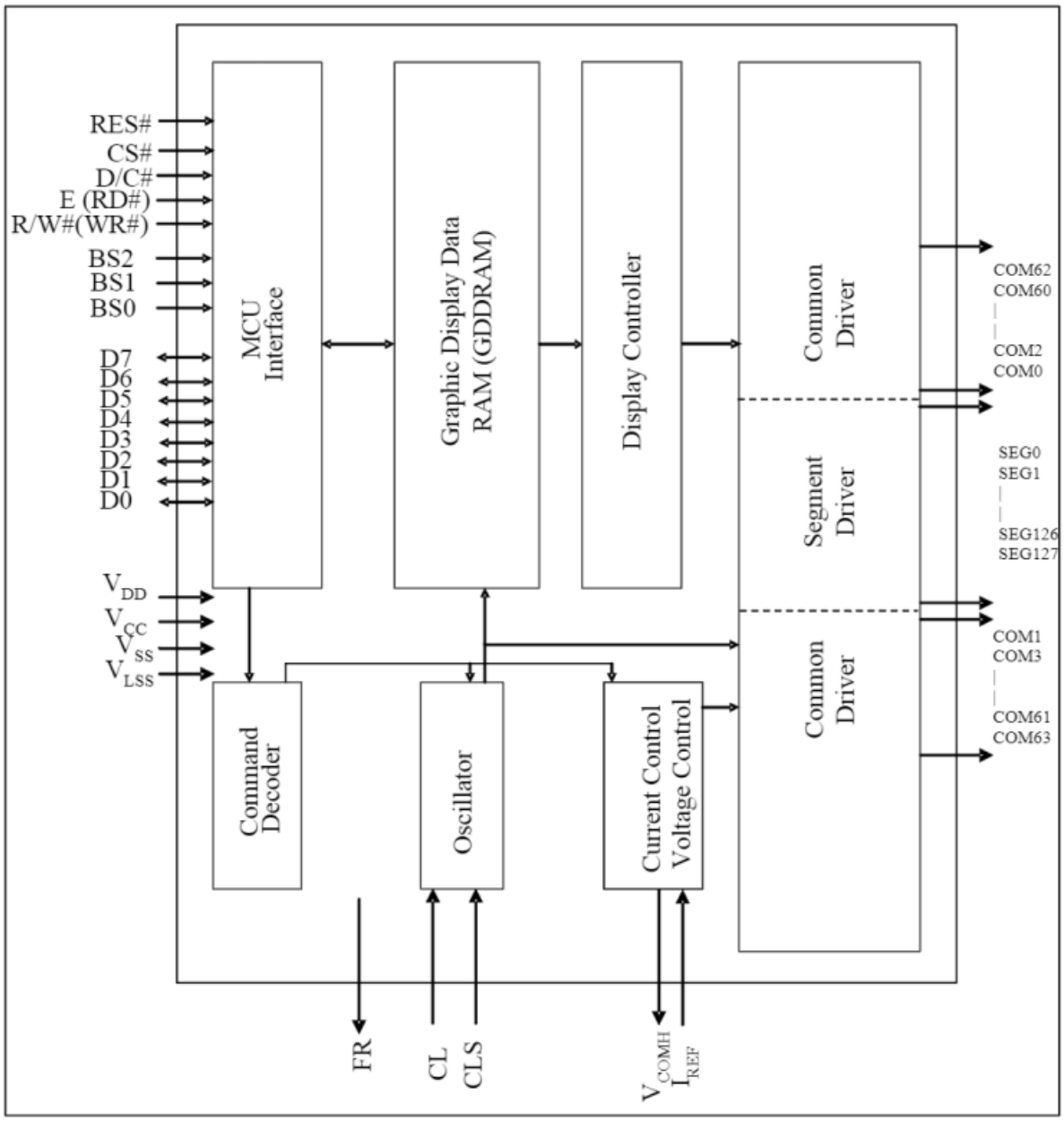
d)- ¿ Que es el controlador ssd1306(i2c). Existe alguna shield para controlar una pantalla oled 128x64?; si es así, implemente una práctica donde muestre el mensaje "Es fácil el desarrollo con shields".

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”.

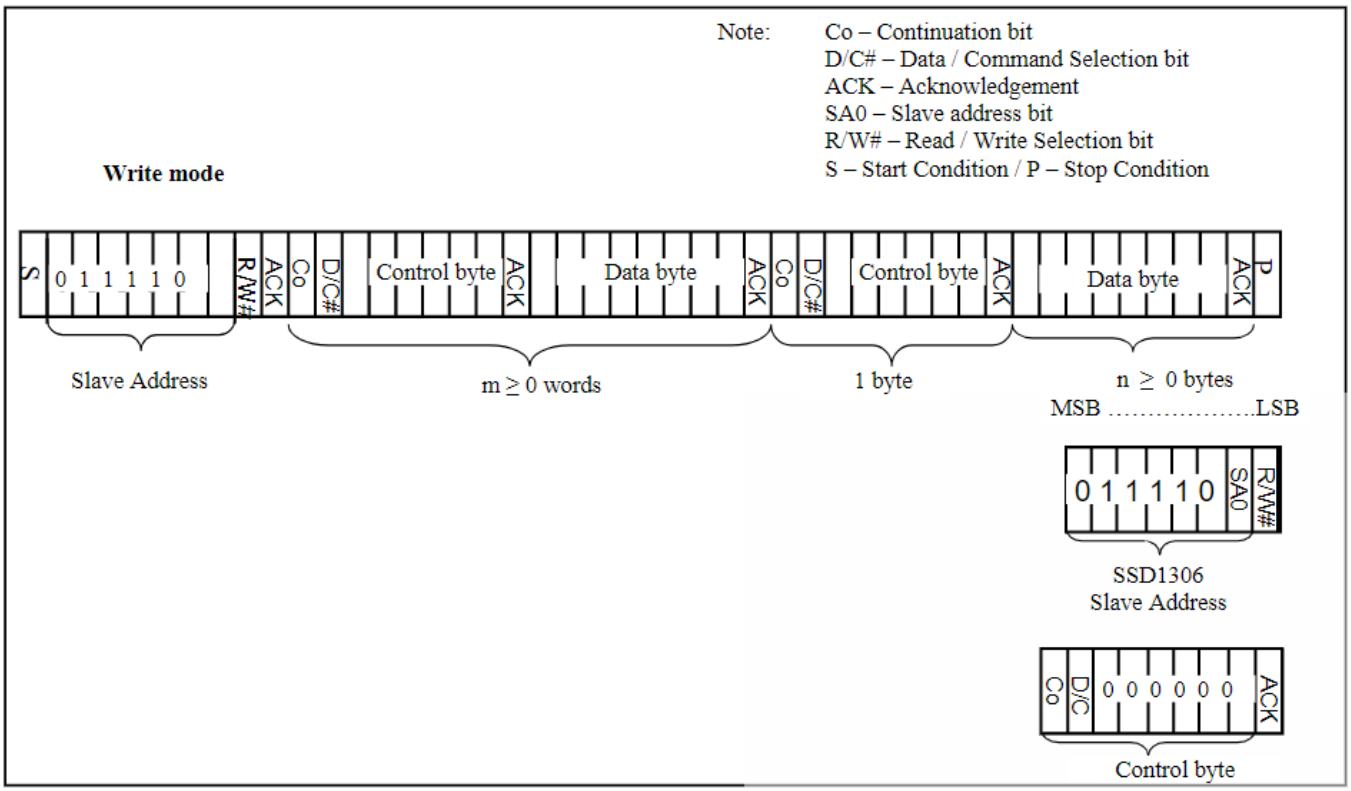
“SDAIN” y “SDAOUT” están unidos y sirven como SDA. El pin "SDAIN" debe estar conectado para actuar como SDA. El pin "SDAOUT" puede estar desconectado. Cuando se desconecta el pin "SDAOUT", la señal de reconocimiento se ignorará en el bus I2C.

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.

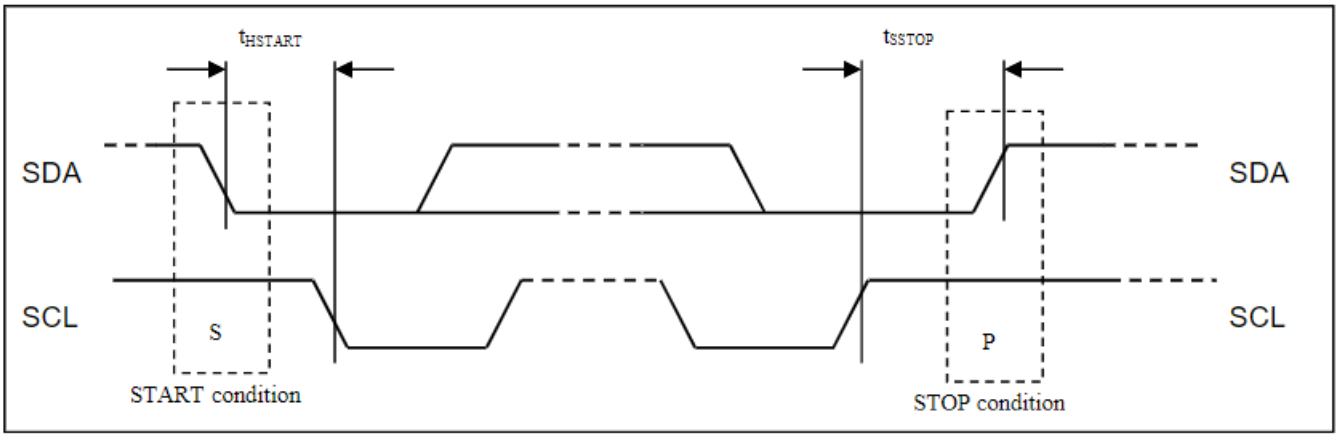


Modo de escritura para I2C

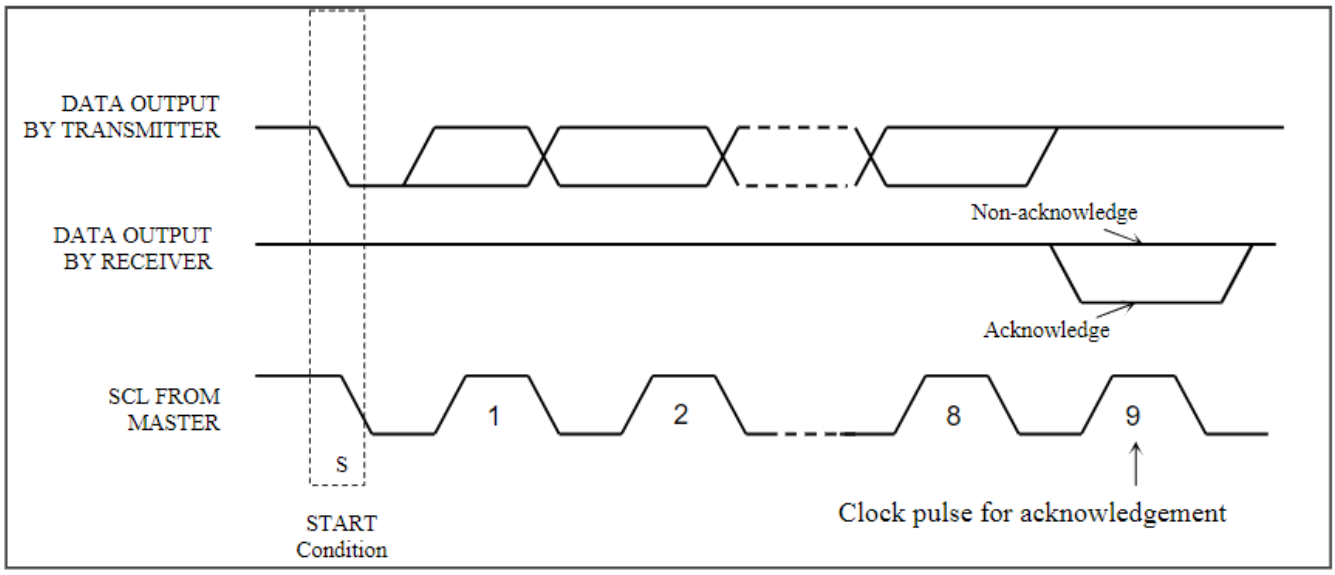
1) El dispositivo maestro inicia la comunicación de datos mediante una condición de inicio. La definición de la condición de inicio se muestra en la figura siguiente. La condición de inicio se establece tirando del SDA de ALTO a BAJO mientras el SCL permanece ALTO.
2) La dirección esclava sigue la condición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



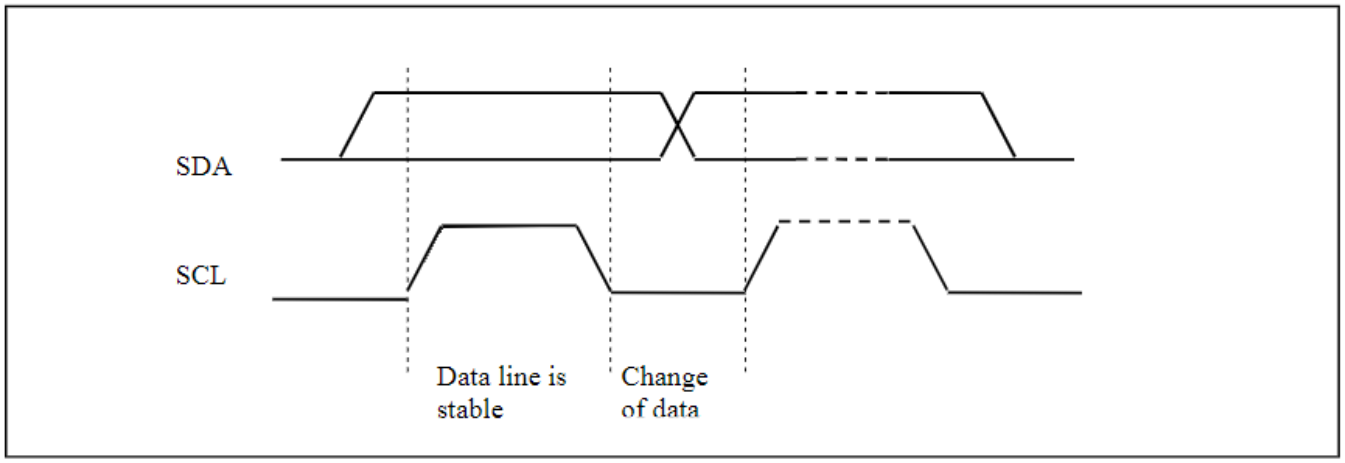
Definición de la condición de acuse de recibo

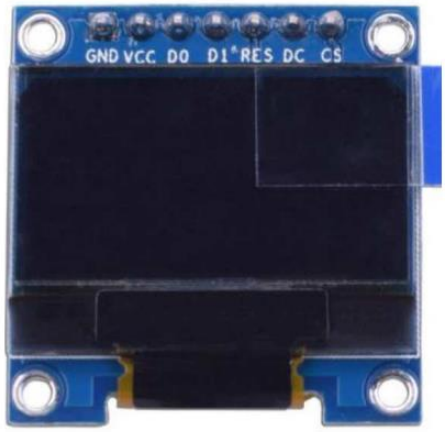


Definición de la condición de inicio y parada

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

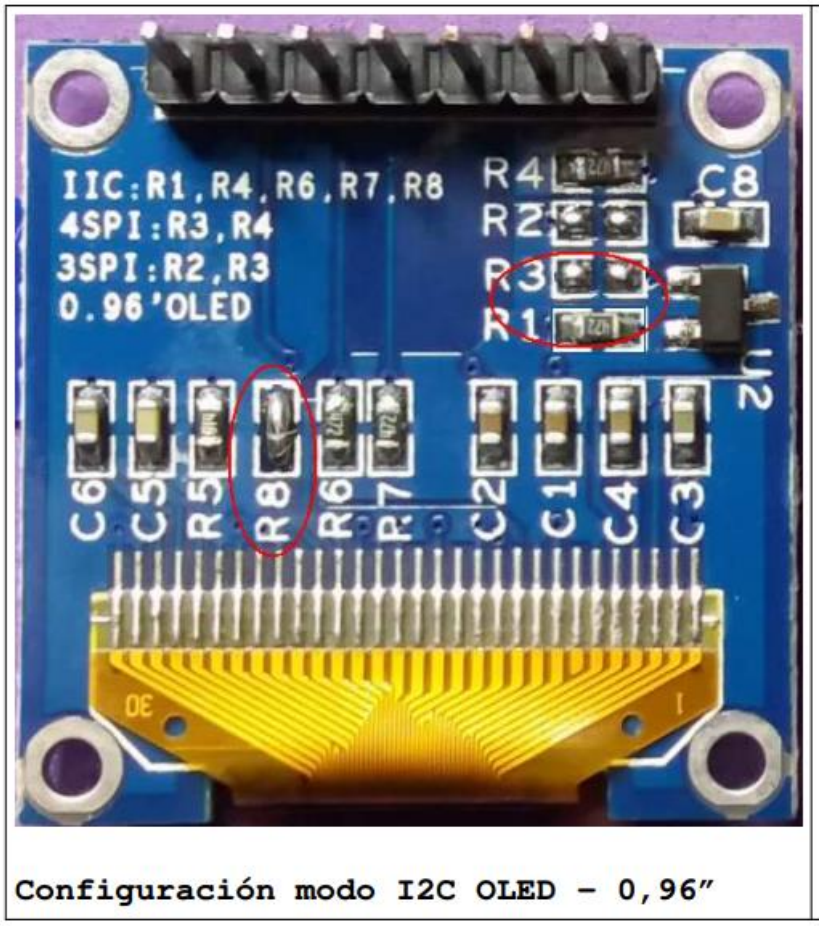




OLED 0,96 “ Chip SDD1306

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**/

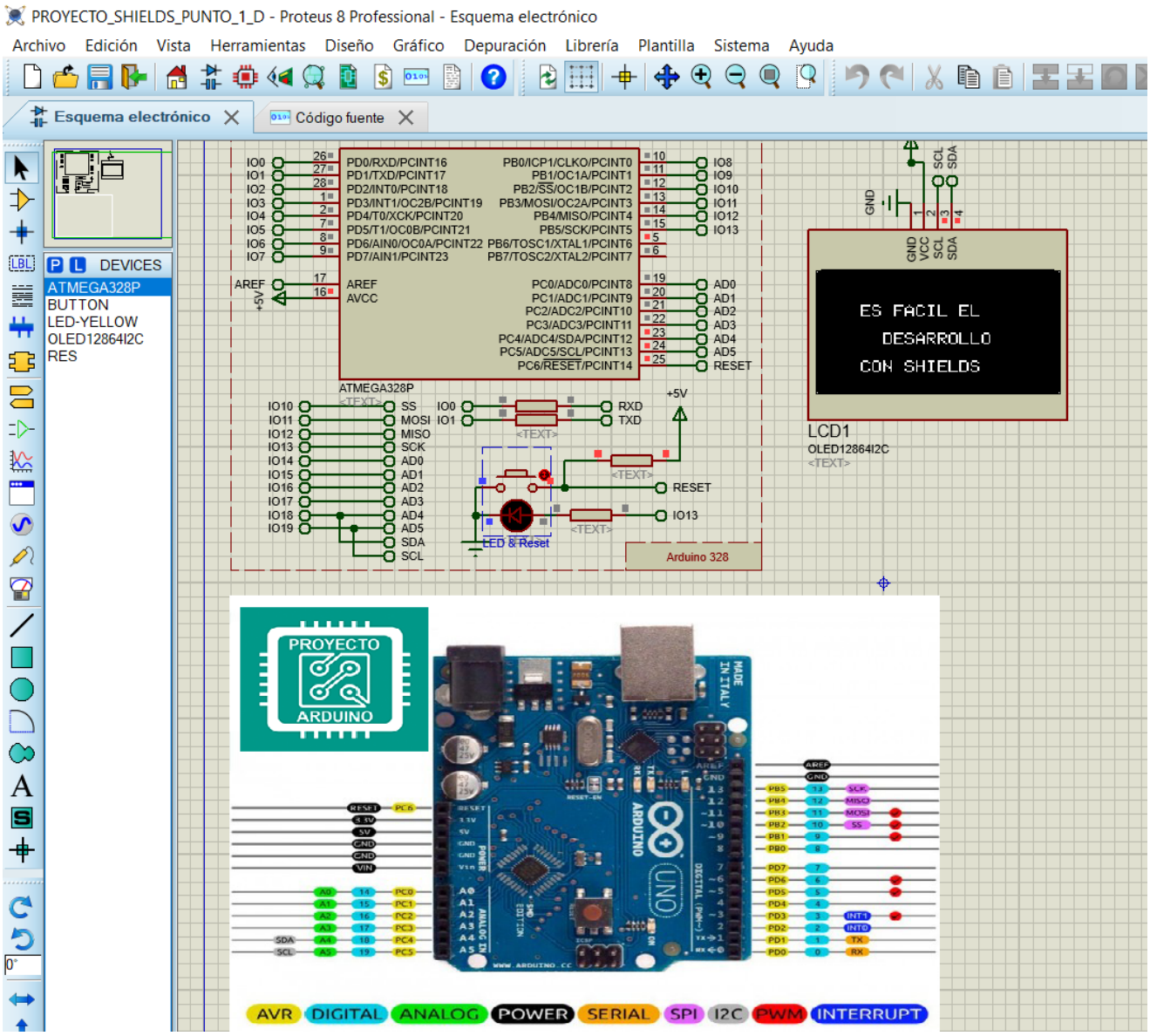


Imagen de la simulación en proteus de la pantalla OLED 128x64 con el controlador

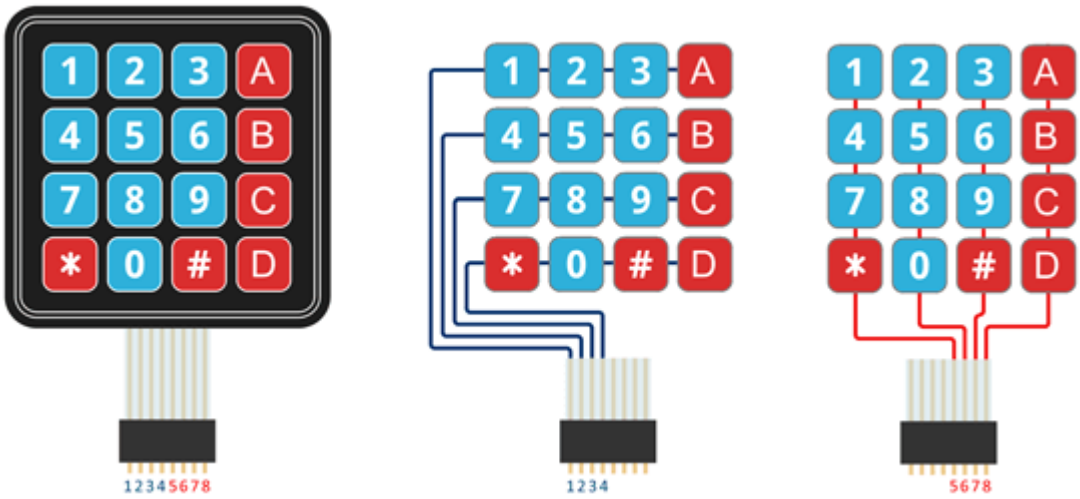
ssd1306. Fuente: elaboración propia.

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.



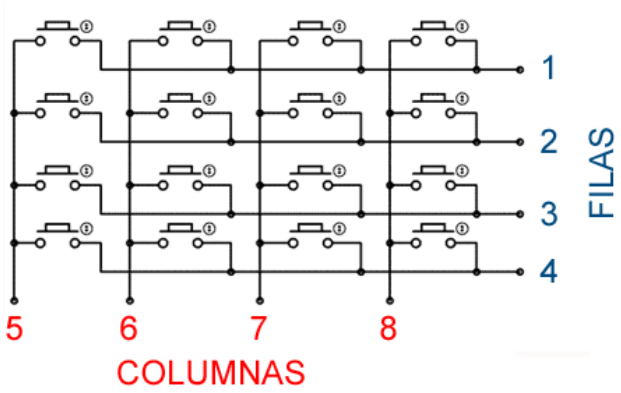
Teclado matricial de 4x4.

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

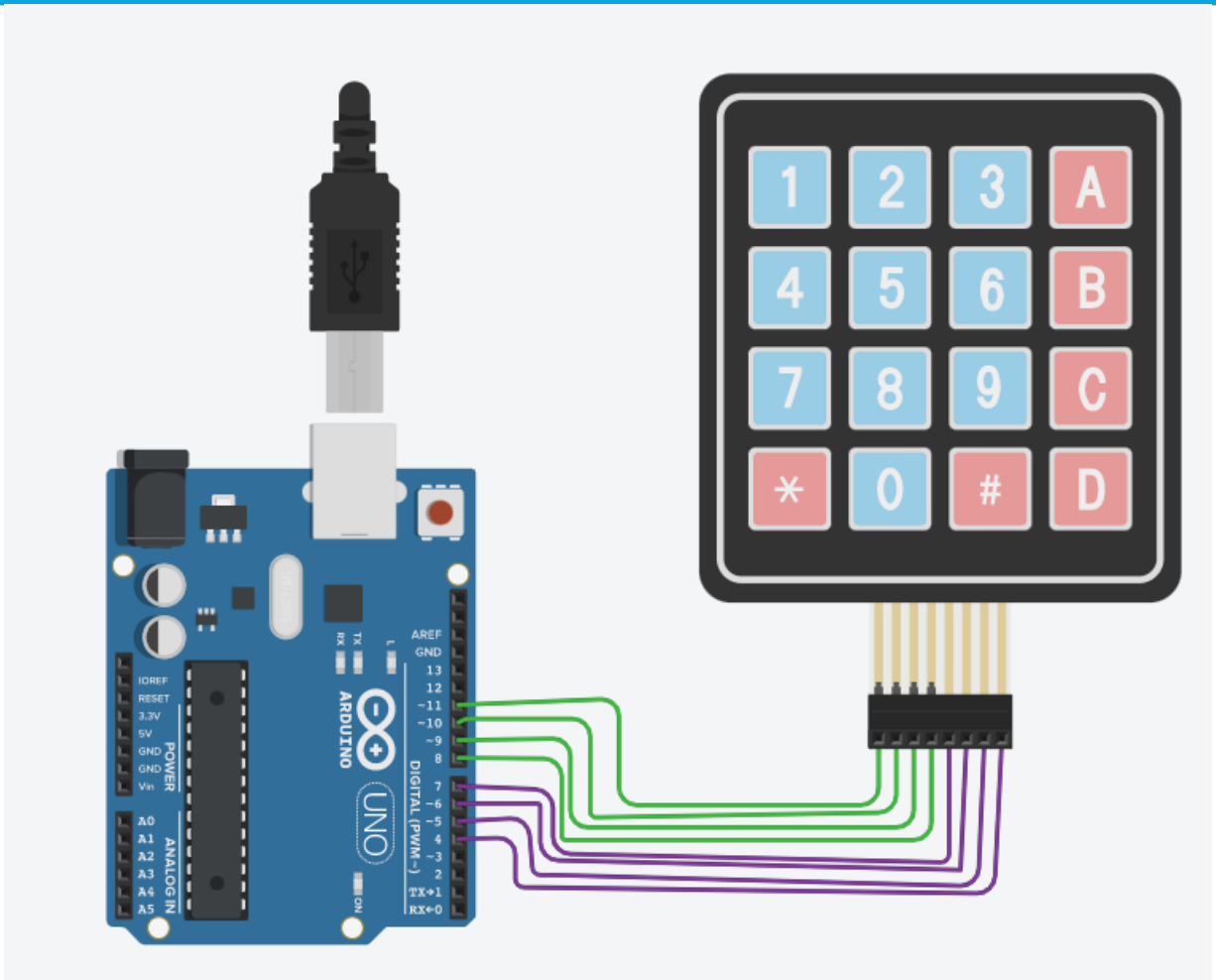
Existen varias librerías diseñaras para facilitar la lectura de teclados matriciales en Arduino. Por ejemplo, la librería Keypad, disponible en este enlace. La librería proporciona ejemplos de código, que resulta aconsejable revisar. El siguiente ejemplo es una modificación a partir de los disponibles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