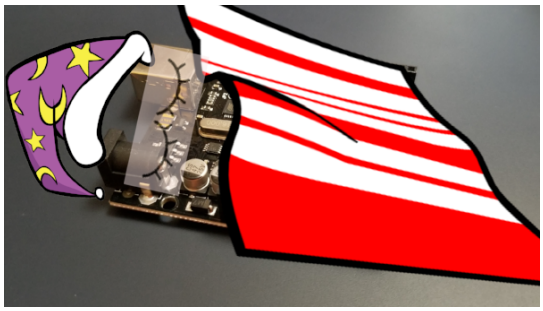
TUTORIAL:A GUIDE TO PUTTING YOUR ARDUINO TO SLEEP

JANUARY 26, 2018 · COMMENT

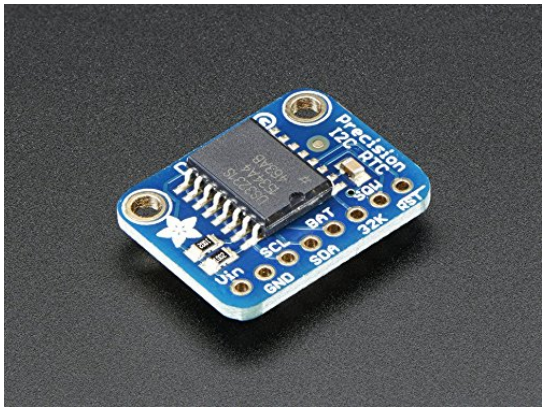
Please put some money in the tip jar by clicking on the donate button to support me so I can continue creating contend like this. P.S. please donate more than $1 as PayPal takes minimum $0.30 per transaction

Donate

Sometimes we are in a situation that requires us to put an Arduino in a place where plugging it in to the power grid is not an option. This happens often when we try to log information in a remote site, or only need to have your Arduino active at a specific interval/action.



In these cases putting your Arduino to sleep is the perfect thing to do. Their attention is only required for a short amount of time e.g. log data in a specific interval, or put out an alert when a predetermined event happens. In this tutorial we are going to experiment with putting your Arduino to sleep and see how to turn your Arduino back on.



Adafruit DS3231 Precision RTC Breakout

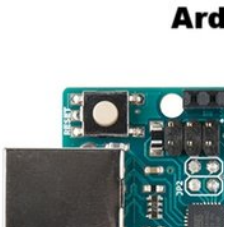
Adafruit

BUY ON AMAZON

This tutorial familiarizes you with the concept and has a small exercise to see what it takes to put an Arduino to sleep. In the next couple of blog posts (in 2 weeks or so) I will show post a couple a projects that will show you how to wake your Arduino using a sensor, or a Real Time Clock module (RTC).

Index

MATERIALS NEEDED IN THIS TUTORIAL



What board to use?

In this tutorial we will be using the Arduino Uno just because it is an easier board to prototype on. In a real live project I would use an Arduino Pro Mini for this. The Arduino Uno and the Arduino Pro Mini have very similar characteristics, the Arduino pro mini has a lot less hardware to power (e.g. the USB portion, extra leds, and some other stuff) thus using a lot less power. This is the reason why the Arduino Pro mini is a better choice.

To give an example a Uno uses between 30-40 mA when awake and about 19 mA when asleep. The Pro Mini uses 25mA when awake and 0.57 mA when asleep. As every mA matters when hooking it up to a battery you can see

> that there are no major differences.

Sleep mode

When you look at the documentation of the ATmega328p (click this link for a copy of this document) processor used for both Arduino Uno and the Arduino Pro mini you notice there are many different sleep modes available. But in a real world scenario there is really only one mode that is useful; The Power down mode (SLEEP_MODE_PWR_DOWN).

When you put your Arduino to sleep it turns off all unnecessary components, reducing the power consumption of the MCU (Microcontroller Unit). In this mode the only way you can wake it up is the use of an external influence (e.g. we give it a nudge to wake up). We will examine how to do this a bit later in this tutorial.

Interrupts

Before we go into the code to put an Arduino to sleep we need to understand the interrupt concept. The best way to describe it is ; You are working on something you really need to concentrate on. You wear headphones blasting your music loud to drown out your surrowndings . You are so concentrated on this that the outside world is lost to you. The only way to get your attention is by giving you a nudge. After you receive this nudge you pay attention to what the interruption is about, and after dealing with it you put the music back on and continue with your task.

> Note: I am not going to go to deep into what interrupts are good for, but if you want to learn more about this concept check out my tutorial (Using Interrupts to improve the functionality of your project) on this topic

Most true Arduino's have a couple of pins that do just that. The Uno and the Pro Mini have 2 pins (d2 and d3) that have the capability to interrupt what the Arduino is doing. With this we can nudge the Arduino back to a waking state.

Putting your Arduino To Sleep

You can download the code for this section for here.

Let's look at the code for this section

```
11 #include <avr/sleep.h>//this AVR library contains the methods that controls the sleep modes
12 #define interruptPin 2 //Pin we are going to use to wake up the Arduino
```

Image_1 click to enlarge

Image 1 contains the code snippet that loads the library that contains everything we need to put your Arduino to sleep, We also declare the variable interruptPin for digital pin 2. We will later use this for making pin 2 an input pin.

```
17  pinMode(LED_BUILTIN,OUTPUT);//We use the led on pin 13 to indecate when Arduino is A sleep
18  pinMode(interruptPin,INPUT_PULLUP);//Set pin d2 to input using the buildin pullup resistor
19  digitalWrite(LED_BUILTIN,HIGH);//turning LED on
20 }
```

Image_2 Click To Enlarge

Image 2 has the code for the Setup() function. It is all straight forward. We declare digital pin 13 as an output pin (LED_BUILTIN is a build in variable for digital pin 13 where an onboard led is connected to). We are using the LED as an indecator for when the Arduino is asleep (when LED is on Arduino is awake, when off the Arduino is asleep).

On line 18 we set digital pin 2 as an input pin. You notice we use INPUT_PULLUP instead of INPUT. By doing this we use the build-in pull-up resistor to prevent the pin from flopping between HIGH and LOW when nothing is attached to it (same thing you would do with a button).

Next we are going to the main loop() function;

```
22 void loop() {
23  delay(5000);//wait 5 seconds before going to sleep
24  Going_To_Sleep();
25 }
```

Image_3

On line 23 we put in a delay of 5 seconds before we call the Going_To_Sleep() function on line 24. This is just so you can see that the onboard LED is on to show your Arduino is awake, and the moment we call this the ***Going_To_Sleep()*** function the LED goes off to indecate the Arduino is asleep.

Next lets look at the Going_To_Sleep() function itself;

```
27 void Going_To_Sleep(){
28     sleep_enable();//Enabling sleep mode
29     attachInterrupt(0, wakeUp, LOW);//attaching a interrupt to pin d2
30     set_sleep_mode(SLEEP_MODE_PWR_DOWN);//Setting the sleep mode, in our case full sleep
31     digitalWrite(LED_BUILTIN,LOW);//turning LED off
32     delay(1000); //wait a second to allow the led to be turned off before going to sleep
33     sleep_cpu();//activating sleep mode
34     Serial.println("just woke up!");//next line of code executed after the interrupt
35     digitalWrite(LED_BUILTIN,HIGH);//turning LED on
36   }
```

Image_4 Click To Enlagre

On line 28 we call the sleep_enable() function which is part of the avr/sleep.h library. It enables us to put the Arduino to sleep, without calling it we can't put the Arduino to sleep. On line 29 we attach an interrupt to pin 2 as I explained in the Interrupt section.

Syntax

attachInterrupt (interrupt, ISR, mode);

happen to the digital pin to call the interrupt. In our case the pin needs to be pulled LOW (to GND).

On line 30 we set the set of sleep mode we want. In our case it goes and shuts down everything it can. On line 31 we turn off the LED, and on line 32 we wait a second to give the board the time to turn the led off. Next we actually put the Arduino to sleep with the sleep_cpu() function.

The code halts here until the interrupt is called. After waking up the Arduino will first execute the code in the ***wakeUp()*** function and then will continue with line 34 printing the wakeup message on the serial monitor, and on line 35 turning the LED back on.

```
38 void wakeUp(){
39   Serial.println("Interrrupt Fired");//Print message to serial monitor
40    sleep_disable();//Disable sleep mode
41   detachInterrupt(0); //Removes the interrupt from pin 2;
42 }
```

Image_5

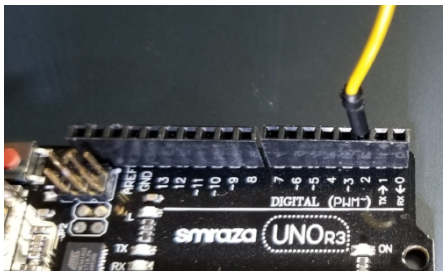
In the ***wakeUp()*** function we print a line to the serial monitor to let you now the interrupt has been called. The next two lines are very important to make sure we do not get an unintentional loop where the sketch can get stuck, and making your project fail.

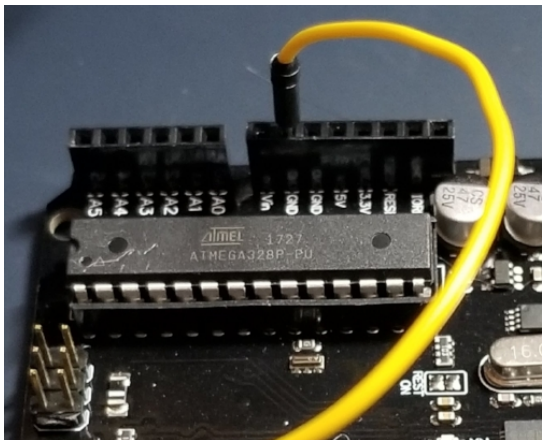
On line 40 we disable the sleep function, and on line 41 we detach the interrupt from pin 2. As I said we do this to prevent a possible endless loop situation.

Exercise 1

STEP 1)

Now it is time to upload the sketch. But before doing that put a jumper wire in d2. Just leave it unplugged on the other end for now. Load your sketch and wait 5 seconds for the LED to turn off and the Arduino to go to sleep.





STEP 2)

After the LED turns off insert the other end of the jumper wire in a GND pin on your Arduino Uno. This will pull pin 2 LOW triggering the interrupt, thus awaking the sleeping Arduino. After the LED comes back on you can remove the jumper wire out of GND and 5 seconds later the Arduino goes back to sleep.

IN CLOSING

Now you know the principles of what it takes to put your Arduino to sleep. As you see it is a very simple process. Finding a good mechanism to control this project is sometimes a bid more problematic. For this reason I will post a couple example projects on how to wake up an Arduino using a sencor (e.g. motion sensor), and a Real Time Clock module (RTC) that will be designed so you can use them and integrate that into your own project.

If you like this tutorial and would like to see more of the same type, please subscribe to my newsletter using the form below or like my Facebook page. This way you get notified when a new post is available. If you have questions or suggestions please email me or leave it in the comments below. Have a great day and see you next time.

Subscribe to our mailing list

* indicates required

Email Address *

First Name

Last Name

Email Format

html
text

Subscribe

PREVIOUS
Project: Example using a RTC to wake-up an Arduino Data Logger

NEXT
Tutorial: How to measure current consumption and why should you do it?

ABOUT