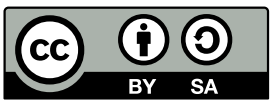
M **24-000**

I2C Protocol

Protocolo I2C para la comunicacion interna

Daniel Vilas

(Draft v0.1)



1. Introduccion

DccDiyTools es un repositorio de Módulos electrónicos, Objetos 3D y Documentación para la realización de maquetas de Tren Digitales. En general los modulos se comunicaran usando protocolos estandard para este tipo de maquetas, tales como DCC, Loconet, X-pressbus, C-Bus,... Y esto es correcto para los modulos individuales y completos que interactúen con la maqueta. Pero hay veces que nos interesa partir estos modulos en componentes modulares o simplemente debemos utilizar diferentes micro controladores para diferentes tareas y debemos poder comunicarnos con todos los modulos.

1.1. Modulo de Origen

La idea de un protocolo de comunicaciones surge de una primera version de un panel de control de la maqueta. Este panel básicamente es una caja donde en la tapa se ha pintado una versión esquemática de la maqueta y en las posiciones de los desvíos se ha puesto unas palancas o interruptores y una cadena de leds WS2812 (usados tanto para indicar el estado de los desvíos, como el estado de ocupación). El sistema se completa con dos interfaces, una usb y otra LocoNet.

La idea original era usar un STM32 pero solo se disponía de un modelo con el que no se era capaz de usar los LEDS (librerías del momento) y tampoco tenia suficientes pins para capturar los pulsos de los interruptores. Así que se opto por usar 3 microcontroladores “básicos”, dos “Arduino” (“Atmega328”) y un stm32F103. Uno de los arduinos se dejo para el bus LocoNet y la tira de leds y el otro para capturar los interruptores. Para comunicar se opto por consultarlos periódicamente por I2C, puesto que requería pocos pines, siendo el STM32F103 el master y el que coordina todo.

1.2. Modulo de Ejemplo

En el modulo anterior, el panel, en realidad la solución seria utilizar un micro controlador màs potente con más pines, mas memoria y mas facilidades para multi-tarea. Y si bien podría ser un buen ejemplo para diseñar este protocolo de comunicaciones vamos a centrarnos en una parte del Hobby que siempre ha sido la menos realista, y nos aprovecharemos de otro Hobby que (gracias a las impresoras 3D) ha dado un gran impulso.

El modulo que vamos usar de modelo es un Panel de conducción más realista. Hoy por hoy los controles de velocidad son una rueda que pone la velocidad del 0 % al 100 %. Hay diseños que la rueda es digital (una barra en una pantalla) o se parece a un regulador de tranvia. Pero en cualquier caso se mantiene, a nivel de uso, el concepto de un potenciomentro/rueda que controla el voltaje en la via.

Las maquinas de tren no funcionan asi, básicamente tienen 3 o 4 “palancas” teniendo su origen en las maquinas de vapor:

- Inversor, de tres posiciones, adelante, atrás o parada.
- Regulador, control de presión que va a los pistones (aceleración)
- Frenos, control de presión que va los frenos (frenos basicos)

La cuarta palanca son otros tipos de frenos (de vacío, del convoy,...). Además según modelos de tren puede haber palancas para los areneros, frenos dinámicos, silbatos,...

La excepción a todo esto el ProtoThrottle y nuestro modulo de ejemplo sera un panel realista de conducción. Diseño actual en desarrollo (20/07/2024):

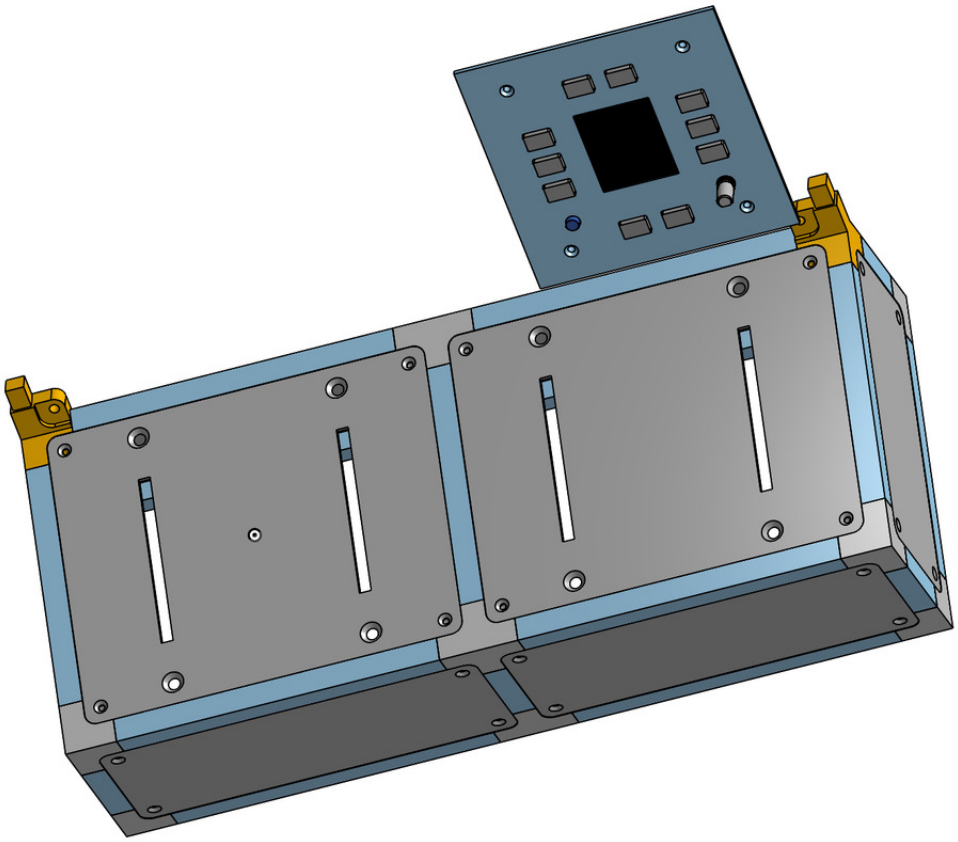


Figura 1: Ejemplo Escritorio Control

Figura 1 Ejemplo de panel en construccion Este sistema tendrá que gestionar pantallas TFT, Palancas y Botones.

Como inspiración también se tiene la creación de cabinas personales de simulación de Avaviacion. En dicho hobby se recrean cabinas muy realista, pero siempre se parte de la modularidad, esto es se divide en paneles independientes, controlados cada uno con su arduino y este gestiona unos pocos botones/led para que sea realista.

En nuestro caso la idea es hacer diferentes paneles (TFT, Botones, dos o tres palancas,...) de tal forma que cada uno se monte el panel como quiera y ajustado a sus necesidades.

Si no tenemos un protocolo de comunicaciones entre los paneles, necesitamos conectores específicos para cada tipo de panel, por lo que el diseño de la placa central se complica y limita las posibilidades. Al añadir un protocolo de comunicaciones podemos simplificar dicho diseño, pero implica un procesador por cada panel que transforme los eventos del usuario a eventos del bus.

Así mismo no tiene sentido que cada panel se muestre al bus LCB emitiendo sus eventos y es mejor que se presente al bus como un solo dispositivo y una placa central gestione cada panel.

1.3. Problemas y Soluciones

Además del problema obvio de pines (para capturar las pulsaciones de botones) el sistema inicial nos ha identificado otros dos problemas, uno de ellos es la limitación de tiempo y coste de procesado, el otro es la disponibilidad de librerías y/o periféricos.

1.3.1. Detectar Pulsaciones

El problema de capturar las pulsaciones de los botones de los usuarios es lineal, un botón ->un pin y ademas nos interesa que sean interrupciones para que se capture la pulsación y no “haya” un coste computacional. Si no son interrupciones debemos comprobar el estado del pin en cada bucle principal, pero esto implica que si el micro procesador esta haciendo cosas largas, podamos perder pulsaciones por el usuario.

Las posibles soluciones son 4:

- Uso de un procesador más poniente, con más pines e interrupciones.
- Matriz de MxN Keyboard (How Key Matrices Works) u otras variaciones basadas en resistencias/diodos. Esta solución require nos impide usar interrupciones y nos obliga a tener un bucle principal rápido y capturar el estado de los botones
- Usar un GPIO Expander o un chip de gestión de keyboard
- Usar un chip barato solo encargado de esto y usar un bus/protocolo propio

1.3.2. Coste Computacional

Ciertas operaciones requiren mucho tiempo de computación y si se realizan a la vez que el usuario interactúe con el sistema es posible que se pierdan dichos eventos.

Por ejemplo el proceso de refrescar todos los leds WS2812 le lleva más de 100ms tiempo suficiente para que un usuario pulse y libere un botón, si empezamos a añadir funciones, la ventana de oportunidad para detectar las pulsaciones se reduce.

En este caso solo tenemos dos soluciones, o bien usamos un procesador más potente, bien usamos un procesador para cada tarea intensiva.

1.3.3. Disponibilidad de Librerías/Perifericos

En el mercado existen diferentes micro-controladores que tienen periféricos diferentes como controladores CAN, de LCD, WiFi, bluetooth y según las necesidad nos puede interesar usar uno u otro. En este caso de una forma u otra deberemos acabar usando varios micro-controladores ya bien sea por usar varios con un bus controlado o con chips dedicados y sus propios protocolos.

En este caso nos podemos encontrar con la (no) existencia de librerías para el micro elegido, con lo que usar un bus propio nos permite tener chips que si nos lo soporte

2. Protocolo Base: I2C

Como base de nuestro Bus, vamos a utilizar I2C puesto que prácticamente todo los microcontroladores tienen hardware dedicado para dicho protocolo y es el que menos pines nos requieren (2). SPI se descarta por requerir mas pines (3+D). Dicho esto se ha evaluado y descartado los Buses CAN y RS-XXX por requerir como mínimo un chip más (Transceivers o PHY) y no tendiendo todos los micros soporte a dichos buses. Estos buses, aun estando pensados para realizar comunicaciones externas, es posible usarlos en cortas distancias, como una placa o entre placas de un dispositivo cerrado.

Basandonos en el Modelo OSI de redes definiremos la capa física, I2C se encargara del transporte y la comunicacion de los datos. Luego nos queda las capas superiores (sesión-aplicación) donde definiremos un protocolo propio, uno base para poder gestionar los dispositivos y varios por encima para cada tipo de panel que queramos hacer.

2.1. Capa Fisica

I2C ya nos marca unos mínimos de como deben ser las conexiones, en este apartado dejaremos simplemente la definición de los conectores y los cables a usar.

La conexión física es en estrella, partiendo del nodo central o maestro. Pero todos los dispositivos compartiran las lineas I2C (SDA y SCL)

Los cables serán AWG 22 e irán a un conector JST_PH2.0 de 5 pines:

- VCC (3.3v 100mA max)
- SDA (PullUp 1K Master)
- SCL (PullUp 1K Master)
- GND
- INT (PullUp 1K Master)

2.2. Capa enlace

En el caso de I2C los microcontroladores contienen hardware para gestionar el bus. En concreto el que hace de master I2C. Este microcontrolador toma el control de las lineas I2C, tanto de SDA como SCL, (siendo esta ultima el reloj para mandar y recivir datos). Como el Master controla el reloj, este debe saber cuando bytes va recibir. Y el slave debe devolver ese mismo numero de bytes (sino los micro se bloquean)

Una trama fisica I2C tipica tiene la forma (Negro Controla Master, Rojo controla Slave):

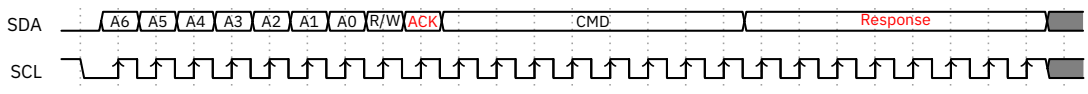


Figura 2: Trama I2C de ejemplo

El bit R/W se supone que es para indicar si una operacion es de escritura o de lectura. Como en las librerias de Arduino no se nos permite controlar este bit, no lo usaremos en el protocolo.

En codigo una peticion de un comando (CMD) que devuele una cantidad de bytes (SIZE) debe hacerse como:

```
Wire.beginTransmission(address);    // Iniciar comunicacion
Wire.write(CMD);                    // Enviar Commando
Wire.endTransmission(false);        // Preparar para response
Wire.requestFrom(address,SIZE);     // Solicitar response
while(Wire.available()>0){          // Tantas veces como SIZE
    byte slaveByte2 = Wire.read();  // Leer byte.
}
Wire.endTransmission(true);        // Liberar Bus
```

Para nuestro protocolo base nos definiremos las siguientes tramas: Request:

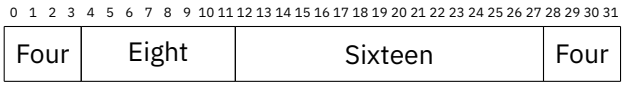


Figura 3: Trama Request

2.3. Protocolo Sesion

Este protocolo sera el base que todos los dispositivos deben implementar, tendra funciones de

- Init
- Reset
- Identificacion
- Gestion de direcciones I2C
- ...

Por una parte se definiran el flujo para iniciar los dispositivos y una serie de Registros y Operaciones I2C para el funcionamiento minimo de los dispostivos

2.4. Protocolos de Aplicacion

Por encima de I2C y de la sesion se definiran otros Registros/Operaciones I2C para cada uno de los tipos de dispositivos que existan, por ejemplo para botones, TFTs, palancas de control,....

3. Capa Sesión

3.1. Estados del dispositivo

Un dispositivo tiene 5 estados tal y como se ve en el diagrama de estados:

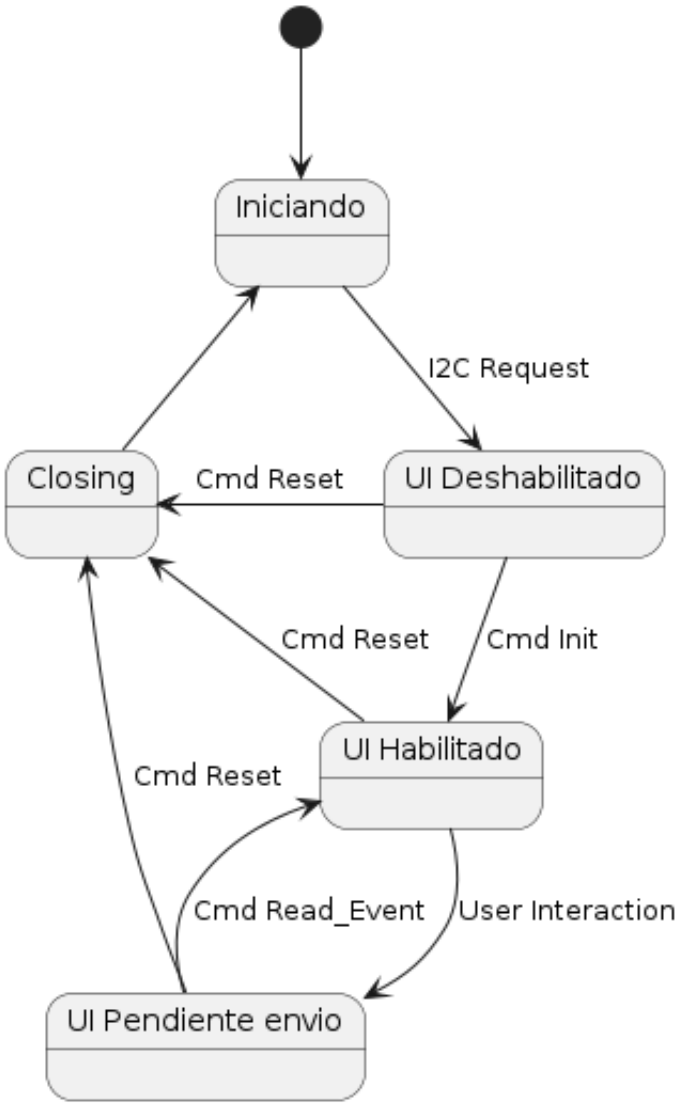


Figura 4: Estados Dispositivos

En iniciando el dispositivo se acaba de iniciar (o de resetear) y esta a la espera de la primera request I2C. En cuyo caso pasa a UI Deshabilitado, estado en el que el dispositivo respondera únicamente a peticiones I2C, ignorando “todo” lo que haga el usuario.

Ante un comando de Iniciar, se pasa el estado UI Habilitado, en este estado se espera que el usuario realice acciones (pulsar botón, mover palancas,...) y en cuanto se captura algun evento se pase a UI Pendiente de envio. En este estado la línea INT debe estar baja. Cuando el máster envíe un comando de leer se vuelve a UI Habilitado y se libera la linea INT.

Se espera que los eventos del usuario se gestionen en una Cola, por lo que UI Habilitado seria equivalente a Cola Vacía y UI Pendiente de envio Cola con Datos.

Una vez recivido el comando de reset el dispositivo se pasa a un estado de reseto donde pue-

de hacer limpieza de recursos antes de hacer un reset real. El proceso de limpieza debe terminar siempre y ejcutar una interrupcion del procesador para hacer un reset.

Los dispositivos son libres de ampliar este diagrama de estados siempre y cuando ante el master respondan de una manera coherente con este diagrama. Para ello se recomienda no ampliarlo y tener un segundo estado interno.

3.2. Proceso de inicio

El master a su vez ejecuta los siguientes pasos:

- Lista los dispositivos I2C. (@, R, Stop), siguiendo los ejemplos “I2C scan”, el resultado es una lista de dispositivos (si podemos capturar el evento, el client bajara un pulso de su INT y registramos el pin INT)
- Para cada dispositivo ejecutara un proceso de identificación mandando varios comandos (ver seccion Identificacion) para comprobar que realmente es un dispositivo nuestro. En este paso detectaremos si hay algun conflicto (varios Dispositivos con la misma dirección I2C)
- Resolucion de conflictos (...)
- Inicializacion global (@=0)

A partir de este momento el master deberá estar mirando los pines de int y consultar el dispositivo cada vez que este le baje a GND su linea de INT

3.3. Identificación

La identificación conlleva dos comandos, Obtener SN y Obtener Capacidades.

Las capacidades serán varios Bytes, siendo los primeros 6bits un identificador de tipo de panel (TFT, Botonera,....) 10 bits reservados para identificar hasta 10 protocolos básicos (BTN, palancas, TFT,...) y luego bytes extra para incluir lista de protocolos soportados

Para cada protocolo se hará una consulta de capacidad a necesidad.

Luego estará el SN, que debe ser una cadena de bytes única para cualquier dipositivo, la idea es utilizar apis de cada fabricante de microntrolador para usar el SN del dispositivo. Para que realmente sea único (por si se da la casulidad de conflicto entre diferentes fabricantes) el primer byte sera un identificador de micro (STM32, ATMEL, ESP32,...)

4. Anexo I: Comandos Protocolo Sesión

Comando	Codigo	Parametros	Respuesta	Condiciones
Query I2C	NA,R		ACK	Pulso Int si es posible
Get_ID	0x01, R		Serial Number	
Get_Caps	0x02, R		Capabilities	
Get_Caps_proto	0x02, R	ID_Proto	Capabilities_proto	
Set_addr	0x03, W	nueva direccion	OK/ERROR	Dirigida o Universal + Int_down (solo uno)
get_addr	0x03, R		direccion	Universal + Int_down Dirigida puede ser ping
reset	0x00,W		ACK	Dirigida o global
e_stop	0x04,W	boolean	ACK	global
sotd	0x05,W		ACK	global
send_int	0x06,W		ACK	
ack_int	0x06,R		Numero Eventos pendientes	
read_event	0x07,R		PrimerEvento	
Init	0x0F,W		ACK	Dirigida o Global
get_status	0x0F,R		ID_Estado	

Cuadro 1: Tabla de commandos

- Query_I2c: Es la consulta de existencia de I2C, no tiene parámetros y es lo que usan los scanners I2C de ejemplo. Si es posible hará un pulso de la linea INT
- Get_ID: obtiene el Número de Serie o identificador del dispositivo, sera un array de bytes no determinado, siendo el primero un identificador de tipo (fabricante u familia de microcontrolador) segido del serial number del silicio.
- get_caps: Obtiene el byte array de las capacidades, 6 bits indincan el tipo panel y 10 bits los protocolos basicos soportados, luego tantos bytes como protocolos extra soporte.
- get_caps_proto: Se pasa como parámetro el id de protocolo y el dispositivo devuelve los bytes específicos de cada protocolo (como por ejemplo numero botones,...)
- set_addr: Cambia la direccion I2C de un dispositivo, se necesita un reset despues. Puede ser dirigida o global, pero si es global la linea INT del dispositivo deberá estar bajo y SOLO uno en este estado.
- get_addr: Lee la direccion de un disposivo, si es dirigido, es un ping y sirve para comprobar que entiende nuestro protocolo. Si es global solo responderá el dispositivo con la linea INT abajo y SOLO habrá uno.
- reset: Reinicia el dispositivo.
- e_stop: señal de Parada, enviada desde el master. Por lo general sera la centralita quien lo lance ante un corto. Es una forma de avisar a los dispositivos el estado. Este commando tiene un parámetro binario con el estado del sistema.
- sotd: Start Of The Day. Señal para reloj Rápido, para indicar un comienzo del día.
- Send_int, El master dirigira esta petición a un dispositivo con la idea de que el mismo baje su señal INT a GND. Se mantendrá bajo hasta que el máster mande un ack_int o lea la última interrupción
- Ack_int. El dispositivo destino devolverá el numero de eventos pendientes de leer, si es 0 liberará la linea INT
- Read_event: El dispositivo destino devolverá al master el primer evento disponible, sera un array de bytes, donde el primer bytes identifica al protocolo y el resto de bytes serán dependiente del protocolo. Si la cola de enventos queda vacía liberarara la Linea INT

- Init: envía a los dispositivos la señal para avanzar de estado. Los dispositivos empezaran a aceptar los eventos de los usuarios.
- get_status: El dispositivo devolverá un byte con el identificador del estado, según la tabla de estados definida en este protocolo.

5. Indice

Índice

Índice de figuras

Índice de cuadros