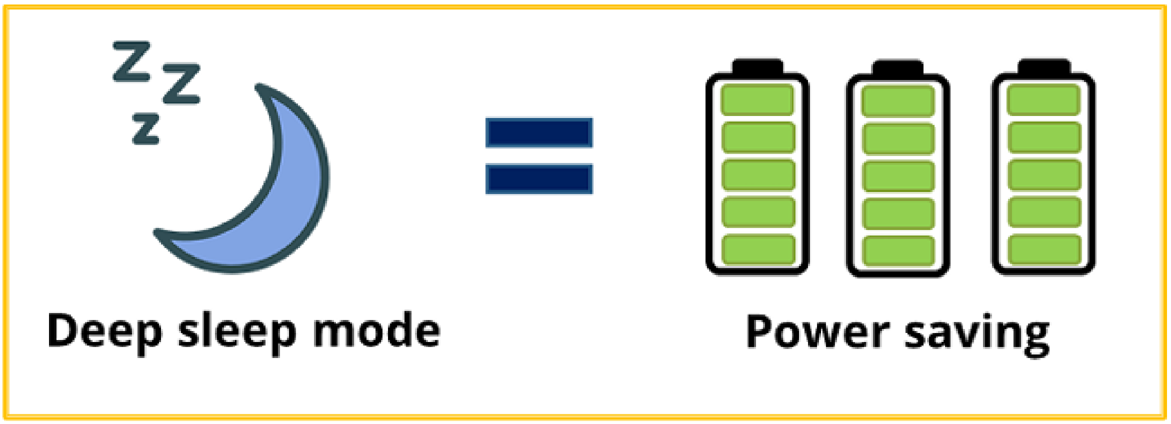
Al-Mustaqbal University
Department of Medical Instrumentation Techniques Engineering
Class: four
Subject: Advanced logic design
Lecturer: Dr. Zahraa hashim kareem
Lecture- 1:Arduino power saving mode

Arduino Power Saving Sleep Modes



Introduction: Arduino is widely used in embedded systems, IoT applications, and battery-powered projects. One of the major challenges in such applications is power consumption. To optimize power usage, Arduino provides various sleep modes that help extend battery life by putting the microcontroller into a low-power state when full operation is not required.

---

1. Understanding Power Consumption in Arduino: Microcontrollers consume power based on their operating states, including active, idle, and sleep modes. Reducing power consumption is crucial for battery-powered applications. Factors affecting power consumption include:

- Operating voltage
- Clock speed
- Peripheral usage (LEDs, sensors, communication modules, etc.)

---

2. Types of Sleep Modes in Arduino: Arduino, particularly the ATmega328P-based boards (e.g., Arduino Uno), offers several sleep modes:

1. **Idle Mode:**
   - CPU is halted, but peripherals like timers and communication interfaces remain active.
   - Suitable for applications needing continuous peripheral operation.
2. **ADC Noise Reduction Mode:**
   - Reduces noise while performing Analog-to-Digital Conversions (ADC).

- Only ADC and external interrupts remain active.

3. **Power-down Mode:**
    - Turns off most of the system, including the main clock.
    - Only external interrupts (INT0, INT1) or a watchdog timer can wake up the system.
    - Ideal for deep sleep applications with minimal wake-up requirements.
4. **Power-save Mode:**
    - Similar to power-down mode but keeps the Timer2 running for periodic wake-ups.
    - Useful for applications needing periodic tasks.
5. **Standby Mode:**
    - The oscillator continues running, allowing faster wake-up times.
    - Consumes slightly more power than power-down mode.
6. **Extended Standby Mode:**
    - Similar to standby mode but allows more flexibility with sleep timers.

---

3. Implementing Sleep Modes in Arduino: To implement sleep modes, the **Arduino Low Power Library** or direct register manipulation can be used.

Example: Entering Power-down Mode with External Interrupt Wake-up

```
#include <avr/sleep.h>
#include <avr/interrupt.h>
void wakeUp() {
  // Interrupt Service Routine (ISR) to wake up from sleep mode
}
void setup() {
  pinMode(2, INPUT_PULLUP);  // Set external interrupt pin
  attachInterrupt(digitalPinToInterrupt(2), wakeUp, LOW);
}
void loop() {
```

```
set_sleep_mode(SLEEP_MODE_PWR_DOWN); // Set power-down mode

sleep_enable();  // Enable sleep mode

sei();  // Enable global interrupts

sleep_cpu(); // Enter sleep mode

sleep_disable(); // Disable sleep after wake-up

}
```

4. Strategies for Power Optimization:

- **Reduce Clock Speed:** Lowering the microcontroller clock speed reduces power consumption.
- **Disable Unused Peripherals:** Turn off modules like ADC, SPI, or Serial when not needed.
- **Use Efficient Power Supplies:** Choose low-dropout regulators or direct battery connections.
- **Optimize Code Execution:** Reduce unnecessary delays and loops that keep the CPU active.

---

5. Applications of Sleep Modes:

- Battery-powered IoT devices (e.g., remote sensors, weather stations)
- Wireless communication modules (e.g., LoRa, Bluetooth, Zigbee nodes)
- Data loggers with periodic wake-ups
- Wearable electronics

Conclusion: Power-saving sleep modes in Arduino are essential for energy-efficient applications. By utilizing different sleep states, disabling unused peripherals, and optimizing software routines, developers can significantly extend battery life while maintaining functionality. Understanding these techniques allows for the creation of sustainable and low-power embedded systems.

H.W/

1- compare between microprocessor and microcontroller.

2- Give me more application about **Sleep Modes.**

Email: zahraa.hashim@uomus.edu.iq