

# IoT Standard and Protocols

More Introduction Stuff

# Agenda

---

## Components of an IoT Application

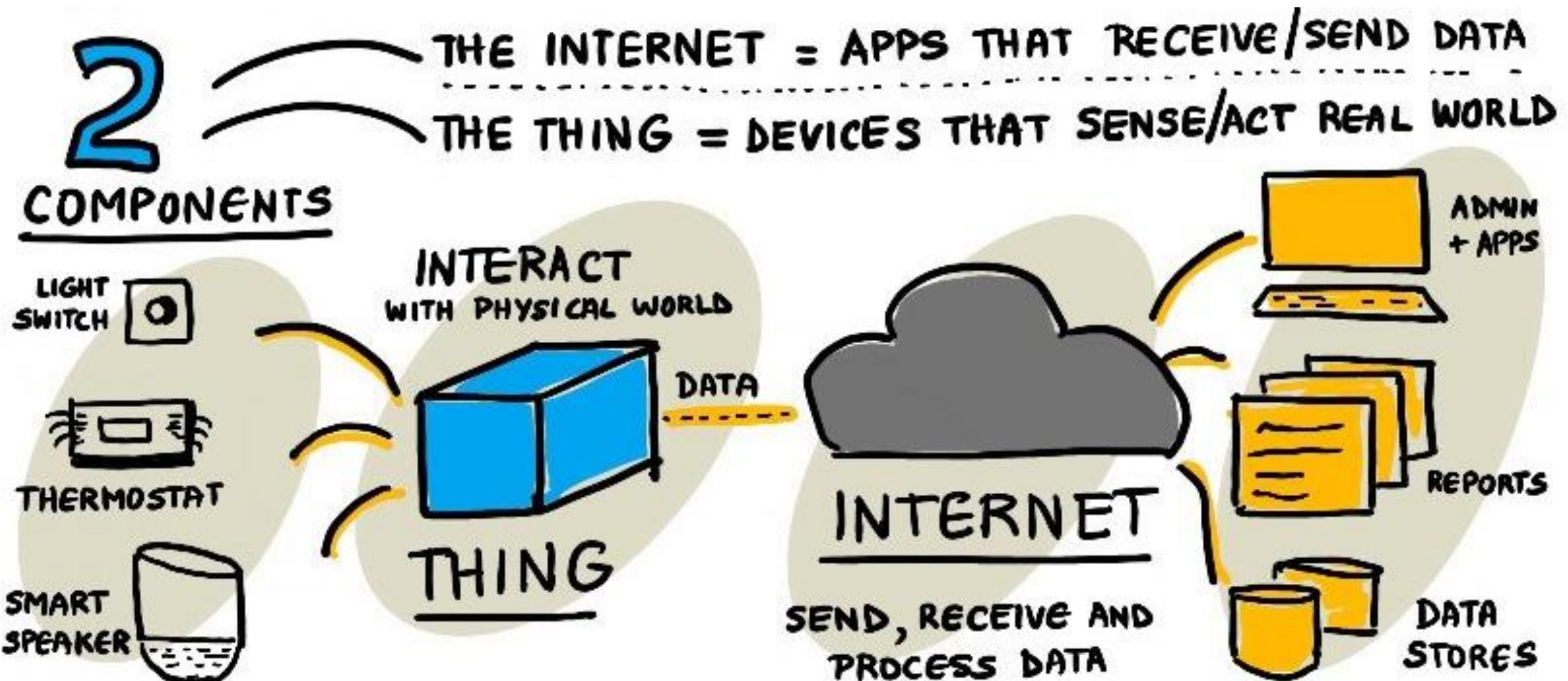
---

### Single Board Computers

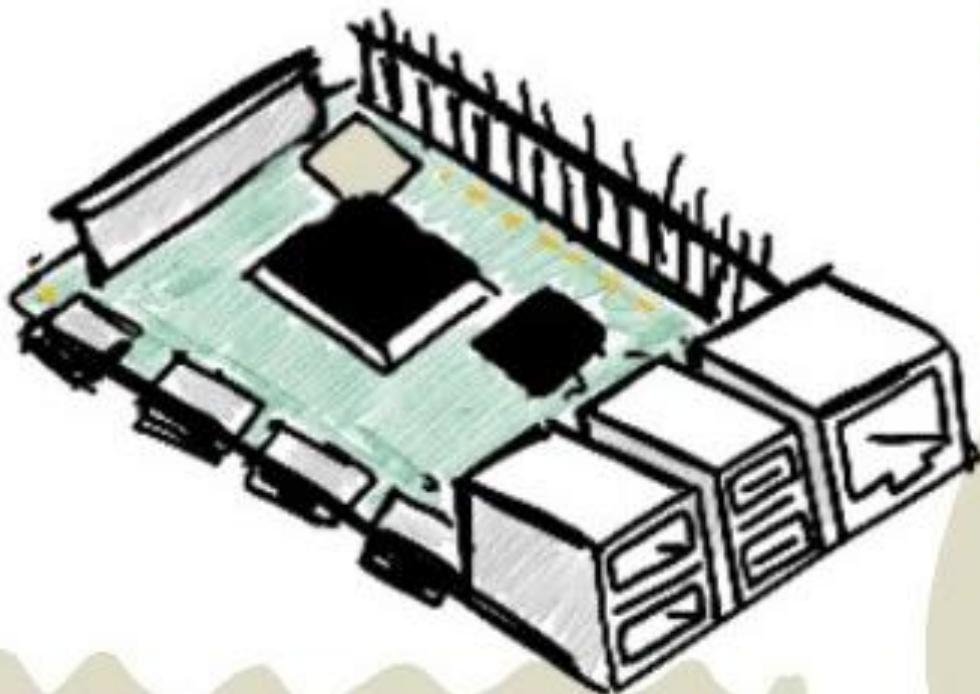
---

### Microcontrollers

# IoT Components



# The “Thing”



A DEVICE THAT CAN  
INTERACT WITH PHYSICAL WORLD

- LOW POWER
  - LOW COST
  - LOW SPEED
- COMPUTERS**

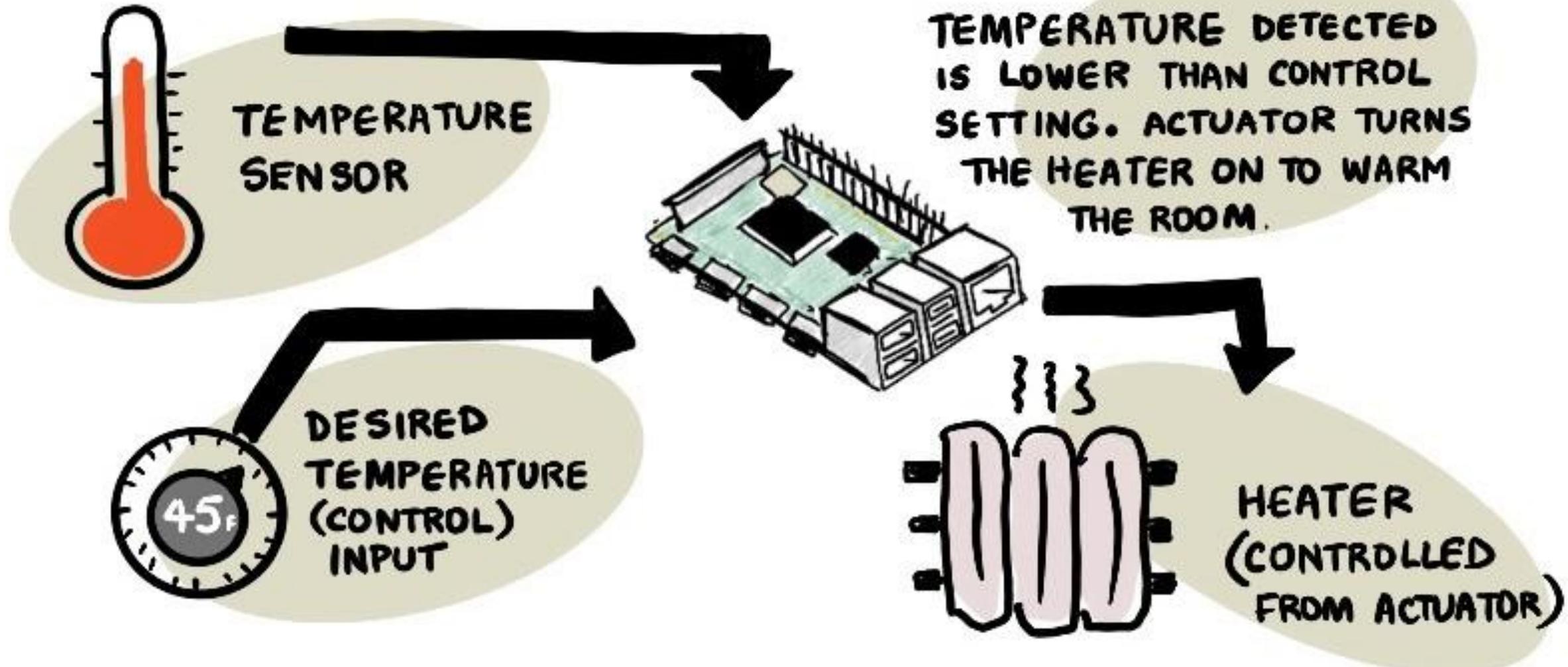
Run for long periods  
Gather data (sensors)  
Take actions (actuators)

## Example

Microcontroller

- RAM in kB
- SPEED in MHz

# EX: A THERMOSTAT!



# QI

---

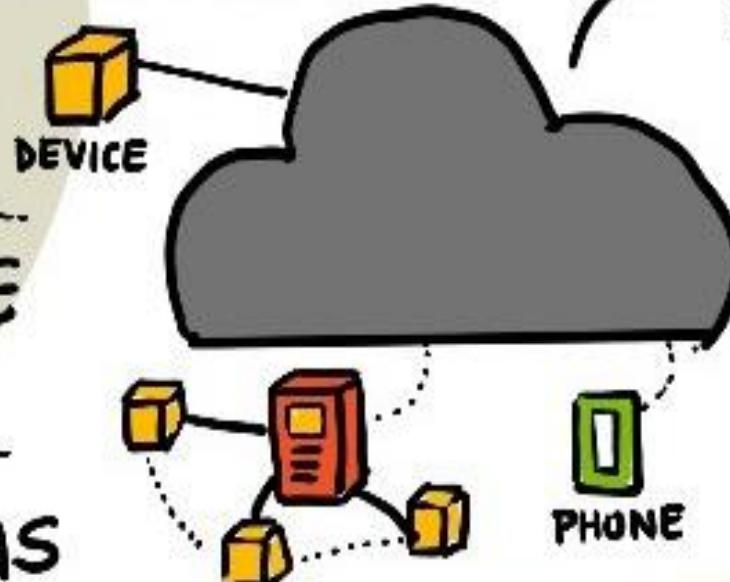
- What other things use sensor data to make decisions?



# Internet

CONSISTS OF APPS THAT

- ❑ PROCESS DATA
- ❑ SEND/RECEIVE MESSAGES
- ❑ TAKE DECISIONS ON REQUESTS TO SEND TO ACTUATORS



TYPICAL SETUP

CLOUD SERVICE  
INTERMEDIARY

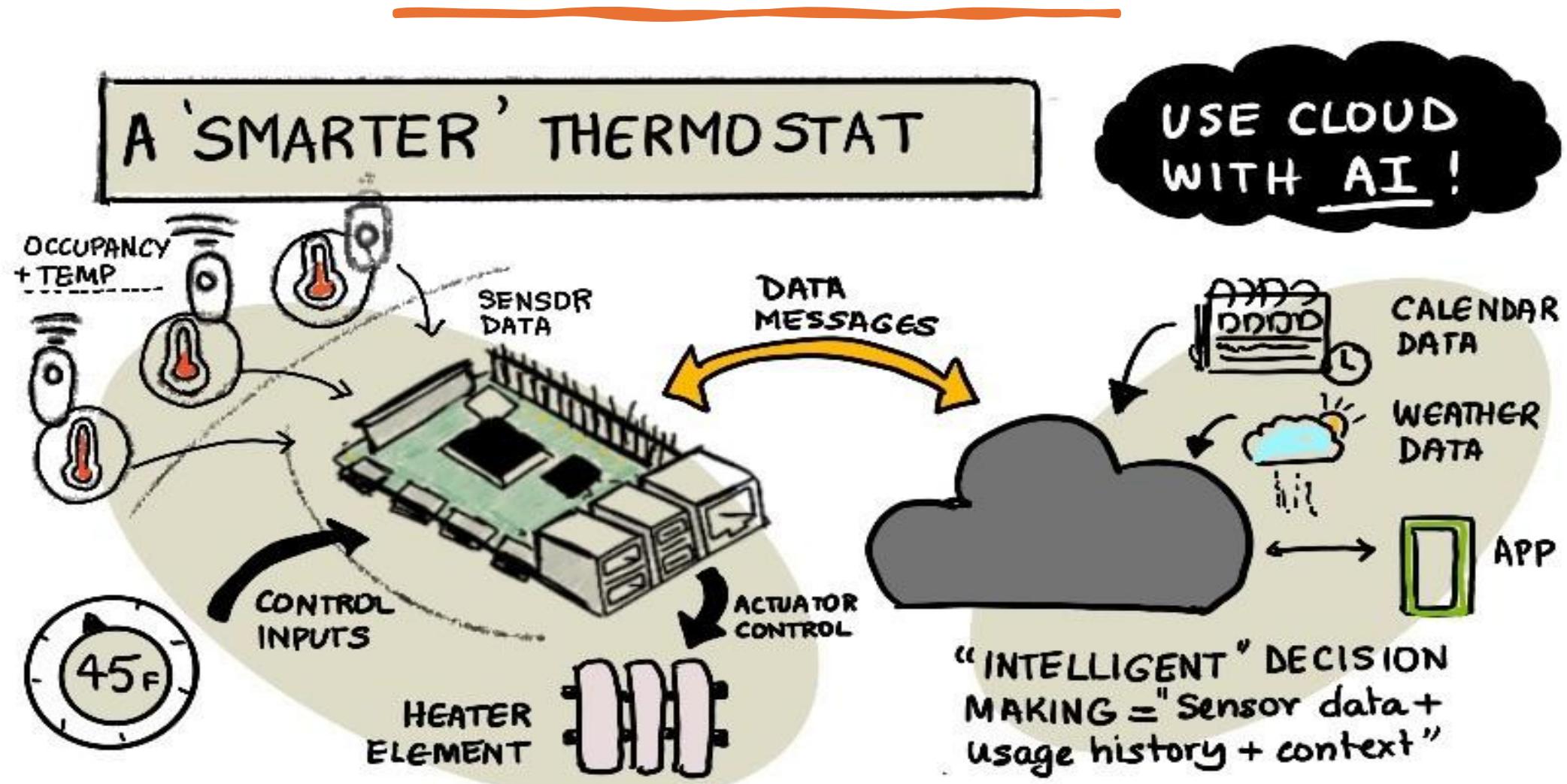
- ✓ HANDLE SECURITY
- ✓ INTERACT WITH SENSORS ON ONE SIDE
- ✓ CONNECT TO APPS ON THE OTHER

MESH  
NETWORKS

PEER TO PEER ROUTES

| DEVICES CONNECT TO EACH OTHER (BT, WiFi) AND TO 'CONNECTED' HUB

# Build a Better Mousetrap



# QI

- What other data could make a smart thermostat even smarter?



# A Device on the Edge

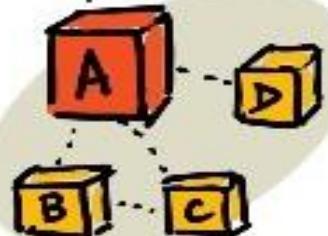


## IOT ON THE EDGE

ALL IOT DEVICES DON'T HAVE  
TO CONNECT TO THE INTERNET

### EDGE

DEVICES ARE 'GATEWAYS'  
CAPABLE OF PROCESSING  
DATA LOCALLY. IOT DEVICES  
CAN CONNECT TO EDGE DEVICES  
OVER LOCAL NETWORKS (WIFI,  
BLUETOOTH)



A = GATEWAY  
(EDGE)

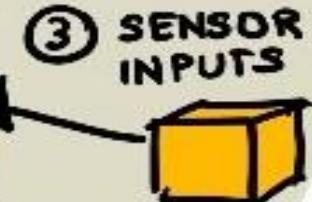
B, C, D = IOT DEVICES  
WITH NO DIRECT INTERNET

① AI MODEL TRAINED  
IN THE CLOUD

EIB



② MODEL DOWNLOADED  
TO EDGE DEVICE



### EXAMPLE

GOOGLE HOME  
AMAZON ALEXA  
APPLE HOMEPOD

# IOT SECURITY



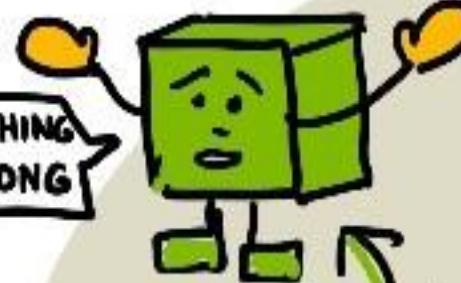
SOMETHING IS WRONG

THERE IS NO  
'S' IN IOT

OH!

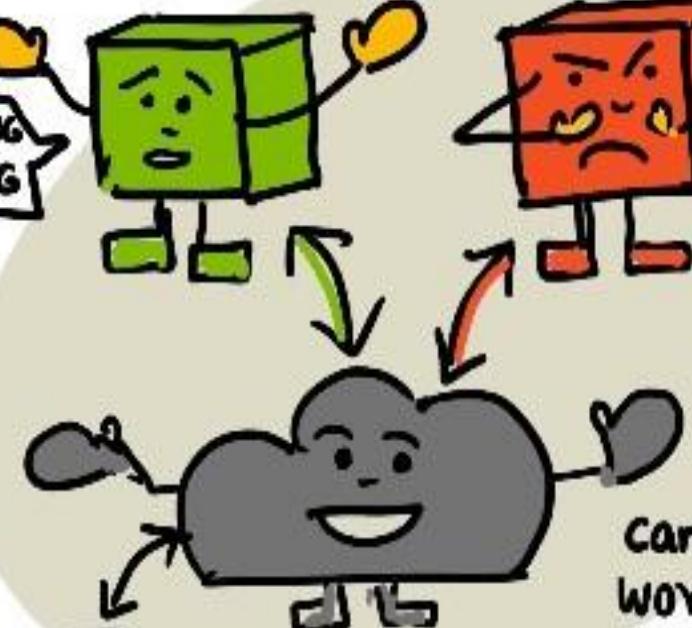


MALICIOUS DATA CAN PROPAGATE



MALICIOUS DEVICE VIRUS ATTACKS

can have real world consequences because IOT devices CONTROL environment



A POPULAR JOKE ON IOT IMPLIES SECURITY DOES NOT EXIST -

IN REALITY

IOT DEVICES CONNECT TO THE CLOUD - AND ARE ONLY AS SECURE AS THE CLOUD (AND NETWORK)

EXAMPLES OF ATTACKS



STUXNET WORM



BABY MONITOR

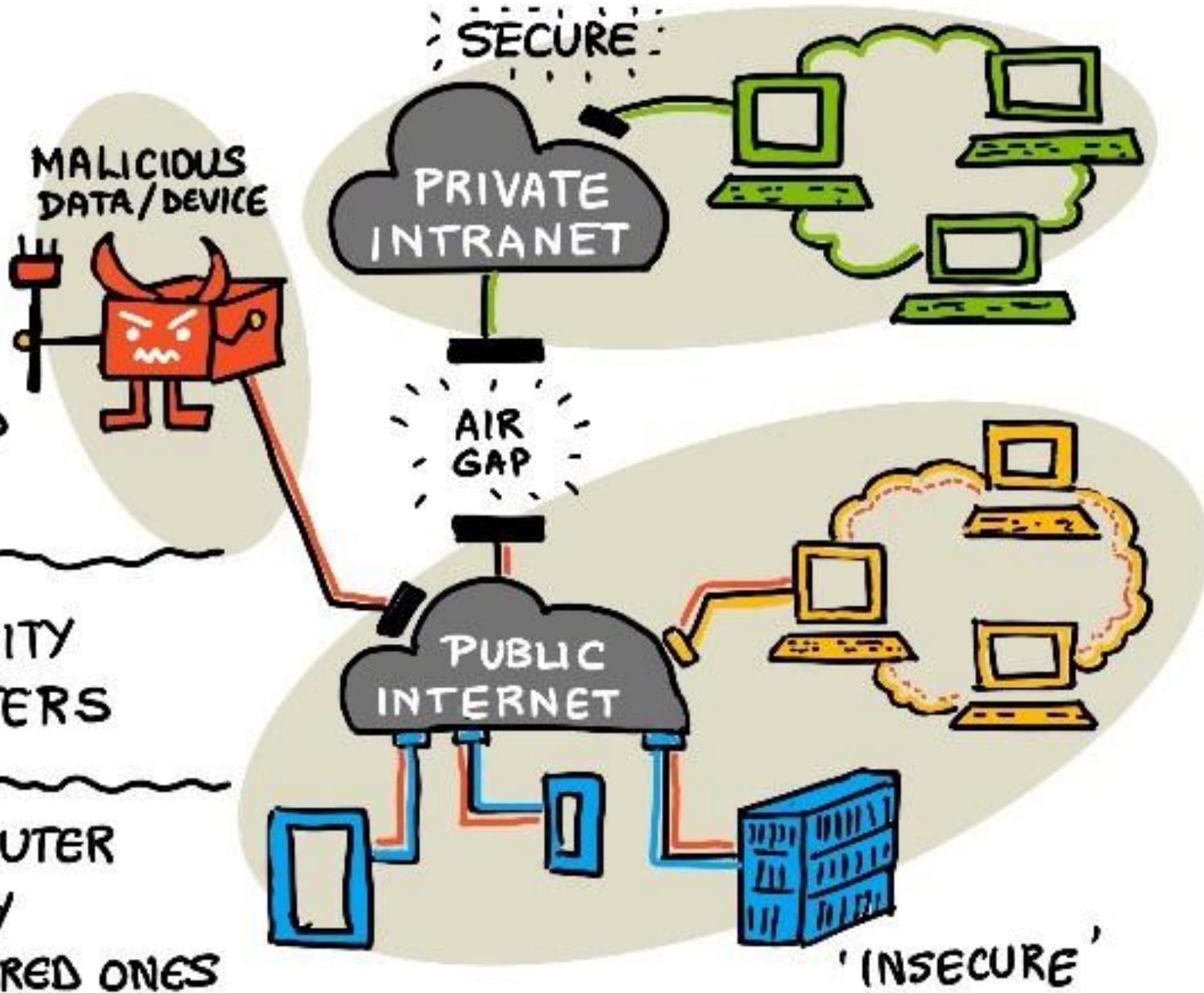
## AIR GAPPING

aka

- ✓ AIR WALL
- ✓ AIR GAP
- ✓ DISCONNECTED NETWORK

IS A NETWORK SECURITY MEASURE FOR COMPUTERS

WHERE A SECURE COMPUTER NETWORK IS PHYSICALLY ISOLATED FROM UNSECURED ONES



# QI

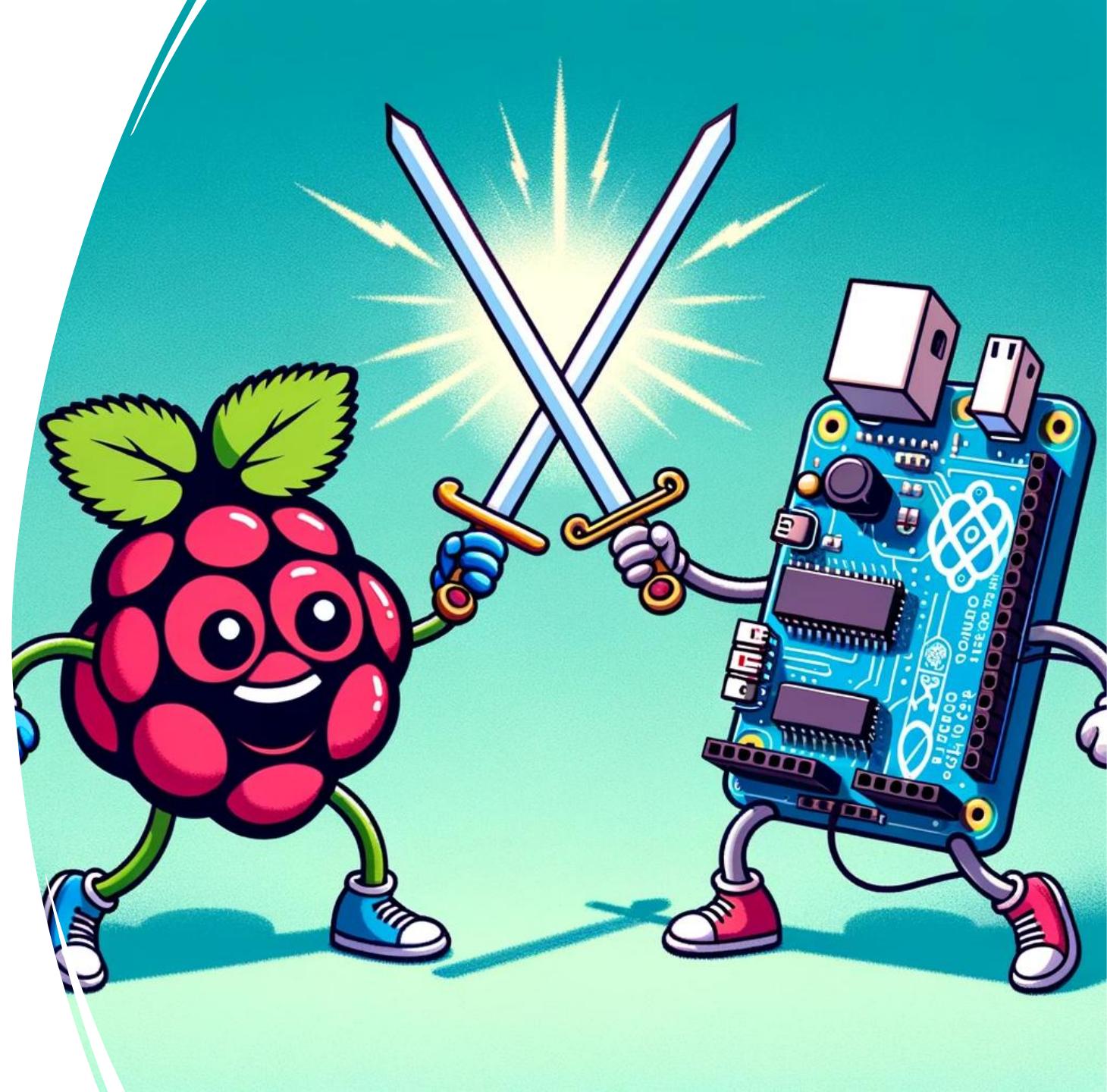
- What other security challenges do you think IoT Systems have?



# Single Board Computers

vs

# Microcontrollers



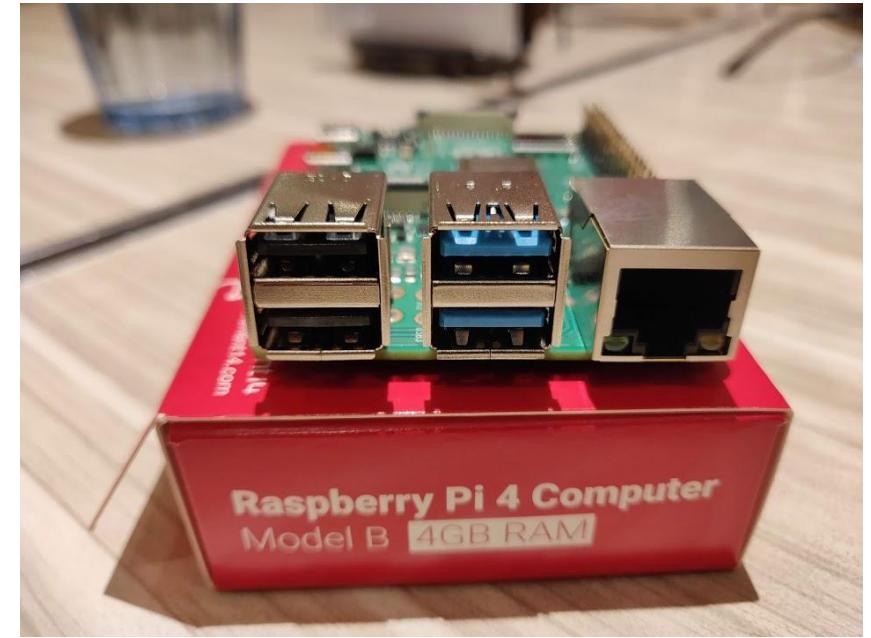
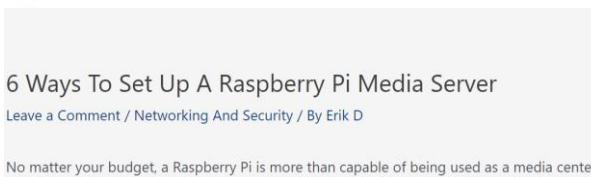
# SBC vs µC

- Single Board Computer (SBC)
  - Example: Raspberry Pi
- Microcontroller (µC)
  - Example: Arduino MKR1010  
(kind of...)
- What are they?
- What are key differences?



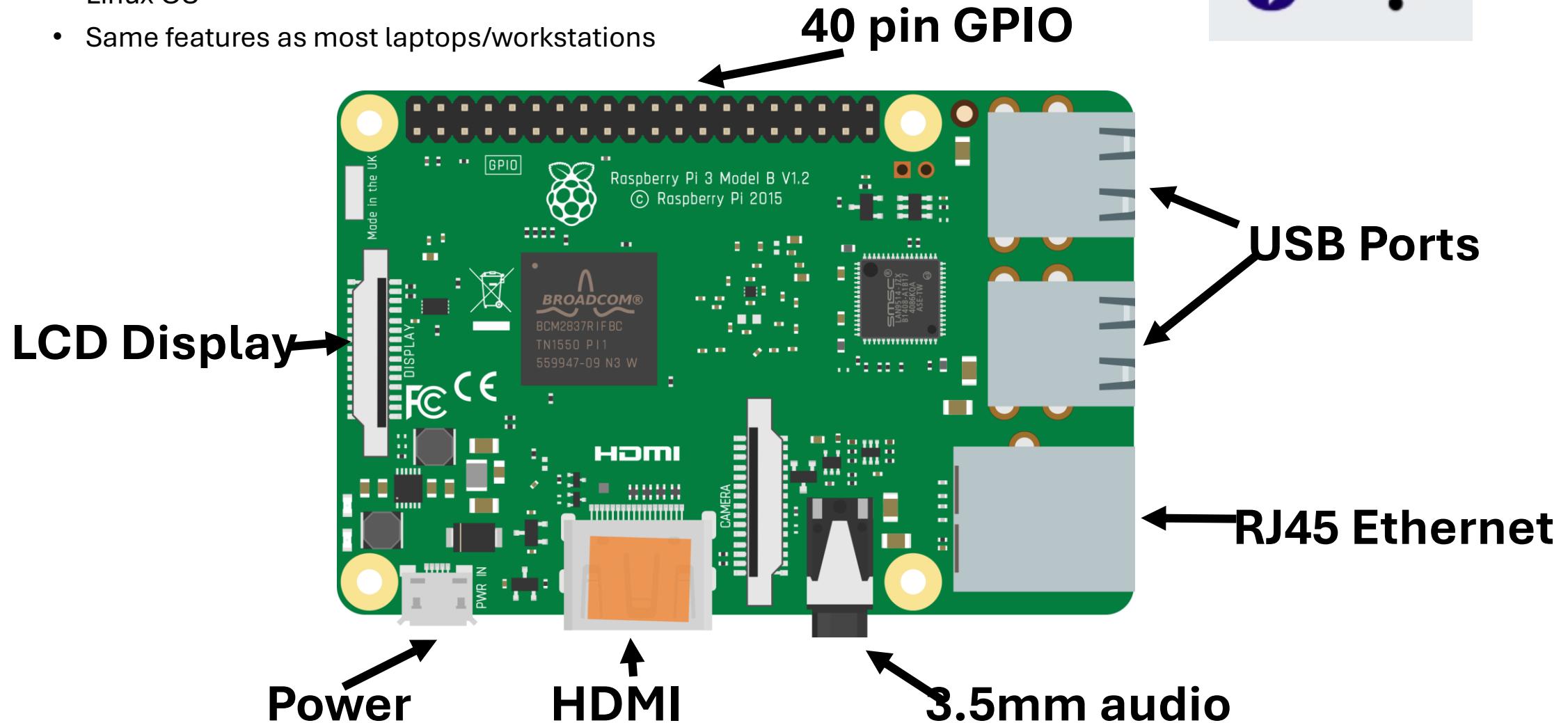
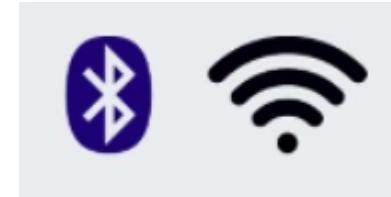
# What is a Single Board Computer(SBC)?

- A complete computer on a single board
  - CPU, RAM, storage, and I/O ports.
- Runs a full-fledged operating system
  - Linux distributions
- Capable of multitasking, web browsing, and running software applications.
- Examples 



# Raspberry Pi

- Low cost, single board computer
- Linux OS
- Same features as most laptops/workstations



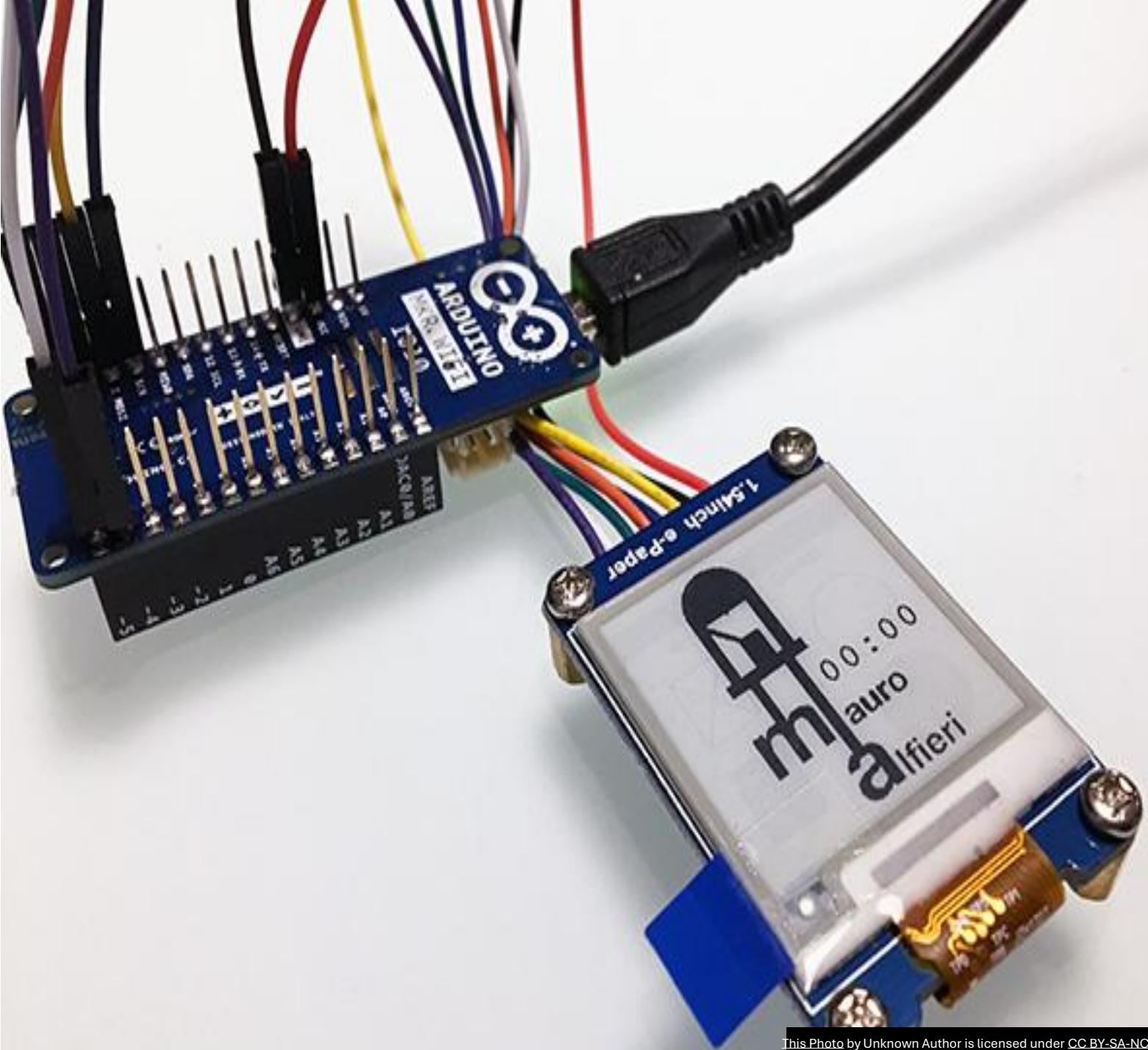
# What is a Microcontroller?

- A compact circuit designed for specific operations in embedded systems.
- Contains a CPU, small RAM, storage, and operates without a full OS.
- Executes pre-programmed tasks, ideal for hardware interactions.
- Example:
  - Automated Plant Watering System: An Arduino reads soil moisture levels and automatically waters a plant when it gets too dry.



# Arduino wifi MKR1010

- Powerful µC platform
- IoT and network connectivity focused
  - Good for networking modules!!!
- Open source
- Large online community



# SBCs vs. Microcontrollers Key Differences

- **Power Consumption:** Arduino is more energy-efficient than Raspberry Pi.
- **Complexity:** Raspberry Pi can handle intricate tasks due to its robust CPU architecture, memory resources and OS.
- **Boot Time:** Arduino starts nearly instantly, while Raspberry Pi requires boot-up time (like your laptop).
- **Cost:** Microcontrollers are generally cheaper than SBCs.
- **Development:** Raspberry Pi offers a typical Operating System-like desktop-like environment, whereas Arduino requires external coding and uploading.



Arduino Nano V3.0  
Nano V3.0 Develop

35 Sold

€3.13

Price includes VAT

Extra 5% off

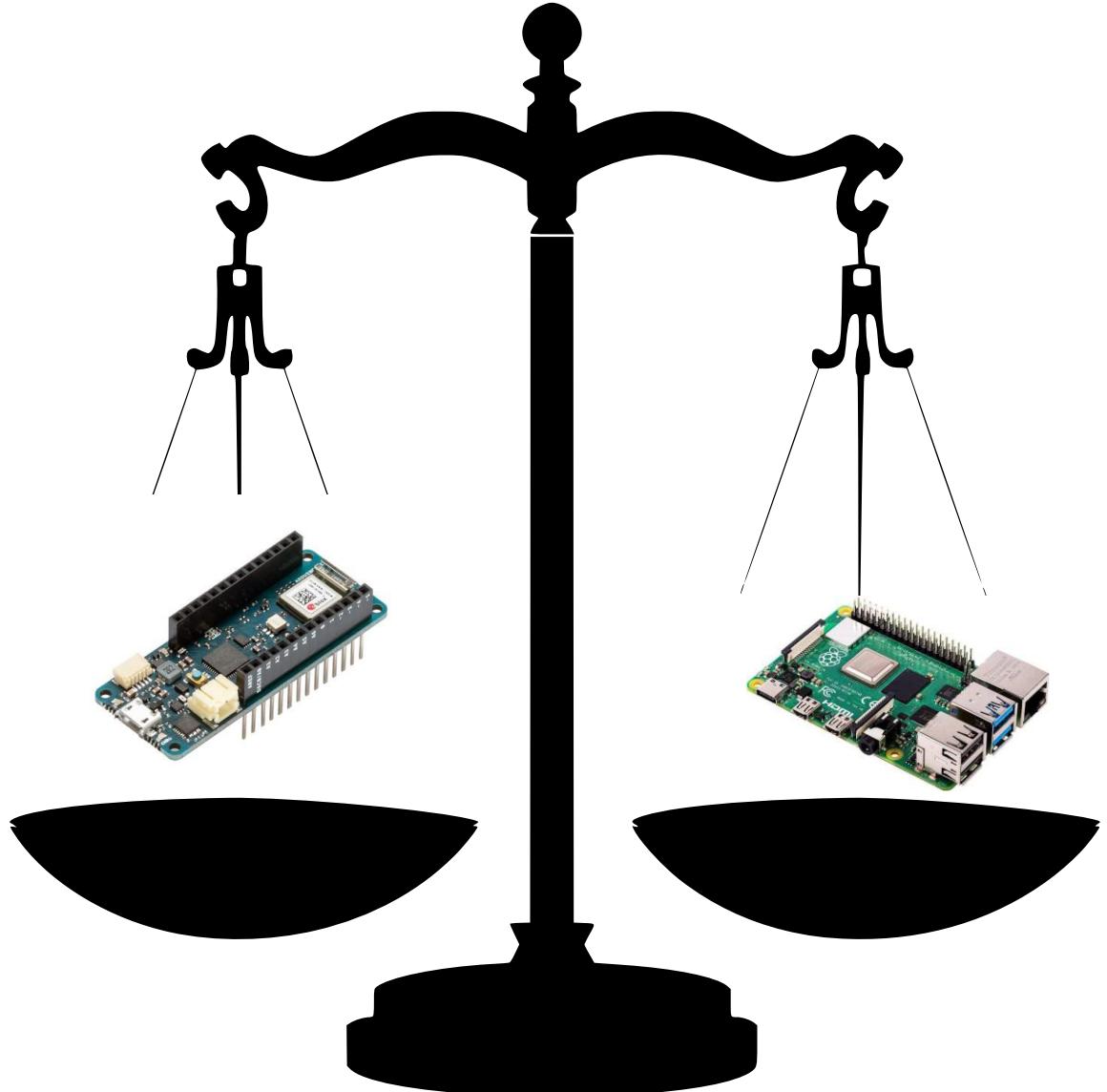
€0.92 Off  
Store Coupon

# SBCs vs. Microcontrollers Key Differences

- **Real-time:**
  - Microcontrollers are generally designed to perform specific tasks without the overhead of an operating system.
  - Microcontrollers loop continuously waiting for input from sensor the react immediately.
  - **Real-time operation:** can guarantee a task is executed in a predictable time frame.
    - Can use “interrupt” handlers to achieve this. Triggered by a sensor input(e.g. tilt sensor in car)
  - SBCs generally run full-fledged operating systems like Linux,
    - can have varying response times due to task scheduling and other processes running in the background.
- **Microcontrollers can operate in Real-time, SBCs do not.**

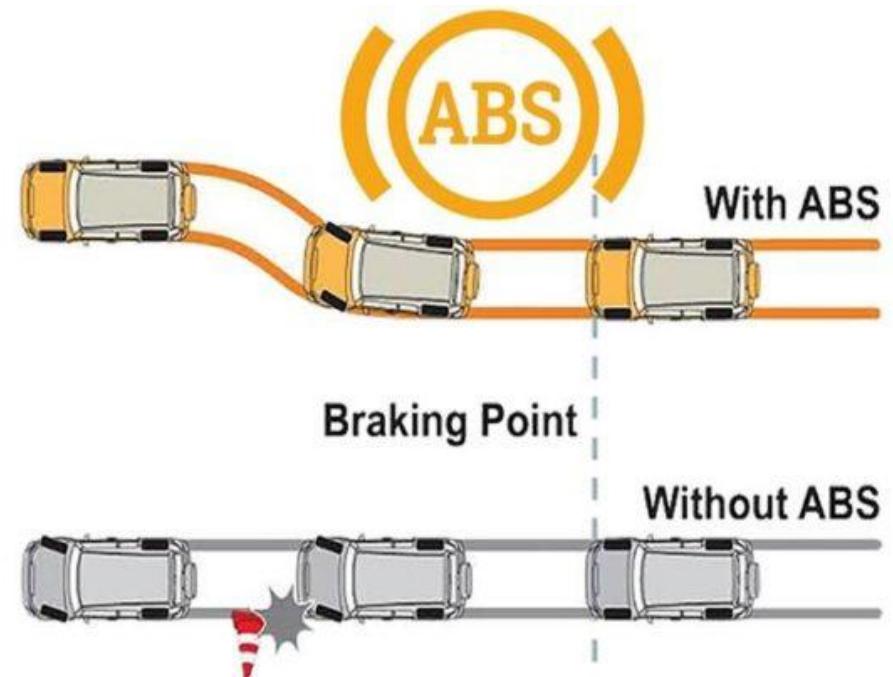
# Which one???

- **Task Simplicity:**
  - Use Arduino for simpler, hardware-focused tasks.
  - Use Raspberry Pi for complex tasks or when a full OS is beneficial (e.g. image processing, DB I/O).
- **Power Constraints:** For battery-operated or energy-efficient systems, Arduino is often a better choice.
- **Integration:** Raspberry Pi is ideal for projects that need internet connectivity, computation & processing, or integration with complex software.
- **Budget:** Consider cost implications, especially for large scale or commercial projects.



# Example Use Case1: ABS

- **Anti-lock Braking System (ABS)**
  - ABS is a safety anti-skid braking system that prevents the wheels from “locking up” during braking, which helps the driver maintain control.
- **How it works:**
  - Wheel Speed Sensors constantly measure the speed of each wheel and send this data to the ABS controller.
  - ABS Controller processes the wheel speed data and determines if a wheel is about to lock up.
  - If the ABS Controller detects a wheel is about to lock, it decreases the pressure to the brake until the wheel starts moving again.
- **Safety Critical System: Has to Work Always, Has to be Reliable, Has to be Fast, Can't be waiting around for processor timeslot**



This Photo by Unknown Author is licensed under CC BY-SA-NC

# Example Use Case1: ABS

- The ABS controller needs to operate in real-time.
- When a driver steps on the brake pedal, the ABS must instantly assess and react to wheel slip conditions to prevent skids
- A delay in processing could reduce the effectiveness of the ABS, leading to potential accidents.
- **Microcontroller** is the best option here and are used extensively in the automotive industry where safety, performance, and reliability are critical.(Incorporated into ECUs(electronic control units))



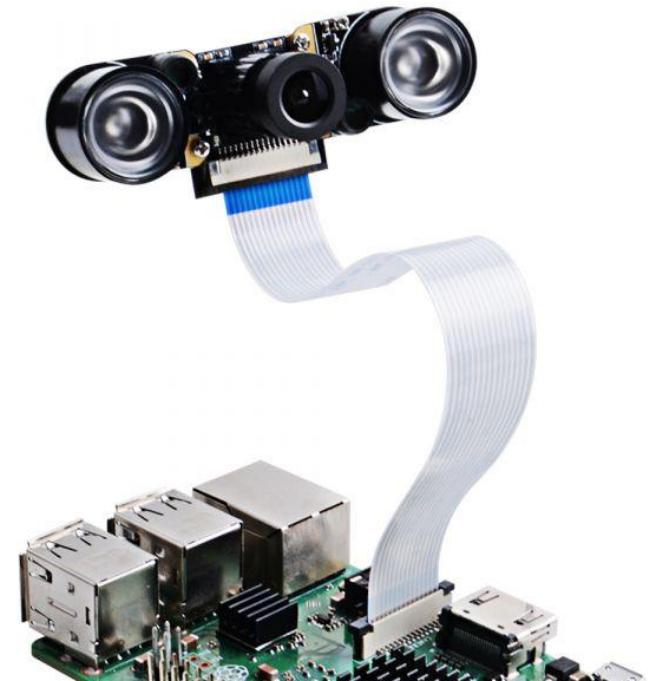
## Example Use Case2: Licence Plate Recognition (LPR)

- LPR systems automate the process of identifying a vehicle's license plate to manage access, billing, security....
- **Capture:** Cameras take pictures of vehicles' as they come and go.
- **Image Processing:** The system processes these images to enhance clarity, adjust lighting, and prepare the image for plate extraction.
- **Plate Extraction:** Program identify the rectangular region of the image containing the license plate.
- **Optical Character Recognition:** The system then processes this extracted portion to recognise and read the characters on the license plate.
- **Database:** The licence plate number is used to query/update a DB



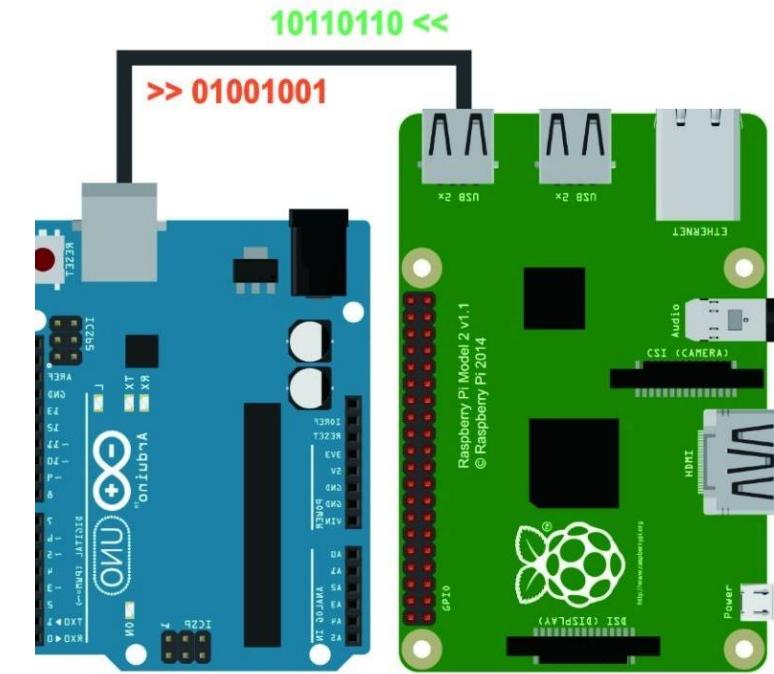
# Example Use Case2: Licence Plate Recognition (LPR)

- A lot of computationally expensive processing here (image processing). May need a lot of CPU power and memory.
- Short period to acquire, process and recognise licence plate is acceptable.
- Quick, but not exact real-time processing, is OK.
- Nobody will be hurt if it fails – not safety critical.
- A networked/connected **Single Board Computer** connected to Camera is a viable option here.

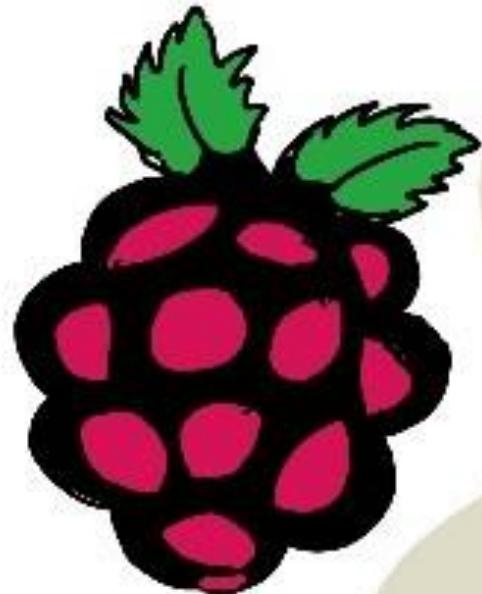


Can you use both together?

- Yep! If you want
  - Why?
    - Microcontrollers are reliable, real-time and interface well with sensors (analog/digital sensors)
    - OS on computer have a lot of processing power/software to work on data but can crash and have security vulnerabilities like any computer.
    - Use Microcontroller to interface/control sensors and actuators
    - Use SBC to process data/connect to other networks and services.



# SINGLE BOARD COMPUTERS



RASPBERRY PI  
FOUNDATION

2009 UK CHARITY

MISSION

PROMOTE THE STUDY OF  
COMPUTER SCIENCE AT  
SCHOOL LEVELS ...



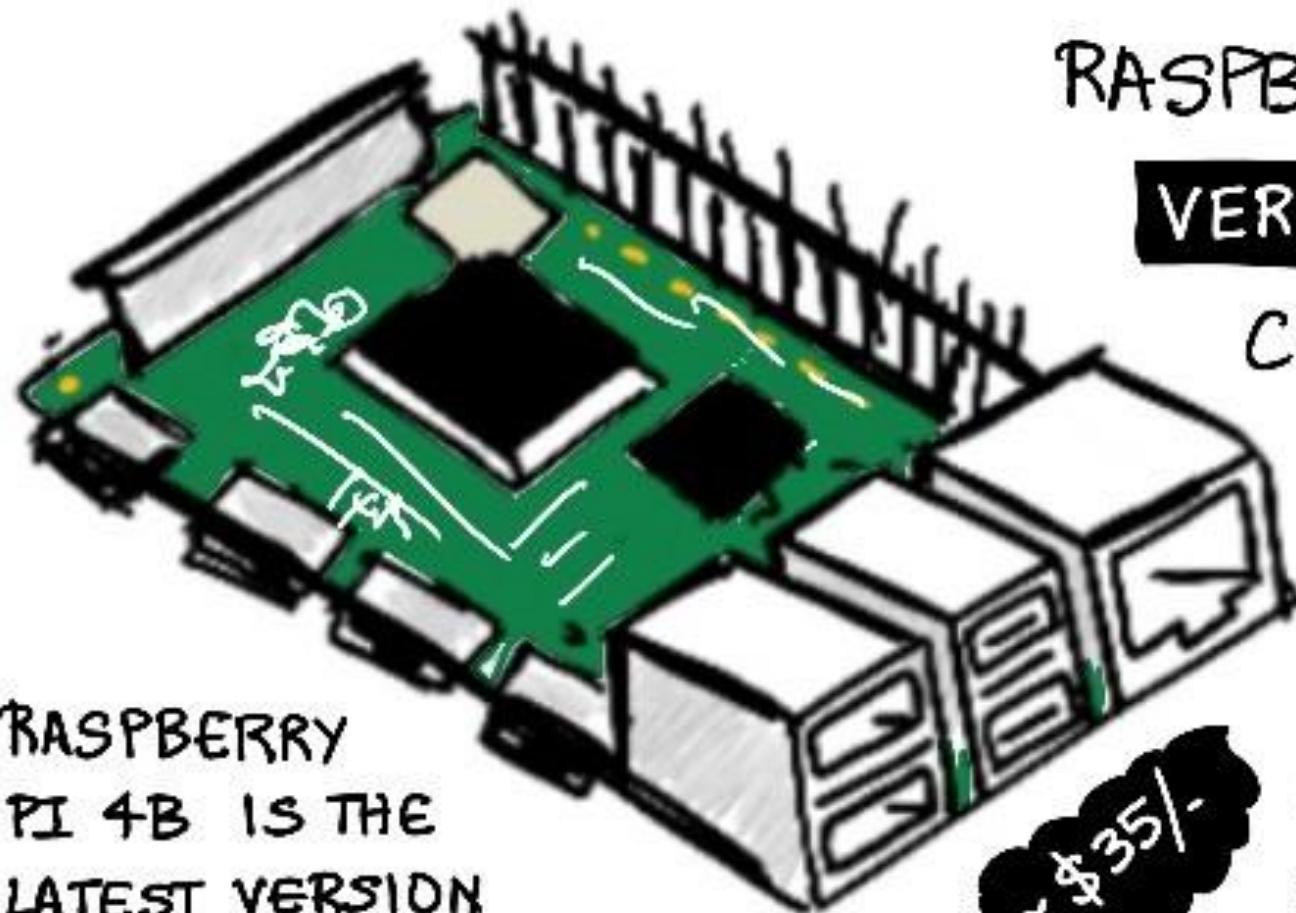
RASPBERRY PI  
SINGLE BOARD  
COMPUTER

**3** VARIANTS

- FULL VERSION
- PI "ZERO"
- COMPUTE MODULE  
THAT CAN BE  
BUILT INTO YOUR  
IOT DEVICE

# RASPBERRY PI 4

COMPARABLE TO DESKTOP  
PC/MAC — BUT CHEAPER

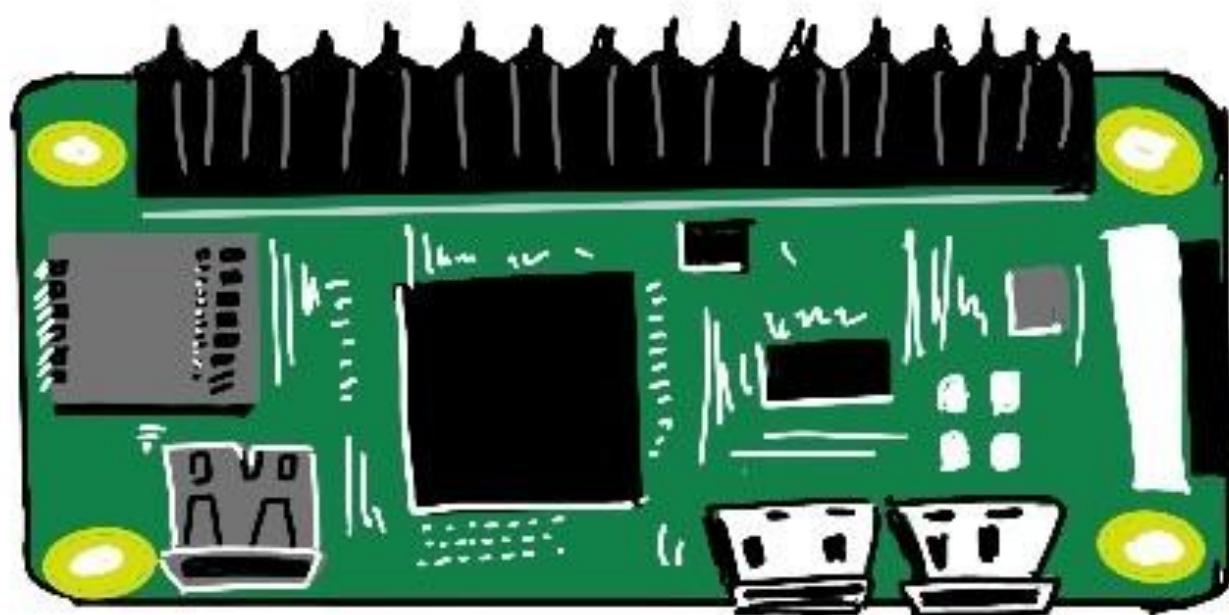


RASPBERRY  
PI 4B IS THE  
LATEST VERSION

RASPBERRY PI 4 IS A **FULL**  
**VERSION** SINGLE BOARD  
COMPUTER WITH **QUAD-CORE**  
**CPU**, **2, 4 or 8** GB of RAM,  
WiFi, Gigabit Ethernet, 2  
HDMI ports, 2 USB 2.0 ports,  
2 USB 3.0 ports, 40 GPIO pins,  
SD card slot, camera connector  
etc.

# RASPBERRY PI ZERO

BY COMPARISON IS **SMALLER** AND HAS



LOWER  
POWER

THAN PI-4B

1 CORE 1GHZ CPU  
512 MB RAM  
1 HDMI PORT  
1 MICRO USB PORT  
40 GPIO PINS  
SD CARD SLOT  
CAMERA CONNECTOR

ALL PI VARIANTS  
RUN RASPBERRY  
PI OS – VERSION OF  
DEBIAN LINUX

LITE  
VERSION  
“HEADLESS”

FULL  
VERSION  
DESKTOP ENV

BOTH PI-ZERO AND PI-4B USE ARM PROCESSORS! = USED IN MOST  
MOBILE PHONES,  
MICROSOFT SURFACE X etc.

# PROGRAMMING: SINGLE BOARD COMPUTERS



WANT TO PROGRAM  
SINGLE BOARD COMPUTERS?

WHAT PROGRAMMING LANG  
DO YOU USE? ARE THEY  
SUPPORTED ON LINUX?

THERE IS A WIDE RANGE  
OF PROGRAMMING LANGUAGES,  
TOOLS AND FRAMEWORKS FOR  
SBC — BECAUSE THEY RUN A  
FULL OPERATING SYSTEM

LARGE  
ECOSYSTEM  
OF HARDWARE  
TO EXTEND PI  
'HATS' = SIT ON  
PI, CONNECT TO 40  
GPIO PINS

Most  
languages  
have libraries  
to access GPIO  
pins and send,  
receive data

MOST COMMON LANGUAGE  
FOR IOT APPS = PYTHON!

## USE OF SINGLE BOARD COMPUTERS

SINGLE BOARD COMPUTERS  
ARE USED FOR BOTH **DEV KITS**  
AND **PROFESSIONAL DEPLOYMENTS**

### USE CASES

- \* CONTROL HARDWARE
- \* RUN COMPLEX  
TASKS (e.g. MACHINE  
LEARNING MODELS)

RASPBERRY PI  
COMPUTE MODULE 4

Designed for  
those building  
custom PCB

COMPUTE MODULE PROVIDES A  
WAY TO MOVE **PROTOTYPE** TO **PRODUCTION**

ALL THE POWER  
OF R.PI 4 **BUT**  
IN A COMPACT AND  
CHEAPER FORM FACTOR

# WHAT'S NEXT?



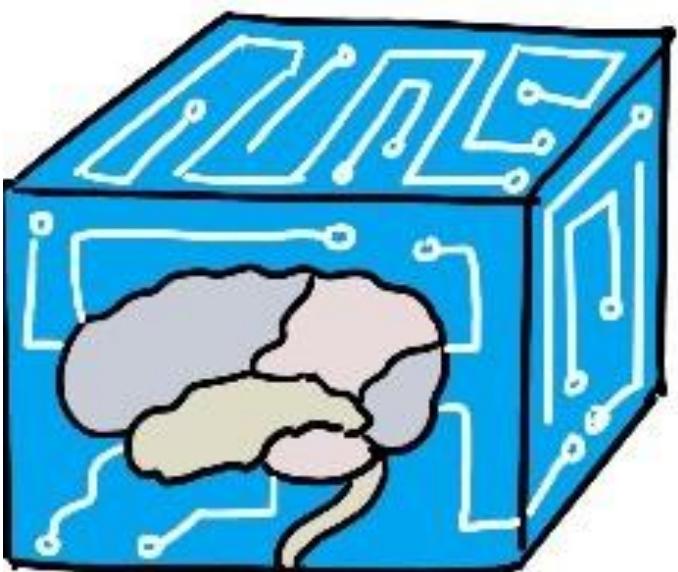
## INTERACT

WITH THE PHYSICAL  
WORLD USING SENSORS  
AND ACTUATORS

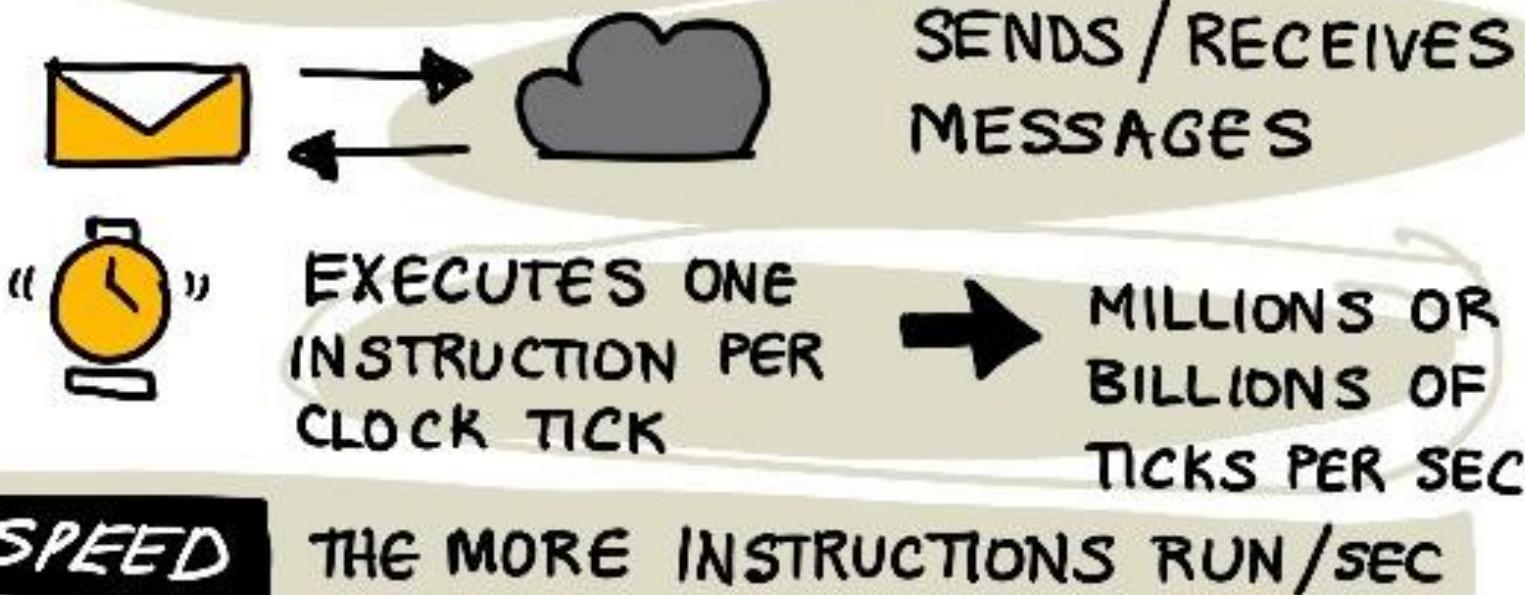
PROJECT  
TIME!

- GATHER DATA
- SEND FEEDBACK
- BUILD NIGHTLIGHT

# MICROCONTROLLERS : CPU

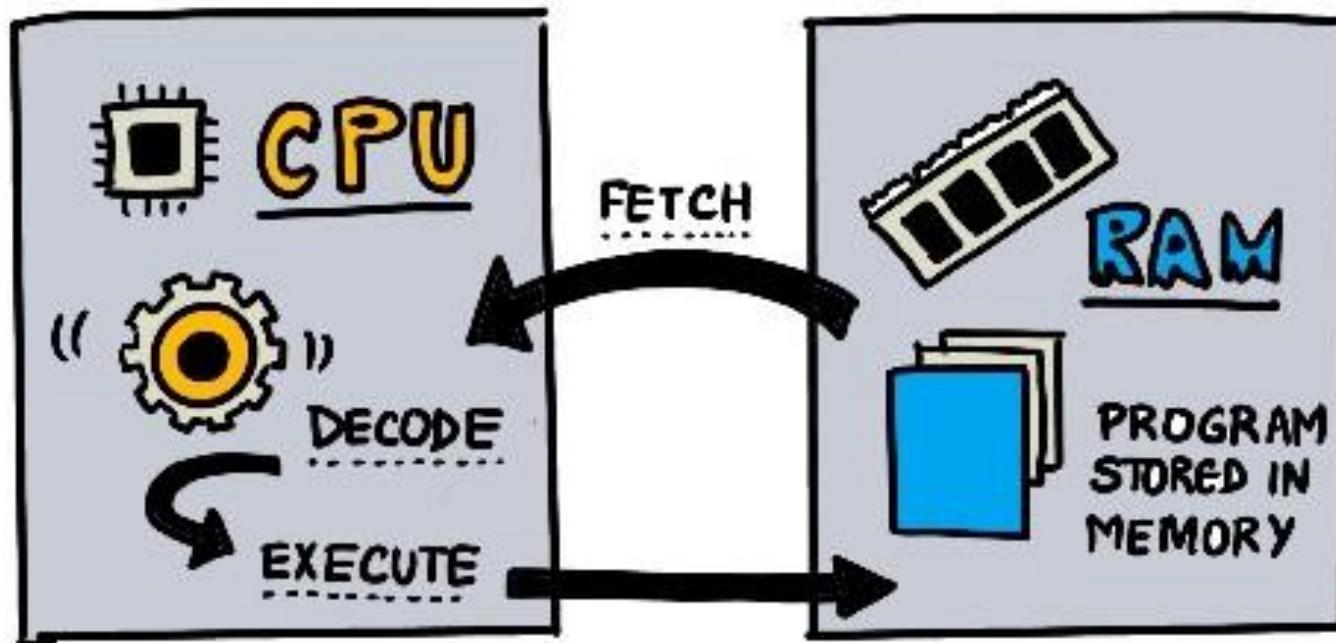


THE CPU (CENTRAL PROCESSING UNIT)  
IS THE **BRAIN** OF THE MICROCONTROLLER.



## MICROCONTROLLERS : CPU

FETCH - DECODE - EXECUTE



ON CLOCK TICK  
-----  
CPU FETCHES  
INSTRUCTION,  
-----  
DECODES IT  
-----

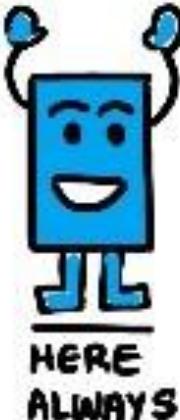
THEN EXECUTES  
THE TASK  
-----

SOME INSTRUCTIONS  
WILL TAKE MULTIPLE  
CLOCK TICKS ...

# MICROCONTROLLERS : MEMORY

THERE ARE 2 TYPES OF MEMORY

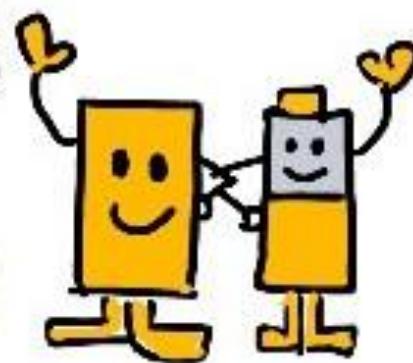
PROGRAM  
— MEMORY —



- STORES YOUR CODE (PROGRAM)
- PERSISTS WHEN THERE IS NO POWER

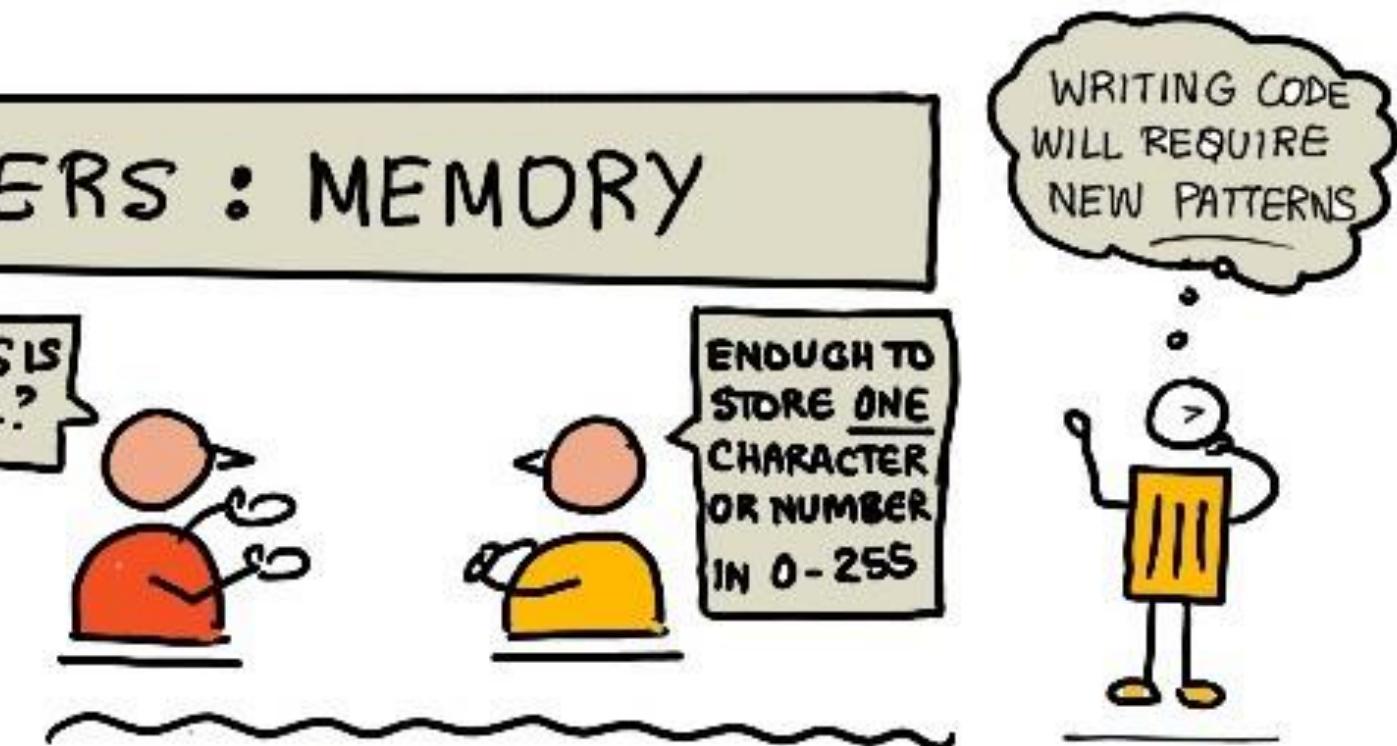
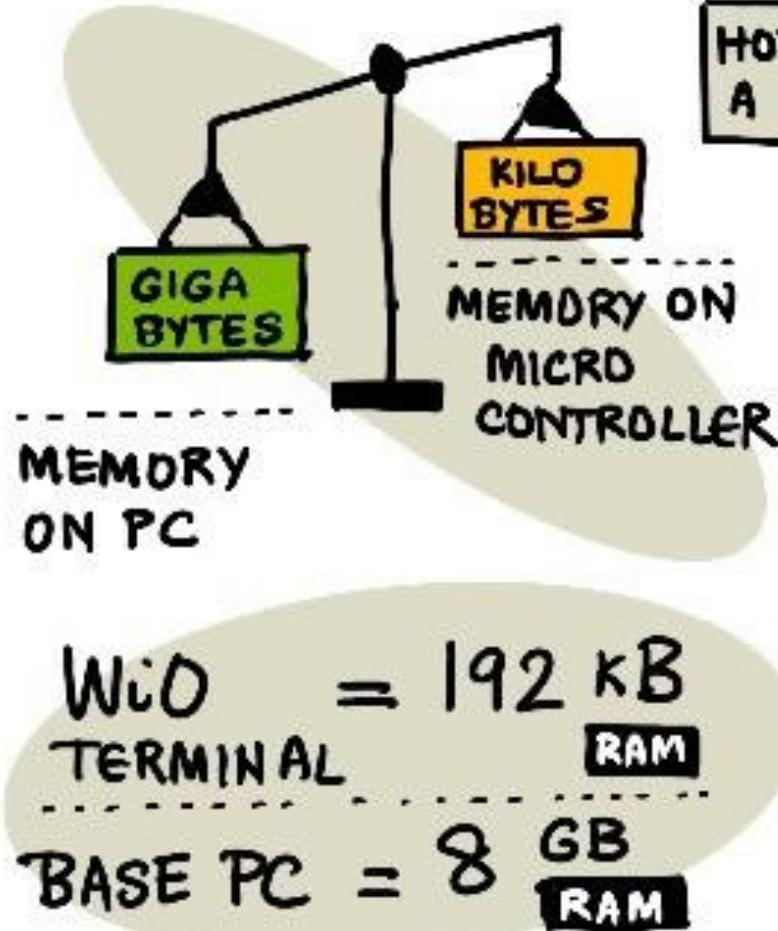
RANDOM ACCESS  
— MEMORY —

- USED TO RUN YOUR CODE WHEN POWERED
- RESETS WHEN THERE IS NO POWER



I NEED POWER  
TO BE ACTIVE

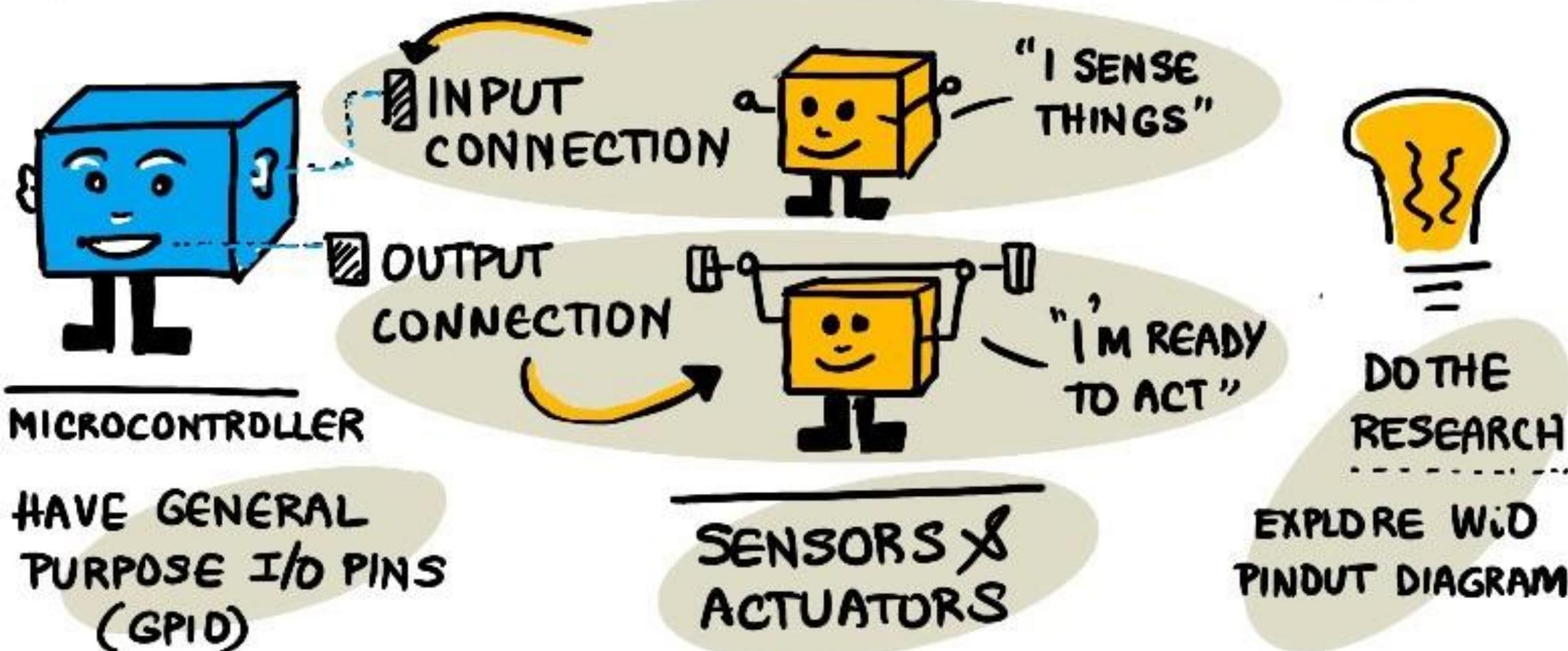
# MICROCONTROLLERS : MEMORY



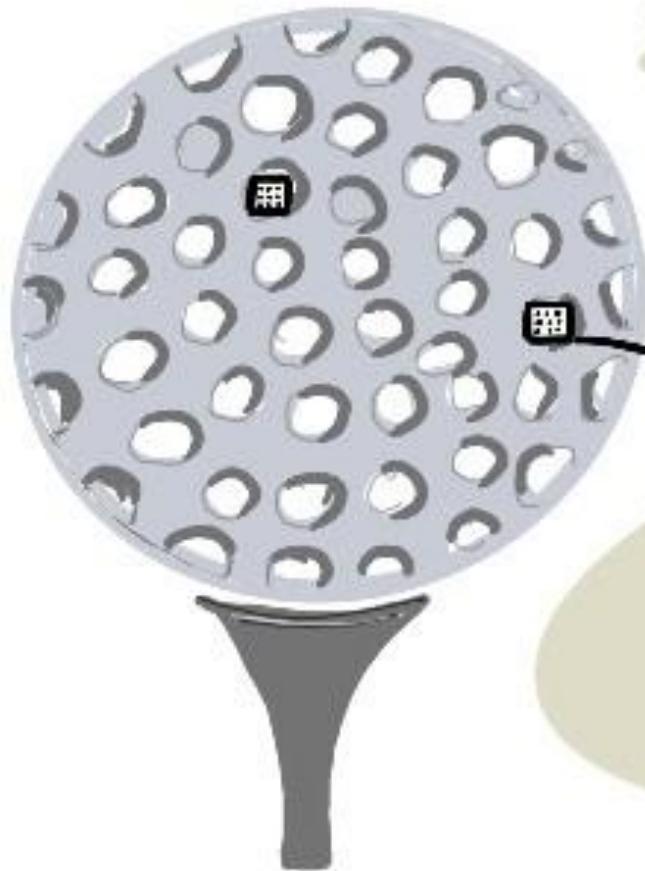
PROGRAM STORAGE is also smaller compared to PC.

WID = 4 MB STORAGE      PC = 500 GB STORAGE

# MICROCONTROLLERS : INPUT/OUTPUT



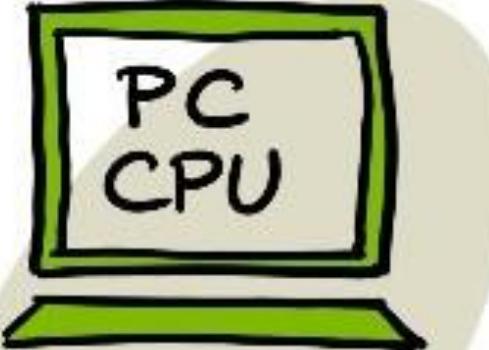
## MICROCONTROLLERS: PHYSICAL SIZE



MICROCONTROLLERS ARE  
SMALL  
IN PHYSICAL SIZE

FREESCALE  
KINETIS KL03  
MCU SMALL  
ENOUGH TO FIT  
IN DIMPLE OF  
GOLF BALLS

1.6mm x  
2mm x  
1mm



136 mm x  
145 mm x  
103 mm



72 mm x  
57 mm x  
12 mm

# FRAMEWORKS & OPERATING SYSTEMS

SEE  
ARDUINO  
FOR  
EXAMPLE

MICROCONTROLLERS  
DON'T RUN A TRADITIONAL  
OPERATING SYSTEM..

- \* THEY HAVE LOW SPEED, MEMORY
- \* THEY PERFORM FOCUSED TASKS

HOW DO I PROGRAM THESE?

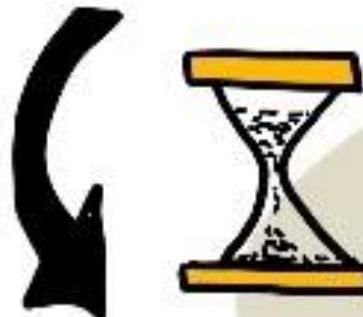


## USE FRAMEWORKS

- \* USE TOOLS TO BUILD CODE IN A WAY THAT WILL RUN ON TARGET MICROCONTROLLER
- \* USE APIs TO TALK TO PERIPHERALS
- \* MANUFACTURERS SUPPORT STANDAR 'FRAMEWORKS' = RECIPES THAT DEVS USE TO RUN CODE ACROSS DIFFERENT MICROCONTROLLER PLATFORMS.

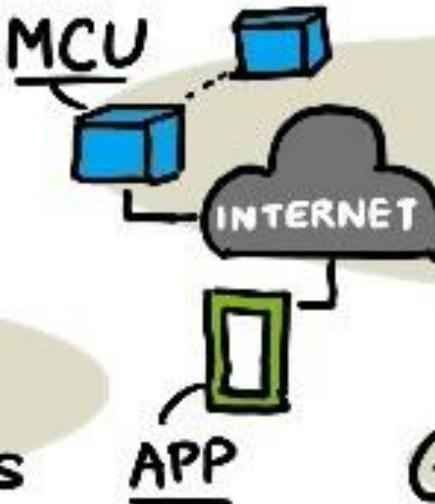
# REAL TIME OPERATING SYSTEMS

USE A REAL TIME  
OPERATING SYSTEM



DESIGNED  
TO HANDLE  
REAL-TIME  
SEND/RECEIVE  
MESSAGE TASKS

RTOS → LIGHTWEIGHT  
CORE FEATURES



MULTITHREADED

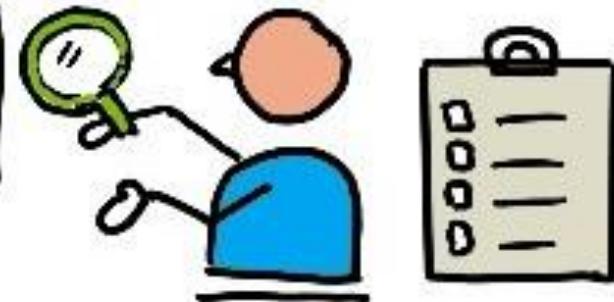
RUN MULTIPLE BLOCKS OF  
CODE IN PARALLEL, ON A  
SINGLE OR MULTIPLE CORES.

NETWORKING

COMMUNICATE SECURELY  
OVER THE INTERNET

GUI COMPONENTS FOR SCREENS

# INVESTIGATE: AZURE RTOS

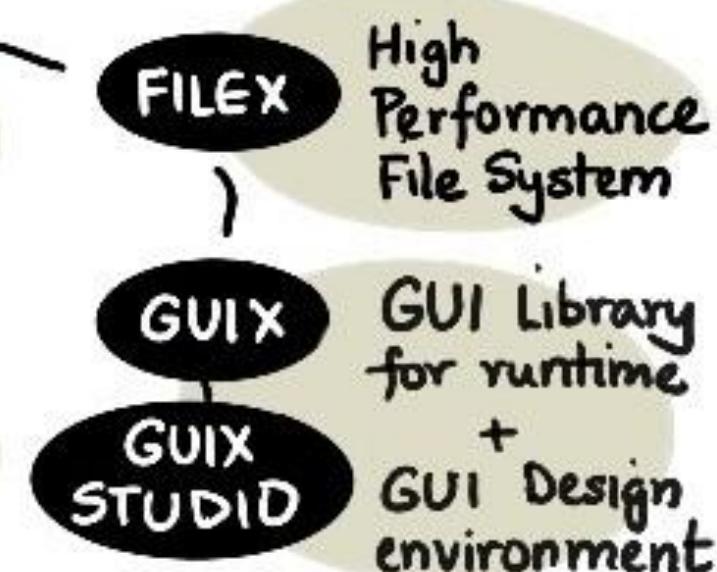
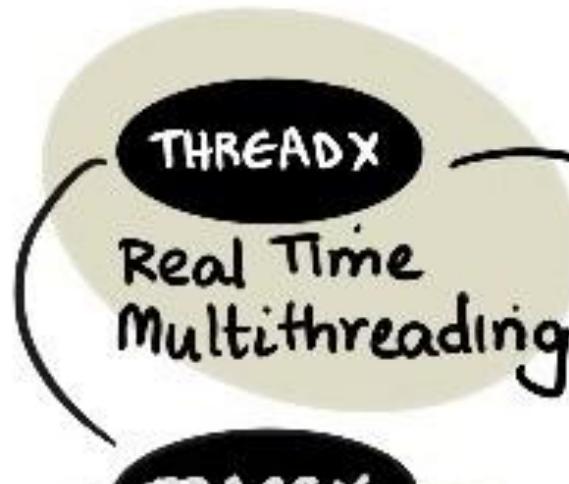


WHAT IS  
AZURE RTOS?

AN EMBEDDED  
DEVELOPMENT  
SUITE WITH

- POWERFUL RTOS
- RELIABLE,  
PERFORMANT
- SUPPORTS POPULAR  
32bit MICROCONTROLLERS

ONE  
EXAMPLE  
SUITE



USBX  
USB host  
and device  
interface

## INVESTIGATE : FREE RTOS, ZEPHYR

[freertos.org](http://freertos.org)

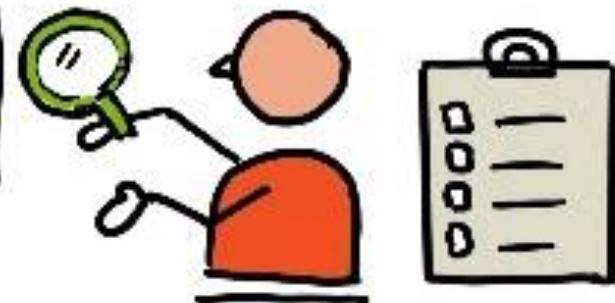
### FREE RTOS

- TRUSTED KERNEL  
MIT LICENSED
- BROAD ECOSYSTEM  
SUPPORT
- KERNEL + IOT DEV  
LIBRARIES

[zephyrproject.org](http://zephyrproject.org)

### ZEPHYR

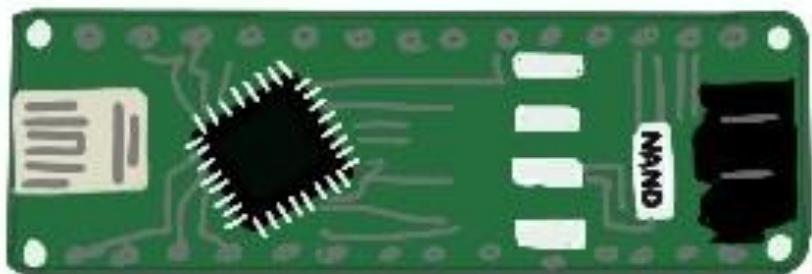
- 200+ BOARDS  
SUPPORTED
- RTOS FOR SECURE,  
SAFE IOT APPS
- OPEN SOURCE WITH  
MULTI-PROTOCOL SUPPORT



### DO YOUR RESEARCH

Explore and  
compare RTOS  
options for IOT

# ARDUINO MICROCONTROLLER FRAMEWORK



ARDUINO IS AN OPEN  
SOURCE ELECTRONICS  
PLATFORM COMBINING  
HARDWARE & SOFTWARE

BUY BOARDS  
FROM ARDUINO  
OR FROM OTHER  
MANUFACTURERS

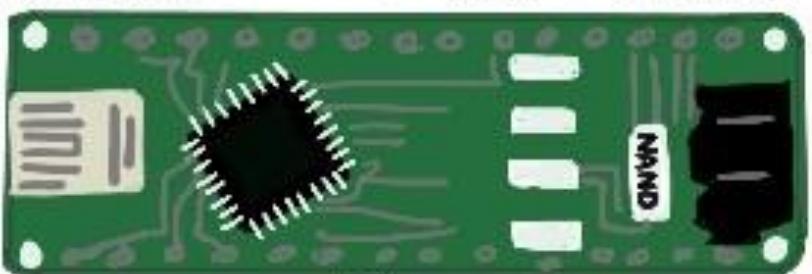
CODE USING  
THE ARDUINO  
FRAMEWORK

CODE USING  
C/C++

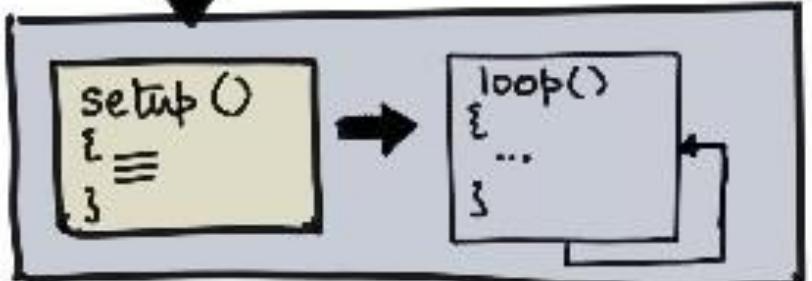
- COMPILED CODE  
IS SMALL IN SIZE
- RUNS FAST EVEN ON  
RESOURCE-LIMITED  
DEVICE PLATFORMS

# ARDUINO : CORE SETUP

ARDUINO COMPLIANT BOARD



ARDUINO FRAMEWORK



## 2 CORE FUNCTIONS

setup()  
loop()

WHEN BOARD POWERS UP

- RUNS `setup()` ONCE
- THEN RUNS `loop()` CONTINUOUSLY (till power off)

# ARCHITECTURE : EVENT LOOP

SETUP

IS FOR ONE-TIME  
INITIALIZATION CODE

→ connect to Wifi, Cloud services etc.

LOOP

IS FOR PROCESSING  
CODE - ADD DELAY TO  
SAVE POWER (sleep/wake cycle)

→ Sensor read  
Send / receive messages

PROGRAM  
ARCHITECTURE

CALLED 'EVENT LOOP'  
OR 'MESSAGE LOOP'

loop() LISTENS FOR

- MESSAGES FROM UI  
(button clicks, keyboard...)
- MESSAGES FROM NETWORK  
(actuator requests)

# ARDUINO : STANDARD LIBRARIES

ARDUINO PROVIDES STANDARD LIBRARIES FOR INTERACTING WITH I/O PINS AND MICRO-CONTROLLERS

EXPOSES CONSISTENT API ACROSS DIVERSE MCU-SPECIFIC IMPLEMENTATION

`delay()`

PAUSE PROGRAM FOR GIVEN PERIOD OF TIME

`digitalRead()`

READ VALUE ON I/O PIN (HIGH OR LOW)

Code can be recompiled for new compliant hardware with minimal effort!

