Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Evaluación del consumo de energía de Arduino

TITULACIÓN Ingeniería Técnica de Telecomunicación, especialidad Sistemas de Telecomunicación

AUTOR: Rubén Martínez Fernández

DIRECTOR: José Polo

FECHA: 19 de Septiembre de 2014

Título: Evaluación del consumo de energía de Arduino

Autor: Rubén Martínez Fernández

Director: José polo

Data: 19 de Septiembre de 2014

Resumen

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus entradas y controlar luces, motores y otros actuadores. Los proyectos hechos con Arduino pueden ejecutarse de forma autónoma, sin estar conectado a un ordenador.

El abanico de aplicaciones desarrolladas mediante esta plataforma es cada vez mayor. Muchas de estas aplicaciones tienen limitados los recursos energéticos, por ello se hace necesaria una optimización del consumo de energía.

Se presenta un estudio del sistema valorando soluciones de software y hardware con el fin del optimizar los recursos del mismo y reducir el consumo de energía.

A su vez se realiza la medición de consumos en función de distintas soluciones y alternativas encontradas.

El objetivo final es crear un patrón que permita a cualquier desarrollador que utilice Arduino reducir el consumo de su proyecto.

Title: Evaluación del consumo de energía de Arduino

Author: Rubén Martínez Fernández

Director: José Polo

Date: September, 19th 2014

Overview

Arduino is an open prototyping software-based, flexible and easy to use electronic hardware platform. It was created for artists, designers, hobbyists and anyone interested in creating environments it or interactive objects.

It can take information from the environment through its inputs and controlling lights, motors, and other actuators. Projects made with Arduino can run without connecting to a computer.

Range of applications developed by this technology is growing. Most of these applications have limited energy resources, the optimization of power consumption is required.

A study evaluating the system software and hardware solutions for the purpose of optimizing its resources and reduce power consumption is presented.

Turn consumption measurement in terms of different solutions and alternatives encountered is performed.

Ultimate goal is to create a pattern to provide any Arduino developer a way to reduce consumption of his project.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

A mis padres, por estar siempre a mi lado
y apoyarme incondicionalmente.

INTRODUCCIÓN

Arduino es una herramienta con la que sentir y controlar el entorno. Su modo de funcionamiento es simple, consta de una plataforma de código abierto basada en una placa con un micro controlador y una aplicación desde la que desarrollar los programas para la placa.

Puede usarse para infinidad de proyectos ya que permite crear objetos que interactúen con interruptores, sensores, luces, motores y otros dispositivos para obtener información diversa. Éstos a su vez pueden funcionar comunicándose con un programa o ser totalmente autónomos, lo que amplía enormemente su rango de aplicaciones.

Algunos proyectos realizados con Arduino van desde un sencillo pero eficaz sistema de seguridad para abrir maletines a un complejo sistema domótico. El límite está en la imaginación y talento de los desarrolladores.

Atendiendo a todas las virtudes del sistema es fácil comprender que sean miles los usuarios de la plataforma y multitud los proyectos llevados a cabo. Uno de los puntos fuertes de tales proyectos es la capacidad que tiene Arduino para funcionar autónomamente. Es aquí de donde nace la necesidad de valorar el consumo de dicha plataforma ya que para que un sistema autónomo sea útil es imperativo poder garantizar su funcionamiento en el tiempo.

En este trabajo se estudiará el consumo de varias placas, con mayor hincapié en Arduino Uno, y se propondrán diferentes soluciones para optimizar y reducir el consumo de dichas placas de modo que sean lo más eficiente posible.

Se ofrece un estudio teórico y práctico de soluciones de software y hardware que pretenden reducir el consumo. Dichas soluciones de software son fácilmente extrapolables al resto de placas arduino ya que comparten el mismo sistema de programación mientras que las de hardware deberían adecuarse a las necesidades de la placa en cuestión.

CAPÍTULO 1. MARCO TECNOLÓGICO

1.1. Arduino

Según el propio fabricante **"***Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos*". Básicamente nos encontramos con un microcontrolador sobre una placa con múltiples puertos de entrada y salida que permiten controlar e interactuar con infinidad de dispositivos.

El éxito de Arduino radica en la facilidad de programación del sistema, el enorme abanico de *shields* disponible para las placas y el bajo coste de las mismas. Además hemos de añadir fiabilidad y un tamaño reducido. Es a todas luces una herramienta práctica y útil para diseñadores, aficionados y cualquier persona interesada en crear objetos o entornos interactivos.

1.1.1. Tipos de Arduino

En la Tabla 1.1 se describen distintos tipos de Arduino y se justifica el modelo sobre el cual se realiza el trabajo.

Se puede encontrar una placa de Arduino prácticamente para cualquier necesidad. Es por ello que a pesar de compartir la idea original de estar formado por un microcontrolador y una placa, los diseños de éstas distan mucho unas de otras.

Tabla 1.1 Comparativa modelos Arduino

Característica de Arduino	UNO	Mega 2560	Leonardo	DUE
Tipo de microcontrolador	Atmega 328	Atmega 2560	Atmega 32U4	AT91SAM3X8E
Velocidad de reloj	16 MHz	16 MHz	16 MHz	84 MHz
Pines digitales de E/S	14	54	20	54
Entradas analógicas	6	16	12	12
Salidas analógicas	0	0	0	2 (DAC)
Memoria de programa (Flash)	32 Kb	256 Kb	32 Kb	512 Kb
Memoria de datos (SRAM)	2 Kb	8 Kb	2.5 Kb	96 Kb
Memoria auxiliar (EEPROM)	1 Kb	4 Kb	1 Kb	0 Kb

Se pueden añadir muchos más modelos a la lista como el Arduino micro, una placa de reducido tamaño con entrada micro USB o el novedoso LilyPad diseñado para integrarse en prendas téxtiles. Merece una mención especial el inminente Arduino Tre, una placa que todavía no se comercializa y que otorga 100 veces más potencia de procesado que los modelos UNO y Leonardo.

1.1.2. Arduino UNO

Arduino Uno es una placa que incorpora un microprocesador ATMega 328, consta de 14 pines digitales de salida/entrada (de los cuales 6 pueden usarse como salidas PWM) y 6 pines de entrada analógica. Además consta de un resonador cerámico de 16MHz, una entrada USB, un conector de alimentación jack y un botón de reset.

Todo esto permite que con tan solo conectar nuestro Arduino mediante USB a nuestro Pc se pueda estar trabajando en cuestión de minutos. Actualmente existe la tercera revisión de la placa que entre otras mejoras aporta un circuito de reset más potente.

Arduino Uno puede alimentarse directamente vía USB o conectándolo a una fuente externa de energía. La fuente de energía se escoge automáticamente en caso de que se alimente por las dos vías.

Para programar la placa únicamente ha de descargarse el software libre desde la misma página de Arduino, escoger la placa con la que vamos a trabajar en el mismo programa y cargar cualquier aplicación ejemplo. Esta facilidad se debe a que el ATM328 viene precargado con un cargador de arranque (bootloader), un programa que cubre las necesidades básicas de un sistema operativo y permite el uso de la placa desde el primer momento.

Podría decirse que Arduino Uno es el modelo Standard. Consta de un número de entradas y salidas aceptable y la facilidad de uso mediante su puerto USB le da un plus extra. De hecho el starter kit de la marca (un pack con varios componentes para llegar a diseñar y montar unos 15 proyectos) incluye el Arduino Uno como placa para introducirse al mundo de la programación de microprocesadores. El precio de la placa ronda los 22 € en infinidad de distribuidores lo que hace todavía más accesible la placa al público en general.

Desarrolladores, escuelas, universidades, artistas, empresas, todos se benefician de un sistema simple, integrado sobre una placa, fácil de usar y barato. Puede afirmarse que Arduino Uno es el modelo más utilizado a la hora de diseñar.

La Tabla 1.2 muestra las especificaciones de la placa Arduino Uno.

Tabla1.2. Especificaciones Arduino Uno

Microcontrolador	ATmega328
Tensión operativa	5V
Tensión de entrada (recomendada)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de reloj	16 MHz

1.2. Estimación de consumo y vida útil

Arduino es una plataforma diseñada para dar vida a múltiples proyectos. En la red aparecen infinidad de ellos. Luces que dibujan figuras en la oscuridad, molinos de viento, riegos automáticos, simuladores de terremotos, etc. Estos ejemplos se diseñaron para permanecer estáticos y cerca de una fuente de alimentación. Pero ¿qué ocurre cuando se proyecta un diseño autónomo? Estaciones meteorológicas, linternas, coches teledirigidos y robots podrían ser implementados perfectamente con Arduino Uno. En estos ejemplos se debe recurrir a una alimentación con baterías (ver Tabla 1.3.).

Tabla 1.3. Capacidad en mA y descarga de baterías

Tipo	Capacidad (mAh)	Descarga (%mes)	Auto descargado (µA)
CR1212	18	1	0,25
CR1620	68	1	0,95
CR2032	210	1	3
NiMH AAA	900	30	375
Alkaline AAA	1250	2	35
NiMH AA	2400	30	1000
Alkaline AA	2890	2	80

Una opción fácil y barata para alimentar nuestro Arduino Uno es utilizar pilas alcalinas AAA. Teniendo en cuenta que ofrecen aproximadamente unos 1250 mAh y atendiendo al valor de corriente contínua reflejado en la tabla 2.1, se obtiene la siguiente relación:

$$\frac{1250\ mAh}{50\ mA} = 25\ h$$

El cálculo anterior muestra como en un caso ideal el sistema apenas estaría operativo un solo día. Hay que añadir el grado de descarga de la pila y es fácil observar que la vida útil alcanzará aproximadamente menos de un día.

Este dato es de vital importancia para aplicaciones a las que no se tenga fácil acceso y requieran cierto grado de autonomía. Estaciones submarinas o meteorológicas, sistemas autónomos de medida, podrían ser alguno de estos ejemplos.

Así pues se tiene una herramienta útil, potente, sencilla y barata que ofrece multitud de posibilidades, pero poco práctica para diseños que deban trabajar de forma autónoma durante largos períodos de tiempo.

La duda de si Arduino Uno es realmente práctico para diseños que deban trabajar largo tiempo con baterías es la motivación principal de este trabajo. Se estudiarán distintas propuestas, tanto de Hardware como de Software, así como la opción de placas alternativas con el fin de reducir y optimizar el consumo de la aplicación.

Se estudian las dos opciones en profundidad. Por motivos de espacio las soluciones de hardware se ubican en los anexos dentro del capítulo 4. En ese capítulo se presentan distintas propuestas para modificar Arduino y modelos alternativos. También se justifica la elección del modelo de bajo consumo sobre el que se realizarán diversas pruebas.

El cuerpo central del trabajo se centra sobre las soluciones de software debido a su facilidad de implementación y resultados inmediatos.

CAPÍTULO 2. SOLUCIONES SOFTWARE

Se presenta un estudio y varias alternativas de software para reducir el consumo de Arduino. Pueden realizarse sobre cualquier placa de Arduino o sobre placas que hayan sufrido las modificaciones de software detallas en el capitulo 4 ubicado en el anexo.

2.1. Sleep modes

Una de las principales herramientas de la que dispone Arduino para optimizar el consumo de energía es la utilización de los modos de trabajo denominados *Sleep*. Básicamente un modo sleep define el estado del microprocesador cuando no está trabajando, de modo que resulta extremadamente útil conocer y saber manejar a la perfección dichos modos de trabajo.

El microprocesador con el que trabaja Arduino Uno, el ATmega 328, dispone de seis modos distintos de trabajo que se pasan a describir. En el capítulo de mediciones se justifica los modos escogidos y se ofrecen resultados del funcionamiento de los mismos.

	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		
Power-down								X(3)	X				X		X
Power-save					X		X(2)	X(3)	X	X			X		X
Standby(1)						X		X(3)	X				X		X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X		X

Tabla 2.1. Modos Sleep del ATMega 328

Cada uno de los distintos modos desactiva una cantidad diferente de recursos del microcontrolador. Se aprecia como el modo Idle es el más conservador mientras que el modo Power-down es el más agresivo. Será el diseñador de la aplicación el encargado de escoger qué modo se ajusta mejor a sus necesidades aunque a priori si el objetivo es reducir el consumo del proyecto el primer modo a tener en cuenta es el modo Power-Down.

2.1.1. Modo Idle

Al escoger este modo se detiene la CPU permitiendo el funcionamiento normal del resto de dispositivos. Básicamente para los relojes CPU y FLASH dejando correr el resto. Este modo permite al MCU despertar tanto mediante interrupciones externas como internas. Un modo para reducir el consumo consiste en desactivar el comparador analógico si dicha interrupción no es necesaria. Este modo de trabajo permite disponer de mayor cantidad de recursos a cambio de sacrificar consumo.

2.1.2. Modo ADC noise reduction

Este modo detiene los relojes I/O, CPU y FLASH dejando funcionar el resto. Esto mejora la percepción de ruido del ADC permitiendo una mejor resolución de medidas. Se puede despertar de este modo de trabajo mediante un reset externo, una interrupción del Watchdog, un reset del Brown-out, una interrupción del Timer/Counter2 o una interrupción externa en INT0 o INT 1 entre otras.

2.1.3. Modo Power-down

En este modo de trabajo el oscilador externo se detiene. Las interrupciones externas y el Watchdog continúan operando si están habilitados. Este modo para todos los relojes permitiendo operar únicamente los módulos asíncronos. Es el modo de trabajo más eficiente en lo que a ahorro de energía se refiere aunque tiene el inconveniente de ser el más restrictivo. A pesar de ello con una buena configuración es fácil de usar y existen varias bibliotecas que permiten su uso de una manera sencilla y eficaz. Es sin duda el modo más usado en todos los diseños pensados para un bajo consumo.

2.1.4. Modo Power-save

Este modo únicamente se diferencia del modo Power-down en que si Timer/Counter2 son habilitados seguirán funcionando durante el modo sueño. Obviamente si no son necesarios este modo se desaconseja dejando su puesto al Power-down.

2.1.5. Modo Stand-by

Un modo idéntico al Power-down con la excepción de que el oscilador sigue en funcionamiento. A la hora de despertar de este modo hay que tener en cuenta que tardará seis cliclos de reloj en hacerlo.

2.1.6. Modo Extended Stand-by

Funciona del mismo modo que el power-save con la diferencia de que el oscilador sigue trabajando. Al igual que el modo Stand-by tarda 6 ciclos de reloj en despertar.

2.2. Desactivación de periféricos

El ATMega 328 dispone de una variedad de periféricos que contribuyen al correcto funcionamiento del mismo. A primera vista pueden parecer imprescindibles pero teniendo en cuenta las especificaciones particulares de cada diseño, se llega a la conclusión de que determinados periféricos pueden desactivarse sin dañar ningún elemento ni alterar el funcionamiento de la aplicación. Desactivar elementos innecesarios contribuye directamente en una mejora de consumo. Hay numerosos estudios que cuantifican este hecho por lo que se procede a su previo estudio y valoración para en capítulos posteriores validarlos e intentar optimizarlos.

2.2.1. Brown Out Detector

El ATmega 328 tiene incorporado un chip Brown-out Detection (BOD). Se utiliza para monitorizar el nivel de Vcc durante el funcionamiento del procesador. Este valor se compara a uno fijado previamente. Si el valor de Vcc cae por debajo de este valor automáticamente se activa el Brown-out reset.

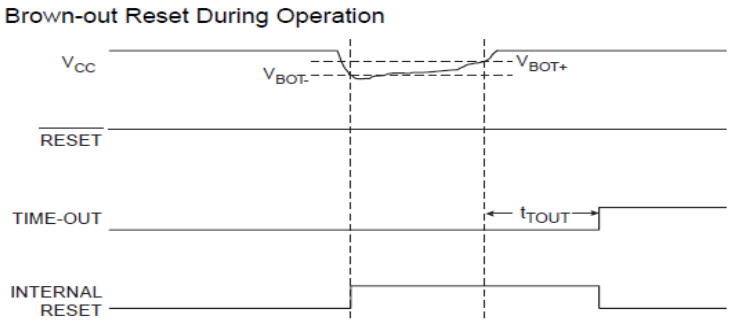


Fig 2.1. Brown-out Detection

Si la aplicación con la que se trabaja no necesita el Brown-out Detector puede desactivarse dicho módulo. Existen varias maneras de desactivar el BOD pero dado que las bibliotecas con las que se trabaja más adelante ofrecen una manera muy sencilla de hacerlo no se considera relevante detallar la cuestión. Atendiendo a las conclusiones de Nick Gammon (ver [21]), el ahorro que se obtiene con este procedimiento es de apenas 25µA. Sin duda es un valor muy pequeño pero que colabora en el descenso consumo. Se debe valorar si para una aplicación en particular un ahorro tan pequeño compensa el hecho de desactivar el BOD.

2.2.2. Analog to Digital Converter (ADC)

El ADC convierte una entrada analogical de voltage en una señal digital de 10 bits a través de sucesivas aproximaciones. El proceso completo no se detalla en este trabajo dado que no es de relevancia para el propósito del mismo. Si el lector desea ampliar información ésta se encuentra a partir de la página 250 del datasheet del fabricante del ATmega 328 (esquema añadido en el Anexo B).

Si este dispositivo se encuentra activo lo estará en todos los modos sleep, por lo que en caso de no ser necesario y con el fin de ahorrar consumo se recomienda su desactivación

Se obtienen valores teóricos nuevamente del trabajo de Nick Gammon (ver [21]) en los que se obtiene un descenso de 335µA. Sabiendo que el consumo de Arduino Uno es teóricamente de unos 40-50 mA no parece una medida importante.

El modo de desactivar el ADC es relativamente sencillo, únicamente hay que añadir la línea de código “ADCSRA=0” en la zona de setup. A pesar de todo no será necesario realizarlo de este modo ya que las bibliotecas con las que se trabaja más adelante lo hacen de un modo todavía más sencillo.

2.2.3. Port Pins

Al entrar en un modo Sleep es necesario configurar los pines para que usen la mínima cantidad de energía. Si algunos pins no se usan se recomienda que tengan un nivel definidos. Es importante que no existan cargas resistivas en los mismos. Hay que tener en cuenta que en ocasiones la entrada lógica será necesaria para despertar de un modo sleep usando interrupciones por lo que deberá estar disponible.

Para configurar un pin lo único que hace falta es definir si es INPUT o OUTPUT y posteriormente el estado, HIGH o LOW. Mediante un sencillo bucle “for” se pueden configurar todos a la vez en un sentido.

```
for (byte i = 0; i <= A5; i++)
    {
    pinMode (i, OUTPUT);    //
    digitalWrite (i, LOW);  //         }
```

Fig 2.2. Definición de pins

Se comparan dos mediciones realizadas en modo Power-down. La primera (ver tabla 2.1.), obtenida del foro wikispace (ver [3]), muestra mejoras del orden de 15mA al definirlos como pines de entrada y estado HIGH. Se debe mencionar que al realizar la medición no había ninguna carga conectada en los pines ya que obviamente aumentaría la corriente consumida.

Tabla 2.2. Mediciones pins

Test 1 with 8.26V in Vin pin:
BLANK SKETCH: **0.445214W** (53.9mA)
Pins 2 - 13 output, HIGH: **0.383264W** (46.4mA)
Pins 2 - 13 output, pins 2 - 12 HIGH: **0.357658W** (43.3mA)
Pins 2 - 13 output, LOW: **0.357658W** (43.3mA)
All pins input, all floating: **0.316358W** (38.3mA)
All pins input, pins 2 - 13 pull-up: **0.261016W** (31.6mA)

Test 2 with 8.22V in Vin pin:
Awake:
BLANK SKETCH: **0.444702W** (54.1mA)
SPI, TWI, USART0 and ADC disabled: **0.359214W** (43.7mA)
Asleep:
Pins 2 - 13 output, HIGH: **0.340308W** (41.4mA)
Pins 2 - 13 output, pins 2 - 12 HIGH: **0.314826W** (38.3mA)
Pins 2 - 13 output, LOW: **0.314826W** (38.3mA)
All pins input, all floating: **0.258108W** (31.4mA)
All pins input, pins 2 - 13 pull-up: **0.25893W** (31.5mA)
All pins input, pins 2 - 12 pull-up: **0.258108W** (31.4mA)

En la siguiente medida, también realizada en modo Power-down, se obtienen valores mucho más modestos de apenas 1µA de diferencia (ver [21]). Ambas mediciones exponen que efectivamente existe diferente consumo de corriente al configurar los pines en un sentido u otro. Desafortunadamente los valores son tan dispares que ofrecen poca fiabilidad. En el capítulo de mediciones se realiza un test para comprobar cuál es el ahorro real, si es que existe, del hecho de modificar los pines.

2.2.4. Power reduction register

El registro Power Reduction (PRR) permite detener el reloj de determinados periféricos con el fin de reducir el consumo de energía. Básicamente se trata de un registro de 8 bits que permite activar o desactivar elementos del procesador [Tabla 2.3.].

Hay que tener en cuenta que el PRR solamente funciona cuando el procesador está activo o en el modo Sleep Idle. En el resto de modos Sleep los módulos ya se encuentran desactivados.

Para desactivar todos los bits del PRR únicamente hay que introducir la línea de código PRR=0xFF. Se realizan pruebas (ver [21]) y se mide un ahorro de 3 mA sobre los 50 mA al desactivar todos los módulos y no trabajar con modos Sleep. Es una herramienta útil a la hora de mejorar el consumo en el modo activo.

Tabla 2.3. PRR

Bit 7	PRTWI: Power Reduction TWI
Bit 6	PRTIM2: Power Reduction Timer/Counter2
Bit 5	PRTIM0: Power Reduction Timer/Counter0
Bit 4	Res: Reserved bit
Bit 3	PRTIM1: Power Reduction Timer/Counter1
Bit 2	PRSPI: Power Reduction Serial Peripheral Interface
Bit 1	PRUSART0: Power Reduction USART0
Bit 0	PRADC: Power Reduction ADC

2.3. Frecuencia de trabajo

La placa de Arduino Uno incorpora un resonador cerámico de 16Mhz. Es una frecuencia de trabajo interesante para la mayoría de aplicaciones. Sin embargo, en proyectos simples o de bajo coste computacional puede ser excesiva. Adaptar la frecuencia de trabajo a las necesidades de la aplicación es un modo de reducir el consumo de la misma.

La gran mayoría de placas con diseño de bajo consumo incorporan cristales de 8Mhz de serie, por lo que es fácil deducir que los diseñadores han llegado a la conclusión de que una frecuencia menor de trabajo repercute directamente en ahorro energético. Una opción sería intercambiar el cristal de la placa por uno de 8Mhz o quizá menos frecuencia. El problema es que se limita el uso de la

placa además de la dificultad de la operación. Afortunadamente es relativamente sencillo modificar dicha frecuencia de trabajo (se obtiene información del foro de Arduino, ver [2]).

La manera de modificar la frecuencia del reloj en un microcontrolador ATMega328, es insertar el siguiente código fuente en el programa allí donde se quiera modificar la frecuencia:

CLKPR = (1<<CLKPCE); *Activa el escalado dinámico.*
CLKPR = B00000011; *Indica la cifra por la que se divide el reloj (Tabla 2.4.).*

Tabla 2.4. Valor división del reloj interno

0000 – 1	0100 - 16
0001 - 2	0101 - 32
0010 - 4	0110 - 64
0011 - 8	0111 - 128

La frecuencia únicamente puede dividirse, no incrementar. Al ralentizar la frecuencia del reloj también se ralentizan las funciones de entrada/salida, delay,etc. Esto hace que si se divide la frecuencia de trabajo por 4 también lo hagan éstas funciones. Puede ser un problema grave. Para que todo funcione correctamente la nueva frecuencia ha de definirse la nueva frecuencia al compilar el código fuente. Se ha de modificar el valor resaltado en negrita (ver Tabla 2.5.) indicando la frecuencia a la que se desea trabajar. En la ruta *C:\Program Files (x86)\Arduino\hardware\arduino* se encuentra el archivo.txt llamado boards. En él se especifican las características de todas las placas Arduino. Accediendo al modelo de placa con el que se trabaja y modificando la información es posible alterar dichas características.

Tabla 2.5. Especificaciones Hardware Arduino Uno

```
uno.name=Arduino Uno
uno.upload.protocol=arduino
uno.upload.maximum_size=32256
uno.upload.speed=115200
uno.bootloader.low_fuses=0xff
uno.bootloader.high_fuses=0xde
uno.bootloader.extended_fuses=0x05
uno.bootloader.path=optiboot
uno.bootloader.file=optiboot_atmega328.hex
uno.bootloader.unlock_bits=0x3F
uno.bootloader.lock_bits=0x0F
uno.build.mcu=atmega328p
```

```
uno.build.f_cpu=16000000L
uno.build.core=arduino
uno.build.variant=standard
```

Para modificar la frecuencia se cambiará el valor remarcado en negrita teniendo en cuenta los factores de división y especificando la velocidad de trabajo deseada. De este modo es fácil y rápido modificar la frecuencia de trabajo. Hay que tener en cuenta que debe guardarse el archivo modificado y cerrar y volver a abrir el IDE de Arduino para que el cambio tenga efecto

2.4. Bibliotecas

Existe una multitud de bibliotecas para trabajar con Arduino. Al ser una plataforma que utiliza el mismo programa para todas las placas se hace muy sencillo utilizar dichas bibliotecas. Para el estudio en el que se centra este trabajo se focaliza sobre aquellas que permitan modos de trabajo en bajo consumo. Básicamente se trata de funciones que permiten acceder de forma sencilla y eficaz a dichos modos de trabajo.

En un origen se había planteado la necesidad de realizar una biblioteca propia, pero al indagar y observar la variedad de bibliotecas existentes para tal fin se consideró más práctico y eficaz realizar una comparativa de al menos tres de estas bibliotecas para valorar cuál era más eficiente y adaptable a las necesidades de los futuros diseñadores. Por supuesto existe la posibilidad de combinar varias de ellas para optimizar todavía más los resultados.

Utilizar una biblioteca es sencillo. Debe descargarse el archivo desde la página del programador. Normalmente se trata de una carpeta que hay que añadir a la carpeta *Libraries* que se encuentra en la ruta `C:\Program Files (x86)\Arduino\libraries`. El siguiente paso es incluir la cabecera en el código que se programa y listo.

En este capítulo se describen tres bibliotecas junto a las funciones que desarrollan. La función de estas bibliotecas es optimizar el consumo del microprocesador. En el capítulo posterior se realiza una comparativa práctica de ellas y se razonan conclusiones respecto a la mismas.

2.4.1. Biblioteca Jeelib

La biblioteca Jeelib ha sido programada por el equipo de Jeelabs. Se trata de una colección de cabeceras, clases y programas sencillos para funcionar con Arduino IDE.

Jeelib tiene disponibles varias clases como Port, PortI2C o MilliTimer pero la que es interesante para el objetivo de este trabajo es la clase Sleepy. Se trata de una clase que permite codificar un ATmega en un modo bajo consumo y que se detalla a continuación.

Jeelib es un código que permite trabajar en modo bajo consumo utilizando el Watchdog Timer (WDT). Para funcionar correctamente requiere un vector para manejar una interrupción del WDT. Está formada por las siguientes funciones:

- Byte Sleepy::loseSomeTime (Word msecs)

Sirve para estar un tiempo (definido en milisegundos) en low power mode. Para usar esta función se debe incluir una definición de la interrupción del WDT. La forma más fácil de hacerlo es mediante la línea “ISR(WDT_vect) {Sleepy::watchdogEvent ();}”

Es una de las funciones más socorridas a la programar ya que sustituye al típico “delay”. Permite poner Arduino en un modo Sleep Power-down por un determinado espacio de tiempo.

- Void Sleepy::powerDown()

Esta función permite entrar en modo Sleep Power-down. Se despierta vía WDT, INT0/1 o un cambio de estado de pin. Además desactiva el Brown Out Detector, el ADC, y otros periféricos como el TWI, SPI y UART antes de pasar al modo sleep. Una vez se restablece el modo de trabajo sus estados vuelven a recuperarse.

Es una función muy útil para modos bajo consumo ya que permite acceder con comodidad al modo más agresivo de sleep a la vez que desactiva periféricos que de otro modo habría que desactivar manualmente.

2.4.2. Biblioteca N0m1

Esta biblioteca de Github permite programar Arduino en cualquier modo Sleep durante un período determinado de tiempo o hasta que una interrupción lo despierte. Resulta muy útil ya que incorpora multitud de ejemplos de programación y la posibilidad de poder trabajar con todos los modos sleep. Como contrapunto no incorpora la opción de habilitar o deshabilitar periféricos. Se realiza una descripción de las funciones que incorpora.

- Función: idleMode

Pone el Arduino en modo sleep Idle. Se detiene el MCU pero los timers y periféricos siguen trabajando.

- Función: adcMode

Pone el Arduino en modo sleep ADC. Se detiene la CPU mientras siguen funcionando el ADC, interrupciones externas, Timer/Counter2 y el WDT.

- Función: pwrSaveMode

Pone el Arduino en modo sleep PowerSave. El timer del cristal sigue funcionando junto el Timer2.

- Función: extStanbyMode

Pone el Arduino en modo sleep extStanby. Una function identical a la pwrSaveMode con la excepción de que el oscilador sigue trabajando para poder despertar más rápido.

- Función: standbyMode

Pone el Arduino en modo sleep standby, modo de trabajo idéntico al de la función pwrDownMode con la excepción de que el oscilador sigue trabajando para poder despertar más rápido.

- Función: pwrDownMode

Pone el Arduino en modo sleep power Down Mode. Todos los sistemas desactivados except el WDT y el reset externo. Extremadamente útil para modos de trabajo de bajo consumo.

- Función: sleepDelay

Imita a la función de Arduino “Delay” poniendo el arduino en un modo sleep durante un período determinado de tiempo. El tiempo se especifica en ms.

- Función: sleepInterrupt

Pone al Arduino en un modo sleep hasta que una interrupción se activa. Se basa en dos valores, el valor que devuelve la interrupción (0,1,etc) y el modo del trigger (LOW, RISING, FALLING, CHANGE).

Las figuras 2.5 y 2.6 muestran dos ejemplos de códigos sencillos en los que usa dicha biblioteca para facilitar la comprensión en el manejo de la misma.

```
#include <Sleep_n0m1.h>

Sleep sleep;
unsigned long sleepTime; // cuanto tiempo pasara el Arduino en modo sleep

void setup()
{
  Serial.begin(9600);
  sleepTime = 50000; // define el valor de sleepTime en ms, tiempo máximo 49,7 días
}

void loop()
{
 delay(100); // delay para dar tiempo a escribir al serial
```

```
  Serial.println("pon aquí tu código");

  Serial.print("durmiendo durante");
  Serial.println(sleepTime);
  delay(100); // delay para dar tiempo a escribir al serial

  sleep.pwrDownMode(); // se escoge modo sleep
  sleep.sleepDelay(sleepTime); // duerme en el modo escogido durante el tiempo predeterminado
}
```

Fig 2.3. Código con la biblioteca N0m1 y ejemplo de la función delay.

```
#include <Sleep_n0m1.h>

Sleep sleep;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
  delay(100); //tiempo para escribir via serial
  Serial.println("ejecuta aquí tu código");
  Serial.print("durmiendo hasta interrupción");
  delay(100); //tiempo para escribir via serial
  sleep.pwrDownMode(); //escoge sleep mode
 sleep.sleepInterrupt(0,FALLING); //duerme hasta interrupción
}
```

Fig. 2.4. Código ejemplo de la biblioteca N0m1 y la función para gestionar interrupciones.

2.4.3. Biblioteca Low-power

Low-power es una biblioteca desarrollada por Rocketscream con el fin de trabajar con Arduino en modos de bajo consumo. Rocketscream es una pequeña empresa ubicada en Malasia que se dedica a comercializar elementos

de electrónica a la par que diseña y fabrica dispositivos de bajo consumo. En uno de los muchos Tags que contiene su web se encuentra el enlace a la biblioteca Low-power.

Low-power es una biblioteca que permite definir una gran cantidad de variables en una sola instrucción permitiendo una programación ágil y fácil de interpretar. Sus funciones permiten escoger el modo sleep que se quiere para el procesador y a su vez que elementos permancen activos o no en el mismo.

En la figura 2.5. se mencionan las funciones de la biblioteca. Cuando se ejecuta una función se selecciona el modo sleep al que pasa el microprocesador y se escoge que elementos de los nombrados en la función permanecen activos o no. Period_t puede definirse con las variables SLEEP_15MS, SLEEP_30MS, SLEEP_60MS, SLEEP_120MS, SLEEP_250MS, SLEEP_500MS, SLEEP_1S, SLEEP_2S, SLEEP_4S, SLEEP_8S, SLEEP_FOREVER. El resto de variables se definen en estado ON-OFF, lo que permite activar y desactivar periféricos con mucha facilidad.

```
void idle(period_t period, adc_t adc, timer2_t timer2,
timer1_ttimer1,timer0_ttimer0,spi_tspi,usart0_tusart0,twi_ttwi);
void powerDown(period_t period, adc_t adc, bod_t bod);
void powerSave(period_t period, adc_t adc, bod_t bod, timer2_t timer2);
void powerStandby(period_t period, adc_t adc, bod_t bod);
void powerExtStandby(period_t period, adc_t adc, bod_t bod, timer2_t timer2);
```

Fig 2.5. Funciones biblioteca Low-power

Para facilitar la comprensión del uso de la biblioteca la figuras 2.6. y 2.7 muestran un ejemplo de un código sencillo haciendo uso de la misma.

```
#include "LowPower.h"

void setup()
{
    // La biblioteca no necesita setup
}

void loop()
{
    // Entra en modo powerdown durante 8 segundos. Desactiva ADC y BOD
```

```
    LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
    // Realiza una acción. Lee un sensor por ejemplo.
}
```

Fig 2.6. Ejemplo de código usando biblioteca Low-power

```
#include "LowPower.h"
// Se define el pin 2 como pin wake up
const int wakeUpPin = 2;

void wakeUp()
{
    // Handler para la interrupción del pin
}

void setup()
{
    // Configura el pin wake up como input
    pinMode(wakeUpPin, INPUT);
}

void loop()
{
    // La interrupción salta cuando el pin wake up es LOW
    attachInterrupt(0, wakeUp, LOW);
    // Entra en modo powerdown con ADC y BOD desactivados
    // Despierta cuando el pin wake up es LOW
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
    // Desactiva interrupción.
    detachInterrupt(0);
    // Realiza cualquier acción. Leer sensor, transmitir, etc.
}
```

Fig. 2.7. Ejemplo de código con biblioteca Low-power y uso de interrupciones.

CAPÍTULO 3. MEDICIONES Y VALIDACIÓN

El objetivo de este capítulo es realizar las medidas necesarias para evaluar el consumo de Arduino y comprobar cómo afectan las soluciones planteadas en los capítulos anteriores.

3.1. Pruebas sobre Arduino Uno

Se realizan dos tipos de mediciones. En la medición de los valores iniciales, es decir, el consumo de Arduino Uno por el simple hecho de estar encendido, se utiliza un multímetro digital standard. Al tratarse de una medición estática (no hay cambios de estado ni fluctuaciones) resulta más sencillo y cómodo. En las medidas dinámicas el consumo de corriente se ha realizado midiendo la tensión sobre una resistencia de control colocada en serie con el Arduino. Se realiza sobre una impedancia flotante de 32.9Ω nominales calculada en función de la corriente esperada. Se usa un osciloscopio portatil.

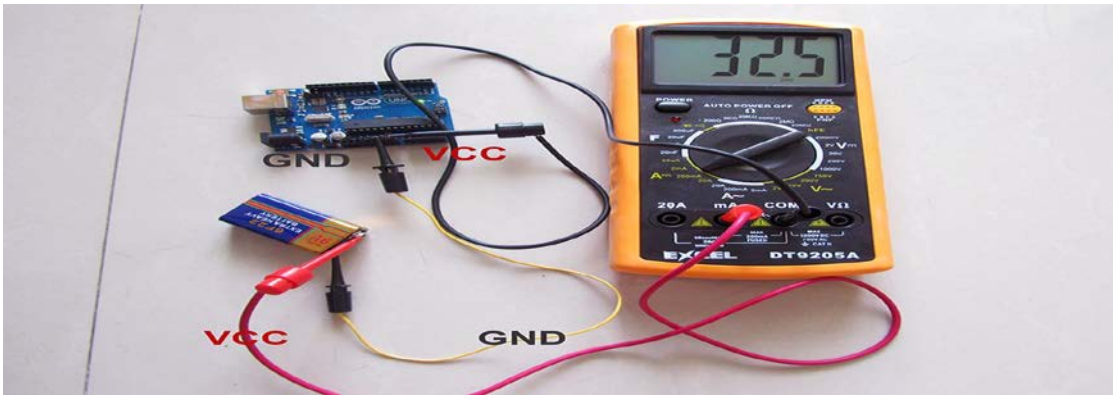


Fig. 3.1. Medida consumo de corriente.

Se utiliza la aplicación “Blink” para realizar el estudio. Es una aplicación sencilla que proporciona el IDE de Arduino. Se encuentra en el apartado “Ejemplos-basic”.La aplicación ilumina un led de manera parpadeante. Se trata de una aplicación de muy bajo coste computacional lo que permite realizar tests a bajas frecuencias. Además, puede iluminar o no el led de modo que también revela el consumo de éste y al requerir tan pocos recursos permite experimentar con los valores de la entrada de alimentación. Por último, al tener una frecuencia de parpadeo de un segundo, hace que la referencia visual de iluminación del led sea muy clara. Esto permite valorar con facilidad si los ajustes en los cambios de frecuencia de trabajo son correctos. El código resulta además sencillo de manejar con las bibliotecas y permite introducir variables con facilidad (Fig 3.2).

Fig.3.2. Código aplicación “Blink”

3.1.1. Valores iniciales

El primer paso en la valoración de consumo de la placa Arduino Uno es valorar qué consumo ofrece por ella misma. Eso significa ponerla a funcionar sin mas, con el código más simple (Fig 3.3.) que existe y valorar los resultados.

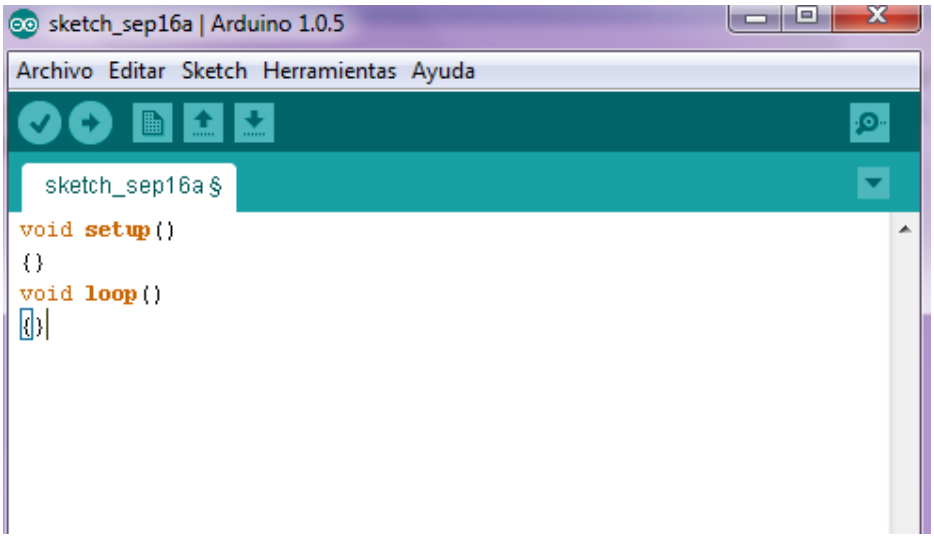


Fig 3.3. Código valor inicial

Se mide con el oscilador externo trabajando a distintas frecuencias y con tensiones de alimentación desde los 5V hasta los 12V que son los márgenes estipulados por el fabricante. Se observa que la placa está operativa a partir de 4.8V pero se decide realizar las comprobaciones a 5V ya que es el valor definido para el óptimo funcionamiento de la placa.

Tabla 3.1. Consumo de Arduino Uno en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	13,3	12,1	12,1	13,3	14,5	15,7
6v	20,5	20,5	21,7	21,7	22,9	24,2
7v	29,1	29,1	29,1	30,2	31,4	33,2
8v	38,1	35,1	38,6	39,9	41,1	42,3
9v	39,8	38,7	39,8	41,1	44,7	48,3
10v	39,8	39,8	39,8	42,3	44,7	48,3
11v	39,8	39,8	39,8	42,3	44,7	48,3
12v	39,8	39,8	39,8	42,3	44,7	48,3

La Tabla 3.1 y la Figura. 3.3. muestran los valores obtenidos en la medición del consumo de la placa en mA. Se mide un valor de 34mA para una alimentación de 7V y trabajando a 16Mhz. Éstos serían los valores standard de trabajo de Arduino Uno ya que es la alimentación y frecuencia que recomienda Arduino. Si se revisan los cálculos anteriores se comprueba que sigue siendo un consumo muy elevado para dispositivos autónomos que deban trabajar con baterías por lo que el estudio de este TFC queda justificado.

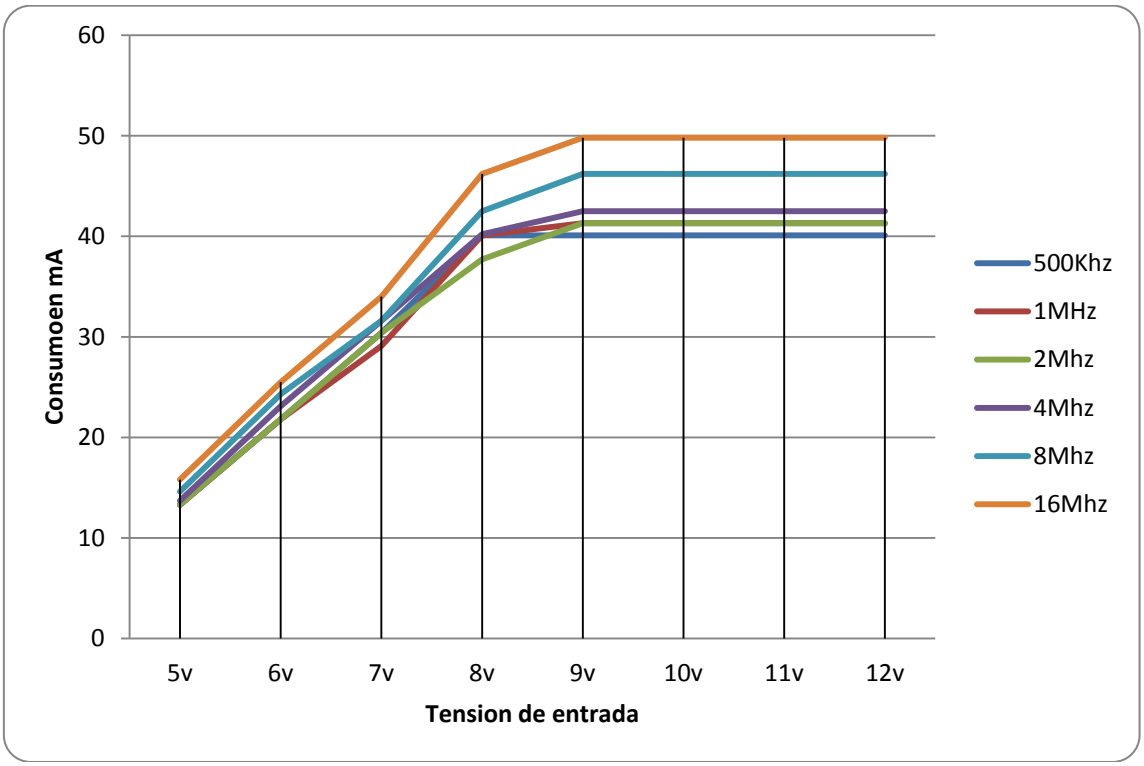


Fig.3.4. Gráfica de la variación de consumo de Arduino Uno

Observando la Figura 3.5. se obtiene una relación prácticamente lineal entre la tensión de entrada y el consumo de la placa que se mantiene al trabajar con distintas frecuencias. Únicamente cuando se trabaja a 1MHz se rompe esta relación. A los 9V de entrada se detiene el aumento de consumo en todas las frecuencias. Se observa un descenso importante al pasar de trabajar con los 7V recomendados a los 5V de alimentación mínima por lo que la tensión de entrada afecta directamente al consumo que tendrá la placa. Obviamente si se quiere un consumo bajo se recomienda alimentar con la mínima tensión posible.

Al observar cómo afecta la reducción de frecuencia de trabajo, puede verse como a mayor tensión de entrada mayor es el ahorro energético, mientras que a tensiones menores la diferencia es menor.

Con estos datos el diseñador puede valorar y escoger que configuración se adapta mejor a sus necesidades y proyectar en función a unos consumos dados para una tensión de entrada y frecuencia de trabajo establecidos.

En las siguientes líneas se muestra el estudio de consumo para los distintos modos Sleep. De esta manera puede observarse directamente de qué modo afecta al consumo la configuración de cada uno de ellos.

Para modificar el modo de trabajo hay que incluir la siguiente expresión al código sobre el que se trabaja en el apartado “set up”:

```
set_sleep_mode (SLEEP_MODE_PWR_DOWN);//Aquí se escoge el
modo sleep
sleep_enable();
sleep_cpu ();
```

Tabla 3.2. Consumo en Modo Idle en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	13,3	13,3	13,3	13,3	14,5	15,7
6v	19,5	21,8	21,8	21,8	24,3	25,5
7v	30,3	30,3	30,3	31,6	31,6	34
8v	38,9	38,9	40,1	41,3	43,7	46,2
9v	41,3	41,3	42,5	42,5	46,2	49,8
10v	41,3	41,3	42,5	42,5	46,2	51
11v	41,3	41,3	42,5	42,5	46,2	51
12v	41,3	41,3	42,5	42,5	46,2	51

Al observar la Figura 3.5 queda manifiesto que el modo IDLE es efectivamente el menos restrictivo en cuanto la desactivación de elementos por lo que como cabía esperar el consumo se mantiene prácticamente idéntico a un funcionamiento normal. En este caso la relación lineal de consumo respecto a

voltaje de entrada sí que se mantiene para todas las frecuencias. A pesar de que se trata de un modo Sleep no se cuantifica ningún descenso en el consumo. Por los datos recogidos no es un modo recomendable para intentar optimizar el consumo.

Tabla 3.3. Consumo en modo ADC en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	13,3	13,3	13,3	13,3	12,1	13,3
6v	21,8	21,8	21,8	21,8	21,8	21,8
7v	30,3	30,3	29,1	29,1	29,1	29,1
8v	40,1	40,1	40,1	40,1	38,9	40,1
9v	40,1	40,1	40,1	40,1	41,3	41,3
10v	40,1	40,1	40,1	40,1	41,3	41,3
11v	40,1	40,1	40,1	40,1	41,3	41,3
12v	40,1	40,1	40,1	40,1	41,3	41,3

En la Tabla 3.3. y la Figura 3.5. se muestra el consumo de la placa trabajando en modo ADC. Se observa la relación lineal entre tensión de entrada y consumo por lo que vuelve a ser recomendable alimentar a bajas tensiones para obtener menor consumo energético. No refleja un ahorro de consumo al trabajar a frecuencias menores. Se mantiene prácticamente idéntico en las 6 frecuencias experimentadas por lo que se recomienda una frecuencia elevada que permita mayor rango de trabajo. Comparando la alimentación a 7 V y 16 Hz se confirma un descenso de consumo de 5 mA respecto el modo normal mientras que el consumo mínimo para los 5V de Vin vuelve a mantenerse en 13 mA.

Tabla 3.4. Tabla consumo en modo PWR_SAVE en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	12,1	12,1	12,1	12,1	13,3	13,3
6v	21,8	21,8	21,8	21,8	21,8	21,8
7v	27,9	27,9	27,9	27,9	26,7	27,9
8v	34	34	34	34	34	34
9v	34	34	34	34	34	34
10v	34	34	34	34	34	34
11v	34	34	34	34	34	34
12v	34	34	34	34	34	34

Para el modo Pwr_save se mantiene una relación lineal (Tabla. 3.4 y Figura 3.5) en la que el valor de saturación a partir de los 8V es idéntico para todas las

frecuencias. Vuelve a no apreciarse diferencia al variar la frecuencia de trabajo y el ahorro para los 7 V y los 16 MHz es de 6 mA respecto al modo operativo normal. El menor consumo que se aprecia vuelve a ser de 12 mA. De momento los modos de trabajo Sleep no han conseguido un valor por debajo de esa medida

Tabla 3.5. Tabla consumo en modo EXT_STANBY en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	12,1	12,1	12,1	12,1	13,3	13,3
6v	21,8	21,8	21,8	21,8	21,8	21,8
7v	27,9	27,9	27,9	27,9	26,7	27,9
8v	34,0	34,0	34,0	34,0	34,0	34,0
9v	34,0	34,0	34,0	34,0,	34,0	34,0
10v	34,0	34,0	34,0	34,0	34,0	34,0
11v	34,0	34,0	34,0	34,0,	34,0	34,0
12v	34,0	34,0	34,0	34,0	34,0	34,0

El modo trabajo Ext_Stanby también refleja un valor de estabilización de 34 mA (ver Tabla 3.5. y Figura 3.5.) y relación casi lineal entre el consumo y la tensión de alimentación. No se aprecia ninguna diferencia de consumo al modificar la frecuencia de trabajo y el valor a 7 V y 16 MHz es de nuevo de 27.9 mA. Los valores obtenidos son prácticamente idénticos a los obtenidos en el modo PWR_SAVE, algo muy razonable ya que se trata del mismo modo de trabajo con la única diferencia de que el oscilador sigue trabajando.

Tabla 3.6. Consumo en modo STANDBY en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	13,3	12,1	12,1	12,1	12,1	13,3
6v	21,8	20,6	21,8	21,8	20,6	21,8
7v	27,9	27,9	27,9	27,9	27,9	27,9
8v	32,8	32,8	32,8	32,8	32,8	32,8
9v	32,8	32,8	32,8	32,8	32,8	32,8
10v	32,8	32,8	32,8	32,8	32,8	32,8
11v	32,8	32,8	32,8	32,8	32,8	32,8
12v	32,8	32,8	32,8	32,8	32,8	32,8

El modo Standby muestra un consumo menor de saturación que el resto de modos (ver Figura 3.5. y Tabla 3.6.). La relación entre tensión y consumo

vuelve a ser lineal y se siguen manteniendo los 27.9 mA para 7 Vin y 16 MHz. Ofrece unos valores muy semejantes a los del modo EXT_STANBY. A este modo de trabajo tampoco le afecta decisivamente la variación de frecuencia aunque a 8 MHz el consumo es algo menor por lo que para este modo de trabajo se recomienda trabajar a 8 MHz.

Tabla. 3.7. Consumo en modo PWR_DOWN en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	13,3	13,3	12,1	13,3	12,1	13,3
6v	20,6	20,6	20,6	20,6	20,6	20,6
7v	27,9	27,9	27,9	27,9	27,9	26,7
8v	32,8	32,8	32,8	32,8	32,8	32,8
9v	32,8	32,8	32,8	32,8	32,8	32,8
10v	32,8	32,8	32,8	32,8	32,8	32,8
11v	32,8	32,8	32,8	32,8	32,8	32,8
12v	32,8	32,8	32,8	32,8	32,8	32,8

Este modo Sleep es teóricamente el que menor consumo debe mostrar (Tabla 3.7 y Figura 3.5.). Los valores son muy semejantes al modo STANDBY, algo totalmente coherente ya que es el mismo modo pero sin oscilador. Curiosamente la mejor frecuencia de trabajo son los 16 MHz ofreciendo 16,7 mA para los 7 Vin. De todos los modos medidos es que el menor consumo muestra por lo que tal y como se esperaba será el modo de trabajo escogido para tratar de operar en modos de bajo consumo.

El siguiente paso consiste en realizar una desactivación progresiva de periféricos desde el modo de trabajo Sleep a 8 MHz y 5 Vin (se escoge esta frecuencia y tensión por ser las que menos consumo ofrecen). De esta manera se intenta conseguir el mínimo consumo a la que la placa Arduino Uno estará operativa. Estas medidas se llevan a cabo con un multimetro digital.

El primer paso consiste en desactivar el Brown Out Detection. Al realizar dicha operación no se cuantifica ninguna variación en el consumo. Teniendo en cuenta que el valor esperado es de 25 µA menos no representa un valor relevante.Posteriormente se desactiva el ADC y se nuevo no se visualiza ninguna variación. Para este caso se espera un descenso de 335 µA pero no se aprecia ningún tipo de descenso.

El siguiente paso consiste en desactivar el PRR y aquí sí que se mejora el consumo en 1,25 mA lo que representa un 10% del consumo de la placa. Este dato refleja la importancia de desactivar el PRR siempre y cuando no sea necesario en nuestro diseño.

Por último se programan los pins de la placa en modo salida/entrada y en estados LOW/HIGH. En ninguna de las 4 variaciones se aprecian diferentes

consumos a pesar de que hay estudios que si que los reflejan. De todos modos se recomienda configurarlos como salidas y en estado HIGH.

Finalmente en el mejor de los casos se obtiene un consumo de 9.7 mA teniendo a la placa en el modo Sleep más profundo sin hacer nada y sin ningún dispositivo acoplado (antenas, sensores, motores, etc). Si se tiene en cuenta que la corriente aportada por un pack de pilas AA será de 2890 mA (Tabla 1.3.) el consumo estimado (no se tiene en cuenta el factor de descarga de las propias pilas) será del orden de 12 días siempre y cuando la placa no haga nada. No parece ni mucho menos una solución atractiva para ningún proyecto que deba funcionar de manera autónoma.

En el siguiente capítulo se realizan medidas o una aplicación simple que ilumina un led con el fin de valorar el consumo de la placa operando y comprobar la eficacia de las bibliotecas anteriormente expuestas.

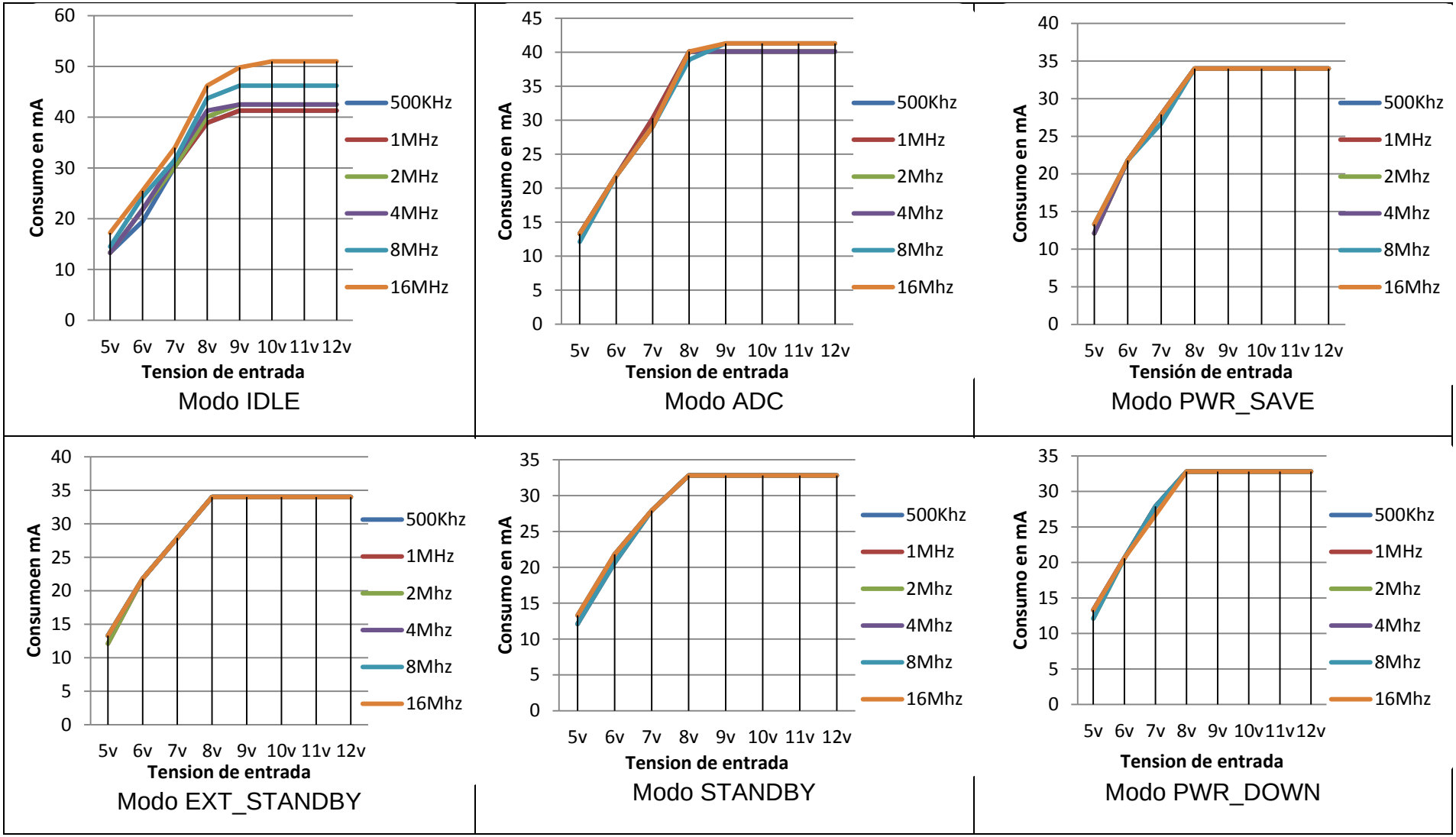


Fig.3.5. Comparativa de consumos de los modos Sleep Arduino Uno

3.1.2. Valores en aplicación

Se realiza una medición sobre la aplicación Blink mencionada anteriormente. Esta aplicación ilumina un led durante un segundo, se apaga otro segundo y vuelve a iluminarse. Solamente se trata de un parpadeo que nos permitirá evaluar el consumo de la placa trabajando.

Para la evaluación de la aplicación se utiliza el led que viene incorporado en la placa en el pin numero 13 de modo que por un lado se ahorrará el consumo de un led externo y por otro se evalúa el coste del mismo.

Los valores se mantuvieron estables a partir de los 10 V de alimentación por lo que el estudio se realizó entre los 5 V y 10 V de alimentación. Los valores reflejados en las tablas son los valores medidos medios cuando el led está encendido y apagado.

Tabla 3.8. Consumo de la aplicación “Blink” en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	13,3	13,3	13,3	13,7	14,6	15,8
6v	21,8	21,8	21,8	23,1	24,3	25,5
7v	30,4	29,1	30,4	31,6	31,6	34,0
8v	40,1	40,1	37,7	40,2	42,5	46,2
9v	40,1	41,3	41,3	42,5	46,2	49,8
10v	40,1	41,3	41,3	42,5	46,2	49,8

Lo primero que se observa (ver Tabla 3.8) es que comparando el consumo de 7V a 16MHz se obtiene un valor prácticamente idéntico al valor original de trabajo sin ninguna aplicación. La relación entre Vin y consumo vuelve a ser lineal y trabajando a 1MHz se aprecia una mejora del orden de 10mA en tensiones elevadas, lo que supone un ahorro a tener en cuenta. Cabe mencionar que el ahorro obtenido si se compara la alimentación a 5 V y a 10 V ronda los 25-30 mA. Siempre y cuando la luminiscencia del led no deba ser elevada (a menor tensión de entrada menor iluminación del led) se recomienda trabajar a frecuencias bajas y una alimentación reducida consiguiendo un ahorro considerable de consumo (ver Figura 3.6.).

Cuando se han realizado modificaciones en el BOD, ADC, PRR y pines de entrada no se detectan alteraciones en el consumo. Es probable que al tratarse de medias fluctuantes y variaciones tan pequeñas no se hayan podido apreciar correctamente.

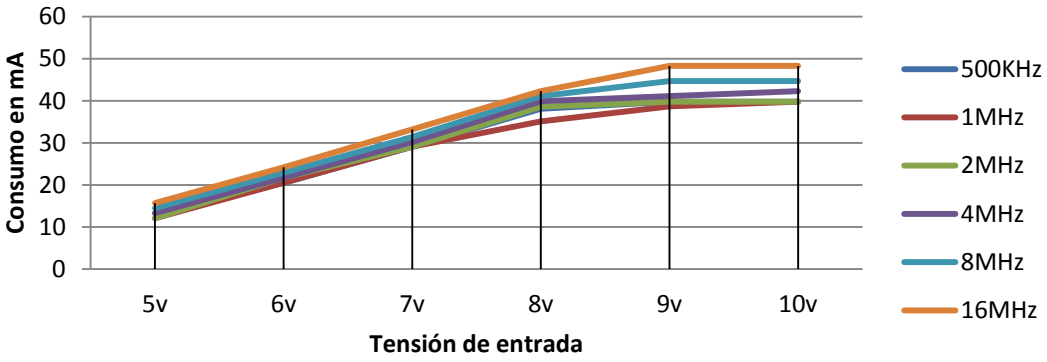


Fig. 3.6. Gráfica consumo aplicación “Blink”

3.1.3. Valores usando bibliotecas

Las siguientes medidas muestran los consumos obtenidos al utilizar las bibliotecas (ver capítulo 2.4) para optimizar el consumo de la placa Arduino Uno funcionando con la aplicación “Blink” mencionada anteriormente. Dicha aplicación utiliza la función “delay” para esperar un segundo entre parpadeos. Es una función muy recurrente ya que permite utilizar tiempos de espera de manera muy fácil. El problema es que durante esa espera no se utiliza ninguna configuración para optimizar recursos. Las bibliotecas presentadas se encargan de optimizar dichos recursos durante esas fases de espera. Esto es extremadamente útil puesto que permite una programación sencilla a la hora de utilizar los modos sleep. Además también ofrecen la posibilidad de desactivar periféricos sin tener que introducir líneas de código.

Se testean en un rango de alimentación de 5V a 10V de entrada y en varias frecuencias de trabajo. El fin de dicha comparativa es valorar si efectivamente consiguen reducir el consumo de la placa y si en caso de hacerlo lo hacen del mismo modo como se espera en un primer momento.

El modo de cómo cargar bibliotecas está reflejado en la página web de Arduino (ver [1]). Se trata de un proceso sencillo y rápido.

La primera biblioteca en comprobar es “Jeelib” (ver cap 2.4.1). Se obtiene una mejora de 10mA trabajando a 7V y 16Mhz. Curiosamente no se observa un descenso de consumo conforme desciende la frecuencia si no todo lo contrario, éste aumenta excepto al trabajar a 2MHz y alimentar a 8V donde se logra un consumo por debajo de la media. Se concluye que para trabajar con esta biblioteca no es necesario modificar la frecuencia de trabajo logrando así poder disponer de toda la potencia de procesado y a la vez un ahorro de energía a tener en cuenta. De nuevo el hecho de modificar periféricos no refleja mejoras de consumo. En algunos puntos los valores son inferiores a los medidos al poner la placa exclusivamente en modo PWR_DOWN por lo que la biblioteca optimiza recursos de la placa otorgando valores muy interesantes.

Tabla 3.9. Consumo de la biblioteca “Jeelib” en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	12	12,08	10,9	11,4	10,2	10,2
6v	19,9	19,9	19	19,3	18,7	17,5
7v	26,3	26	25,4	25,4	24,7	23,5
8v	32,6	32,6	27,8	31,4	31,4	29,6
9v	32,6	32,6	31,4	31,4	31,4	31,4
10v	32,6	32,6	31,4	31,4	31,4	31,4

Tabla 3.10. Consumo de la biblioteca “N0m1” en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	11,5	11,5	11,5	11,5	12,7	13,3
6v	18,7	18,7	19,9	19,9	21,1	22,1
7v	27,8	27,8	27,8	26,6	28,3	29,6
8v	35	35	36,2	36,2	37,5	38,6
9v	37,5	37,5	37,4	38	37,5	38,6
10v	37,5	37,5	37,4	38	37,5	38,6

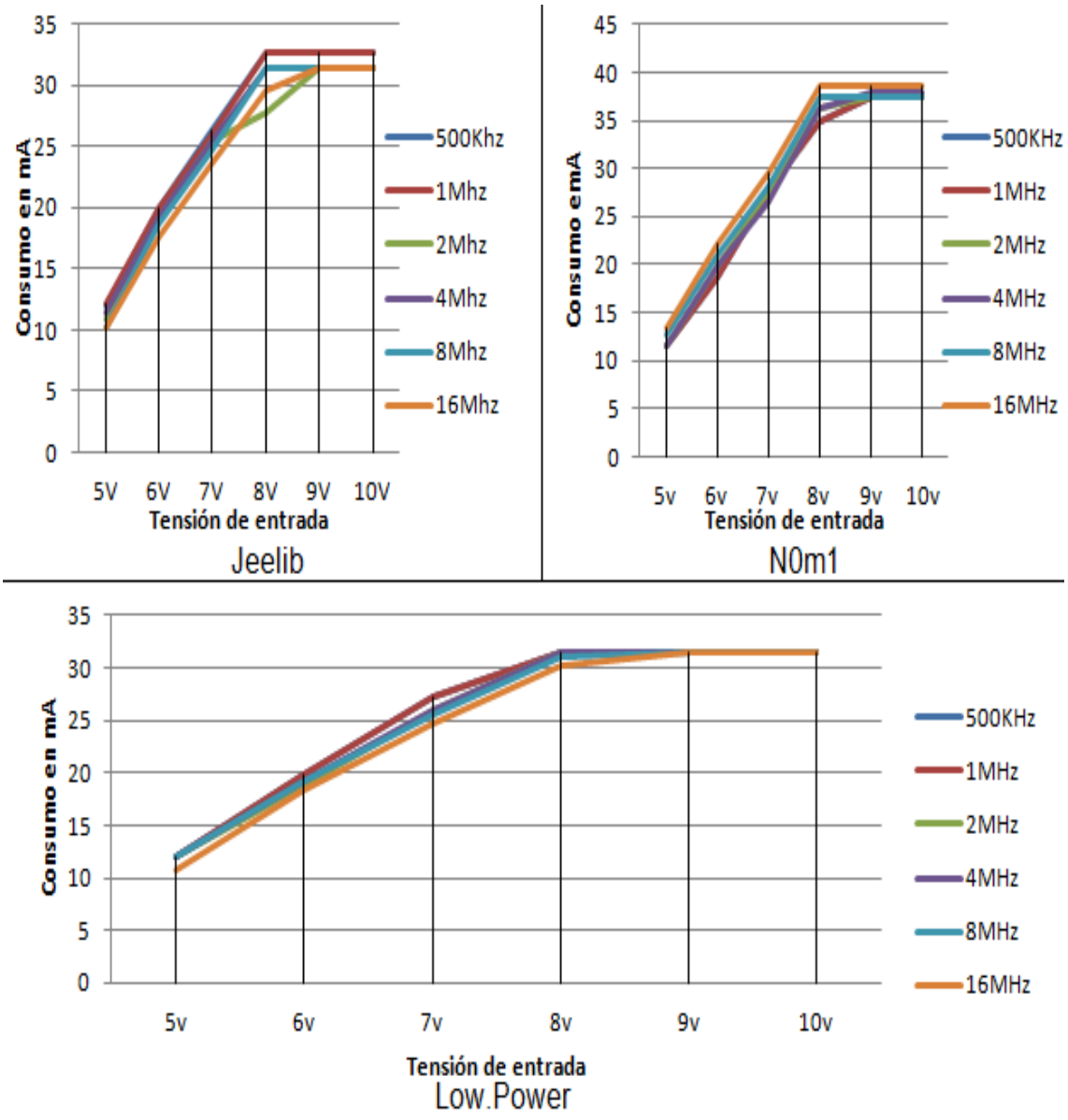
La biblioteca “N0m1” ofrece valores algo más elevados que la de “Jeelib”. Se ha escogido el modo sleep PWR_DOWN dado que es el más restrictivo y el mayor ahorro ofrece. Esta vez sí que se aprecia un menor consumo trabajando a frecuencias menores (ver fig.3.7.) y de nuevo una relación lineal hasta los 9V donde los valores se mantienen. La mejora a 7Vin y 16 MHz es de 5mA y el consumo mínimo obtenido refleja 11,5mA (ver tabla 3.10). No se observan modificaciones al desactivar periféricos.

La última biblioteca es Low.Power. De las tres es la que más opciones ofrece a la hora de configurar la placa por lo que a priori es la más interesante.
Se escoge de nuevo el modo Sleep PWR_DOWN y se desactiva el BOD y el ADC. El PRR se desactivó sin mostrar ninguna diferencia de consumo. Para 7Vin y 16Mhz se miden 24,7mA (ver tabla 3.11) lo que supone un descenso respecto los 34mA iniciales. Ofrece una relación lineal y esta vez los valores se estabilizan a 9V de entrada respecto a los 10V de las anteriores. El valor máximo obtenido es de 31.4mA, idéntico a “Jeelib”. Dado que los valores son muy parecidos a “Jellib”, menores a “N0m1” y que permite trabajar con los periféricos y escoger los modos Sleep con facilidad se recomienda esta biblioteca como biblioteca de trabajo en modo Low power para futuras aplicaciones.

Tabla 3.11. Consumo de la biblioteca “Low.power” en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	12,1	12,1	12,1	12,1	12,1	10,8
6v	19,9	19,9	18,7	19,3	19,3	18,4
7v	27,2	27,2	25,9	25,9	25,6	24,7
8v	31,4	31,4	31,4	31,4	31,1	30,2
9v	31,4	31,4	31,4	31,4	31,4	31,4
10v	31,4	31,4	31,4	31,4	31,4	31,4

Fig.3.7. Comparativa de consumos bibliotecas



3.2. Pruebas sobre Mini Ultra 8MHz

A continuación se realizan las medidas sobre la placa Mini Ultra 8MHz. La placa se compró en rocketscream (ver [17]) al precio de 14 dólares que al cambio son aproximadamente 10.5 euros. A la hora de utilizar la placa hay que tener unas consideraciones básicas.

La primera es que tiene una cabecera FTDI de 6 pines. No incorpora una entrada USB para cargar los programas por lo que es necesario un adaptador FTDI con entrada USB para cargar los programas (ver fig.3.7.). Puede conseguirse fácilmente en cualquier distribuidor de componentes electrónicos

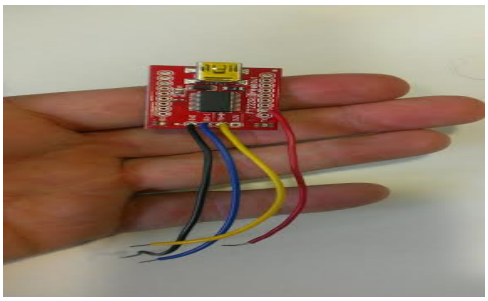


Fig 3.8. Adaptador FTDI con entrada mini USB

Para poder trabajar cómodamente con las placas, se opta por soldar cables en los pines DTR (Data Terminal Ready), GND, TX y RX del adaptador FTDI y a la vez en los mismo pines más los que corresponden al Vin y GND de la placa Mini Ultra 8MHz. Una vez Hecho esto lo único que hay que hacer es conectar ambos DTR y GND, el pin TX de la placa FTDI con el RX del Mini Ultra 8Mhz y el RX de la placa FTDI con el TX de la placa Mini Ultra 8MHz. Sólo que alimentar la placa Mini Ultra a 3.3V y conectarla a tierra (Ver Fig.3.8.)

Usando el Arduino IDE hay que seleccionar la tarjeta “Arduino pro o pro mini (3.3, 8MHz) w/ ATmega328” en la opción Herramientas/Tarjetas. El siguiente paso es cargar los programas de la misma manera que con cualquier otra placa de Arduino.

Hay que tener presente que la placa Mini Ultra 8MHz tiene un rango de alimentación de 3.3 a 6V, por lo que hay que ser cuidadoso con la fuente de alimentación y aplicar la tensión adecuada en todo momento.

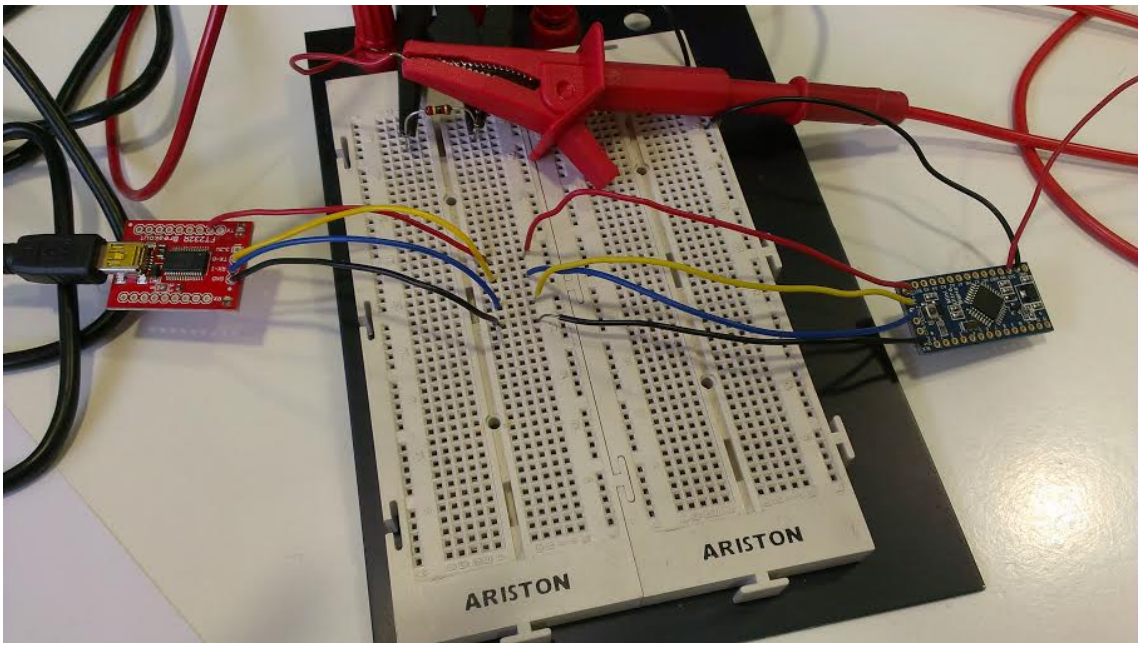


Fig. 3.9. Conexión de la placa FTDI y el Mini Ultra 8MHz

A continuación se investiga el modo de alterar la frecuencia de trabajo de la placa (ver cap. 3.3). En la ruta `C:\Program Files (x86)\Arduino\hardware\arduino` aparece el archivo boards.txt. Ahí es donde se definen las características de las tarjetas con las que se trabaja. Sabiendo que trabajamos con la Minipro 3.3V 8MHz y ATmega328 podemos acceder a la misma y modificar la frecuencia de trabajo.

Tabla 3.12. Especificaciones de Hardware Mini Ultra 8MHz

```
pro328.name=Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
pro328.upload.protocol=arduino
pro328.upload.maximum_size=30720
pro328.upload.speed=57600
pro328.bootloader.low_fuses=0xFF
pro328.bootloader.high_fuses=0xDA
pro328.bootloader.extended_fuses=0x05
pro328.bootloader.path=atmega
pro328.bootloader.file=ATmegaBOOT_168_atmega328_pro_8MHz.hex
pro328.bootloader.unlock_bits=0x3F
pro328.bootloader.lock_bits=0x0F
pro328.build.mcu=atmega328p
pro328.build.f_cpu=8000000L
```

```
pro328.build.core=arduino
pro328.build.variant=standard
```

Al modificar el valor pro328.build.f_cpu=**8000000**L se modifica el valor de la frecuencia de la cpu del mismo modo que en el Arduino Uno.

Como ambas placas trabajan sobre el mismo IDE de Arduino las aplicaciones sirven para ambas y las bibliotecas encontradas también son aplicables. Los mecanismos para alterar el consumo de la placa también son idénticos.

3.2.1. Valores iniciales

Del mismo modo que sobre la placa Arduino Uno, se realizan medidas iniciales con la placa funcionando pero sin correr ningún programa (ver fig.3.3.). Dado que se trata de una placa para trabajar en modo bajo consumo se mide a la tensión de entrada que recomienda el fabricante que es de 3.3V y a la mínima tensión operativa que para la placa de pruebas resultó ser de 2.8V. Como se ha visto en el capítulo anterior, incrementar la tensión de entrada supone incrementar el consumo de la placa, y dado que lo que se busca es el mínimo consumo posible se desestiman las medidas con tensiones superiores al rango operativo estipulado por el fabricante.

Tabla 3.13. Consumo Mini Ultra 8Mhz en mA

	500KHz	1MHz	2MHz	4MHz	8MHz
2,8V	0,56	0,76	1,13	1,86	2,94
3,3V	0,64	0,9	1,34	2,23	3,59

Al medir el consumo de la placa los valores obtenidos son realmente interesantes. Si bien es cierto que se parte de la mitad de frecuencia de trabajo, comparando el consumo mínimo del Arduino Uno (ver tabla 3.1) se pasa de unos 13mA, a frecuencias inferiores a 4 MHz a aproximadamente 0,5 mA. Si se opera a una frecuencia más elevada, por ejemplo 8 MHz se miden alrededor de 3 mA lo cual sigue siendo un valor a tener en cuenta.

Se aprecia claramente el descenso de consumo conforme se disminuye la frecuencia de trabajo. La diferencia entre valores al disminuir la tensión de entrada es mayor conforme mayor es la frecuencia de trabajo.

El siguiente paso es medir el consumo de la placa para los distintos modos Sleep y comparar los resultados con los obtenidos sobre la placa de Arduino Uno.

Se realiza una comparativa de los modos Sleep alimentando la placa a 2.8V y a 3.3V y trabajando a varias frecuencias.

Las medidas alimentando a 3.3V (ver tabla 3.14) muestran un valor de apenas 120µA en el modo más restrictivo. Si se compara con los 3.52mA que ofrece el modo Idle a 8MHz se obtiene una mejora más que considerable.

Tabla 3.14. Valores de modos Sleep Mini Ultra a 3.3V en mA

	500KHz	1MHz	2MHz	4MHz	8MHz
IDLE	0,64	0,89	1,34	2,23	3,52
ADC	0,39	0,41	0,45	0,62	0,7
PWR_SAVE	0,35	0,35	0,38	0,44	0,52
EXT_STANDBY	0,33	0,35	0,35	0,44	0,53
STANBY	0,31	0,31	0,31	0,31	0,31
PWR_DOWN	0,12	0,12	0,12	0,13	0,12

De todos los modos de trabajo el menos restrictivo es el modo Idle. Además es el único modo en el cual el descenso de frecuencia afecta significativamente. De hecho, en los modos más restrictivos el consumo no se ve alterado por la variación de frecuencia. Se concluye que en caso de utilizar el modo PWR_DOWN conviene trabajar a 8MHz para disponer de más potencia de procesado y en modo Idle reducir la frecuencia lo máximo que permita la aplicación con el fin de reducir el consumo al máximo.

A continuación se muestran las mismas medidas alimentando a 2.8V (fig. 3.15). Solamente hay una diferencia de medio voltio por lo que los valores que se esperan no deberían diferir demasiado de los anteriormente expuestos.

Tabla 3.15. Valores de modos Sleep Mini Ultra a 2.8V en mA

	500KHz	1MHz	2MHz	4MHz	8MHz
IDLE	0,56	0,77	1,13	1,87	2,95
ADC	0,36	0,37	0,4	0,46	0,57
PWR_SAVE	0,32	0,33	0,35	0,4	0,46
EXT_STANDBY	0,32	0,32	0,36	0,4	0,46
STANDBY	0,28	0,28	0,28	0,28	0,28
PWR_DOWN	0,12	0,12	0,11	0,12	0,11

Se observa una mejora en los consumos pero el consumo mínimo trabajando a PWR_DOWN vuelve a ser el mismo. Dado que normalmente se usará este modo de trabajo para diseños de bajo consumo y que el fabricante establece los 3.3V como tensión operativa se recomienda operar a esa tensión y modo

Sleep. Además se comprueba que la reducción de frecuencia de trabajo no afecta al consumo en ese modo por lo que permite trabajar con más potencia de procesado sin incrementar el coste energético.

Se comprueba cómo es al modo Idle al que más afecta la reducción de frecuencia mientras que el resto de modos mantienen el mismo consumo. El modo pwr_down vuelve a ser el más ecónomico y restrictivo.

Una vez obtenidos los datos de consumo se establece que el menor consumo que ofrece la placa es de 110µA trabajando a 2.8V y 120µA a 3.3V en modo Power_down.

Se desactiva el BOD en ambas situaciones y no se mide ninguna diferencia de consumo. Al desactivar el ADC se miden 21.3µA a 2.8V y 24.3µA a 3.3V lo cual supone una mejora del orden de 100µA. Un consumo de apenas 25µa representa un dato realmente interesante.

La desactivación del PRR se mide sobre la placa sin ningún programa ya que no funciona con los modos sleep. Se parte de 3.52 mA y cuando se desactiva se logran 3.03 mA. Supone un ahorro de 0,5 mA. Añadiendo la desactivación del ADC el valor final es de 2.92mA, que coincide con los 100 uA medidos anteriormente.

De nuevo las distintas configuraciones de los pines de entrada/salida no revelan ninguna diferencia de consumo.

3.2.2. Valores en aplicación

Se carga la aplicación “Blink” sin problemas y gracias al led que incorpora la placa se visualiza que las modificaciones de frecuencia son correctos y que efectivamente la placa esta operativa a 2.8 V. Si se trabaja alimentando a 3.3 V el led muestra más intensidad pero a 2.8 V se ilumina lo suficiente como para diferenciar claramente el parpadeo.

Al comparar el consumo a varias frecuencias (ver Tabla 3.16.) se comprueba que reducir la frecuencia de trabajo repercute directamente y de modo agresivo en la reducción de consumo de la placa. Se consiguen ahorros de más de 2 mA y teniendo en cuenta que se tienen máximas de 3.6 mA y 4.3 mA es un dato importante. Comparando con los valores máximos de la Tabla 3.13 el hecho de iluminar el led representa un incremento de consumo de 700 µA lo cual no supone un valor elevado.

La gráfica (Figura 3.10) demuestra que la relación entra la tensión de entrada y el consumo de la placa es lineal. El consumo de la placa aumenta conforme aumenta la alimentación. También muestra claramente que el trabajo a frecuencias menores optimiza el consumo.

Es fácil deducir que para diseños de bajo consumo con la placa Mini Ultra 8 MHz, siempre y cuando el proyecto lo permita, se limitará la tensión de entrada

al minimo valor operativo y la frecuencia al mínimo valor que requiera la situación.

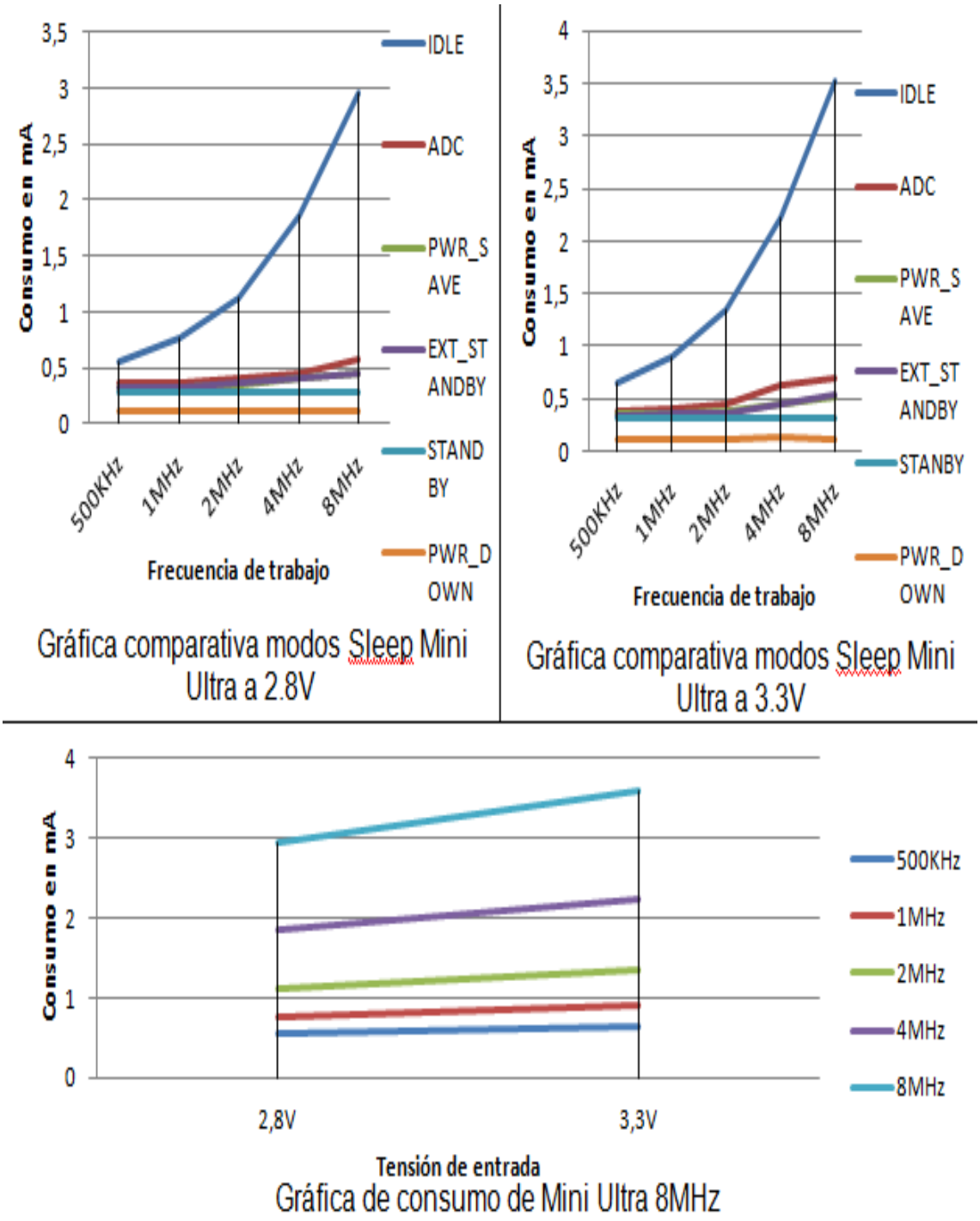


Fig.3.10. Comparativa consumos Mini Ultra 8MHZ

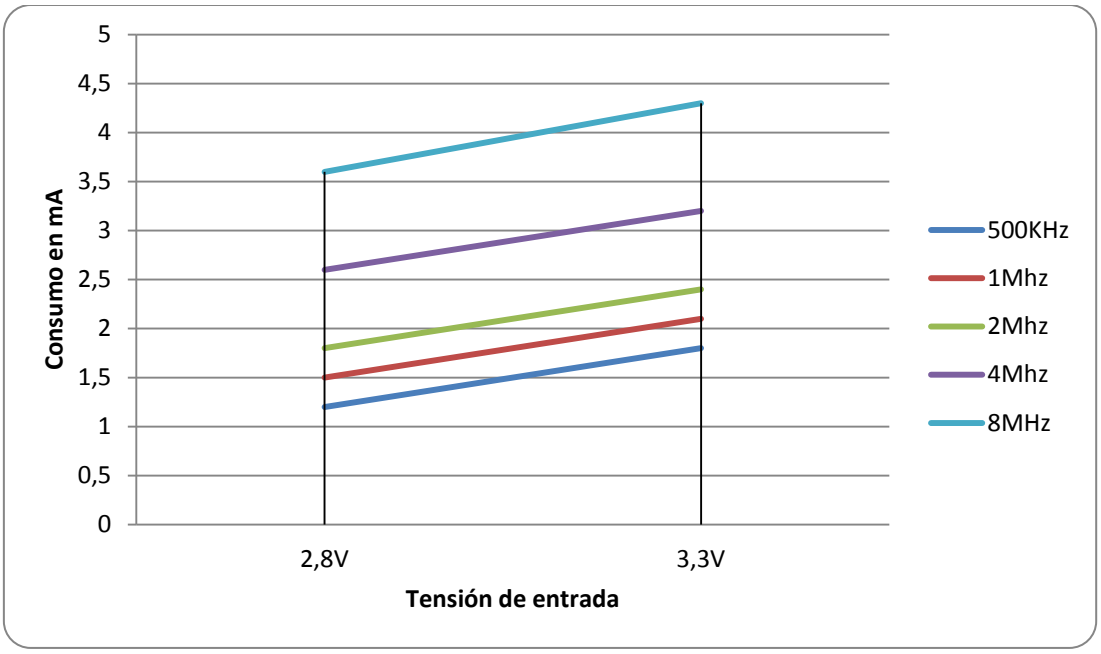


Fig.3.11. Gráfica de consumo Mini Ultra 8MHz aplicación Blink

Tabla 3.16. Consumo Mini Ultra 8Mhz en mA. Aplicación Blink

	500KHz	1Mhz	2Mhz	4Mhz	8MHz
2,8V	1,2	1,5	1,8	2,6	3,6
3,3V	1,8	2,1	2,4	3,2	4,3

Una vez medidos los valores de la aplicación trabajando se procede a desactivar el BOD, ADC, PRR y se configuran los pines de entrada. Aunque anteriormente se miden mejoras de unos 100 µA al ejecutar la aplicación no se registra ninguna modificación en el consumo.

Con Arduino Uno trabajando a 1 MHz y alimentando a 5 V se consigue un consumo de 12.1 mA (Tabla 3.8). Al cambiar a la placa Mini Ultra 8 MHz sin ningún tipo de configuración se obtienen 1.5 mA para 1MHz y 3.3 V de entrada. Es una mejora de 10.6 mA y un consumo final aceptable.

3.2.3. Valores usando bibliotecas

Al medir con las bibliotecas se encuentran problemas al trabajar a la frecuencia de 500 KHz. Las medidas no son estables (oscilan continuamente e incluso hay momentos en los que la placa deja de funcionar) y el parpadeo del led tampoco. Dado que es una frecuencia de trabajo poco práctica debido a la poca potencia de cálculo que ofrece se desestima trabajar a esa frecuencia y se empieza a medir a 1 MHz y alimentar a 3.3 V (Tabla 3.17). Se vuelven a medir valores menores con la aplicación low.power.

Tabla 3.17. Medidas bibliotecas Mini Ultra a 3.3 V

	1MHz	2MHz	4MHz	8MHz
Jeelib	1,5	1,5	1,5	1,6
N0m1	1,6	1,7	1,8	1,8
Low.power	1,5	1,5	1,4	1,5

En la aplicación Low.power los valores no varían al reducir la frecuencia de trabajo por lo que se recomienda utilizar la frecuencia de 8 MHz (Tabla 3.17). Al utilizar la biblioteca Low.power el BOD y ADC se configuran en modo OFF. Esa acción es la que lleva a un consumo menor. Al realizar medidas a 2.8V se obtiene un valor de apenas 845µA (tabla 3.18). En este caso Jeelib y Low.power se comportan del mismo modo. Siguiendo el razonamiento anterior se escoge de nuevo la biblioteca Low.power y la frecuencia de trabajo de 8 MHz. Vuelve a observarse que alterar la frecuencia de trabajo no modifica el consumo de la placa.

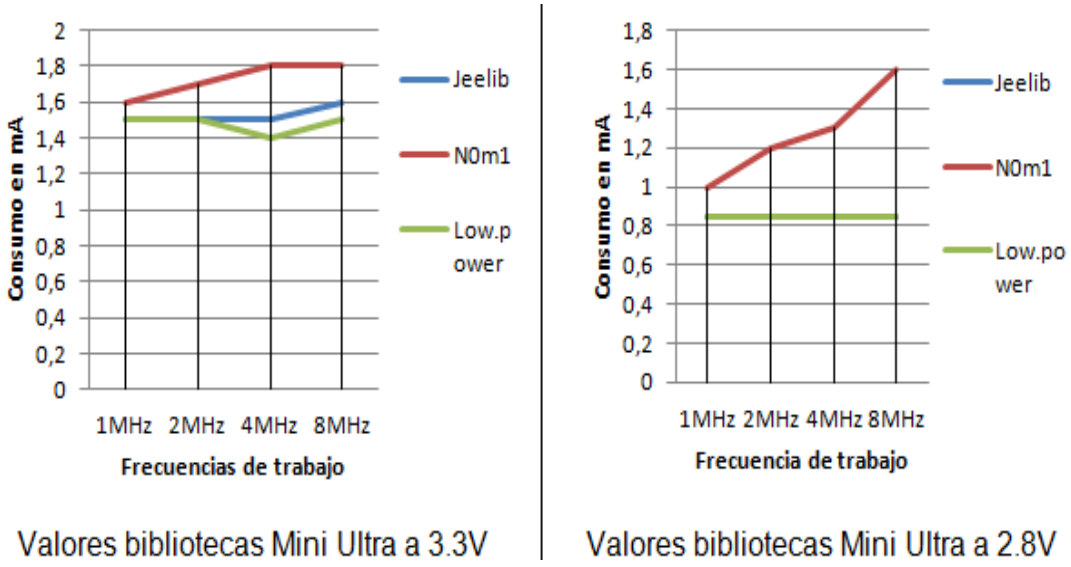


Fig.3.12. Comparativa consumos Mini Ultra a distintas Vin

Tabla 3.18. Medidas bibliotecas Mini Ultra a 2.8 V en mA sin perifericos

	1MHz	2MHz	4MHz	8MHz
Jeelib	0,845	0,845	0,845	0,845
N0m1	1	1,2	1,3	1,6
Low.power	0,845	0,845	0,845	0,845

3.3. Pruebas sobre Arduino Mega y Xbee

Una aplicación muy común realizada con sistemas Arduino consiste en montar distintos dispositivos capaces de comunicarse entre ellos. Sensores remotos, redes de comunicación, servidores Ip, etc. Una de las herramientas para la comunicación entre Arduinos más utilizada son los módulos Xbee. Los módulos Xbee son económicos, potentes y fáciles de utilizar. Son módulos de radio frecuencia que trabajan en la banda de 2.4 GHz con protocolo de comunicación 802.15.4 fabricados por Maxstream.

Los módulos tienen 6 convertidores análogo-digital y 8 entradas digitales. Trabajan a 2.4 GHz y generan una red propia a la que se puede conectar o desconectar.

Tabla 3.19. Características módulo Xbee

Alcance: 100 mts para los módulos Xbee y 1.6 Km para los módulos Xbee Pro
9 entradas/salidas con entradas analógicas y digitales.
Consumo <50mA en funcionamiento y <10uA en modo sleep
Interfaz serial.
65,000 direcciones para cada uno de los 16 canales disponibles.
Fáciles de integrar.

En este apartado se estudia un módulo Arduino Mega. Se trata de una de las placas de mayor tamaño y potencia disponibles en Arduino. La principal diferencia con Arduino Uno es el incremento de pines digitales (tabla 3.20) y que incorpora un procesador ATmega1280, además de un incremento de memoria considerable.

Dado que para un diseño de sensores Xbee puede hacerse necesario una mayor gestión de datos y de que ya se ha recopilado información sobre la placa Arduino Uno, se opta por valorar los consumos de la placa Arduino Mega en un supuesto montaje con módulos Xbee. El hecho de disponer de mayor numero de pines y memoria puede suponer una ventaja para el diseño final.

Tabla 3.20. Características Arduino Mega

Microcontrolador	ATmega1280
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	54 (14 proporcionan salida PWM)
Pines de entrada analógica	16
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	128 KB de las cuales 4 KB las usa el gestor de arranque(bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

3.3.1. Valores Arduino Mega

Al observar las características de la placa Arduino Mega todo parece indicar que el consumo será más elevado que el de Arduino Uno. Dado que anteriormente queda demostrado que el modo de trabajo de menor consumo es el PWR_DOWN (ver cap 3.1.1), se realizan mediciones en modo operativo normal y en ese modo Sleep. Se busca valorar el comportamiento de esta placa a distintos valores de alimentación, frecuencias de trabajo y comprobar si los valores teóricos de consumo se corresponden con la realidad.

La primera medición se realiza con la placa trabajando con una aplicación normal (ver fig 3.2.) para cuantificar el consumo standard y su comportamiento. Siguiendo el procedimiento detallado en el capítulo 2.3, se consigue modificar la frecuencia de trabajo alterando la línea “mega2560.build.f_cpu=**16000000**L. Para comprobar que el cambio es efectivo y correcto se realiza un test con la aplicación “Blink” (ver fig. 3.2.) en la que el destello luminoso del led indica que los cambios son correctos.

No se realiza un estudio sobre la aplicación “Blink” ni respecto al efecto de las bibliotecas dado que carece de relevancia. Al tratarse del mismo IDE el funcionamiento de las bibliotecas esperado es el mismo y se centra el esfuerzo en otras medidas que otorgarán mas información.

Se realiza la valoración de consumo y se muestra en la Tabla 3.21.

Tabla 3.21. Consumo de Arduino Mega en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	18,1	18,8	19,6	21,8	25,4	35,2
6v	26,2	26,9	28	30,7	35,3	44,2
7v	49	49,8	51,2	54,3	58,3	67,4
8v	49,7	50,3	51,4	54,8	59,7	71,3
9v	49,7	50,3	51,4	54,8	59,7	71,3
10v	49,7	50,3	51,4	54,8	59,7	71,3
11v	49,7	50,3	51,4	54,8	59,7	71,3
12v	49,7	50,3	51,4	54,8	59,7	71,3

Al observar la tabla 3.21 se comprueba cómo a partir de 8 V los valores se mantienen estables. Se obtiene un descenso de consumo al alimentar por debajo de los 7 V recomendados por el fabricante. La placa sometida a estudio se mostraba operativa a 5 V y dado el ahorro energético contrastado se recomienda alimentar a esa tensión. La reducción de frecuencia repercute en el ahorro directo de 20 mA en el caso de pasar de 16 MHz a 1 MHz. La menor diferencia de valores se encuentra entre los 500 KHz, 1 MHz y los 2 MHz por lo que dado que el consumo no difiere significativamente, se recomienda operar a 2 MHz con el fin de disponer de más potencia de procesado. Con esta frecuencia y alimentando a 5V se consigue un consumo de alrededor de 20 mA. No se aprecia un incremento de consumo tan lineal como en las mediciones de Arduino Uno y los valores obtenidos son significativamente mayores si se comparan los 34mA obtenidos al alimentar un Arduino Uno a 7 V y 16 MHz (ver Tabla 3.1.) y los 67.4 mA del Arduino Mega en las mismas condiciones.

Obviamente si los valores de Arduino Uno no eran adecuados para aplicaciones que requirieran un bajo consumo, los mostrados por Arduino Mega son totalmente inaceptables para tal fin. Con todo, cabe recordar que Arduino Mega es una placa diseñada para ofrecer potencia de procesado y recursos elevados y no ha sido pensada para entornos de bajo consumo.

A todo esto hay que añadir el consumo del shield necesario para acoplar el módulo Xbee a la placa. En el caso que se estudia se utiliza un “wireless proto Shield” de Arduino. Es una placa sobre la que no se ha podido actuar y que consume por si misma 1.9mA. Existen en el mercado multitud de placas con la misma función de diferentes marcas, pero no se ha encontrado ninguna diseñada para modos de trabajo bajo consumo. Ésta puede ser una ventana abierta a futuras investigaciones.

Se sigue a continuación con las mediciones en modo Sleep PWR_DOWN. Al tratarse de una placa de mayores prestaciones no se espera el mismo descenso de consumo que en Arduino Uno.

Lo primero que se observa al estudiar la Tabla 3.22 es que los valores máximos de han reducido considerablemente respecto a los valores de la 3.21. De hecho al comparar el valor de trabajar a 7V y 16MHz en modo normal a modo sleep pwr_down existe una diferencia del 50%.

Tabla 3.22. Consumo de Arduino Mega en modo PWR_DOWN en mA

	500KHz	1MHz	2MHz	4MHz	8MHz	16MHz
5v	16,9	16,7	16,8	16,8	17,3	17,1
6v	23,8	23,4	23,3	23,3	23,7	23,7
7v	31,2	31,2	31,2	31,2	31,2	31,2
8v	31,3	31,2	31,2	31,2	31,2	31,2
9v	31,3	31,2	31,2	31,2	31,3	31,2
10v	31,3	31,3	31,3	31,3	31,3	31,2
11v	31,4	31,3	31,3	31,3	31,3	31,2
12v	31,4	31,3	31,4	31,4	31,4	31,2

Al igual que en Arduino Uno no existe diferencia de consumo al modificar la frecuencia de trabajo. Esto es debido a que en este modo de trabajo el oscilador externo se detiene y las modificaciones de frecuencia no surgen efecto. La relación entre la alimentación de entrada y el consumo es lineal y se mantiene estable entre los 7V y 12V que son los recomendados por el fabricante. Otra vez la tensión que menor consumo ofrece son los 5 V por lo que sigue siendo la tensión recomendada de trabajo.

Para valorar el ahorro de consumo obtenido al desactivar el ADC se mide el consumo a 5V y 16MHz. Al desactivarlo se miden 125µA menos. Este valor aumentó hasta 150µA al subir la tensión de entrada a 12V. Dado que los valores medidos oscilan sobre los 30mA realmente es un valor que se podría ignorar. No se recomienda desactivar el ADC en los trabajos de esta placa.

Desactivar el BOD y configurar los pins tampoco produjo nigún cambio a pesar del elevado número de éstos. Por lo tanto tampoco es recomendable modificar estos parámetros.

Modificar el PRR tiene sentido en otros modos de trabajo. Al realizarlo se miden 1.4 mA de mejora pero realmente se obtienen mejores resultados de consumo utilizando el modo pwr_down. Con todo si el diseñador escoge otro modo de trabajo sí que se recomienda desactivar el PRR.

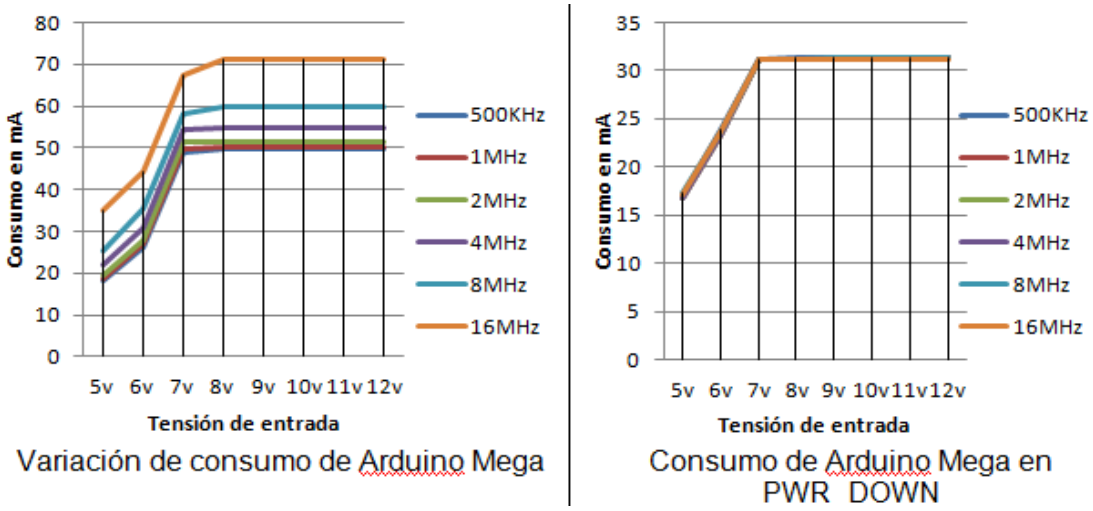


Fig.3.13. Comparativa consumos Arduino Mega

3.3.2. Valores Xbee

Se pasa a medir el valor de consumo de los módulos Xbee y a comprobar si los consumos especificados (Tabla 3.19) se corresponden con los valores medidos. Para las mediciones se trabaja con los módulos Serie 1. Antes de realizar las medidas se realiza un pequeño estudio sobre el modo de funcionamiento y diseño de los módulos.

Los módulos Xbee se alimentan de la misma placa de Arduino por lo que la alimentación viene definida por el sistema. Intentar reducir el consumo del módulo Xbee reduciendo el valor de alimentación no es por tanto factible directamente. A nivel de hardware hay pocas piezas susceptibles de mejora o sustitución por lo que se pasa a valorar el aspecto software. La única opción práctica y sencilla que se encuentra para reducir el consumo se encuentra en saber manejar con soltura los modos Sleep que incorpora.

La manera más sencilla de programar los módulos Xbee consiste en utilizar un adaptador FTDI conectado via USB al pc (Figura 3.13.). El programa X-CTU permite configurar las especificaciones del módulo. Es un tema que se podría ampliar pero no es el objetivo de este trabajo.

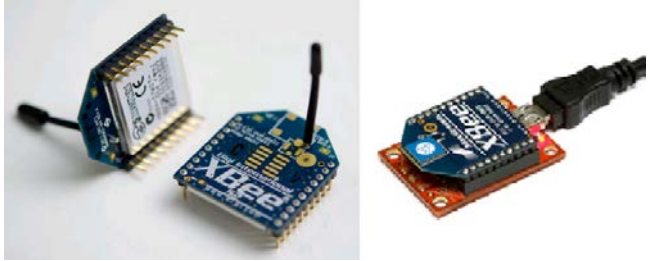


Fig. 3.14. Módulos Xbee y adaptador FTDI

El programa X-CTU (Fig. 3.14.) pone a disposición del programador una carpeta de modos de trabajo Sleep. En ella aparecen 5 pestañas que permiten modelar dicho modo de trabajo. La pestaña SM permite escoger el modo Sleep a utilizar y la ST define la cantidad de tiempo antes de que el módulo de ponga a dormir. La configuración de los ciclos de sueño se realiza principalmente con el comando SM. Por defecto, los modos de sueños están deshabilitados (SM=0), permaneciendo el módulo en estado de reposo/recepción. En este estado el módulo está siempre preparado para responder a un comando, ya sea por el puerto serial o la interfaz RF.

SM=1 Pin de Hibernación. Este modo minimiza el consumo de energía cuando el módulo se encuentra en reposo. Se habilita cuando Sleep_RQ (pin 9) está en alto, el módulo terminará cualquier transmisión, recepción o procedimientos de asociación y entrará en modo de reposo y luego en modo de sueño. En este estado el módulo no responderá a comandos entrantes, desde la interfaz serial o RF. Cuando se baja el estado lógico de Sleep_RQ (pin 9) el módulo saldrá del modo de sueño y estará listo para recibir o enviar.

SM=2 Pin Doze. Este modo cumple la mimsa función que el anterior con la diferencia de que su tiempo para despertar es más elevado y consume más energía.

SM=4 Cyclic sleep remote. Este modo permite que el módulo Xbee revise periódicamente datos vía RF. El modulo está configurado para dormir, entonces se despierta una vez pasado un ciclo completo (definido en SP, cyclic sleep period). Si no hay datos en cola para el receptor, el coordinador no transmitirá y el receptor volverá al dormir para otro ciclo. Si los datos en cola se transmiten de vuelta al receptor, éste se quedará despierto para permitir la comunicación de ida y vuelta hasta que el ST (Tiempo antes de dormir) expire.

SM=5 Cyclic sleep remote w/pin wake up. Unifica los modos SM 1 y 4.

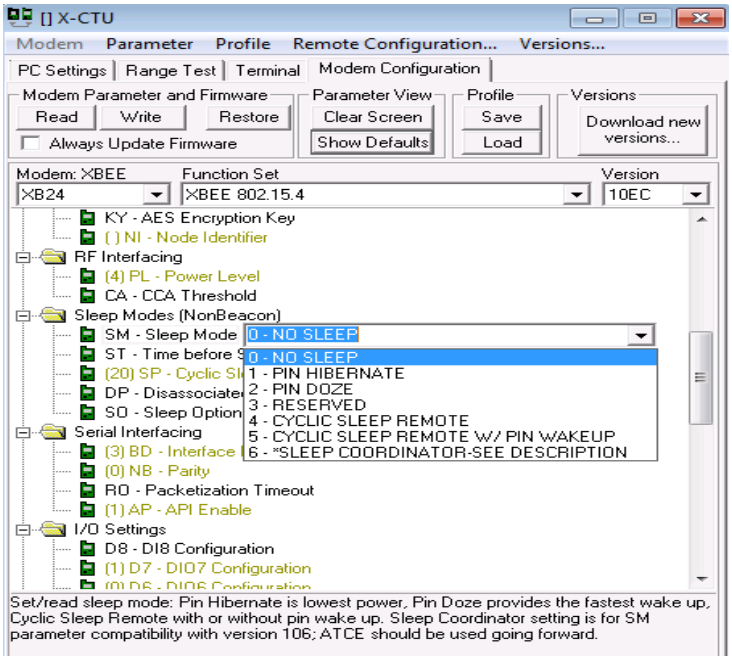


Fig 3.15. X-CTU

Para medir el consumo del módulo Xbee previamente se mide el consumo de la placa más el shield trabajando a 7 V y 16 MHz. Se obtiene un valor de 69.3 mA. Al acoplar el Xbee en modo operativose alcanza un valor de 114.1mA por lo que el consumo que aporta el módulo es de 44.8 mA. Siguiendo este procedimiento se mide el consumo de Xbee en los distintos modos SM (ver tabla 3.23).

Tabla 3.23. Consumo Xbee

	Consumo total en mA	Consumo XBEE en mA
SM1	69,4	0,1
SM2	70,1	0,8
SM3	114,1	44,8
SM4	114,1	44,8
SM5	114,1	44,8
SM6	114,1	44,8

Los modos 4 y 5 muestran el valor del Xbee funcionando durante un tiempo determinado definido en el ciclo de trabajo. Mientras estaban en reposo mostraban un consumo igual al del modo SM2. Los modos SM3 y SM6 no son relevantes. Se aprecia cómo en el modo SM1 el consumo es realmente mínimo reduciéndose a tan solo 100µA (a pesar de que las especificaciones marcan menos de 50µA) y el modo SM2 ofrece 0,8mA lo cual es una cifra muy interesante si se valora que es el resultado que también ofrecen los modos SM 4 y 5. Probablemente estos modos son los más prácticos dada la posibilidad de trabajar bajo ciclos de tiempo determinados por lo que el consumo que refleja en tal modo es el más interesante a la hora de calcular el consumo de la aplicación final.

No se aprecian variaciones de consumo del módulo Xbee al variar la entrada de la alimentación entre 5 V y 12 V. Sí que se modifica el consumo pero es la placa la que consume más corriente, el consumo del módulo se mantiene estable.

Por último se realiza una comparativa del mínimo consumo posible del conjunto y el consumo estándar sin modificaciones. Alimentando la placa a 5V y a 1MHz de frecuencia de trabajo, desactivando el ADC y con el módulo Xbee en el modo Sleep más profundo SM1 se logran medir 18,68 mA. El resultado final procede de la suma de los consumos de los tres elementos integrados, placa, shield y xbee con los valores 16.7 mA+1.84 mA+0.14 mA=18,68 mA. Se aprecia el valor en la figura 3.15.

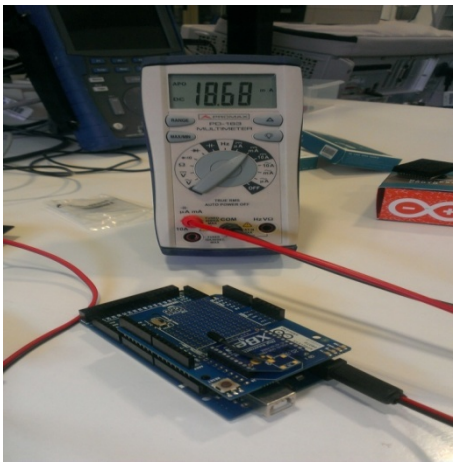


Fig. 3.15. Consumo mínimo conjunto Xbee

Al realizar una medición a valores normales alimentando a 7 V y trabajando a 16 MHz, con todos los periféricos activados se mide un consumo de placa de 67,7 mA más 1,9 mA del shield más 44,8mA del módulo Xbee trabajando con el modo SM a 0. El resultado final es un consumo del conjunto de 114.4 mA.

Comparando un valor y otro se obtiene una diferencia de 95,7 mA lo cual supone un ahorro de consumo del 83%.

Con un Arduino Uno se pasaría a consumir 15,2mA en el modo más restrictivo posible lo que únicamente supone una diferencia de apenas 3mA. Valorando las prestaciones extras que nos ofrece la placa Mega los 3mA parecen un precio asequible a pagar.

Atendiendo a los capítulos anteriores (ver cap.1.2) queda demostrado que el conjunto de Arduino Mega y el módulo Xbee no es útil como un diseño de trabajo autónomo. A pesar de ello ofrece multitud de posibilidades y potencia de procesado por lo que puede ser totalmente válido para diseños en los que la alimentación de los equipos no suponga un inconveniente como por ejemplo la distribución de sensores en un domicilio de cara a realizar una vivienda plenamente domotizada.

Estudiando un nodo comercial, por ejemplo el modelo “Waspmote Mote Runner 6LoWPAN” de Libelium, se observa que tiene un consumo en modo operativo de 15mA a 14MHZ y en reposo de 55µA. Los valores de alimentación varían entre 3.3V y 4.2V. El precio del kit básico ronda los 199€ (el precio final dependerá del distribuïdor final).

Los valores obtenidos de Arduino Mega no pueden competir contra estos consumos ya que para 16MHZ su consumo es de 67,4mA trabajando a los standard 7V de Vin. Sin embargo Arduino Uno logra trabajar a 15.7mA a 5V y 16MHZ ofreciendo un consumo en reposo de 13,3mA. El principal handicap es el elevado consumo en modo reposo que es considerablemente mayor. Sin

embargo el coste de un starter kit de Arduino oscila entre los 95€ y los 100€ lo que supone la mitad del coste de Waspmote.

Dado que para optmizar el consumo de cualquier aplicación es imprescindile trabajar en modos de bajo consumo Arduino no puede considerarse una opción real a un nodo comercial dado el elevado consumo de su modo bajo consumo (13,3mA vs 55µA). En cambio dado que su coste es aproximadamente la mitad (97€ vs 199€) es una alternativa perfectamente válida para diseños poco exigentes en consumo.

La placa Mini Ultra 8MHz otorga un consumo de 3.6mA en modo operativo y 120µA en modo reposo por lo que sí es una alternativa real en cuanto a consumos se refiere. El reducido coste de la placa, aproximadamente 9€, la define como una aternativa a tener en cuenta a falta de comprobar la adaptabilidad y consumo una vez incorporados los periféricos externos.

Tabla 3.24. Comparativa de consumos minimos distintos modelos

	Arduino Uno	**Arduino Mini Ultra 8MHz**	**Arduino Mega**	**Waspmote Libelium**
Consumo operativo	12.1 mA	0.76 mA	18.1 mA	15 mA
Consumo en reposo	12.1 mA	0.12 mA	16.7mA	0.05 mA
Precio	Kit 97€ Unidad 14€	Unidad 9€	Unidad 47,19 €	Kit 199€

CONCLUSIONES

Objetivos alcanzados

El objetivo de este trabajo es evaluar el consumo de Arduino y valorar su adecuación para entornos de bajo consumo. Al realizar las mediciones se comprueba que Arduino no es un sistema pensado para trabajar en estos entornos y que aparentemente no proporciona soluciones inmediatas. Se ha planteado afrontar esta situación estudiando dos vías posibles de modificación, una alternativa vía hardware y otra vía software. Debido a la facilidad de implementación se han considerado de mayor utilidad las soluciones de software y por ello se ha hecho más hincapié sobre éstas.

A pesar de que no se ha conseguido un consumo que realmente pueda considerarse de bajo consumo para la placa Arduino Uno, sí que se han registrado mejoras importantes. Además se ha comparado la efectividad de distintas bibliotecas diseñadas para modos de trabajo bajo consumo con resultados positivos. De las tres bibliotecas estudiadas se muestra cuál es la que mejor rendimiento ofrece. Todo esto otorga a futuros diseñadores una guía sencilla y útil para optimizar el consumo de proyectos en Arduino.

El éxito del estudio se logra con la placa Arduino Mini Ultra 8 MHz alcanzando consumos de apenas 25 µA para el modo en reposo. Si por ejemplo se diseñara una aplicación que trabajara un segundo cada minuto, teniendo en cuenta que el consumo en modo trabajo obtenido es de 1.5 mA, se obtendría un consumo medio de solamente 45 µA. Alimentando con una batería estándar de 2890 mAh se logra una vida útil de algo más de 7 años. Obviamente habría que valorar la descarga de la propia batería y el consumo de posibles dispositivos añadidos. Con todo, es un resultado muy positivo y que aporta una solución real a diseños pensados para trabajar en modo bajo consumo.

Por último se ha valorado el conjunto de una placa de Ardunio Mega con un modúlo Xbee. Es un sistema que claramente no está pensado para entornos de bajo consumo y con el cual no se logra un resultado del todo eficaz. De todos modos sí que se logra reducir el consumo original en un 83% trabajando con los modos Sleep y las frecuencias de trabajo, lo que resulta muy útil de cara a optimizar recursos de futuras aplicaciones.

Siguientes pasos

De cara a profundizar en la investigación lo primero que se recomienda es llevar a cabo las pruebas de hardware estudiadas para con el Arduino Uno. El controlador de potencia externo unificado a una placa de Arduino mejorada (sustitución del regulador de tensión) y configurada para un modo de trabajo en bajo consumo puede dar resultados óptimos en cuanto a mejora de rendimiento.

El primer paso sería sustituir el regulador de tensión de la placa Arduino Uno por el modelo LTC3525. En el capítulo 4.2. ubicado en el anexo se detalla el procedimiento paso a paso.

Otro paso a realizar sería testear el nuevo Arduino Mini Ultra de 16 MHz (no se ha utilizado en este estudio ya que todavía no está disponible en el mercado) para poder apreciar su consumo y compararlo al de 8 MHz. Dado el buen resultado alcanzado por el Mini Ultra de 8 MHz todo hace pensar que el nuevo modelo de 16 MHz también sea un éxito otorgando un consumo mínimo y doblando la potencia de procesado. También sería interesante comprobar el rendimiento de estas dos placas con el regulador de potencia externo descrito en el capítulo 2.4. Si se confirman los datos ofrecidos pueden alcanzarse cotas de consumo mínimas y una potencia operativa muy aceptable.

Por último, la tarea de buscar/encontrar shields de bajo consumo para las placas de Arduino supondría una mejora constante en todas las aplicaciones que requirieran acoplar módulo a las placas. Es muy interesante también estudiar la compatibilidad de las placas de bajo consumo Mini Ultra 8MHz y 16MHz con los módulos Xbee. Dado el bajo consumo de las placas y de los módulos Xbee cuando están en reposo los consumos podrían considerarse extremadamente útiles para diseños autónomos.

Estudio de ambientalización

Gran cantidad de diseños con Arduino funcionan con baterias y pilas alcalinas. Existen gran variedad de estudios que demuestran el impacto negativo que producen estas baterias cuando no se desechan correctamente. Cursos de agua y suelos se ven muy afectados por el mercurio de las pilas y la flora y la fauna se resienten considerablemente.

Optimizar la vida útil de dichas pilas y baterías así como la correcta gestión de los residuos repercute directamente sobre nuestro ecosistema. Por lo tanto, gestionar y optimizar la vida útil de las baterias, pilas y fuentes de alimentación repercute directamente y de forma positiva en la conservación y mejora del medio ambiente.

BIBLIOGRAFÍA

[1] Arduino URL: http://www.arduino.cc/

[2] Forum Arduino URL: http://forum.arduino.cc/

[3] Wiki espacio de Arduino, espacio donde compartir ideas sobre Arduino URL: http://arduino720.wikispaces.com/Power+Saving+Techniques

[4] Surprisingedege, Espacio destinado a la electrónica en general URL: http://www.surprisingedge.com/low-power-atmegatiny-with-watchdog-timer/

[5] The green jornal. Blog destinado a la implementación de soluciones domóticas
URL: http://tae09.blogspot.com.es/2012/10/arduino-low-power-tutorial.html

[6] Blog personal del Dr. Jeffrey Groff URL: http://www.fiz-ix.com/2012/11/low-power-arduino-using-the-watchdog-timer/

[7] Electrical Engineering Stack Exchange, lugar de encuentro de ingenieros eléctronicos.
URL: http://electronics.stackexchange.com/questions/49182/how-can-a-let-my-atmega328-run-for-a-year-on-batteries

[8] Seta43. Blog de artículos de diseños electrónicos. URL: http://seta43.blogspot.com.es/2013/03/arduino-reduciendo-consumo-parte-2.html

[9] EngBlaze. Web abierta de recursos electrónicos. URL: http://www.engblaze.com/low-power-libraries-for-arduino-control-sleep-with-single-function-calls/

[10] Blog personal de Donald Morrisey URL: http://donalmorrissey.blogspot.com.es/2010/04/putting-arduino-diecimila-to-sleep-part.html

[11] Citizen-sensing. Site de la universidad de St. Andrews, departamento de ciencia computacional. URL: http://citizen-sensing.org/2013/07/arduino-watchdog/

[12] Openhomeautomation. Site de automatismos para el hogar. URL: http://www.openhomeautomation.net/arduino-battery/

[13] Laboratoriy for Experimental Computer Science at the Academy of Media Arts Cologne. URL: http://interface.khm.de/index.php/lab/experiments/sleep_watchdog_battery/

[14] Hackaday. Blog personal de proyectos electrónicos. URL: http://hackaday.com/2012/08/18/making-the-arduino-sleep-the-long-sleep/

[15] Olimex. Web de herramientas de desarrollo. URL: http://olimex.wordpress.com/2013/11/05/experimenting-with-low-power-modes-and-arduino/

[16] Blog personal de Alan Mitchel. URL: http://alanbmitchell.wordpress.com/2011/10/02/operate-arduino-for-year-from-batteries/

[17] RocketScream. Enlace a la placa Arduino Mini Ultra 8MHz. URL: http://www.rocketscream.com/shop/mini-ultra-8-mhz-arduino-compatible

[18] Lowpowerlab. Web de diseño de Moteino. URL: http://lowpowerlab.com/moteino/

[19] Geekfactory. Web/blog de suministros electrónicos. http://www.geekfactory.mx/arduino/mitos-de-arduino-las-baterias-duran-poco-para-alimentar-un-arduino/

[20] Blog personal de Hans Crijns, ingeniero eléctrico y desarrollador. URL: http://hwstartup.wordpress.com/2013/03/11/how-to-run-an-arduino-on-a-9v-battery-for-weeks-or-months/

[21] Gammon Software Solutions. Web dedicada a soluciones electrónicas. URL: http://www.gammon.com.au/forum/?id=11497

[22] Rocket Scream. Acceso a biblioteca low.power. URL: http://www.rocketscream.com/blog/2011/07/04/lightweight-low-power-arduino-library/

[23] Sparkfun. Web de suministros electrónicos y soporte al diseño de aplicaciones. URL: https://www.sparkfun.com/tutorials/309

[24] Tim.cexx.org. Web de desarrollo de mosquino. URL: http://tim.cexx.org/?page_id=760

[25] Blog personal de Patrick Thalin. URL: http://www.thalin.se/2013/04/arduino-pro-mini-low-power-modification.html

[26] Tutorial de Arduino básico. URL: http://www.seta43.netau.net/ardu1.html

[27] Dos botones. Blog de diseño de proyectos electrónicos. URL: http://blog.dosbotones.com/2011/09/reducir-el-consumo-de-arduino.html

[28] Blog personal de un diseñador electrónico. URL: http://cybergibbons.com/uncategorized/arduino-misconception-3-it-isnt-low-power-enough-to-be-run-from-battery/

[29] Forefront. Web de diseños electrónicos. URL: http://forefront.io/a/beginners-guide-to-arduino

[30] Jeelabs. Web de soluciones de software. Enlace a la biblioteca Jeelib. URL: http://jeelabs.net/pub/docs/jeelib/classSleepy.html

[31] Engblaze. URL: http://www.engblaze.com/hush-little-microprocessor-avr-and-arduino-sleep-mode-basics/

[32] Blog personal de Donal Morrissey. URL: http://donalmorrissey.blogspot.com.es/2010/04/putting-arduino-diecimila-to-sleep-part.html

[33] Github. Web de recursos software. Enlace con acceso a la biblioteca N0m1. URL: https://github.com/n0m1/Sleep_n0m1

[34] Rocketscream. Enlace a la placa Arduino Mini Ultra 16MHz URL: http://www.rocketscream.com/blog/2013/09/13/introducing-mini-ultra-16-mhz/

[35] 5Herzt. Web de suministros y apoyo electrónico. URL: http://5hertz.com/tutoriales/?p=377#4.ardconv

[36] Makezine. Revista digital sobre electrónica. URL: http://makezine.com/2012/04/24/soapbox-my-top-10-favorite-arduino-compatible-clones-and-derivatives/

[37] Sutsburbiablog. Blog inglés sobre electrónica. URL: http://sustburbia.blogspot.co.uk/2014/03/further-wanderings-in-low-power-land.html

[38] Socgadget. Web sobre gadgets electrónicos y configuraciones. URL: http://www.socgadget.com/2011/07/low-power-xbee-module/

[39] Internet de las cosas. Web sobre diseños wireless e internet de las cosas. URL: http://www.socgadget.com/2011/07/low-power-xbee-module/

[40] Blog personal de John Henry Sammer. Detalla modos de trabajo del módulo Xbee S1. URL: http://www.johnhenryshammer.com/WOW2/pagesHowTo/xbeeSeries1.php#index

[41] Fiz-ix. Blog sobre electrónica general. URL: http://www.fiz-ix.com/2012/11/low-power-xbee-sleep-mode-with-arduino-and-pin-hibernation/

[42] Arduino projects. URL: http://arduprojects.blogspot.com.es/2013/05/comunicacion-entre-arduinos-mediante.html

[43] Sustitución regulador de tensión URL: http://5hertz.com/tutoriales/?p=377#4.ardconv

[44] Especificaciones wapsmote URL: http://www.libelium.com/products/waspmote-mote-runner-6lowpan/

ANEXOS

CAPÍTULO 4. SOLUCIONES HARDWARE

A lo largo de este capítulo se presentan diferentes soluciones de implementación física con el fin de optimizar el consumo de la placa Arduino Uno. Se explica cómo cambiar componentes de la placa, crear una placa en una protoboard con los componentes mínimos, el uso de un controlador externo y el estudio de distintos clones de la placa.

Los estudios de este capítulo son teóricos. No se comprueba empíricamente el resultado de las soluciones presentadas.

4.1. Detalle de la placa

Se procede a estudiar la placa para identificar los elementos que la componen y valorar su posible mejora, intercambio o incluso eliminación con el fin de reducir el consumo.

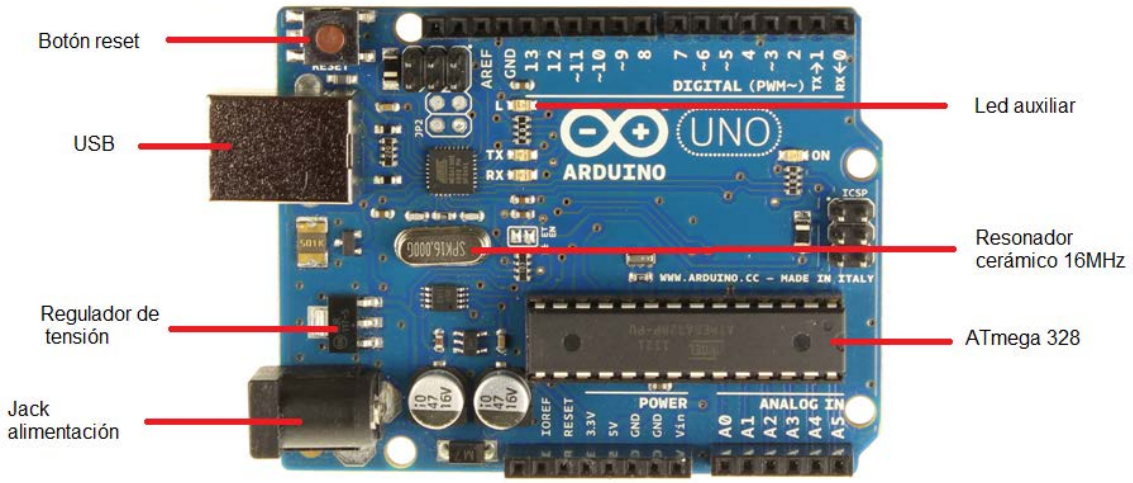


Fig 4.1. Vista frontal Arduino Uno r3

La figura 4.1. muestra los componentes básicos que conforman la placa. Nos encontramos con el microprocesador ATM328, un cristal de 16Mhz, un regulador de tensión que es una variante del LM1117, el puerto USB, la alimentación y un botón de *reset* junto a varios leds. Estas piezas son susceptibles a ser estudiadas. El resto de componentes (pines de entrada y salida, condensadores y resistencias) a priori no deben afectar considerablemente en el consumo de la placa y su modificación es más compleja.

La figura 4.2 muestra la parte trasera de la placa. Tiene unos orificios que permite su acoplamiento a otros dispositivos o placas de mayores de dimensiones para integrarla a distintos proyectos. Todos los elementos están soldados a la placa.

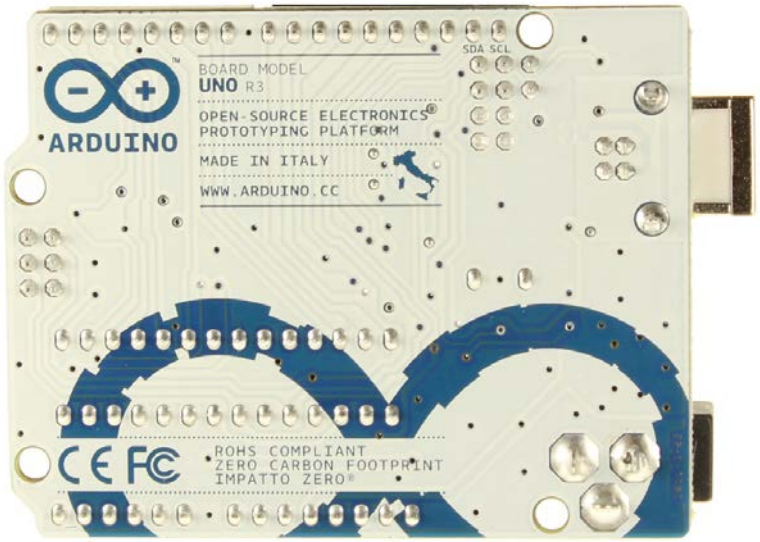


Fig 4.2. Vista posterior Arduino Uno r3

4.2. Regulador tensión

De todos los elementos de la placa el que mayor consumo energético y facilidad de sustitución tiene es el regulador de tensión.

Un regulador de tensión es básicamente un elemento que proporciona una tensión estable. Hablando de Arduino se establecen 5 V de tensión operativa por lo que cualquier tensión de entrada superior será transformada a dicho valor.

La mayoría de las tarjetas Arduino (y compatibles) usan un regulador lineal basado en alguna variante del LM1117. Arduino Uno usa el LM117-5. Un regulador lineal tiene la desventaja de ser poco eficiente, ya que transforma la energía que no llega a la carga en calor.

Si se imagina por un momento que se utiliza una batería de plomo-acido de 12 volts para alimentar un Arduino UNO que consume 50mA, y cuyo voltaje regulado de operación es de 5 voltios. Entonces la energía disipada en forma de calor en el regulador es de $P = (12 – 5) * 0.05A = 0.35 W$ mientras que la energía usada por el microcontrolador es de $P = 5V * 0.05A = 0.25W$. Como podemos observar, la mayoría de la energía (58%) se va como calor en el regulador en vez de realizar un trabajo útil.

Este efecto puede incrementar considerablemente el consumo de la placa, llegando a duplicar el coste de energía necesaria. Así pues se hace imprescindible una adaptación correcta de las tensiones de entrada para cualquier diseño en el que se quiera un consumo moderado.

Los reguladores lineales siempre utilizan corriente, aunque no tengan carga conectada. Esta es una característica del dispositivo y se puede encontrar información sobre este parámetro en la hoja de datos del regulador bajo el nombre de *"Quiescent Current"*. Es una medida necesaria para la operación y polarización de los circuitos internos del regulador y aunque generalmente es una corriente bastante pequeña, es importante mencionarla. Existen reguladores de ultra baja corriente en espera que ayudan a mitigar este problema, pero siempre se va a requerir de una pequeña corriente para el funcionamiento del regulador.

Se plantean dos alternativas para abordar el gasto energético del regulador de tensión. La primera es investigar si existe algún modo de configurar o modificar el regulador que incorpora Arduino Uno para que trabaje en modo bajo consumo. Desafortunadamente, se han revisado las especificaciones técnicas del mismo y no hay ninguna opción para tal fin. La segunda opción es la de sustituir dicho regulador por uno más eficiente y que además otorgue un "*Quiescent Current*" lo más bajo posible. De varios modelos estudiados se llega a la conclusión de que el modelo LTC3525 es el más adecuado para tal fin. Ofrece una salida constante de 5V con una eficiencia cercana al 95% y con un valor de "*Quiescent Current" de 7µA.*

Otra ventaja de modificar el regulador es que con el nuevo modelo se pueden obtener entradas de 3.3V lo cual permite su fácil integración con otros dispositivos o sensores que funcionen a dicha tensión. Acelerómetros o giroscopios son algunos de ellos. A continuación se muestran unas imágenes del procedimiento de intercambio de dicho regulador (ver[43]).

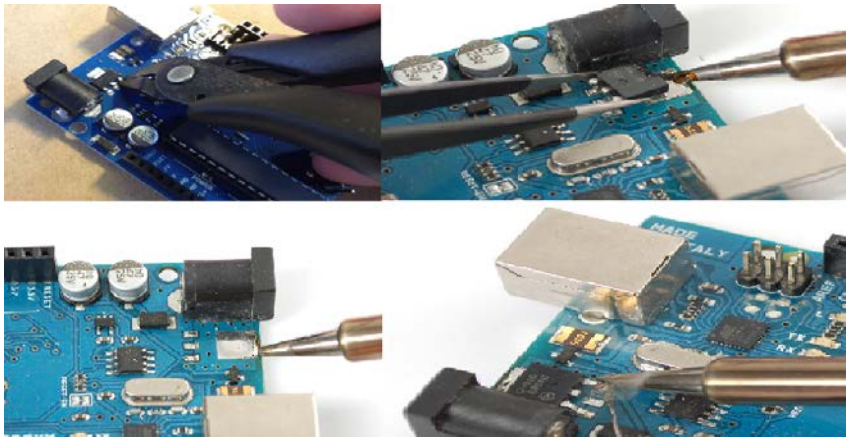


Fig. 4.3. Desmontaje y montaje regulador de tensión en Arduino Uno

4.3. Montaje sobre protoboard

Una opción tomada por muchos diseñadores consiste en montar sobre una “protoboard” los componentes necesarios para imitar una placa de Arduino. De esta manera se puede prescindir de multitud de elementos que consumen batería y que no siempre serán necesarios para el diseño en cuestión. Por ejemplo se puede prescindir de pines de entrada y salida que no sean necesarios o incluso llegar a prescindir del regulador de tensión (asumiendo los riesgos que ello conlleva).

El diseño sobre la “protoboard” sería la implementación rápida y sencilla de ésta alternativa. Una solución definitiva sería la creación de la placa sobre un circuito impreso. Dada la cantidad de placas que se comercializan y los buenos resultados obtenidos con la placa Mini Ultra 8MHz no se considera necesario la realización de dicho circuito.

Básicamente el resultado final es una placa de Arduino (ver[12]) pero sin un puerto USB, botón de reset o leds. Este montaje resulta útil para proyectos en los que sea necesario ahorrar componentes y gasto energético.

Para que el nuevo diseño se asemeje lo máximo posible a Arduino Uno se utiliza un ATmega 328 con el bootloader precargado (es fácil encontrarlo en multitud de distribuidores web por apenas 7,5 €). De esta manera todo el proceso de montaje es mucho más sencillo. A parte del resto de componentes necesarios para el montaje como son resistencias, un reloj de 16MHz, condensadores y algunos leds, es necesaria una placa de carga FTDI para cargar aplicaciones en el microprocesador directamente.

Es importante valorar que ahora la nueva placa no dispone de regulador de tensión (se prescinde de él para reducir consumo aunque podría añadirse al diseño sin problemas) por lo que la alimentación deberá estar comprendida entre los 1,8 V y 5,5 V que especifica el fabricante.

A continuación se muestra una imagen del montaje final y el esquema del mismo con el detalle del montaje y los componentes.

Algunos inconvenientes de esta solución son la poca robustez de la placa (poca protección ante subidas de tensión, desajustes de los componentes y el cableado, etc) y el hecho del aumento considerable de tamaño del proyecto.

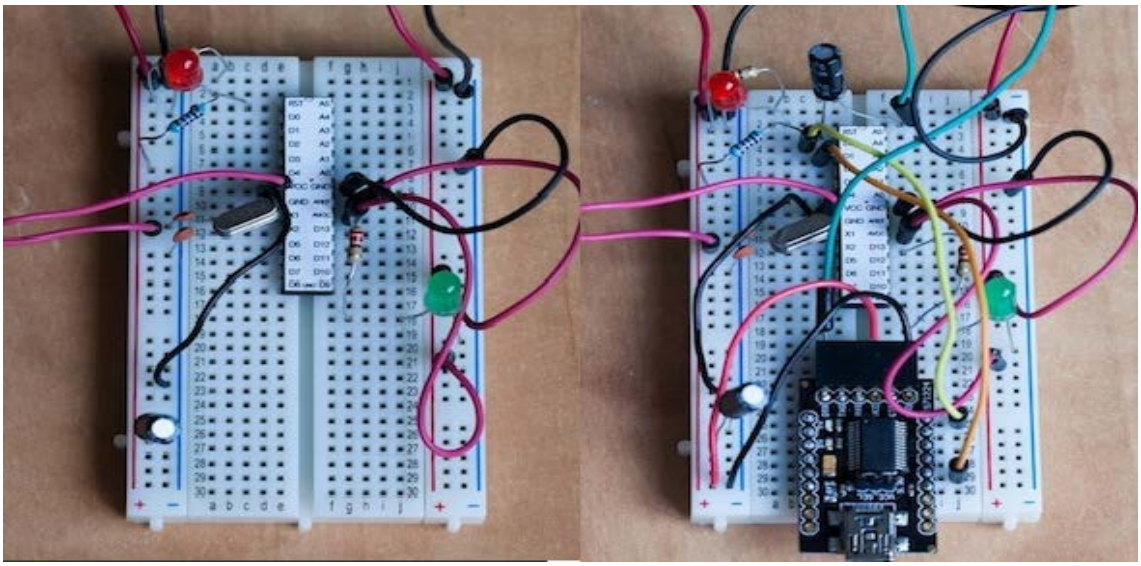


Fig. 4.4. Resultado final con y sin placa FTDI de un Arduino Uno sobre protoboard.

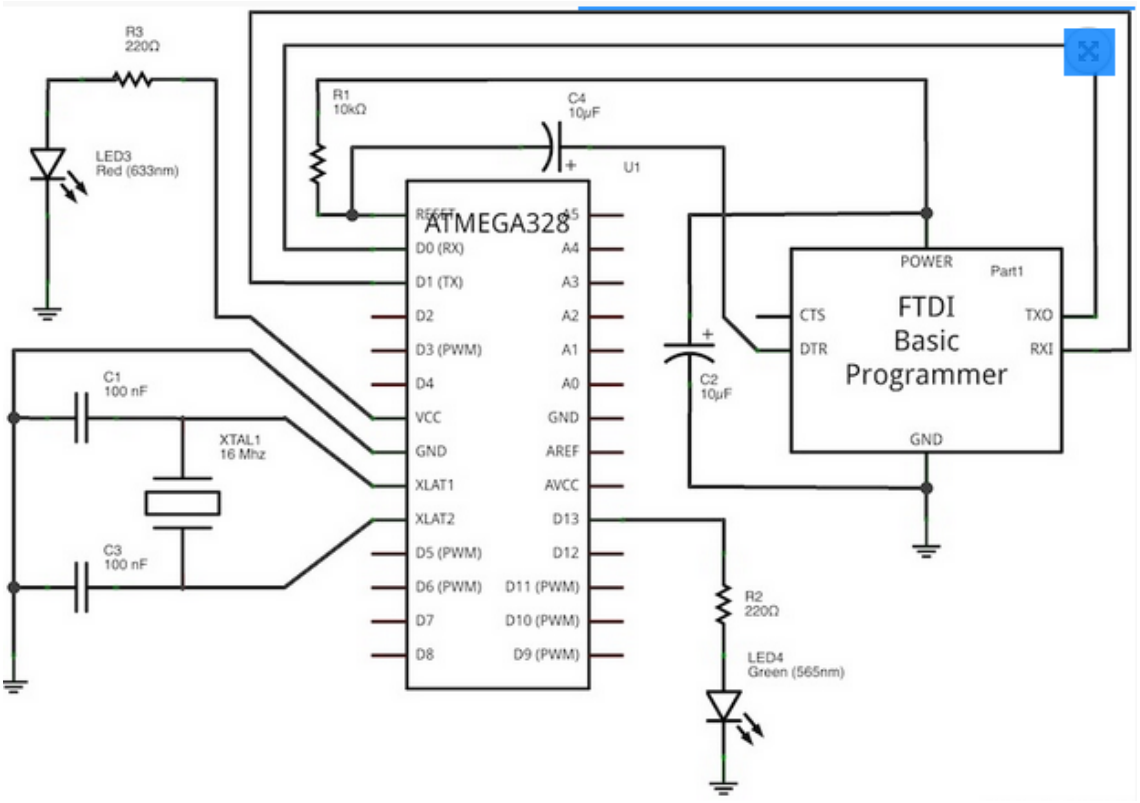


Fig 4.5. Esquema del montaje con relación de componentes.

4.4. Controlador externo

La idea de un controlador externo se basa en la idea de mantener apagada la placa de Arduino Uno mientras ésta no esté operativa. Evidentemente, esta solución solamente será válida en proyectos con unas características puntuales. La información de este apartado se encuentra reflejada en la referencia [16] de la bibliografía y se recomienda su consulta.

Se pone de ejemplo un colector de información. El controlador de potencia alimenta periódicamente la placa, una vez ésta ha capturado y transmitido la información envía una señal para que el controlador corte la alimentación. Hay que tener muy en cuenta que esta solución funcionará en casos de adquisición de datos (se debe usar la memoria EEPROM) y no en las de control ya que al apagar el Arduino se pierde la información.

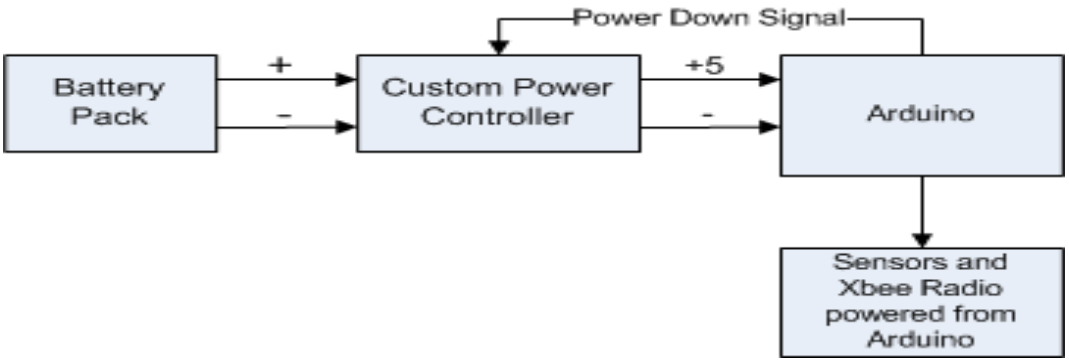


Fig. 4.6. Diagrama del Arduino de bajo consumo.

Para justificar la incorporación de un controlador en aplicaciones de este tipo se hacen los siguientes cálculos.

Una pila alcalina AA aporta unos 2,5 W/h. Si se alimenta el sistema con un pack de tres se obtienen 7,5 W/h. Suponiendo que el sistema pudiera pasar la mayor del tiempo durmiendo el regulador de tensión consumirá del orden de 10mA, lo cual nos lleva a un consumo de 0.045 W. Haciendo números redondos y sin tener en cuenta perdidas se obtienen apenas 7 días de funcionamiento en un caso ideal y sin funcionamiento.

$$\frac{\frac{7.5W}{h}}{0.045W} = 166{,}6\ h = 6{,}9\ días$$

En el proyecto que se estudia se necesita leer y enviar información cada 10 minutos. Arduino Uno tarda 72 milisegundos en despertar, 10 milisegundos en leer del sensor y alrededor de 1 segundo en transmitir los datos. Teniendo en cuenta el consumo del sensor se obtiene un consumo aproximado de 70 mA cuando el Arduino está operativo. Dado que el sistema se alimenta

directamente por el Vin de la placa con 5V se obtiene una potencia 70mA*5V=0.35W. Hay que tener en cuenta que el circuito está operativo únicamente 1 segundo cada 10 minutos por lo que el consumo real es:

$$1\frac{seg}{600seg(10min)} * .035W = 0.583mW$$

A este valor hay que añadir los 11μW que consume el propio controlador de potencia por lo que se obtiene un valor total de 0.594mW.

$$\frac{\frac{7.5W}{h}}{0.583mW} = 12.864\ h = 536\ días = 17\ meses$$

Si se comparan los aproximadamente 7 días con los 17 meses teóricos el resultado es apabullante.

Dado que el propósito de este trabajo es evaluar el consumo de Arduino por si mismo, no se ha desarrollado ni comprobado el resultado de este estudio. No se valida por tanto dicho resultado. A pesar de todo se considera que es una aportación interesante de cara a futuros desarrolladores y a cualquier aplicación basada en la adquisición y transmisión de datos. En el enlace expuesto anteriormente (consultar [12]) se detalla el diseño del propio controlador.

4.5. Clones de Arduino

En el mercado se encuentra una enorme variedad de sustitutos de Arduino. Son muchas las empresas que diseñan sus propias placas con el IDE de Arduino y las venden a un precio algo más barato y con unas características parecidas. De entre todas estas propuestas aparecen diseños "*low power*", es decir, diseños con características semejantes a las de Arduino pero con un consumo teórico menor. De entre varios modelos se escogen 3 que se asemejan a Arduino Uno y se realiza una comparativa.

4.5.1. Moteino

Se empieza la lista con la placa Moteino. La definición de la placa dice que se trata de una placa de bajo coste y bajo consumo, wireless y que incluye un procesador ATmega328 100% compatible con Arduino IDE. A diferencia de Arduino Uno no incluye una entrada USB por lo que la comunicación con el ordenador debe llevarse a cabo mediante un adaptador externo FTDI. Lo que lo hace más interesante es que además es compatible con el transceptor RFM12B o el modelo posterior RFM69, por lo que solamente soldando la antena tenemos un equipo listo para enviar y recibir información.

El montaje de placa es artesanal, se realiza a mano y puede conseguirse desde apenas 11€, lo que supone la mitad del coste de un Arduino Uno. Hay que recordar que no incluye el adaptador FTDI. Debido a las numerosas peticiones de los usuarios se ha diseñado una placa que incluye una entrada mini-USB. El precio de esta placa asciende a los 22€ que cuesta un Arduino Uno por lo que sin comprobar si realmente se obtienen consumos menores la única diferencia es el tamaño y la compatibilidad sin usar *shileds* con las antenas mencionadas anteriormente.

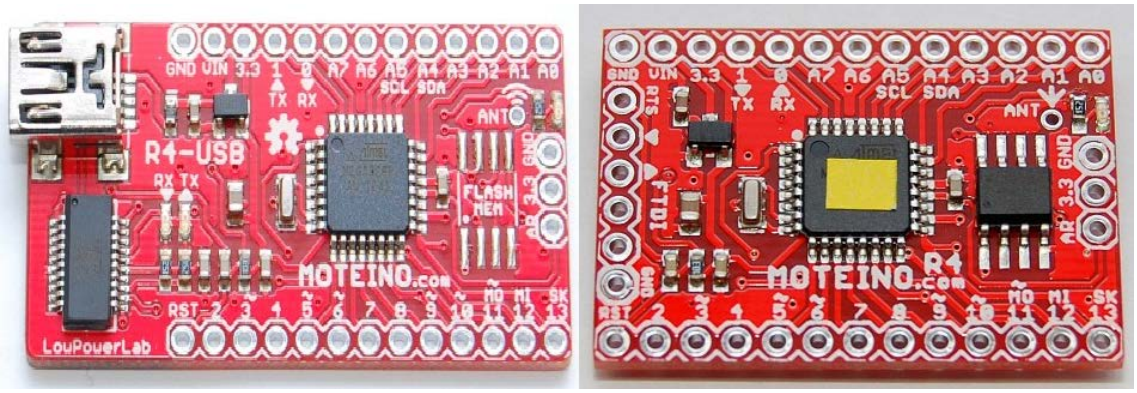


Fig. 4.7. Moteino

Evaluando las especificaciones de Moteino (véase tabla 4.1.) se observa que una de las principales diferencias es la alimentación (opera a 3,3 V respecto a los 5 V de Arduino Uno) y la corriente es de 30 mA respecto a los 40 mA de Arduino Uno.

Tabla. 4.1. Especificaciones Moteino

Microcontroller	ATmega328
Transceiver	RFM12B (all revisions) and RFM69 W/HW/CW (R3, R4)
Transceivers frequencies	434Mhz (universal), 868Mhz (EU), 915Mhz (US, Australia, etc.)
Operating Voltage	3.3V
Input Voltage (recommended)	3.3V-9V
Input Voltage (max limits)	3.3V – 13V
Digital I/O Pins	14+6 (6 PWM capable: marked with "~" symbol)
Analog Input Pins	8 (2 analog-only pins more than regular Arduinos)
DC Current per I/O Pin	30 mA
DC Current for 3.3V Pin	40 mA
Flash Memory	32 KB of which 1 KB used by DualOptiboot bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
MISC	Onboard LED on pin D9 instead of D13 (D9 is PWM!)
	RFM12B or RFM69 SPI-CS on D10
	FLASH SPI-CS on D8
	A6 and A7 are analog pins only, cannot be used as digital pins

4.5.2. Mini Ultra 8Mhz

Esta placa se ha diseñado con el claro objetivo de realizar una placa Arduino de bajo consumo. Uno de los puntos fuertes es que además proporciona una biblioteca para poder gestionar esos modos de trabajo en bajo consumo. Ademas incorpora un reloj de 8Mhz respecto al de 16Mhz de Arduino Uno y su tensión de trabajo es de 3.3V respecto a los 5 de Arduino Uno. Tambíen incorpora un regulador de tensión con apenas 1,6µA de "*quiescent current*". Se hace necesario un dispositivo externo FTDI para cargar los programas dado que no incorpora ningún puerto USB. El coste ronda los 12€ lo que supone la mitad de lo que cuesta Arduino Uno.

Tabla 4.2. Especificaciones Mini Ultra 8Mhz

Compatible with Arduino IDE using “Arduino Pro Mini 3.3 V 8 MHz” as the board option
Microcontroller – ATmega328P-AU
Clock – External ceramic 8 MHz resonator
Operating voltage – 3.3 V
Low quiescent current (1.6 µA) on-board 3.3 V 250 mA regulator
Ultra low power – Minimum 1.7 µA in power down mode
External DC source range – 3.4 – 6 V
Indicator LED connected to digital pin 13 through a MOSFET
FTDI 6-pin header
Emphasis on low noise design approach:
Analog reference voltage with LC filter
Digital pins (D0-D13) breakout on 1 side of the board
Analog pins (A0-A7) breakout on 1 side of the board:
Analog pins A6 & A7 are pure analog input pins only
600 mil in between 2 breakout headers - suitable for breadboard usage
Dimension – 18.41 mm x 39.37 mm (0.725″ x 1.55″)
RoHS compliant – Yes

Observando la tabla 4.2 se comprueba que la placa dispone de 14 pins digitales y 8 analógicos. Las medidas son menores que las de Arduino Uno. A priori las desventajas que se observan son que dispone de menos potencia de procesado debido al reloj de 8Mhz y que no dispone de puerto USB lo que conlleva a utilizar un adaptador FTDI incrementando el precio. Un adaptador FTDI esta rondando los 14 euros en varios distribuidores web.

Fig. 4.8. Mini Ultra 8Mhz

4.5.3. Tinyduino

Tinyduino puede resumirse como un Arduino Uno en pequeño. Incorpora el mismo procesador ATmega 328 pero el tamaño de la placa es de apenas 2x2 cm. Tiene además la característica de que funciona a base de *shields*. Por ejemplo, ya que no incorpora una entrada USB para cargar programas dispone de un shield con una entrada USB para tal fin.

Tabla 4.3. Especificaciones Tinyduino

Arduino and LilyPad Compatible
Expandable with Stackable TinyShield Boards
Optional battery connector for CR1612-CR1632 coin cell batteries
0.1″ spaced solder holes for external power source
Robust Gold Finish – makes soldering easy and is non-corrosive
Ultra compact size and weight (smaller than a US Quarter!)
Square Version: 20mm x 20mm (.787 inches x .787 inches)
Circular Version: 20mm (.787 inches) diameter
Max Height (without battery holder): 2.9mm (0.12 inches)
Max Height (with battery holder): 6.58mm (0.26 inches)
Ultra-thin 0.61mm (0.024 inches) PCB
Weight: TBD grams (TBD ounces)
Atmel ATmega328P Microcontroller
32KB Flash, 2KB RAM, 1KB EEPROM
1.2mA (typical) @ 3V, 4MHz
Default Clock speed: 8MHz
2.7V – 5.5V operating voltage (Arduino mode)

Aporta prácticamente las mismas especificaciones que Arduino Uno con la excepción de que el reloj es de 8Mhz. Funciona con un modo de trabajo modular de manera que si por ejemplo no se necesita un regulador de tensión se puede eludir dicho shield, o si no se necesita el puerto USB éste puede ser retirado una vez cargada la aplicación. De esta manera se reduce y optimiza el tamaño de la placa considerablemente. Cabe mencionar que el fabricante además dispone de placas cuadradas o redondas en función de las necesidades finales del proyecto.

Es interesante mencionar que dispone de sus propios Tinyshields. Es decir, no funciona con los *shields* que podemos encontrar en el mercado. El fabricante, Tinycircuits, ha diseñado una gran variedad de *shields* que se ajustan a la perfección al Tinyduino creando una unidad muy compacta.

No se especifica si ha diseñado para trabajar en modo bajo consumo ni aparece en la página del anunciante referencia alguna en cuanto a mediciones de consumo. Por lo tanto es una alternativa a Arduino Uno sobre todo en

cuanto a optimización de espacio. El precio de la placa base ronda los 17€ mientras que el shield con el puerto USB alcanza los 15€ por lo que el precio final (teniendo en cuenta el adaptador USB) es de unos 32€, unos 10€ más caro que el Arduino Uno.

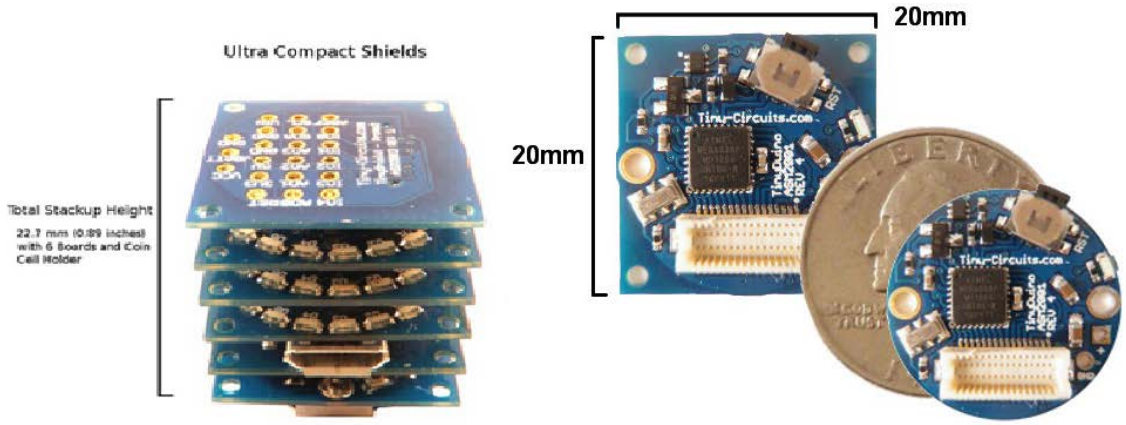
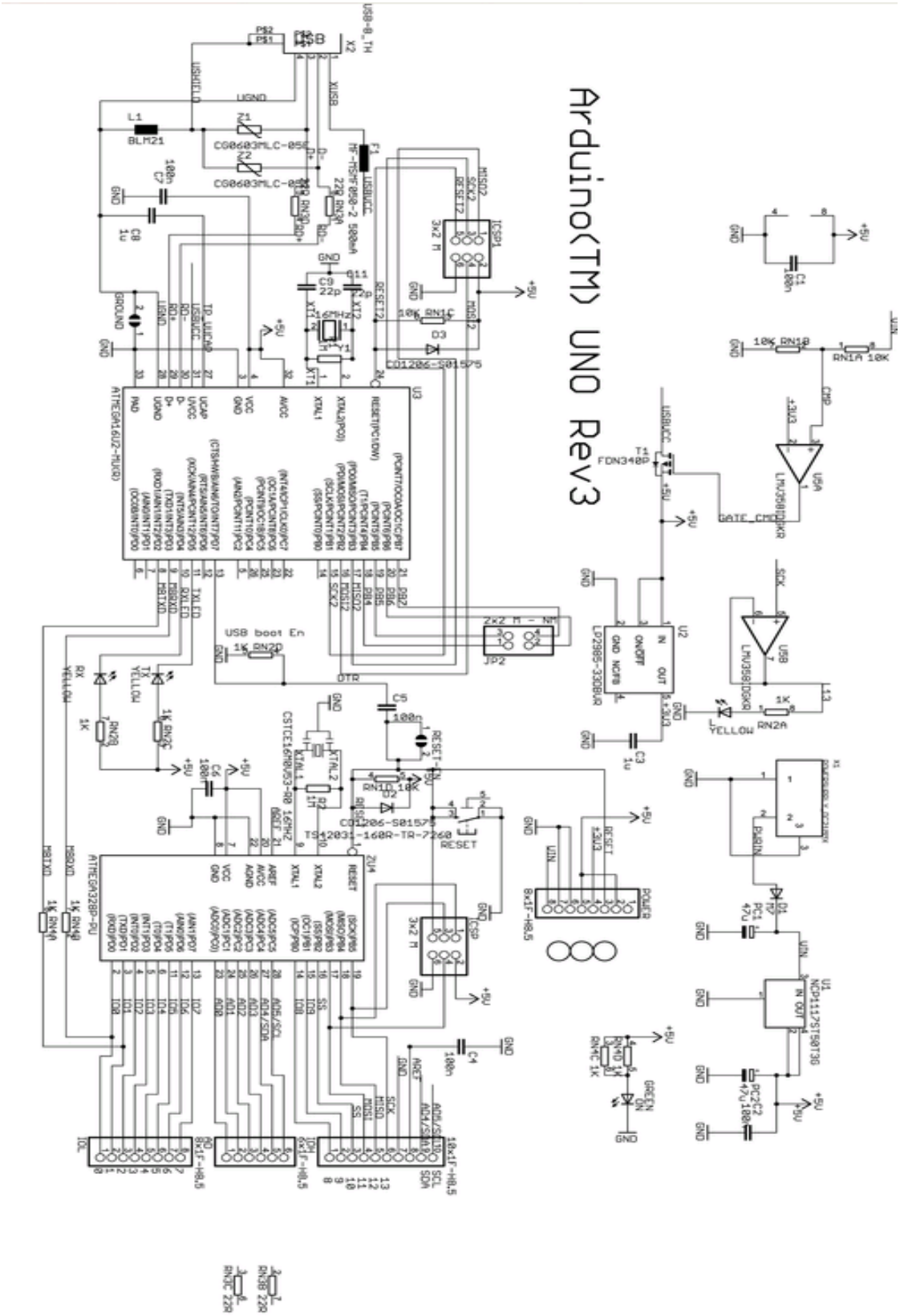


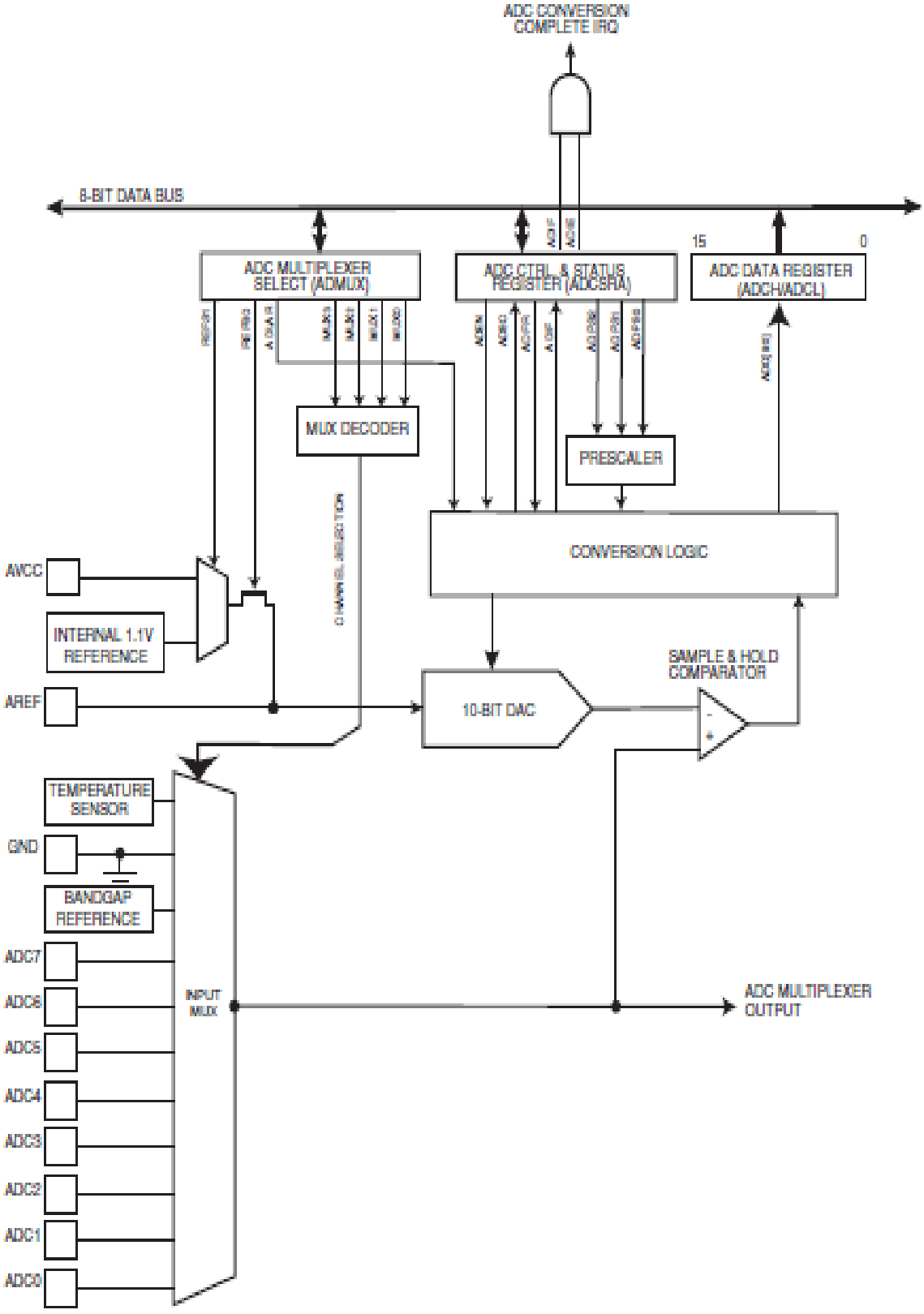
Fig. 4.9. Tinyduino

Finalmente se escoge como candidata a estudio la placa Mini Ultra 8 MHz. Al tratarse de una placa diseñada específicamente para modos de bajo consumo, contar con un tamaño reducido y un precio ajustado encaja en las necesidades de este trabajo.

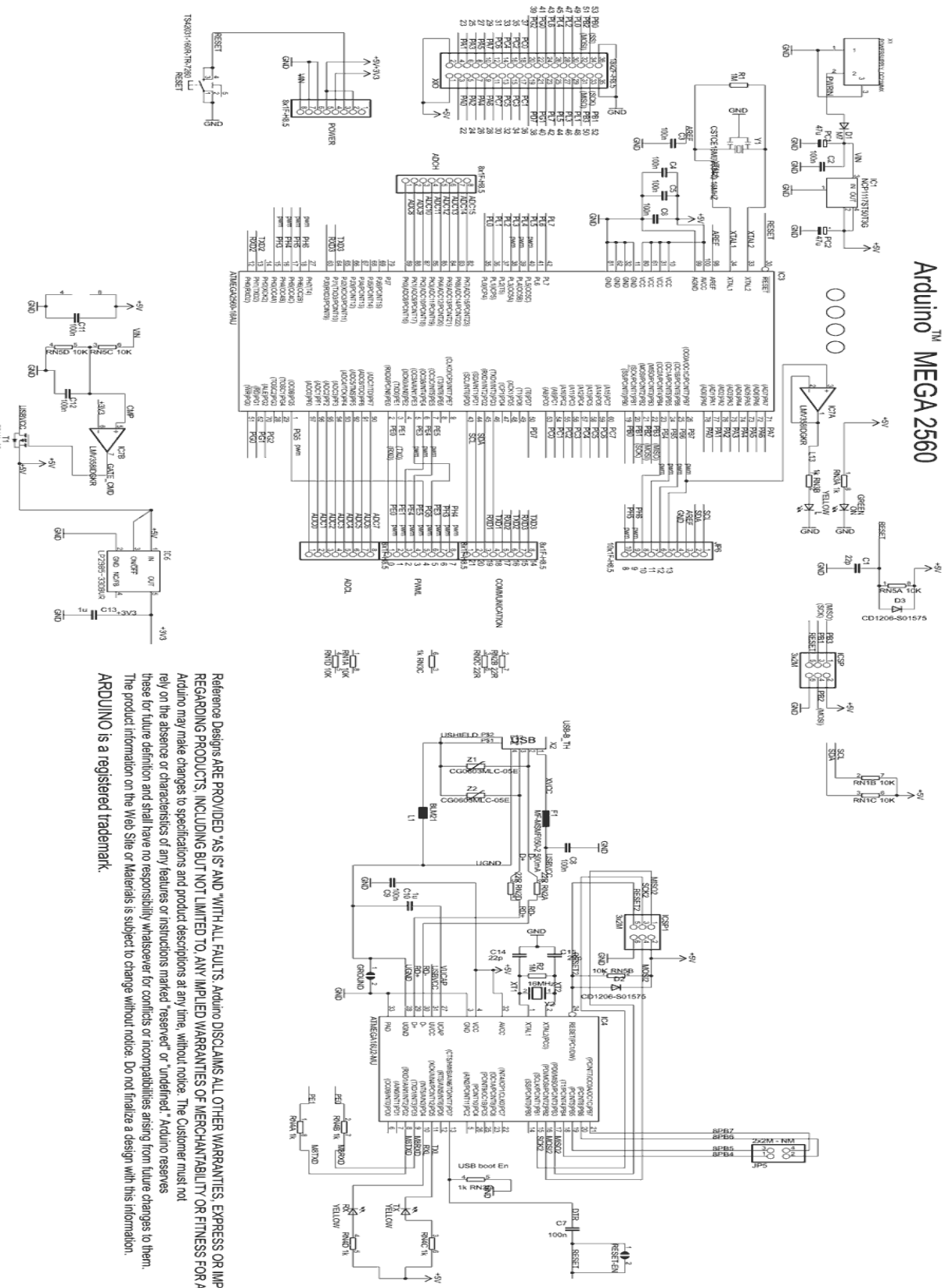
ANEXO A. Schematichs Arduino UNO



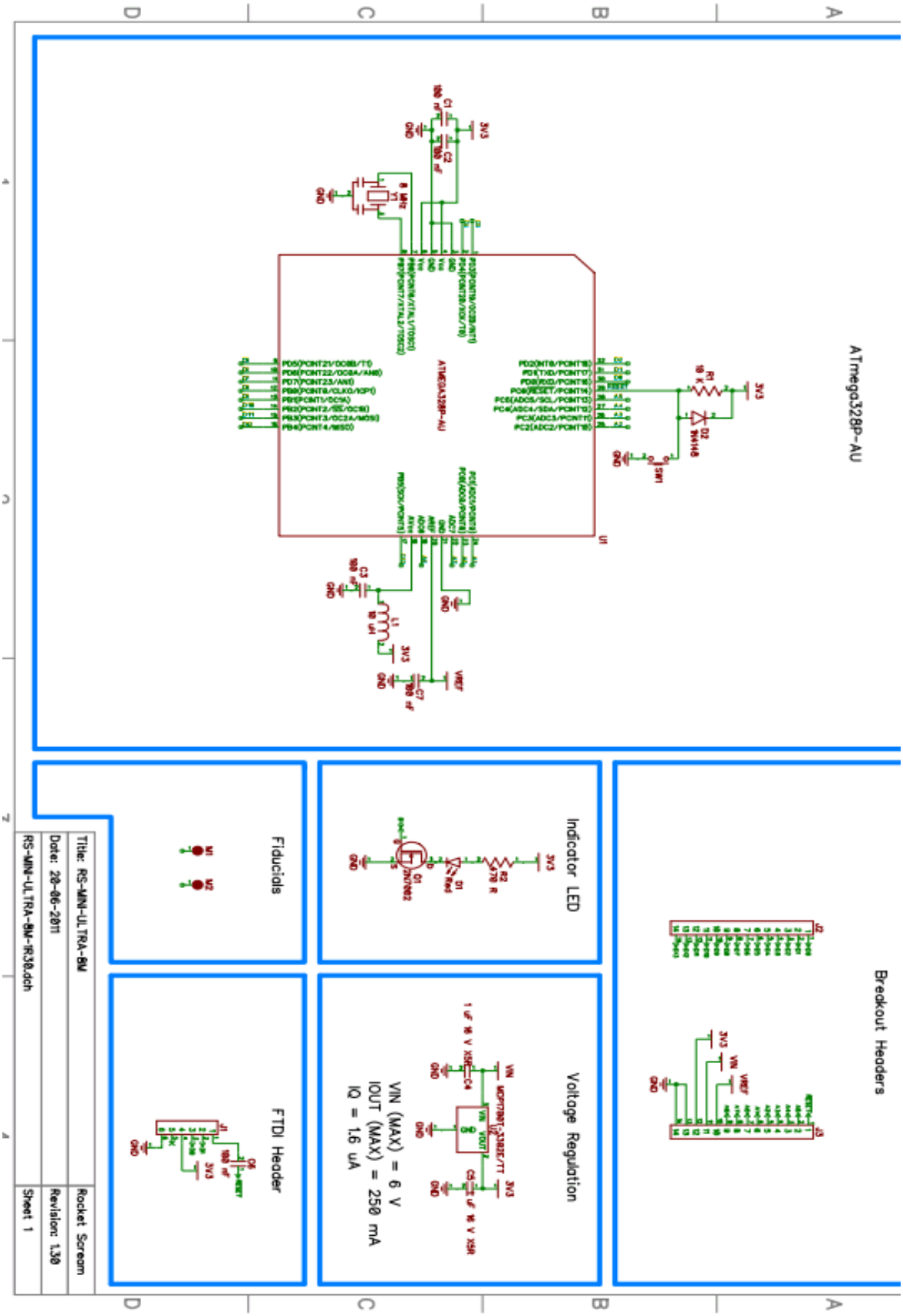
ANEXO B. Bloque ADC



ANEXO C. Schematichs Arduino MEGA



ANEXO D. Schematichs Arduino Mini Ultra 8MHz



ANEXO E. Datasheet LTC3525

LINEAR TECHNOLOGY

LTC3525-3/ LTC3525-3.3/LTC3525-5

400mA Micropower Synchronous Step-Up DC/DC Converter with Output Disconnect

FEATURES

- **Up to 95% Efficiency**
- **Output Disconnect and Inrush Current Limit**
- **Fixed Output Voltages of 3V, 3.3V or 5V**
- **Delivers 65mA at 3V from a 1V Input**
- **Delivers 60mA at 3.3V from a 1V Input, or 140mA at 3.3V from a 1.8V Input**
- **Delivers 175mA at 5V from a 3V Input**
- **Burst Mode® Operation: I_Q = 7µA**
- Only Three External Components
- $V_{IN} > V_{OUT}$ Operation
- <1µA Shutdown Current
- Antiringing Control
- Short-Circuit and Overtemperature Protection
- Very Low Profile of 1mm
- Tiny 6-Pin SC70 Package

APPLICATIONS

- MP3 Players
- Portable Instruments
- Glucose Meters
- Digital Cameras

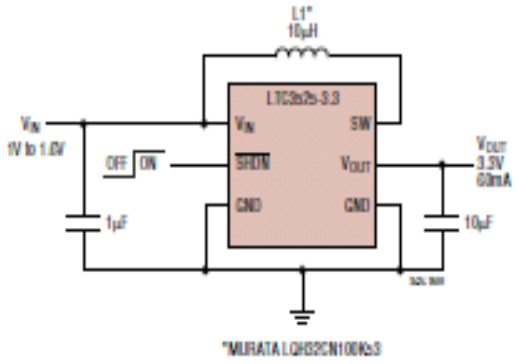
DESCRIPTION

The LTC®3525-3/LTC3525-3.3/LTC3525-5 are high efficiency synchronous step-up DC/DC converters with output disconnect that can start up with an input as low as 1V. They offer a compact, high efficiency alternative to charge pumps in single cell or dual cell alkaline or Li-Ion applications. Only three small external components are required. The LTC3525 is offered in fixed output voltages of 3V, 3.3V or 5V.

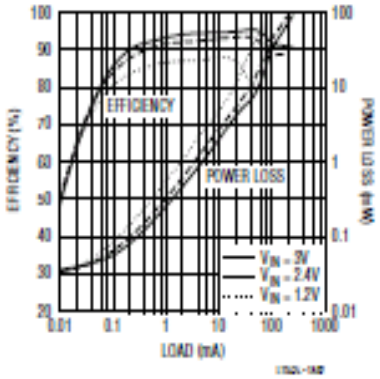
The device includes a 0.5Ω N-channel MOSFET switch and a 0.8Ω P-channel synchronous rectifier. Peak switch current ranges from 150mA to 400mA, depending on load, providing enhanced efficiency. Quiescent current is an ultralow 7µA, maximizing battery life in portable applications.

Other features include <1µA shutdown current, antiringing control and thermal shutdown. The LTC3525 is available in a tiny 6-pin SC70 package.

TYPICAL APPLICATION



LTC3525-3.3 Efficiency and Power Loss vs Load Current

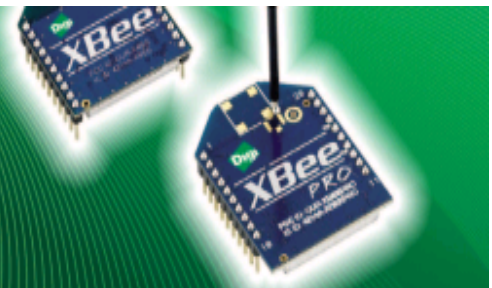


ANEXO F. Datasheet Xbee module

RF Modules

Embedded RF Modules for OEMs

Low-cost, easy-to-deploy modules provide critical end-point connectivity to Digi's Drop-in Networking product family.



Overview

XBee Product Family

The XBee family of embedded RF modules provides OEMs with a common footprint shared by multiple platforms, including multipoint and ZigBee/Mesh topologies, and both 2.4 GHz and 900 MHz solutions. OEMs deploying the XBee can substitute one XBee for another, depending upon dynamic application needs, with minimal development, reduced risk and shorter time-tomarket.

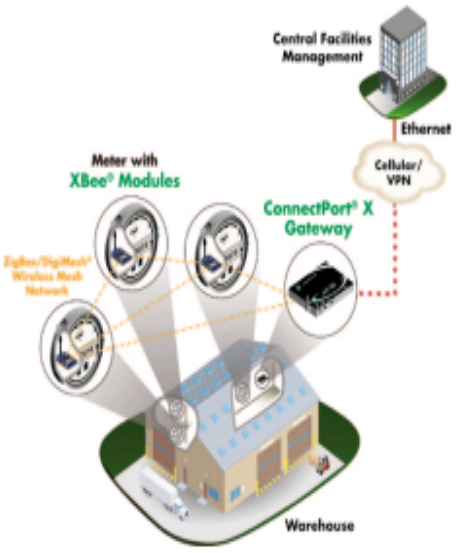
Why XBee Multipoint RF Modules?

XBee multipoint RF modules are ideal for applications requiring low latency and predictable communication timing. Providing quick, robust communication in point-to-point, peer-to-peer, and multipoint/star configurations, XBee multipoint products enable robust end-point connectivity with ease. Whether deployed as a pure cable replacement for simple serial communication, or as part of a more complex hub-and-spoke network of sensors, XBee multipoint RF modules maximize wireless performance and ease of development.

Drop-in Networking End-Point Connectivity

XBee OEM RF modules are part of Digi's Drop-in Networking family of end-to-end connectivity solutions. By seamlessly interfacing with compatible gateways, device adapters and extenders, XBee embedded RF modules provide developers with true beyond-the-horizon connectivity.

Application Highlight



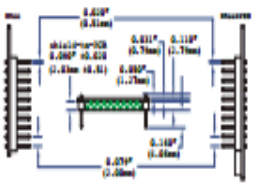
Features/Benefits

- 802.15.4/Multipoint network topologies
- 2.4 GHz for worldwide deployment
- 900 MHz for long-range deployment
- Fully interoperable with other Digi Drop-in Networking products, including gateways, device adapters and extenders
- Common XBee footprint for a variety of RF modules
- Low-power sleep modes
- Multiple antenna options
- Industrial temperature rating (-40° C to 85° C)
- Low-power and long-range variants available

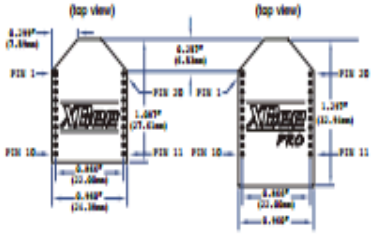
Platform	XBee® 802.15.4 (Series 1)	XBee-PRO® 802.15.4 (Series 1)	XBee-PRO® XSC
Performance			
RF Data Rate	250 kbps	250 kbps	10 kbps / 9.6 kbps
Indor/Urban Range	100 ft (30 m)	300 ft (100 m)	Up to 1200 ft (370 m)
Outdoor/RF Line-of-Sight Range	300 ft (100 m)	1 mi (1.6 km)	Up to 6 mi (9.6 km)
Transmit Power	1 mW (+0 dBm)	60 mW (+18 dBm)*	100 mW (+20 dBm)
Receiver Sensitivity (1% PER)	-92 dBm	-100 dBm	-106 dBm
Features			
Serial Data Interface	3.3V CMOS UART	3.3V CMOS UART	3.3V CMOS UART (5V Tolerant)
Configuration Method	API or AT Commands, local or over-the-air	API or AT Commands, local or over-the-air	AT Commands
Frequency Band	2.4 GHz	2.4 GHz	902 MHz to 928 MHz
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)	DSSS (Direct Sequence Spread Spectrum)	FHSS (Frequency Hopping Spread Spectrum)
Serial Data Rate	1200 bps - 250 kbps	1200 bps - 250 kbps	1200 bps - 57.6 kbps
ADC Inputs	(6) 10-bit ADC inputs	(6) 10-bit ADC inputs	None
Digital I/O	8	8	None
Antenna Options	Chip, Wire Whip, U.FL, & RPSMA	Chip, Wire Whip, U.FL, & RPSMA	Wire Whip, U.FL, RPSMA
Networking & Security			
Encryption	128-bit AES	128-bit AES	No
Reliable Packet Delivery	Retries/Acknowledgments	Retries/Acknowledgments	Retries/Acknowledgements
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 Channels	PAN ID, 64-bit IEEE MAC, 12 Channels	PAN ID, 32-bit Address, 7 Channels
Power Requirements			
Supply Voltage	2.8 - 3.4VDC	2.8 - 3.4VDC	3.0 - 3.6VDC
Transmit Current	45 mA @ 3.3VDC	215 mA @ 3.3VDC	265 mA typical
Receive Current	50 mA @ 3.3VDC	55 mA @ 3.3VDC	65 mA typical
Power-Down Current	<10 uA @ 25° C	<10 uA @ 25° C	45 uA pin Sleep
Regulatory Approvals			
FCC (USA)	OUR-XBEE	OUR-XBEEPRO	MCQ-XBEEXSC
IC (Canada)	4214A-XBEE	4214A-XBEEPRO	1846A-XBEEXSC
ETSI (Europe)	Yes	Yes* Max TX 10 mW	No
C-TICK Australia	Yes	Yes	No
Telec (Japan)	Yes	Yes*	No

802.15.4 - Star

(side views)



(top view) (top view)



Visit www.digi.com for part numbers.