ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS

GRADO EN INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA DE ADQUISICIÓN DE DATOS PARA LA MONITORIZACIÓN DE UNA PILA DE COMBUSTIBLE BASADA EN ARDUINO

TRABAJO DE FIN DE GRADO

AUTOR

Rubén Fernando Prades Mateu

DIRECTOR

Ignacio Peñarrocha Alós

Castelló de la Plana, octubre de 2020

ÍNDICE

ÍNDICE 5
ÍNDICE DE ILUSTRACIONES 7
ÍNDICE DE TABLAS 8
1 MEMORIA 10
1.1 Objeto 10
1.2 Alcance 10
1.3 Antecedentes 11
1.4 Normas y referencias 13
1.4.1 Disposiciones legales y normas aplicadas 13
1.4.2 Programas de desarrollo 13
1.4.3 Bibliografía 15
1.5 Definiciones y abreviaturas 16
1.5.1 Definiciones 16
1.5.2 Abreviaturas 17
1.6 Requisitos de diseño 18
1.6.1 Adquisición de datos 18
1.6.2 Comunicaciones 18
1.6.3 Almacenamiento 18
1.6.4 Interfaz de usuario 18
1.7 Análisis de soluciones 19
1.7.1 Configuración general del dispositivo 19
1.7.2 Unidad central de proceso 19
1.7.3 Lectura de temperaturas 20
1.7.4 Lectura de corrientes 20
1.7.5 Protocolo de comunicación 21
1.7.6 Software 21
1.8 Resultados finales 22
1.8.1 Hardware general 23
1.8.2 RTC 24
1.8.3 Lectura de tensiones 24
1.8.4 Lectura de corrientes 24
1.8.5 Lectura de temperaturas 25
1.8.6 Entradas analógicas 25

1.8.7 Pulsador y led de estado 25
1.8.8 Tarjeta SD 27
1.8.9 Salida MODBUS e interfaz de usuario 27
1.9 Planificación 29
1.10 Orden de prioridad entre los documentos 31
1.11 Conclusiones y trabajo futuro 31
2 ANEXOS 34
2.1 Cálculos 34
2.1.1 Divisor de tensión 34
2.1.2 Circuito amplificador para la lectura de corrientes 35
2.1.3 Alimentación del dispositivo 36
2.2 Código Arduino Due 38
2.3 Código de la interfaz visual 47
2.4 Datasheet MAX31865 51
2.5 Datasheet DS3231 57
2.6 Datasheet LM321 62
2.7 Datasheet SN65176b 65
3 PLANOS 69
3.1 Esquema eléctrico Arduino Due 69
3.2 Layout Arduino Due 71
3.3 Esquema eléctrico Due Shield 72
3.4 Layout Due Shield 74
4 PLIEGO DE CONDICIONES 75
4.1 Condiciones técnicas y operativas 75
4.1.1 Equipamiento necesario 75
4.1.2 Directrices para el correcto funcionamiento del dispositivo. 75
5 MEDICIONES 77
6 PRESUPUESTO 79
6.1 Presupuesto de ejecución material (PEM) 79
6.2 Presupuesto de ejecución por contrata parcial (PEC) 81
6.3 Presupuesto de ejecución por contrata total 81

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Logo de la Universidad CEU Cardenal Herrera..... 10

Ilustración 2: Pila de combustible de alta temperatura utilizada en el prototipo del CEU-UCH..... 11

Ilustración 3: Ápeiron, el avión no tripulado alimentado por hidrógeno. 12

Ilustración 4: Captura de pantalla de Eagle 13

Ilustración 5: Captura de pantalla de Visual Studio Code..... 14

Ilustración 6: Captura de pantalla de Spyder (Anaconda) 14

Ilustración 7: Arduino Due 19

Ilustración 8: Sonda de temperatura PT100 de 3 hilos..... 20

Ilustración 9: Resistencias Shunt de diferentes tamaños y sensibilidades..... 20

Ilustración 10: Esquema de funcionamiento general del dispositivo 22

Ilustración 11: Montaje completo del dispositivo 23

Ilustración 12: Interfaz de usuario. Selección de puerto. 27

Ilustración 13: Interfaz de usuario. Menú general. 28

Ilustración 14: Interfaz de usuario. Gráfica de tensiones. 28

Ilustración 15: Divisor de tensión utilizado en el dispositivo..... 34

Ilustración 16: Circuito amplificador utilizado en las entradas de corriente..... 35

Ilustración 17: Esquema eléctrico final de las entradas de corriente..... 36

ÍNDICE DE TABLAS

Tabla 1: Esquema de conexiones del dispositivo. 23

Tabla 2: Configuración de los jumpers para las entradas de tensión. 24

Tabla 3: Configuración de los jumpers para las entradas de temperatura. 25

Tabla 4: Planificación del proyecto. Mes de julio. 29

Tabla 5: Planificación del proyecto. Mes de agosto. 30

Tabla 6: Planificación del proyecto. Mes de septiembre. 30

Tabla 7: Consumo máximo de los componentes alimentados a 3,3v. 36

Tabla 8: Consumo máximo de los componentes alimentados a 5v. 37

Tabla 9: Componentes electrónicos utilizados. 77

Tabla 10: Materiales y herramientas utilizados. 78

Tabla 11: Horas de trabajo empleadas. 78

Tabla 12: Coste de los componentes electrónicos. 79

Tabla 13: Coste de materiales y herramientas. 80

Tabla 14: Coste de Mano de Obra 80

Tabla 15: Coste de Ingeniería 80

Tabla 16: Presupuesto de Ejecución Material 80

Tabla 17: Presupuesto de ejecución por Contrata Parcial 81

Tabla 18: Presupuesto de ejecución por Contrata Total 81

1 MEMORIA

1.1 Objeto

El objetivo de este proyecto es el diseño y desarrollo de un sistema de adquisición de señales capaz de leer y monitorizar, a tiempo real, diferentes magnitudes tales como temperaturas, tensiones y corrientes con el fin de incorporarlo al prototipo de central de cogeneración basada en pila de combustible que se está desarrollando en la Universidad CEU Cardenal Herrera.

Ilustración 1: Logo de la Universidad CEU Cardenal Herrera.

El desarrollo de dicho sistema responde a la necesidad de substituir los actuales sistemas de captación de datos que se están utilizando en dicho proyecto de investigación, por un sistema personalizado y más optimizado que unifique todos los datos en una única plataforma.

1.2 Alcance

El ámbito de aplicación de este proyecto es principalmente, el campo de la investigación de nuevas fuentes de energía, en concreto, en el proyecto de investigación que se realiza en la universidad CEU-UCH a cerca de las pilas de combustible de alta temperatura alimentadas por hidrógeno.

El dispositivo que se va a desarrollar pretende satisfacer la monitorización integral de dicho sistema, aunque, dado su carácter general, podría utilizarse en cualquier ámbito que requiera de un dispositivo de monitorización de magnitudes eléctricas y térmicas.

A pesar de ello, el presente proyecto no contempla los algoritmos necesarios para el control de la pila de combustible, siendo puramente, un dispositivo de adquisición de datos.

1.3 Antecedentes

Las fuentes de energía convencionales han sido una pieza clave del desarrollo tecnológico durante muchísimas décadas dado que han abastecido de energía eléctrica a, prácticamente, cualquier lugar del mundo. Dichas fuentes poseen el inconveniente común de ser limitadas, dado que el consumo de la materia prima necesaria se realiza a un ritmo muy superior al de producción de las mismas.

A esto hay que sumar los problemas medioambientales derivados de su uso. En un escenario en el que cada vez es más evidente el cambio climático que está sufriendo el planeta, estas fuentes de energía suponen un problema dada la gran cantidad de emisiones de gases que expulsan a la atmosfera, derivados de la quema de energías fósiles, o los residuos peligrosos que producen, como en el caso de la energía de fisión nuclear.

Por todo lo anteriormente mencionado, el desarrollo energético de los últimos años está centrando sus esfuerzos en encontrar nuevas fuentes de energía que sean limpias e ilimitadas, como la energía solar, la eólica o, en este caso, el hidrógeno.

Ilustración 2: Pila de combustible de alta temperatura utilizada en el prototipo del CEU-UCH

En la Universidad CEU Cardenal Herrera, en concreto en el campus de Moncada, se está llevando a cabo un proyecto de investigación el cual pretende diseñar y construir un modelo de planta de cogeneración basado en pila de combustible de hidrógeno de alta temperatura.

Dicho modelo se desarrolla para implementarlo, entre otros, en modelos de viviendas eléctricamente autónomas y prototipos de vehículos no tripulados alimentados con hidrogeno, tales como aviones o barcos.

Ilustración 3: Ápeiron, el avión no tripulado alimentado por hidrógeno.

Este prototipo requiere una monitorización constante de las magnitudes de control de la pila para su correcto funcionamiento. Por ello, el proyecto aquí descrito tiene como objetivo cubrir dicha necesidad, mediante el diseño del hardware y software necesario para ello.

1.4 Normas y referencias

1.4.1 Disposiciones legales y normas aplicadas

Para la realización de este proyecto se ha seguido la normativa que se enumera a continuación:

- UNE-EN-157001: Criterios generales para la elaboración de proyectos.
- UNE-EN ISO 9000: Sistemas de gestión de la calidad. Fundamentos y vocabulario.
- UNE 1032:1982: Dibujos técnicos. Principios de representación.
- UNE 1-027-95: Dibujos técnicos. Plegado de planos.
- UNE 1039:1994: Dibujos técnicos. Acotación, Principios generales, métodos de ejecución e indicaciones especiales.
- UNE 1-035-95: Dibujos técnicos. Cuadros de rotulación.

1.4.2 Programas de desarrollo

Para el desarrollo de este proyecto en todas sus etapas han sido utilizados el siguiente software informático:

- **Eagle:** Este potente software de diseño electrónico propiedad de Autodesk permite realizar todo el desarrollo de cualquier placa PCB al completo, desde el diseño de su esquema eléctrico hasta la disposición y conexión de los componentes en la placa, permitiendo además exportar los archivos GERBERS de la misma para su posterior producción.

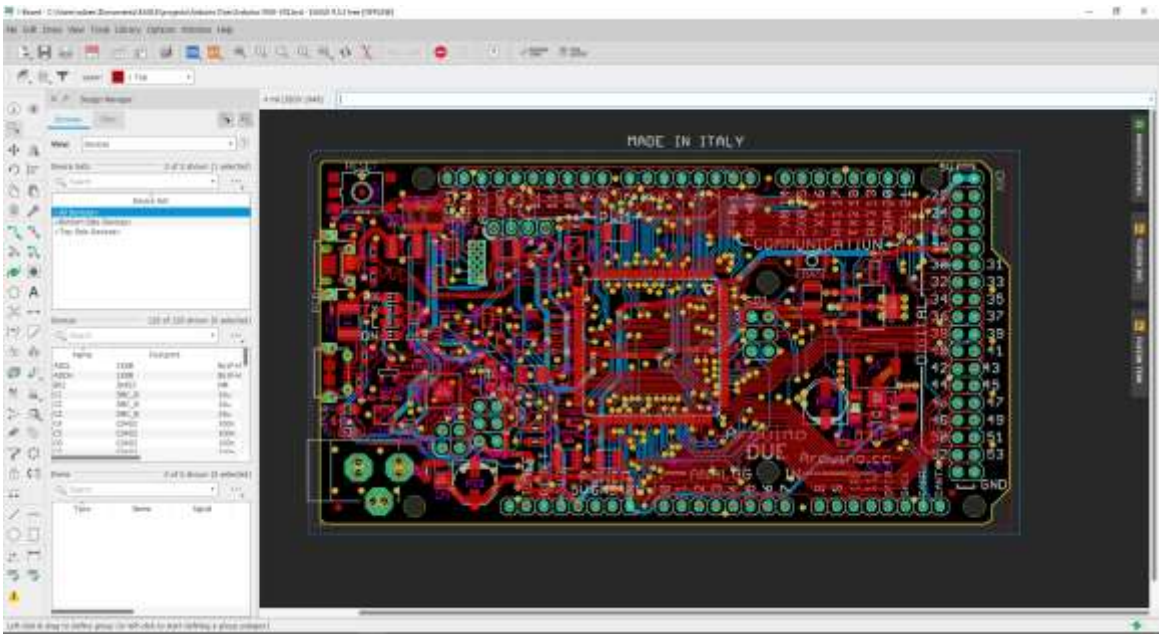


Ilustración 4: Captura de pantalla de Eagle

- **Visual Studio Code:** Software libre de desarrollo de código compatible con prácticamente cualquier lenguaje y plataforma. Su intuitiva interfaz visual lo convierte en una herramienta perfecta para gestionar códigos extensos.

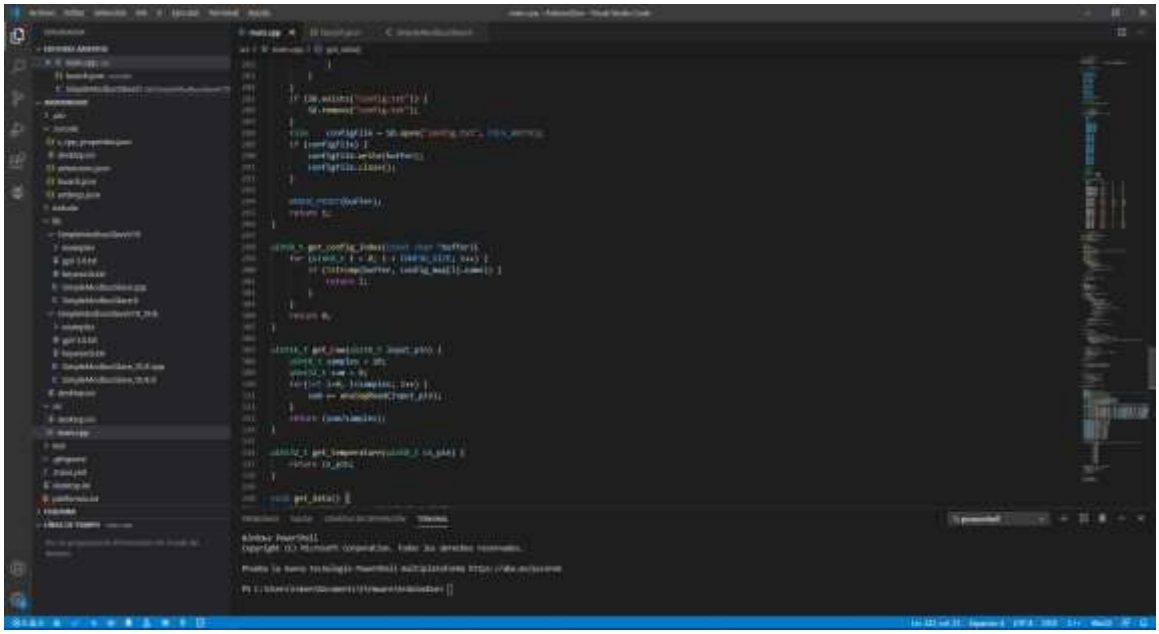


Ilustración 5: Captura de pantalla de Visual Studio Code

- **Spyder:** Entorno de desarrollo de código abierto para Python integrado en Anaconda orientado a científicos, analistas de datos e ingenieros dadas sus potentes herramientas de gestión de código.

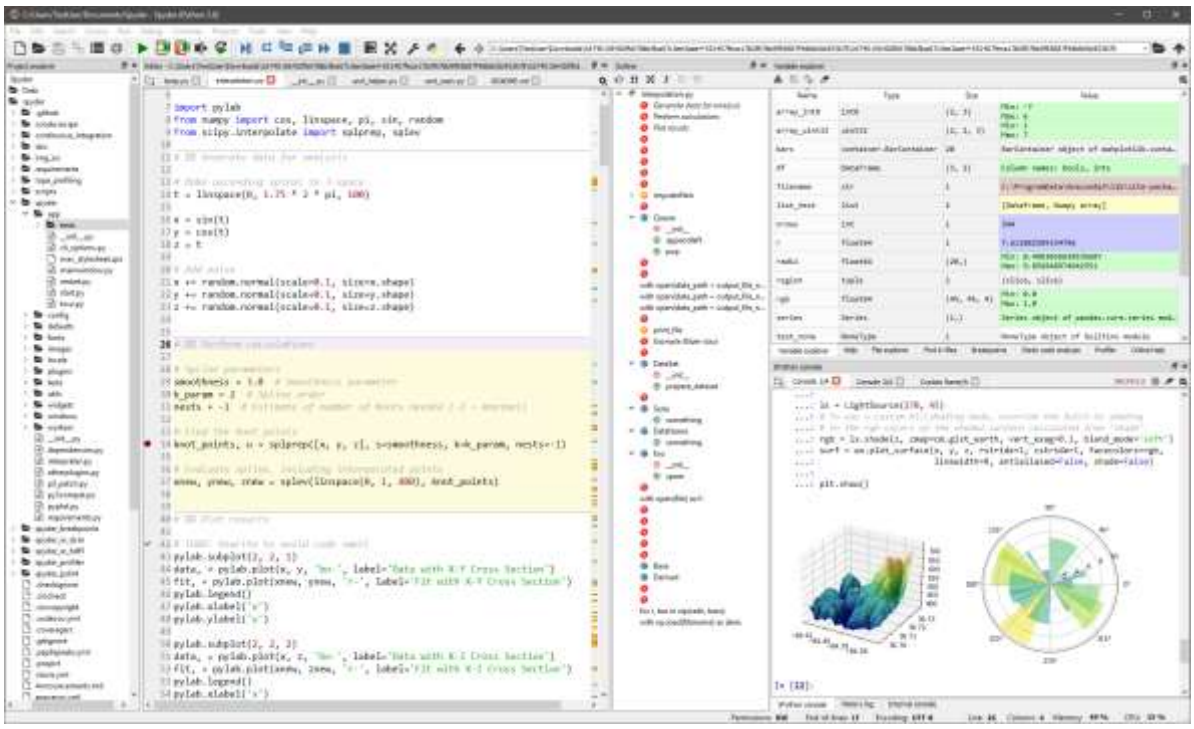


Ilustración 6: Captura de pantalla de Spyder (Anaconda)

1.4.3 Bibliografía

- http://energetica21.com/noticia/nuevo-sistema-para-la-cogeneracin-de-calor-y-electricidad-basado-en-una-pila-de-combustible-que-consume-hidrgeno
- https://www.digikey.es/
- https://jlcpcb.com/
- https://store.arduino.cc/arduino-due
- https://datasheets.maximintegrated.com/en/ds/DS3231.pdf
- https://datasheets.maximintegrated.com/en/ds/MAX31865.pdf
- https://www.ti.com/lit/ds/symlink/lm321.pdf?ts=1602633145134&ref_url=https%253A%252F%252Fwww.google.com%252F
- https://www.ti.com/lit/ds/symlink/sn65176b.pdf?ts=1602633186581&ref_url=https%253A%252F%252Fwww.google.com%252F
- https://guia-tkinter.readthedocs.io/es/develop/
- https://programacion.net/articulo/introduccion_a_la_libreria_matplotlib_de_python_1599
- Apuntes de la asignatura EE1018 – Electrónica
- Apuntes de la asignatura EE1034 – Programación de sistemas
- Apuntes de la asignatura EE1035 – Instrumentación, Medida i Tratamiento de Señal

1.5 Definiciones y abreviaturas

1.5.1 Definiciones

- **Arduino:** Es una plataforma de código libre que desarrolla hardware y software conocida por sus microcontroladores.
- **Diodo Zener:** Diodo de silicio diseñado para trabajar en la zona de ruptura donde, al superar una tensión máxima, el diodo abre el paso a la corriente.
- **Gerbers:** Archivos generados por los programas de diseño de PCBs que contienen la información por capas de su composición, necesarios para la fabricación de las mismas.
- **Hardware:** Nombre que recibe el conjunto de componentes físicos de un dispositivo electrónico.
- **I2C:** Bus de comunicación serie diseñado para la comunicación entre máquinas.
- **Jumpers:** Componente electrónico con dos o más terminales que permite abrir o cerrar un circuito eléctrico, bien sea mediante un puente de un material conductor, como por medio de soldadura de estaño.
- **Matlab:** Programa informático de cálculo matemático muy utilizado en el campo de la ciencia y la tecnología.
- **Microcontrolador:** Es un circuito integrado programable que incluye en su interior memoria, cpu y periféricos.
- **Modbus:** Es un protocolo de comunicación industrial de arquitectura maestro/esclavo que permite una sencilla comunicación entre máquinas.
- **Pila de combustible:** Es un dispositivo en el cual se produce una reacción química controlada para producir energía eléctrica mediante un flujo continuo de los reactivos que la alimentan.
- **Python:** Lenguaje de programación interpretado de código abierto multiplataforma.
- **Shield:** Tarjeta electrónica que complementa a un microcontrolador encajando sobre los pines de este.
- **Shunt:** Resistencia de valor mínimo que permite el paso de electricidad produciendo una caída de tensión reducida entre sus bornes. Midiendo dicha tensión se puede calcular la intensidad que la atraviesa.
- **Software:** Conjunto de procesos instalados en una computadora que le permiten realizar determinadas tareas.

1.5.2 Abreviaturas

- **CEU-UCH:** CEU – Universidad Cardenal Herrera.
- **CPU:** Unidad central de proceso (del inglés *Central Process Unit*).
- **COM:** Puerto de comunicación del pc.
- **PC:** Ordenador personal (del inglés *Personal Computer*).
- **PCB:** Placa de circuito impreso (del inglés *Printed Circuit Board*).
- **RTC:** Reloj de tiempo real (del inglés *Real Time Clock*).
- **SD:** Secure Digital. Nombre que reciben las targetas de memoria de este tipo.
- **SPI:** Interfaz de periféricos síncrona (del inglés *Syncronous Peripheral Interface*).
- **USB:** Bus serie universal (del inglés *Universal Serial Bus*).

1.6 Requisitos de diseño

El prototipo de central eléctrica de cogeneración basada en pila de combustible que se está desarrollando en la UCH-CEU necesita, para su correcto funcionamiento, un dispositivo capaz de obtener las magnitudes necesarias para definir su estado en diferentes zonas de dicha instalación. Dicho dispositivo debe ser capaz de mostrar a tiempo real la evolución de las mismas y almacenarlas para su posterior análisis.

Según su carácter, podemos clasificar y definir los requisitos necesarios para dicho dispositivo en los siguientes subgrupos:

1.6.1 Adquisición de datos

- Es necesario contar, como mínimo, con entradas preparadas para leer:
  - 3 señales de tensión que podrán oscilar entre 0 y 25 voltios
  - 3 señales de corriente que pueden oscilar entre 0 y 20 amperios
  - 3 señales de temperatura que pueden oscilar entre 0 y 150ºC
  - 1 señal de temperatura ambiente
- Dichas entradas deben poder leerse de forma simultánea e independiente.
- El usuario debe ser capaz de calibrar dichas entradas antes de cada uso para asegurar su correcto funcionamiento.

1.6.2 Comunicaciones

- El dispositivo debe ser capaz de mostrar a tiempo real los datos obtenidos en un PC, por lo que deberá establecerse un método de conexión con este.
- El sistema de comunicación debe basarse en el protocolo de comunicación industrial MODBUS para permitir una mayor versatilidad a la hora de interaccionar con otros dispositivos industriales, si fuera necesario.

1.6.3 Almacenamiento

- El dispositivo debe ser capaz de funcionar de forma autónoma sin la necesidad de estar conectado a un PC o a cualquier otro dispositivo vía MODBUS. Para ello, deberá contar con una ranura SD para almacenar la captura de datos.
- Dicho almacenamiento de datos debe realizarse en un archivo de texto plano en formato .csv y con una marca temporal que permita identificar el momento de la medición.
- El dispositivo debe contar también con la capacidad de almacenar los parámetros de configuración cuando este se encuentre apagado, recuperando dichos valores cuando vuelva a iniciarse.

1.6.4 Interfaz de usuario

- El sistema deberá contar con una interfaz de monitorización compatible con Windows donde visualizar las señales capturadas a tiempo real, pudiendo activar/desactivar las señales que en un momento dado se encuentren desconectadas o no sean de utilidad.
- Debe poseer una pantalla de calibración donde, para cada señal, el usuario sea capaz de ajustar el sensor para su correcto funcionamiento.
- Dicha interfaz debe comunicar con el dispositivo vía USB, usando el protocolo MODBUS.

1.7 Análisis de soluciones

1.7.1 Configuración general del dispositivo

Para cumplir con los requisitos de diseño, se necesita, básicamente, una unidad central de proceso y una serie de circuitos electrónicos que adapten las señales provenientes de los sensores a las entradas de dicha CPU.

En un primer momento, se planteó la opción de integrar las dos partes en la misma placa. Esto reduciría notablemente los costes de producción para grandes lotes y dotaría al dispositivo de un tamaño más reducido, a costa de incrementar los costes de diseño.

Dado que no se contempla la producción en masa y el tamaño del dispositivo no es un problema a tener en cuenta, se ha optado por utilizar un microcontrolador comercial como unidad de proceso y diseñar un Shield que contenga todos los elementos necesarios para la conexión directa de los sensores a este.

1.7.2 Unidad central de proceso

Se necesita una unidad central de procesos que sea capaz de leer tantas entradas simultáneas como se definen en los requisitos de diseño, además de contar con suficientes puertos de comunicación para intercambiar datos con el resto de hardware.

De entre todo el abanico de opciones posibles, se focalizó principalmente en los productos de la familia Arduino, dado que cuentan con una excelente relación calidad-precio y, gracias a su popularidad, cuentan con infinidad de documentación en la red que facilita su programación. La mayoría de estas tarjetas no poseen grandes procesadores, ni procesador gráfico, ni conexión de red, como el caso de las Raspberry Pi pero, tampoco son características necesarias para este proyecto dado que la CPU solo debe leer datos, procesarlos y enviarlos al dispositivo que se encargará de la monitorización.

Ilustración 7: Arduino Due

Solo dos productos de la familia Arduino poseen suficientes entradas analógicas para satisfacer los requisitos de diseño: Arduino Mega y Arduino Due. Se toma la decisión de utilizar este último, puesto que posee conversores AD de 12 bits frente a los 8 bits de Arduino Mega. Característica que dotará al dispositivo de mayor precisión a la hora de adquirir datos.

1.7.3 Lectura de temperaturas

La lectura de temperaturas se realiza con sensores PT100/PT1000. Dichos sensores producen cambios muy pequeños de su resistencia interna al variar su temperatura, por lo que se necesita un circuito amplificador que eleve sus salidas a los valores de entrada de Arduino.

En un primer momento se estudió la posibilidad de realizar dicho circuito mediante operacionales pero, si se desea una compatibilidad con PT100 y PT1000 tanto de 2, 3 y 4 hilos, el circuito se complica y aumenta su tamaño en placa.

Ilustración 8: Sonda de temperatura PT100 de 3 hilos

Por ello, finalmente se optó por recurrir al circuito integrado MAX31865. Este integrado permite la lectura de resistencias PT100/PT1000 de 2, 3 y 4 hilos con una resolución de 15 bits, además de contar con salida SPI para comunicar bidireccionalmente con la CPU.

1.7.4 Lectura de corrientes

En cuanto a la lectura de corrientes, se focalizó en las sondas de efecto Hall puesto que no son invasivas, pero sí suficientemente precisas. A pesar de ello, estos sensores poseían el problema de captar mucho ruido electromagnético a causa de su principio de funcionamiento, por lo que fueron descartados y substituidos por resistencias Shunt. Resistencias que, a pesar de ser mínimamente invasivas alterando la resistencia total del circuito, permiten una lectura mucho más limpia en cualquier entorno, puesto que no son vulnerables a las ondas electromagnéticas.

Ilustración 9: Resistencias Shunt de diferentes tamaños y sensibilidades

1.7.5 Protocolo de comunicación

El dispositivo necesita comunicarse con el pc para transmitirle los datos leídos. Si el número de señales hubiera sido menor, se habría optado por una transmisión mediante cadenas de caracteres a través del puerto COM, conectado por USB. Pero, puesto que el número de señales es elevado, se desea una comunicación bidireccional maestro-esclavo y se busca la versatilidad del dispositivo, se decide finalmente implementar un protocolo MODBUS, permitiendo, además, la comunicación con otros dispositivos industriales que también lo posean.

1.7.6 Software

La interfaz visual de usuario se planteó inicialmente en Matlab dada la facilidad de programación y cálculo que permite con sus múltiples herramientas, pero dado que no es un programa gratuito, se decidió utilizar Python, por ser un lenguaje multiplataforma, de licencia abierta, y que no requiere la instalación de ningún programa para la ejecución de sus códigos, tan solo poseer el intérprete de Python, que en dispositivos Linux ya viene de serie.

1.8 Resultados finales

Definidas las soluciones adoptadas para el diseño del dispositivo, pueden definirse las características generales de este.

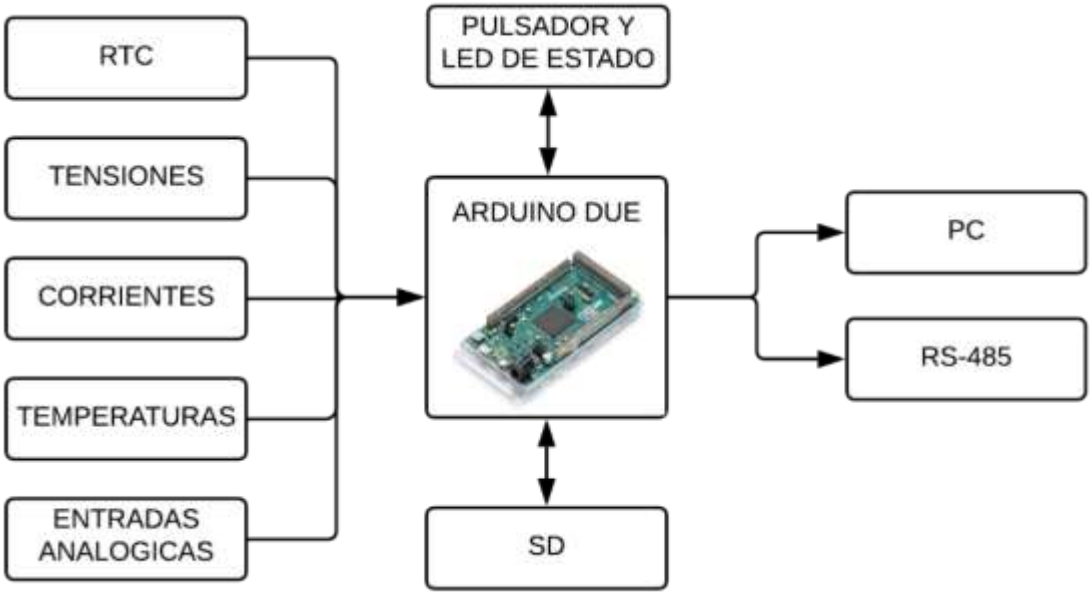


Ilustración 10: Esquema de funcionamiento general del dispositivo

En términos generales, se ha diseñado un dispositivo que posee cuatro entradas para la lectura de tensiones, cuatro entradas para la lectura de corrientes, cuatro entradas para la lectura de temperaturas y cuatro entradas analógicas para uso de carácter general. A todo esto, se le suma un RTC que proporciona la marca temporal a dichas lecturas. Para su control, el dispositivo incorpora un pulsador y un led que permite al usuario activar y desactivar el guardado de datos y visualizar su estado respectivamente.

La tarjeta SD permite el almacenamiento de los datos obtenidos, así como la lectura y escritura de las variables de configuración. Finalmente, el dispositivo incorpora conexión con protocolo MODBUS tanto USB como RS-485 para la visualización de las lecturas.

A continuación, se exponen con detalle las características de cada uno de los elementos que lo conforman.

1.8.1 Hardware general

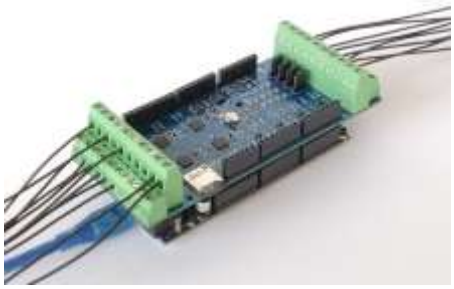
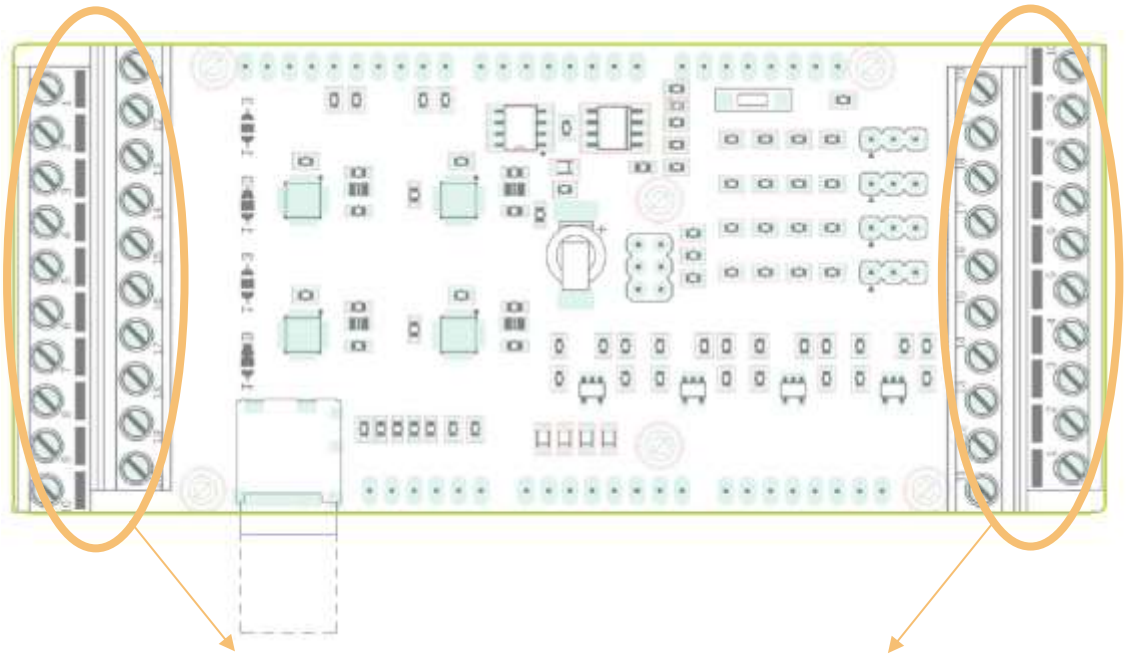


Ilustración 11: Montaje completo del dispositivo

El dispositivo está formado por una shield que se acopla a un Arduino Due a través de los pines. En ella se encuentran todos los circuitos electrónicos que se describirán a continuación y que permiten la conexión de los sensores con Arduino Due. El conexionado de estos sensores se realiza mediante 40 bornes a tornillo tal y como se indica en el siguiente esquema.



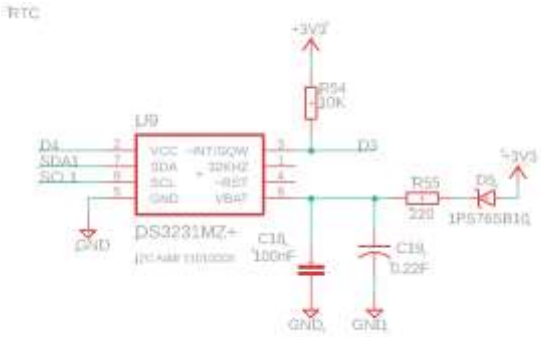
IZQUIERDA	
1. T1_FORCE+	11. T1_RTDIN+
2. T1_FORCE-	12. T1_RTDIN-
3. T2_FORCE+	13. T2_RTDIN+
4. T2_FORCE-	14. T2_RTDIN-
5. T3_FORCE+	15. T3_RTDIN+
6. T3_FORCE-	16. T3_RTDIN-
7. T4_FORCE+	17. T4_RTDIN+
8. T4_FORCE-	18. T4_RTDIN-
9. GND	19. RS485_A
10. VIN	20. RS485_B

DERECHA	
20. V4_IN	10. GND
19. V3_IN	9. GND
18. V2_IN	8. GND
17. V1_IN	7. GND
16. SHUNT4_V+	6. SHUNT4_V-
15. SHUNT3_V+	5. SHUNT3_V-
14. SHUNT4_V+	4. SHUNT2_V-
13. SHUNT4_V+	3. SHUNT1_V-
12. ANALOG_IN_9	2. ANALOG_IN_8
11. ANALOG_IN_11	1. ANALOG_IN_10

Tabla 1: Esquema de conexiones del dispositivo.

1.8.2 RTC

Los datos deben almacenarse con una marca temporal, dado que Arduino Due no posee reloj y el dispositivo tiene la posibilidad de funcionar desconectado del PC, se necesita un elemento que proporcione la fecha y hora al sistema y pueda mantenerla cuando este se desconecte. Para ello se ha añadido a la shield el RTC DS3231MZ+, un circuito integrado que cuenta con reloj, calendario y comunicación I2C, alimentado a través de un condensador de 0.22F que le permite mantenerse despierto, aunque se corte la alimentación de la placa.



1.8.3 Lectura de tensiones

El dispositivo cuenta con cuatro entradas para leer tensión. Cada una de estas, cuenta con un divisor resistivo de 3 escalas, permitiendo la lectura en tres rangos de tensión diferentes. La selección de las escalas se realiza mediante un jumper colocado en el lado derecho de la placa (junto a la entrada) y su configuración es la siguiente:

Esquema	Conexión	Rango de tensión
	Desconectado	0v <= Vin <= 50v
	Terminales central y derecho	0v <= Vin <= 25v
	Terminales central e izquierdo	0v <= Vin <= 12v

Tabla 2: Configuración de los jumpers para las entradas de tensión.

1.8.4 Lectura de corrientes

El dispositivo tiene preparadas cuatro entradas para leer corriente mediante resistencias shunt. La conexión de estas debe realizarse de acuerdo al esquema de conexiones respetando en todo momento la polaridad de estas. El circuito amplificador cuenta con una ganancia de 51, lo que quiere decir que si, por ejemplo, se conecta una resistencia Shunt de 10A/60mV, se obtienen 3.06v a la entrada del dispositivo. Dado que una selección o conexión incorrecta de la Shunt pueden provocar tensiones negativas o superiores a las máximas de entrada de Arduino, se han protegido estas con un Diodo Zener.

1.8.5 Lectura de temperaturas

El dispositivo cuenta con cuatro entradas para leer temperaturas mediante sondas RTD PT100 de 2, 3 y 4 hilos. Esta lectura se realiza mediante el circuito integrado MAX31865, que permite la lectura de dichos sensores con una resolución de 15 bits y comunica con la CPU por medio de SPI.

En la parte trasera de la shield, se encuentran 4 grupos de jumpers soldables encargados de la configuración de las entradas. Según el número de hilos que disponga el sensor, se deberán soldar/desoldar de acuerdo con la siguiente tabla:

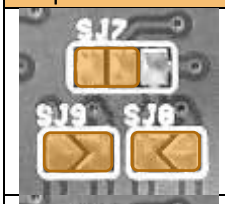
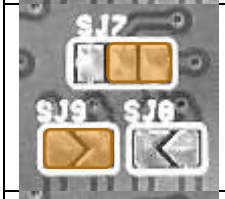
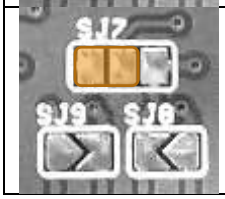
Esquema	Configuración Jumpers	Configuración hilos
	Jumper superior conectado a la izquierda Jumpers de abajo conectados	2 hilos Conectados a RTDIN+ y RTDIN-
	Jumper superior conectado a la derecha Jumper izquierdo conectado Jumper derecho desconectado	3 hilos Conectar a RTDIN+ el lado con un hilo y a RTDIN- y FORCE- el lado con dos hilos
	Jumper superior conectado a la izquierda Jumpers inferiores desconectados	4 hilos Conectar los hilos de un lado a RTDIN+ y FORCE+ y los del otro a RTDIN- y FORCE-

Tabla 3: Configuración de los jumpers para las entradas de temperatura.

Si se deseara utilizar sensores PT1000, en lugar de PT100, se debería substituir la resistencia R5/R7/R9/R11 (según el circuito que se desee modificar, ver esquemático) por una resistencia de 4.3K.

1.8.6 Entradas analógicas

El dispositivo cuenta con 4 entradas analógicas conectadas a los pines 1, 2, 11 y 12 del bornero derecho. Estas entradas no están asignadas a ninguna magnitud en concreto, pero pueden usarse en cualquier momento si fueran necesarias. Cuentan con un rango de trabajo de 0 a 3,3v y una resolución de 12 bits.

1.8.7 Pulsador y led de estado

En la parte superior derecha, se ha incorporado a la shield un pulsador que permite activar y desactivar la grabación de lecturas en la SD. Para ello, basta con pulsarlo una vez para cambiar de un modo a otro.

El led azul que se encuentra a su lado, parpadeará si se encuentra adquiriendo datos. En caso contrario, este led permanecerá apagado.

1.8.8 Tarjeta SD

El dispositivo cuenta con una ranura push-pull para tarjeta SD. En esta tarjeta se almacenarán los datos adquiridos en un archivo de texto plano llamado "data.txt" para su posterior consulta. La tarjeta SD además contiene un archivo llamado "config.txt". En él se almacenan las variables de configuración que han cambiado su valor por defecto. Al inicio de cada sesión, el dispositivo carga desde este archivo su configuración y, cada vez que una de estas variables es modificada, se vuelve a guardar en el archivo.

Este archivo puede ser creado/modificado de forma manual por el usuario para establecer su propia configuración, siempre que se mantenga el formato correcto.

1.8.9 Salida MODBUS e interfaz de usuario

Finalmente, el dispositivo cuenta con salida MODBUS para comunicar con la interfaz de usuario o con otros dispositivos industriales.

Por un lado, se ha incorporado el circuito integrado SN65176D para adaptar uno de los puertos serie de Arduino Due al estándar RS-485. Por otro lado, el dispositivo puede comunicar mediante MODBUS directamente sobre la conexión USB. Por defecto viene configurada la salida a través de USB, en caso cambiar a la salida RS-485, bastaría con añadir en el archivo de configuración "config.txt" una línea con el texto "rs485=1"

La interfaz de usuario para diseñada para pc permite visualizar las variables capturadas por el dispositivo. Para ello, se deben seguir los siguientes pasos:

1- Comprobar que el pc tiene el intérprete de Python instalado.

2- Conectar el dispositivo al pc via USB (programming port) asegurándose de no tener el parámetro de configuración "rs485=1" en la SD.

3- Comprobar que el pc ha inicializado el dispositivo correctamente y en que puerto COM lo ha hecho.

4- Abrir el archivo "dataviewer.py"

5- Aparecerá una pantalla donde se debe indicar el puerto COM en el que se encuentra conectado el dispositivo. Una vez escrito el nombre del puerto, hacer clic en "conectar"

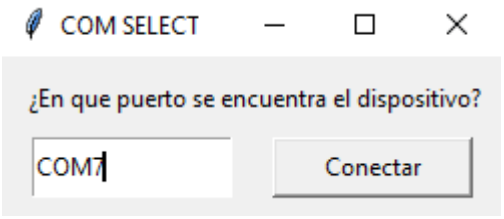


Ilustración 12: Interfaz de usuario. Selección de puerto.

6- Si el puerto se encuentra disponible y no hay ningún error de comunicación, aparecerá una nueva ventana donde aparecen las magnitudes disponibles para su visualización a tiempo real.

Ilustración 13: Interfaz de usuario. Menú general.

7- Al hacer clic sobre cualquiera de ellas, se abrirá una gráfica donde se mostrarán a tiempo real la evolución de todos los cuatro canales de esa magnitud.

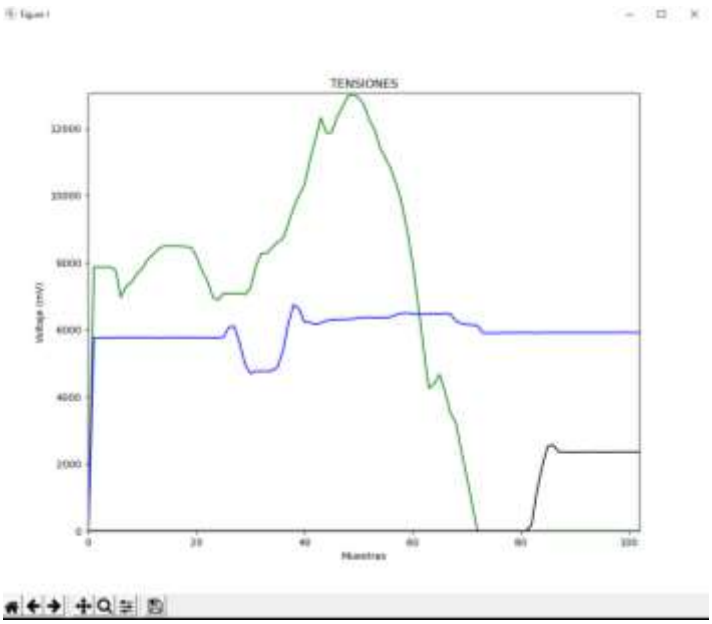


Ilustración 14: Interfaz de usuario. Gráfica de tensiones.

1.9 Planificación

La realización de este proyecto se ha dividido en las siguientes etapas:

1. Planteamiento del problema
2. Recopilación de información
3. Planteamiento de las diferentes soluciones
4. Análisis de soluciones
5. Selección de componentes
6. Diseño del esquema eléctrico
7. Disposición de los elementos en la placa y ruteado de señales
8. Fabricación de la placa
9. Soldadura de los componentes
10. Programación del código de Arduino Due
11. Programación de la Interfaz de usuario en Python
12. Redacción de los documentos del proyecto

JULIO

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1																															
2																															
3																															
4																															
5																															
6																															
7																															
8																															
9																															
10																															
11																															
12																															

Tabla 4: Planificación del proyecto. Mes de julio.

AGOSTO

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1																															
2																															
3																															
4																															
5																															
6																															
7																															
8																															
9																															
10																															
11																															
12																															

Tabla 5: Planificación del proyecto. Mes de agosto.

SEPTIEMBRE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1																															
2																															
3																															
4																															
5																															
6																															
7																															
8																															
9																															
10																															
11																															
12																															

Tabla 6: Planificación del proyecto. Mes de septiembre.

1.10 Orden de prioridad entre los documentos

El orden de prioridad entre los documentos para el actual proyecto se encuentra definido en la norma UNR 157001:2014 y establece la siguiente prioridad:

1. Planos
2. Pliego de condiciones
3. Presupuesto
4. Memoria

1.11 Conclusiones y trabajo futuro

En el proyecto presentado se ha realizado el diseño completo de una tarjeta de adquisición de datos, formada por un Arduino Due y un shield con todos los componentes electrónicos necesarios para la conexión directa de los sensores.

Se ha desarrollado el firmware interno del microcontrolador, con funcionalidades tales como la lectura de datos, el ajuste de los mismos en función de los parámetros de calibración, la escritura de datos en la SD, la carga de datos desde la SD y la comunicación mediante protocolo MODBUS mediante el puerto USB o RS-485.

También ha sido desarrollada una interfaz de usuario en lenguaje Python que permite la visualización a tiempo real de los datos capturados por el dispositivo.

Se puede concluir, por tanto, que se ha alcanzado el objetivo planteado al principio del proyecto ya que, gracias a este dispositivo, el proyecto de la universidad CEU Cardenal Herrera cuenta con un sistema de adquisición de datos con visualización a tiempo real más preciso y ajustado a sus necesidades.

Sin embargo, el dispositivo puede mejorarse añadiendo funcionalidades que podrían ser de gran ayuda y con las que aún no cuenta.

En primer lugar, el dispositivo no cuenta con filtrado de los datos adquiridos por las entradas analógicas. Una posible mejora sería establecer algoritmos de filtrado digital para limpiar las señales de posibles ruidos indeseados.

Por otro lado, el protocolo MODBUS ha sido diseñado teniendo en cuenta el intercambio de parámetros de calibración y ajuste, así como las funciones de control de la tarjeta SD, que también contemplan la lectura y escritura de dichos parámetros. Sin embargo, no ha sido desarrollado un entorno visual desde el cual pueda calibrarse con sencillez el dispositivo, quedando pendiente como tarea futura.

Así pues, la interfaz visual que permite la visualización de datos podría incorporar nuevas funcionalidades tales como: la activación/desactivación de señales que no se encuentran en uso, la posibilidad de calcular y graficar magnitudes derivadas de las captadas por el dispositivo, o el guardado de datos directamente sobre el disco duro del PC.

2 ANEXOS

2.1 Cálculos

2.1.1 Divisor de tensión

El circuito de entrada para la lectura de tensiones consiste en un divisor resistivo de 3 etapas como el que se muestra en la imagen.

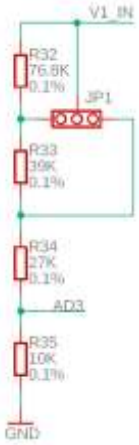


Ilustración 15: Divisor de tensión utilizado en el dispositivo.

El cálculo de las resistencias de este divisor se ha realizado mediante las tensiones máximas que puede admitir. Tomando como partida que la tensión máxima de la entrada analógica de Arduino Due (AD3) es de 3,3v, despreciando la corriente que pueda absorber esta y estableciendo que R35=10kΩ; el cálculo de las resistencias restantes se puede obtener mediante la ley de ohm.

$$I = \frac{V}{R} \tag{1}$$

Haciendo uso de esta fórmula, se obtienen valores de resistencias con decimales que deben aproximarse a los valores comerciales de estas. Para evitar problemas a la entrada de Arduino, se han redondeado los valores al alza, aumentando mínimamente las tensiones máximas admitidas en cada escalón.

Las resistencias obtenidas tras realizar el cálculo son:

R32 =$76.8k\Omega$ R33 =$39k\Omega$ R34 =$27k\Omega$ R35 =$10k\Omega$

Por tanto, la tensión máxima para cada escalón será:

$$V1_{max} = \frac{3.3v}{R35}(R35 + R34) = 12{,}21V \tag{2}$$

$$V2_{max} = \frac{3.3v}{R35}(R35 + R34 + R33) = 25{,}08V \tag{3}$$

$$V3_{max} = \frac{3.3v}{R35}(R35 + R34 + R33 + R32) = 50{,}424V$$

(4)

Y la corriente máxima residual:

$$I_{max} = \frac{3.3v}{R35} = 0.33mV$$

(5)

2.1.2 Circuito amplificador para la lectura de corrientes

Para la correcta lectura de las corrientes a través de las resistencias Shunt, se debe utilizar un circuito amplificador. Para este caso concreto se ha utilizado el esquema siguiente, que corresponde a un amplificador de tensión diferencial.

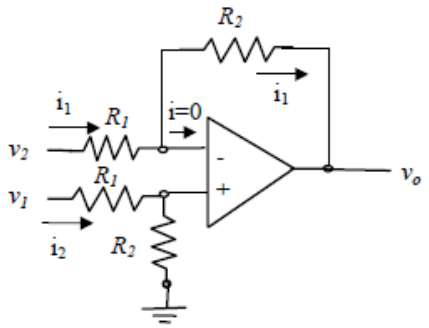


Ilustración 16: Circuito amplificador utilizado en las entradas de corriente.

En este circuito con realimentación negativa se obtiene que:

$$v^+ = v_1 \frac{R_2}{R_1 + R_2} = v^-$$

(6)

$$i_1 = \frac{(v_2 - v^-)}{R_1} = \frac{(v^- - v_o)}{R_2}$$

(7)

Unificando estas dos fórmulas y despejando v_0 se obtiene una tensión de salida:

$$v_o = \frac{R_2}{R_1}(v_1 - v_2)$$

(8)

Se determina por tanto que la ganancia del circuito es:

$$Gain = \frac{R_2}{R_1}$$

(9)

Tomando como base que se desea leer corrientes una resistencia shunt de 20A/60mV, la tensión diferencial máxima de la resistencia será de 60mV con 20 amperios de corriente. Como la tensión máxima admisible a la entrada es de 3,3v, la ganancia máxima será, por tanto:

$$Gain_{max} = \frac{R_2}{R_1} = \frac{3{,}3V}{0{,}06V} = 5{,}5$$

(10)

Despejando la anterior ecuación y buscando valores de resistencias comerciales, se concluyen los siguientes valores:

R1 =$1k\Omega$ R2 =$51k\Omega$

Donde finalmente la ganancia total del circuito solo sería de 51, pero seguiría cumpliendo con los requisitos de diseño.

Finalmente, se ha añadido una resistencia de pull-down con valor de $10k\Omega$ a la entrada positiva del shunt para evitar valores ambiguos cuando esta se encuentre desconectada. Así como un diodo Zener a la salida que proteja las entradas analógicas de Arduino Due en caso de sobrepasar la tensión limite o de producirse tensiones negativas.

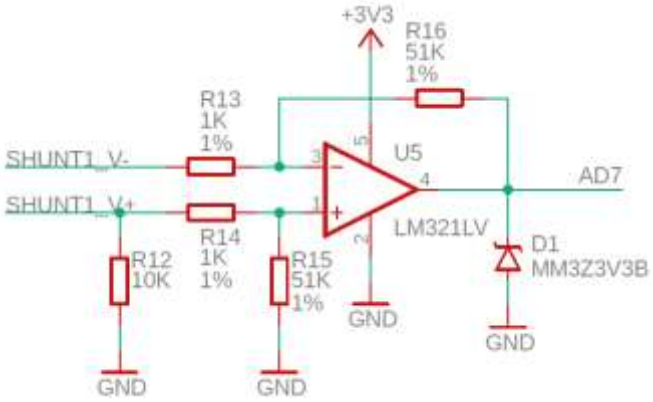


Ilustración 17: Esquema eléctrico final de las entradas de corriente.

2.1.3 Alimentación del dispositivo

El dispositivo cuenta con unos componentes electrónicos activos en la shield que necesitan ser alimentados. Dado que las salidas de tensión de Arduino Due tienen sus limitaciones de corriente, se va a proceder al cálculo de las corrientes máximas que consumen los componentes que necesitan ser alimentados.

Para una alimentación a 3,3v se obtiene el siguiente calculo:

Consumo máximo de los componentes de la placa alimentados a 3,3V			
Concepto	Unidades	Consumo unitario (mA)	Consumo total (mA)
SD CARD	1	200	200
MAX31865	4	3,5	14
LM321LV	4	50	200
DS3231M+	1	0,3	0,3
Consumo total a 3,3v (mA)			414,3

Tabla 7: Consumo máximo de los componentes alimentados a 3,3v

Dado que la salida de 3,3v de Arduino Due según sus especificaciones puede suministrar hasta 800mA, se puede afirmar que no se necesita alimentación externa a 3,3v.

Para una alimentación a 5v se obtiene el siguiente calculo:

Consumo máximo de los componentes de la placa alimentados a 5V			
Concepto	Unidades	Consumo unitario (mA)	Consumo total (mA)
SN65176b	1	70	70
Consumo total a 5v (mA)			70

Tabla 8: Consumo máximo de los componentes alimentados a 5v

Dado que la salida de 5v de Arduino Due según sus especificaciones puede suministrar hasta 800mA, se puede afirmar que no se necesita alimentación externa a 5v.

Finalmente, se procede a calcular la potencia máxima requerida por los componentes anteriormente mencionados, más el consumo de Arduino Due, para comprobar el consumo total del sistema.

La potencia total requerida de los componentes sería la siguiente:

$$P_{Comp} = V_{3.3v} \cdot I_{3.3v} + V_{5v} \cdot I_{5v} = 1717{,}19mW \quad (11)$$

La potencia consumida por Arduino es de 500mW, que sumada al anterior valor hace un consumo total de:

$$P_{total} = 2217.19mW$$

Teniendo en cuenta que un USB 2.0 común proporciona un máximo de 500mA a 5v se obtendría una potencia de alimentación total de 2500mW por lo que el dispositivo podría ser alimentado perfectamente a través del cable USB, no necesitando ninguna fuente de alimentación externa.

2.2 Código Arduino Due

```
#include <Arduino.h>
#include <SPI.h>
#include <SD.h>
#include <stdio.h>
#include <string.h>
#include <SimpleModbusSlave_DUE.h>
#include <Adafruit_MAX31865.h>
#include <DS3231.h>
#include <Wire.h>


File configFile;
File dataFile;


#define SD_CS_PIN               (5)

#define TEMPERATURE_1_CS_PIN    (9)
#define TEMPERATURE_2_CS_PIN    (13)
#define TEMPERATURE_3_CS_PIN    (8)
#define TEMPERATURE_4_CS_PIN    (12)

#define TEMPERATURE_1_RD_PIN    (6)
#define TEMPERATURE_2_RD_PIN    (10)
#define TEMPERATURE_3_RD_PIN    (7)
#define TEMPERATURE_4_RD_PIN    (11)

#define VOLTAGE_1_PIN           (A0)
#define VOLTAGE_2_PIN           (A1)
#define VOLTAGE_3_PIN           (A2)
#define VOLTAGE_4_PIN           (A3)

#define CURRENT_1_PIN           (A4)
#define CURRENT_2_PIN           (A5)
#define CURRENT_3_PIN           (A6)
#define CURRENT_4_PIN           (A7)

#define ANALOG_1_PIN            (A8)
#define ANALOG_2_PIN            (A9)
#define ANALOG_3_PIN            (A10)
#define ANALOG_4_PIN            (A11)

#define LED_PIN                 (14)
#define SWITCH_PIN              (15)



//      **********************
//              DEBUG
//      **********************
#define DBG_SERIAL Serial1
#define DEBUG_ENABLE        (1)
#define DEBUG_PRINT(s__)            if (DEBUG_ENABLE) {DBG_SERIAL.print(s__);}
#define DEBUG_PRINTLN(s__)          if (DEBUG_ENABLE) {DBG_SERIAL.println(s__);}
```

```
//      **********************
//              MODBUS
//      **********************

enum
{
    modbus_temperature_1,
    modbus_temperature_2,
    modbus_temperature_3,
    modbus_temperature_4,

    modbus_voltage_1,
    modbus_voltage_2,
    modbus_voltage_3,
    modbus_voltage_4,

    modbus_current_1,
    modbus_current_2,
    modbus_current_3,
    modbus_current_4,

    modbus_analog_1,
    modbus_analog_2,
    modbus_analog_3,
    modbus_analog_4,

    modbus_calibration_channel,
    modbus_calibration_value,

    HOLDING_REGS_SIZE
};

unsigned int holdingRegs[HOLDING_REGS_SIZE];



//      **********************
//              CONFIG
//      **********************
    uint8_t  save_config();
    uint8_t  get_config();
    uint8_t  get_config_index(const char *buffer);
    uint8_t  set_zero();

    typedef enum {
        SAMPLERATE  =1,
        RS485,
        SAVE,

        TEMPERATURE_1_ZERO,
        TEMPERATURE_1_MAX,
        TEMPERATURE_2_ZERO,
        TEMPERATURE_2_MAX,
        TEMPERATURE_3_ZERO,
        TEMPERATURE_3_MAX,
        TEMPERATURE_4_ZERO,
```

```
        TEMPERATURE_4_MAX,

        VOLTAGE_1_ZERO,
        VOLTAGE_1_MAX,
        VOLTAGE_2_ZERO,
        VOLTAGE_2_MAX,
        VOLTAGE_3_ZERO,
        VOLTAGE_3_MAX,
        VOLTAGE_4_ZERO,
        VOLTAGE_4_MAX,

        CURRENT_1_ZERO,
        CURRENT_1_MAX,
        CURRENT_2_ZERO,
        CURRENT_2_MAX,
        CURRENT_3_ZERO,
        CURRENT_3_MAX,
        CURRENT_4_ZERO,
        CURRENT_4_MAX,

        ANALOG_1_ZERO,
        ANALOG_1_MAX,
        ANALOG_2_ZERO,
        ANALOG_2_MAX,
        ANALOG_3_ZERO,
        ANALOG_3_MAX,
        ANALOG_4_ZERO,
        ANALOG_4_MAX,

        CONFIG_SIZE

    } config_addr_t;

    struct config{
        config_addr_t   index;
        const char      name[15];
        int             value;
        const int       def_value;
    };

    config config_map[CONFIG_SIZE] = {
        {},
        {SAMPLERATE,            "samplerate",       0,              0},
        {RS485,                 "rs485",            0,              0},
        {SAVE,                  "save",             0,              0},

        {TEMPERATURE_1_ZERO,    "t1_zero",          0,              0},
        {TEMPERATURE_1_MAX,     "t1_max",           0,              0},
        {TEMPERATURE_2_ZERO,    "t2_zero",          0,              0},
        {TEMPERATURE_2_MAX,     "t2_max",           0,              0},
        {TEMPERATURE_3_ZERO,    "t3_zero",          0,              0},
        {TEMPERATURE_3_MAX,     "t3_max",           0,              0},
        {TEMPERATURE_4_ZERO,    "t4_zero",          0,              0},
        {TEMPERATURE_4_MAX,     "t4_max",           0,              0},

        {VOLTAGE_1_ZERO,        "v1_zero",          0,              0},
        {VOLTAGE_1_MAX,         "v1_max",           33000,          33000},
```

```
        {VOLTAGE_2_ZERO,        "v2_zero",          0,              0},
        {VOLTAGE_2_MAX,         "v2_max",           33000,          33000},
        {VOLTAGE_3_ZERO,        "v3_zero",          0,              0},
        {VOLTAGE_3_MAX,         "v3_max",           33000,          33000},
        {VOLTAGE_4_ZERO,        "v4_zero",          0,              0},
        {VOLTAGE_4_MAX,         "v4_max",           33000,          33000},

        {CURRENT_1_ZERO,        "c1_zero",          0,              0},
        {CURRENT_1_MAX,         "c1_max",           0,              0},
        {CURRENT_2_ZERO,        "c2_zero",          0,              0},
        {CURRENT_2_MAX,         "c2_max",           0,              0},
        {CURRENT_3_ZERO,        "c3_zero",          0,              0},
        {CURRENT_3_MAX,         "c3_max",           0,              0},
        {CURRENT_4_ZERO,        "c4_zero",          0,              0},
        {CURRENT_4_MAX,         "c4_max",           0,              0},

        {ANALOG_1_ZERO,         "a1_zero",          0,              0},
        {ANALOG_1_MAX,          "a1_max",           0,              4095},
        {ANALOG_2_ZERO,         "a2_zero",          0,              0},
        {ANALOG_2_MAX,          "a2_max",           0,              4095},
        {ANALOG_3_ZERO,         "a3_zero",          0,              0},
        {ANALOG_3_MAX,          "a3_max",           0,              4095},
        {ANALOG_4_ZERO,         "a4_zero",          0,              0},
        {ANALOG_4_MAX,          "a4_max",           0,              4095},
    };

//      **********************
//           VARIABLES
//      **********************
uint16_t get_raw(uint8_t input_pin);
uint32_t get_temperature(uint8_t cs_pin);
void     get_data();
uint32_t get_unix_time ();
uint16_t provisional_time = 0;


//      **********************
//           MAX31865
//      **********************
Adafruit_MAX31865 thermo1 = Adafruit_MAX31865(TEMPERATURE_1_CS_PIN);
Adafruit_MAX31865 thermo2 = Adafruit_MAX31865(TEMPERATURE_2_CS_PIN);
Adafruit_MAX31865 thermo3 = Adafruit_MAX31865(TEMPERATURE_3_CS_PIN);
Adafruit_MAX31865 thermo4 = Adafruit_MAX31865(TEMPERATURE_4_CS_PIN);

#define RREF      430.0
#define RNOMINAL  100.0

//************************************


void setup() {
    //Comunicación
    DBG_SERIAL.begin(9600);
    delay(100);

    modbus_configure(&Serial, 9600, 1, 2, HOLDING_REGS_SIZE, holdingRegs);
    modbus_update_comms(9600, 1);
```

```
    //Entradas
    analogReadResolution(12);

    DEBUG_PRINTLN("DUE ON");
    delay (1000);
    //SD
    DEBUG_PRINTLN(get_raw(A0));
    if (SD.begin(5)) {
        DEBUG_PRINTLN("SD OK");
        if (!SD.exists("data")){
            if (SD.mkdir("data")){
                DEBUG_PRINTLN("data dir created");
            }
            else{
                DEBUG_PRINTLN("Error creating data dir");
            }
        }
        get_config();
    }
    else {
        DEBUG_PRINTLN("SD not found");
    }
    pinMode(14, OUTPUT);

    thermo1.begin(MAX31865_3WIRE);
    thermo2.begin(MAX31865_3WIRE);
    thermo3.begin(MAX31865_3WIRE);
    thermo4.begin(MAX31865_3WIRE);

}

void loop() {
    get_data();
    modbus_update();
    digitalWrite(14, 0);
    digitalWrite(14, 1);
}


//*************** FUNCIÓN DE CARGA DE LA CONFIGURACIÓN *****************
uint8_t get_config() {
    File configFile = SD.open("config.txt");
    char c[2]="";
    char buffer[32] = "";
    uint8_t index = 0;
        if (configFile) {
            while(configFile.available()) {
                c[0] = configFile.read();
                if (!strcmp(c, "=")) {
                    index = get_config_index(buffer);
                    buffer[0] = '\0';
                } else if (!strcmp(c, ";")) {
                    if (index > 0){
                        config_map[index].value = atoi(buffer);
                    }
                    buffer[0] = '\0';
                } else if (!strcmp(c, "\n")) {
```

```
                    buffer[0] = '\0';
                } else {
                    strcat(buffer, c);
                }
            }
            configFile.close();
            DEBUG_PRINTLN("Config loaded");
        }
        else {
            DEBUG_PRINTLN("error opening config.txt");
        }
    return 1;
}


//*************** FUNCIÓN DE GUARDADO DE CONFIGURACIÓN *****************
uint8_t save_config() {

    char    buffer[2048] = "";
    uint8_t buffer_len = 0;

    for (uint8_t i = 0; i < CONFIG_SIZE; i++) {
        if (config_map[i].value != config_map[i].def_value) {
            DEBUG_PRINTLN(i);
            buffer_len = sprintf(buffer, "%s%s=%u;\n", buffer, config_map[i].name,
config_map[i].value);
            if (buffer_len > 1000) {
                DEBUG_PRINTLN ("SD buffer overload");
                return 0;
            }
        }
    }
    if (SD.exists("config.txt")) {
        SD.remove("config.txt");
    }
    File    configFile = SD.open("config.txt", FILE_WRITE);
    if (configFile) {
        configFile.write(buffer);
        configFile.close();
    }

    DEBUG_PRINT(buffer);
    return 1;
}


//*************** ADQUISICIÓN DEL INDICE DE UN ELEMENTO CONFIG *****************
uint8_t get_config_index(const char *buffer){
    for (uint8_t i = 0; i < CONFIG_SIZE; i++) {
        if (!strcmp(buffer, config_map[i].name)) {
            return i;
        }
    }
    return 0;
}

//*************** ADQUISICIÓN DEL VALOR RAW DE UNA ENTRADA ANALOGICA *****************
```

```
uint16_t get_raw(uint8_t input_pin) {
    uint8_t samples = 10;
    uint32_t sum = 0;
    for(int i=0; i<samples; i++) {
        sum += analogRead(input_pin);
    }
    return (sum/samples);
}

//*************** ADQUISICIÓN DE TEMPERATURAS *****************
uint32_t get_temperature(uint8_t cs_pin) {
    switch (cs_pin)
    case TEMPERATURE_1_CS_PIN:
        uint16_t rtd1 = thermo1.readRTD();
        return thermo1.temperature(RNOMINAL, RREF);
        break;

    case TEMPERATURE_2_CS_PIN:
        uint16_t rtd1 = thermo1.readRTD();
        return thermo1.temperature(RNOMINAL, RREF);
        break;

    case TEMPERATURE_3_CS_PIN:
        uint16_t rtd1 = thermo1.readRTD();
        return thermo1.temperature(RNOMINAL, RREF);
        break;

    case TEMPERATURE_4_CS_PIN:
        uint16_t rtd1 = thermo1.readRTD();
        return thermo1.temperature(RNOMINAL, RREF);
        break;

    default:
        return 0;
}

//*************** ADQUISICIÓN DE DATOS Y GUARDADO EN SD *****************
void get_data() {

    int32_t t1_value = get_temperature(TEMPERATURE_1_CS_PIN);
    int32_t t2_value = get_temperature(TEMPERATURE_2_CS_PIN);
    int32_t t3_value = get_temperature(TEMPERATURE_3_CS_PIN);
    int32_t t4_value = get_temperature(TEMPERATURE_4_CS_PIN);

    int32_t v1_value = map(get_raw(VOLTAGE_1_PIN), 0, 4095, config_map[VOLTAGE_1_ZERO].value,
config_map[VOLTAGE_1_MAX].value);
    int32_t v2_value = map(get_raw(VOLTAGE_2_PIN), 0, 4095, config_map[VOLTAGE_2_ZERO].value,
config_map[VOLTAGE_2_MAX].value);
    int32_t v3_value = map(get_raw(VOLTAGE_3_PIN), 0, 4095, config_map[VOLTAGE_3_ZERO].value,
config_map[VOLTAGE_3_MAX].value);
    int32_t v4_value = map(get_raw(VOLTAGE_4_PIN), 0, 4095, config_map[VOLTAGE_4_ZERO].value,
config_map[VOLTAGE_4_MAX].value);

    int32_t c1_value = map(get_raw(CURRENT_1_PIN), 0, 4095, config_map[CURRENT_1_ZERO].value,
config_map[CURRENT_1_MAX].value);
    int32_t c2_value = map(get_raw(CURRENT_2_PIN), 0, 4095, config_map[CURRENT_2_ZERO].value,
config_map[CURRENT_2_MAX].value);
```

```
    int32_t c3_value = map(get_raw(CURRENT_3_PIN), 0, 4095, config_map[CURRENT_3_ZERO].value,
config_map[CURRENT_3_MAX].value);
    int32_t c4_value = map(get_raw(CURRENT_4_PIN), 0, 4095, config_map[CURRENT_4_ZERO].value,
config_map[CURRENT_4_MAX].value);

    int32_t a1_value = map(get_raw(ANALOG_1_PIN), 0, 4095, config_map[ANALOG_1_ZERO].value,
config_map[ANALOG_1_MAX].value);
    int32_t a2_value = map(get_raw(ANALOG_2_PIN), 0, 4095, config_map[ANALOG_2_ZERO].value,
config_map[ANALOG_2_MAX].value);
    int32_t a3_value = map(get_raw(ANALOG_3_PIN), 0, 4095, config_map[ANALOG_3_ZERO].value,
config_map[ANALOG_3_MAX].value);
    int32_t a4_value = map(get_raw(ANALOG_4_PIN), 0, 4095, config_map[ANALOG_4_ZERO].value,
config_map[ANALOG_4_MAX].value);

    holdingRegs[modbus_temperature_1] = t1_value;
    holdingRegs[modbus_temperature_2] = t2_value;
    holdingRegs[modbus_temperature_3] = t3_value;
    holdingRegs[modbus_temperature_4] = t4_value;
    holdingRegs[modbus_voltage_1] = v1_value;
    holdingRegs[modbus_voltage_2] = v2_value;
    holdingRegs[modbus_voltage_3] = v3_value;
    holdingRegs[modbus_voltage_4] = v4_value;
    holdingRegs[modbus_current_1] = c1_value;
    holdingRegs[modbus_current_2] = c2_value;
    holdingRegs[modbus_current_3] = c3_value;
    holdingRegs[modbus_current_4] = c4_value;
    holdingRegs[modbus_analog_1] = a1_value;
    holdingRegs[modbus_analog_2] = a2_value;
    holdingRegs[modbus_analog_3] = a3_value;
    holdingRegs[modbus_analog_4] = a4_value;

    char writeArray[2048] = "";
    sprintf(writeArray,
("%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;%lu;\n"),
                    get_unix_time(),
                    t1_value,
                    t2_value,
                    t3_value,
                    t4_value,
                    v1_value,
                    v2_value,
                    v3_value,
                    v4_value,
                    c1_value,
                    c2_value,
                    c3_value,
                    c4_value,
                    a1_value,
                    a2_value,
                    a3_value,
                    a4_value
                    );

    dataFile = SD.open("data.txt", FILE_WRITE);
```

```
    if(dataFile) {
        dataFile.print(writeArray);
    } else {
        DEBUG_PRINTLN("SD error");
    }
    dataFile.close();


}
//*************** ADQUISICIÓN DEL TIEMPO *****************
uint32_t get_unix_time() {
    provisional_time++;
    return provisional_time;
}
```

2.3 Código de la interfaz visual

```
import minimalmodbus
import serial
import time
import threading
import tkinter
import numpy as np

import matplotlib.pyplot as plt
import matplotlib.animation as animation
i = 0

gData = []
for n in list(range(13)):
    gData.append([0.0])


def com_connect():
    global entry0
    global label0
    global button0

    print(entry0.get())
    instrument = minimalmodbus.Instrument(entry0.get(),1)
    instrument.serial.baudrate = 9600
    instrument.serial.bytesize = 8
    instrument.serial.parity   = serial.PARITY_NONE
    instrument.serial.stopbits = 1
    instrument.serial.timeout  = 2
    instrument.mode = minimalmodbus.MODE_RTU
    instrument.clear_buffers_before_each_transaction = True
    instrument.close_port_after_each_call = False
    time.sleep(1)

    label0.place_forget()
    entry0.place_forget()
    button0.place_forget()

    def plot_temperature():
        fig_t = plt.figure(figsize=(10,8))

        hl_t1,  = plt.plot(gData[0], gData[1], 'r')
        hl_t2,  = plt.plot(gData[0], gData[2], 'g')
        hl_t3,  = plt.plot(gData[0], gData[3], 'b')
        hl_t4,  = plt.plot(gData[0], gData[4], 'k')
        plt.title("TEMPERATURAS")
        plt.xlabel("Muestras")
        plt.ylabel("Temperatura (°C)")

        ani_t = animation.FuncAnimation( fig_t, update_temperature, fargs = (hl_t1,
hl_t2, hl_t3, hl_t4, gData), interval = 100, blit = False)

        plt.show()

    def plot_voltage():
        fig_v = plt.figure(figsize=(10,8))
```

```
        hl_v1,  = plt.plot(gData[0], gData[5], 'r')
        hl_v2,  = plt.plot(gData[0], gData[6], 'g')
        hl_v3,  = plt.plot(gData[0], gData[7], 'b')
        hl_v4,  = plt.plot(gData[0], gData[8], 'k')
        plt.title("VOLTAGES")
        plt.xlabel("Muestras")
        plt.ylabel("Voltage (V)")

        ani_v = animation.FuncAnimation( fig_v, update_voltage, fargs = (hl_v1,
hl_v2, hl_v3, hl_v4, gData), interval = 100, blit = False)

        plt.show()

    def plot_current():
        fig_i = plt.figure(figsize=(10,8))

        hl_i1,  = plt.plot(gData[0], gData[9], 'r')
        hl_i2,  = plt.plot(gData[0], gData[10], 'g')
        hl_i3,  = plt.plot(gData[0], gData[11], 'b')
        hl_i4,  = plt.plot(gData[0], gData[12], 'k')
        plt.title("CORRIENTES")
        plt.xlabel("Muestras")
        plt.ylabel("Corriente (A)")

        ani_i = animation.FuncAnimation( fig_i, update_current, fargs = (hl_i1,
hl_i2, hl_i3, hl_i4, gData), interval = 100, blit = False)

        plt.show()


    def update_temperature(num, hl_t1, hl_t2, hl_t3, hl_t4, data):
        dx  = np.array(data[0])
        dt1 = np.array(data[1])
        dt2 = np.array(data[2])
        dt3 = np.array(data[3])
        dt4 = np.array(data[4])

        hl_t1.set_data(dx, dt1)
        hl_t2.set_data(dx, dt2)
        hl_t3.set_data(dx, dt3)
        hl_t4.set_data(dx, dt4)
        plt.ylim(min(min(data[1]), min(data[2]), min(data[3]), min (data[4])),
max(max(data[1]), max(data[2]), max(data[3]), max (data[4]))+20)
        plt.xlim(min(data[0]),max(data[0]))

        return hl_t1, hl_t2, hl_t3, hl_t4


    def update_voltage(num, hl_v1, hl_v2, hl_v3, hl_v4, data):
        dx  = np.array(data[0])
        dv1 = np.array(data[5])
        dv2 = np.array(data[6])
        dv3 = np.array(data[7])
        dv4 = np.array(data[8])

        hl_v1.set_data(dx, dv1)
```

```
        hl_v2.set_data(dx, dv2)
        hl_v3.set_data(dx, dv3)
        hl_v4.set_data(dx, dv4)


        plt.ylim(min(min(data[5]), min(data[6]), min(data[7]), min (data[8])),
max(max(data[5]), max(data[6]), max(data[7]), max (data[8]))+20)
        plt.xlim(min(data[0]),max(data[0]))

        return hl_v1, hl_v2, hl_v3, hl_v4


    def update_current(num, hl_i1, hl_i2, hl_i3, hl_i4, data):
        dx  = np.array(data[0])
        di1 = np.array(data[9])
        di2 = np.array(data[10])
        di3 = np.array(data[11])
        di4 = np.array(data[12])

        hl_i1.set_data(dx, di1)
        hl_i2.set_data(dx, di2)
        hl_i3.set_data(dx, di3)
        hl_i4.set_data(dx, di4)

        plt.ylim(min(min(data[9]), min(data[10]), min(data[11]), min (data[12])),
max(max(data[9]), max(data[10]), max(data[11]), max (data[12]))+20)
        plt.xlim(min(data[0]),max(data[0]))

        return hl_i1, hl_i2, hl_i3, hl_i4


    def getData(out_data):
        while True:
            global i
            i = i + 1
            t1 = instrument.read_register(0)
            t2 = instrument.read_register(1)
            t3 = instrument.read_register(2)
            t4 = instrument.read_register(3)
            v1 = instrument.read_register(4)
            v2 = instrument.read_register(5)
            v3 = instrument.read_register(6)
            v4 = instrument.read_register(7)
            i1 = instrument.read_register(8)
            i2 = instrument.read_register(9)
            i3 = instrument.read_register(10)
            i4 = instrument.read_register(11)

            gData[0].append(i)
            gData[1].append(t1)
            gData[2].append(t2)
            gData[3].append(t3)
            gData[4].append(t4)
            gData[5].append(v1)
            gData[6].append(v2)
            gData[7].append(v3)
            gData[8].append(v4)
```

```
            gData[9].append(i1)
            gData[10].append(i2)
            gData[11].append(i3)
            gData[12].append(i4)

            if len(gData[1]) > 200:
                for n in list(range(13)):
                    gData[n].pop(0)

    dataCollector = threading.Thread(target = getData, args = (gData,))
    dataCollector.start()


    root.geometry("150x180")
    tkinter.Label(root, text="¿Que desea visualizar?").place(x=15, y=15)
    tkinter.Button(root, text="TENSIONES", command=plot_voltage).place(x=20, y=50,
width=100, height=30)
    tkinter.Button(root, text="CORRIENTES", command=plot_current).place(x=20, y=90,
width=100, height=30)
    tkinter.Button(root, text="TEMPERATURAS", command=plot_temperature).place(x=20,
y=130, width=100, height=30)

    root.wm_title("DATA VIEWER")



root = tkinter.Tk()
root.wm_title("COM SELECT")
root.geometry("250x80")

label0 = tkinter.Label(root, text="¿En que puerto se encuentra el dispositivo?")
entry0 = tkinter.Entry(root)
button0 = tkinter.Button(root, text="Conectar", command=com_connect)

label0.place(x=10, y=10)
entry0.place(x=15, y=40, width=100, height=30)
button0.place(x=135, y=40, width=100, height=30)


entry0.get()
root.mainloop()

instrument.serial.close()
```

2.4 Datasheet MAX31865

MAX31865 **RTD-to-Digital Converter**

General Description

The MAX31865 is an easy-to-use resistance-to-digital converter optimized for platinum resistance temperature detectors (RTDs). An external resistor sets the sensitivity for the RTD being used and a precision delta-sigma ADC converts the ratio of the RTD resistance to the reference resistance into digital form. The MAX31865's inputs are protected against overvoltage faults as large as ±45V. Programmable detection of RTD and cable open and short conditions is included.

Applications

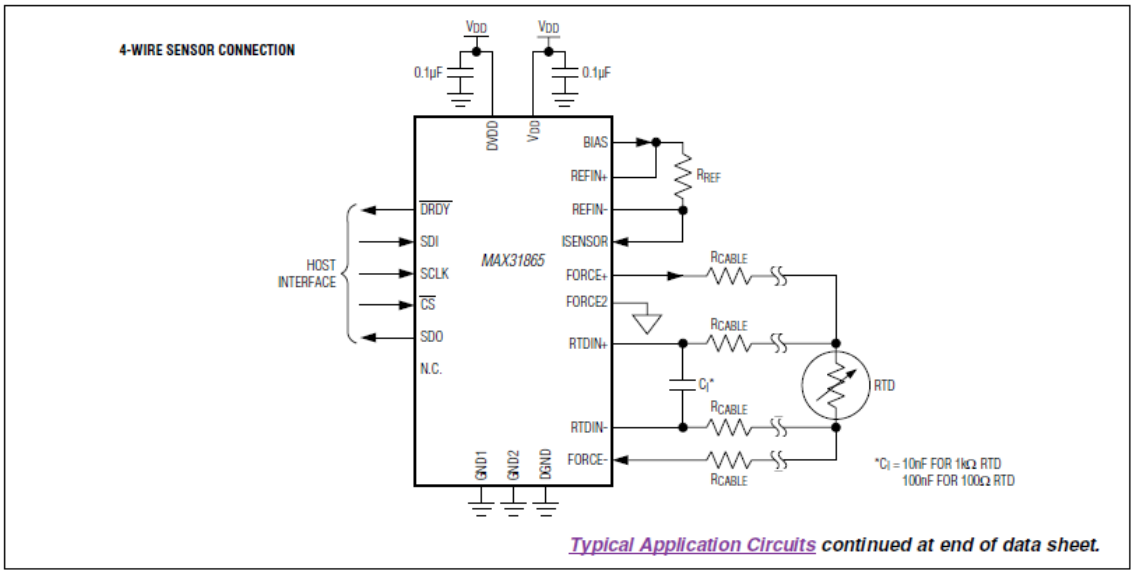
- Industrial Equipment
- Medical Equipment
- Instrumentation

Ordering Information appears at end of data sheet.

Benefits and Features

- Integration Lowers System Cost, Simplifies Design Efforts, and Reduces Design Cycle Time
  - Simple Conversion of Platinum RTD Resistance to Digital Value
  - Handles 100Ω to 1kΩ (at 0°C) Platinum RTDs (PT100 to PT1000)
  - Compatible with 2-, 3-, and 4-Wire Sensor Connections
  - SPI-Compatible Interface
  - 20-Pin TQFN and SSOP Packages
- High Accuracy Facilitates Meeting Error Budgets
  - 15-Bit ADC Resolution; Nominal Temperature Resolution 0.03125°C (Varies Due to RTD Nonlinearity)
  - Total Accuracy Over All Operating Conditions: 0.5°C (0.05% of Full Scale) max
  - Fully Differential V_{REF} Inputs
  - 21ms (max) Conversion Time
- Integrated Fault Detection Increases System Reliability
  - ±45V Input Protection
  - Fault Detection (Open RTD Element, RTD Shorted to Out-of-Range Voltage, or Short Across RTD Element)

Typical Application Circuits



Typical Application Circuits continued at end of data sheet.

maxim integrated™

19-6478; Rev 3; 7/15

Absolute Maximum Ratings

Voltage Range on V_{DD} Relative to GND1.....-0.3V to +4.0V
Voltage Range on BIAS, REFIN+, REFIN-, ISENSOR..... -0.3V to (V_{DD} + 0.3V)
Voltage Range on FORCE+, FORCE2, FORCE-, RTDIN+, RTDIN- Relative to GND1 ....-50V to +50V
Voltage Range on DVDD Relative to DGND.....-0.3V to +4.0V
Voltage Range on All Digital Pins Relative to DGND .....-0.3V to (V_{DVDD} + 0.3V)

Continuous Power Dissipation (T_A = +70°C)
TQFN (derate 34.5mW/°C above +70°C).....2758.6mW
SSOP (derate 11.9mW/°C above +70° C).....952.4mW
ESD Protection (all pins, Human Body Model).....±2kV
Operating Temperature Range.....-40°C to +125°C
Junction Temperature .....+150°C
Storage Temperature Range.....-65°C to +150°C
Soldering Temperature (reflow) .....+260°C
Lead Temperature (soldering,10s) .....+300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Package Thermal Characteristics (Note 1)

TQFN
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....29°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) .....2°C/W

SSOP
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....84°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) .....32°C/W

Note 1: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to www.maximintegrated.com/thermal-tutorial.

Recommended DC Operating Conditions

(T_A = -40°C to +125°C, unless otherwise noted.) (Notes 2 and 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{DD}	V_{DD}		3.0	3.3	3.6	V
DVDD	V_{DVDD}		3.0	3.3	3.6	V
Input Logic 0	V_{IL}	$\overline{CS}$, SDI, SCLK	-0.3		0.3 x V_{DVDD}	V
Input Logic 1	V_{IH}	$\overline{CS}$, SDI, SCLK	0.7 x V_{DVDD}		V_{DVDD} + 0.3	V
Analog Voltages (FORCE+,FORCE2, FORCE-, RTDIN+, RTDIN-)		Normal conversion results	0		V_{BIAS}	V
Reference Resistor	R_{REF}		350		10k	Ω
Cable Resistance	R_{CABLE}	Per lead	0		50	Ω

Electrical Characteristics

(3.0V ≤ V_{DD} ≤ 3.6V, T_A = -40°C to +125°C, unless otherwise noted. Typical values are T_A= +25°C, V_{DD} = V_{DVDD} = 3.3V.) (Notes 2 and 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
ADC Resolution		No missing codes		15		Bits
ADC Full-Scale Input Voltage (RTDIN+ - RTDIN-)				REFIN+ - REFIN-		V

Electrical Characteristics (continued)

(3.0V ≤ V_{DD} ≤ 3.6V, T_A = -40°C to +125°C, unless otherwise noted. Typical values are T_A= +25°C, V_{DD} = V_{DVDD} = 3.3V.) (Notes 2 and 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
ADC Common-Mode Input Range			0		V_{BIAS}	V
Input Leakage Current		RTDIN+, RTDIN-, 0°C to +70°C, on-state		2		nA
		RTDIN+, RTDIN-, -40°C to +85°C, on-state		5		
		RTDIN+, RTDIN-, -40°C to 100°C, on-state		14		
Bias Voltage	V_{BIAS}		1.95	2.00	2.06	V
Bias Voltage Output Current	I_{OUT}		0.2		5.75	mA
Bias Voltage Load Regulation		I_{OUT} ≤ 5.75mA		30		mV/mA
Bias Voltage Startup Time		(Note 4)			10	ms
ADC Full-Scale Error				±1		LSB
ADC Integral Nonlinearity		Differential Input, endpoint fit, 0.3 x V_{BIAS} ≤ V_{REF} ≤ V_{BIAS}		±1		LSB
ADC Offset Error			-3		+3	LSB
Noise (over Nyquist Bandwidth)		Input referred		150		µV RMS
Common-Mode Rejection				90		dB
50/60Hz Noise Rejection		Fundamental and harmonics		82		dB
Temperature Conversion Time (Note 5)	t_{CONV}	Continuous conversion (60Hz notch)		16.7	17.6	ms
		Single conversion (60Hz notch)		52	55	
		Single conversion (50Hz notch)		62.5	66	
		Continuous conversion (50Hz notch)		20	21	
Automatic Fault Detection Cycle Time		From $\overline{CS}$ high to cycle complete		550	600	µs
Power-Supply Rejection				1		LSB/V
Power-Supply Current (Note 6)	I_{DD} Shutdown	Bias off, ADC off		1.5	3	mA
	I_{DD}	Bias on, active conversion		2	3.5	mA
Power-On Reset Voltage Threshold			2	2.27		V
Power-On Reset Voltage Hysteresis				120		mV
Input Capacitance	C_{IN}	Logic inputs		6		pF
Input Leakage Current	I_L	Logic inputs	-1		+1	µA
Output High Voltage	V_{OH}	I_{OUT} = -1.6mA	V_{DVDD} - 0.4			V
Output Low Voltage	V_{OL}	I_{OUT} = 1.6mA			0.4	V

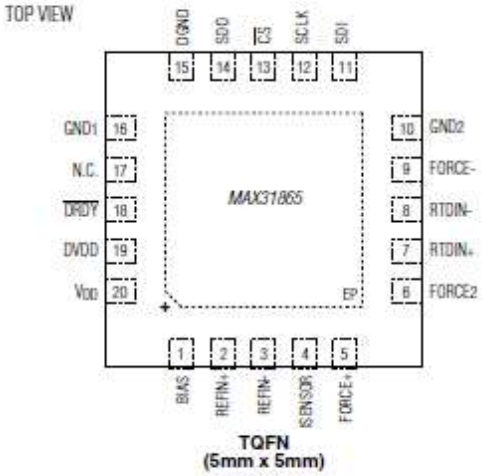
AC Electrical Characteristics: SPI Interface

(3.0V ≤ V_{DD} ≤ 3.6V, T_A = -40°C to +125°C, unless otherwise noted. Typical values are T_A= +25°C, V_{DD} = V_{DVDD} = 3.3V.) (Notes 3 and 7) (Figure 1 and Figure 2)

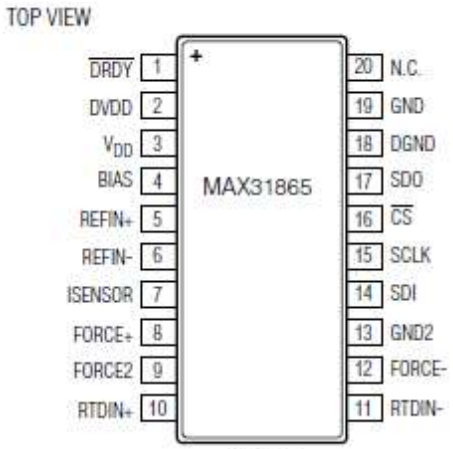
PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Data to SCLK Setup	t_{DC}	(Notes 8, 9)	35			ns
SCLK to Data Hold	t_{CDH}	(Notes 8, 9)	35			ns
SCLK to Data Valid	t_{CDD}	(Notes 8, 9, 10)			80	ns
SCLK Low Time	t_{CL}	(Note 9)	100			ns
SCLK High Time	t_{CH}	(Note 9)	100			ns
SCLK Frequency	t_{CLK}	(Note 9)	DC		5.0	MHz
SCLK Rise and Fall	t_R, t_F	(Note 9)			200	ns
$\overline{CS}$ to SCLK Setup	t_{CC}	(Note 9)	400			ns
SCLK to $\overline{CS}$ Hold	t_{CCH}	(Note 9)	100			ns
$\overline{CS}$ Inactive Time	t_{CWH}	(Note 9)	400			ns
$\overline{CS}$ to Output High-Z	t_{CDZ}	(Notes 8, 9)			40	ns
Address 01h or 02h Decoded to $\overline{DRDY}$ High	t_{DRDYH}	After RTD register read access (Note 9)		50		ns

Note 2: All voltages are referenced to ground when common. Currents entering the IC are specified positive.
Note 3: Limits are 100% production tested at T_A= +25°C and/or T_A= +85°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization. Typical values are not guaranteed.
Note 4: For 15-bit settling, a wait of at least 10.5 time constants of the input RC network is required. Max startup time is calculated with a 10kΩ reference resistor and a 0.1µF capacitor across the RTD inputs.
Note 5: The first conversion after enabling continuous conversion mode takes a time equal to the single conversion time for the respective notch frequency.
Note 6: Specified with no load on the bias pin as the sum of analog and digital currents. No active communication. If the RTD input voltage is greater than the input reference voltage, then an additional 400µA I_{DD} can be expected.
Note 7: All timing specifications are guaranteed by design.
Note 8: Measured at V_{IH} = 0.7V x V_{DVDD} or V_{IL} = 0.3 x V_{DVDD} and 10ms maximum rise and fall times.
Note 9: Measured with 50pF load.
Note 10: Measured at V_{OH} = 0.7 x V_{DVDD} or V_{OL} = 0.3 x V_{DVDD}. Measured from the 50% point of SCLK to the V_{OH} minimum of SDO.

Pin Configurations



TQFN
(5mm x 5mm)



SSOP

Pin Description

PIN		NAME	FUNCTION
TQFN	SSOP		
1	4	BIAS	Bias Voltage Output (V_{BIAS})
2	5	REFIN+	Positive Reference Voltage Input. Connect to BIAS. Connect the reference resistor between REFIN+ and REFIN-.
3	6	REFIN-	Negative Reference Voltage Input. Connect the reference resistor between REFIN+ and REFIN-.
4	7	ISENSOR	Low Side of R_{REF}. Connect to REFIN-.
5	8	FORCE+	High-Side RTD Drive. Connect to FORCE2 when using the 3-wire connection configuration. Protected to ±45V.
6	9	FORCE2	Positive Input Used in 3-Wire Only. When in the 3-wire connection configuration, connect to FORCE+. When in the 2-wire or 4-wire connection configuration, connect to ground. Protected to ±45V.
7	10	RTDIN+	Positive RTD Input. Protected to ±45V.
8	11	RTDIN-	Negative RTD Input. Protected to ±45V.
9	12	FORCE-	Low-Side RTD Return. Protected to ±45V.
10	13	GND2	Analog Ground. Connect to GND1.
11	14	SDI	Serial-Data Input
12	15	SCLK	Serial-Data Clock Input
13	16	$\overline{CS}$	Active-Low Chip Select. Set $\overline{CS}$ low to enable the serial interface.
14	17	SDO	Serial-Data Output
15	18	DGND	Digital Ground
16	19	GND1	Analog Ground. Connect to GND2.
17	20	N.C.	Do Not Connect
18	1	$\overline{DRDY}$	Active-Low, Push-Pull, Data-Ready Output. $\overline{DRDY}$ goes low when a new conversion result is available in the data register. When a read operation of an RTD resistance data register occurs, $\overline{DRDY}$ returns high.
19	2	DVDD	Digital Supply Voltage Input. Connect to a 3.3V power supply. Bypass to DGND with a 0.1µF bypass capacitor.
20	3	V_{DD}	Analog Supply Voltage Input. Connect to a 3.3V power supply. Bypass to GND1 with a 0.1µF bypass capacitor.
—	—	EP	Exposed Pad (Bottom Side of Package). Connect to GND1. Applies to TQFN package only.

2.5 Datasheet DS3231

DS3231M **±5ppm, I²C Real-Time Clock**

General Description

The DS3231M is a low-cost, extremely accurate, I²C real-time clock (RTC). The device incorporates a battery input and maintains accurate timekeeping when main power to the device is interrupted. The integration of the microelectromechanical systems (MEMS) resonator enhances the long-term accuracy of the device and reduces the piece-part count in a manufacturing line. The DS3231M is available in the same footprint as the popular DS3231 RTC.

The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a 1Hz output are provided. Address and data are transferred serially through an I²C bidirectional bus. A precision temperature-compensated voltage reference and comparator circuit monitors the status of V_{CC} to detect power failures, to provide a reset output, and to automatically switch to the backup supply when necessary. Additionally, the RST pin is monitored as a pushbutton input for generating a microprocessor reset. See the Block Diagram for more details.

Applications

Power Meters

Industrial Applications

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
DS3231MZ+	-45°C to +85°C	8 SO
DS3231MZ/V+	-45°C to +85°C	8 SO
DS3231M+	-45°C to +85°C	16 SO

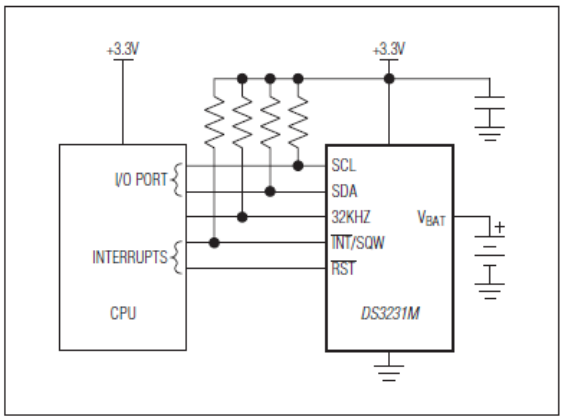
+Denotes a lead(Pb)-free/RoHS-compliant package.
/V denotes an automotive qualified part.

Underwriters Laboratories is a registered certification mark of Underwriters Laboratories Inc.

Benefits and Features

- Highly Accurate RTC With Integrated MEMS Resonator Completely Manages All Timekeeping Functions
  - Complete Clock Calendar Functionality Including Seconds, Minutes, Hours, Day, Date, Month, and Year, with Leap-Year Compensation Up to Year 2100
  - Timekeeping Accuracy ±5ppm (±0.432 Second/Day) from -45°C to +85°C
  - Footprint and Functionally Compatible to DS3231
  - Two Time-of-Day Alarms
  - 1Hz and 32.768kHz Outputs
  - Reset Output and Pushbutton Input with Debounce
  - Digital Temp Sensor with ±3°C Accuracy
  - +2.3V to +5.5V Supply Voltage
- Simple Serial Interface Connects to Most Microcontrollers
  - Fast (400kHz) I²C Interface
- Battery-Backup Input for Continuous Timekeeping
  - Low Power Operation Extends Battery-Backup Run Time
- Operating Temperature Range: -40°C to +85°C
- 8-Pin or 16-Pin SO Packages
- Underwriters Laboratories® (UL) Recognized

Typical Operating Circuit



19-5312; Rev 7; 3/15

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to GND .....-0.3V to +6.0V
Operating Temperature Range .....-45°C to +85°C
Storage Temperature Range.....-55°C to +125°C
Junction Temperature .....+150°C
Lead Temperature (soldering, 10s) .....+300°C
Soldering Temperature (reflow) .....+260°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Package Thermal Characteristics (Note 1)

8 SO
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....120°C/W

16 SO
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....90°C/W

Note 1: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to www.maximintegrated.com/thermal-tutorial.

Recommended Operating Conditions

(T_A = -45°C to +85°C, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		2.3	3.3	5.5	V
	V_{BAT}		2.3	3.0	5.5	
Logic 1	V_{IH}		0.7 x V_{CC}		V_{CC} + 0.3	V
Logic 0	V_{IL}		-0.3		0.3 x V_{CC}	V

Electrical Characteristics—Frequency And Timekeeping

(V_{CC} or V_{BAT} = +3.3V, T_A = -45°C to +85°C, unless otherwise noted. Typical values are at V_{CC} = +3.3V, V_{BAT} = +3.0V, and T_A = +25°C, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
1Hz Frequency Tolerance	$\Delta f/f_{OUT}$	Measured over ≥ 10s interval			±5	ppm
1Hz Frequency Stability vs. V_{CC} Voltage	$\Delta f/V$			±1		ppm/V
Timekeeping Accuracy	tK_A				±0.432	Seconds/ Day
32kHz Frequency Tolerance	$\Delta f/f_{OUT}$				±2.5	%

DC Electrical Characteristics—General

(V_{CC} = +2.3V to +5.5V, T_A = -45°C to +85°C, unless otherwise noted. Typical values are at V_{CC} = +3.3V, V_{BAT} = +3.0V, and T_A = +25°C, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Active Supply Current (I²C Active: Includes Temperature Conversion Current)	I_{CCA}	V_{CC} = +3.63V			200	µA
		V_{CC} = V_{CCMAX}			300	
Standby Supply Current (I²C Inactive: Includes Temperature Conversion Current)	I_{CCS}	V_{CC} = +3.63V			130	µA
		V_{CC} = V_{CCMAX}			200	
Temperature Conversion Current (I²C Inactive)	$I_{CCSCONV}$	V_{CC} = +3.63V			575	µA
		V_{CC} = V_{CCMAX}			650	

DC Electrical Characteristics—General (continued)

(V_{CC} = +2.3V to +5.5V, T_A = -45°C to +85°C, unless otherwise noted. Typical values are at V_{CC} = +3.3V, V_{BAT} = +3.0V, and T_A = +25°C, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Power-Fail Voltage	V_{PF}		2.45	2.575	2.70	V
Logic 0 Output (32KHZ, $\overline{INT}$/SQW, SDA)	V_{OL}	I_{OL} = 3mA			0.4	V
Logic 0 Output ($\overline{RST}$)	V_{OL}	I_{OL} = 1mA			0.4	V
Output Leakage (32KHZ, $\overline{INT}$/SQW, SDA)	I_{LO}		-0.1		+0.1	µA
Input Leakage (SCL)	I_{LI}		-0.1		+0.1	µA
$\overline{RST}$ I/O Leakage	I_{OL}		-200		+10	µA
V_{BAT} Leakage	I_{BATLKG}			25	100	nA
Temperature Accuracy	$TEMP_{ACC}$	V_{CC} or V_{BAT} = +3.3V		±3		°C
Temperature Conversion Time	t_{CONV}			10		ms
Pushbutton Debounce	PB_{DB}			250		ms
Reset Active Time	t_{RST}			250		ms
Oscillator Stop Flag (OSF) Delay	t_{OSF}	(Note 3)		125	200	ms

DC Electrical Characteristics—V_{BAT} Current Consumption

(V_{CC} = 0V, V_{BAT} = +2.3V to +5.5V, T_A = -45°C to +85°C, unless otherwise noted. Typical values are at V_{CC} = 0V, V_{BAT} = +3.0V, and T_A = +25°C, unless otherwise noted.)

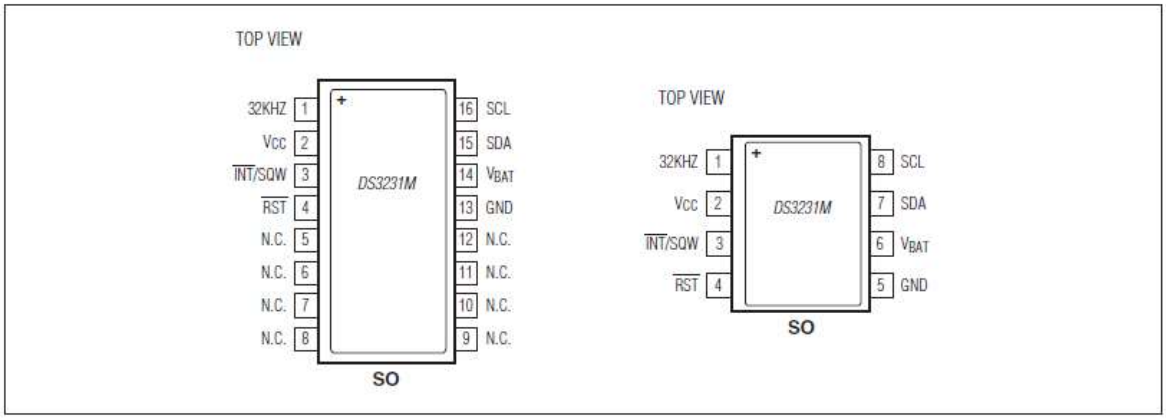
PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Active Battery Current (I²C Active) (Note 4)	I_{BATA}	V_{BAT} = +3.63V			70	µA
		V_{BAT} = V_{BATMAX}			150	
Timekeeping Battery Current (I²C Inactive) (Note 4)	I_{BATT}	V_{BAT} = +3.63V, EN32KHZ = 0		2	3.0	µA
		V_{BAT} = V_{BATMAX}, EN32KHZ = 0		2	3.5	
Temperature Conversion Current (I²C Inactive)	I_{BATTC}	V_{BAT} = +3.63V			575	µA
		V_{BAT} = V_{BATMAX}			650	
Data Retention Current (Oscillator Stopped and I²C Inactive)	I_{BATDR}	T_A = +25°C			100	nA

AC Electrical Characteristics—Power Switch

(T_A = -45°C to +85°C, unless otherwise noted.) (Figure 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{CC} Fall Time, V_{PFMAX} to V_{PFMIN}	t_{VCCF}		300			µs
V_{CC} Rise Time, V_{PFMIN} to V_{PFMAX}	t_{VCCR}		0			µs
Recovery at Power-Up	t_{REC}	(Note 5)		250	300	ms

Pin Configuration



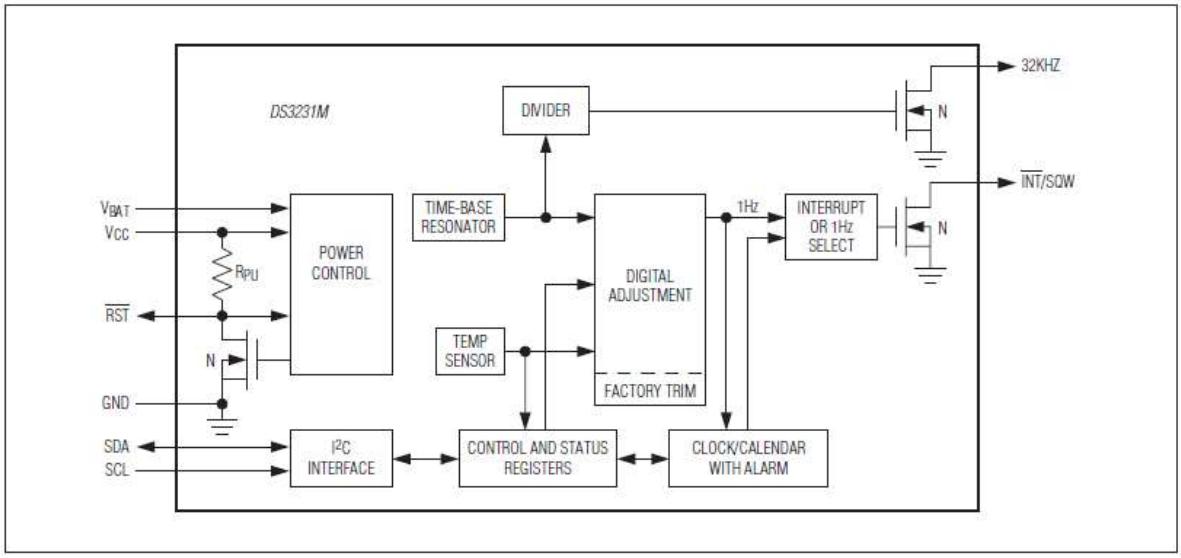
Pin Description

PIN		NAME	FUNCTION
8 SO	16 SO		
1	1	32KHZ	32.768kHz Output (50% Duty Cycle). This open-drain pin requires an external pullup resistor. When enabled with the EN32KHZ bit in the Status register (0Fh), this output operates on either power supply. This pin can be left open circuit if not used.
2	2	V_{CC}	DC Power Pin for Primary Power Supply. This pin should be decoupled using a 0.1µF to 1.0µF capacitor. Connect to ground if not used.
3	3	$\overline{INT}$/SQW	Active-Low Interrupt or 1Hz Square-Wave Output. This open-drain pin requires an external pullup resistor connected to a supply at 5.5V or less. It can be left open if not used. This multifunction pin is determined by the state of the INTCN bit in the Control register (0Eh). When INTCN is set to logic 0, this pin outputs a 1Hz square wave. When INTCN is set to logic 1, a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled.
4	4	$\overline{RST}$	Active-Low Reset. This pin is an open-drain input/output. It indicates the status of V_{CC} relative to the V_{PF} specification. As V_{CC} falls below V_{PF}, the $\overline{RST}$ pin is driven low. When V_{CC} exceeds V_{PF}, for t_{RST}, the $\overline{RST}$ pin is pulled high by the internal pullup resistor. The active-low, open-drain output is combined with a debounced pushbutton input function. This pin can be activated by a pushbutton reset request. It has an internal 50kΩ (R_{PU}) nominal value pullup resistor to V_{CC}. No external pullup resistors should be connected. If the oscillator is disabled, t_{REC} is bypassed and $\overline{RST}$ immediately goes high.
—	5–12	N.C.	No Connection. These pins must be connected to ground.
5	13	GND	Ground
6	14	V_{BAT}	Backup Power-Supply Input. When using the device with the V_{BAT} input as the primary power source, this pin should be decoupled using a 0.1µF to 1.0µF low-leakage capacitor. When using the device with the V_{BAT} input as the backup power source, the capacitor is not required. If V_{BAT} is not used, connect to ground. The device is UL recognized to ensure against reverse charging when used with a primary lithium battery. Go to www.maximintegrated.com/qa/info/ul for more information.

Pin Description (continued)

PIN		NAME	FUNCTION
8 SO	16 SO		
7	15	SDA	Serial-Data Input/Output. This pin is the data input/output for the I²C serial interface. This open-drain pin requires an external pullup resistor. The pullup voltage can be up to 5.5V, regardless of the voltage on VCC.
8	16	SCL	Serial-Clock Input. This pin is the clock input for the I²C serial interface and is used to synchronize data movement on the serial interface. The pullup voltage can be up to 5.5V, regardless of the voltage on VCC.

Block Diagram



Detailed Description

The DS3231M is a serial real-time clock (RTC) driven by an internal, temperature-compensated, microelectromechanical systems (MEMS) resonator. The oscillator provides a stable and accurate reference clock and maintains the RTC to within ±0.432 seconds-per-day accuracy from -45°C to +85°C. The RTC is a low-power clock/calendar with two programmable time-of-day alarms. INT/SQW provides either an interrupt signal due to alarm conditions or a 1Hz square wave. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. The internal registers are accessible though an I²C bus interface. A temperature-compensated voltage reference and comparator circuit monitors the level of VCC to detect power failures and to automatically switch to the backup supply when necessary. The RST pin provides an external pushbutton function and acts as an indicator of a power-fail event.

Operation

The *Block Diagram* shows the device's main elements. Each of the major blocks is described separately in the following sections.

2.6 Datasheet LM321

Product Folder | Sample & Buy | Technical Documents | Tools & Software | Support & Community

TEXAS INSTRUMENTS

LM321

SNOS935C – FEBRUARY 2001 – REVISED DECEMBER 2014

LM321 Low Power Single Operational Amplifier

1 Features

- (V_{CC} = 5 V, T_A = 25°C. Typical values unless specified.)
- Gain-Bandwidth Product 1 MHz
- Low Supply Current 430 µA
- Low Input Bias Current 45 nA
- Wide Supply Voltage Range 3 V to 32 V
- Stable With High Capacitive Loads
- Single Version of LM324

2 Applications

- Chargers
- Power Supplies
- Industrial: Controls, Instruments
- Desktops
- Communications Infrastructure

3 Description

The LM321 brings performance and economy to low power systems. With a high unity gain frequency and a specified 0.4-V/µs slew rate, the quiescent current is only 430-µA/amplifier (5 V). The input common mode range includes ground and therefore the device is able to operate in single supply applications as well as in dual supply applications. It is also capable of comfortably driving large capacitive loads.

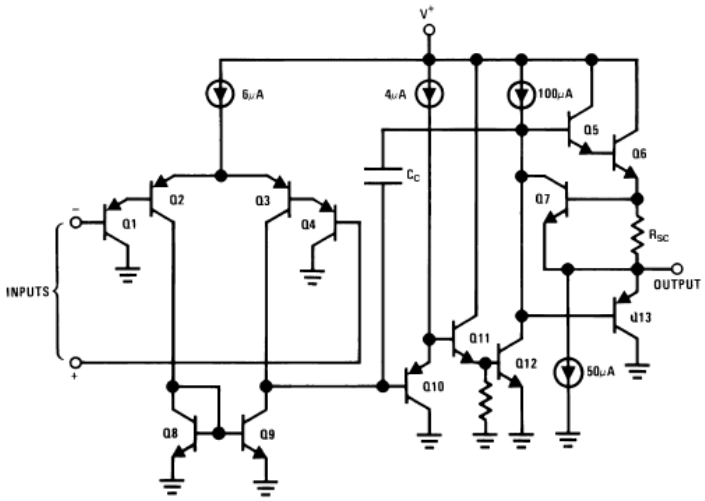
The LM321 is available in the SOT-23 package. Overall the LM321 is a low power, wide supply range performance operational amplifier that can be designed into a wide range of applications at an economical price without sacrificing valuable board space.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM321	SOT (5)	2.90 mm × 1.60 mm

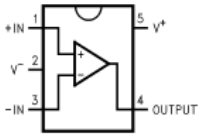
(1) For all available packages, see the orderable addendum at the end of the datasheet.

Simplified Schematic



5 Pin Configuration and Functions

DBV Package
5-Pin SOT-23
Top View



Pin Functions

PIN		I/O	DESCRIPTION
NAME	NO.		
+IN	1	I	Noninverting input
V–	2	—	Negative (lowest) power supply
–IN	3	I	Inverting input
OUTPUT	4	O	Output
V+	5	—	Positive (highest) power supply

6 Specifications

6.1 Absolute Maximum Ratings (1)

	MIN	MAX	UNIT
Differential Input Voltage	±Supply Voltage		
Input Current (V_{IN} < –0.3 V) (2)		50	mA
Supply Voltage (V^+ - V^-)		32	V
Input Voltage	–0.3	32	V
Output Short Circuit to GND, V^+ ≤ 15 V and T_A = 25°C (3)	Continuous		
Junction Temperature (4)		150	°C
Mounting Temperature: Lead temperature (Soldering, 10 sec)		260	°C
Mounting Temperature: Infrared (10 sec)		215	°C
Storage temperature, T_{stg}	–65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) This input current will only exist when the voltage at any of the input leads is driven negative. It is due to the collector base junction of the input PNP transistors becoming forward biased and thereby acting as input diode clamps. In addition to this diode action, there is also lateral NPN parasitic transistor action on the IC chip. This transistor action can cause the output voltages of the operational amplifer to go to the V+ voltage level (or to ground for a large overdrive) for the time duration that an input is driven negative. This is not destructive and normal output states will re-establish when the input voltage, which was negative, again returns to a value greater than –0.36V (at 25°C).
(3) Short circuits from the output V+ can cause excessive heating and eventual destruction. When considering short circuits to ground the maximum output current is approximately 40mA independent of the magnitude of V+. At values of supply voltage in excess of +15V, continuous short circuits can exceed the power dissipation ratings and cause eventual destruction.
(4) The maximum power dissipation is a function of TJ(MAX), θJA , and TA. The maximum allowable power dissipation at any ambient temperature is PD = (TJ(MAX) - TA)/ θJA. All numbers apply for packages soldered directly onto a PC board.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001 (1)	±300	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

	MIN	MAX	UNIT
Temperature Range	−40	85	°C
Supply Voltage	3	30	V

6.4 Thermal Information

THERMAL METRIC(1)		LM321 DBV 5 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance	265	°C/W

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.

6.5 Electrical Characteristics

Unless otherwise specified, all limits specified for at T_A = 25°C; V^+ = 5 V, V^- = 0 V, V_O = 1.4 V

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OS}	Input Offset Voltage				(1)2	7	mV
			(1), –40°C ≤ T_J ≤ 85°C			9	
I_{OS}	Input Offset Current				5	50	nA
			–40°C ≤ T_J ≤ 85°C			150	
I_B	Input Bias Current (2)				45	250	nA
			–40°C ≤ T_J ≤ 85°C			500	
V_{CM}	Input Common-Mode Voltage Range		V^+ = 30 V (3), for CMRR > = 50dB	0		V^+ - 1.5	V
			V^+ = 30 V (3), for CMRR > = 50dB, –40°C ≤ T_J ≤ 85°C			V^+ - 2	
A_V	Large Signal Voltage Gain		(V^+ = 15 V, R_L = 2kΩ, V_O = 1.4 V to 11.4 V)	25	100		V/mV
			(V^+ = 15 V, R_L = 2kΩ, V_O = 1.4 V to 11.4 V), –40°C ≤ T_J ≤ 85°C	15			
PSRR	Power Supply Rejection Ratio		R_S ≤ 10kΩ, V^+ ≤ 5 V to 30 V	65	100		dB
CMRR	Common Mode Rejection Ratio		R_S ≤ 10kΩ	65	85		dB
V_O	Output Swing	V_{OH}	V^+ = 30 V, R_L = 2kΩ, –40°C ≤ T_J ≤ 85°C	26			V
			V^+ = 30 V, R_L = 10kΩ, –40°C ≤ T_J ≤ 85°C	27	28		
		V_{OL}	V^+ = 5 V, R_L = 10kΩ, –40°C ≤ T_J ≤ 85°C		5	20	mV
I_S	Supply Current, No Load		V^+ = 5 V		0.430	1.15	mA
			V^+ = 5 V, –40°C ≤ T_J ≤ 85°C		0.7	1.2	
			V^+ = 30 V		0.660	2.85	
			V^+ = 30 V, –40°C ≤ T_J ≤ 85°C		1.5	3	
I_{SOURCE}	Output Current Sourcing		V_{ID} = +1 V, V^+ = 15 V, V_O = 2 V	20	40		mA
			V_{ID} = +1 V, V^+ = 15 V, V_O = 2 V, –40°C ≤ T_J ≤ 85°C	10	20		
I_{SINK}	Output Current Sinking		V_{ID} = −1 V, V^+ = 15 V, V_O = 2 V	10	20		mA
			V_{ID} = −1 V, V^+ = 15 V, V_O = 2 V, –40°C ≤ T_J ≤ 85°C	5	8		
			V_{ID} = −1 V, V^+ = 15 V, V_O = 0.2 V	12	100		µA

(1) VO ≅ 1.4 V, RS = 0Ω with V+ from 5 V to 30 V; and over the full input common-mode range (0 V to V+ - 1.5 V) at 25°C.
(2) The direction of the input current is out of the IC due to the PNP input stage. This current is essentially constant, independent of the state of the output so no loading change exists on the input lines.
(3) The input common-mode voltage of either input signal voltage should not be allowed to go negative by more than 0.3 V (at 25°C). The upper end of the common-mode voltage range is V+ - 1.5 V at 25°C, but either or both inputs can go to +32 V without damage, independent of the magnitude of V+.

2.7 Datasheet SN65176b

Product Folder | Sample & Buy | Technical Documents | Tools & Software | Support & Community

TEXAS INSTRUMENTS

SN65176B, SN75176B

SLLS101F – JULY 1985 – REVISED JANUARY 2015

SNx5176B Differential Bus Transceivers

1 Features

- Bidirectional Transceivers
- Meet or Exceed the Requirements of ANSI Standards TIA/EIA-422-B and TIA/EIA-485-A and ITU Recommendations V.11 and X.27
- Designed for Multipoint Transmission on Long Bus Lines in Noisy Environments
- 3-State Driver and Receiver Outputs
- Individual Driver and Receiver Enables
- Wide Positive and Negative Input/Output Bus Voltage Ranges
- ± 60-mA Max Driver Output Capability
- Thermal Shutdown Protection
- Driver Positive and Negative Current Limiting
- 12-kΩ Min Receiver Input Impedance
- ± 200-mV Receiver Input Sensitivity
- 50-mV Typ Receiver Input Hysteresis
- Operate From Single 5-V Supply

2 Applications

- Chemical/Gas Sensors
- Digital Signage
- HMI (Human Machine Interfaces)
- Motor Controls: AC Induction, Brushed and Brush-less DC, Low- and High-Voltage, Stepper Motors, and Permanent Magnets
- TETRA Base Stations
- Telecom Towers: Remote Electrical Tilt Units (RET) and Tower Mounted Amplifiers (TMA)
- Weigh Scales
- Wireless Repeaters

3 Description

The SN65176B and SN75176B differential bus transceivers are designed for bidirectional data communication on multipoint bus transmission lines. They are designed for balanced transmission lines and meet ANSI Standards TIA/EIA-422-B and TIA/EIA-485-A and ITU Recommendations V.11 and X.27.

The SN65176B and SN75176B devices combine a 3-state differential line driver and a differential input line receiver, both of which operate from a single 5-V power supply. The driver and receiver have active-high and active-low enables, respectively, that can be connected together externally to function as a direction control. The driver differential outputs and the receiver differential inputs are connected internally to form differential input/output (I/O) bus ports that are designed to offer minimum loading to the bus when the driver is disabled or V_{CC} = 0. These ports feature wide positive and negative common-mode voltage ranges, making the device suitable for party-line applications.

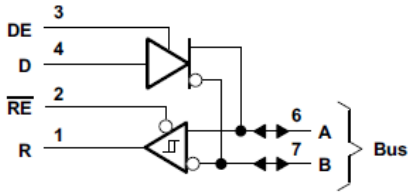
The driver is designed for up to 60 mA of sink or source current. The driver features positive and negative current limiting and thermal shutdown for protection from line-fault conditions. Thermal shutdown is designed to occur at a junction temperature of approximately 150°C. The receiver features a minimum input impedance of 12 kΩ, an input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

Device Information(1)

PART NUMBER	PACKAGE (PIN)	BODY SIZE (NOM)
SNx5176	SOIC (8)	4.90 mm × 3.91 mm
	PDIP (8)	9.81 mm × 6.35 mm
	SOP (8)	6.20 mm × 5.30 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

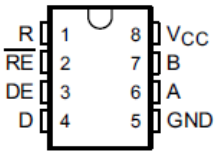
4 Simplified Schematic



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

6 Pin Configuration and Functions

Top View



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
R	1	O	Logic Data Output from RS-485 Receiver
$\overline{RE}$	2	I	Receive Enable (active low)
DE	3	I	Driver Enable (active high)
D	4	I	Logic Data Input to RS-485 Driver
GND	5	—	Device Ground Pin
A	6	I/O	RS-422 or RS-485 Data Line
B	7	I/O	RS-422 or RS-485 Data Line
V_{CC}	8	—	Power Input. Connect to 5-V Power Source.

7 Specifications

7.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

		MIN	MAX	UNIT
V_{CC}	Supply voltage(2)		7	V
	Voltage range at any bus terminal	–10	15	V
V_I	Enable input voltage		5.5	V
T_J	Operating virtual junction temperature		150	°C
T_{stg}	Storage temperature range	–65	150	°C
	Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds		260	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions* is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values, except differential input/output bus voltage, are with respect to network ground terminal.

7.2 Recommended Operating Conditions

			MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage		4.75	5	5.25	V
V_I or V_{IC}	Voltage at any bus terminal (separately or common mode)		-7		12	V
V_{IH}	High-level input voltage	D, DE, and $\overline{RE}$	2			V
V_{IL}	Low-level input voltage	D, DE, and $\overline{RE}$			0.8	V
V_{ID}	Differential input voltage(1)				±12	V
I_{OH}	High-level output current	Driver			–60	mA
		Receiver			–400	µA
I_{OL}	Low-level output current	Driver			60	mA
		Receiver			8	
T_A	Operating free-air temperature	SN65176B	–40		105	°C
		SN75176B	0		70	

(1) Differential input/output bus voltage is measured at the non-inverting terminal A, with respect to the inverting terminal B.

7.3 Thermal Information

THERMAL METRIC(1)		SNx5176			UNIT
		BD	BP	BPS	
		8 PINS			
$R_{\theta JA}$	Junction-to-ambient thermal resistance	97	85	95	°C/W

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report (SPRA953).

7.4 Electrical Characteristics – Driver

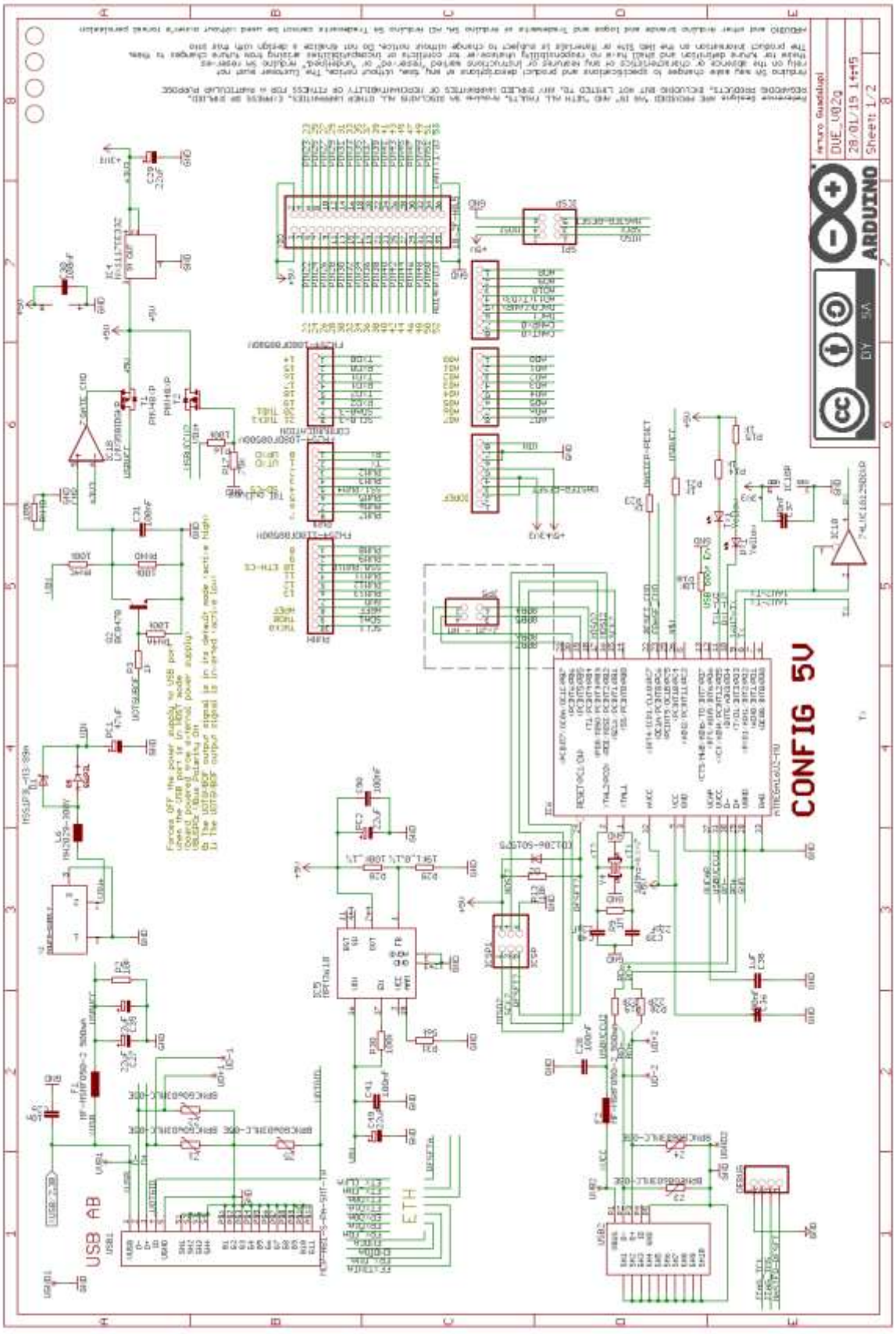
over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

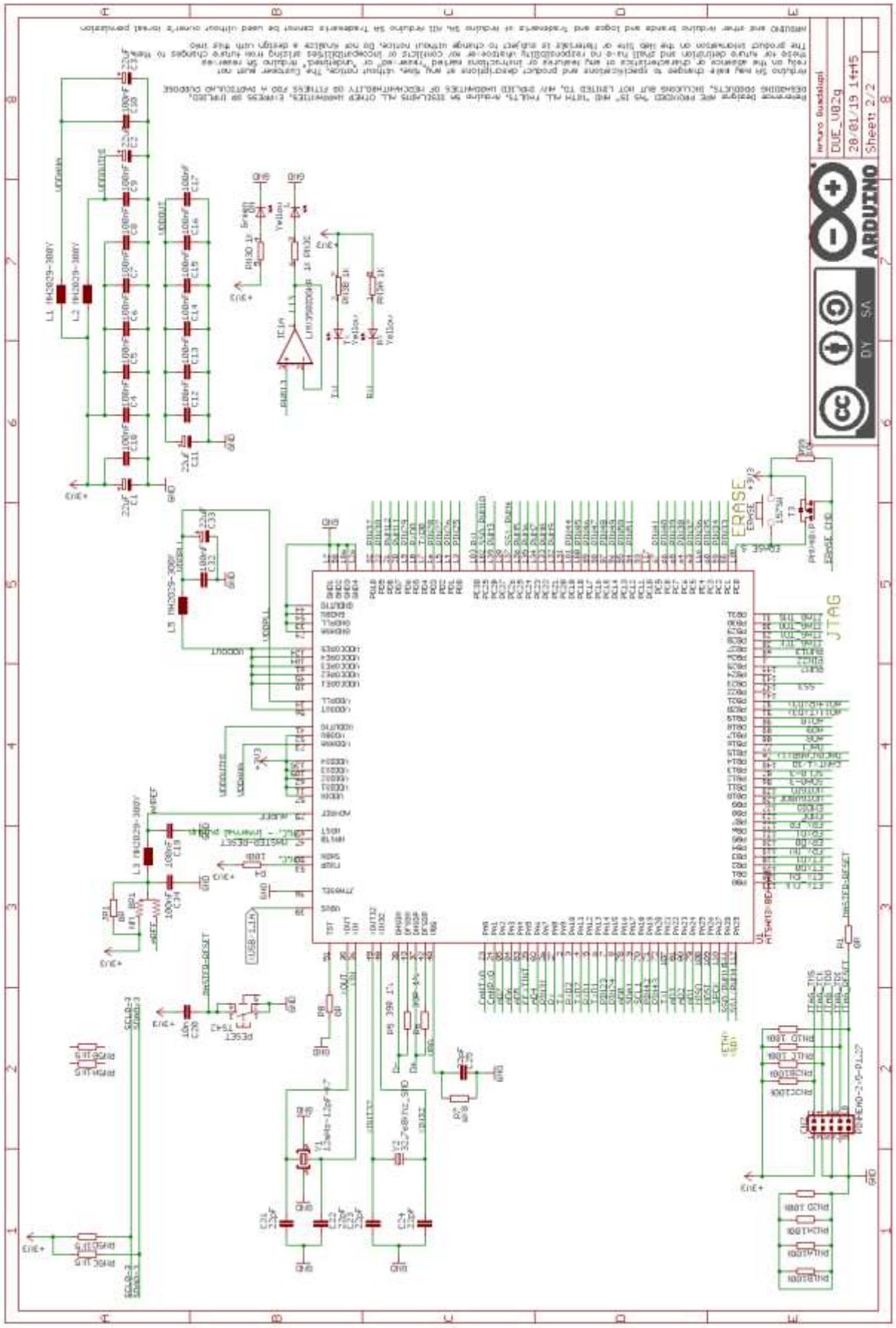
PARAMETER		TEST CONDITIONS(1)		MIN	TYP(2)	MAX	UNIT
V_{IK}	Input clamp voltage	I_I = –18 mA				–1.5	V
V_O	Output voltage	I_O = 0		0		6	V
$\|V_{OD1}\|$	Differential output voltage	I_O = 0		1.5	3.6	6	V
$\|V_{OD2}\|$	Differential output voltage	R_L = 100 Ω, see Figure 10		½ V_{OD1} or 2 (3)			V
		R_L = 54 Ω, see Figure 10		1.5	2.5	5	
V_{OD3}	Differential output voltage	See (4)		1.5		5	V
$\Delta\|V_{OD}\|$	Change in magnitude of differential output voltage(5)	R_L = 54 Ω or 100 Ω, see Figure 10				±0.2	V
V_{OC}	Common-mode output voltage	R_L = 54 Ω or 100 Ω, see Figure 10		-1		+3	V
$\Delta\|V_{OC}\|$	Change in magnitude of common-mode output voltage(5)	R_L = 54 Ω or 100 Ω, see Figure 10				±0.2	V
I_O	Output current	Output disabled(6)	V_O = 12 V			1	mA
			V_O = –7 V			–0.8	
I_{IH}	High-level input current	V_I = 2.4 V				20	µA
I_{IL}	Low-level input current	V_I = 0.4 V				–400	µA
I_{OS}	Short-circuit output current	V_O = –7 V				–250	mA
		V_O = 0				–150	
		V_O = V_{CC}				250	
		V_O = 12 V				250	
I_{CC}	Supply current (total package)	No load	Outputs enabled		42	70	mA
			Outputs disabled		26	35	

(1) The power-off measurement in ANSI Standard TIA/EIA-422-B applies to disabled outputs only and is not applied to combined inputs and outputs.
(2) All typical values are at V_{CC} = 5 V and T_A = 25°C.
(3) The minimum V_{OD2} with a 100-Ω load is either ½ V_{OD1} or 2 V, whichever is greater.
(4) See ANSI Standard TIA/EIA-485-A, Figure 3.5, Test Termination Measurement 2.
(5) $\Delta|V_{OD}|$ and $\Delta|V_{OC}|$ are the changes in magnitude of V_{OD} and V_{OC}, respectively, that occur when the input is changed from a high level to a low level.
(6) This applies for both power on and off; refer to ANSI Standard TIA/EIA-485-A for exact conditions. The TIA/EIA-422-B limit does not apply for a combined driver and receiver terminal.

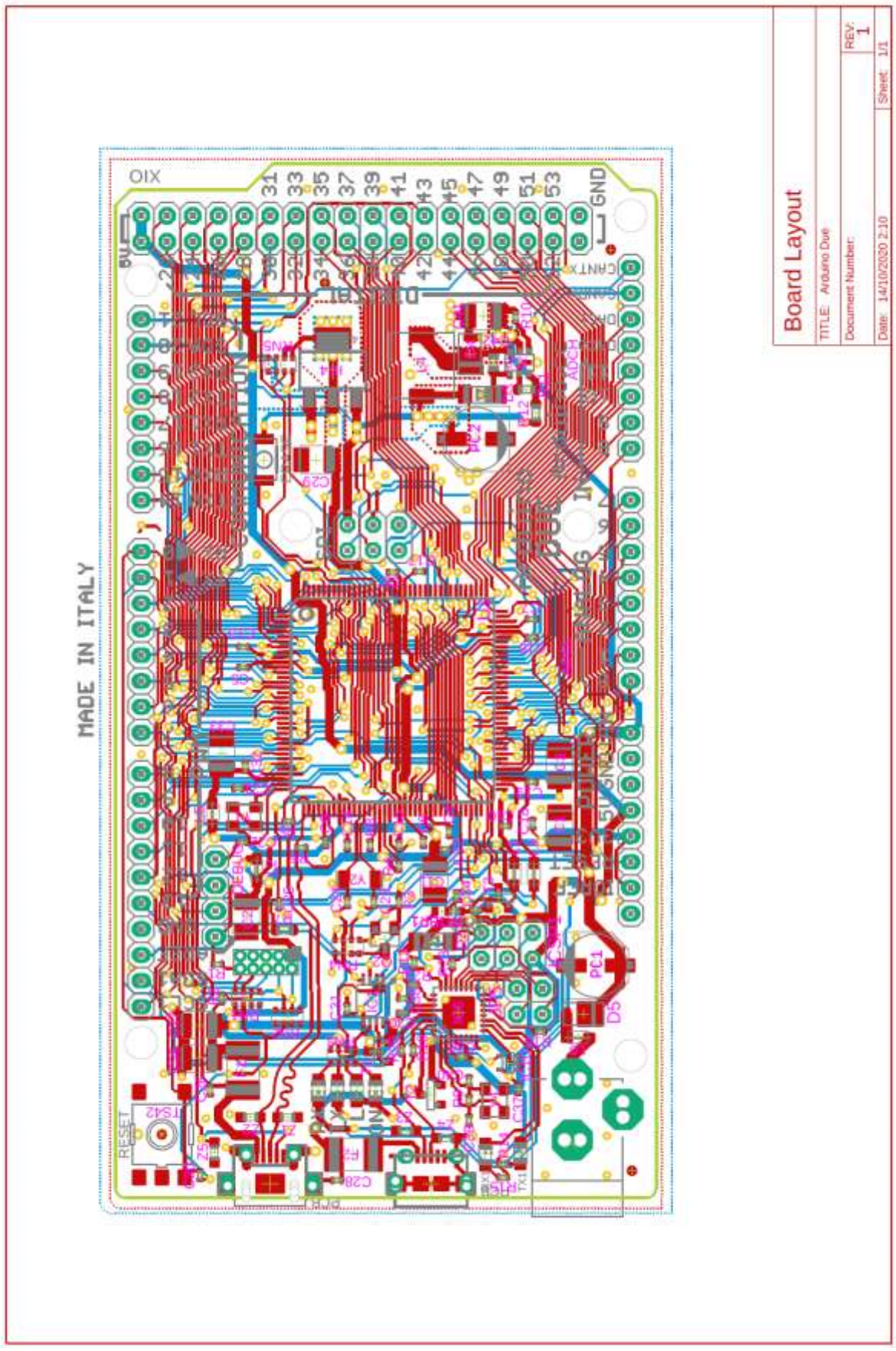
3 PLANOS

3.1 Esquema eléctrico Arduino Due

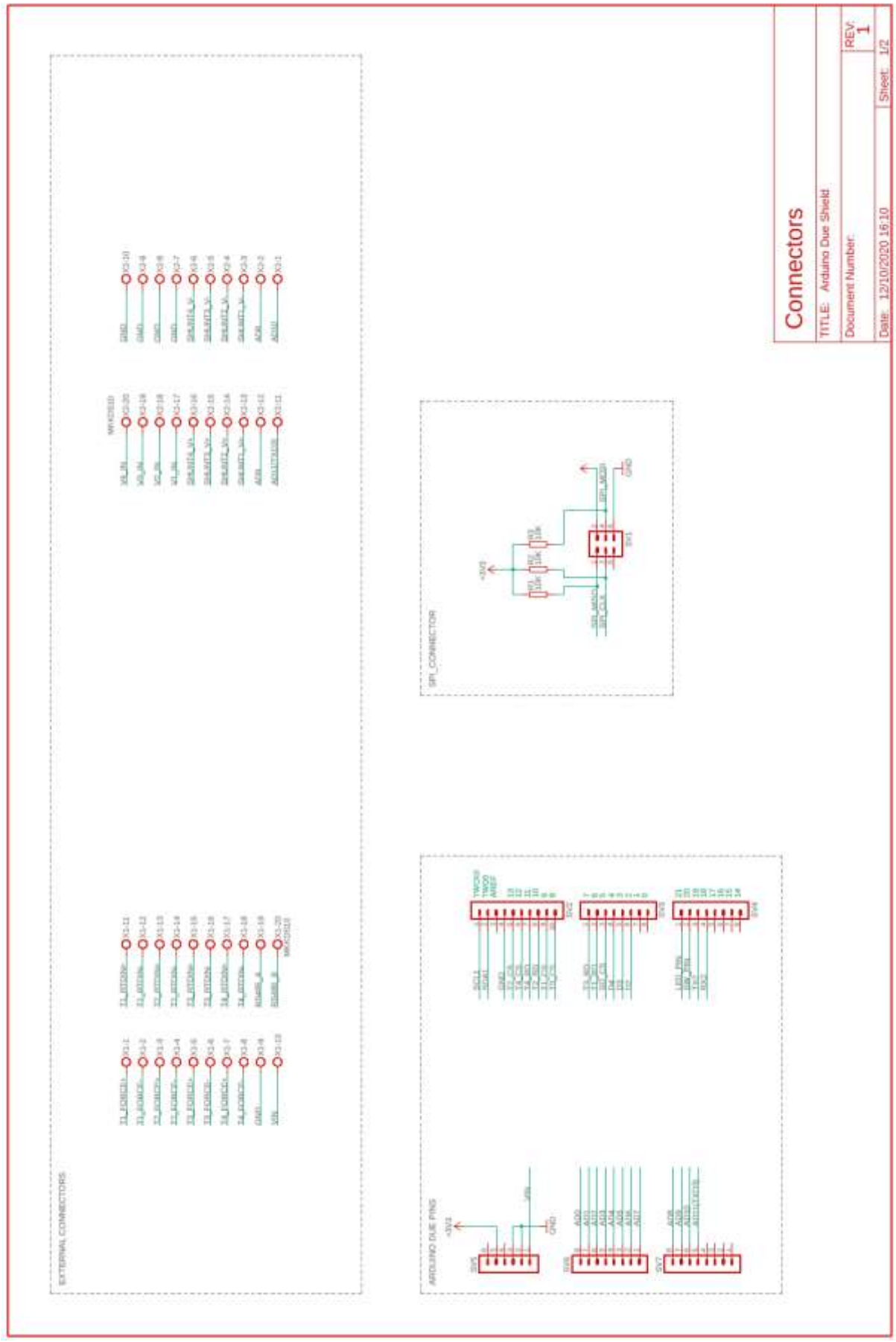


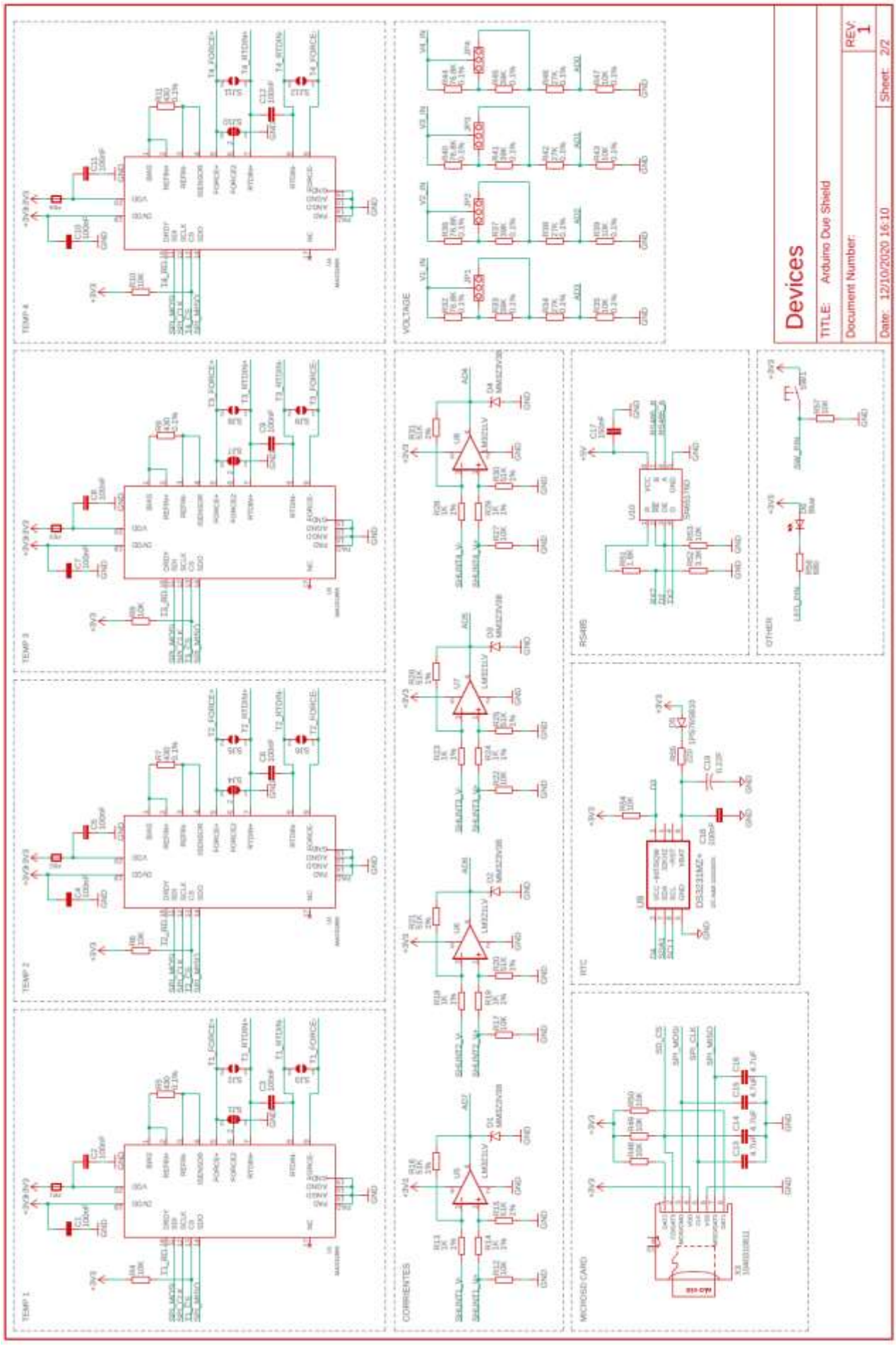


3.2 Layout Arduino Due

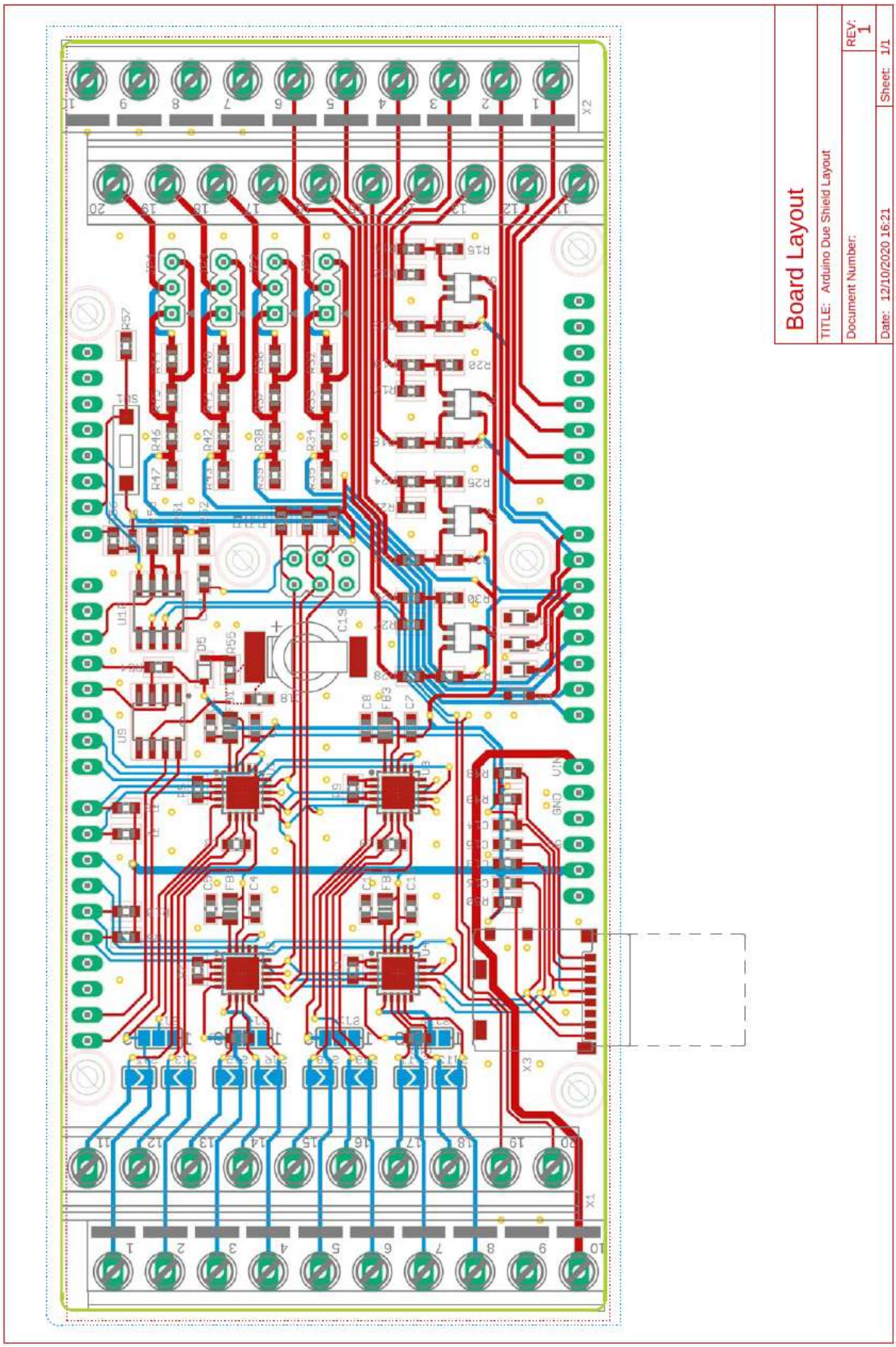


3.3 Esquema eléctrico Due Shield




Devices
TITLE: Arduino Due Shield
Document Number:
REV: 1
Date: 12/10/2020 16:10
Sheet: 2/2
TEMP 4
TEMP 3
TEMP 2
TEMP 1
VOLTAGE
CORRIENTES
RS485
OTHER
RTC
MICROSD CARD

3.4 Layout Due Shield



4 PLIEGO DE CONDICIONES

4.1 Condiciones técnicas y operativas

4.1.1 Equipamiento necesario

Para la correcta realización de este proyecto es necesario el siguiente material:

- PC con Microsoft Windows 8.1 o superior, procesador de 64 bits con al menos 4 núcleos, y al menos 3 GB de memoria Ram.
- Un microcontrolador Arduino Due.
- Una tarjeta microSD.
- Cable micro-USB.
- Estación de soldadura con pistola de aire caliente.
- Fuentes de corriente y de tensión para realizar los tests del dispositivo.
- Toma de corriente con protección.

4.1.2 Directrices para el correcto funcionamiento del dispositivo.

- Las entradas de tensión nunca deben ser sometidas a tensiones superiores a las marcadas en las directrices de este proyecto, pudiendo dañar severamente el dispositivo si esto ocurriera. Así mismo, tampoco deben ser sometidas a tensiones de valor negativo.
- Todas las entradas de tensión están referenciadas a una misma masa común, al menos una de ellas debe de estar conectada con el dispositivo.
- Las entradas de tensión deben ser calibradas antes de su primer uso y después de cada cambio en la escala del divisor de tensiones. Así pues, se recomienda comprobar el estado de dicha calibración antes de cada uso del dispositivo.
- Se debe respetar siempre la polaridad de las Shunt en las entradas de corriente.
- Las resistencias Shunt deben colocarse siempre en el cable de retorno de la corriente, estando el borne de menor tensión conectado a 0v.
- Al ser la Shunt un elemento externo al dispositivo, de debe de prestar atención a la sensibilidad de esta, asegurándose de que, multiplicándola por la ganancia del circuito, no se excede la tensión máxima de entrada.
- Las entradas de corriente deben ser calibradas antes de su primer uso y cada vez que se conecte de nuevo una Shunt a dicha entrada. Se recomienda comprobar el funcionamiento de dicha entrada antes de cada uso.
- Aunque no es estrictamente necesario, se recomienda la calibración de las entradas de temperatura antes de cada uso.
- No se deben ajustar los jumpers o conectar/desconectar ningún elemento al dispositivo encontrándose este conectado a la alimentación.

5 MEDICIONES

Los componentes electrónicos necesarios, así como las herramientas y las horas de trabajo, de este proyecto vienen indicados a continuación:

Componentes electrónicos	Unidades
CAP CER 0.1UF 25V X7R 0603	16
CAP CER 0.15UF 16V X7R 0603	2
CAP CER 4.7UF 16V X5R 0603	5
CAP 220MF -20% +80% 3.3V SMD	1
RES SMD 220 OHM 1% 1/10W 0603	2
RES SMD 10K OHM 5% 1/10W 0603	20
RES SMD 1.8K OHM 5% 1/10W 0603	2
RES SMD 3.3K OHM 5% 1/10W 0603	2
RES SMD 680 OHM 5% 1/10W 0603	2
RES SMD 10K OHM 0.1% 1/5W 0603	6
RES SMD 27K OHM 0.1% 1/5W 0603	6
RES SMD 39K OHM 0.1% 1/5W 0603	6
RES SMD 430 OHM 0.1% 1/5W 0603	6
RES SMD 76.8K OHM 0.1% 1/5W 0603	6
RES SMD 51K OHM 1% 1/10W 0603	10
RES 1K OHM 1% 1/10W 0603	10
DIODE ZENER 3.3V 200MW SOD323F	5
DIODE SCHOTTKY 30V 200MA SOD323	1
IC OPAMP GP 1 CIRCUIT SOT23-5	5
IC RTD TO DIGITAL CONVERT 20QFN	5
IC RTC DS3231MZ+	1
IC TRANSCEIVER HALF 1/1 8SOIC	1
FERRITE BEAD 80 OHM 0805 1LN	5
SWITCH TACTILE SPST-NO 0.05A 32V	1
LED BLUE CLEAR 2SMD	1
TERM BLK 5P SIDE ENT 5.08MM PCB	4
CONN MICRO SD CARD PUSH-PULL R/A	1
CONN JUMPER SHORTING GOLD FLASH	6
CONN HDR 6POS 0.1 TIN PCB	2

Tabla 9: Componentes electrónicos utilizados.

Materiales y Herramientas	Unidades
Arduino Due	1
Estaño para soldadura	1
Soldador con pistola de aire caliente	1
Fabricación de la placa a medida	1

Tabla 10: Materiales y herramientas utilizados.

Horas de trabajo	Unidades
Diseño del esquema eléctrico	40
Diseño de la shield	40
Programación del firmware	100
Programación de la Interfaz Visual	80
Soldadura de los componentes	40

Tabla 11: Horas de trabajo empleadas.

6 PRESUPUESTO

6.1 Presupuesto de ejecución material (PEM)

Coste de los componentes Electrónicos			
Concepto	Unidades	Precio unitario (€)	Coste (€)
CAP CER 0.1UF 25V X7R 0603	16	0,029	0,464
CAP CER 0.15UF 16V X7R 0603	2	0,09	0,18
CAP CER 4.7UF 16V X5R 0603	5	0,24	1,2
CAP 220MF -20% +80% 3.3V SMD	1	1,74	1,74
RES SMD 220 OHM 1% 1/10W 0603	2	0,09	0,18
RES SMD 10K OHM 5% 1/10W 0603	20	0,039	0,78
RES SMD 1.8K OHM 5% 1/10W 0603	2	0,09	0,18
RES SMD 3.3K OHM 5% 1/10W 0603	2	0,09	0,18
RES SMD 680 OHM 5% 1/10W 0603	2	0,09	0,18
RES SMD 10K OHM 0.1% 1/5W 0603	6	0,23	1,38
RES SMD 27K OHM 0.1% 1/5W 0603	6	0,23	1,38
RES SMD 39K OHM 0.1% 1/5W 0603	6	0,23	1,38
RES SMD 430 OHM 0.1% 1/5W 0603	6	0,23	1,38
RES SMD 76.8K OHM 0.1% 1/5W 0603	6	0,23	1,38
RES SMD 51K OHM 1% 1/10W 0603	10	0,054	0,54
RES 1K OHM 1% 1/10W 0603	10	0,015	0,15
DIODE ZENER 3.3V 200MW SOD323F	5	0,18	0,9
DIODE SCHOTTKY 30V 200MA SOD323	1	0,23	0,23
IC OPAMP GP 1 CIRCUIT SOT23-5	5	0,29	1,45
IC RTD TO DIGITAL CONVERT 20QFN	5	4,62	23,1
IC RTC DS3231MZ+	1	7,09	7,09
IC TRANSCEIVER HALF 1/1 8SOIC	1	1,81	1,81
FERRITE BEAD 80 OHM 0805 1LN	5	0,16	0,8
SWITCH TACTILE SPST-NO 0.05A 32V	1	0,46	0,46
LED BLUE CLEAR 2SMD	1	0,39	0,39
TERM BLK 5P SIDE ENT 5.08MM PCB	4	7,35	29,4
CONN MICRO SD CARD PUSH-PULL R/A	1	1,86	1,86
CONN JUMPER SHORTING GOLD FLASH	6	0,09	0,54
CONN HDR 6POS 0.1 TIN PCB	2	0,56	1,12
Coste total (€)			81,82 €

Tabla 12: Coste de los componentes electrónicos.

Coste de materiales y herramientas			
Concepto	Unidades	Precio unitario (€)	Coste (€)
Arduino Due	1	42,00	42,00
Estaño para soldadura	1	3,00	3,00
Soldador con pistola de aire caliente	1	15,00	15,00
Fabricación de la placa	1	5,25	5,25
Envío de la placa	1	7,06	7,06
Coste total (€)			72,31 €

Tabla 13: Coste de materiales y herramientas.

Coste de Mano de Obra			
Concepto	Unidades	Precio unitario (€)	Coste (€)
Soldadura de los componentes	40	15,00	600,00
Coste total (€)			600,00 €

Tabla 14: Coste de Mano de Obra

Coste de Ingeniería			
Concepto	Unidades	Precio unitario (€)	Coste (€)
Diseño del esquema eléctrico	40	20,00	800,00
Diseño de la shield	40	20,00	800,00
Programación del firmware	100	20,00	2000,00
Programación de la Interfaz Visual	80	20,00	1600,00
Coste total (€)			5.200,00 €

Tabla 15: Coste de Ingeniería

Presupuesto de ejecución material	
Concepto	Unidades
Coste de los componentes Electrónicos	81,82 €
Coste de Material y Herramientas	72,31 €
Coste de Mano de Obra	600,00 €
Coste de Ingeniería	5.200,00 €
Coste total (€)	5.954,13 €

Tabla 16: Presupuesto de Ejecución Material

6.2 Presupuesto de ejecución por contrata parcial (PEC)

Presupuesto de ejecución por contrata parcial	
Concepto	Unidades
Presupuesto de ejecución material	5.954,13 €
Gastos generales (18%)	1.071,74 €
Beneficio industrial (6%)	357,25 €
Coste total (€)	7.383,13 €

Tabla 17: Presupuesto de ejecución por Contrata Parcial

6.3 Presupuesto de ejecución por contrata total

Presupuesto de ejecución por contrata total	
Concepto	Unidades
Presupuesto de ejecución por contrata parcial	7.383,13 €
I.V.A. (21%)	1.550,46 €
Coste total (€)	8.933,58 €

Tabla 18: Presupuesto de ejecución por Contrata Total

El coste total del proyecto es de:

OCHO MIL NOVECIENTOS TRENTA Y TRES EUROS CON CINCUENTA Y OCHO CÉNTIMOS