Power management and Energy consumption in smart things

Professor: Dr.Mahdi Rasti

Lecturer: HamidReza Hemati

AmirKabir University of Technology
Fall 2021

Agenda

1- Big concept

- IoT definition
- IoT architecture
- Technical research challenges

2- Things

- Defination of Things
- sensors
- Actuators
- Communication technologies and devices

3- Power management

- Power supllies in Iot
- Power consumption
- Battery capacity
- Different kind of batteries
- Importance of power management

4- software technics to reduce power consumption

- esp8266
- Arduino boards

5- Hardware approaches to reduce power consumption

- Designing Personalized Boards
- Removing extra parts
- Working Voltage and clock speed

6- Important pins

- Analog to Digital converter
- pulse width modulation

1

BIG CONCEPT

Let’s start with big concept about "Internet of Thing" and “Power Management"

Introduction

- The Internet of Things (IoT) is an extension of the Internet in which large numbers of “things”, including sensors, actuators and processors, in addition to human users, are networked and able to provide high resolution data on their environment and exercise a degree of control over it[1].
- The principal advantage of IoT consists of its ability to enable communication between an infinite amount of machines incorporated into a large-scale wireless network[1].
- These automated devices and sensors together produce and transmit information in real-time, which is useless in the case of incorrect or insufficient filtering and data processing[1].

Introduction

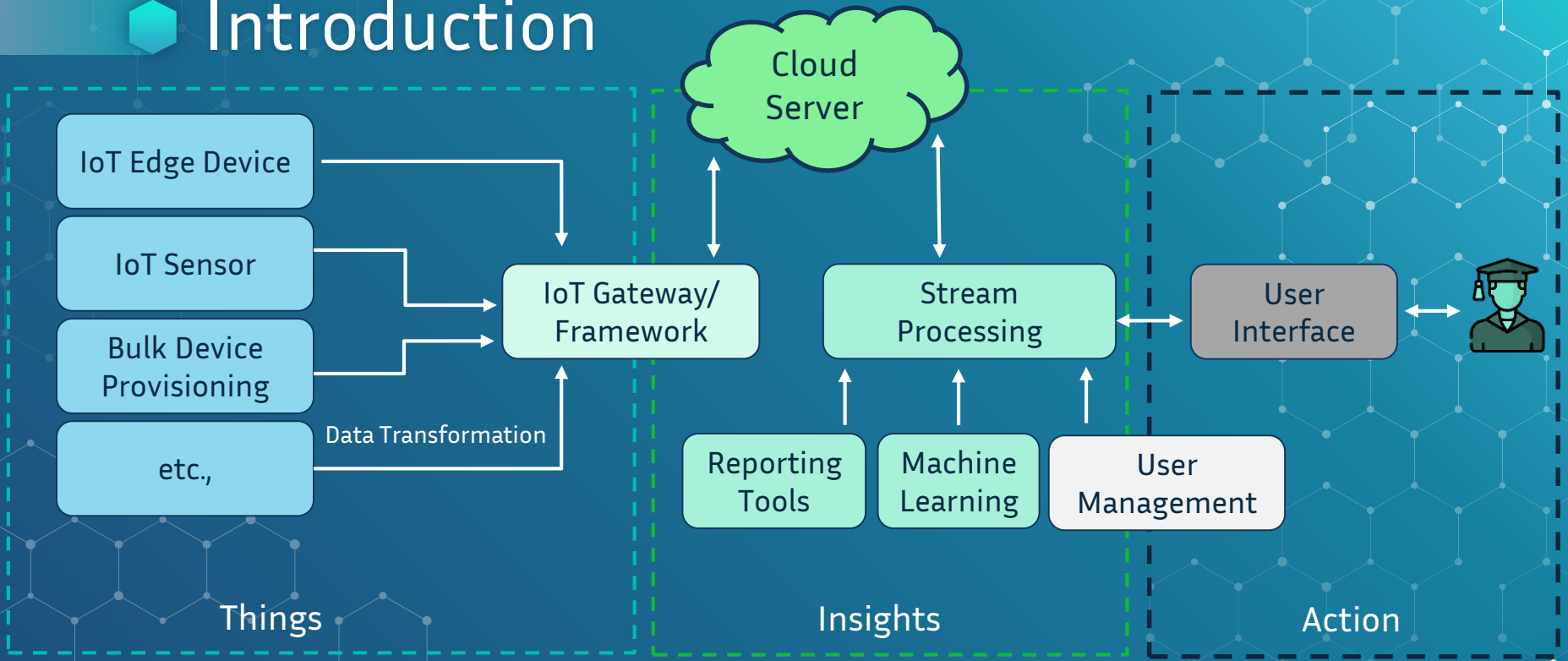


Figure 1: Abstract architectures[2].

The main IoT technical research challenges

- Design
  - Architecture
  - Interoperability
  - Scalability
  - Mobility
  - Security and Privacy
- Scientific/Engineering
  - Energy efficiency/Power
  - Reliability /Robustness
- Management/ Operation
  - Software
  - Development
  - Availability
  - Data management
  - Cloud computing

[1] Peter J. Ryan 1 and Richard B. Watson, “Research Challenges for the Internet of Things: What Role Can OR Play?”, systems journal, 14 March 2017 .

Introduction

- Many of these challenges are technical, including interoperability and scalability
- Reducing power consumption and increasing battery life is one of the most important challenges especially in LP applications
- How to invest in the IoT is a challenge for business
- Major social, legal and ethical challenges, including security and privacy
- As the future IoT will be a multi-national, multi-industry, multi-technology infrastructure

2 Things

What are things?

- A thing, in the context of the Internet of things, is an entity or physical object that has a unique identifier, an embedded system and the ability to transfer data over a network.
- A smart object has at a minimum, the following four defining characteristics
  1. Processing unit
  2. Sensor(s) and/or Actuator(s)
  3. Communication Device
  4. Power Source

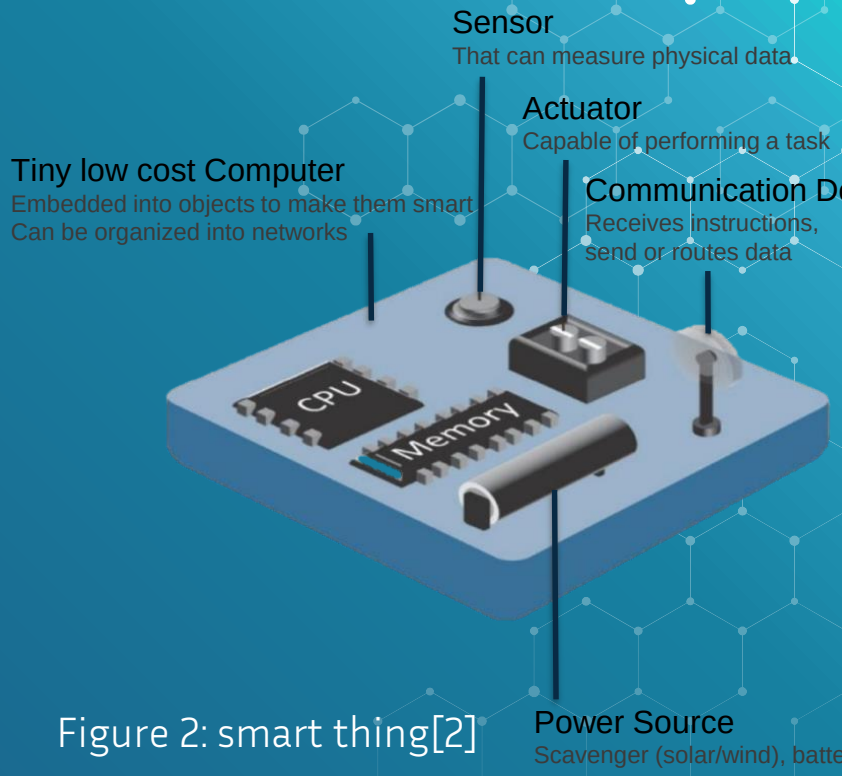


Figure 2: smart thing[2]

[2] IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Thing, Cisco press, 2017

Sensors

- Measures some physical quantity and converts it into a digital representation (ADC in section 7).
- Each sensor consumes different power depending on its type or application

Figure 3: different sensors

[2] IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Thing, Cisco press, 2017

Actuators

- Actuators receive some type of control signal (commonly an electric signal or digital command, PWM in section 7) that triggers a physical effect, usually some type of motion, force, voice, light and so on.
- Like sensors each kind of actuator consume different amount of power. For example motor power consumptions is much higher than buzzer

[2] IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Thing, Cisco press, 2017

Communication technologies

- An essential requirement for implementing a large and cooperating network of devices is the ability to have a flexible, inexpensive and adaptive connectivity layer
- For these reasons, during the years, several wireless connectivity standards have emerged, ranging from very short-range, low power connections to long range connectivity solutions based on cellular technologies.
- While short-range protocols can be successfully used for a number of applications (e.g., smart homes, wearable, etc.), for many other applications larger coverage radii become mandatory.

[3] LPWAN white paper, Leverege LLC, 2020 | www.leverege.com

Communication Devices and technologies

- Each communication technology has its own characteristics. To determine the type of communication device of objects, we must pay attention to these features.

Standard	ZigBee (WPAN)	Low Power Wi-Fi (WLAN)	6LoWPAN (LPWAN)	LoRaWAN (LPWAN)	NB-IoT (LPWAN - cellular)	LTE-M (LPWAN - cellular)	5G (cellular)	Wi-Sun (WNAN)
Nominal range	10 -100 m	70 m - 225 m	25 - 50 m	2 - 15 Km	1 - 15 Km	1 - 11 Km	up to 100 km	5 - 10 km
Max Data Rate (Kbit/s)	250 Kbps	15 Mbps	250 Kbps	50 Kbps	250 Kbps	1 Mbps	599 Mbps	300 Kbps
Power consumption	Medium	Low to medium	Low	Low to medium	Low	Low	Low to medium	Medium to high

Figure 4: characteristics of communication technologies and their impact on the power consumption

3 Power management

Power supplies in IoT

- One of the first decisions in IoT system design should be the power source
- The power source of smart things is determined by the area of application and The location which the node is placed
  - Urban and suburban areas
  - Power grid access
  - smart homes, wearable, agriculture, industry, etc.
- In applications where nodes do not have access to the power grid, power management becomes very important and vital

[4] Internet of Things A to Z : Technologies and Applications

Power supplies in IoT

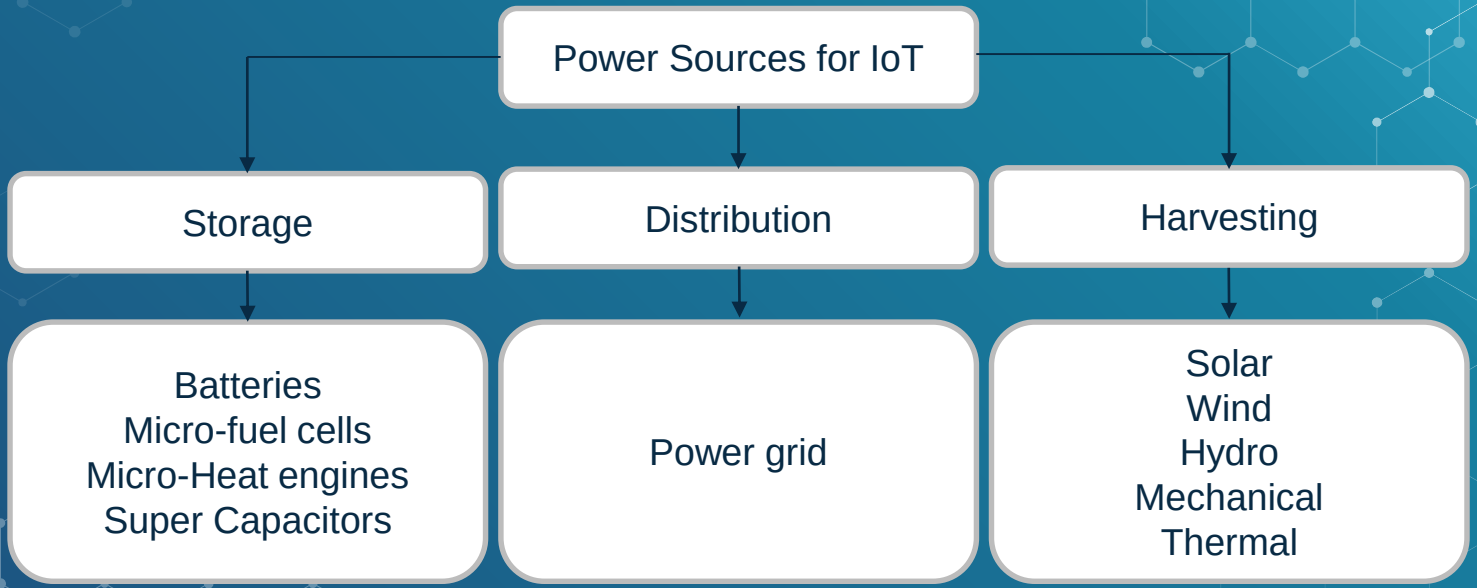


Figure 5: Power Sources for IoT [5]

Power consumption

- If the device will be powered by batteries then all design decisions must consider how to preserve power
- Many networking technologies will not be a good fit with battery power.
- Frequency of communication does have an influence on power selection.

[4] Internet of Things A to Z : Technologies and Applications

Battery capacity

- battery capacity is how much energy is stored in the battery.
- This power is often expressed in Watt-hours (the symbol Wh).
- A Watt-hour is the voltage (V) that the battery provides multiplied by how much current (Amps) the battery can provide for some amount of time (generally in hours).
- Voltage * Amps * hours = Wh.

www.learn.adafruit.com/all-about-batteries/power-capacity-and-power-capability

Battery capacity

- voltage is pretty much fixed for a battery type due to its internal chemistry (alkaline, lithium, lead acid, etc)
- For this reason, battery capacity is often represented by Amps*hour, expressed in Ah or mAh (1000mAh = 1Ah).
- For example, lets say we have a 3V nominal battery with 1Amp-hour capacity, therefore it has 3 Wh of capacity. 1 Ah means that in theory we can draw 1 Amp of current for one hour, or 0.1A for 10 hours, or 0.01A (also known as 10 mA) for 100 hours.

Battery Capacity and Battery Capability

- Battery Capability means the amount of current that the battery can deliver. It is also known as C-rating.
- the amount of current we can really draw (the Battery capability) from a battery is often limited.
- for example, a coin cell that is rated for 1 Ah can't actually provide 1 Amp of current for an hour, in fact it cant even provide 0.1 Amp without overextending itself (Its like saying a human has the capability to travel up to 30 KM: of course running 30 KM is a lot different than walking!)
- Likewise, a 1Ah coin cell has no problem providing a 1mA for 1000 hours but if you try to draw 100mA from it, it'll last a lot less than 10 hours.

www.learn.adafruit.com/all-about-batteries/power-capacity-and-power-capability

Battery Capacity and Battery Capability

- Example:
  2500 mAh it means that the battery has a capability to deliver 2.5A/2500mA of current to the load for 1 hour. The time that the battery works continuously depends upon the load current that it consumes. So if the load consumes only 25 mA of current then the battery can stay alive for 100 hours.

Different kind of Batteries

- Basically, all the electrochemical cells and batteries are classified into two types:
  - Primary (non-rechargeable)
  - Secondary (rechargeable)

- Even though there are several other classifications within these two types of batteries, these two are the basic types. Simply speaking, Primary Batteries are non-rechargeable batteries i.e., they cannot be recharged electrically while the Secondary Batteries are rechargeable batteries i.e., they can be recharged electrically.

Different type of Batteries | electronicshub.org

Different kind of Batteries

- comparison of non-rechargeable Batteries

Battery Type	Characteristics	Applications
Zinc – Carbon	Common, low cost, variety of sizes	Radios, toys, instruments
Magnesium (Mg/MnO2)	High capacity, long shelf life	Military and aircraft Radios
Mercury (Zn/HgO)	Very high capacity, long shelf life	Medical (hearing aids, pacemakers), photography
Alkaline (Zn/Alkaline/MnO2)	Very popular, moderate cost, high performance	Most popular primary batteries
Silver/Zinc (Zn/Ag2O)	Highest capacity, costly, flat discharge	Hearing aids, photography, pagers
Lithium/Soluble Cathode	High energy density, good performance, wide temp range	Wide range of applications with capacity between 1 – 10,000 Ah
Lithium/Solid Cathode	High energy density, low temp performance, long shelf life	Replacement for button and cylindrical cells
Lithium/Solid Electrolyte	Low power, extremely long shelf life	Memory circuits, medical electronics

table 1: primary batteries | electronicshub

Different kind of Batteries

- comparison of rechargeable Batteries

Battery Comparison	NiMH (LSD)	NiMH	Lithium-Iron	Lithium- Ion	NiCd
Output	1.2 Volts	1.2 Volts	1.5 Volts	3.7 Volts	1.2 Volts
Capacity AA (Varies by Manufacturer)	2000 mAh	2500 mAh	3000 mAh	2500 mAh	1000 mAh
Charging Cycles (Varies by Manufacturer)	500 - 2000	1500- 2000	N/A (Not Rechargable)	500 - 1000	300 - 800
Self Discharge (1 Year Average)	2 - 3%	15 - 25%	1 - 2%	15 - 20%	7 -10% (PER MONTH)
Shelf Life (Varies by Manufacturer)	10 - 20 Years	5 Years	20 Years	3 - 5 Years	7 - 10 Years
Sizes Available	AA/AAA (Adapters Available)	AA/AAA/C/D 9V	AA/AAA/9V	Variable Sizes	Variable Sizes
Pricing (4 Pack of AA)	12$	16$	12$	16$	5$

table2: secondary batteries

How to Choose a Battery?

- Selecting a battery for your application can be dialed down to just two characteristics: Performance and Cost. But if we dig a little bit deeper, then the following are determining factors in choosing the right battery for your application.
  - Primary or Secondary
  - Energy or Power
  - Shelf Life
  - Energy Efficiency and Recharge Rate
  - Battery Life
  - Battery Temperature
  - cost

Different type of Batteries | electronicshub.org

Connecting battery

- For powering SoCs we need to connect battery to the Vin (voltage input) pin.
- Important thig that we must consider is the board working voltage and on-board voltage regulator Specifications.

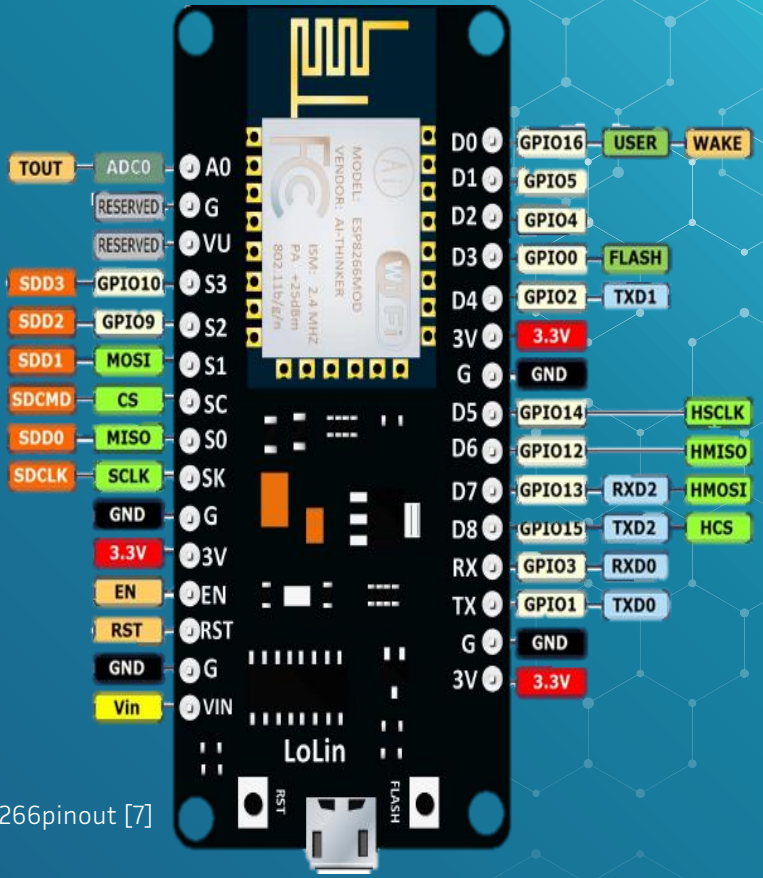


Figure 7: esp8266pinout [7]

Connecting battery

- The NodeMCU has a build in AMS1117 3.3V voltage regulator that transforms the input voltage from the 5V USB connection as well as the voltage of the VIN pin to the stable 3.3V output voltage.

Microcontroller	Minimum Voltage	Typical Voltage	Maximum Voltage
ESP8266	2.58	3.3V	3.6V

Voltage Regulator	Output Voltage	Maximum Input Voltage	Maximum Output Current
AMS1117	3.3V	15V	1A

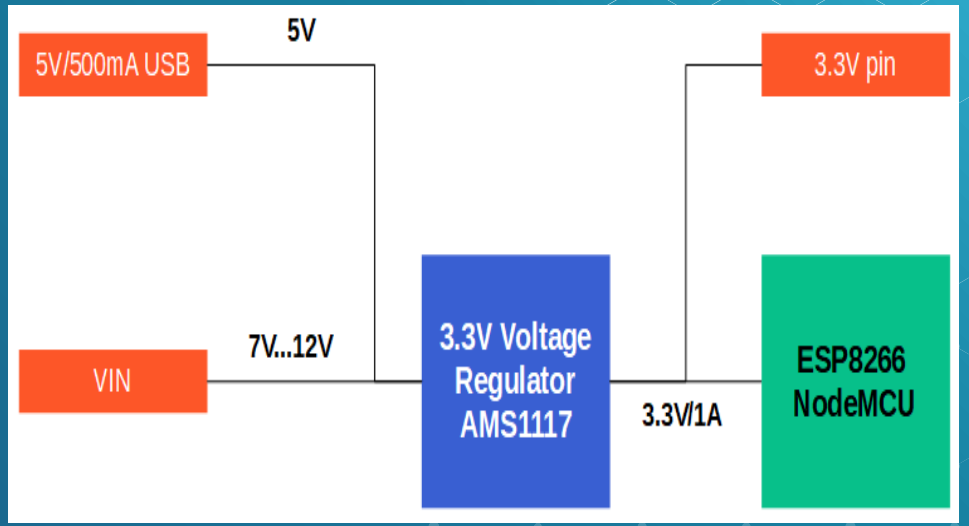


Figure 8: esp8266 voltage levels

https://diyi0t.com/best-battery-for-esp8266/

Importance of power management

- A Key requirement for IoT devices is the ability to consume very low power while maintaining an acceptable level of performance, which could enable them to run on batteries for long periods of time.

- Depending on usage patterns and the power requirements of the attached sensors, actuators, or integrated circuits (ICs), a device may be put into sleep mode or into low-power mode periodically to conserve power.

[4] Internet of Things A to Z : Technologies and Applications

Importance of power management

- In some applications power management is not very critical. For example smart home, some of smart city applications and etc.[8]
- Why is that? Connecting to power grid

Figure 9: smart home

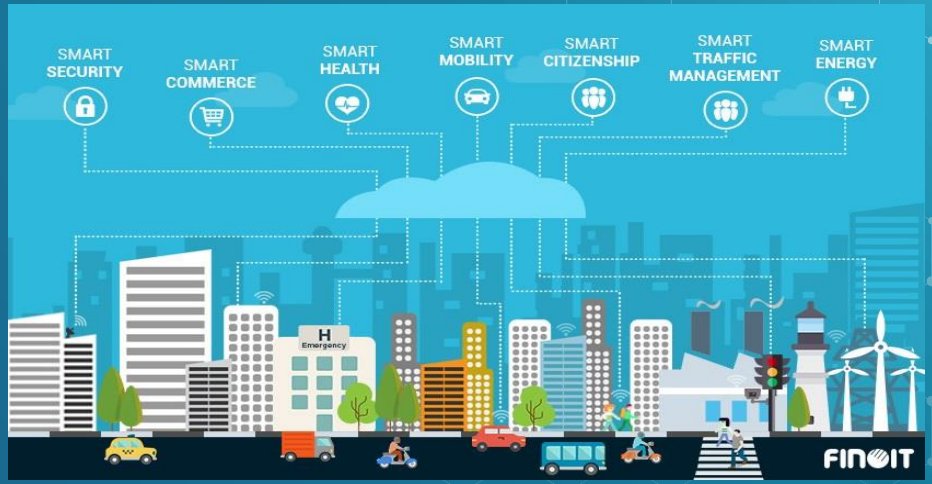


Figure 10: smart city

Importance of power management

- Power management is a critical factor for portable and wearable IoT devices that rely on a wireless power source, such as batteries or photovoltaic cells (solar).

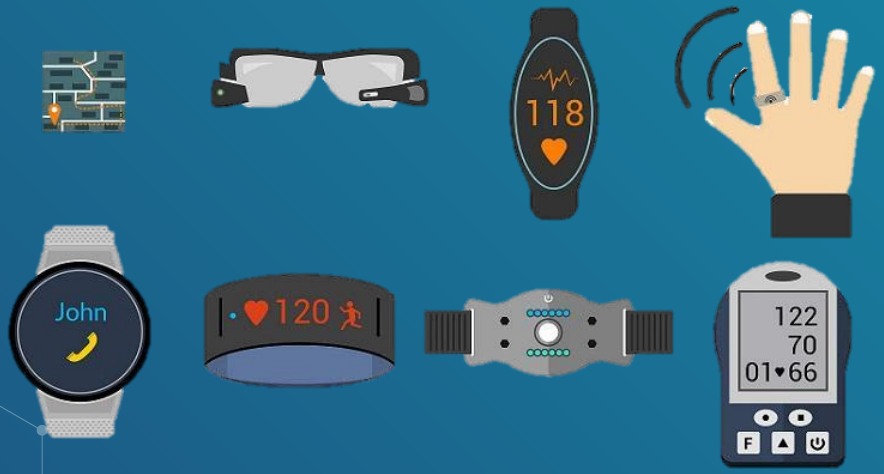


Figure 11: wearable devices

power management – example

- Let's look at an example of the importance of power consumption management.
  - Suppose we want to make an agricultural land smart using the Internet of Things approaches.
  - This agricultural land has an area of 10 hectares

Figure 12: smart agriculture

power management – example

- We need to collect important data from all over the farm Using the nodes we designed
- What if we do not consider power management in nodes?
  - Overhead become more than profit
- What can we do?
  - Does the node need to be always on?
  - Duty-cycle
  - Frequency of data transmission
- Other examples?

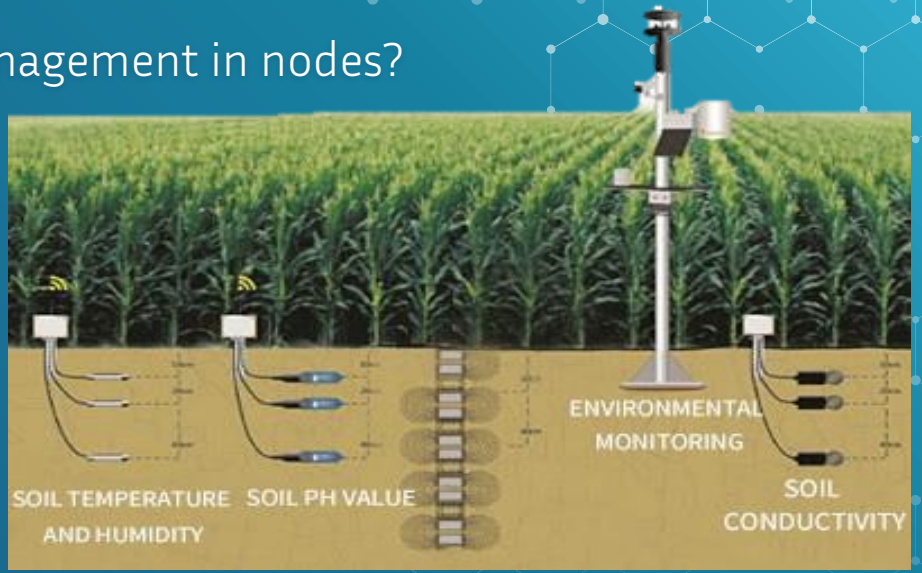


Figure 13: collecting data in farm land

Importance of power management

- So we need to consider power consumption management in designing smart things.
- There are two important categories for power management:
  - Software technics (section 4)
  - Hardware approaches (section 5)

4 software technics to reduce power consumption

Sleep mode process

- The Arduino main loop typically runs continuously
- By using sleep modes we can pause execution of the loop
- Number of wake-up sources can be use to wake the Arduino up and resume the program execution
- Once in sleep mode the ATmega328p can be woken up via an external signal or internal time-based events

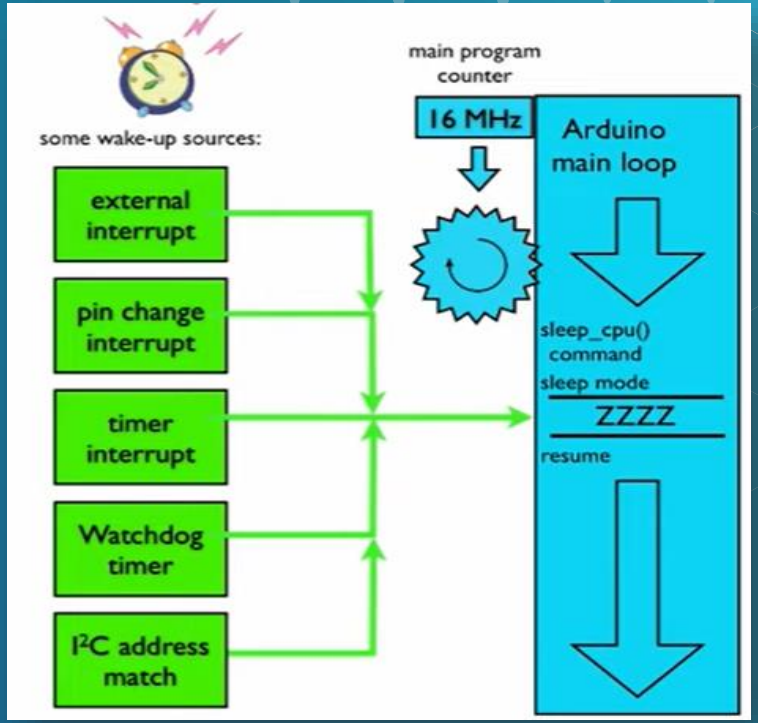


Figure 14: sleep process | makecourse

[9] ATMega328P Datasheet

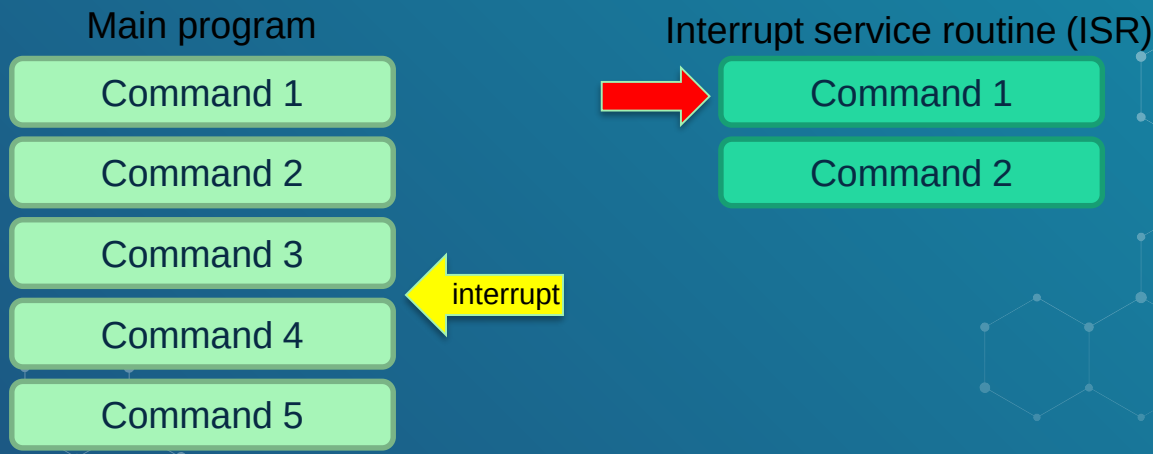
Wake-up events

- There are two general types of wake-up events
  1. External interrupts (using external signals)
  2. Internal interrupts (using internal time-based events)

[9] ATMega328P Datasheet

Interrupts

- **What is an Interrupt?**
  - Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.



[10] Embedded Systems: Introduction to ARM®CORTEX-M Microcontrollers

External Interrupts

- **External Interrupts**
  - External interrupts triggered by a state change on one of the digital pins
- **Trigging Interrupts**
  - An interrupt can be triggered when a digital pin goes from low to high or from high to low (rising and falling edge)

[10] Embedded Systems: Introduction to ARM®CORTEX-M Microcontrollers

External Interrupts

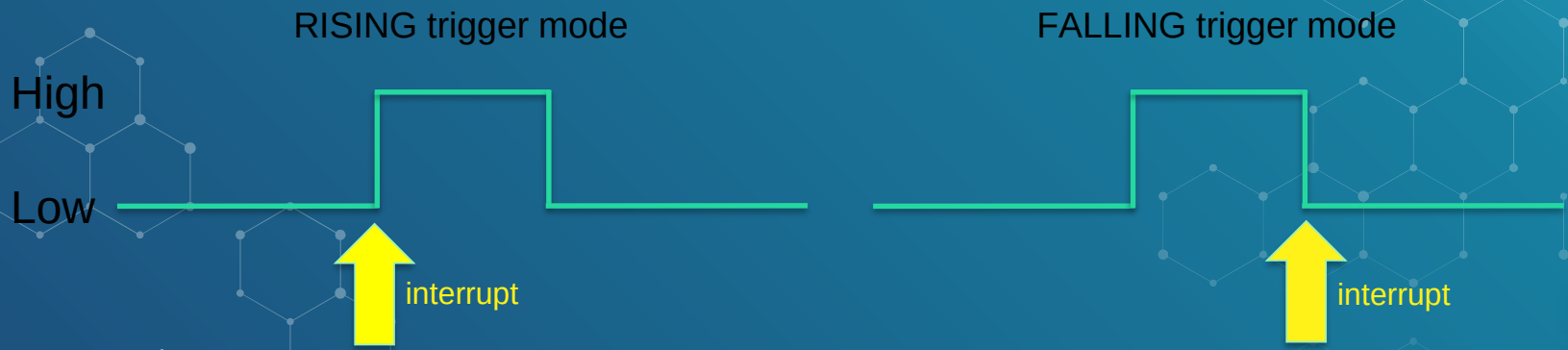
- **External Interrupts pins**

Board	INT0	INT1	INT2	INT3	INT4	INT5	…	INT15
UNO	pin2	pin3						
Nano	pin2	pin3						
Pro mini	pin2	pin3						
Mega	pin2	pin3	pin21	pin20	pin19	pin18		
esp8266	GPIO 0	GPIO 1	GPIO 2	GPIO 3	GPIO 4	GPIO 5	...	GPIO 15

Table3: interrupt pins of Arduino and esp8266 boards [6][9]

External Interrupts

- **How to use External Interrupts in Arduino**
  - attachinterrupt(0 , buttonPressed , RISING)
    - 0 -> INT 0 = pin 2
    - buttonPressed -> ISR name (name of th function to run after interrupt)
    - RISING -> Trigger mode



[9] ATMega328P Datasheet

External Interrupts

- Changing LED state with and without external interrupt.

```
volatile boolean ledOn = false;

void setup() {
  pinMode(13,OUTPUT);
  pinMode(2,INPUT);
  attachInterrupt(0,buttonPressed,RISING);
}

void loop() {

}

void buttonPressed()
{
  if(ledOn)
  {
    ledOn = false;
    digitalWrite(13,LOW);
  }else
  {
    ledOn = true;
    digitalWrite(13,HIGH);
  }
}
```

Figure 15: changing LED state using external interrupt

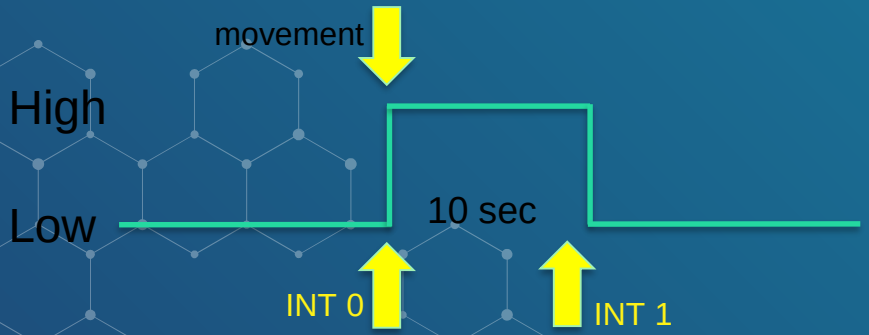
```
boolean ledOn = false;
int button_state = 0;

void setup() {
  pinMode(13,OUTPUT);
  pinMode(2,INPUT);
}

void loop() {
  button_state = digitalRead(2);
  if(button_state == 1){
    buttonPressed();
  }
  delay(100);
}

void buttonPressed()
{
  if(ledOn)
  {
    ledOn = false;
    digitalWrite(13,LOW);
  }else
  {
    ledOn = true;
    digitalWrite(13,HIGH);
  }
}
```

Figure 16: changing LED state without external interrupt

[9] ATMega328P Datasheet

External Interrupts

- Any sensor that produces digital output can be used as an external interrupt maker.
  - Motion detector, smoke detector, humidity and water level sensors and etc.




```
1  #include "LowPower.h"
2
3  void setup() {
4    pinMode(13,OUTPUT);
5    pinMode(2,INPUT);
6    pinMode(3,INPUT);
7    attachInterrupt(0,turnLEDOn,RISING);
8    attachInterrupt(1,turnLEDOff,FALLING);
9  }
10
11 void loop() {
12  LowPower.powerDown(SLEEP_FOREVER , ADC_OFF, BOD_OFF);
13 }
14
15 void turnLEDOn()
16 {
17   digitalWrite(13,HIGH);
18 }
19
20 void turnLEDOff()
21 {
22   digitalWrite(13,LOW);
23 }
```

Figure 17: changing LED state using motion detector external interrupt

Internal Interrupts

- Internal Interrupts are time-based events. They are managed using clock and timers of the system.
- We use periodic timers to create internal interrupts
- For example:
  - lowPower.powerDown(SLEEP_8S, ADC_OF, BOD_OF)
  - **Sleep_8s** means that the Arduino board wakes up after 8 seconds due to an internal interrupt..

[9] ATMega328P Datasheet

Sleep modes

- Arduino boards have 6 different sleep modes:
  - Idle — the least power saving
  - ADC noise reduction
  - Power save
  - Extended standby
  - Standby
  - Power down — the most power saving

[9] ATMega328P Datasheet

Sleep modes

	Active Clock Domains					Oscillators		Wake-up Sources							
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		
Power-down								X(3)	X				X		X
Power-save					X		X(2)	X(3)	X	X			X		X
Standby(1)						X		X(3)	X				X		X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X		X

Notes:
1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Table4: Active Clock Domains and Wake-up Sources in the Different Sleep Modes [9]

Atmega328p system clock

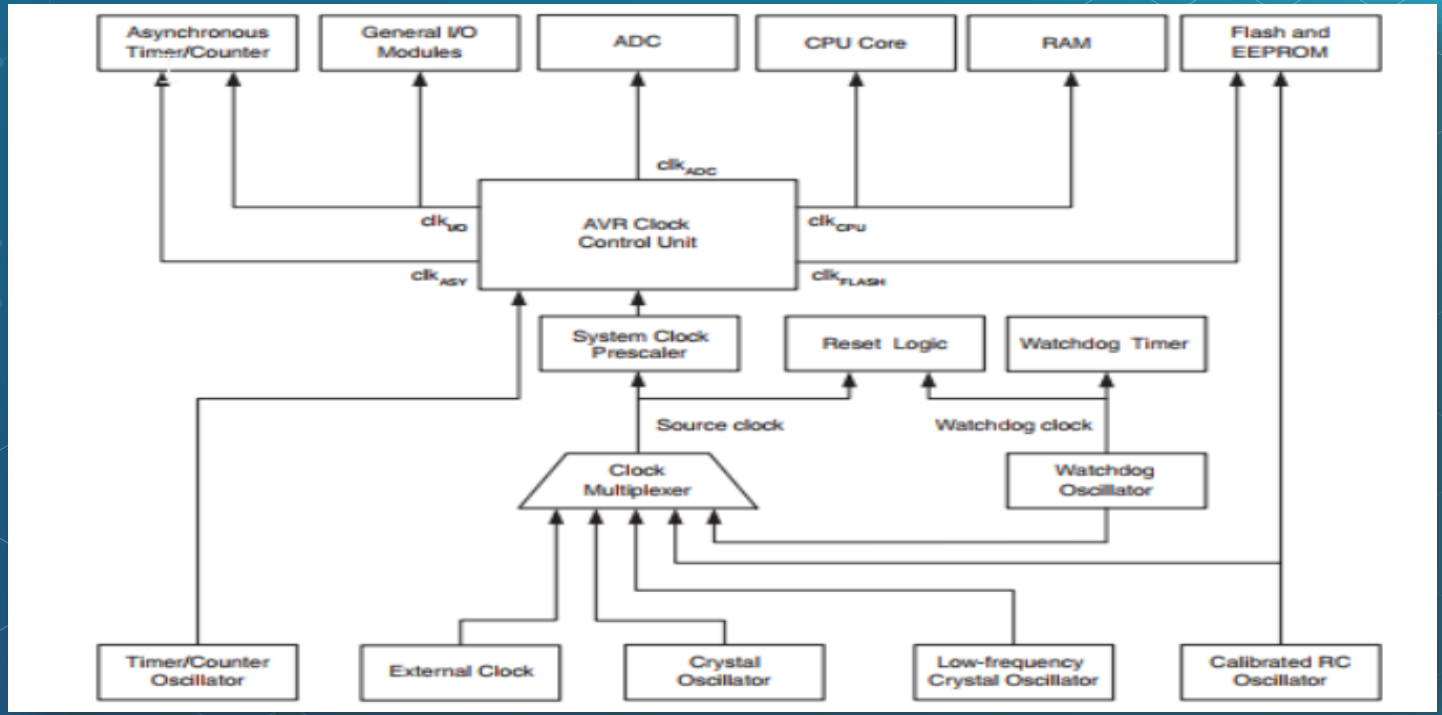


Figure 18: Atmega328p clock distribution System | arnabkumardas.com

Sleep mode libraries

- There is two library to manage sleep modes in Arduino

1. <avr/sleep.h> AVR
2. <lowPower.h> rocket scream github

Power-down sleep mode

- Using lowPower library
- LowPower.powerDown(period, adc, bod);

- Period -> Duration of low power mode.
- adc -> ADC module disable control.
  - ADC_OFF - Turn off ADC module
  - ADC_OFF - Leave ADC module in its default state
- Bod -> Brown Out Detector (BOD) module disable control
  - BOD_OFF - Turn off BOD module
  - BOD_ON - Leave BOD module in its default state

Low power library | github

Power-down sleep mode

- Lets see the effect of powerDown sleep mode on power consumption.

- We are going to test the power consumption of 6 different Arduino boards(nano, uno, mega, pro macro, 5 volt pro mini, 3.3 pro mini) using the same battery

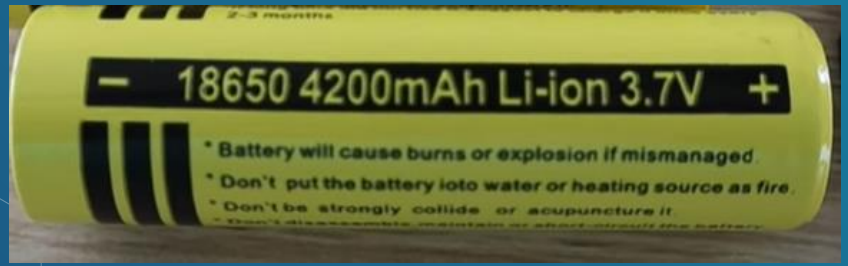


Figure 19: 4200mAh Li-ion battery

Power-down sleep mode

- Lets test battery life of Arduino boards in normal and low power mode

```
1  void setup() {
2    pinMode(LED_BUILTIN, OUTPUT);
3  }
4
5  void loop() {
6    delay(8000);
7    digitalWrite(LED_BUILTIN, HIGH);
8    delay(100);
9    digitalWrite(LED_BUILTIN, LOW);
10 }
```

Figure 20: normal code

```
1  #include "LowPower.h"
2
3  void setup()
4  {
5    pinMode(LED_BUILTIN, OUTPUT);
6  }
7
8  void loop()
9  {
10   LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
11   digitalWrite(LED_BUILTIN, HIGH);
12   delay(100);
13   digitalWrite(LED_BUILTIN, LOW);
14 }
```

Figure 21: low power code

Comparison of normal and low power mode in Arduino | YouTube

Power-down sleep mode

	Normal			Low Power		
	Current [mA]	Hours	Days	Current [mA]	Hours	Days
Uno	54.4	77	3	38.2	110	5
Mega	77.7	54	2	31.72	132	6
Nano	25.5	165	7	6.4	656	27
Pro Micro	43.4	97	4	9.25	454	19
Pro Mini 5V	19.1	220	9	3.2	1313	55
Pro Mini 3.3V	5.5	764	32	1.6	2625	109
Pro Mini 3.3V at 3.7V	5.1	824	34	1.5	2800	117
Battery Capacity	4200	mAh				

Figure 22: comparison of Arduino boards in normal and low power mode | YouTube

ESP8266

- ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications.[6]

- The low-power architecture operates in the following modes:
  - Active mode: Wi-Fi is powered on. It can receive, transmit, or listen.
  - Modem-sleep mode: The CPU is operational. The Wi-Fi is disabled.
  - Light-sleep mode: The CPU and all peripherals are paused. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
  - Deep-sleep mode: Only the RTC is operational and all other part of the chip are powered off.

[6] esp8266ex Datasheet

ESP8266

- ESP8266 series chip provides 3 configurable sleep modes. Users can choose and configure the sleep mode as required.[6]

- Table 1 shows the differences between the 3 sleep modes.
  - RTC (Real-Time Clock).
  - DTIM (Delivery Traffic Indication Message).
  - Deep-sleep mode is controlled by users. Users can call the function to enable Deep-sleep instantly. For details, please refer to Section 4. Deep-sleep.

ESP8266

Item		Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi		OFF	OFF	OFF
System clock		ON	OFF	OFF
RTC		ON	ON	ON
CPU		ON	Pending	OFF
Substrate current		15 mA	0.4 mA	~ 20 μA
Average current	DTIM = 1	16.2 mA	1.8 mA	-
	DTIM = 3	15.4 mA	0.9 mA	
	DTIM = 10	15.2 mA	0.55 mA	

Table 5: comparison of sleep modes in ESP8266 [6]

ESP8266 – Modem-Sleep

- Modem-sleep mode is enabled only when ESP8266 connects to the router in station mode. ESP8266 stays connected to the router through the DTIM beacon mechanism.
  - The DTIM beacon interval of the router is usually 100ms to 1000ms
- In Modem-sleep mode, ESP8266 will close the Wi-Fi module circuit between the two DTIM Beacon intervals in order to save power.
- ESP8266 will be automatically woken up before the next Beacon arrival. The sleep time is decided by the DTIM Beacon interval time of the router
- During sleep, ESP8266 can stay connected to the Wi-Fi and receive the interactive information from a mobile phone or server.

[6] esp8266ex Datasheet

ESP8266 Modem-Sleep

- **Interface:**
  - Modem-sleep:
    The system goes into Modem-sleep via the following interface
    - wifi_set_sleep_type(MODEM_SLEEP_T)
    - In Modem-sleep, the system can be woken up automatically. Users don't need to configure the interface.
- **Application:**
  Modem-sleep is generally used in the applications that need the CPU powered on. An example of the applications is Pulse Width Modulation (PWM) light that needs real-time CPU control.

[6] esp8266ex Datasheet

ESP8266 Light-Sleep

- The working mode of Light-sleep is similar to that of Modem-sleep. The difference is that, during Light-sleep mode, except from Wi-Fi circuit, ESP8266 also powers off clock and suspends internal CPU, resulting in less power than in Modem-sleep mode.

- **Interface:**
  - Light-sleep:
    The system goes into Light-sleep via the following interface
    - wifi_set_sleep_type(LIGHT_SLEEP_T)
    - ESP8266 automatically enters Light-sleep mode when connected to Wi-Fi with the CPU idle.

[6] esp8266ex Datasheet

ESP8266 Light-Sleep

- **External Wake-up :**
  - During Light-sleep, the CPU is suspended and will not respond to the signals and interrupts from the peripheral hardware interfaces. Therefore, ESP8266 needs to be woken up via external GPIO. The waking process is less than 3 ms
    The GPIO wake-up function can only be enabled by level triggers. The interface is as follows.
    - void gpio_pin_wakeup_enable(uint32 i, GPIO_INT_TYPE intr_state);
    - GPIO16 cannot be used for wake-ups.

Uint32 i	The IO serial number of the wake-up function
GPIO_INT_TYPE Intr_state	The trigger mode of wake-up • GPIO_PIN_INTR_LOLEVEL • GPIO_PIN_INTR_HILEVEL

Table6: external wake-up function inputs[6]

ESP8266 Light-Sleep

- **Application:**
  Light-sleep mode can be used in the scenarios where the applications need to stay connected to the router and can respond to the sending data from the router in real time. The CPU can be idle before receiving commands. An example is the Wi-Fi switch whose CPU is idle for most of the time and only performs GPIO operations until receiving the control commands.[7]
- **Note:**
  If a task interval is shorter than the DTIM beacon interval, the system cannot go into Light-sleep mode, as the figure below shows

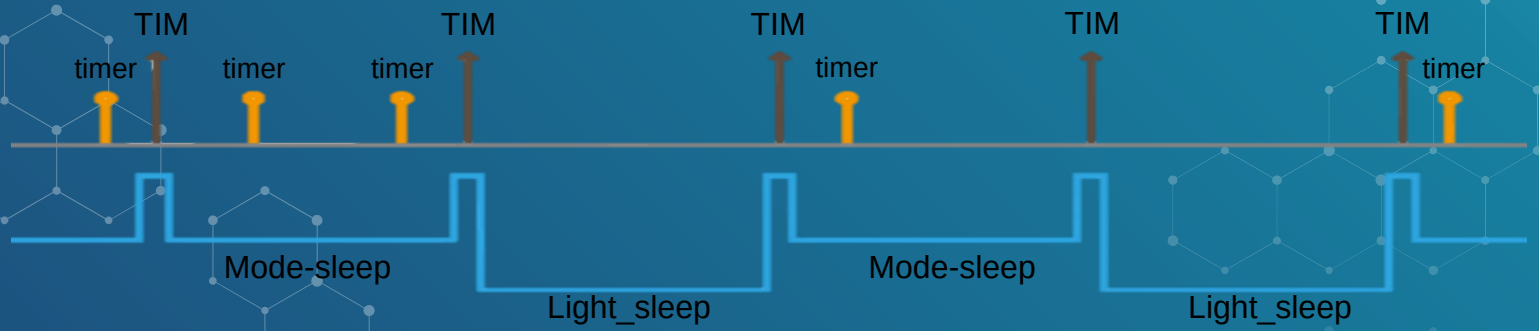


Figure 23: DTIM beacon intervals [8]

ESP8266 Deep-Sleep

- Unlike the other two modes, the system cannot go into Deep-sleep automatically
- Users can call the interface function system_deep_sleep to immediately enable Deep-sleep.
- In this mode, the chip will turn off Wi-Fi connectivity and data connection; only the RTC module is still working, responsible for periodic wake-ups.

[8] esp8266ex Datasheet

ESP8266 Deep-Sleep

- **Interface:**
  - Enable Deep-sleep:
    The system goes into Deep-sleep mode via the following interface
    - void system_deep_sleep(uint32 time_in_us)
    - To enable Deep-sleep, you need to connect GPIO16 to the EXT_RSTB pin of ESP8266

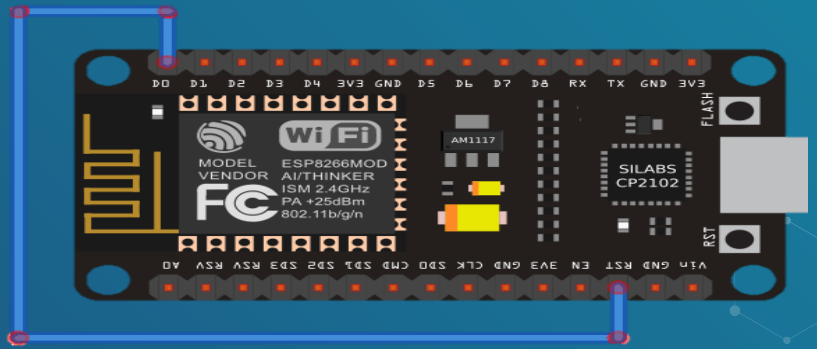


Figure 24: connecting GPIO16 to RST pin | randomnerdtutorials.com

[7] esp8266ex Datasheet

ESP8266 Deep-Sleep

- Why connect GPIO16 to RST pin?
  - Especial wake pin

- RESET pin **HIGH** → ESP running
- RESET pin **LOW** → ESP restarts

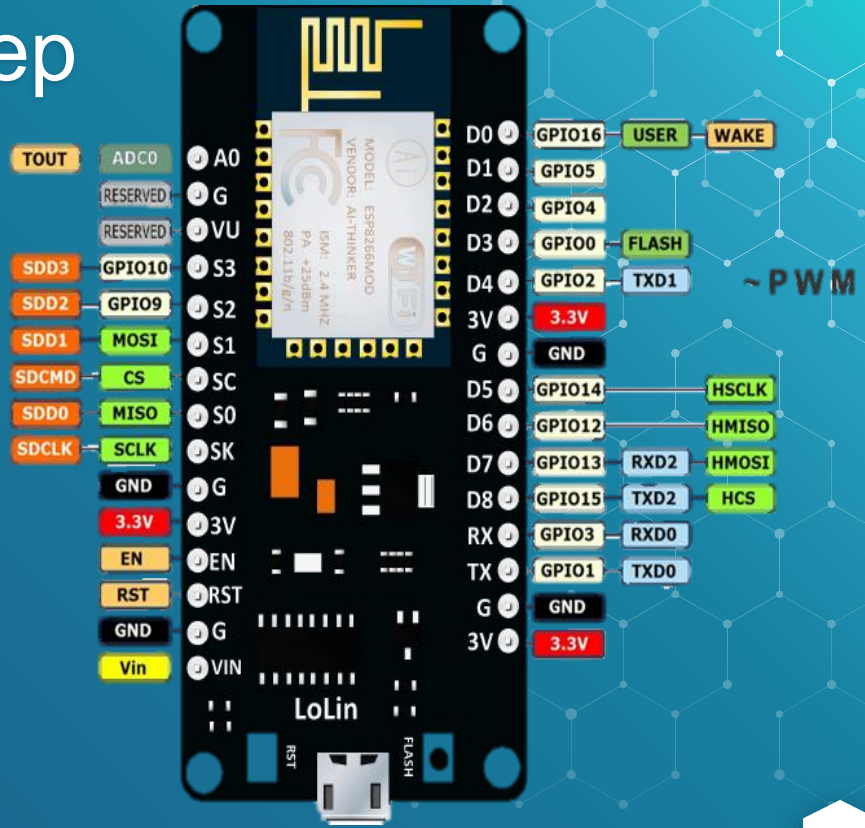


Figure 25: esp8266pinout [7]

ESP8266 Deep-Sleep

```
/*
 * ESP8266 Deep sleep mode example
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 */

void setup() {
  Serial.begin(115200);
  Serial.setTimeout(2000);

  // Wait for serial to initialize.
  while(!Serial) { }

  // Deep sleep mode for 30 seconds, the ESP8266 wakes up by itself when GPIO 16 (D0 in NodeMCU board) is connected to the RESET pin
  Serial.println("I'm awake, but I'm going into deep sleep mode for 30 seconds");
  ESP.deepSleep(30e6);

  // Deep sleep mode until RESET pin is connected to a LOW signal (for example pushbutton or magnetic reed switch)
  //Serial.println("I'm awake, but I'm going into deep sleep mode until RESET pin is connected to a LOW signal");
  //ESP.deepSleep(0);
}

void loop() {
}
```

Figure 26: deep sleep mode | randomnerdtutorials.com

ESP8266 Deep-Sleep

- **Application:**
  Deep-sleep can be used in low-power sensor applications or in the scenarios where data transmission is not required for most of the time. The device wakes up from Deep-sleep at intervals to measure and upload data, and then goes to Deep-sleep again. The device can also store data in the RTC memory (which can still save data in Deep-sleep mode) and then send it at a time.[7]

5 Hardware approaches to manage power consumption

Removing extra parts

- Every small change has a big impact
- Lets start with a simple test
  - Removing power LED from Arduino pro mini

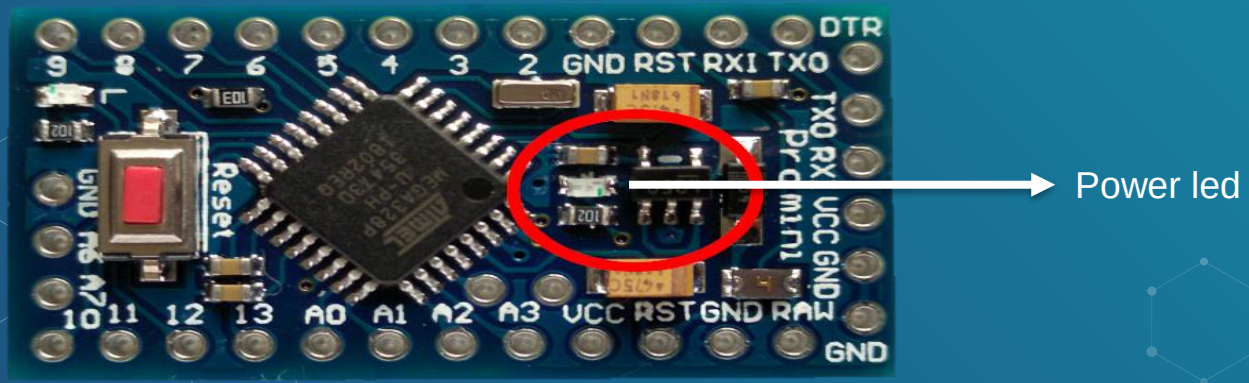


Figure 27: Arduino pro mini

Removing extra parts

- Table below show the current draw of a Arduino pro mini with and without power LED

		Test - 4200 mAh			Common - 3500 mAh		
	Current [mA]	Hours	Days	Years	Hours	Days	Years
Normal	5.4	778	32	0	648	27	0
Low Power	1.5	2800	117	0	2333	97	0
No Led	0.058	72414	3017	8	60345	2514	7

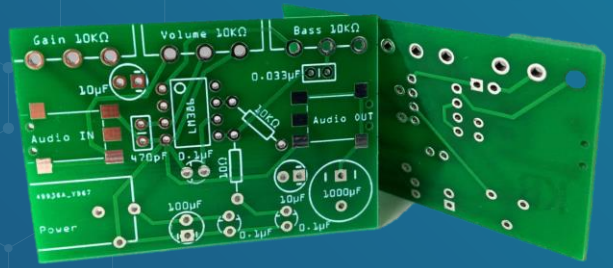
Table 7: comparison of power consumption in Arduino pro mini | YouTube

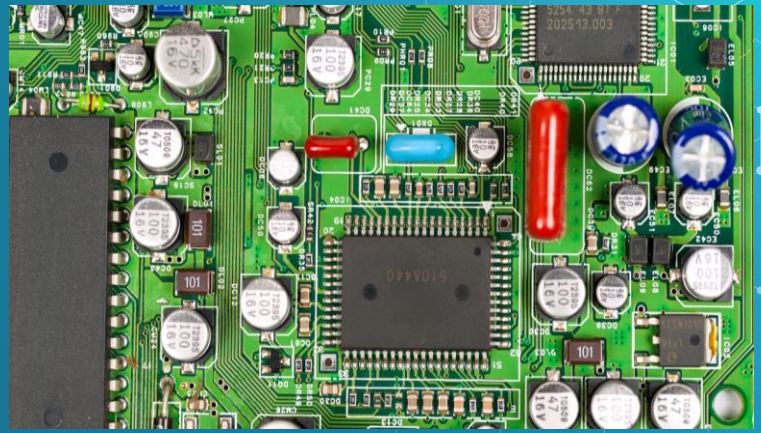
Using required peripherals

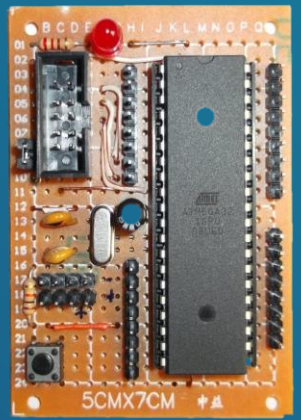
- As we saw in the last example, it is very important to use only required and necessary peripherals, no more and no less
  - We have to design the hardware according to this principle to reduce power consumption in nodes.
- Microcontroller development boards are PCBs with additional circuitry to support the microcontroller to make it more convenient to prototype with and program the chip
- SoC and SBC are good as prototypes but not usually ideal for real product

https://developer.ibm.com/articles/iot-lp101-best-hardware-devices-iot-project/

PCB design







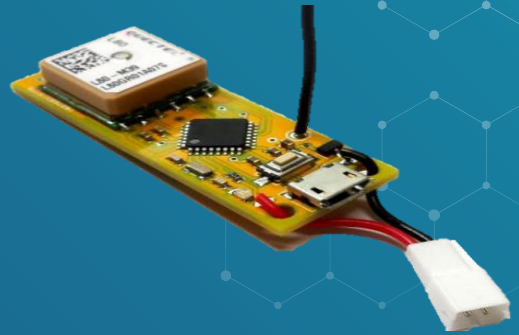


Figure 28: PCBs

Lowering the Voltage

- One of the easiest ways to reduce current is to lower the voltage you give to the board.[12]

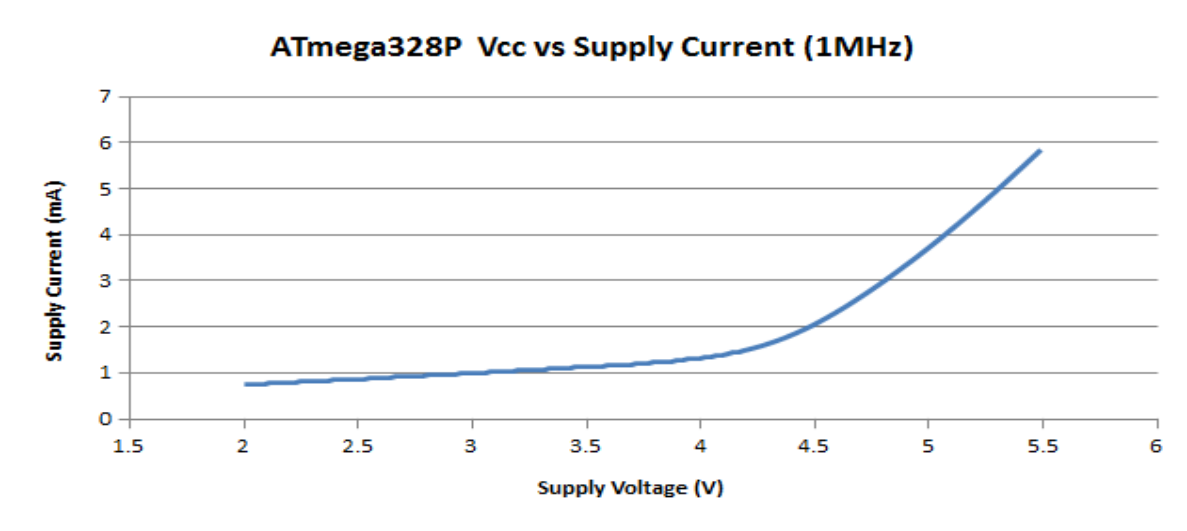


Figure 29: impact of working voltage on current draw [12]

[12] A SURVEY ABOUT POWER CONSUMPTION FOR ARDUINO

Lowering the Voltage

- Reducing the voltage, does have a catch though. Reducing the system voltage too much, without reducing clock speed, could make the microcontroller start to behave strangely. In Arduino boards The Mega, Uno, nano, and 5V Pro Mini all use a 16MHz crystal but 3.3V pro mini use a 8MHz crystal.
  - The relationship between clock speed and system voltage is the reason our 3.3V Pro Mini uses a 8MHz clock instead of 16MHz.

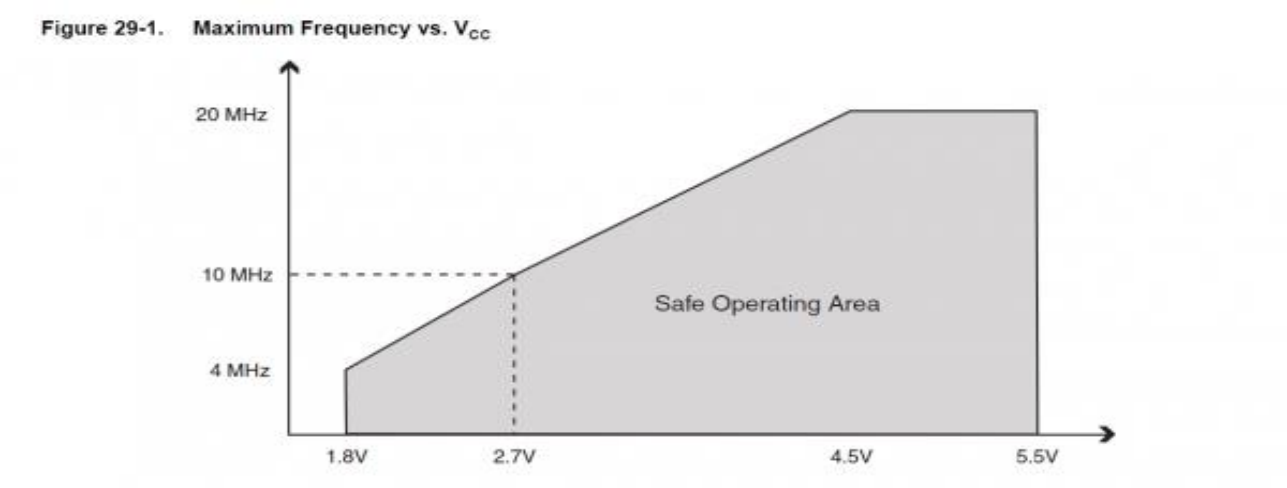


Figure 30: clock speed and voltage [12]

[12] A SURVEY ABOUT POWER CONSUMPTION FOR ARDUINO

Reducing the Clock Speed

- In projects where doesn't need to execute a large number of instructions in a short amount of time or in projects where timing isn't an issue, reducing the clock speed of the microcontroller can shave a few milliamps off the supply current.[12]

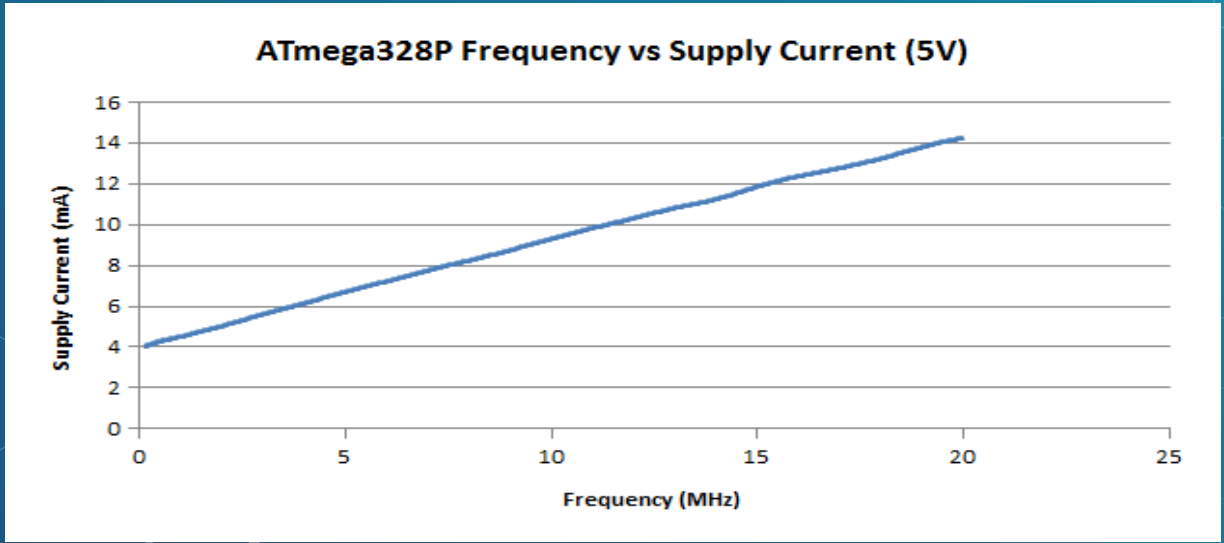


Figure 31: impact of frequency on current draw [12]

[12] A SURVEY ABOUT POWER CONSUMPTION FOR ARDUINO

6 Important pins

ADC, PWM

Analog to Digital converter (ADC)

- Everything in the physical world is an analog signal
  - Sound, light, temperature, pressure
- Need to convert into digital signals

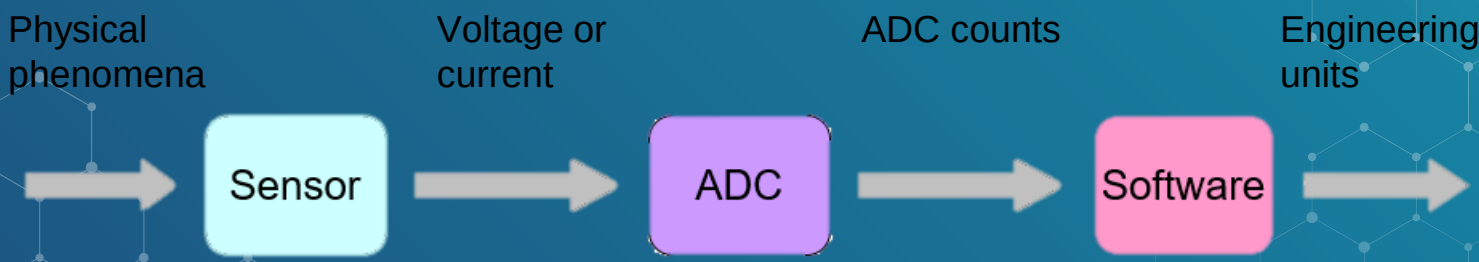
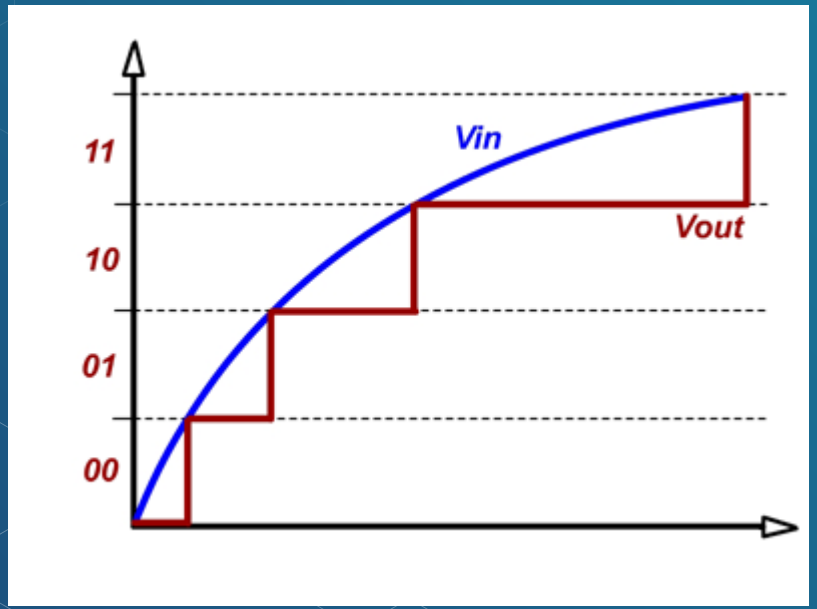


Figure 32: ADC [12]

[13] EECS 373 Design of Microprocessor-Based Systems

Analog to Digital converter (ADC)

- **resolution**



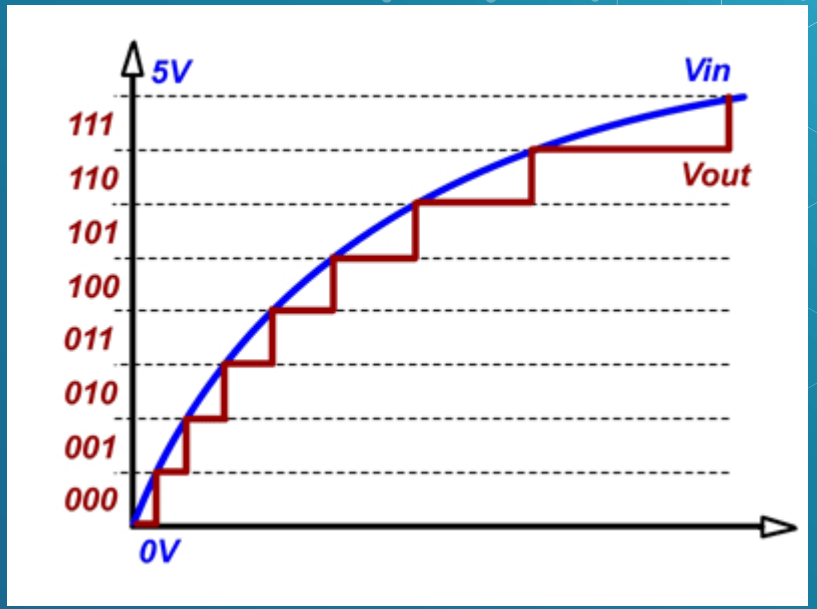


Figure 35: different resolutions

Analog to Digital converter (ADC)

- **Sampling time,** nyquist theorem

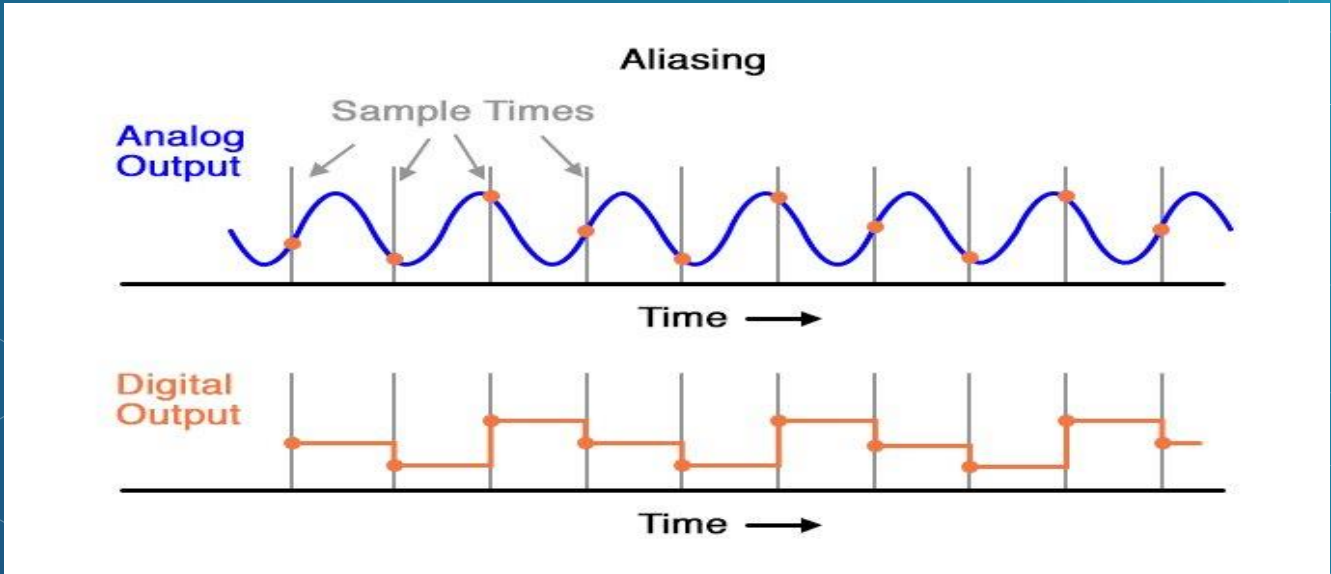


Figure 34: sample time

pulse width modulation (PWM)

- PWM is a way to control analog devices with a digital output
  - Drive analog devices like variable-speed motors, dimmable lights, actuators, and speakers

- PWM is not true analog output
  - PWM “fakes” an analog-like result by applying power in pulses, or short bursts of regulated voltage

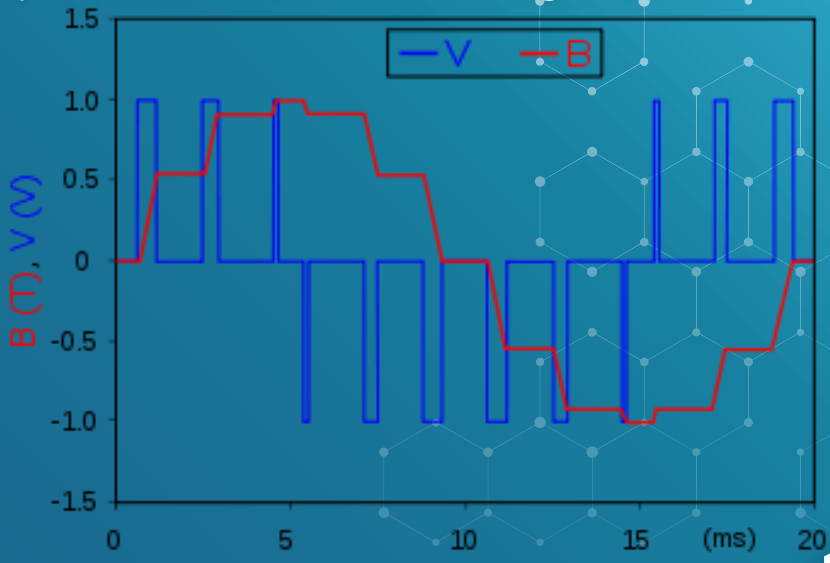


Figure 30: pwm

https://www.analogictips.com/pulse-width-modulation-pwm/

pulse width modulation (PWM)

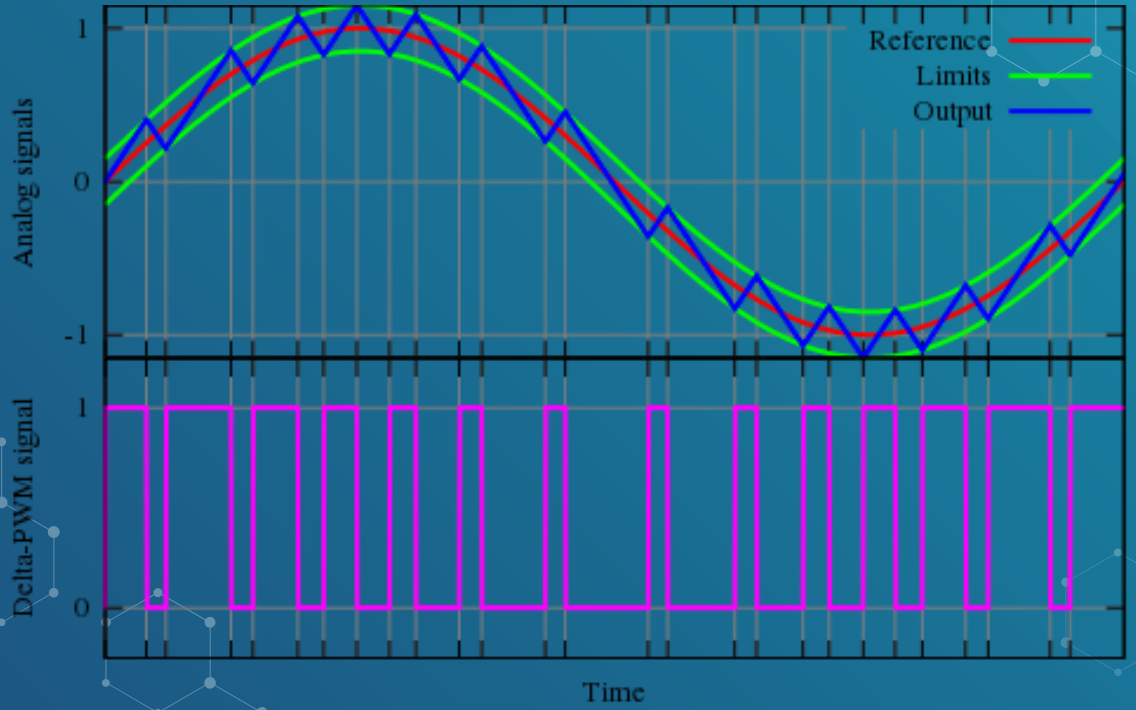


Figure 36: https://commons.wikimedia.org/wiki/File:Delta_PWM.svg

pulse width modulation (PWM)

- Duty cycle

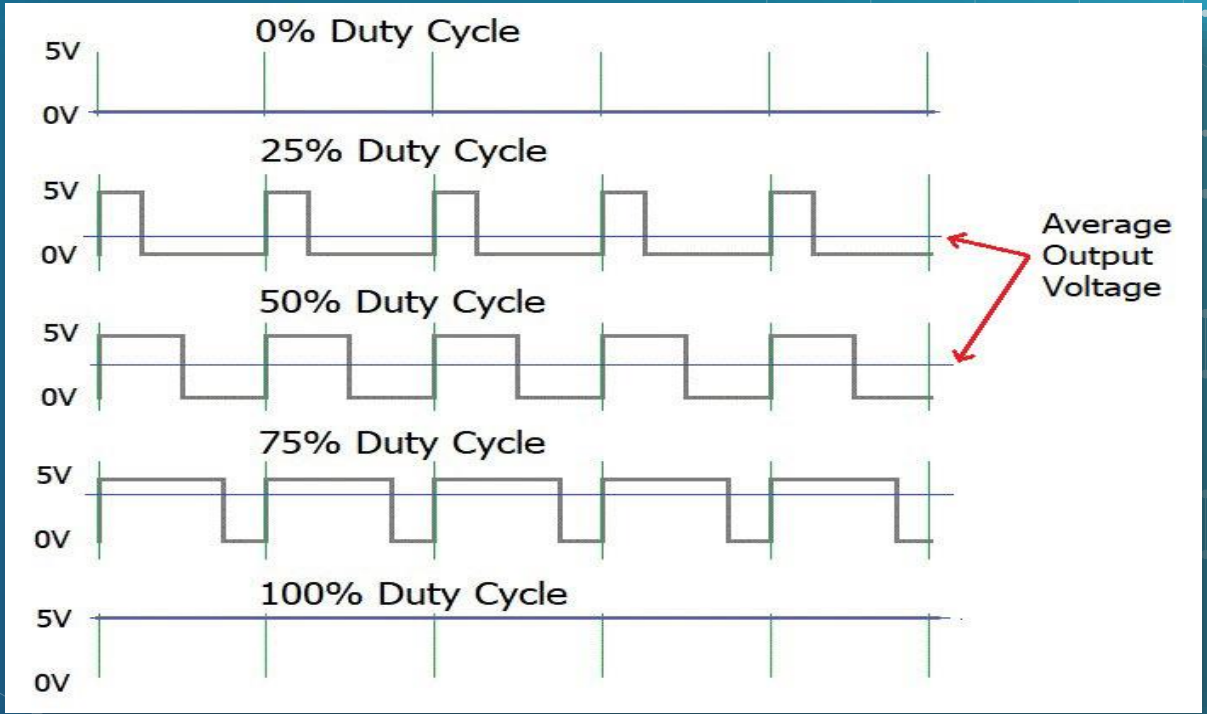


Figure 37: duty cycle

Reference

[1] Peter J. Ryan 1 and Richard B. Watson, *"Research Challenges for the Internet of Things: What Role Can OR Play?"* , systems journal, 14 March 2017.

[2] IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Thing, Cisco press, 2017

[3] LPWAN white paper, Leverege LLC, 2020 | www.leverege.com

[4] Internet of Things A to Z : Technologies and Applications, IEEE Press, 2018

[5] P.P.Ray, A survey on Internet of Things architectures, Journal of King Saud University - Computer and Information Sciences, Volume 30, Issue 3, July 2018, Pages 291-319.

[6] Abhi Raj and Dan Steingart, "Review—Power Sources for the Internet of Things", Electrochemical Society Journal, 25 April 2018.

Reference

[7] esp8266ex Datasheet

[8] 9B-esp8266-low_power_solutions

[9] Ricardo Reis, “Strategies for Reducing Power Consumption and Increasing Reliability in IoT”, IFIP International Internet of Things Conference, 2018 - Springer

[10] ATMega328P Dataheet

[11] Embedded Systems:Introduction to ARM®CORTEX-M Microcontrollers, Vol. 1, 5th Ed., Jonathan W. Valvano, 2014

[12] Sabou sebastian, “A SURVEY ABOUT POWER CONSUMPTION FOR ARDUINO” Carpathian Journal of Electrical Engineering, 2020

[13] EECS 373 Design of Microprocessor-Based Systems

THANKS!

You can find me at:

Email: hrhhemati@gmail.com

Telegram: @Hamidrezahematii