Research Inventy: International Journal of Engineering And Science
Vol.12, Issue 12 (December 2022), PP 33-39
Issn (e): 2278-4721, Issn (p):2319-6483, www.researchinventy.com

Power Saving Techniques for IoT Devices

Saša Salapura
[1] *Associate Professor, Faculty of Computer Science, PIM University, Banja Luka, Bosnia and Herzegovina*

Abstract: *We witness an increased number of IoT devices in almost every aspect of life. These devices are expected to work for many years, on their own and without any slowdowns. The key goal for devices that are completely battery-operated is energy efficiency and reducing that consumption to the minimum. In this paper, I will describe different techniques of consumption reduction of IoT devices that can be successfully applied in real projects. The described techniques reduce and take care of the consumption of IoT devices. The following techniques will be covered: sleep mode, the change of working effort and frequency, disabling, and not using intern modules.*

Keywords: *IoT, low power, deep sleep, lower voltage, lower frequency.*

Date of Submission: 04-12-2022 Date of Acceptance: 16-12-2022

I. INTRODUCTION

We are surrounded by IoT devices that are expected to work interrupted for a long period of time. When designing, special attention is given to devices that operate entirely on batteries. Besides microcontrollers/processors, IoT devices contain both elements for measuring some of the physical quantities (sensors) as well as executing elements (actuators) which is why it is necessary to carefully design each of these subassemblies of the device, keeping in mind minimal consumption.

There are different techniques for reducing the consumption of IoT devices: Techniques that reduce the consumption of the processor by putting it into sleep mode (extremely small consumption mode), or techniques that lead to a noticeable reduction of the consumption by changing working effort and/or changing the frequency on which the processor operates. The processor itself contains several internal functioning modules which can be turned off when necessary, and therefore disabling and not using the intern models can be used as another technique for the reduction of consumption. Techniques for reducing the consumption of IoT devices could generally be divided into techniques based on actions on the existing hardware and techniques based on changes in the software. If these techniques are applied to specific processors (as with the processor ATMega328P) it is possible to achieve a very low consumption, as stated in theFigure *1*.

Figure 1ATMega328P: Power-Down Supply Current vs. Vcc Watchdog Timer Disabled (Origin: Figure 31-342) [1]

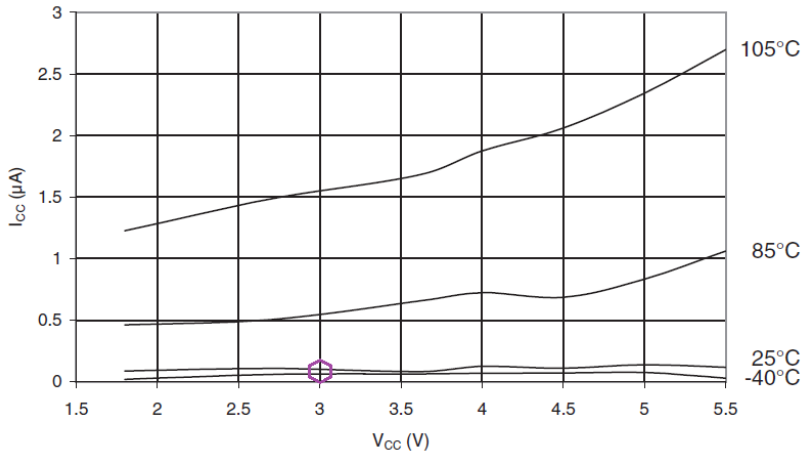


Figure 1 shows the consumption of 0.1uA (100nA), in the conditions of working at room temperature and with a power supply stress of 3V. This consumption is significantly below the power of self-discharging of most rechargeable batteries (Li-ion battery, self-discharge rate 0.3-2.5% per month [6]).

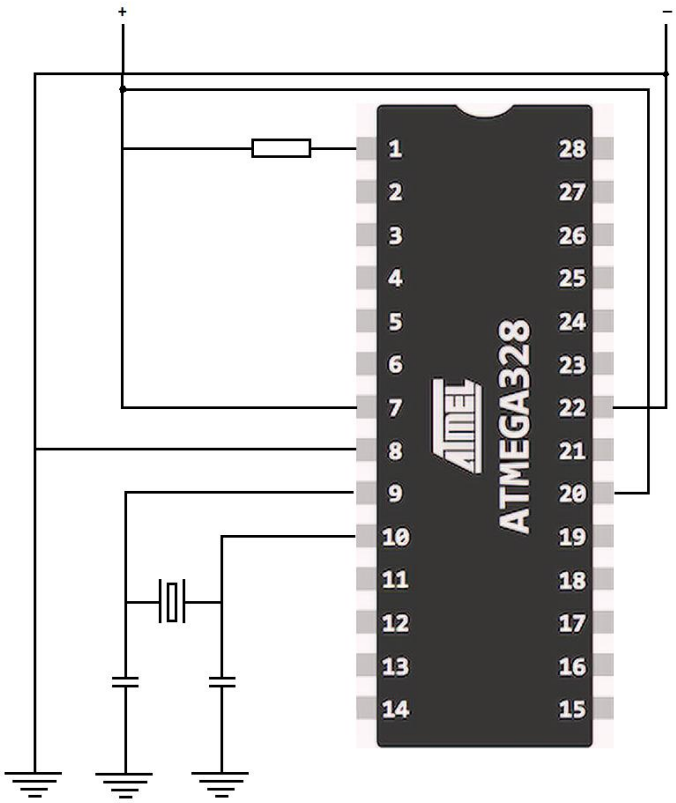
In this paper, the following techniques will be used: voltage lowering, clock speed reduction as well as power saving with software. All tests are done on the ATMega 328P processor only, without the parts that make up the Arduino experimental board (the USB port, the on-board serial to USB converter, voltage regulators, etc.) that is, after removing extra hardware. [***Figure 2***]

II. TESTING PROCEDURE

Barebone board

All tests were done on the barebone board PBC which is a cost-effective and miniaturized alternative to Arduino experimental board. Basically, it is an ATMega328P processor only, without all the other parts that make up the Arduino experimental board (the USB port, the on-board serial to USB converter, voltage regulators, etc.) that is, after removing extra hardware [***Figure 2***].

Figure 2 ATMega328P barebone (minimum circuit) [3]



Understanding battery self-discharge

Self-discharge rates are faster at the start and depend on the ambient temperature, type of battery, and battery technology. Self-discharge is considerably increased at temperatures above +55°C. Batteries will reach these temperatures when left in storage rooms during hot summers [7].

The following *Table 1* table shows typical shelf life and self-discharge rates for common rechargeable cells.

Table 1 Typical shelf life and self-discharge rates for common rechargeable cells [7]

Typical shelf life		Typical self-discharge rates	
Zinc Carbon	2 - 3 years	Lead Acid	4%-6% /month
Alkaline	5 years	Nickel Cadmium	15%-20% /month
Lithium	10+ years	Nickel Metal Hydride	30% /month
		Lithium	2%-3% /month

The self-discharge rate of X% per month can often be found along with the battery capacity. This means that an unused battery with capacity Q (mAh) and self-discharge rate X% per month will lose Q * X/100 of capacity after one month.

For example, a battery of capacity Q = 3000 mAH and self-discharge rate (monthly discharge rate) X=3% will lose 90mAh every month, or 0.125 mAH every hour – same with current consumption of 0.125 mA for an hour: the self-discharge rate is effectively the same as using 125 µA continuously [Equation 1]! It is smart to keep the current consumption of the device in sleep mode at that level.

Equation 1: Formula for self-discharge rates

$$I_{cc}\,[mA] = \frac{X\,[\%]}{100} \times \frac{C\,[mAh]}{24*30}$$

IoT devices usually have a periodic measurement mode that sends the measured values. This working mode is used for measuring river flood stage or temperature and infrared radiation level. It measures the values and sends the data every 30 or 60 minutes. It takes only 1-2 seconds to measure and send the measured values, while the rest of the time, the processor and the sensors can stay in deep sleep mode.

The average consumption of this kind of device can be measured with the Equation2:

Equation 2: The average consumption of electricity of devices

$$I_{avg} = (t_0*I_0 + t_1*I_1 \ldots + t_x*I_x) / (t_0 + t_1 \ldots + t_x)$$

The average consumption equals sum of consumptions in their own time, divided by the sum of times. For example, if the consumption of microcontrollers is 11.55mA, and of sensors, 2.5mA in the awake period which lasts 2 seconds, and the consumption of microcontrollers in deep sleep (or when the sensor is off) equals 6.7µA (from*Figure 5*), then we can calculate that the average consumption of the device is (11.5mA+2.5mA)*2sec + 0.067mA*3598sec)/(2sec+3598sec) = 0.0747mA (74.4 µA).

The measure of battery performance and longevity can be quantified in several ways. The average run time on a full charge (in hours) can be calculated using the formula below (Equation 3).

Equation 3: Average battery lifespan [h]

```
Average battery life[h] = capacity of battery[mAh]/average current[mA]
```

If a battery with a capacity of 3000mA is used for power supply, with the average consumption of 0.0747mA mentioned above, the device would work for 1670 days (around 40 000 hours).

For this reason, the self-discharge rate of the same battery is 125 µA continuously, the average consumption of the device will be 133 µA (from Equation 2) and the device will work for 940 and not 1,670 days.

Power reduction register (PRR)

Microcontrollers include several sleep modes to save power. ATMega328P processor can lower power consumption by shutting down the clock, for select peripherals, via a register setting. That register is called the Power Reduction Register (PRR). It can cut off TWI, SPI, USART0, timers, counters, and ADC.

Figure 3 PRR – Power Reduction Register (Origin: Figure 10.11.3)[1]

Bit	7	6	5	4	3	2	1	0
	PRTWI0	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI0	PRUSART0	PRADC
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

The various bits in this register turn off internal devices, as follows [1]:

Bit 7 – PRTWI: Power Reduction TWI – Set this bit to 1 and shut down the TWI by stopping the clock to the module.

Bit 6 – PRTIM2: Power Reduction Timer/Counter2 - Set this bit to 1 and shut down the Timer/Counter2 module in synchronous mode (AS2 is 0).

Bit 5 – PRTIM0: Power Reduction Timer/Counter0 - Set this bit to 1 and shut down the Timer/Counter0 module.

Bit 4 – Reserved - This bit is reserved and will always read as zero.

Bit 3 – PRTIM1: Power Reduction Timer/Counter1 - Set this bit to 1 and shut down the Timer/Counter1 module.

Bit 2 – PRSPI: Power Reduction Serial Peripheral Interface - Set this bit to 1 and shut down the Serial Peripheral Interface by stopping the clock to the module.
Bit 1 – PRUSART0: Power Reduction USART0 - Set this bit to 1 and shut down the USART by stopping the clock to the module.
Bit 0 – PRADC: Power Reduction ADC - Set this bit to 1 and shut down the ADC. [5]

Note that PRR only applies in active (non-sleep) or idle modes. In sleep mode, those modules are already turned off!

ADC subsystem will be turned off with one line in software (ADCSRA = 0) and power consumption will drop from µA to nA (1000 times!).
Moreover, by adding all digital pins to work as outlets, the consumption decreases for an additional 4-5mA.

BOD disable

Many microcontrollers have a protection circuit that detects when the supply voltage goes below the level required for reliable operation and puts the device into a reset state to ensure proper startup when power returns. This action is called a "Brown Out Reset" or BOR. Brown Out Reset is an important function to increase the reliability of a microcontroller after start-up. BOR will solve problems with the power supply. Many processors have an on-chip Brown-out Detection (BOD) circuit for monitoring the Operating Voltage (Vcc) level during operation. By comparing the Vcc to a fixed trigger level it can determine if the device needs to be put into the reset mode to prevent erratic operation. [4]

When the Brown-out Detector (BOD) is enabled, it actively monitors the power supply voltage even during a sleep period. To save power, it is possible to disable the BOD by software for some of the sleep modes. The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses. If BOD is disabled in the software, the BOD function is turned off immediately after entering sleep mode. After wake-up, BOD is automatically enabled and ensures safe operation in case of lower Vcc.

Power consumption in sleep mode

The most significant consumption saving is realized by using the sleep mode. The table below shows typical and maximal consumption values:

Table 2 ATmega328P DC characteristics (Origin: Table 29-8.)[1]

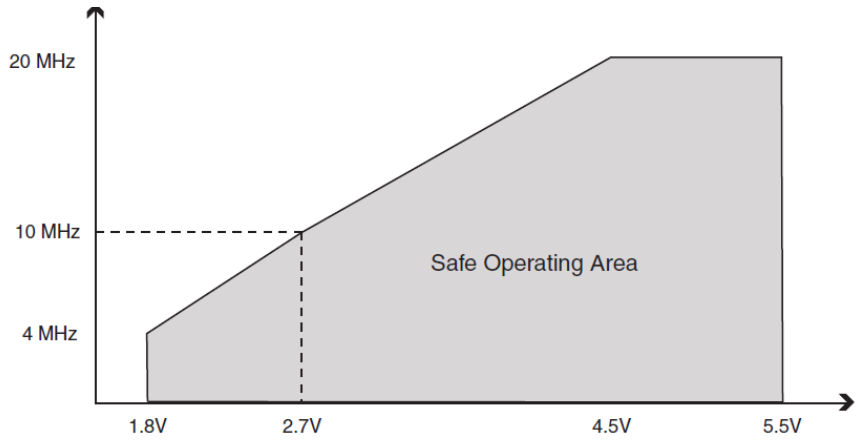
Parameter	Condition	Typical	Maximum	
Power Supply Current	Active 1MHz, VCC = 2V	0.3	0.5	mA
	Active 4MHz, VCC = 3V	1.7	2.5	
	Active 8MHz, VCC = 5V	5.2	9	
	Idle 1MHz, VCC = 2V	0.04	0.15	
	Idle 4MHz, VCC = 3V	0.3	0.7	
	Idle 8MHz, VCC = 5V	1.2	2.7	
Power-save mode	32kHz TOSC enabled, VCC = 1.8V	0.8		µA
	32kHz TOSC enabled, VCC = 3V	0.9		
Power-down mode	WDT enabled, VCC = 3V	4.2	8	
	WDT disabled, VCC = 3V	0.1	2	

Software changes in the microcontroller itself lead to very low consumption in deep sleep mode (around 5 µA, or even less than 1µA when the watchdog timer is turned off with the software), while the consumption when awake is less than 5mA - to this value we should also add the consumption of the sensors, LED diodes, etc.

Voltage and frequency scaling

Frequency is dependent on V_{CC}.

Figure 4 Maximum Frequency vs. VCC (Origin: Figure 29-1.)[1]



As shown in ***Figure 4***, the Frequency vs. VCC curve is linear [1]:

a) for frequency lower than 4MHz, use at least 1.8V (will not operate below 1.8V)
b) for frequencies between 4MHz and 10MHz use Vcc from 1.8V to 2.7V and calculate it with formula

Equation 4: Formula for the required voltage for speed (between 4MHz and 10MHz)

$$Vcc\,[V] = 1.8 + \Big(Fq - 4\Big)*0.15, \quad 4 \le Fq\,[MHz] \le 10$$

c) for frequencies between 10MHz and 20MHz use Vcc from 2.7V to 4.5V (or to maximal 5.5V) and calculate it with formula

Equation 5: Formula for the required voltage for speed (between 10MHz and 20MHz)

$$Vcc\,[V] = 2.7 + \Big(Fq - 10\Big)*0.18, \quad 10 \le Fq\,[MHz] \le 20$$

d) For example, with 3.7Vcc you can run on 16MHz.

The maximal speed for the given voltage value can also be calculated:

a) from 1.8V to 2.7V:

Equation 6: Formula for maximum speed for given Vcc

$$Fq\,[MHz] = 4 + \frac{(Vcc - 1.8)}{0.15}, \quad 1.8 \le Vcc \le 2.7$$

b) from 2.7V to 4.5V

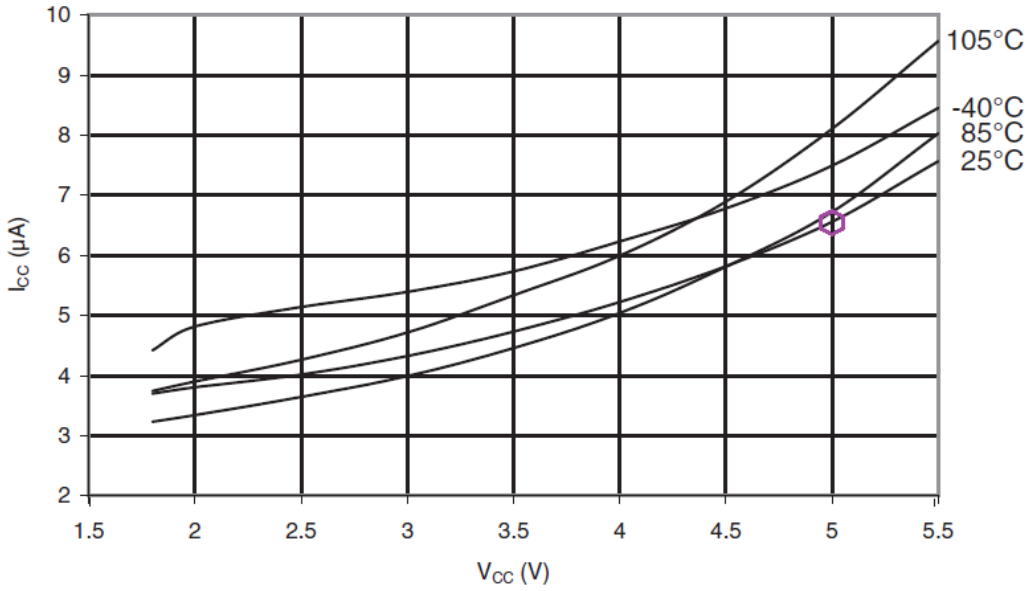
Equation 7: Formula for maximum speed for given Vcc

$$Fq\,[MHz] = 10 + \frac{\Big(Vcc - 2.7\Big)}{0.18}, \quad 2.7 \le Vcc \le 4.5$$

c) For example, to run on 8MHz, minimum 2.4V is needed.

Figure 5 ATmega328P: Power-Down Supply Current vs. VCC - Watchdog Timer Enabled

(Orig: Figure 35-12)[1]

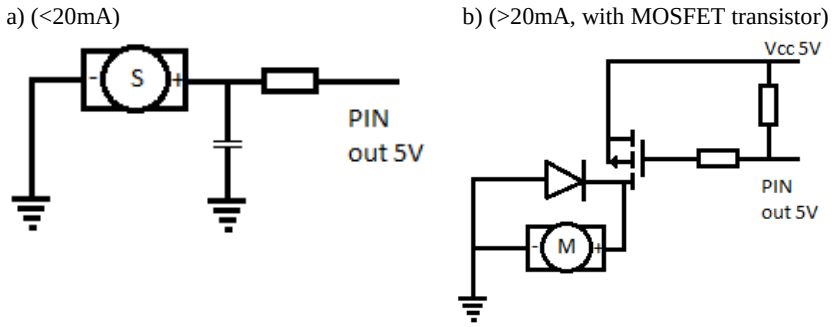


Powering off external devices

In the case of sensors having really low consumption (<20mA), it is possible to power the device directly from a digital pin of the processor (for ATMega328P it is any pin from PortD: 2, 3, 4, 5, 6, 11, 12, 13) and just set the pin to output and high (+5V) when required to turn the sensor on. "Most sensors can be powered with a digital pin, but you have to be careful with I2C chips. When powered down, a current will flow through the pullup resistors, so have to power the pullup resistors also with the same digital pins. When you use the internal pullup resistors (as set by default in the Wire library) you have to disable them. When some current is flowing into a sensor via the signal pins, it might work half-half and it can be unstable and shortcutting. The sensor must either work with a normal voltage (set pin HIGH: the sensor is powered up) or be completely unpowered (set pin LOW: the sensor is powered down)." [8]

Besides disabling pull-up resistors, it is preferable to set the pin high before switching it to output (to save forcing the capacitor low). When finished, first set the pin to input, and then to LOW (to save draining the capacitor). When working with I_2C, turn off the I_2C hardware and then turn off the I_2C pull-ups to save power.

Figure 6 Powering from pin



Special attention should be paid to the consumption of LED diodes used in the device. They indicate the status of the device, whether it is turned on, currently active, or inactive. Even though LEDs have a very low consumption (between 5 to 20mA), they have a significant effect on the total consumption of the device. This is why it is recommended that LEDs should constantly turn on and off, instead of being turned on all the time. The

LEDs are turned on periodically every 1 to 5 seconds, and they can stay turned on for 5 to 20 seconds. The rest of the time they should remain turned off. There is a significant difference between the consumption of the LEDs which are turned on for 5 seconds and the ones turned on for 15 seconds, and both of the flashes can be seen with the naked eye. Therefore, choosing the shortest flash possible visible with the naked eye can lead to additional savings in consumption.

III. CONCLUSION

It is necessary to know the purpose of the device, surroundings and conditions during design of different IoT devices, in order to choose the technology which the device will be developed on. In this paper you will find documented different techniques for reducing the consumption of the IoT devices, based on the changes in hardware as well as changes in software. Some methods are related to removing extra hardware, or replacing some components (replacing existing voltage regulator, replacing charging mode through USB port, removing LED diodes). Changes in the application are described, which contribute to a significant device consumption reduction. It is also indicated that using Bare-bones board, that is, bare microcontrollers in minimal configuration, instead of experimental plates, leads to excellent results. The best results are gained by combining all the methods proposed, that is, with the changes in software in minimal configuration of microcontrollers, combined with turning off the sensors. Applying said principles in design of IoT battery-operated devices, from processors, to sensors and connection methods, provides very long work duration with the minimal consumption of the built-in batteries, where self-discharging of batteries in use plays a great role play.

Conflict of interest

There is no conflict to disclose.

REFERENCES

[1]. megaAVR® Data Sheet (ATmega48A/PA/88A/PA/168A/PA/328/P), Data Sheet Complete, 2020 Microchip Technology Inc., DS40002061B
[2]. Per Andreas Gulbrandsen, AN2515 AVR® Low-Power Techniques, © 2018, Microchip Technology Inc., AN2515
[3]. Saša Salapura, Nebojša Kuduz, "OPTIMIZING ENERGY CONSUMPTION OF AVR BASED IOT DEVICES", X INTERNATIONAL CONFERENCE OF SOCIAL AND TECHNOLOGICAL DEVELOPMENT, Trebinje, June, 03-06, 2021, Republic of Srpska, BiH, 2021
[4]. Kevin Darrah, "Low Power Arduino! Deep Sleep", 2016, last access 05.11.2022. https://www.kevindarrah.com/
[5]. Nick Gammon, "Power saving techniques for microprocessors", 2015, last access 05.11.2022. https://www.gammon.com.au/power
[6]. Eduardo Redondo-Iglesias, Pascal Venet, Serge Pelissier. Measuring Reversible and Irreversible Capacity Losses on Lithium-ion Batteries. VPPC, Oct 2016, Hangzhou, China. ff10.1109/VPPC.2016.7791723ff. ffhal-01393614v2f
[7]. Battery Performance Characteristics, The Electropedia, 2019. https://www.mpoweruk.com/performance.htm
[8]. Powering sensor from digital pin, Forum Arduino.cc, 2014, last access 05.11.2022. https://forum.arduino.cc/t/powering-sensor-from-digital-pin/276587
[9]. Microcontroller Design Recommendation for 8-Bit Devices, 2020, last access 05.11.2022.,
[10]. https://microchipdeveloper.com/8bit:guide

Saša Salapura. "Power Saving Techniques for IoT Devices." *International Journal of Engineering and Science*, vol. 12, no. 12, 2022, pp. 33-39.