CARRERA: TECNICATURA SUPERIOR EN TELECOMUNICACIONES
MATERIA: ELECTRONICA MICROCONTROLADA
DOCENTES:

- JORGE E. MORALES, INGENIERO ELECTRICISTA ELECTRÓNICO (UNIVERSIDAD NACIONAL DE CÓRDOBA)
- C. GONZALO VERA, TECNICO SUPERIOR EN MECATRONICA (U.T.N.)

SHIELDS

CORDOBA, 4 DE SEPTIEMBRE DEL 2022.-

GRUPO N°2:

- DARIO ARRIOLA
- JEREMIAS CASTRO
- CARLA ARGENTINA WAYAR
- GAZZOLA, Oscar Antonio

Shields v1.0

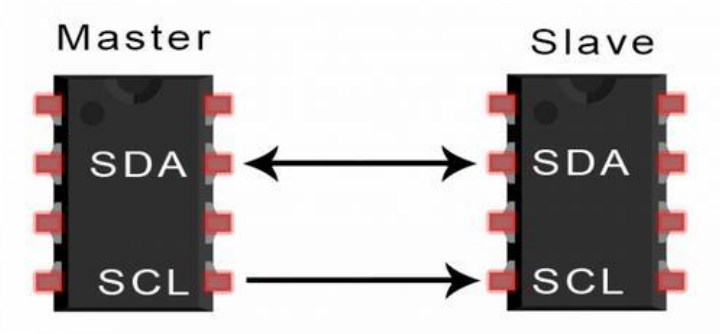
Ejercicio #1

a) Explique el funcionamiento del protocolo I2C

El bus de **comunicaciones I2C** (nombrado a veces como **I cuadrado C**, I^2) es un protocolo que se efectúa por medio de DOS hilos, bidireccional y sencillo desarrollado por Philips. Solo necesita 2 cables para transferir información y a través de estos dos hilos pueden conectarse diferentes dispositivos donde algunos de ellos serán **MAESTROS** en cuanto muchos otros dispositivos serán **ESCLAVOS**.

I2C combina las ventajas de SPI y UART. Con **I2C**, se pueden conectar varios dispositivos esclavos a un solo dispositivo maestro (como SPI), y varios dispositivos maestros pueden controlar uno o varios dispositivos esclavos. Esto es útil cuando desea que varios microcontroladores registren datos en una sola tarjeta de memoria o muestren texto en una sola pantalla LCD.

Al igual que la comunicación UART, **I2C** solo usa dos cables para transferir datos entre dispositivos:



SDA (DATOS EN SERIE): LA LÍNEA UTILIZADA POR EL MAESTRO Y EL ESCLAVO PARA ENVIAR Y RECIBIR DATOS.

SCL (SERIAL CLOCK): LA LÍNEA QUE TRANSPORTA LA SEÑAL DEL RELOJ.

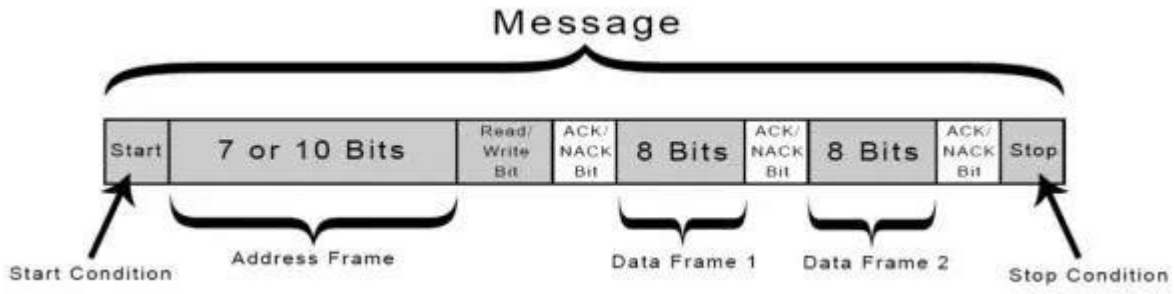
I2C ES UN PROTOCOLO DE COMUNICACIÓN EN SERIE, POR LO QUE LOS DATOS SE TRANSMITEN BIT A BIT A LO LARGO DE UN SOLO CABLE (CABLE SDA).

COMO SPI, I2C ES SÍNCRONO, POR LO QUE LA SALIDA DE BITS SE SINCRONIZA CON EL MUESTREO DE BITS A TRAVÉS DE UNA SEÑAL DE RELOJ COMPARTIDA ENTRE EL MAESTRO Y EL ESCLAVO. LA SEÑAL DEL RELOJ SIEMPRE LA CONTROLA EL ANFITRIÓN.

Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

Cómo funciona I2C

Cuando se usa **I2C**, los datos se convierten en mensajes y los mensajes se descomponen en marcos de datos. Cada mensaje tiene una trama de dirección, que contiene la dirección binaria de la estación esclava y una o más tramas de datos que contienen los datos que se transmiten. El mensaje también incluye las condiciones de inicio y parada entre cada trama de datos, bits de lectura / escritura y bits ACK / NACK:



Condición de inicio: antes de que la línea SCL cambie de nivel alto a nivel bajo, la línea SDA cambia de nivel alto a nivel bajo.

Condición de parada: después de que la línea SCL cambia de nivel bajo a nivel alto, la línea SDA cambia de nivel bajo a nivel alto.

Trama de dirección: una secuencia única de 7 o 10 bits para cada estación esclava, que se utiliza para identificar la estación esclava cuando la estación maestra desea comunicarse con ella.

Bit de lectura / escritura: un solo bit que especifica si el dispositivo maestro envía datos al dispositivo esclavo (nivel de voltaje bajo) o si el dispositivo esclavo solicita datos (nivel de voltaje alto).

Bit ACK / NACK: cada trama del mensaje va seguida de un bit de reconocimiento / no reconocimiento. Si la trama de dirección o la trama de datos se recibe con éxito, el bit ACK se devuelve desde el dispositivo receptor al remitente.

I2C no tiene una línea de selección esclava como SPI, por lo que necesita otra forma de que el dispositivo esclavo sepa que se le están enviando datos en lugar de otro dispositivo esclavo. Lo hace por dirección. La trama de dirección es siempre la primera trama después del bit de inicio en un mensaje nuevo. El dispositivo maestro envía la dirección del dispositivo esclavo que se comunica con él a cada dispositivo esclavo conectado a él. Luego, cada dispositivo esclavo compara la dirección enviada desde el dispositivo maestro con su propia dirección. Si la dirección coincide, el bit ACK de bajo voltaje se envía de vuelta al host. Si la dirección no coincide, el dispositivo esclavo no realiza ninguna operación y la línea SDA permanece alta.

Bit de lectura / escritura

La trama de dirección incluye un bit al final para notificar al dispositivo esclavo si el dispositivo maestro desea escribir datos en él o recibir datos del dispositivo maestro. Si el dispositivo maestro desea enviar datos al dispositivo esclavo, el bit de lectura / escritura es bajo. Si el dispositivo maestro solicita datos del dispositivo esclavo, este bit es alto.

Marco de datos

Una vez que el dispositivo maestro detecta el bit ACK del dispositivo esclavo, está listo para enviar la primera trama de datos.

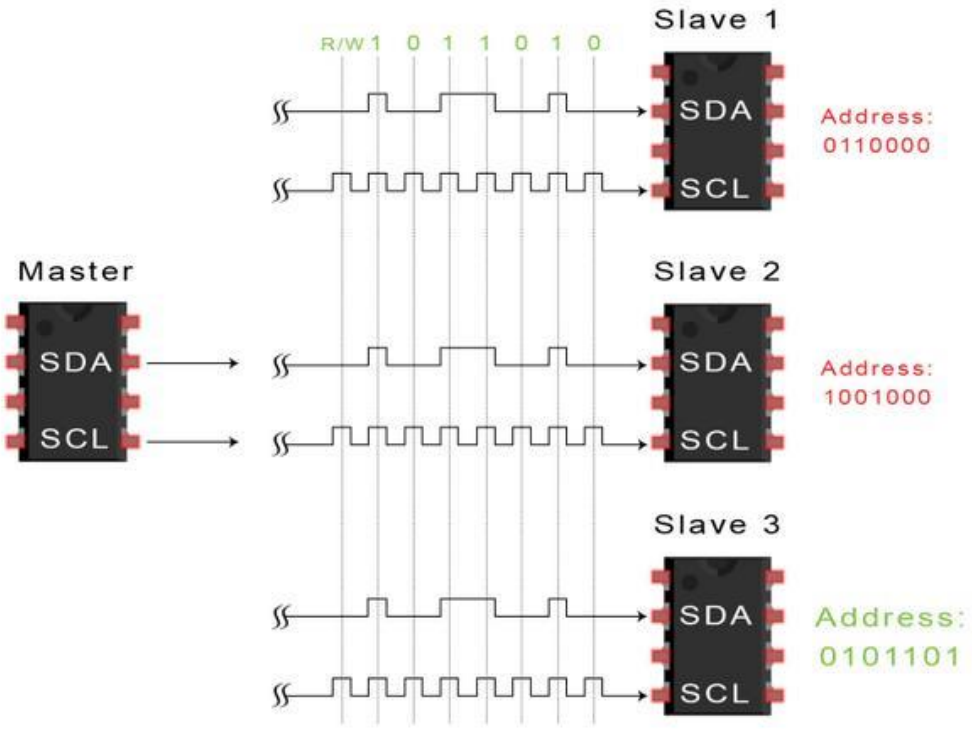
La trama de datos siempre tiene 8 bits de longitud y se envía con el bit más significativo primero. Siga el bit ACK / NACK de cada trama de datos para verificar que la trama se haya recibido correctamente. Antes de enviar la siguiente trama de datos, el maestro o esclavo debe recibir el bit ACK (dependiendo de la persona que envía los datos).

Después de enviar todas las tramas de datos, el dispositivo maestro puede enviar una condición de parada al dispositivo esclavo para detener la transmisión. La condición de parada es que después de que la línea SCL cambia de baja a alta, la línea SDA cambia de baja a alta y la línea SCL permanece alta.

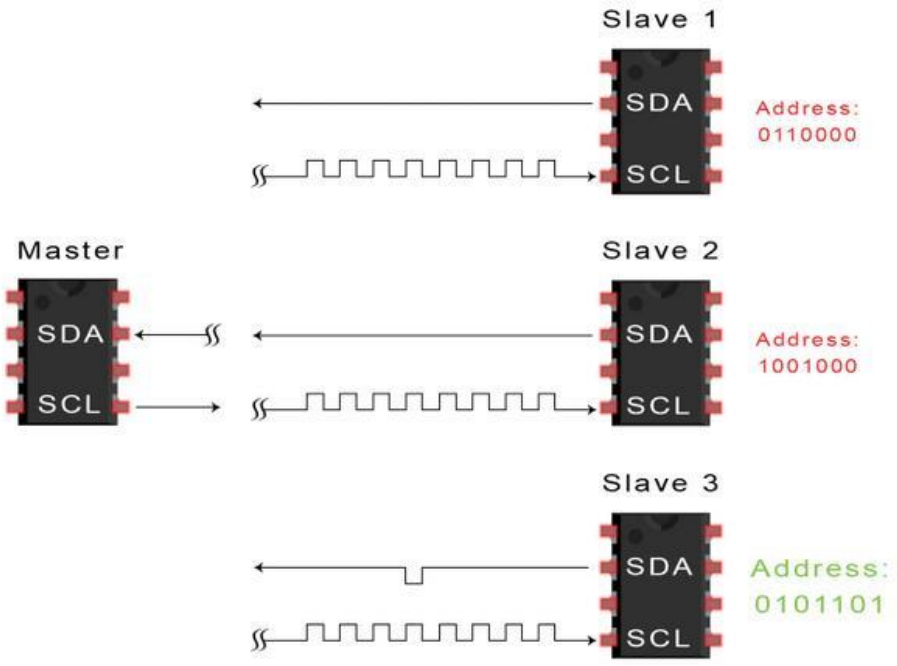
Pasos de transmisión de datos I2C

1. El host envía datos a cada dispositivo esclavo conectado y luego cambia la señal SDA de alta a baja, y luego cambia SCL de alta a baja.

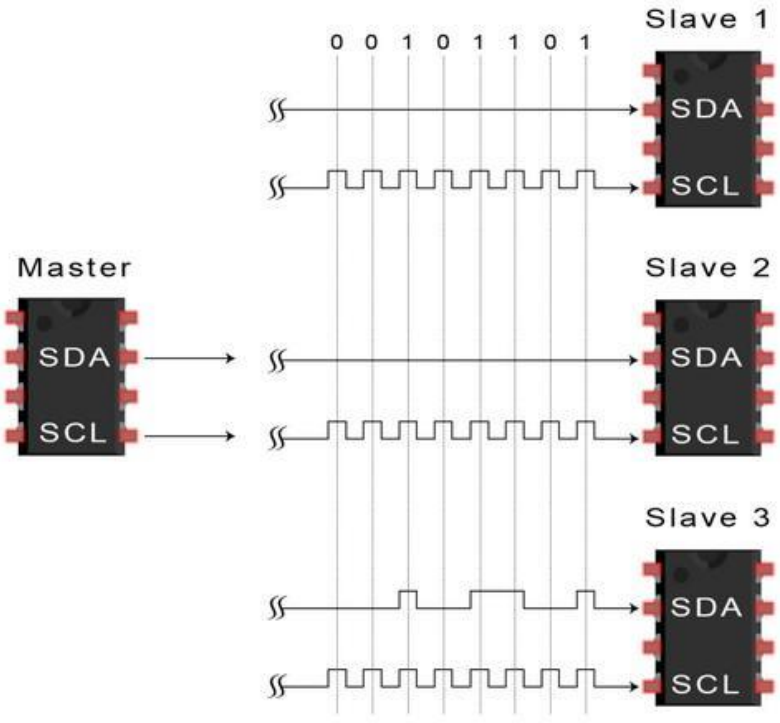
2. El dispositivo maestro envía a cada dispositivo esclavo la dirección de 7 o 10 bits del dispositivo esclavo con el que desea comunicarse y el bit de lectura / escritura:



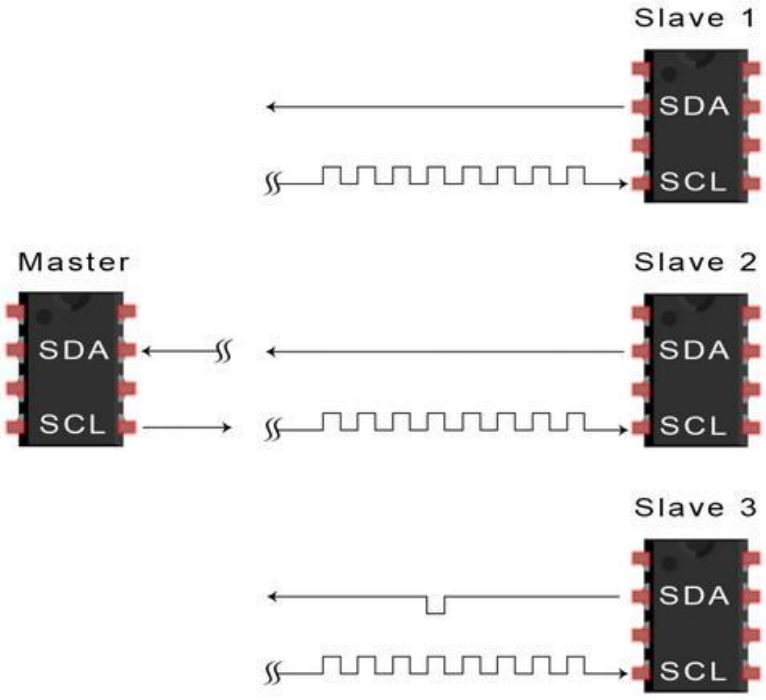
3. Cada dispositivo esclavo compara la dirección enviada por el dispositivo maestro con su propia dirección. Si la dirección coincide, el dispositivo esclavo devuelve el bit ACK bajando la línea SDA. Si la dirección del dispositivo maestro no coincide con la dirección del dispositivo esclavo, el dispositivo esclavo mantiene la línea SDA alta.



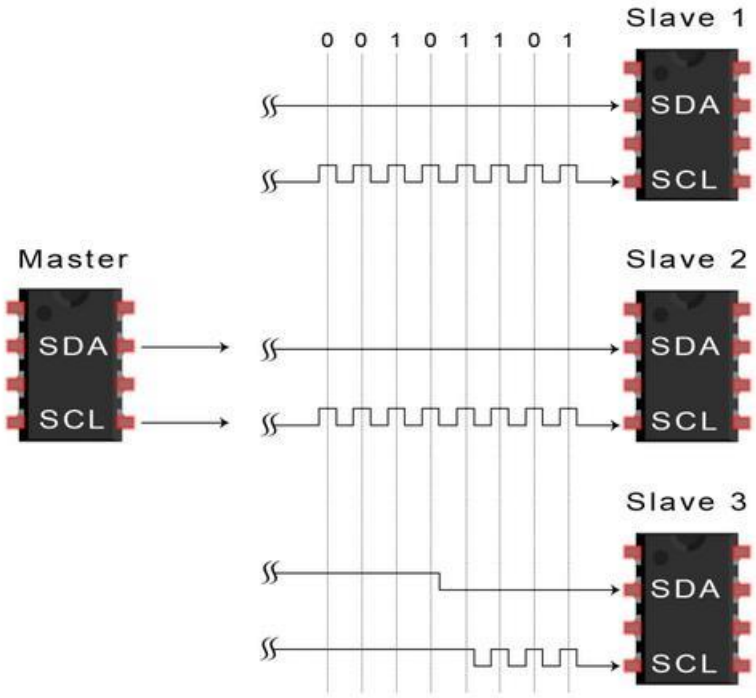
4. El dispositivo maestro envía o recibe tramas de datos:



5. Después de transmitir cada trama de datos, el dispositivo receptor devuelve otro bit ACK a remitente para confirmar la recepción exitosa de la trama:

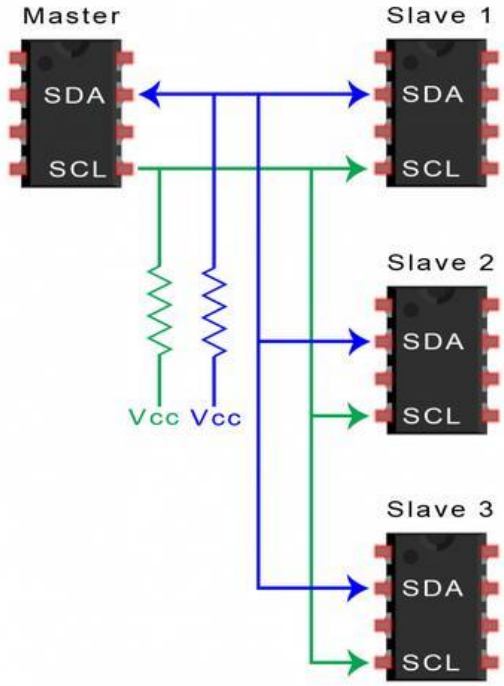


6. Para detener la transmisión de datos, el maestro envía una condición de detención al esclavo cambiando SCL a alto antes de cambiar SDA a alto:



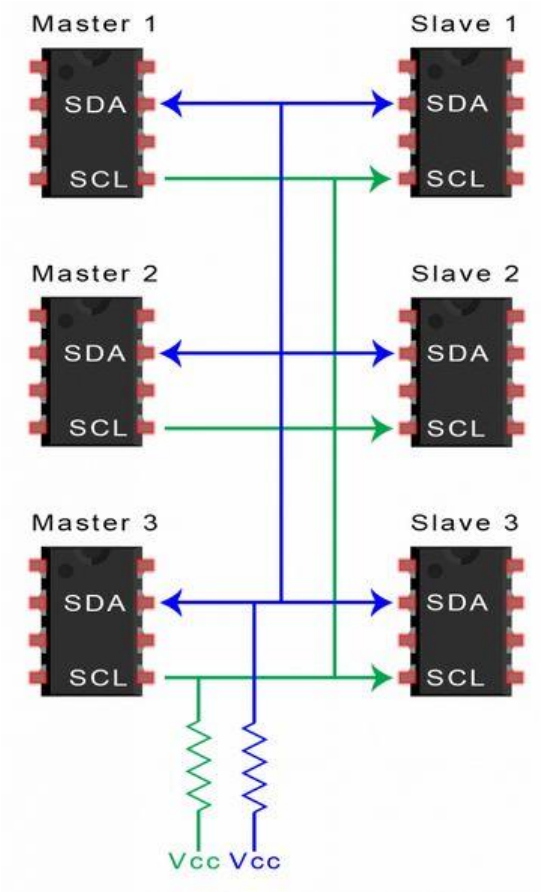
Maestro único con varios esclavos

Debido a que **I2C** utiliza direccionamiento, se pueden controlar varios dispositivos esclavos desde un solo dispositivo maestro. Con direcciones de 7 bits, se pueden utilizar 128 (27) direcciones únicas. No es común utilizar direcciones de 10 bits, pero se proporcionan 1.024 (210) direcciones únicas. Para conectar varios dispositivos esclavos a un solo dispositivo maestro, conéctelos así, usando resistencias pull-up de 4.7K ohmios para conectar las líneas SDA y SCL a Vcc:



Hay varios dispositivos esclavos y varios dispositivos maestros

Se pueden conectar varios dispositivos maestros a un solo dispositivo esclavo o múltiples dispositivos esclavos. Cuando dos dispositivos maestros intentan enviar o recibir datos a través de la línea SDA al mismo tiempo, existe un problema con varios dispositivos maestros en el mismo sistema. Para resolver este problema, cada dispositivo maestro necesita detectar si la línea SDA es baja o alta antes de enviar un mensaje. Si la línea SDA es baja, significa que otro controlador principal ha controlado el bus y el controlador principal debe esperar para enviar un mensaje. Si la línea SDA es alta, la transmisión de información es segura. Para conectar múltiples dispositivos maestros a múltiples dispositivos esclavos, use el siguiente diagrama para conectar las líneas SDA y SCL a Vcc con resistencias pull-up de 4.7K ohmios:



Ventajas y desventajas de I2C

Comparado con otros protocolos, I2C suena muy complicado, no es fácil de implementar en el programa y causa problemas como pérdida de datos, no respuesta, "muertos, etc.".

Ventajas:

Utiliza solo dos cables
Admite múltiples servidores maestros y múltiples servidores esclavos
El bit ACK / NACK confirma que todas las tramas se han transmitido correctamente
El HW no es tan complicado como UART
Protocolo conocido y ampliamente utilizado

Desventajas:

La tasa de Transferencia de datos es más lenta que SPI
El tamaño del marco de datos esta imitado a 8 bits
Implementar HW más complejo que SPI

b) Qué son los sensores resistivos?

Los Sensores Resistivos son los sensores cuyo principio físico de funcionamiento es la variación de la resistencia eléctrica de un componente eléctrico o electrónico.

Básicamente, un sensor resistivo es un dispositivo que convierte cantidades físicas medidas tales como desplazamiento, deformación, fuerza, aceleración, humedad, temperatura, etc. en valores de resistencia.

Según la variable física que hace variar la resistencia se clasifican en:

- Potenciómetros (Potentiometers)
- Galgas extensométricas (Strain gages)
- Sensores de temperatura de resistencia metálica o RTD (Resistance Temperature Detectors) Termistores [Thermistors (Thermal Resistors)]
- Magnetorresistencias (Magnetoresistors)
- Fotorresistencias o LDR (Light Dependent Resistors)
- Higrómetros (Hygrometers)
- Sensores y detectores de gases
- Sensores de conductividad de líquidos

Cómo se conectan a través de un divisor resistivo?

Un divisor Resistivo ó Divisor de Tensión, es una configuración de circuito eléctrico que reparte la tensión eléctrica de una fuente entre una o más impedancias conectadas en serie.

En electrónica y electricidad se usa para alimentar (proporcionar tensión de alimentación) a un aparato, con una tensión más pequeña que la que proporcionan las pilas o baterías disponibles.

Se trata de un circuito práctico de amplia utilización en electrónica que permite obtener a partir de un generador de tensión de un valor dado, otro generador con una fracción de tensión cualquiera. Mediante este circuito es posible alimentar (proporcional tensión de alimentación o polarización) a un equipo de bajo consumo o a un componente electrónico como por ejemplo un transistor.

En definitiva, sirve para **obtener una tensión más pequeña partiendo de una tensión mayor**.

Qué es el acondicionamiento de señales?

El acondicionamiento de señal es un proceso de adquisición de datos que se lleva a cabo mediante un instrumento llamado acondicionador de señal. Ese instrumento convierte un tipo de señal eléctrica o mecánica (señal de entrada) en otro (señal de salida). El objetivo consiste en amplificar la señal y convertirla a otro formato fácil de leer y compatible con fines de adquisición de datos o de control de una máquina.

Un acondicionador de señal ayuda a obtener medidas precisas, como condición esencial para la exactitud de la adquisición de datos o del control de máquinas.

Cuando deseamos realizar una medida, lo primordial es tener un transductor que pueda transformar la medida física en una medida eléctrica, Esa medida eléctrica es necesario acondicionarla.

Para entender esto, acondicionar una señal significa realizar el siguiente proceso: Convertir la señal, modificar el nivel de la señal, linealizacion para una respuesta y por ultimo filtrar la señal. Un acondicionador de señal son pasos de adquisición de datos que es llevado acabo con un dispositivo que puede convertir un tipo de señal electrónica a otra. Su principal objetivo es convertir una señal que puede ser difícil de leer y convertirlas a otro formato para que tengan de una manera fácil de leer los datos adquiridos.

La mayoría de estas señales requieren de alguna manera de preparación antes de que sean digitalizadas. Por ejemplo las señales de un termopar, estas señales proporcionan unos niveles muy pequeños de tensión que deben ser amplificados antes que sean digitalizados. Podemos mencionar de otros sensores, como detectores de temperatura mediante resistencia (RTD), los termistores, las galgas extensiométricas y los acelerómetros, toda esta tecnología que se requiere un trabajo de preparación con lleva un acondicionador de señales.

Etapas del Acondicionamiento de señales

Amplificación.

Filtrado.

Linealizacion.

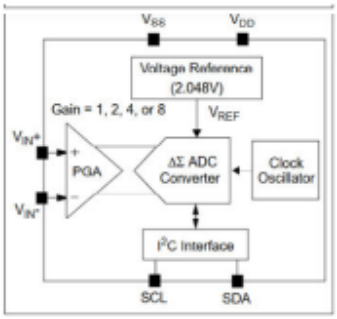
Conversión de **señal**.

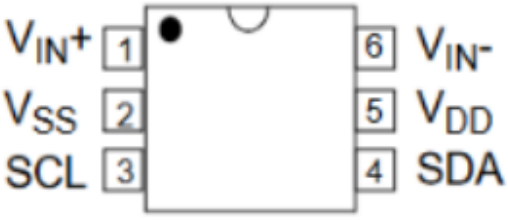
Multiplexores.

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

El MPC3421 se trata de un conversor analógico-digital de 18 bits con un único canal para señales diferenciales. Las características de este dispositivo son las siguientes:

- 18 bits de resolución.
- Auto calibración interna con offset y ganancia interna en cada conversión.
- Voltaje de referencia interno de 2.048V.
- Amplificador programable integrado.
- Oscilador integrado.
- INL de 10 ppm del fondo de escala.
- Cuatro velocidades de transmisión de datos configurable; 3.75 SPS (18 bits), 15 SPS (16 bits), 60 SPS (14 bits), 240 SPS (12 bits).
- Bajo consumo de funcionamiento.
- Interfaz I2C de 100kHz a 400 kHz.
- Tensión de operación de 2.7V a 5.5V.
- Encapsulado SOT23.

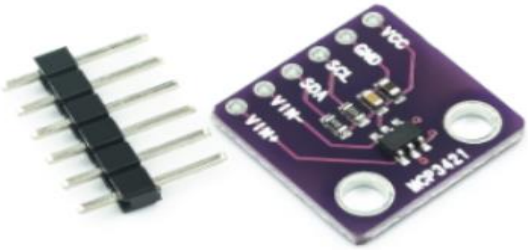


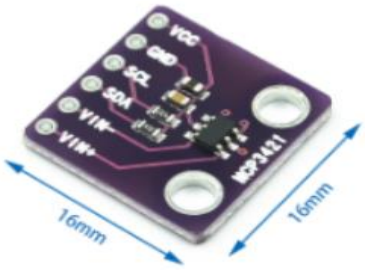


Esta board permite al usuario medir el voltaje de entrada desconocido con una cantidad mínima de preparación.

Muchas almohadillas de prueba, incluidas las SCL y SDA, permiten controlar las comunicaciones de datos de señales I2C.

El usuario puede conectar los pines SDA y SCL a su propia placa PCB y evaluar el rendimiento del dispositivo mcp3421.



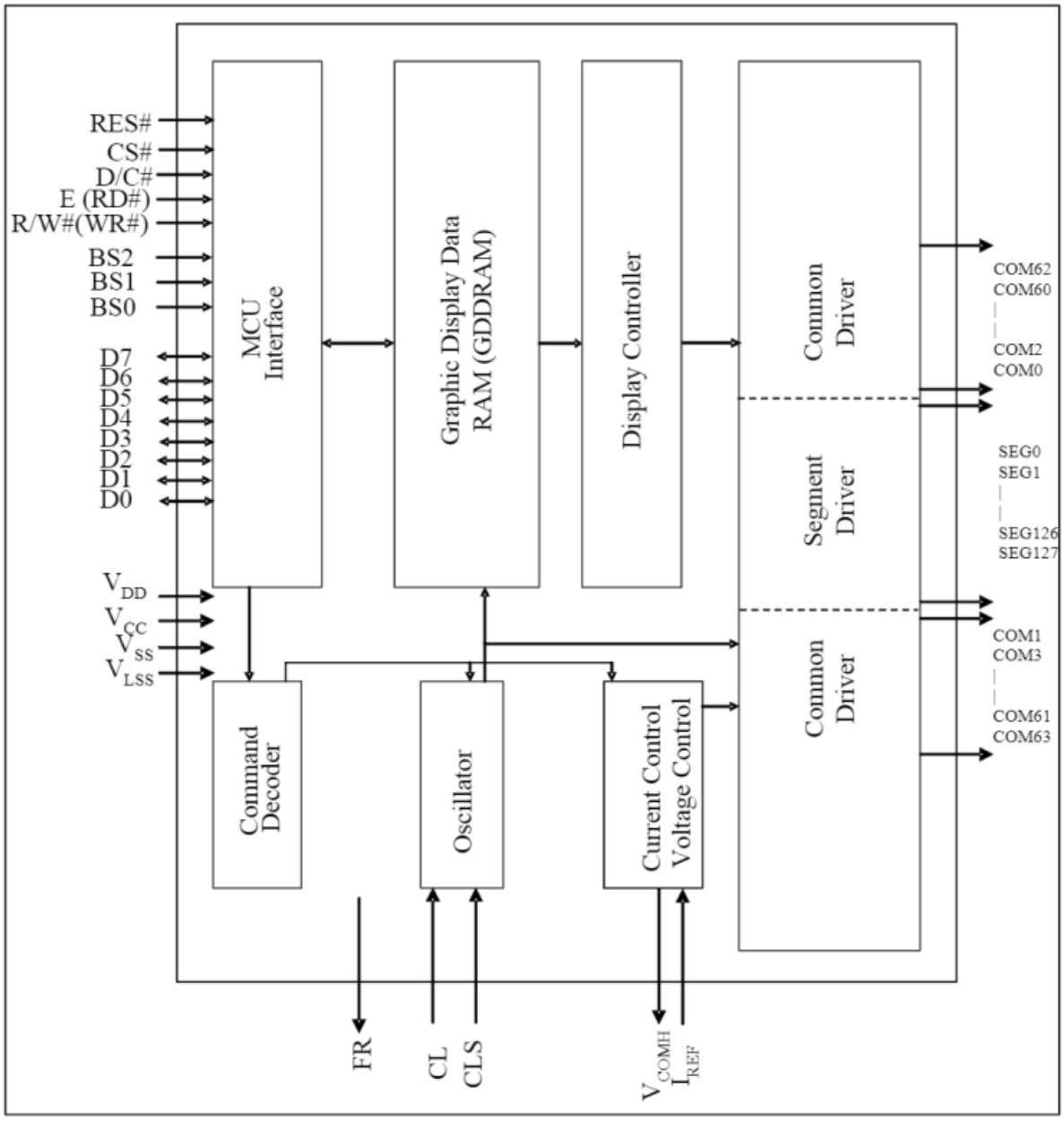


d)- ¿ Que es el controlador ssd1306(i2c). Existe alguna shield para controlar una pantalla oled 128x64?; si es así, implemente una práctica donde muestre el mensaje "Es fácil el desarrollo con shields".

El controlador ssd1306, es un controlador OLED CMOS de un solo chip con controlador para diodos de emisión de luz orgánica, el sistema de Visualización de gráficos de matriz de puntos de diodo consta de 128 segmentos y 64 comunes. Este circuito integrado es diseñado para panel OLED tipo cátodo común. El SSD1306 incorpora control de contraste, RAM de pantalla y oscilador, lo que reduce la cantidad de componentes externos y consumo de energía. Tiene control de brillo de 256 pasos. Los datos/comandos son enviado desde MCU general a través de la interfaz paralela compatible con la serie 6800/8000 seleccionable por hardware, I2C o SPI. Es adecuado para muchas aplicaciones portátiles compactas, como pantalla secundaria de teléfono móvil, reproductor de MP3 y calculadora, etc.

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, por ejemplo, la placa Arduino uno, se realiza mediante SPI o I2C. Generalmente, la comunicación SPI es más rápida que la comunicación I2C. Por el contrario, la comunicación SPI requiere de más pines que la comunicación I2C.

Diagrama de bloques SSD 1306:



Interfaz MCU I2C:

La interfaz de comunicación I2C consta del bit de dirección del esclavo SA0, la señal de datos del bus I2C SDA (SDAOUT/D2 para salida y SDAIN/D1 para la entrada) y la señal de reloj del bus I2C SCL (D0). Tanto las señales de datos como las de reloj deben estar conectadas a resistencias pull-up. RES# se utiliza para la inicialización del dispositivo.

I. Bit de dirección del esclavo (SA0)

SSD1306 tiene que reconocer la dirección esclava antes de transmitir o recibir cualquier información por el bus I2C. El dispositivo responderá a la dirección del esclavo seguido del bit de dirección del esclavo (bit "SA0") y el bit de selección de lectura/escritura (bit "R/W#") con el siguiente formato de byte, b7 b6 b5 b4 b3 b2 b1 b0 0 1 1 1 1 0 SA0 R/W# El bit “SA0” proporciona un bit de extensión para la dirección del esclavo. Se puede seleccionar "0111100" o "0111101" como la dirección esclava de SSD1306. El pin D/C# actúa como SA0 para la selección de

dirección de esclavo.
El bit “R/W#” se usa para determinar el modo de operación de la interfaz de bus I2C. R/W#=1, está en modo lectura. R/W#=0, está en modo escritura.

II. Señal de datos de bus I2C (SDA)

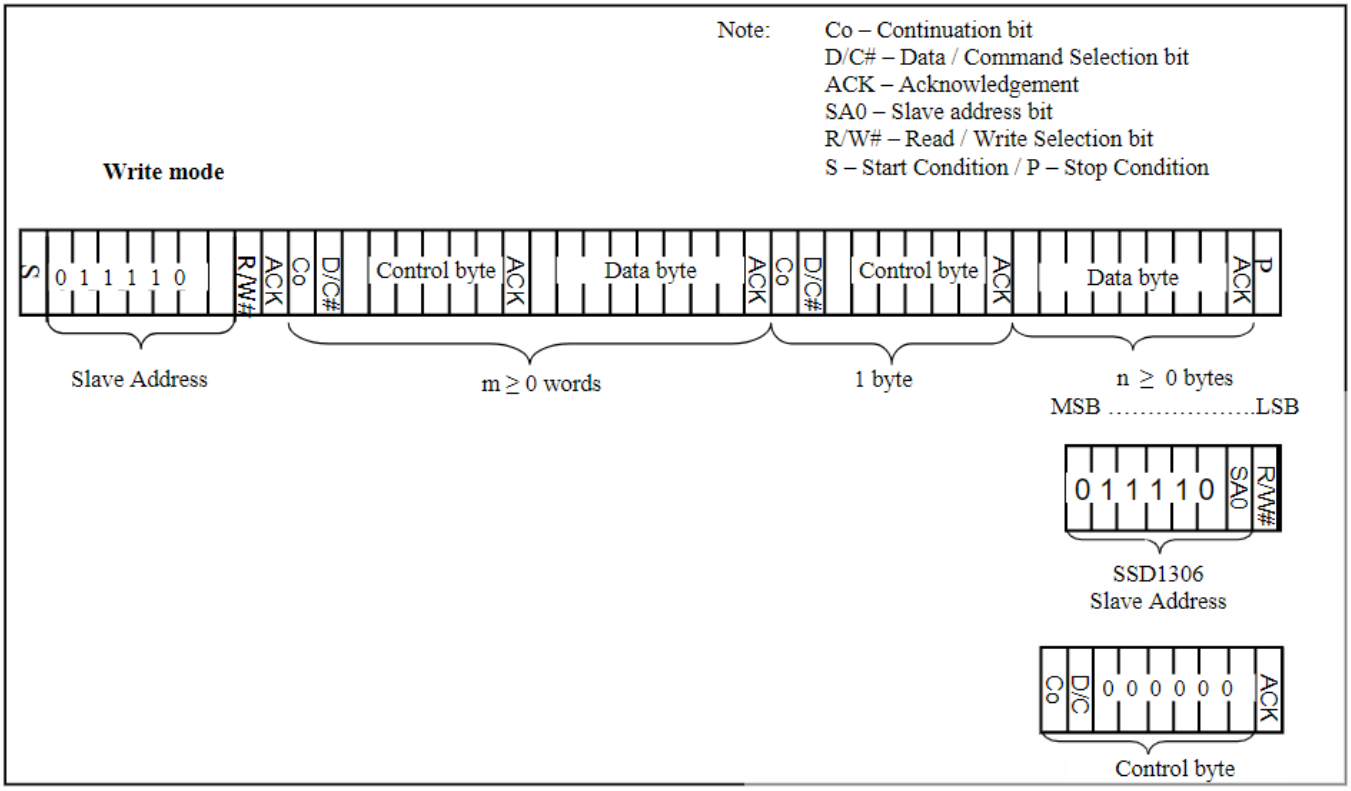
SDA actúa como un canal de comunicación entre el transmisor y el receptor. Los datos y el acuse de recibo se envían a través de la SDA.
Debe notarse que la resistencia de la pista ITO y la resistencia levantada en el pin "SDA" se convierte en un divisor de potencial de voltaje. Como resultado, el reconocimiento no sería posible alcanzar un nivel 0 lógico válido en “SDA”.
“SDAIN” y “SDAOUT” están unidos y sirven como SDA. El pin "SDAIN" debe estar conectado para actuar como SDA. El pin "SDAOUT" puede estar desconectado. Cuando se desconecta el pin "SDAOUT", la señal de reconocimiento se ignorará en el bus I2C.

III. Señal de reloj de bus I2C (SCL)

La transmisión de información en el bus I2C sigue una señal de reloj, SCL. Cada transmisión de bit de datos tiene lugar durante un solo período de reloj de SCL.

Bus I2C Escritura de datos

La interfaz de bus I2C da acceso para escribir datos y comandos en el dispositivo. Consulte la Figura 8-7 para conocer el modo de escritura del bus I2C en orden cronológico.

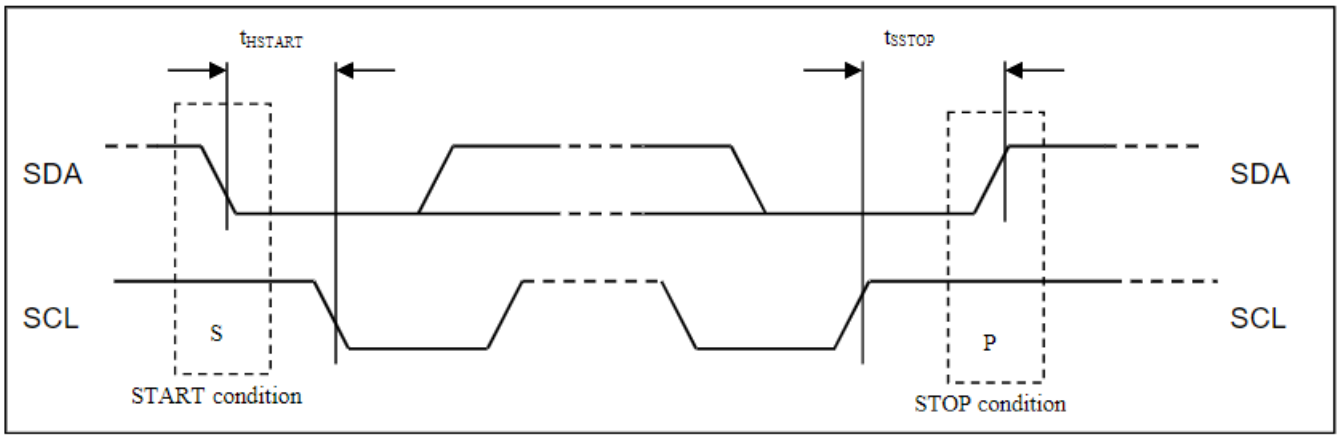


Modo de escritura para I2C

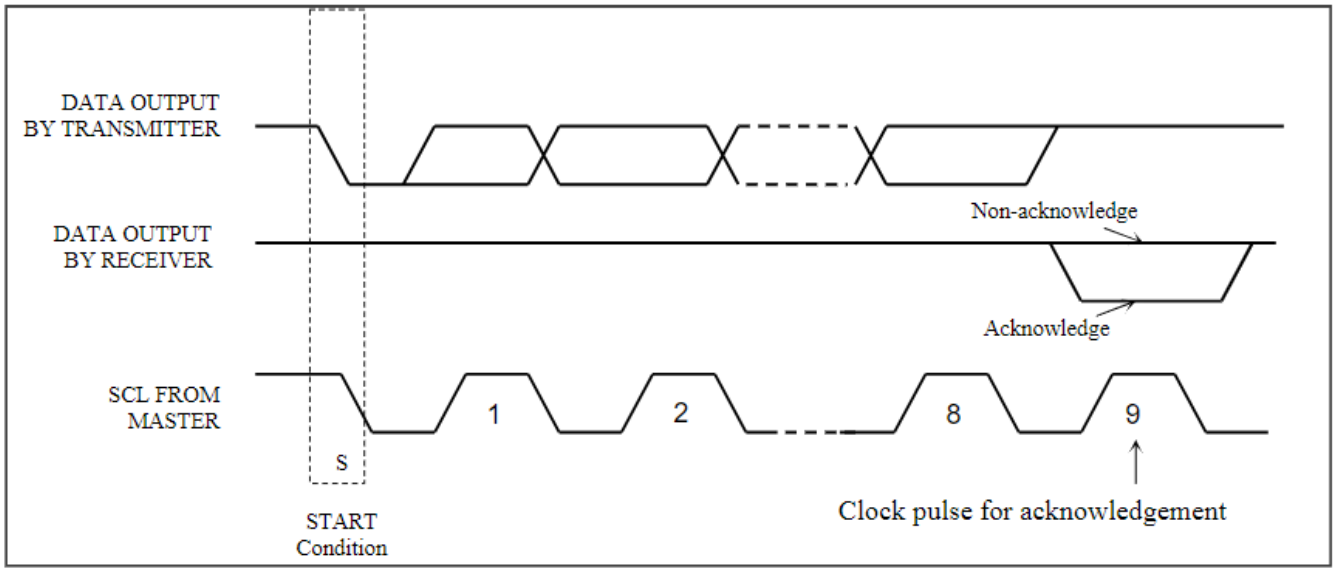
1) El dispositivo maestro inicia la comunicación de datos mediante una condición de inicio. La definición de la condición de inicio se muestra en la figura siguiente. La condición de inicio se establece tirando del SDA de ALTO a BAJO mientras el SCL permanece ALTO.
2) La dirección esclava sigue la condición de inicio para uso de reconocimiento. Para el SSD1306,

la dirección del esclavo es "b0111100" o "b0111101" al cambiar el SA0 a BAJO o ALTO (el pin D/C actúa como SA0).
3) El modo de escritura se establece configurando el bit R/W# en "0" lógico.
4) Se generará una señal de reconocimiento después de recibir un byte de datos, incluida la dirección del esclavo y el bit R/W#. Consulte la Figura siguiente para ver la representación gráfica de la señal de reconocimiento. El bit de reconocimiento se define cuando la línea SDA se baja durante el período ALTO del pulso de reloj relacionado con el reconocimiento.
5) Después de la transmisión de la dirección del esclavo, se puede enviar el byte de control o el byte de datos a través del SDA. Un byte de control consta principalmente de bits Co y D/C# seguidos de seis "0".
a. Si el bit Co se establece como "0" lógico, la transmisión de la siguiente información contendrá solo bytes de datos.
b. El bit D/C# determina que el siguiente byte de datos se actúa como un comando o un dato. Si el bit D/C# se establece en "0" lógico, define el siguiente byte de datos como un comando. Si el bit D/C# se establece en "1" lógico, define el siguiente byte de datos como datos que se almacenarán en la GDDRAM.
El puntero de dirección de la columna GDDRAM se incrementará en uno automáticamente después de cada escritura de datos.
6) Se generará un bit de reconocimiento después de recibir cada byte de control o byte de datos.
7) El modo de escritura finalizará cuando se aplique una condición de parada. La condición de parada se establece tirando del "SDA adentro" de BAJO a ALTO mientras el "SCL" permanece ALTO.

Definición de la condición de inicio y parada



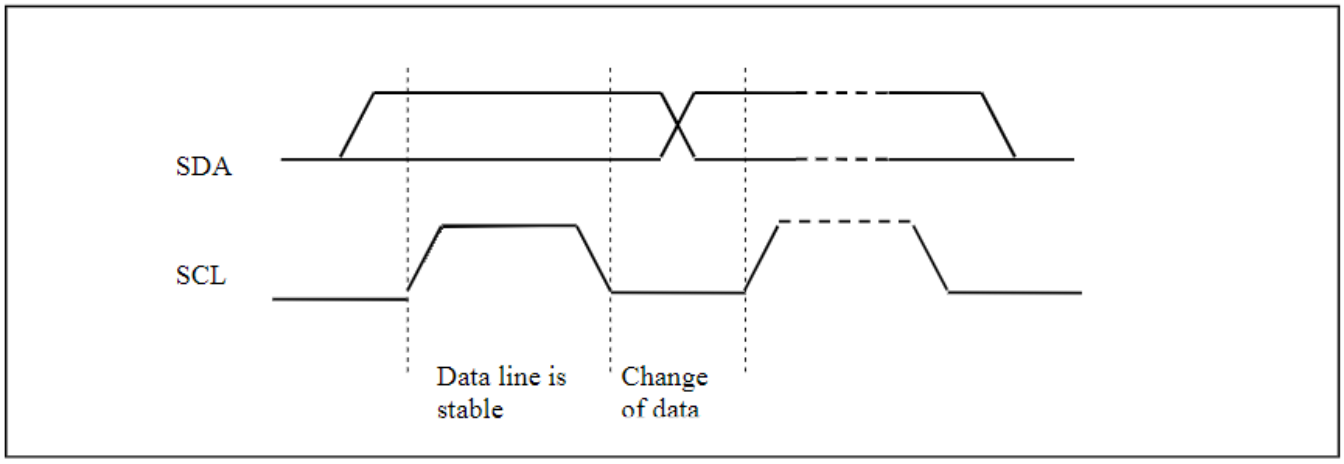
Definición de la condición de acuse de recibo

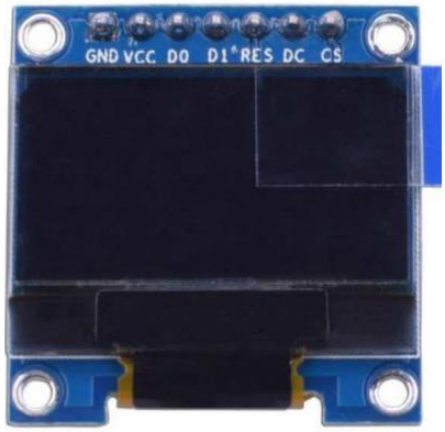


Definición de la condición de inicio y parada

Se debe tener en cuenta que la transmisión del bit de datos tiene algunas limitaciones. 1. El bit de datos, que se transmite durante cada pulso SCL, debe mantenerse en un estado estable dentro del período "ALTO" del pulso del reloj. Consulte la Figura 8-10 para ver las representaciones gráficas. Excepto en las condiciones de inicio o parada, la línea de datos se puede cambiar solo cuando el SCL es BAJO. 2.Tanto la línea de datos (SDA) como la línea de reloj (SCL) deben ser levantadas por resistencias externas.

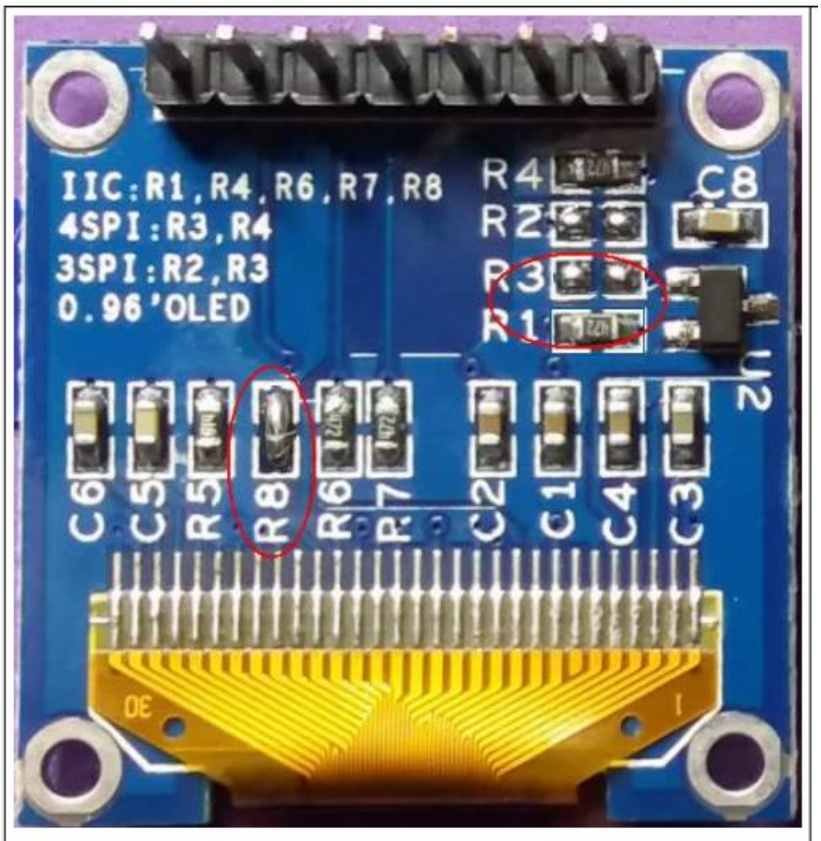
Definición de la condición de transferencia de datos





OLED 0,96 “ Chip SDD1306

NOTA: De fábrica vienen configuradas para trabajar en modo SPI. Para usarlas en modo I2C debe configurarse cambiando una resistencia de lugar y un puente.
Configuración OLED 0,96”
En este caso es similar. En R8 realizamos el puente y trasladamos R3 a la posición libre de R1.-



Configuración modo I2C OLED – 0,96”

Implementación de una práctica donde muestre elmensaje "Es fácil el desarrollo con shields".

Conexión I2C

- OLED GND – Arduino GND
- OLED VCC – Arduino +5V
- OLED CLK/(D0) – Arduino Uno A5 - SCL
- OLED MOSI7(D1) – Arduino Uno A4 - SDA
- OLED RES – Circuito Reset R/C ó pin Arduino por soft
- OLED DC - Arduino GND
- OLED CS – Arduino GND

Código: archivo main.cpp del repositorio del grupo 2 **Grupo2**/Proyecto Shields v1.0/**EJERCICIO 1**/

Archivo de la simulación: archivo PROYECTO_SHIELDS_PUNTO_1_D. del proyecto en el repositorio del grupo 2 **Grupo2**/Proyecto Shields v1.0/**EJERCICIO 1**/

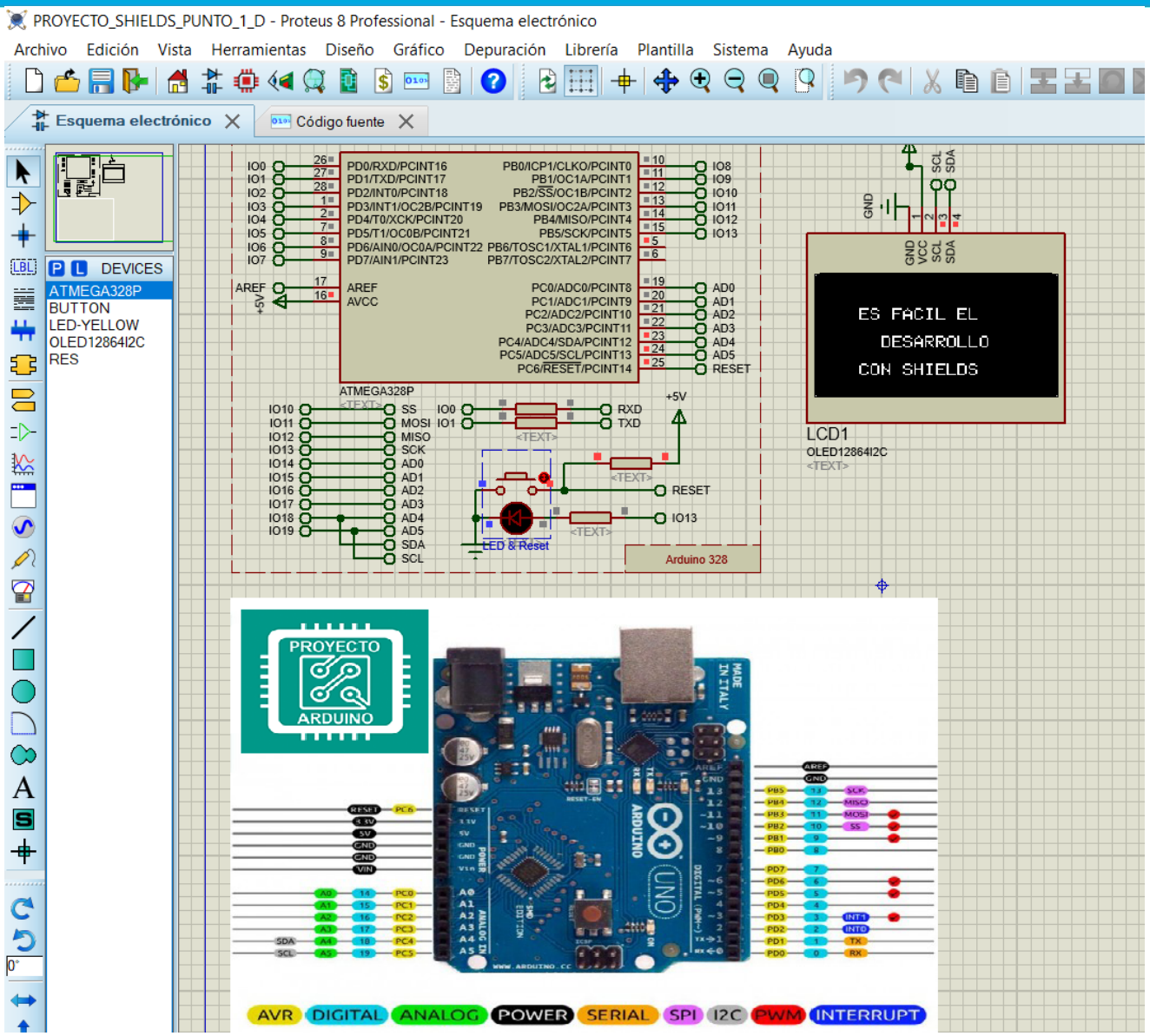


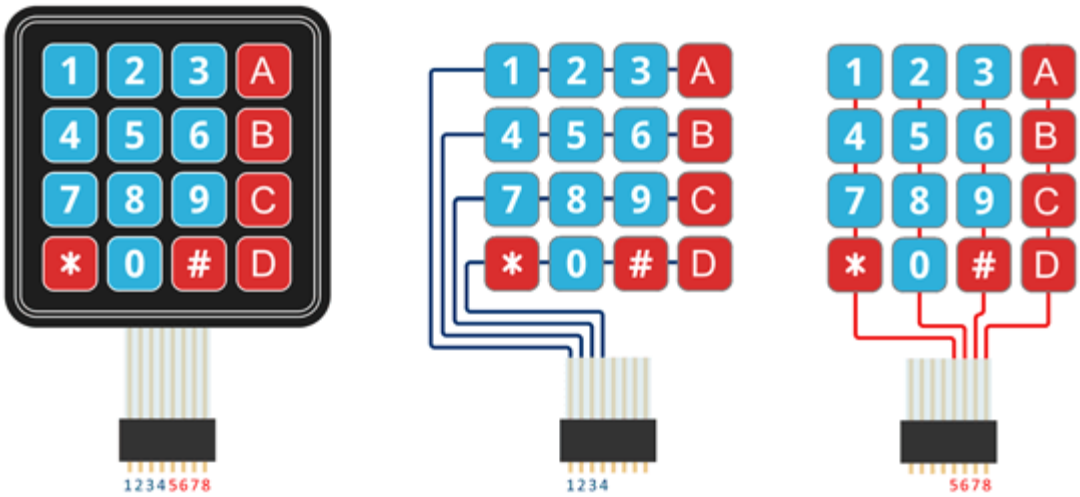
Imagen de la simulación en proteus de la pantalla OLED 128x64 con el controlador ssd1306.
Fuente: elaboración propia.

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

El teclado por si solo no es un shield, ya que no solo se trata de un hardware sino de un software que debe acompañar la implementación, como lo son generalmente las librerías, y cuya finalidad sea la de facilitar en un proyecto su implementación, al no especificar o no dar mas detalles no lo es.

El conjunto {teclado matricial + librerías de uso + repositorio (implementación, ejemplos, etc)} Arduino Shield.

Un teclado matricial es un dispositivo que agrupa varios pulsadores y permite controlarlos empleando un número de conductores inferior al que necesitaríamos al usarlos de forma individual. Podemos emplear estos teclados como un controlador para un autómata o un procesador como Arduino. Existen múltiples teclados con diferente número de teclas, siendo los más habituales las configuraciones de 3×3, 3×4 y 4×4.



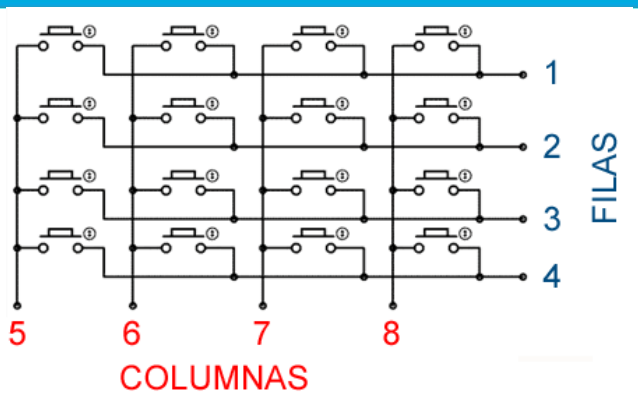
Teclado matricial de 4x4.

La detección de la pulsación de una tecla se hace de forma similar a la lectura simple de un pulsador. En resumen, se pone a tierra un extremo del pulsador, y el otro se conecta a una entrada digital con una resistencia de pull-up.

Para leer todas las teclas se haceun barrido por filas. En primer lugar, se pone todas las filas a 5V, y se define a todas las columnas como entradas con resistencia de pull-up.

Progresivamente se pone una fila a 0V, y se lee las entradas de la columna. Una vez realizada la lectura se vuelve a ponerla a 5V, se pasa a la siguiente fila, y se vuelve a realizar el progreso hasta recorrer todas las filas.

Para detectar NxM pulsadores necesitamos sólo N+M conductores.



Conexión interna de los pulsadores de un teclado matricial de 4x4

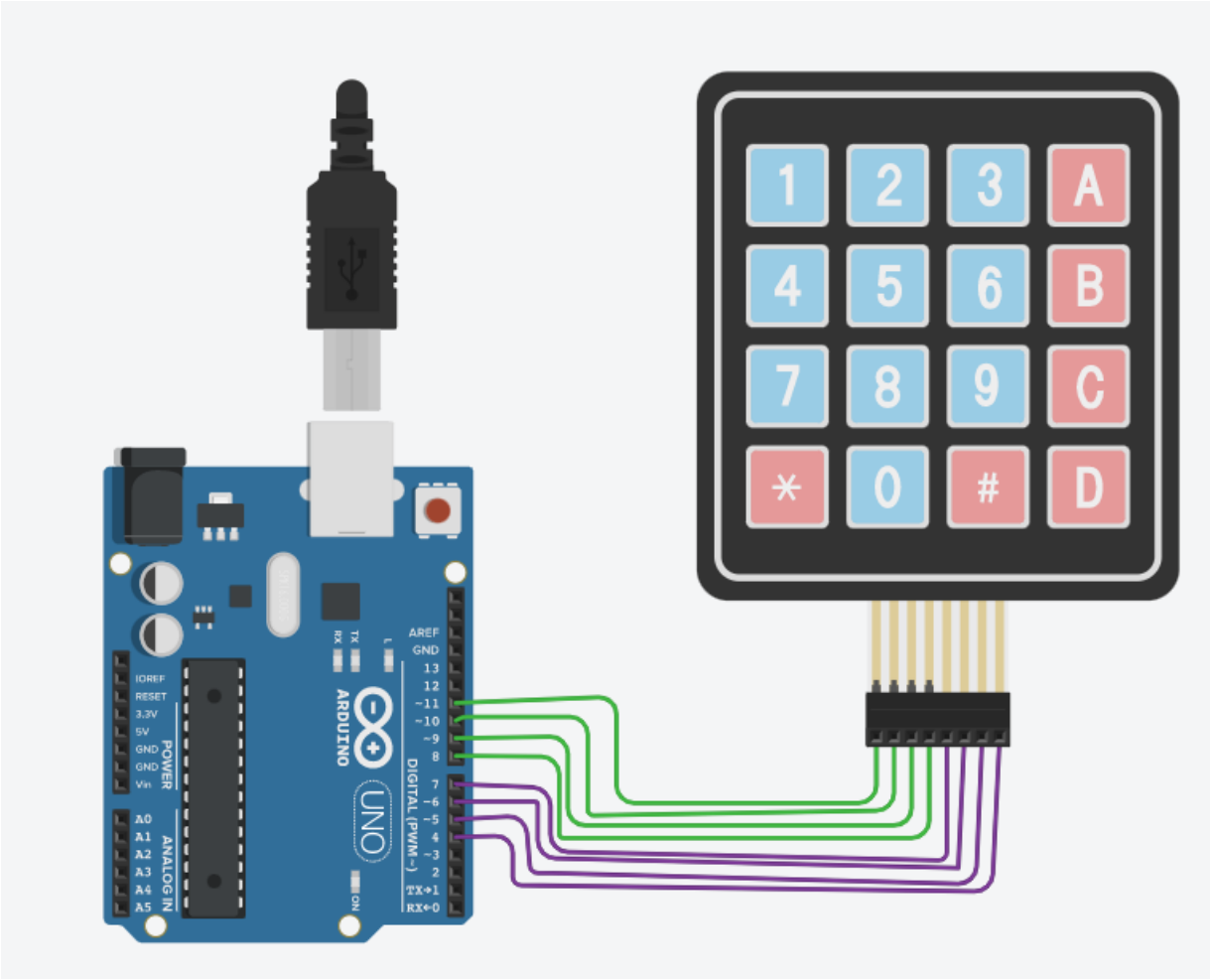
EJEMPLOS DE CÓDIGO CON LIBRERÍA

Existen varias librerías diseñaras para facilitar la lectura de teclados matriciales en Arduino. Por ejemplo, la librería Keypad, disponible en este enlace. La librería proporciona ejemplos de código, que resulta aconsejable revisar. El siguiente ejemplo es una modificación a partir de los disponibles en la librería.

```
#include <Keypad.h> // se incluye la librería para trabajar con teclado matricial

const byte fila = 4;// se define constantes para la cantidad de filas
const byte columna = 4;// se define constantes para la cantidad de columnas

 // se define la estructura del teclado matricial usando una matriz
char matricial [fila][columna] = {
  { '1','2','3', 'A' },
  { '4','5','6', 'B' },
  { '7','8','9', 'C' },
  { '#','0','*', 'D' }
};

byte pinfila[fila] = { 11, 10, 9, 8 };// se define los pines que se usan de la placa Arduino para las filas
byte pincolumna[columna] = { 7, 6, 5, 4 };// se define los pines que se usan de la placa Arduino para
      las columnas
 // se crea el objeto teclado con su estructura
Keypad teclado= Keypad(makeKeymap(matricial), pinfila, pincolumna, fila, columna);

void setup() {
  Serial.begin(9600);// se inicializa la comunicación serial
}

void loop() {
char tecla; // se define variable del tipo carácter
tecla = teclado.getKey();// se le asigna el valor de la tecla presionada

  if (tecla) {
    Serial.println(tecla);// se visualiza por el monitor serie la tecla presionada
  }
}
```

ESQUEMA DE CONEXIÓN

El esquema de conexión es sencillo. Simplemente se conecta todos los pines del teclado matricial a los pines de entradas digitales de la placa Arduino. Por ejemplo, en el ejemplo de un teclado de 4x4 el esquema quedaría de la siguiente forma.



Esquema de conexión , Fuente propia

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

Un shield es una placas o módulos de hardware libre o de una implementación que me facilita justamente el desarrollo de dispositivos electrónicos, se tiene que tener en cuenta que no solo es el hardware sino también el software, las librerías que se usan para su implementación por lo tanto también los protocolos de comunicación dentro de lo que es el mismo sistema embebido. Los shields tienen como objetivo facilitar el desarrollo de proyectos electrónicos, el montaje de todos los dispositivos necesarios para ese sistema embebido tanto como para la parte de entrada como la de salida o la visualización con mayor facilidad tanto a nivel hardware como software.
Como ejemplo de los shields es el ejemplo anterior, pero también se puede ver otros ejemplos de las placas- módulos que en conjunto con sus librerías que se pueden descargar de repositorios de forma libre con ejemplos de aplicación.

Ejemplo de placas- módulos – shields que se pueden usar con la placa Arduino.

¿Qué es el SPI?

La SPI fue desarrollada por Motorola (ahora parte de NXP Semiconductors) aproximadamente en 1985. Se trata de una interfaz serial síncrona prevista para la comunicación entre dispositivos a corta distancia. Desde entonces, se ha convertido en un estándar de-facto empleado por muchos fabricantes de semiconductores, especialmente en microprocesadores y microcontroladores.

El motivo de la popularidad de SPI radica en sus muchas ventajas. La primera es que es una interfaz direccionada de hardware simple que ofrece completa flexibilidad para la cantidad de bis transferidos. Usa un modelo de maestro-secundario con un maestro simple y puede manejar varios dispositivos secundarios usando comunicaciones dúplex que operan a velocidades de reloj de hasta 50 MHz. No usa un protocolo estándar y transfiere solo paquetes de datos, lo que la hace ideal para transferir flujos de datos largos.

SPI usa un máximo de cuatro líneas de señal (Figura 1). El dispositivo maestro, por lo general un procesador o controlador, suministra y controla el reloj (SCK) y líneas de selección de chip (CS). La operación multiplexor completa se maneja a través de las líneas de datos Master Out Slave In (MOSI) y Master In Slave Out (MISO). En un maestro individual simple, con configuración del dispositivo secundario individual, la línea de selección de chip puede eliminarse y se puede forzar la entrada de CS al dispositivo secundario al estado lógico habilitado. Si el dispositivo secundario solo puede enviar datos (comunicación semidúplex), luego la línea MOSI también puede eliminarse, y así reducir el conteo de señales adicionalmente. Los datos salen a través de la señal del reloj de tal forma que la transferencia de datos se asemeja a un registro de turnos con un bit cambiado para cada reloj.

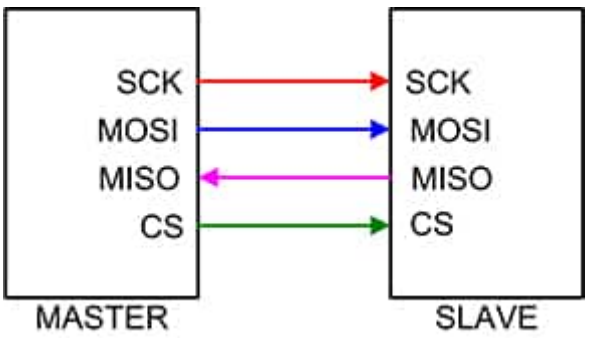


Figura 1: La conexión dúplex SPI básica usa dos líneas de datos (MOSI, MISO), una línea de reloj (SCK) y una línea de selección de chip (CS). MOSI en un dispositivo secundario es, a veces, rotulada como entrada de datos de dispositivo secundario (SDI). MISO puede

denominarse como salida de datos secundarios (SDO). (Fuente de la imagen: Digi-Key Electronics)

Hay dos enfoques para manejar varios dispositivos secundarios (Figura 2).

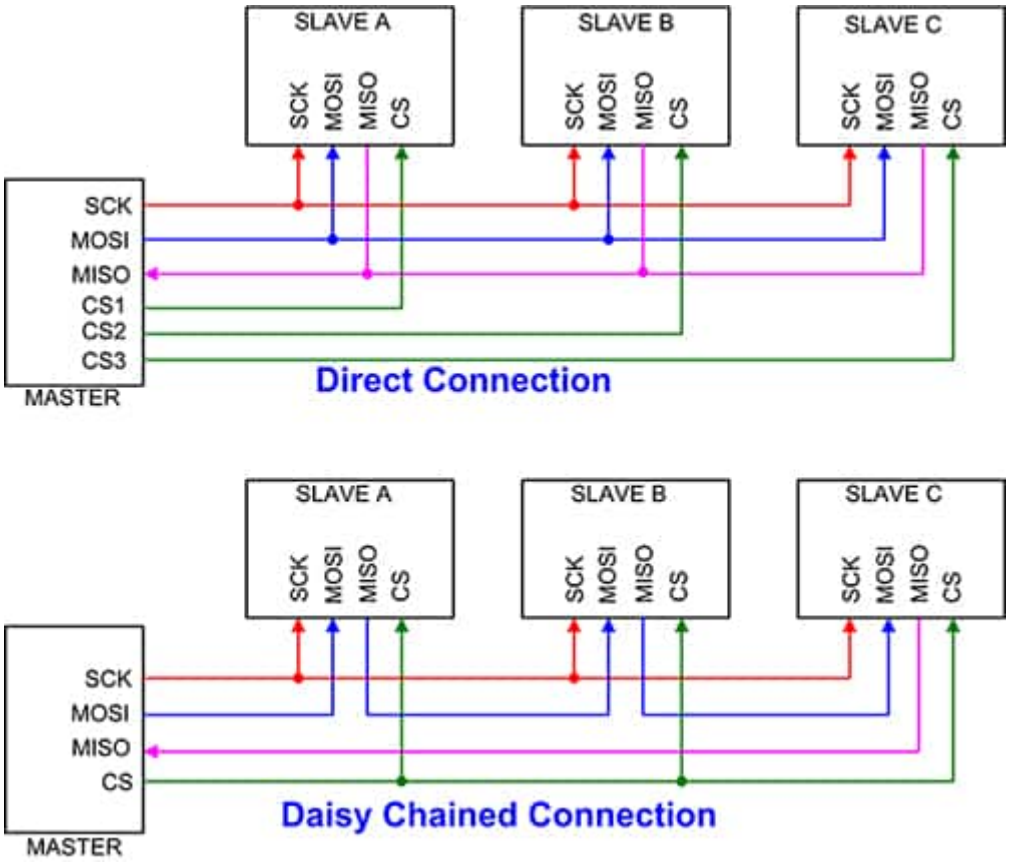


Figura 2: Dos configuraciones para tratar con interfaces de varios dispositivos secundarios. La conexión directa requiere una selección de chips para cada dispositivo secundario. La conexión encadenada usa una selección de chip individual y combina todos los datos en una sola línea. (Fuente de la imagen: Digi-Key Electronics)

La conexión directa usa una línea de selección de chips para cada dispositivo secundario. La mayoría de los microprocesadores posee tres o cuatro líneas de selección de chip. Esto limita la cantidad máxima de dispositivos secundarios al número de líneas de selección de chip. En la mayoría de los casos, esto no resulta ser un problema, pero si un diseño requiere más dispositivos en el bus, se pueden configurar algunos usando el enfoque encadenado. Con un encadenamiento, se usa una selección de chips comunes para varios dispositivos secundarios y los datos se transfieren hacia afuera en una línea de datos común. Una vez más, si se usa el modelo de los dispositivos secundarios SPI como un registro de turnos, los datos de los dispositivos secundarios se propagan en un flujo multiplexado serial.

ventajas y desventajas del SPI

Algunas de las ventajas más importantes de el bus SPI

- Tiene total flexibilidad para los bits transferidos, es decir, no se limita a una palabra de 8 bits
- Tiene una interfaz de hardware muy simple
- No se limita a ninguna velocidad de reloj máxima, lo que permite una velocidad potencialmente alta
- Es más rápido que el serial asíncrono(UART)
- Soporta múltiples esclavos.
- Soporta comunicaciones full duplex.
- Tiene una señal de bus única por dispositivo llamada SS y todas las demás señales se comparten
- No hay modos de fallo de arbitraje.
- No necesita de transceptores(adaptadores)

Por otro lado, entre las desventajas más importantes del bus SPI:

- Requiere más pines de conexíon con respecto al bus I2C.
- No admite nodos en forma dinámica (en caliente)
- Solo soporta un dispositivo maestro.
- No hay protocolo de comprobacíon de errores.
- El bus SPI requiere generalmente de líneas SS separadas para cada esclavo, lo que puede ser problemático si se necesitan numerosos esclavos.
-

test de funcionamiento

Libreria Arduino SPI

Existen diversas formas de verificar el correcto funcionamiento de la comunicación SPI, una de ellas es usando las librerías que nos provee arduino (Para microcontroladores avanzados)

https://docs.arduino.cc/tutorials/due/due-extended-spi

Proyectos OpenSource.

También tenemos proyectos opensource como pueden ser Logic Sniffer, que está escrito en java, usa un puerto USB para los datos y FPGA, lo que hace es analizar una entrada digital de estados, la vuelca en la memoria RAM y luego la vuelca en la PC.

hay otras opciones, por ejemplo usando arduino

Este es el codigo de el programa:

Codigo: https://github.com/gillham/logic_analyzer

Programa : https://www.lxtreme.nl/ols/

Repositorio Electgpl: https://sites.google.com/site/electgpl/storage

Blog: https://electgpl.blogspot.com/

Video de el funcionamiento de el programa:

https://www.youtube.com/watch?v=Q2t70aMPV9c

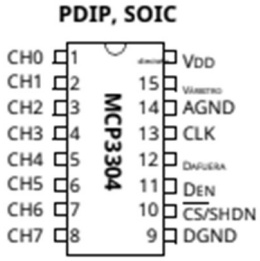
Usando un Osciloscopio

También podemos usar un osciloscopio en este caso el MSO-X 2014A para decodificar y disparar el tráfico de BUS serial SPI de dos hilos:

Video de ejemplo: https://www.youtube.com/watch?v=6AiAgZxPfp8

¿QUÉ ES EL mcp3304?

El mcp3304 es un microcontrolador de entrada diferencial de 13 bits, convertidor A/D de baja potencia con interfaz serie SPI(Serial Peripheral Interface). El convertidor MCP3304 presenta entradas diferenciales completas y bajo consumo de energía en un paquete pequeño que es ideal para sistemas alimentados por batería y aplicaciones de adquisición de datos remota. También es programable por el usuario para configurarlo en cuatro pares de entradas diferenciales u ocho entradas de un solo extremo.



MCP3304 PDIP, SOIC	Símbolo	Descripción
1	CH0	Analog Input
2	CH1	Analog Input
3	CH2	Analog Input
4	CH3	Analog Input
5	CH4	Analog Input
6	CH5	Analog Input
7	CH6	Analog Input
8	CH7	Analog Input
9	DGND	Digital Ground
10	$\overline{CS}$/SHDN	Chip Select / Shutdown Input
11	D_{IN}	Serial Data In
12	D_{OUT}	Serial Data Out
13	CLK	Serial Clock
14	AGND	Analog Ground
15	V_{REF}	Reference Voltage Input
16	V_{DD}	+4.5V to 5.5V Power Supply
—	NC	No Connection

A- Entradas analógicas (CH0-CH7)

Estos pines tienen un rango de voltaje absoluto de VSS- 0.3V a VDD+ 0,3 V. El rango de entrada diferencial de escala completa se define como el valor absoluto de (IN+) - (IN-). Esta diferencia no puede exceder el valor de Vref- Se producirá saturación de 1 LSB o código digital.

B-Tierra digital (DGND)

Conexión a tierra al circuito digital interno. Para garantizar la precisión, este pin debe estar conectado a la misma tierra que AGND. Si hay disponible un plano de tierra analógico, se recomienda conectar este dispositivo al plano de tierra analógico del circuito.

C - Select/Shutdown de chip (CS/SHDN)

El pin CS/SHDN se usa para iniciar la comunicación con el dispositivo cuando se baja. Este pin finalizará una conversión y pondrá el dispositivo en modo de espera de bajo consumo cuando se coloque en alto. El pin CS/SHDN debe colocarse alto entre conversiones y no puede vincularse bajo para conversiones múltiples.

D - Entrada de datos en serie (D_{IN})

El pin de entrada de datos en serie del puerto SPI se utiliza para sincronizar los datos de configuración del canal de entrada. Los datos se bloquean en el flanco ascendente del reloj.

E - Salida de datos en serie (D_{OUT})

El pin de salida de datos en serie SPI se utiliza para cambiar los resultados de la conversión A/D. Los datos siempre cambiarán en el flanco descendente de cada reloj a medida que se realiza la conversión.

F - Reloj serie (CLK)

El pin de reloj SPI se utiliza para iniciar una conversión, así como para registrar cada bit de la conversión a medida que se lleva a cabo.

G - Tierra analógica (AGND)

Conexión a tierra al circuito analógico interno. Para garantizar la precisión, este pin debe estar conectado a la misma tierra que DGND. Si hay disponible un plano de tierra analógico, se recomienda conectar este dispositivo al plano de tierra analógico del circuito.

H - Referencia de voltaje (V_{REF})

Este pin de entrada proporciona el voltaje de referencia para el dispositivo, que determina el rango máximo de la señal de entrada analógica y el tamaño de LSB.

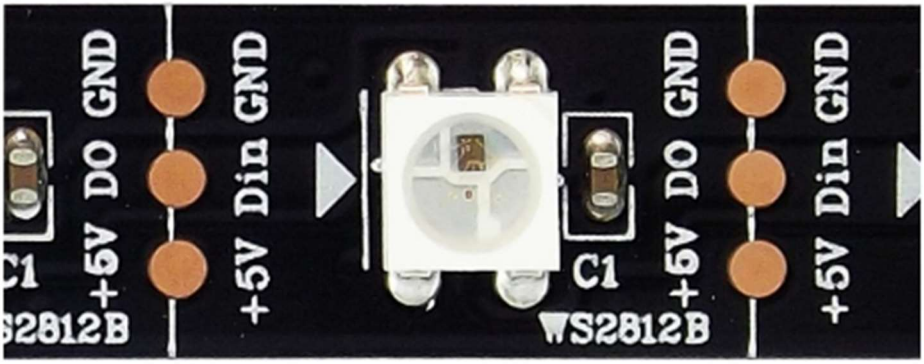
I - Fuente de alimentación (V_{DD})

El dispositivo puede funcionar de 2,7 V a 5,5 V, pero el rendimiento de conversión de datos es de 4,5 V a 5,5 V rango de suministro. Para garantizar la precisión, se debe colocar un condensador de derivación de cerámica de 0,1 µF lo más cerca posible del pin.

EL Led ws2812

El led WS2812 es un chip led encapsulado de tipo smd 5050 que internamente tiene un driver capaz de controlar 1 led RGB, además, el chip WS2812 posee un estructura interna simple para poder controlar de manera práctica un led RGB. El WS2812 dispone de 4 pines: V_{dd} (3,5 a 5.3VDC), GND, D_{in} y D_{out}. A través de los pines D_{in} y D_{OUT} es donde sucede la magia. Los bits de datos que representan el brillo RGB son alimentados en serie en el pasador D_{in}, y el chip de tiras a cabo los primeros 24 bits (8 bits cada uno de R, G, B) y envía los bits restantes a través del pin D_{out}. Mediante la conexión de los LED en una cadena, con el D_{out} de un LED debe ir a la D_{in} de el siguiente LED, cada led solo necesita 24 bits, y envía el resto de la secuencia de datos afuera al siguiente LED. En

teoría no hay límite para el número de LEDs que puede manejar con una sola línea de datos, la única limitación es que el tiempo que tarda en actualizar todos los LEDs.



Los led RGB se conectan con una fuente de alimentacion externa de +5V DC para lo cual se puede seguir un esquema como el de la siguiente imagen: conectar la entrada de + 5V de la tira led RGB al terminal + (positivo) en la fuente de alimentación (no se conecta a la de Arduino), DIN al pin digital D6 del Arduino, y – de nuestra fuente a GND de nuestra cinta led RGB, además de conectar nuestro terminal negativo a un pin GND de nuestro Arduino. Para el uso de la cinta LED, se utiliza la librería NeoPixel de Adafruit (`#include "Adafruit_NeoPixel.h"`)

Esquema de conexiones