Alumnos: Leonardo Gonzalez - Juan Diego Gonzalez Antoniazzi - Andres Montaño - Rodolfo Paz - Carolina Nis - Fernando Vexenat

Shields v1.0

Bienvenido a las practicas con sistemas embebidos, en la misma desarrollaremos la ejercitación del lenguaje CPP para sistemas embebidoscon VsCode @ PlatformIO, con el framework de Arduino.

La modalidad será la siguiente:

Cada practica se desarrollará en forma grupal, debiendo subir el desarrollo de la misma al repositorio (respetando la estructura de monorepositorio) establecido por grupo. Los ejercicios serán implementados de forma que a cada integrante le corresponda 1 o más tareas (issues); por lo que deberán crear el proyecto correspondiente, con la documentación asociada si hicierafalta, y asignar los issues por integrante. De esta forma quedara documentada la colaboración de cada alumno.

Ejercicio #1

a) Explique el funcionamiento del protocolo I2C?

I2C es un puerto y protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA y SCL. Además el protocolo permite conectar hasta 127 dispositivos esclavos con esas dos líneas, con hasta velocidades de 100, 400 y 1000 kbits/s. También es conocido como IIC ó TWI – Two Wire Interface.

El protocolo I2C es uno de los más utilizados para comunicarse con sensores digitales, ya que a diferencia del puerto Serial, su arquitectura permite tener una confirmación de los datos recibidos, dentro de la misma trama, entre otras ventajas.

La conexión de tantos dispositivos al mismo bus, es una de las principales ventajas. Además, si comparamos a I2C con otro protocolo serial, como Serial TTL, este incluye más bits en su trama de comunicación que permite enviar mensajes más completos y detallados.

Los mensajes que se envían mediante un puerto I2C, incluye además del byte de información, una dirección tanto del registro como del sensor. Para la información que se envía siempre existe una confirmación de recepción por parte del dispositivo. Por esta razón es bueno diferenciar a los distintos elementos involucrados en este tipo de comunicación.

I2C – Esquema de comunicación y elementos

Siempre que hablamos de una comunicación oral, se entiende que es entre dos o más personas. Como consecuencia podemos también indicar que en una comunicación digital existen distintos dispositivos o elementos. En el caso de I2C se diferencian dos elementos básicos, un MAESTRO y un ESCLAVO. La Figura-1, muestra una conexión típica de tres dispositivos, el bus consiste de dos líneas llamadas, Serial DAta – SDA y Serial CLock – SCL. Es decir, Datos Seriales y Reloj Serial. En particular al bus se le conectan dos resistencias en arreglo pull-up, de entre 2.2K y 10K.

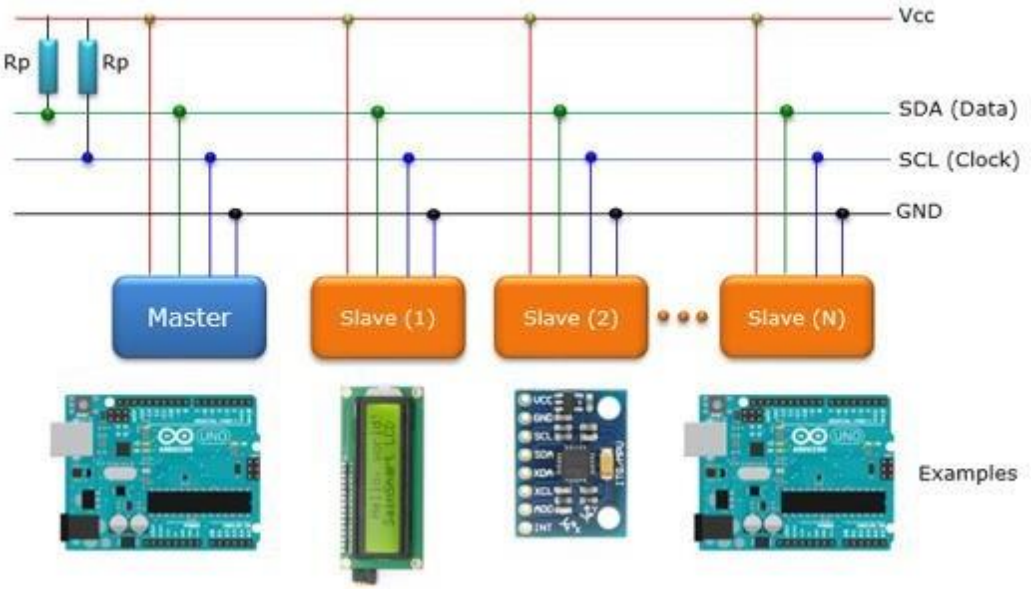


Figura-1. Conexión de tres dispositivos a un bus de comunicación I2C.

El MAESTRO I2C se encarga de controlar al cable de reloj, por sus siglas en inglés llamada SCL – Serial CLock. Además, el MAESTRO se encarga de iniciar y parar la comunicación. La información binaria serial se envía sólo por la línea o cable de datos seriales, en inglés se llama SDA – Serial DAta. Dos Maestros no pueden hacer uso de un mismo puerto I2C. Puede funcionar de dos maneras, como maestro-transmisor o maestro-receptor. Sus funciones principales son:

Iniciar la comunicación – S
Enviar 7 bits de dirección – ADDR
Generar 1 bit de Lectura o Escritura – R/W
Enviar 8 bits de dirección de memoria
Transmitir 8 bits de datos –
Confirmar la recepción de datos – ACK – ACKnowledged
Generar confirmación de No-recepción, NACK – No-ACKnowledged
Finalizar la comunicación

El ESCLAVO I2C, generalmente suele ser un sensor. Este elemento suministra de la información de interés al MAESTRO. Puede actuar de dos formas: esclavo-transmisor o esclavo-receptor. Un dispositivo I2C esclavo, no puede generar a la señal SCL. Sus funciones principales son:

Enviar información en paquetes de 8 bits.

Enviar confirmaciones de recepción, llamadas ACK

I2C – Bits de la trama del puerto

El protocolo de comunicación I2C se refiere al conjunto de bits que son necesarios para enviar uno o varios bytes de información. En lo particular, para este protocolo existen los siguientes bits importantes:

Inicio ó Start – S
Parada – P
Confirmación – ACK
No Confirmación – NACK
Lectura-/Escritura – L/W
7 bits para la dirección del dispositivo esclavo/maestro
8 bits de dirección (para algunos sensores pueden ser 16 bits)
8 bits de datos

El conjunto de estos bits y su orden va formando distintas tramas de comunicación. Existen distintos modos de comunicación dependiendo del arreglo de estos bits. Tanto el maestro como el esclavo pueden o no generar los bits anteriores, según los modos de comunicación.

El puerto I2C está disponible si las dos líneas, SDA y SCL están en un nivel lógico alto.

I2C – modos de comunicación

Los modos de comunicación en I2C se refieren a las distintas tramas que pueden formarse en el bus. Estas tramas o modos dependen de, por ejemplo, si queremos leer al sensor esclavo, o si lo queremos configurar. Existen principalmente dos modos de comunicación:

Maestro-Transmisor y Esclavo-Receptor. Este modo se usa cuando se desea configurar un registro del esclavo I2C.
Maestro-Receptor Y Esclavo-Transmisor. Se usa cuando queremos leer información del sensor I2C.

Velocidad del puerto I2C

La velocidad del puerto I2C se refiere al tiempo que le toma al puerto transferir un bit de información. Entonces, este valor se mide en bits/segundo. Típicamente vamos a encontrar la referencia en ciclos/segundo o Herz. Existen tres velocidades estándar, 100Khz, 400Khz y 1Mhz, es decir, 100kbits/s, 400kbits/s y 1000kbits/s. Por ejemplo, para la trama de la Figura-2, ese paquete de datos tiene 29 bits, por lo que a una velocidad de 100kbits/s le tomaría al puerto 0.29 ms enviar la información.

El dato digital o lógica que leerá cada uno de los dispositivos, corresponde el voltaje en los flancos de subida de la señal de reloj – SCL.

Condiciones eléctricas

Cada uno de los bits antes descritos (Inicio-S, ACK, NACK, Parada, RS) significan condiciones eléctricas en el bus I2C. Las condiciones de voltaje son las siguientes:

Inicio. La condición de inicio se genera cuando el bus está disponible, cuando mientras la línea de SCL está en alto (1), existe un flanco de bajada (un cambio de estado lógico de alto a bajo), en la línea de SDA). Este bit sólo lo puede generar el MAESTRO

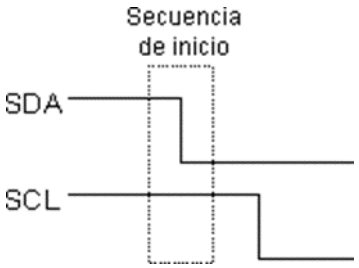


Figura-4. Condición de INICIO-I2C. Estando el bus disponible, hay un flanco de bajada en la línea SDA.

Paro. La señal o bit de PARO se genera cuando hay un flanco de subida en la línea de datos, mientras que la línea de reloj se encuentra en alto. Está condición sólo es posible generarla desde el MAESTRO.

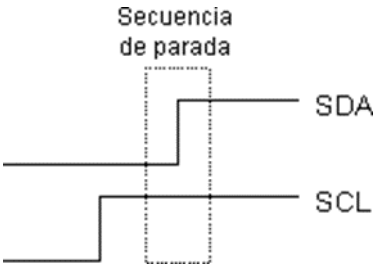


Figura-5. Condición de paro en el puerto I2C.

ACK. Confirmación de recepción. Está condición se crea cuando estando la señal SCL en alto, SDA está en bajo. Está señal la puede generar tanto el MAESTRO como el ESCLAVO.
NACK. Este bit es usado en el esquema de comunicación donde se leen varios bytes de un ESCLAVO en una sola transmisión. Entonces, el bit NACK se usa cuando ya no se quieren recibir más bytes. La condición sólo la puede generar el MAESTRO.

b) Que son los sensores resistivos? Como se conectan a través de un divisor resistivo? Que es el acondicionamiento de señales?

Un sensor resistivo es un dispositivo que convierte cantidades físicas medidas tales como desplazamiento, deformación, fuerza, aceleración, humedad, temperatura, etc. en valores de resistencia. Dispositivo sensor resistivo. Existen principalmente sensores de deformación resistiva, como el tipo de deformación por resistencia, tipo piezoresistivo, resistencia térmica, sensible al calor, sensible al gas y a la humedad. El medidor de deformación en el sensor tiene un efecto de deformación del metal, es decir, deformación mecánica bajo la acción de una fuerza externa, de modo que el valor de resistencia cambia en consecuencia. Hay dos tipos de galgas extensométricas de resistencia: metal y semiconductor. Los medidores de deformación de metal están disponibles en alambre, papel de aluminio y película. Los medidores de tensión de semiconductores tienen las ventajas de una alta sensibilidad (generalmente docenas de veces de seda y papel de aluminio) y pequeños efectos laterales.

Los sensores piezoresistivos son dispositivos fabricados por resistencia a la difusión sobre un sustrato de un material semiconductor de acuerdo con el efecto piezoresistivo del material semiconductor. El sustrato puede usarse directamente como un elemento sensor de medición, y la resistencia de difusión está conectada en forma de un puente en el sustrato. Cuando el sustrato se deforma por una fuerza externa, los valores de resistencia cambiarán y el puente producirá una salida desequilibrada correspondiente. El material del sustrato (o diafragma) utilizado como sensor piezoresistivo es principalmente una oblea de silicio y una oblea de silicio. Los sensores piezoresistivos de silicio hechos de obleas de silicio se están volviendo cada vez más populares, especialmente para los sensores

piezoresistivos de estado sólido que miden la presión y la velocidad. El sensor de termistor mide principalmente la temperatura y los parámetros relacionados con la temperatura utilizando la característica de que el valor de resistencia cambia con la temperatura. Este tipo de sensor es adecuado cuando la precisión de detección de temperatura es relativamente alta. Una gama más amplia de materiales de resistencia térmica son platino, cobre, níquel y similares. Tienen las características de gran coeficiente de resistencia a la temperatura, buena linealidad, rendimiento estable, amplio rango de temperatura y fácil procesamiento. Se utiliza para medir temperaturas en el rango de -200 ° C a 500 ° C.

ACONDICIONAMIENTO DE CIRCUITOS RESISTIVOS

Existen esencialmente 3 formas de acondicionar un sensor resistivo: divisor de tensión, fuente de corriente y puente de Wheatstone. Se describen cada una de estas opciones, señalando algunas ventajas y desventajas de su uso.

Acondicionamiento por divisor de tensión

Probablemente la forma más simple de acondicionar un sensor resistivo es formar un divisor de tensión entre el sensor y un resistor fijo, como se ve en la Figura 6-1, donde R1 es una resistencia de valor fijo y RS Representa el sensor resistivo.

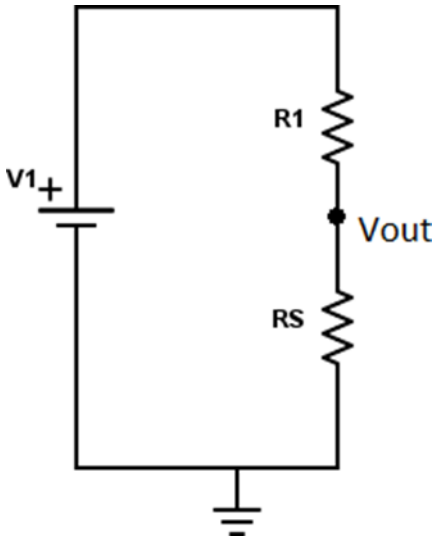


Figura 6-1. Conexión de sensor resistivo por divisor de tensión

Si analizamos este circuito podemos observar que el voltaje Vout está dado por

$Vout = V1\ Rs\ Rs+R1$ Eq. 6.1

De este modo, si el valor de RS varía, el valor de Vout también los hace proporcionalmente.

Evidentemente este circuito es muy simple y económico de implementar, por lo cual puede resultar conveniente. Sin embargo, debemos analizar un poco más sus características para encontrar las consideraciones que debemos tener a la hora de implementarlo y las posibles desventajas de esta configuración.

Relación directa o inversa

En el circuito de la Figura 6-1, vemos que la ecuación nos presenta una relación directa entre el voltaje y a la resistencia, puesto que a mayor resistencia tendremos mayor voltaje. En caso de requerir que la relación sea inversa, bastará con invertir el orden de las resistencias como se aprecia en la Figura 6-2.

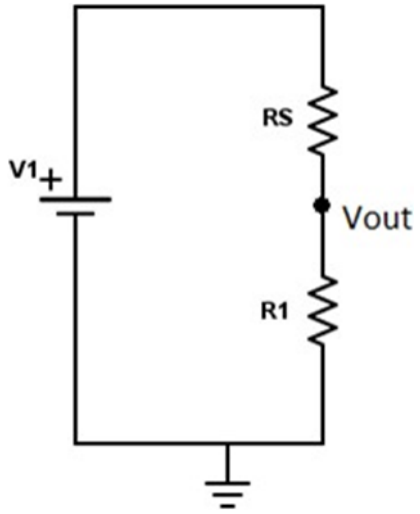


Figura 6-2. Conexión de sensor resistivo por divisor de tensión con relación inversa
En este caso la ecuación para el voltaje Vout sería:

$Vout = V1\ R1\ Rs{+}R1$ Eq. 6.2

Y tendríamos que, a medida que sube la resistencia, baja el voltaje de salida. La elección acerca de cuál de las dos opciones es más conveniente, depende de cuál sea la relación entre la resistencia y la variable a medir (si es directa o inversa) y de la conveniencia o necesidades propias del sistema de medición.

Variación de la resistencia y variación del voltaje

Algunos sensores resistivos presentan una amplia variación de la resistencia con la variable a medir, pero algunos otros tienen una variación de resistencia muy limitada. Para ilustrar esto veamos el ejemplo de dos sensores resistivos diferentes:

Las fotorresistencias son elementos resistivos sensibles a la luz. Cuando una luz incide sobre este elemento, su resistencia se reduce y cuando están en condiciones de oscuridad, la resistencia aumenta. Los valores de una fotorresistencia pueden variar desde un valor de 100Ω a plena luz hasta 1MΩ o más en la oscuridad. La variación exacta depende del elemento específico, dado que existen de varios valores; pero en general tenemos una variación bastante amplia.

La pt100 es un RTD, un tipo de sensor de temperatura muy utilizado que varía su resistencia de acuerdo a la temperatura. Su nombre se deriva de la resistencia que presenta a 0°C (es decir, la resistencia a =°C es de 100Ω). La variación de la resistencia puede consultarse n tablas que se especifican en la base de datos, pero como referencia podemos mencionar que a 100°C la resistencia de la pt100 es de 138.5Ω. ¡En 100 °C la resistencia sólo ha cambiado en 38!5Ω!

Por su puesto, si la variación de la resistencia es pequeña, la variación en el voltaje también lo será

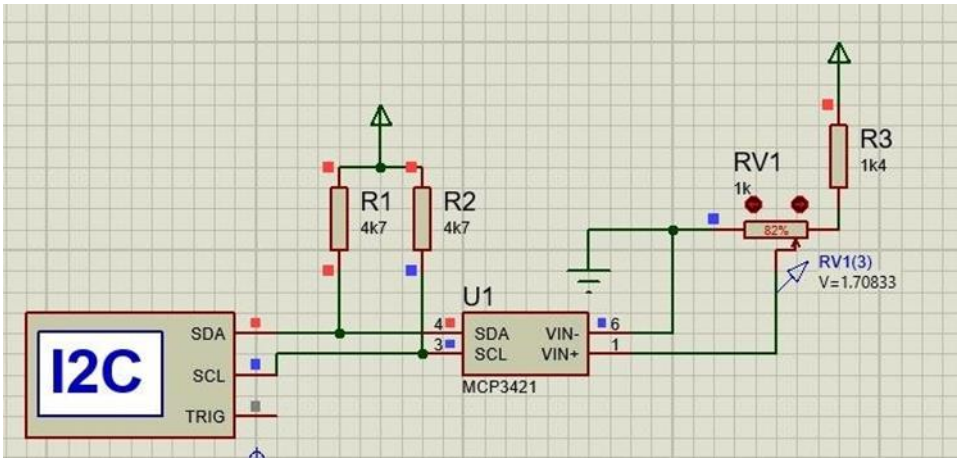
c) Como funciona el integrado mcp3421 y como lo utilizaría para construir una shield de transducción resistiva?

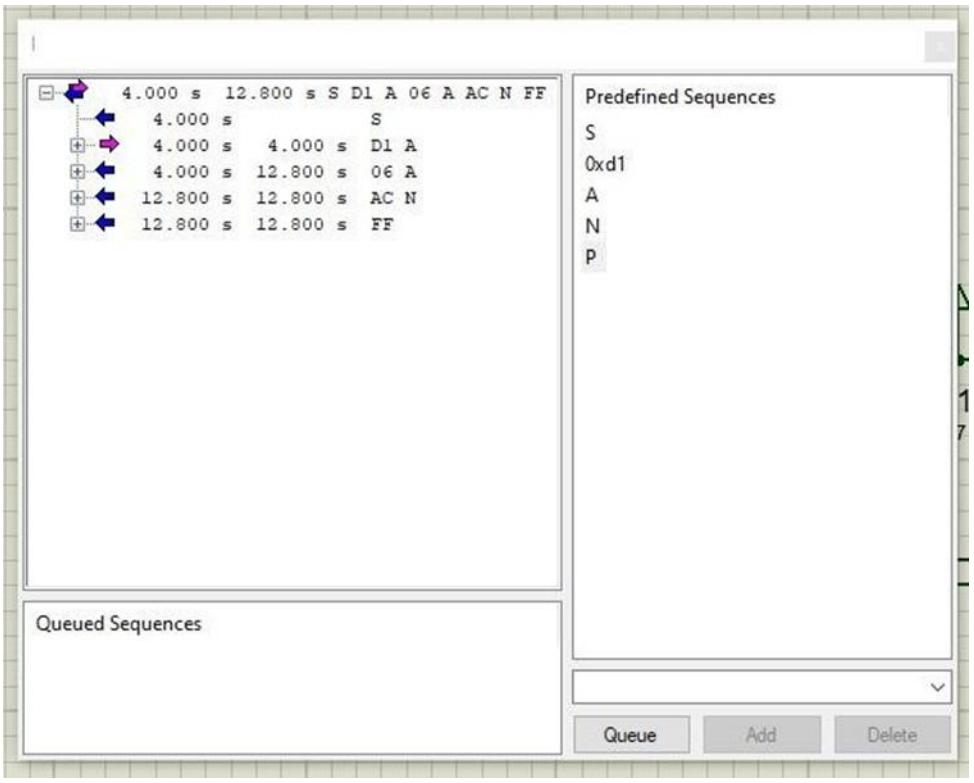
Descripción:

El dispositivo MCP3421 funciona de la siguiente manera realiza la conversión a un ritmo de 3.75, 15,60 o 240 muestras por segundo (MPS) en función de la configuración de los bits de control sobre el bus I2C. Este dispositivo tiene un amplificador interno de ganancia programable (PGA). El usuario puede seleccionar la ganancia en x1, x2, x4, x8. Esto permite que el dispositivo MCP3421 realice las conversiones con mayor precisión. El dispositivo tiene dos modos de trabajo: (a) el modo continuo y
(b) el modo One-Shot. En el modo One-Shot, el dispositivo entra en un modo de baja corriente de espera automáticamente después de una conversión. Esto reduce el consumo de corriente en gran medida durante los períodos de inactividad. El dispositivo MCP3421 se puede utilizar para varias aplicaciones de alta precisión de conversión analógica donde la simplicidad de diseño, bajo consumoy pequeño tamaño son las consideraciones principales.

El MCP3421 es un solo canal de bajo ruido, alta precisión convertidor ΔΣ análogo digital con entradas diferenciales y hasta 18 bits de resolución en un pequeño encapsulado SOT-23-6. La tensión de la precisión en la placa de referencia 2.048V permite un rango de entrada de ± 2.048V diferencial
(tensión Δ = 4.096V). El dispositivo utiliza dos con interfaz I2C y funciona con una sola fuente de alimentación de 2.7V a 5.5V

La Shields de traducción resistiva se muestra en el circuito a continuación:





Estas serían las tramas, la resolución de micro es 0,001, como vemos el micro nos responde que lee 06AC = 1708, entonces para convertirlo a tensión es la lectura por la resolución del micro = 1708*0,001 = 1,7v.
Si se fija en la imagen lo que no da
La S seria la palabra de inicio, oxd1 lo que queremos leer, A es reconocimiento, la N no reconocimiento se debe enviar antes de finalizar la comunicación, la P es de STOP.
A continuación, se envía el archivo SHIEDLS de traducción resistiva.

d) Que es el controlador ssd1306(i2c). Existe alguna shield para controlar una pantalla oled 128x64?; si es así, implemente una practica donde muestre el mensaje “Es fácil el desarrollo con shields”.

Las pantallas OLED no son más que eso, pantallas. Por si solas no hacen nada. Para poder mostrar datos e información en ellas se necesita un driver o controlador.

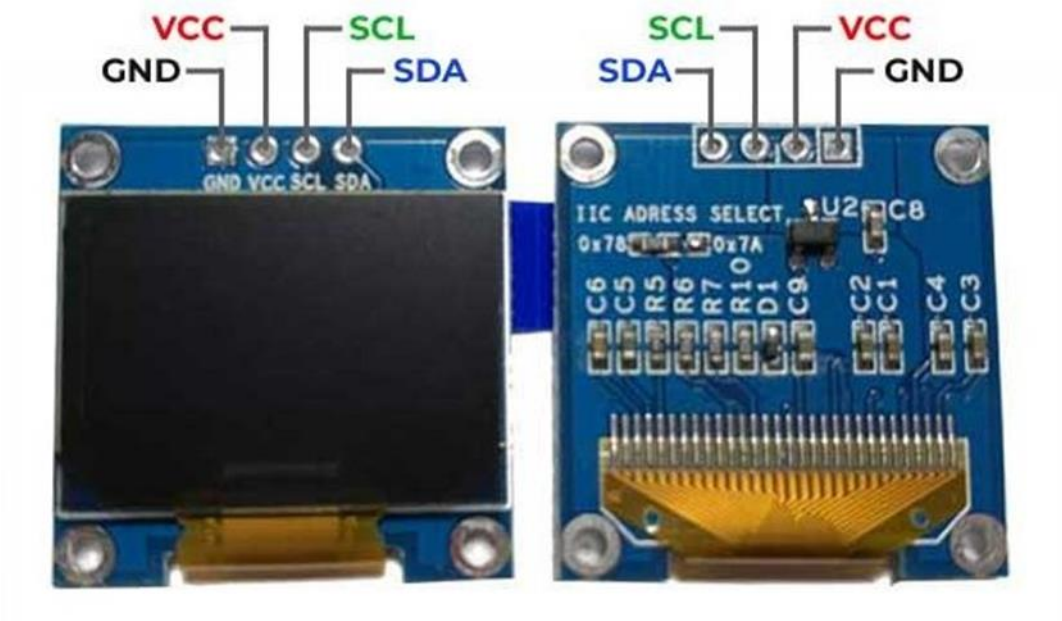
Normalmente estos módulos utilizan un controlador bastante conocido que se llama **SSD1306**. Pero pueden utilizar otro tipo de controladores. Se trata de un potente controlador **OLED CMOS**. Básicamente lo que hace el **SSD1306** es **comunicar con el microcontrolador** para obtener los datos y enviarlos a la pantalla OLED para que dibuje esos datos.

La comunicación entre el SSD1306 y el microcontrolador ya sea un Arduino o un ESP8266, se realiza mediante **SPI o I2C**. Generalmente, la comunicación **SPI** es **más rápida** que la comunicación **I2C**. Por el contrario, la

comunicación **SPI** requiere más pines que la comunicación **I2C**.

Al final una cosa compensa a la otra. Puedes elegir una comunicación más rápida en detrimento del número de pines o puedes elegir liberar más pines en detrimento de la velocidad. No deja de ser una elección personal.

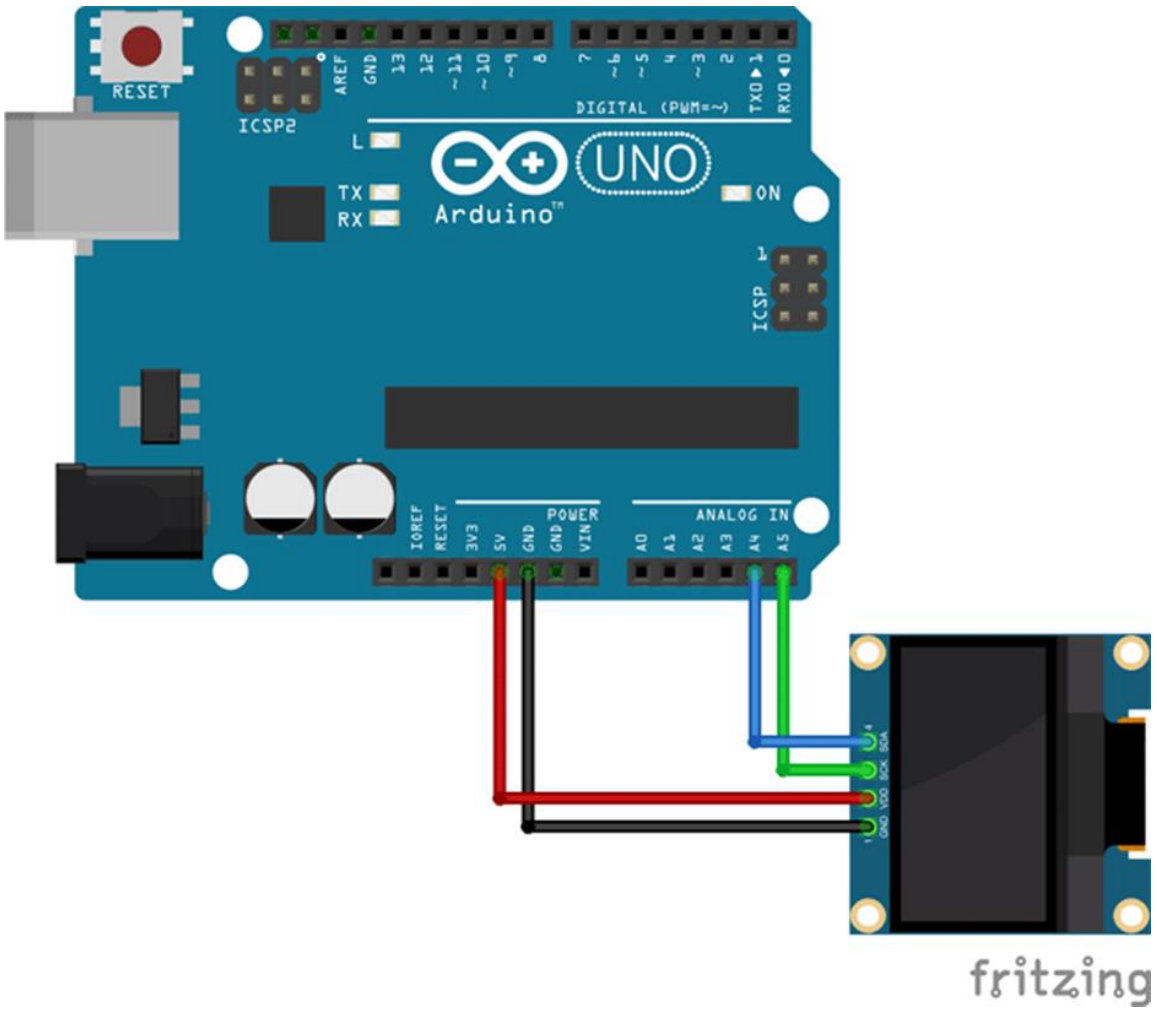
La pantalla OLED que voy a mostrar a continuación es una pantalla que utiliza la interfaz I2C. En este tipo de pantallas vas a encontrar 4 pines.



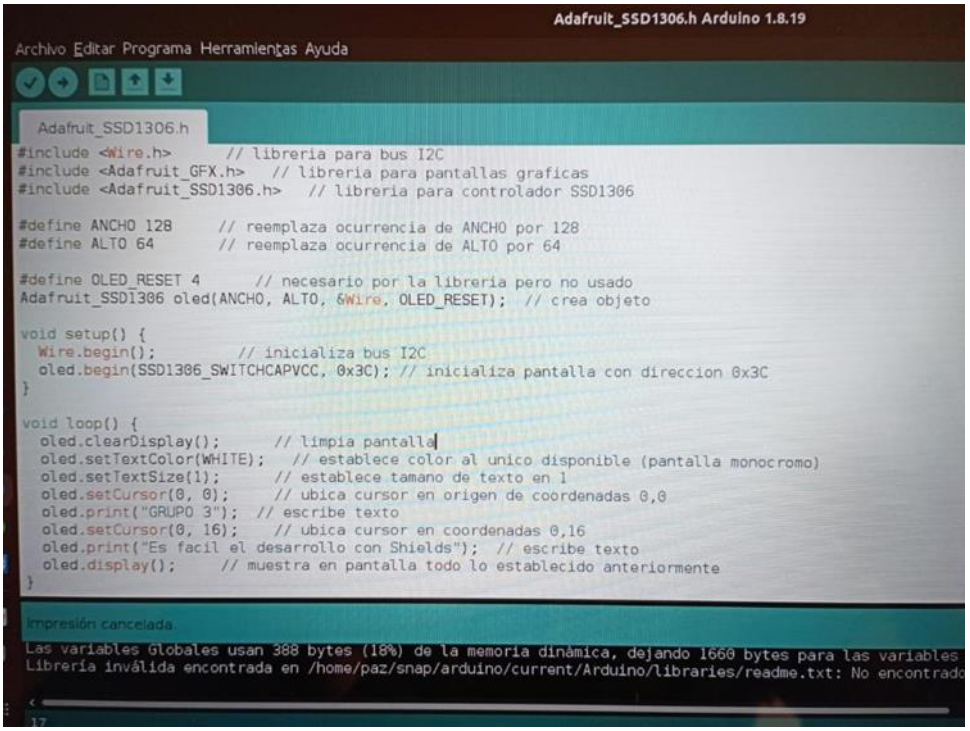
- **GND**: pin de tierra.
- **VCC**: es el pin de alimentación. Se puede alimentar la pantalla entre 1,8V y 6V.
- **SCL**: es el pin de la señal de reloj de la interfaz I2C.
- **SDA**: es el pin de la señal de datos de la interfaz I2C.

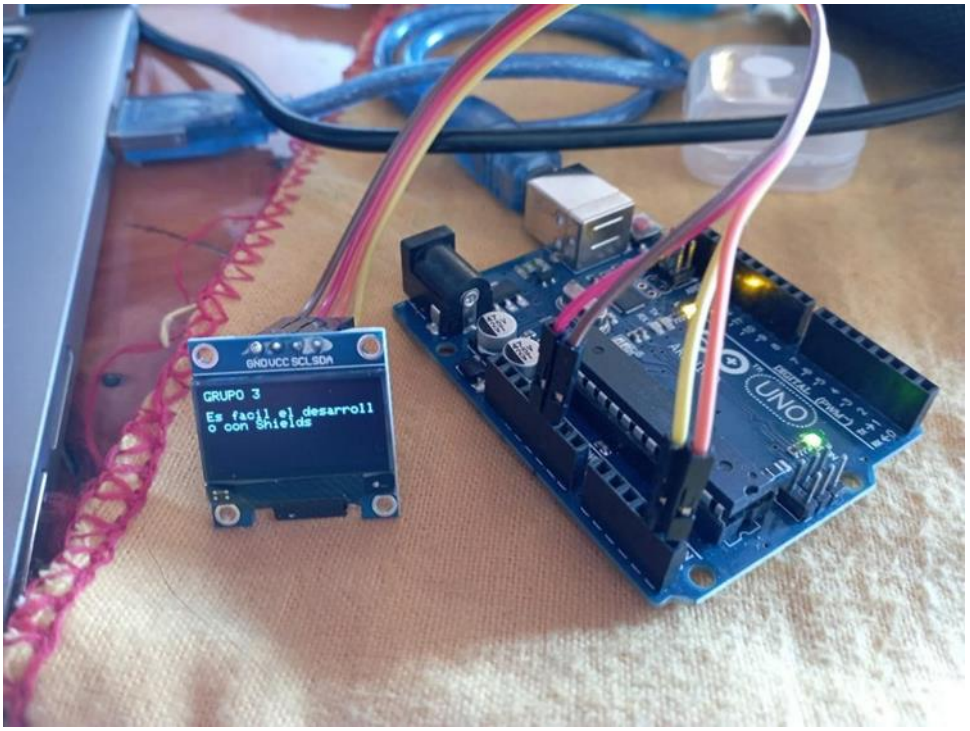
Conexión pantalla OLED con Arduino I2C

El siguiente esquema te muestro cómo debes conectar la pantalla OLED con un Arduino Uno.



El código hecho en la plataforma Arduino





Conclusión:

Probablemente el controlador SSD1306, no viene ajustado para su pantalla OLED de forma predeterminada, por lo tanto, el tamaño de visualización deberá cambiarse antes de usarse en caso contrario saltará un error como

este #error (“Altura incorrecta, corrija Adafruit_SSD1306.h!”).

Abra la carpeta y el archivo Adafruit_SSD1306.h con un editor de texto en mi caso el Arduino. Busque y comente la línea #define SSD1306 128x64 y luego guarde el archivo.

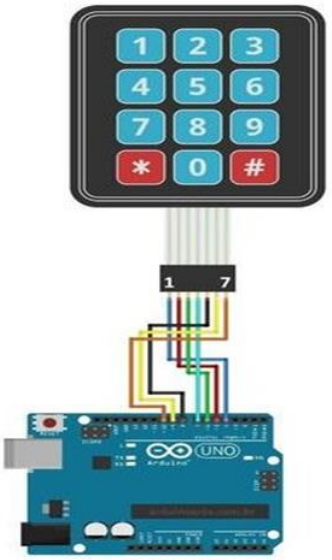
e) Un teclado, tipo telefónico, de membrana es una shield?; El conjunto {teclado + librerías de uso + repositorio (implementación, ejemplos, etc.)} es una shield?

Un shield Arduino es una placa de extensión para las placas Arduino. Estos módulos Arduino son placas quepodremos conectar a las ciertas tarjetas Arduino y dotarlas de nuevas funcionalidades. Normalmente estas placas tienen el tamaño y pinout preciso para que encajen en los pines de algunas placas Arduino, lo cual el teclado numérico 4×3 SCM Matricial de 12 teclas, es un dispositivo de entrada “periférico de entrada” y no una shield, este dispositivo nos ayudara a crear proyectos de control de acceso así como otros sistemas de codificación donde necesites una interfaz con botones, tiene cuatro perforaciones en las orillas ideales para montar en paneles, cajas o carcasas impresas en 3D. Es compatible para las tarjetas de Arduino, AVR, PIC, ARM y otro micro controladores.

Los teclados matriciales usan una combinación de filas y columnas para conocer el estado de los botones. Cada tecla es un pulsador conectado a una fila y a una columna. Cuando se pulsa una de las teclas, se cierra una conexión única entre una fila y una columna.

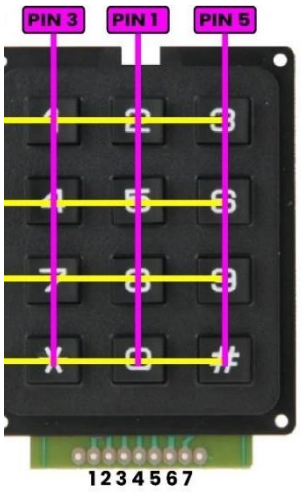
ESPECIFICACIONES Y CARACTERÍSTICAS

- Tipo: Teclado Numérico 4×3 SCM
- Teclado: Matricial
- Teclas: 12
- Contenido: Números del 0 al 9 y dígitos (*) y (#)
- Fuerza de accionamiento de 100 ±30g
- Capacidad nominal de contacto de 20mA a 24VDC
- Resistencia de contacto Máx.: 200Ω
- Rango de temperatura: -20ºC a +60ºC
- Vida útil de 1,000,000 ciclos por tecla
- Aconsejable utilizar resistencias en Pull-up
- Dimensiones: 7×5.2×0.9cm
  Peso: 19g



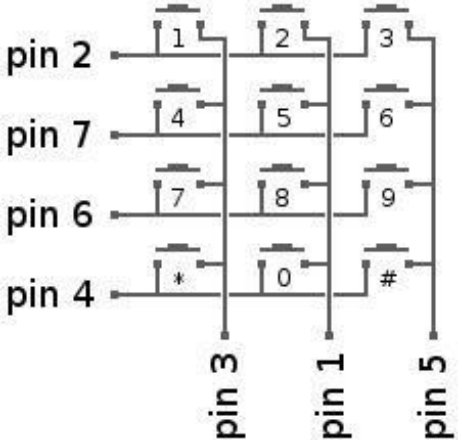
¿Cómo utilizar el teclado numérico 4×3?

Para utilizar en teclado con cualquier micro controlador es necesario entender como está conformado internamente la distribución de los botones. En las siguientes imágenes se puede detallar como está compuesto el teclado de acuerdo a los botones y los pines que tiene el teclado así mismo el diagrama deconexión de cómo están configurados internamente los botones son respecto a los pines:



Teclas + Pines

1:	2+3
2:	1+2
3:	2+5
4:	3+7
5:	1+7
6:	5+7
7:	3+6
8:	1+6
9:	5+6
*:	3+4
0:	1+4
#:	4+5



Para programar el Teclado o Keypad en Arduino o en cualquier otro microcontrolador, basta con seguir enorden la siguiente secuencia:

Inicialmente conectamos el teclado matricial en PINES DIGITALES del ARDUINO. Para eso será importantepoder identificar cuáles son las columnas y las filas del teclado.

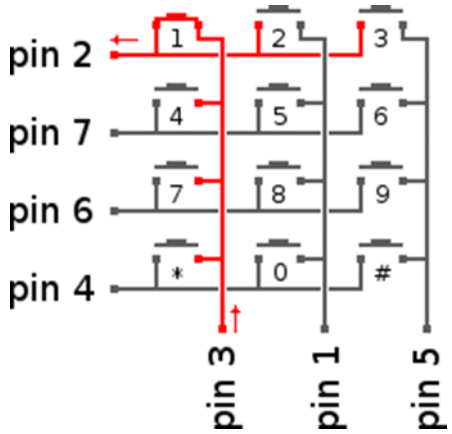
Las Filas del Teclado estarán conectadas en PINES DIGITALES configurados como SALIDAS.

Las Columnas del Teclado estarán conectadas en PINES DIGITALES configurados como ENTRADAS y conel PULL UP (por lo tanto, estas entradas siempre estarán recibiendo un 1 lógico, si ningún botón es presionado).

Configurar TODAS las SALIDAS (Filas) en 1 lógico o 5V

Aplicamos el concepto de la MULTIPLEXACIÓN: Aquí vamos a mandar un 0 lógico por cada fila y vamos a leer todas las columnas, si se detecta que alguna columna recibió el cero lógico, indica que el botón que comparte la fila y la columna fue presionado, en caso contrario vuelvo a colocar la fila en 1 lógico y verifico lasiguiente fila.

En la siguiente animación podemos ver como se establece el presionado de un botón sobre el keypad.Notarás más adelante, que cuando se programa el teclado sin librería, básicamente usamos el tecladomatricial con for (ciclo) para poder preguntar por las columnas del teclado



Un shield Arduino es una placa de extensión para las placas Arduino. Estos módulos Arduino son placas que podremos conectar a las ciertas tarjetas Arduino y dotarlas de nuevas funcionalidades. Normalmente estas placas tienen el tamaño y pinout preciso para que encajen en los pines de algunas placas Arduino

f) Explique con sus propias palabras que es una shield.

Un shield es una placa secundaria que se puede conectar a la placa base del Arduino, está es una placa de circuito impreso y se puede conectar sin ninguna otra conexión, al conectar los pines del shield a la placa base, amplía las capacidades de la misma.

El shield es compatible con diferentes placas Arduino, entre ellas está diseñado para funcionar con las placas Arduino MEGA y UNO.

Los shields con múltiples funciones a menudo tienen capacidades apilables, por ellos, antes de comprar uno de estos, se debe verificar la información que se brinda en la documentación del mismo para asegurarse de que los diferentes pines del shield no se usen en los mismos pines.

El Arduino puede suministrar hasta aproximadamente 500 miliamperios, cualquier shield que use más electricidad que eso superaría la corriente máxima que el dispositivo puede proporcionar. Además, cualquier shield que use una gran cantidad de componentes puede requerir más energía de la que puede proporcionar el Arduino, independientemente del tamaño o tipo del componente.

Ejercicio #2

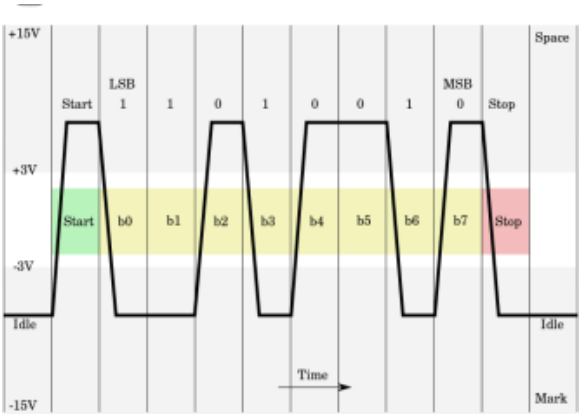
a) Que es el protocolo SPI y cuales son sus características.

Fue desarrollado por Motorola allá por los 80's, y que es un protocolo muy utilizado para comunicarse con distintos módulos de hardware. Este tipo de comunicación se ve comúnmente en el mundo de los microcontroladores, utilizado para comunicarse con otros dispositivos como memorias, pantallas,tarjetas de sonido, y dispositivos de interfaz serie-paralelo, lo cual nos brinda la oportunidad de crear cosas muy robustas e interesantes.

El problema

La comunicación serial del tipo RX y TX (USART/UART) es una comunicación asíncrona, que noposee control de los datos que se envían sobre su bus y sin garantía de que ambos dispositivos estén funcionando siempre a la misma velocidad. Esto causa un problema ya que los dispositivos a conectar no siempre corren a la misma frecuencia, dificultando la comunicación entre ambos.

Para solucionar este error de sincronización, se agrega un bit de inicio (start) y un bit de parada(stop)al final de cada dato enviado, con lo cual el receptor sincroniza su reloj para la comunicación en cada bit de inicio.

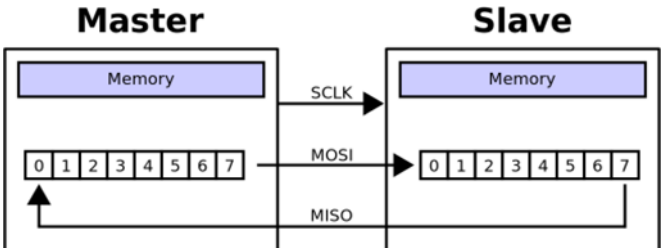


La solución

La solución a este problema es el protocolo SPI. Es un protocolo con un bus síncrono, lo que significa que utiliza líneas separadas para datos y reloj, lo que dota a ambos dispositivos de perfecta sincronización.

El reloj le indica al dispositivo receptor el momento exacto en que puede tomar el bit de la línea de datos enviado por el transmisor. El pulso de reloj puede ser tanto de subida (rising) como de bajada (fallin). Cuando el receptor recibe este pulso, inmediatamente toma el dato de la línea y lo almacena en un registro de corrimiento. Un dato importante a tener en cuenta son los límites de velocidad del reloj del dispositivo receptor, ya que la frecuencia de

la señal generada por el reloj del transmisor, puede ser más alta que la soportada por el receptor, causando problemas en la comunicación.



Una de las razones por las que el protocolo SPI es muy popular es porque el hardware que lo compone es un simple registro de desplazamiento (shift register), el cual tiene un costo mucho más bajo que los chips USART/UART.

Muchos sistemas digitales tienen periféricos que necesitan existir, pero no ser rápidos. Las ventajas de un bus serie es que minimiza el número de conductores, pines y el tamaño del circuito integrado. Esto reduce el coste de fabricar montar y probar la electrónica. Un bus de periféricos serie es la opción másflexible cuando se tiene tipos diferentes de periféricos serie. El hardware consiste en señales de reloj, data in, data out y chip select para cada circuito integrado que tiene que ser controlado. Casi cualquier dispositivo digital puede ser controlado con esta combinación de señales. Los dispositivos se diferencian en un número predecible de formas. Unos leen el dato cuando el reloj sube otros cuando elreloj baja. Algunos lo leen en el flanco de subida del reloj y otros en el flanco de bajada. Escribir escasi siempre en la dirección opuesta de la dirección de movimiento del reloj.

El bus SPI se define mediante 4 pines:

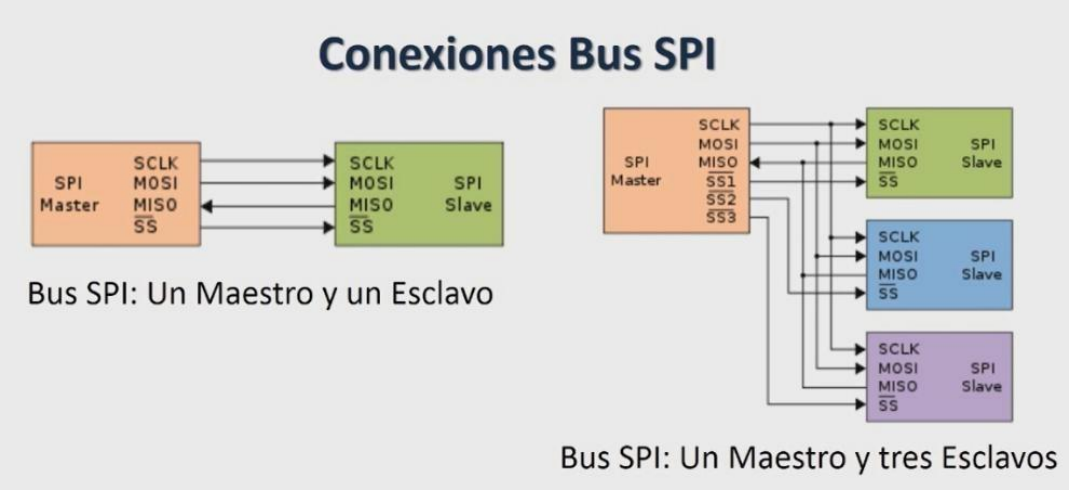
- SCLK o SCK : Señal de reloj del bus. Esta señal rige la velocidad a la que se transmitecada bit.
- MISO (Master Input Slave Output): Es la señal de entrada a nuestro dispositivo, por aquíse reciben los datos desde el otro integrado.
- MOSI (Master Output Slave Input): Transmisión de datos hacia el otro integrado.
- SS o CS: Chip Select o Slave Select, habilita el integrado hacia el que se envían losdatos. Esta señal es opcional y en algunos casos no se usa.

A diferencia de otros buses el SPI no implementa el nivel del enlace entre dispositivos, es decir no hay un campo para la dirección ni un campo para ACK, etc. El SPI se comporta como un shift register donde a cada golpe de clock se captura un bit. En parte no es necesaria hacer un direccionamiento de los chips ya que mediante la señal Chip select, habilitamos al integrado al que queremos enviar los datos. El funcionamiento para un envió de un Master es el siguiente:

- Se habilita el chip al que hay que enviar la información mediante el CS (Opcional).
- Se carga en el buffer de salida el byte a enviar.
- La línea de Clock empieza a generar la señal cuadrada donde normalmente por cadaflanco de bajada se pone un bit en MOSI.

El receptor normalmente en cada flanco de subida captura el bit de la línea MISO y loincorpora en el buffer. Se repite el proceso 8 veces y se ha transmitido un byte. Si se ha terminado de transmitir se vuelve a poner la

línea CS en reposo. Hay que tener en cuenta que a la vez que el Master está enviando un dato también lo recibe así que, si el Slave ha depositado algún byte en el buffer de salida, este tambiénserá enviado y recibido por el Master.



Funcionamiento de SPI en Atmega8

El SPI Master (servidor) inicializa el ciclo de comunicación cuando se coloca en bajo el Selector de Esclavo (SS-Selector Slave) (cliente). Master y Slave (servidor y cliente) preparan los datos a ser enviados en sus respectivos registros de desplazamiento y el Master genera el pulso del reloj en el pin SCK para el intercambio de datos. Los datos son siempre intercambiados desde el Maestro al Esclavo en MasterOut-SlaveIn, MOSI, y desde Esclavo al Maestro en MasterIn-SlaveOut, MISO. Después de 7 cada paquete de datos el Maestro debe sincronizar el esclavo llevando a 'alto' el selector de Esclavo, SS.

Cuando se configure como Maestro, la interfaz SPI no tendrá un control automático de la línea SS. Este debe ser manejado por software antes de que la comunicación pueda empezar, cuando esto es realizado, escribiendo un byte en el registro de la SPI comienza el reloj de la SPI, y el hardware cambia los 8 bits dentro del Esclavo. Después de cambiar un Byte, el reloj del SPI para, habilitando el fin de la transmisión (SPIF). Si la interrupción del SPI está habilitado (SPIE) en el registro SPCR, una interrupción es requerida. El Master podría continuar al cambio del siguiente byte escribiendo dentro del SPDR, o señalizar el fin del paquete colocando en alto el Esclavo seleccionado, línea SS. El último byte llegado se mantendrá en el registro Buffer para luego usarse.

Cuando lo configuramos como un Esclavo, la interfaz SPI permanecerá durmiendo con MISO en tres estados siempre y cuando el pin SS este deshabilitado. En este estado, por el software se podría actualizar el contenido del registro SPDR, pero los datos no serán desplazados por la llegada del pulsode reloj en el pin SCK hasta que el pin SS no sea habilitado ('0'). Será visto como un byte completamente desplazado en el fin de la transmisión cuando SPIF se habilite. Si la interrupción SPI, SPIE en SPCR, está habilitada, una interrupción es solicitada. El Esclavo podría continuar para colocarnuevos datos para ser enviados dentro del SPDR antes de seguir leyendo la data que va llegando. El último byte que entra permanecerá en el buffer para luego usarse.

Para usar el bus SPI en Arduino, se usa la librería: http://arduino.cc/en/Reference/SPI

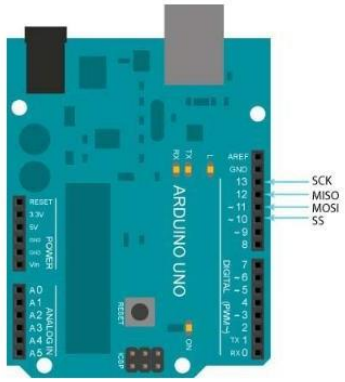
En un primer paso importamos la librería del SPI con #include. En el setup hay que iniciar y configurarel SPI con SPI.begin() y además hay que definir el pin SS como salida.
Finalmente, mediante la función SPI.transfer enviamos el byte que queremos.

Métodos SPI:

- begin () — Inicializa el bus SPI
- end () — Deshabilita el bus SPI.
- setBitOrder () — Configura el orden de los bits enviados como el menos significativo primero o elmás significativo primero.
- setClockDivider () — Configura del divisor de reloj en el bus SPI. Es decir, configura la velocidaddel bus.
- setDataMode () — Configura el modo de dato del bus SPI, es decir, polaridad y fase del reloj.
- transfer () — Transfiere un byte sobre el bus SPI, tanto de envío como de recepción.

En las placas **Arduino UNO R3** o **Arduino Duemilanove**, los pines utilizados son:

- **SS**: digital 10. Puede usar otros pines digitales, pero 10 es generalmente el valor predeterminado. Si se piensa usar el bus SPI en modo esclavo, este pin es de uso obligatorio.
- **MOSI**: digital 11
- **MISO**: digital 12
- **SCK**: digital 13



Inicialización del bus SPI

```
Inicialización de bus SPI                                   Arduino
SPI.begin();           // Inicializa el bus SPI
SPI.transfer(dato);    // Envía un byte
SPI.attachInterrupt(); // Activa las interrupciones para recibir datos
```

Configuración el orden de transmisión de bits del bus SPI

```
Configuracion del orden de transmisión en bus SPI           Arduino
SPI.setBitOrder(LSBFIRST); // Bit menos significativo primero
SPI.setBitOrder(MSBFIRST); // Bit mas signigicativo primero
```

Configurar la polaridad del reloj del bus SPI

```
Configuracion polaridad del reloj en bus SPI                Arduino
SPI.setDataMode(SPI_MODE0); // Reloj normalmente bajo, muestreo en flanco subida
SPI.setDataMode(SPI_MODE1); // Reloj normalmente bajo, muestreo en flanco bajada
SPI.setDataMode(SPI_MODE2); // Reloj normalmente alto, muestreo en flanco subida
SPI.setDataMode(SPI_MODE3); // Reloj normalmente alto, muestreo en flanco bajada
```

b) Cuales son las ventajas y desventajas del protocolo SPI

Ventajas de SPI sobre I2C.

- I2C No es Full-Dúplex por lo que no permite envíos y recepciones al mismo tiempo.
- I2C un poco más complejo que SPI.
- I2C no tiene control de errores, por ejemplo, mediante paridad etc. Aunque se puede realizar por Software.
- Velocidades de comunicación relativamente elevadas. En el caso de Arduino de hasta 8 MHz.
- Completo control sobre la trama de bits al no exigir direccionamiento ni ACK.
- Se requiere un hardware sencillo (Barato)
- Requiere un menor consumo y menor electrónica de conexión que el I2C
- Como el Clock lo proporciona el master, los esclavos no necesitan osciladores (más barato).
- Alta velocidad de trasmisión (hasta 8 MHz en Arduino) y Full Dúplex

- Los dispositivos necesarios son sencillos y baratos, lo que hace que esté integrado en muchos dispositivos.
- Puede mandar secuencias de bit de cualquier tamaño, sin dividir y sin interrupciones.

Desventajas de SPI:

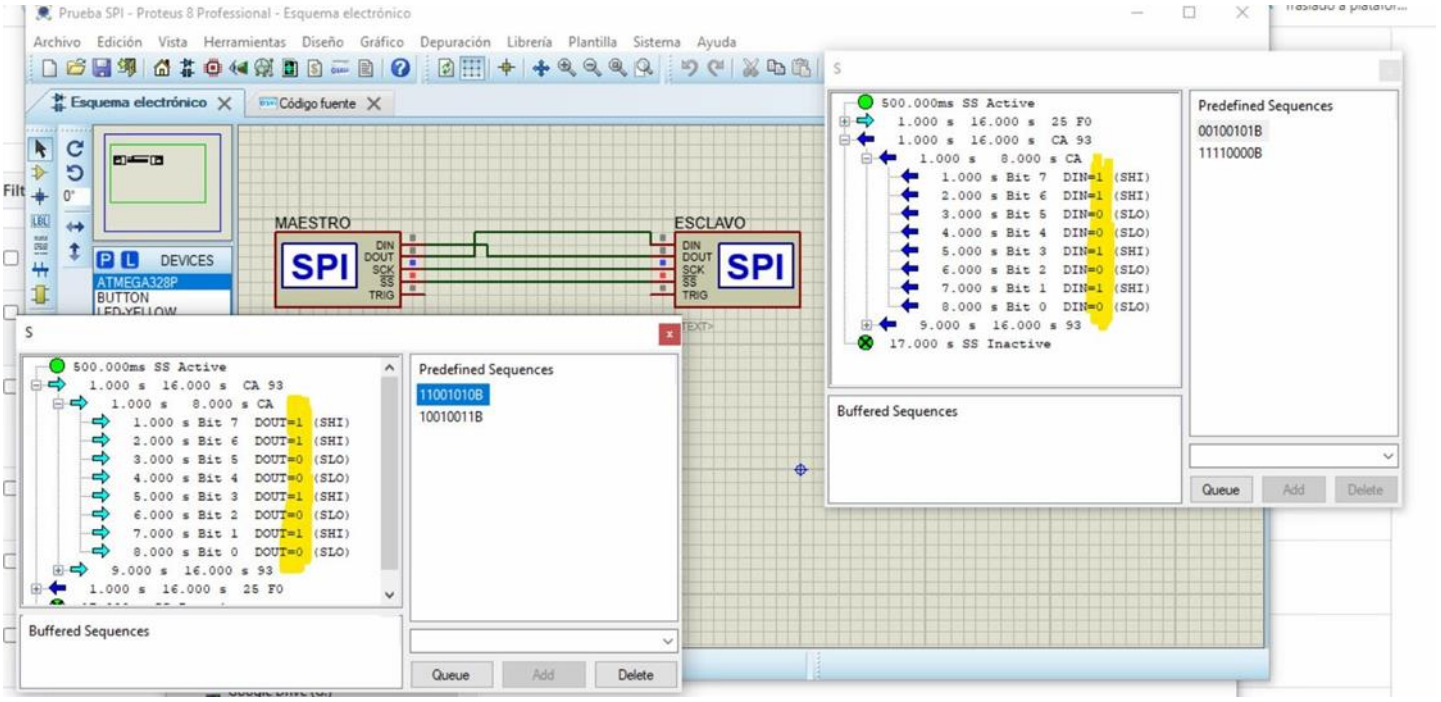
- No hay control del flujo por hardware.
- Las comunicaciones tienen que estar perfectamente establecidas de antemano. No puedesenviar mensajes de diferentes longitudes cuando convenga.
- No hay confirmación de la recepción como ocurre en I2C con el ACK. Es decir, no sabemos si elmensaje a llegado al destino.
- Usa más pines que otros buses, ya que necesita uno por cada esclavo. Eso implica que no hay direccionamiento en la propia trama. A menos que se diseñe por software.
- Funcionamiento a distancias cortas
- Master único y casi sin posibilidad de master múltiple
- Se requiere 3 cables (SCK, MOSI y MISO) + 1 cable adicional (SS) por cada dispositivoesclavo.
- Solo es adecuado acorta distancias (unos 30cm) s
- No se dispone de ningún mecanismo de control, es decir, no podemos saber si el mensaje hasido recibido y menos si ha sido recibido correctamente.
- La longitud de los mensajes enviados y recibidos tiene que ser conocida por ambos dispositivos.

Tanto el bus SPI como el I2C son llamados buses de tarjeta, es decir están pensados para trabajar a distancias pequeñas del entorno de una tarjeta, en caso de necesitar un bus serie a larga distanciahay que ir a buses de campo como RS485, CAN, etc.…

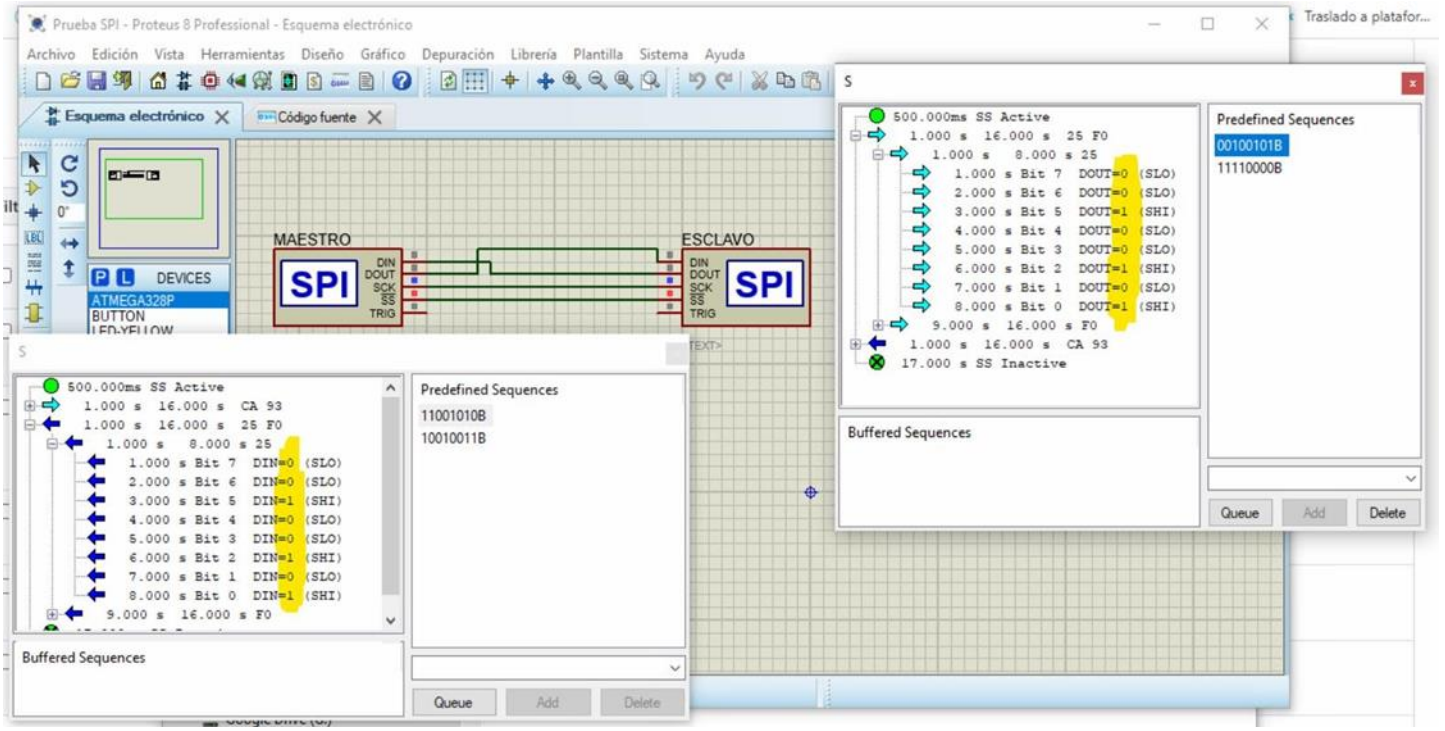
- El canal SPI fue diseñado para aplicaciones de transmisión de datos a velocidades altas (10 Mbps) y distancias cortas, del orden de 10 a 20 cms, o bien dentro de un mismo PCB (circuito impreso), entre 2 circuitos integrados como podrían ser un microcontrolador y otro dispositivo, por ejemplo, un circuito integrado con la función RFID. Las señales de transmisión de datos y control del canal SPI, usan niveles de voltaje TTL o bien 3.3 volts, dependiendo de la tecnología de fabricación del dispositivo

c) Como probaría si una comunicación SPI funciona correctamente en su laboratorio?

Podría probarlo a través del SPI Debugger que tiene Proteus, en este caso vamos a hacer una prueba con un dispositivoMaestro y uno Esclavo.
Como podemos ver en la siguiente imagen, vemos el primer dato que envía el maestro (11001010) DOUT (datossalientes) y luego lo vemos como DIN (datos entrantes) en el esclavo aquí vemos cada bit recibido.



Luego en la siguiente imagen podemos ver cuando el esclavo le envía datos al maestro, aquí vemos los bits que envía elesclavo (DOUT) y luego los vemos reflejados en el maestro (DIN)



Dejo aquí algunos videos desde los cuales me guie para hacer las pruebas

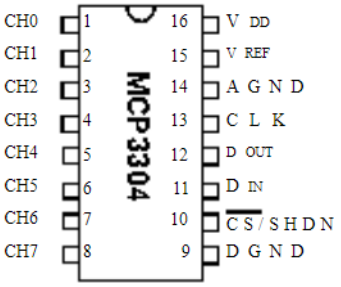
https://www.youtube.com/watch?v=onW8q1IX5IU

https://www.youtube.com/watch?v=Mb3dR_6a0us&ab_channel=SETISAEDU

d) Que es el mcp3304(spi) y como lo utilizaría para hacer una shield que controle 8 sensores?

El MCP3304 es un convertidor A/D (analógico/digital) de baja potencia, con interfaz serial SPI.

PDIP, SOIC



Características

- Entradas diferenciales completas
- ±1 LSB máx. DNL
- ±1 LSB máx. INL (MCP3302/04-B)
- ±2 LSB máx. INL (MCP3302/04-C)
- Operación de suministro único: 2.7V a 5.5V
- Frecuencia de muestreo de 100 ksps con tensión de alimentación de 5 V
- Frecuencia de muestreo de 50 kbps con tensión de alimentación de 2,7 V
- Corriente de espera típica de 50 nA, 1µ A m á x.
- 4 5 0 µA corriente máxima activa a 5V
- Rango de temperatura industrial: -40 °C a +85 °C
- Paquetes PDIP, SOIC y TSSOP de 14 y 16 pines
    - Kit de evaluación MXDEV TM disponible

Aplicaciones

- Sensores Remotos
- Sistemas operados por batería
- Interfaz del transductor

Descripción general

Los convertidores A/D de 13 bits MCP3302/04 de Microchip Technology Inc. cuentan con entradas diferenciales completas y bajo consumo deenergía en un paquete pequeño que es ideal para sistemas alimentados por batería y aplicaciones de adquisición remota de datos.

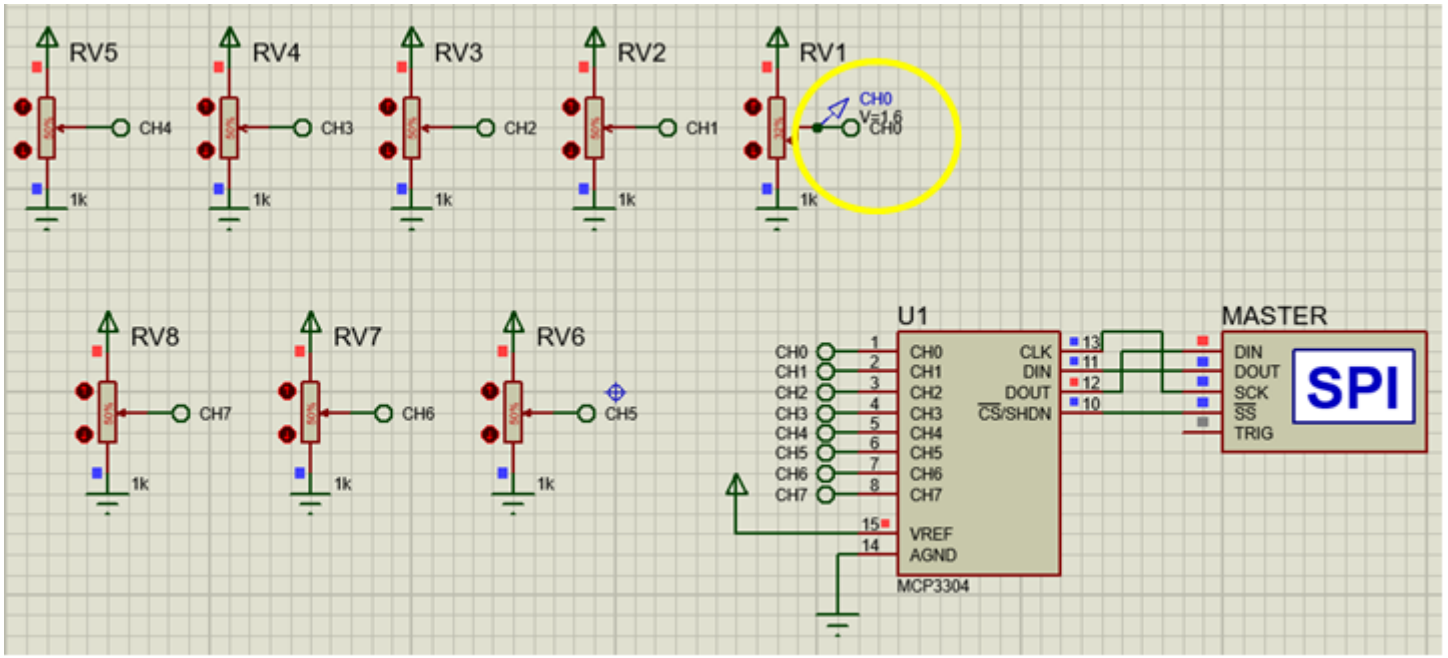
El MCP3304 es programable y proporciona cuatro pares de entradas diferenciales u ocho entradas de un solo extremo.

Estos convertidores A/D de 13 bits, que incorporan una arquitectura de aproximación sucesiva con circuitos integrados de muestreo y retención, están especificados para tener una no linealidad diferencial (DNL) de ±1 LSB; ±1 LSB de no linealidad integral (INL) para dispositivos de grado B y ±2 LSB para dispositivos de grado C. La interfaz serial SPI™ estándar de la industria permite agregar capacidad de convertidor A/D de 13 bits a cualquier microcontrolador PICmicro.

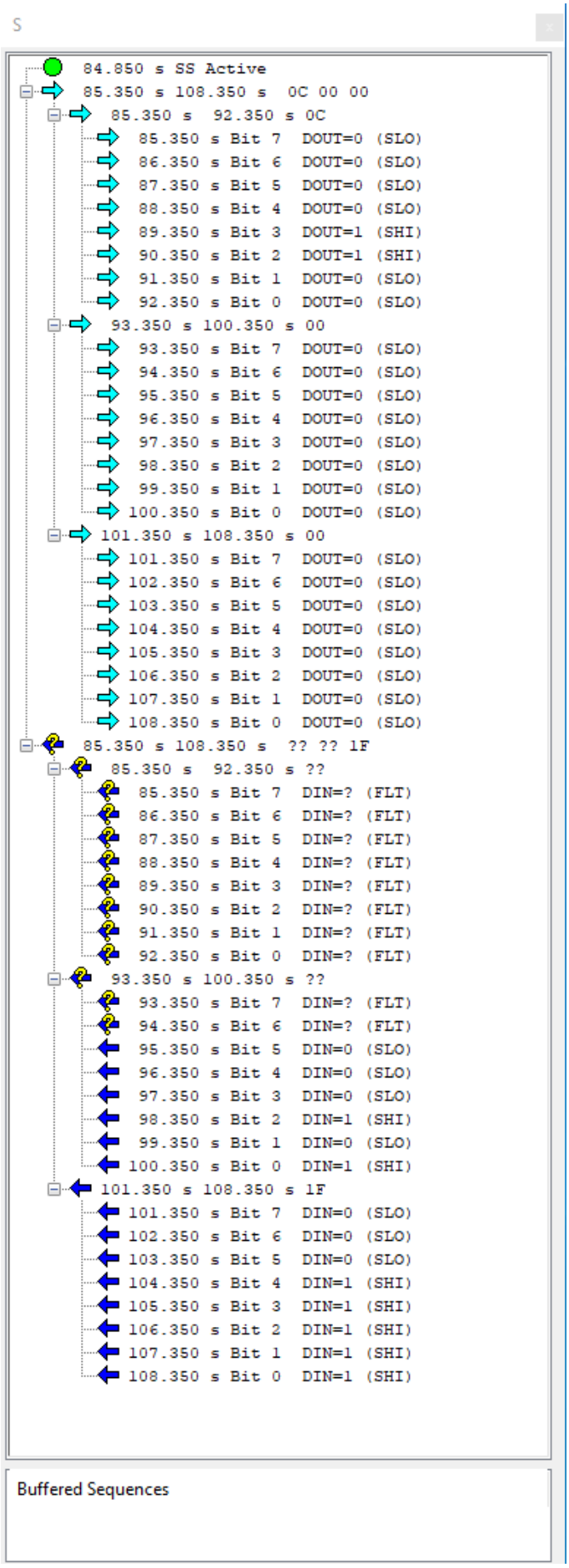
Los dispositivos MCP3302/04 cuentan con un diseño de baja corriente que permite el funcionamiento con corrientes típicas en espera y activas de solo 50 nA y 300 µA, respectivamente. Los dispositivos funcionan en un amplio rango de voltaje de 2,7 V a 5,5 V y son capaces de tasas de conversión de hasta 100 kbps. El voltaje de referencia se puede variar de 400 mV a 5 V, lo que produce una resolución referida a la entrada entre 98 µV y 1,22 mV.

El MCP3304 está disponible en paquetes PDIP de 16 pines y SOIC de 150 mil. Las entradas diferenciales completas de este dispositivo permite el uso de una amplia variedad de señales en aplicaciones tales como adquisición remota de datos, instrumentación portátil y aplicaciones que funcionan con baterías

Al Shield lo podríamos implementar como en la siguiente imagen, en donde cada potenciómetro representaría un sensor conectado a cada pin del MCP3304



En donde deberíamos especificar luego del bit de inicio si queremos realizar una lectura sigle-ended (lectura de un solo canal) o pseudo-diferencial (lectura compuesta por dos canales), para la primera deberíamos poner el primer bit en 1, y para la segunda este debe estar en 0, luego de esto seleccionar el canal Ej. Lectura de canal 0 = 000, quedando así para single-ended 1000.



Como podemos observar en la imagen, enviamos (DOUT) los últimos 4 bits en 0, luego el tercer bit en 1 (estos 5bit conformanla trama de inicio). En el bit 2 enviado en 1 le decimos al micro que queremos hacer la lectura en modo single, luego en el bit 1, 2, y 7 del segundo octeto le especificamos el canal, en este caso el 0 (000). Luego los demás bit´s no interesan por eso los coloco todos en cero.

Aquí vemos lo que nos responde el micro (DIN), vemos que nosha respondido 10100011111 = 1311.

Si hacemos 1311*((2*Vref)

/8192)8192 = Resolución

del micro 1311*((2*5) /8192

= 1.6v

Nos da la tensión que marca la sonda (circulo amarillo) e n laimagen anterior de Proteus.

Se dejan precargados los bytes para poder leer el canal 0 en elarchivo de Proteus

Explicación de las tramas a enviar al micro para poder realizar la comunicación

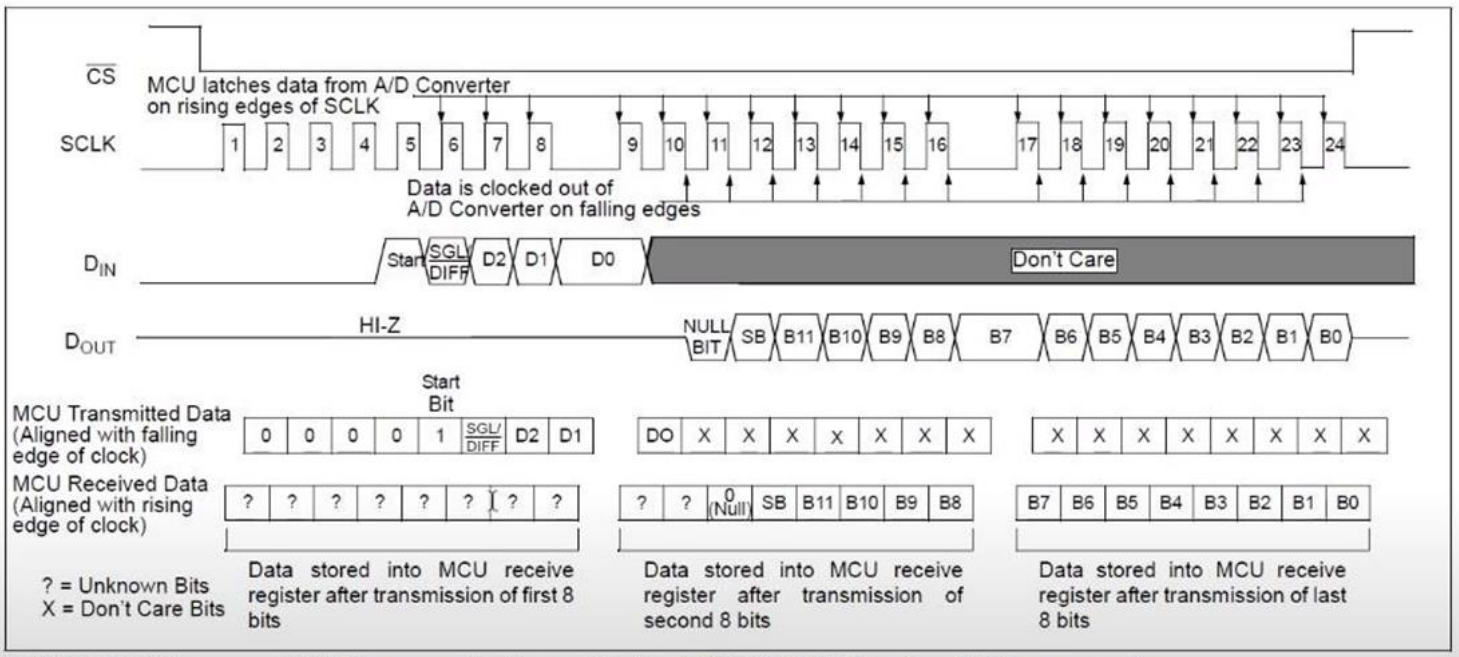


FIGURE 6-4: *SPI Communication with the MCP3302/04 using 8-bit segments (Mode 0,0: SCLK idles low).*

Tabla de selección de modo de lectura y canal a leer.

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

Hoja de datos del MCP3304

https://html.alldatasheet.com/html-pdf/98036/MICROCHIP/MCP3304/406/1/MCP3304.html

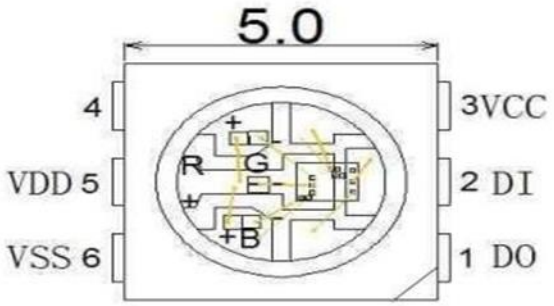
e) Que protocolo utilizan los led ws2812? Hay alguna librería para controlarlos con módulos ESP12 o ESP32? Que se podría armar con los WS2812 y porque el fabricante no utilizo protocolos SPI, UART o I2C?

Dos cosas importantes a considerar son que el WS2812 comparte una sola línea tanto para reloj como para datos, y que admite una velocidad de transmisión de datos mínima de 400 kBits por segundo. Esto hace queel desarrollo sea complicado tanto para el desarrollador de hardware como para el de firmware.

Dado que los datos de los parámetros RGB se modulan dentro de la señal de reloj, la precisión es obligatoria para el microcontrolador (MCU) que controla el LED. Esto les da a los desarrolladores de firmware dos opciones. Pueden seleccionar un microcontrolador de 32 bits de alta gama con una potencia de procesamiento inmensa u optar por un microcontrolador de 8 bits de bajo presupuesto. La MCU de 32 bits simplificará el desarrollo del firmware, pero es significativamente más cara. Como alternativa, la MCU de 8 bitspodría no generar el paquete de datos WS2812 con precisión. El tiempo es clave para estos LED y una diferencia de aproximadamente un microsegundo podría dar como resultado un color diferente.

La configuración ws2812 es algo muy diferente a led típico. Es un led (RGB) que está integrado con un chip de control inteligente en un solo factor de forma 5050. admite un protocolo de transmisión de una sola línea NZR, donde se envían señales de reloj y datos al ws2812, a un mínimo de 400 kbits por segundo, para controlar el valor RGB del led el ws2812 se puede conectar en cascada conectando el pin de "salida de datos"de un led a la "entrada de datos" del otro led.

Configuración del PIN



Función PIN

NO.	Símbolo	Descripción de la función
1	DOUT	Salida de la señal de control de datos
2	DIN	Entrada de la señal de control
3	VCC	Circuito de control de la fuente de alimentación
4	NC	
5	VDD	LED de alimentación
6	VSS	Tierra

Por un lado, tenemos la librería NeoPixel de Adafruit que **es sencilla de usar, pero relativamente lenta**, loque la hace inadecuada para efectos complejos.

Por otro lado, tenemos la librería FastLED, **algo más difícil, pero a cambio permite patrones mucho más**

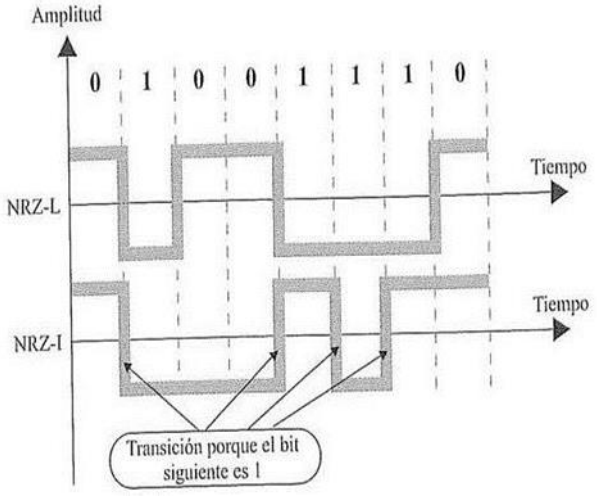
complejos. Lo cual permite realizar proyectos de una manera simple y escalable ya que se tiene el control decada pixel y de cada color, ya sea rojo, verde y azul a través de un solo cable. Además, tienen la ventaja de poder encadenarse unos tras otros (la salida de uno se puede conectar a la entrada de la siguiente) para obtener cadenas más largas además de formas interesantes gracias a su flexibilidad.

Dentro de los protocolos usados por el ws2812 se encuentra sin retorno a cero (nrz):

En la codificación NRZ, el nivel de la señal es siempre positivo o negativo, los dos métodos más usuales son:

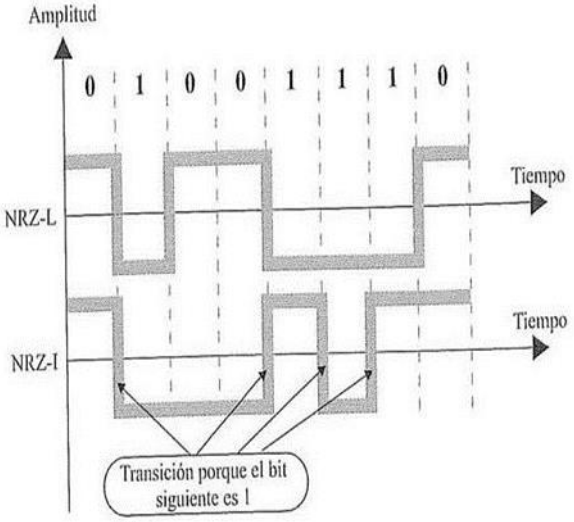
- NZR-L. En esta codificación, el nivel de la señal depende del tipo de bit que representa, habitualmente un valor de tensión positiva indica que el bit es un 0 y un valor de tensión negativa indica que el bit es un 1 (o viceversa), por tanto, el nivel de la señal depende del estado del bit. Pero cuando hay un flujo grande de ceros o unos en los datos puede surgir el problema de la sincronización.

- NZR-I. En esta codificación, una inversión de la tensión representa un bit, es la transición entre el valor de la tensión positiva y negativa, no la tensión en sí misma, lo que representa un bit. Un bit 0 se representa sin ningún cambio. NZR-I es mejor que NZR-L debido a la sincronización implícita provista por el cambio de señal cada vez que se encuentra un 1. La existencia de unos en el flujo de datos permite al receptor sincronizar su temporizador con la llegada real de la transmisión. Las tiras de ceros todavía pueden causar problemas.



La figura muestra las representaciones NZR-L y NZR-I de la misma serie de bits. En la secuencia NZR-L las tensiones positiva y negativa tienen un significado específico: positivo para el 0, negativo para el 1. En la secuencia NZR-I, las tensiones no tienen significado por sí mismas, es el cambio de nivel la base para reconocerlos unos. El siguiente enlace explica con mayor profundidad este modelo codificación NZR.

Con retorno a cero (RZ): si los datos originales contienen tiras de unos o ceros, el receptor puede sufrir pérdidas por sincronización. Otra de las soluciones es incluir de alguna forma la sincronización dentro de la señal codificada, algo similar a la solución de NZR-I pero capaz de manejar tiras de unos y de ceros.



La figura muestra las representaciones NZR-L y NZR-I de la misma serie de bits. En la secuencia NZR-L las tensiones positiva y negativa tienen un significado específico: positivo para el 0, negativo para el 1. En la secuencia NZR-I, las tensiones no tienen significado por sí mismas, es el cambio de nivel la base para reconocerlos unos. El siguiente enlace explica con mayor profundidad este modelo codificación NZR.

PROTOCOLOS DE COMUNICACIÓN SERIE

UART (receptor-transmisor asíncrono universal): el protocolo UART es uno de los protocolos de comunicación en serie más antiguos y fiables que todavía utilizamos en la actualidad. Este protocolo utilizados cables conocidos como Tx (transmitir) y Rx (recibir) para comunicarse con ambos componentes. Para transmitir datos, tanto el transmisor como el receptor deben estar de acuerdo con cinco configuraciones comunes, estas son:

- **Velocidad de transmisión:** La velocidad de transmisión de la velocidad de transmisión de datos.
- **Longitud de datos:** El número acordado de bits que el receptor guardará en sus registros.
- **Bit de inicio:** Una señal baja que le permite al receptor saber cuándo se van a transferir datos.
- **Detener bit:** Una señal alta que le permite al receptor saber cuándo se ha enviado el último bit (bit mássignificativo).
- **Bit de paridad:** Una señal alta o baja utilizada para verificar si los datos enviados eran correctos oestaban corruptos.

SPI (interfaz periférica en serie): SPI es otro protocolo en serie popular utilizado para velocidades de datos más altas de alrededor de 20 Mbps. Utiliza un total de cuatro cables, a saber, SCK (línea de reloj en serie), MISO (Master Out Slave In), MOSI (Master In Slave Out) y SS / CS (selección de chip). A diferencia de UART,SPI utiliza un formato de maestro a esclavo para controlar varios dispositivos esclavos con un solo maestro.

MISO y MOSI actúan como Tx y Rx de UART utilizados para transmitir y recibir datos. Chip Select se utilizapara seleccionar con qué esclavo quiere comunicarse el maestro.

Dado que SPI es un protocolo síncrono, utiliza un reloj integrado del maestro para garantizar que tanto el

dispositivo maestro como el esclavo funcionen en la misma frecuencia. Esto significa que los dos dispositivosya no tienen que negociar una velocidad en baudios.

I2C (circuito interintegrado): I2C es otro protocolo serial síncrono como SPI, pero con varias ventajas. Estos incluyen la capacidad de tener múltiples maestros y esclavos, direccionamiento simple (no se requiere Chip Select), operación con varios voltajes y usar solo dos cables conectados a dos resistencias pull-up. I2C se utiliza a menudo en muchos dispositivos de IoT, equipos industriales y electrónica de consumo. Los dos pinesen un protocolo I2C son la línea de datos en serie (SDA) que transmite y recibe datos y el pin de la línea de reloj en serie (SCL), que funciona como un reloj.

SE PUEDE LLEGAR A LA CONCLUSION QUE LOS PROTOCOLOS **SPI, UART o I2C NO SE USAN ENESTA CONFIGURACION YA QUE NO SON LOS ADECUADOS EN ESTE TIPO DE CONEXIÓN NI CONFIGURACION.**

f) Enumere otros protocolos de comunicación que no sean de campo y descríbalos brevemente.

Los protocolos que se utilizan en los sistemas embebidos son los que se obtienen de los protocolos existentes en las computadoras y los sistemas más grandes y, en ocasiones, se adoptan de los que se utilizan en la electrónica industrial o de consumo. Estos mismos protocolos se han modificado a lo largo de los años, y algunos se han desechado, mientras que otros son completamente funcionales y se mantienen robustos incluso en las condiciones más adversas.

Protocolos Paralelos
La comunicación en paralelo transmite todos los bits de datos al mismo tiempo, por lo que es rápida, pero tiene la desventaja de usar muchos cables, lo que la hace más costosa y susceptible al ruido eléctrico. Esto es lo que los hace casi inexistentes en los campos relacionados con los microcontroladores, ya que aumenta la complejidad de los circuitos periféricos del microcontrolador y sacrifica una gran cantidad de pines para mantener el bus de comunicación.
Estos inconvenientes son los que llevaron a la adopción de protocolos en serie.

Existen dos formas básicas de realizar transferencias de datos, las cuales son:

La *comunicación sincrónica* es una técnica que consiste en enviar una trama de datos (juego de caracteres) que configura un bloque de información que comienza con un conjunto de bits de sincronización (SYN) y finaliza con otro conjunto de bits de fin de bloque (ETB). En este caso, el bit de sincronización tiene la función de sincronizar los relojes existentes en el transmisor y el receptor, controlando así la duración de cada bit y carácter.

La *comunicación* asincrónica en donde el emisor decide cuándo enviar el mensaje a través de la red, y el receptor no sabe cuándo llega el mensaje. Para ello se utiliza un bit de cabecera que va al principio de cada carácter y uno o dos bits de parada al final del mismo carácter, esto se hace para que tanto el emisor como el receptor puedan sincronizar sus relojes y poder decodificar el mensaje.
En este tipo de transmisión no se maneja mucha velocidad ya que cada carácter es transmitido de uno en uno y por lo tanto puede ser un poco lenta.

CAN (Controller Area Network) es un protocolo serial creado por la empresa alemana Bosch a mediados de los años ochenta. Está optimizado para enviar pequeñas cantidades de datos entre múltiples nodos, además tiene una tasa de transferencia máxima de 1 MB/s, sin embargo, su funcionamiento a baja velocidad lo hace resistente al ruido y le permite cubrir largas distancias.
El bus CAN se diseñó originalmente para automóviles, pero también es popular en otras áreas, como el control de la línea de ensamblaje industrial, los protocolos de transmisión en el barco y más. A pesar del creciente éxito, las especificaciones de Bosch no definen estándares para voltajes, conectores o cables: cada organización define múltiples estándares a nivel físico. La capa física más común y utilizada es la norma ISO 11898-1, pero se puede implementar de otras formas.

El protocolo ***RS-232*** fue introducido por primera vez en 1962 por la División de Radio de Electronic Industries Alliance (EIA). Este protocolo se utilizó originalmente para la comunicación entre dispositivos y se conoce en la jerga como dispositivos DTE (Data Terminal Equipment) y dispositivos DCE (Data Communication Equipment). DTE es un dispositivo que convierte la información del usuario en señales o convierte las señales recibidas. Los primeros dispositivos DTE fueron teletipos, mientras que los primeros dispositivos DCE solían ser módems, que a su vez transmitían datos a través de líneas telefónicas o transmisores de radio para formar paquetes. La versión actual del protocolo es TIA-232-F "Interfaz entre el equipo terminal de datos y el equipo de terminación del circuito de datos mediante el intercambio de datos binarios en serie" publicado en 1997.

El protocolo ***RS-485***, también conocido como EIA-485, lleva el nombre del comité que formuló el estándar en 1983, es el estándar de comunicación de bus de capa física del modelo OSI.
Se define como un sistema de bus de transmisión multipunto diferencia, es ideal para transmisiones de alta velocidad a larga distancia (35 Mbit/s hasta 10 metros y 100 kbit/s en 1200 metros) y a través de canales de ruido, ya que reduce el ruido que aparece en el voltaje generado por la línea de transmisión.
El medio físico de transmisión es un par intercalado, soporta hasta 32 estaciones en 1 solo hilo, tiene una longitud máxima de 1200 metros, opera entre 300 y 200 bits/seg, la comunicación half-duplex (semi duplex) soporta hasta 32 transmisiones y 32 receptores.
Permite múltiples drivers para brindar la posibilidad de configuración multipunto. Dado que es un estándar bastante abierto, permite muchas configuraciones y usos muy diferentes, y su inmunidad al ruido lo hace ideal para entornos sensibles o industriales.

El acrónimo ***USB***, mejor conocido como Universal Serial Bus, es un bus estándar de la industria que define los cables, conectores y protocolos utilizados en un bus para realizar conexiones entre

computadoras, periféricos y dispositivos electrónicos, de comunicación y de alimentación.
Su desarrollo comenzó con un grupo de empresas de la industria que buscaban unificar la forma en que los periféricos se conectaban a las computadoras, que en ese momento no eran muy compatibles entre sí, incluidas Intel, Microsoft, IBM, Compaq, DEC, NEC y Nortel.
La primera especificación 1.0 completa se lanzó en 1996, pero se volvió ampliamente utilizada en 1998 con la especificación 1.1.
A partir de 2004, hay alrededor de 6 mil millones de dispositivos en el mercado global, con alrededor de 2 mil millones vendidos anualmente.