



**TECNOLÓGICO
NACIONAL DE MÉXICO**



IER
Instituto de Energías
Renovables

“Sensores embebidos para el flujo de calor.”

REPORTE DE RESIDENCIA PROFESIONAL

PRESENTA:

Gonzalo Octavio Gaona Terrones.

No. 14091070.

gonzalogaona96@gmail.com

Tel. celular (044) 734 146 51 12

EMPRESA:

Instituto de Energías Renovables.

Calle priv. Xochicalco S/N.

Municipio de Temixco, Morelos.

C.P - 62580 México.

Tels. (52) 777 362 0090 (ext. 29744) y (52) 555 622 9744.

RFC: UNA2907227Y5

ASESOR INTERNO:

Asesor: MC. José Efraín Ruiz Domínguez.

ASESOR EXTERNO:

Asesor externo: Dr. Guillermo Barrios del Valle.

Período de realización: 10 de julio de 2020 al 10 de enero del 2021.

Agradecimientos.

En primer lugar, agradezco a Dios por permitirme llegar hasta este punto de mi vida.

Al Instituto de Energías Renovables por permitirme realizar mi residencia profesional ahí, así como por proporcionarme el material necesario para el presente proyecto.

Al Dr. Guillermo Barrios del Valle me quien brindó asesorías de programación en el lenguaje MicroPython.

Al Dr. Guillermo Ramírez Zúñiga, por las asesorías en tópicos de electrónica y programación del microcontrolador PSoC 5LS.

Al MC. José Efraín Ruiz Domínguez por su disposición a apoyarme.

Finalmente, quiero agradecer a mis padres Gonzalo Gaona Sánchez y Griselda Terrones Sánchez quienes siempre me brindaron su apoyo, al igual que a Consuelo Gonzales quien me motivó a seguir con mis estudios.

Resumen.

Durante el desarrollo del presente proyecto se diseña y construye un sistema de instrumentación para sensores de flujo de calor, con la finalidad de posteriormente se pueda emplear para medir la transferencia de calor en las dos caras de un muro.

Adicionalmente, el presente proyecto se realizó de dos maneras distintas, las cuales se describen a continuación:

La primera es utilizando un ESP32 para leer el voltaje y convertirlo a unidades de flujo de calor ($W\ m^2$), para lo cual, es necesario realizar un sistema de acondicionamiento de señal que permita interpretar la señal eléctrica proveniente del sensor de calor y amplificarla. Además, cuenta con la capacidad de publicar los datos obtenidos en una plataforma IoT.

La segunda manera es la utilización de un PSoC 5, el cual cuenta con la resolución necesaria para conectar el sensor directamente, es decir, sin tener que amplificar la señal del voltaje; adicionalmente, se utiliza el ESP32 para poder publicar los datos, ya que el PSoC no cuenta con un módulo wifi como lo tiene el ESP32; sin embargo, por cuestiones de tiempo se plantea llegar hasta realizar la comunicación serial entre el ESP32 y el PSoC 5LS para que posteriormente sea posible publicar los datos en la plataforma ThingsBoard .

Índice general.

CAPÍTULO 1. GENERALIDADES DEL PROYECTO.	1
1.1. Introducción.	1
1.2. Descripción de la empresa u organización y del puesto o área del trabajo el estudiante.	1
1.2.1. Ubicación.	1
1.2.2. Misión.	2
1.2.3. Logotipo de la empresa.	2
1.3. Problema a resolver.	2
1.4. Objetivo general.	3
1.5. Objetivos específicos.	3
1.6. Justificación.	3
CAPÍTULO 2. MARCO TEÓRICO.	4
1.1. Mecanismos de transferencia de calor.	4
1.2. Sensores de flujo de calor.	7
1.3. Amplificadores operacionales.	11
1.4. Microcontrolador y publicación de datos.	15
CAPÍTULO 3. DESARROLLO DE ACTIVIDADES.	30
3.2. Equipo utilizado.	30
3.3. Diseño del circuito y cálculos.	31
3.4. Desarrollo de la programación.	37
3.5. Pruebas de funcionamiento.	48
CAPÍTULO 4. RESULTADOS.	53
CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES.	57
CAPÍTULO 6. FUENTES DE INFORMACIÓN.	58
CAPITULO 7. ANEXOS.	60
7.1. Códigos completos.	60
7.2. Fichas técnicas.	69

Índice de ilustraciones.

Ilustración 1. Logotipo del instituto y de la UNAM. ("IER-UNAM", 2020)	2
Ilustración 2. Transmisión de calor por conducción. ("Ricardo Cabrera, CALOR Y TERMODINAMICA, conducción, ejercicios ", 2020)	4
Ilustración 3. Ejemplo de convección. ("¿Qué es la convección?", 2020).....	6
Ilustración 4. Transferencia de calor por radiación. ("La pérdida de calor corporal - thetrekkinglife.com", 2020).....	7
Ilustración 5. Sensor de flujo de calor Fluxteq PHFS-01e. (Fluxteq.com, 2020)	8
Ilustración 6. Composición de una termopila. ("Termopares-la física y la química", 2020)	9
Ilustración 7. Curva de calibración de un sensor. (Calibración de Sensores, 2020)	10
Ilustración 8. Se calibran teniendo una temperatura conocida que se pueda aplicar al sensor para poderlo calibrar. (Laurila, H. 2020)	11
Ilustración 9 Diagrama de un amplificador operacional. ("Introducción a los Amplificaciones Operacionales", 2020).....	12
Ilustración 10. AD620 físicamente. ("AD620 Instrumentation Amplifiers", 2020)	12
Ilustración 11. presenta los pines de conexión del AD620.	13
Ilustración 12. LM741 físicamente. ("741cn dip-8 UA741 LM741 chips amplificadores operacionales", 2020)	14
Ilustración 13. Diagrama de integrado LM741.	15
Ilustración 14. Diagrama bloques del ESP32.	19
Ilustración 15. Pinout del ESP32. (OLED, 2020)	20
Ilustración 16. Pinout del PSoC 5.	23
Ilustración 17. Diagrama de bloques del PSoC 5. ("cypress", 2020)	25
Ilustración 18. Transmisión de datos en paralelo.....	26
Ilustración 19. Transmisión de datos en serie.	26
Ilustración 20. Valores típicos de transmisión. ("Vitesse de transmission les plus utilisées", 2021)	27
Ilustración 21. voltaje representado por el flujo de calor.	33
Ilustración 22. voltaje representado por el flujo de calor.....	34
Ilustración 23. voltaje representado por el flujo de calor.....	36
Ilustración 24. Diagrama de conexión del acondicionamiento de la señal.	37
Ilustración 25. Diagrama de flujo del programa.	38
Ilustración 26. Código "main" en MicroPython para leer el flujo de calor y publicarlo en ThingsBoard.....	39
Ilustración 27. Configuración de ADC.	40
Ilustración 28. Configuración de ADC.	41
Ilustración 29. Configuración del multiplexor (AMux).	42
Ilustración 30. Configuración de los pines.	43
Ilustración 31. AMux y ADC conectados.	43
Ilustración 32. Configuración del puerto UART.	44
Ilustración 33. Puerto UART.	45

Ilustración 34. Configuración de los pines en PSoC 5.	45
Ilustración 35. Pines en el microcontrolador.....	46
Ilustración 36. Ciclo for en el código del PSoC 5LS.	47
Ilustración 37. Código para realizar la comunicación en Python.....	47
Ilustración 38. Código para leer un voltaje en MicroPython.....	48
Ilustración 39. Valores del voltaje impresos en la terminal.	49
Ilustración 40. Diagrama del divisor de voltaje.	49
Ilustración 41. Resultado del divisor de voltaje.....	50
Ilustración 42. Valor del voltaje en la segunda etapa de amplificación.....	51
Ilustración 43. Sistema completo de la instrumentación del sensor de flujo de calor para el ESP32.	53
Ilustración 44. Flujo de calor a través del tiempo, utilizando el ESP32.....	54
Ilustración 45. Sistema completo de la instrumentación del sensor de flujo de calor para el PSoC 5LS.....	55
Ilustración 46. Datos recibidos por el sensor de flujo de calor.	56

CAPÍTULO 1. GENERALIDADES DEL PROYECTO.

1.1. Introducción.

El presente trabajo fue realizado en el Instituto de Energías Renovables de la Universidad Nacional Autónoma de México (IER-UNAM), bajo el nombre “Sensores embebidos para el flujo de calor.” El IER-UNAM se encuentra ubicado en la privada Xochicalco S/N, colonia Centro de Temixco, Morelos, México.

La transferencia de calor se presenta en diferentes procesos cotidianos y conocer el flujo de calor a través de una pared ayuda a comprender el comportamiento térmico de una edificación y contribuye al ahorro de energía. Por lo que se convierte en una necesidad el contar con un sistema que sea capaz de monitorear el flujo de calor al cual se encuentra la edificación.

El objetivo de este proyecto es el de diseñar y construir un sistema para medir el flujo de calor en muros y que los datos sean publicados en una plataforma IoT. Para esto, se requiere diseñar un sistema con la capacidad de interpretar la señal eléctrica mediante un sensor de flujo de calor de la marca “Fluxteq”.

1.2. Descripción de la empresa u organización y del puesto o área del trabajo el estudiante.

El instituto en donde es realizado el presente trabajo lleva por nombre “Instituto de Energías Renovables”, el cual pertenece a la UNAM.

La misión del Instituto es realizar investigación científica básica y aplicada en energía, con énfasis en energías renovables, que coadyuven al desarrollo de tecnologías energéticas sustentables; llevar a cabo estudios, asesorías y capacitación a los distintos sectores de la sociedad; formar recursos humanos especializados, y difundir los conocimientos adquiridos para el beneficio del país. Somos un referente nacional y la principal institución de México activa en la investigación, innovación, divulgación y formación de especialistas en tecnologías energéticas sustentables. (“IER-UNAM - Identidad - Por un futuro sustentable”, 2020)

1.2.1. Ubicación.

El Instituto de Energías Renovables se encuentra ubicado en Priv. Xochicalco S/N Temixco, Morelos México, con código postal 62580.

1.2.2. Misión.

Realizar investigación científica básica y aplicada en energía, con énfasis en energías renovables, que coadyuven al desarrollo de tecnologías energéticas sustentables; llevar a cabo estudios, asesorías y capacitación a los distintos sectores de la sociedad; formar recursos humanos especializados, y difundir los conocimientos adquiridos para el beneficio del país. ("IER-UNAM - Identidad - Por un futuro sustentable", 2020)

1.2.3. Logotipo de la empresa.



Ilustración 1. Logotipo del instituto y de la UNAM. ("IER-UNAM", 2020)

1.3. Problema a resolver.

El confort térmico al interior de las edificaciones la satisfacción con la temperatura o el flujo de calor de un lugar. Es un conjunto de condiciones en las que la mayoría de las personas se sienten cómodas térmicamente. El confort térmico se encuentra entre las condiciones más importantes para mejorar la comodidad y la satisfacción de los ocupantes con su ambiente interior por lo cual, se considera de suma importancia.

El flujo de calor en edificaciones se da por la radiación, convección y conducción. En este particular, lo que se desea medir es transferencia de calor por conducción, a través de las dos caras de un muro.

Es de suma importancia saber el flujo de calor que existe al interior del recinto, para esto, es indispensable contar con un dispositivo que sea capaz de realizar la medición del flujo de calor en un muro.

En este proyecto, se realiza el diseño para medir el flujo de calor utilizando un sensor Fluxteq y poder publicar el resultado en una plataforma de IoT

1.4. Objetivo general.

El principal objetivo de este trabajo es realizar el diseño y construcción, así como realizar las pruebas de funcionamiento de un dispositivo para medir el flujo de calor en muros, usando tecnologías libres.

1.5. Objetivos específicos.

Los objetivos específicos que se desarrollaron en este trabajo son los siguientes:

- Diseño del circuito de ganancia: Elaboración del diseño del circuito de ganancia para amplificar el voltaje proporcionado específico por el sensor de flujo de calor utilizado.
- Armado del circuito: Una vez realizado el diseño del circuito, se continúa al armado del mismo.
- Programación: Realizar la programación del microcontrolador utilizado.
- Análisis de ruido: Realizar un análisis de las interferencias de la señal eléctrica, pueden ser más perjudiciales.
- Adquisición y visualización de datos: A través del microcontrolador, se obtienen los datos que sean captados por el sensor y enviados a una interfaz gráfica.
- Pruebas del sistema: Al finalizar el proyecto, se realizan pruebas de funcionamiento para verificar su desempeño.

1.6. Justificación.

Actualmente, el Instituto de Energías Renovables requiere de un sensor que mida el flujo de calor en un muro para completar otros proyectos, por lo que es de suma importancia el desarrollo de dicho sensor.

El presente proyecto brindará la lectura del flujo de calor que existe en los muros, esto utilizando tecnologías libres y a un bajo costo.

CAPÍTULO 2. MARCO TEÓRICO.

1.1. Mecanismos de transferencia de calor.

El calor se define como la forma de energía que se puede transferir de un sistema a otro, como resultado de la diferencia de temperatura. (Cengel, 2011)

En general, existen tres mecanismos distintos por los cuales se puede transmitir el calor, los cuales son:

- Conducción.
- Convección.
- Radiación.

Estos se presentan en la vida cotidiana y son descritos a continuación.

La conducción es la transferencia de energía de las partículas con más energía hacia las contiguas con menos energía, como resultado de interacciones entre esas partículas. La conducción se puede dar en los sólidos, líquidos o gases. Se debe a las colisiones y a la difusión de las moléculas cuando se mueven, en líquidos y gases. Mientras que en los sólidos se debe a la combinación de las vibraciones de las moléculas en una retícula y al transporte de energía por parte de los electrones libres. (Cengel, 2011)

En la ilustración 2 se muestra que el calor Q es transferido por conducción de una fuente de calor caliente hacia una fuente fría. El calor será transmitido a mayor tasa cuando el área A sea menor y la distancia ΔX sea menor.

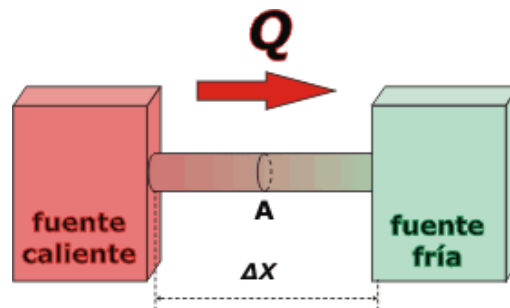


Ilustración 2. Transmisión de calor por conducción. ("Ricardo Cabrera, CALOR Y TERMODINAMICA, conducción, ejercicios", 2020)

La razón de la conducción en estado permanente de calor a través de una capa plana es proporcional a la diferencia de temperatura a través de ésta y al área de transferencia de calor, pero es inversamente proporcional al espesor de esa capa. (Cengel, 2011)

Es posible cuantificar los procesos de transferencia de calor gracias a los modelos adecuados matemáticos. Estos modelos nos son de utilidad para conocer la cantidad de energía que se transfiere por unidad de tiempo. Para la conducción de calor en estado permanente, la ecuación o modelo se conoce como ley de Fourier. Este modelo se expresa como:

$$q''_x = -k \frac{dT}{dX},$$

(Incropera, 2007) donde q'' es el calor, k es la constante de conductividad térmica del material y $\frac{dT}{dX}$ es el gradiente térmico en una distancia x .

Los experimentos han demostrado que la razón de la transferencia de calor, q'' , a través de la pared se duplica cuando se duplica la diferencia de temperatura dT de uno a otro lado de ella, o bien, se duplica el área A perpendicular a la dirección de la transferencia de calor; pero se reduce a la mitad cuando se duplica el espesor L de la pared. Por lo tanto, se concluye que la razón de la conducción de calor a través de una capa plana es proporcional a la diferencia de temperatura a través de ésta y al área de transferencia de calor, pero es inversamente proporcional al espesor de esa capa; es decir,

$$q = kA \frac{T_1 - T_2}{\Delta x} = kA \frac{T_1 - T_2}{L}$$

(Cengel, 2011) donde q'' es el calor, k es la constante del material, A es el área, T_1 y T_2 son las temperaturas y L es el espesor de la pared.

Este trabajo está realizado para medir la transferencia de calor por conducción.

Otro mecanismo de transferencia de calor la convección.

La convección es el modo de transferencia de energía entre una superficie sólida y el líquido o gas adyacente que está en movimiento y comprende los efectos combinados de la conducción y el movimiento de fluidos. Entre más rápido es el

movimiento de un fluido, mayor es la transferencia de calor por convección. En ausencia de cualquier movimiento masivo de fluido, la transferencia de calor entre una superficie sólida y el fluido adyacente es por conducción pura. La presencia de movimiento masivo del fluido acrecienta la transferencia de calor entre la superficie sólida y el fluido, pero también complica la determinación de las razones de esa transferencia. (Cengel, 2011)

En la ilustración 3 se muestra un ejemplo típico de transferencia de calor por convección en el cual dos fluidos a distinta temperatura están interactuando y en movimiento.

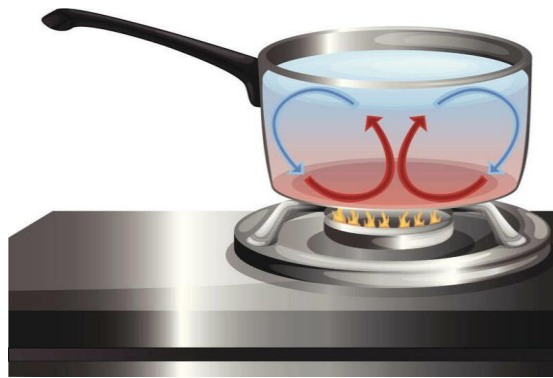


Ilustración 3. Ejemplo de convección al interior de la olla. ("¿Qué es la convección?", 2020)

La convección recibe el nombre de convección forzada si el fluido es forzado a fluir sobre la superficie mediante medios externos como un ventilador, una bomba o el viento. Como contraste, se dice que es convección natural (o libre) si el movimiento del fluido es causado por las fuerzas de empuje que son inducidas por las diferencias de densidad debidas a la variación de la temperatura en ese fluido. La transferencia de calor entre el bloque y el aire circundante será por conducción si la diferencia de temperatura entre el aire y el bloque no es suficientemente grande como para vencer la resistencia de ese aire al movimiento y, por consiguiente, para iniciar corrientes naturales de convección. Los procesos de transferencia de calor que comprenden cambio de fase de un fluido también se consideran como convección a causa del movimiento de ese fluido inducido durante el proceso, como la elevación de las burbujas de vapor durante la ebullición o la caída de las gotitas de líquido durante la condensación. A pesar de la complejidad de la convección, se observa que la rapidez de la transferencia de calor por convección es proporcional a la diferencia de temperatura y se expresa en forma conveniente por la ley de Newton del enfriamiento como:

$$q_{conv} = hA_s(T_s - T_{\infty})$$

donde h es el coeficiente de transferencia de calor por convección, en $\text{W/m}^2 \cdot ^\circ\text{C}$ o, A_s es el área superficial a través de la cual tiene lugar la transferencia de calor por convección, T_s es la temperatura de la superficie y T_{∞} es la temperatura del fluido suficientemente alejado de esta superficie. (Cengel, 2011)

El último, pero no menos importante, es la transferencia de calor por radiación.

La radiación es la energía emitida por la materia en forma de ondas electromagnéticas (o fotones) como resultado de los cambios en las configuraciones electrónicas de los átomos o moléculas. A diferencia de la conducción y la convección, la transferencia de calor por radiación no requiere la presencia de un medio interventor. (Cengel, 2011)

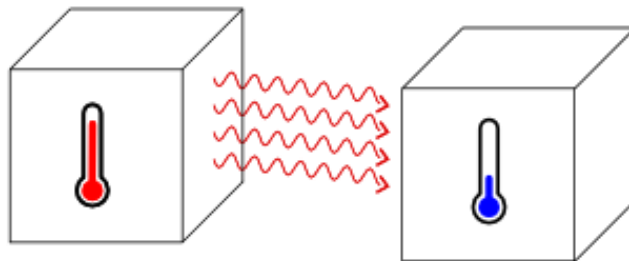


Ilustración 4. Transferencia de calor por radiación. ("La pérdida de calor corporal - thetrekkinglife.com", 2020).

La radiación es un fenómeno volumétrico y todos los sólidos, líquidos y gases emiten, absorben o transmiten radiación en diversos grados. Sin embargo, la radiación suele considerarse como un fenómeno superficial para los sólidos que son opacos a la radiación térmica, como los metales, la madera y las rocas, ya que las radiaciones emitidas por las regiones interiores de un material de ese tipo nunca pueden llegar a la superficie, y la radiación incidente sobre esos cuerpos suele absorberse en unas cuantas micras hacia adentro de dichos sólidos.

1.2. Sensores de flujo de calor.

Un sensor es un dispositivo que capta magnitudes físicas u otras alteraciones de su entorno. (Oxford, 2020)

El presente proyecto se enfoca en los sensores de flujo de calor de la marca Fluxteq debido a que el instituto IER-UNAM contaba con dicho recurso. Cabe señalar que estos sensores utilizan el efecto termopila.



Ilustración 5. Sensor de flujo de calor Fluxteq PHFS-01e. (Fluxteq.com, 2020)

Una termopila es un dispositivo electrónico que convierte energía térmica en energía eléctrica. Está compuesto de varios termopares conectados normalmente en serie o, menos generalmente, en paralelo. Las termopilas generan un voltaje proporcional a una diferencia de temperatura local o gradiente de temperatura.

Proporcionan un voltaje de salida en respuesta a un gradiente de temperatura, formando parte de dispositivos como sensores de flujo térmico y en controles de seguridad de quemadores de gas.

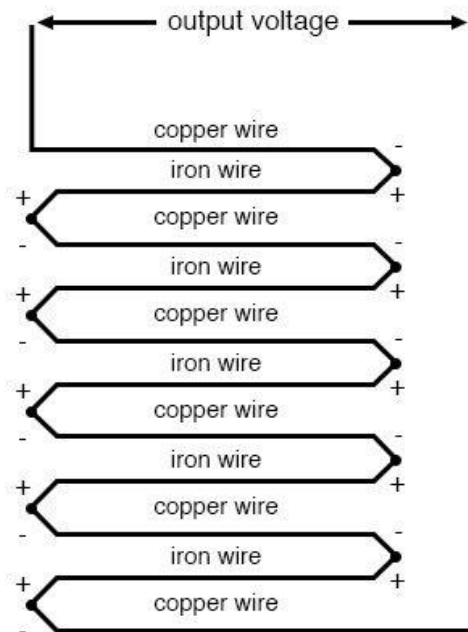


Ilustración 6. Composición de una termopila. ("Termopares-la física y la química", 2020)

La mayoría de los sensores de flujo de calor que están disponibles en el mercado son termopilas de temperatura diferencial. Operan midiendo una diferencia de temperatura relativamente pequeña a través de una capa de resistencia térmica (TRL) que es el material del núcleo del sensor de flujo de calor. Usando la ley de Fourier de conducción de calor con un supuesto de transferencia de calor unidimensional en condiciones de estado estacionario. (Fluxteq.com, 2020)

$$q''_x = -k \frac{dT}{dX}$$

En términos eléctricos, una termopila se compone de varios termopares conectados en serie. Juntos, generan un voltaje que es proporcional a la diferencia de temperatura entre dos puntos; Esta diferencia da una medida de temperatura relativa. ("Diarioelectronico hoy", 2020)

La calibración del sensor es necesaria para poder interpretar la información que proporciona el sensor permitiendo corregir inexactitudes.

Calibrar un sensor consiste en obtener la curva de calibración, la cual representa la equivalencia entre la magnitud física medida y la respuesta eléctrica del sensor. (Fluxteq.com, 2020)

Histéresis es la máxima diferencia entre las señales de salida correspondientes a un mismo valor de la magnitud a medir, que se obtiene cuando dicho valor se obtiene mediante el aumento y la disminución de dicha magnitud. (Calibración de Sensores, 2020)

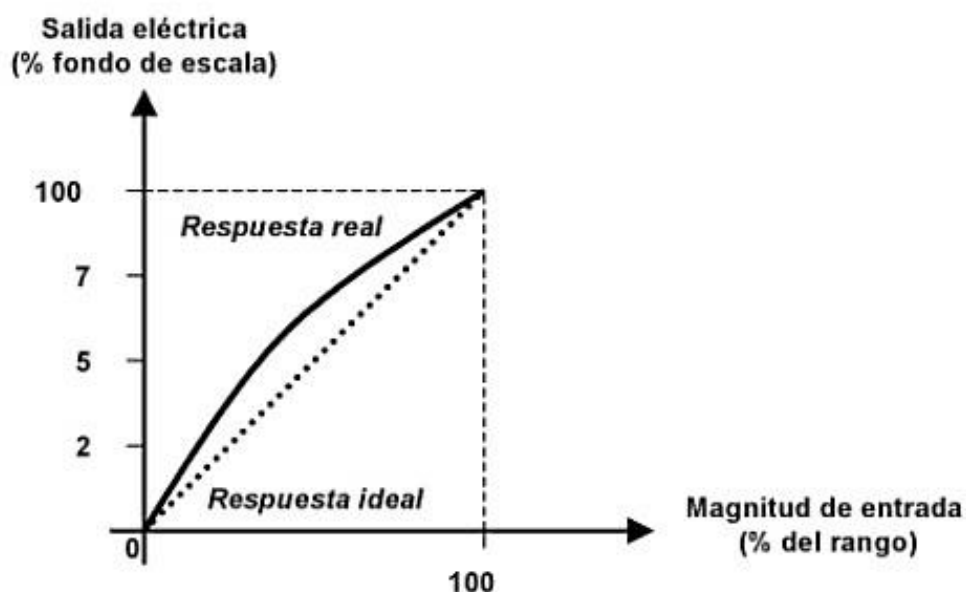


Ilustración 7. Curva de calibración de un sensor. (Calibración de Sensores, 2020)

Actualmente, existen dos sistemas de calibración para los sensores Fluxteq:

- **Sistemas de calibración de conducción.**

FluxTeq ha desarrollado dos sistemas de calibración de conducción y se han calibrado de forma cruzada entre sí para garantizar datos de medición precisos y una calibración del sensor de alta precisión. Un pequeño sistema utiliza placas calentadas y enfriadas por agua y un sensor de referencia para calibrar un sensor de prueba intercalado entre las almohadillas de separación.

- **Sistema de calibración de radiación.**

Se utiliza un sensor de referencia para calibrar un banco de lámparas de cuarzo en un rango de configuraciones de flujo de calor. Luego, el sensor de prueba se monta

en una placa enfriada por agua y la lámpara se ilumina sobre el rango calibrado. Los voltajes del sensor de salida se registran para varios ajustes de la lámpara y se toman datos de sensibilidad para un rango de flujos de calor por radiación. (Fluxteq.com, 2020)

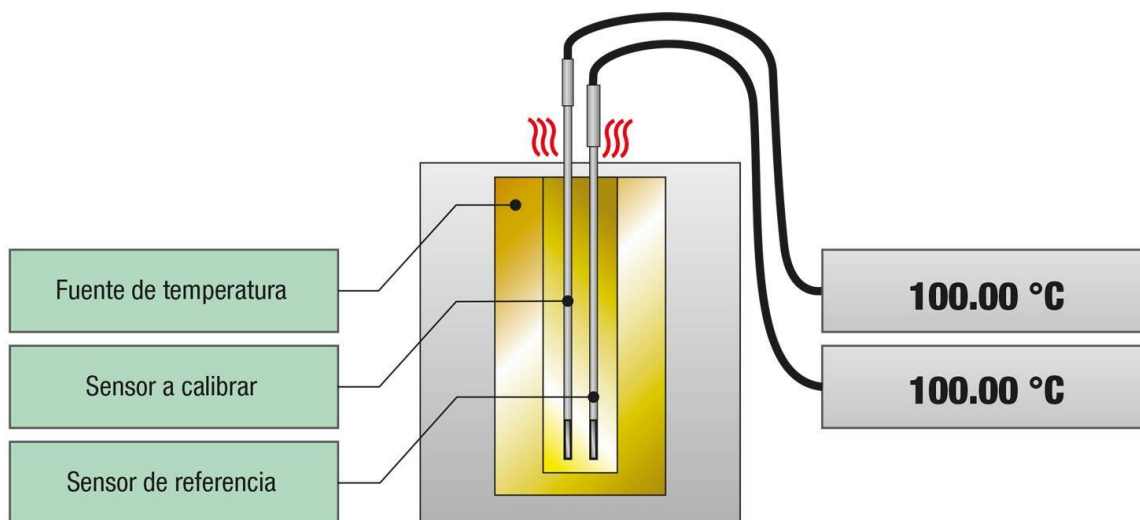


Ilustración 8. Se calibran teniendo una temperatura conocida que se pueda aplicar al sensor para poderlo calibrar. (Laurila, H. 2020)

1.3. Amplificadores operacionales.

Un amplificador operacional, o amp-op, es un amplificador diferencial de muy alta ganancia con alta impedancia de entrada y baja impedancia de salida, ya que permite el paso de una cantidad pequeña de corriente por unidad de voltaje aplicado en ese punto. Los usos típicos del amplificador operacional son proporcionar cambios en la amplitud del voltaje (amplitud y polaridad), en osciladores, en circuitos de filtrado y en muchos tipos de circuitos de instrumentación. Un amplificador operacional contiene varias etapas de amplificadores diferenciales para alcanzar una muy alta ganancia de voltaje. (Boylestad, Nashelsky & Suárez Fernández, 2009).

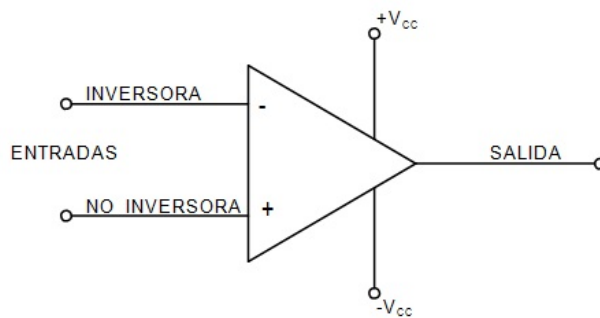


Ilustración 9 Diagrama de un amplificador operacional. ("Introducción a los Amplificaciones Operacionales", 2020)

En este proyecto se requirió de dos tipos de amplificadores operacionales, los cuales son AD620 y LM741. El AD620 se utilizó en este proyecto para incrementar el voltaje de salida y el LM741 se utilizó para que el sensor pudiera ser leído para voltajes positivos y negativos.

ADI presenta la serie AD620 que viene en paquetes SOIC y DIP de 8 conductores y son amplificadores de instrumentación de bajo costo y alta precisión que, con una resistencia externa, permiten al usuario establecer ganancias de 1 a 10.000. El diseño compacto y bajo consumo de energía (solamente 1.3 mA fuente de corriente) lo hacen apto para aplicaciones portátiles o remotas que requieren una batería. ("DigiKey Electronics México", 2020)

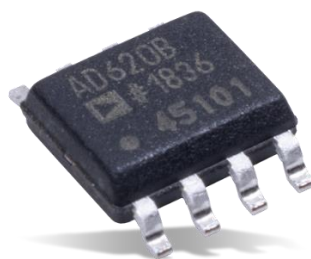


Ilustración 10. AD620 físicamente. ("AD620 Instrumentation Amplifiers", 2020)

El AD620, con su alta precisión de no linealidad de 40 pulsaciones por minuto (ppm) (máximo), baja tensión de deriva de 50 μV (máximo) y compensación de deriva de 0,6 $\mu\text{V} / ^\circ \text{C}$ (máxima), es ideal para uso en sistemas de adquisición de datos precisos tales como balanzas de peso e interfaces de transductor. Además, el bajo

ruido, baja corriente de polarización de entrada y baja potencia de entrada del AD620 lo hacen apto para aplicaciones médicas tales como ECG y monitores de presión sanguínea no invasivos. ("DigiKey Electronics México", 2020)

En la siguiente ilustración se presenta los pines de conexión del AD620, donde R_G es la resistencia de ganancia, IN es la entrada de voltaje, V_S es el voltaje que alimenta al amplificador, REF es el voltaje referenciado. Y OUTPUT es el voltaje que sale.

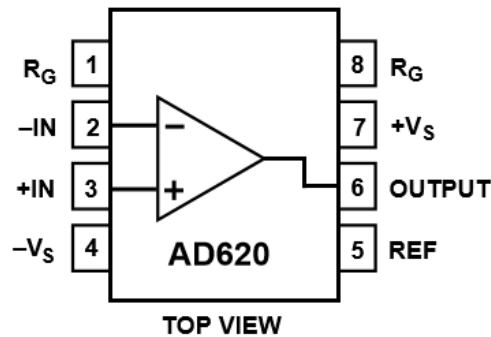


Ilustración 11. presenta los pines de conexión del AD620.

Sus características son las siguientes:

- Fácil de usar
- Rango de ganancia de 1 a 10.000 con una resistencia externa
- Amplio rango de potencia (± 2.3 V a ± 18 V)
- Rendimiento más alto que tres diseños IA de op amp
- Baja alimentación, 1.3 mA (máx.) de corriente de alimentación
- Voltaje de polarización de entrada 50 μ V (máx.).
- Corriente de polarización de entrada 1,0 nA (máx.),
- Relación de rechazo de modo común de 100 dB (mínimo) ($G = 10$)
- Bajo nivel de ruido
- Ruido de voltaje de entrada de 9 nV/ $\sqrt{\text{Hz}}$ a 1 kHz
- 0.28 μ V p-p ruido (0.1 Hz a 10 Hz)
- Ancho de banda de 120 kHz ($G = 100$)

El LM741 es un amplificador operacional de propósito general que cuenta con un rendimiento mejorado sobre los estándares de la industria como el LM709. El LM741 ofrece muchas características que hacen que su aplicación sea casi a

prueba de protección de sobrecarga en la entrada y la salida, ausencia de enclavamiento cuando se excede el rango de modo común y libre de oscilaciones. Son repuestos de inserción directa para 709C, LM201, MC1439 y 748 en la mayoría de las aplicaciones. ("LM741CN", 2020)



Ilustración 12. LM741 físicamente. ("741cn dip-8 UA741 LM741 chips amplificadores operacionales", 2020)

- No se produce enclavamiento cuando se excede el rango de modo común.
- Producto ecológico sin Sb/Br.
- Aplicaciones: Procesado de señal.
- Protección contra sobrecarga en entrada y salida.

Sus características son las siguientes:

- Tipo amplificador: Uso general.
- Número de amplificadores: 1.
- Tensión de alimentación mínima: 10 V.
- Tensión de alimentación máxima: 44 V.
- Ancho de banda: 1.5 MHz.
- Velocidad de cambio (slew rate): 0.5 V/ μ s.
- Temperatura de funcionamiento mínima: 0 ° C.
- Temperatura de funcionamiento máxima: 70 ° C.
- Encapsulado: DIP
- 8 pines.

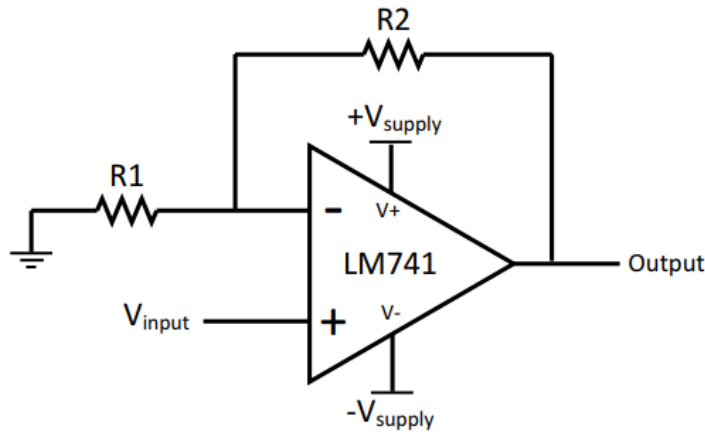


Ilustración 13. Diagrama de integrado LM741.

1.4. Microcontrolador y publicación de datos.

En el presente proyecto, se optó por utilizar un microcontrolador ESP y transferir los datos obtenidos a la plataforma ThingsBoard.

El Internet of Things (IoT) o en español, El Internet de las Cosas describe la red de objetos físicos (cosas) que llevan sensores integrados, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet. Estos dispositivos abarcan desde objetos domésticos cotidianos hasta sofisticadas herramientas industriales. ("¿Qué es Internet of Things (IoT)?", 2020)

En los últimos años, IoT se ha convertido en una de las tecnologías más importantes del siglo XXI. Ahora que podemos conectar objetos cotidianos como aparatos de cocina, vehículos, termostatos, monitores de bebés a Internet mediante dispositivos integrados, es posible la comunicación fluida entre personas, procesos y cosas.

Por medio de la informática de bajo costo, la nube, big data, la analítica y las tecnologías móviles, los objetos físicos pueden compartir y recopilar datos con una intervención humana mínima. En este mundo hiperconectado, los sistemas digitales pueden grabar, supervisar y ajustar cada interacción entre los objetos conectados. Este mundo físico se combina con el mundo digital de modo que pueden cooperar.

Los microcontroladores ESP32 y ESP8266, son placas programables que tienen una característica principalmente; cuentan con conectividad Wifi y son de muy bajo costo, se pueden adquirir desde \$200.00.

Las aplicaciones de IoT son aplicaciones prediseñadas de software como servicio que pueden analizar y presentar los datos captados por los sensores de IoT a los usuarios del negocio a través de cuadros de mando. ("¿Qué es Internet of Things (IoT)?", 2020).

Las aplicaciones de IoT pueden utilizar algoritmos de machine learning para analizar las enormes cantidades de datos de los sensores conectados en la nube. Mediante el uso de cuadros de mando y alertas de IoT en tiempo real, puede obtener visibilidad en indicadores de rendimiento claves, estadísticas del tiempo medio entre fallos y otra información. Los algoritmos basados en machine learning pueden identificar anomalías y enviar alertas a los usuarios e incluso activar correcciones automatizadas o medidas de respuesta proactivas. ("¿Qué es Internet of Things (IoT)?", 2020).

Con las aplicaciones de IoT basadas en la nube, los usuarios empresariales pueden mejorar rápidamente los procesos existentes para las cadenas de suministro, el servicio al cliente, los recursos humanos y los recursos financieros. No es necesario volver a crear todos los procesos empresariales. ("¿Qué es Internet of Things (IoT)?", 2020).

La capacidad de IoT para proporcionar información de los sensores, así como para hacer posible la comunicación entre dispositivos da impulso a un amplio conjunto de aplicaciones.

A continuación, se enumeran algunas de las aplicaciones más populares y lo que hacen.

- **Mejorar el seguimiento y la “delimitación” de activos físicos.** El seguimiento permite a las empresas determinar rápidamente la ubicación de los activos. La delimitación les permite asegurarse de que los activos de alto valor están protegidos del robo y la extracción.
- **Usar wearables para supervisar el análisis de la salud humana y las condiciones ambientales.** Los wearables de IoT hacen que las personas entiendan mejor su propia salud y permiten a los médicos supervisar de forma remota a los pacientes. Esta tecnología también hace posible que las empresas puedan hacer un seguimiento de la salud y la seguridad de sus empleados, lo que es especialmente útil para los empleados que trabajan en condiciones peligrosas.
- **Facilitar los cambios de procesos empresariales.** Un ejemplo de ello es el uso de dispositivos de IoT para supervisar el estado de máquinas remotas y activar las llamadas de servicio para el mantenimiento preventivo. La capacidad de supervisar las máquinas de forma remota también hace posible nuevos modelos empresariales de producto como servicio, modelos en los

que los clientes ya no tienen la necesidad de comprar un producto, sino que pagan por su uso. ("¿Qué es Internet of Things (IoT)?", 2020).

ThingsBoard es una plataforma IoT de código abierto en la que podemos almacenar, visualizar y analizar los datos de nuestros dispositivos, datos que serán enviados por el ESP una vez que sea acondicionada la señal del sensor de flujo de calor.

A continuación, son descritas las partes que forman la plataforma:

- **Activos:** ThingsBoard permite jerarquizar entidades a través de las relaciones entre los activos; siguiendo el ejemplo anterior, cada empresa de mantenimiento podría crear un activo por cada barrio, y dentro de ellos un activo por cada edificio, y dentro de ellos un activo por cada vivienda.
- **Deashboards:** Los datos enviados por los nodos se denominan telemetrías, y podemos visualizarlos utilizando multitud de widgets (tablas, gráficos de líneas, indicadores analógicos...).
- **Reglas:** ThingsBoard incluye un motor de reglas similar a Node-RED con el que podemos analizar las telemetrías y realizar acciones, como enviar mensajes en caso de que se supere cierto umbral de temperatura.
- **Alarmas:** Mediante las alarmas podemos mantener un registro de las situaciones en las que un determinado sensor se encuentra en estado de excepción, es decir, está registrando un valor fuera del rango tolerable (por ejemplo, un sensor de temperatura que supera cierto umbral), o no ha actualizado su telemetría desde hace más tiempo del que tiene asignado.

Es importante mencionar que no es necesario instalar ningún software adicional.

Por otro lado, el ESP permite ser programado con una variedad de lenguajes; como puede ser Arduino, Lua, C, Node.js y por supuesto MicroPython. (Jaquería, 2020)

En este proyecto, la programación es realizada con MicroPython y el microcontrolador ESP32.

MicroPython es un pequeño pero eficiente interprete del lenguaje de programación Python 3 que incluye un subconjunto mínimo de librerías y que además está optimizado para que pueda correr en microcontroladores y ambientes restringidos. (Morales, 2020)

Con MicroPython tienes la posibilidad de escribir códigos más simples, en lugar de usar Lenguajes de Programación de más bajo nivel como C o C++, que es el que utiliza Arduino, por ejemplo. (Morales, 2020)

Hay algunas características que MicroPython tiene y es lo que lo hace único y diferente de otros sistemas embebidos:

- Tiene una REPL Interactiva (Read-Eval-Print Loop Por sus siglas en Ingles). Que es un pequeño programa que lee e interpreta los comandos del usuario, los evalúa y después imprime el resultado. Esto permite conectar alguna tarjeta (microcontrolador que soporte Python) y esta tiene que ejecutar el código sin necesidad de compilar ni cargar el programa.
- Muchas librerías. Así como el lenguaje de programación Python cuenta con un sin fin de librerías para la ejecución de tareas, MicroPython también viene bien cargado con bastantes paquetes para ahorrar trabajo. Es posible ejecutar análisis de datos JSON desde un servicio web, búsqueda de texto en expresiones regulares o hasta levantar un Socket dentro de una red tan solo con las funciones ya precargadas.
- Extensibilidad. Para los usuarios avanzados de MicroPython, pueden extender de Python a funciones de más bajo nivel como C o C++, pudiendo mezclar códigos que requieran de ejecución más rápida a bajo nivel con MicroPython que es de más alto nivel. (Morales, 2020).

A continuación, se presenta el diagrama bloques del ESP32.

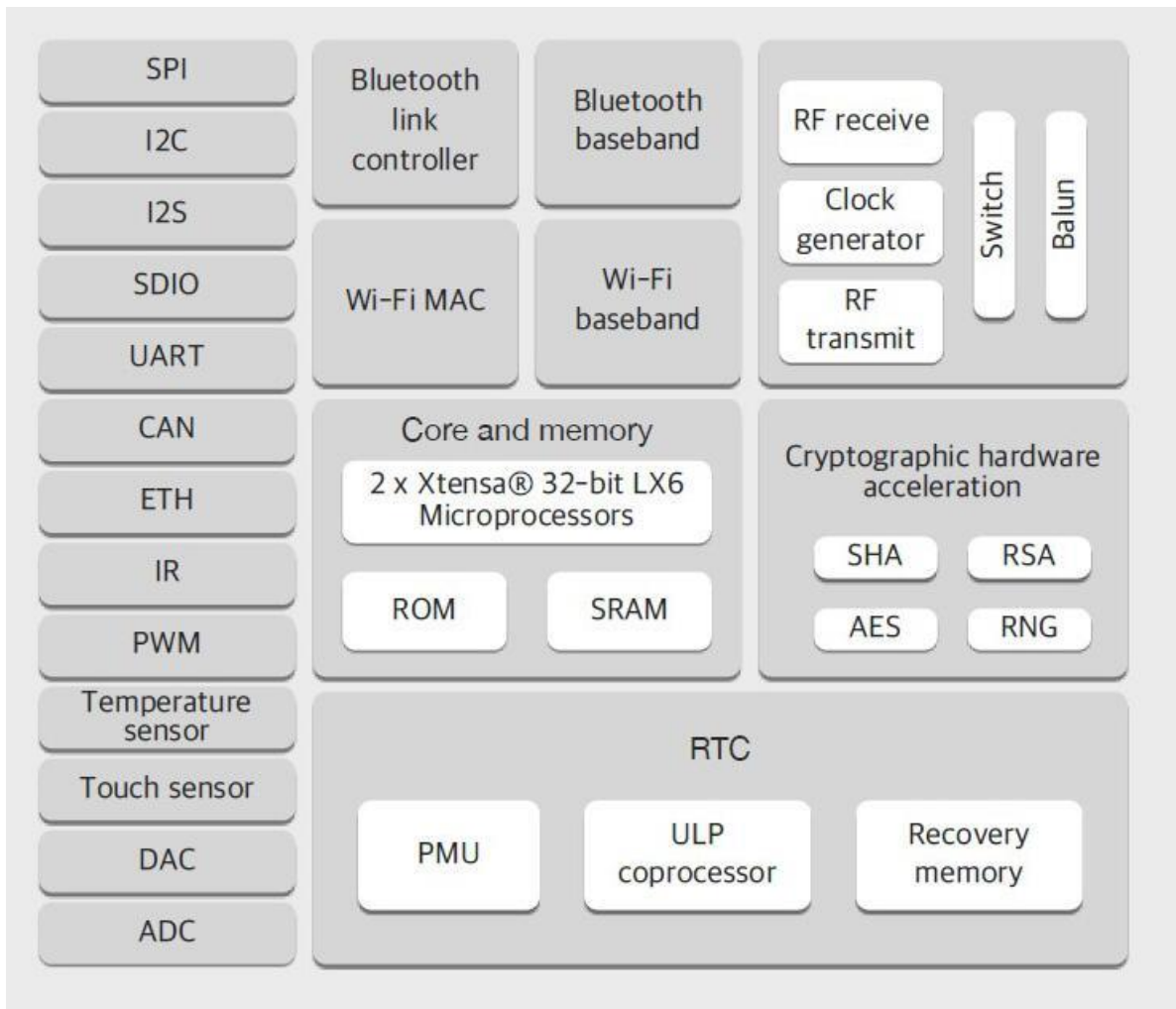


Ilustración 14. Diagrama bloques del ESP32.

El ESP32 es capaz de funcionar de forma fiable en entornos industriales, con una temperatura de funcionamiento de -40°C a $+125^{\circ}\text{C}$. Alimentado por circuitos de calibración avanzados.

Está altamente integrado con interruptores de antena incorporados, balun de RF, amplificador de potencia, amplificador de recepción de bajo ruido, filtros y módulos de administración de energía. ESP32 agrega funcionalidad y versatilidad invaluable a sus aplicaciones con requisitos mínimos de placa de circuito impreso (PCB).

Diseñado para dispositivos móviles, dispositivos electrónicos portátiles y aplicaciones de IoT, el ESP32 logra un consumo de energía ultra bajo con una combinación de varios tipos de software patentado. El ESP32 también incluye características de vanguardia, como la sincronización de reloj de grano fino, varios modos de potencia y escalado dinámico de potencia.

ESP32 puede funcionar como un sistema independiente completo o como un dispositivo esclavo de una MCU host, lo que reduce la sobrecarga de la pila de comunicación en el procesador de la aplicación principal. ESP32 puede interactuar con otros sistemas para proporcionar funcionalidad Wi-Fi y Bluetooth a través de sus interfaces SPI / SDIO o I2C / UART. ("ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems", 2020)

A continuación, se muestra los pines del ESP32.

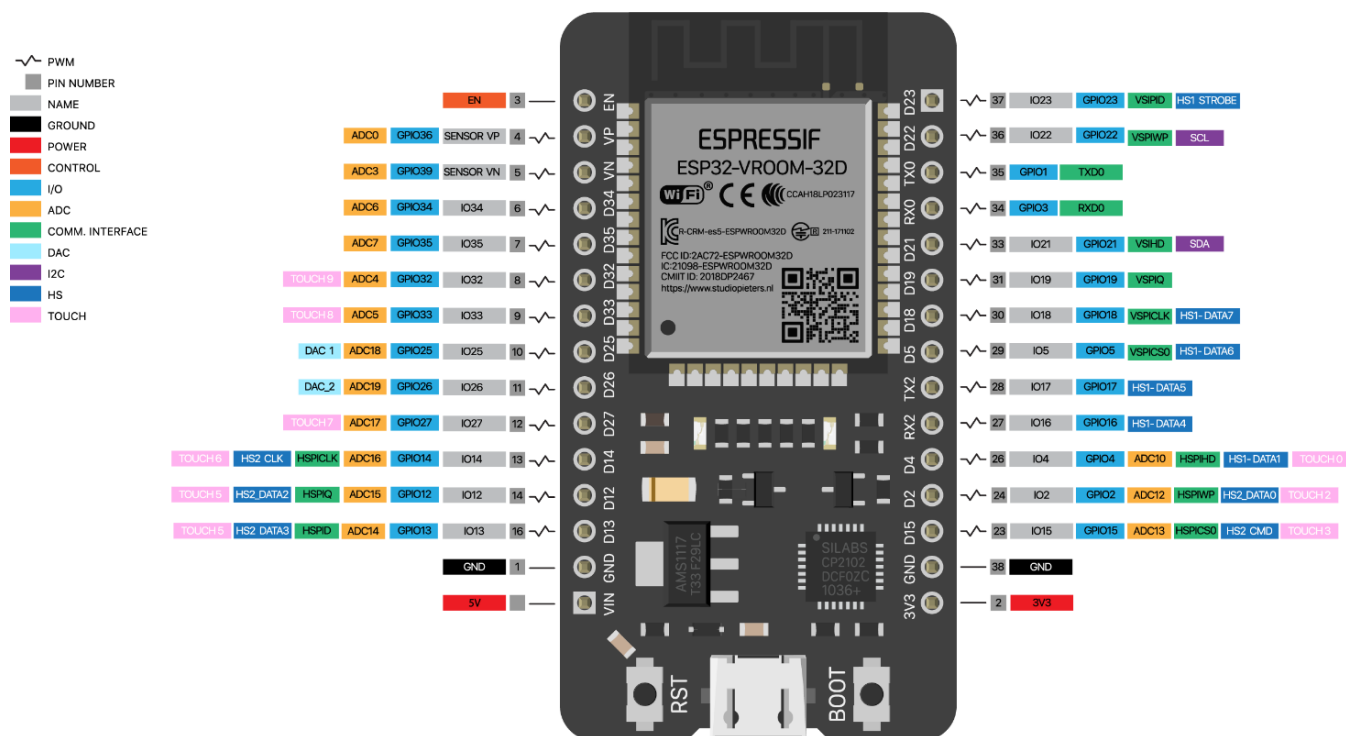


Ilustración 15. Pinout del ESP32. (OLED, 2020)

Por otro lado, se utilizó el microcontrolador PSoC 5LP.

PSoC® 5LP es sistema integrado en chip programable, que integra periféricos analógicos y digitales configurables, memoria y un microcontrolador en un solo chip. La arquitectura PSoC 5LP aumenta el rendimiento a través de:

- Arm Cortex-M3 core de 32 bits más controlador DMA y procesador de filtro digital, hasta 80 MHz.
- Potencia ultra baja con el rango de voltaje más amplio de la industria.
- Los periféricos digitales y analógicos programables permiten funciones personalizadas.
- Enrutamiento flexible de cualquier función periférica analógica o digital a cualquier pin.

Los dispositivos PSoC (Programmable System on Chip) emplean una arquitectura de sistema en chip altamente configurable para el diseño de control integrado. Integran circuitos analógicos y digitales configurables, controlados por un microcontrolador en chip. Un sólo dispositivo PSoC puede integrar hasta 100 funciones periféricas digitales y analógicas, lo que reduce el tiempo de diseño, el espacio en la placa, el consumo de energía y el costo del sistema al tiempo que mejora la calidad del sistema.

Sus características de funcionamiento son las siguientes.

- Rango de voltaje: 1.71 a 5.5 V, hasta 6 dominios de potencia.
- Rango de temperatura (ambiente) -40 a 85°C .
- Partes de temperatura extendida: -40 a 105°C .
- Funcionamiento de CC a 80 MHz.
- Modos de energía:
 - Modo activo 3,1 mA a 6 MHz y 15,4 mA a 48 MHz.
 - Modo de suspensión de 2 μA .
 - Modo de hibernación 300-nA con retención de RAM.
- Aumente el regulador desde una entrada de 0.5 V hasta una salida de 5 V.
- Actuación.

- CPU Arm Cortex-M3 de 32 bits, 32 entradas de interrupción.
- Controlador de acceso directo a memoria (DMA) de 24 canales.
- Procesador de filtro digital de punto fijo (DFB) de 24 bits y 64 tomas.
- Memoria.
 - Flash de programa de hasta 256 KB, con caché y funciones de seguridad.
 - Hasta 32 KB de flash adicional para código de corrección de errores (ECC).
 - Hasta 64 KB de RAM.
 - EEPROM de 2 KB.

En la siguiente ilustración se muestra el pinout del microcontrolador PSoC 5LS.

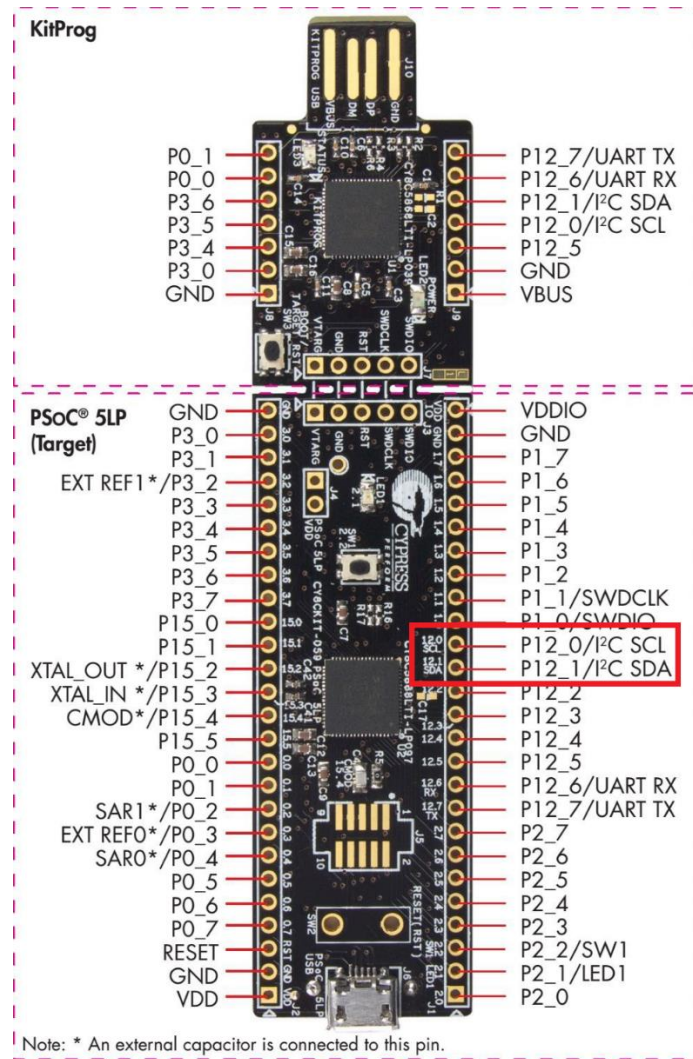


Ilustración 16. Pinout del PSoC 5.

El PSoC es programado con PSoC Creator.

PSoC Creator es un entorno de diseño integrado (IDE) que permite la edición, compilación y depuración simultánea de hardware y firmware de sistemas PSoC y FM0+. Las aplicaciones se crean mediante captura esquemática y más de 150 componentes periféricos pre-verificados y listos para producción.

Los componentes son periféricos analógicos y digitales representados por un símbolo que los usuarios arrastran y sueltan en sus diseños y configuran para adaptarse a una amplia gama de requisitos de aplicación. Cada componente en el rico catálogo de componentes Cypress de señal mixta está configurado con un cuadro de diálogo de personalización e incluye un conjunto completo de bibliotecas API generadas dinámicamente. Después de configurar todos los periféricos, el

firmware se puede escribir, compilar y depurar dentro de PSoC Creator o exportarse a IDE de terceros líderes como IAR Embedded Workbench®, Arm® Microcontroller Development Kit y Eclipse TM.

Los sistemas PSoC y FM0 + tienen una energía optimizada más allá de un MCU típico porque PSoC Creator optimiza los diseños para habilitar solo la funcionalidad requerida. Los usuarios pueden incluso crear componentes personalizados utilizando diagramas de máquinas de estado o Verilog para optimizar aún más el uso de hardware y energía.

PSoC Creator es un IDE gratuito que corre en Windows e incluye:

- Diseño de hardware con captura esquemática completa y herramienta de cableado fácil de usar.
- Más de 150 componentes previamente verificados y listos para producción.
- Biblioteca de comunicaciones completa que incluye I2C, USB, UART, SPI, CAN, LIN y Bluetooth Low Energy.
- Periféricos digitales con potentes herramientas de configuración gráfica.
- Amplio soporte de cadena de señal analógica con amplificadores, filtros, ADC y DAC.
- Bibliotecas API generadas dinámicamente.
- Compilador de código fuente C gratuito sin limitaciones de tamaño de código.
- Editor de fuentes integrado con diagnósticos en línea, autocompletar y fragmentos de código.
- Depurador incorporado.

La familia CY8C58LP proporciona bloques configurables de circuitos analógicos, digitales y de interconexión alrededor de un subsistema de CPU. La combinación de una CPU con un subsistema analógico flexible, subsistema digital, enrutamiento y E / S permite un alto nivel de integración en una amplia variedad de aplicaciones médicas, industriales y de consumo. ("cypress", 2020)

A continuación, se presenta la arquitectura del dispositivo.

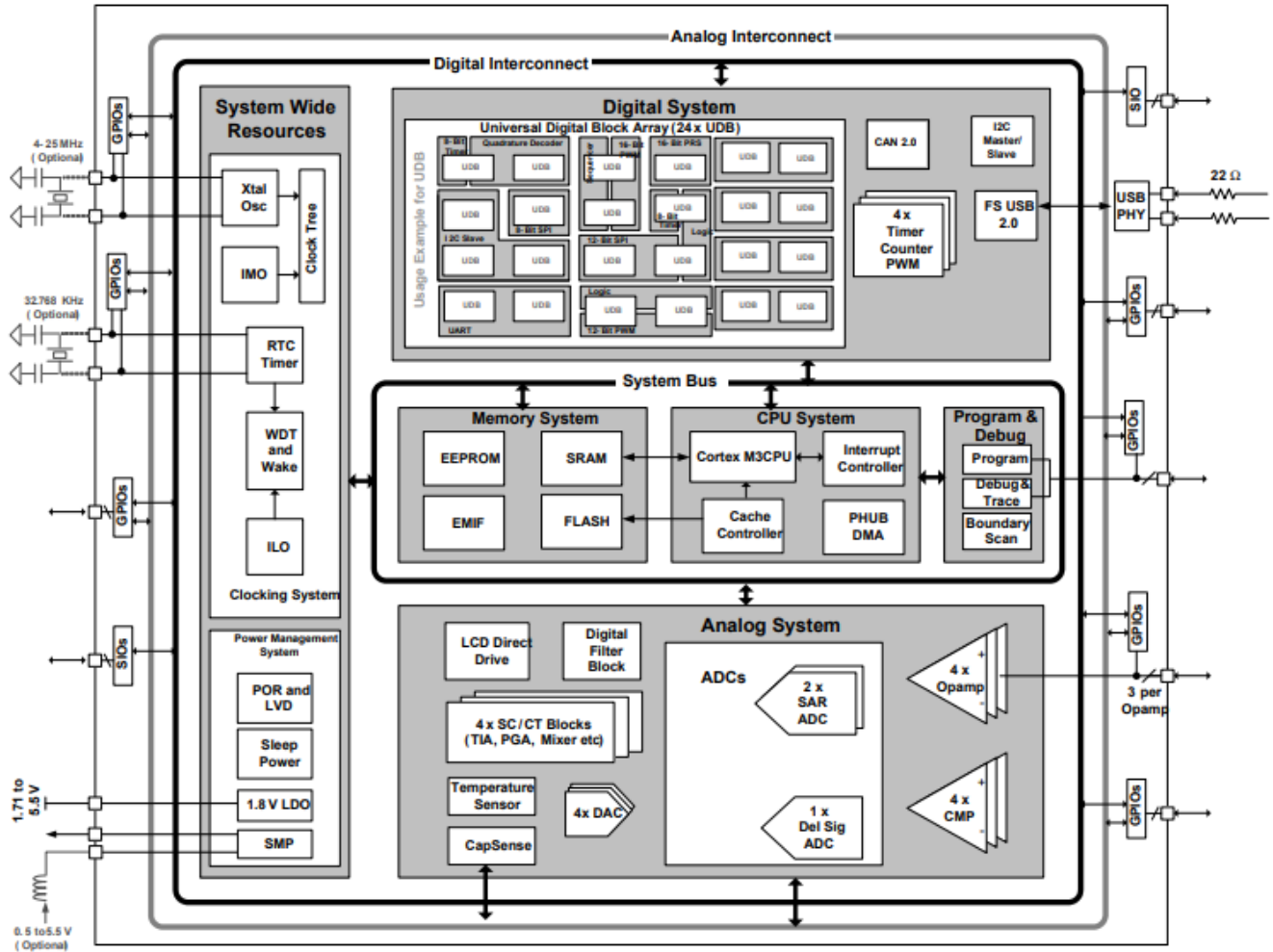


Ilustración 17. Diagrama de bloques del PSoC 5. ("cypress", 2020)

La comunicación se realizó utilizando el puerto UART con un ESP32. Esto es requerido para enviar los datos que lee el sensor a la plataforma ThingsBoard, ya que el PSoC no cuenta con un módulo wifi.

El controlador del Receptor / Transmisor Asíncrono Universal (UART) es el componente clave del subsistema de comunicaciones en serie de una computadora. El UART toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo UART vuelve a ensamblar los bits en bytes completos.

La transmisión en serie se usa comúnmente con módems y para la comunicación no en red entre computadoras, terminales y otros dispositivos. ("Serial and UART Tutorial", 2021)

Existen dos formas de transmitir los datos, en paralelo y en serie.

Las interfaces paralelas transfieren múltiples bits simultáneamente. Por lo general, requieren barras (buses) de datos, que se transmiten a través de ocho, dieciséis o más cables. Los datos se transfieren en amplios oleajes de 1s y 0s.

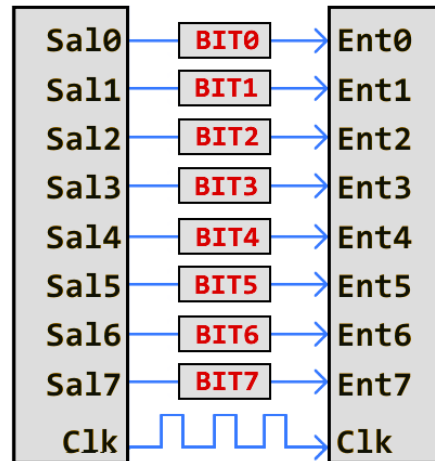


Ilustración 18. Transmisión de datos en paralelo.

las interfaces serie transmiten sus datos un bit a la vez. Estas interfaces pueden operar con tan solo un cable, por lo general nunca más de cuatro.

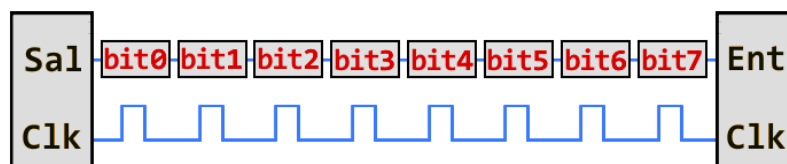


Ilustración 19. Transmisión de datos en serie.

La comunicación paralela ciertamente tiene sus beneficios. Es rápida, directa y relativamente fácil de implementar. Pero requiere muchas más líneas de entrada/salida (E/S). ("¿Qué es la comunicación serie? | Robots Didácticos", 2021)

El baud es una medida de la velocidad de transmisión en la comunicación asincrónica. Debido a los avances en la tecnología de comunicación por módem, este término se usa incorrectamente con frecuencia al describir las velocidades de datos en dispositivos más nuevos.

Tradicionalmente, una velocidad en baudios representa la cantidad de bits que realmente se envían a través de los medios, no la cantidad de datos que se mueven realmente de un dispositivo DTE a otro. El recuento en baudios incluye los bits de sobrecarga Start, Stop y Parity que son generados por el UART emisor y eliminados por el UART receptor. Esto significa que las palabras de datos de siete bits en realidad necesitan 10 bits para transmitirse por completo. Por lo tanto, un módem capaz de mover 300 bits por segundo de un lugar a otro normalmente solo puede mover 30 palabras de 7 bits si se usa la paridad y hay un bit de inicio y parada. ("Serial and UART Tutorial", 2021)

Si se utilizan palabras de datos de 8 bits y también se utilizan bits de paridad, la tasa de datos cae a 27,27 palabras por segundo, porque ahora se necesitan 11 bits para enviar las palabras de ocho bits, y el módem aún envía solo 300 bits por segundo.

Bauds	Transmission speed			Real transmission speed	
	Bit/s	Bit duration	Speed	Speed	Byte duration
50 Bd	50 bits/s	20.000 ms	6.25 bytes/s	5 bytes/s	200.000 ms
75 Bd	75 bits/s	13.333 ms	9.375 bytes/s	7.5 bytes/s	133.333 ms
110 Bd	110 bits/s	9.091 ms	13.75 bytes/s	11 bytes/s	90.909 ms
134 Bd	134 bits/s	7.463 ms	16.75 bytes/s	13.4 bytes/s	74.627 ms
150 Bd	150 bits/s	6.667 ms	18.75 bytes/s	15 bytes/s	66.667 ms
200 Bd	200 bits/s	5.000 ms	25 bytes/s	20 bytes/s	50.000 ms
300 Bd	300 bits/s	3.333 ms	37.5 bytes/s	30 bytes/s	33.333 ms
600 Bd	600 bits/s	1.667 ms	75 bytes/s	60 bytes/s	16.667 ms
1200 Bd	1200 bits/s	833.333 µs	150 bytes/s	120 bytes/s	8.333 ms
1800 Bd	1800 bits/s	555.556 µs	225 bytes/s	180 bytes/s	5.556 ms
2400 Bd	2400 bits/s	416.667 µs	300 bytes/s	240 bytes/s	4.167 ms
4800 Bd	4800 bits/s	208.333 µs	600 bytes/s	480 bytes/s	2.083 ms
9600 Bd	9600 bits/s	104.167 µs	1200 bytes/s	960 bytes/s	1.042 ms
19200 Bd	19200 bits/s	52.083 µs	2400 bytes/s	1920 bytes/s	520.833 µs
28800 Bd	28800 bits/s	34.722 µs	3600 bytes/s	2880 bytes/s	347.222 µs
38400 Bd	38400 bits/s	26.042 µs	4800 bytes/s	3840 bytes/s	260.417 µs
57600 Bd	57600 bits/s	17.361 µs	7200 bytes/s	5760 bytes/s	173.611 µs
76800 Bd	76800 bits/s	13.021 µs	9600 bytes/s	7680 bytes/s	130.208 µs
115200 Bd	115200 bits/s	8.681 µs	14400 bytes/s	11520 bytes/s	86.806 µs
230400 Bd	230400 bits/s	4.340 µs	28800 bytes/s	23040 bytes/s	43.403 µs
460800 Bd	460800 bits/s	2.170 µs	57600 bytes/s	46080 bytes/s	21.701 µs
576000 Bd	576000 bits/s	1.736 µs	72000 bytes/s	57600 bytes/s	17.361 µs
921600 Bd	921600 bits/s	1.085 µs	115200 bytes/s	92160 bytes/s	10.851 µs

Ilustración 20. Valores típicos de transmisión. ("Vitesse de transmission les plus utilisées", 2021)

La fórmula para convertir bytes por segundo en una velocidad en baudios y viceversa era simple hasta que aparecieron los módems correctores de errores. Estos módems reciben el flujo de bits en serie del UART en la computadora host (incluso cuando se utilizan módems internos, los datos todavía se serializan con frecuencia) y vuelven a convertir los bits en bytes. Estos bytes luego se combinan en paquetes y se envían a través de la línea telefónica usando un método de transmisión sincrónica. Esto significa que los bits de parada, inicio y paridad agregados por el UART en el DTE (la computadora) fueron eliminados por el módem antes de la transmisión por el módem emisor. Cuando estos bytes son recibidos por el módem remoto, el módem remoto agrega bits de inicio, parada y paridad a las palabras, los convierte a un formato en serie y luego los envía al UART receptor en la computadora remota.

La razón por la que se realizan todas estas conversiones adicionales es para que los dos módems puedan realizar la corrección de errores, lo que significa que el módem receptor puede pedirle al módem emisor que reenvíe un bloque de datos que no se recibió con la suma de comprobación correcta. Esta verificación es manejada por los módems, y los dispositivos DTE generalmente ignoran que el proceso está ocurriendo. ("Serial and UART Tutorial", 2021)

Al dividir los bits de inicio, parada y paridad, los bits de datos adicionales que los dos módems deben compartir entre sí para realizar la corrección de errores se ocultan en su mayoría de la tasa de transmisión efectiva que ve el equipo DTE emisor y receptor. Por ejemplo, si un módem envía diez palabras de 7 bits a otro módem sin incluir los bits de inicio, parada y paridad, el módem emisor podrá agregar 30 bits de su propia información que el módem receptor puede utilizar para corregir errores. sin afectar la velocidad de transmisión de los datos reales.

El uso del término baudios se confunde aún más con los módems que realizan compresión. Una sola palabra de 8 bits transmitida por la línea telefónica podría representar una docena de palabras que se transmitieron al módem emisor. El módem receptor expandirá los datos a su contenido original y pasará esos datos al DTE receptor.

Los módems modernos también incluyen búfer que permiten que la velocidad a la que los bits se mueven a través de la línea telefónica (DCE a DCE) sea diferente a la velocidad a la que se mueven los bits entre el DTE y el DCE en ambos extremos de la conversación. Normalmente, la velocidad entre el DTE y el DCE es mayor que la velocidad del DCE al DCE debido al uso de compresión por los módems.

Como el número de bits necesarios para describir un byte varió durante el viaje entre las dos máquinas más las diferentes velocidades de bits por segundo que se utilizan

en los enlaces DTE-DCE y DCE-DCE, el uso del término Baudios para describir la velocidad de comunicación general causa problemas y puede tergiversar la verdadera velocidad de transmisión. Por lo tanto, Bits por segundo (bps) es el término correcto para describir la velocidad de transmisión observada en la interfaz DCE a DCE y Baudios o Bits por segundo son términos aceptables para usar cuando se realiza una conexión entre dos sistemas con una conexión por cable, o si se está utilizando un módem que no realiza corrección de errores o compresión. ("Serial and UART Tutorial", 2021)

Los módems modernos de alta velocidad (2400, 9600, 14,400 y 19,200bps) en realidad todavía funcionan a 2400 baudios o menos, o más exactamente, a 2400 símbolos por segundo. El módem de alta velocidad puede codificar más bits de datos en cada símbolo utilizando una técnica llamada Constellation Stuffing, razón por la cual la tasa efectiva de bits por segundo del módem es mayor, pero el módem continúa funcionando dentro del ancho de banda de audio limitado que el teléfono. proporciona el sistema. Los módems que funcionan a velocidades de 28.800 y superiores tienen velocidades de símbolo variables, pero la técnica es la misma. ("Serial and UART Tutorial", 2021)

CAPÍTULO 3. DESARROLLO DE ACTIVIDADES.

El presente proyecto se realizó bajo la supervisión y asesoría de los doctores Guillermo Barrios del Valle y Guillermo Ramírez Zúñiga y la asesoría del MC. José Efraín Ruiz Domínguez, catedrático del Instituto Tecnológico de Zacatepec.

Como ya se mencionó con anterioridad, el principal objetivo de este trabajo es realizar el diseño y construcción, así como realizar las pruebas de funcionamiento de un dispositivo para medir el flujo de calor en muros usando tecnologías libres.

Se realizó la programación para leer voltajes en el software MicroPython haciendo uso de un microcontrolador ESP32 y posteriormente se realizó el diseño del circuito de amplificación puesto que el ESP32 no tiene la resolución suficiente para detectar al sensor de flujo de calor.

3.1. Especificaciones y Requerimientos.

A continuación, se describen las principales especificaciones y requerimientos necesarios para cumplir con los objetivos.

Como especificaciones, se tomaron amplificar el voltaje del sensor de flujo de calor Fluxteq en tres rangos (-5 a $5 \frac{W}{m^2}$, -10 a $10 \frac{W}{m^2}$, -20 a $20 \frac{W}{m^2}$) para que pueda ser reconocido por un ESP32 y que publique los datos en Thingsboard.

3.2. Equipo utilizado.

A continuación, se menciona el material utilizado para el proyecto utilizando el microcontrolador ESP32, así como el material utilizado realizando el cambio de microcontrolador al PSoC 5.

Para el microcontrolador ESP32:

- ESP32.
- Software MicroPython.
- 1 amplificador operacional LM741 en forma de sumador inversor.
- 2 amplificadores de instrumentación AD620.
- 2 capacitores de 0.1μ .
- 3 potenciómetro de $100k\Omega$.
- 5 resistencias de $100k\Omega$.
- 1 resistencia de 47Ω .

- Fuente bipolar.

Para el microcontrolador PSoC:

- PSoC 5.
- Software para PSoC.

3.3. Diseño del circuito y cálculos.

Para el ESP32 se diseña un circuito de amplificación, dado que su resolución no es la suficiente para reconocer al sensor de flujo de calor. A continuación, se describen los cálculos correspondientes.

Sabiendo que el sensor cuenta con una sensibilidad nominal de aprox. 9,0 mV / (W / cm ^ 2) y que el microcontrolador tiene 10 bits de resolución, se tiene la necesidad de diseñar un circuito que amplifique la señal del voltaje.

Se indicó que realizara el diseño para tres rangos distintos de flujo de calor, por lo que a continuación, se presentan los cálculos pertinentes.

Teniendo un flujo de calor en un rango de -5 a 5 $\frac{W}{m^2}$ y una sensibilidad del sensor de $0.9 \times 10^{-6} \frac{V m^2}{W}$, el voltaje representado se expresa de la siguiente manera:

$$(\pm 5 \frac{W}{m^2})(0.9 \times 10^{-6} \frac{V m^2}{W}) = \pm 4.5\mu V$$

Debido a que el microcontrolador no es capaz de leer un voltaje de magnitud tan reducida, se utiliza un amplificador operacional “AD620”, el cual aumenta el voltaje inicial. Sabiendo que el amplificador operacional antes mencionado aumenta el voltaje hasta mil veces, es decir, puede tener una ganancia (G) de hasta mil y con ello, se calcula la resistencia (RG) que necesita; se realizó el cálculo correspondiente, lo que resulta:

$$(4.5\mu V)(1000) = 0.0045V$$

$$G=1000$$

$$R_G = \frac{49.4K\Omega}{1000 - 1} = 49.4\Omega$$

De acuerdo al resultado anterior, se puede observar que el voltaje es insuficiente para que sea detectado por el microcontrolador, debido a que cuenta con un ADC de 10 bits de resolución y trabaja con un voltaje de alimentación de 3.3V, con ayuda de la siguiente ecuación se puede determinar el voltaje mínimo reconocido:

$$\frac{3.3}{1024} = 3.22 \text{ mV}$$

Por lo que es necesario utilizar un segundo amplificador operacional. A continuación, se muestran los cálculos para obtener la ganancia requerida, así como el valor de resistencia necesario. Cabe señalar que debido a que, en la tercera etapa, se utiliza un LM741 el cual funciona de forma de sumador no inversor por lo que se tiene que dividir a la mitad el voltaje con el cual funciona el ESP32.

$$G = \frac{1.65V}{0.0045V} = 166.6$$

$$R_G = \frac{49.4K\Omega}{166.6 - 1} = 135.12\Omega$$

En la ilustración 21 se muestra gráficamente el voltaje representado por el flujo de calor, en donde se observa el voltaje con el que se alimenta el ESP32, la línea roja marca la división del voltaje positivo y negativo.

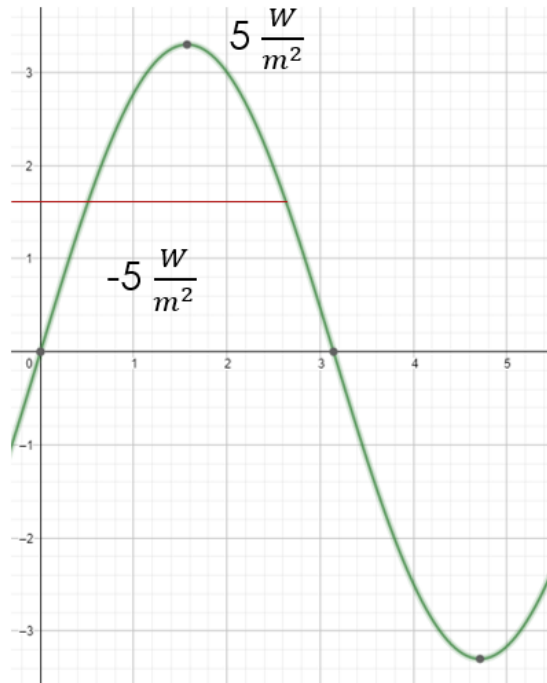


Ilustración 21. voltaje representado por el flujo de calor.

De igual manera, se calculó teniendo un flujo de calor en un rango de -10 a $10 \frac{W}{m^2}$ el voltaje representado se expresa de la siguiente manera:

$$\left(\pm 10 \frac{W}{m^2}\right) \left(0.9 \times 10^{-6} \frac{V \cdot m^2}{W}\right) = \pm 9 \mu V$$

Sabiendo que el amplificador operacional antes mencionado aumenta el voltaje hasta mil veces, se realizó el cálculo correspondiente, lo que resultó:

$$(9 \mu V) (1000) = 0.009 V$$

$$G=1000$$

$$R_G = \frac{49.4 K\Omega}{1000 - 1} = 49.4 \Omega$$

A continuación, se muestran los cálculos para obtener la ganancia requerida, así como el valor de resistencia necesario:

$$G = \frac{1.65V}{0.009V} = 183.3$$

$$R_G = \frac{49.4K\Omega}{183.3 - 1} = 269.5\Omega$$

En la siguiente ilustración se muestra gráficamente el voltaje representado por el flujo de calor, en donde se observa el voltaje con el que se alimenta el ESP32, la línea roja marca la división del voltaje positivo y negativo.

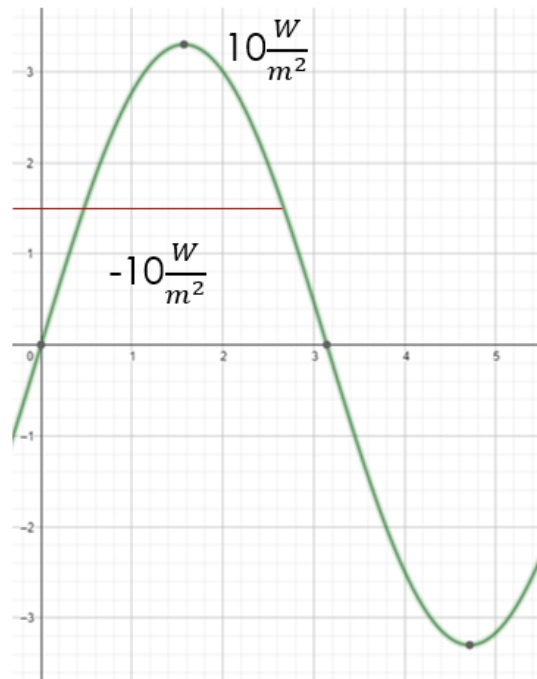


Ilustración 22. voltaje representado por el flujo de calor.

Finalmente, teniendo un flujo de calor en un rango de -20 a $20 \frac{W}{m^2}$ el voltaje representado se expresa de la siguiente manera:

$$(\pm 20 \frac{W}{m^2})(0.9 \times 10^{-6} \frac{V m^2}{W}) = \pm 18 \mu V$$

Sabiendo que el amplificador operacional antes mencionado aumenta el voltaje hasta 1000 veces, se realizó el cálculo correspondiente, lo que resultó:

$$(18 \mu V) (1000) = 0.018 V$$

$$G=1000$$

$$R_G = \frac{49.4 K\Omega}{1000 - 1} = 49.4 \Omega$$

A continuación, se muestran los cálculos para obtener la ganancia requerida, así como el valor de resistencia necesario:

$$G = \frac{1.65 V}{0.018 V} = 91.6$$

$$R_G = \frac{49.4 K\Omega}{91.6 - 1} = 545.2 \Omega$$

En la siguiente ilustración se muestra gráficamente el voltaje representado por el flujo de calor, en donde se observa el voltaje con el que se alimenta el ESP32, la línea roja marca la división del voltaje positivo y negativo.

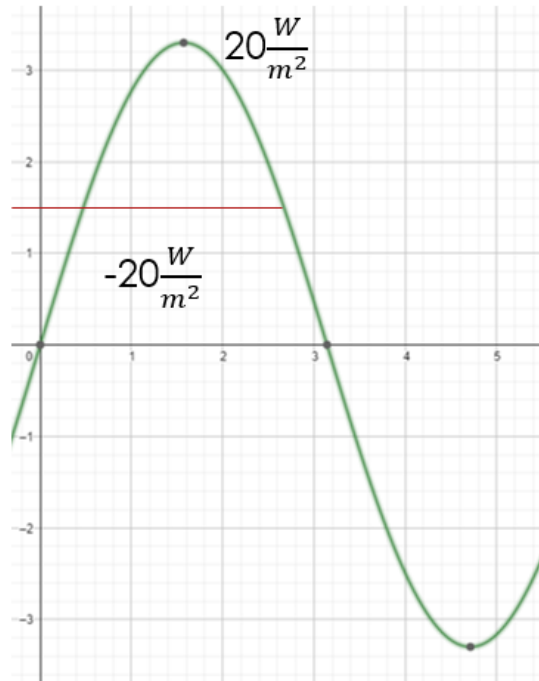


Ilustración 23. voltaje representado por el flujo de calor.

Para que el circuito sea capaz de leer valores de flujo de calor, tanto positivos como negativos, se utiliza un amplificador operacional LM741, el cual permite realizar tal acción.

Con base a lo anteriormente calculado, se realizó el diagrama del circuito eléctrico del acondicionamiento de la señal.

En la siguiente ilustración se muestra el circuito de ganancia, el cual amplifica dos veces el voltaje para que pueda ser reconocido por el microcontrolador.

Con base a los cálculos realizados con anterioridad, se llevó a cabo la realización del diseño del circuito de acondicionamiento que a continuación se muestra.

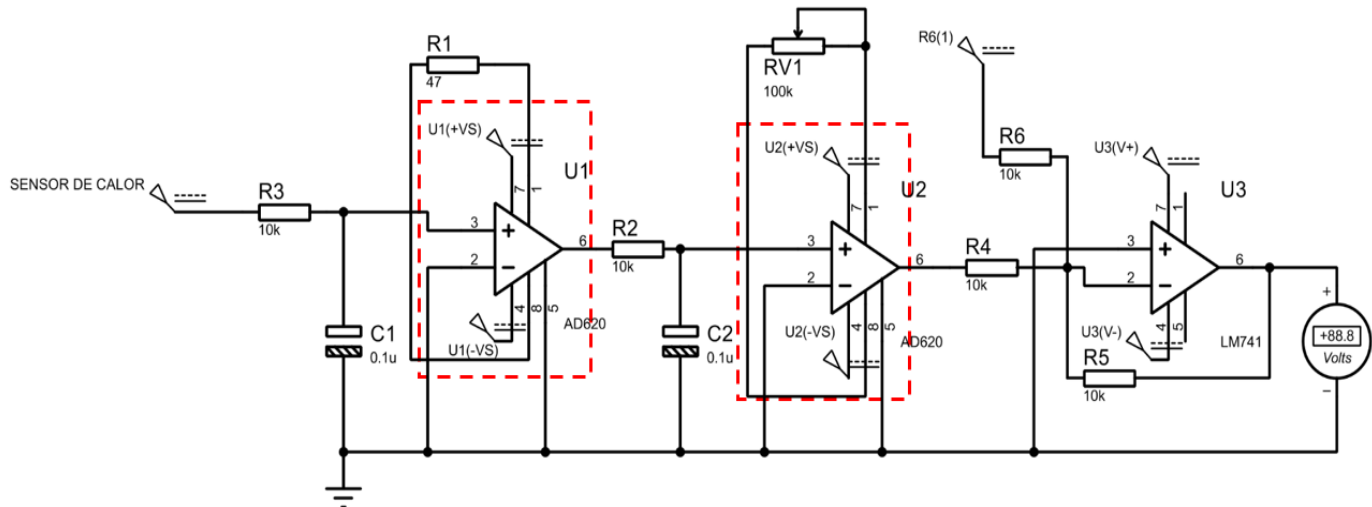


Ilustración 24. Diagrama de conexión del acondicionamiento de la señal.

En la ilustración 24 se muestra el diagrama del circuito de amplificación, en el cual se resaltan las dos etapas de amplificación.

3.4. Desarrollo de la programación.

A continuación, presenta el diagrama de flujo del programa del sensor de flujo de calor, en el cual se muestra la estructura desde que inicia el programa. Una vez que inicia, se configuran los pines necesarios para su funcionamiento, seguido de esto, se requiere el convertidor analógico/digital (ADC), posteriormente, el programa entra a un ciclo en el cual “llama” a la medición correspondiente, seguido de tomar la lectura del sensor y realizar la conversión necesaria para finalmente, publicar los resultados esperados.

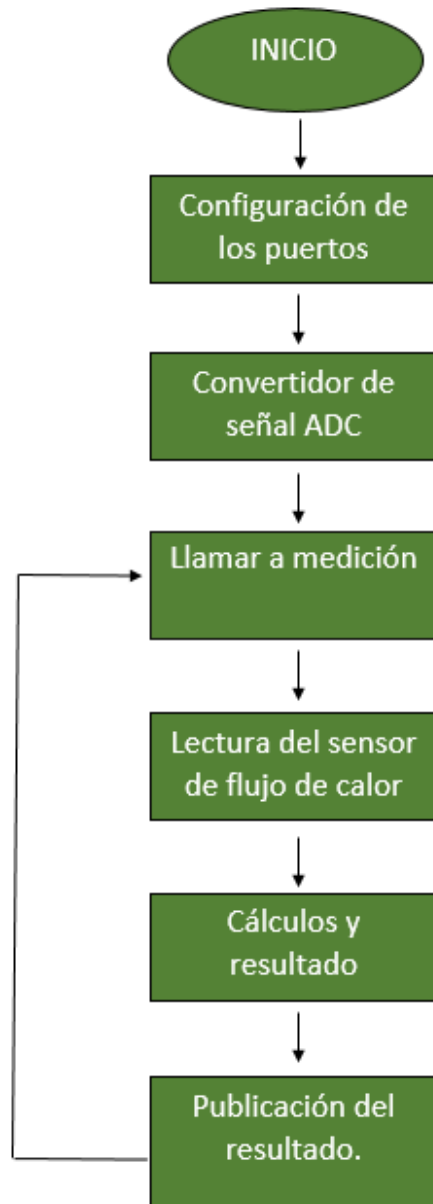


Ilustración 25. Diagrama de flujo del programa.

A continuación, se presenta el código principal “main” de programación en MicroPython para el microcontrolador ESP32, donde se observan las librerías utilizadas en las líneas 1-8, posteriormente las líneas 10 y 11 corresponden a la red y contraseña del modem donde es conecta a internet, las líneas 13 y 14 corresponden los datos de la plataforma ThingsBoard, así como la etiqueta “label”. Posteriormente, el programa entra a un ciclo en el que lee el voltaje para posteriormente convertirlo a unidades de flujo de calor. Finalmente publica el resultado en ThingsBoard cada 5 segundos.



```
1 # from wifi import do_connect
2 import time
3 # import network
4 from function import *
5 from wifi import *
6 from machine import UART
7 from machine import ADC
8 from machine import Pin
9
10 red = 'cb61c0'
11 clave = '280865312'
12
13 unique_id = '66f5d0f0-5de2-11eb-9c3f-d1ead9980bc3'
14 token = 'tDzaQ1x1jJoWdk5l0rtg'
15
16 label = 'Flujo de calor'
17
18
19 do_connect(red,clave)
20
21
22 while True:
23
24     adc = ADC(Pin(32))
25     adc.atten(ADC.ATTN_11DB)
26     voltaje=adc.read()
27
28     resultado = ((3.3*voltaje/1024)/1000000)/0.00000144
29     valor = resultado
30     data = {'label': valor}
31     publish_thingsboard(red,clave,token, unique_id,data)
32     time.sleep(5)
```

Ilustración 26. Código “main” en MicroPython para leer el flujo de calor y publicarlo en ThingsBoard.

A continuación, se presenta la programación para PSoC 5.

Este programa se divide en tres secciones, las cuales son: el esquemático, el código y la configuración de los pines.

El esquemático es donde se colocarán los componentes electrónicos necesarios para comenzar a configura la tarjeta.

El código se realiza en el lenguaje C.

Es la ventana donde se configuran y se asignan los pines analógicos y digitales que se han colocado en el esquemático.

A continuación, se presenta la configuración que fue utilizada, según lo requerido, así como los componentes necesarios para realizar la programación de este microcontrolador.

Primero se configura el ADC que en este caso se optó por utilizar un Delta-Sigma. Este ADC cuenta con una resolución máxima de 20 bits, por lo cual no es necesario utilizar un amplificador operacional

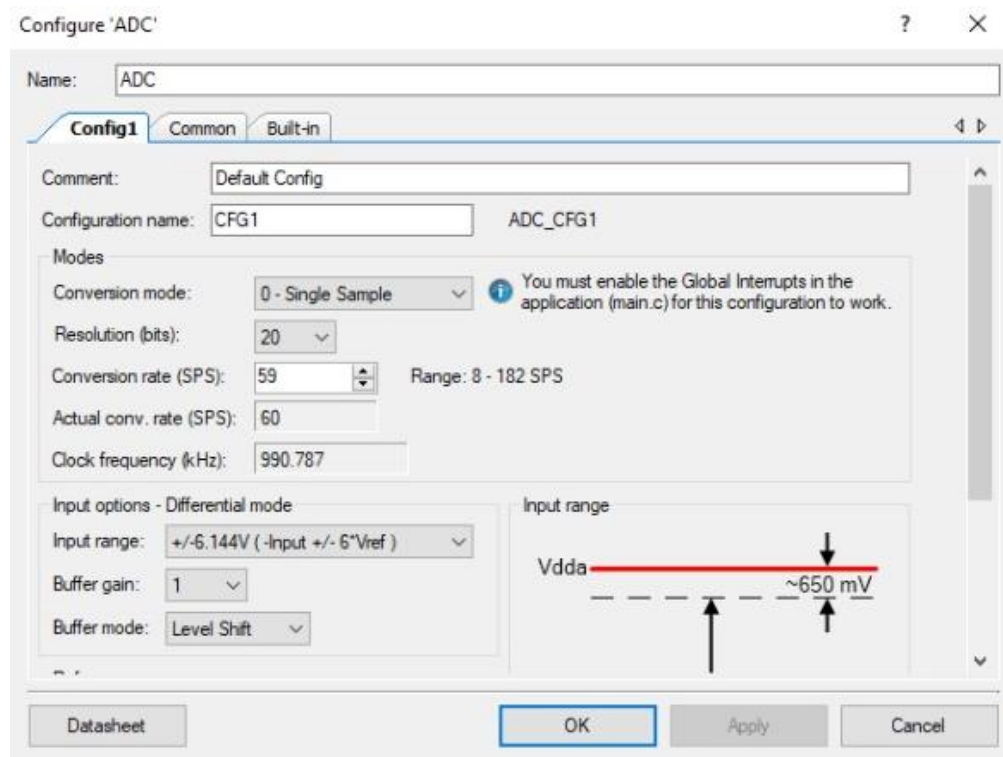


Ilustración 27. Configuración de ADC.

La entrada diferencial es para que el dispositivo sea capaz de leer voltajes negativos como se muestra en la siguiente ilustración.

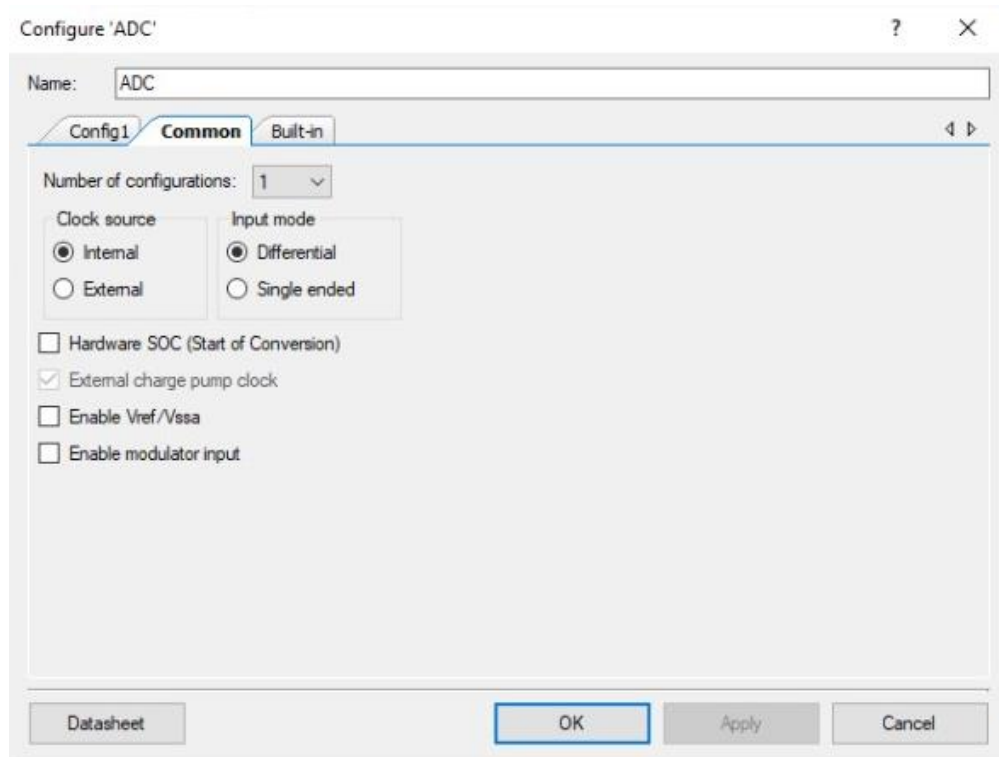


Ilustración 28. Configuración de ADC.

Puesto que se requiere de más de un sensor, se optó por utilizar multiplexor, (mismo que es incluido en el software) que multiplique el número de entradas, las entradas diferenciales sirven para que los sensores puedan leer voltajes positivos y negativos.

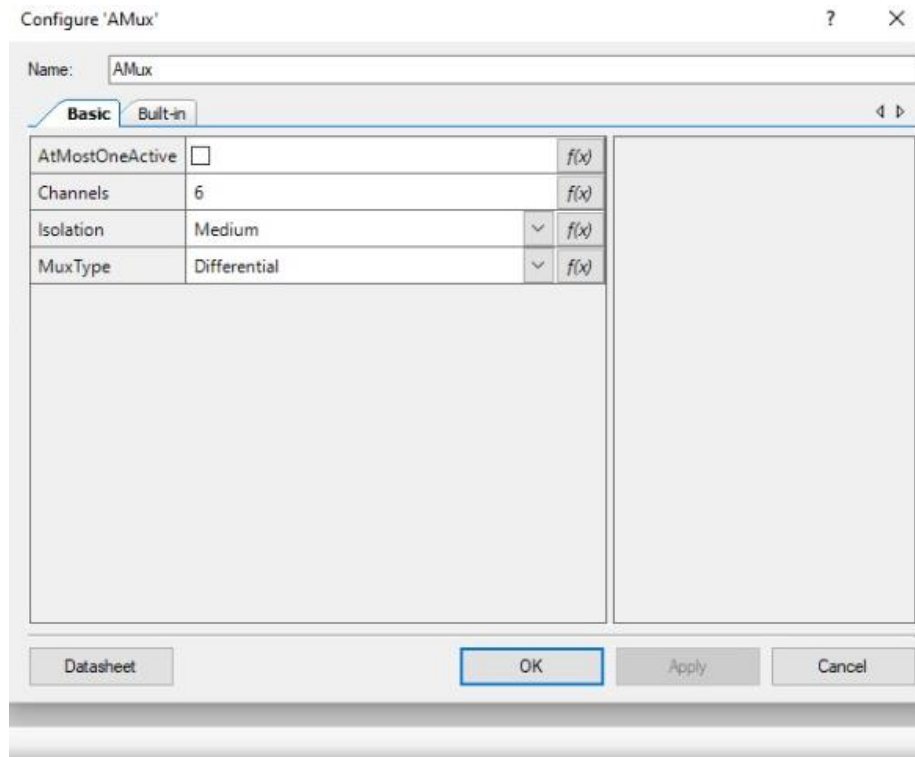


Ilustración 29. Configuración del multiplexor (AMux).

Los pines se configuran como se muestra en la siguiente ilustración, se elige digitales o analógicos según sea el caso y se nombra la variable.

Este procedimiento se realiza en todos los pines que se van a utilizar en el esquemático para cada módulo.

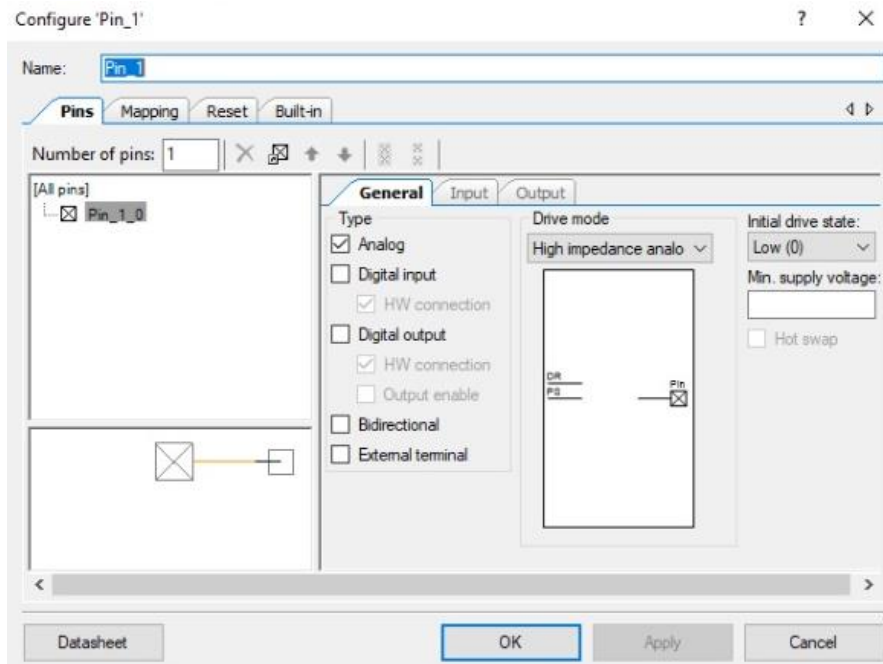


Ilustración 30. Configuración de los pines.

En la siguiente ilustración se aprecia la conexión del multiplexor y el ADC. Cabe resaltar que el multiplexor en su forma diferencial, necesita de pines por cada entrada. En este caso son seis sensores, por lo tanto, son 12 pines.

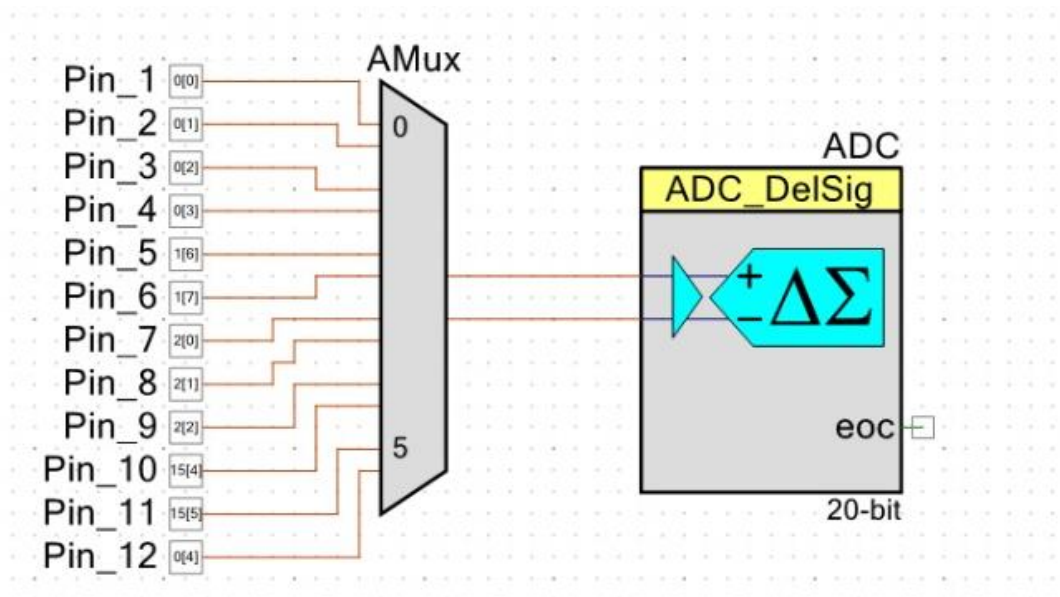


Ilustración 31. AMux y ADC conectados.

Por otro lado, se configura el puerto UART, que se utiliza para realizar la comunicación serial. En este caso se configuró para que transmitiera 57600 bits por segundo.

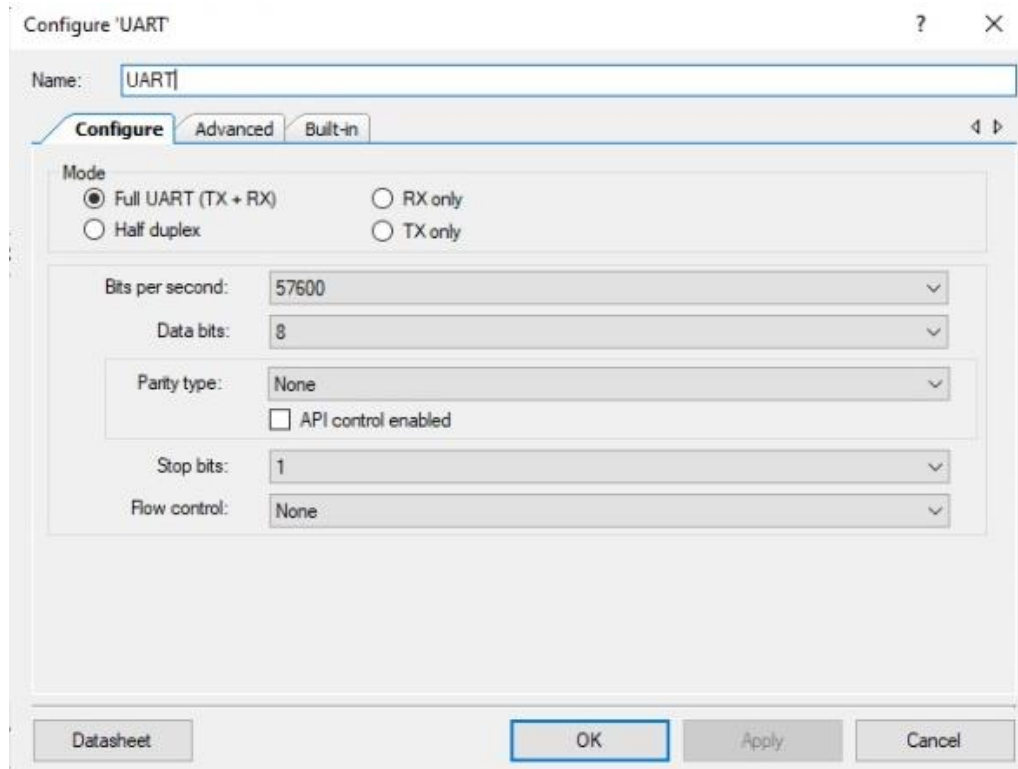


Ilustración 32. Configuración del puerto UART.

A continuación, se observa el puerto UART, el cual consta de los pines de comunicación Tx y Rx, los cuales son de transmisión y recepción, respectivamente. Cabe destacar el dispositivo que envía la información, así como el que recibe dicha información, deben de estar a la misma velocidad de transmisión que en este caso son 57600 baudios, de lo contrario, se puede mal interpretar la información.

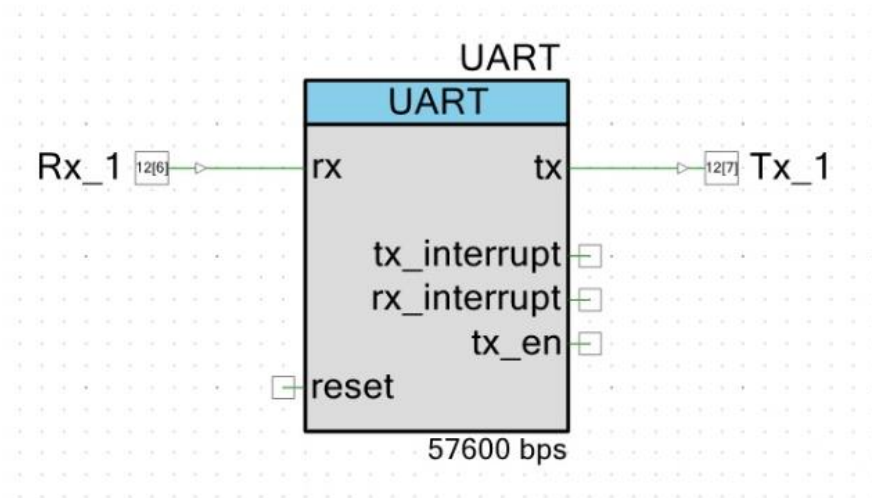


Ilustración 33. Puerto UART.

En la siguiente ilustración se puede apreciar el apartado de la configuración de los pines, en la cual se muestra el nombre del pin, el puerto y el número del pin. Cabe señalar que del “pin_1” al “pin_12”, son los pines para los sensores y los pines “Rx_1” y “Tx_1” se utilizan para la comunicación serial.

	Name	Port	Pin	Lock
<input checked="" type="checkbox"/>	Pin_1	P0[0]	48	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_2	P0[1]	49	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_3	P0[2]	50	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_4	P0[3]	51	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_5	P1[6]	18	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_6	P1[7]	19	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_7	P2[0]	62	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_8	P2[1]	63	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_9	P2[2]	64	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_10	P15[4]	60	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_11	P15[5]	61	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_12	P0[4]	53	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Rx_1	P12[6]	20	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Tx_1	P12[7]	21	<input checked="" type="checkbox"/>

Ilustración 34. Configuración de los pines en PSoC 5.

A continuación, se muestra la ubicación de los pines en el dispositivo PSoC 5, en donde los pines que se muestran de color azul son los pines que están ocupados.

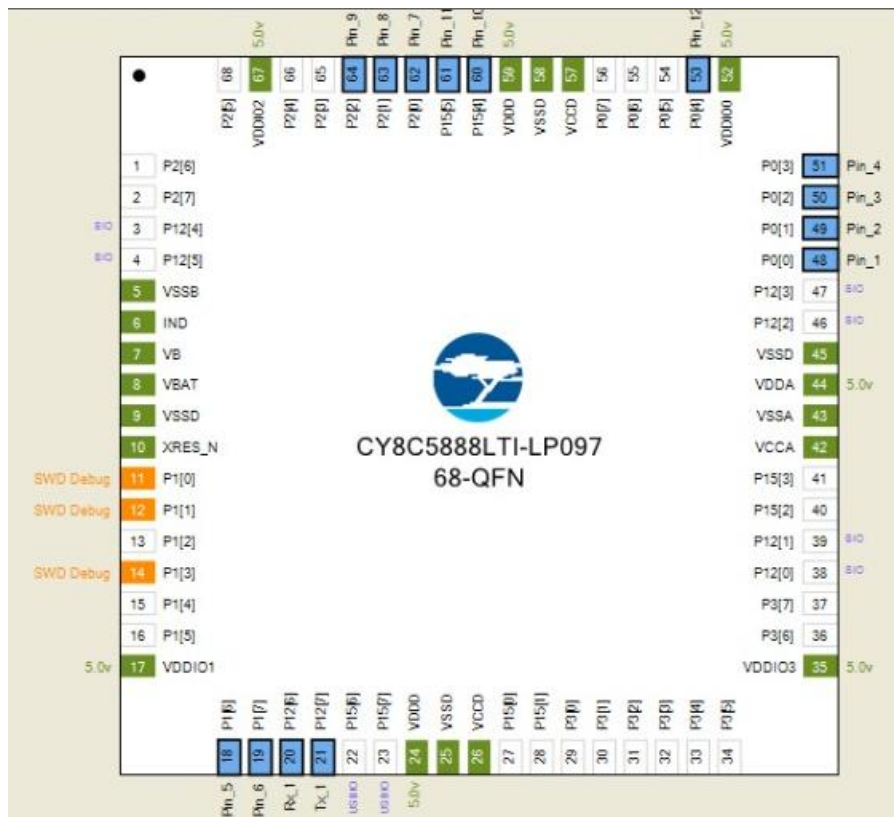


Ilustración 35. Pines en el microcontrolador.

A continuación, se muestra el ciclo for del código en C en el cual se muestra la utilización del AMux, así la lectura de un voltaje y su conversión a unidades de flujo de calor, posteriormente, se observa la utilización del UART, el cual se encarga de enviar los datos por el puerto serial.

```

19     for(;;)
20     {
21
22         AMux_FastSelect(0);
23         ADC_StartConvert();
24         ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
25         enteros[0]=ADC_GetResult32();
26         ADC_StopConvert();
27         voltajes[0] = ADC_CountsTo_Volts(enteros[0]);
28         flujo_calor[0] = ((voltajes[0])/0.00000144);
29         sprintf(Text, "%.6f", flujo_calor[0]);
30         UART_PutString("A0=");
31         UART_PutChar(10);
32         UART_PutString(Text);
33         UART_PutChar(10);
34         CyDelay(2000);
35
36

```

Ilustración 36. Ciclo for en el código del PSoC 5LS.

Finalmente, se realizó el código para la comunicación serial el cual, se muestra a continuación. En las líneas 1 y 2 se importan las librerías necesarias, posteriormente, en las líneas 4 y 5 se configura los parámetros necesarios para que la comunicación serial sea óptima, cabe resaltar que los valores entre el ESP32 y el PSoC 5LS tienen que ser los mismos. Finalmente, el programa entra en un ciclo para imprimir y leer los valores correspondientes.

```

1 from machine import UART
2 import time
3
4 uart = UART(2, 57600)
5 uart.init(57600 ,bits=8, parity=None, stop=1)
6
7
8 while True:
9     uart.any()
10    uart.read()
11    sleep (2)
12    print (uart.read())
13

```

Ilustración 37. Código para realizar la comunicación en Python.

3.5. Pruebas de funcionamiento.

Se realizaron pruebas de funcionamiento a cada componente que integra el proyecto, como son:

- Código para el leer un voltaje en MicroPython.
- Circuito de ganancia.
- Código para el leer un voltaje en PSoC 5.

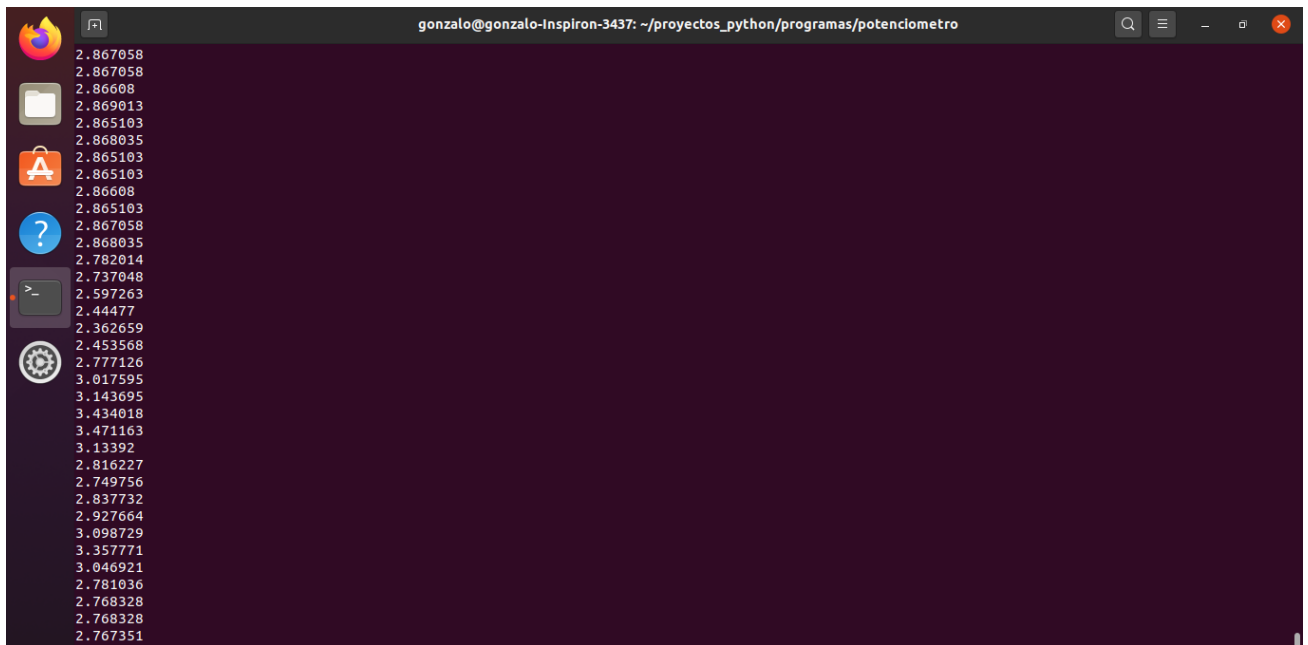
Primero, se realiza el código en el lenguaje MicroPython para que lea un voltaje, como se muestra a continuación.

```
1 while True:
2     adc = ADC(Pin(32))
3     adc.atten(ADC.ATTN_11DB)
4     voltaje=adc.read()
5     print (voltaje/1024)
6     sleep(0.15)
```

Ilustración 38. Código para leer un voltaje en MicroPython.

En la línea uno del código se muestra una función de control “while True” que es una estructura de control para el bucle infinito, en la segunda línea de código se aprecia la importación del ADC, así como del pin 32, posteriormente, se realiza la conversión del ADC a voltaje que como ya se mencionó con anterioridad, el microcontrolador ESP32 cuenta con un ADC y haciendo la conversión a números decimales resulta 1024. Finalmente, se le da una pausa al programa, esto se da en segundos.

Posteriormente, se conecta una fuente de voltaje y un potenciómetro para verificar que realmente cambie el valor del voltaje proporcionado, como se puede observar en la siguiente ilustración.



```
gonzalo@gonzalo-Inspiron-3437: ~/proyectos_python/programas/potenciometro
2.867058
2.867058
2.86608
2.869013
2.865103
2.868035
2.865103
2.865103
2.86608
2.865103
2.867058
2.868035
2.782014
2.737048
2.597263
2.44477
2.362659
2.453568
2.777126
3.017595
3.143695
3.434018
3.471163
3.13392
2.816227
2.749756
2.837732
2.927664
3.098729
3.357771
3.046921
2.781036
2.768328
2.768328
2.767351
```

Ilustración 39. Valores del voltaje impresos en la terminal.

Para verificar que el circuito funcione de manera correcta, se optó por alimentar el circuito con 0.2 V, para lo cual se utilizó un divisor de voltaje como se muestra a continuación.

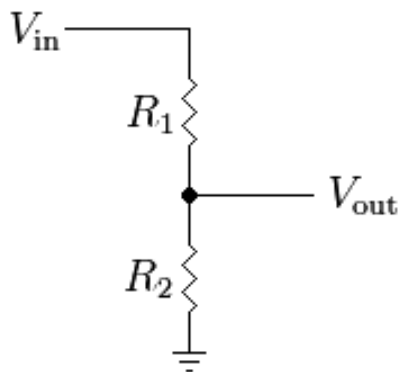


Ilustración 40. Diagrama del divisor de voltaje.

Teniendo una entrada de voltaje de 3.3 V y requiriendo una salida de voltaje, se realizó el siguiente cálculo:

$$V_{in} = 3.3V$$

$$V_{out} = 0.2V$$

$$R_1 = ?$$

$$R_2 = ?$$

Suponiendo que $R_1 = 10\Omega$:

$$R_2 = 10\Omega \left(\frac{3.3V - 0.2V}{0.2V} \right) = 155\Omega$$

$$V_{out} = 3.3V \left(\frac{10\Omega}{150\Omega + 10\Omega} \right) = 0.206V$$

Después de realizar los anteriores cálculos, se procede a medir el voltaje, lo que resultó de la siguiente manera.

En la siguiente ilustración se muestra el valor del voltaje a la salida del divisor de voltaje:



Ilustración 41. Resultado del divisor de voltaje.

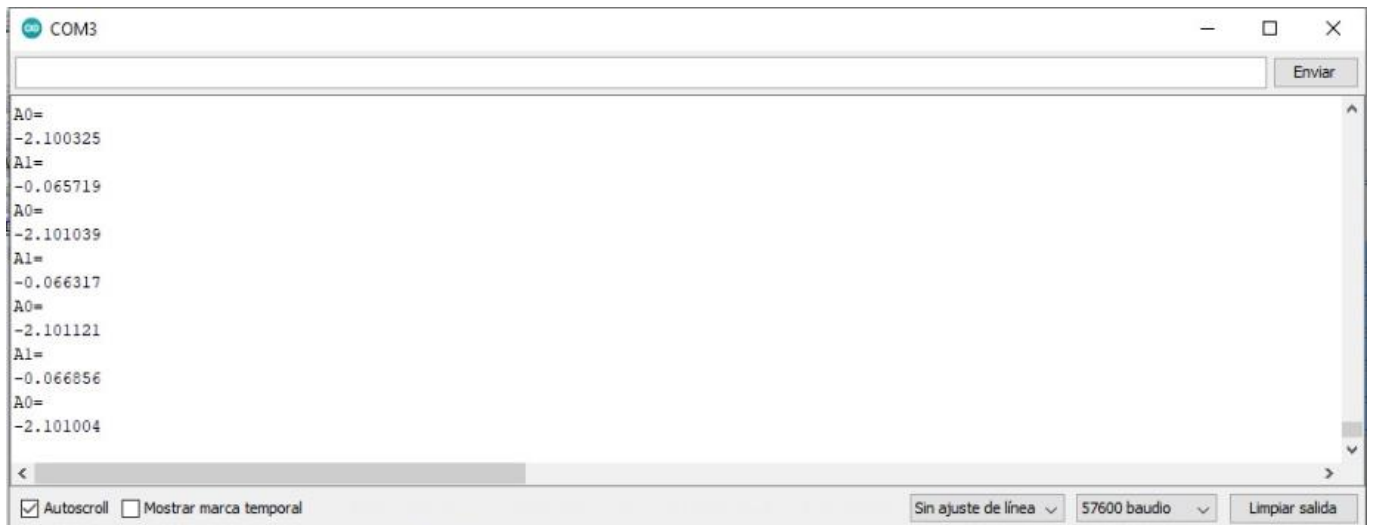
De la misma manera, se realizó la medición del voltaje en la primera etapa de amplificación, lo cual resultó un voltaje de 0.41 V. Finalmente, se realizó la medición

del voltaje en la segunda etapa de amplificación, la cual resultó de 0.81, como se puede observar en la ilustración.



Ilustración 42. Valor del voltaje en la segunda etapa de amplificación.

Por otro lado, se le realizan pruebas al programa del PSoC de la misma manera que al ESP32, es decir, se conecta una fuente de voltaje y un potenciómetro para verificar se funcionamiento. En este caso se utiliza el monitor serial del IDE de Arduino enviando la señal por el puerto UART, puesto que el software del PSoC no cuenta con un monitor serial. Esto se muestra en la siguiente ilustración.



CAPÍTULO 4. RESULTADOS.

En la siguiente ilustración se aprecia al sistema completo de acondicionamiento de la señal, en donde se observa a la fuente de alimentación, la cual se utiliza para proveer energía para que el circuito de amplificación funcione. También se observa al sensor conectado al circuito de amplificación y finalmente se aprecia al ESP32 conectado a una computadora para visualizar los datos obtenidos.

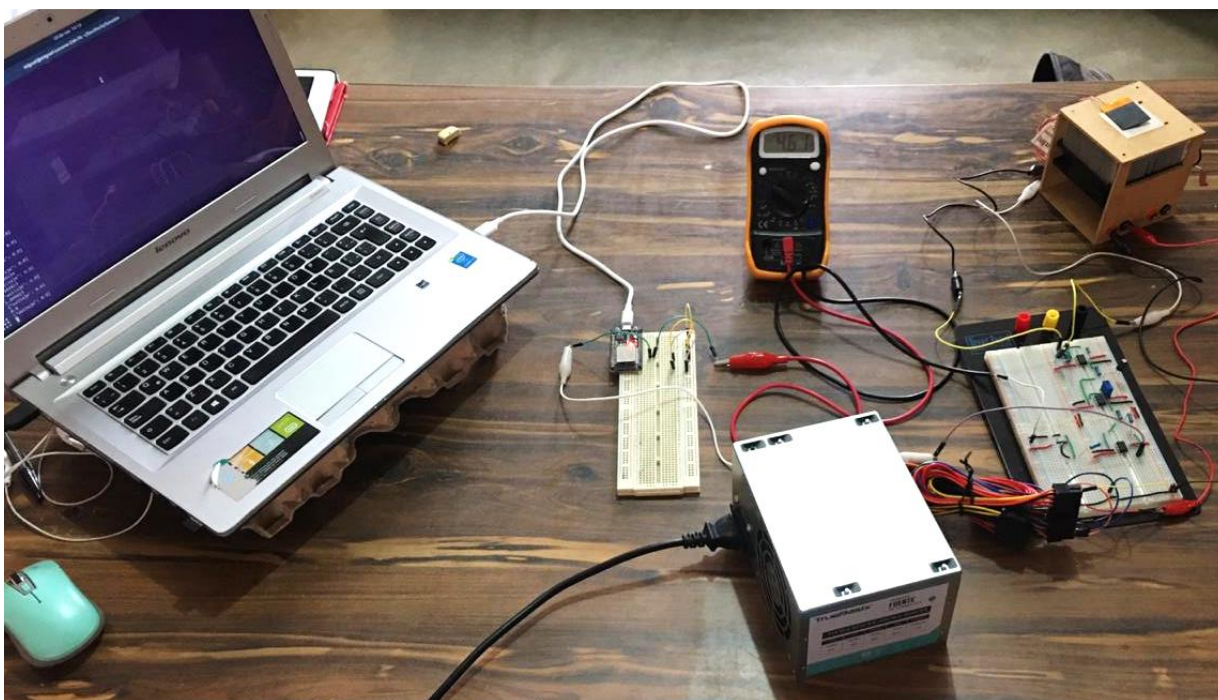


Ilustración 43. Sistema completo de la instrumentación del sensor de flujo de calor para el ESP32.

A continuación, se muestran los resultados del sensor de flujo de calor en ThingsBoard, utilizando el ESP32 y el circuito de ganancia, donde se muestra el flujo de calor a través del tiempo en donde se observa que el flujo de calor ronda alrededor de los $7 \frac{W}{m^2}$.

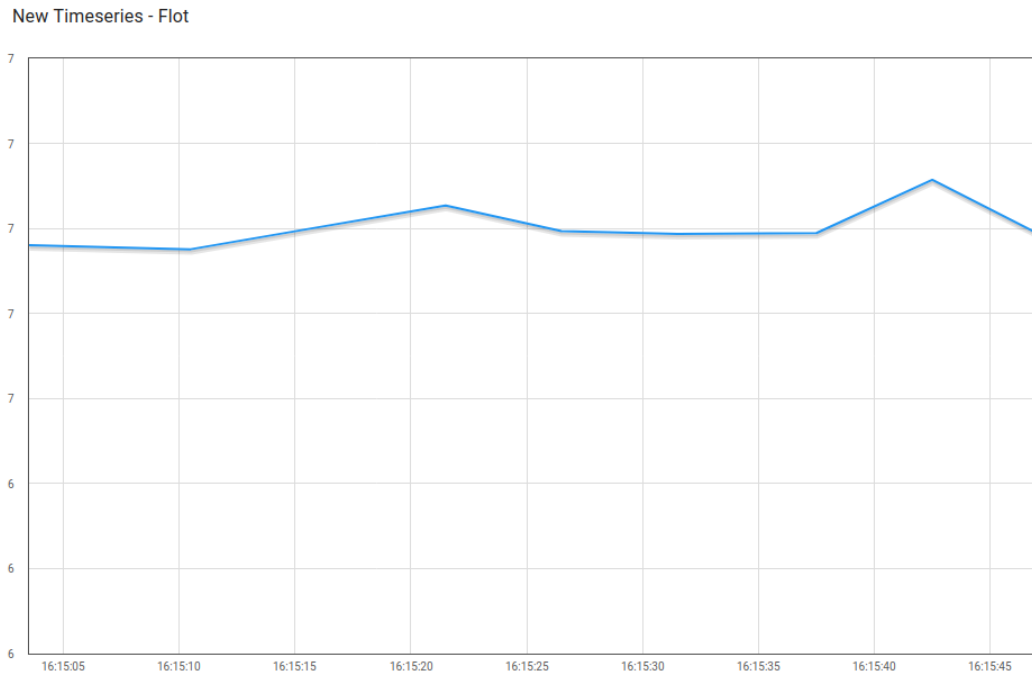


Ilustración 44. Flujo de calor a través del tiempo, utilizando el ESP32.

En la siguiente ilustración se aprecia al PSoC 5LS conectado al sensor de flujo de calor y al ESP32 recibiendo los datos por el puerto serial.

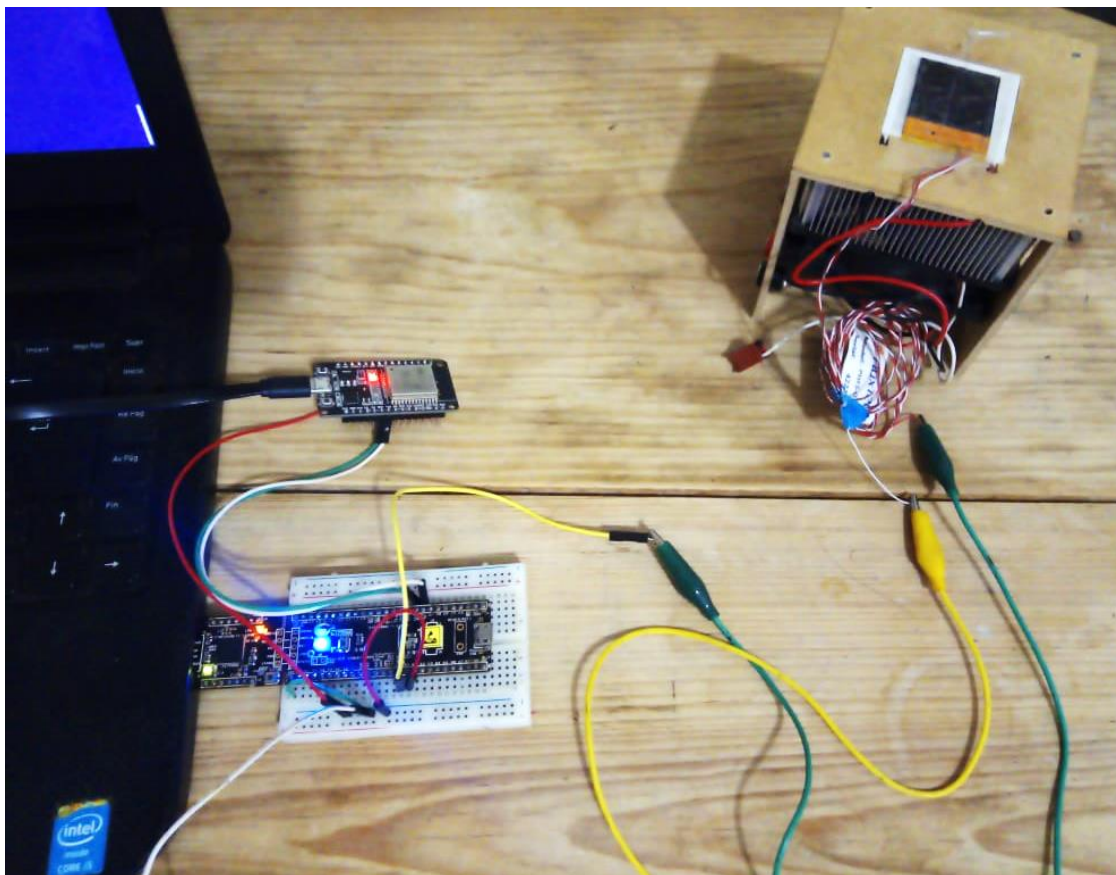
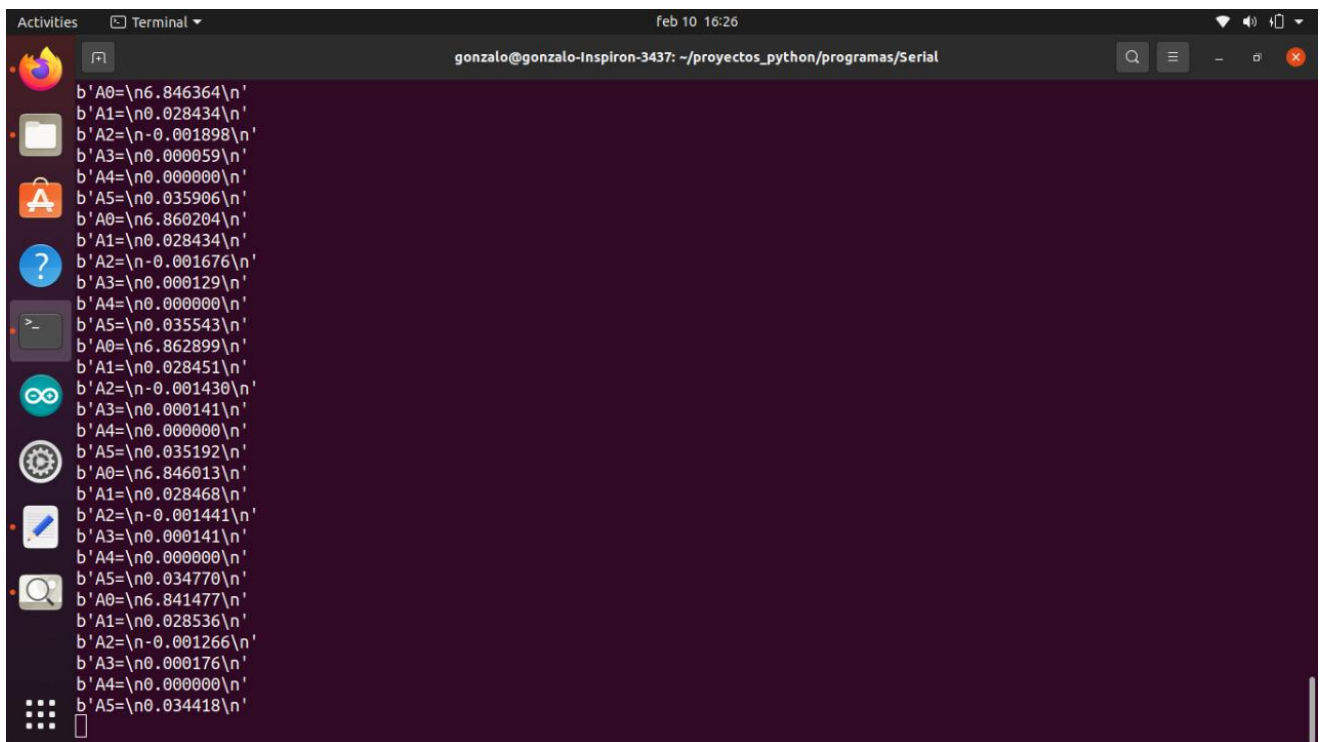


Ilustración 45. Sistema completo de la instrumentación del sensor de flujo de calor para el PSoC 5LS.

Como se puede observar, se muestran los valores del sensor de flujo de calor los cuales son muy cercanos a los que brinda el sensor de flujo de calor utilizando el ESP32, mismos que son recibidos por comunicación serial. En este caso, solamente fue proporcionado un sensor de flujo de calor



The image shows a terminal window on a Linux system. The title bar indicates the user is 'gonzalo' on a machine named 'gonzalo-Inspiron-3437', and the current directory is '~/proyectos_python/programas/Serial'. The terminal displays a series of hexadecimal data points, each preceded by a label (A0, A1, A2, A3, A4, A5) and a backslash-n character, indicating line breaks. The data appears to be received from a sensor, likely a temperature flow sensor as mentioned in the caption. The data is organized into groups of five lines each, with some groups having a label change (e.g., from A0 to A1).

```
b'A0=\n6.846364\n'  
b'A1=\n0.028434\n'  
b'A2=\n-0.001898\n'  
b'A3=\n0.000059\n'  
b'A4=\n0.000000\n'  
b'A5=\n0.035906\n'  
b'A0=\n6.860204\n'  
b'A1=\n0.028434\n'  
b'A2=\n-0.001676\n'  
b'A3=\n0.000129\n'  
b'A4=\n0.000000\n'  
b'A5=\n0.035543\n'  
b'A0=\n6.862899\n'  
b'A1=\n0.028451\n'  
b'A2=\n-0.001430\n'  
b'A3=\n0.000141\n'  
b'A4=\n0.000000\n'  
b'A5=\n0.035192\n'  
b'A0=\n6.846013\n'  
b'A1=\n0.028468\n'  
b'A2=\n-0.001441\n'  
b'A3=\n0.000141\n'  
b'A4=\n0.000000\n'  
b'A5=\n0.034770\n'  
b'A0=\n6.841477\n'  
b'A1=\n0.028536\n'  
b'A2=\n-0.001266\n'  
b'A3=\n0.000176\n'  
b'A4=\n0.000000\n'  
b'A5=\n0.034418\n'
```

Ilustración 46. Datos recibidos por el sensor de flujo de calor.

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES.

El presente proyecto se efectuó satisfactoriamente de las dos maneras posibles cumpliendo con objetivo que se planteó al inicio el cual fue realizar el diseño y construcción de un dispositivo para medir el flujo de calor en muros ya que, como se puede observar, los resultados obtenidos son muy similares, teniendo en cuenta que las lecturas se realizaron en días distintos, pero en la misma época del año.

Recomiendo al Instituto de Tecnologías Renovables que continúen implementando proyectos de investigación, así como que siga explorando la utilización del microcontrolador PSoC 5LS, ya que cuenta con muchas ventajas y beneficios.

CAPÍTULO 6. FUENTES DE INFORMACIÓN.

- IER-UNAM - Identidad - Por un futuro sustentable. (2020). Retrieved 7 November 2020, from <https://www.ier.unam.mx/nosotros/identidad.html>
- Cengel, Y. (2011). Heat and mass transfer. Brantford, Ont.: W. Ross MacDonald School Resource Services Library.
- Incropera, F. (2007). Introduction to heat transfer. Hoboken, NJ: Wiley.
- Ricardo Cabrera, CALOR Y TERMODINAMICA, conducción, ejercicios. (2020). Retrieved 16 November 2020, from https://ricuti.com.ar/no_me_salén/TERMO/TEOR_conduccion.html
- ¿Qué es la convección? (2020). Retrieved 16 November 2020, from <https://www.vix.com/es/btg/curiosidades/5074/que-es-la-conveccion>
- sensor noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com. (2020). Retrieved 16 November 2020, from <https://www.oxfordlearnersdictionaries.com/us/definition/english/sensor?q=sensor>
- Fluxteq.com. 2020. [online] Available at: <<https://www.fluxteq.com/>> [Accessed 16 August 2020].
- Boylestad, R., Nashelsky, L., & Suárez Fernández, A. (2009). Electrónica (10th ed.). Naucalpan de Juárez (México): Pearson Educación de México.
- Introducción a los Amplificaciones Operacionales - Electrónica Gonzalo Boscana. (2020). Retrieved 23 November 2020, from <https://sites.google.com/site/electronicainet/introduccion-a-los-amplificaciones-operacionales>
- La pérdida de calor corporal - thetrekkinglife.com. (2020). Retrieved 23 November 2020, from <https://thetrekkinglife.com/articulos/seguridad-y-supervivencia/112-la-perdida-de-calor-corporal>
- DigiKey Electronics México | Distribuidor de componentes electrónicos. (2020). Retrieved 23 November 2020, from <https://www.digikey.com.mx/>
- AD620 Instrumentation Amplifiers. (2020). Retrieved 23 November 2020, from <https://www.mouser.mx/new/analog-devices/adi-ad620-amplifiers/>
- LM741CN. (2020). Retrieved 23 November 2020, from <https://www.carrod.mx/products/lm741-amplificador-operacional>
- 10pcs ua741cn dip-8 UA741 LM741 st chips ic amplificadores operacionales. (2020). Retrieved 23 November 2020, from <https://www.teknistore.com/es/microcontroladores-de-chips/2589-10pcs-ua741cn-dip-8-ua741-lm741-st-chips-ic-amplificadores-operacionales.html>

- Jaquería, L. (2020). Preparando un taller de MicroPython en ESP32. Retrieved 28 November 2020, from <https://lajaqueria.org/proyectos/2019/03/30/preparando-un-taller-de-microPython-en-esp32.html#:~:text=Los%20microcontroladores%20ESP32%20o%20ESP8266,js%20y%20por%20supuesto%20Python.>
- OLED, W. (2020). BricoGeek.com. Retrieved 28 November 2020, from <https://tienda.bricogeek.com/arduino-compatibles/1121-wemos-esp32-oled.html>
- Nyambi, W. (2020). The IoT Revolution. Retrieved 28 November 2020, from <https://www.gbnews.ch/the-iot-revolution/>
- Medición de temperatura sin contacto - diarioelectronicohoy.com. (2020). Retrieved 29 November 2020, from <https://www.diarioelectronicohoy.com/medicion-de-temperatura-sin-contacto/#:~:text=En%20t%C3%A9rminos%20el%C3%A9ctricos%2C%20una%20termopila,varios%20termopares%20conectados%20en%20serie.&text=Como%20tiene%20una%20masa%20t%C3%A9rmica,como%20una%20diferencia%20de%20temperatura.>
- Termopares-la física y la química. (2020). Retrieved 29 November 2020, from <https://lafisicayquimica.com/termopares/>
- perfil, V. (2020). Calibración de Sensores. Retrieved 29 November 2020, from <http://mecatronicasensoresyactuadores.blogspot.com/2013/06/calibracion-de-sensores.html>
- Laurila, H. (2020). Cómo calibrar los sensores de temperatura. Retrieved 29 November 2020, from <https://blog.beamex.com/es/como-calibrar-los-sensores-de-temperatura>
- ¿Qué es Internet of Things (IoT). (2020). Retrieved 26 December 2020, from <https://www.oracle.com/mx/internet-of-things/what-is-iot/>
- ESP32 Wi-Fi & Bluetooth MCU | Espressif Systems. (2020). Retrieved 27 December 2020, from <https://www.espressif.com/en/products/socs/esp32>
- (2021). Retrieved 1 January 2020, from <https://www.cypress.com/>
- Serial and UART Tutorial. (2021). Retrieved 13 January 2021, from https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/index.html#uart

CAPITULO 7. ANEXOS.

7.1. Códigos completos.

Código del PSoC:

```
#include "project.h"
```

```
#include "stdio.h"
```

```
#include "math.h"
```

```
#include "stdlib.h"
```

```
int32 enteros[6];
```

```
float32 voltajes[6];
```

```
float flujo_calor [6];
```

```
char Text[20];
```

```
int canal;
```

```
int main(void)
```

```
{
```

```
    CyGlobalIntEnable;
```

```
    UART_Start();
```

```
    ADC_Start();
```

```
    AMux_Start();
```

```
    for(;;)
```

```
    {
```

```

AMux_FastSelect(0);

ADC_StartConvert();

ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);

enteros[0]=ADC_GetResult32();

ADC_StopConvert();

voltajes[0] = ADC_CountsTo_Volts(enteros[0]);

flujo_calor[0] = ((voltajes[0])/0.00000144);

sprintf(Text, "%.6f", flujo_calor[0]);

UART_PutString("A0=");

UART_PutChar(10);

UART_PutString(Text);

UART_PutChar(10);

CyDelay(2000);

```

```

AMux_FastSelect(1);

ADC_StartConvert();

ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);

enteros[1]=ADC_GetResult32();

ADC_StopConvert();

voltajes[1] = ADC_CountsTo_Volts(enteros[1]);

flujo_calor[1] = ((voltajes[1])/0.00000144);

sprintf(Text, "%.6f", flujo_calor[1]);

UART_PutString("A1=");

UART_PutChar(10);

```

```
UART_PutString(Text);  
UART_PutChar(10);  
CyDelay(2000);
```

```
AMux_FastSelect(2);  
ADC_StartConvert();  
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);  
enteros[2]=ADC_GetResult32();  
ADC_StopConvert();  
voltajes[2] = ADC_CountsTo_Volts(enteros[2]);  
flujo_calor[2] = ((voltajes[2])/0.00000144);  
sprintf(Text, "%.6f", flujo_calor[2]);  
UART_PutString("A2=");  
UART_PutChar(10);  
UART_PutString(Text);  
UART_PutChar(10);  
CyDelay(2000);
```

```
AMux_FastSelect(3);  
ADC_StartConvert();  
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);  
enteros[3]=ADC_GetResult32();  
ADC_StopConvert();
```

```
voltajes[3] = ADC_CountsTo_Volts(enteros[3]);  
flujo_calor[3] = ((voltajes[3])/0.00000144);  
sprintf(Text, "%.6f", flujo_calor[3]);  
UART_PutString("A3=");  
UART_PutChar(10);  
UART_PutString(Text);  
UART_PutChar(10);  
CyDelay(2000);
```

```
AMux_FastSelect(4);  
ADC_StartConvert();  
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);  
enteros[4]=ADC_GetResult32();  
ADC_StopConvert();  
voltajes[4] = ADC_CountsTo_Volts(enteros[4]);  
flujo_calor[4] = ((voltajes[4])/0.00000144);  
sprintf(Text, "%.6f", flujo_calor[4]);  
UART_PutString("A4=");  
UART_PutChar(10);  
UART_PutString(Text);  
UART_PutChar(10);  
CyDelay(2000);
```

```

    AMux_FastSelect(5);

    ADC_StartConvert();

    ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);

    enteros[5]=ADC_GetResult32();

    ADC_StopConvert();

    voltajes[5] = ADC_CountsTo_Volts(enteros[5]);

    flujo_calor[5] = ((voltajes[5])/0.00000144);

    sprintf(Text, "%.6f", flujo_calor[5]);

    UART_PutString("A5=");

    UART_PutChar(10);

    UART_PutString(Text);

    UART_PutChar(10);

    CyDelay(2000);

}

}

/* [] END OF FILE */

```

Códigos del ESP32:

Main.py

```

# from wifi import do_connect

import time

# import network

from funcion import *

from wifi import *

```

```
from machine import UART
```

```
from machine import ADC
```

```
from machine import Pin
```

```
red = 'cb61c0'
```

```
clave = '280865312'
```

```
unique_id = '66f5d0f0-5de2-11eb-9c3f-d1ead9980bc3'
```

```
token = 'tDzaQ1x1jJoWdK5lOtg'
```

```
label = 'Flujo de calor'
```

```
do_connect(red,clave)
```

```
while True:
```

```
    adc = ADC(Pin(32))
```

```
    adc.atten(ADC.ATTN_11DB)
```

```
    voltaje=adc.read()
```

```
    resultado = ((3.3*voltaje/1024)/1000000)/0.00000144
```

```
    valor = resultado
```

```
    data = {label: valor}
```

```
publish_thingsboard(red,clave,token, unique_id,data)
```

```
time.sleep(5)
```

boot.py

```
from time import sleep
```

funcion.py

```
from umqtt.simple import MQTTClient
```

```
import gc
```

```
import json
```

```
import machine
```

```
import time
```

```
from wifi import do_connect
```

```
def settimeout(duration):
```

```
    pass
```

```
def t3_publication(topic, msg):
```

```
    print (topic, ';', msg)
```

```
    pycom.rgbled(0xff00)
```

```
def publish_thingsboard(red, clave,token,UNIQUE_ID,data):
```

```
    # from umqtt.simple import MQTTClient
```

```
    # import gc
```

```
    # import json
```

```
    # import machine
```

```
    # import utime
```



```

contador = 0

condicion = True

while condicion:

    try:

        client = MQTTClient(UNIQUE_ID, "iot.ier.unam.mx", port = 1883,
user=token, password=")

        client.settimeout = settimeout

        client.connect()

        print(json.dumps(data))

        client.publish('v1/devices/me/telemetry', json.dumps(data) )

        client.disconnect()

        condicion = False

    except Exception as inst:

        do_connect(red,clave);

        time.sleep(10)

        contador += 1

        print("Falla ",contador)

        if contador >= 10:

            machine.reset()

```

wifi.py

```

def do_connect(red,clave):

    import network

    import time

    sta_if = network.WLAN(network.STA_IF)

    if not sta_if.isconnected():

```

```

print('connecting to network...')

sta_if.active(True)

sta_if.connect(red,clave)

#sta_if.connect('IER', 'acadier2014')

while not sta_if.isconnected():

    pass

else:

    print("Already connected")

print('network config:', sta_if.ifconfig())

time.sleep(5)

```

Códigos del ESP32 para comunicación serial:

Main.py

```

from machine import UART

import time


uart = UART(2, 57600)

uart.init(57600 ,bits=8, parity=None, stop=1)


while True:

    uart.any()

    uart.read()

    sleep (2)

    print (uart.read())

```

boot.py.

from time import sleep

7.2. Fichas técnicas.

A continuación, se muestran las fichas técnicas de los componentes eléctricos que se utilizan:

- Amplificador de instrumentación AD620.
- Amplificador operacional LM741.
- ADC DeltaSigma.
- Multiplexor.



Low Cost, Low Power Instrumentation Amplifier

AD620

FEATURES

EASY TO USE

Gain Set with One External Resistor

(Gain Range 1 to 1000)

Wide Power Supply Range (± 2.3 V to ± 18 V)

Higher Performance than Three Op Amp IA Designs

Available in 8-Lead DIP and SOIC Packaging

Low Power, 1.3 mA max Supply Current

EXCELLENT DC PERFORMANCE ("B GRADE")

50 μ V max, Input Offset Voltage

0.6 μ V/ $^{\circ}$ C max, Input Offset Drift

1.0 nA max, Input Bias Current

100 dB min Common-Mode Rejection Ratio ($G = 10$)

LOW NOISE

9 nV/ $\sqrt{\text{Hz}}$, @ 1 kHz, Input Voltage Noise

0.28 μ V p-p Noise (0.1 Hz to 10 Hz)

EXCELLENT AC SPECIFICATIONS

120 kHz Bandwidth ($G = 100$)

15 μ s Settling Time to 0.01%

APPLICATIONS

Weigh Scales

ECG and Medical Instrumentation

Transducer Interface

Data Acquisition Systems

Industrial Process Controls

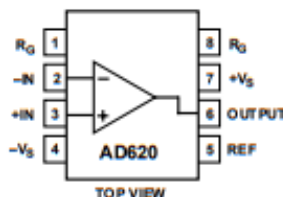
Battery Powered and Portable Equipment

PRODUCT DESCRIPTION

The AD620 is a low cost, high accuracy instrumentation amplifier that requires only one external resistor to set gains of 1 to

CONNECTION DIAGRAM

8-Lead Plastic Mini-DIP (N), Cerdip (Q)
and SOIC (R) Packages



1000. Furthermore, the AD620 features 8-lead SOIC and DIP packaging that is smaller than discrete designs, and offers lower power (only 1.3 mA max supply current), making it a good fit for battery powered, portable (or remote) applications.

The AD620, with its high accuracy of 40 ppm maximum nonlinearity, low offset voltage of 50 μ V max and offset drift of 0.6 μ V/ $^{\circ}$ C max, is ideal for use in precision data acquisition systems, such as weigh scales and transducer interfaces. Furthermore, the low noise, low input bias current, and low power of the AD620 make it well suited for medical applications such as ECG and noninvasive blood pressure monitors.

The low input bias current of 1.0 nA max is made possible with the use of SuperBeta processing in the input stage. The AD620 works well as a preamplifier due to its low input voltage noise of 9 nV/ $\sqrt{\text{Hz}}$ at 1 kHz, 0.28 μ V p-p in the 0.1 Hz to 10 Hz band, 0.1 pA/ $\sqrt{\text{Hz}}$ input current noise. Also, the AD620 is well suited for multiplexed applications with its settling time of 15 μ s to 0.01% and its cost is low enough to enable designs with one in-amp per channel.

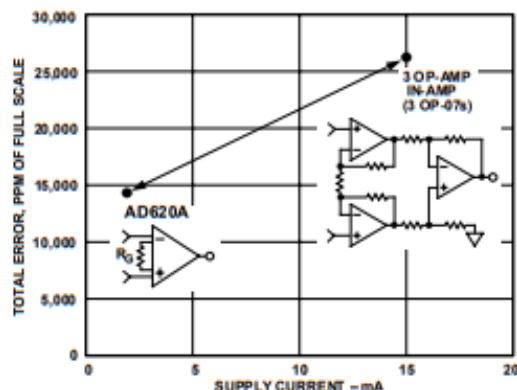


Figure 1. Three Op Amp IA Designs vs. AD620

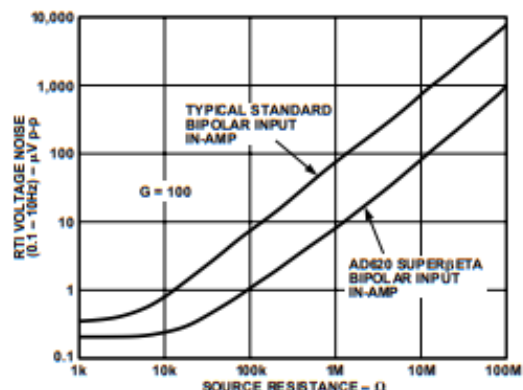


Figure 2. Total Voltage Noise vs. Source Resistance

REV. E

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: <http://www.analog.com>
Fax: 781/326-8703 © Analog Devices, Inc., 1999

AD620—SPECIFICATIONS

(Typical @ +25°C, $V_S = \pm 15$ V, and $R_L = 2$ k Ω , unless otherwise noted)

Model	Conditions	AD620A			AD620B			AD620S ¹			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
GAIN	$G = 1 + (49.4 \text{ k}/R_G)$										
Gain Range		1		10,000	1		10,000	1		10,000	
Gain Error ²	$V_{OUT} = \pm 10$ V										
$G = 1$			0.03	0.10		0.01	0.02		0.03	0.10	%
$G = 10$			0.15	0.30		0.10	0.15		0.15	0.30	%
$G = 100$			0.15	0.30		0.10	0.15		0.15	0.30	%
$G = 1000$			0.40	0.70		0.35	0.50		0.40	0.70	%
Nonlinearity, $G = 1$ –1000	$V_{OUT} = -10$ V to $+10$ V, $R_L = 10$ k Ω		10	40		10	40		10	40	ppm
$G = 1$ –100	$R_L = 2$ k Ω		10	95		10	95		10	95	ppm
Gain vs. Temperature	$G = 1$ Gain $> 1^2$			10 –90			10 –50			10 –50	ppm/°C ppm/°C
VOLTAGE OFFSET	(Total RTI Error = $V_{OS} + V_{OSD}/G$)										
Input Offset, V_{OS}	$V_S = \pm 5$ V to ± 15 V		30	125		15	50		30	125	μ V
Over Temperature	$V_S = \pm 5$ V to ± 15 V			185			85			225	μ V
Average TC	$V_S = \pm 5$ V to ± 15 V		0.3	1.0		0.1	0.6		0.3	1.0	μ V/°C
Output Offset, V_{OSD}	$V_S = \pm 15$ V		400	1000		200	500		400	1000	μ V
	$V_S = \pm 5$ V			1500			750			1500	μ V
Over Temperature	$V_S = \pm 5$ V to ± 15 V			2000			1000			2000	μ V
Average TC	$V_S = \pm 5$ V to ± 15 V		5.0	15		2.5	7.0		5.0	15	μ V/°C
Offset Referred to the Input vs. Supply (PSR)	$V_S = \pm 2.3$ V to ± 18 V										
$G = 1$		80	100		80	100		80	100		dB
$G = 10$		95	120		100	120		95	120		dB
$G = 100$		110	140		120	140		110	140		dB
$G = 1000$		110	140		120	140		110	140		dB
INPUT CURRENT											
Input Bias Current			0.5	2.0		0.5	1.0		0.5	2	nA
Over Temperature				2.5			1.5			4	nA
Average TC			3.0			3.0			8.0		pA/°C
Input Offset Current			0.3	1.0		0.3	0.5		0.3	1.0	nA
Over Temperature				1.5			0.75			2.0	nA
Average TC			1.5			1.5			8.0		pA/°C
INPUT											
Input Impedance											
Differential			10 ¹²			10 ¹²			10 ¹²		G Ω pF
Common-Mode			10 ¹²			10 ¹²			10 ¹²		G Ω pF
Input Voltage Range ³	$V_S = \pm 2.3$ V to ± 5 V	$-V_S + 1.9$		$+V_S - 1.2$	$-V_S + 1.9$		$+V_S - 1.2$	$-V_S + 1.9$		$+V_S - 1.2$	V
Over Temperature	$V_S = \pm 5$ V to ± 18 V	$-V_S + 2.1$		$+V_S - 1.3$	$-V_S + 2.1$		$+V_S - 1.3$	$-V_S + 2.1$		$+V_S - 1.3$	V
		$-V_S + 1.9$		$+V_S - 1.4$	$-V_S + 1.9$		$+V_S - 1.4$	$-V_S + 1.9$		$+V_S - 1.4$	V
Over Temperature		$-V_S + 2.1$		$+V_S - 1.4$	$-V_S + 2.1$		$+V_S - 1.4$	$-V_S + 2.3$		$+V_S - 1.4$	V
Common-Mode Rejection Ratio DC to 60 Hz with 1 k Ω Source Imbalance	$V_{CM} = 0$ V to ± 10 V										
$G = 1$		73	90		80	90		73	90		dB
$G = 10$		93	110		100	110		93	110		dB
$G = 100$		110	130		120	130		110	130		dB
$G = 1000$		110	130		120	130		110	130		dB
OUTPUT											
Output Swing	$R_L = 10$ k Ω , $V_S = \pm 2.3$ V to ± 5 V	$-V_S + 1.1$		$+V_S - 1.2$	$-V_S + 1.1$		$+V_S - 1.2$	$-V_S + 1.1$		$+V_S - 1.2$	V
Over Temperature	$V_S = \pm 5$ V to ± 18 V	$-V_S + 1.4$		$+V_S - 1.3$	$-V_S + 1.4$		$+V_S - 1.3$	$-V_S + 1.6$		$+V_S - 1.3$	V
		$-V_S + 1.2$		$+V_S - 1.4$	$-V_S + 1.2$		$+V_S - 1.4$	$-V_S + 1.2$		$+V_S - 1.4$	V
Over Temperature		$-V_S + 1.6$		$+V_S - 1.5$	$-V_S + 1.6$		$+V_S - 1.5$	$-V_S + 2.3$		$+V_S - 1.5$	V
Short Current Circuit			± 18			± 18			± 18		mA

Precision V-I Converter

The AD620, along with another op amp and two resistors, makes a precision current source (Figure 37). The op amp buffers the reference terminal to maintain good CMR. The output voltage V_X of the AD620 appears across R_1 , which converts it to a current. This current less only the input bias current of the op amp, then flows out to the load.

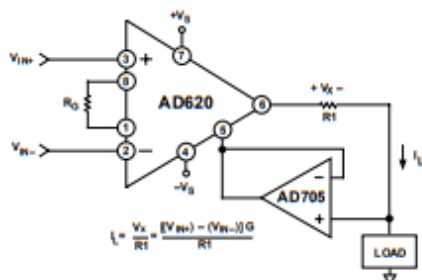


Figure 37. Precision Voltage-to-Current Converter (Operates on 1.8 mA, ± 3 V)

GAIN SELECTION

The AD620's gain is resistor programmed by R_G , or more precisely, by whatever impedance appears between Pins 1 and 8. The AD620 is designed to offer accurate gains using 0.1%–1% resistors. Table II shows required values of R_G for various gains. Note that for $G = 1$, the R_G pins are unconnected ($R_G = \infty$). For any arbitrary gain R_G can be calculated by using the formula:

$$R_G = \frac{49.4 \text{ k}\Omega}{G - 1}$$

To minimize gain error, avoid high parasitic resistance in series with R_G to minimize gain drift. R_G should have a low TC—less than 10 ppm/°C—for the best performance.

Table II. Required Values of Gain Resistors

1% Std Table Value of R_G , Ω	Calculated Gain	0.1% Std Table Value of R_G , Ω	Calculated Gain
49.9 k	1.990	49.3 k	2.002
12.4 k	4.984	12.4 k	4.984
5.49 k	9.998	5.49 k	9.998
2.61 k	19.93	2.61 k	19.93
1.00 k	50.40	1.01 k	49.91
499	100.0	499	100.0
249	199.4	249	199.4
100	495.0	98.8	501.0
49.9	991.0	49.3	1,003

INPUT AND OUTPUT OFFSET VOLTAGE

The low errors of the AD620 are attributed to two sources, input and output errors. The output error is divided by G when referred to the input. In practice, the input errors dominate at high gains and the output errors dominate at low gains. The total V_{OS} for a given gain is calculated as:

$$\text{Total Error RTI} = \text{input error} + (\text{output error}/G)$$

$$\text{Total Error RTO} = (\text{input error} \times G) + \text{output error}$$

REFERENCE TERMINAL

The reference terminal potential defines the zero output voltage, and is especially useful when the load does not share a precise ground with the rest of the system. It provides a direct means of injecting a precise offset to the output, with an allowable range of 2 V within the supply voltages. Parasitic resistance should be kept to a minimum for optimum CMR.

INPUT PROTECTION

The AD620 features 400 Ω of series thin film resistance at its inputs, and will safely withstand input overloads of up to ± 15 V or ± 60 mA for several hours. This is true for all gains, and power on and off, which is particularly important since the signal source and amplifier may be powered separately. For longer time periods, the current should not exceed 6 mA ($I_{IN} \leq V_{IN}/400 \Omega$). For input overloads beyond the supplies, clamping the inputs to the supplies (using a low leakage diode such as an FD333) will reduce the required resistance, yielding lower noise.

RF INTERFERENCE

All instrumentation amplifiers can rectify out of band signals, and when amplifying small signals, these rectified voltages act as small dc offset errors. The AD620 allows direct access to the input transistor bases and emitters enabling the user to apply some first order filtering to unwanted RF signals (Figure 38), where $RC \approx 1/(2\pi f)$ and where $f \geq$ the bandwidth of the AD620; $C \leq 150$ pF. Matching the extraneous capacitance at Pins 1 and 8 and Pins 2 and 3 helps to maintain high CMR.

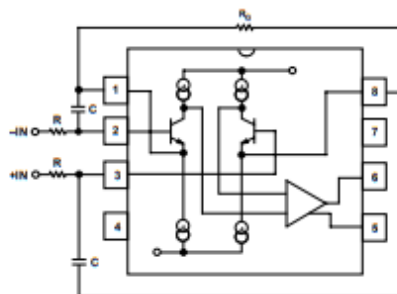


Figure 38. Circuit to Attenuate RF Interference

LM741

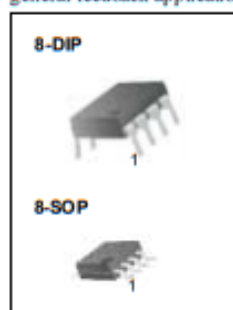
Single Operational Amplifier

Features

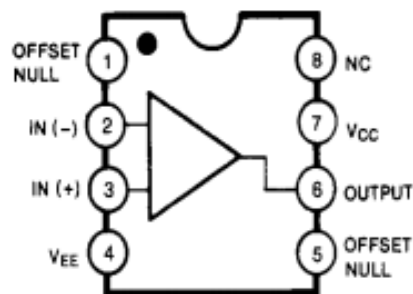
- Short circuit protection
- Excellent temperature stability
- Internal frequency compensation
- High Input voltage range
- Null of offset

Description

The LM741 series are general purpose operational amplifiers. It is intended for a wide range of analog applications. The high gain and wide range of operating voltage provide superior performance in integrator, summing amplifier, and general feedback applications.



Internal Block Diagram

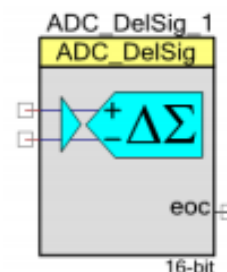


Delta Sigma Analog to Digital Converter (ADC_DelSig)

3.30

Features

- Selectable resolutions, 8 to 20 bits
- Eleven input ranges for each resolution
- Sample rate 8 sps to 384 ksp/s
- Operational modes:
 - Single sample
 - Multi-sample
 - Continuous mode
 - Multi-sample (Turbo)
- High input impedance input buffer
 - Selectable input buffer gain (1, 2, 4, 8) or input buffer bypass
- Multiple internal or external reference options
- Automatic power configuration
- Up to four run-time ADC configurations



General Description

The Delta Sigma Analog to Digital Converter (ADC_DelSig) provides a low-power, low-noise front end for precision measurement applications. You can use it in a wide range of applications, depending on resolution, sample rate, and operating mode. It can produce 16-bit audio; high speed and low resolution for communications processing; and high-precision 20-bit low-speed conversions for sensors such as strain gauges, thermocouples, and other high-precision sensors. When processing audio information, the ADC_DelSig is used in a continuous operation mode. When used for scanning multiple sensors, the ADC_DelSig is used in one of the multi-sample modes. When used for single-point high-resolution measurements, the ADC_DelSig is used in single-sample mode.

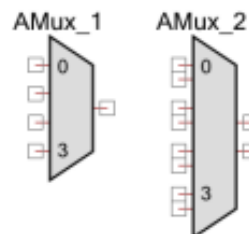
Delta-sigma converters use oversampling to spread the quantization noise across a wider frequency spectrum. This noise is shaped to move most of it outside the input signal's bandwidth. An internal low-pass filter is used to filter out the noise outside the desired input

Analog Multiplexer (AMux)

1.80

Features

- Single or differential connections
- Adjustable between 1 and 256 connections
- Software controlled
- Connections may be pins or internal sources
- Multiple simultaneous connections
- Bidirectional (passive)



General Description

The analog multiplexer (AMux) Component can be used to connect none, one, or more analog signals to a different common analog signal. The ability to connect more than one analog signal at a time provides cross-bar switch support, which is an extension beyond traditional mux functionality.

When to Use an AMux

Use an AMux any time you need to multiplex multiple analog signals into a single source or destination. Because the AMux is passive, it can be used to multiplex input or output signals.

Input/Output Connections

This section describes the various input and output connections for the AMux. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

I/O	Description
0-255 – Analog	The AMux is capable of having between 1 and 256 analog switchable connections.
0-255 (paired) – Analog *	The paired switchable connections are only used when the MuxType parameter is set to Differential .
common – Analog	The "common" signal is the common connection; it is not labeled. The channel selected with the <code>AMux_Select()</code> function is connected to this terminal.