POWER OPTIMIZATION - HOW LOW CAN YOU GO?
Goal

• Designing for low power in both hardware and software
• How to calculate power consumption, and theoretical battery life
• Implementing sleep mode on WildLogger
  • use RTC alarm
  • disable peripherals in sleep mode
  • protect against brownouts

Lab

• Install the LowPower Arduino library
• Implement a command to put the WildLogger into sleep mode
• Use the pushbutton to wake WildLogger out of sleep mode
Why Is This Important?

• After functionality or accuracy, battery life usually most important attribute
• Cost of deployment usually outweighs cost of actual device
• Longer field devices can stay in the field:
  • more data collected
  • reduce costs
Power Consumption

• Power consumption is a property of complete system ie. hardware and software
  • Power usage of electronic components eg. MCU
  • Power requirements/usage of peripherals eg. sensors, LCD, SD cards
  • Different layers of software eg. firmware and application software
• Determined by interaction between all these elements
Power Optimisation

- Usually power optimise devices with batteries
- Discussed in terms of amps (A), milliamperes (mA) or microamperes (µA)
- Batteries capacity rated in amp-hours (Ah)
- Power optimization takes place in both hardware and software
WildLogger Power Optimisation Strategy

- Keep device into sleep mode most of the time
- Only wake up when need to do something important – collect, save or send data.
- Device has two states:
  - Awake / Active mode (1% of the time)
  - Sleep / Low Power mode (99% of the time)
Approach

• Determine battery life requirements
• Hardware Optimisations
  • Choose sensors and components that can go into low power mode
• Software Optimisations
  • Program device to stay in sleep mode as long as possible
• Calculate average power consumption of device. 3 ways:
  • Take representative sample of data
  • Calculate average power consumption and drain
  • Extrapolate to estimate usable battery life
• Can we meet our battery life requirements?
Power Optimisation on WildLogger

Battery Life Requirement
• As long as possible

Hardware Optimizations
• AVR-based ATmega 1284P – supports deep sleep mode
• PIR Motion Sensor - consumes 0.060 mA power
• DHT11 – consumes 0.100 mA
• No LCD, SD consumes a lot of power.
  • For more info, check this link: http://bit.ly/microsd-power

Software Optimisations
• Put into sleep mode, wake up with RTC interrupt
• Add watchdog timer to check if the system has ‘hung’ or crashed
## Microcontroller Power Modes

<table>
<thead>
<tr>
<th>Sleep Mode</th>
<th>Active Clock Domains</th>
<th>Oscillators</th>
<th>Wake-up Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clkCPU</td>
<td>clkFLASH</td>
<td>clkIO</td>
</tr>
<tr>
<td>Idle</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ADCNRM</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Power-down</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Power-save</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Standby^{(1)}</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Extended Standby</td>
<td>X^{(2)}</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Note: X indicates the power state is active in that mode.
Disable All Peripherals

LED lights – turn off when deploying
Measuring Current

![Diagram showing a multimeter measuring current](image)

The multimeter reads 23.24 mA (milliamps).
Example Battery Life Estimation

Example:
Energy Capacity of 3 x AA = 2000 mA-hours
Duty cycle: 99% sleep, 1% active
Sleep current = 1 mA
Active current = 30 mA
Average Current = \((1 \text{ mA} \times 0.99) + (30 \text{ mA} \times 0.01)\) = 1.29 mA
Max theoretical battery life = 1550 hours
\[= 64 \text{ days}\]
\[\sim 2 \text{ months}\]
Wildlogger Battery Estimation

Energy Capacity of 3 x AA = 2000 mA-hours

Duty cycle: 99.998% sleep, 0.002% active

Sleep current = 0.25 mA (0.18 mA + 0.06 mA with PIR sensor added)

Active current = 30 mA

Average Current = (0.25 mA * 0.99998) + (30 mA * 0.00002) = 0.251 mA

Max theoretical battery life = 7,968 hours

= 332 days

~ 11 months

Battery Life (hours) = \frac{\text{Energy Capacity (mA-hours)}}{\text{Average Current (mA)}}
Battery Requirements met?

Yes! We’re a genius

- Adjust battery life requirements
e.g. more frequent trips out

No! We’re still a genius

- Adjust reading frequency (if feasible)

- Add extra battery capacity
  - add batteries (in parallel)
  - add solar capacity

eg. more frequent trips out
Increase Battery Capacity

- Add batteries in parallel to increase battery capacity (adding batteries in a series increases voltage)
WildLogger Brownout Reset

5.0V

1.8V

MCU Reset
Battery Life Optimization

Functionality

Frequency

Battery Capacity
What Do I Need to Know?

- Install Low Power library for Arduino

  - LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_ON);
  - SLEEP_FOREVER: stay in sleep mode forever or until interrupt occurs
  - ADC_OFF: turn off ADC
  - BOD_ON: keep brownout detection on
Installing LowPower Library
```c
#include <cmdArduino.h>
#include <LowPower.h>

#define DEBOUNCE_DELAY 200

int intpButton = 2;
volatile int flagButton = false;

void setup()
{
    cmd.begin(57600);
    Serial.println("Lab 5a - Power Management - How Low Can You Go");
    attachInterrupt(intpButton, isrButton, FALLING);
    cmd.add("sleep", cmdSleep);
}

void loop()
{
    cmd.poll();
    if (flagButton)
    {
        delay(DEBOUNCE_DELAY);
        flagButton = false;
        Serial.println("The button was pushed.");
    }
}
```
void cmdSleep(int argCnt, char **args)
{
    Serial.println("Going to sleep now...zzzz");
    delay(100);
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_ON);
    Serial.println("Waking up now....");
}

void isrButton()
{
    flagButton = true;
}
Lab 5a - Power Management - How Low Can You Go

CMD: Command not recognized.

************* CMD *************

CMD -> sleep
Going to sleep now...zzzz
Waking up now....

************* CMD *************

CMD -> The button was pushed.
COMING UP

Module 5.1: Who Let the Watchdogs Out!