Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Placa de control WiFi para dispositivo de orientación temporal

Autor
Roberto Pérez Cacho

Director
José Ignacio Artigas

Especialidad
Electrónica Industrial

Convocatoria
Diciembre 2012

Resumen

El presente proyecto trata de la implementación de la tecnología WiFi en un dispositivo de orientación temporal. Es un proyecto perteneciente al grupo de investigación de la Universidad de Zaragoza, Tecnodiscap.

El dispositivo de orientación temporal consiste en una tira de leds a los que se les debe de poder modificar tanto el color como el brillo. También incluye la funcionalidad de parpadeo. La configuración del dispositivo se realiza desde un ordenador a través de un módulo WiFi.

Las partes de las que consta son: diseño y montaje de una placa de circuito impreso (PCB) que contenga un microcontrolador y el módulo WiFi, comunicación con los leds mediante I2C, comunicación con el módulo WiFi por puerto UART, comunicación vía WiFi entre el módulo y el ordenador.

Es un trabajo más bien práctico, ya que el grueso del proyecto ha consistido en la programación y depuración del microcontrolador.

Como conocimientos adquiridos cabe destacar: tecnología I2C, comunicación por puerto serie (UART), redes WiFi, programación en lenguaje C, selección de componentes a utilizar, diseño y montaje de PCBs.

Índice

Índice de figuras

Índice de tablas

1 - Introducción

Este proyecto fin de carrera se ha realizado en el grupo de investigación Tecnodiscap de la Universidad de Zaragoza, dentro del proyecto global de realizar un dispositivo de orientación temporal que permita a los alumnos de un colegio de educación especial comprender mejor cuándo deben de realizar las actividades programadas por medio de leds de colores. El estado de estos leds (color, intensidad, parpadeo…) debe de ser modificado desde un ordenador, a través de una red wifi.

Se hace necesaria la aplicación de la tecnología para mejorar el aprendizaje de alumnos con capacidades educativas especiales.

Como punto de partida del proyecto se tenía un dispositivo de orientación temporal que constaba de 5 placas de leds, una placa controladora y la carcasa. Los objetivos de este PFC eran sustituir la placa controladora existente con tecnología ZigBee por una nueva con tecnología WiFi, adaptándola a la carcasa y las placas de leds existentes, así como programar las funciones definidas para esta nueva versión del proyecto.

Las partes básicas del proyecto son el diseño de una nueva placa de circuito impreso (PCB), adaptada a la nueva tecnología, y la programación del microconrolador, mediante el cual se controlan el módulo Wifi mediante UART, y las placas de leds mediante I2C. A través de I2C también se implementa un sensor de luz, para adecuar la intensidad luminosa de los leds a las necesidades ambientales. Aunque el control de este sensor no se ha desarrollado en esta versión del proyecto, se ha incluido en el diseño de la PCB.

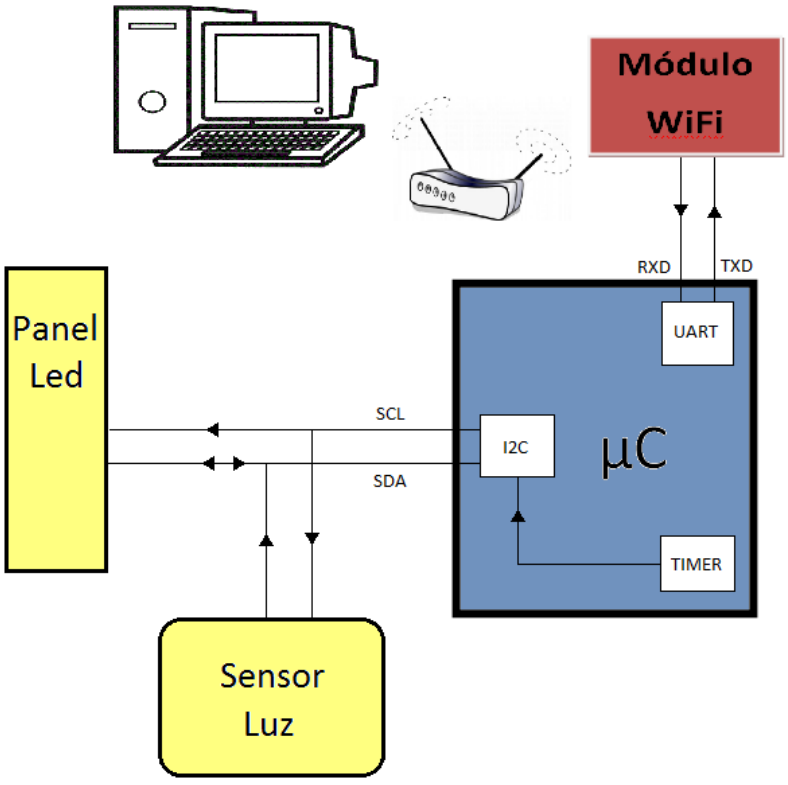


Fig. 1 – Diagrama general del proyecto

Fig. 2 – Dispositivo de orientación temporal. Vista frontal

Fig. 3 – Dispositivo de orientación temporal. Reverso

Fig. 4 – Dispositivo de orientación temporal. Detalle

2 - Diseño de la PCB

El diseño de la nueva placa de circuito impreso, está influenciado por la necesidad de adaptación a los elementos ya fabricados. El tamaño es el mismo que el de la antigua PCB para que encaje en el hueco habilitado a tal efecto en la carcasa. También los taladros están en el mismo sitio para adaptarse a los agujeros de la carcasa. Además, tanto el conector de alimentación como el que une la nueva PCB con las placas de led ya fabricadas son los mismos y están colocados en la misma posición.

En el caso de la unión entre PCB's es obvio que los conectores han de ser los mismos para que encajen. En este caso son unos conectores que utilizan la técnica del machihembrado. En lo referente al conector de alimentación, se ha decidido utilizar el mismo modelo para poder aprovechar la fuente de alimentación de 3 voltios que se utilizaba con la PCB antigua.

Se ha realizado una placa de circuito impreso de doble cara, ya que el tamaño es relativamente reducido. A diferencia de la antigua PCB, se han colocado los componentes THD en una cara y los SMD en la otra, utilizando planos de masa en ambas.

Existía la necesidad de colocar los elementos luminosos (placas de led e indicador de alimentación) y el sensor de luz de manera que cuando el dispositivo de orientación temporal esté colgado en una pared, éstos queden del lado visible. Por tanto los componentes SMD (los leds y el sensor son SMD) se han colocado en la cara de la PCB que queda hacia el exterior. Los componentes THD se han colocado en la cara que queda hacia la pared.

Los componentes principales que debía contener la PCB, además de los condensadores y resistencias necesarios, son:

- Un microcontrolador, se seleccionó el MSP430F5172 de Texas Instruments.

- El amplificador de la señal I2C P82B715 (ya que puede haber mucha distancia desde el microcontrolador hasta los controladores de leds MAX7315).

- Los conectores para albergar el módulo WiFi.

- El conector para unir la PCB con las placas de led.

- El conector de alimentación.

- Un led rojo para indicar si el dispositivo está alimentado.

- Un conector para las tareas de programación del microcontrolador por medio del MSP-FET430UIF.

También se ha introducido en el diseño una memoria Flash adicional, aunque en la fabricación no se ha montado, por si en futuras modificaciones del proyecto es necesaria una mayor capacidad de memoria.

Lo mismo ocurre con el sensor de intensidad luminosa, está en el diseño de la placa, pero no está soldado en la placa física. Ésta funcionalidad se implementará en las versiones posteriores.

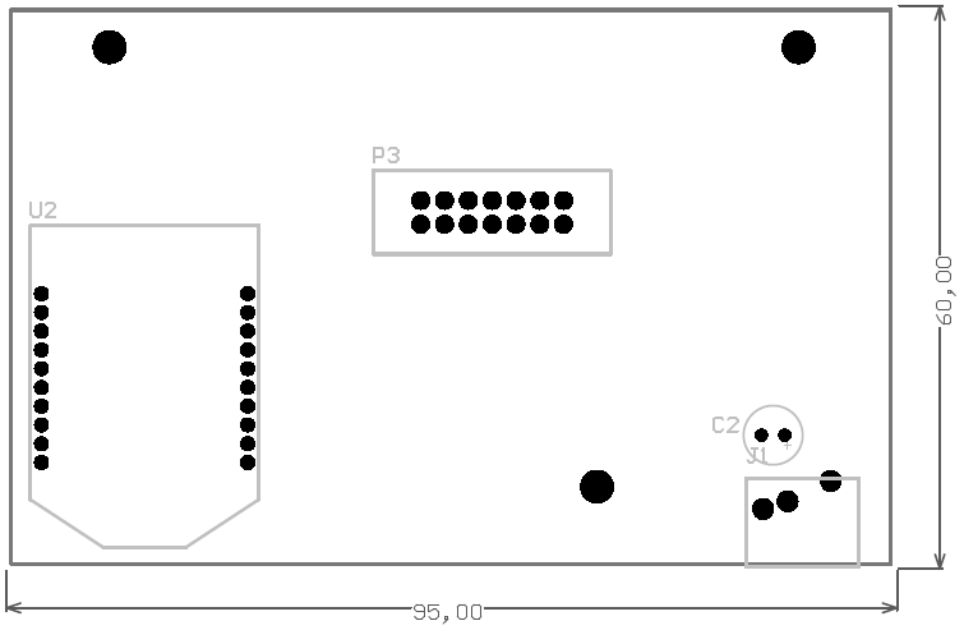


Fig. 5 – Serigrafía PCB cara top

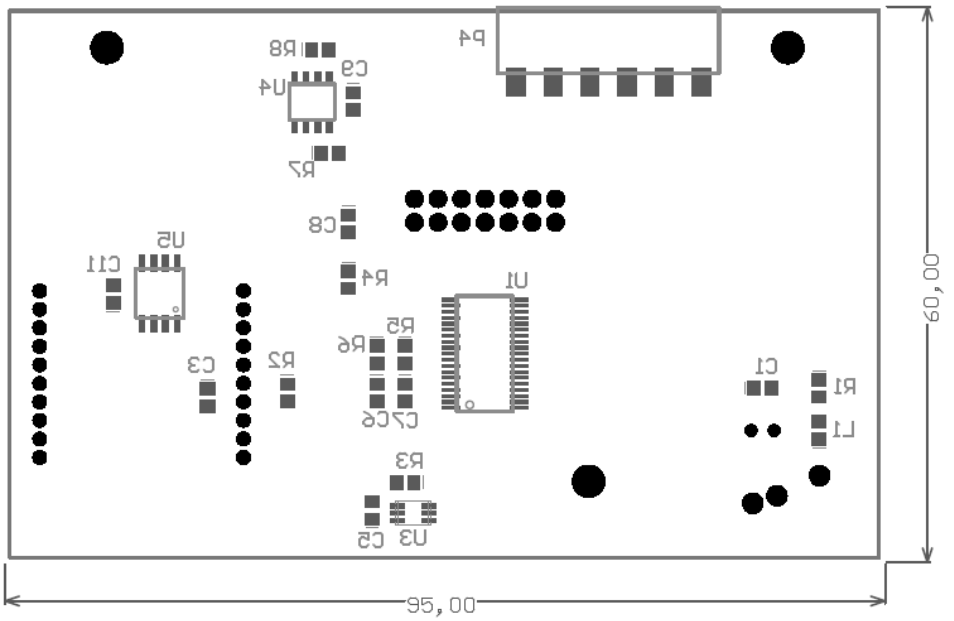


Fig. 6 – Serigrafía PCB cara bottom

3 - Microcontrolador

Los principales requisitos que debía cumplir el microcontrolador a elegir en este proyecto eran que tuviera puerto UART, I2C y timer, además de tener suficiente memoria Flash.

El microcontrolador elegido finalmente ha sido el MSP430F5172. Pertenece a la familia de ultra-bajo consumo MSP430 de Texas Instruments. Este microcontrolador tiene dos timers de alta resolución 16bits, interfaces de comunicación serie universal (USCI_A0 y USCI_B0), tres canales DMA, I/Os tolerantes con 5V, y 29 pines I/O, entre otras características.

Device	Flash (KB)	SRAM (KB)	Timer_A	Timer_D	USCI		ADC10_A (Ch)	Comp_B (Ch)	I/O	Package
					Channel A: UART, IrDA, SPI	Channel B: SPI, I²C				
MSP430F5172	32	2	3	3, 3	1	1	9 ext, 2 int	16	31	40 QFN
							8 ext, 2 int	15	29	38 TSSOP

Fig. 7 – Características MSP430F5172

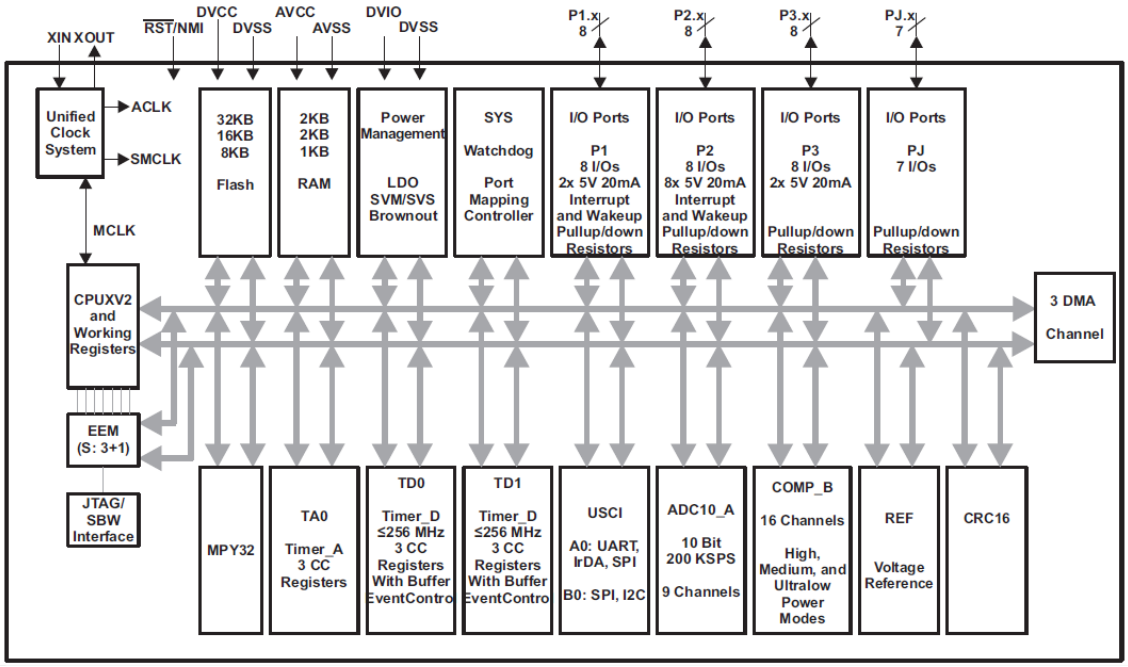


Fig. 8 – Diagrama de bloques MSP430F5172

Para programar el microcontrolador, he utilizado Code Composer Studio v5. Es un entorno de desarrollo integrado para las familias de microcontroladores de Texas Instruments. Además, contiene simulador y depurador por hardware, permitiendo el uso de puerto paralelo o serie (USB).

En este caso, he usado la depuración por puerto USB. La herramienta utilizada ha sido el MSP-FET430UIF. Es un potente emulador adaptado a la familia MSP430 de Texas Instruments, que se conecta con el microcontrolador a través de un interfaz JTAG y no necesita alimentación externa.

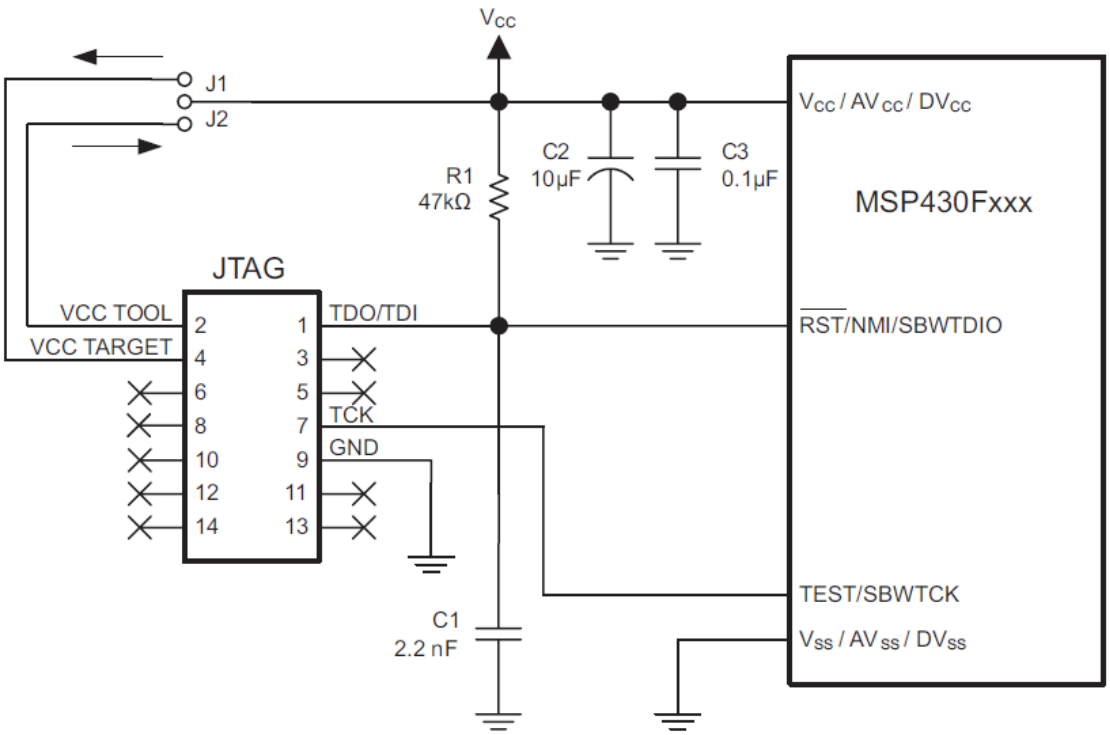


Fig. 9 – Conexión MSP-FET430UIF

En la Fig. 9 se ve el esquema de cómo se debe realizar la conexión del emulador al microcontrolador. Hay que tenerlo en cuenta a la hora de diseñar la PCB. Dependiendo de si Vcc se conecta a J1 o a J2, habrá que alimentar la PCB con una fuente externa o no.

En este proyecto, he conectado Vcc a J1, por tanto, he utilizado la alimentación externa de la PCB para depurar el código.

4 - I2C

I2C es un bus de comunicaciones en serie empleado habitualmente para la comunicación entre circuitos integrados en una PCB. Son necesarias dos líneas bidireccionales de comunicación, SDA y SCL. La línea SDA es la correspondiente a los datos, y la SCL es el reloj. Para su correcto funcionamiento, debemos poner una resistencia pull-up en cada una de las líneas.

Para la transmisión de datos es necesario un dispositivo que actue como maestro (master), y los demás como esclavos (slaves). El dispositivo master es el encargado de generar la señal de reloj e iniciar la transferencia de datos.

El bus I2C permite que el dispositivo maestro no tenga que ser siempre el mismo, esto se denomina modo multi-master. No se va a profundizar en este tema ya que se aleja del contenido del presente proyecto.

Los cuatro modos de operación posibles son:

- Maestro transmite – El maestro transmite datos al esclavo
- Maestro recibe – El maestro recibe datos desde el esclavo
- Esclavo transmite – El esclavo transmite datos desde el maestro
- Esclavo recibe – El esclavo recibe datos desde el maestro

Cada dispositivo tiene una dirección, que se usa para seleccionar con cual se desea establecer la comunicación de entre los que están conectados a las líneas SDA y SCL. Esta dirección ocupa 7 bits. El siguiente bit es el de lectura/escritura. Este bit es “0” si queremos escribir, o “1” si queremos leer.

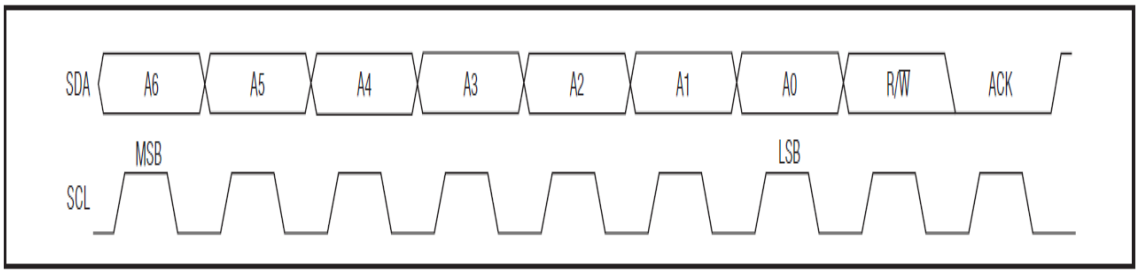


Fig. 10 – Slave Address

Para iniciar la comunicación, el dispositivo master debe enviar una señal de start. Seguidamente hay que enviar la dirección del esclavo y el bit de lectura/escritura. Si el dispositivo esclavo tiene diferentes registros internos, hay que enviar el registro sobre el que queremos actuar. Lo siguiente es enviar los datos de 8 bits. Cuando se acaba es necesario que el master envie una señal de stop.

La señal de start corresponde a un cambio de estado alto a estado bajo de la línea de datos mientras la línea SCL está en estado alto. La señal de stop, por su parte, se realiza cuando se sube la línea SDA desde el estado bajo mientras la línea de reloj está en alto.

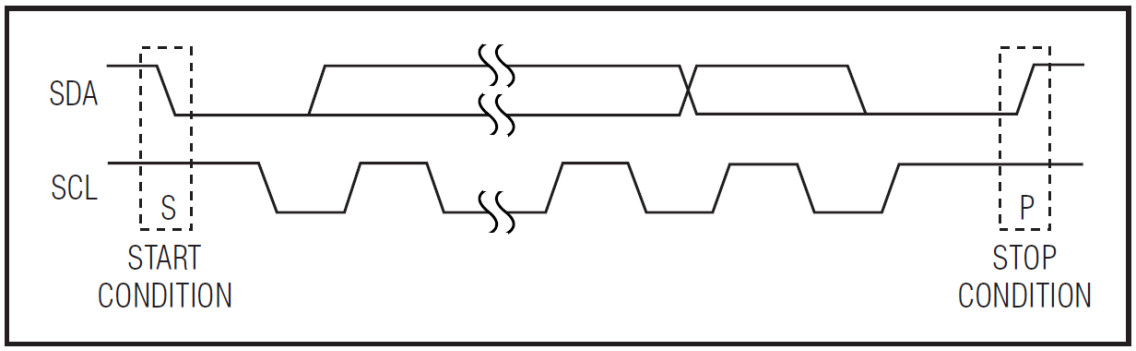


Fig. 11 – Start and Stop conditions

Los cambios de bits se realizan mientras la línea SCL está en estado bajo. La aceptación del dato se produce durante el estado alto del reloj. Por tanto, la línea SDA debe mantener el bit mientras SCL permanece en estado alto.

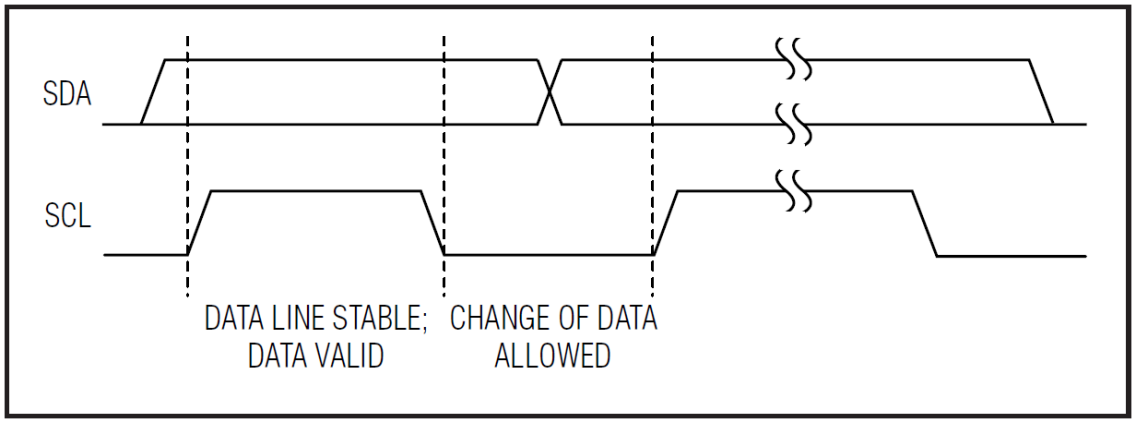


Fig. 12 – Bit transfer

El dispositivo receptor, por su parte, debe de enviar un bit de reconocimiento (ACK) después de cada byte recibido. Si el receptor es el esclavo, y no envía el ACK al master, éste enviará la señal de stop y dará por concluida la comunicación. Si por el contrario el receptor es el maestro, es éste el que decide enviar un ACK o terminar la comunicación con un bit de stop.

Como conclusión, en la Fig. 13 se muestra el diagrama de flujo de un dispositivo master en comunicación I2C.

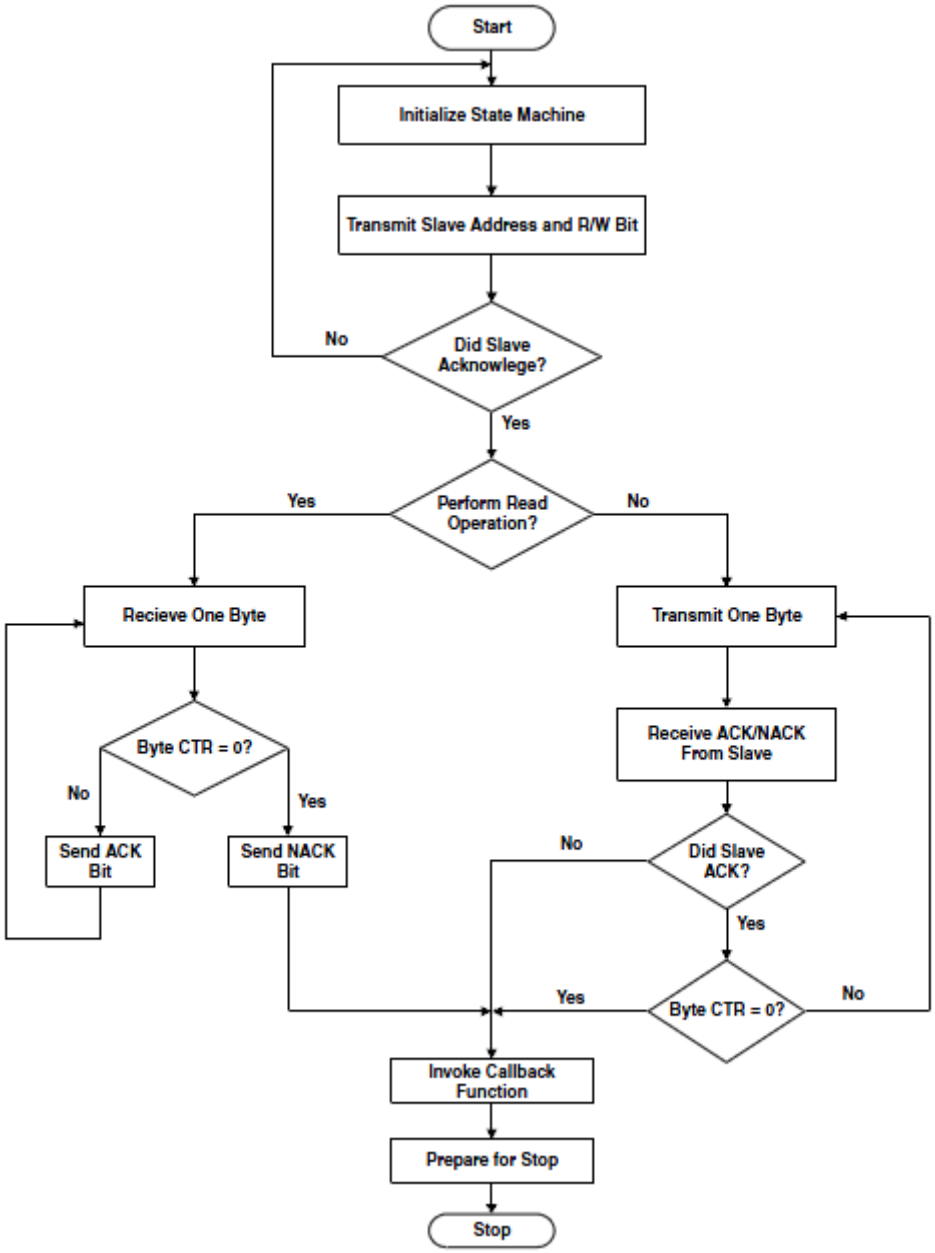


Fig. 13 – Diagrama de flujo I2C dispositivo master

4.1 - I2C en el proyecto

En el caso específico del dispositivo de orientación temporal, el dispositivo maestro, como es obvio, es el microcontrolador (MSP430F5172), y los dispositivos esclavos son los controladores de los leds (MAX7315), y en las futuras modificaciones del proyecto también lo será el sensor de intensidad luminosa.

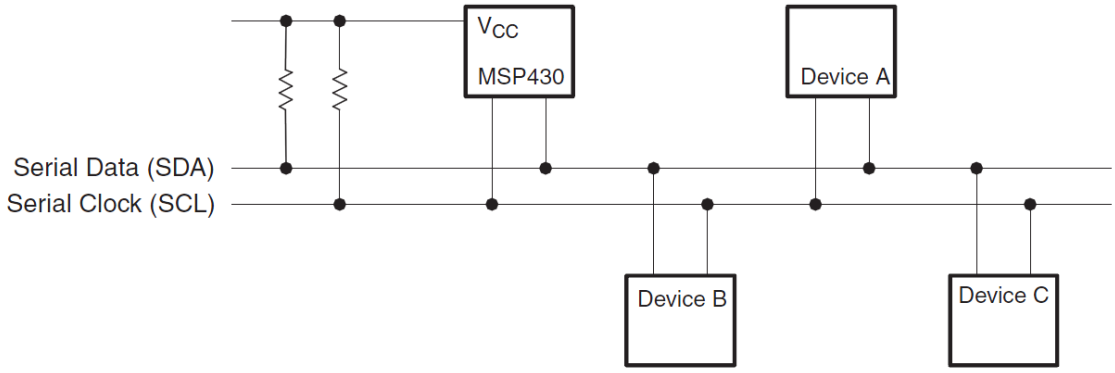


Fig. 14 – Diagrama de conexión del bus I2C

En el caso de los dispositivos MAX7315, se utilizará el modo escritura sobre el esclavo, ya que lo que se desea es modificar el estado de los leds, mientras que en el sensor, lo que se desea es leer la información proporcionada.

4.2 - I2C en MSP430

Los principales registros para la configuración, transmisión y recepción de datos son los siguientes:

- P1SEL: Función I2C en los pines P1.4 y 1.5.

- UCB0CTL0:

    - UCMST: Selección del modo master/slave.

    - UCMODE: Modo de trabajo del USCI_B0.

    - UCSYNC: Modo síncrono.

- UCB0CTL1:

    - UCSSEL: Selecciona la fuente de reloj.

    - UCTR: Transmitir/recibir.

    - UCTXNACK: Envía un NACK.

    - UCTXSTP: Envía un STOP.
    - UCTXSTT: Envía un START.
    - UCSWRST: Mantiene congelado al USCI.

- UCB0BR0: Low Byte del Prescaler Baud Rate.
- UCB0BR1: High Byte del Prescaler Baud Rate.
- UCB0I2COA: Registro que almacena la dirección I2C propia.
- UCB0I2CSA: Registro que almacena la dirección del esclavo.
- UCB0STAT:
    - UCBBUSY: Bus inactivo/ocupado.
    - UCNACKIFG: Se ha recibido un NACK.
    - UCSTPIFG: Se ha recibido un STOP.
    - UCSTTIFG: Se ha recibido un START.
- IFG2:
    - UCB0TXIFG: Transmisión. Se activa cuando el buffer UCB0TXBUF está vacío.
    - UCB0RXIFG: Recepción. Se activa cuando el buffer UCB0RXBUF está cargado.
- UCB0TXBUF: Aquí se guarda el dato que va a ser enviado.
- UCB0RXBUF: Contiene el último dato recibido.

5 – Controlador de LEDs MAX7315

Para fijar el color y el brillo de los leds, se utilizan controladores MAX7315. Estos dispositivos permiten, de una manera simple, cambiar la intensidad de sus salidas, estando cada una de ellas conectada a un led.

Estos controladores se controlan por I2C a una velocidad máxima de 400kbps. El rango de tensión de operación está entre 2 y 3.6V, teniendo protección contra sobretensiones de 5.5V. Consta de 8 pines de entrada/salida, proporcionando cada uno un máximo de 20mA de corriente cuando funcionan todos al mismo tiempo.

En las figuras 15 y 16 se muestran fotografías de las dos caras de una placa de leds, indicando dónde están posicionados los controladores MAX7315, los microinterruptores encargados de seleccionar la dirección slave y los leds.

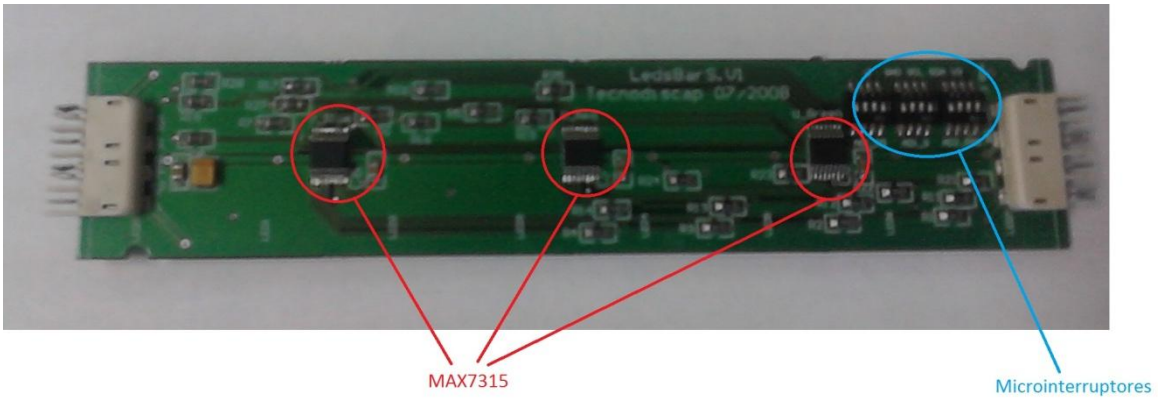


Fig. 15 – Placa de leds: reverso

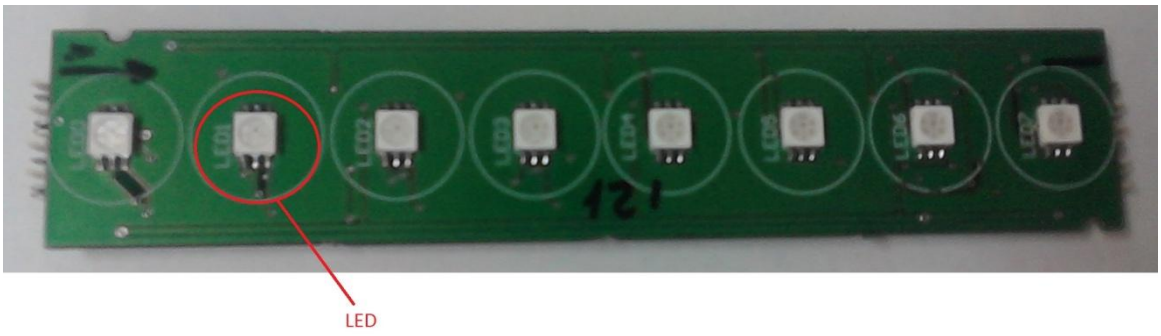


Fig. 16 – Placa de leds: anverso

Cada placa de leds tiene tres MAX7315, uno para el color rojo, otro para el azul, y otro para el verde. Cada placa tiene ocho leds RGB, por tanto cada MAX7315 controla ocho leds. Cada dispositivo MAX7315 tiene una dirección slave, que se puede modificar mediante unos microinterruptores existentes en la misma placa.

En la Fig. 17 se puede ver un esquema básico de conexionado. Los pines AD0, AD1 y AD2 están conectados a los microinterruptores, y son los encargados de seleccionar la dirección slave del MAX7315. Cada salida P0 - P7 está conectada al Led 0 – Led 7 de esa placa. En este esquema faltarían las resistencias pull-up necesarias para el correcto funcionamiento de la comunicación I2C y las resistencias limitadoras de corriente de los leds.

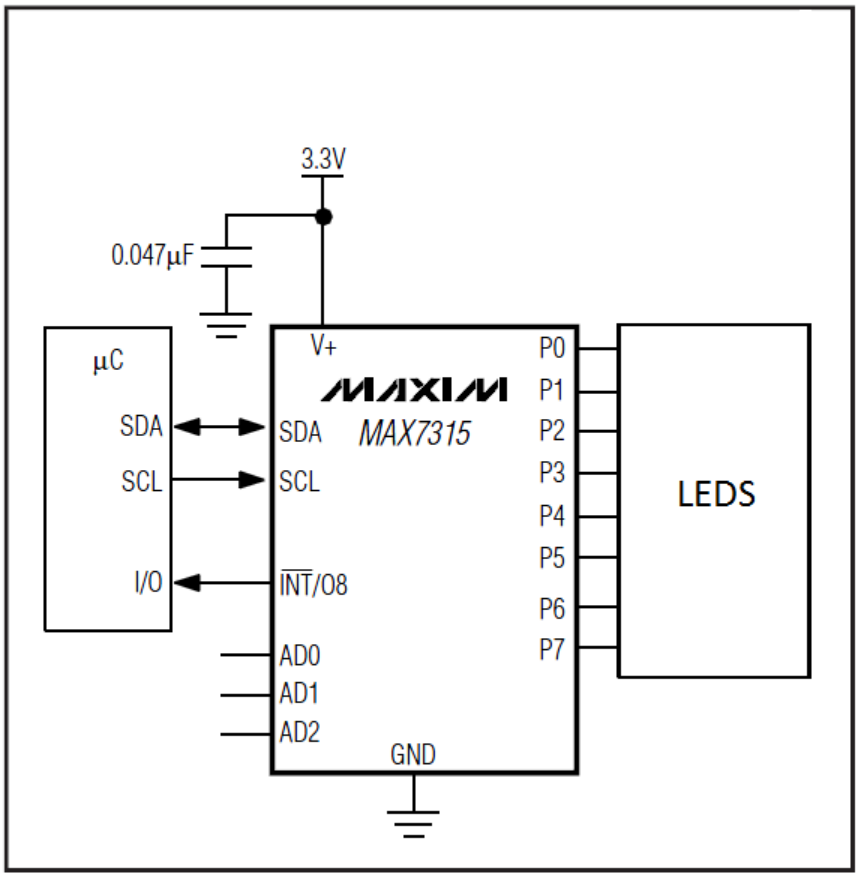


Fig. 17 – Esquema conexión MAX7315

El MAX7315 permite realizar tanto control global de las salidas como control PWM, además del control de intensidad master. También tiene posibilidades de parpadeo y de interrupciones.

Para el presente proyecto sólo he utilizado el control PWM y el control de intensidad global. Como el control global debe de estar desactivado para usar el PWM, se ha decidido utilizar el control de intensidad master para el brillo, y el control PWM para modificar el color. El control de intensidad master tiene 15 niveles de control (ver tabla 1), del 1 al 15, ya que el nivel 0 deshabilita el PWM. El PWM tiene 16 niveles de control para cada nivel de intensidad master, por tanto combinándolos se pueden alcanzar los 240 niveles.

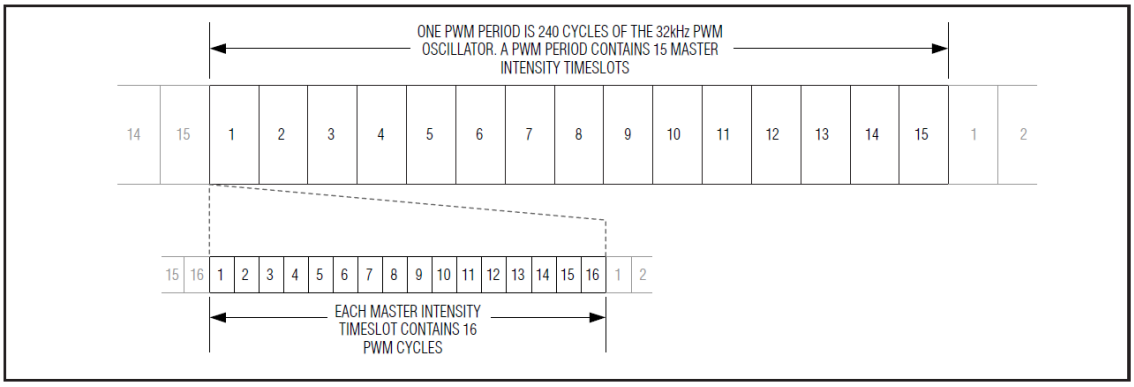


Fig. 18 – PWM

Para despejar dudas del funcionamiento del MAX7315, se ha utilizado el conversor USB-I2C S310425 que permite manejar el bus I2C desde un PC de forma sencilla, con un programa de tipo terminal serie. Así se pueden probar los comandos que hay que mandar en cada caso, antes de entrar en la programación del microcontrolador.

Fig. 19 – Módulo comunicaciones USB-I2C S310425

Cuando se usa el conversor USB-I2C, el comando que hay que enviar contiene la dirección del MAX7315 que queremos modificar, el registro interno al que nos referimos, y el dato. Cuando se realiza la programación del microcontrolador, la dirección del MAX7315 se especifica aparte, por tanto el comando solo contiene la dirección interna y el byte o bytes de datos.

Cuando queremos actuar sobre los leds, los primeros registros que hay que modificar son el “Blink Phase 0 Outputs” (0x01), que hay que ponerlo a 0x00 y, el registro “Configuration” (0x0F) que también hay que ponerlo a 0x0, ya que queremos que el control global esté desactivado.

El siguiente registro es el “Ports Configuration” (0x03), en el que hay que declarar los puertos como entrada o salida. Para el caso de los leds que deban estar encendidos, el valor de ese bit en el registro es “0”. En los leds que queramos apagados, su bit vale “1”.

Los siguientes datos que hay que enviar son los que modifican el control de intensidad master y el control PWM. El control de intensidad master se controla con el registro 0x0E, y el PWM con los registros 0x10, 0x11, 0x12 y 0x13, controlando cada uno dos

leds. Además, estos cuatro registros de PWM se auto incrementan, es decir, se puede enviar el mensaje con la dirección slave 0x10 y los datos de los cuatro registros sin necesidad de mandar cada uno con su dirección slave.

Para comprender mejor los registros encargados del brillo (master) y del color (PWM), a continuación se muestran las tablas 1 y 2 respectivamente.

REGISTER	R/$\overline{W}$	ADDRESS CODE (HEX)	REGISTER DATA							
			D7	D6	D5	D4	D3	D2	D1	D0
MASTER AND GLOBAL INTENSITY			MSB MASTER INTENSITY			LSB	MSB O8 INTENSITY			LSB
Write master and global intensity	0		M3	M2	M1	M0	G3	G2	G1	G0
Read back master and global intensity	1									
Master intensity duty cycle is 0/15 (off); **internal oscillator is disabled;** all outputs will be static with no PWM	—		0	0	0	0	—	—	—	—
Master intensity duty cycle is 1/15	—		0	0	0	1	—	—	—	—
Master intensity duty cycle is 2/15	—		0	0	1	0	—	—	—	—
Master intensity duty cycle is 3/15	—		0	0	1	1	—	—	—	—
—	—		—	—	—	—	—	—	—	—
Master intensity duty cycle is 13/15	—	0X0E	1	1	0	1	—	—	—	—
Master intensity duty cycle is 14/15	—		1	1	1	0	—	—	—	—
Master intensity duty cycle is 15/15 (full)	—		1	1	1	1	—	—	—	—
O8 intensity duty cycle is 1/16	—		—	—	—	—	0	0	0	0
O8 intensity duty cycle is 2/16	—		—	—	—	—	0	0	0	1
O8 intensity duty cycle is 3/16	—		—	—	—	—	0	0	1	0
—	—		—	—	—	—	—	—	—	—
O8 intensity duty cycle is 14/16	—		—	—	—	—	1	1	0	1
O8 intensity duty cycle is 15/16	—		—	—	—	—	1	1	1	0
O8 intensity duty cycle is 16/16 (static output, no PWM)	—		—	—	—	—	1	1	1	1

Tabla 1 - Master, 08 Intensity register

REGISTER	R/W	ADDRESS CODE (HEX)	REGISTER DATA							
			D7	D6	D5	D4	D3	D2	D1	D0
OUTPUTS P1, P0 INTENSITY		0X10	**MSB OUTPUT P1 INTENSITY LSB**				**MSB OUTPUT P0 INTENSITY LSB**			
Write output P1, P0 intensity	0		P1I3	P1I2	P1I1	P1I0	P0I3	P0I2	P0I1	P0I0
Read back output P1, P0 intensity	1									
Output P1 intensity duty cycle is 1/16	—		0	0	0	0	—	—	—	—
Output P1 intensity duty cycle is 2/16	—		0	0	0	1	—	—	—	—
Output P1 intensity duty cycle is 3/16	—		0	0	1	0	—	—	—	—
—	—		—	—	—	—	—	—	—	—
Output P1 intensity duty cycle is 14/16	—		1	1	0	1	—	—	—	—
Output P1 intensity duty cycle is 15/16	—		1	1	1	0	—	—	—	—
Output P1 intensity duty cycle is 16/16 (static logic level, no PWM)	—		1	1	1	1	—	—	—	—
Output P0 intensity duty cycle is 1/16	—		—	—	—	—	0	0	0	0
Output P0 intensity duty cycle is 2/16	—		—	—	—	—	0	0	0	1
Output P0 intensity duty cycle is 3/16	—		—	—	—	—	0	0	1	0
—	—		—	—	—	—	—	—	—	—
Output P0 intensity duty cycle is 14/16	—		—	—	—	—	1	1	0	1
Output P0 intensity duty cycle is 15/16	—		—	—	—	—	1	1	1	0
Output P0 intensity duty cycle is 16/16 (static logic level, no PWM)	—		—	—	—	—	1	1	1	1
OUTPUTS P3, P2 INTENSITY		0x11	**MSB OUTPUT P3 INTENSITY LSB**				**MSB OUTPUT P2 INTENSITY LSB**			
Write output P3, P2 intensity	0		P3I3	P3I2	P3I1	P3I0	P2I3	P2I2	P2I1	P2I0
Read back output P3, P2 intensity	1									
OUTPUTS P5, P4 INTENSITY		0x12	**MSB OUTPUT P5 INTENSITY LSB**				**MSB OUTPUT P4 INTENSITY LSB**			
Write output P5, P4 intensity	0		P5I3	P5I2	P5I1	P5I0	P4I3	P4I2	P4I1	P4I0
Read back output P5, P4 intensity	1									
OUTPUTS P7, P6 INTENSITY		0x13	**MSB OUTPUT P7 INTENSITY LSB**				**MSB OUTPUT P6 INTENSITY LSB**			
Write output P7, P6 intensity	0		P7I3	P7I2	P7I1	P7I0	P6I3	P6I2	P6I1	P6I0
Read back output P7, P6 intensity	1									
OUTPUT O8 INTENSITY			See the master, O8 intensity register (Table 13).							

Tabla 2 - Outputs Intensity Register

A continuación se expone un ejemplo de cómo hay que configurar el MAX7315 para poner los leds de una placa al 80 % de brillo y con un nivel del 40 % de color.

La cadena a enviar sería:

START (S) 0x01 0x00 (S) 0x0F 0x00 (S) 0x03 0x00 (S) 0x0E 0xCF (S) 0x10 0x66 0x66 0x66 0x66 STOP

Como se ve, hay que enviar una señal de Start antes de introducir otro registro. A partir del 0x10, los registros se autoincrementan.

6 - UART

A diferencia del I2C, el UART es un puerto serie asíncrono (Universal Asynchronous Receiver-Transmitter). El UART transmite los datos bit a bit de manera secuencial. En el dispositivo de destino el otro UART recompone cada byte.

Son necesarias dos líneas para la comunicación, una para transmitir (TX), y otra para recibir (RX). Cuando conectamos dos dispositivos que queremos comunicar por UART debemos de conectar la línea TX del primero con la RX del segundo, y viceversa. Así conseguimos que lo que uno transmite, el otro lo reciba.

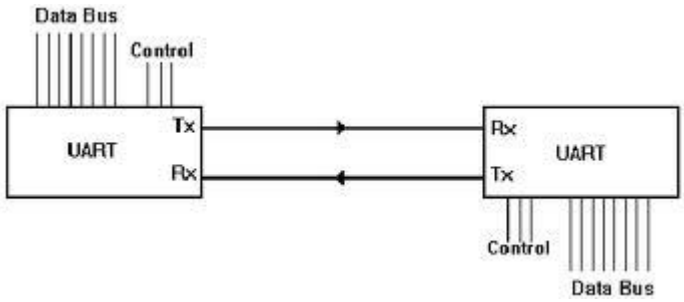


Fig. 20 – Conexión UART

Para que la transmisión y recepción de datos mediante UART sea satisfactoria, ambos módulos deben de tener programada la misma velocidad de transmisión (baudrate), longitud de caracter, paridad y bits de stop.

El protocolo de comunicación se muestra en la Fig 21. Estando la señal a “1”, el comando empieza con un “0” correspondiente al bit de start. A continuación los bits de datos (7 u 8), que empiezan por el bit menos significativo y terminan por el más significativo. El siguiente bit corresponde a la paridad (par o impar). En un caso concreto sin paridad, este bit se omite. Por último, la señal de stop (1 ó 2 bits), “1” lógico.

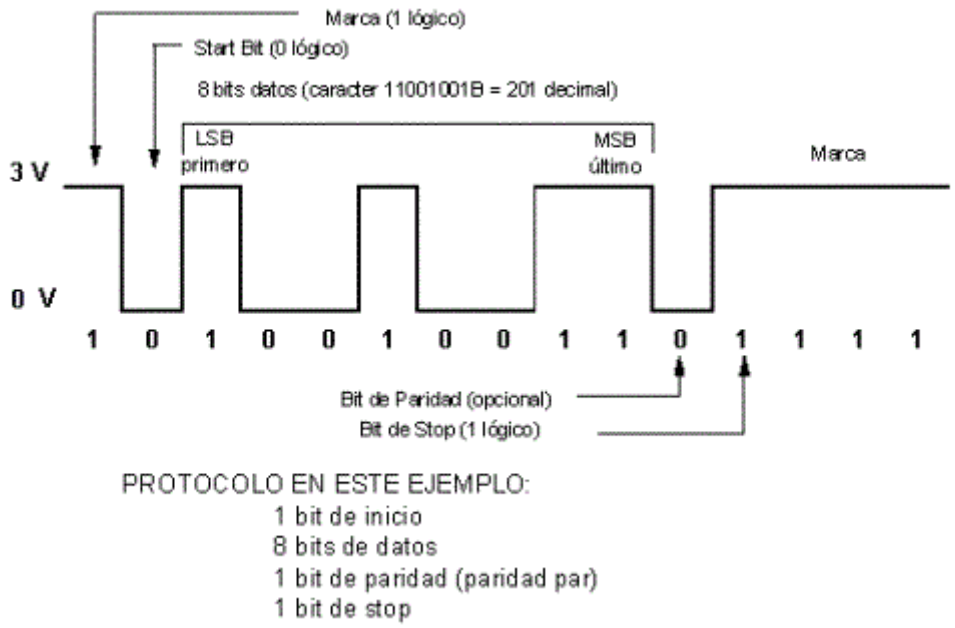


Fig. 21 – Protocolo comunicación puerto serie.

6.1 - UART en el proyecto

El bus UART se usa para enviar información desde el microcontrolador al módulo wifi, y viceversa.

En este caso, he usado un baudrate de 9600, datos de 8 bits, sin paridad, y 1 bit de stop. Por consiguiente, para cada byte de datos que se quiera enviar, habrá que mandar un "0" lógico (start), 8 bits del dato (empezando por el LSB), y un "1" lógico (bit de stop).

6.2 - UART en MSP430

Los principales registros para la configuración, transmisión y recepción de datos son los siguientes:

- P1SEL: Función UART en los pines P1.1 y 1.2.

- UCA0CTLW0:

    - UCPEN: Habilita o deshabilita paridad.

    - UCPAR: Paridad par o impar.

    - UC7BIT: Longitud dato 7 u 8 bits.

    - UCSPB: Uno o dos bits stop.

    - UCMODEx: Selección modo de trabajo.

    - UCSYNC: Modo síncrono o asíncrono.

- UCSSELx: Selecciona la fuente de reloj.

- UCSWRST: Mantiene congelado al USCI.

-UCA0BR0: Low Byte del Prescaler Baud Rate.

-UCA0BR1: High Byte del Prescaler Baud Rate.

-UCA0MCTLW:

- UCBRSx: Second modulation stage.

- UCBRFx: First modulation stage.

- UCOS16: Oversampling mode.

- UCA0STATW:

UCBUSY: Transmisión o recepción en progreso.

- UCA0RXBUF: Buffer de recepción.

- UCA0TXBUF: Buffer de transmisión.

- UCA0IE:

- UCTXIE: Habilita interrupción de transmisión.

- UCRXIE: Habilita interrupción de recepción.

- UCA0IFG:

- UCTXIFG: Flag interrupción de transmisión.

- UCRXIFG: Flag interrupción de recepción.

7 - WiFi

WiFi es un mecanismo de conexión de dispositivos de manera inalámbrica. Es necesario un punto de acceso para que los dispositivos puedan realizar la conexión.

El modo de identificar cada dispositivo en la red WiFi es mediante su dirección IP. Cada uno de ellos tiene asignada una dirección IP, que de forma lógica y jerarquizada, permite dirigirnos al dispositivo deseado. Esta dirección IP puede ser estática o dinámica, dependiendo de si el dispositivo tiene siempre la misma IP o, por el contrario, la va cambiando constantemente.

La red WiFi tiene como identificador el SSID o nombre de la red. Está compuesto por un máximo de 32 caracteres alfanuméricos.

En lo referente a la seguridad, la red puede requerir una clave de acceso o no. Si bien, hoy en día, la mayoría de redes usan clave de acceso, puesto que así son mucho menos vulnerables. Las claves pueden ser de tres tipos: WEP, WPA y WPA2. Temporalmente, aparecieron por ese orden, siendo cada una más segura que la anterior.

A continuación se muestra una pequeña comparativa de los módulos WiFi existentes en el mercado:

- Xbee Wifi

    Fabricante: Digi International.

    Precio: $35.00

    Configuración: Mediante comandos API o AT.

    Entradas/salidas digitales: 10.

    Datos serie: UART hasta 1Mbps, SPI hasta 3.5Mbps

    Encriptación: WPA, WPA2.

    Standard: 802.11/b/g/n.

    Diferentes posibilidades de antena.

- RN-XV (Elegido)

    Fabricante: Roving Networks.

    Precio: 30€.

    Entradas/salidas digitales: 8.

    Datos serie: UART hasta 464Kbps.

    Encriptación: WEP, WPA, WPA2.

Standard: 802.11/b/g.

- WF121A

Fabricante: Bluegiga Technologies.

Precio: 30.39€.

Entradas/salidas digitales: 38.

Datos serie: UART hasta 20Mbps.

Encriptación: WPA, WPA2, CCMP, TKIP.

Standard: 802.11/b/g/n.

- WiFi GA1000

Fabricante: Tibbo Technology.

Precio: 44€.

Entradas/salidas digitales: 5.

Datos serie: SPI.

Standard: 802.11/b/g.

7.1 - Módulo RN-XV

RN-XV es un módulo WiFi de ultra bajo consumo. Es fácilmente configurable por WiFi y por UART, usando comando ASCII. Este módulo tiene el firmware precargado, así, una implementación sencilla solo requiere conectar los pines de alimentación, masa, TX y RX.

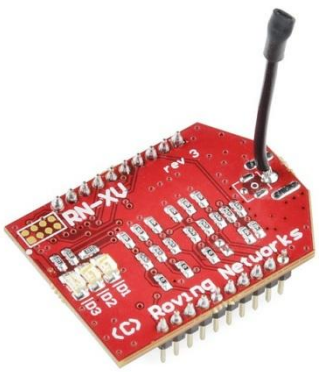


Fig. 22 – Módulo WiFi RN-XV

El RN-XV puede realizar conexiones a través de un punto de acceso, o en modo ad-hoc. Además soporta claves de autenticación WEP, WPA y WPA2.

Las características eléctricas son:

- Tensión de alimentación típica de 3.3V.

- Consumo en modo Sleep de 4mA.

- Consumo en modo Standby de 15mA.

- Consumo con RX conectado de 40mA.

- Consumo con TX conectado de 180mA.

Las características para la comunicación por UART son, por defecto:

- Baudrate 9600.

- Longitud de caracter 8 bits.

- Sin paridad.

- 1 bit de stop.

7.2. - Configuración y manejo del módulo WiFi

A continuación se explican los pasos a realizar para conseguir una comunicación exitosa entre el módulo WiFi y el ordenador. Solo se exponen las acciones a realizar en este proyecto, puesto que las funcionalidades del módulo son muy extensas y además no aportan nada para este caso.

En este proyecto se ha optado por una configuración mediante punto de acceso.

Para la configuración del módulo, hay que entrar en “command mode”. Para entrar en este estado hay que enviarle los caracteres “$$$”. El módulo responde con CMD.

Una vez en command mode, ya podemos configurar la conexión. Enviamos el comando “set ip dhcp 1” para poner DHCP a ON. Con esto conseguimos que tome la dirección ip y el gateway del punto de acceso.

El siguiente comando es “set ip protocol 2”. Conseguimos que solo se reciba datos de la ip host con la que está conectado.

Después se manda “set wlan join 0”. Así el módulo deja de intentar conectarse constantemente a la red por defecto.

Con el comando “set ip host <ip del ordenador>” que enviamos a continuación declaramos la dirección IP del ordenador con la que se tiene que comunicar el módulo.

El siguiente comando es para introducir la contraseña WPA de la red WiFi si la hubiese. Este comando pues, es opcional, dependiendo de si hay o no contraseña. “set wlan phrase <contraseña>”.

Si la contraseña es del tipo WEP, en lugar del comando anterior hay que enviar “set wlan key <contraseña>”.

Ya solo queda hacer un “join <nombre de la red>” para conectarnos a la red WiFi.

Si hacemos “save”, la próxima vez solo hay que escribir “join <nombre de la red>”.

Así pues, en el programa del microcontrolador solo habrá que enviar por UART “$$$” y “join <nombre de la red>”.

El cómo abrir la conexión con el PC no es incumbencia de este proyecto, ya que la parte del programa que haya que realizar en el ordenador se abordará en otro proyecto. Sin embargo, se necesita abrir la conexión para poder usar el DOT. Por tanto, explico a continuación el modo del que lo llevo a cabo.

En este PFC se ha utilizado el programa TCPIP Builder, de licencia abierta, con el que se puede establecer una comunicación TCP con una dirección IP. Una vez dentro se introducen las direcciones del ordenador y del módulo WiFi, y se pulsa en Connect. El sistema responde con un *HELLO*. Los datos a enviar se escriben en el cuadro de diálogo y se pulsa Send. Así, podemos enviar y recibir información entre el PC y el módulo WiFi.

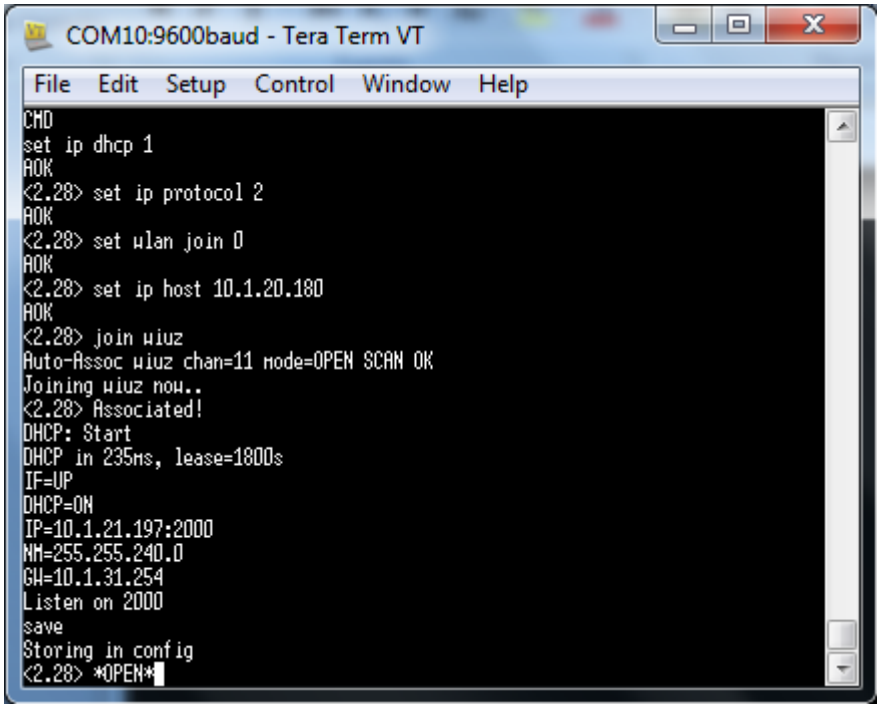


Fig. 23 – Ejemplo conexión a red WiFi

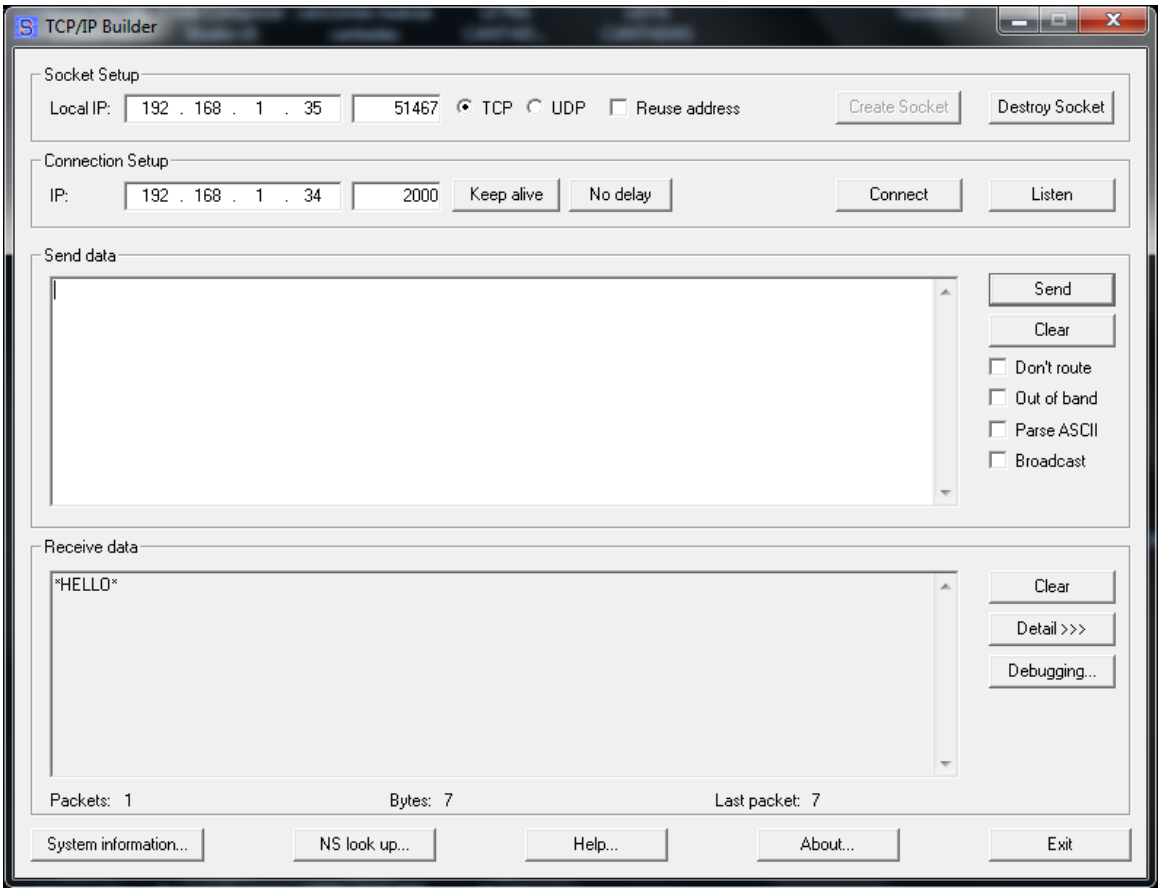


Fig. 24 – Captura TCPIP Builder

8 – Protocolo de comunicación vía WiFi

El protocolo que se va a seguir en la comunicación vía WiFi es el osgi4ami. Es un protocolo promovido por Tecnodiscap en varios foros relacionados con Ambient Assisted Living. Este protocolo busca homogeneizar las comunicaciones en los ambientes inteligentes. Es válido tanto para la comunicación WiFi como para cualquier otro estándar que se pudiera plantear. El hecho de que no tenga interfaces predefinidas para los dispositivos hace que sea un protocolo muy extendido en las aplicaciones relacionadas con los entornos inteligentes.

Entrando en la explicación del protocolo, hay que diferenciar entre los mensajes enviados desde el PC al nodo, y los enviados desde el nodo al PC.

8.1 - Mensajes PC->Nodo

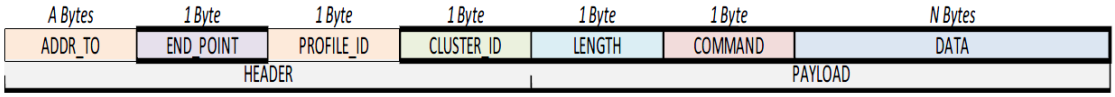


Fig. 25 – Formato mensajes PC->Nodo

Los bytes de ADDR_TO y PROFILE_ID no se utilizan.

El End_Point es el identificador del dispositivo lógico al que se refiere el mensaje. Cada componente físico implementa por lo menos dos dispositivos lógicos, un Base_Device y uno o más Virtual_Device.

El Cluster_Id es el grupo de atributos al que pertenece el mensaje. Cada dispositivo lógico puede implementar varios Cluster.

El campo Length es la longitud en bytes del resto del payload (N+1).

El Command es el comando del mensaje. Cada Cluster tiene definidos una serie de comandos que se pueden invocar.

El campo Data son los parámetros, la información específica del comando Command, perteneciente al cluster Cluster_Id correspondiente al dispositivo End_Point para el que se ha construido el mensaje.

8.2 - Mensajes Nodo->PC

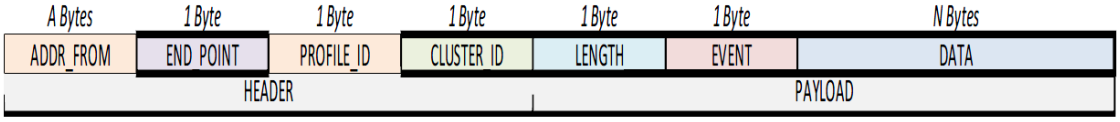


Fig. 26 – Formato mensajes Nodo->PC

Los campos en este caso son los mismos que en el caso anterior. La única diferencia es que el Command ahora es el campo Event.

8.3 - End_Point

Cada End_Point tiene los siguientes atributos:

- Dev_Category.

- Dev_Type.

- Dev_Implementation.

- Dev_Version.

- Clusters.

El End_Point del Base Device (encargado de gestionar las comunicaciones) es 0x10.

Los End_Point correspondientes a los dispositivos lógicos implementados en el nodo (Virtual Devices) pueden ir desde el 0x11 hasta el 0x1F. En nuestro caso, como solo hay un dispositivo implementado, se usa el 0x11.

En cuanto al Dev_Category, para el Base Device es 0x01 (04A_DEVICE), y para el Virtual Device es 0x30 (Simple_HMI). El valor Dev_Type es 0x01 (04A_DEVICE) y 0x04 (Led_Pannel) respectivamente.

8.4 - Cluster_Id

En cuanto a los cluster implementados son:

- Device (0x00): Conjunto de operaciones asociadas al dispositivo base de OSGi4AMI.

- Simple_HMI_Output (0x32): Conjunto de operaciones asociadas al dispositivo Simple HMI de salida.

8.5 - Command

Los command que se han implementado en esta versión del proyecto se pueden dividir en dos bloques:

- Los referentes al nodo de comunicaciones (Cluster_Id 0x00).

- Los referentes al panel de leds (Cluster_Id 0x32).

Los command del nodo de comunicaciones son:

- Get_Type (0x00).

- Ping (0x02).

Los del panel de leds son:

- Get_Max_Spots (0x00).

- Set_Backg_Color (0x01).

- Set_All_Color (0x02).

- Get_All_Color (0x03).

- Set_Spot_Color (0x04).

- Set_Backg_Effect (0x05).

- Set_Spot_Effect (0x06(.

- Set_Progress_Bar (0x07).

Get_Type

Este comando consulta el tipo de dispositivo. No necesita enviar ningún parámetro. El dispositivo debe responder con el evento Dev_Type.

Ping

Con el comando ping se comprueba que los mensajes llegan al dispositivo. No envía parámetros. La respuesta del dispositivo es un ACK.

Get_Max_Spots

La funcionalidad del Get_Max_Spots es consultar el número total de puntos del panel. No envía parámetros, y responde con el evento Dev_Max_Spots.

Set_Backg_Color

Se utiliza este comando para configurar todos los puntos con el mismo color y brillo. Los parámetros que envía son 4 bytes con la información de los colores. El primer byte contiene la información del color rojo, el segundo del verde, y el tercero del azul. El cuarto byte corresponde al brillo. “Red, Green, Blue, Bright”.

La respuesta es un ACK, si el envío es satisfactorio, o un Error si los parámetros enviados no son correctos.

Set_All_Color

Con Set_All_Color se configura una lista de puntos, cada uno con su color y su brillo. Los parámetros a enviar son una lista que contiene para cada led el número de punto, la información del rojo, el verde, el azul, y el brillo. “Npunto, Red, Green, Blue, Bright”.

La respuesta, como en el caso anterior, puede ser un ACK o un Error.

Get_All_Color

Este comando sirve para tener la información de cada punto del panel. No envía parámetros, y responde con el evento Dev_All_Color.

Set_Spot_Color

Este comando configura el color y el brillo de un único punto del panel. Los parámetros necesarios son el número de punto, la intensidad del rojo, la del verde, la del azul, y el brillo. “Npunto, Red, Green, Blue, Bright”.

La respuesta es un ACK si todo es correcto, o un Error si hay fallos en los parámetros.

Set_Backg_Effect

Configura todos los puntos del panel con el mismo efecto. Los parámetros que envía son “Effect_Type, y Time”. El Effect_Type informa del efecto que se debe aplicar a los leds. Por el momento el único efecto que se aplica a este panel es el parpadeo. Cuando el parámetro Effect_Type vale “1” se aplica parpadeo, cuando vale “0” no hay efecto. Por su parte, el parámetro Time informa del periodo que debe tener el parpadeo. Este parámetro, a diferencia del resto, ocupa 2 bytes, y contiene la información del periodo en milisegundos.

El evento con el que responde este comando es un ACK o Error si los parámetros son incorrectos.

Set_Spot_Effect

Se comporta de la misma manera que el anterior, pero solo afecta a un punto del panel. Así pues, esta función necesita un parámetro más, el número de punto, “Npunto, Effect_Type, Time”. La respuesta también es un ACK o un Error.

Set_Progress_Bar

Este comando se utiliza para encender un porcentaje de puntos del panel, empezando desde el uno hacia arriba. Los parámetros necesarios son, además de la información de los tres colores y el brillo, el porcentaje de puntos que deben de estar encendidos, “Percent On, Red, Green, Blue, Bright”. La respuesta es, como en los casos anteriores, un ACK.

8.6 - Event

ACK

El evento ACK es una respuesta como confirmación a un comando. Los parámetros que envía son “Cluster_Id, y Command”.

Error

El evento de Error notifica de este con los siguientes parámetros: “Cluster_id, Command, y Error_Code”.

El Error_Code puede ser “error_numero_parametros” (0x01) o “periodo_elevado” (0x02). El primero se usa para los command “set_spot_color”, “set_backg_color” y “set_all_color”, cuando la cantidad de parámetros que se reciben es distinta de la correcta. El segundo tiene lugar en los comandos “set_spot_effect” y “set_backg_effect”, cuando el parámetro “time” tiene un valor superior al periodo máximo definido. El periodo máximo que se ha definido es de 8 segundos, ya que un parpadeo que permanece más de cuatro segundos encendido y cuatro apagado no tiene sentido.

Not_Supported

Cuando el dispositivo recibe un comando que no es ninguno de los anteriores, responde con este evento. A diferencia del evento Error, que indica que los parámetros no son correctos, el Not_Supported notifica que el comando no está soportado.

Los parámetros que acompañan a este evento son: “Cluster_Id, y Command”.

Dev_Type

Éste informa de la lista de EndPoints implementados por el dispositivo físico. Los parámetros con los que responde son una lista que incluye para cada EndPoint: “End_Point, Dev_Category, Dev_Type, Dev_Implement, y Dev_Version”.

Dev_Max_Spots

Esta respuesta lleva el parámetro Max_Spots, que es el número de leds.

Dev_All_Color

Este evento contiene como parámetros una lista con el número de punto, el color rojo, el verde, el azul, y el brillo, de cada uno de los leds del panel. “Npunto, Red, Green, Blue, Bright”.

8.7. – Ejemplo

A continuación se muestra un pequeño ejemplo de comunicación.

Si quisiéramos poner todos los leds de color azul a la máxima intensidad (set_backg_color), el mensaje que habría que enviar desde el ordenador sería:

0x11 0x32 0x05 0x01 0xFF 0x00 0x00 0xFF

El dispositivo respondería con un ACK:

0x11 0x32 0x03 0x00 0x32 0x01

9 - Programa

A la hora de programar el microcontrolador se ha tratado siempre de que el código resultante fuera fácilmente ampliable en las posteriores modificaciones del proyecto. La mayoría de cambios leves que puedan llevarse a cabo en el dispositivo supondrán que la única modificación del código sea introducir los nuevos datos en la declaración de constantes.

El lenguaje elegido para la programación del microcontrolador ha sido el lenguaje C.

La estructura básica del programa consiste en atender mediante interrupción los datos que llegan del módulo wifi, enviar a la función correspondiente, y actuar.

La recepción por wifi se atiende con la interrupción UART Rx. La cadena de datos recibidos se almacena en el array “dato_recibido”. Una vez que la cadena de datos está completa, se pasa a procesar el dato. Para esto, se cuenta el número de bytes recibidos con la variable “z”, y se compara con la longitud que debe tener el dato (byte length del mensaje), cuando la cadena de datos ha terminado se pone z=255. Desde el main se comprueba cuándo “z” es igual a 255, entonces se pone a 0 y se llama a la función “elegir_opción”.

Mediante la función “elegir_opción” se almacenan los datos recibidos por UART en sus correspondientes variables. Atendiendo al valor de las variables “cluster_id” y “command” se averigua a qué función hay que ir.

Para el control de I2C se ha creado el array “guardado” donde se almacenan los valores de cada registro para todos los leds con sus tres colores.

La implementación de una interrupción “timer” que actúa cada 100 ms, pone a “1” el flag “enviar”, de modo que en el main se ejecutarán las funciones “enviar_I2C” y “actualiza_efecto” cuando este flag esté a “1”, o lo que es lo mismo, cada 100 ms. La función “actualiza_efecto” se explica más adelante.

La función “enviar_I2C” se encarga de crear el array de datos a enviar a partir del array “guardado”. Introduce la dirección I2C correspondiente, y envía el array de datos. Cada vez que se llama a la función “enviar_I2C” (cada 100 ms) se actúa sobre todos los leds. Así, cuando se quiera modificar alguno solo hay que cambiar el valor correspondiente del array “guardado”.

Conviene explicar, antes de entrar a ver específicamente lo que hacen las funciones que utilizan I2C para modificar el estado de los leds, que para apagar del todo el color de los puntos se ha optado por poner los pines correspondientes del MAX 7315 como entrada. Así pues, cuando se quiera que ese led emita de nuevo, se deberá configurar el pin como salida.

La función “set_backg_color” pone a todos los puntos con el mismo color y brillo. Para empezar comprueba si algún color debe de estar apagado del todo. Si es así configura los pines de los MAX7315 correspondientes a ese color como entradas. Si no, como salidas. Asimismo, si el brillo es cero, configura los tres colores como cero, esto es, poner los pines como entadas. Con esos datos de entrada/salida, y con los de brillo y

colores recibidos por UART, compone el array “guardado”. Este array es modificado totalmente, ya que esta función afecta a todos los puntos.

La función “set_spot_color” hace lo mismo que la anterior, pero solo con un punto. Así pues, mira si en el punto en cuestión el valor de alguno de los tres colores o del brillo debe ser cero. Como en el caso anterior configura el pin correspondiente a ese led como entrada o salida, según corresponda. En cuanto a modificar el valor de los colores, se ha explicado en el punto 5 de esta memoria que cada registro afecta a dos puntos (ver tabla 2).

Por tanto, como esta función solo afecta a uno, hay que crear el dato a guardar en el array con el nuevo valor de ese led y con el valor antiguo del array “guardado” correspondiente al otro led. En este caso no se modifica todo el array “guardado”, si no solo la parte correspondiente a esa pareja de puntos.

Atendiendo al protocolo de comunicaciones, debería de haber otra función que modifique una lista de puntos (“set_all_color”). Este command no se ha implementado como función en el código fuente, y se ha optado por llamar a la función “set_spot_color” para cada punto de la lista.

La función “set_progress_bar” calcula el número de puntos que deben de estar encendidos a partir del porcentaje recibido. A continuación llama a la función “set_spot_color” tantas veces como puntos deba haber encendidos.

Las otras dos funciones que también modifican el estado de los puntos son “set_spot_effect” y “set_backg_effect”. Lo que se busca con ellas es que un determinado led o todos parpadeen.

Para estas dos funciones, se ha creado un array “efecto”, que almacena el efecto que atañe a cada punto y su periodo. Este array se forma con dos bits correspondientes al tipo de efecto, y el resto con el periodo dividido entre 200 ms para ahorrar memoria RAM del microcontrolador (Fig. 27). Al guardarse el tipo de efecto en dos bits, se ha dejado espacio por si en otras versiones se quiere incluir otros tipos, además del parpadeo. Caben un total de tres efectos diferentes.

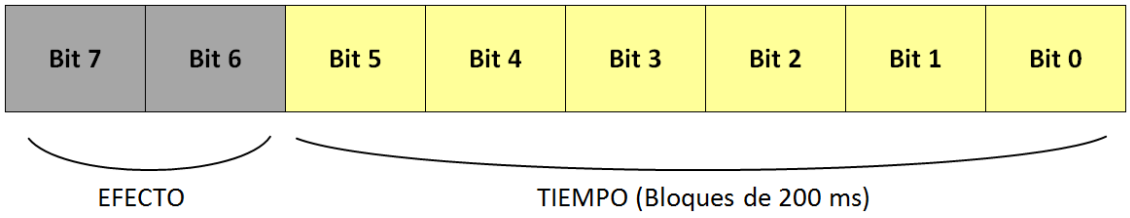


Fig. 27 – Array “efecto[ ]”

Para producir el parpadeo, además de crear el array “efecto”, necesitamos una función que periódicamente encienda y apague los puntos que corresponda. Para esto, en la

interrupción “timer” que hemos definido antes cada 100 ms, además de activar el flag “enviar”, incrementa un contador llamado “contador_efecto”.

La idea de este contador es que cuando sea múltiplo del semiperiodo expresado en bloques de 100 ms, conmute el estado del led. Para limitar el tamaño del contador, y se pueda reiniciar sin que esto afecte a la temporización que realiza, se ha hecho la siguiente operación:

$$2^4 * 3^3 * 5 * 7 = 15120$$

Por consiguiente el contador va de 0 a 15119. Los números que no se pueden formar multiplicando algunos de estos números entre sí, se han evitado con una lista de números prohibidos. Si el número de bloques de 100 ms es igual a alguno de estos números prohibidos se le suma 1.

La lista de números prohibidos es: 11, 13, 17, 19, 22, 23, 25, 26, 29, 31, 32, 33, 34, 37, 38, y 39.

En la función “actualiza_efecto”, a la que se llama cuando el flag “enviar” está a uno (cada 100 ms), se comprueba todo el array “efecto”, y en los puntos que tienen parpadeo, se mira si les toca conmutar. Esto es cuando el resto de la división entre el contador y el número de bloques de 100 ms es cero. Si les toca conmutar, hay que ver si alguno de los tres colores está definido como salida, y por tanto el punto está encendido. Si es así, se configuran los tres colores como entrada. Si por el contrario, el punto está apagado, se configuran como salida los colores que deban de estar encendidos.

En lo que respecta a las funciones que no modifican el estado de los puntos, deben de enviar datos por WiFi. Para esto hay que enviar los datos por el buffer Tx de la UART, así los recibe el módulo WiFi RN-XV y los envía al ordenador.

Al principio del código, en el main, hay que conectar el módulo WiFi a la red. Para esto, enviamos por el buffer Tx la secuencia para entrar en command mode y conectar el módulo. Hay que tener en cuenta los retardos necesarios antes y después de entrar en command mode especificados en el manual de usuario del módulo RN-XV. Una vez conectado, ya podemos enviar y recibir datos por UART.

Todas las funciones cuya misión es enviar datos al módulo WiFi lo que hacen es componer el array “frase”, que es el que se envía por UART, y llamar a la función “enviar_wifi”. Ésta función va colocando los sucesivos bytes del array “frase” en el buffer Tx.

9.1 - Diagramas de flujo

A continuación, se muestran los principales diagramas de flujo.

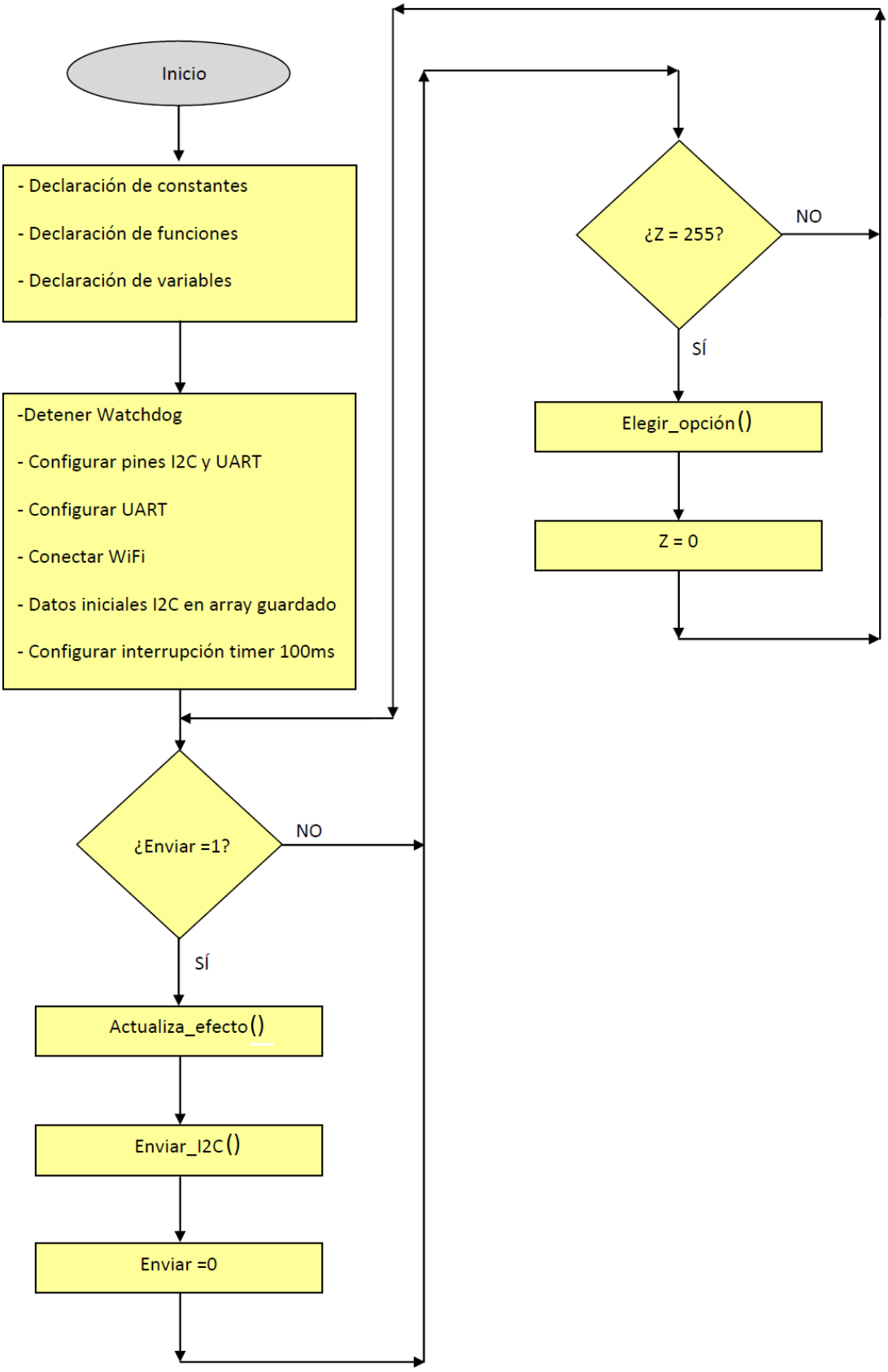


Fig. 28 – Diagrama de flujo main

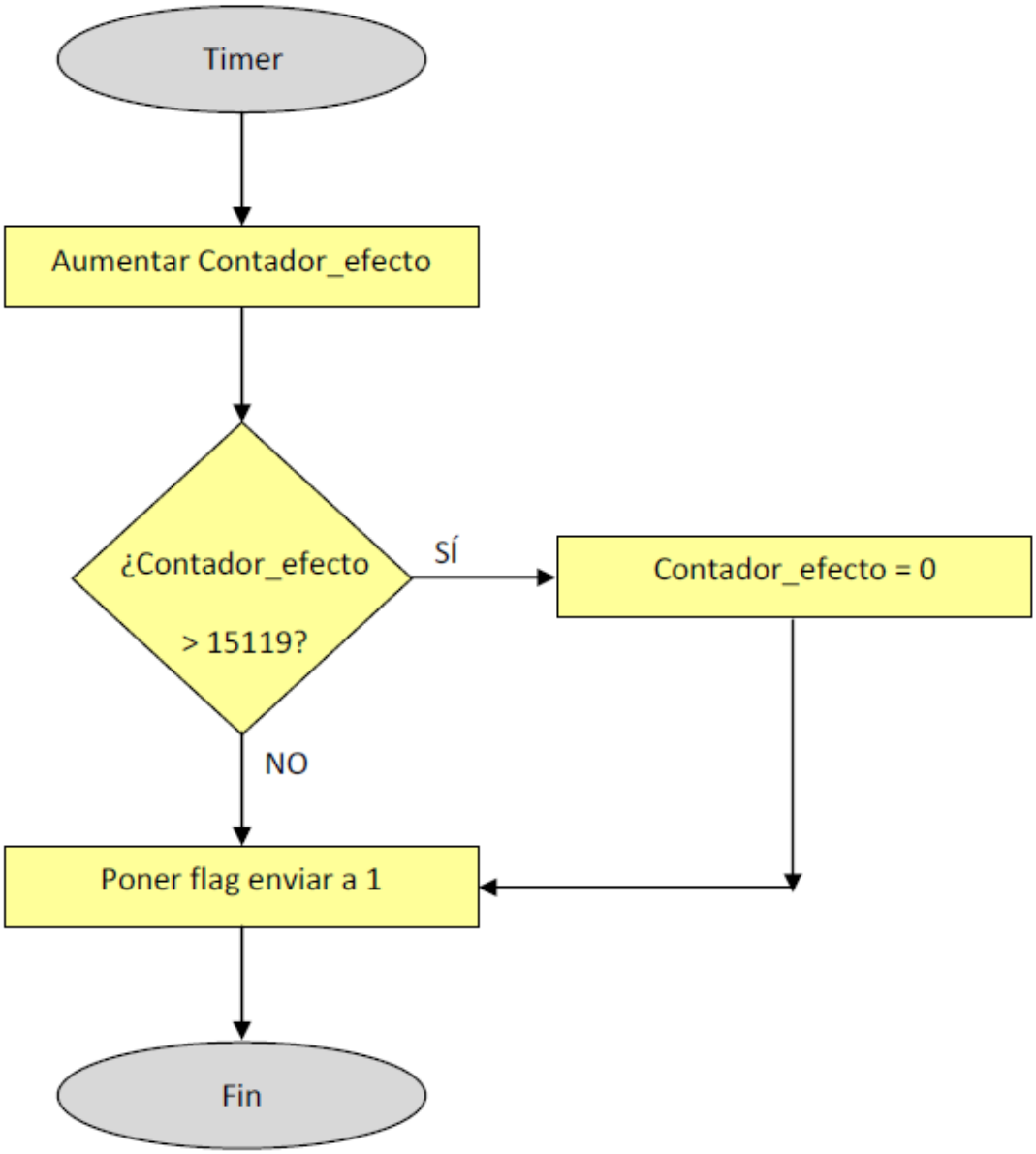


Fig. 29 – Diagrama de flujo interrupción timer

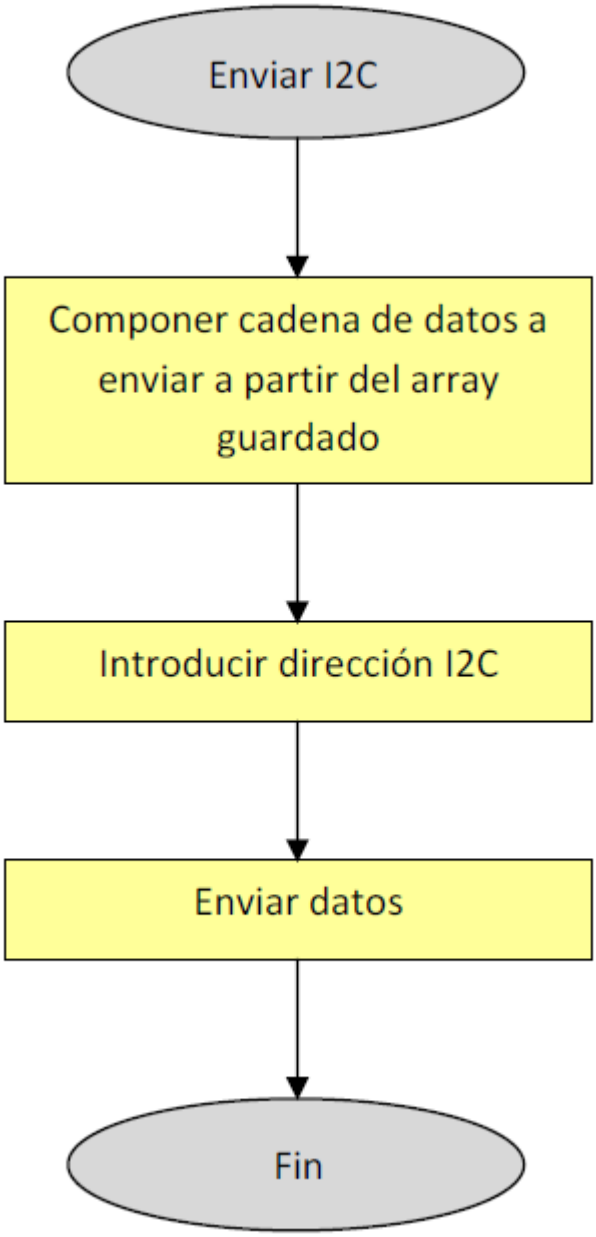


Fig. 30 – Diagrama de flujo envío de datos por I2C

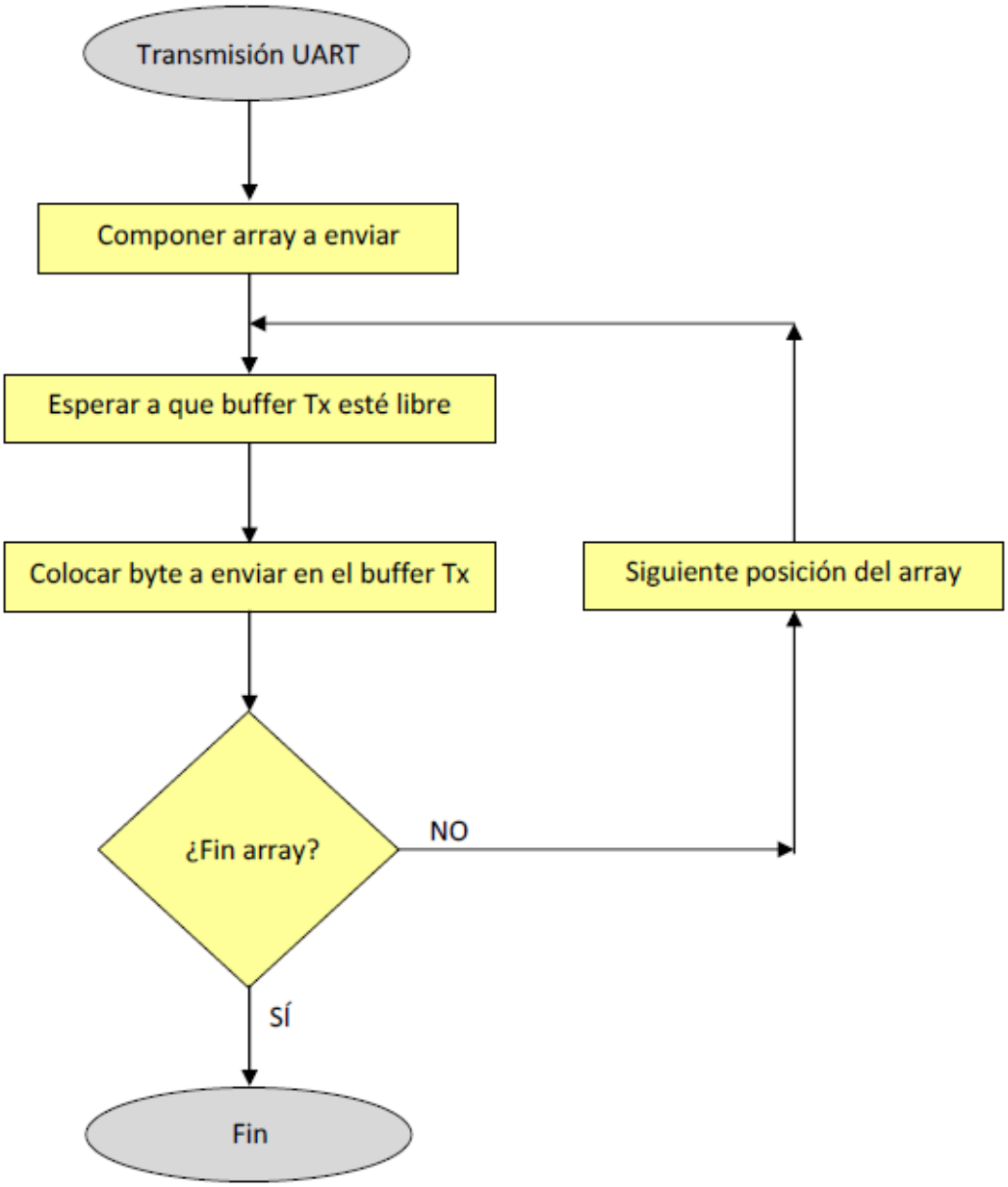


Fig. 31 – Diagrama de flujo transmisión de datos por UART

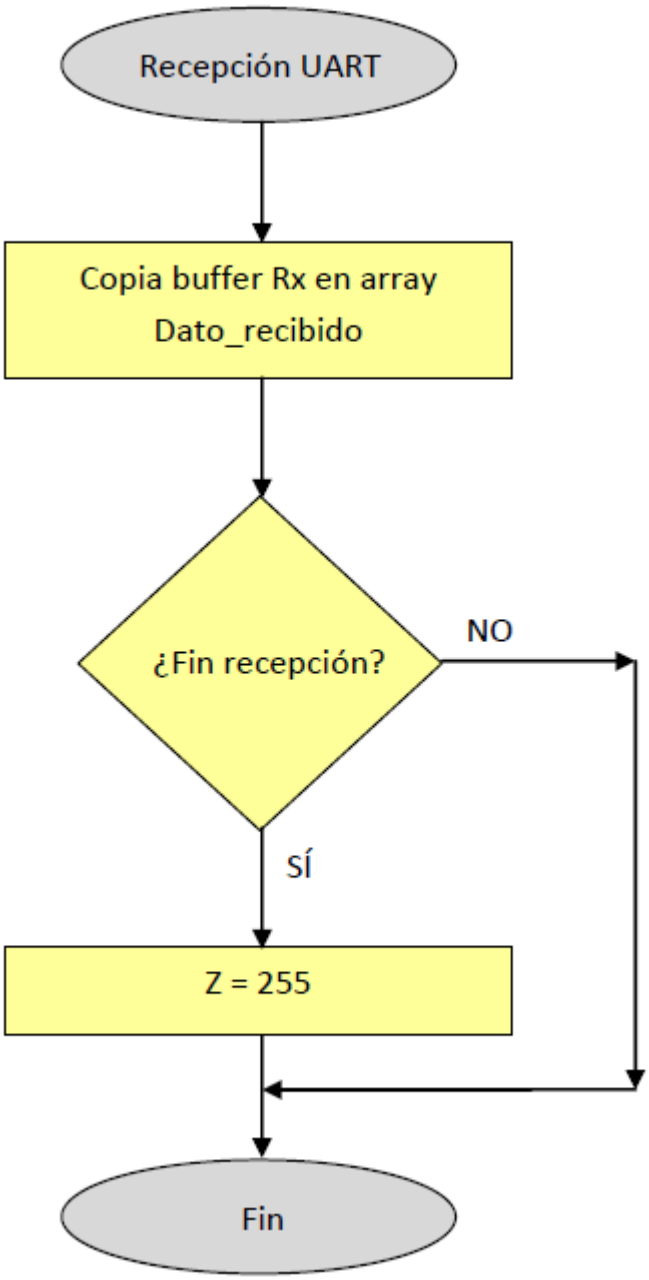


Fig. 32 – Diagrama de flujo interrupción recibir datos por UART

10 – Conclusiones

Este PFC ha mejorado mis conocimientos sobre diseño de PCBs, programación de microcontroladores y lenguaje C, y me ha aportado conocimientos de campos que desconocía, como la comunicación I2C, UART y WiFi.

Como resumen de las tareas realizadas en el proyecto destaco los siguientes puntos:

- Diseño de PCB.

- Montaje de PCB.

- Uso del módulo WiFi RN-XV.

- Manejo de los controladores de leds MAX7315.

- Programación del microcontrolador en lenguaje C.

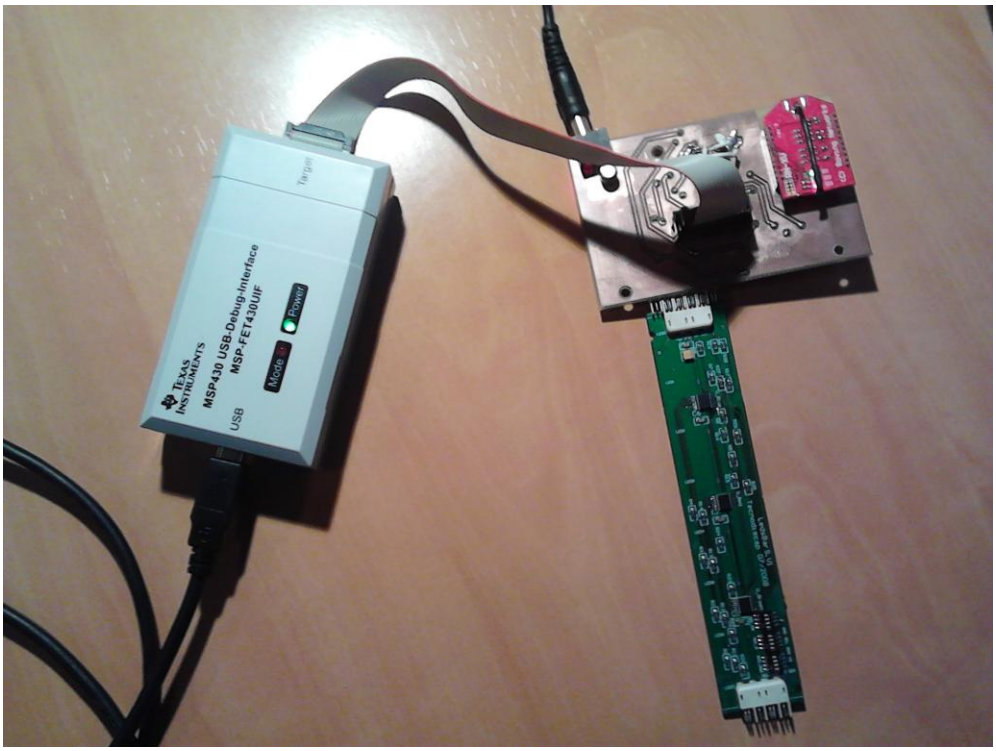
Con respecto a los problemas que me han surgido a lo largo del proyecto destacaría los errores en el diseño de los pines de dos componentes de la PCB, así como la conexión errónea de dos de las pistas. Estos problemas los he solucionado soldando cables en la PCB de manera que las conexiones fueran las correctas. Todos los fallos que había en el diseño de la PCB están corregidos en los planos adjuntos en el Anexo A.

Otro motivo de problemas durante la ejecución del proyecto, fue la conexión a red WiFi con el módulo RN-XV a la red “wiuz” de la escuela, debido a su baja intensidad de señal y los continuos cambios de dirección IP. Para solventar este problema he utilizado un router con el que he creado una red, usando direcciones IP fijas.

En cuanto al estado actual del proyecto, ha quedado solucionada la comunicación WiFi con el ordenador y UART con el microcontrolador. También están implementadas todas las funciones respectivas a la modificación de los leds por I2C.

En los desarrollos futuros lo que queda por hacer es introducir el sensor de intensidad luminosa y programar el resto de funciones respectivas a la comunicación con el ordenador. En mi opinión, creo que se deberían de introducir funcionalidades que permitan la manipulación de las redes WiFi y las direcciones IP desde el microcontrolador.

También se deberá desarrollar un programa de ordenador que permita la comunicación con el dispositivo de orientación temporal de una manera sencilla e intuitiva.



Fi. 33 – Emulador conectado a la placa

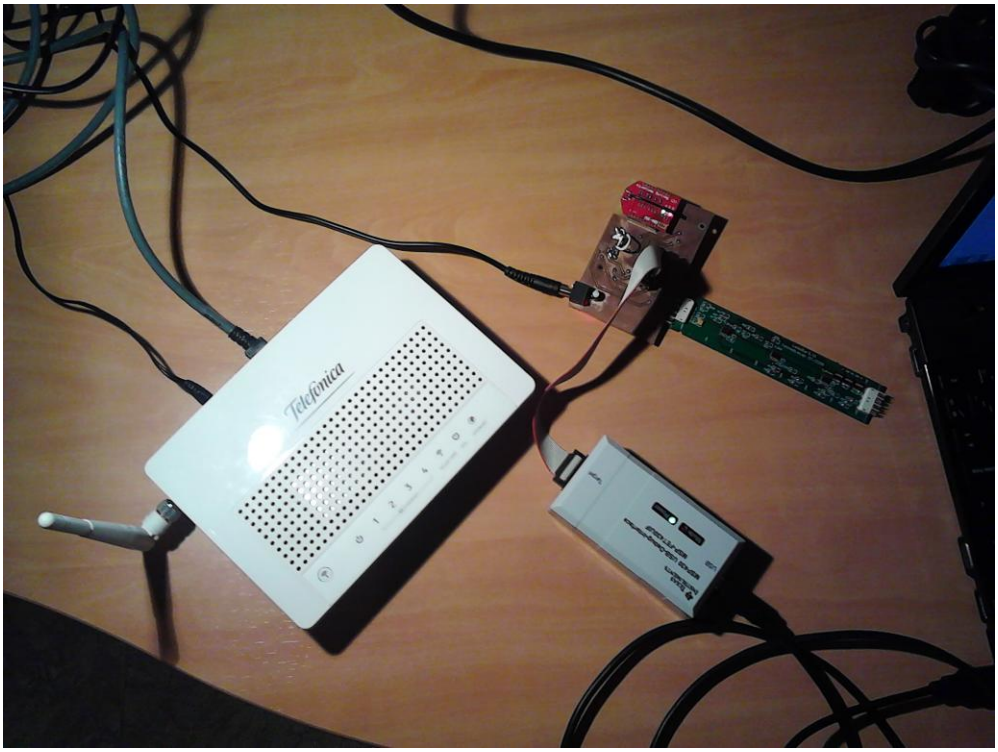


Fig. 34 – Puesto de trabajo

11 – Bibliografía

http://www.digikey.com

http://es.farnell.com

http://www.ti.com

http://es.wikipedia.org/

http://en.wikipedia.org/

http://www.rovingnetworks.com

http://www.digi.com

http://www.superrobotica.com/s310425.htm

Multitud de foros y páginas web sobre comunicación UART e I2C.

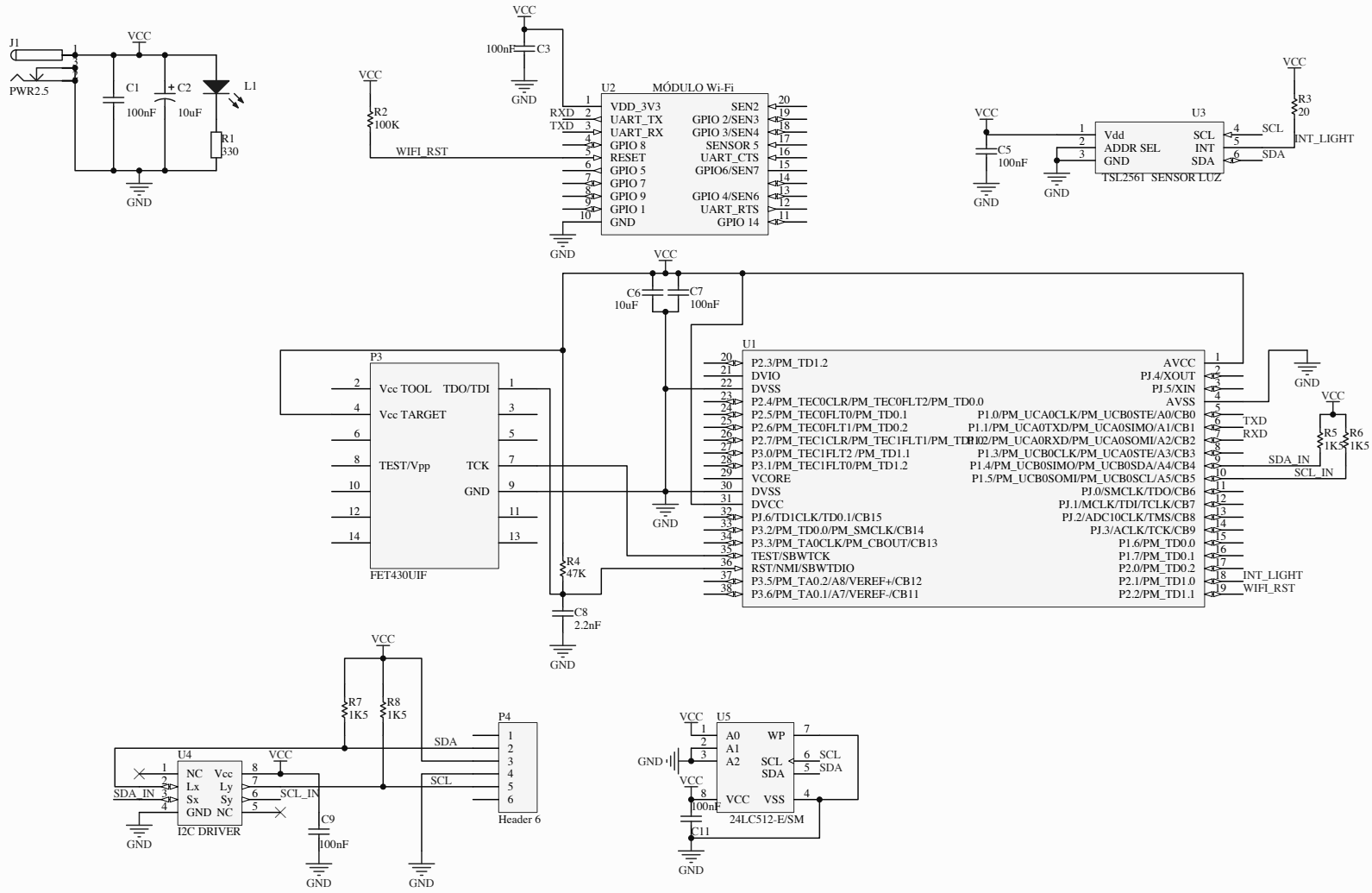
Guía de usuario de la familia MSP430x5xx y MSP430x6xx de Texas Instruments.

Manual de usuario y referencia de comandos WiFly-EZX RN-XV, de Roving Networks.

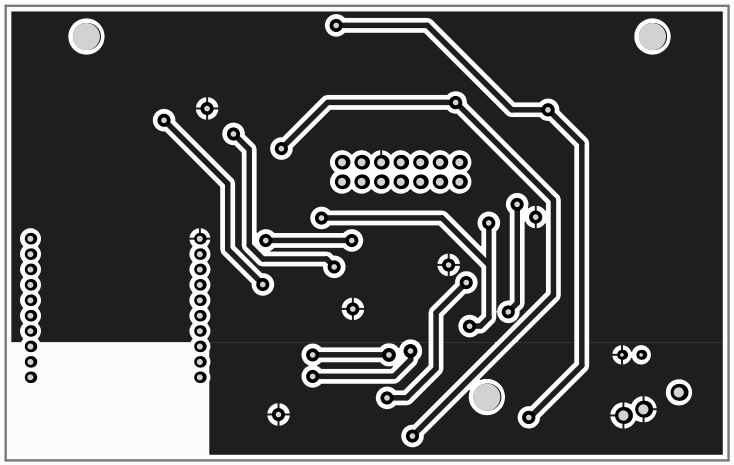
Multitud de datasheet en formato pdf extraídas tanto de las páginas web de los fabricantes como de diversas web de distribuidores de componentes electrónicos.

Anexo A

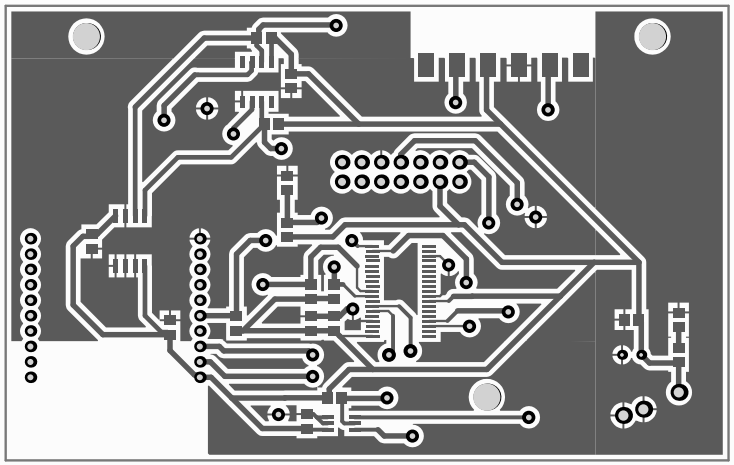
Planos



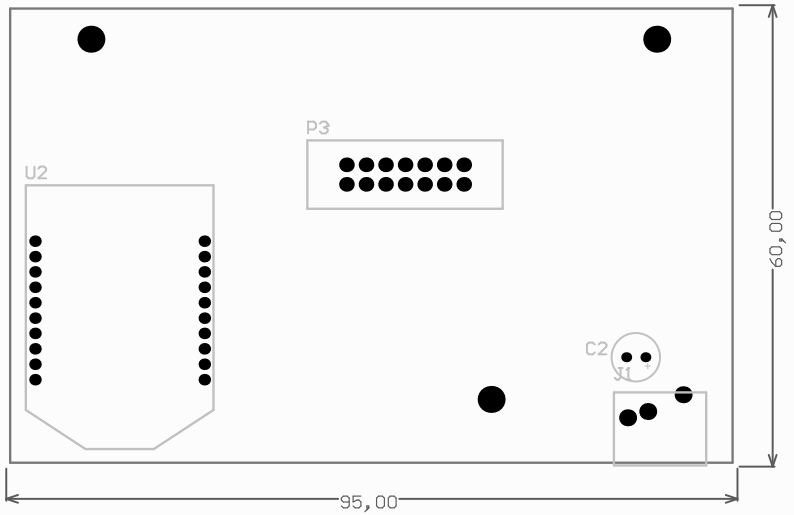
Nombre: Roberto Pérez Cacho	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Proyecto: Placa de control wifi para dispositivo de orientación temporal	Plano nº: 1
Escala: S/E	Plano: Esquema general del circuito



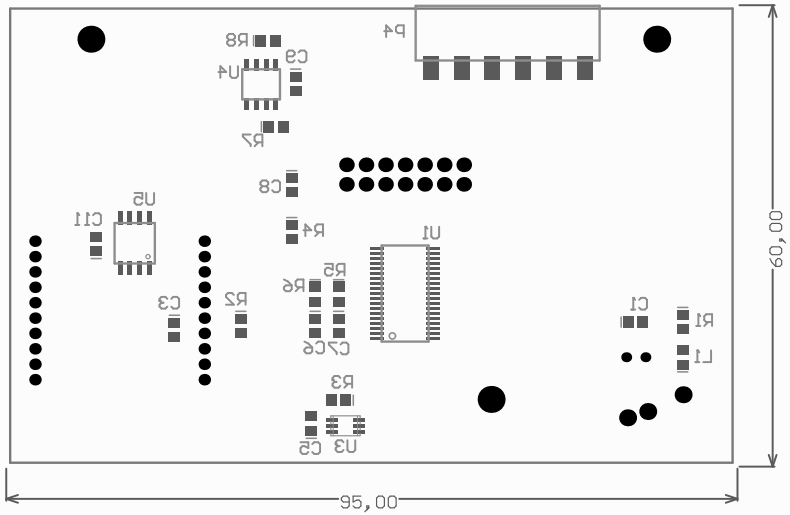
Nombre: Roberto Pérez Cacho		ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
	Proyecto: Placa de control wifi para dispositivo de orientación temporal	Plano nº: 2.1
	Escala: 1:1	Plano: Circuito impreso cara top



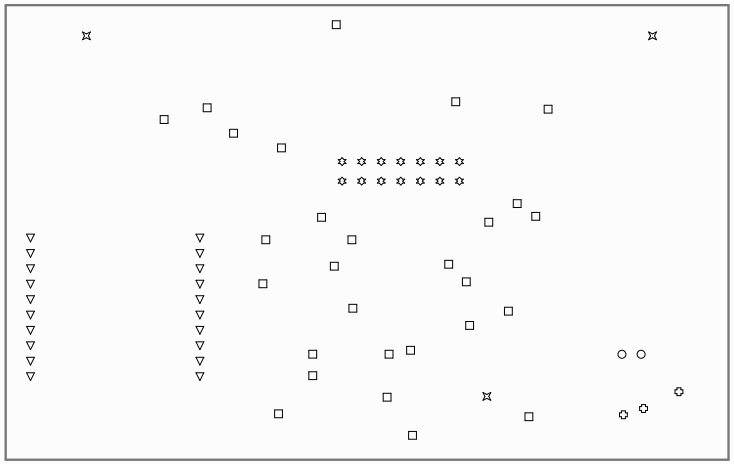
Nombre: Roberto Pérez Cacho	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Proyecto: Placa de control wifi para dispositivo de orientación temporal	Plano nº: 2.2
Escala: 1:1	Plano: Circuito impreso cara bottom



Nombre: Roberto Pérez Cacho	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
Proyecto: Placa de control wifi para dispositivo de orientación temporal	Plano nº: 3.1
Escala: 1:1	Plano: Serigrafía de componentes cara top



Nombre: Roberto Pérez Cacho		ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA
	Proyecto: Placa de control wifi para dispositivo de orientación temporal	Plano nº: 3.2
	Escala: 1:1	Plano: Serigrafía de componentes cara bottom



Symbol	Hit Count	Tool Size	Plated	Hole Type
○	2	0.45mm (17.716mil)	PTH	Round
□	28	0.7112mm (28mil)	PTH	Round
▽	20	0.75mm (29.528mil)	PTH	Round
✡	14	1.02mm (40.157mil)	PTH	Round
✪	3	1.3mm (51.181mil)	PTH	Round
⌑	3	3.5mm (137.795mil)	PTH	Round
	70 Total			

Nombre: Roberto Pérez Cacho	ESCUELA DE INGENIERÍA Y ARQUITECTURA UNIVERSIDAD DE ZARAGOZA	
	Proyecto: Placa de control wifi para dispositivo de orientación temporal	Plano nº: 4
	Escala: 1:1	Plano: Taladrado

Anexo B
Listado de componentes

Cantidad	Descripción	Referencia (Distibuidor)	Precio (hasta 10u.)	Precio (+500u)
1	Conector de alimentación	1217039 (Farnell)	1,02	0,66
1	Conector 6 vías	1638946 (Farnell)	1,91	1,32
1	Conector emulador	1099256 (Farnell)	0,60	0,26
1	Sensor luz	1226888 (Farnell)	2,31	1,79
1	Amplificador bus I2C	1627047 (Farnell)	2,95	1,89
1	Led rojo alimentación	1226392 (Farnell)	0,23	0,127
1	Condensador 10uF	1962120 (Farnell)	0,057	0,039
5	Condensadores 100nF	1740665 (Farnell)	0,041	0,019
1	Condensador 2,2nF	1301778 (Farnell)	0,036	0,032
1	Condensador electrolítico 10uF	9452338 (Farnell)	0,093	0,038
1	Resistencia 330 ohm	1099797 (Farnell)	0,029	0,023
1	Resistencia 100 Kohm	1469860 (Farnell)	0,024	0,018
1	Resistencia 20 ohm	2057582 (Farnell)	0,021	0,01
4	Resistencias 1500 ohm	1099801 (Farnell)	0,03	0,025
1	Resistencia 47 Kohm	1469929 (Farnell)	0,024	0,018
1	Módulo WiFi RN-XV	Cooking Hacks	40,80	40,80
1	Microcontrolador MSP430F5172	2057047RL (Farnell)	5,34	1,85