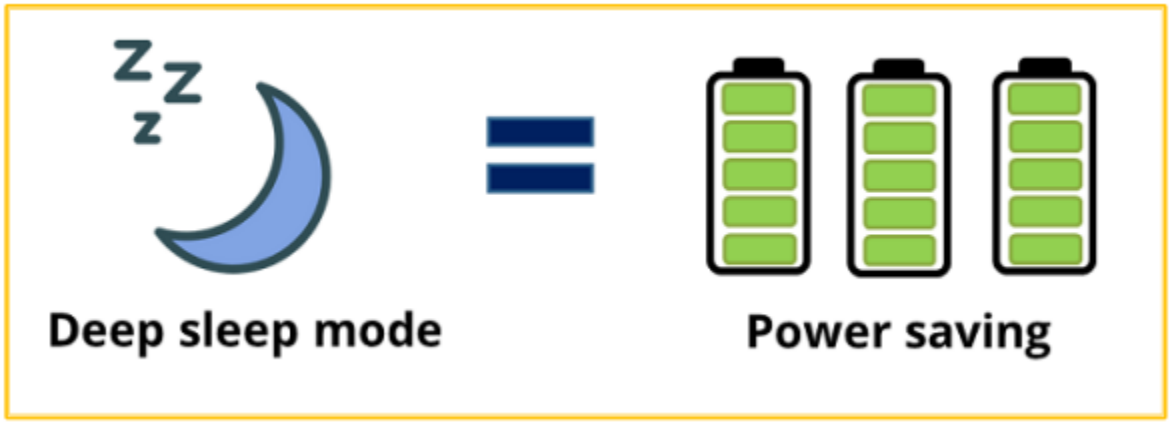
Al-Mustaqbal University
Department of Medical Instrumentation Techniques Engineering
Class: four
Subject: Advanced logic design
Lecturer: Dr. Zahraa hashim kareem
Second term / Lec.- 4: Arduino Power Saving Sleep Modes

Arduino Power Saving Sleep Modes



This lecture focuses on using power saving sleep modes on the Arduino. Six types of sleep modes but five of these are supported in the C header (avr/sleep.h):
1-Idle (SLEEP_MODE_IDLE)

2- ADC Noise Reduction (SLEEP_MODE_ADC)

3- Power-save (SLEEP_MODE_PWR_SAVE)

4- Standby (SLEEP_MODE_STANDBY)

5- Power-down (SLEEP_MODE_PWR_DOWN)

The most power saving sleep mode "Power Down" will be demonstrated in the tutorial. A current measurement demonstrates the power savings potential.

Goals:

- Understand how Arduino sleep modes work.
- Learn to use the "Power Down" sleep mode and an external interrupt as wake-up mode.

Idle mode: is the default mode where pretty much everything will wake the CPU and use power.

ADC noise reduction mode: disables the I/O clocks and is mainly useful for improving the performance of the MCU's on-board ADC.

Email: zahraa.hashim@uomus.edu.iq

Al-Mustaqbal University
Department of Medical Instrumentation Techniques Engineering
Class: four
Subject: Advanced logic design
Lecturer: Dr. Zahraa hashim kareem
Second term / Lec.- 4: Arduino Power Saving Sleep Modes

The last three modes are pretty similar in that they all disable pretty much everything the processor can do.

The power-save mode: allows the CPU to wake from a timer interrupt, so it's useful if you want the Arduino to periodically wake up and do some work before going back to the low power state.

The standby mode: keeps the main clock running so it can wake up more quickly (6 cycles) but it can only wake from external interrupts since it disables the timers.

In the power-down mode, the AVR disables everything except the external interrupts so

there's no other way to wake it and it takes a while to come back up — this is the lowest power

mode.

The sleep mode can be changed by including the avr-libc header avr/sleep.h in your project.
There are six functions in this header but you really only need to use two unless you're doing something fancy. So, to put the Arduino to sleep you can do something like this:

```
#include <avr/sleep.h>
void setup()
{
 set_sleep_mode(SLEEP_MODE_POWER_DOWN); // Set the sleep mode
 sleep_mode(); // Go to sleep
}
```

That's it. The Arduino will now sleep pretty much indefinitely since we didn't configure an interrupt or anything else that would wake it up. Once in power-down mode, the only things that will wake the CPU are:

Al-Mustaqbal University
Department of Medical Instrumentation Techniques Engineering
Class: four
Subject: Advanced logic design
Lecturer: Dr. Zahraa hashim kareem
Second term / Lec.- 4: Arduino Power Saving Sleep Modes

- An external reset
- A watchdog timer interrupt/reset
- A serial address match
- An external level interrupt
- A pin change interrupt

Once the CPU wakes it will continue execution from where it left off, servicing interrupts and so on until it hits another sleep instruction. Since my project has a big button on top I was able to use power-down mode and wake on a pin change interrupt.

Example

```
/****************************************
PURPOSE:    Tutorial on using the Power Down sleep mode of the ATmega328p
****************************************/
#include <avr/sleep.h>      //this AVR library contains the methods that controls the sleep modes
void setup()
{
  Serial.begin(9600);          //start serial communication at 9600 baud
  pinMode(13,OUTPUT);     // set pin 13 as an output so we can use LED to monitor
  digitalWrite(13,LOW);     // turn pin 13 LED off
  pinMode(2, INPUT);       //Set interrupt pin 2 as input
  digitalWrite(2,HIGH);     //activate the internal pull-up resistor to pull it high when
                            //jumper is not connected to GND
}
void loop()
{
 delay(5000); // Stay awake for 5 seconds, then sleep. These are the 5 seconds when we see ~40mA on the multimeter.
 sleepSetup();  //now we see ~27 mA until pin 2 is connected to GND
}
```

Email: zahraa.hashim@uomus.edu.iq

Al-Mustaqbal University
Department of Medical Instrumentation Techniques Engineering
Class: four
Subject: Advanced logic design
Lecturer: Dr. Zahraa hashim kareem
Second term / Lec.- 4: Arduino Power Saving Sleep Modes

Example

```
/*****************************************
PURPOSE:    Tutorial on using the Power Down sleep mode of the ATmega328p
*****************************************/
#include <avr/sleep.h>      //this AVR library contains the methods that controls the sleep modes
void setup()
{
  Serial.begin(9600);          //start serial communication at 9600 baud
  pinMode(13,OUTPUT);     // set pin 13 as an output so we can use LED to monitor
  digitalWrite(13,LOW);     // turn pin 13 LED off
  pinMode(2, INPUT);       //Set interrupt pin 2 as input
  digitalWrite(2,HIGH);     //activate the internal pull-up resistor to pull it high when
                            //jumper is not connected to GND
}
void loop()
{
 delay(5000); // Stay awake for 5 seconds, then sleep. These are the 5 seconds when we see ~40mA on the multimeter.
 sleepSetup();  //now we see ~27 mA until pin 2 is connected to GND
}
```

Al-Mustaqbal University
Department of Medical Instrumentation Techniques Engineering
Class: four
Subject: Advanced logic design
Lecturer: Dr. Zahraa hashim kareem


```
  sleep_enable();
  attachInterrupt(0, pinInterrupt, LOW);  //Set pin 2 as interrupt and attach Interrupt
                                          //Service Routine (ISR)
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);  //define power down sleep mode
  digitalWrite(13,LOW);                 // turn LED off to indicate sleep
  sleep_cpu();                  //Set sleep enable (SE) bit, this puts the ATmega to sleep
  Serial.println("just woke up!");    //When it wakes up due to the interrupt the
                        //program continues with the instruction following sleep_cpu()
  digitalWrite(13,HIGH);  // turn LED on to indicate wake up
}
void pinInterrupt()                 //ISR
{
sleep_disable();        //this is important. It is possible that the interrupt is called
                        //between executing "attachInterrupt(...)" and sleep_CPU() in the
                        //main loop if that happens without the sleep_disable() in the
                       //ISR, the ISR would be called, the interrupt detached and the
                       //device put to sleep. since the interrupt would be disabled at that
                       //point, there would be no way to wake the device up anymore.
                       //by putting sleep_disable() in the ISR, sleep_cpu() would not be
                       //effective during that loop, i.e. the main loop would run one
                       //more time and then properly attach the interrupt before hitting
                       //the sleep_cpu() a second time. At that point the device would
                      //go to sleep, but the interrupt would now be activated, i.e. wake-
                      //up can be induced.
 detachInterrupt(0);     //disable the interrupt. This effectively debounces the interrupt
                         //mechanism to prevent multiple interrupt calls.
}
```