
    Serial.print("0");
    datafile.print("0");
  }
  Serial.print(minute, DEC);
  minuto=minute;
  datafile.print(minute, DEC);
  Serial.print(":");
  datafile.print(":");
  if (second<10)
  {
    Serial.print("0");
    datafile.print("0");
```

```
  }
  Serial.print(second, DEC);
  segundo = second;
  datafile.print(second, DEC);
  Serial.print(" ");
  datafile.print(" ");
  Serial.print(dayOfMonth, DEC);
  dia=dayOfMonth;
  datafile.print(dayOfMonth, DEC);
  Serial.print("/");
  datafile.print("/");
  Serial.print(month, DEC);
  datafile.print(month, DEC);
  Serial.print("/");
  datafile.print("/");
  Serial.println(year, DEC);
  datafile.println(year, DEC);
  sprintf(filename, "HP%02d%02d.txt", hour, minute );
}
void loop()
{
   tecladatos();
       if (pulsacion == 'D') {
    lcd.clear();
    lcd.setCursor ( 0, 0);//cursor en columna 1 y fila 1
    lcd.print("LUZ SOLAR 7 dias");
        displayTime();
        dia1 = dia;
        hora1= hora;
        minuto1 = minuto;
        segundo1 = segundo;
        archivo1 = filename;
        Serial.println("SOLAR");
```

```
  datafile = SD.open(archivo1, FILE_WRITE);//abrimos  el archivo
   if (datafile) {
       displayTime();
       tiempo1 = millis();
      datafile.println("SOLAR");
      Serial.println("INICIO1");
      datafile.println("T1,T2,T3,TH,tiempo");
      Serial.println("T1(°C) , T2(°C), T3(°C) , TH(°C),tiempo(s)");
      } else {
            Serial.println("Error al abrir el archivo");
           }
  datafile.close();
          while(tiempo<10079) { //tiempo de toma de datos (7 dias=10080 restarle 1)
      // displayTime();
     tiempo2=millis();
    datafile = SD.open(archivo1, FILE_WRITE);//abrimos  el archivo
    if (datafile) {
               tomadedatos();
            } else {
             Serial.println("Error creando el archivo datafile.txt");
             lcd.print("Error archivo");
             }
       datafile.close();
       delay(59400); //retardo 1 minuto
      }
       lcd.clear();
       lcd.setCursor ( 0, 2);//cursor en columna 1 y fila 3
       lcd.print("Fin toma de datos");
       Serial.println("FIN");
}
 if (pulsacion == 'U') {
 lcd.clear();
```

```
  lcd.setCursor ( 0, 0);//cursor en columna 1 y fila 1
  lcd.print(" Luz artificial");
  Serial.println("SIMULACION");
  lcd.setCursor ( 0, 1);//cursor en columna 1 y fila 2
  lcd.print("C Periodico");
  lcd.setCursor ( 0, 2);//cursor en columna 1 y fila 3
  lcd.print("M Continuo");
  lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
  lcd.print("                    ");
  lcd.setCursor ( 0, 3);//cursor en columna 2 y fila 4
 }
 if (pulsacion == 'C') {
  tecladatos();
  lcd.clear();
  lcd.setCursor ( 0, 0);//cursor en columna 1 y fila 1
  lcd.print("Ingrese semiperiodo");
  lcd.setCursor ( 0, 1);//cursor en columna 1 y fila 2
  lcd.print("1 1h   2 2h");
  lcd.setCursor ( 0, 2);//cursor en columna 1 y fila 3
  lcd.print("3 3h   4 4h");
  tecladatos();
 }
 if (pulsacion == '1') {
  semiperiodo = 3600000; //60000x60ms=1h
  intervalOn = semiperiodo;         // tiempo en segundos ON
  intervalOff = semiperiodo;       // tiempo en segundos OFF
  intervalo = "1h";
  lcd.setCursor ( 3, 3);//cursor en columna 2 y fila 4
  lcd.print(intervalo);
 }
 if (pulsacion == '2') {
  semiperiodo = 7200000; //2h
  intervalOn = semiperiodo;         // tiempo en segundos ON
```

```
  intervalOff = semiperiodo;       // tiempo en segundos OFF
  intervalo = "2h";
  lcd.setCursor ( 3, 3);//cursor en columna 2 y fila 4
  lcd.print(intervalo);
 }
 if (pulsacion == '3') {
  semiperiodo = 10800000; //3h
  intervalOn = semiperiodo;         // tiempo en segundos ON
  intervalOff = semiperiodo;       // tiempo en segundos OFF
  intervalo = "3h";
  lcd.setCursor ( 3, 3);//cursor en columna 2 y fila 4
  lcd.print(intervalo);
 }
 if (pulsacion == '4') {
  semiperiodo = 14400000; //4h
  intervalOn = semiperiodo;         // tiempo en segundos ON
  intervalOff = semiperiodo;       // tiempo en segundos OFF
  intervalo = "4h";
  lcd.setCursor ( 3, 3);//cursor en columna 2 y fila 4
  lcd.print(intervalo);
 }
  switch (pulsacion) {
    tecladatos();
  case 'E':
    lcd.clear();
    beep1();
    lcd.setCursor ( 0, 0);//cursor en columna 1 y fila 1
    lcd.print("Inicio toma de datos");
    relayState == HIGH;
    digitalWrite(RelayControl1, relayState); // NO1 y COM1 desconectados (LED
apagado)
    lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
    lcd.print("RELAY1 OFF");
```

```
// Obtener fecha actual y mostrar por Serial
  displayTime();
  archivo1 = filename;
datafile = SD.open(archivo1, FILE_WRITE);//abrimos  el archivo
 if (datafile) {
     displayTime();
     datafile.println("SIMULACION");
     Serial.println("INICIO2");
     datafile.println("relaystate,T1,T2,T3,TH,tiempo");
     Serial.println("relaystate , T1(°C) , T2(°C), T3(°C) , TH(°C),tiempo(s)");
     } else {
           Serial.println("Error al abrir el archivo");
          }
datafile.close();
relayState == HIGH;
digitalWrite(RelayControl1, relayState); // NO1 and COM1 disconnected (LED off)
lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
lcd.print("RELAY1 OFF");
ciclo1 = 0;
tiempo = 0;
tiempo1 = millis();
while (ciclo1 <= 6) {
  currentMillis = millis();
  if (relayState == LOW) {
    if (currentMillis - previousMillis > intervalOff) {
      previousMillis = currentMillis;
      relayState = HIGH;
      ciclo1 ++;
      lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
      lcd.print("RELAY1 OFF");
    }
  } else {
    if (currentMillis - previousMillis > intervalOn) {
```

```
      previousMillis = currentMillis;
      relayState = LOW;
      lcd.setCursor ( 0, 3);//cursor en columan 1 y fila 3
      lcd.print("RELAY1 ON ");
     }
    }
    digitalWrite(RelayControl1, relayState); // NO1 and COM1 Conectado (LED on)
    tiempo2 = millis();
    datafile = SD.open(archivo1, FILE_WRITE);//abrimos  el archivo
    if (datafile) {
                Serial.print(relayState);
                datafile.print(relayState);
                Serial.print(",");
                datafile.print(",");
                tomadedatos();
              } else {
               Serial.println("Error creando el archivo datafile.txt");
               lcd.print("Error archivo");
               }
      datafile.close();
  }
  lcd.clear();
  lcd.setCursor ( 0, 0);//cursor en columna 1 y fila 1
  lcd.print("Fin toma de datos");
  Serial.println("FIN");
  digitalWrite(RelayControl1, HIGH); // NO1 and COM1 disconnected (LED off)
  lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
  lcd.print("RELAY1 OFF ");
  break;
 case 'M':
  lcd.clear();
  relayState = HIGH;
  digitalWrite(RelayControl1, relayState); // Relay OFF
```

```
lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
lcd.print("RELAY1 OFF");
  displayTime();
dia1 = dia;
hora1= hora;
minuto1 = minuto;
segundo1 = segundo;
archivo1 = filename;
datafile = SD.open(archivo1, FILE_WRITE);//abrimos  el archivo
 if (datafile) {
      displayTime();
      Serial.println("INICIO");
     datafile.println("relaystate,T1,T2,T3,TH,tiempo");
     Serial.println("relaystate , T1(°C) , T2(°C), T3(°C) , TH(°C),tiempo(s)");
      } else {
           Serial.println("Error al abrir el archivo");
          }
    datafile.close();
    tiempo1=millis();
         while(tiempo<5) {
      relayState = LOW;
      digitalWrite(RelayControl1, relayState); // Relay ON
      lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
      lcd.print("RELAY1 ON ");
      tiempo2=millis();
   datafile = SD.open(archivo1, FILE_WRITE);//abrimos  el archivo
  if (datafile) {
              Serial.print(relayState);
              datafile.print(relayState);
              Serial.print(",");
              datafile.print(",");
              tomadedatos();
           } else {
```

```
            Serial.println("Error creando el archivo datafile.txt");
            lcd.print("Error archivo");
            }
          datafile.close();
        }
          lcd.clear();
          relayState = HIGH;
          digitalWrite(RelayControl1, relayState); // Relay OFF
          lcd.setCursor ( 0, 3);//cursor en columna 1 y fila 4
          lcd.print("RELAY1 OFF");
          lcd.setCursor ( 3, 0);//cursor en columna 1 y fila 1
          lcd.print("Fin toma de datos");
          Serial.println("FIN");
     break;
   case 'B':
     lcd.setCursor ( 0, 3);//cursor en columna 2 y fila 4
     lcd.print(pulsacion2);
     break;
   }
}
void tomadedatos() {
  lcd.setCursor ( 0, 1);//cursor en columan 1 y fila 1
  sensors1.requestTemperatures();   //Se envía el comando para leer la temperatura
  float temp1 = sensors1.getTempCByIndex(0); //Se obtiene la temperatura en ºC del
sensor 1
  Serial.print(temp1); //dato T1
  datafile.print(temp1);
  lcd.print("T1=");
  lcd.print(temp1);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor ( 10, 1);//cursor en columan 1 y fila 1
  sensors2.requestTemperatures();   //Se envía el comando para leer la temperatura
```

```
  float temp2 = sensors2.getTempCByIndex(0); //Se obtiene la temperatura en ºC del
sensor 2
  Serial.print(",");
  datafile.print(",");
  Serial.print(temp2); // dato T2
  datafile.print(temp2);
  lcd.print("T2=");
  lcd.print(temp2);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor ( 10, 2);//cursor en columan 1 y fila 1
  sensors3.requestTemperatures();   //Se envía el comando para leer la temperatura
  float temp3 = sensors3.getTempCByIndex(0); //Se obtiene la temperatura en ºC del
sensor 3
  Serial.print(",");
  datafile.print(",");
  Serial.print(temp3);  // dato T3
  datafile.print(temp3);
  lcd.print("T3=");
  lcd.print(temp3);
  lcd.print((char)223);
  lcd.print("C");
  Serial.print(",");
  datafile.print(",");
  lcd.setCursor ( 0, 2);//cursor en columan 1 y fila 3
  lcd.print("TH=");
  Serial.print(ktc.readCelsius()); //dato TH
  datafile.print(ktc.readCelsius());
  lcd.print(ktc.readCelsius());
  lcd.print((char)223);
  lcd.print("C");
  tiempo = (tiempo2 - tiempo1) / 60000.0 ;
  Serial.print(",");
```

```
  datafile.print(",");
  lcd.setCursor (10, 3);//cursor en columna 1 y fila 4
  lcd.print(tiempo, 2);
  lcd.print(" min");
  Serial.println(tiempo, 2); // dato t en minutos
  datafile.println(tiempo, 2);
  }
void tecladatos() {
 pulsacion = Teclado1.getKey() ;
 if (pulsacion != 0) {     // Si el valor es 0 es que no se ha pulsado ninguna tecla
 if (pulsacion == '1') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '2') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '3') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '4') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '5') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '6') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '7') {
     pulsacion2 = pulsacion;
   }
   if (pulsacion == '8') {
     pulsacion2 = pulsacion;
```

```
    }
    if (pulsacion == '9') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == '0') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == 'U') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == 'D') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == 'B') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == 'C') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == 'M') {
      pulsacion2 = pulsacion;
    }
    if (pulsacion == 'E') {
      pulsacion2 = pulsacion;
    }
  }
  else {
    char waitForKey();
  }
}
void beep1() {
      analogWrite (speakerPin, 0);
      delay (100);
```

```
    analogWrite (speakerPin, 255);
}
```

Figura 33. Programa HeatPipe9.ino en el entorno Arduino IDE.

F. Programa Python para la Interfaz remota

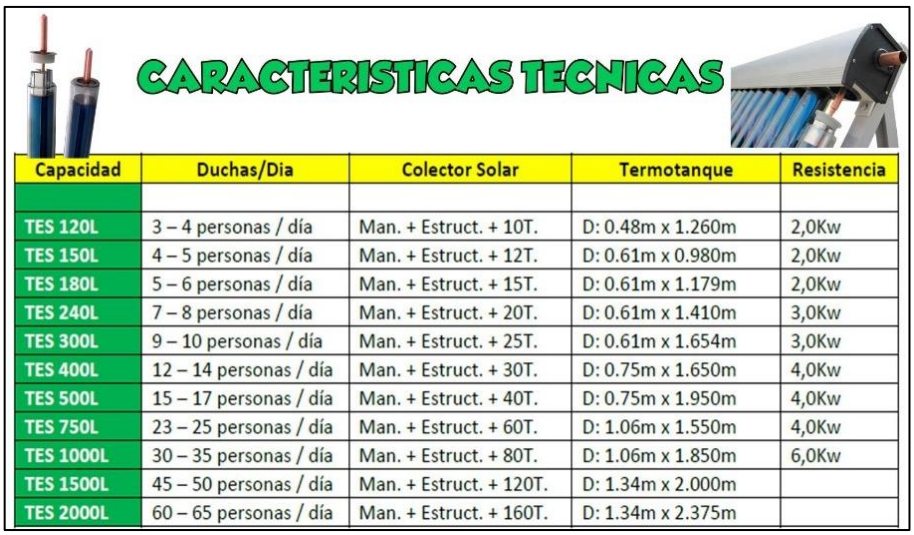
El programa Python usado es **heatpipe2.py**:

```
import serial, time
import os
os.chdir('Datos')
#
arduino = serial.Serial('/dev/ttyUSB0', 9600) # /dev/ ttyACM0
String1 ="INICIO"
String3 = "FIN"
String4 = "SOLAR"
String5 = "SIMULACION"
String7 = "INICIO2"
while True:
   rawString = arduino.readline()
   String2 = rawString.strip() #eliminar retorno de carro y salto de linea
   String2 = String2.decode('utf-8') # quitar 'b'
   if  String4 == String2 :
        print('SOLAR')
        String6 = time.strftime("%d%m%y%I%M%S")
        print(String6)
   if  String1 == String2 :
        print('INICIO')
        while String3 != String2:
           file = open('SOLAR'+String6 +'.txt', 'a')
           rawString = arduino.readline()
           String2 = rawString.strip() #eliminar retorno de carro y salto de linea
           String2 = String2.decode('utf-8') # quitar 'b'
           print(String2)
           file.write(String2+'\n')
   if  String5 == String2 :
        print('SIMULACION')
        String6 = time.strftime("%d%m%y%I%M%S")
```

```
   if  String7 == String2 :
        print('INICIO')
        while String3 != String2:
            file = open('SIMUL'+String6 +'.txt', 'a')
            rawString = arduino.readline()
            String2 = rawString.strip() # eliminar retorno de carro y salto de linea
            String2 = String2.decode('utf-8') # quitar 'b'
            print(String2)
            file.write(String2+'\n')
        print('FIN')
        file.close()
arduino.close()
```

G. Características técnicas de las partes del equipo colector solar.

- La terma solar usada es marca GEO-Energía.



Capacidad	Duchas/Dia	Colector Solar	Termotanque	Resistencia
TES 120L	3 – 4 personas / día	Man. + Estruct. + 10T.	D: 0.48m x 1.260m	2,0Kw
TES 150L	4 – 5 personas / día	Man. + Estruct. + 12T.	D: 0.61m x 0.980m	2,0Kw
TES 180L	5 – 6 personas / día	Man. + Estruct. + 15T.	D: 0.61m x 1.179m	2,0Kw
TES 240L	7 – 8 personas / día	Man. + Estruct. + 20T.	D: 0.61m x 1.410m	3,0Kw
TES 300L	9 – 10 personas / día	Man. + Estruct. + 25T.	D: 0.61m x 1.654m	3,0Kw
TES 400L	12 – 14 personas / día	Man. + Estruct. + 30T.	D: 0.75m x 1.650m	4,0Kw
TES 500L	15 – 17 personas / día	Man. + Estruct. + 40T.	D: 0.75m x 1.950m	4,0Kw
TES 750L	23 – 25 personas / día	Man. + Estruct. + 60T.	D: 1.06m x 1.550m	4,0Kw
TES 1000L	30 – 35 personas / día	Man. + Estruct. + 80T.	D: 1.06m x 1.850m	6,0Kw
TES 1500L	45 – 50 personas / día	Man. + Estruct. + 120T.	D: 1.34m x 2.000m	
TES 2000L	60 – 65 personas / día	Man. + Estruct. + 160T.	D: 1.34m x 2.375m	

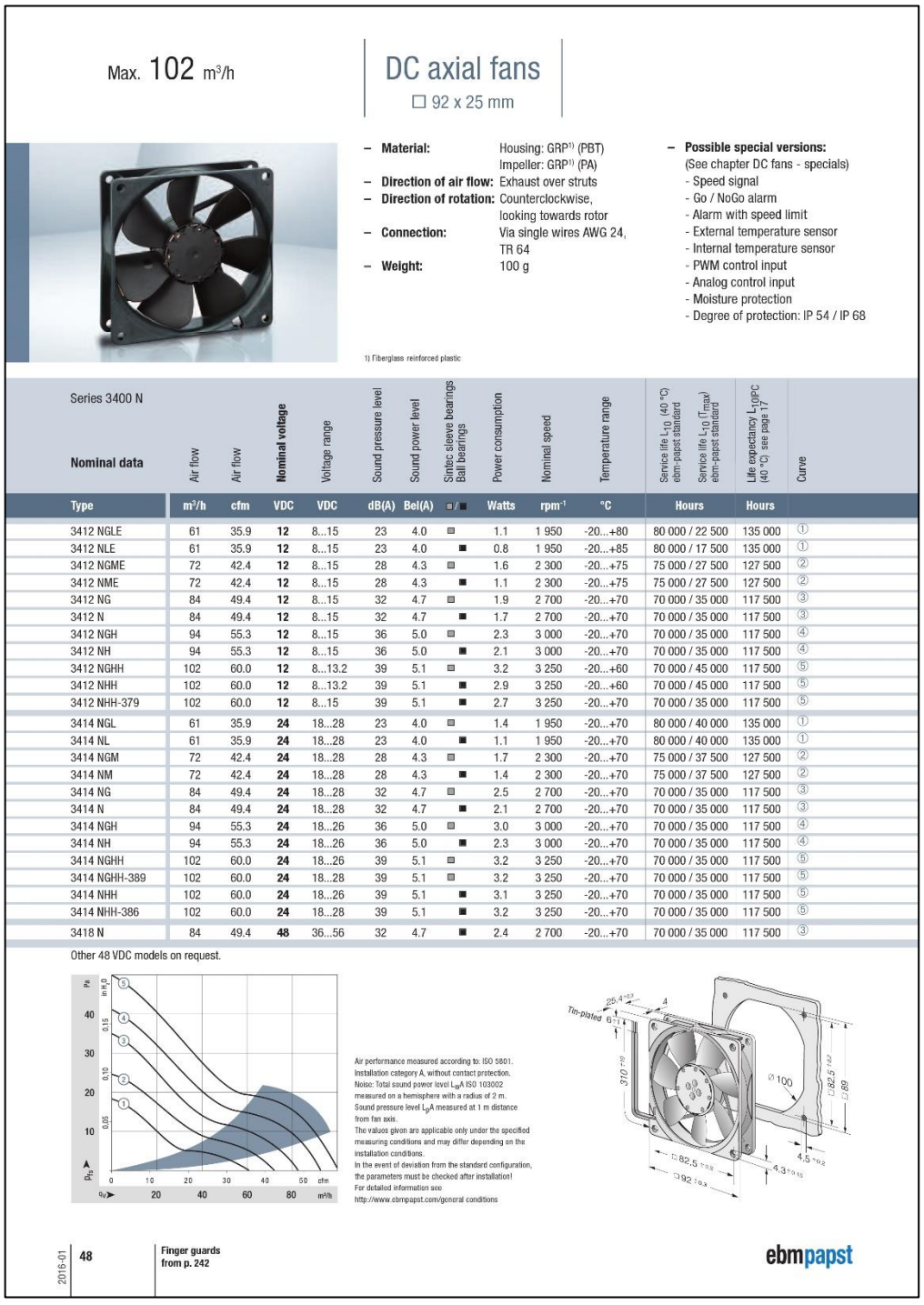
Figura 34. Características de la terma solar. Fuente: GeoEnergía Perú(2020), termas solares, http://geoenergiaperu.com/termas-solares/

- Para la simulación del tubo de calor se utilizaron focos Dicroico.

Tabla 17. Características del foco dicroico. Fuente: Secom Iluminación S.L. (2017), Ficha técnica, http://www.secom.es/descargas/fijo-circular-gu53438_fichatecnica.pdf

Lámpara / Lamp: Halógena Dicroica / Dichroic Halogen
Equipos / Equipment: Equipo Electrónico / Electronic equipment
Watios / Watt: 50W
W/consumo / consumption: 53,7 W
Portalámparas / Lampholder: GU5.3
Vida útil / Life span: 3000 h.
Temperatura de color / Colour temperature: 3000°K
Ángulo de apertura (grados) / Opening angle (degrees): 36°
Grado de protección / Protection degree: IP20
Índice de reprod. crom. IRC / Chromatic reprod. index CRI: 100

- Se utilizo un ventilador Tubeaxial serie 3400.



Max. 102 m³/h

DC axial fans

□ 92 x 25 mm

– **Material:** Housing: GRP[1)] (PBT)
Impeller: GRP[1)] (PA)
– **Direction of air flow:** Exhaust over struts
– **Direction of rotation:** Counterclockwise, looking towards rotor
– **Connection:** Via single wires AWG 24, TR 64
– **Weight:** 100 g

– **Possible special versions:**
(See chapter DC fans - specials)
- Speed signal
- Go / NoGo alarm
- Alarm with speed limit
- External temperature sensor
- Internal temperature sensor
- PWM control input
- Analog control input
- Moisture protection
- Degree of protection: IP 54 / IP 68

1) Fiberglass reinforced plastic

Series 3400 N Nominal data	Air flow	Air flow	Nominal voltage	Voltage range	Sound pressure level	Sound power level	Sintec sleeve bearings Ball bearings	Power consumption	Nominal speed	Temperature range	Service life L_{10} (40 °C) ebm-papst standard	Service life L_{10} (T_{max}) ebm-papst standard	Life expectancy L_{10} IPC (40 °C) see page 17	Curve
Type	**m³/h**	**cfm**	**VDC**	**VDC**	**dB(A)**	**Bel(A)**	□/■	**Watts**	**rpm⁻¹**	**°C**	**Hours**		**Hours**	
3412 NGLE	61	35.9	**12**	8...15	23	4.0	□	1.1	1 950	-20...+80	80 000 / 22 500		135 000	①
3412 NLE	61	35.9	**12**	8...15	23	4.0	■	0.8	1 950	-20...+85	80 000 / 17 500		135 000	①
3412 NGME	72	42.4	**12**	8...15	28	4.3	□	1.6	2 300	-20...+75	75 000 / 27 500		127 500	②
3412 NME	72	42.4	**12**	8...15	28	4.3	■	1.1	2 300	-20...+75	75 000 / 27 500		127 500	②
3412 NG	84	49.4	**12**	8...15	32	4.7	□	1.9	2 700	-20...+70	70 000 / 35 000		117 500	③
3412 N	84	49.4	**12**	8...15	32	4.7	■	1.7	2 700	-20...+70	70 000 / 35 000		117 500	③
3412 NGH	94	55.3	**12**	8...15	36	5.0	□	2.3	3 000	-20...+70	70 000 / 35 000		117 500	④
3412 NH	94	55.3	**12**	8...15	36	5.0	■	2.1	3 000	-20...+70	70 000 / 35 000		117 500	④
3412 NGHH	102	60.0	**12**	8...13.2	39	5.1	□	3.2	3 250	-20...+60	70 000 / 45 000		117 500	⑤
3412 NHH	102	60.0	**12**	8...13.2	39	5.1	■	2.9	3 250	-20...+60	70 000 / 45 000		117 500	⑤
3412 NHH-379	102	60.0	**12**	8...15	39	5.1	■	2.7	3 250	-20...+70	70 000 / 35 000		117 500	⑤
3414 NGL	61	35.9	**24**	18...28	23	4.0	□	1.4	1 950	-20...+70	80 000 / 40 000		135 000	①
3414 NL	61	35.9	**24**	18...28	23	4.0	■	1.1	1 950	-20...+70	80 000 / 40 000		135 000	①
3414 NGM	72	42.4	**24**	18...28	28	4.3	□	1.7	2 300	-20...+70	75 000 / 37 500		127 500	②
3414 NM	72	42.4	**24**	18...28	28	4.3	■	1.4	2 300	-20...+70	75 000 / 37 500		127 500	②
3414 NG	84	49.4	**24**	18...28	32	4.7	□	2.5	2 700	-20...+70	70 000 / 35 000		117 500	③
3414 N	84	49.4	**24**	18...28	32	4.7	■	2.1	2 700	-20...+70	70 000 / 35 000		117 500	③
3414 NGH	94	55.3	**24**	18...26	36	5.0	□	3.0	3 000	-20...+70	70 000 / 35 000		117 500	④
3414 NH	94	55.3	**24**	18...26	36	5.0	■	2.3	3 000	-20...+70	70 000 / 35 000		117 500	④
3414 NGHH	102	60.0	**24**	18...26	39	5.1	□	3.2	3 250	-20...+70	70 000 / 35 000		117 500	⑤
3414 NGHH-389	102	60.0	**24**	18...28	39	5.1	□	3.2	3 250	-20...+70	70 000 / 35 000		117 500	⑤
3414 NHH	102	60.0	**24**	18...26	39	5.1	■	3.1	3 250	-20...+70	70 000 / 35 000		117 500	⑤
3414 NHH-386	102	60.0	**24**	18...28	39	5.1	■	3.2	3 250	-20...+70	70 000 / 35 000		117 500	⑤
3418 N	84	49.4	**48**	36...56	32	4.7	■	2.4	2 700	-20...+70	70 000 / 35 000		117 500	③

Other 48 VDC models on request.

Air performance measured according to: ISO 5801.
Installation category A, without contact protection.
Noise: Total sound power level L_{WA} ISO 13002 measured on a hemisphere with a radius of 2 m.
Sound pressure level L_{pA} measured at 1 m distance from fan axis.
The values given are applicable only under the specified measuring conditions and may differ depending on the installation conditions.
In the event of deviation from the standard configuration, the parameters must be checked after installation!
For detailed information see http://www.ebmpapst.com/general conditions

2016-01 | 48 | Finger guards from p. 242 | ebmpapst

Figura 35. Características del ventilador. Fuente: Mouser electronics (2020), ebm-papst 3400N Serie Ventiladores tangenciales CC, https://ve.mouser.com/datasheet/2/120/3400N-275986.pdf

- Características del radiómetro CMP11.

Tabla 18. Características del radiómetro CMP11. Fuente: Campbell Scientific Inc (2019), Manual Pyranometers, https://s.campbellsci.com/documents/us/manuals/cmp6-cmp11-cmp21.pdf

CMP-series Specifications			
Specification	**CMP6**	**CMP10/CMP11**	**CMP21**
ISO Classification	First Class	Secondary Standard	
Maximum irradiance	2000 W•m^{-2}	4000 W•m^{-2}	
Spectral range (50% points)	285 to 2800 nm		
Response time (95 %)	<18 s	<5 s	
Expected daily uncertainty	<5%	<2%	
Zero offset due to thermal radiation (200 W•m^{-2})	<15 W•m^{-2}	<7 W•m^{-2}	
Zero offset due to temperature change (5 K•hr^{-1})	<4 W • m^{-2}	<2 W•m^{-2}	
Non-stability (change/year)	<1 %	<0.5%	
Non-linearity (0 to 1000 W•m^{-2})	<1%	<0.2%	
Directional error (up to 80° with 1000 W•m^{-2} beam)	<20 W•m^{-2}	<10 W•m^{-2}	
Tilt error (at 1000 W•m^{-2})	<1%	<0.2%	
Level accuracy	0.1°		
Operating temperature	–40 to 80 °C		
Temperature dependence of sensitivity	<4% (–10 to 40 °C)	<1% (–10 to 40 °C)	<1% (–20 to 50 °C)
Sensitivity	5 to 20 µV / W•m^{-2}	7 to 14 µV / W•m^{-2}	
Typical signal output for atmospheric applications	0 to 20 mV	0 to 15 mV	
Weight	0.6 kg (1.3 lb) without cable; 0.9 kg (2 lb) with 10 m (33 ft) cable		
Impedance[1]	20 to 200 Ω	10 to 100 Ω	
[1] Impedance is defined as the total electrical impedance at the radiometer output connector fitted to the housing. It arises from the electrical resistance in the thermal junctions, wires, and passive electronics within the radiometer.			

H. Curva característica de rendimiento de los módulos de Temperatura.

- Curva característica de rendimiento del módulo sensor de temperatura DS18B20.

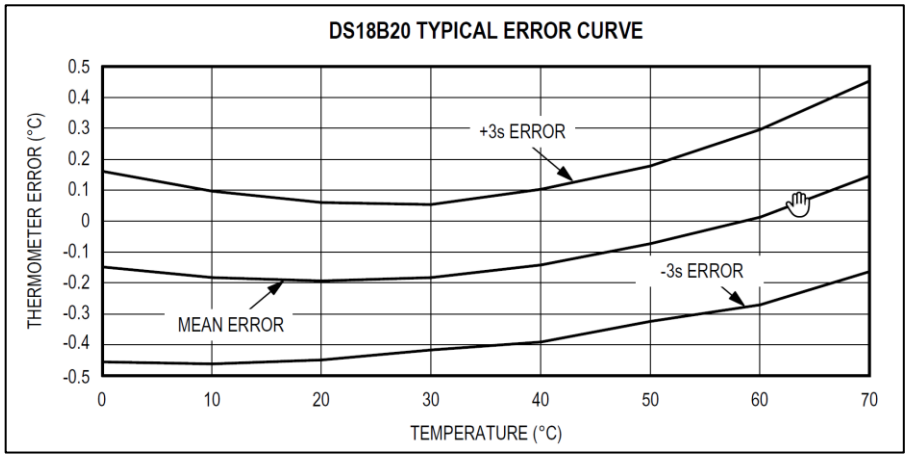


Figura 36. Curva de rendimiento típica para el sensor DS18B20. Fuente: Maxim Integrated Products, Inc. (2018), DS18B20, www.maximintegrated.com

- Curva característica de rendimiento del módulo sensor de temperatura MAX6675.

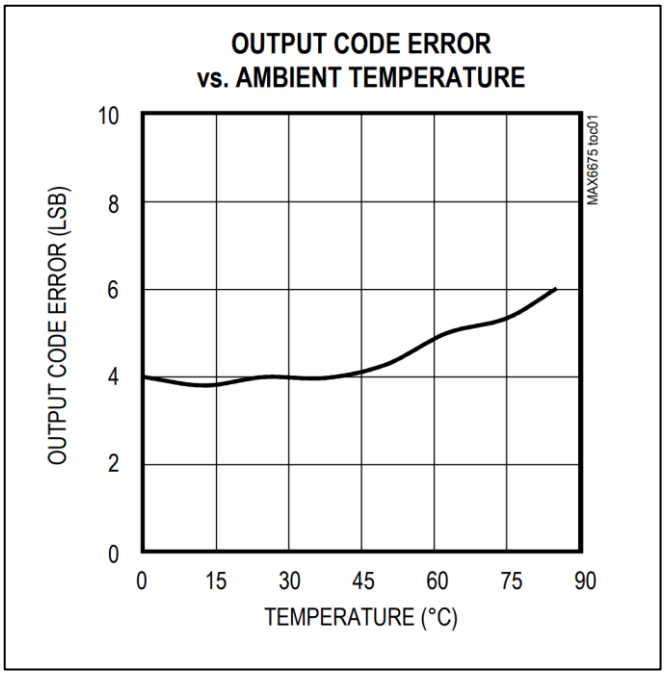


Figura 37. Características de operación típicas para el módulo MAX6675, convertidor de termocupla tipo K compensado en frío a digital. Fuente: Maxim Integrated Products, Inc. (2018), DS18B20, www.maximintegrated.com

I. Hardware y software utilizado

Para la elaboración de la presente Tesis se ha utilizado el siguiente hardware y software:

- La placa Raspberry Pi 3 B+ es producto de Raspberry Pi Foundation desarrollado en el Reino Unido. El sistema operativo usado en esta Tesis para la placa Raspberry es Raspbian, se descarga de la página web: https://www.raspberrypi.org/downloads/raspbian/

- La placa Arduino Mega es producto que inicialmente empezó en Italia y que después compro parte de las acciones la multinacional ARM Holdings (ARM). El programa Arduino IDE usado en esta Tesis se descarga de la página web: https://www.arduino.cc/en/Main/Software

- Las figuras de las conexiones Arduino y Raspberry fueron hechas con el programa fritzing que se descarga de la página web: https://fritzing.org/download/

- Algunas figuras utilizadas en la Tesis fueron elaboradas con el programa Visio de Microsoft. Algunos gráficos fueron elaborados con Excel de Microsoft.

- Para la elaboración de las imágenes a partir de las fotos tomadas, se utilizó el software Movavi Photo Noir y Movavi Photo Editor.

- Teamviewer es producto de GFI Software, se utiliza para conexión remota, el programa se puede descargar para Windows de la página web: https://www.teamviewer.com/es-mx/descarga/windows/ y para Raspberry de la página web: https://www.teamviewer.com/es-mx/descarga/raspberry-pi/