ESCUELA DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

PERIFÉRICOS E INTERFACES

PRÁCTICAS DE LABORATORIO

Módulo 2 – Práctica 2

El Bus I2C

Actualizado: 23 febrero 2016

1. Competencias y objetivos de la práctica

La segunda práctica de la asignatura Periféricos e Interfaces tiene como objetivo principal reforzar y completar los conocimientos teóricos adquiridos en el módulo 2 dedicado a los buses de interconexión. Para alcanzar este propósito lo que se propone es la conexión de un módulo I2C usando un puerto de entrada/salida paralelo de propósito general. Concretamente, se utilizará como primer elemento a conectar el módulo de memoria M24C01 de 128 bytes de capacidad (1 kbit) (o también M24C02 de 256 bytes, 2 Kbit)
Para realizar la práctica se dispone de una tarjeta "*Arduino Uno*" y el circuito de memoria M24C01/M24C02 adecuadamente montado en una tarjeta experimental de laboratorio, según el esquema eléctrico que se detalla más adelante en esta memoria y la correspondiente explicación y análisis llevados a cabo en la sesión de presentación de la práctica.

La práctica se divide en **dos etapas**.

- **La primera** es la etapa básica, donde se familiarizará con el entorno de programación del Arduino en general y con la tarjeta "*Arduino Uno*". En especial es importante ejercitarse previamente en la comunicación con hardware externo, para poder leer información desde la tarjeta y mostrarla en la consola simulada en el PC y también poder enviarle información leída desde el teclado. Para esto aprovecharemos la herramienta "Serial Monitor" presente en el entorno de desarrollo. Será especialmente importante prestar atención a las conversiones que es necesario realizar de la información que se intercambia.

- **La segunda** etapa tiene como objetivo darle a las competencias adquiridas una aplicación práctica lo más cerca posible al mundo real. Así, se desarrollará y experimentará con el protocolo del bus I2C, para profundizar en su conocimiento y conseguir dominar su uso. Para esto se tendrá que implementar a través de los terminales o pines de entrada/salida del Arduino (que se especifican más adelante) el protocolo de comunicación correspondiente al bus síncrono I2C y comprobar que las operaciones de lectura y escritura sobre el dispositivo externo se realizan correctamente. Se utilizarán habilidades de programación y acceso a las señales físicas.

Es tarea del estudiante planificar y desarrollar el software de bajo nivel para programar funcionamiento de estos dispositivos y realizar las transferencias de datos sincronizando la actividad de comunicación y el almacenamiento y lectura en la memoria I2C. Con la realización de esta práctica el alumno alcanzará las siguientes competencias:

1. Capacidad para entender los diversos aspectos software y hardware involucrados en la gestión de un bus serie de entrada salida.
2. Capacidad para diseñar e implementar el software necesario para la gestión mediante un procesador de propósito general del acceso a un periférico I2C.
3. Capacidad para hacer uso de la información aportada por un fabricante de un dispositivo periférico real para utilizarlo en la resolución de un problema práctico real.
4. Capacidad para aprender y aplicar nuevos conceptos de forma autónoma e interdisciplinar.
5. Capacidad para emplear la creatividad en la resolución de los problemas.

Para alcanzar estas competencias se plantea la consecución de los siguientes objetivos:

1. Conocer y usar el bus I2C y su protocolo de comunicación básico.
2. Conocer y entender los aspectos básicos de funcionamiento de los periféricos I2C y, concretamente, la memoria M24C01/M24C02 a partir de la hoja de características proporcionada por el fabricante.

3. Realizar rutinas sencillas que permitan operaciones de escritura y lectura en cualquier dirección de memoria del dispositivo y en los distintos modos soportados.
4. Verificar el correcto funcionamiento del hardware y del software de comprobación desarrollado. Observación de las señales, si fuese necesario.
5. Planificar el desarrollo software completo de la segunda parte de la práctica. Expresarlo en un organigrama general que refleje la secuencia de acciones necesarias a realizar.

2. Documentación

La documentación básica a manejar en la realización de esta práctica está disponible en la plataforma Moodle institucional, y se concreta en:

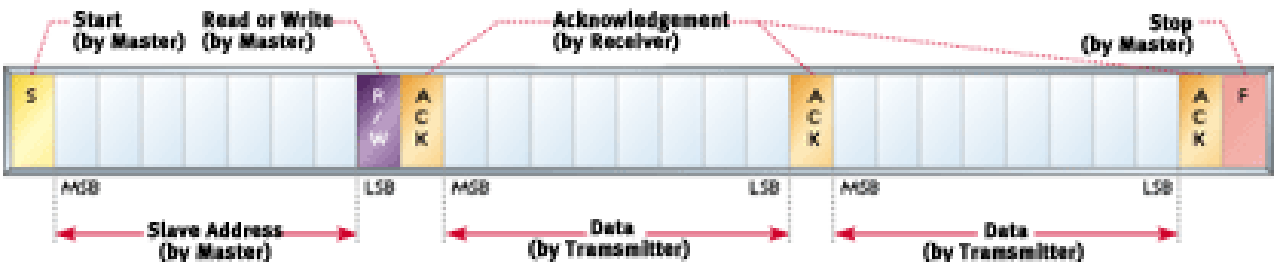
- Enunciado de la práctica (*este documento*)
- Material explicado en clase de teoría (*Bus-I2C.pdf*) y prácticas: "El bus I2C" (*Presentacion-Prac2_I2C.pdf*)
- Documentos varios relativos al funcionamiento del bus I2C
- Documentación del fabricante de la Memoria M24C01/M24C02 (*stmicroelectronics_24c01.pdf* y *ks24011.pdf*).
- Documentación (*http://arduino.cc/en/Reference/HomePage*) del entorno de programación del Arduino.
- Documentación (*BusI2C.pdf*) de la Práctica y del "*Arduino Uno*" (*http://arduino.cc/en/Main/ArduinoBoardUno*), y en especial:
    - Esquema de conexiones
    - Técnicas de acceso a los Puertos Digitales y canal serie

3. Conceptos previos

En las Clases de Teoría han sido explicados los conceptos previos sobre el bus I2C. Concretamente se han detallado los criterios de uso y características del bus, así como algunos ejemplos de uso que superan incluso a los que utilizaremos en esta práctica. En la presentación de la práctica, además se planteó un bosquejo de cómo se podía utilizar el hardware de la práctica 2 para controlar un dispositivo I2C. Esto está recogido en los documentos citados anteriormente.

Aquí se continuará, desde ese punto, detallando el esquema del circuito a utilizar en el desarrollo de la práctica 2. Previamente interesa dedicar espacio a hacer un repaso de los conceptos necesarios y relacionarlo con el detalle concreto del dispositivo que vamos a utilizar.

Es muy importante tener conocimiento claro y detallado del funcionamiento del bus I2C. Si repasamos el documento de la explicación en clase, encontramos un ejemplo típico de comunicación:



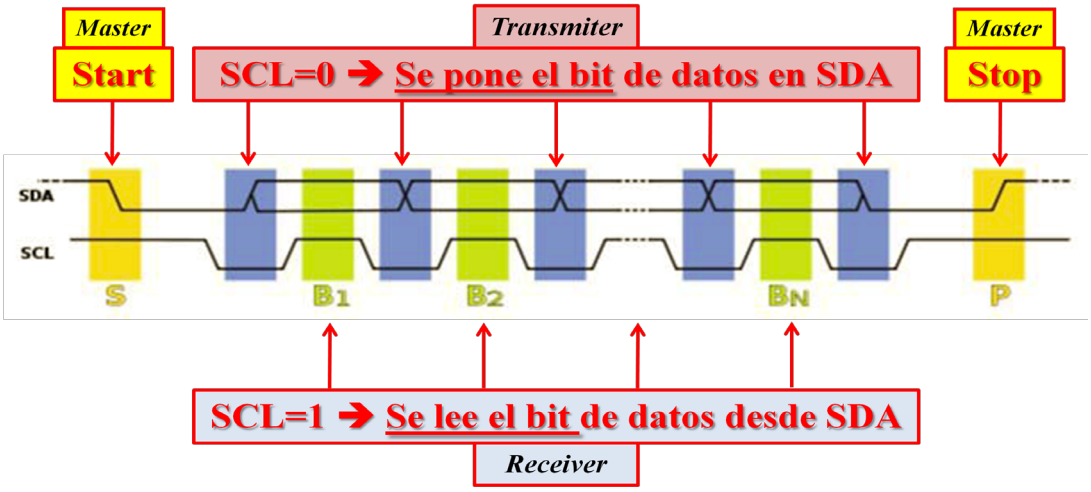
Recuérdese, que nosotros vamos a utilizar un entorno muy sencillo, donde solo existirá un "master" que será siempre nuestro Procesador Arduino. A partir de ese momento sabemos que nosotros vamos a iniciar la comunicación (generando una condición "start") y que también vamos a terminarla (generando la condición de "stop") para dejar el bus libre. No va a existir otro Maestro del bus, lo cual nos simplifica la gestión.

OPERACIONES DE START Y STOP

El Maestro adquiere el uso del bus enviando la "condición de START". Antes de hacerlo debería asegurarse de que el bus está libre: SDA="1" y SCL="1". Esto lo haría poniendo SDA,SCL="1,1" e inmediatamente leyendo SDA y SCL y comprobando que efectivamente están a 1,1 (¿SDA,SCL ="1,1" ?). Si se cumple esta condición el bus puede ser tomado por el Maestro enviando la condición de START.

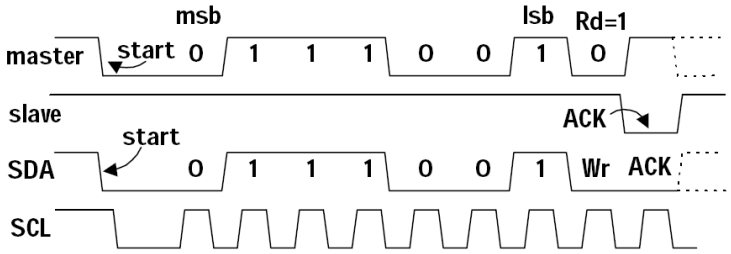
En el siguiente esquema vemos que la "condición de start" se produce cuando el "master" hace una transición de "1" a "0" de la señal SDA mientras SCL está a "1". La condición de stop es la opuesta, es decir, cuando el "master" hace una transición de "0" a "1" de la señal SDA mientras SCL está a "1". Esto es lo que se refleja en el siguiente gráfico:

Téngase en cuenta que en circunstancias normales (distintas de "start", "stop" y otras condiciones atípicas) solo se producen cambios en la señal SDA mientras SCL está a "0" y solo se lee SDA cuando SCL está a "1", como se refleja en el siguiente gráfico:

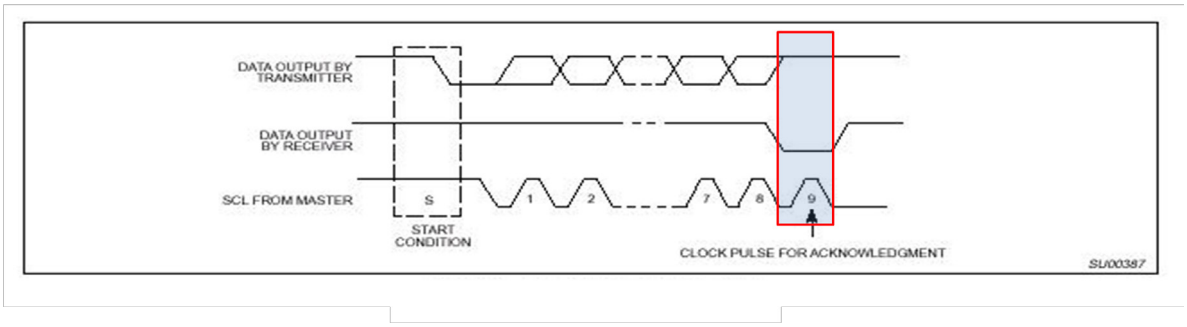


Téngase muy presente que el master actúa como "transmitter" cuando escribe y como "Receiver" cuando lee. Además, recuérdese que **las líneas físicas SCL y SDA implementan un AND-implícito** entre todos los dispositivos conectados, de forma que si queremos leer un dato en SDA, debemos poner nuestra salida SDA (que es open-collector) a "1" durante todo el tiempo que estamos leyendo hasta que completamos la lectura del último bit, tras lo que ponemos un "0" para enviar el ACK. En la siguiente secuencia hay un ejemplo de inicio donde el master (transmitter) comienza enviando la dirección y todos los Slaves (receivers) están escuchando (SDA=1), hasta que quien ha sido direccionado contesta con el ACK (y en ese

momento el master ha puesto el SDA=1 y el slave SDA=0, así, por el AND-implícito, el valor observable en la línea SDA será 0)



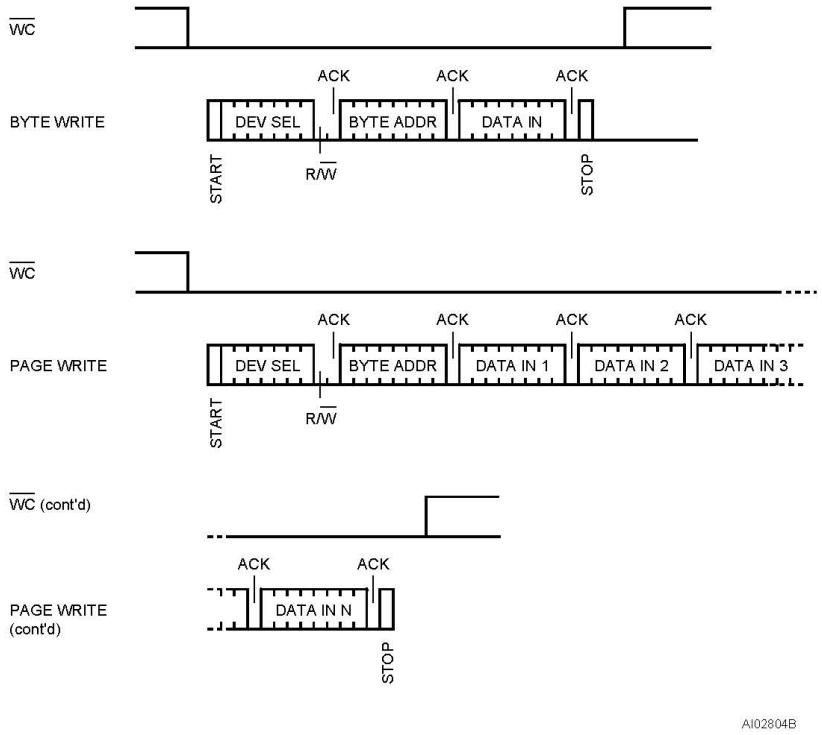
Y en general, en términos de transmisor - receptor, la gráfica sería la siguiente:



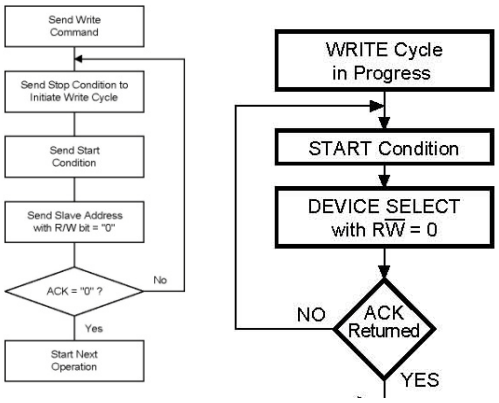
OPERACIÓN DE ESCRITURA

Vamos a ver un primer caso en que el maestro (transmitter todo el tiempo) escribe datos sobre un esclavo (receiver todo el tiempo). La secuencia de operaciones está esquematizada en la siguiente gráfica:

En el caso concreto de la memoria M24C01, los diferentes modos de operación cuando es direccionada para escritura (con wc=0) están detallados en la siguiente gráfica:



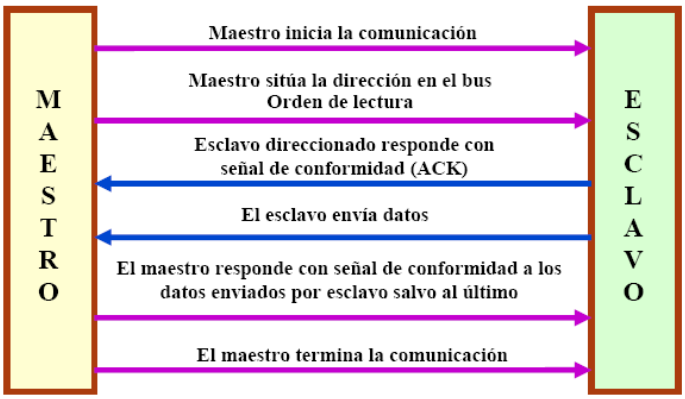
Puede darse la circunstancia de que la memoria esté ocupada en un ciclo interno de escritura (que puede durar hasta 10 milisegundos, según los datos del fabricante). En este caso, sucede que al enviar la dirección del dispositivo después de la condición de "start", la memoria no confirma la aceptación de la información enviada (no enviando "ack") por lo que debemos establecer un bucle de espera para consultar (polling) tal condición de "ack". Cuando se reciba el "ack" entonces podremos continuar con el resto del proceso para realizar la operación deseada. El siguiente esquema (optimizado y más detallado en la figura8 de *stmicroelectronics_24c01.pdf* muestra el proceso descritoAdemás, téngase en cuenta que el máximo número de bytes que podemos transferir en una sola operación de escritura (PAGE WRITE) es de 16 debido que existe un contador interno de solo 4 bits para apuntar a la dirección dentro de una página. Por tanto, si el contador del modo PAGE está a 1111 la siguiente escritura se realizará en la posición 15 de la página y el contador se autoincrementará pasando a la cuenta 0000. La escritura del siguiente byte se realizaría sobreescribiendo en la posición cero de la página.



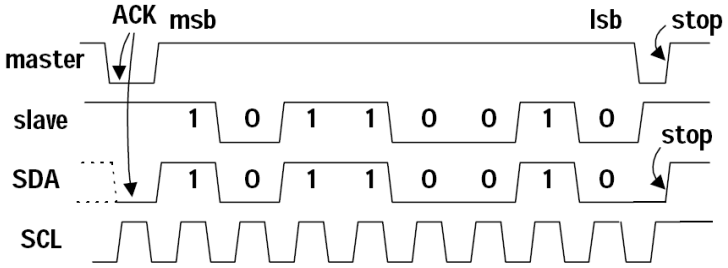
En esta figura una vez se envía "stop" y en el otro flujograma START??

OPERACIÓN DE LECTURA

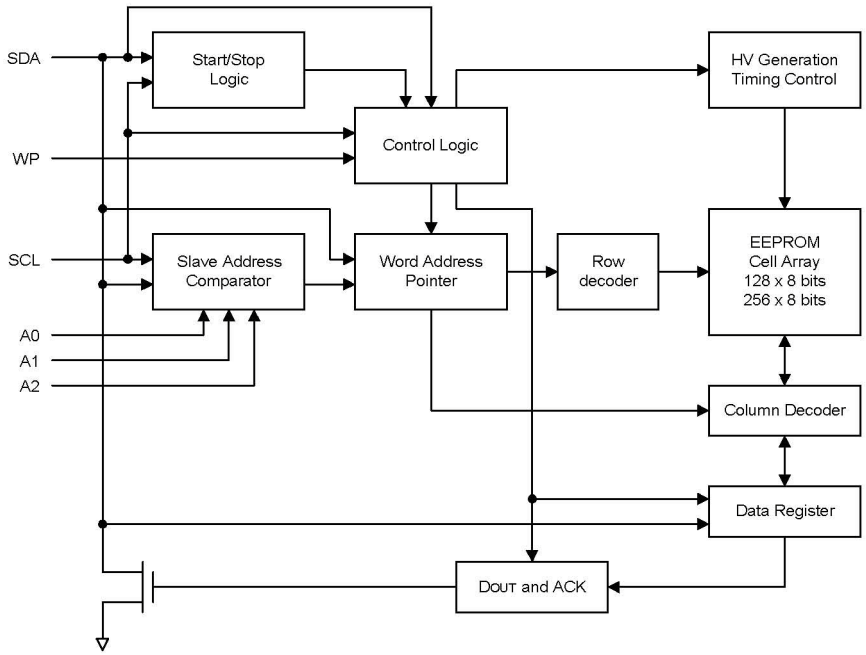
En la siguiente gráfica se recoge la secuencia de pasos mediante la cual el Maestro lee información desde el Esclavo. Aquí, como se observa, tanto el Maestro como el Esclavo usan y se alternan los papeles de "transmitter" y "receiver".



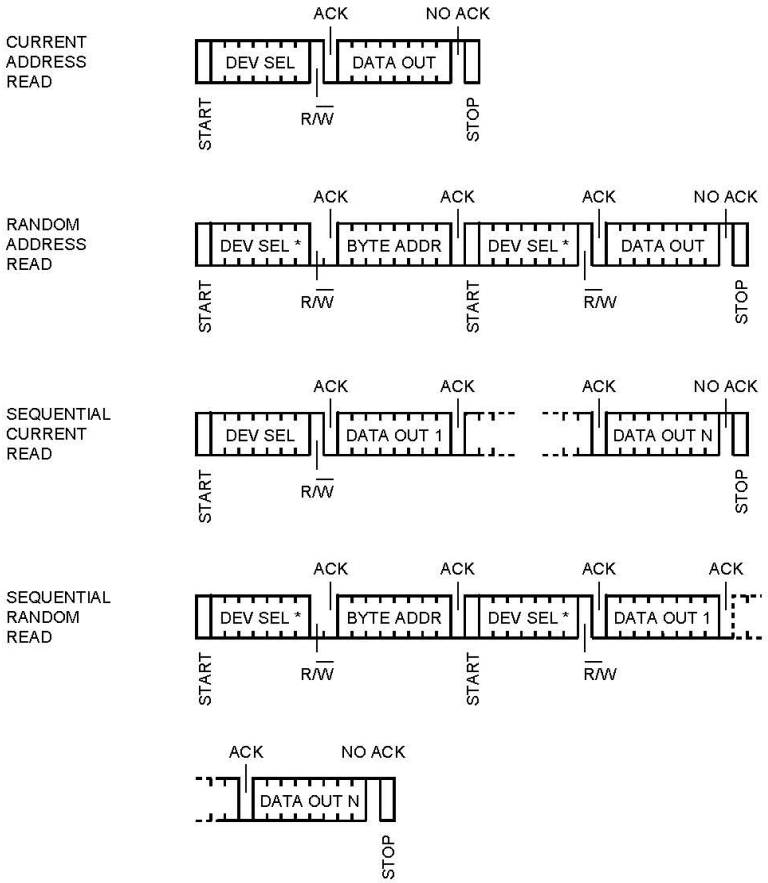
En la siguiente gráfica vemos un detalle de la señal SDA puesta por el Maestro (cuando actúa como receiver), la SDA puesta por el Esclavo (transmitter) y el SDA observable en la línea física y que es el resultante del AND-implícito.



En la siguiente figura se detalla los distintos bloques funcionales de la memoria M24C01 donde se aprecia el "Word Address Pointer" que señala la posición que va a ser accedida.



En el caso concreto de la memoria M24C01, los diferentes modos de operación cuando es direccionada para lectura están detallados en la siguiente gráfica:



Como podemos apreciar existen cuatro modos de lectura. La memoria tiene un contador interno que apunta a la posición actual a ser leída. Cada vez que se realiza la lectura el contador se auto-incrementa.

- En el primer modo de lectura, el "**current address read**" enviamos la condición de START, seleccionamos el dispositivo con R/W=1(Read) y este nos envía el dato almacenado en la posición de memoria actual (la señalada por el puntero) y además incrementa el puntero que quedará apuntando a la siguiente posición de memoria (salvo que fuese la última posición, en cuyo caso quedará apuntado a la posición cero). Para terminar, **el Maestro no envía el ACK** y seguidamente envía la condición de STOP.

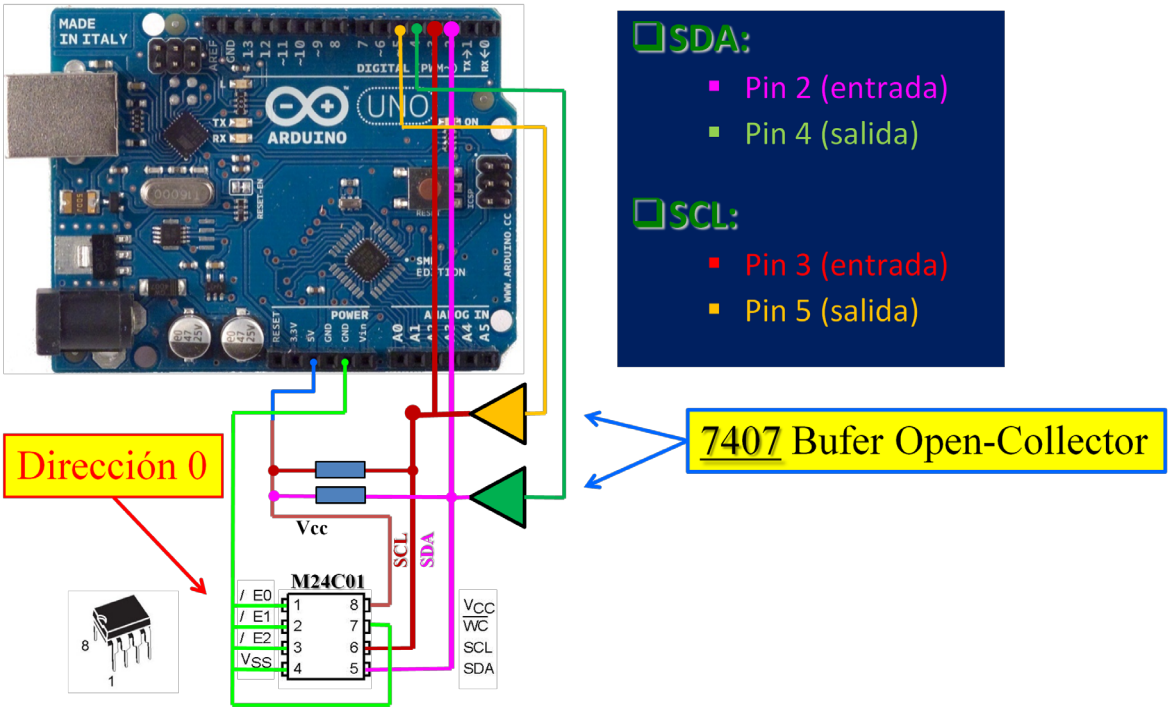
- En el segundo modo de lectura, el "**random address read**" hacemos un "dummy write" o intento de escritura para cargar la dirección que queremos leer en el contador interno. Para esto enviamos la condición de START, seleccionamos el dispositivo con R/W=0 (Write), enviamos la dirección y tras recibir el ACK enviamos la condición de START. Ahora volvemos a seleccionar el dispositivo para operación de lectura y éste nos envía el dato almacenado en la posición de memoria actual (la señalada por el puntero) y además incrementa el puntero para quedar apuntando a la siguiente posición de memoria (salvo que fuese la última posición, en cuyo caso quedará apuntado a la posición cero). Para terminar **el Maestro no envía el ACK** y seguidamente envía la condición de STOP.

- El tercer y cuarto modo son los mismos que el primero y segundo, pero en modo "**Sequential**". Esto es debido a que el dispositivo seguirá enviándonos los datos o contenidos de las sucesivas posiciones de memoria mientras el Maestro responda enviando el ACK y siga enviando ciclos de SCL. Para terminar **el Maestro no envía el ACK** tras el último dato y seguidamente envía la condición de STOP. Cuando el puntero interno alcanza el valor máximo y lo incrementamos, continúa desde cero.

4. Esquema de conexiones

Como explicamos en clase, el microcontrolador ATmega328 posee internamente la capacidad de gestionar el protocolo del I2C mediante los terminales A4 y A5. Sin embargo el objetivo principal de esta práctica es realizar el control de I2C utilizando las líneas de entrada salida de propósito general y gestionar los protocolos de la comunicación en el bus I2C mediante software escrito totalmente por el estudiante.

Para llevarlo a cabo, utilizaremos en esquema de conexiones recogido en la figura siguiente:



En este caso tenemos el bus de datos SDA (dibujado en color violeta) y la señal de reloj SCL (en color rojo) accesibles en los terminales 2 y 3, respectivamente. **Es críticamente importante que estos terminales los programemos como entrada en la inicialización (setup)**; a través de ellos podremos leer (monitorizar) en todo momento el estado de las señales en el bus. Para poder escribir en el bus utilizaremos los terminales 4 para actuar sobre el SDA y el terminal 5 para actuar sobre el SCL (programándolos como salidas en el setup).

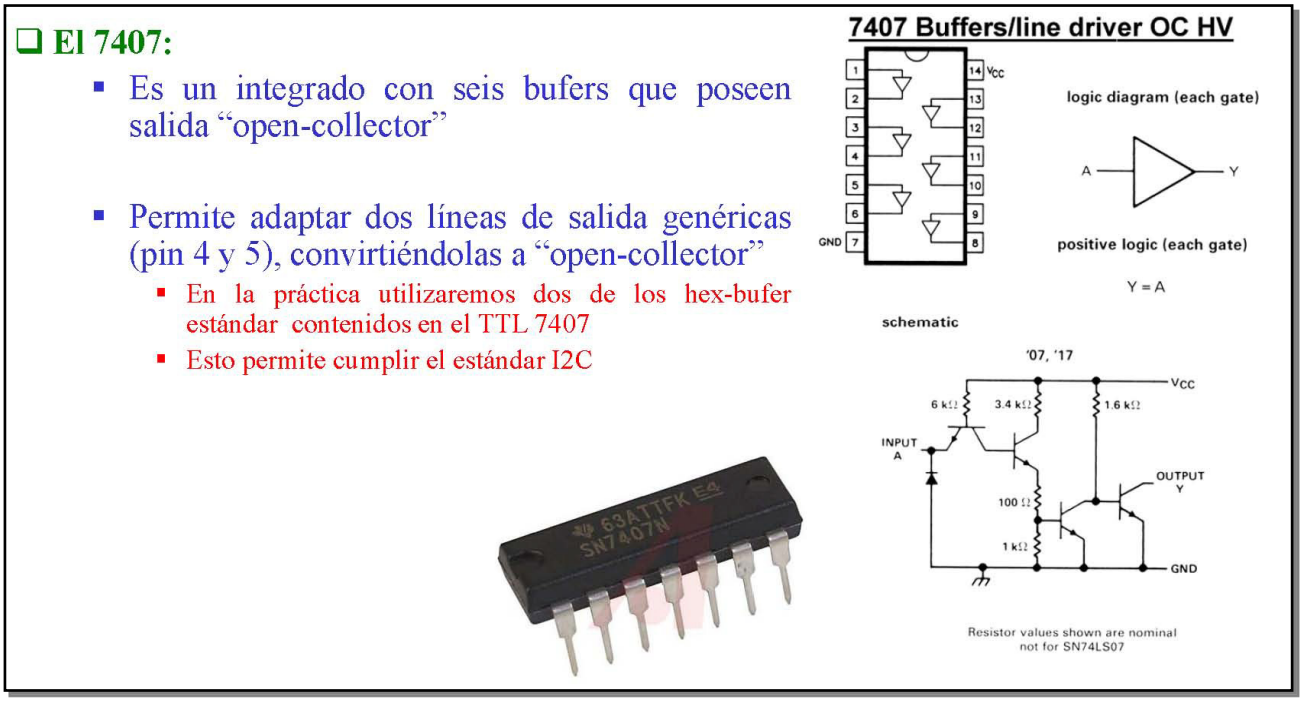
Seguidamente se muestra un ejemplo de la inicialización que se debe realizar en el "setup" en relación con las líneas de E/S para el SDA y el SCL, para llevar a cabo la práctica:

Ejemplo de inicialización del Arduino…

```
const int LEE_SDA = 2;  // puerto de entrada para leer el estado de la linea SDA
const int LEE_SCL = 3;  // puerto de entrada para leer el estado de la linea SCL
const int ESC_SDA = 4;  // puerto de salida para escribir el valor de la linea SDA
const int ESC_SCL = 5;  // puerto de salida para escribir el valor de la linea SCL

void setup() {

 // Inicializacion del canal serie para comunicarse con el usuario
 Serial.begin(9600);

 // Inicializacion de los terminales de entrada
 pinMode(LEE_SDA, INPUT);
 pinMode(LEE_SCL, INPUT);
// Inicialización de los terminales de salida
 pinMode(ESC_SDA, OUTPUT);
 pinMode(ESC_SCL, OUTPUT);
// Asegurarse de no intervenir el bus poniendo SDA y SCL a "1"....
 digitalWrite(ESC_SDA, HIGH);
 digitalWrite(ESC_SCL, HIGH);
}
```

También se puede apreciar como se ha conectado la señal **WC a "0"**, con lo que se **habilita permanentemente la escritura** en el dispositivo. Los terminales E0, E1 y E2 han sido también conectados a "0", con lo que **la dirección del dispositivo dentro del bus I2C será "1010000"** (consultar la tabla 2, página 5 del *stmicroelectronics_24c01.pdf* para aclaración).

Como se ha explicado anteriormente, estos terminales, cuando los programamos como salida, no son de tipo "open-collector" u "open-drain" como necesitamos para cumplir con las especificaciones del I2C. Por este motivo utilizamos un acoplamiento mediante un buffer estándar TTL 7407 para poder conectar adecuadamente cada una de las salidas del Arduino al bus I2C que hemos implementado. En la siguiente figura se recoge el esquema de uso del 7407.



5. Realización práctica básica

Como ya se ha comentado, la realización de la presente práctica se divide en dos partes claramente diferenciadas. En la primera parte, se pretende alcanzar un doble objetivo por un lado familiarizarse con el entorno del Arduino y por otro, implementar operaciones básicas sobre el bus I2C comprendiendo su funcionamiento y ganar experiencia en el manejo del mismo. Asimismo, se ha de garantizar el correcto funcionamiento tanto del hardware como del software empleando los medios y procesos de depuración que procedan o estime oportunos. En la segunda parte, se le dará una utilidad práctica a la experiencia ganada para lo que tendrá que implementar todas las tareas relacionadas en el apartado de "tareas de realizar" de esta memoria.

Estudio e implementación del protocolo y depuración del hardware (Memoria de 128 bytes M24C01)

En primer lugar tenemos que aprender a utilizar el bus I2C. Para facilitar la realización de la práctica, seguidamente se van a proporcionar una **serie de sugerencias** desde el punto de vista del implementador. Tienen como finalidad darles un punto de partida y unas indicaciones que pretenden facilitar la consecución de los objetivos. Evidentemente otros enfoques de resolución son posibles, y que para algunos incluso podrían ser más directos o sencillos. Siéntanse libres de elegir el procedimiento de implementación que le resulte de mayor comodidad o efectividad (siempre que cumplan con los objetivos de la práctica y que funcionen correctamente).

Operaciones elementales del Maestro:

Seguidamente establecemos una lista de las operaciones más elementales que tiene que realizar el Maestro, y por lo tanto que tenemos que implementar en nuestro software de control.

Tipos de acciones Elementales		
Actividad del Maestro	Acrónimo	Cometario
Condición de START	**START**	
Condición de STOP	**STOP**	
Envío un bit a "1"	**E-BIT-1**	
Envío un bit a "0"	**E-BIT-0**	
Envío el "ACK"	**E-ACK** (= E-BIT-0)	Es igual a Envío bit a "0"
Recibo el "ACK"	**R-ACK** (= R-BIT)	Es igual a Recibo un bit
Recibo un bit	**R-BIT**	

En realidad vemos que en este modo simplificado, solo necesitamos cinco de estas operaciones elementales, con las que podremos implementar todas las demás.

La secuencia de señales y acciones que se deben generar se han resumido en las siguientes cinco tablas:

START				
	Sec. temporal/tipo			
	1	2	3	4
Señal	Esc	Lec	Esc	Esc
SDA	1	¿1?	0	0
SCL	1	¿1?	1	0

STOP				
	Sec. temporal/tipo			
	1	2	3	4
Señal	Esc	Esc	Esc	Esc
SDA	0	0	1	1
SCL	0	1	1	1

E-BIT-1				
	Sec. temporal/tipo			
	1	2	3	4
Señal	Esc	Esc	Esc	Esc
SDA	1	1	1	1
SCL	0	1	1	0

E-BIT-0 = **E-ACK**				
	Sec. temporal/tipo			
	1	2	3	4
Señal	Esc	Esc	Esc	Esc
SDA	0	0	0	0
SCL	0	1	1	0

R-BIT = **R-ACK**				
	Sec. temporal/tipo			
	1	2	3	4
Señal	Esc	Esc	Lec	Esc
SDA	1	1	¿*?	1
SCL	0	1	¿1?	0
	* = Valor leído 0,1			

Los valores leídos (en START, R-BIT y R-ACK) serán gestionados adecuadamente por el software del usuario, por ejemplo, en el caso de START, si no se lee SDA, SCL=1,1 hay que permanecer esperando a que el bus se libere.

Utilizando lo anterior como guía, podremos hacer las distintas operaciones necesarias. Por ejemplo, la secuencia para realizar un "BYTE WRITE" sería como se especifica en el siguiente cuadro:

"BYTE WRITE"	
Secuencia	Acción
START	
E-BIT-1	Los 7 bits de dirección del dispositivo 1010 000
E-BIT-0	
E-BIT-1	
E-BIT-0	
E-BIT-0	
E-BIT-0	
E-BIT-0	
E-BIT-0	R/W = 0 (W)
R-ACK	Leer ack del dispositivo; si ok, sigo
E-BIT-0	Pongo la dirección del byte = 0000 0000 Envío los ocho ceros (esto pondrá el contador interno a 000 0000)
E-BIT-0	
E-BIT-0	
E-BIT-0	
E-BIT-0	
E-BIT-0	
E-BIT-0	
E-BIT-0	
R-ACK	Leer ack del dispositivo; si ok, sigo
E-BIT-0	Envío el dato, por ej.: 0101 0101
E-BIT-1	
E-BIT-0	
E-BIT-1	
E-BIT-0	
E-BIT-1	
E-BIT-0	
E-BIT-1	
R-ACK	Leer ack del dispositivo; si ok, sigo
STOP	

Consultar las correspondientes gráficas presentes en la documentación para implementar la secuencia similar a la anterior para las operaciones que nos interese utilizar.

6. Tareas a realizar:

Como se ha explicado, se puede seguir un procedimiento de menos a más como por ejemplo:

1. Comprobar el funcionamiento escribiendo un valor a una posición fija y leyéndolo después.
2. Comprobar la memoria completa escribiendo una secuencia de 128 bytes y luego leer los 128.
3. Utilizando la experiencia anterior, para completar y presentar la práctica, **hay que realizar un programa que muestre y proporcione la funcionalidad del siguiente menú**:

1. Guardar un dato (de 0 a 255) en cualquier dirección de memoria del dispositivo M24C01 (o M24C02 según el disponible). Tanto el dato como la dirección se han de solicitar al usuario. *(pidiéndolos por teclado/pantalla)*
2. Leer una posición (de 0 a 127) del M24C01 *(pidiendo la posición por pantalla)*
3. Inicializar toda la memoria del M24C01 a un valor *(que se solicita por pantalla)*
4. Mostrar el contenido de los 128 bytes del M24C01 *(en ocho columnas, en hexadecimal)*
5. Inicializar toda la memoria del M24C01 a un valor (Page Write) *(que se solicita por pantalla)*
6. Mostrar el contenido de los 128 bytes del M24C01 (Sequential Read) *(en ocho columnas, en hexadecimal)*

La comunicación entre el PC y el Arduino se realizará a través del "Monitor Serial" del entorno de programación de Arduino que nos permitirá, después de una programación adecuada, la comunicación con el chip de memoria I2C con objeto de implementar las funcionalidades solicitadas.

Cuando se ejecute el programa, se mostrará el menú anterior y permanecerá a la espera de que se le solicite una del las opciones. Cuando se haya completado la toma de datos y/o visualización de resultados, se volverá al principio, es decir, mostrar el menú y permanecer a la espera. En la lectura de valores, se comprobará siempre la validez de los datos proporcionados por el usuario, y se implementará el tratamiento de errores necesario.

Para lo anterior resultará útil crear procedimientos que sirvan de vínculo para programar más cómodamente. Por ejemplo, crear, al menos, un procedimiento que podríamos denominar “Escribe-en-Mem-I2C(*dir*,*valor*)” para escribir un “valor” en una determinada dirección “dir”; y otro procedimiento para “Lee-de-Mem-I2C(*dir*)”, para leer una posición “dir” de la memoria. Para implementar procedimientos que transfieran bloque mayores, pero recuérdese que el máximo número de bytes que podemos transferir en una sola operación de escritura (PAGE WRITE) es de 16.

7. Realización práctica mejorada (opcional , hasta el 30%)

Esta parte de la práctica corresponde a la realización práctica con mejoras, ampliación o modificaciones a la básica para cambiar la funcionalidad y siempre haciendo uso de las técnicas de sincronización y transferencias de datos.

8. Defensa y entrega de la práctica

Una vez realizada la práctica, el estudiante deberá elaborar una memoria que consistirá en los listados de los programas adecuadamente documentados y los diagramas de flujo que se requieran en cada apartado. Además, el estudiante deberá mostrar al profesor el funcionamiento correcto de los programas y contestar a cuantas preguntes éste formule.

La fecha límite para la realización y entrega de memoria será la fijada a través de la plataforma de tele-enseñanza institucional.