Universidad Tecnológica Nacional

Proyecto Final

Kit de desarrollo para Aplicaciones de Bajo Nivel Industrial con Arduino

---

Autores:

- Millicovsky Martín Javier
- Tomé Julián Amir

Proyecto final presentado para cumplimentar los requisitos académicos para acceder al título de Ingeniero Electrónico

en la

Facultad Regional Paraná

Fecha (Septiembre de 2019)

Declaración de autoría:

Yo/nosotros declaro/declaramos que el Proyecto Final “Título” y el trabajo realizado son propio/s. Declaro/declaramos:

- Este trabajo fue realizado en su totalidad, o principalmente, para acceder al título de grado de Ingeniero _____, en la Universidad Tecnológica Nacional, Regional Paraná.
- Se establece claramente que el desarrollo realizado y el informe que lo acompaña no han sido previamente utilizados para acceder a otro título de grado o pre-grado.
- Siempre que se ha utilizado trabajo de otros autores, el mismo ha sido correctamente citado. El resto del trabajo es de autoría propia.
- Se ha indicado y agradecido correctamente a todos aquellos que han colaborado con el presente trabajo.
- Cuando el trabajo forma parte de un trabajo de mayores dimensiones donde han participado otras personas, se ha indicado claramente el alcance del trabajo realizado.

Firmas:

-
-
-

Fecha:

Agradecimientos:

En agradecimiento a nuestras familias, las cuales fueron el pilar fundamental por brindarnos el privilegio de estudiar la presente carrera, brindando el apoyo sentimental y económico necesario. A las amistades que acompañaron en este difícil proceso de manera incondicional y aquellos profesores que además de brindar conocimientos fomentaron la formación humana ante todo.

Millicovsky Martín Javier

Tomé Julián Amir

Universidad Tecnológica Nacional

Abstract

Facultad Regional Paraná
Ingeniero en Electrónica

Kit de desarrollo para aplicaciones de bajo nivel Industrial con Arduino

Millicovsky Martín Javier
Tomé Julián Amir

Abstract:

Aiming to obtain a device that emulates a remote terminal unit for automations sistems or control, where the industrial exigency isn´t relatively high, a set of tools were developed to conform in its entirety a development kit for industrial applications. In order to make it, some modules and sistems that belong to today´s market were used. All is oriented to open systems and universality. It´s about Arduino and its modules. Own designs were implemented, which are going to adapt different signals for processing and finish it in an unification on a 3D printed cabinet

Finally, a total functional kit was obtained as an example, with different communication ports and big capacity for several signal proccesing. Also with benefits capables of compete with similar devices that appear at the industrial market.

Keywords: PLC, Arduino, Open System, Automatization, Control.

Resumen:

Con el fin de obtener un dispositivo emulando una unidad terminal remota, orientado al control o automatización de sistemas donde la robustez industrial no sea relativamente exigente, se desarrolló un conjunto de herramientas las cuales en su totalidad conformarán un kit de desarrollo para aplicaciones Industriales. Para la realización del mismo se acudió al uso de sistemas o módulos existentes en el mercado actual, orientados a sistemas abiertos y de carácter universal, como Arduinos y sus respectivos módulos, además se emplearon diseños propios los cuales adaptarán diferentes señales a procesar para culminar en su unificación sobre un gabinete impreso en 3D.

Al final se obtuvo un ejemplo de kit totalmente funcional, con diferentes puertos de comunicación y capacidad amplia de procesamiento para diferentes señales, con prestaciones capaces de competirle a dispositivos similares presentes en el mercado Industrial.

Palabras Clave: PLC, Arduino, Sistema Abierto, Automatización, Control.

Reconocimientos:

Se reconoce a los Ingenieros Katzenelson Gustavo, Pico Martín y Weber Adrián los cuales colaboraron de manera incondicional en la realización del proyecto, de la misma manera al compañero Arribillaga Marcos, futuro Ingeniero, el cual siempre estuvo a disposición para colaborar en lo que se necesitara.

Índice:

Lista de Figuras

Lista de Tablas

Lista de Abreviaciones

Abreviación	Significado
AC	Corriente alterna
ANSI	Instituto Nacional Estadounidense de Estándares
ARPANET	Red de la Agencia de Proyectos de Investigación Avanzada
ASCII	Código Estadounidense Estándar para el Intercambio de Información
CAN	Controlador de Red de Zona
CC	Corriente continua
CCR	Centro de Control de la Red
CLK	Señal de reloj
CPU	Unidad central de procesamiento
CRC	Verificación por redundancia cíclica
DARPA	Agencia de Proyectos de Investigación Avanzados de Defensa
DC	Corriente directa
EEPROM	ROM programable y borrable eléctricamente
FSK	Modulación por desplazamiento de frecuencia
GFSK	Modulación por desplazamiento de frecuencia gaussiana
GND	Tierra o referencia
GPIO	Entrada/Salida de Propósito General
HDLC	Control de enlace de datos de alto nivel
I2C	Inter integrated circuits
ID	Identification
IDE	Entorno de desarrollo integrado
IEEE	Instituto de Ingeniería Eléctrica y Electrónica
IP	Protocolo de Internet
IRAM	Instituto Argentino de Normalización y Certificación
JTAG	Grupo de Acción Conjunta de Prueba
KIT	Equipo
LED	Diodo emisor de luz
LoRa	Redes de baja potencia y área amplia
LOT	Internet de las cosas
M2M	Máquina a máquina
MAC	Control de acceso a medios
MCU	Microcontrolador
MISO	Salida de datos del Esclavo y entrada al Master
MOSI	Salida de datos del Master y entrada de datos al Esclavo
OSI	Interconexión de sistemas abiertos

PC	Computadora personal
PCB	Placa de circuito impreso
PCI	Estándar de Seguridad de Datos para la Industria de Tarjeta de Pago
PCMCIA	Asociación internacional de tarjetas de memoria para equipos personales
PLC	Controlador lógico programable
PWM	Modulación por ancho de pulsos
RTU	Unidad terminal remota
SCADA	Supervisión, Control y Adquisición de Datos
SCK	Pulsos de reloj
SMD	Componentes de montaje superficial
SPI	Interfaz periférica serial
SRAM	Memoria estática de acceso aleatorio
TCP	Protocolo de Control de Transmisión
TTL	Lógica transistor a transistor
UART	Transmisor-Receptor Asíncrono Universal
UHF	frecuencia ultra alta
USB	Bus Universal en Serie
UTP	Par trenzado no blindado
UTR	Unidad terminal remota
VCC	Voltaje de corriente continua
Vin	Voltaje de entrada
VLAN	Red de área local virtual
Vout	Voltaje de salida

Dedicado a:

Se dedica este proyecto en recuerdo y con mucho afecto al Ing. Isaac Millicovsky, padre de Martín Millicovsky.

Capítulo 1: Introducción

Pensado inicialmente en desarrollar un dispositivo que sirva para tareas de control y sea dirigido a pequeños laboratorios, empresas y fábricas. Se habla de un producto que sea la primera opción al momento de encarar un problema. Se busca abrir la mente a los clientes, opciones basadas en mejores precios y con una fiabilidad aceptable.

Se elige el control como el punto de partida para el proyecto debido a que existe un sector del mercado con gran potencial y es una parte de la ingeniería que está en constante avance y debate. El hecho de optimizar y buscar puntos que mejoren tanto en prestaciones como en lo económico en un motivo adicional en la presente elección.

Cuando se habla de control se refiere a un área de la ingeniería que se encarga de controlar sistemas dinámicos (sistemas donde el estado evoluciona con el tiempo, los efectos actuales son el resultado de causas actuales y previas) mediante controladores, para que el comportamiento sea el deseado, como por ejemplo el control de temperatura en una habitación. Actualmente se habla en muchos de casos de control automático, donde se busca manejar con una o más entradas (referencia), una o más salidas.

El objetivo del proyecto es realizar una aplicación de cualidades industriales relativamente bajas usando un Arduino como cerebro, una opción para pequeños propósitos, dado que la idea es introducir en el mercado un producto que ofrezca numerosas prestaciones y a un precio razonable.

El dispositivo a comparar a primera vista, de acuerdo a lo pretendido, es un controlador lógico programable. Estos se emplean en la ingeniería automática. En función de ciertas indicaciones que se le brinden, denominadas entradas, este dispositivo será capaz de elaborar salidas que solucionen el inconveniente. Analizando los productos de marca del mercado, nos referimos a PLC y PLC basados en Arduino, se observa la falta de unificación en sus características. Este último comentario hace referencia a que, al adquirir un artefacto de una determinada marca, en muchos casos si se desea realizar una expansión o modificación del dispositivo deberán ser de la misma firma, resaltando que el costo económico es notable. Distinto es una computadora de escritorio, en la cual se puede elegir el mouse, teclado, placa de video, u otra parte que queramos. Este es el punto de partida para lo buscado, una base a partir de la cual el usuario pueda agregar módulos libremente, según lo que esté desarrollando, por lo que el producto en sí sería en pocas palabras un gabinete o kit de inicio con el cual empezar a trabajar. Todo esto marca claramente que lo que se busca no es unir y hacer un aparato que englobe todas las prestaciones, sino que para múltiples propósitos podamos elegir el kit o gabinete adecuado con su respectivo Arduino. A diferencia de la mayoría de productos en el mercado, el hecho de usar un Arduino, una Raspberry o cualquier sistema embebido abierto, da la posibilidad de poder agregar herramientas de forma flexible y libre al kit.

Al sensar temperatura para cortar un motor por emergencia, partimos del kit con el Arduino y agregamos las herramientas necesarias para que este pueda cumplir la meta.

La diferencia con lo que se ve en el mercado es que se tiene que elegir el dispositivo terminado, esto implica que solo sirve para determinadas aplicaciones o en

caso inverso es demasiado para lo necesario. Sin embargo, con el actual proyecto un usuario puede configurarlo y amoldarlo según las necesidades, gastando solo lo necesario en cuando a dinero. Si bien se habla de partir de una base, también hay que tener en cuenta que es lo que se agrega, en función de la tarea a realizar.

Indagando en internet se puede observar como dispositivo “similar” a los denominados Ardbox “https://www.industrialshields.com”. Aquí veremos muchos productos basados en Arduino y Raspberry que fueron realizados para solucionar inconvenientes industriales. Es cuestión de analizar las características y compararlas con el que se requiera usar.

Cada idea o planteo va a ser diferente, por lo que el Arduino o sistema embebido a usar y las placas y módulos que lo complementen también. Todo esto es parte de un estudio previo: dado un cliente con una situación o aplicación a realizar hay que determinar el cerebro del kit, que precisamos en función de sus recursos, y si contamos con los extras necesarios para completar el desarrollo. Si este estudio es positivo, se puede optar por el kit y adquirir todo para comenzar el armado del dispositivo final, para dicho usuario. Por esta razón es que se habla de un proyecto de concepto abierto, una experiencia que busca reflejar ventajas pero que tiene puntos a tratar y mejorar. Para corroborar que el producto funcione se requiere desarrollar una aplicación, usando un Arduino Due, y observar los resultados. El objetivo es realizar tareas de diferentes tipos para ver cómo responde sobre todo complementándose con máquinas de alta exigencia. Se realizarán comunicaciones con Lora WiFi y Modbus RS485 y TCP IP. También placas para la comprensión de señales de entrada y el manejo de actuadores a la salida.

Los pasos para encarar una aplicación son los siguientes: empezamos por estudiar las tareas a realizar y que Arduino es el indicado para poder realizarlas. Luego adquirir los módulos y realizar los circuitos necesarios. Finalmente es cuestión de ubicar todo en un gabinete. El paso más importante de todos es el primero, debido a que será la base en lo cual se determinará su factibilidad y será la base de los demás pasos.

Capítulo 2: Desarrollo

En el presente capítulo se detallará en tres partes el proyecto, desmenuzando cada bloque en distintos puntos, para comprender de mejor forma como se fue desarrollando la aplicación.

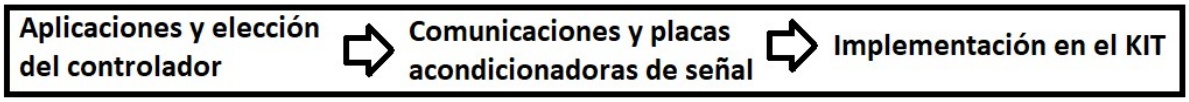


Imagen 1. Diagrama en bloques del Proyecto

A) Investigación de componentes: Análisis y selección

En este bloque se comienza por entender que es lo que se desea realizar, es decir, la aplicación o conjunto de tareas que el KIT final será capaz de implementar de esta forma se sabrá lo que se necesita posteriormente.

Las prestaciones buscadas serán poder comunicarse con otros dispositivos y poder manejar señales tanto de entrada como de salida.

Los dispositivos que quieran comunicarse lo harán mediante el protocolo Modbus, ya sea por RS485 o TCP IP. También estará la opción de hacerlo vía LoRa WiFi.

El manejo de señales es con el propósito de realizar acciones en función del análisis de datos que ingresen. En cuanto a las señales que entren para su análisis, éstas podrán ser digitales y/o analógicas y contará con placas que adapten la información proveniente al controlador elegido para su posterior procesamiento.

A.1) Introducción a las comunicaciones RS485-Ethernet-Lora Wifi

Se comienza por explicar que significa comunicarnos: hacer saber algo a un receptor determinado. Uno de los fines más importantes del proyecto es el de manejar y transportar información. En este apartado se hablará de los métodos que se usarán para transportar la información hacia otro dispositivo. El emisor seria el producto mientras que el receptor sería quien esté solicitando.

Para conectar dos partes de una maquinaria es necesario tener una comunicación entre ellas. La comunicación está compuesta de dos partes, una parte física o hardware y una parte de programación o firmware. El hardware está compuesto por los materiales necesarios para la comunicación, ya sea el cableado, el tipo de conector o el conversor, mientras que el firmware está compuesto por las instrucciones necesarias o tramas para el funcionamiento de la máquina.

Antes de continuar se definirán dos conceptos importantes: red y protocolo. Una red es una configuración de computadora que intercambia información. Pueden proceder de una variedad de fabricantes y es probable que tenga diferencias tanto en hardware como en software, para posibilitar la comunicación entre estas es necesario un conjunto de reglas formales para su interacción. A estas reglas se les denominan protocolos. Un protocolo es un conjunto de reglas establecidas entre dos dispositivos para permitir la comunicación entre ambos.

A.1.1) Comunicación por RS485

Para una comunicación a grandes distancias y velocidades de transmisión se utiliza la norma RS485. Permite la trasmisión multipunto, es decir un ordenador central conectado con varias UTR. Se denomina trasmisión diferencial y permite velocidades de hasta 10 Mbps, sobre distancias de hasta 1.3 kms.

Se usan dos señales para trasmitir y dos para recibir, además de la tierra, la cual se suele conectar a la malla del cable. Se envía la señal de trasmisión y su complemento, mientras que, en el receptor, la señal original se obtiene restando las dos señales, lo cual ayuda a reducir notablemente el ruido generado en la línea, ya que éste se adhiere en ambas líneas por igual, con lo que se consigue cancelarlo al restar ambas señales. Para la comunicación por RS485 se usan 2 señales A-B, el GND y es necesario un protocolo "half dúplex", ya que las líneas son usadas tanto para la trasmisión como para la recepción. Half Duplex transmite y recibe en ambas direcciones, pero sólo ocurre una transmisión a la vez, es decir, no hay comunicación bidireccional simultáneamente, se debe esperar a que se termine de transmitir para poder recibir.

En la imagen se puede observar el dispositivo y el cable. Desde Arduino se emite una señal que pasa por el conversor y sale con las señales AB. El cable RS485 de la imagen es concreto, se tienen los dos hilos y un conector industrial. Este conector se puede ilustrar de la siguiente forma:

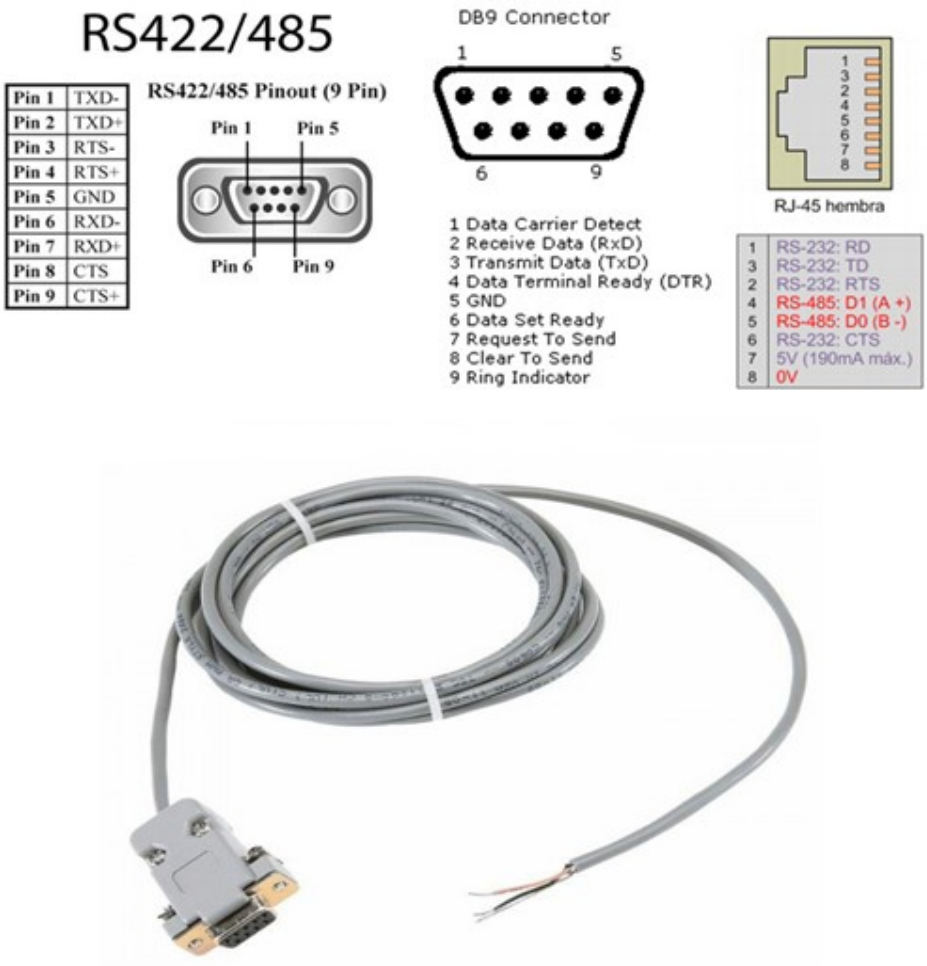


Imagen 2. Conector RS485-Pines

A.1.2) Comunicación por TCP IP- Ethernet

Se han desarrollado diferentes familias de protocolos para comunicación por red de datos para los sistemas UNIX. El más ampliamente utilizado es el Internet Protocol Suite, comúnmente conocido como TCP / IP. Es un protocolo DARPA que proporciona transmisión fiable de paquetes de datos sobre redes. El nombre TCP / IP Proviene de dos protocolos importantes de la familia, el Transmission Control Protocol (TCP) y el Internet Protocol (IP). Todos juntos llegan a ser más de 100 protocolos diferentes definidos en este conjunto.

El TCP / IP es la base del Internet que sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local y área extensa. Fue desarrollado y demostrado por primera vez en 1972 por el departamento de defensa de los Estados Unidos, ejecutándolo en el ARPANET, una red de área extensa del departamento de defensa.

Una red TCP/IP transfiere datos mediante el ensamblaje de bloques de datos en paquetes, cada paquete comienza con una cabecera que contiene información de control; tal como la dirección del destino, seguido de los datos. Cuando se envía un archivo por la red TCP/IP, su contenido se envía utilizando una serie de paquetes diferentes. El Internet Protocol (IP), un protocolo de la capa de red, permite a las aplicaciones ejecutarse transparentemente sobre redes interconectadas. Cuando se utiliza IP, no es necesario conocer que hardware se utiliza, por tanto, ésta corre en una red de área local.

El TCP, un protocolo de la capa de transporte, asegura que los datos sean entregados, que lo que se recibe, sea lo que se pretendía enviar y que los paquetes que sean recibidos en el orden en que fueron enviados. TCP terminará una conexión si ocurre un error que haga la transmisión fiable imposible.

Ethernet es un estándar de redes de área local para computadores. Define las características de cableado y señalización, de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI. Ahora veremos el formato de la trama:

-El primer campo es el preámbulo que indica el inicio de la trama y tienen el objeto de que el dispositivo que lo recibe detecte una nueva trama y se sincronice.

-El delimitador de inicio de trama indica que el frame empieza a partir de él.

-Los campos de MAC (o dirección) de destino y origen indican las direcciones físicas del dispositivo al que van dirigidos los datos y del dispositivo origen de los datos, respectivamente.

-La etiqueta es un campo opcional que indica la pertenencia a una VLAN o prioridad en IEEE P802.1p

-Ethernetype indica con que protocolo están encapsulados los datos que contiene la Payload, en caso de que se usase un protocolo de capa superior.

-La Payload es donde van todos los datos y, en el caso correspondiente, cabeceras de otros protocolos de capas superiores. Tiene un mínimo de 64 Bytes (o 42 si es la versión 802.1Q) hasta un máximo de 1518 Bytes. Los mensajes inferiores a 64 bytes se llaman tramas enanas (runt frames) e indican mensajes dañados y parcialmente transmitidos.1

-La secuencia de comprobación es un campo de 4 bytes que contiene un valor de verificación CRC (control de redundancia cíclica). El emisor calcula el CRC de toda la trama, desde el campo destino al campo CRC suponiendo que vale 0. El receptor lo recalcula, si el valor calculado es 0 la trama es válida.

-El gap de final de trama son 12 bytes vacíos con el objetivo de espaciado entre tramas.

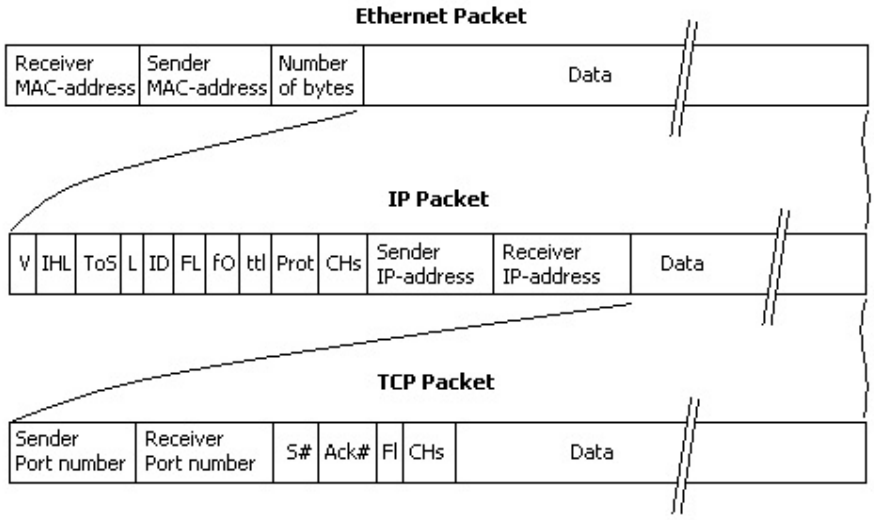


Imagen 3. Trama Ethernet

Hace ya mucho tiempo que Ethernet consiguió situarse como el principal protocolo del nivel de enlace. Ethernet 10Base2 consiguió, ya en la década de los 90s, una gran aceptación en el sector. Hoy por hoy, 10Base2 se considera como una "tecnología de legado" respecto a 100BaseT. Hoy los fabricantes ya han desarrollado adaptadores capaces de trabajar tanto con la tecnología 10baseT como la 100BaseT y esto ayuda a una mejor adaptación y transición. Las tecnologías Ethernet que existen se diferencian en estos conceptos entre ellos:

Velocidad de transmisión: velocidad a la que transmite la tecnología.

Tipo de cable: tecnología del nivel físico que usa la tecnología.

Longitud máxima: distancia máxima que puede haber entre dos nodos adyacentes.

Topología: forma física de la red. Si se usan conectores T y estrella, hubs o switchs.

En función de lo utilizado para establecer comunicaciones tendremos tres tipos:

Tecnología	Velocidad de transmisión	Tipo de cable	Distancia máxima	Topología
100BaseT4	100 Mbit/s	Par Trenzado (categoría 3UTP)	100 m	Estrella. Half Duplex (hub) y Full Duplex (switch)
100BaseTX	100 Mbit/s	Par Trenzado (categoría 5UTP)	100 m	Estrella. Half Duplex (hub) y Full Duplex (switch)
1000BaseT	1000 Mbit/s	(categoría 5e ó 6UTP)	100 m	Estrella. Full Duplex (switch)

Tabla 1. Tecnologías Ethernet a disposición

Para la aplicación se empleará un cable derecho debido a que se mandan datos hacia la computadora por el puerto Ethernet.

Hay que tener en cuenta es que las redes de ordenadores no utilizan los 4 pares del cable UTP (8 cables) en su totalidad, utilizan solamente 4 cables: 2 para transmitir y 2 para recibir. La razón de esta distinción entre cable cruzado y directo son estos pines de transmisión y recepción de los distintos dispositivos dentro de una red.

Por ejemplo, si conectamos un cable directo entre dos PCs, no conseguiremos comunicación entre ellos ya que estaremos transmitiendo datos hacia el pin de transmisión del otro PC cuando debemos hacerlo al pin de recepción. Lo mismo ocurriría con dos routers o Switches.

Los equipos activos más modernos, detectan automáticamente el tipo de cable van a recibir, si uno recto o uno cruzado, facilitando así la tarea de instalación y mantenimiento de una red. El cable cruzado es utilizado para conectar dos PCs directamente o equipos activos entre sí, como hub con hub, con switch, router, etc. Un cable cruzado es aquel donde en los extremos la configuración es diferente. Dicha configuración o montaje lo podemos comprobar fijándonos en los colores de los hilos dentro del conector RJ45. El cable cruza las terminales de transmisión de un lado para que llegue a los pines de recepción del otro, y la recepción del origen a transmisión del final. Este tipo de cable se usa para conectar equipos “iguales”, es decir, PC con PC y equipos activos de red entre sí (router con router, switch con hub, hub con router, etc).

Para crear el cable de red cruzado, se arma con un rj45 un extremo del cable siguiendo la norma T568A y el otro extremo siguiendo la T568B. El cable directo es sencillo, hay que tener la misma norma en ambos extremos del cable.

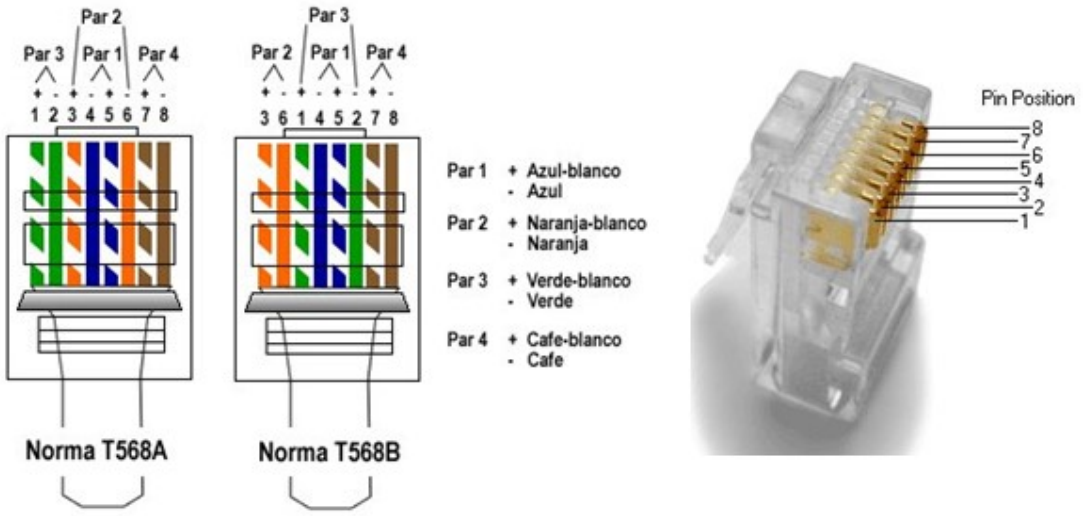


Imagen 4. Normas de cableado UTP para RJ45

A.1.3) Comunicación por LoRa Wifi

LoRa es una tecnología patentada (EP2763321 de 2013 y US7791415 de 2008) desarrollada por Cycleo (Grenoble, Francia) y adquirida por Semtech en 2012. LoRa utiliza bandas de frecuencia de radio subgigahertz sin licencia como 169 MHz, 433 MHz, 868 MHz (Europa) y 915 MHz (América del Norte).

Dirigido al mercado M2M (máquina a máquina) e IoT (internet de las cosas), LoRa es ideal para proporcionar conectividad intermitente de baja velocidad de datos en distancias significativas. La interfaz de radio ha sido diseñada para permitir que se reciban niveles de señal extremadamente bajos y, como resultado, incluso las transmisiones de baja potencia pueden recibirse en rangos significativos.

La modulación y la interfaz de radio de LoRa se han diseñado y optimizado para proporcionar exactamente el tipo de comunicaciones necesarias para los nodos IoT y M2M remotos. La tecnología ahora se está implementando ampliamente. Se está incorporando a muchos sistemas, e incluso las computadoras pequeñas, como Arduino, tienen opciones de LoRa. En consecuencia, es muy fácil desarrollar aplicaciones para LoRa tanto para la fabricación a gran escala como para las aplicaciones más especializadas. La baja potencia y las capacidades de largo alcance significan que los puntos finales se pueden implementar en una amplia variedad de lugares, en edificios y en el exterior.

Las aplicaciones para la tecnología inalámbrica LoRa incluyen: medición inteligente; seguimiento de inventario, datos de máquinas expendedoras y monitoreo; industria automotriz; aplicaciones de utilidad, entre otros. En resumen, en cualquier lugar donde sea necesario el reporte y control de datos.

El "padre" de LoRa es la empresa Semtech, que desarrolló la tecnología y posee la patente. Ahora la fundación LoRa Allience, se encarga del desarrollo del estándar y su evolución.

El standard de 868 MHz porque ofrece unas características superiores en cuanto al alcance (es 2 o 3 veces mayor que los estándares mencionados anteriormente en la banda de 2,4 GHz). El coste de la implementación es relativamente bajo. El consumo de energía es bajo. La versión de 868 MHz no está disponible en todo el mundo. En los EEUU se requiere una banda de 915 MHz. El hardware para ambos estándares es idéntico sin embargo requieren versiones de software distintas.

La banda de 433 MHz es ligeramente mejor respecto al alcance, pero no garantiza una transmisión segura de los datos. Debido a que muchos dispositivos trabajan en esta banda, especialmente controles remotos, es frecuente que se produzcan perturbaciones en la transmisión.

LoRa y LoRaWan

LoRaWan es una especificación de redes LPWAN (Low Power Wide Area Network). Atendiendo a los niveles OSI, sería el nivel 2 (red). Es lo que se conoce como MAC (Media Access Control). Digamos que LoRaWAN se encarga de unir diferentes dispositivos LoRa gestionando sus canales y parámetros de conexión: canal, ancho de banda, cifrado de datos, etc.

En el nivel 1 de OSI, nivel físico, encontramos la tecnología LoRa de comunicación. Esta tecnología permite el envío y recepción de información punto-a-punto. Lo que caracteriza a un dispositivo LoRa es su largo alcance con un mínimo dispositivo. Para ello emplea la técnica de espectro ensanchado, donde la señal a mandar utiliza más ancho de banda que el necesario teóricamente pero que permite una recepción de múltiples señales a la vez que tengan distinta velocidad.

Las frecuencias de comunicaciones que LoRa usan son principalmente las de la banda ISM, aunque la tecnología puede operar en cualquier frecuencia por debajo del 1 GHz.

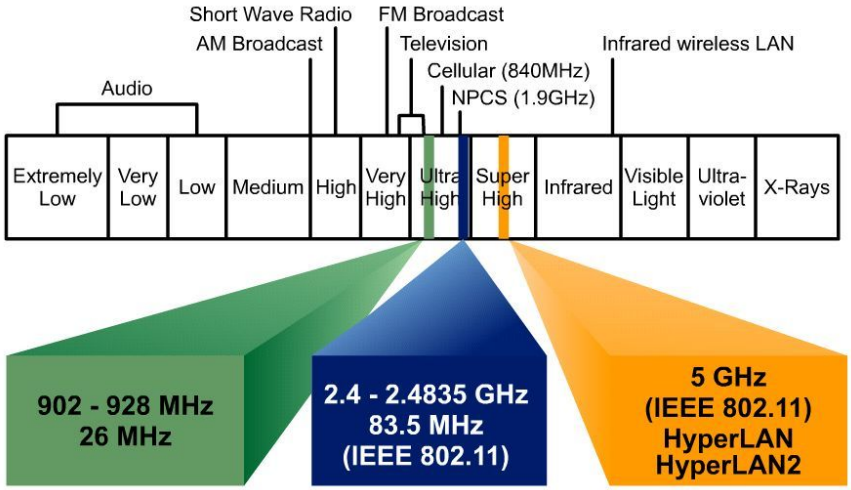


Imagen 5. Espectro de Frecuencia

El uso de estas frecuencias se debe a que mientras se respete los valores de emisión, cualquier persona o empresa puede hacer uso de ella sin necesidad de licencia.

Los parámetros de comunicación LoRa son:

Canal dentro de la banda: frecuencia central que representa el canal. El canal 10 dentro de la banda 868 MHz tiene un valor de 865.200.000 Hz.

Spreading factor (SF): define el número de bits usados para codificar un símbolo. A mayor SF, menor velocidad de transferencia tendremos, pero mayor inmunidad a interferencias.

Coding rate (CR): indica la forma de codificar para corrección de errores. Es decir, según la técnica especificada, añade símbolos de control para saber si los datos son correctos o no e incluso poder determinar los valores correctos.

Bandwidth (BW): indica el ancho de frecuencia que vamos a usar

Establecer una comunicación punto-a-punto con LoRa es relativamente sencillo. Para distancias muy cortas (menor a 1 KM) no suele haber mucho problema. Se complica para distancias más grandes. Además de usar unas antenas con mayor ganancia y que sean visibles directamente unas a las otras sin obstáculos de por medio, es necesario una correcta configuración de canal, velocidad, etc. Aquí es donde entra en escena LoRaWAN, subiendo de nivel y encargándose también de cifrar los datos para que ajenos no hagan uso de ellos. Lo único de lo que debemos de preocuparnos es de registrar e identificar nuestro dispositivo dentro de la red.

LoRa en el mundo Arduino

Poco a poco LoRa va teniendo presencia en el mundo y ya es posible realizar proyectos en plataformas como Arduino o Raspberry PI.

Existen varias empresas que comercializan shields y módulos para integrar, desde LoRa a incluso LoRaWAN. Al final todas estas soluciones integran en su base el chip de radio de Semtech. Los principales integrados usados son: SX1272, SX1276 y SX1278.

La mayoría de integradores añaden circuitería adicional para que sea prácticamente plug&play ponerse a desarrollar con LoRa. La comunicación con los microcontroladores se hace mediante SPI. Entre los principales productos encontramos:

SX1278 LoRA + ESP32 (WiFi/Bluetooth SoC), ideal para empezar y practicar en el mundo LoRa. Tiene pantalla OLED y se puede programar desde Arduino IDE.

Libelium Waspmote LoRa, producto de calidad, bien acabado y precio acorde a la calidad. Disponible para Arduino y Raspberry PI.

LoPy, SoC que lleva WiFi+Bluetooth 4+LoRa, ideal para prototipar en Python (Micropython) proyectos y experimentar con LoRa.

ModTronix inAir4/9/9B, módulo que requiere cierto montaje pero que no es para nada complejo. Fácil de integrar en una protoboard.

HopeRF, módulo más complejo de integrar, pues lo ideal es integrarlo sobre PCB.

Open Source LoRa Gateway, si no se quiere acomplejarse para intercomunicar dispositivos LoRa, pero te quieres reservar el derecho a poder modificar en un futuro, este es tu Gateway.

Existen algunas soluciones que directamente implementan LoRaWAN (no siendo posible acceder a bajo nivel). Soluciones como la de Microchip, RN2483.

El ADN IoT

Diseñado para las comunicaciones de Internet de las cosas, la tecnología LoRa permite la conexión entre dispositivos remotos de punto de uso y LPWAN para su entrega a aplicaciones de análisis.

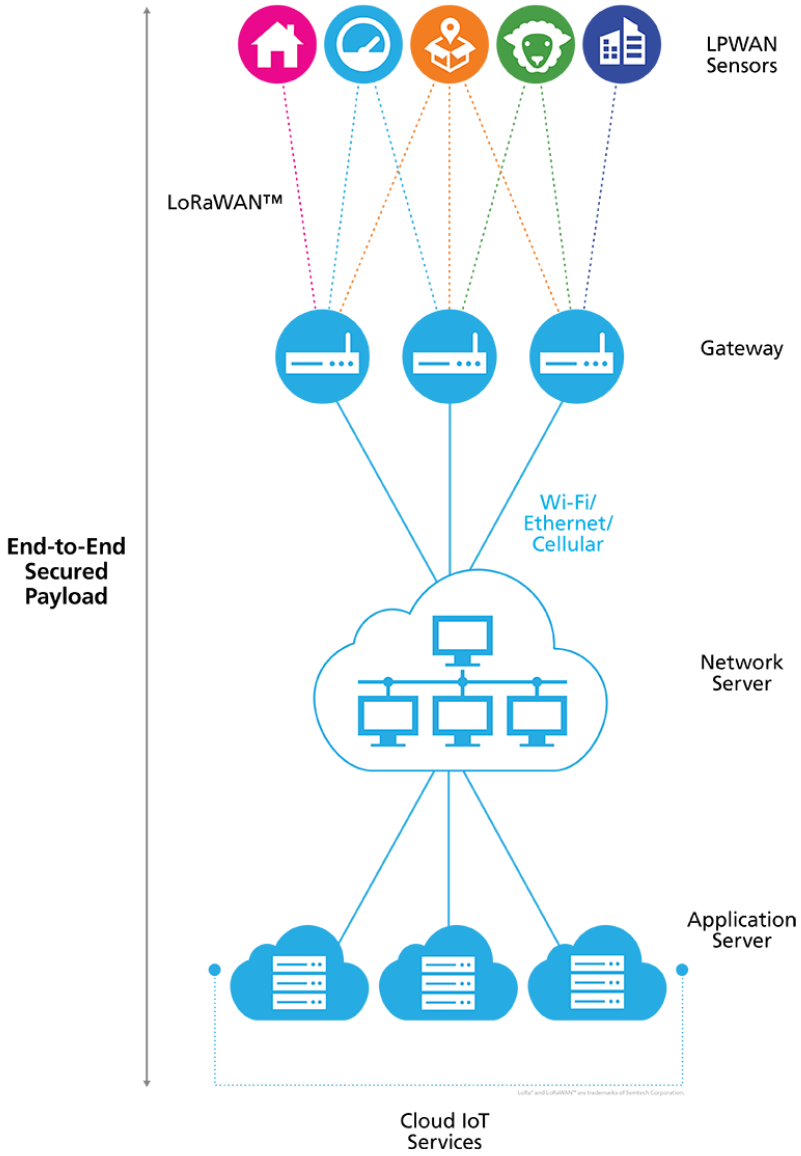


Imagen 6. Diseño de un sistema con LoRa

Modulación LoRa: la modulación inalámbrica, utilizada para crear el enlace de comunicación de largo alcance.

Transceptores y nodos finales: los transceptores configurados con tecnología LoRa están integrados en nodos finales o dispositivos sensores diseñados para una multitud de aplicaciones de la industria.

Picocélulas y pasarelas: los sensores capturan y transmiten datos a las pasarelas a distancias cercanas y lejanas, interiores y exteriores, con un requisito mínimo de energía.

Servidor de red: los Gateways envían información a través de Wi-Fi, Ethernet o celular al servidor de red, que es responsable de las funciones de administración de la red, como la activación por aire, la de duplicación de datos, el enrutamiento dinámico de cuadros, el control de velocidad adaptable y la administración del tráfico. y administración.

Servidores de aplicaciones y servicios de IoT en la nube: las aplicaciones interpretan los datos recopilados por los dispositivos habilitados con LoRa, aplicando técnicas como el aprendizaje automático y la inteligencia artificial para resolver problemas de negocios para un planeta más inteligente.

LoRa complementa Wi-Fi, Bluetooth y celular

Al igual que Wi-Fi, LoRa funciona en la banda sin licencia y es compatible con aplicaciones en interiores, al igual que el celular. La tecnología es altamente segura desde los dispositivos finales hasta el servidor de aplicaciones, y es adecuada para aplicaciones en exteriores. Combina estas características de redes Wi-Fi y celulares para ofrecer una solución de conectividad eficiente, flexible y económica ideal para aplicaciones de IoT, ya sea en interiores o exteriores e instaladas en redes públicas, privadas o híbridas. Los datos de sensores simples pueden alimentar las plataformas de análisis, como las de inteligencia artificial y aprendizaje automático. Estos requieren una diversidad de datos que es posible gracias a los sensores de bajo costo habilitados para LoRa.

A.1.4) Protocolos Modbus RS485-TCP IP

Un protocolo de comunicación describe un estándar a seguir para que la comunicación y el entendimiento entre dos o más equipos sea favorable, dentro de un protocolo se define tanto la velocidad de comunicación, como la longitud de la trama. Depende de cada protocolo esta trama puede variar de longitud, ya que se pueden aplicar bits especiales como el de inicio o el de Stop. Para continuar deberemos explicar conceptos sobre el modelo OSI.

El modelo OSI consiste en un estándar que regula la comunicación entre distintos sistemas. Está compuesto por siete capas distintas.

Imagen 7. Modelo OSI

El modelo OSI consiste en un estándar que regula la comunicación entre distintos sistemas. Está compuesto por siete capas distintas. Cada capa tiene su función y debe prestar unos servicios que serán usados por la capa superior.

Existen dos tipos de comunicación dentro del modelo OSI, la comunicación horizontal entre dos equipos en la misma capa y la comunicación vertical entre una capa y su superior. Las principales funciones de las capas de aplicación son:

- Capa Física: transmite la información entre dos equipos.
- Capa Enlace de Datos: estructura la información bajo una trama.
- Capa de Red: recibe y envía las tramas.
- Capa de Transporte: controla el flujo de comunicaciones.
- Capa de Sesión: sincroniza los equipos.
- Capa de Presentación: descodifica los datos recibidos.
- Capa de Aplicación: transfiere los archivos.

Modbus consiste en un protocolo de solicitud-respuesta entre dos dispositivos. Éste funciona con una relación de maestro-esclavo. El maestro es el que inicia un requerimiento de servicio y el esclavo es cualquier recurso de cómputo dedicado a responder a los requerimientos del maestro. La capa física del protocolo Modbus puede ser a través de RS232, RS422 o RS485.

Modbus es un protocolo de comunicaciones situado en los niveles 1, 2 y 7 del Modelo OSI, basado en la arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP), diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de comunicaciones estándar de facto en la industria, es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales.

Las principales razones por las cuales el uso de Modbus en el entorno industrial se ha impuesto a otros protocolos de comunicaciones son:

Se diseñó teniendo en cuenta su uso para aplicaciones industriales

Es público y gratuito

Es fácil de implementar y requiere poco desarrollo

Maneja bloques de datos sin suponer restricciones

Hay muchas variantes de protocolos Modbus, existen versiones del protocolo Modbus para el puerto serie, para Ethernet, y otros protocolos que soportan el conjunto de protocolos TCP/IP de Internet:

- Modbus RTU: es la implementación más común disponible para Modbus. Se utiliza en la comunicación serie y hace uso de una representación binaria compacta de los datos para el protocolo de comunicación. El formato RTU sigue a los comandos/datos con una suma de comprobación de redundancia cíclica (CRC) como un mecanismo de comprobación de errores para garantizar la fiabilidad de los datos. Un mensaje Modbus RTU debe transmitirse continuamente sin vacilaciones entre caracteres. Los mensajes Modbus son entramados (separados) por períodos inactivos (silenciosos).
- Modbus ASCII: se utiliza en la comunicación serie y hace uso de caracteres ASCII para el protocolo de comunicación. El formato ASCII utiliza un checksum de control de redundancia longitudinal (LRC). Los mensajes Modbus ASCII están entramados por los dos puntos principales (":") y la nueva línea (CR/LF).
- Modbus TCP/IP o Modbus TCP: se trata de una variante Modbus utilizada para comunicaciones a través de redes TCP/IP, conectándose a través del puerto 502. No requiere un cálculo de suma de verificación (checksum), ya que las capas inferiores ya proporcionan protección de checksum.
- Modbus sobre TCP/IP o Modbus sobre TCP o Modbus RTU/IP: esta es una variante de Modbus que difiere del Modbus TCP en que se incluye una suma de comprobación en la carga útil como en Modbus RTU.
- Modbus sobre UDP: Algunos han experimentado con el uso de Modbus sobre UDP en redes IP, lo que elimina los gastos generales necesarios para TCP.
- Modbus Plus (Modbus+, MB+ o MBP): es una versión extendida del protocolo y privativa de Schneider Electric y a diferencia de las otras variantes, soporta comunicaciones peer-to-peer entre múltiples masters. Requiere un co-procesador dedicado para manejar HDLC. Utiliza par trenzado a 1 Mbit/s y sus especificaciones son muy semejantes al estándar EIA/RS-485 aunque no guarda compatibilidad con este, e incluye transformador de aislamiento en cada nodo. Se requiere hardware especial para conectar Modbus Plus a un ordenador, normalmente una tarjeta diseñada para bus ISA, PCI o PCMCIA.

- Pemex Modbus: esta es una extensión de Modbus estándar con soporte para datos
- Históricos y de flujo. Fue diseñado para la compañía petrolera Pemex para su uso en el control de procesos y nunca alcanzó un uso generalizado.
- Enron Modbus: esta es otra extensión del estándar Modbus desarrollada por Enron Corporation con soporte para variables enteras de 32 bits y de punto flotante y datos históricos y de flujo. Los tipos de datos se asignan utilizando direcciones estándar. Los datos históricos cumplen con un estándar de la industria del American Petroleum Institute (API, por sus siglas en inglés) según la forma en que deben almacenarse los datos.

El modelo de datos en Modbus distingue entre entradas digitales (discrete input), salidas digitales (coils), registros de entrada (input register) y registros de retención (holding registers). Las entradas y salidas digitales ocupan, evidentemente, un bit; mientras que los registros, tanto de entrada como de retención, ocupan dos Bytes.

El objetivo de la comunicación vía RS485 va a ser a través de Modbus RTU, las más simple, usada y estandarizada. Abarcaremos ASCII para ver las diferencias. La codificación de datos dentro de la trama puede hacerse en modo ASCII o en modo puramente binario, según el estándar RTU (Remote Transmisión Unit). En redes, una trama es una unidad de envío de datos. Es una serie sucesiva de bits, organizados en forma cíclica, que transportan información y que permiten en la recepción extraer esta información. Viene a ser el equivalente de paquete de datos o Paquete de red, en el Nivel de red del modelo OSI.

En cualquiera de los dos casos (RTU-ASCII), cada mensaje obedece a una trama que contiene cuatro campos principales, según se muestra en la figura. La única diferencia está en que la trama ASCII incluye un carácter de encabezamiento y los caracteres CR y LF al final del mensaje. Pueden existir también diferencias a la hora de calcular el CRC, puesto que el formato RTU emplea una fórmula polinómica en vez de la simple suma en módulo 16. A continuación vemos las tramas para ASCII y RTU:

(3AH)	Nº Esclavo	Código de operación	Subfunciones, Datos	LRC(16)	CR	LF

Codificación ASCII

Nº Esclavo	Código de operación	Subfunciones, Datos	CRC (16)

Codificación RTU

Imagen 8. Trama ASCII y RTU

A continuación, se detallan brevemente los campos:

- Número de esclavo (1 byte): permite direccionar un máximo de 63 esclavos con direcciones que van del 01H hasta 3FH. El número 00H se reserva para los mensajes difundidos. Las direcciones están en Hexadecimal.
- Código de operación o función (1 byte): cada función permite transmitir datos u órdenes al esclavo. Existen dos tipos básicos de órdenes:
- Ordenes de lectura/escritura de datos en los registros o en la memoria del esclavo.
- Ordenes de control del esclavo y el propio sistema de comunicaciones (RUN/STOP, carga y descarga de programas, verificación de contadores de intercambio, etc.)

- Campo de subfunciones/datos (n bytes): este campo suele contener, en primer lugar, los parámetros necesarios para ejecutar la función indicada por el byte anterior. Estos parámetros podrán ser códigos de subfunciones en el caso de órdenes de control (función 00H) o direcciones del primer bit o byte, número de bits o palabras a leer o escribir, valor del bit o palabra en caso de escritura, etc.
- Palabra de control de errores (2 bytes): en código ASCII, esta palabra es simplemente la suma de comprobación (checksum) del mensaje en módulo 16 expresado en ASCII. En el caso de codificación RTU el CRC se calcula según una fórmula polinómica según un algoritmo.

A nivel general de buses de campo, el nivel de aplicación de MODBUS no está cubierto por un software estándar, sino que cada fabricante suele suministrar programas para controlar su propia red. No obstante, el nivel de concreción en la definición de las funciones permite al usuario la confección de software propio para gestionar cualquier red, incluso con protocolos de distintos fabricantes. A continuación, se detallan las funciones según el código:

Función	Código (hexadecimal)	Tarea
0	00	Control de estaciones esclavas
1	01	Lectura de n bits de salida o internos
2	02	Lectura de n bits de entradas
3	03	Lectura de n palabras de salidas o internos
4	04	Lectura de n palabras de entradas
5	05	Escritura de un bit
6	06	Escritura de una palabra
7	07	Lectura rápida de 8 bits
8	08	Control de contadores número 1 a 8
9	09	No utilizado
10	0A	No utilizado

11	0B	Control de contadores número 9
12	0C	No utilizado
13	0D	No utilizado
14	0E	No utilizado
15	0F	Escritura de n bits
16	10	Escritura de n palabras

Tabla 2. Funciones Modbus

Modbus TCP / IP es un protocolo Modbus simple que se ejecuta en Ethernet a través de una interfaz TCP. Modbus es un protocolo de aplicación que asigna las formas de administrar y pasar datos entre varias capas sin verse afectado por el protocolo utilizado por la siguiente capa inmediata.

Utiliza el Protocolo de control de transmisión y el Protocolo de Internet para la transmisión de mensajes desde Modbus entre dispositivos compatibles a través de varios sistemas. Es decir, utiliza una red física (Ethernet), con un estándar de red (TCP / IP), y ofrece un método de representación de datos (Modbus como protocolo de aplicación). Un mensaje es básicamente un mensaje de datos de comunicación Modbus comprimido en una cubierta Ethernet TCP / IP, que es simplemente un protocolo de capa de transporte y no cambia la forma en que se almacenan o interpretan los datos dentro del mensaje.

A.2) Introducción a Arduino

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas, pero igualmente funcionales al partir de la misma base.

El software libre son los programas informáticos cuyo código es accesible por cualquiera para que quien quiera pueda utilizarlo y modificarlo.

Arduino ofrece la plataforma Arduino IDE (Entorno de Desarrollo Integrado), que es un entorno de programación con el que cualquiera puede crear aplicaciones para las placas Arduino, de manera que se les puede dar todo tipo de utilidades.

El IDE se compone de un editor de texto, un compilador, un debug o depurador y un GUI (constructor de interfaz gráfica) en un mismo entorno de

programación, además de poder incluir herramientas para poder cargar un programa ya compilado, como es nuestro caso.

Arduino trabaja con ficheros con la extensión “ino”, aunque es necesario que el programa principal este dentro de una carpeta con el mismo nombre, se pueden añadir programas secundarios donde incluir llamadas del programa principal.

Al tratarse de un software “OpenSource” facilita el auto aprendizaje ya que es posible encontrar por internet gran cantidad de ejemplos de diferentes aplicaciones compartidas por otros usuarios, además de facilitar la localización de un gran abanico de shields de sensores compatibles con Arduino.

El Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

También cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que reproducir los datos procesados, pero también pueden ser otras placas o controladores.

Debido a que Arduino es una de las primeras plataformas microcontroladoras del mundo OpenSource se han desarrollado varias versiones sobre la aplicación. A continuación, vemos gráficamente algunos:

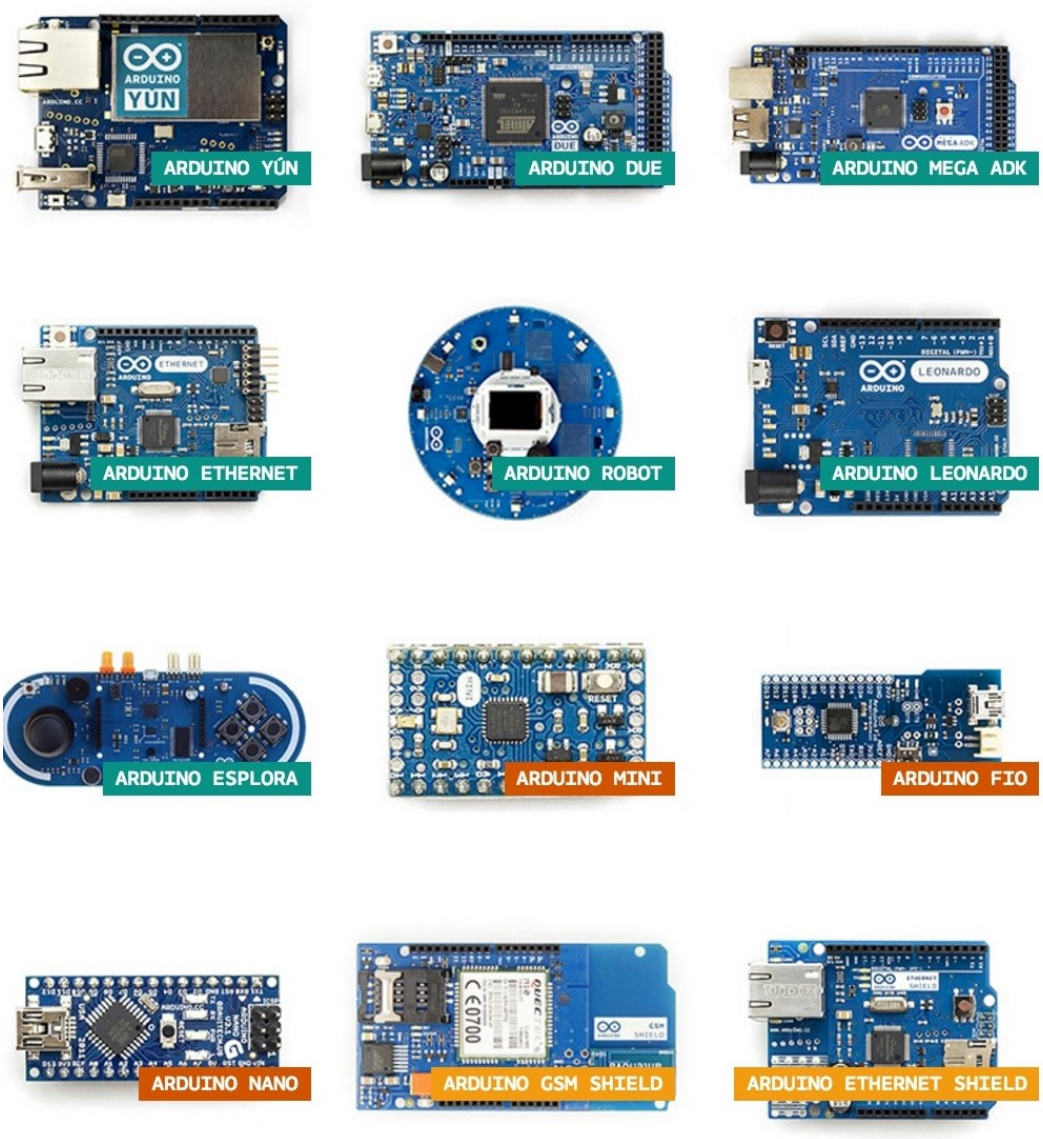


Imagen 9. Algunos tipos de Arduino

Cuando se decide utilizar un Arduino para un proyecto, lo primero que se observa son las características que ofrece cada uno. En la página de Arduino tenemos una tabla comparativa: “https://www.arduino.cc/en/products/compare”. Se procede a conocer los principales puntos a tener en cuenta para luego poder comparar los recursos del Arduino adquirido y los que se usan en el mercado para el proyecto:

-Tipo de procesador, UART.

-Voltaje de operación/entrada-Corrientes de trabajo.

-Velocidad de CPU.

-Pines analógicos de entrada/salida, digitales entrada-salida/PWM, USB.

-Memoria EEPROM-SRAM-FLASH en [kB].

A continuación, se verá una imagen del Arduino Uno y sus partes, para tener una noción de cómo se distribuye en el PCB:

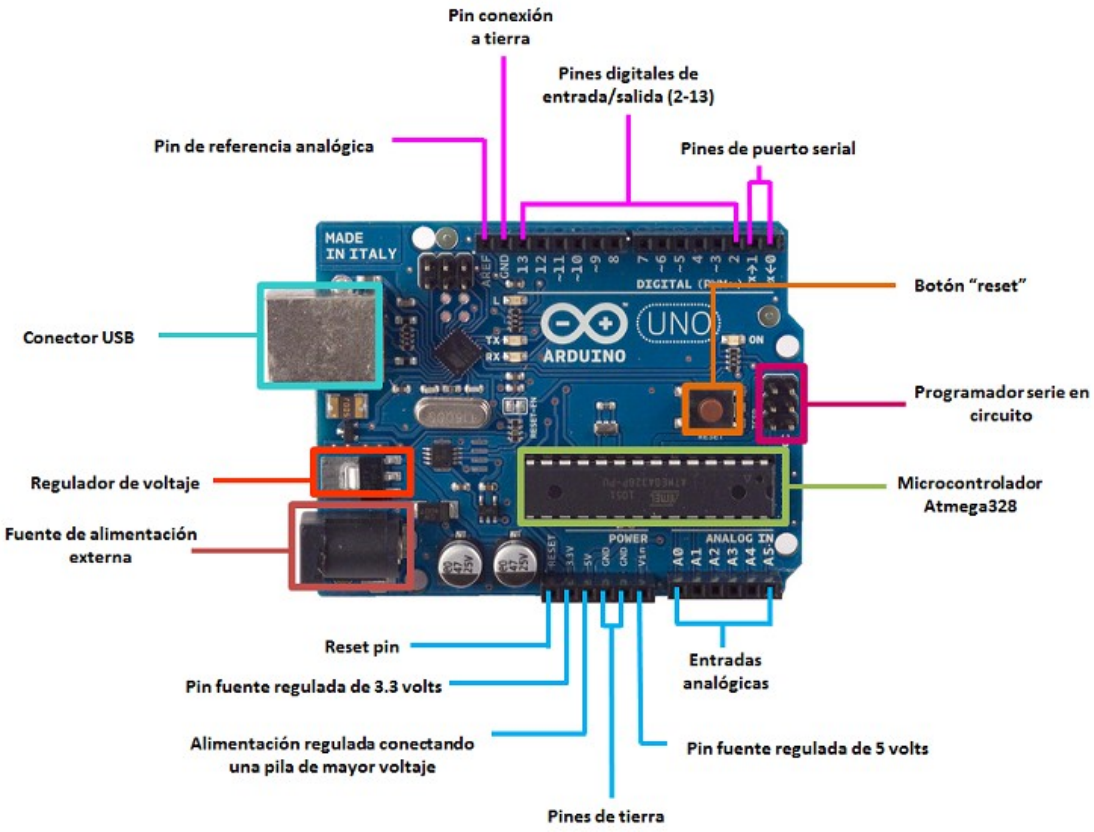


Imagen 10. Arduino Uno-partes

Selección de componentes: Arduino Due

El Due es una placa de microcontrolador basada principalmente en Atmel SAM3X8E ARM Cortex-M3 (microcontrolador ARM de 32 bits). Es un controlador grande en comparación con el de Arduino Uno o Nano y viene con más número de pines y espacio de memoria en comparación con ellos. Se enunciarán algunos datos técnicos:

Contiene 54 digitales que pueden funcionar en ambos sentidos: entrada o salida. De estos pines digitales, 12 pueden usarse para generar salidas PWM.
Este módulo contiene todo lo necesario para el proyecto de automatización, incluidas 12 entradas analógicas y 4 módulos serie UART.
Viene con una frecuencia de reloj de aproximadamente 84 MHz, lo que ayudará a que la velocidad de procesamiento crezca disparándose.
Hay dos puertos USB disponibles donde uno se usa para programación mientras que otro es nativo. Ambos puertos se pueden usar para fines de programación, sin embargo, el nativo también actúa como un host USB para periféricos conectados como teclados y teléfonos inteligentes.

La mayoría de las placas Arduino se ejecutan a 5V, pero este módulo es una excepción que funciona a 3.3V. Los pines incorporados en el no pueden soportar una tensión superior a esta. Hacerlo puede afectar drásticamente el rendimiento de la placa y puede hacer que sus pines queden nulos.
JTAG se agrega a la placa que se usa principalmente para probar la conexión física entre los pines integrados.
El Arduino Due se puede programar utilizando un software común de Arduino (IDE), que es compatible con todas las placas Arduino y puede funcionar en ambos sentidos: en línea y fuera de línea.
Este módulo incorpora 2 DAC (digital a analógico), 2 TWI (interfaz de dos cables) incorporados en la placa, también conocidos como protocolo I2C, un conector de alimentación (puede encender el dispositivo conectándolo con una computadora a través del cable USB o usando este conector de alimentación), un botón de reinicio del encabezado SPI, un botón de borrado y Botón de reinicio. Un montón de funciones, haciendo su tarea fácil.
Según la restricción de voltaje, los Shields Arduino que operan a 5V no son compatibles con este módulo Due. Sin embargo, los escudos que vienen con el diseño de Arduino R3 funcionan de manera eficiente, incluyendo el escudo WiFi de Arduino y el escudo Ethernet, ya que funcionan a 3.3V.
No tiene EEPROM. Usar memoria flash sería una solución gracias al segundo puerto USB, pero habría que analizar si hay librerías que lo posibiliten la escritura.
Tiene Flash de 512 KB y SRAM de 96 KB en dos bancos de 64-32.
Tenemos un Arduino de 32 bits pensando en un hardware capaz de dar soluciones a proyectos cada vez más serios.
Corriente DC para 3.3V de 800mA y de salida total de CC en todas las líneas E/S de 130mA. 3.3V es el voltaje de operación de cada pin, mientras que Vin es el voltaje de entrada con el rango de voltaje recomendado de 7V a 12V. Puede alimentar el controlador con Vin o 5V.

Programación y comunicación

Se puede programar mediante Arduino IDE. Es fácil de usar y una persona sin un conocimiento técnico previo puede aprender el software sin mucha dificultad. No se requiere un quemador externo para grabar el código en el controlador. El software funciona perfectamente con sistemas operativos como Windows, Linux o MAC. Viene con muchos protocolos para comunicarse con dispositivos externos. El UART es útil para configurar una comunicación en serie. Hay cuatro de ellos, brindando la flexibilidad de establecer una comunicación en serie con más de un dispositivo. La interfaz de dos cables también se incluye en el dispositivo que viene con dos líneas SDA y SCL. Hay dos canales TWI disponibles en el tablero. Arduino Software Wire Library se utiliza para acceder al bus TWI.

Arduino Due viene con una interfaz periférica en serie (SPI) que desempeña un papel vital en la comunicación entre el microcontrolador y otros dispositivos periféricos, como los registros de desplazamiento y los sensores. Hay dos pines utilizados para la comunicación SPI, es decir, MOSI (Master Slave Output Output) y MISO (Master Input Slave Output). El primero se utiliza para recibir los datos, mientras que más tarde ayuda a enviar los datos por el microcontrolador.

A continuación, una imagen de la placa con algunas indicaciones:

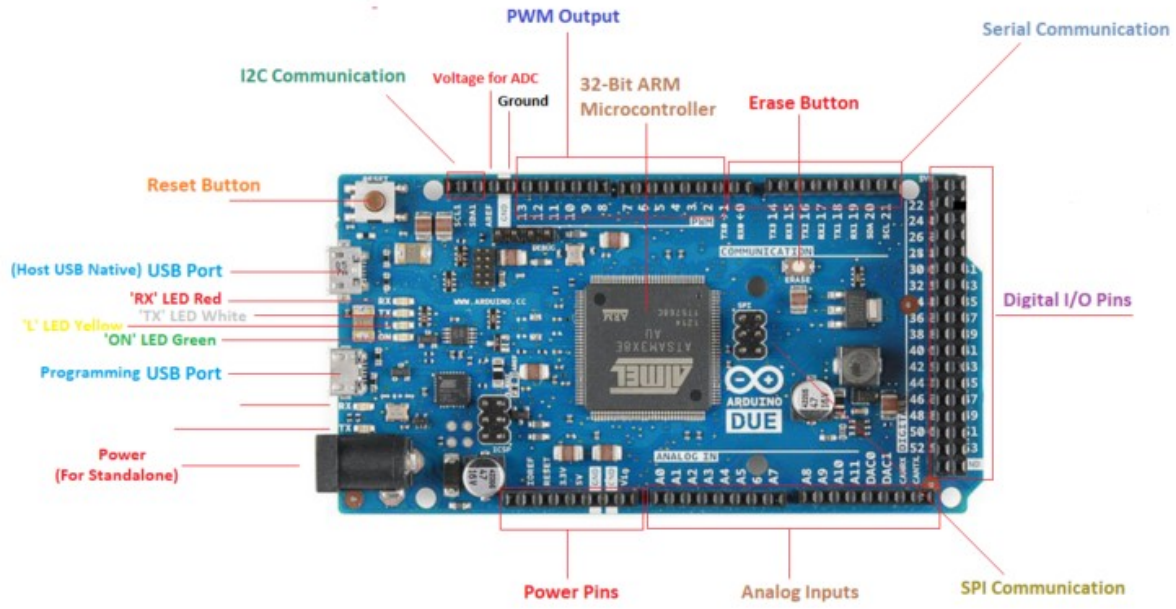


Imagen 11. Arduino Due-partes

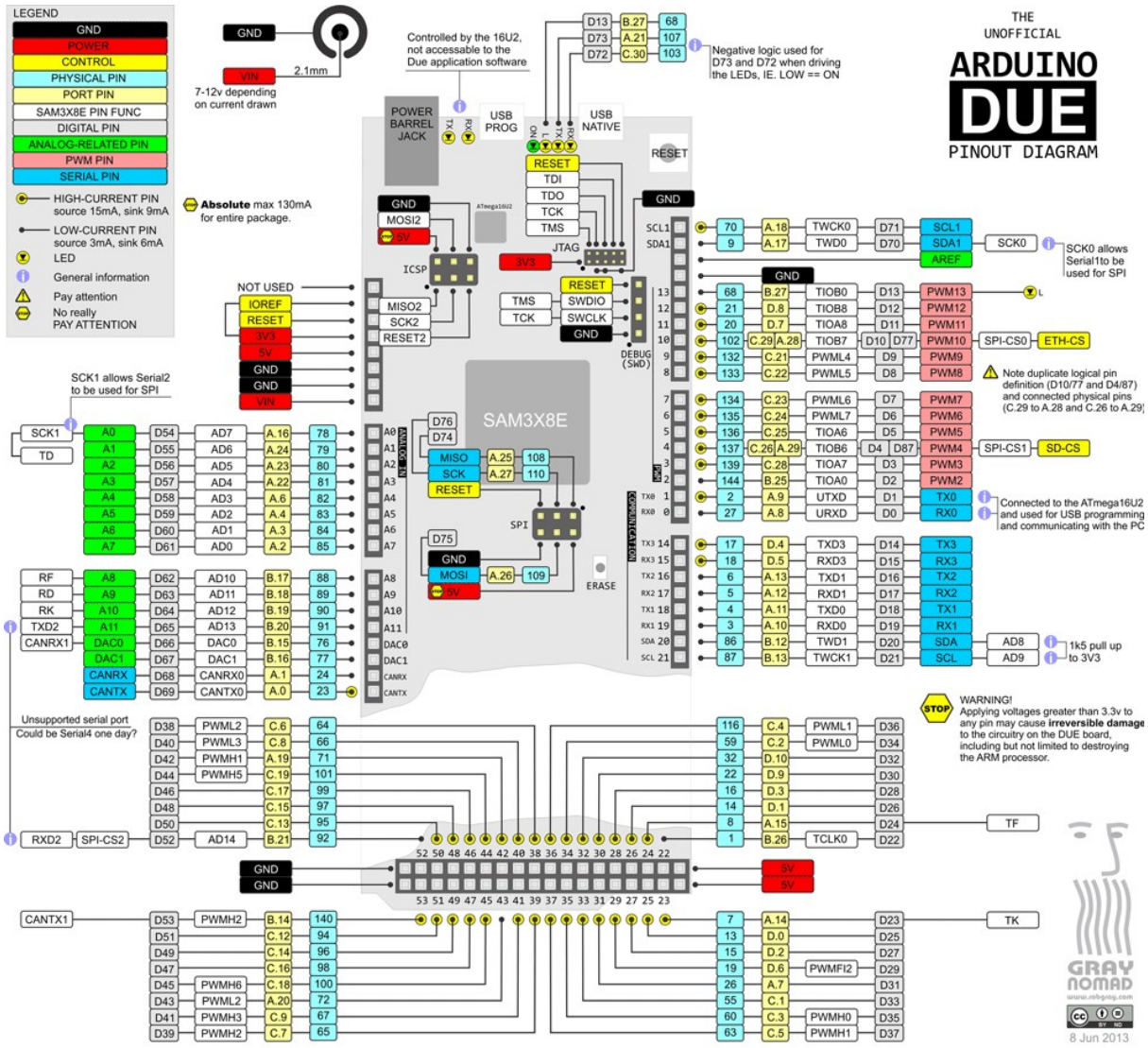


Imagen 12. Arduino Due-Esquema de Placa

Existen dos puertos USB disponibles en la placa de los cuales hablamos previamente. Hay un conector de alimentación en la placa para encender el dispositivo.

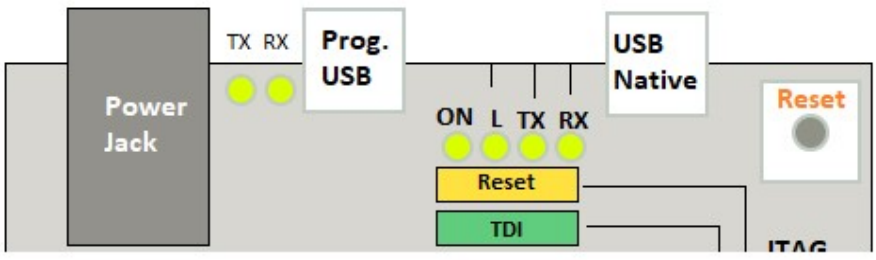


Imagen 13. Arduino Due-Puertos USB

Arduino Due viene con cuatro puertos llamados PORTA, PORTB, PORTC y PORTD. Hay 54 pines de E / S digitales. Los pines con color rosa se utilizan como pines de E / S digitales.

Hay 12 pines analógicos que son parte de PORTA y PORTB y aparecen de la siguiente manera:

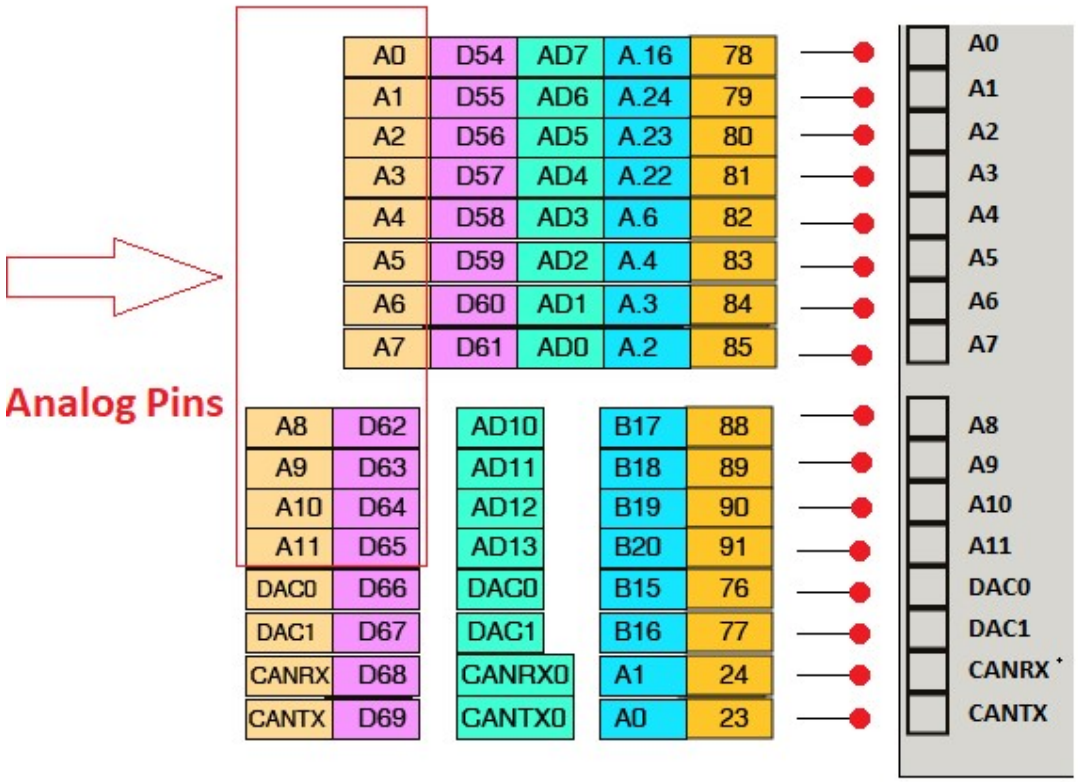


Imagen 14. Arduino Due-Pines Analógicos

Presenta cuatro canales UART añadidos en el tablero. Se utilizan para la comunicación en serie con los dispositivos externos donde TX es el Pin de transmisión en serie, mientras que RX es el Pin de recepción en serie. Aparecen en la siguiente imagen:

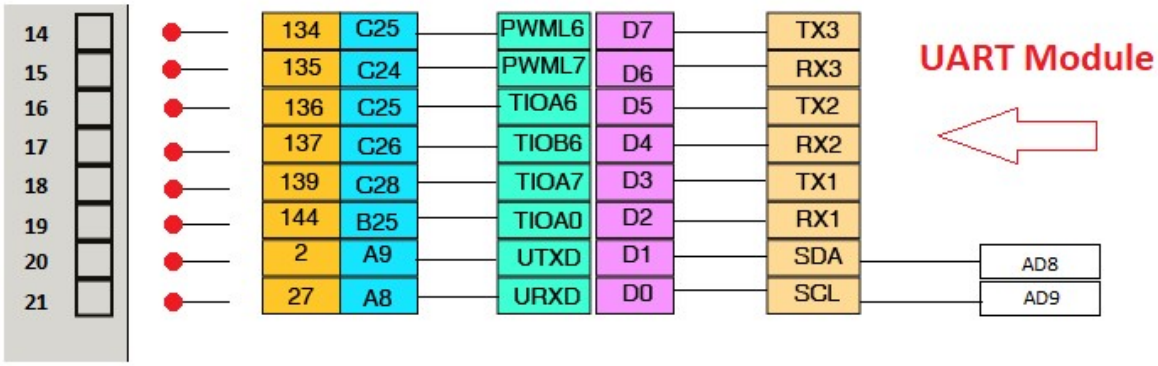


Imagen 15. Arduino Due-Canales UART

Contiene dos módulos TWI (interfaz de dos cables) incorporados en la placa, también conocidos como protocolo I2C, y se utilizan para establecer la comunicación entre dispositivos de baja velocidad como los conversores y microcontroladores ADC y DAC. Es una comunicación a dos hilos y viene con dos líneas: Serial Clock (SCL) y Serial Data (SDA). La primera es una señal de reloj que se utiliza para sincronizar la transferencia de datos entre los dispositivos, mientras que la última se utiliza para mantener los datos deseados. A continuación, lo vemos en la imagen:

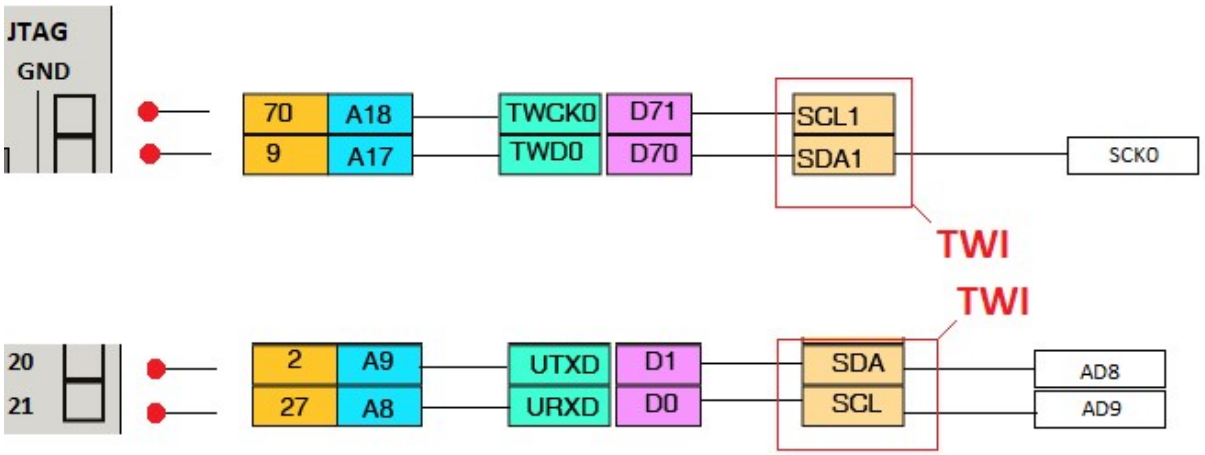


Imagen 16. Arduino Due-Puertos TWI

El encabezado JTAG es una interfaz de hardware común que establece un camino para comunicarse directamente con chips externos en una placa. Fue introducido por el Grupo de Acceso de Prueba Conjunto (europeo) con la intención de probar las conexiones físicas entre los pines en un chip. Para ello se cuenta con TCK, TMS, TDI y TDO. Se puede observar en la siguiente imagen:

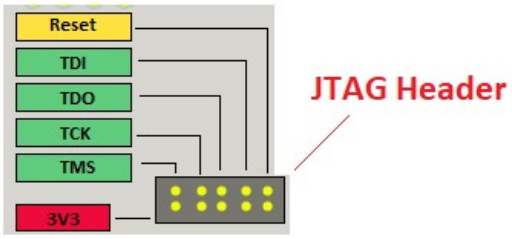


Imagen 17. Arduino Due-JTAG Header

Presenta cuatro fuentes de alimentación mencionadas como 5V, 3.3V, Vin y Ground, de las cuales hablamos previamente. Se procede a ilustrarlas:

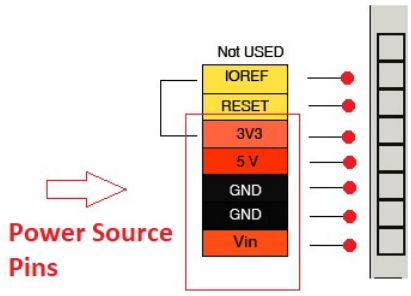


Imagen 18. Arduino Due-Pines de Alimentación

De los 54 pines digitales, 12 se utilizan para la salida PWM. Aparecen en la de la siguiente manera.

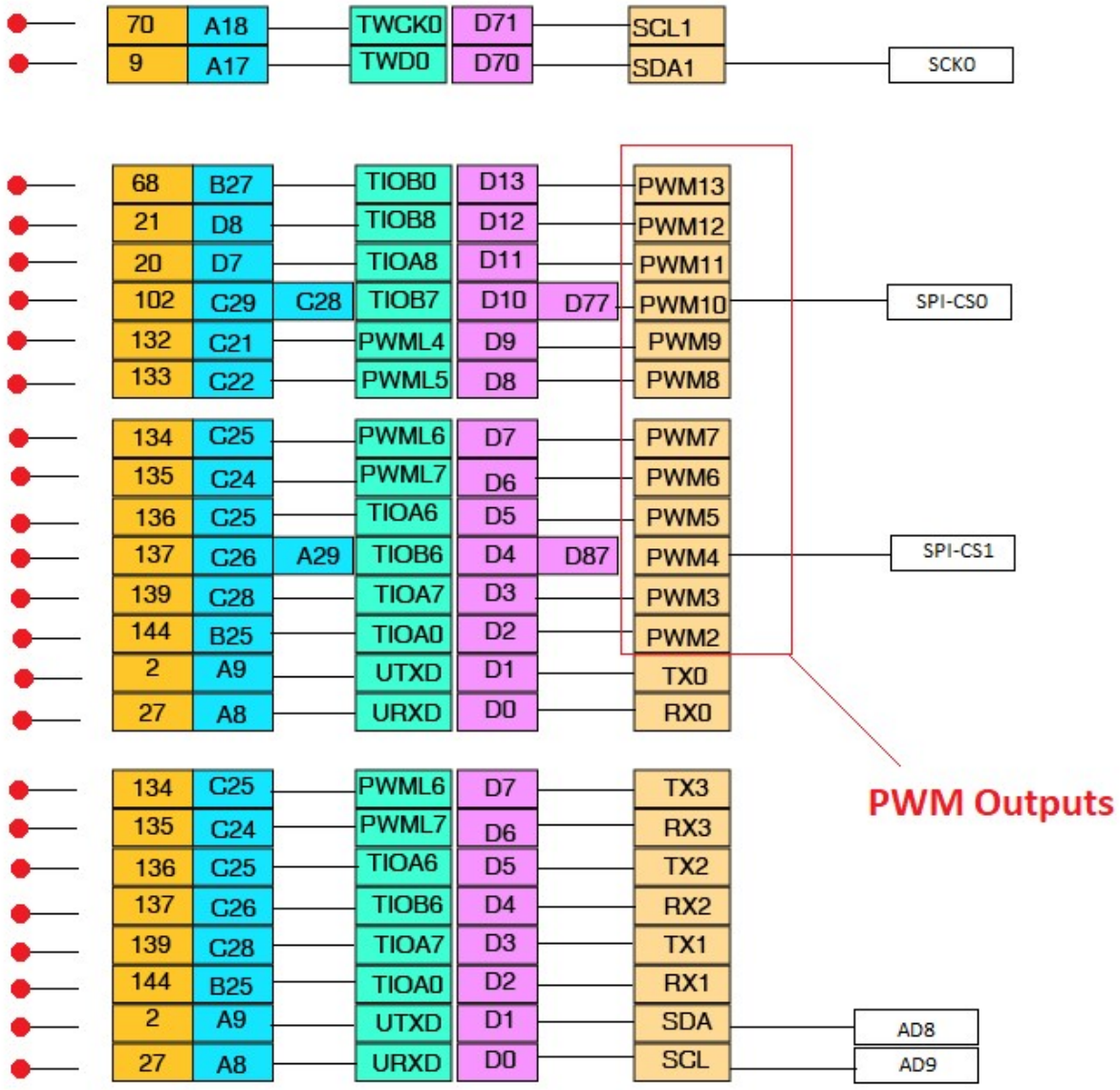


Imagen 19. Arduino Due-Pines Digitales-PWM

Cuenta con un botón para el reinicio del controlador y también se agrega un botón de borrado de la información almacenada en la placa en el dispositivo.

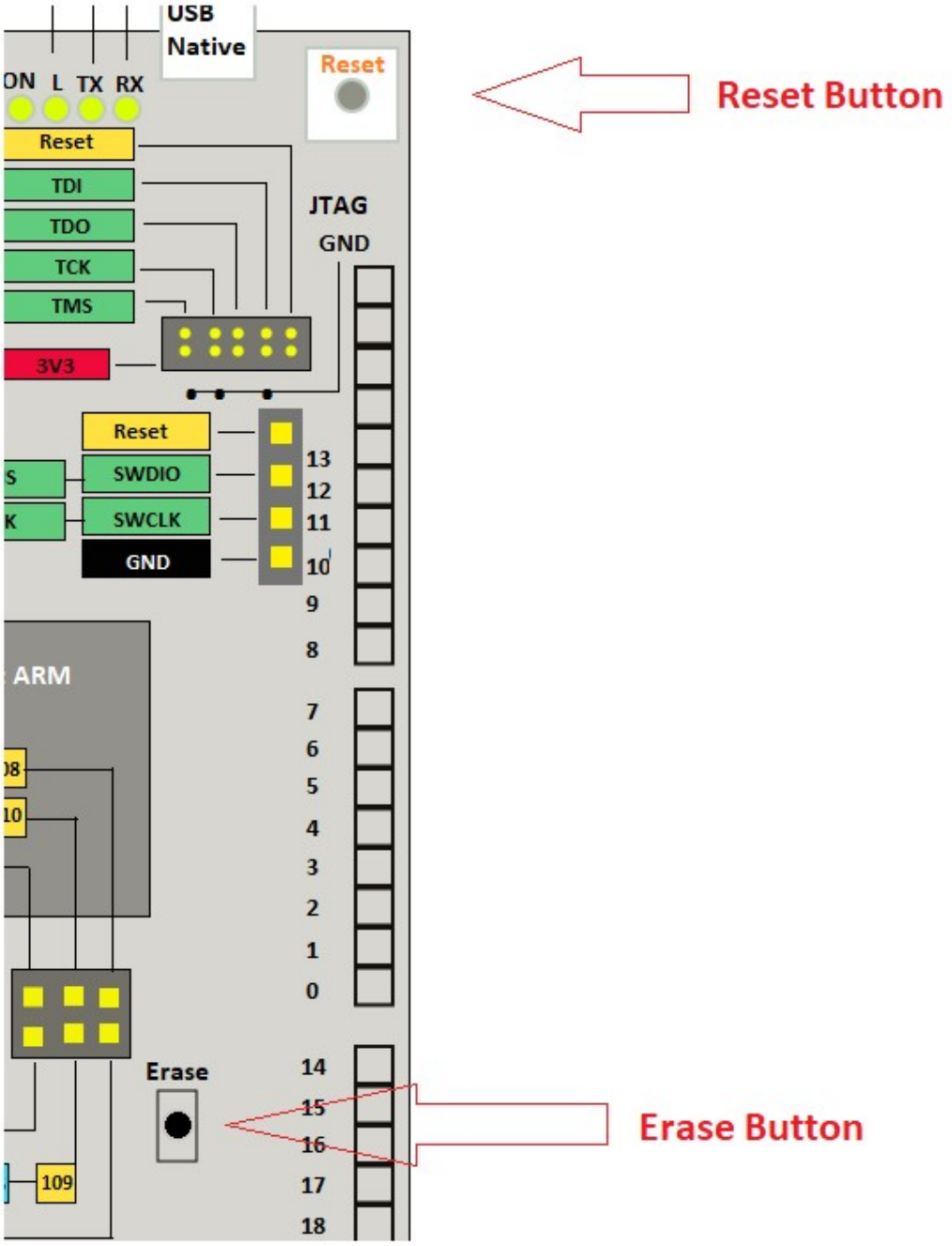


Imagen 20. Arduino Due-Botones de Reinicio y Borrado

Además de todo lo nombrado, se mostrará una imagen de los puertos CAN.

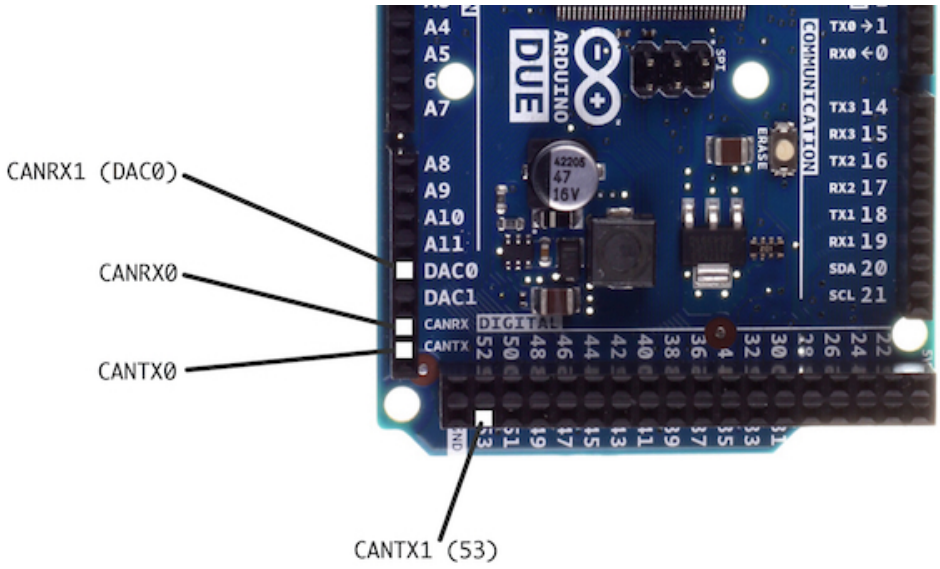


Imagen 21. Arduino Due-Puerto CAN

Puertos Serie UART

Los puertos serie son la forma principal de comunicar una placa Arduino con un ordenador. Gracias al puerto serie se puede, por ejemplo, mover el ratón o simular la escritura de un usuario en el teclado.

También es el nombre genérico con que se denomina a los interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos.

Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). No obstante, pueden existir otros conductores para referencia de tensión, sincronismo de reloj, etc.

Por el contrario, un puerto paralelo envía la información mediante múltiples canales de forma simultánea. Para ello necesita un número superior de conductores de comunicación, que varían en función del tipo de puerto. Igualmente existe la posibilidad de conductores adicionales además de los de comunicación.

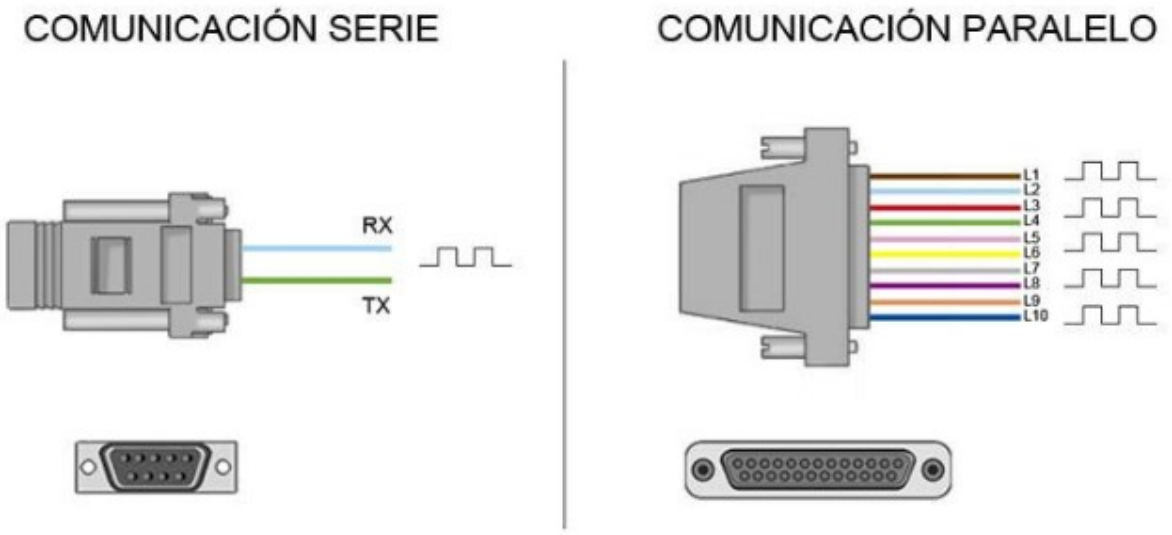


Imagen 22. Puerto Serie-Paralelo

Prácticamente todas las placas Arduino disponen al menos de una unidad UART. Las placas Arduino UNO y Mini Pro disponen de una unidad UART que operan a nivel TTL 0V / 5V, por lo que son directamente compatibles con la conexión USB. Por su parte, Arduino Mega y Arduino Due disponen de 4 unidades UART TTL 0V / 5V.

Redes CAN

Arduino Due viene con los pines CANTX y CANRX para el desarrollo de una red de área de controlador (CAN) y existen algunas librerías disponibles para su programación.

El protocolo de comunicaciones CAN proporciona los siguientes beneficios:

Ofrece alta inmunidad a las interferencias, habilidad para el autodiagnóstico y la reparación de errores de datos.
Es un protocolo de comunicaciones normalizado, con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común o bus.
El procesador anfitrión (*host*) delega la carga de comunicaciones a un periférico inteligente, por lo tanto el procesador anfitrión dispone de mayor tiempo para ejecutar sus propias tareas.
Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto, excepto en los enganches.

CAN (Controller Area Network) y RS485 son estándares populares en sistemas de bus de campo. Como RS485 solo aborda la capa 1 (la capa física) y

CAN también agrega la capa 2 (la capa de enlace de datos) en el modelo OSI, es difícil comparar los dos estándares.

El estándar RS485 se creó en 1983 y ofrece una capacidad de comunicación multipunto para dispositivos con al menos un UART. Cualquier dispositivo conectado al bus RS485 puede comunicarse con cualquier otro dispositivo en modo semidúplex. Aunque la mitad dúplex permite una comunicación bidireccional, solo un dispositivo puede transmitir a la vez y los otros dispositivos tienen que escuchar.

En este caso, RS485 no es un protocolo, solo proporciona las reglas básicas y el enlace físico para el intercambio de datos, lo que permite la transmisión de mensajes serie utilizando un bus multipunto, mientras que el contenido del mensaje es completamente definido por el usuario. Esto también significa que, por ejemplo, la estructura del marco de comunicación, el esquema para direccionar otros nodos, los mecanismos para evitar colisiones de datos y otras tareas deben ser implementadas por el desarrollador en forma de un software de protocolo.

La red de área del controlador (CAN) fue creada en la década de 1980 por Robert Bosch GmbH. Al principio, CAN se dirigía principalmente a la industria automotriz, pero hoy en día se utiliza en una variedad de aplicaciones, como automatización industrial, médica, transporte, etc.

Comparado con RS485, CAN no solo proporciona los medios físicos para la comunicación, sino que también proporciona todos los demás mecanismos necesarios para direccionar paquetes de datos (mensajes), evitar colisiones de datos, detectar fallas en los datos transmitidos, repetición automática de mensajes perturbados y garantizar la consistencia de los datos. sobre todos los nodos en una red. Además, CAN especifica la estructura de la trama de datos, con identificador de mensaje, datos y bytes de control.

Al igual que RS485, un nodo CAN puede comunicarse con cualquier otro nodo en modo semidúplex. Un mensaje consta principalmente de un identificador (ID), que representa la prioridad del mensaje con respecto a la prevención de colisiones y hasta ocho bytes de datos. Se transmite en serie por el bus. Este patrón de señal está codificado en non-return-to-zero (NRZ) y es detectado por todos los nodos.

Si el bus está libre, cualquier nodo puede comenzar a transmitir. Si dos o más nodos comienzan a enviar mensajes al mismo tiempo, el mensaje con la prioridad más alta, como lo define la ID más dominante (que tiene más bits dominantes, es decir, ceros), sobrescribirá las ID menos dominantes de otros nodos. Esto asegura que solo los demás nodos transmitan y reciban el mensaje dominante. Este mecanismo se conoce como arbitraje de bus basado en la prioridad. Los mensajes con valores numéricamente más pequeños de ID tienen

mayor prioridad y se transmiten primero en caso de que dos mensajes se envíen simultáneamente.

Finalmente veremos una tabla comparativa:

Característica	RS48	PUEDE
Interfaz requerida	UART	Controlador CAN
Capas de modelo ISO soportadas	Capa fisica	Capa física y capa de enlace de datos
Detección de colisiones de datos.	No se ha implementado	Sí, CSMA / CR
Velocidad de transmisión máxima	10 Mbit / s (hasta 12 m)	1 Mbit / s (hasta 50 m)
Longitud máxima del autobús	1200 m (a 100 kbit / s)	1600 m (a 50 kbit / s)
Principios de arbitraje de bus soportados	Maestro / esclavo o Token Ring	Multimaster y todos los principios, que pueden derivarse de eso como Master / Slave o Token Ring
Cantidad máxima de datos por cuadro	Ilimitado	8 bytes
Ejemplos de protocolos populares.	Modbus RS485, Profibus	CANopen, DeviceNet, J1939

Tabla 3. Comparación CAN-RS485

Comunicación por bus I2C

Philips Semiconductors (ahora NXP Semiconductors) desarrolló un simple bus bidireccional de 2 hilos para un control eficiente entre CI. Este bus se llama Inter-IC o I2C-bus. Solo se requieren dos líneas de bus: una línea de datos en serie (SDA) y una línea de reloj en serie (SCL). Se pueden realizar transferencias de datos bidireccionales en serie orientadas a 8 bits a hasta 100 kbit / s en modo Estándar, hasta 400 kbit / s en modo rápido, hasta 1 Mbit / s en modo rápido Plus (Fm +), o hasta 3,4 Mbit / s en el modo de Alta velocidad. El modo Ultra Fast es un modo unidireccional con transferencias de datos de hasta 5 Mbit / s.

I2C para Arduino admite 127 dispositivos conectados a través de los pines (SDA) y (SCL). Los pines I2C pueden variar en diferentes placas Arduino. I2C no solo se utiliza en placas individuales, sino también para conectar componentes que están conectados mediante un cable. La simplicidad y la flexibilidad son características clave que hacen que este bus sea atractivo para muchas aplicaciones.

Las características más significativas incluyen:

Sólo se requieren dos líneas de autobús. Sin requisitos estrictos de velocidad de transmisión como, por ejemplo, con RS232, el maestro genera un reloj de bus.

Existen relaciones simples de maestro / esclavo entre todos los componentes. Cada dispositivo conectado al bus es direccionable por software por una dirección única.

I2C es un verdadero bus multi-maestro que proporciona arbitraje y detección de colisiones

Vemos un ejemplo de conexionado básico:

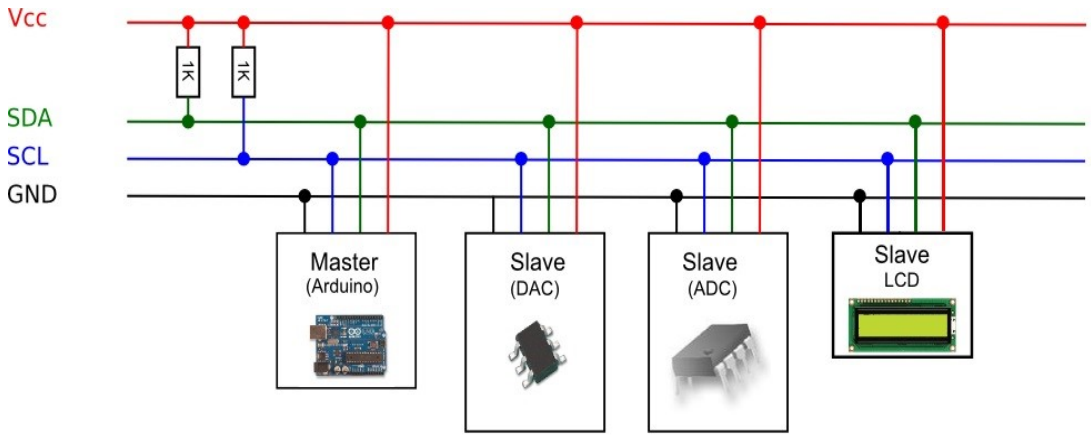


Imagen 23. Ejemplo de conexionado por I2C

La idea es que todos los componentes se conecten en paralelo a las dos líneas del Bus, SDA y SCL. Respecto a la imagen podemos hacer algunos comentarios:

Puede haber más de un master. La norma propone un sistema de arbitraje, para transferir el control de uno a otro, pero en un instante dado, solo uno puede ser el master.
Hay unas resistencias de Pullup conectadas a SDA y SCL. Son imperativas, ya que el bus es activo bajo. Esto es, la señal activa es un 0, no un 1. Al conectar algo al busI2C, es imprescindible que saber si los Pullups hay que agregarlos, o vienen puestos en el componente. En el caso del display I2C, normalmente incluyen los Pullups.

Este es un puerto muy útil para aplicaciones que un usuario desee y en el presente caso, donde se va a agregar un display, claramente vemos su utilidad.

Comunicación por bus SPI

El SPI es un protocolo de comunicación síncrona de 4 hilos, entre dispositivos electrónicos presentado por Motorola en 1982, que ha ganado bastante aceptación en la industria como sistema de comunicación de muy corta distancia, normalmente dentro la placa de circuito impreso.

Es un protocolo de transmisión que permite alcanzar velocidades muy altas y que se diseñó pensando en comunicar un micro controlador con distintos periféricos y que funciona a full dúplex (enviar y recibir datos al mismo tiempo).

SPI utiliza una solución síncrona, porque utiliza unas líneas diferentes para los datos y el Clock. El Clock es una señal que indica al que escucha exactamente cuándo leer las líneas de datos, con lo que el problema de perdida de sincronía se elimina de raíz.

Por eso mismo, no se necesita pactar la velocidad de transmisión, ya que será el Clock quien fije la velocidad y puede ser variable a lo largo de la comunicación sin que sea un problema (Aunque por supuesto según el dispositivo habrá un límite de velocidad).

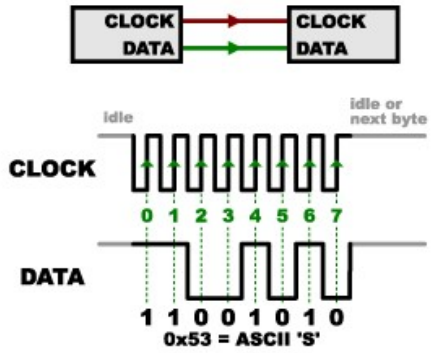


Imagen 24. Ejemplo de sincronismo SPI

Uno de los motivos por los que SPI es tan popular es que el hardware de recepción puede ser un sencillo Shift register, una forma sencilla de aumentar el número de salidas digitales de nuestros Arduinos sin demasiada complicación, como el 74HC595. Esto es una solución mucho más simple que una UART (Universal Asíncronous Receiver Transmitter o sistema universal asíncrono de recepción y transmisión serie) de comunicación serie.

En un bus SPI una de las partes genera el Clock al que llamamos master, y el resto son los esclavos, pudiendo haber uno o varios en el bus. A la señal de reloj se le suele llamar CLK por Clock o SCK por Serial Clock.

Cuando el master envía información, lo hace por una línea de datos que normalmente de nombre MOSI (Master Out Slave In) y si el esclavo responde lo hace a través de una línea llamada MISO (Master In Slave Out).

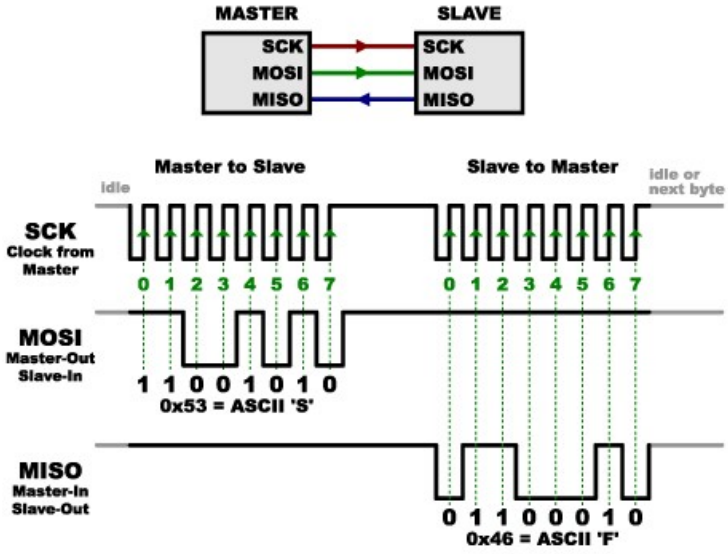


Imagen25. Funcionamiento de MOSI-MISO

Como es el master quien genera el Clock, necesita saber de antemano, si un esclavo va a devolver una respuesta y de que longitud, para mantener el Clock hasta que la transferencia esté completa. Esto no es normalmente un problema porque cuando el master pide a, digamos un sensor, que le envié una lectura, normalmente sabemos que van ser 2 bytes. Hay una última línea de control, llamada Slave Select o SS, que indica a un esclavo que el mensaje que viene es para él, o bien que se reclama que envié una respuesta a una petición del master.

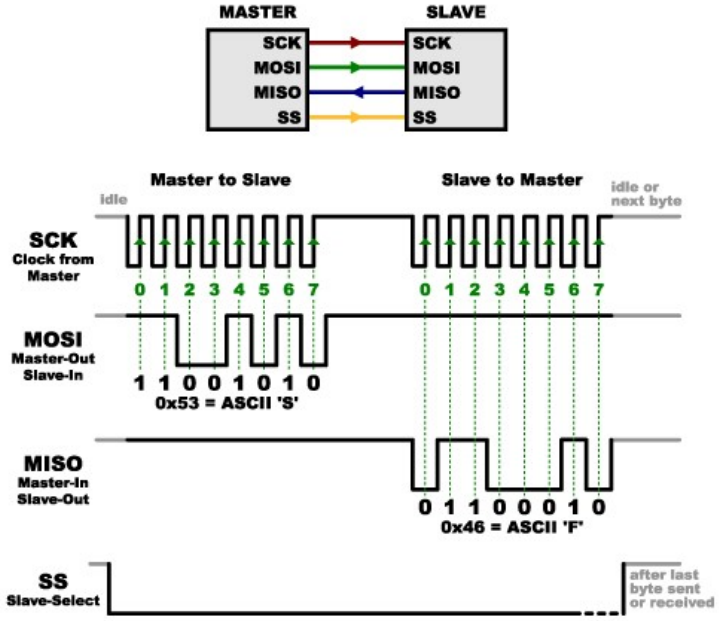


Imagen 26. Funcionamiento de MOSI-MISO junto a SS

La línea SS normalmente se mantiene HIGH y se activa con LOW, lo que despierta al esclavo seleccionado. Cuando se termina la transferencia la línea se levanta a HIGH y el esclavo se desactiva.

Hay dos maneras de conectar múltiples esclavos a un bus SPI. Con una línea SS por cada esclavo o en cascada.

En general cada esclavo requiere su propia línea SS para ser activado, y así evitar que dos hablen a la vez, porque el resultado sería ruido inútil. Basta con activar la línea correspondiente y el esclavo está listo para recibir sus órdenes

Es un sistema cómodo, siempre y cuando no sean muchos esclavos porque podemos necesitar muchas líneas.

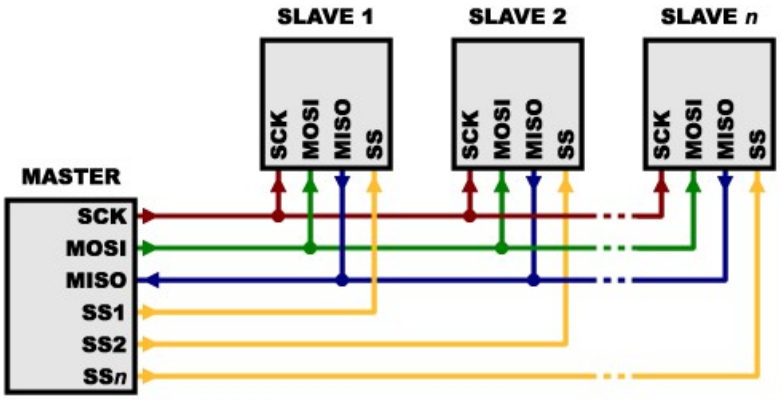


Imagen 27. Conexión de esclavos por SS

Cuando el número de esclavos crece suele ser más frecuente conectarlos en cascada, con el MISO de uno (Salida), conectado al MOSI (Entrada) del siguiente. En este caso solo usamos una única línea SS, que se comparte entre todos los esclavos.

Esta configuración es típica de una situación en la que le master envía datos, pero no recibe nada de vuelta, como en el caso de una cadena de múltiples display LED, en los que se envía información para ser mostrada, pero, no hay datos de vuelta. En este caso incluso podemos desconectar la línea MISO

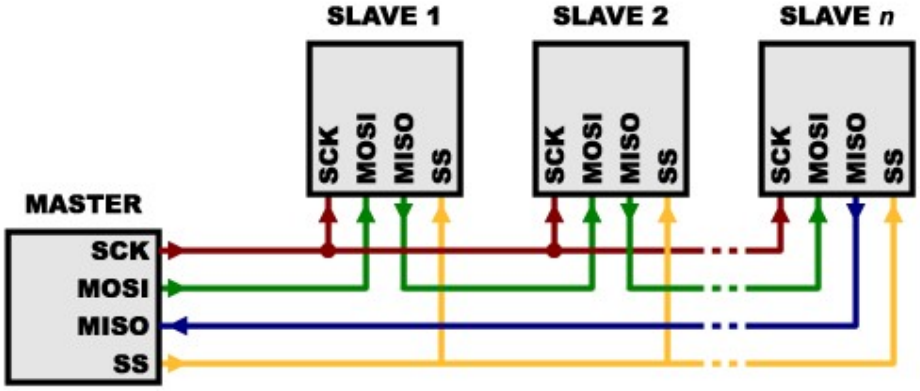


Imagen 28. Conexión de esclavos en cascada

Por último y para cerrar esta sesión, comentar que naturalmente Arduino soporta el bus SPI, con una librería estándar que sorprendentemente se llama SPI y gestiona todas las complicaciones y el arbitraje del protocolo.

A.3) Módulos para RS485 y Ethernet

A.3.1) Conversor TTL-RS485

Existen distintos productos para que el Arduino pueda entablar comunicaciones por RS485, pero para el proyecto se elige el de Waveshare debido a que cuenta con tres salidas distintas y la posibilidad de alimentarse con 3.3[V].

Los precios en comparación con un producto típico de 5V y una sola salida A-B son apenas menores. A continuación, se aprecia el dispositivo:

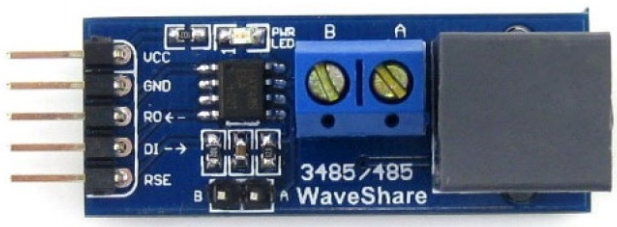


Imagen 29. Conversor TTL-RS485

La conexión es sumamente sencilla, tenemos 5 pines y las salidas A-B ya en RS485 para conectar a la maquina o dispositivo que se necesite. Inicialmente aparece la alimentación con VCC y GND. Luego tendremos el pin R0 y DI que se conectan a uno de los puertos serie, en Rx y Tx respectivamente. El último, RSE, se conecta a un PIN digital, y determina si se encuentra en modo emisión o recepción.

A.3.2) Ethernet Mini W5100

Para conectar a internet o Lan el Arduino se necesita un módulo más complejo que el RS485. Hay opciones un poco más económicas como el ENC28J60, y variables del W5100 como el W5500 y W5200 por dar ejemplos. Sin embargo, en este caso, no se trata solo buscar el más fiable y económico, sino que también pueda programarse sin problemas. En la imagen se visualiza el pinout. En Arduino Due se tienen los tres pines SPI (MOSI-SCK-MISO) y tres pines SS, un máximo de tres esclavos. El punto a observar es que solo uno de esos tres SS se especializa en ETH, es el pin 10. La alimentación es de 5V.

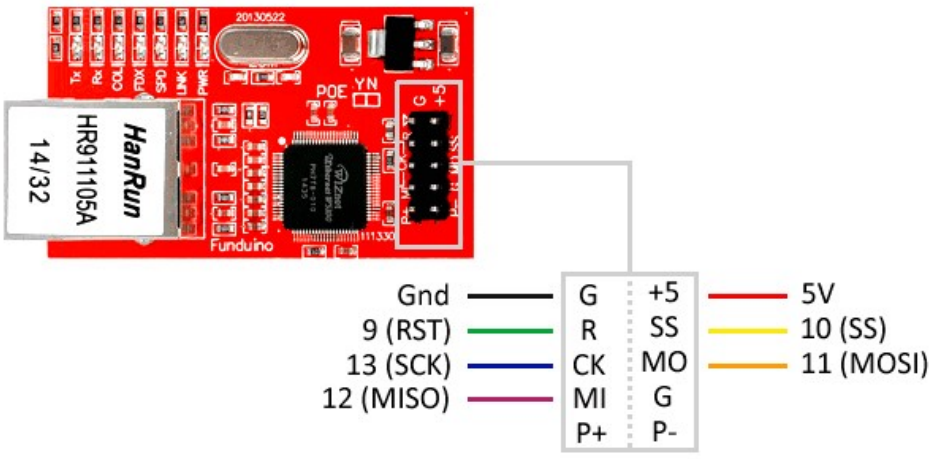


Imagen 30. Mini Ethernet Shield W5100 - Pines

A.3.3) Módulo SX1278 LoRA WiFi

El SX1278-433M está basado en el transceptor de RF SX1278 de SEMTECH. Refiere a un módulo wifi UHF de pequeñas dimensiones y ultra baja potencia. Es un producto económico diseñado específicamente con propósitos de aplicaciones wifi.
Este tiene, las siguientes características:

Alimentación: 1.8V a 3.7 V.
Rango de Frecuencia: 433MHz.
Potencia máxima de salida: 20 dBm.
Velocidad de transferencia de datos: @ FSK, 1.2-300kbps.
Lora TM, FSK, GFSK y el modo de modulación OOK.
Equipo de paquete de hasta 256 bytes con CCR.
Rango de temperatura: -40 ~ + 85 ° C.
Sensor de temperatura incorporado e indicador de batería baja.
Excelente inmunidad de bloqueo.

El alcance de este módulo puede llegar a los 5 kilómetros en una zona abierta. A continuación, se observa gráficamente la placa y sus pines:

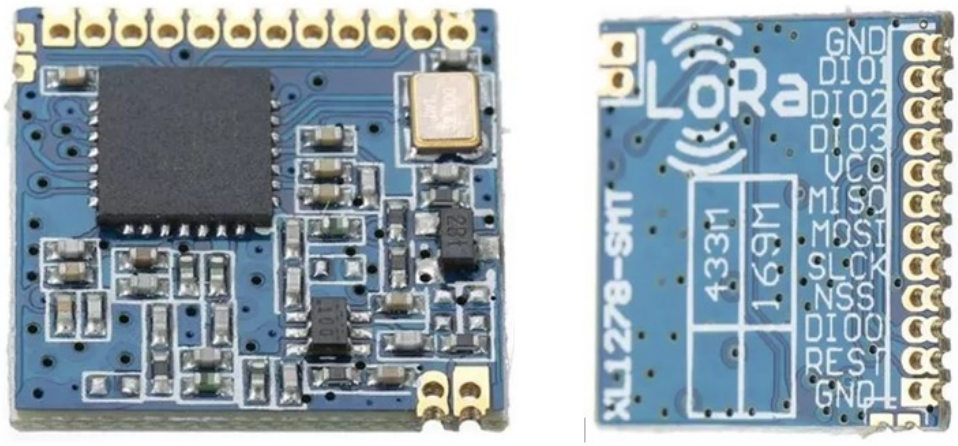


Imagen 31. Módulo SX1278 Lora Wifi

En la imagen se visualiza la función de los pines. Será el segundo módulo con SPI, por lo que solo quedará un esclavo disponible. Los dos que se ubican distantes hacen referencia a la antena y gnd. Del datasheet del dispositivo:

Pin name	Pin type	Description
VCC	Power(Analog)	1.8 V - 3.7 V analog power supply connection.
GND	Ground	Connect to the system ground.
RST	Digital Input	Reset, active low
MOSI	Digital Input	Serial configuration interface, data input.
SCK	Digital Input	Serial configuration interface, clock input.
MISO	Digital Output	Serial configuration interface, data output. Optional general output pin when CSN is high.
NSS	Digital Input	Serial configuration interface, chip select, active low.
DIOx	Input/output	GPIO
ANT	RF I/O	RF output signal from PA, connect to the Antenna.

Tabla 4. Pinout-Módulo SX1278 Lora Wifi

Las tres antenas más comunes para estos módulos:

Imagen 32. Posibles antenas para el Módulo SX1278

De izquierda a derecha tenemos primero la antena sprint o resorte: pequeña y de bajo precio, una opción económica para satisfacer necesidades básicas. A medio camino aparece la antena plástica SMA, dicha sigla hace referencia al tipo de conector dorado que vemos. Esta antena es de media escala, bajo costo y alta ganancia. Finalmente observamos la antena de montaje magnético, útil para un montaje en caja de hierro y que posee una ganancia considerablemente alta.

Como punto extra se puede decir que existen convertidores RS485-LoRa, por lo que existe compatibilidad.

Imagen 33. Conversores RS485-LoRa

Como último punto, previo al desarrollo, comentaremos acerca de los otros medios de comunicación disponibles en nuestro Arduino Due: CAN, I2C, SPI, UART (puerto serie).

A.3.4) Señales de Salida de sensores

-Sensores con señal de salida binaria (Tipo A)

Los sensores que tienen dos estados de salida, con lo cual dan una señal binaria indicando activado o detectando, o en el caso que el sensor esté en reposo estará inactivo. Son los finales de carrera, sensores de proximidad, presostatos, etc.

-Sensores con señal de salida por pulsos (Tipo B)

Los sensores cuya señal de salida es por trenes de pulsos suelen controlar la velocidad de los movimientos rotativos de algún sistema como puede ser un motor o polea estando este engranado al giro del sistema a medir generando unos pulsos que al ser más continuos estos pulsos nos indican mayor velocidad y en caso contrario a menor número de pulsos por unidad de tiempo la velocidad es más reducida. En muchos casos estos trenes de pulsos también nos informan del sentido de giro.

Los sensores con salida por trenes de pulsos pueden ser sensores incrementales de longitud, es decir, lineal o rotativos. Los sensores de este tipo suelen ser encoders.

-Sensores con salida analógica sin amplificador (Tipo C)

Los sensores con salida analógica tendrán una señal que puede tener infinidad de valores en función del dato a registrar, pero este tipo de sensores sin amplificador tienen una señal analógica muy baja, necesitando de un circuito complementario para poder visualizar la señal.

Estos sensores analógicos son potenciómetros lineales, sensores piezoeléctricos, piezoresistivos, pt-100 o células termoeléctricas, etc.

-Sensores con salida analógica amplificada (Tipo D)

Los sensores con salida analógica que el mismo sensor amplifica y convierte a una señal de salida analógica con valores legibles con equipos de control. Estos sensores también se le denominan transductores de señal.

Los tipos de señales que pueden tener de salida estos sensores y suelen ser los más utilizados son de 0 a 10V., de 0 a 20 mA y de 4 a 20mA.

-Sistemas de sensores con señal de salida estandarizada (Tipo E)

La señal de salida estandarizarada que utilizan estos sensores o sistemas de sensores son para comunicar por buses de datos, como pueden ser la RS 232, RS 485, etc. Son sistemas que permiten comunicar la señal obtenida con menos cableado y perdidas de señal despreciables.

A.4) Comunicaciones

En esta sección se tratarán los códigos implementados para RS485, TCP-IP y LoRA WiFi. Se explicarán las funciones y partes más importantes para comprenderlos y llegar a la aplicación en cuestión. En cuanto a las placas, se detallarán los circuitos y como es que acondicionan la señal.

En este apartado se introducirá a las librerías elegidas para los códigos de maestro y esclavo en Modbus RS485 y TCP-IP, enviar y recibir con interrupciones mensajes por Lora Wifi con el SX1278 y la justificación de los componentes usados para las placas acondicionadoras. En cuanto a la programación, la base fue extraída de la página “GitHub”. Para simular una comunicación usamos la PC y el software Modbus Poll y Slave. Como se mencionó, el contenido en general es obsoleto, con errores y hasta inconclusos. Sirve como base para adaptar a los desarrollos con Arduino Due.

Librerías para Modbus RS485 como esclavo

Para que el Due funcione como esclavo las librerías a utilizar son Modbus y ModbusSerial. Puede consultarse en http://github.com/andresarmento/modbus-arduino. Con ambas se tiene la posibilidad de manejar coils, discrete coils, input registers y holding register. Da la posibilidad de asignar direcciones y valores muy sencillamente para que el maestro lea y consulte sin problemas.

Librerías para Modbus RS485 como maestro

En el caso de usar el Arduino como maestro se usará SimpleModbusMaster, la cual puede ser consultada en https://github.com/jecrespo/simple-modbus.

Al igual que como esclavo se puede usar los cuatro tipos de datos. Se podrá escribir y leer, como también tratar una variable o multiples.

Librerías para Modbus TCP IP como esclavo

Para el caso de un esclavo TCP IP la tarea fue complicada ya que no se pudo manejar distintos datos, pero la comunicación puede establecerse. La librería utilizada fue Mudbus y se encuentra en https://github.com/luizcantoni/mudbus. Se podrá crear hasta 125 coils y 128 holding registers. Estos se podrán leer y escribir.

Librerías para Modbus TCP IP como maestro

En este caso se dispone de las librerías Ethernet y ModbusTCP. La información puede consultarse en https://github.com/goddland16/Modbus-TCP. Se tendrá la posibilidad de manejar todas las variables y funciones. A su vez, permite manejar dos dispositivos muy conocidos en el mercado como en el enc28j60 y el esp8266.

Librerías para Lora Wifi

El contenido que puede consultarse es más fiable, actual y amplio que en los casos anteriores y se decidió optar por LoraLib. La información referente puede verse en la página https://github.com/jgromes/LoRaLib. La librería permite enviar y recibir mensajes como también ver datos técnicos del dispositivo en pleno funcionamiento.

A.5) Circuito Acondicionador de Señales

Debido a la gran cantidad de tipos de señales presentes en un sistema eléctrico/electrónico, ya sea en un ambiente industrial pequeño, un hogar, cualquier entorno donde se desee implementar el dispositivo desarrollado, debe de tenerse en cuenta la incompatibilidad existente entre las señales externas a tratar y las señales que maneja internamente el Arduino o sistema embebido implementado, dichas incompatibilidades pueden ser en cualquier propiedad que caracteriza a una señal, ya sea voltaje, corriente, frecuencia, etc. Es por lo mencionado anteriormente que entre el Arduino y las señales externas debe existir una interfaz que “adapte” ambas partes, para ello se hace uso de distintos circuitos adaptadores de señal. A continuación, se hará mención de posibles situaciones en las cuales se desee llevar a cabo lo mencionado y los ejemplos que se utilizaron en el actual proyecto.

Adaptador de señales digitales o booleanas

El presente caso trata de señales las cuales pueden presentar solo dos estados “lógicos”, dichos estados serán on-off, alto-bajo, 220V-0V, etc. Dichas señales por su facilidad de procesamiento son empleadas en un amplio rango de aplicaciones, desde la recepción y/o actuación de una sola variable, hasta para el almacenamiento o codificación de información. Se recuerda que el sistema Arduino implementado para señales digitales solo admite un máximo de tensión de 3,3V por lo cual necesita en prácticamente la totalidad de los casos una adaptación.

Entradas digitales

En lo que refiere a la recepción de entradas digitales el sistema buscado es el siguiente:

Imagen 34. Diagrama en bloques de adaptador de entrada digital

Para el caso en que la señal digital externa sea de amplitud menor que los 3,3V admitidos por el Arduino el adaptador debería realizar una amplificación o pre amplificación, debido a que el dispositivo realizado no está orientado a dichas señales y que en la gran mayoría de los sensores la amplificación ya viene realizada por defecto, no se detallará sobre el caso tratado. Sin embargo, cuando se trata de una señal digital Externa de amplitud superior a los 3,3V admitidos por el Arduino se pueden emplear diferentes alternativas, en el mercado existen atenuadores ya compensados en frecuencia y aislados, los cuales bastaría solo con adquirirlos y emplearlos, dichos circuitos son relativamente básicos y sencillos por lo cual a continuación se procede a detallar la implementación de uno de ellos.
El adaptador ideal deberá contar con un divisor de voltaje, una aislación galvánica y una adaptación en frecuencia, aunque empleando un divisor de voltaje ya sería suficiente en muchos casos, pero de ninguna manera recomendado.

Partiendo de un divisor de voltaje se comienza a detallar el circuito, el mismo puede constar de cualquier elemento pasivo, en el proyecto se emplearán resistencias en configuración de divisor de voltajes, las mismas deben poseer un valor el cual sean lo suficientemente grandes para que el consumo de corriente sea despreciable, pero no lo demasiado para que el ruido eléctrico/electrónico interfiera relativamente.

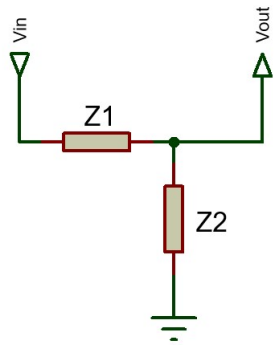


Imagen 35. Divisor de voltaje

En el esquema anterior Vin representa la señal digital externa a tratar, mientras que Vout representa la entrada digital del Arduino. En base al sistema anterior la ecuación que lo caracteriza es la siguiente:

$$Vout = Vin \times \frac{Z2}{Z1 + Z2}$$

Reemplazando los valores de voltaje necesarios y dando un valor al azar a una de las impedancias se puede calcular el total del sistema, sin embargo, el presente esquema no cuenta con una aislación galvánica ni estará compensado en frecuencia para cualquier caso, es por ello que se deben adicionar más componentes al adaptador.

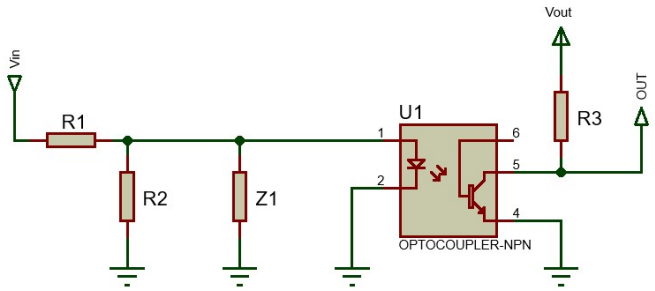


Imagen 36. Adaptador de entrada digital

En la figura previa se puede visualizar un adaptador adecuado para la mayoría de las aplicaciones, el funcionamiento del mismo se detalla a continuación. R2 y R1 deben elegirse en base a la ecuación de divisor de tensión con el objetivo de polarizar el diodo del optoacoplador empleado cuando se desee su activación, dicho optoacoplador ofrece una aislación galvánica, separando las señales externas del sistema interno. La resistencia R3 se utiliza como limitador de corriente y Z1 representa un filtro en el caso que se necesite una adaptación en frecuencia, si las velocidades son relativamente lentas Z1 puede despreciarse, en caso de ser necesario debe realizarse un análisis en base a la señal de entrada para armar el compensador en frecuencia adecuado. Para el caso que la señal externa de entrada presente un valor alto en forma de voltaje alterno debe agregarse un rectificador previo al optoacoplador.

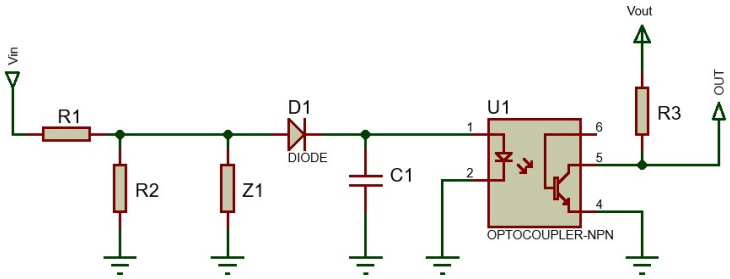


Imagen 37. Circuito entrada digital completo

El valor de Vout será el voltaje que interprete como valor lógico alto el Arduino, en el presente caso 3,3V. De este modo el adaptador está completo, cuando se presente una señal activa en la entrada, el optoacoplador actuará y pondrá a GND la salida. Cuando no se presente señal de activación en la entrada, la salida estará sometida a Vout, por lo que podría decirse que el circuito adapta e invierte una señal Vin, de entrada, a una Vout de salida.

Salidas digitales

En lo que respecta a las salidas digitales existen tres tipos empleadas en controladores lógicos programables, salidas a relé, salidas a transistor y salida a tiristor. Dependiendo el nivel de potencia y la velocidad de conmutación que se desee se recomienda una u otra, pero todas se encargan de realizar la misma función, adaptar el nivel lógico del controlador a un valor de señal distinto deseado. A grandes rasgos se puede especificar como características principales de la salida a relé que puede manejar gran cantidad de potencia con una velocidad de conmutación relativamente baja, mientras que en una salida a transistor o tiristor se reduce el nivel de potencia capaces de manejar en relación a la salida a relé pero posee la ventaja de una velocidad de conmutación mayor, debido a que no posee desplazamiento mecánico, sin embargo con el avance de la tecnología estos márgenes se ven acortados cada vez más. En lo que respecta a la aislación entre el sistema interno y externo el relé presenta una aislación galvánica, mientras que en la salida a transistor o tiristor no.

En el presente proyecto se optó por salidas a relé debido a las características nombradas previamente, cabe mencionar que igual al caso de las entradas digitales, debido a que se trabaja en base a un sistema abierto como Arduino, existe en el mercado gran variedad de módulos de salida los cuales pueden ser adquiridos y empleados sin ningún problema, dependiendo de la aplicación buscada. A continuación, se detalla la implementación de una salida a relé básica y funcional. El eje del adaptador buscado es la activación de un relé, por lo cual se necesita amplificar la salida del Arduino para lograr la excitación de la bobina de dicho relé.

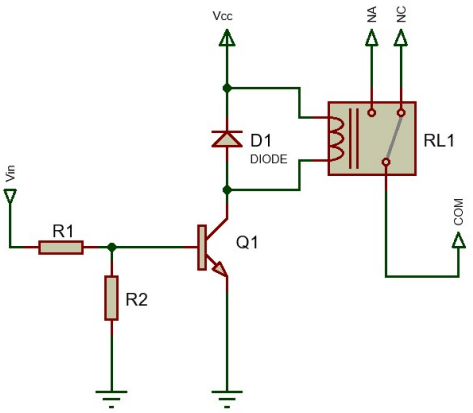


Imagen 38. Adaptador salida digital

El circuito anterior ejemplifica un buen adaptador para una salida digital buscada, mediante la salida del controlador, identificada con Vin, se procede a saturar el transistor Q1, dicha saturación provoca que la bobina del relé RL1 sea energizada y de este modo el contacto común (COM) dejara de estar en contacto con el normal cerrado (NC) y pasara a estarlo con el normal abierto (NA) de este modo se puede adaptar una señal de entrada a una de salida distinta, con la ventaja de una aislación galvánica. Cabe destacar que una vez el transistor vuelva a estar en corte el relé volverá a su posición de reposo, por lo que si se requiere una retención se deberá buscar la manera de enclavar la salida o mantener el transistor en saturación el tiempo que sea requerido. El diodo D1 simplemente se pone para evitar picos de tensión inversa provocados por la bobina al desenrizarse.

Por último, se presenta un posible adaptador de salida digital mediante tiristor a modo informativo, se aclara que en el presente proyecto solo se emplearon salidas a relé.

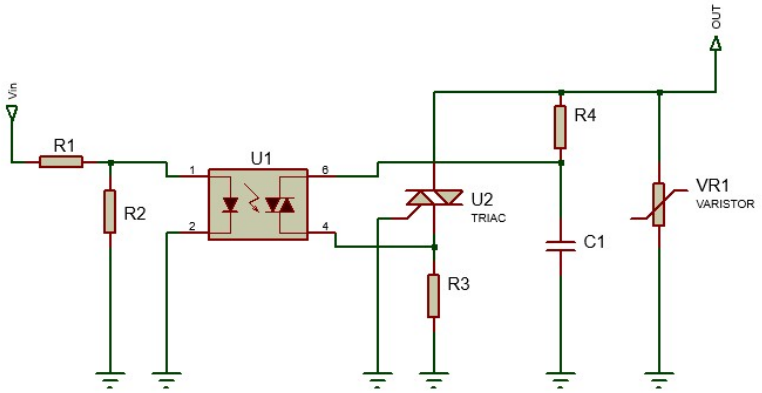


Imagen 39. Circuito salida digital con triac

El circuito anterior es un ejemplo de salida discreta en AC con triac.

Adaptador de señales analógicas

Existen casos en los cuales se desea adaptar, ya sea linealmente o no, una variable la cual posee infinitos valores acotada dentro de cierto margen, ya que dicha variable representa algún fenómeno físico de interés. Esta puede provenir de un sensor o lazo de control, pero además puede ser utilizado como una salida del controlador para la regulación o control de un actuador, debido a que el conversor analógico digital de un controlador está limitado a un margen generalmente distinto al de las señales externas, debe realizarse la correcta adaptación entre ellas, un ejemplo empleado en el proyecto es la adaptación de una señal de entrada analógica acotada entre 0V-10V a 0V-3,3V. Es por ello que se requiere un adaptador para dichas señales.

El principio fundamental de estos adaptadores es la representación fiel de una señal y escalarla perdiendo la menor cantidad de información posible.

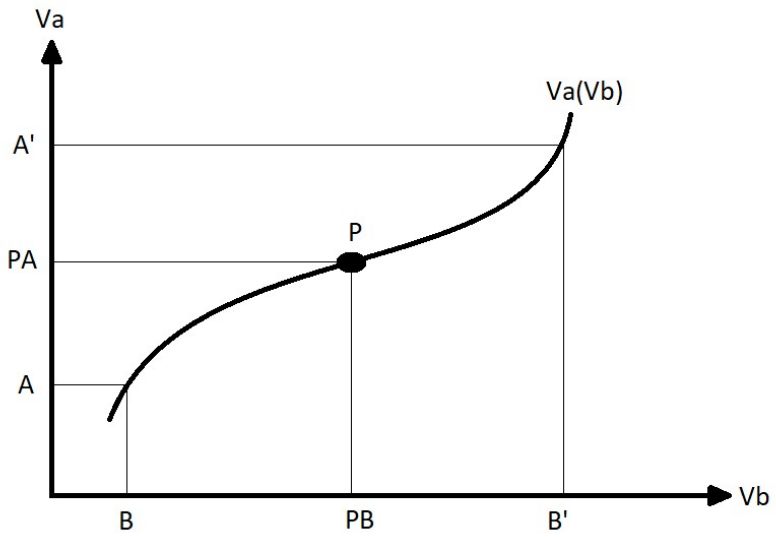


Imagen 40. Conversión de señal analógica

Gráficamente para una variable Vb se debe obtener una representación lineal que dependa de cada valor de Vb, tanto Vb como Va(Vb) deben estar acotadas, en la imagen anterior los márgenes están presentados con B-B' y A-A' respectivamente. Para un punto PB cualquiera dentro del rango admitido corresponderá un PA escalado linealmente.

En base a lo expuesto anteriormente se concluye que el adaptador buscado debe seguir la siguiente ecuación:

$$Vo = k \times Vin$$

El voltaje de salida Vo debe ser igual al voltaje de entrada Vin afectado por una constante, por lo cual un circuito simple capaz de cumplir dicha función puede ser un atenuador en caso de que k sea menor que 1 o un amplificador en caso que k sea mayor que 1.

Por lo general cuando se trate de una entrada analógica la cual presenta un valor superior al que es capaz de manejar el controlador estaremos presente en el caso que k debe ser menor que 1, un circuito posible a implementar es un atenuador acompañado de un seguidor de tensión mediante un amplificador operacional, el cual nos servirá además como adaptador de impedancia.

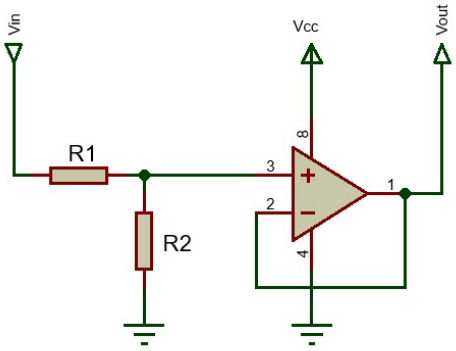


Imagen 41. Adaptador de entrada analógica

En el circuito anterior se puede apreciar que la relación de atenuación “k” está determinada por el divisor resistivo conformado por R1 y R2, en este sentido se deben tener en cuenta las características mencionadas anteriormente en el apartado de adaptadores de entradas digitales, además no se de obviar la frecuencia a la cual pueden trabajar los amplificadores operacionales debido a que estos no poseen buena respuesta a las altas velocidades de conmutación.
En el caso que se requiera un circuito adaptador de salida, caso en que k es mayor que 1, se deberá optar por un amplificador que se crea adecuado dependiendo la aplicación, para bajas potencia es conveniente la utilización de una clase A debido a su gran linealidad.

A.6) Gabinete

Una vez obtenidos todos los módulos y circuitos se debe desarrollar una interfaz física para el sistema competo. Existen diferentes normas para el diseño y construcción de gabinetes, ya sean ANSI, IRAM, etc, las mismas se encargan de detallar el correcto procedimiento y especificaciones técnicas las cuales deben tenerse en cuenta a la hora del desarrollo del gabinete. No es intención del presente proyecto explicar las normas, dependiendo el trabajo solicitado se deberá optar por una correcta elección del gabinete, ya sea obteniendo uno en venta o realizando el diseño y mandarlo a su futura realización, esta última opción será empleada en el trabajo actual, el cual solo quiere demostrar una aplicación “general” del kit.

Existen diferentes aspectos fundamentales a tener en cuenta, seguidamente se describen una serie de ellos a modo ilustrativo para la compresión del lector.

Circuitería interna

Dependiendo el tamaño, cantidad y disposición del total de circuitos desarrollados para el kit se tendrá en cuenta un correcto diseño, estos serán de aspecto fundamental a la hora de planificar las dimensiones del gabinete ya que la circuitería interna constituye la mayoría del sistema empelado

Entradas y salidas del kit con el exterior

Uno de los aspectos principales a visualizar en el futuro gabinete es la cantidad de entradas y salidas del sistema con el exterior, estas pueden variar tanto en forma como en cantidad, dependiendo los módulos empleados y las borneras o pines de conexión empelados debe preverse el espacio adecuado para ello.

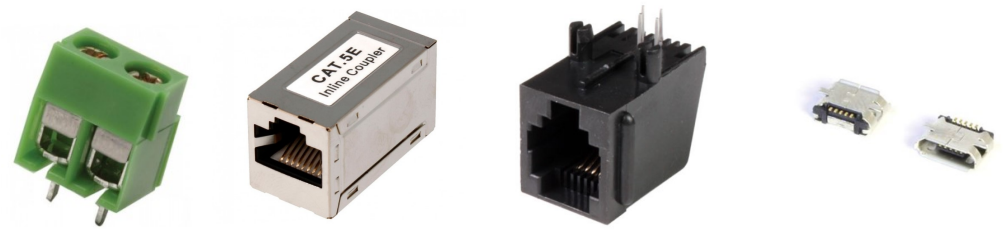


Imagen 42. Interfaces de conexión

Grado de protección

Dependiendo el ambiente de campo donde será dispuesto el kit, debe tenerse presente las condiciones ambientales, además de la ventilación y disipación necesaria. Es por ello que existen como norma general los “grados ip” los cuales ayudan a determinar el correcto diseño en base al grado adecuado. A modo de ejemplo se detallarán a continuación los grados ip existentes.

PROTECCIÓN ANTE CUERPOS SÓLIDOS			PROTECCIÓN CONTRA EL AGUA	
	Sin protección	0	Sin protección	
	Protección contra objetos con diámetro superior a 50mm	1	Protección ante un goteo vertical	
	Protección contra objetos con diámetro superior a 12mm	2	Protección contra goteo con incluinación de 15º	
2.5mm	Protección contra objetos con diámetro superior a 2.5mm	3	Protección ante pulverización	
1mm	Protección contra objetos con diámetro superior a 1mm	4	Protección ante salpicaduras	
	Protección ante el polvo	5	Protección ante chorros de agua	
	Protección totalmente estanco ante el polvo	6	Protección ante chorros continuos de agua	
		7	Protección contra inmersiones temporales	
		8	Protección contra inmersiones permanentes	

Tabla 5. Grado de protección IP

Recordar que un caso especial es la flora y la fauna del lugar donde se instalará el kit, desde simples hormigas hasta abejas, roedores y demás pueden

ocasionar una falla, por lo mismo el presente paso no debe ser tomado a la ligera y debe tenerse en cuenta lo anterior descripto.

B) Propuestas de Circuitos o Esquemas: pruebas, problemas y soluciones

Para la programación no se han tenido problemas dado que las fuentes explican las librerías de manera detallada. Los puntos complicados fueron los errores en cuanto a compatibilidad con Due, donde hubo que redefinir puertos y formatos, y la falta de librerías extras para su correcto funcionamiento. Luego se pudo poner en funcionamiento todo sin problemas. En el caso de RS485 se adquirió un conversor USB-RS485 para usar la PC como maestro o esclavo atrás del software ya mencionado. Para Ethernet simplemente se configuró el puerto de la PC y se conectó con un cable UTP directo. El mismo software que se usa para RS485 también puede configurarse para TCP-IP indicándole la IP del Arduino. En cuanto a Lora Wifi el medio de comunicación es el aire, usamos dos Arduinos y dos SX1278. El principal error es que el mensaje empieza a llegar con datos erróneos al poco tiempo, pero pudo solucionarse limpiando el buffer.

Para RS485 se puede comprar un integrado y realizar una placa, pero los costos se elevan y resulta mucho más práctico el módulo. De igual manera para Ethernet y Lora Wifi. Lo importante en este caso es contar con un buen código y librería. Para elegir el módulo correcto hay que asegurarse que va a funcionar en Due, pero no es menos importante que existan códigos desarrollados. Este trabajo es fundamental, consiste en buscar si se han realizado trabajos con los dispositivos RS485, Ethernet y Lora Wifi que se van a adquirir.

Se realizarán pruebas con códigos en el IDE de Arduino para el Board Due, probando distintas librerías y códigos hasta finalmente optar por lo que mejor funcione. Salvo por el caso de Lora Wifi, tanto en RS485 y Ethernet, la mayor parte del contenido encontrado por no decir un 90% es considerado obsoleto, los mismos autores explican que no están más en el asunto. Una página muy consultada fue “github”. El trabajo para consolidar y poner en marcha códigos obsoletos y con errores no fue sencillo, pero tampoco imposible. En el caso de RS485 los inconvenientes que ocurrían eran por incompatibilidad de Boards, un código diseñado para UNO no es compatible con DUE y el más común por librerías incompletas. Una vez que decidimos los códigos a usar, podemos adquirir los módulos y probar.

Para señales de entrada se tiene en cuenta la posibilidad de que estas sean digitales o analógicas. También es de suma importancia que el valor máximo que deben tener es 3.3V, por eso estas placas, por ejemplo, acondicionaran una señal de 0-10V de un sensor de temperatura a 0-3.3V para poder ser analizada.

En cuanto a la salida simplemente es una salida a relé de 5VDC, mientras que para las entradas se usa un optoacoplador si la entrada es digital y un LM741 si es analógica.

B.1) Propuestas de circuitos adaptadores de señales

Los circuitos con los cuales se llevó a cabo el proyecto fueron los siguientes:

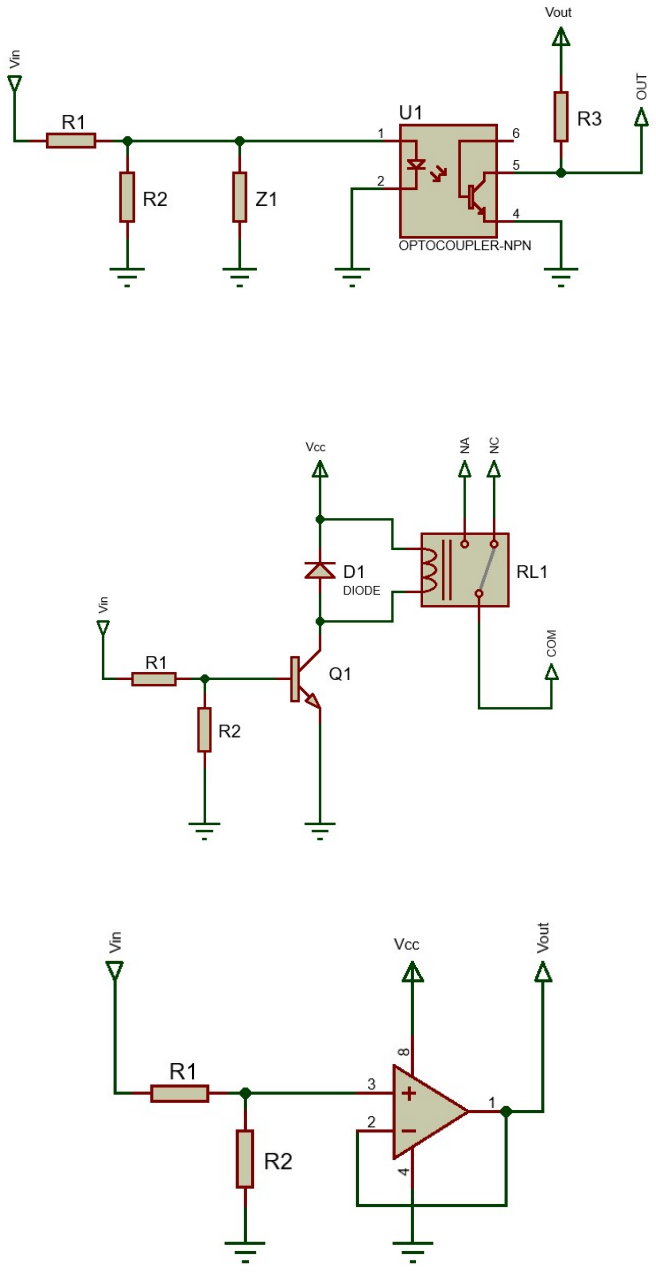


Imagen 43. Circuitos adaptadores de señales empleados

Los mismos fueron seleccionados debido a sus prestaciones, las cuales por su simplicidad y funcionamiento satisfacen los requisitos básicos buscados en el

presente trabajo. Cabe destacar que dependerá de las características de campo para determinar los valores de algunos componentes, así también la cantidad requerida y demás aspectos que se requieran.

Se procedió a la confección del diseño PCB mediante el programa Proteus 8 Professional, usando componentes no de montaje superficial, con el fin de simplificar la puesta a prueba y su adquisición, lo ideal a la hora de realizar el kit completo y finalizado es emplear componentes de montaje superficial para una mayor optimización en los espacios, así también dependiendo del entorno del campo de trabajo donde se empleará debe tenerse en cuenta la contaminación del ruido electrónico, vibraciones, temperatura, humedad, con el fin de una correcta selección del tipo de componentes a emplear.

El diseño final de los PCB quedo de la siguiente manera:

Placa maestra

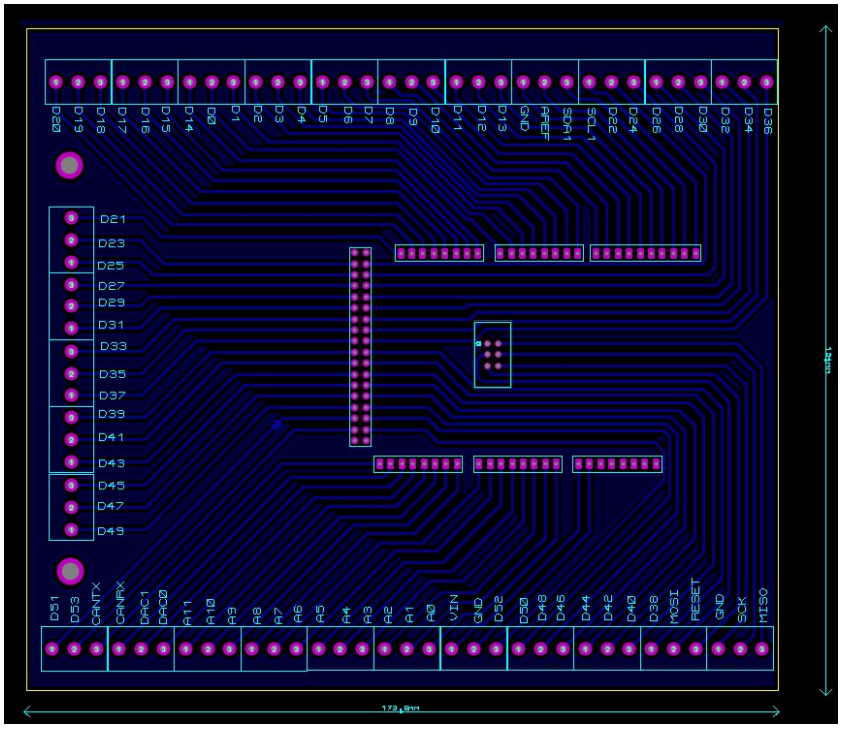


Imagen 44. PCB placa madre

La misma es la placa fundamenta en el kit, la misma está diseñada para contener el sistema embebido a emplear, en la imagen se encuentra realizada para un Arduino Due, la misma se encarga de presentar un conexionado confiable de los pines del Arduino con el exterior, debido a que los que trae de fábrica no son lo suficientemente robustos para un entorno semi industrial, también se observa que no todos los pines del Arduino son llevados hacia los bornes de conexión externos, esto es debido a lo maleable que se desea presentar el proyecto, en el mismo se

recalca el uso de únicamente lo necesario para el trabajo solicitado, de este modo se reducen los costos de manufactura así también el tiempo y la complejidad.

Adicionalmente la presente placa tendrá espacio superior para colocación de módulos o circuitos complementarios.

Adaptador de entrada discreta

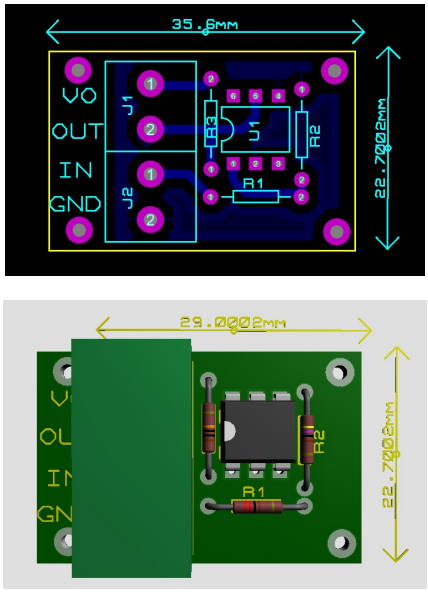


Imagen 45. PCB adaptador de entrada digital

El mismo fue pensado para activar una entrada discreta de 5V de entrada y una salida de 3,3V. Los valores de los componentes empleados fueron los siguientes:

R1 = 2,2K
R2 = 4,7K
R3 = 100K
U1 = 4N25

Donde las etiquetas de las borneras representan lo siguiente:

VO = Voltaje de salida que tomará como referencia la salida (3,3V)
OUT = Salida la cual estará negada con respecto a la entrada
IN = Voltaje de entrada de valor lógico a adaptar (5V)
GND = Masa

Adaptador de entrada analógica

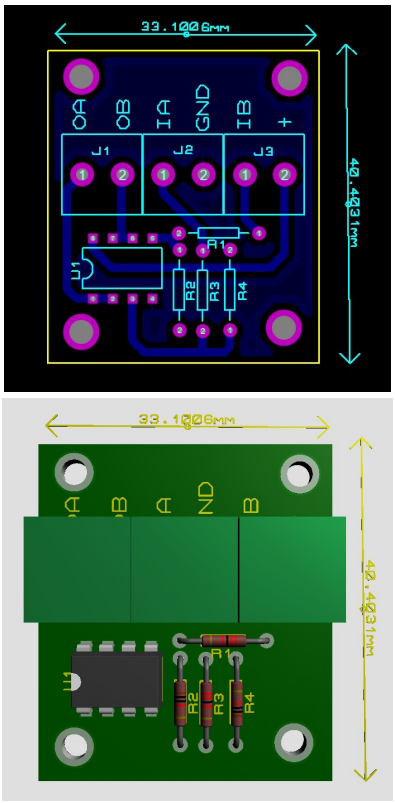


Imagen 46. PCB adaptador de entrada analógica

Aprovechando la capacidad de operacionales con los que cuenta gran cantidad de integrados de este tipo en el mercado se realizó el diseño de dos conversores de voltaje en vez de uno. Los mismos serán encargados de adaptar un voltaje de entrada con un margen de 0-10V a 0-3,3V.

Los valores de los componentes empleados fueron los siguientes:

R1 y R3 = 22K
R2 y R4 = 10K
U1 = LM393

Donde las etiquetas de las borneras representan lo siguiente:

+ = Voltaje de alimentación de los operacionales
IA y IB = Entradas de voltaje analógico (0-10V)
OA y OB = Salidas de voltaje analógico (0-3,3V)
GND = Masa

Adaptador salida a relé

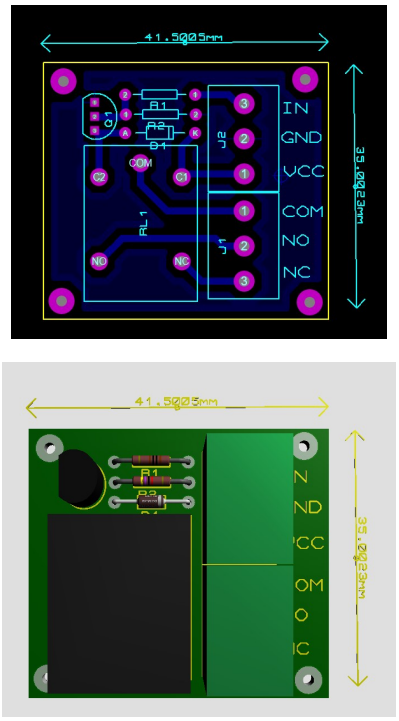


Imagen 47. PCB adaptador de salida digital

La presente placa debe encargarse de activar o desactivar un relé, se optó por la utilización de un relé de 5V de bobina, capaz de tolerar hasta 220V 7A en CA.

Los valores de los componentes empleados fueron los siguientes:

R1 = 1K
R2 = 47K
D1 = 1N4148
Q1 = 2N3904
RL1 = Relé

Donde las etiquetas de las borneras representan lo siguiente:

IN = Voltaje de activación
VCC = Voltaje de alimentación del circuito
COM = Conector del común del relé
NO = Conector del contacto normal abierto del relé
NC = Conector del contacto normal cerrado del relé
GND = Masa

En lo que respecta a las pruebas una vez realizados los circuitos no se presentaron inconvenientes en las placas diseñadas, en el caso del adaptador de entrada

analógico existe una desviación que se detallará en el apartado de problemas y soluciones.

B.2) Diseño de Gabinete

Debido a la gran flexibilidad que ofrecen las impresiones 3D en la actualidad y la buena respuesta del material de construcción se optó por dicha opción a la hora de su elección para llevar a cabo el gabinete del kit. Para el mismo se realizó el diseño de un kit el cual contará con módulo RS485-TTL, módulo de interfaz TCP-IP y borneras de conexionado. A continuación, se presentan imágenes del diseño inicial del mismo:

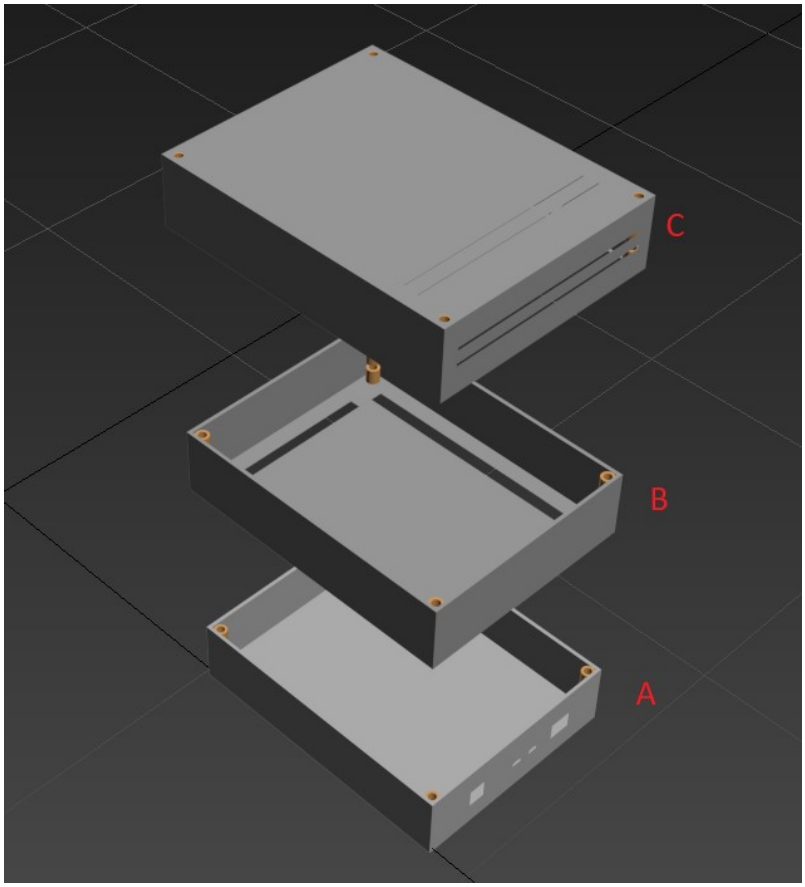


Imagen 48. Gabinete por partes

El diseño actual fue pensado teniendo en cuenta tres partes fundamentales. La parte A será la base del gabinete, el mismo contendrá la placa madre y aprovechará el espacio para presentar en su parte posterior las interfaces de los módulos de comunicaciones principales.

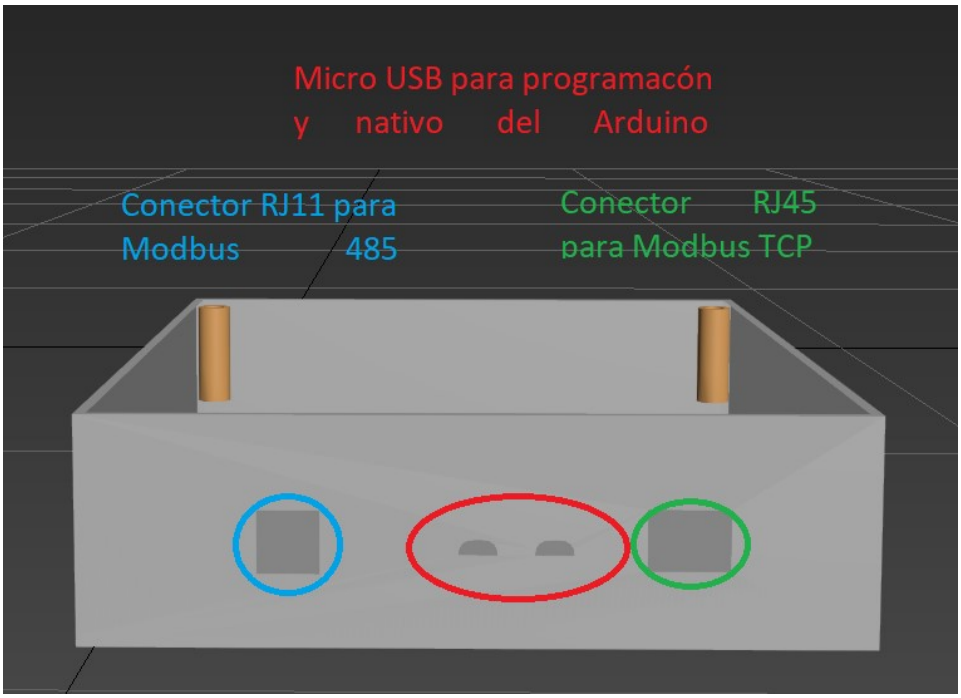


Imagen 49. Base de gabinete

La parte B representa uno o más slots necesarios para disponer de lugar físico a los circuitos y módulos adicionales requeridos en el sistema, además contará con espacio para realizar el conexionado entre la parte A, B y C mutuamente.

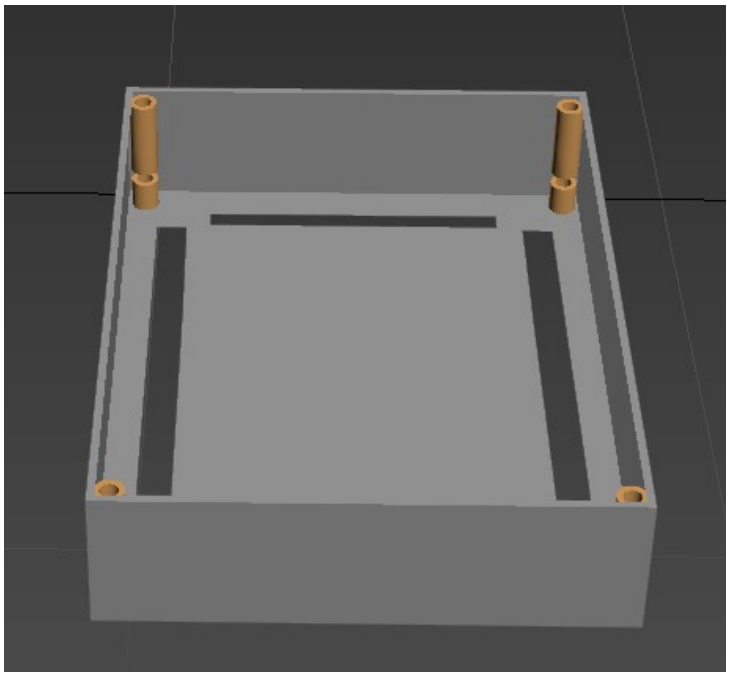


Imagen 50. Slot medio del gabinete

Por último, la parte C se encarga de portar las borneras de conexionado hacia el exterior del kit además de ser la tapa del gabinete.

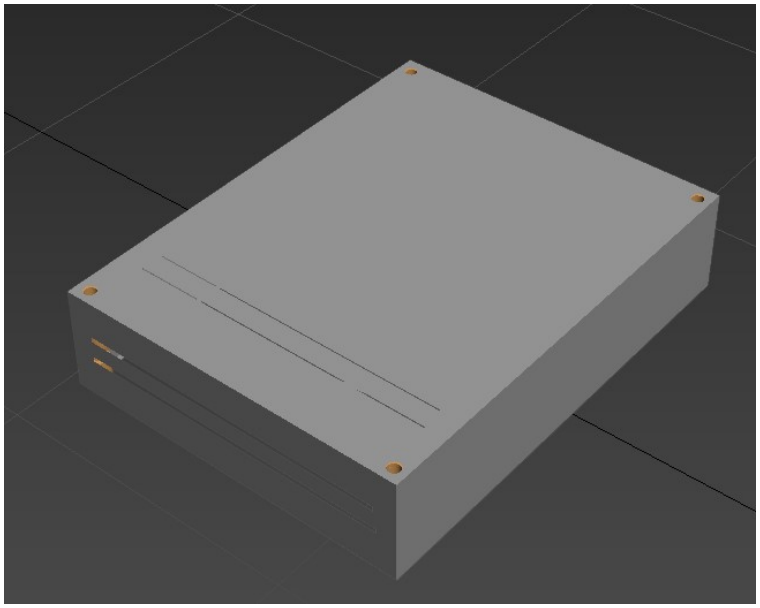


Imagen 51. Tapa del gabinete

Por lo que el gabinete una vez ensamblado se estima que presentara la siguiente forma.

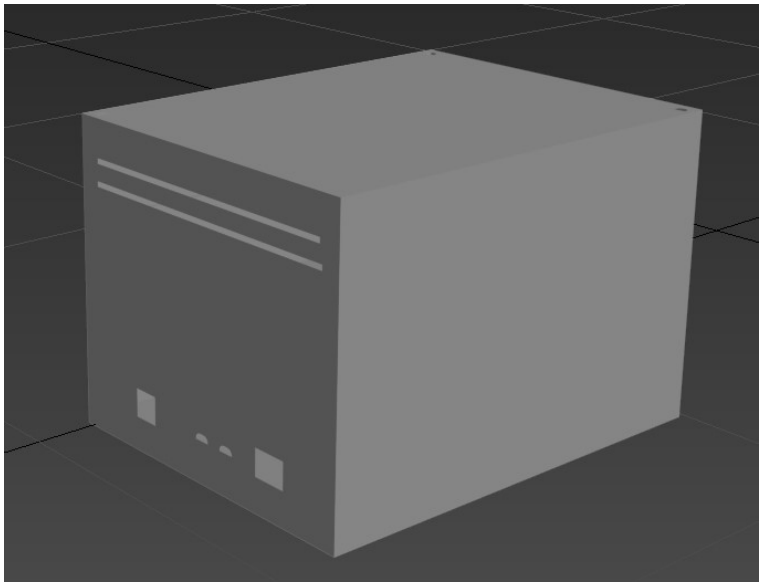


Imagen 52. Diseño completo del gabinete

Un posible problema detectado a la hora de la impresión fue la disposición de una impresora la cual cuente con un área de impresión capaz de contener el

total del volumen de cada parte, debido a que cada parte posee un volumen relativamente superior al del promedio encontrado en las impresoras de la zona. Para este problema simplemente se puede optar por mandar a imprimir a distancia o se puede realizar una impresión por subdivisión de partes para un posterior ensamble, esta última fue la seleccionada para el actual proyecto y su ejecución no presento dificultad alguna.

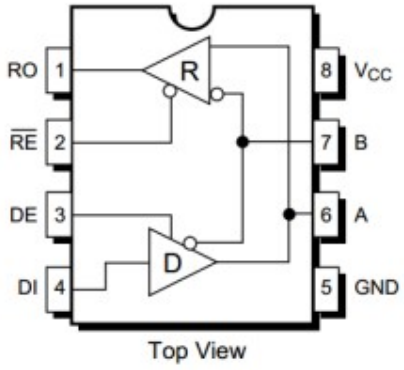
Otro posible inconveniente, no funcional sino estético, es el relieve o patrón que dejan las impresoras al realizar la impresión, el mismo puede ser solucionado con la implementación de un revestimiento y lijado posterior en tanto y cuanto no interfiera con las propiedades pensadas para el gabinete.

C) Análisis del funcionamiento

Se analizan las funciones más importantes de las librerías de comunicaciones y la electrónica de las placas, con el objetivo de llegar a la aplicación sin dudas.

Cuando se habla del código se tienen en cuenta ciertos aspectos: librerías y ciertas variables se definen fue del setup. En este último se realizan configuraciones previas para finalmente realizar lo necesario en el loop. De esta forma se definen las tres áreas de trabajo de un típico programa.

En RS485, todo es realizado por el MAX3485. En cuanto a TCP-IP, el módulo cuenta con el Wiznet W5100. El diagrama en bloque nos da un pantallazo del funcionamiento del mini Shield. Para Lora Wifi, una placa muy pequeña basada en el integrado SX1278 de Semtech ofrece una salida adaptada en 50 [ohm] para la antena. Comprender el funcionamiento en profundidad de los circuitos de los módulos se escapa a lo que estamos tratando en el proyecto. El análisis completo de las placas realizadas se detalla en el segundo bloque del proyecto.



Top View

SP485/MAX485/SP3485/MAX3485
Pin layout (Top view)

Pin	Name	Description
1	RO	Receiver Output
2	$\overline{RE}$	Receiver Output Enable Active LOW
3	DE	Driver Output Enable Active HIGH
4	DI	Driver Input
5	GND	Ground Connection
6	A	Driver Output/Receiver Input. Non-inverting
7	B	Driver Output/Receiver Input. Inverting
8	V_{CC}	V_{CC} < 5.25V

SP485 / MAX485 are RS485 transceivers requiring 5V power supply.
SP3485 / MAX3485 are RS485 transceivers requiring 3.3V power supply.

Imagen 53. Integrado para RS485

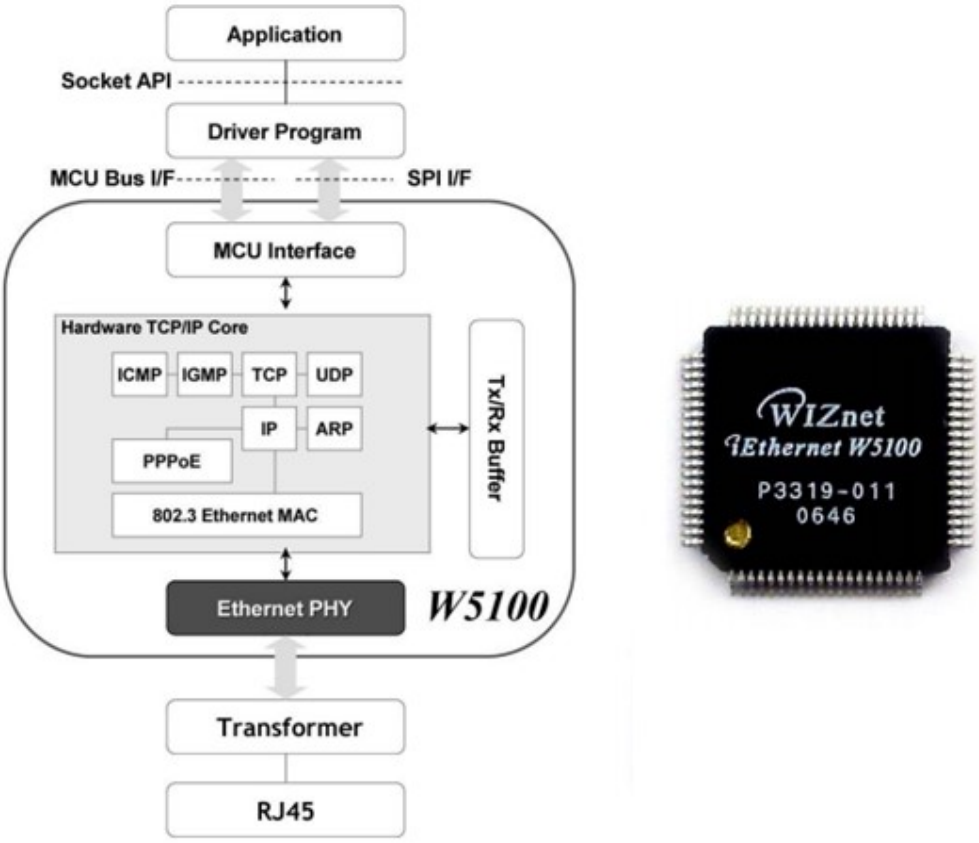


Imagen 54. Wiznet W5100 – Diagrama en Bloques

C.1) Funciones para Modbus RS485 como esclavo

Para empezar, se crea un objeto fuera del setup mediante ModbusSerial, con el que se configurará el esclavo mediante config y la ID con setSlaveID (1-247).

Se explican dos conceptos que aparecen en la configuración:

BaudRate: la velocidad a la que se transfiere la información en un canal de comunicación. En el contexto del puerto serie, "9600 baudios " significa que el puerto serie es capaz de transferir un máximo de 9600 bits por segundo.

Configuración del puerto: la más usada para un puerto serial es:

8 bits de datos 1 bit de parada Velocidad de 9600 BAUD

Sin bit de paridad 1 bit de inicio

Para que pueda haber una sincronización de los datos enviados, se requiere que ambos dispositivos que usen el mismo puerto serial, tengan la misma configuración.

En la configuración se indica el puerto serie usado del Due, el BaudRate, la configuración del puerto y el pin digital que cambia al módulo a emisor-receptor.

ModbusSerial mb; mb. setSlaveId (1); mb.config(&Serial1, 9600, SERIAL_8N1, 5);

Para agregar un dato tenemos que darle a la función add la dirección y el valor que va a tener. Daremos un ejemplo para cada tipo:

mb.addCoil(LAMP1_COIL,true); Agregamos una bobina con valor 1

mb.addIsts(DI1,false); Agremos una bobina discreta con valor 0

mb.addHreg(HREG1, 127); Agregamos un holding register de valor 127

mb.addIreg(IREG1,9); Agregamos un input register de valor 9

En todos los casos la dirección viene dada como la primera variable y debe ser un entero, el cual puede declararse fuera del código como un const int. Un punto no menos importante es que, por ejemplo, si en las bobinas no damos valor, da false por defecto.

Finalmente llegamos al loop, donde ejecutamos el task: mb.task();

Esta función actualiza los datos que hemos creado. Si el maestro decide cambiar un valor, cada vez que se ejecute task, sabremos si ocurrió algo. Es muy útil si se realiza un programa donde en función de algún cambio que haga el maestro, el esclavo tenga que realizar alguna acción.

Un punto sumamente importante a tener en cuenta es que el maestro debe seleccionar correctamente lo que desea leer del esclavo. A modo de ejemplo: si tenemos 5 coils creadas en el esclavo en las direcciones 4,5,6,7 y 8, si se le pide al

maestro que lea desde la dirección 2 a las 10 de nuestro esclavo, ocurrirá un error porque en las direcciones 2,3,9 y 10 no se creado nada en el esclavo.

C.2) Funciones para Modbus RS485 como maestro

Para ser maestro se emplea una librería distinta que se maneja de una forma totalmente diferente a la del esclavo anterior. Se comienza por definir variables de datos y configuración para el setup:

```
#define baud 9600          BaudRate
#define timeout 1000       Tiempo de espera en milisegundos
#define polling 200  Tiempo por cada llamada al esclavo en milisegundos
#define retry_count 10     Tiempo de reintento por fallo de comunicación en milisegundos
```

Brevemente, si la conexión es exitosa, el maestro solicita algo al esclavo cada 0,2 segundos. En caso ocurrir un fallo, se espera 1 segundo más el retry count y se reintenta.

```
#define TxEnablePin 5      Pin para setear en emisor-receptor el módulo
#define TOTAL_NO_OF_REGISTERS 10      Número de registros con valores
#define TOTAL_NO_OF_PACKETS 10        Número de paquetes de acciones
Enum                                  Creamos paquetes
{
  PACKET1,
  PACKET2,
  PACKET3,
};
Packet packets[TOTAL_NO_OF_PACKETS];      Vector de paquetes de acciones
unsigned int regs[TOTAL_NO_OF_REGISTERS];     Vector de valores de variables
```

Para entender con más detalle debemos avanzar las funciones que ser realizan dentro del setup.

```
modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 25, 1, 1);

modbus_construct(&packets[PACKET3], 1, PRESET_MULTIPLE_REGISTERS, 30, 1, 3);

modbus_construct(&packets[PACKET2], 1, READ_HOLDING_REGISTERS, 46, 1, 2);

modbus_configure (&Serial2, baud, SERIAL_8N1, timeout, polling, retry_count, TxEnablePin, packets, TOTAL_NO_OF_PACKETS, regs);
```

Se empieza por entender que es un paquete de acciones. Cada solicitud que se realice debe construirse con modbus_construct y una acción determinada necesita un paquete del vector de paquetes creado previamente. Para construir la acción, la primera variable es un paquete del vector, seguido del ID del esclavo en cuestión. Luego sigue la función a realizar, leer o escribir uno o más datos.

Finalmente, y no menos importante se debe entender cómo funciona el vector de registro de valores. El vector de registros del tipo unsigned int creado previamente es un vector donde estarán todos los valores que se usen en las acciones. Entonces procedemos a entender la última variable, formada por tres números, de la función modbus_construct. Primero se debe decir la dirección inicial del esclavo a partir de la cual se realiza la acción y desde ese lugar, cuanto más vamos a seguir. El último número es de vital importancia e indica a partir de qué dirección del registro de valores vamos a guardar la información que usamos, si se escribe o recibe, si se lee, al realizar una acción. Es complejo y requiere de un orden esencial, debido a que es muy factible sobrescribir datos si confundimos números y direcciones. El último punto es que, si tenemos un total de 3 acciones a realizar, se deben ordenar en orden creciente, en función de la dirección del registro de esclavo que se lee, tal como se mostró. Una vez entendido esto, se puede terminar por configurar la conexión con modbus_configure. Las variables que recibe son: el puerto del Arduino usado, el BaudRate, la configuración de puerto serie, timeout, polling, retry count, el pin de cambio de modo, el vector de paquetes, el número de paquetes y el vector de valores. Un aspecto no mencionado pero lógico es que el orden de las variables nunca debe modificarse, porque sigue las leyes de la librería.

Para manejar el maestro, se emplea el monitor serie, para eso se inicia con Serial. Begin(BaudRate). Es muy útil para solicitar información al esclavo y también observar que es lo que llega.

Para el loop, al igual que en el esclavo se cuenta con una función que actualiza la información del vector de valores cada vez que se requiera, es modbus_update (). Dentro del loop también se puede realizar cambios en los valores para cambiar la información en acciones que solicitan escritura del esclavo,

pero las acciones en sí, no pueden cambiarse. Una acción, una vez construida, no se le puede cambiar ninguna variable. La única excepción es lo nombrado previamente con el vector de valores.

C.3) Funciones para Modbus TCP IP como esclavo

En el caso del esclavo TCP IP, la librería Mudbus es muy sencilla. También precisa de la librería Ethernet.

El primer paso es crear un Objeto Mudbus: Mudbus Mb;

Ya en el setup se necesita configurar el puerto IP. Solo necesita la Mac del moduló w5100 y la ip del Arduino que se usará. Esta debe ser la siguiente respecto al dispositivo que haga de maestro. A continuación, se muestra la configuración:

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1 , 2);
Ethernet.begin(mac, ip);
```

Para iniciar la comunicación se precisa la Mac y la IP, si bien hay inicializaciones más complejas, que incluyen Gateway, DNS y subnet. El dispositivo maestro fue emulado con una PC, se observa su configuración:

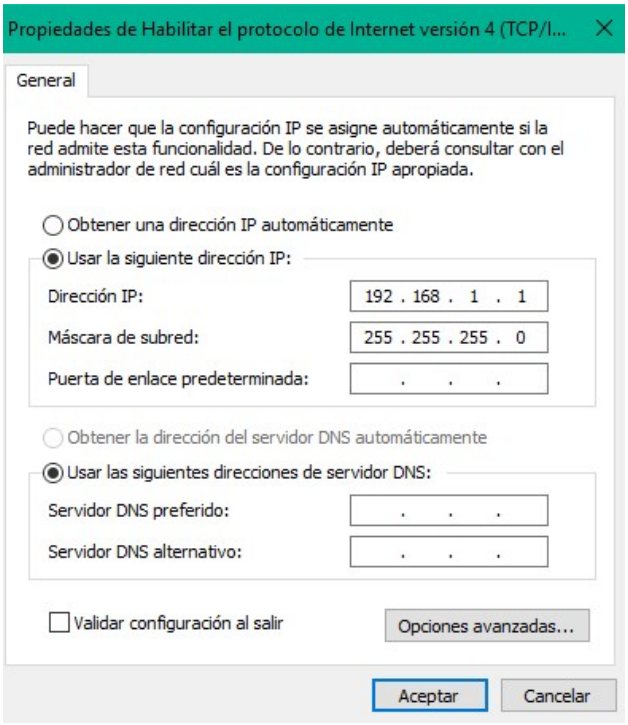


Imagen 55. Configuración Modbus TCP

Esto es todo en cuanto a la configuración. Al crear el objeto Mudbus, automáticamente se tendrá a disposición 0-125 HoldingRegister que toman datos enteros con signo y 0-128 coils que toman datos booleanos. Para referirse a ellos, se utiliza el objeto creado y la dirección: Mb.R[ ] - Mb.C[ ].

Al igual que en RS485 se tiene una función que actualiza los datos del objeto, cada vez que se necesite, se usa en el loop : Mb.Run().

C.4) Funciones para Modbus TCP IP como maestro

El manejo de esta librería es un tanto más complejo respecto a las demás. Se comienza, previo al setup, por definir el módulo que se tiene:

#define WIZNET_W5100 1

El paso siguiente es crear un objeto referente al dispositivo que hace de esclavo y asignarle la ID. Al igual que en el caso de TCP IP esclavo, se necesita la librería ethernet.

ModbusTCP node(1); node es el objeto esclavo con ID 1

Ya en el setup, los pasos para arrancar la comunicación son los mismos, con tan solo la Mac del módulo y la IP del Arduino. Para setear la IP de esclavo se tiene:

node.setServerIPAddress(ModbusDeviceIP);

Ahora nos centraremos en todas las funciones que pueden ser usadas para leer, escribir y de otras características, dentro del loop.

node.setTransactionID(random(100));

La función setea un tiempo de inactividad para la devolución de una llamada de función o solicitud, en 0-100 ms para el esclavo. Es algo totalmente optativo.

node.clearResponseBuffer(); Borra el buffer de respuesta Modbus

Buffer: es un espacio de memoria, en el que se almacenan datos de manera temporal, normalmente para un único uso. Sirve para evitar que el programa o recurso que los requiere, ya sea hardware o software, se quede sin datos durante una transferencia (entrada/salida) de datos irregular o por la velocidad del proceso.

node.clearTransmitBuffer(); Limpia el buffer de transmisión Modbus

int len = node.getResponseBufferLength();

Se utiliza para saber la longitud del paquete de información recibido, se guarda en una variable del tipo entero.

node.getResponseBuffer(byte); Devuelve en byte la respuesta en el buffer

Esta función puede usarse para conocer los valores tras una solicitud de lectura.

node.writeSingleRegister(5, 3); Escribe 3 en holding register de dirección 5

Para hacer una escritura multiple de registros se procede de la siguiente forma:

for (byte i = 0; i < 5; i++) {node.setTransmitBuffer(i, (9+i));}

node.writeMultipleRegisters(2, 5);

Se escribe en formato byte 5 registros, para eso realizamos un for. Para la información que se transmite, en cada ciclo for se usa setTransmitBuffer, indicando el número de registro y la información. Todo es en byte. El método del for también se usa para leer multiples registros con getResponseBuffer.

node.writeSingleCoil(20, 1); Escribimos true en el coil de dirección 20

node.writeMultipleCoils(20, 16); Escribimos desde el coil 20 , 16 veces

Para escribir usamos nuevamente setTransmitBuffer con una cadena de “1” y “0”

Para leer coils, discrete inputs, input registers y holding registers

node.readCoils(dir, cant);

node.readDiscreteInputs(dir, cant);

node.readHoldingRegisters(dir, cant);

node.readInputRegisters(dir, cant);

En todos los casos se especifica la dirección de inicio y la cantidad

Otras dos funciones interesantes pero que no hemos utilizado son:

maskWriteRegister(uint16_t u16WriteAddress,uint16_t u16AndMask, uint16_t u16OrMask): modifica el contenido de un holding register específico, usando una combinación de máscaras AND, OR y la del registro en cuestión. Puede usarse para setear o limpiar bits de un registro de forma individual. La máscara en informática es el conjunto de datos que, junto con una operación, permiten extraer selectivamente ciertos datos almacenados en otro conjunto. En la llamada a función se solicita el registro a setear, la máscara AND y la OR.

readWriteMultipleRegisters(uint16_t u16ReadAddress, uint16_t u16ReadQty, uint16_t u16WriteAddress, uint16_t u16WriteQty): como su nombre lo indica, da la posibilidad de leer y escribir. Se podrá realizar una combinación donde habrá una operación de lectura y otra escritura en una sola transmisión Modbus. La escritura se realiza antes de la lectura. Se especifica dirección primer registro y número de registros. La información a ser escrita se especifica en el buffer de transmisión.

C.5) Funciones para Lora Wifi

La aplicación del proyecto no lleva Wifi, pero se experimentó y se consiguieron buenos resultados, así como un alcance cercano al esperado. La librería ofrece muchas opciones, pero simplemente la operación fue enviar un mensaje desde un Arduino y recibirlo en otro a la mayor distancia posible. Abarcaremos las funciones básicas de configuración del módulo, envío y recepción.

Se comienza por incluir la librería LoraLib y crear un objeto indicando los pines NSS, DIO0 y DIO1. El primero se usa para el esclavo SPI, el segundo como interrupción para alertar el Arduino, mientras que el último puede ir en cualquier pin o desconectado.

```
SX1278 lora = new LoRa(4,2,3);
```

Ahora ya se puede pasar al setup, donde es útil utilizar el monitor serie del Arduino para mostrar si se pudo configurar correctamente el módulo. Se creará una variable del tipo entero que almacenará la respuesta para la configuración del módulo, donde deben ir las características técnicas:

```
Serial.begin(9600);
// initialize SX1278 with default settings
Serial.print(F("Initializing ... "));
// carrier frequency:           434.0 MHz
// bandwidth:                   125.0 kHz
// spreading factor:            9
// coding rate:                 7
// sync word:                   0x12
// output power:                17 dBm
// current limit:               100 mA
// preamble length:             8 symbols
// amplifier gain:              0 (automatic gain control)
 int state = lora.begin(434.0, 125.0, 9, 7, 0x12, 17, 100, 8);
```

Luego se consulta comparando nuestra variable de estado con ERR_NONE. Si son iguales, el dispositivo se conectó y configuró como corresponde. Ahora se va a dividir esta sección en dos bloques: transmisión y recepción.

Transmisión: ya configurado el módulo se define la función que se realizará cuando la transmisión o recepción de un paquete se haya completado, alertada por DIO0.

lora.setDio0Action(setFlag);

setFlag es una función que asigna true a una bandera si se hizo la transmisión.

Finalmente se procede a realizar una simple transmisión y reusando la variable de estado se consulta si fue exitoso, comparándolo con ERR_NONE.

state = lora.startTransmit("Hello World!");

Hasta aquí el setup para una transmisión. Ya en el loop para comenzar se debe verificar que el mensaje de prueba se haya realizado chequeando que la bandera tenga asignado el valor true. Luego simplemente es usar starTransmit, verificar la bandera y la variable de estado.

Recepción: se vuelve al punto donde se setea el DIO0 con la bandera. Aquí la bandera se pone en true cuando llega un paquete. Para recibir la función es lora.startReceive(). En el caso de querer leer la información proveniente como una cadena de caracteres usamos lora.readData(variable). La variable es un String. También puede leer como un arreglo en forma de byte lora.readData(byte,8). Donde byte es un vector de tamaño 8. Esta función puede almacenarse en una variable de estado para compararse con ERR_NONE.

Otras funciones de características simplemente demostrativas son:

lora.getRSSI(); Indicador de la Fuerza de la señal recibida en dBm

lora.getSNR(); Indicador de la relación señal a ruido en dB

Existen muchas funciones más, pero como se dijo previamente no serán necesarias para las pruebas que se debían realizar.

C.6) Observaciones en los circuitos adaptadores de señal

En la realización de los anteriores circuitos se prestaron una serie de inconvenientes los cuales algunos son nombrados a continuación con posibles soluciones llevadas a cabo.

Al momento de la colocación del Arduino en la placa madre, no presentaba un buen calce debido al conector de alimentación del mismo, dicho problema fue sencillo de solucionar debido a que simplemente se requirió desoldar el conector.

Un posible problema que pude llegar a presentarse, pero que en el actual proyecto no se presentó, es la intolerancia del sistema al ruido electrónico, esto principalmente en un ambiente industrial o similar, en caso de presentarse algún inconveniente de este tipo existen diferente técnicas y soluciones que se pueden emplear, las mismas pueden consistir en buen blindaje sumado a un plano de masa, la utilización de componentes con tecnología de fabricación más robustos al

ruido, mejoras en el diseño electrónico de las placas empleadas, mallados en el conexionado, etc.

En el caso del adaptador de voltaje analógico se presentó una desviación en las medidas esperadas a las obtenidas, se observó luego de múltiples mediciones una variación en la pendiente de la recta de respuesta, esto se produce debido a la tolerancia presente en los componentes empleados, confeccionando una tabla de valores del circuito se obtiene lo siguiente:

Valor de entrada [V]	Valor de salida obtenido[V]	Valor de salida esperado [V]
0,00	0,00	0,00
1,20	0,13	0,38
2,10	0,32	0,66
3,10	0,52	0,97
4,00	0,75	1,25
5,10	1,06	1,59
6,10	1,35	1,91
7,00	1,64	2,19
8,10	1,99	2,53
9,10	2,30	2,84
10,20	2,65	3,19

Tabla 6. Datos del adaptador de entrada analógica

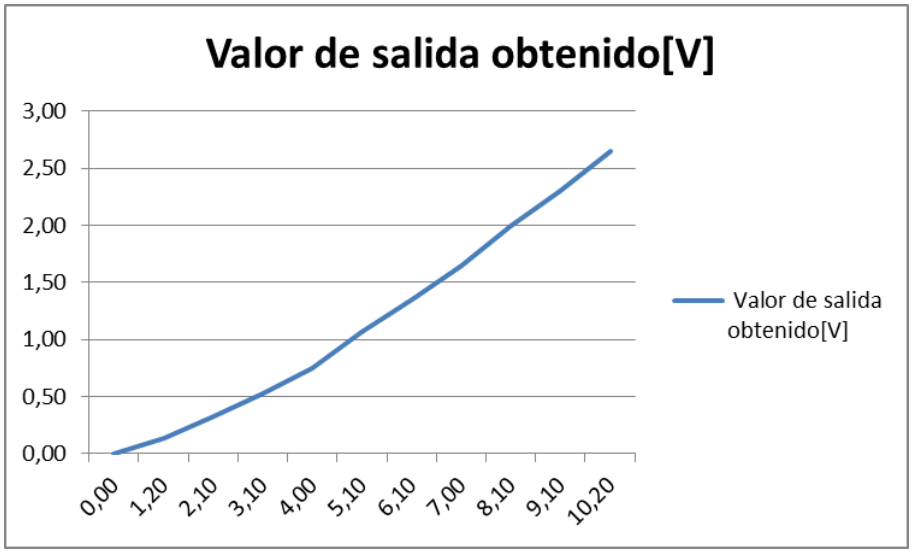


Imagen 56. Gráfica de datos analógicos adquiridos

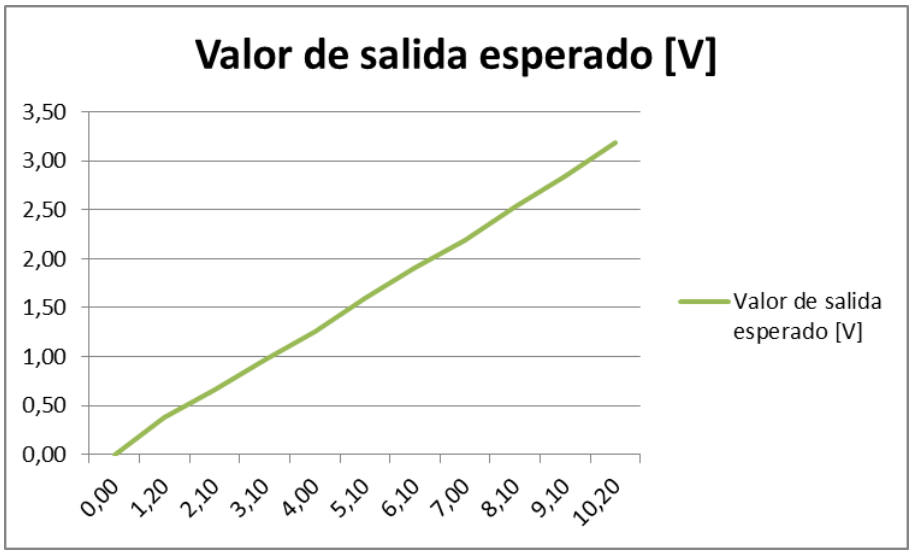


Imagen 57. Gráfica de datos analógicos esperados

Claramente en los datos recolectados se observa la discrepancia, sin embargo, existen soluciones sencillas para el actual problema, una de ellas podría ser una corrección modificando el hardware, la misma constaría con una resistencia variable afectando al divisor resistivo, una vez adicionada debería calibrarse hasta obtener una pendiente buscada, el problema es que las resistencias variables también son afectadas por una gran tolerancia y no sería una solución recomendada. La solución optada y propuesta a implementar es una corrección por software, aprovechando el procesamiento del Arduino, se aplica una corrección a los datos de entrada mediante una tabla interna de valores.

D) Desarrollo del Software

Para realizar los códigos de prueba en las tres comunicaciones siempre se consideró Half Duplex, es decir, bidireccional pero no simultáneo. En los tres casos tenemos que enviar información desde el Arduino y esperar que el receptor la procese y nos responda. En modbus se desarrollaron códigos tanto para maestro como para esclavo. Si el Arduino funciona como maestro le solicita información a un esclavo y este responde. En caso inverso una máquina puede solicitar información al Due. Para Lora Wifi las pruebas se remiten a enviar una cadena de texto. Para ello se cuenta con dos SX1278, donde uno emite un mensaje y el otro lo recibe comprobando que llegó sin problemas.

E) Diseño Completo

Debido a que el presente trabajo es simplemente un desarrollo general de las posibles aplicaciones que podrían llevarse a cabo, el mismo para su puesta a prueba completa contará con una de cada una de sus características o funciones previamente analizadas, de este modo se podrá comprobar que todas sus

funciones, de cada tipo, no tendrán inconvenientes a la hora de realizar una implementación con cualquiera de ellas y en la cantidad que cada futura aplicación lo requiera. Se entiende en base a lo mencionado que, posteriormente dependiendo el problema a solucionar, será determinación de la persona que arme el kit elegir correctamente que circuitos y código emplear para la correcta solución

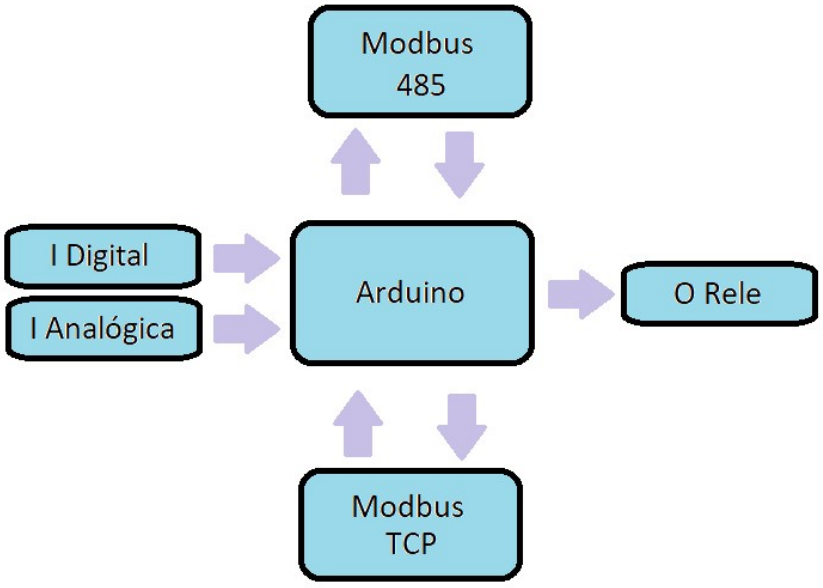


Imagen 58. Diagrama en bloques del kit completo

Prestaciones

a) Comunicación vía modbus RS485 vía conector RJ11 o bornera de conexión (A,B): El mismo puede ser empleado para conectarse a un bus modbus y trabajar como maestro o esclavo. En el caso del actual proyecto será empleado para solicitarle datos a un variador de velocidad.

b) Comunicación vía modbus TCP vía conector RJ45: El mismo puede ser empleado para conectarse a un bus modbus y trabajar como maestro o esclavo, empleando las funciones que se comprenda sean necesarias en la comunicación de diferentes dispositivos, en el caso del actual proyecto será empleado para enviarle datos a un sistema emulando un SCADA.

c) Entrada digital aislada: La misma puede ser empleada para cualquier señal discreta que se desee tratar, en este caso emulará un botón de parada de emergencia el cual deberá ingresar una señal de 5[V] para que el sistema pueda funcionar normalmente

d) Entrada analógica: Empleada en la conversión de un valor de voltaje analógico que puede significar diferentes variables físicas, en este caso se tratará de una señal de 0 a 10[V] la cual estará representando la temperatura que presentará el motor manejado por el variador de velocidad

e) Salida a Relé puede ser empleada en cualquier circunstancia que se desee cerrar cualquier circuito en cuanto no superé las características del rele, en el diseño final sirve para conectar un contacto de entrada de un plc dispuesto en el tablero.

f) Prestaciones propias del Arduino: cuenta con todas sus funciones y bondades en cuanto no estén solicitadas ya por los anteriores puntos mencionados. En el presente proyecto no serán empleados

El código que utiliza el Arduino puede manejarse sin problemas. Uno de los tantos objetivos del proyecto es que toda aplicación sea flexible y moldeable y de igual manera el kit y su respectiva programación.

No es difícil ni sencillo manejar todo a la vez, simplemente es cuestión de conocer las bases de las librerías que se usan. Se usó el dispositivo como maestro Modbus tanto para RS485 como TCP. En el primer caso se solicita información a un variador de velocidad altivar31. Esa información se encuentra en holding registers con direcciones específicas que pueden observarse en el manual de variables de comunicación. A su vez, se deben ajustar las características de comunicación para que el enlace sea sin inconvenientes. En este caso el SERIAL 8N1. Para conectar el altivar disponer de una entrada RJ45 donde la información que se precisa proviene solo de dos de los ocho cables. Por ende, se usó un cable ethernet donde un extremo tiene ficha y el otro los cables con A y B para el Arduino.

En cuanto a la programación del Modbus RS485 como esclavo, es tal cual se vió. Se declaran variables básicas y se configuran los paquetes para leer registros en nuestro caso. También la configuración de comunicación. Luego en el loop simplemente se administra el vector de información. De este último nos va a interesar lo que obtenemos del variador porque será comunicado por modbus TCP IP a una computadora que simula de esclavo. Entonces podemos decir que en RS485 solo realizamos lectura y en TCP IP solo escritura.

Para mandar la información a la computadora, se realiza la configuración ya explicada y se escribe en registros del esclavo PC la información que se extra del vector de información de RS485. Para enlazar se usa un cable UTP derecho.

Lo último que quedaría explicar es como se manejan las placas. Sencillamente se definen y configuran los pines que usamos como entradas analógica y digital. De igual manera para la salida por relé. Para activar el relé la operación es simple. Por entrada analógica al superar un voltaje la salida se activa. En caso que baje a un cierto valor se pone en off la salida. Si se detecta señal en la entrada digital

automáticamente se activa la salida, mientras que cuando no se detecte el relé no va a funcionar. El sistema es simple, porque la entrada analógica simula un sensor de temperatura, es decir, una automatización donde si superamos un valor el relé se activa para que se corte el sistema. La entrada digital sería en el mejor de los casos una acción manual, un pulsador de emergencia. De igual manera la sección de código de las placas adaptadoras está programada con una lógica para que la salida se encuentre activada por consecuencia de una sola entrada. Mientras que, si el relé está activo, la única forma de apagarlo es que no se detecte nada en ambas entradas. Distinto a la forma de encenderlo, donde con solo una entrada ya funciona. Esto significa que, si se supera una temperatura y también se pulsa el accionador de emergencia, el sistema no va entrar en error.

Para diseñar el código se segmenta en los tres sectores de los cuales ya hemos hablado, pero a su vez, dividimos estos en tres subsectores más dedicados a las placas y las dos comunicaciones. La programación es secuencial, y empieza con la consulta de detección de señal en las placas. Seguidamente se actualiza el flujo de datos por RS485 y se adquiere la información que se solicita al altivar. Finalmente, esa información se escribe por TCP IP al esclavo PC. Es un ciclo sin fin, donde si bien el relé podría cortar el sistema, las comunicaciones van a seguir funcionando y mandando solicitudes de escritura y lectura.

Capítulo 3: Resultados

En lo que respecta a las pruebas realizadas con el tablero no se presentaron problemas y se obtuvo un funcionamiento correcto, gracias a ello se puede concluir que debido a la sencillez presentada para su construcción y la accesibilidad económica que presenta es una gran opción para campos de funcionamiento como el evaluado. El producto termina presento el siguiente aspecto:

Imagen 59. Vista posterior del gabinete terminado

Imagen 60. Viste superior del gabinete terminado

Se puede apreciar que el gabinete impreso y montado presenta gran aproximación a lo buscado, presentado en su parte posterior las interfaces de comunicación Modbus, mientras que en su parte superior presenta borneras para el conexionado de las señales externas, tantas las de alimentación, como las de entrada y salida.

Visto de frente y enumerando de izquierda hacia la derecha las borneras presentarán las siguientes funciones:

Bornera número	Función
0	Entrada analógica 0-10[V]
1	Común de Relé
2	Normal abierto de Relé
3	Normal cerrado de Relé
4	Pin digital del Arduino D11
5	Pin digital del Arduino D6
6	Pin digital del Arduino D2
7	Pin digital del Arduino D7
8	Pin digital del Arduino D10
9	GND
10	Alimentación 24[V]
11	Modbus B
12	Modbus A
13	Pin digital del Arduino D52
14	Entrada digital 5[V]

Tabla 7. Conexionado del kit

Internamente la entrada analógica de 0-10[V] se conecta a la entrada de la placa adaptadora de voltaje analógico para posteriormente conectarse en el pin analógico 0 del Arduino. La entrada digital de 5[V] posterior al circuito adaptador de señal digital se conecta con el pin digital D3 del Arduino. Por último, el pin digital D38 del Arduino se conecta al circuito adaptador de señal para activar la salida que contiene el Relé.

A continuación, se muestran imágenes de la prueba realizada con el variador de velocidad conectado al kit.

Imagen 61. Variador de velocidad probado

Imagen 62. Motor supervisado

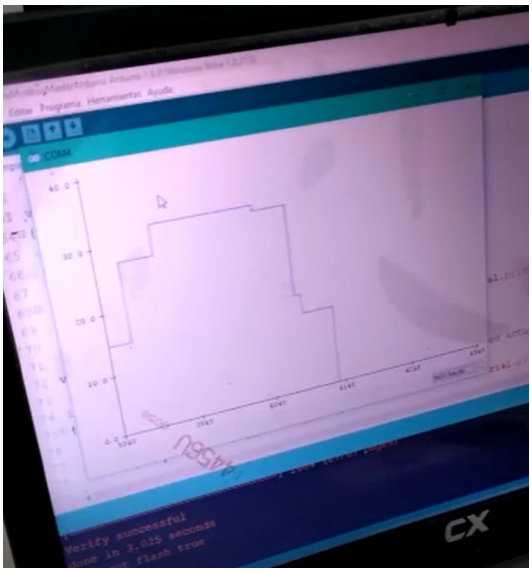


Imagen 63. Gráfica de velocidad supervisada

Las imágenes anteriores exponen la conexión realizada al variador de velocidad Altivar 31, el motor de 1HP que maneja y como es visualizada la velocidad del motor en [Hz] en una PC emulando un sistema de recepción, dicha velocidad es extraída de un registro interno del variador de velocidad mediante Modbus 485, demostrando que es lo suficientemente robusto el kit empelado para soportar una aplicación básica de nivel Industrial.

Capítulo 4: Análisis de Costos

Debido al hecho de ser un desarrollo muy flexible del kit, el costo del trabajo dependerá de la aplicación a satisfacer, es por ello que posteriormente se detallara el costo aproximado que presenta cada factor, ya sea fijo o variable, presente para el desarrollo del kit de manera unitaria. Dependiendo los factores empleados y su cantidad se desarrollará un costo. Otro aspecto importante a tener en cuenta es que al ser un proyecto en busca del empleo universal, presentando un aspecto flexible y abierto, el software empleado y el diseño del sistema no presentara costo y serán obviados.

Las estimaciones de precios por unidad en base a lo desarrollado en el presente trabajo son los siguientes

Para el desarrollo de la placa madre:

Placa de cobre virgen simple faz	237,00 pesos
Bornera	7,70 pesos
Pulsador reductor LM2596	117,00 pesos
Pines machos pack 40 unidades	20,90 pesos
Pines hembra pack 40 unidades	28 pesos
Arduino DUE	2228 pesos
Arduino UNO	669 pesos

Módulos probados:

Mini Ethernet Shield W5100	656 pesos
Lora SX1278	649 pesos
Conversor RS485 SP3485	275 pesos

Placa conversora DC-DC:

Bornera	7,70 pesos
Placa de cobre virgen simple faz	15 pesos
Optoacoplador 4N25	17,40 pesos
Resistencias	0,70 pesos

Placa conversora AC-AC:

Bornera	7,70 pesos
Placa de cobre virgen simple faz	15 pesos
Resistencias	0,70 pesos
Operacional LM393	26,00 pesos

Placa salida a relé:

Bornera	7,70 pesos
Placa de cobre virgen simple faz	15 pesos
Resistencias	0,70 pesos
Transistor 2N3904	7,00 pesos
Diodo 1N4148	3,00 pesos
Relé simple inversor	39,50 pesos

Hora de trabajo	227 pesos
Hora de impresión 3D	150 pesos
Estaño	63 pesos

Se estima que, en gastos adicionales como cables, tornillos, tuercas, servicios, etc, alrededor de un 10% adicional a la suma del total de los gastos anteriores.

Teniendo en cuenta lo anterior descrito se procederá a contrastar un producto comercial contra la implementación del kit desarrollado, en base un cierto problema o trabajo a solucionar, de este modo se podrán extraer conclusiones en lo que respecta a lo económico y funcional.

Se toma como ejemplo una aplicación en la cual se necesita sensar la temperatura de un motor trifásico, en caso que se exceda cierto nivel de temperatura el sistema debe interrumpirse, además se debe disponer de un interruptor de emergencia en caso que se desee detener el funcionamiento de forma manual. El dispositivo debe tomar datos mediante RS485 del variador de velocidad del motor y enviarlos vía modbus TCP hacia un maestro que emula un sistema SCADA.

Para el caso anterior nombrado actualmente en el mercado existen diferentes alternativas, por similitud al kit a emplear y adaptación al problema se contrastará con un PLC Arduino ARDBOX 20 I/Os Analog HF Modbus, actualmente al momento de la realización del presente proyecto el dispositivo cuenta 141,75 euros, a los cuales debe sumarse el costo de envió e impuestos, lo que da un estimado de 10580 pesos aproximadamente en el mejor de los casos.

Realizando un detalle de costos para el problema requerido se obtiene lo siguiente:

Factores	Costo Unitario en pesos	Cantidad	Costo total
Placa de cobre virgen simple faz	277,00	1,00	277,00
Bornera	7,70	25,00	192,50
LM2596	117,00	1,00	117,00
Pines machos pack 40 unidades	20,90	1,00	20,90
Pines hembra pack 40 unidades	28,00	1,00	28,00
Arduino DUE	2228,00	1,00	2228,00
Mini Ethernet Shield W5100	656,00	1,00	656,00
Conversor RS485 SP3485	275,00	1,00	275,00
Optoacoplador 4N25	17,40	1,00	17,40
Resistencias	0,70	16,00	11,20
Operacional LM393	26,00	1,00	26,00
Transistor 2N3904	7,00	1,00	7,00
Diodo 1N4148	3,00	1,00	3,00
Rele simple inversor	39,50	1,00	39,50
Hora de trabajo	250,00	8,00	2000,00
Hora de impresión 3D	70,00	43,00	3010,00
Estaño	63,00	1,00	63,00
Gastos varios	10 % de lo anterior	1,00	897,15
Total			9868,65

Tabla 8. Costos de aplicación implementada

El kit desarrollado en el actual proyecto es ideal en un sector de mercado el en el cual se requiera realizar el control y automatización de pequeños trabajos, el hecho de contar con flexibilidad a la hora de su implementación hace posible la competencia con productos existentes en el mercado. Con ventaja de ser un producto open source y posee una capacidad adaptativa si se requiere una modificación del mismo para su utilización en un proyecto distinto y se quiere seguir empleando el mismo producto.

Como observación cabe destacar que en el presente proyecto, las horas de trabajo del diseño realizado teniendo en cuenta el tiempo empleado en el desarrollo e inconvenientes en el transcurso de la realización del mismo, se estima que se emplearon un aproximado de 50 a 70 horas de trabajo, este costo adicional debería tenerse en cuenta a la hora de realizar la venta del diseño mismo, pero debido al carácter libre que posee el proyecto estos gastos serán obviados para el uso de cualquier persona interesada.

Capítulo 5: Discusión y Conclusión.

En base a lo analizado durante el desarrollo del proyecto se puede observar las bondades y defectos que presenta sobre el sector del mercado a intervenir, como características principales se puede decir que presenta ventajas en cuanto a su flexibilidad, precio, orientación a lo universal y abierto. Mientras que como desventaja podría decirse que es de dimensiones superiores a los dispositivos en el mercado, se encuentra en etapas de desarrollo y por ello mismo existen diferentes áreas en las cuales se puede mejorar notoriamente su desempeño.

En principio el proyecto estaba orientado a la integración de la mayor cantidad de protocolos de comunicación, de tal manera que no exista barrera alguna en la comunicación entre diferentes marcas y sistemas del mercado actual en lo que respecta a pequeñas automatizaciones y controles, a medida que se avanza en la investigación se llega a la conclusión que dicha tarea presentaba dificultades que harían prácticamente imposible su implementación, esto es debido que gran parte de los fabricantes de equipos orientados al control prefieren ocultar el modo de funcionamiento del protocolo que emplean sus dispositivos. De este modo se orientó específicamente a los protocolos y sistemas universales actualmente presente. Lo anterior dicho no quita que el proyecto simplemente es un punta pie inicial, es un desarrollo el cual se encuentra abierto para ser retomado y expandido posteriormente, en ese sentido está la posibilidad de ir agregando diferentes sistemas y protocolos en cuanto y tanto sea legal y accesible.

Al momento de realizar el análisis del costo del kit el mismo arrojo resultados esperados, los cuales permitían la competencia con otros productos, tener en cuenta que los mismos fueron pensados como un comprador casual, ¿a qué nos referimos con esto?, que es el precio que obtendría un comprador cualquiera, al por menor, es por ello que para la producción en masa sería aún más viable en lo que respecta a costos, lo mismo con el empleo de tecnologías SMD también se mejoraría la performance y el volumen ocupado. No debe menospreciarse el hecho de poder expandirse o modificarse agregando o cambiando simples circuitos y conexionados, lo que presenta una gran ventaja frente a expansiones o cambios. Lo anterior es posible debido a la sencillez con la cual fue pensado, aunque se necesiten mínimos conocimientos técnicos para su modificación, no dejan de ser mínimos, ya que el desarrollo tanto de hardware como de software están detallados de tal forma que dependiendo lo que se desee simplemente se debe agregar el bloque o circuito ya facilitado en el proyecto, además del desarrollo universal sobre Arduino en la actualidad.

Capítulo 6: Literatura Citada

Información útil:

https://www.xataka.com/makers/empezar-con-arduino-genuino-como-elegir-la-placa-modelos-compatibles-y-kits-de-iniciacion

https://www.arduino.cc/

https://www.industrialshields.com/es_ES/

https://lorawan.es/

https://aprendiendoarduino.wordpress.com

https://www.prometec.net/

https://www.iot-store.com.au

https://automantenimiento.net

https://www.luisllamas.es/arduino-ethernet-shield-w5100/

Datasheets:

http://copperhilltech.com/content/RS485-Board-UserManual.pdf

https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf

https://www.futurashop.it/Allegato_PDF_ENG/2846-DRF1278F.pdf

https://www.egr.msu.edu/eceshop/Parts_Inventory/datasheets/lm741.pdf

https://www.vishay.com/docs/83725/4n25.pdf

http://www.robgray.com/temp/Due-pinout.pdf

https://www.schneider-electric.co.in/en/download/document/1624595/

https://www.se.com/hk/en/download/document/1624597/

https://www.se.com/es/es/download/document/1624589/

Librerías:

http://github.com/andresarmento/modbus-arduino

https://github.com/jecrespo/simple-modbus

https://github.com/luizcantoni/mudbus

https://github.com/goddland16/Modbus-TCP

https://github.com/jgromes/LoRaLib

Capítulo 7: Anexos de Datasheets de componentes y código final

RS485 Board (3.3V/5V) User Manual

1. Description

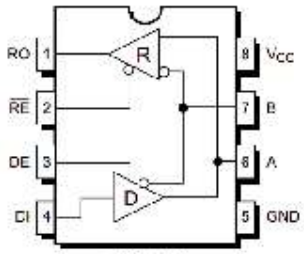
RS-485 is a serial communication protocol standard based on differential signal transmission. Comparing with RS-232 which has the transmission distance limit of 15m, RS485 can be used effectively over long distances and is widely used in many industrial applications.

RS-485 communication is a half-duplex communication using a twist pair to transmit differential signal, so it requires data receive and transmission modes switch.

- The RS485 Board is an accessory board used for adding the RS485 transceiver to your application board.
- SP485 / MAX485 are RS485 transceivers requiring 5V power supply.
- SP3485 / MAX3485 are RS485 transceivers requiring 3.3V power supply.

2. Hardware description

2.1 Pin function



Top View

SP485/MAX485/SP3485/MAX3485
Pin layout (Top view)

Pin	Name	Description
1	RO	Receiver Output
2	$\overline{RE}$	Receiver Output Enable Active LOW
3	DE	Driver Output Enable Active HIGH
4	DI	Driver Input
5	GND	Ground Connection
6	A	Driver Output/Receiver Input. Non-inverting
7	B	Driver Output/Receiver Input. Inverting
8	V_{CC}	V_{CC} < 5.25V

SP485 / MAX485 are RS485 transceivers requiring 5V power supply.
SP3485 / MAX3485 are RS485 transceivers requiring 3.3V power supply.

WIZnet

W5100 Datasheet

W5100 Datasheet

Version 1.1.6

WIZnet

For more information, visit our website at http://www.wiznet.co.kr

DRF1278F
20dBm LoRa Long Range RF Front-end Module V1.11

Features:

- Frequency Range: 433MHz
- Modulation: FSK/GFSK/MSK/LoRa
- SPI Data Interface
- Sensitivity: -139dBm
- Output Power: +20dBm
- Data Rate: <300 kbps
- 127dB dynamic Range RSSI
- Excellent blocking immunity
- Preamble detection
- Automatic RF sense and CAD monitor
- Built-in bit synchronizer for clock recovery
- Packet engine up to 256 bytes with CRC
- Working Temperature: -40°C ~+80°C
- Build-in temperature sensor
- Standby current: ≤ 1uA
- Supply voltage: 1.8~3.6V

Applications

- Remote Control
- Smart metering
- Home Automation
- Personal data logger
- Wireless sensor network
- Remote keyless entry
- Wireless PC peripherals

DESCRIPTION

DRF1278F is a type of low cost RF front-end transceiver module based on SX1278 from Semtech Corporation. It keeps the advantages of RFIC SX1278 but simplifies the circuit design. The high sensitivity (-139dBm) in LoRa modulation and 20dBm high power output make the module suitable for low range and low data rate applications.

DRF1278F module consists of RFIC SX1278, thin SMD crystal and antenna matching circuit. The antenna port is well matched to standard 50 Ohm impedance. Users don't need to spend time in RF circuit design and choose suitable antennas for different applications. DRF1278F operates at 1.8~3.6V with extra low standby current which makes it suitable for battery powered-up applications. Because DRF1278F is purely hardware module and it adopts ±10ppm crystal which the resolution of it places a important role in calculating spreading factor, bandwidth, etc. Users need to read the datasheet of SX1278 carefully in order to use the module in the best performance. DORJI also provides sample codes based on microcontroller and Arduino platform. The customers who buy the modules can contact the sales of DORJI for modifiable copies.

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

LM741

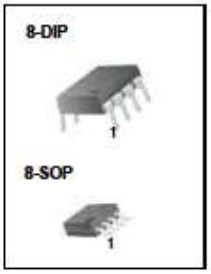
Single Operational Amplifier

Features

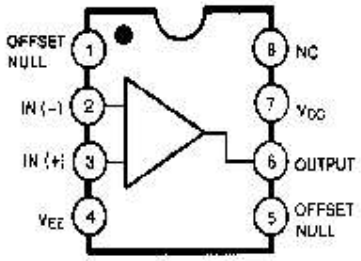
- Short circuit protection
- Excellent temperature stability
- Internal frequency compensation
- High Input voltage range
- Null of offset

Description

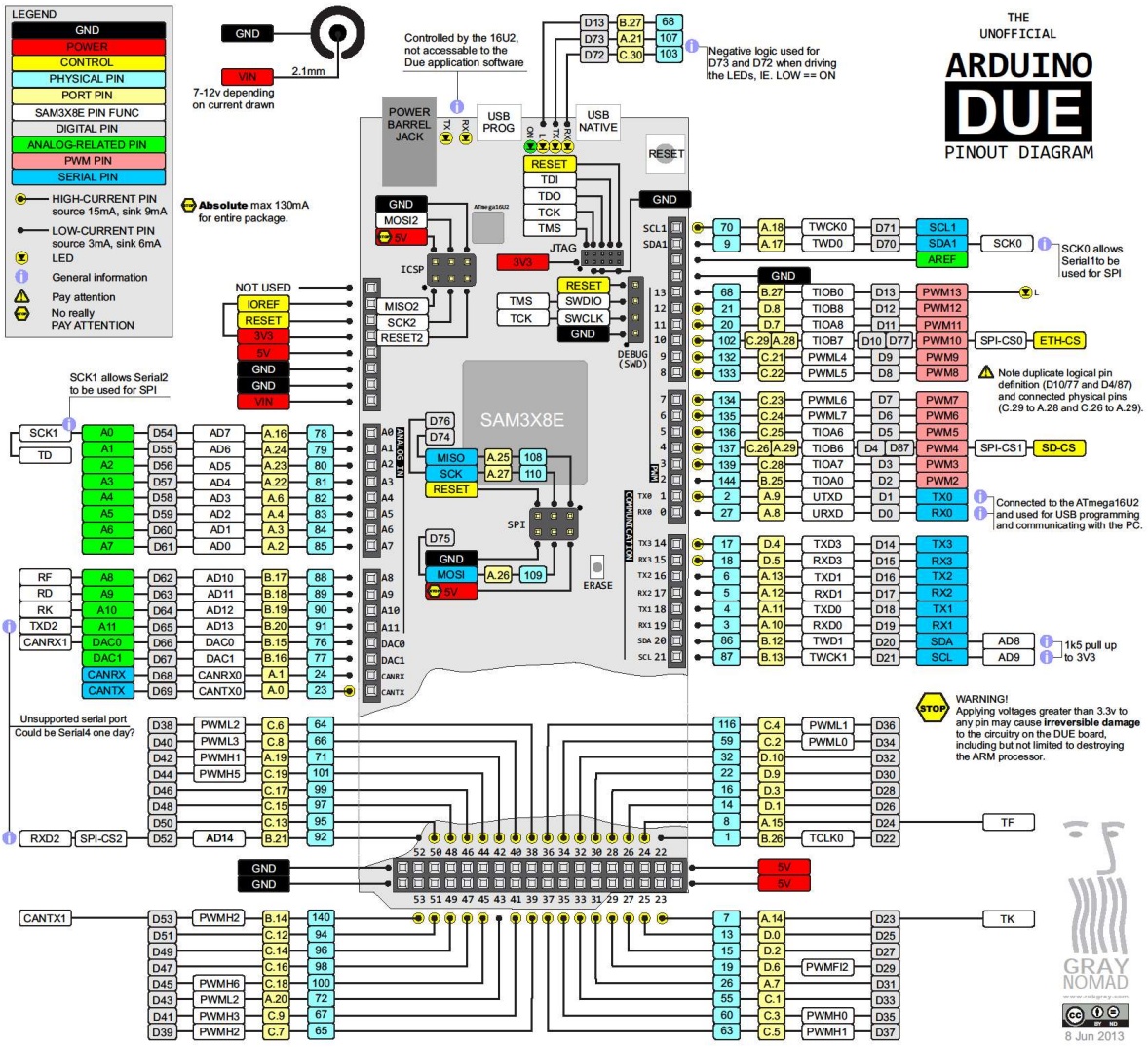
The LM741 series are general purpose operational amplifiers. It is intended for a wide range of analog applications. The high gain and wide range of operating voltage provide superior performance in intergrator, summing amplifier, and general feedback applications.



Internal Block Diagram



Rev. 1.0.1


THE UNOFFICIAL
ARDUINO DUE
PINOUT DIAGRAM
LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
SAM3X8E PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
HIGH-CURRENT PIN source 15mA, sink 9mA
LOW-CURRENT PIN source 3mA, sink 6mA
LED
General information
Pay attention
No really PAY ATTENTION
Absolute max 130mA for entire package.
7-12v depending on current drawn
Controlled by the 16U2, not accessable to the Due application software
Negative logic used for D73 and D72 when driving the LEDs, IE. LOW == ON
POWER BARREL JACK
USB PROG
USB NATIVE
RESET
JTAG
ICSP
DEBUG (SWD)
SAM3X8E
SPI
ERASE
NOT USED
SCK1 allows Serial2 to be used for SPI
SCK0 allows Serial1to be used for SPI
Note duplicate logical pin definition (D10/77 and D4/87) and connected physical pins (C.29 to A.28 and C.26 to A.29).
Connected to the ATmega16U2 and used for USB programming and communicating with the PC.
1k5 pull up to 3V3
Unsupported serial port
Could be Serial4 one day?
WARNING!
Applying voltages greater than 3.3v to any pin may cause irreversible damage to the circuitry on the DUE board, including but not limited to destroying the ARM processor.
GRAY NOMAD
8 Jun 2013

ESPAÑOL

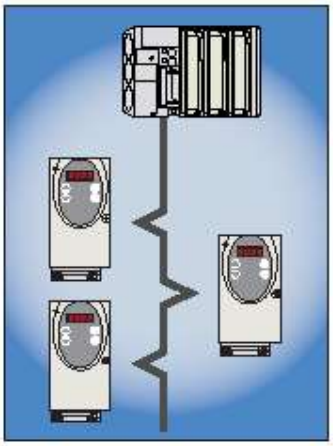
Altivar 31

Communication variables

User's manual

Software V3.7

10/2009



1524695

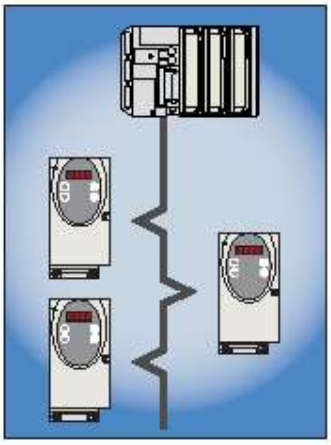
www.schneider-electric.com

Schneider Electric

Altivar 31
Modbus

User's manual

11/2009



1624597

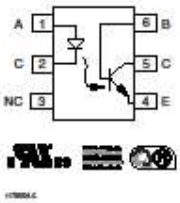
www.schneider-electric.com

Schneider Electric

4N25, 4N26, 4N27, 4N28

Vishay Semiconductors

Optocoupler, Phototransistor Output, with Base Connection



FEATURES

- Isolation test voltage 5000 V_{RMS}
- Interfaces with common logic families
- Input-output coupling capacitance < 0.5 pF
- Industry standard dual-in-line 6 pin package
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

APPLICATIONS

- AC mains detection
- Reed relay driving
- Switch mode power supply feedback
- Telephone ring detection
- Logic ground isolation
- Logic coupling with high frequency noise rejection

AGENCY APPROVALS

- UL1577, file no. E52744
- BSI: EN 60065:2002, EN 60950:2000
- FIMKO: EN 60950, EN 60065, EN 60335

DESCRIPTION

The 4N25 family is an industry standard single channel phototransistor coupler. This family includes the 4N25, 4N26, 4N27, 4N28. Each optocoupler consists of gallium arsenide infrared LED and a silicon NPN phototransistor.

ORDER INFORMATION	
PART	**REMARKS**
4N25	CTR > 20 %, DIP-6
4N26	CTR > 20 %, DIP-6
4N27	CTR > 10 %, DIP-6
4N28	CTR > 10 %, DIP-6

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Surge current	t ≤ 10 µs	I_{FSM}	3	A
Power dissipation		P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	t ≤ 1 ms	I_C	100	mA
Power dissipation		P_{diss}	150	mW

```
//CODIGO FINAL
//DEFINICIONES RS485
#include <SimpleModbusMaster.h>

//DEFINICIONES TCP
#define WIZNET_W5100 1
unsigned int param_value_int[7];
#include <Ethernet.h>
#include <SPI.h>
#include <ModbusTCP.h>

//-------PARTE RTU 485 ------------------------------------
#define baud 9600
#define timeout 1000
#define polling 200
#define retry_count 10
#define TxEnablePin 5
#define TOTAL_NO_OF_REGISTERS 10
#define TOTAL_NO_OF_PACKETS 10
enum
{
  PACKET1,
  PACKET2,
  PACKET3,
};
Packet packets[TOTAL_NO_OF_PACKETS];
unsigned int regs[TOTAL_NO_OF_REGISTERS];
```

```
//-------PARTE TCP IP ------------------------------------
IPAddress ModbusDeviceIP(192, 168, 1, 1);
IPAddress moduleIPAddress(192, 168 , 1, 2);
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
 ModbusTCP node(1);

void setup()
{
//-----------PARTE PLACAS----------------------------
pinMode(3,INPUT);
pinMode(38,OUTPUT);
digitalWrite(38,LOW);

//-----------PARTE MODBUS RTU 485 -------------------------
  Serial.begin(9600);

  modbus_construct(&packets[PACKET1], 1, READ_HOLDING_REGISTERS, 3205, 1, 1);
        //torque motor
  modbus_construct(&packets[PACKET2], 1, READ_HOLDING_REGISTERS, 3207, 1, 2);
        //volt linea
  modbus_construct(&packets[PACKET3], 1, READ_HOLDING_REGISTERS, 8604, 1, 3);
        //vel (const 0.018)

  modbus_configure(&Serial1, baud, SERIAL_8N1, timeout, polling, retry_count,
        TxEnablePin, packets, TOTAL_NO_OF_PACKETS, regs);

//-----------PARTE MODBUS TCP IP----------------------------------
Ethernet.begin(mac, moduleIPAddress);
  Serial.println("Ethernet interface started");
```

```
Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++)
  {Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print("."); }

  node.setServerIPAddress(ModbusDeviceIP);
  delay(3000);
}

void loop()
{
  //------------------PARTE RTU 485
  modbus_update();

  //---------PARTE PLACAS
  int sdg;
  float enanvalue = 0;
  float anaux = 0;

  for (int i = 0; i <= 200; i++)
    {anaux=analogRead(A0);
     enanvalue=enanvalue+anaux;}
  enanvalue=enanvalue/200;

  sdg=digitalRead(3);

  if (sdg==LOW)
    {digitalWrite(38,HIGH);}
  if (sdg==HIGH)
    {digitalWrite(38,LOW);}
```

```
  if (enanvalue>100)
    {digitalWrite(38,HIGH);}
  if (enanvalue<80 && sdg==HIGH)
    {digitalWrite(38,LOW);}

  //------------------PARTE RTU 485
  modbus_update();

//---------PARTE TCP IP---------------------------------------

  node.writeSingleRegister(5, regs[1]);
  node.writeSingleRegister(6, regs[2]);
  node.writeSingleRegister(7, regs[3]);
}
```