Arduino Sleep Modes Automatic and Manual to Save Power, Arduino Deep Sleep

Engr Fahad • November 25, 2020 0 9,276 14 minutes read

Table of Contents

1. Arduino Sleep Modes, Description:
2. Amazon Purchase Links:
3. Power Management and Sleep Modes:
4. Sleep Modes:
5. Arduino Idle Sleep Mode:
    5.1. Arduino Idle Sleep Mode Code:
    5.2. Increase the Arduino Sleep Time using the For Loop:
6. Add Noise Reduction Mode:
7. Power-Down Mode:
    7.1. Arduino Power-Down Sleep Mode Code:
    7.2. Arduino Code for Power-Down Interrupt Mode:
    7.3. Arduino Power-Down Sleep Mode Code using a sensor and external interrupt:
8. Power-Save Mode:
9. Standby Mode:
10. Extended Standby Mode:
11. Weather Station project:
    11.1. Battery Optimized Arduino Weather Station Code:
        11.1.1. Code explanation:
12. Arduino sleep mode – Waking up when receiving data on the USART
    12.1. Arduino wake up code using USART:
    12.2. Arduino Wake up from the idle mode using using USART:

Arduino Sleep Modes, Description:

Arduino Sleep Modes Automatic and Manual to Save Battery Power – I have been making Arduino based projects battery. Some Arduino projects you make might be needed to run outside or a long way from power lines. This at that point requires compactness making you depend on battery power, and potentially, solar based charging, making it a bit more confounded. In this article, we will examine strategies to lessen power utilization in the Arduino. We will make use of the LowPower.h library and also avr/power.h and avr/sleep.h to make a lowe-power system.

Power consumption is a basic issue for a device running constantly for quite a while without being turned off. So to beat this issue pretty much every controller accompanies a sleep mode, which help engineers to plan electronic devices for ideal power utilization. Sleep mode places the device in power saving mode by turning off the unused module.

Amazon Purchase Links:

12v Adaptor:

Arduino Uno

Arduino Nano

Mega 2560:

Other Tools and Components:

Super Starter kit for Beginners

Digital Oscilloscopes

Digital Multimeter

Soldering iron kits

PCB small portable drill machines

**Please Note: These are affiliate links. I may make a commission if you buy the components through these links. I would appreciate your support in this way!*

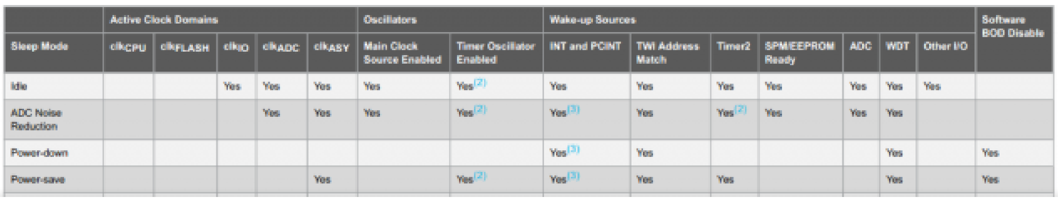
Power Management and Sleep Modes:

Sleep modes allow the application to close down unused modules in the microcontroller to save power. The Arduino boards are provided with different sleep modes allowing the users to tailor the power utilization to the application requirements. When the sleep mode is activated the Brown-out Detector in short BOD effectively monitors the power supply voltage during the sleep periods. If you want to further save power you can even disable the Brown-out Detector BOD in some sleep modes. The Brown-out Detector disable option is only available for the ATmega328P, this is the same microcontroller that is used in the Arduino Uno, Arduino Nano, and Pro-Mini. I will test these different Sleep modes on the Arduino Nano. You can also use the Arduino Uno. So, now let's take a look at the Sleep Modes.

Sleep Modes:

As per the AVR Microcontroller ATmega328P Datasheet there are a total of 6 sleep modes.

Idle Mode, ADC Noise Reduction Mode, Power-down Mode, Power-save mode, Standby Mode, and the Extended Standby Mode.



	Active Clock Domains					Oscillators		Wake-up Sources							Software BOD Disable
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT and PCINT	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	
Idle			Yes	Yes	Yes	Yes	Yes[2]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
ADC Noise Reduction				Yes	Yes	Yes	Yes[2]	Yes[3]	Yes	Yes[2]	Yes	Yes	Yes		
Power-down								Yes[3]	Yes				Yes		Yes
Power-save					Yes		Yes[2]	Yes[3]	Yes	Yes			Yes		Yes

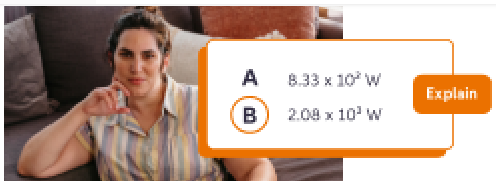
To can activate any of the 6 sleep modes, the Sleep Enable bit in the Sleep Mode Control Register (SMCR.SE) must be written to ‘1’ and a SLEEP instruction must be executed. Sleep Mode Select bits (SMCR.SM[2:0]) select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, Standby, or Extended Standby) will be initiated by the SLEEP instruction.

You can wake up the Arduino from the sleep mode using the Reset or you can use an internal or external Arduino Interrupts. I will explain this practically in the programming examples. For now we can focus on the six Sleep modes of the Arduino.

Arduino Idle Sleep Mode:

To activate the Idle Sleep Mode in the Arduino when the SM[2:0] bits are written to ‘000’. The Idle Sleep Mode stops the CPU but allow the SPI, USART, Analog Comparator, 2-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to remain functional. So, the Idle Sleep Mode as per the above table only halts the clk_{cpu} and clk_{FLASH}, while the other clocks run in the normal way.

You can wake up the Arduino from the Idle sleep Mode through

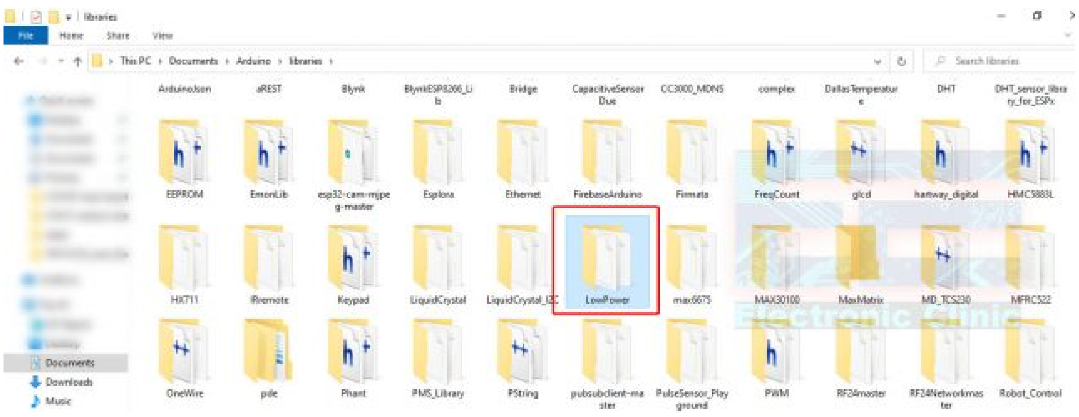


button or any sensor with the interrupt pin and wake up the Arduino. So, you can put Arduino to Sleep and waked it up at anytime using simple code. Take a look at the following code.

Idle Sleep Mode Arduino Code:

LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_OFF, USART0_OFF, TWI_OFF);

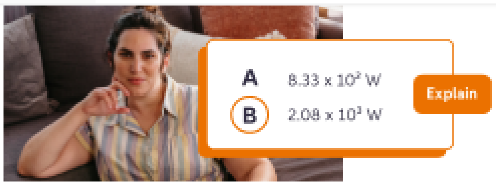
This is not the complete code but just an example how to use the idle sleep mode. To activate the Arduino's different low power modes you will need to download the LowPower library. After you down the LowPower library, simple extract the folder. Copy the LowPower folder and paste it into the Arduino libraries folder.



Now you can open the Latest version of the Arduino IDE.

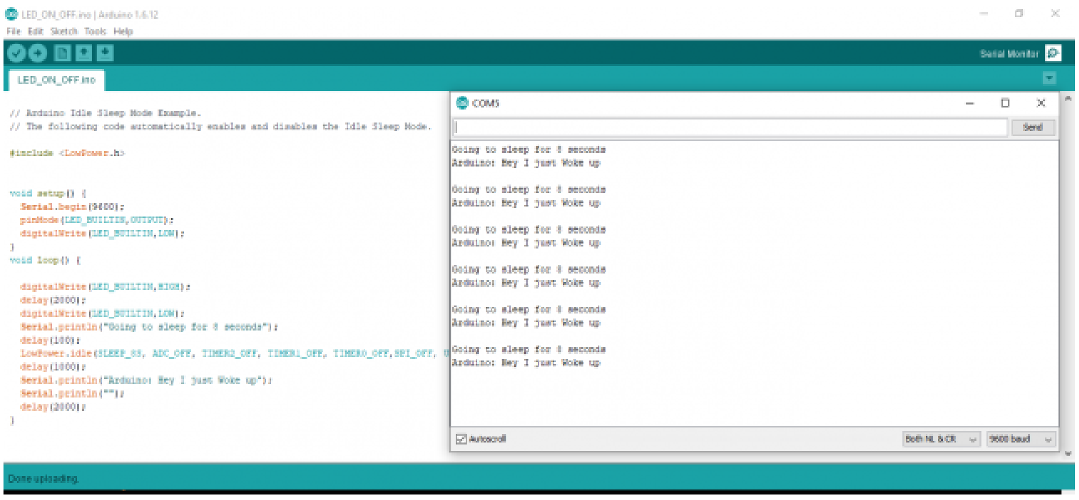
Arduino Idle Sleep Mode Code:

```
// Arduino Idle Sleep Mode Example.
// The following code automatically enables and disables the Idle Sleep Mo

#include <LowPower.h>


void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,LOW);
```



```
  digitalWrite(LED_BUILTIN,HIGH);
  delay(2000);
  digitalWrite(LED_BUILTIN,LOW);
  Serial.println("Going to sleep for 8 seconds");
  delay(100);
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_
  delay(1000);
  Serial.println("Arduino: Hey I just Woke up");
  Serial.println("");
  delay(2000);
}
```

Upload the code given above and open the serial monitor.



```
// Arduino Idle Sleep Mode Example.
// The following code automatically enables and disables the Idle Sleep Mode.

#include <LowPower.h>


void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,LOW);
}
void loop() {

  digitalWrite(LED_BUILTIN,HIGH);
  delay(2000);
  digitalWrite(LED_BUILTIN,LOW);
  Serial.println("Going to sleep for 8 seconds");
  delay(100);
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_OFF, U
  delay(1000);
  Serial.println("Arduino: Hey I just Woke up");
  Serial.println("");
  delay(2000);
}
```

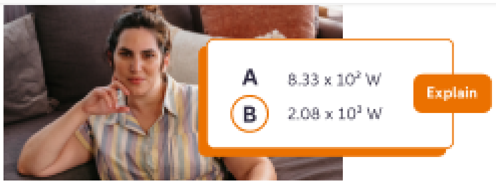
```
Going to sleep for 8 seconds
Arduino: Hey I just Woke up

Going to sleep for 8 seconds
Arduino: Hey I just Woke up

Going to sleep for 8 seconds
Arduino: Hey I just Woke up

Going to sleep for 8 seconds
Arduino: Hey I just Woke up

Going to sleep for 8 seconds
Arduino: Hey I just Woke up

Going to sleep for 8 seconds
Arduino: Hey I just Woke up
```

The purpose of the above program is to turn ON and turn OFF the LED and then enter the Arduino into sleep mode for 8 seconds. Then wakes up the Arduino automatically and then again the LED turns ON and turns OFF, this process continues forever. The above code is quite simple, it's just the LED blinking code, the only thing that you need to learn is

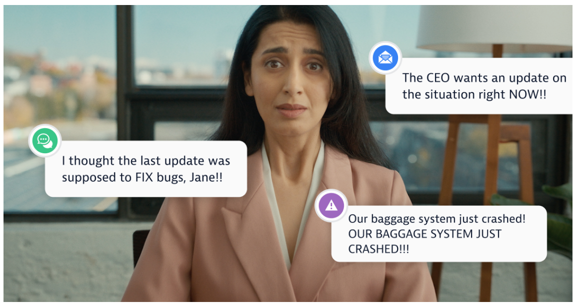
LowPower.idle(SLEEP_8S, ADC_OFF,TIMER2_OFF,TIMER1_OFF,TIMER0_OFF,SPI_OFF, USART0_OFF, TWI_OFF);

This line of code puts the Arduino to sleep for 8 seconds and also turns off the timers, SPI, USART, and the 2-wire interface TWI.

The argument SLEEP_8S put the Arduino to sleep for 8

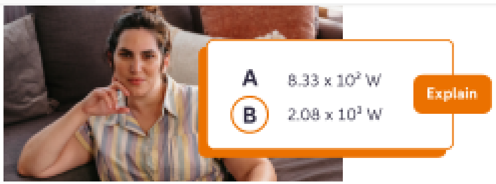


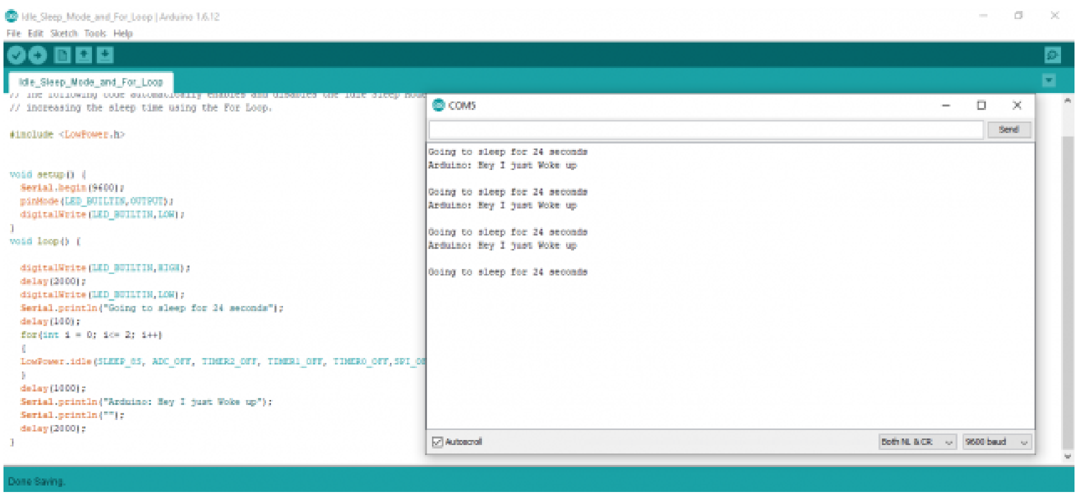
increase the sleep duration by using the FOR loop which I am going to demonstrate next.


The CEO wants an update on the situation right NOW!!
I thought the last update was supposed to FIX bugs, Jane!!
Our baggage system just crashed! OUR BAGGAGE SYSTEM JUST CRASHED!!!

Increase the Arduino Sleep Time using the For Loop:

```
// Arduino Idle Sleep Mode Example.
// The following code automatically enables and disables the Idle Sleep Mo
// increasing the sleep time using the For Loop.

#include <LowPower.h>



void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,LOW);
}
void loop() {

  digitalWrite(LED_BUILTIN,HIGH);
  delay(2000);
  digitalWrite(LED_BUILTIN,LOW);
  Serial.println("Going to sleep for 24 seconds");
  delay(100);
  for(int i = 0; i<= 2; i++)
  {
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_
  }
  delay(1000);
  Serial.println("Arduino: Hey I just Woke up");
```




Idle_Sleep_Mode_and_For_Loop | Arduino 1.6.12
File Edit Sketch Tools Help
Idle_Sleep_Mode_and_For_Loop
COM5
Send
Autoscroll
Both NL & CR
9600 baud
Done Saving.

```
// The following code automatically enables and disables the Idle Sleep Mod
// increasing the sleep time using the For Loop.

#include <LowPower.h>


void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,LOW);
}
void loop() {

  digitalWrite(LED_BUILTIN,HIGH);
  delay(2000);
  digitalWrite(LED_BUILTIN,LOW);
  Serial.println("Going to sleep for 24 seconds");
  delay(100);
  for(int i = 0; i<= 2; i++)
  {
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_O
  }
  delay(1000);
  Serial.println("Arduino: Hey I just Woke up");
  Serial.println("");
  delay(2000);
}
```

```
Going to sleep for 24 seconds
Arduino: Hey I just Woke up

Going to sleep for 24 seconds
Arduino: Hey I just Woke up

Going to sleep for 24 seconds
Arduino: Hey I just Woke up

Going to sleep for 24 seconds
```

This can be quite useful in reducing the power consumption. Let's say you can make Arduino sleep for 10 mints and then wake up the Arduino only for 10 to 20 seconds read the sensors values and then again go into the sleep mode. You can save a lot of power. You will be able to power up the Arduino using a battery for a longer period of time.

Add Noise Reduction Mode:

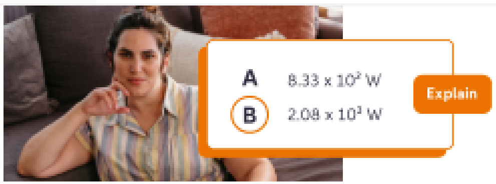
To activate the Add Noise Reduction Sleep Mode in the Arduino when the SM[2:0] bits are written to '001'. This mode also allows the CPU to stop, but allows the ADC, The external interrupts, the 2-wire Serial interface TWI, Timers/Counter, and the Watchdog to continue to operate. The ADC Noise Reduction Sleep Mode actually stops the $\text{clk}_{\text{I/O}}$, clk_{CPU}, and $\text{clk}_{\text{FLASH}}$, while it allows the other clocks to run.

We can wake up the Arduino from the ADC Noise Reduction Sleep Mode by using one of the following methods:

- External Reset
- Pin change interrupt
- Brown-out Reset
- Timer/Counter interrupt
- Watchdog System Reset
- Watchdog Interrupt
- 2-wire Serial Interface address match
- External level interrupt on INT
- SPM/EEPROM ready interrupt

Power-Down Mode:

To activate the Power-Down Sleep Mode in the Arduino when the SM[2:0] bits are written to '010'. The Power-Down Sleep Mode only allows the operation of asynchronous modules while stops all the generated clocks. In this Arduino sleep mode the external Oscillator is stopped, while allows the external interrupts, the 2-wire Serial interface address watch, and the Watchdog continue to operate.



- External Reset
- Pin change interrupt
- External level interrupt on INT
- Brown-out Reset
- Watchdog System Reset
- Watchdog Interrupt
- 2-wire Serial Interface address match

Arduino Power-Down Sleep Mode Code:

```
#include <LowPower.h>
void setup() {
   pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
  digitalWrite(LED_BUILTIN, LOW);
  LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
}
```

Pretty much the same code ;) except this time I am using the LowPower.powerDown. The first argument is the SLEEP_2S which means this will put the Arduino in sleep mode for 2 seconds. So, the purpose of this program is to turn on the LED then put the Arduino to sleep and then turn OFF the LED and again put the Arduino to Sleep. The same process repeats again and again.

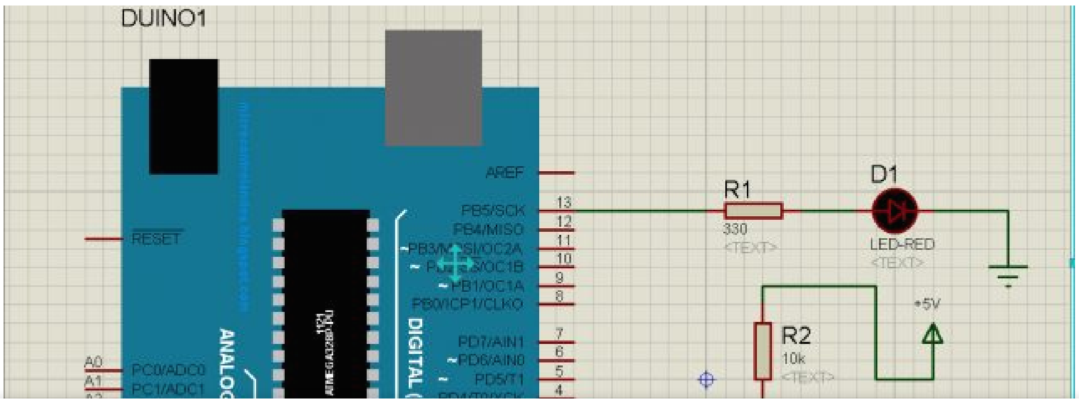
This is completely automatic, you don't need to do anything, it will automatically enter into the sleep mode and then automatically wakes up, a periodic process.

Let's take this to another level. What if we enter the Arduino into the sleep mode forever and then we can only wake it up using the external or internal interrupt. Let's do it.

Arduino Code for Power-Down Interrupt Mode:

```
void wakeUp()
{
  Serial.println("woke up");

}

void setup()
{
  Serial.begin(9600);
   pinMode(LED_BUILTIN, OUTPUT);
   pinMode(wakeUpPin, INPUT_PULLUP);
   attachInterrupt(digitalPinToInterrupt(wakeUpPin), wakeUp, FALLING);
}

void loop()
{
  Serial.println("Going to Sleep");
  delay(1000);
   LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
   delay(1000);
   digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
}
```

The purpose of this program is to toggle the LED between the ON and OFF states each time a signal is detected on the Arduino's pin number 2. You can connect a push button with this pin. This is such an amazing code. You press a button to change the LED state and then the Arduino enters into the sleep mode forever. The Arduino will remain in the sleep mode until you press the button connected with the Pin 2. Connect the button as per the circuit diagram given below.

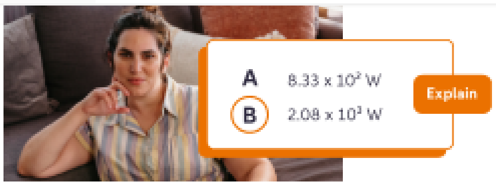


A 10k Pull-up resistor is connected with the pushbutton. When the button is open 5V is given to the pin 2 and when the pushbutton is pressed then GND is given as the signal. So each time the button is pressed an interrupt will be generated.

Let's take this to another level. Let's write a program that enters the Arduino into the sleep mode forever when the sensor values drops below 100. The Arduino then remains in the sleep mode forever until you press the button connected with pin 2 of the Arduino which generates an interrupt. We will use the same circuit connections except this time you can also add a Pot with the Analog pin A0.

Arduino Power-Down Sleep Mode Code using a sensor and external interrupt:

```
#include "LowPower.h"

const int wakeUpPin = 2;
int Pot = A0;
int PotData = 0;

void wakeUp()
{
  Serial.println("woke up");

}

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



```
    attachInterrupt(digitalPinToInterrupt(wakeUpPin), wakeUp, FALLING);
}

void loop()
{
  PotData = analogRead(Pot);
  Serial.println(PotData);
  if(PotData <  100)
  {
      Serial.println("Going to Sleep");
      delay(1000);
      LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
  }

  if(PotData >= 100)
  {
    digitalWrite(wakeUpPin, LOW);
    delay(100);
    digitalWrite(wakeUpPin, HIGH);
  }

}
```

When waking up from Power-Down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period.

Power-Save Mode:

To activate the Power-Save Sleep Mode in the Arduino when the SM[2:0] bits are written to '011'. The Power-Save Mode is identical to the Power-down sleep mode, with only one exception: If Timer/counter2 is enabled, it will keep running even during the sleep mode. Arduino can be waked up by using the timer overflow or the output compare event. If you are working on a project in which you don't need the timer/counter then my recommendation is to use the Power-down sleep mode instead of using the power-save mode.

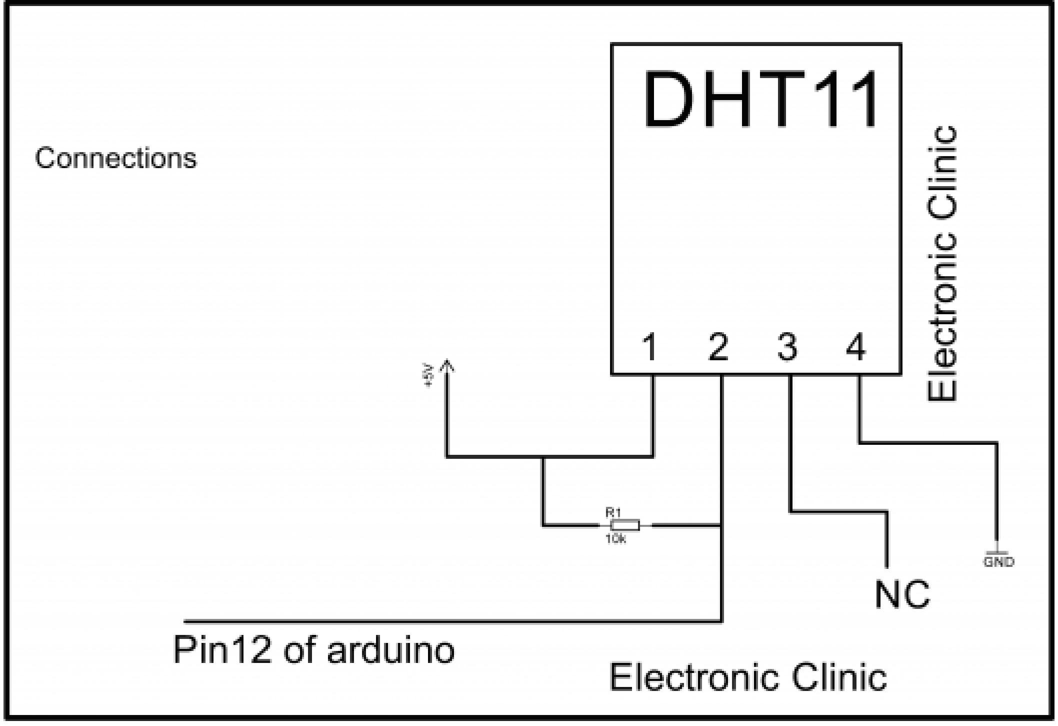
To activate the Standby Sleep Mode in the Arduino when the SM[2:0] bits are written to '110'. If you look in the table given above, you will find that the Standby mode is quite similar to the Power-down mode, the only difference between the two is that in the Standby mode the external Oscillator will keep running.

Extended Standby Mode:

To activate the Extended Standby Sleep Mode in the Arduino when the SM[2:0] bits are written to '111'. The Extend Standby Mode is identical to the Power-save sleep mode, but with only one exception that is the Oscillator will keep running. When in the Extended Standby Sleep Mode the Arduino will take six clock cycles to wake up.

Weather Station project:

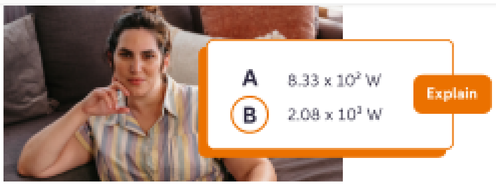
Now, let's make a battery operated Weather Station using the famous DHT11 Temperature and humidity sensor. The goal of this project is to save the battery power by using one of the Arduino Sleep modes. In this project we will not use any external hardware interrupts, but we will display the data on the serial monitor after every 8 seconds. Download the dht library.



These are the basic connections which I have already explained in my previous videos and articles.

Battery Optimized Arduino Weather Station Code:

```
/* Weather station example with sleep mode used to save the battery power.
 *
 */
#include <dht.h>
#include <LowPower.h>
#define dataPin 12
dht DHT;
void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,LOW);
}
void loop() {
  Serial.println("Get Data From DHT11");
  delay(1000);
  digitalWrite(LED_BUILTIN,HIGH);
  int readData = DHT.read11(dataPin); // DHT11
  float t = DHT.temperature;
  float h = DHT.humidity;
```



```
  Serial.print(" C | ");
  Serial.print("Humidity = ");
  Serial.print(h);
  Serial.println(" % ");
  delay(2000);
  Serial.println("Arduino: I am going to sleep");
  delay(200);
  digitalWrite(LED_BUILTIN,LOW);
  LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,
                SPI_OFF, USART0_OFF, TWI_OFF);
  Serial.println("Arduino: Hey I just Woke up");
  Serial.println("");
  delay(2000);
}
```

Code explanation:

#include <dht.h>

You will need the dht library for the dht11 sensor.

#include <LowPower.h>

The LowPower library I have already explained. This is used to activate any of the six Arduino Sleep modes.

#define dataPin 12

Dht11 Temperature and humidity sensor is connected with the Arduino pin 12

dht DHT;

void setup() {

Serial.begin(9600);

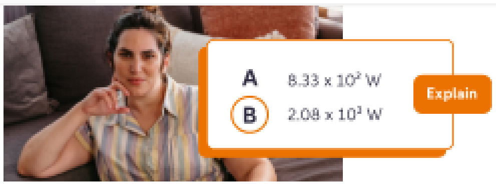
pinMode(LED_BUILTIN,OUTPUT);

digitalWrite(LED_BUILTIN,LOW);

}

```
void loop() {

Serial.println("Get Data From DHT11");

delay(1000);

digitalWrite(LED_BUILTIN,HIGH);

int readData = DHT.read11(dataPin); // DHT11

float t = DHT.temperature;

float h = DHT.humidity;



Serial.print("Temperature = ");

Serial.print(t);

Serial.print(" C | ");

Serial.print("Humidity = ");

Serial.print(h);

Serial.println(" % ");

delay(2000);
```

All the above instructions are used to read the temperature and humidity values, and then print the values on the serial monitor.

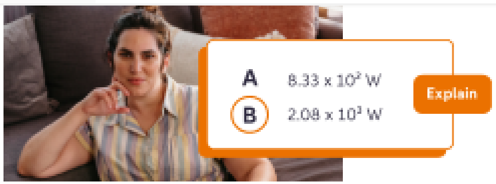


```
digitalWrite(LED_BUILTIN,LOW);

LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF,
TIMER1_OFF, TIMER0_OFF,

SPI_OFF, USART0_OFF, TWI_OFF);
```

This line of code you are already familiar with, which I already explained in very detail.

```
Serial.println("Arduino: Hey I just Woke up");

Serial.println("");

delay(2000);

}
```

This code will significantly reduce the power. You can even increase the sleep time using the for loop as already explained. You can also define a timer to keep track of the seconds, minutes, and hours. This way you can enter the Arduino into sleep mode for even hours.



Arduino sleep mode – Waking up when receiving data on the USART

Currently, I am working on the battery voltage monitoring system, the battery is charged using the Solar panels. I have designed an android app for this. In the previous examples I explained how to use the external hardware interrupts and how to use the automatic sleep and wakeup.

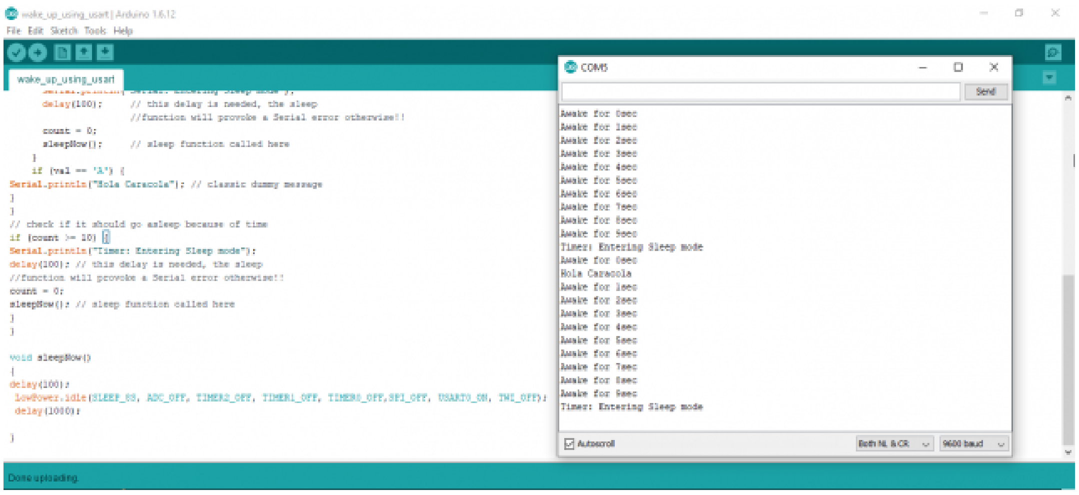
For this particular project I don't want the Arduino to send me data on regular intervals, and I also don't want to use the external hardware interrupts to wake up the Arduino, as my goal is to wireless monitor the Battery voltage using the Bluetooth and android cell phone application. I want to wirelessly wake up the Arduino, and then the Arduino send me the desired battery voltage reading.

In the example code below, I will try to keep things simpler so that you can get the whole idea and learn things quickly. We will write a program that will count the 10 seconds before the Arduino enters into the sleep mode. When the Arduino is in the sleep mode it can be waked up using the command "A" sent from the android app. The Arduino when receives the letter "A" it starts the counting and then again enters into the sleep mode. This way we won't need to go near the battery to wake up the Arduino. We can do everything wirelessly. This was we can save a lot of power as for the maximum time Arduino will be in sleep mode.

Arduino wake up code using USART:

```
#include <avr/power.h>
#include <avr/sleep.h>

int sleepStatus = 0; // variable to store a request for sleep
int count = 0; // counter

void setup()
{
```

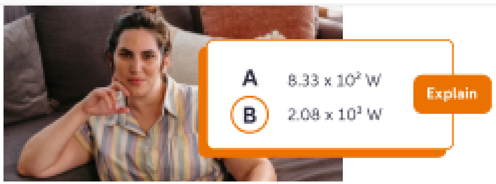
```
void sleepNow()
{
/*
 *
 * In the avr/sleep.h file, the call names of these sleep modus are to be
 *
 * The 5 different modes are:
 * SLEEP_MODE_IDLE -the least power savings
 * SLEEP_MODE_ADC
 * SLEEP_MODE_PWR_SAVE
 * SLEEP_MODE_STANDBY
 * SLEEP_MODE_PWR_DOWN -the most power savings
 *
 */

set_sleep_mode(SLEEP_MODE_IDLE); // sleep mode is set here

sleep_enable(); // enables the sleep bit in the mcucr register
// so sleep is possible. just a safety pin

power_adc_disable();
power_spi_disable();
power_timer0_disable();
power_timer1_disable();
power_timer2_disable();
power_twi_disable();

sleep_mode(); // here the device is actually put to sleep!!

// THE PROGRAM CONTINUES FROM HERE AFTER WAKING UP
sleep_disable(); // first thing after waking from sleep:
// disable sleep...

power_all_enable();

}

void loop()
{
// display information about the counter
count++;
delay(1000); // waits for a second

// compute the serial input
if (Serial.available()) {
int val = Serial.read();
if (val == 'S') {
      Serial.println("Serial: Entering Sleep mode");
      delay(100);     // this delay is needed, the sleep
                      //function will provoke a Serial error otherwise!!
      count = 0;
      sleepNow();     // sleep function called here
    }
    if (val == 'A') {
Serial.println("Hola Caracola"); // classic dummy message
}
}

// check if it should go asleep because of time
if (count >= 10) {
Serial.println("Timer: Entering Sleep mode");
delay(100); // this delay is needed, the sleep
//function will provoke a Serial error otherwise!!
count = 0;
sleepNow(); // sleep function called here
}
}
```



```
      delay(100);     // this delay is needed, the sleep
                      //function will provoke a Serial error otherwise!!
      count = 0;
      sleepNow();     // sleep function called here
    }
    if (val == 'A') {
Serial.println("Hola Caracola"); // classic dummy message
}
}
// check if it should go asleep because of time
if (count >= 10) {
Serial.println("Timer: Entering Sleep mode");
delay(100); // this delay is needed, the sleep
//function will provoke a Serial error otherwise!!
count = 0;
sleepNow(); // sleep function called here
}
}

void sleepNow()
{
delay(100);
 LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_OFF, USART0_ON, TWI_OFF);
 delay(1000);

}
```

```
Awake for 0sec
Awake for 1sec
Awake for 2sec
Awake for 3sec
Awake for 4sec
Awake for 5sec
Awake for 6sec
Awake for 7sec
Awake for 8sec
Awake for 9sec
Timer: Entering Sleep mode
Awake for 0sec
Hola Caracola
Awake for 1sec
Awake for 2sec
Awake for 3sec
Awake for 4sec
Awake for 5sec
Awake for 6sec
Awake for 7sec
Awake for 8sec
Awake for 9sec
Timer: Entering Sleep mode
```

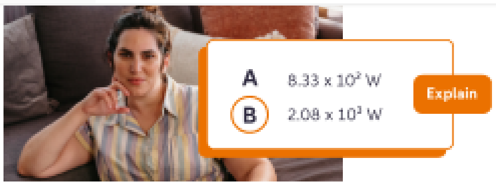
After uploading the above code open the serial monitor. The Arduino will count 10 seconds and then enters into the Sleep



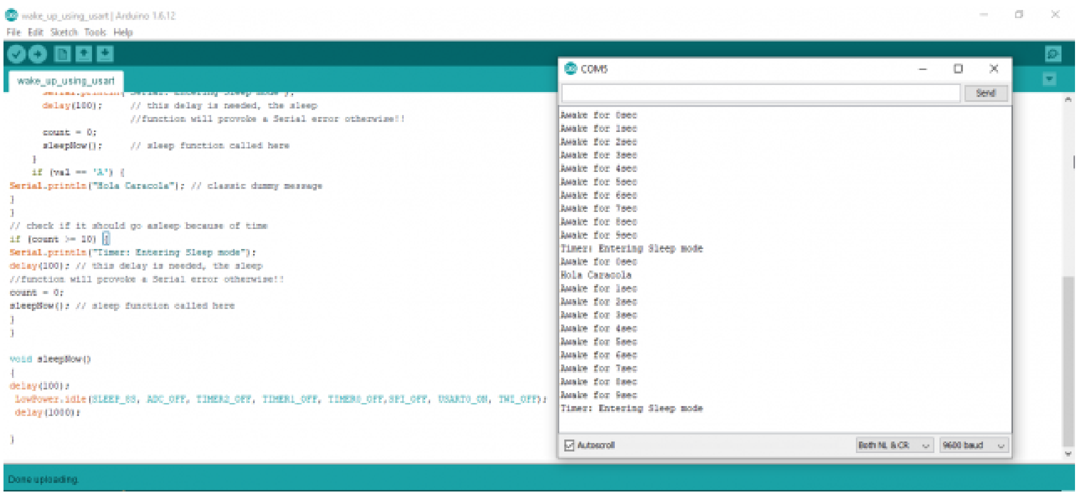
If it’s difficult for you to understand this code. You can try the following code. Which does the same exact thing.

Arduino Wake up from the idle mode using using USART:

```
// Arduino Idle Sleep Mode Example with wakeup using USART.
// this code enters the Arduino into sleep mode, and then remains in the s


#include <LowPower.h>

int sleepStatus = 0; // variable to store a request for sleep
int count = 0; // counter

void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,LOW);
}
void loop() {

// display information about the counter
Serial.print("Awake for ");
Serial.print(count);
Serial.println("sec");
count++;
delay(1000); // waits for a second
// compute the serial input
if (Serial.available()) {
int val = Serial.read();
```



```
                                //function will provoke a Serial error otherwise!!
        count = 0;
        sleepNow();     // sleep function called here
      }
      if (val == 'A') {
Serial.println("Hola Caracola"); // classic dummy message
}
}
// check if it should go asleep because of time
if (count >= 10) {
Serial.println("Timer: Entering Sleep mode");
delay(100); // this delay is needed, the sleep
//function will provoke a Serial error otherwise!!
count = 0;
sleepNow(); // sleep function called here
}
}

void sleepNow()
{
delay(100);
 LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_O
 delay(1000);


}
```

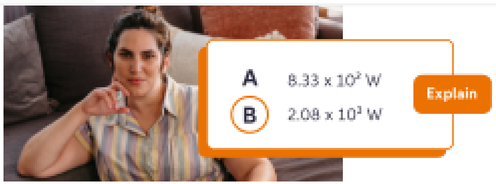


```
      delay(100);       // this delay is needed, the sleep
                        //function will provoke a Serial error otherwise!!
      count = 0;
      sleepNow();       // sleep function called here
    }
    if (val == 'A') {
Serial.println("Hola Caracola"); // classic dummy message
}
}
// check if it should go asleep because of time
if (count >= 10) {
Serial.println("Timer: Entering Sleep mode");
delay(100); // this delay is needed, the sleep
//function will provoke a Serial error otherwise!!
count = 0;
sleepNow(); // sleep function called here
}
}

void sleepNow()
{
delay(100);
 LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_OFF, USART0_ON, TWI_OFF);
 delay(1000);

}
```

```
Awake for 0sec
Awake for 1sec
Awake for 2sec
Awake for 3sec
Awake for 4sec
Awake for 5sec
Awake for 6sec
Awake for 7sec
Awake for 8sec
Awake for 9sec
Timer: Entering Sleep mode
Awake for 0sec
Hola Caracola
Awake for 1sec
Awake for 2sec
Awake for 3sec
Awake for 4sec
Awake for 5sec
Awake for 6sec
Awake for 7sec
Awake for 8sec
Awake for 9sec
Timer: Entering Sleep mode
```

This program also works the same way, but this program is quite simple as compared to the previous one.

In the previous examples



TWI_OFF);

I set the USART0_OFF as I was not using serial communication. But in this example

LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF,SPI_OFF, USART0_ON, TWI_OFF);

I set the USAT0_ON as in this particular example I am using the serial communication.

In the same way you can ON and OFF any of the arguments.

Tags arduino deep sleep arduino power save mode Arduino sleep arduino wake up using interrupt arduino wake up using usart deep sleep idle sleep mode lowpower library periodic sleep power save sleep mode sleep mode wakeup wakeup arduino