IoT Internet of Things

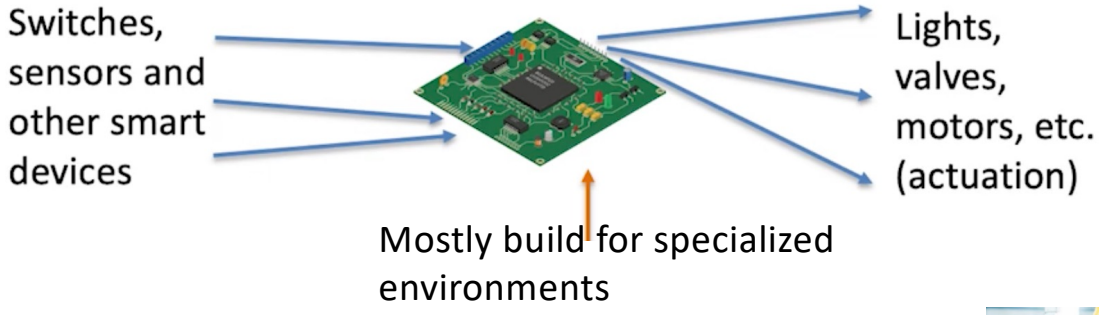
Hartmut Seitter

Agenda – Topics which should be covered in this course

- Introduction
  - IoT Definition
  - What are the disciplines of IoT
  - A brief history about IoT
- Application scenarios for IoT
  - Cool applications and a lot of opportunities
- The IoT Value Stack a real world example
- Sensors
  - Wireless, sensor, networks and IoT
  - From Sensors and Actuators to ….
  - … embedded systems and computation
- **Circuits**
  - **Energy and Wireless**
  - **Digital & Analog**
- **Embedded Systems**
  - **Technology Drivers,**
  - **Energy,**
  - **Microcontroller,**
  - **Software**

PLCs in IoT

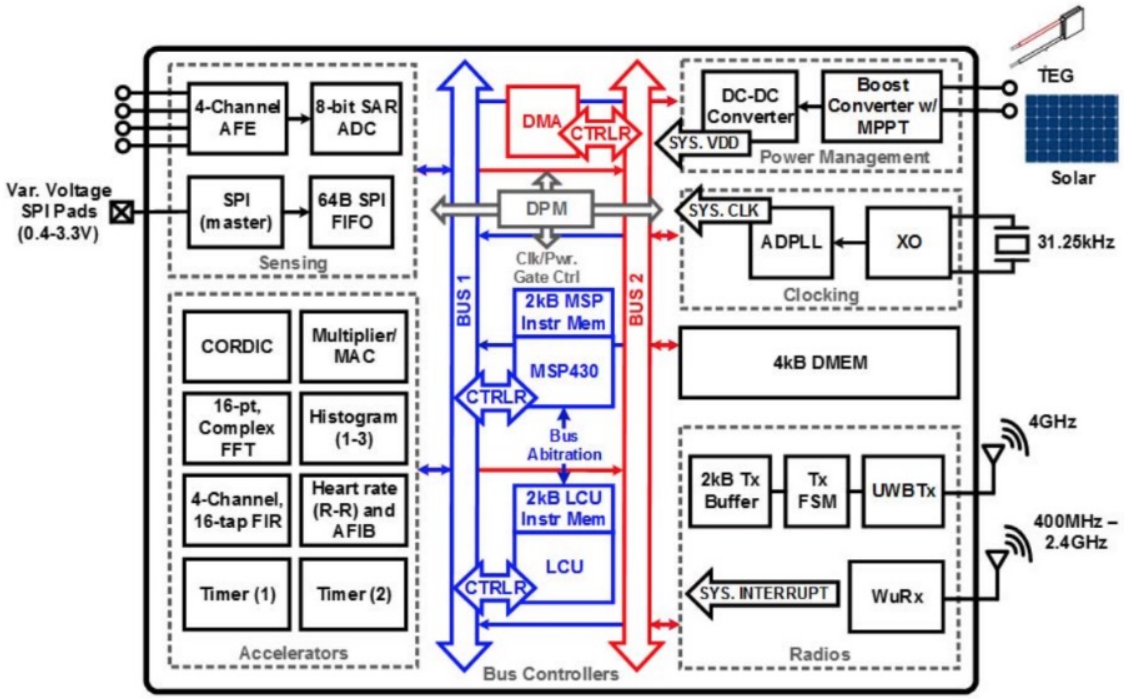
- **Programmable logic computers**
  - **Miniature industrial computers that contain hw and sw used to perform input translation**

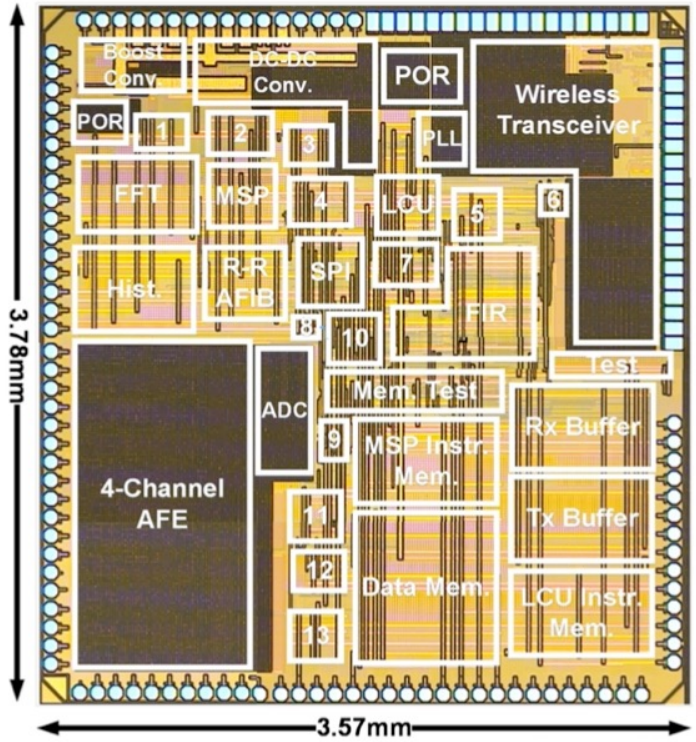


- **Human machine interface**
  - **Interface between human and machine to control something**
  - **Operating system:**
    - **Specialized. Linux or Windows-based**

- **Machines and Robots**
  - **Perform the actuation**
  - **Often equipped with sensors**
  - **Trend is to make the machine more autonomous (ML)**

How does a typical IoT device look like – it's the MSP430

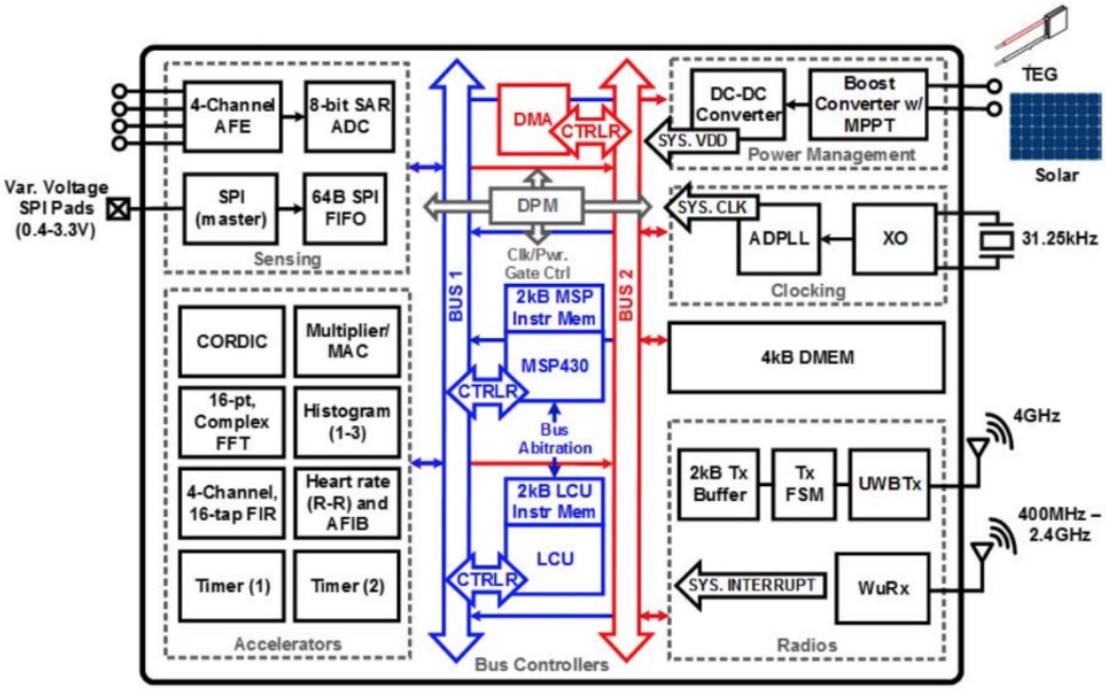




Under the hood of an IoT chip

Talk to the outside world
IO Ports (digital and analogue)

Accelerator / in HW implemented
Functions such as Filter, Fourier-Transformation



Energy management

Communication

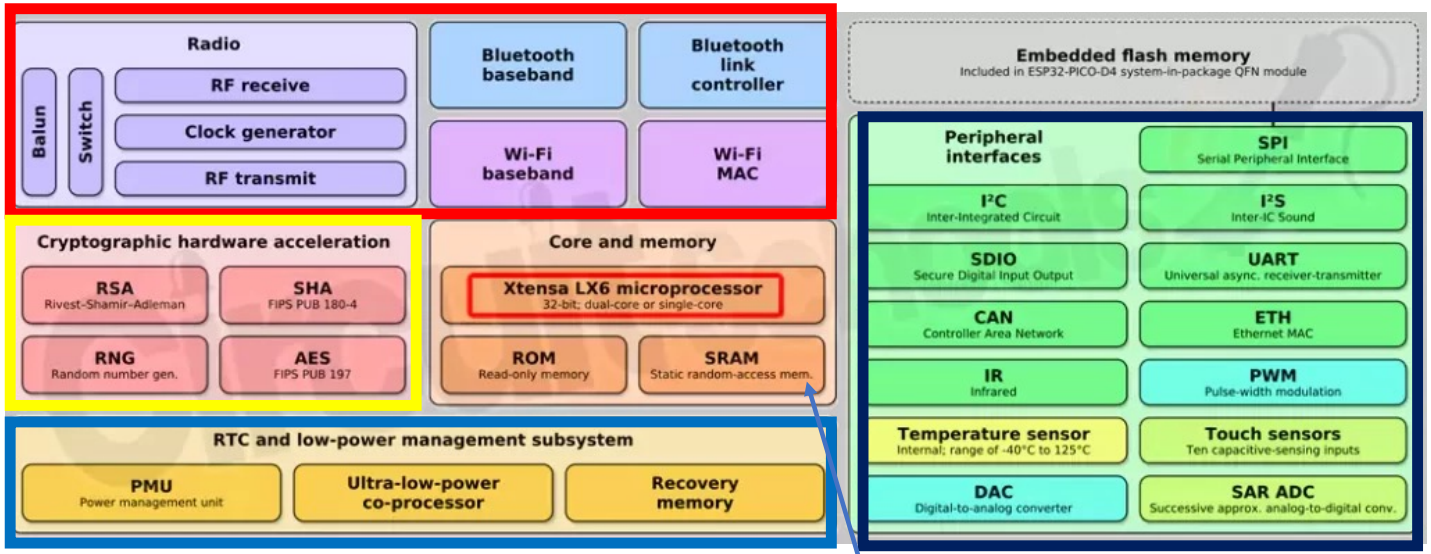
WiFi, BLE, ……

CPU processor unit
Bus system

Another common PLC – ESP32 Architectural Block Diagram –

Depending on the type it may vary

Wifi connectivity



HW Accelerator

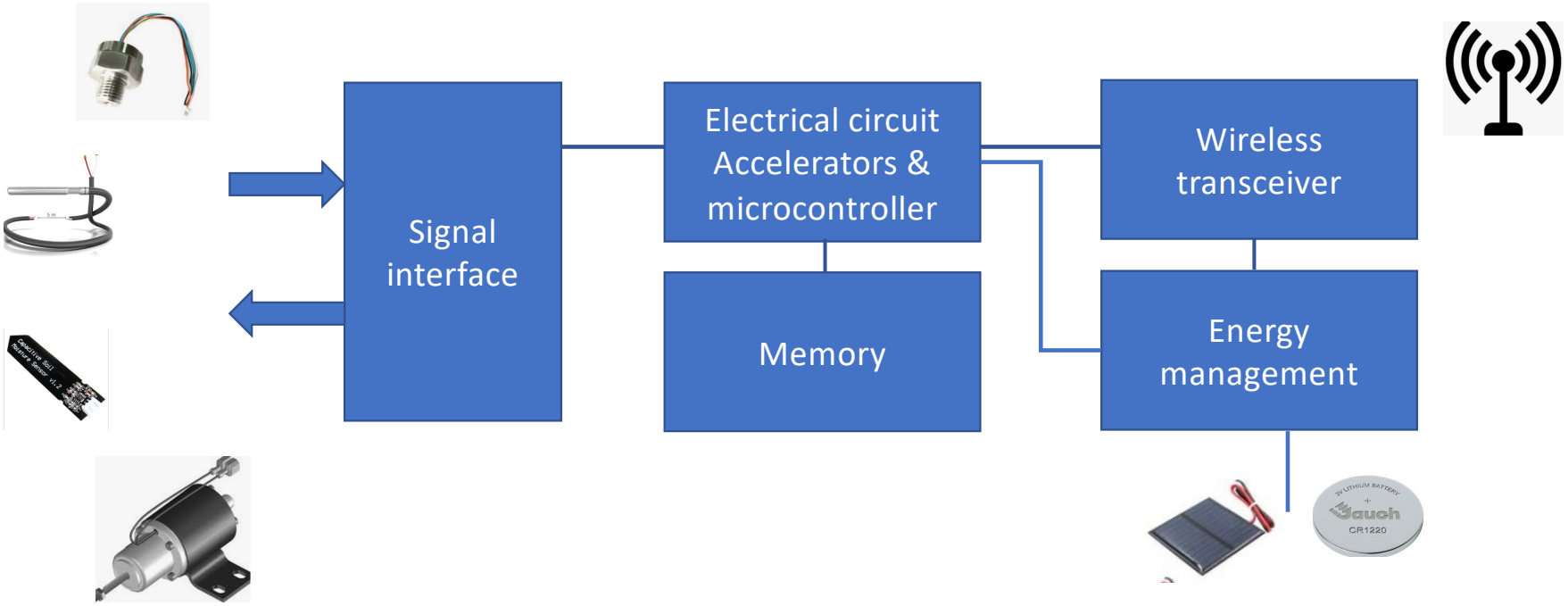
Device Interfaces

Power Management
-different power modes

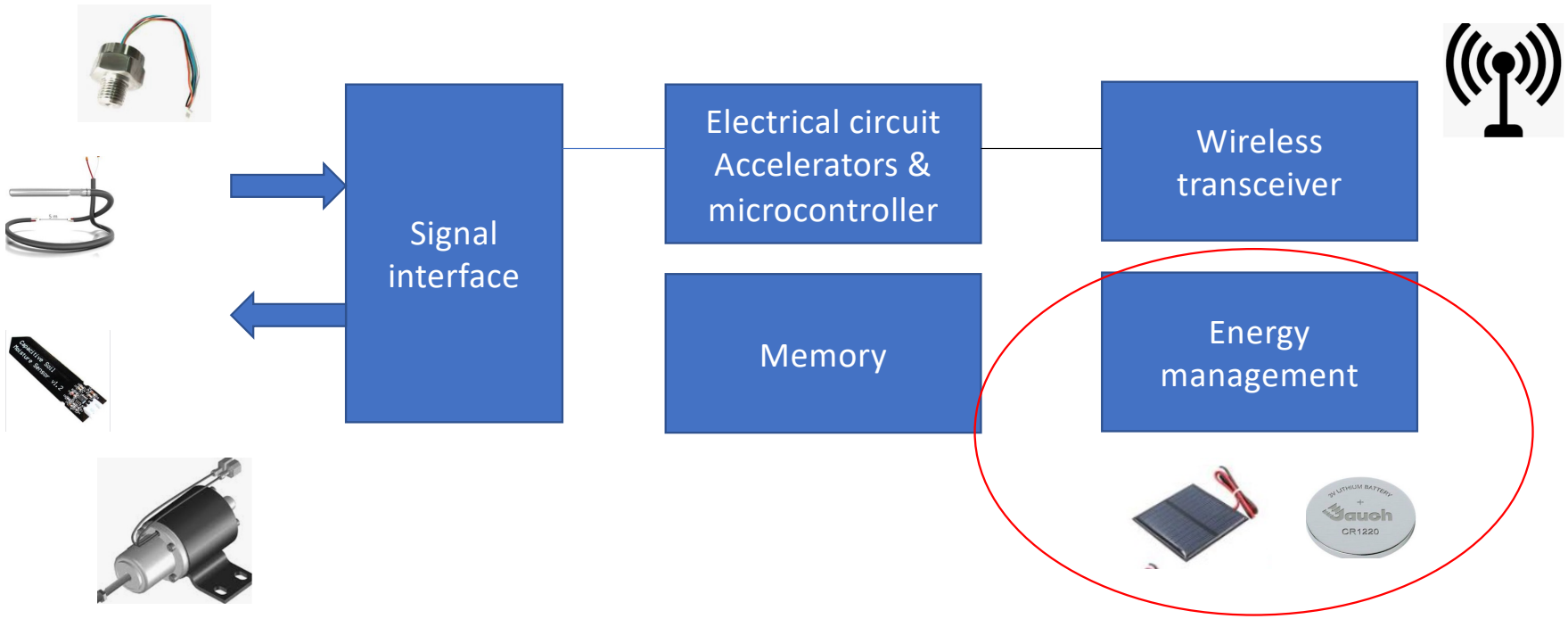
Processing unit

Generic block diagram – typical IoT device

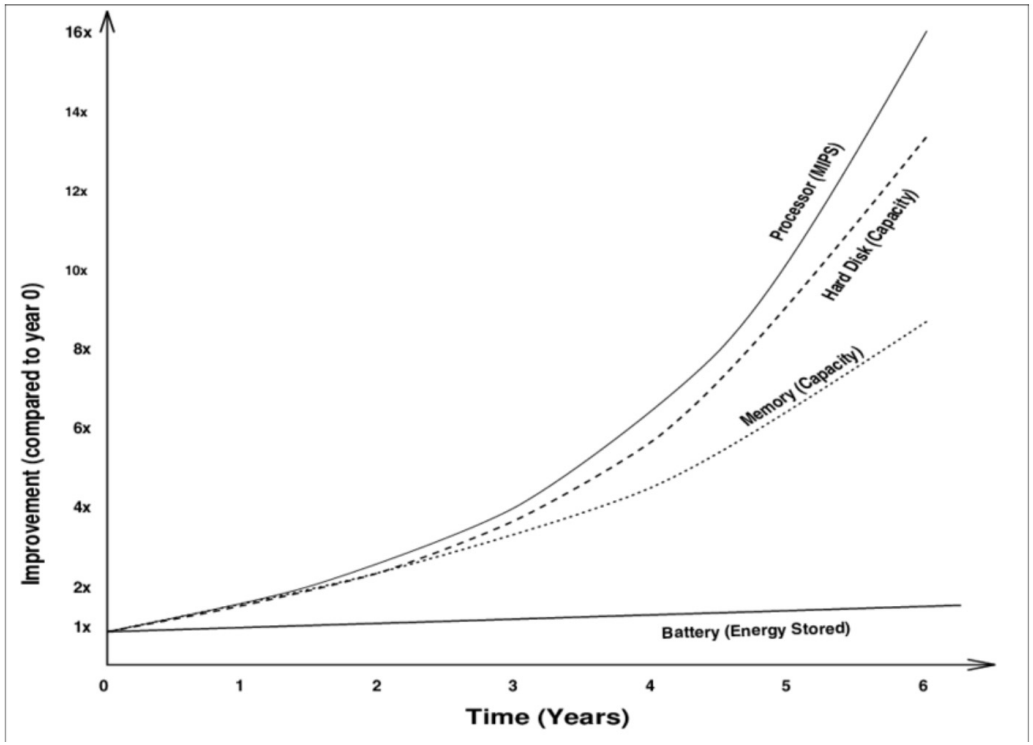
Sensors and Actuators



Generic block diagram – typical IoT device -. The energy management

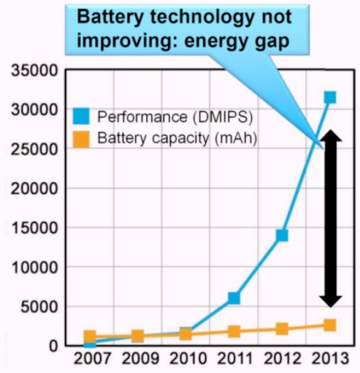


Improvements “battery capacity vs other electronic components” over time



This is still valid also in the last 5 years

Batteries improve slowly



- Semiconductor chip performance tends to improve rapidly
  - See e.g. the blue curve on the left for microprocessors
  - DMIPS = Dhrystone Million Instructions per Second
- Battery capacity improves slowly
  - About 5-8% per year
  - Doubling in ~10 years
- **Key challenge: How to fill this gap via creative circuit & system design?**

Tsai, 2014 International Solid-State Circuits Conference (ISSCC), Keynote Talk

Energy management in ‘battery powered IoT Devices’ is still a big challenge

- **Very often battery powered IoT devices are often installed in a remote location**
- **Very often the IoT devices are very small, and large batteries cannot be used**
  - **Changing batteries is very expensive**
  - **The battery lifetime should be designed for years**
  - **Constant voltage is required to guarantee operation of the electronic components (sensor and microcontroller)**
  - **Efficient power management is required**
    - **Power reduction and**
    - **Power harvesting**

    is essential

In microcontroller special operating modes are available to reduce power consumptions

A view words about the energy management to illustrate the problem

	AA Batteries	CR2032	CR2	CR132A
Voltage	3V	3V	3V	3V
Capacity	2700mAh or more	225mAh	950mAh	1500mAh
Retention	88% after 24 months			
weight	46g	2,8g	11g	17g
height	45mm	3,2mm	15,6mm	32mm
diameter	10,5mm	20mm	27mm	17mm

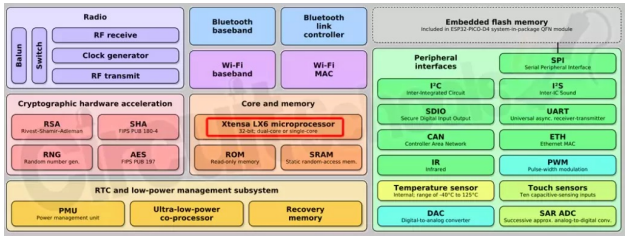
Simple power consumption sample

- **Estimate the average current that can be sustained for a battery life of 10 years?**
  - **Battery AA with 2500mAh**
  - **Average current = 2500mAh/365*10*24h = 28,5.µA**
    - **The LED operational time 2500mAh/20mA appr. 5 Tage**
  - **Battery CR2032 with 225mAh**
  - **Average current 225mAh/365*10*24h = 2,56 µA**
    - **The LED operational time 225mAh/20mA appr. 11 hour**
- **Typical microcontroller requires something around 200..250mA**
  ->>> about 10…15hours

Artikel-Nr.: KBT L-7104GD
LED, 3 mm, bedrahtet, grün, 20 mcd, 40°

Typ:	LED
Farbe:	grün
Wellenlänge:	565 nm
Lichtstärke:	20 mcd
Gehäuseausführung:	diffus, grün
Abstrahlwinkel:	40 °
Betriebsstrom:	20 mA
Betriebsspannung max.:	2,2 V
Betriebstemperatur:	-40 ... +85 °C

The microcontroller ESP32 components and it's power consumption



- **The Wifi connectivity**
  - Approximately 80-240 mA on 3,3V power consumption depending on mode of operation
  - The ESP32 supports mainly the following wireless communication protocols
    - WiFi
    - Bluetooth and BLE
    - Thread (experimental)

    - *There are boards available which support other communication protocols such as Zigbee, LoRaWAN, ….. by using additional radio modules*

- **The CPU**
  - Approximately 80-260 mA on 3,3V power consumption depending on mode of operation

- **The peripheral interfaces**
  - UART, SPI, I2C**:** active interfaces consume around 0.1-10 mA during data transfer, depending on data rate and load.
  - ADC (Analog-to-Digital Converter): approx. 100-150 µA per channel when active.
  - DAC (Digital-to-Analog Converter): around 100-150 µA during operation.

esp32 – we will use it in the IoT Lab – power consumption

Table 6: Power Consumption by Power Modes

Power mode	Description			Power consumption
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 15 for details.
	Wi-Fi/BT Tx packet			
	Wi-Fi/BT Rx and listening			
Modem-sleep	The CPU is powered on.	240 MHz *	Dual-core chip(s)	30 mA ~ 68 mA
			Single-core chip(s)	N/A
		160 MHz *	Dual-core chip(s)	27 mA ~ 44 mA
			Single-core chip(s)	27 mA ~ 34 mA
Power mode	Description			Power consumption
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA
			Single-core chip(s)	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	The ULP coprocessor is powered on.			150 μA
	ULP sensor-monitored pattern			100 μA @1% duty
	RTC timer + RTC memory			10 μA
Hibernation	RTC timer only			5 μA
Power off	CHIP_PU is set to low level, the chip is powered off.			1 μA

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ESP32 – power consumption on Wifi communication

$$P(W) = 10^{\frac{P(dBm)}{10}} \times 1\,mW$$

Remark:
0 dBm = 1 mW
20dBm = 100mW

Table 15: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

Remark:
Smartphone on 4G app.
23…27dBm (200…500mW)

Example
The cr2032 battery has a capacity of app. 225mAh
When all this capacity can be used for BLE transmitting which is not realistic at all!!
You can switch on the BLE Transmitter in the esp32 for appr. 1,7h

You see – power consumption is a very hot topic of IoT devices

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ESP32 – power consumption on Bluetooth and BLE communication

$$P(W) = 10^{\frac{P(dBm)}{10}} \times 1\,mW$$

Remark:
0 dBm = 1 mW
10dBm = 10mW

Table 5-12. Transmitter Characteristics –Bluetooth LE

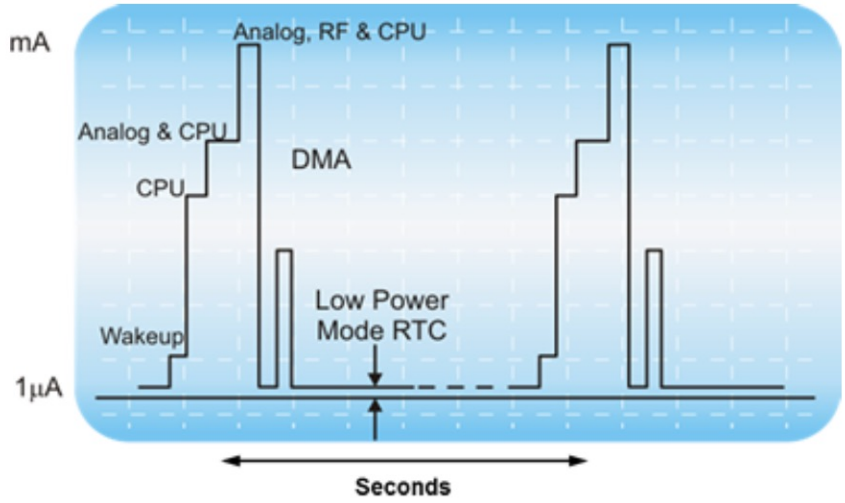
Parameter	Description	Min	Typ	Max	Unit
RF transmit power (see note under Table 5-8)	—	—	0	—	dBm
Gain control step	—	—	3	—	dB
RF power control range	—	–12	—	+9	dBm
Adjacent channel transmit power	F = F0 ± 2 MHz	—	–52	—	dBm
	F = F0 ± 3 MHz	—	–58	—	dBm
	F = F0 ± > 3 MHz	—	–60	—	dBm
$\Delta f1_{avg}$	—	—	—	265	kHz
$\Delta f2_{max}$	—	247	—	—	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	—	—	0.92	—	—
ICFT	—	—	–10	—	kHz
Drift rate	—	—	0.7	—	kHz/50 μs
Drift	—	—	2	—	kHz

Bluetooth designed for continuous audio streaming, file transfers, and higher data rates. (Constant sending and receiving data required more power)

BLE is specifically optimized for low power consumption, suitable for devices like sensors, fitness trackers, and IoT devices that transmit small amounts of data intermittently. (BLE is engineered to minimize active radio time, therefore less power is consumed compared to classic Bluetooth)

Energy management – embedded device

Typical mode of operation of an IoT controller



Most of the time the IoT controller will be in the low power mode (sleep or better deep sleep).
Only for a view msec the IoT controller will be turned on for a measurement and transmitting data

The power consumption calculation is very complicated, and you have to know a lot of details of the mode of operation

https://uk.farnell.com/calculating-battery-life-in-iot-applications#calculator

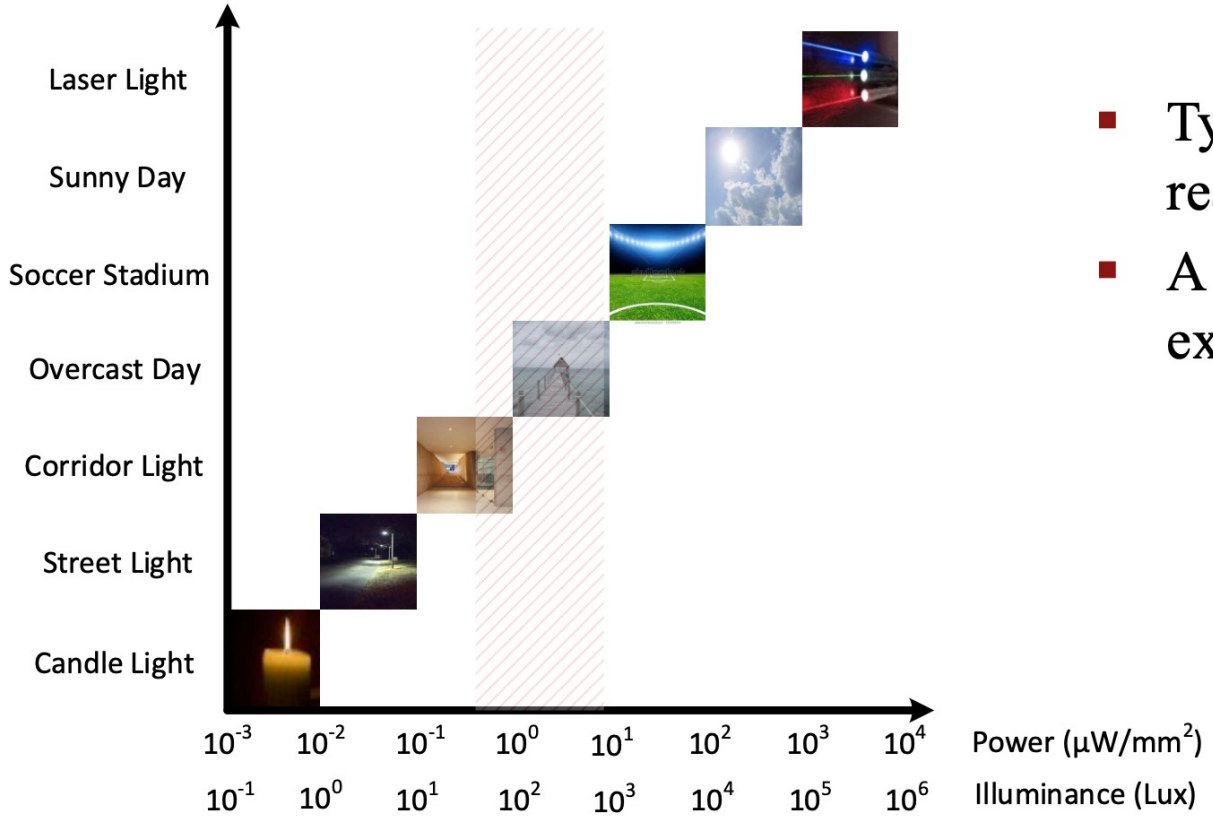
Energy management – embedded device – energy harvesting

Most popular

Source	Limitation	Power Density
Inductive Coupling	Short range (cm), inefficient	$\propto D, Q, 1/d^3$
Far-field RF	Base station (a few m), safety	$\propto 1/d^2$
Solar (Indoor)	Available artificial lighting	10 $\mu W/cm^2$
Solar (Outdoor)	Direct sunlight	10,000 $\mu W/cm^2$
Vibration	Relatively constant movement	4 $\mu W/cm^2$
Thermoelectric	Thermal gradient	25 $\mu W/cm^2$

A. Klinefelter et al., "A 6.45µW self-powered IoT SoC with integrated energy-harvesting power management and ULP asymmetric radios," 2015 IEEE International Solid-State Circuits, pp. 384-385, Feb. 2015.

How much solar power can we expect?



- Typically expect 1 µW/mm^2 under reasonable lighting conditions
- A few cm^2 of solar cells may help extend the lifetime to “infinity”

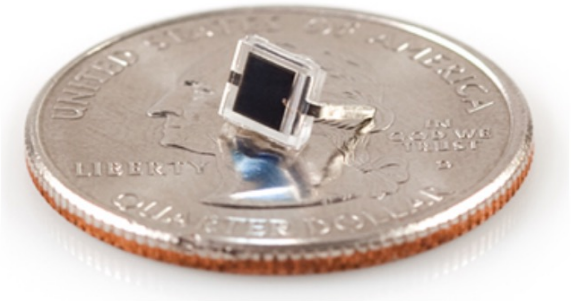


Chart by Nishit Shah, Stanford

Energy management – embedded device – energy harvesting DC-DC converter

- **We need a constant voltage on our microcontroller for operation!**

- **The energy harvesting sensor does not provide a constant voltage!!!**
  - **We cannot use the energy from the sensor directly (sensors -> light, vibration, temperature difference, etc.)**
  - **DC-DC converter in combination with a storage capacitor (cloud also be a rechargeable battery) are used to provide an 'constant' output voltage for our microprocessors**
  - **The DC-DC converter are also high sophisticated circuits.**

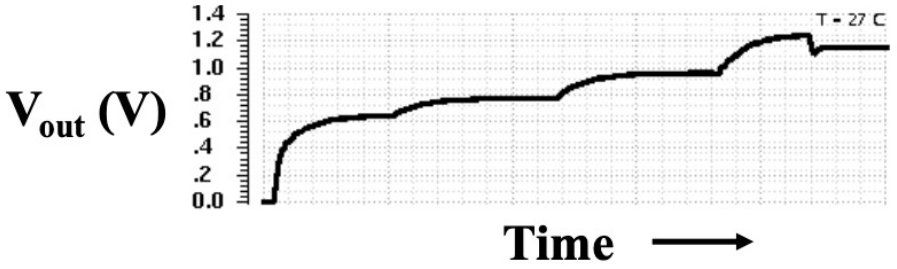
For the interested students -> to charge a capacitor in one step (switch voltage on and capacitor will be loaded) you will lose app. ½ of the energy (Elektrotechnik)

The solution in the energy harvesting DC-DC converter the storage capacitor will be loaded in several steps which result in not loosing so much energy – energy gain about factor 10!!!

Energy management – embedded device – energy harvesting DC-DC converter

Step charging example

$$C_{store} = 100nF$$
$$V_{in} = 0.8V$$
$$V_{out} = 1.3V$$



Energy loss without step charging:

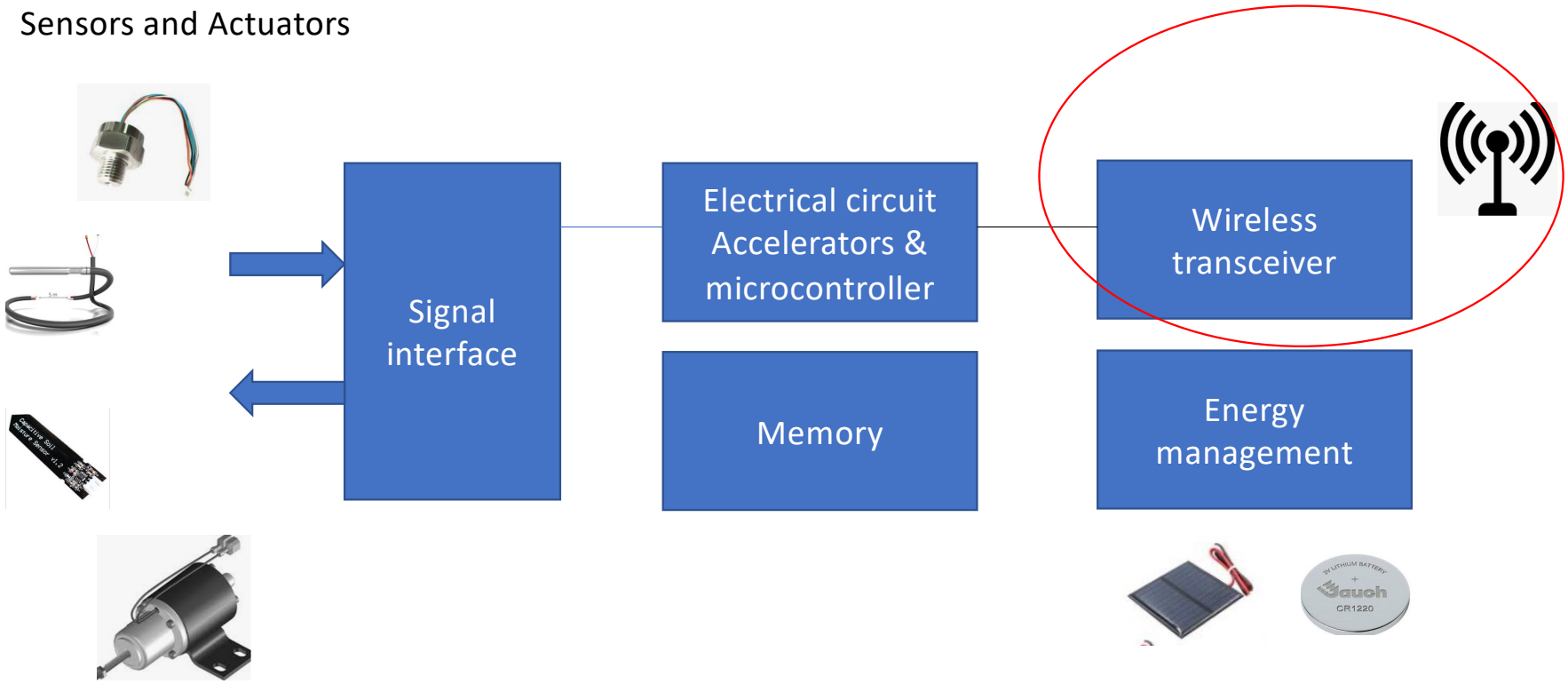
$$E_{loss} = \frac{1}{2} C_{store} V_{out}^2 = 192\ nJ$$

Energy loss with step charging:

$$E_{loss} = \sum_k \frac{1}{2} C_{store} \left(V_{out,k} - V_{out,k-1}\right)^2 = 18\ nJ$$

M. Saadat and B. Murmann, "A 0.6V - 2.4V Input, Fully Integrated Reconfigurable Switched-Capacitor DC-DC Converter for Energy Harvesting Sensor Tags," in Proc. IEEE Asian Solid-State Circuits Conf., Nov. 2015.

Generic block diagram – typical IoT device



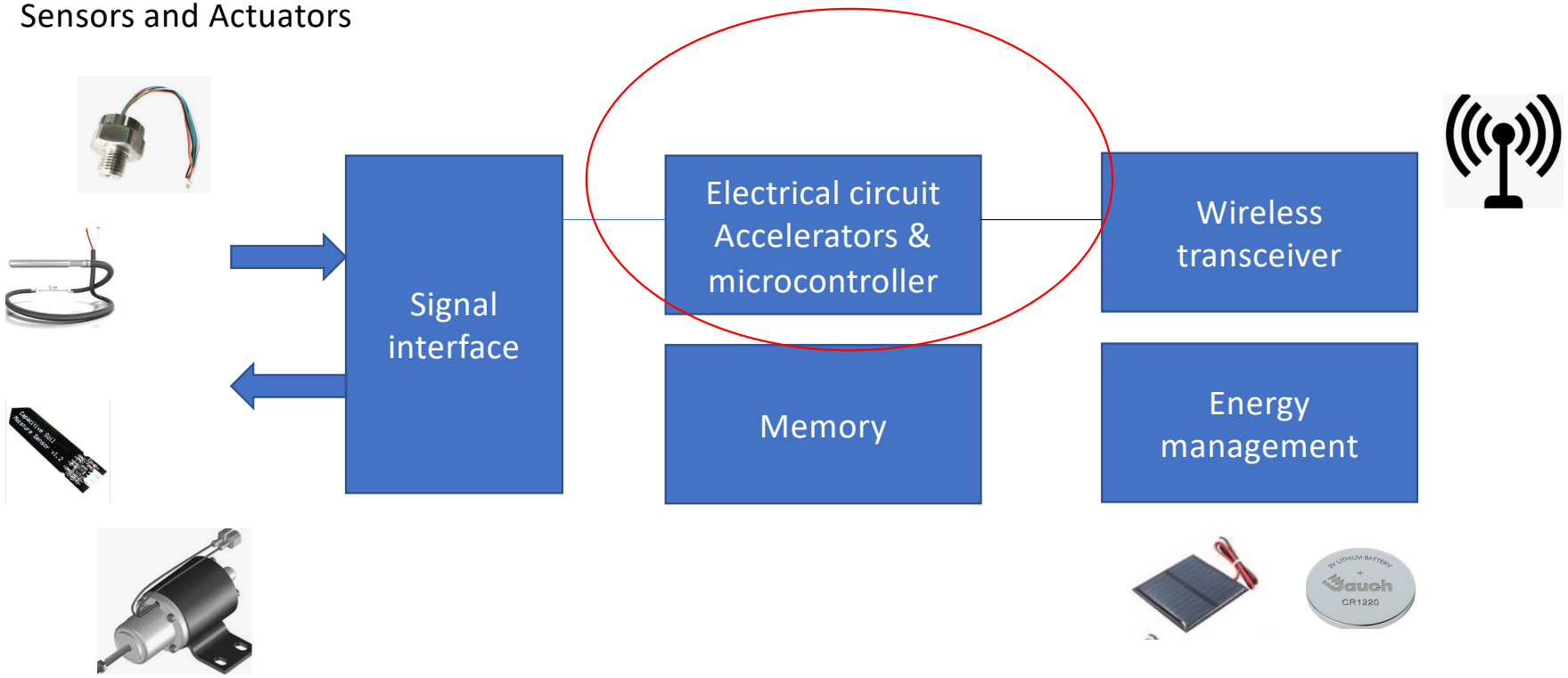
ESP32 wireless connection functions

- **The radio module consists of the following blocks:**
  - 2.4 GHz receiver
  - 2.4 GHz transmitter
  - Clock Generator
  - Implements a TCP/IP and full 802.11 b/g/n Wi-Fi MAC protocol.
  - Bluetooth up to 4Mbps data rate
  - BLE Modes Advertising, scanning, multiple connections, asynch data reception and transmitting

Atmega *processor (arduino boards) does not have internal wireless connectivity capabilities, they use external modules for WiFi capabilities*

Raspi Zero *supports*
802.11 b/g/n wireless LAN.
Bluetooth 4.1.
Bluetooth Low Energy (BLE)

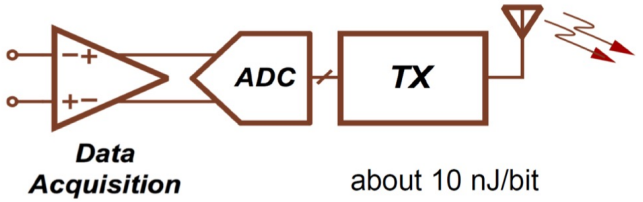
Generic block diagram – typical IoT device



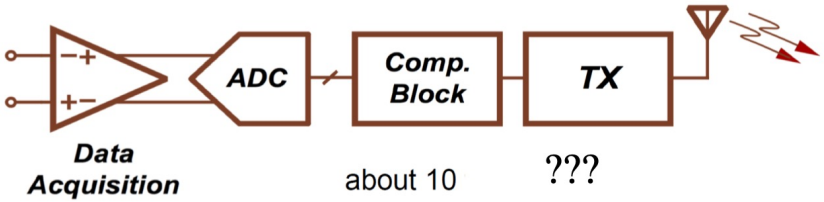
Transmit or not to transmit?

- **We learnt – to transmit data is expensive from a power consumption point of view**
- **How about adding some signal processing to reduce to number of bits to transfer?**
- **What about data compression?**

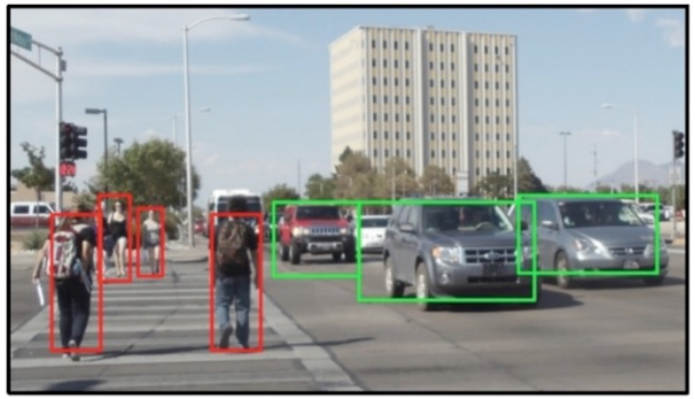
Sense → Communicate:



Sense → Compute → Communicate:



Energy of image processing accelerators



Let’s assume you have an IoT solution which takes pictures every xx minutes

Now you have to decide whether you transfer the raw image or whether you do some post processing before you transfer the image

Post processing e.g. video compression

-> video compression takes app. the same amount of energy as transferring a pixel

->object detection on the edge – it takes less energy than transferring the data

A lot of research work currently takes place

- in object detection post processing can reduce the amount of data transmitted by the factor of 20.

DSP – Digital Signal Processing –sample use cases – Hardware Accelerator

- **Audio Signal Processing:**
  - Noise reduction in microphones.
  - Echo cancellation.
  - Audio equalization.
  - Voice recognition systems.

- **Image Processing:**
  - Filtering, edge detection, and image enhancement.
  - Stereo image processing.
  - Example: Real-time video streaming enhancement or object detection.

- **Sensor Data Analysis:**
  - Filtering vibrations from accelerometers.
  - Analyzing ECG signals in medical devices.
  - Example: Detecting frequencies of machinery vibration for predictive maintenance.

- **Communication Systems:**
  - Modulation/demodulation.
  - Noise filtering in wireless signals.
  - Error detection and correction.

DSP – Digital Signal Processing – FFT – Fast Furrier Transformation

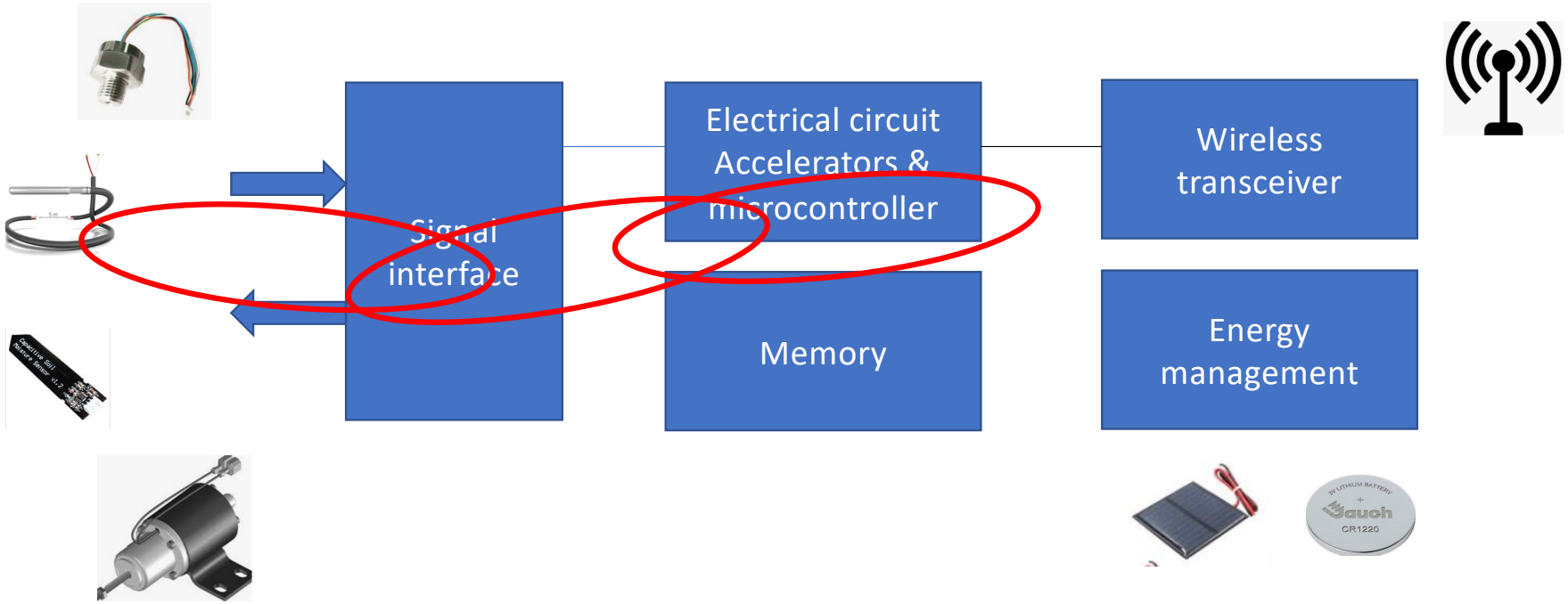
- Suppose you have a microphone or analog sensor connected to the ESP32's ADC pin, and you want to analyze the frequency components of the signal in real-time—like detecting specific tones or identifying frequency content in sound, vibration, or other signals.

- **When to use FFT:**
  - Analyze the frequency spectrum of an incoming signal.
  - Detect dominant frequencies or monitor frequency changes over time.
  - In audio processing, vibration analysis, or spectrum sensing in IoT.

- **Processing sequence**
  - **Sample analog input:** Reads a signal at regular intervals into a buffer.
  - **Apply windowing:** Reduces spectral leakage.
  - **Compute FFT:** Converts time domain signal to frequency spectrum.
  - **Find peak frequency:** Finds the dominant frequency component.
  - **Output:** Prints the peak frequency to Serial Monitor.

Generic block diagram – typical IoT device

Sensors and Actuators



How to link sensors and actuators to microcontroller

- **Digital 0 and 1 signal – between 0 and max. voltage representing binary 0 and 1**

  Remember the discussion on hardware switches and their behaviour

- **Analog signal – between 0 and max. voltage representing a binary value between 0 and number of digits the AD converter provides**
  - 9 bit resolution 0 ...511
  - 10bit resolution 0…1023
  - 11bit resolution 0…2046
  - 12bit resolution 0…4095
  - Attention on ESP32 there are ADC1 and ADC2 AD GPIO Pins – ADC2 share resources with other resources like WiFi Bluetooth …

  - Digital to analog Pins are also available

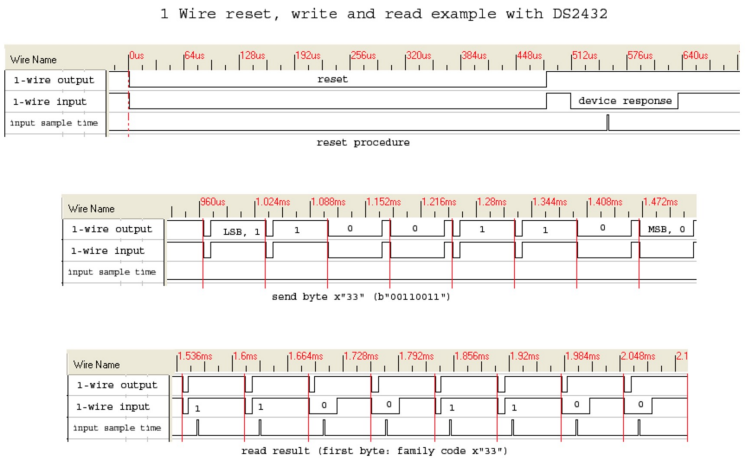
  - Touch sensors GPIOs are also available on an ESP32

Energy management IoT devices

- There are two major components on IoT devices which must be considered from an energy management point of view
  - The signal interface
    - The AD converter is used in most IoT solutions, so will discuss it during the MC Controller discussion
  - The microcontroller itself
    - We will discuss the energy management of the microcontroller when we discuss the MC in general
    - What I can say at this point – MC might be the unit which consume the most energy especially the RF transmission
      - Take care about power saving modes of the microprocessor!!

How to link sensors and actuators to microcontroller

- **Serial communication**
  - **OneWire Protocol** (used e.g. Temperatur sensors DS18B20 and others)
  - It's a half duplex serial bus protocol designed by Dallas Semiconductors
    - (half duplex = communication from master to slave or from slave to master at the same time)
  - **One Signal line is used for the communication (Ground is the other contact)**
  - **Device Addressing – each device on the Bus has a unique 64 bit address (operates on low power mode)**
  - **Multiple devices can be connected to the single data line**
  - **There is a bus muster initiating the activity on the bus (most of or Arduino or RasPi etc**
  - **The one wire combines the clock and data in a single wire**



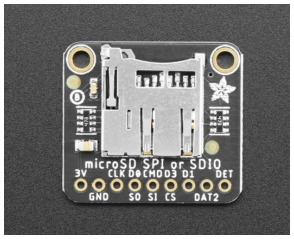
How to link sensors and actuators to microcontroller

- **Serial communication**
  - **SPI Protocol** (Serial Peripheral Interface)
  - It's a full duplex serial bus protocol
    - You can communicate from master to slave and from slave to master at the same time

  - **Several Signal Lines are required**
    - **MOSI = Master Out Slave In**
    - **MISO = Master In Slave Out**
    - **SCLK = Serial Clock**
      - Each clock puls allows data bits to be sent and received
    - **SS/CS = Slave Select Chip Select**

  - **SPI protocol provides a high data rate communication between master and slave**

  - **E.g. SD Card modules are provided with SPI Interface**



  - **More than one slave device can be connected on the same SPI lines however for each slave device separate CS Line (Signal) is required**

How to link sensors and actuators to microcontroller

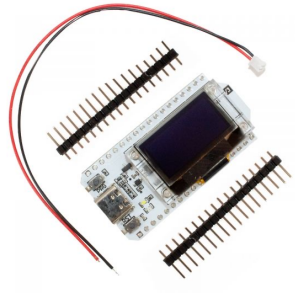
- **Serial Communication**
  - **I2C Protocol** (Inter integrated Circuit)
  - It's a half duplex serial bus protocol
    - You can transmit data in one direction at a time.

  - **Two lines are required for communication**
    - **SCL = Serial Clock Line**
    - **SDA = Serial Data Line**

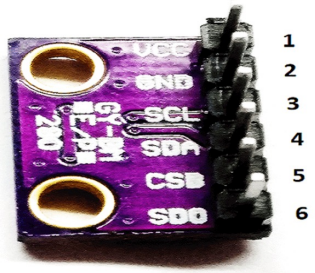
  - **It is also a master slave protocol – the master controls the clock and initiates the communication**

  - **I2C provides different communication speeds**
    - **Standard mode 100kbit/sec**
    - **Fast mode 400kbit/sec**
    - **Fast mode plus 1Mbit/sec**
    - **High speed up to 3,4Mbit/sec**

How to link sensors and actuators to microcontroller



- **Serial Communication**
  - **I2C Protocol** (Inter integrated Circuit)
    - Start Condition: The master initiates communication by pulling the SDA line low while the SCL line is high.
    - Addressing: The master sends the address of **the target slave** and a read/write bit.
    - Acknowledgment: The addressed slave acknowledges by pulling the SDA line low.
    - Data Transfer: Data is transmitted in byte format, with each byte followed by an acknowledgment from the receiver.
    - Stop Condition: The master releases the SDA line high while the SCL line is high to signal the end of communication.
  - **A lot of sensor and actuators use the I2C Bus interface e.g. the BMP280**
    - **Check the data sheet for the I2C address**
  - **ATTENTION**
    Check the I2C SDC and SCL pins whether they require a pull a resistor. Very often the are implemented as open Collector output



GY-BMP280 Module Pinout

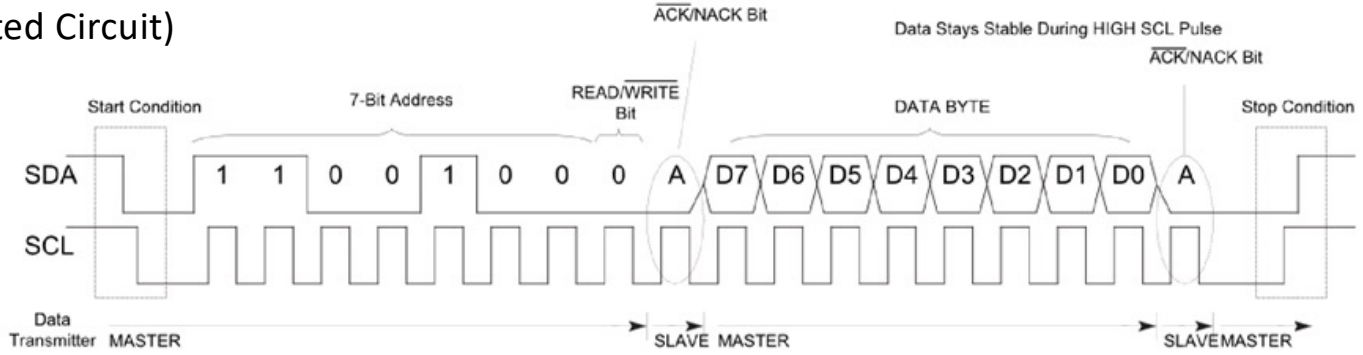
Leave pin 6 of the module (SDO) unconnected to set the I²C address to 0x76 – the on-board resistor pulls the SDO pin low setting the address to 0x76.

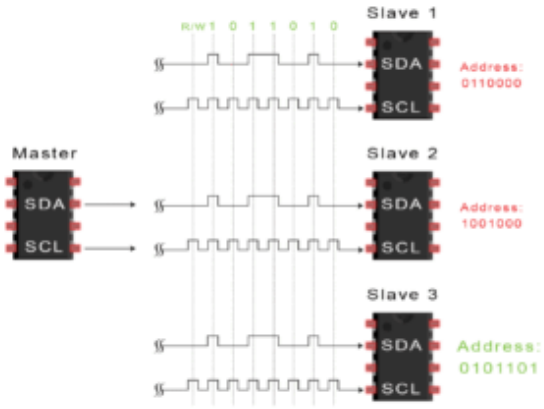
To change the I²C address to 0x77, connect pin 6 of the module (SDO) to Vcc which would typically be the 3.3V supply.

How to link sensors and actuators to microcontroller

Serial Communication

I2C Protocol (Inter integrated Circuit)





How to link sensors and actuators to microcontroller

- **Serial Communication**
  - **UART Protocol** (Universal Async Receiver Transmitter Protocol)

    - **It is a commonly used protocol for short distance communication between devices**
    - **It supports full duplex mode (for RX and TX different lines are used)**

    - **It does not have a clock line/signal**

    - **To start the communication a START BIT is sent**
    - **The data bits follow (typically 5…9 bits)**
    - **A parity bit for error checking**
    - **Check the data sheet for the**
    - **And a STOP bit**

GPS Modules often use the UART protocol
It also very often in Industrial Automations used

Camera interface and a lot more

How to link sensors and actuators to microcontroller

- **Other serial communication protocols**
  - **RS485 Protocol used in industrial environments for long distance communication**
    - **2 wires are used**
    - **Up to 32 devices can be connected to the bus**
    - **Communication speed up to 10kps**
    - **Support half and full duplex (full duplex required 4 wires)**
  - **RS232 Protocol**
    - **Simpler than RS 485, Point-to-point (typically 2 devices), suitable for short-range, commonly used in computers and peripherals.**

How to link sensors and actuators to microcontroller

Serial interfaces differences

FULL NAME:	UART	I2C	SPI
COMPLEXITY:	Very low	Low, even with many devices	Complexity rises as more devices are added
MINIMUM WIRES REQUIRED:	2	2	4
DUPLEX:	Full-duplex (send and receive over the same connections at the same time)	Half-duplex (send and receive over the same connections but not at the same time)	Full-duplex (send and receive over the same connections at the same time)
TYPE OF COMMUNICATION:	Asynchronous (messages do not need to be consistent in length)	Synchronous (all transmissions look the same, a requirement for synchronous buses)	Synchronous (all transmissions look the same, a requirement for synchronous buses)
NUMBER OF TARGETS:	Connections are normally device-to-device	Up to 128 when using a 7-bit address, with a default of 27	In theory, a serial chain can support any number of targets
NUMBER OF CONTROLLERS:	Connections are normally device-to-device	Multiple controllers can be supported	1
DISTANCE:	Up to 1000 m	Up to 100 m	Over 10 m
SPEED:	Normally up to 5 Mbs	Up to 5 Mbs	As much as 100 Mbs

https://www.avnet.com/wps/portal/us/resources/article/comparing-common-mcu-interface-protocols/?srsltid=AfmBOorGMFJerDc_cd2ArO6jhvWP9E7C7_6UihZ9I8bV0FanDMt1g6x0

Protocol/ Standard	Max. Nodes	Max. Bit Rate [kbps]	Max. Length [m]	Advantages	Disadvantages	Notes
UART	2			1, Full-duplex 2. uses only two wires 3. Can provide both synchronous and asynchronous communication	1. Can connect only two devices 2. controller should adjust settings to the controlled device 3. Usually slower compared to I2C and SPI	1. Configurable baud rate 2. Can use only one wire if one-sided communication is required 3. Uses internal IC clock system 4. Basic IC to IC or device to device communication
I2C	127 or 1023	5000	Undefined	1. Uses only two wires 2. Multi-master and multi slave 3. More slaves do not require more pins 4. More reliable than UART	1. Slower compared to SPI 2. More complex hardware than SPI 3. Slaves should have addresses defined	1. Requires pull-up resistors 2. Basic IC to IC communication
SPI	Depends on SS pins	Up to 10000	Undefined	1. Full-duplex 2. very simple drivers 3. More reliable than UART 4. Very fast serial communication 5. Multi slave communication	1. Uses three or more wires (traces) 2. More slaves increase pin count 3. Only one master 4. Slaves cannot communicate freely with each other.	1. Basic IC to IC communication 2. Low-resolution display/image data
1-Wire	2^48	16.3	300	1. Requires only one wire 2. Parasitic power configuration 3. Very low power consumption 4. Very simple few pin devices	1. Only Maxim devices have this serial communication	1. Requires Pull-up
CAN	128	1000	500	1. Very robust 2. Multi slave 3. Error detection	1. Quite expensive	1. 120 Ω impedance transmission line 2. Uses differential pair 3. Used primarily on automotive electronics
LIN	16	19.2	40	1. Very cheap 2. Requires only one wire 3. Can have up to 15 slaves	1. Slow data rate	1. Used primarily on automotive electronics
RS-485	256	Up to 10000	1330	1. High achievable data rate 2. High achievable communication distance 3. High receiver sensitivity	1. Higher power consumption 2. Pretty complex hardware	1. Uses one or two differential pairs 2. Device to device communication
RS-232	2	128	15	1. Cheap	1. Slow data rate 2. Modern devices rarely have this connection or do not use this standard 3. Low receiver sensitivity	1. It can be seen often in many previous-generation devices 2. There are a lot of converters 3. Device to device communication

https://resources.altium.com/p/comparing-all-serial-communications-protocols

Embedded Systems and microcontroller

- **Before we discuss the microcontroller for IoT devices in more detail I would make a clear statement**

The "Thing" in IoT

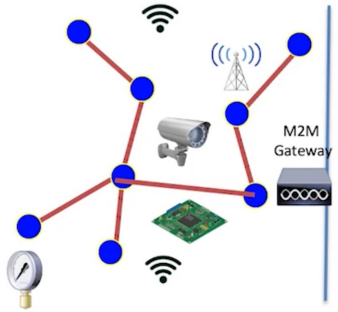
Computing systems are designed for a particular app
smart light
medical implant
personal fitness tracker
engine controller
…

Cost and energy define the design significantly

Hardware and software are often selected together, driven by the application requirements

- **In the IoT Lab and further lectures we will focus on ESP8266 / ESP32 / Arduino Boards / RasPi which are more or less *general purpose* microcontroller / minicomputer**

Access domain –there are different use cases



Each type of device uses a type of access technology that matches its requirements
how much data, how often, sent at what distance by how many devices

Gateways

Laptops, tablets, high throughput, high power consumption

WiFi Gateway

IoT Gateway

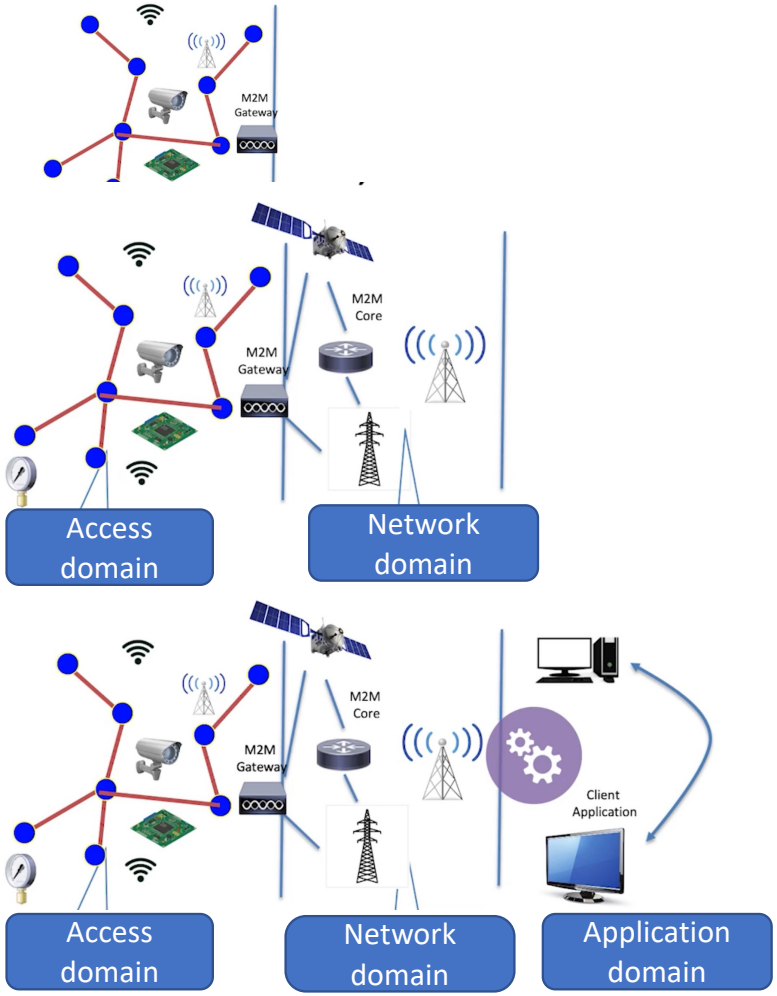
Satellite or cellular distribution link

Optica fiber distribution link

IoT: low throughput, low battery consumption

Gateways provide a bridge between different Communication technologies They translate protocols and may provide additional functions (e.g. aggregation, computation)

Access domain –there are different use cases



Each type of device uses a type of access technology that matches its requirements
how much data, how often, sent at what distance by how many devices

Access protocol, IEEE 802 …, …. , etc

Bring data as fast, as reliably, and as securely as possible between the places where it is produced and place where it is consumed

LTE, MPLS, etc

“Application” is a generic term: can look at the object or the data it produces

Smart energy mgnt, connectivity mgnt, data analysis, fleet management

Fog Computing -> bring processing power closer to the devices

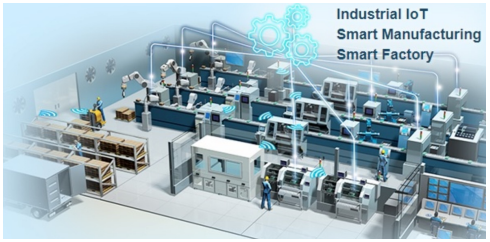
Processing data in the could causes bandwidth consumption
(one oil rig may produce 1TB/day)
Not all data is needed in the could
Processing some data closer to the IoT object makes sens

... Big Data problems of the marine sector .....

....the sheer volume of data. Even for the basic model of the seabed beneath a wind farm site, **10 terabytes of data are generated**. Much more data is added during a turbine life cycle. To constantly transfer such volumes of data would be time-consuming and expensive, even with fiber optic lines.

Embedded Systems – Connectivity and MGC (eMbedded Gateway Cloud) architecture

- **Remember IoT solutions are available nearly everywhere – the IoT Segments (lecture 1)**
- **IoT solution for different segments have different requirements and standards …..**



Industrial automation

Controlled environment
Industrial requirements
Controlled network
Thousands of units/people
Monitor and control machines

Typical networks
WirelessHART, 802.15.24, IEEE, IIC
LoraWan

Home area networks

Uncontrolled environment
Unlicensed spectrum
Convenience
Consumer requirements
Hundreds of units/person

Typical networks
ZigBee, ZWave, 6loWPAN, …
LoraWan

Personal Area Networks

Uncontrolled environment
Unlicensed spectrum
Convenience
Consumer requirements
Tens of units/person
Fashion vs. functions

Typical networks
Bluetooth/BLE/3G/LTE/IEEE

Network Devices

Uncontrolled environment
Unlicensed spectrum
Convenience
Tens of units/person
Powered

Typical networks
WiFi, 802.11, TCP/IP…

Embedded Systems – Connectivity and MGC (eMbedded Gateway Cloud) architecture

Some LoRaWAN use cases

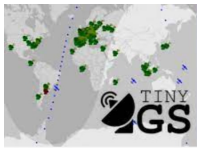
Internet of Things use cases for LoRaWAN – source Cisco Solution for LoRaWAN brochure – PDF opens

IoT MGC Architecture

embedded devices

6lowpan
ZigBee,
ZWave,
BLE,
WiFi,
WirelessHART
LoraWan
MQTT (TCP)
…

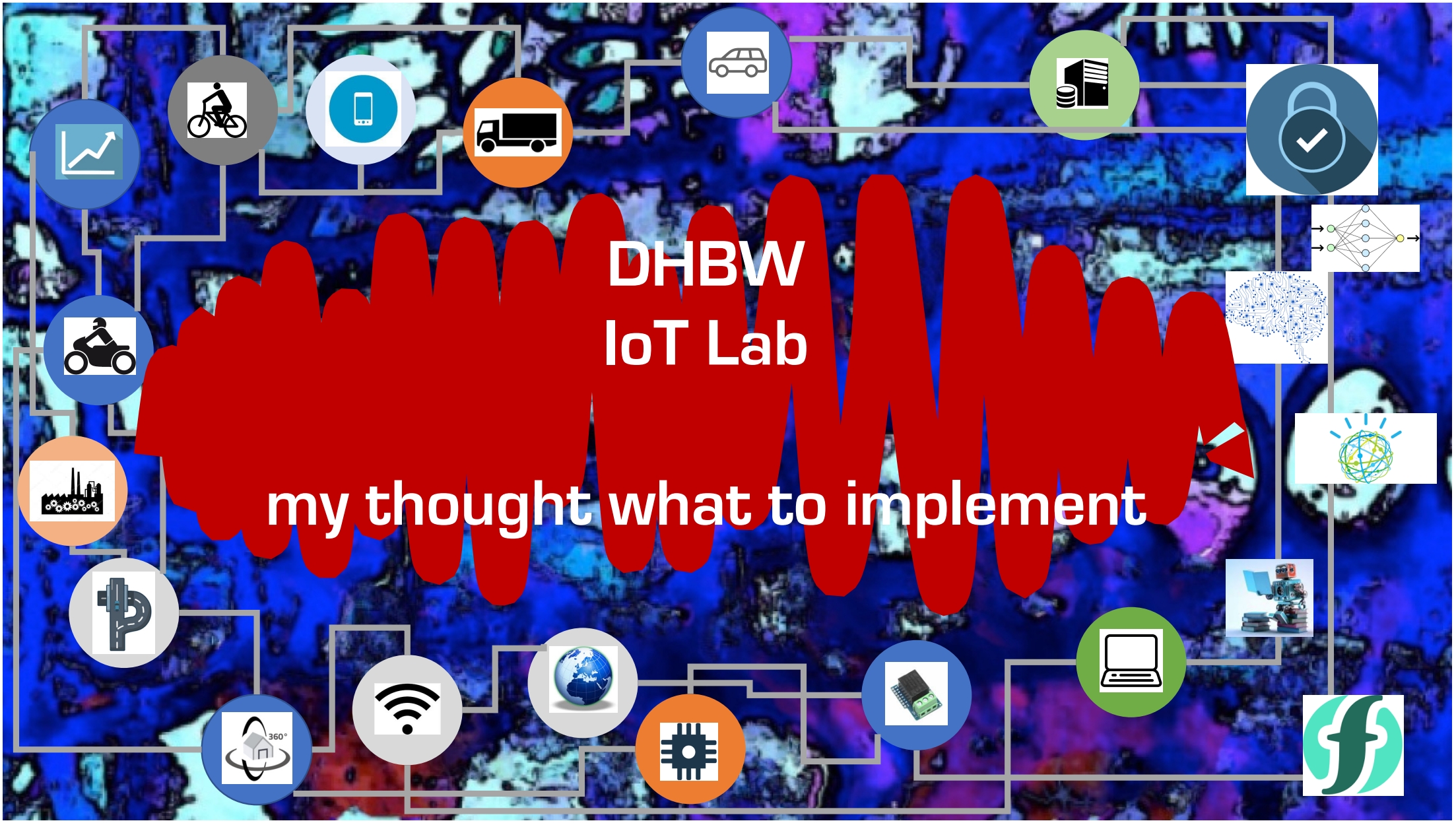
gateways



3G/4G/5G
Ethernet
TCP/IP
HTTP

MQTT

Cloud server environment


DHBW
IoT Lab
my thought what to implement

Goal of the IoT lab

- **Build a real IoT solution**
  - **Select Hardware and put it together**
  - **Develop a program which reads sensor data, do some pre-calculation, transmit data to a IoT gateway, forward it to Server app.**

- **Get familiar with IoT sensors (from a HW point of view)**
- **Practice some microcontroller programming (which is slightly different from other IT programming)**
- **Get familiar with IoT connectivity topics**
- **Learn how to use different communication protocols (BLE, LoraWan, WiFi, MQTT)**
- **Analyse the collected data using analytics functions**

Block diagram IoT Lab

Accelerator sensor

Gyroscope sensor

Moisture sensor

Ultrasonic sensor

Temperature sensor

Micro-controller

LoRaWAN

Connectivity Hub LoraWan e.-g. TTN

WiFi

WiFi

BLE

Connectivity Hub e.g. smartphone

LTE

WiFi

IoT server

Analytics on sensor data

MQTT will also be used

you can check this -https://tutorials-raspberrypi.com/raspberry-pi-sensors-overview-50-important-components/

My recommendations for getting started

- **Chose you scale**
- **Plans to expand, start small then expand step by step**
- **Review communication bricks**
- **RTFM (read the …… *fine* manual)**
- **Install and setup**
  - **Expect extra time for troubleshooting**

How to be prepared for the IoT Lab

- You need some hardware (sensor and microcontroller)
  - I have some stuff and I may order some stuff on request.
- You need a development environment to develop the microcontroller program
  - e.g. Arduino IDE or Visual Studio Code & PlatformIO
- You have to choose the commination network to transfer your data to a server
  - two or three tier architecture
    - e.g. BLE to a gateway (smartphone) LTE or WiFi to the server
    - WiFi direct to the server
    - LoRaWAN to TheThingsNetwork – TCPIP (MQTT) to the server
- A server side is needed to save and analyse the sensor data
  - Node-Red is an easy of use programming environment e.g. for IoT applications.

Today we can start to install Node-Red locally

Smart Home Levels

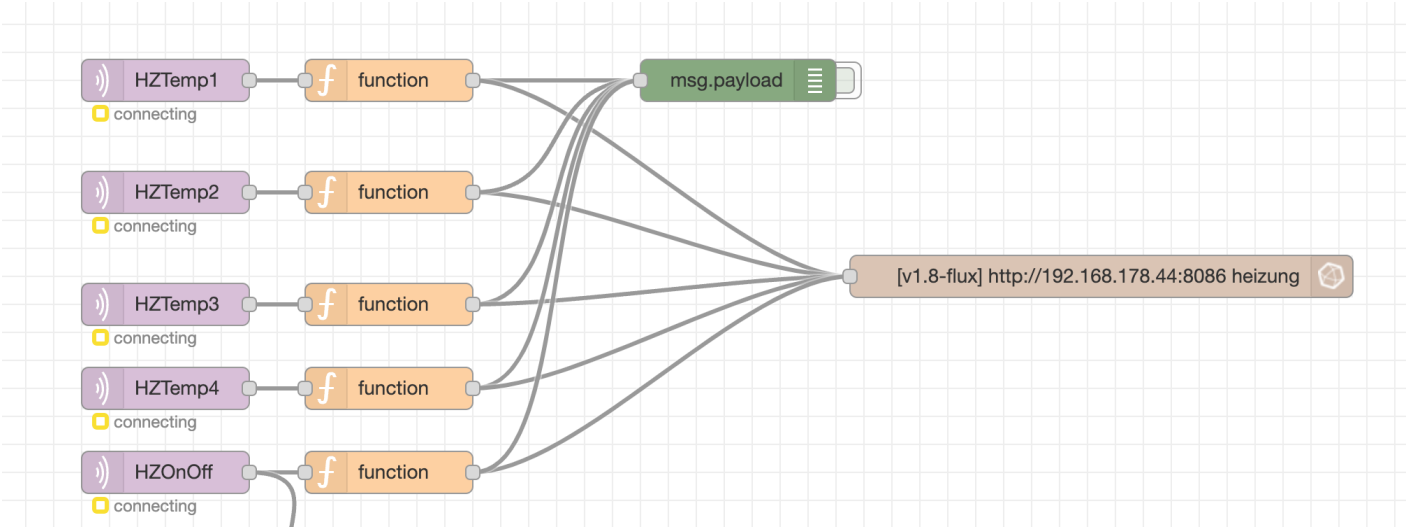
- **Level 1:** thermostat
- **Level 2:** sensors everywhere; on doors, windows, mailbox, appliance (power consumption, status), outside (weather, lawn humidity, wind level)
- **Level 3:** basic control (door locks, light up/down/color, audio volume up/down, machines start/stop, sprinklers on/off, with or without timer)
- **Level 4:** environmentally aware control (light color based on you, sprinklers start/stop based on lawn humidity and weather, door lock based on presence, music moves with you, etc.
- **Level 5:** home is an extension of you (smart fridge, smart oven, smart chairs/desks, etc.)

A few words about node-red

- **https://nodered.org/#get-started**

Nodes

A node-red app is assembled by nodes which are combined to a flow



Each node can perform a specific action

- An input node can receive messages form a source (MQTT, HTTP, ….)
- Processing node can process data (function node can be programmed in JavaScript)

A flow is a collection of connected nodes that define how the data moves through the app
Each flow can be deployed and executed separately

A Dashboard function is also available to create a user interface for monitoring and controlling the app