Proyecto de ejemplo de AVR de baja potencia

◎ Objetivo

Esta página ilustra varios métodos para configurar un MCU AVR ® de 8 bits para que funcione con bajo consumo de energía. Con este ejercicio lograrás:

- Cree un proyecto y agregue un código simple para hacer parpadear un LED
- Ejecutar el proyecto y medir el consumo de energía
- Hacer modificaciones al código para reducir la potencia
- Ejecute el proyecto modificado y observe un consumo de energía reducido

☑ Materiales

Requisitos de Software

		Instaladores			
		ventanas	linux	Mac OS X	
Herramienta	Sobre				Instrucciones de instalación
Entorno de desarrollo integrado Atmel® Studio	(/atstudio:start)	(http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe)			(/install:atstudi

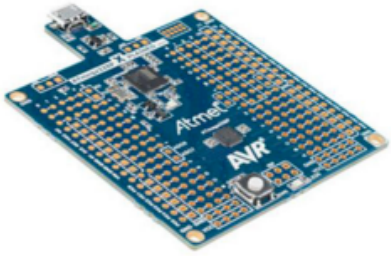
Requisitos de hardware

- ATmega328PB Mini tablero explicado
- Kit depurador de energía
- Dos cables micro USB
- Tres cables hembra a macho y un cable macho a macho

Herramienta		Sobre	
(http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)	Mini kit de evaluación ATmega328PB Xplained	(http://www.atmel.com/tools/MEGA328PB-XMINI.aspx)	(https://www.microchipdirect.co

ATmega328PB Tarjeta explicada

El **mini kit de evaluación ATmega328PB Xplained** es una plataforma de hardware para evaluar el microcontrolador Atmel ATmega328PB. NO se necesita un depurador externo para ejecutar estos ejercicios. El ATmega328PB tiene un depurador incorporado completamente integrado.



(/local--files/8avr:low-power-example/xplained-board.png)

Kit depurador de energía

Power Debugger es un depurador compatible con CMSIS-DAP que funciona con Atmel Studio v7.0 o posterior. El depurador de energía envía datos de depuración de aplicaciones y medidas de energía en tiempo de ejecución al visualizador de datos.

(/local--files/8avr:low-power-example/power-debug.png)

El Power Debugger tiene dos canales de detección de corriente independientes para medir y optimizar el consumo de energía de un diseño.
El canal 'A' es el canal recomendado para medir con precisión corrientes bajas.
El canal 'B' es el canal recomendado para medir corrientes más altas con una resolución más baja.

configuración de hardware

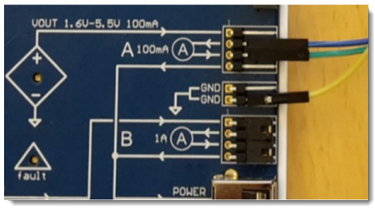
Conexión del depurador de energía a la placa Xplained ATmega328PB.

Los puertos de medición de corriente de los canales 'A' y 'B' en la placa Power Debugger se representan con símbolos de amperímetro en la serigrafía. El suministro de voltaje está conectado a la entrada del amperímetro y la carga (objetivo) está conectada a la salida. Con los siguientes pasos, el depurador de energía mide el consumo en el núcleo AVR.
Tabla 1-1 Power Debugger y **ATmega328PB Explicación de la** conexión Mini.

Power Debugger	ATmega328PB Xplained Mini
Port A input : Blue wire	5V
Port A output: Green wire	VCC
GND : Yellow Wire	GND

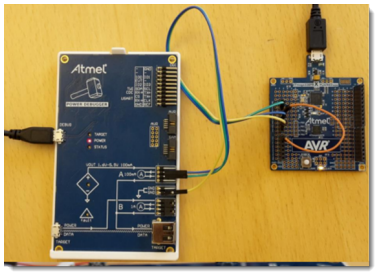
(/local--files/8avr:low-power-example/table.png)

Figura 1-1 Conexión de Power Debugger y **ATmega328PB Xplained Mini** .



(/local--files/8avr:low-power-example/connection.png)

Configuración de hardware



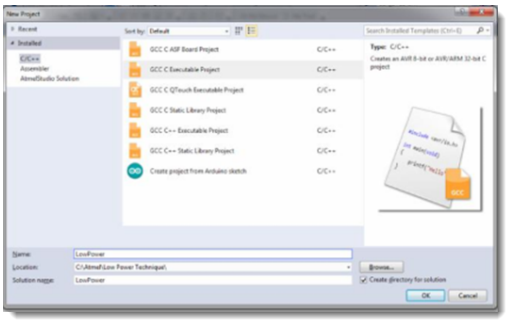
(/local--files/8avr:low-power-example/hw.png)

Medición de la corriente en modo activo

Procedimiento

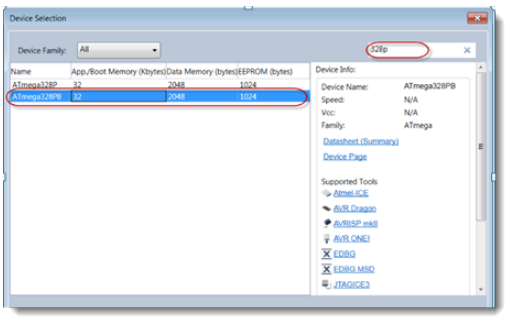
A Creación de proyectos

- Abrir Atmel Studio 7
- Seleccione **Archivo > Nuevo > Proyecto**
- Seleccione **GCC C Executable Project** y asígnele el nombre **LowPower** .
- Elija una ubicación para guardar el proyecto en su computadora.



(/local--files/8avr:low-power-example/location.png)

- Cuando aparezca la ventana de selección de dispositivo, ingrese 328P en la ventana de búsqueda y luego seleccione Atmega328PB. Haga clic en Aceptar.



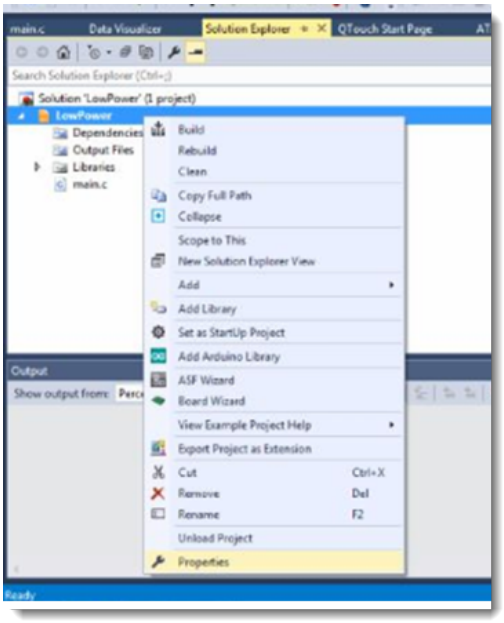
(/local--files/8avr:low-power-example/device-selection.png)

- Se creará un proyecto que contiene un `ciclo while(1)` vacío en `main()` .

```c
#include <avr/io.h>

int main(void)
{
    /* Replace with your appli
    while (1)
    {
    }
}
```

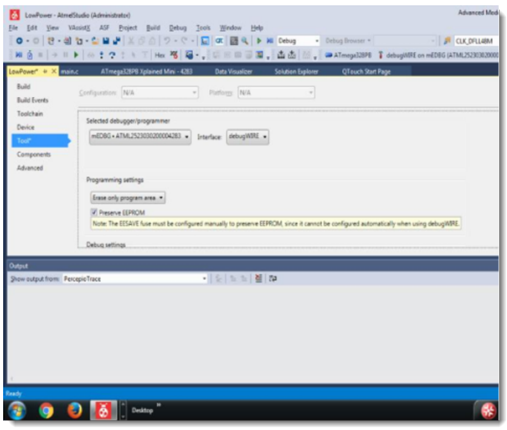
Seleccione **Ver > Explorador** de soluciones en la barra de menú.

En el Explorador de soluciones, haga clic con el botón derecho en el proyecto y seleccione **Propiedades** .



(/local--files/8avr:low-power-example/se.png)

En **Herramienta** , seleccione **mEDBG** y **debugWIRE**



(/local--files/8avr:low-power-example/debugwire.png)

B Agregar código al proyecto

- Agregue las siguientes dos declaraciones en `main()` para llamar a `TMR0_init` y establecer un pin de E/S como salida:

```
TMR0_init();

DDRB |= 1<<DDRB5;   // Direction o
```

Agregue la rutina de inicialización del temporizador 0

```
/*********************************
                                TM
*********************************
void TMR0_init( void ){

    // enable timer overflow inter
    TIMSK0=(1<<TOIE0)
    // set timer0 counter initial
    TCNT0=0x00;
    // start timer0 with /1024 pre
    TCCR0B = (1<<CS02) | (1<<CS00)

    // enable interrupts
    sei();

}
```

- Proporcione la rutina de servicio de interrupción TMR0 al proyecto y haga que el LED parpadee a aproximadamente 1 Hz.

```
uint8_t    count;
ISR(TIMER0_OVF_vect){

    count++;
    if(count==20){
        count=0;
        // Toogle LED
        PORTB=PORTB ^ 0x20;
    }
}
```

El programa completo es el siguiente,

```
#include <avr/io.h>
#include "avr/interrupt.h"

    uint8_t    count;
/* Timer 0 Interrupt */
ISR(TIMER0_OVF_vect){

    count++;
    if(count==20){
        count=0;
        // Toogle LED
        PORTB=PORTB ^ 0x20;
    }
}

int main(void)
{
    TMR0_init();

    DDRB |= 1<<DDRB5;   // Directi
    /* Replace with your applicati
    while (1)
    {
    }
}

/*********************************
                                TM
*********************************
void TMR0_init( void ){

    // enable timer overflow inter
    TIMSK0=(1 <<TOIE0) ;

    // set timer0 counter initial
    TCNT0=0x00;

    // start timer0 with /1024 pre
    TCCR0B = (1 <<CS02) | (1<<CS00

    // enable interrupts
    sei();

}
```

C Verificar las ejecuciones del proyecto

- Compile el proyecto y programe el dispositivo seleccionando **Depurar > Continuar** .

El LED parpadeará si el proyecto funciona correctamente

- Asegúrese de que la depuración del proyecto finalice seleccionando **Depurar > Deshabilitar debugWire** y luego **Cerrar** .

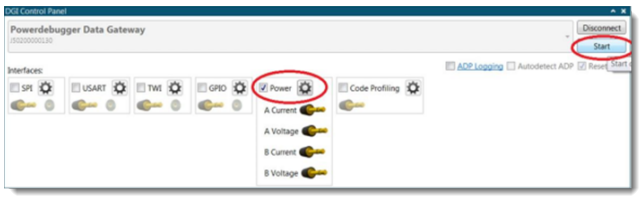
D Medir el consumo de energía en modo activo

- En Atmel Studio 7, abra el menú **Herramientas > Visualizador de datos**

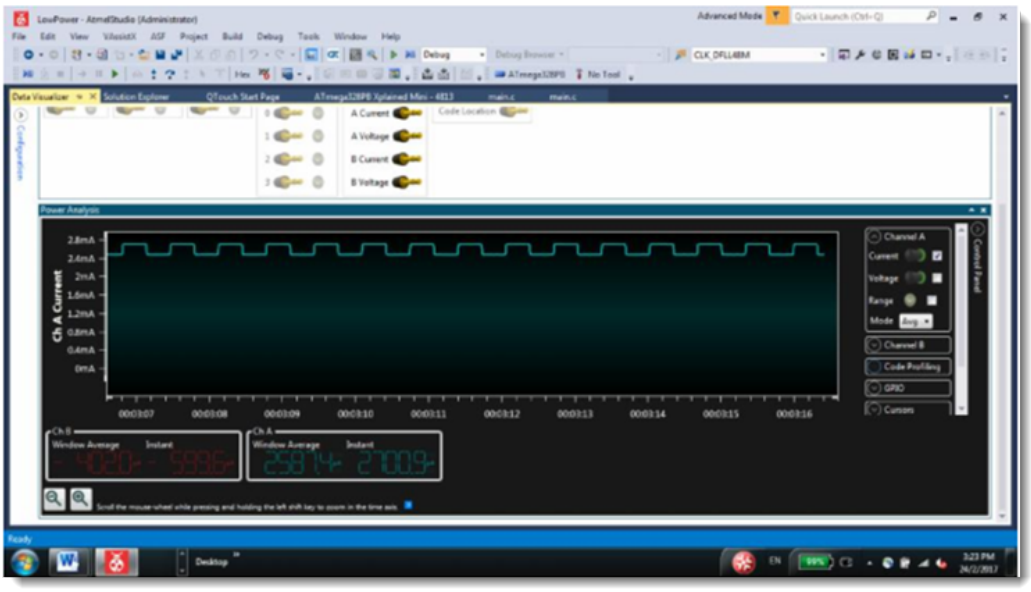
Power Debugger Data Gateway debe seleccionarse de forma predeterminada en el Panel de control de DGI; selecciónelo si no lo está. Haga clic en **Conectar** .

(/local--files/8avr:low-power-example/connect.png)

- Seleccione **Encendido** y **arranque**



(/local--files/8avr:low-power-example/power-and-start.png)



(/local--files/8avr:low-power-example/power-monitor.png)

Apague la fuente de alimentación de destino a la placa explicada y habilite la medición de energía

(/local--files/8avr:low-power-example/enable-power.png)

- Cierre la programación del dispositivo
- Verifique que el consumo medio de corriente sea de unos 2,6 mA

Reducir el consumo de energía

Existen varios métodos que puede utilizar para reducir el consumo de energía de un microcontrolador AVR® . Este ejemplo reduce el poder por:

- apagar los periféricos no utilizados
- Detener la corriente de fuga en los pines de E/S digital
- Uso del modo de suspensión. Las técnicas de optimización del consumo de energía se implementan en la función `Optimize_power_consumption()` que se llama desde `main()` .

Deshabilitar los búferes de entrada digital y el comparador analógico

Para los pines de entrada analógica, el búfer de entrada digital debe estar deshabilitado.

El búfer de entrada digital mide el voltaje en el pin y representa el valor como uno o cero lógico. Un nivel de señal analógica cercano a VCC/2 en un pin de entrada puede causar un consumo de corriente adicional significativo. Si el pin se usa como pin analógico, no hay necesidad de saber si el valor digital de la señal analógica sería uno o cero; estos pines digitales deben estar deshabilitados.

Apague los periféricos no utilizados

Deshabilite los periféricos no utilizados en la aplicación. El **registro de reducción de energía** (**PRR0**) y (**PRR1**) pueden detener el reloj de periféricos individuales para reducir el consumo de energía. Los recursos utilizados por el periférico permanecen ocupados cuando se detiene el reloj. En la mayoría de los casos, los periféricos deben desactivarse antes de que se detenga el reloj.

1. Incluya el archivo <power.h> en el programa principal **#include <avr/power.h>** 2. Agregue el código a continuación para `optimizar_el_consumo_de_energía()` .

```
/* Power shutdown to unused peri
PPR0 = 0xDF;
PPR1 = 0x3F;
```

3. Agregue e #include para <wdt.h> a main.c. **#include <avr/wdt.h>** 4. Agregue el siguiente código enoptimize_power_consumption `()` para desactivar el temporizador de vigilancia.

```
/* Disable interrupts*/
Cli();
Wdt_reset();
MCUSR&= ~(1<<WDRF);
```

Aplicar resistencias pull-up

Los pines no utilizados y desconectados consumen energía. La carga de energía innecesaria de los pines flotantes se puede evitar usando las resistencias pull-up internas del AVR en los pines no utilizados. Los pines del puerto se pueden configurar para ingresar pull-ups configurando el bit **DDxn** en 0 y el bit **PORTxn** en 1. (donde **x** es PORT B, C, D, E y **n** es 0 a 7)

```
1  /* Unused pins set as inpu
2       DDRB  &= 0xE0;
3       DDRC &= 0xC9;
4       DDRD &= 0x03;
5       DDRE &= 0xF3;
6
7       PORTB |= ~(0xE0);
8       PORTC |= ~(0xC9);
9       PORTD |= ~(0x03);
10      PORTE |= ~(0xF3);
```

Usar la función de suspensión

El modo de suspensión permite que la aplicación ahorre energía al apagar los módulos no utilizados. El AVR proporciona varios modos de suspensión que permiten al usuario adaptar el consumo de energía a los requisitos de la aplicación.

1. **Incluya el archivo de encabezado <avr/sleep.h>** de la biblioteca AVR en la parte superior del archivo.
2. Establezca el modo de suspensión deseado en la función Optimize_Power_consumption `()` configurando el microcontrolador para usar el modo de suspensión **SLEEP_MODE_PWR_DOWN** para un consumo mínimo de energía.

```
   /* Set sleep mode */
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
```

3. Llame a la `función sleep_mode()` desde `main()` para ingresar al modo de suspensión.

```
1                while (1)
2                   {
3                      sleep_mode();
4   }
```

Uso de la interrupción de cambio de pin

Para despertar el microcontrolador de **SLEEP_MODE_PWR_DOWN** , se utiliza la **interrupción de cambio de pin** . El interruptor de la **mini placa ATmega328PB Xplained** (SW0) está conectado a **PB7** . El pin **PB7** es la fuente de la interrupción por cambio de pin. Haz las siguientes adiciones a `main()` :

#include avr/interrupt.h

Configuración del bit PCINT7 y registro PCICR:

```
PCMSK0 |= (1<<PCINT7); //Enable I
   PCICR |= (1<<PCIE0); //Enable t
   sei();
```

Agregue la rutina de servicio de interrupción:

Para habilitar las rutinas ISR, el nombre del vector para PCINT7 es ISR (PCINT0_vect), definido en el archivo de encabezado del dispositivo.

```
   ISR (PCINT0_vect)
{
     if (!(PINB & (1<<PINB7))) //
    {
        PORTB |= (1 <<PORTB5); //
    }
    else
    {
        PORTB &= ~(1<<PORTB5); //
    }
}       }
```

Código completado

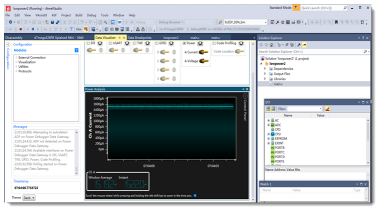
Después de realizar los cambios anteriores, el código completo debería ser como el siguiente:

```
/*
 * lowpower2.c
 */

//#include <avr/io.h>

#include <avr/io.h>
#include "avr/interrupt.h"
#include "avr/wdt.h"
#include "avr/sleep.h"

#include <avr/power.h>
#include <avr/interrupt.h>
//#include <util/delay.h>

void    optimize_power_consumption

    /* Disable digital input buffe
    DIDR0 = (1 << ADC5D) | (1 << A
    DIDR0 |= 0xC0 ;   /*ADC7D and

    /* Disable digital input buffe
    DIDR1 |= (1 << AIN1D) | (1 <<
    /* Disable Analog Comparator *
    ACSR |= (1 << ACD);

    /*Power shutdown to unused per
    PRR0 = 0xDF;
    PRR1 = 0x3F;
    /*Unused pins set as input pul
    DDRB &= 0xE0;
    DDRC &= 0xC9;
    DDRD &= 0x03;
    DDRE &=0xF3;

    PORTB |=~(0xE0);
    PORTC |=~(0xC9);
    PORTD |=~(0x03);
    PORTE |=~(0xf3);

    /*Watchdog Timer OFF*/

    /* Disable interrupts */
```

```
    cli();
    /* Reset watchdog timer */
    wdt_reset();
    /* Clear WDRF in MCUSR */
    MCUSR &= ~(1<<WDRF);
    /* Turn off WDT */
    WDTCSR = 0x00;

    /* Set sleep mode */
    set_sleep_mode(SLEEP_MODE_PWR_

}

/*********************************
                                TM
*********************************
 void TMR0_init( void ){

    // enable timer overflow inter
    TIMSK0=(1<<TOIE0) ;

    // set timer0 counter initial
    TCNT0=0x00;

    // start timer0 with /1024 pre
    TCCR0B = (1<<CS02) | (1<<CS00)

    // enable interrupts
    sei();

}
     uint8_t    count;
/* Timer 0 Interrupt */
 ISR(TIMER0_OVF_vect){

    count++;
    if(count==20){
        count=0;
        // Toogle LED
        PORTB=PORTB ^ 0x20;
    }
}

ISR (PCINT0_vect)
{
     if (!(PINB & (1<<PINB7))) //
    {
        PORTB |= (1<<PORTB5); // T
    }
    else
    {
        PORTB &= ~(1<<PORTB5); //
    }
}

int main(void)
{
    TMR0_init();

    DDRB |= 1<<DDRB5;   // Directi
    DDRB &= ~(1<<DDB7); //Set PORT

    optimize_power_consumption();

    PCMSK0 |= (1<<PCINT7); //Enabl
    PCICR |= (1<<PCIE0); //Enable
    sei();

    //set_sleep_mode(SLEEP_MODE_PW
    //sleep_enable();
    //sleep_cpu();

    /* Replace with your applicati
    while (1)
    {
        sleep_mode();
    }
}
```

Programe la aplicación y mida el consumo de energía.

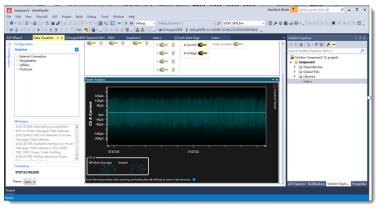
1. Programe el código seleccionando **Depurar > Continuar** .
2. Espere hasta que la aplicación esté programada y el mensaje en la parte inferior de la ventana aparezca como **En ejecución** .
3. Salga del depurador seleccionando **Depurar > Deshabilitar debugWire** y luego **Cerrar** .
4. Abra la **ventana** del Visualizador de datos y verifique el consumo de energía como se muestra a continuación.



(/local--files/8avr:low-power-example/one-five.png)

El consumo de corriente observado debería ser de alrededor de 1,52 mA.

- Establezca el interruptor de alimentación objetivo apagado: **Herramientas > Programación de dispositivos >** Configuración de herramientas



(/local--files/8avr:low-power-example/threema.png)

El consumo de corriente debería haberse reducido a aproximadamente 20,7 µA.

Tenga en cuenta que el temporizador no activa el dispositivo desde el modo SLEEP. La aplicación está configurada para salir del modo de suspensión cuando se produce la **interrupción de cambio de pin .**

Conclusiones

Este ejemplo demostró varias técnicas diferentes para reducir el consumo de energía de una aplicación AVR. El consumo de energía se puede reducir significativamente mediante:

- Diseño inteligente
- Uso de los modos de suspensión
- Desactivación de periféricos no utilizados

Este ejemplo usó el **visualizador de datos Atmel Studio 7** y la placa de **depuración de energía** para medir la energía.