

# IoT Standard and Protocols

More Introduction Stuff

# Agenda

---

## Components of an IoT Application

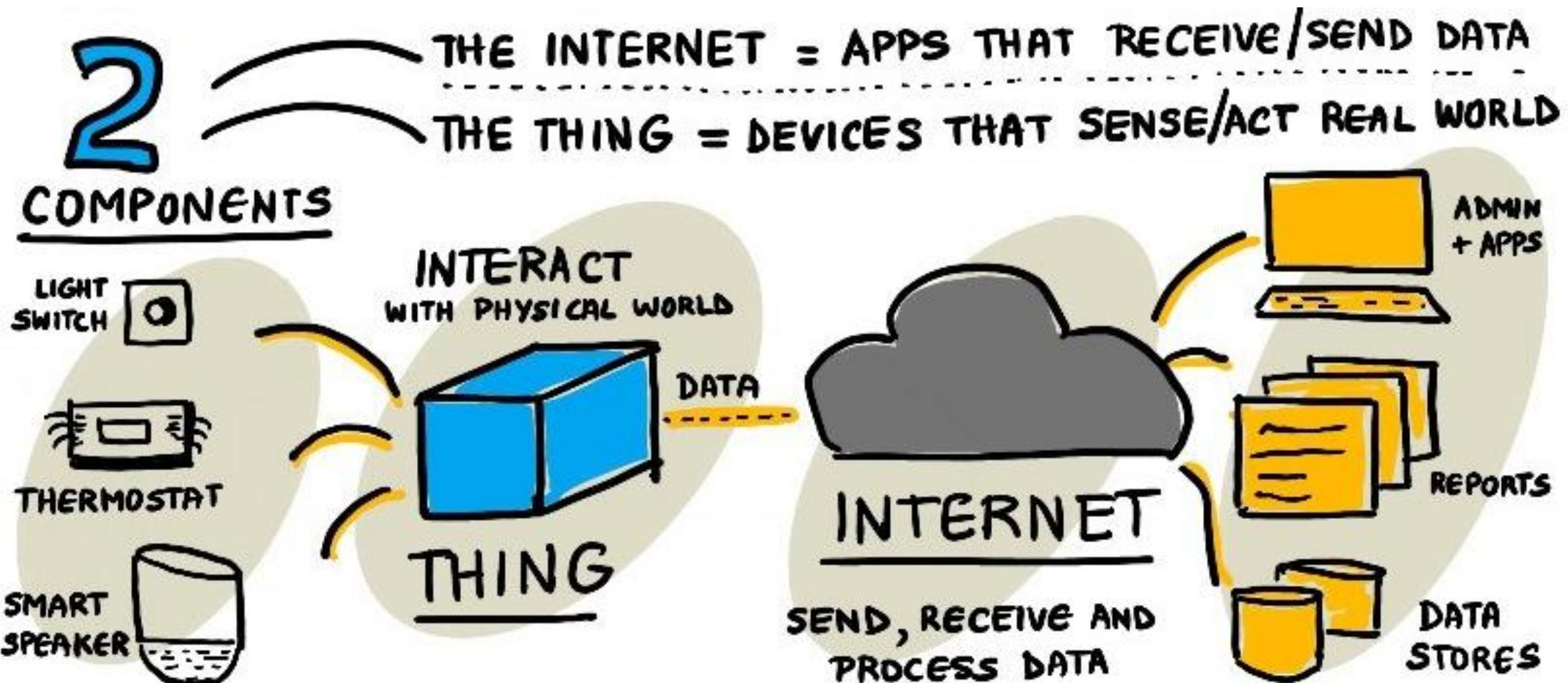
---

### Single Board Computers

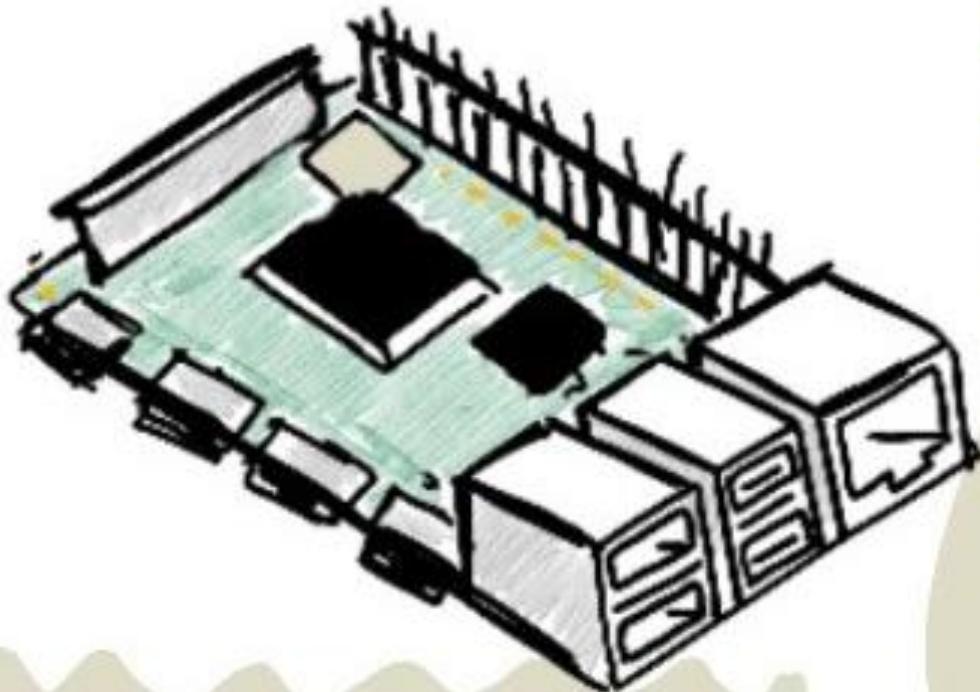
---

### Microcontrollers

# IoT Components



# The “Thing”



A DEVICE THAT CAN  
INTERACT WITH PHYSICAL WORLD

- ❑ LOW POWER
  - ❑ LOW COST
  - ❑ LOW SPEED
- COMPUTERS

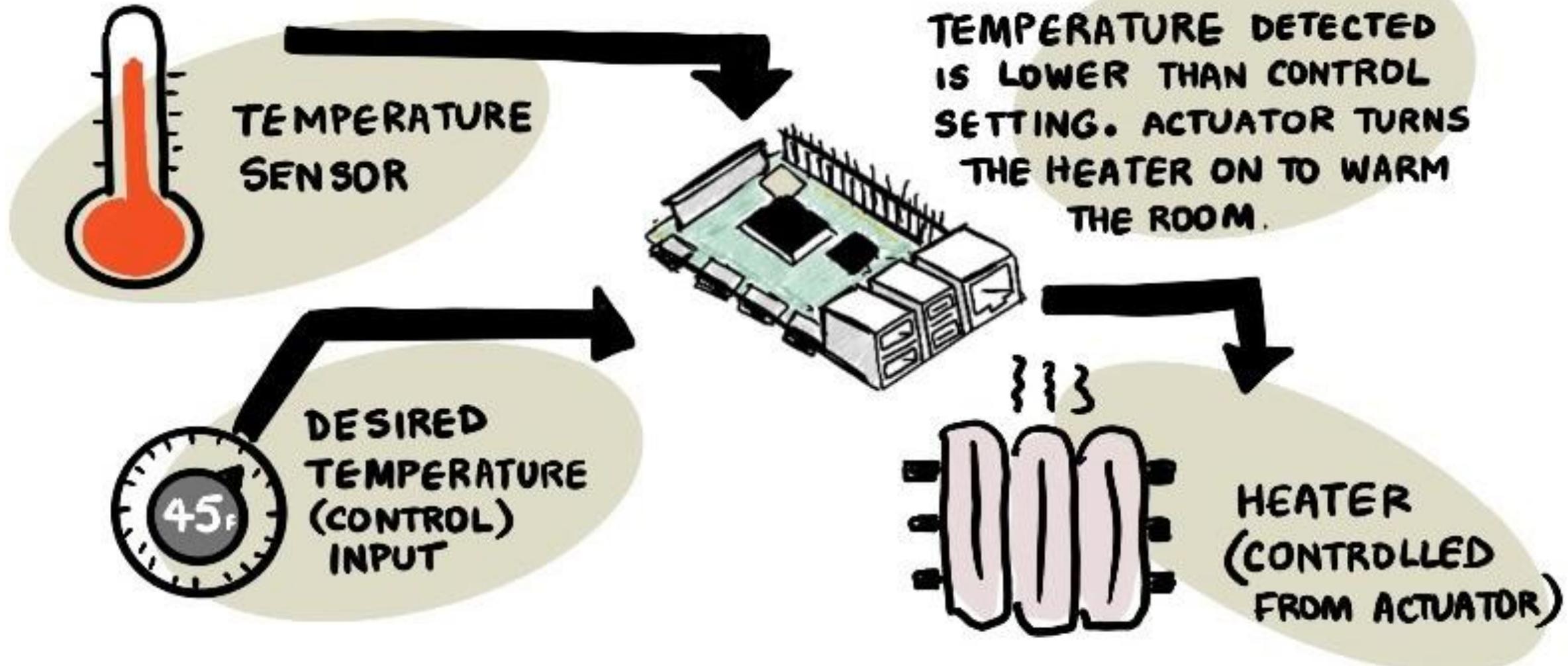
Run for long periods  
Gather data (sensors)  
Take actions (actuators)

## Example

Microcontroller

- ❑ RAM in kB
- ❑ SPEED in MHz

# EX: A THERMOSTAT!



# QI

---

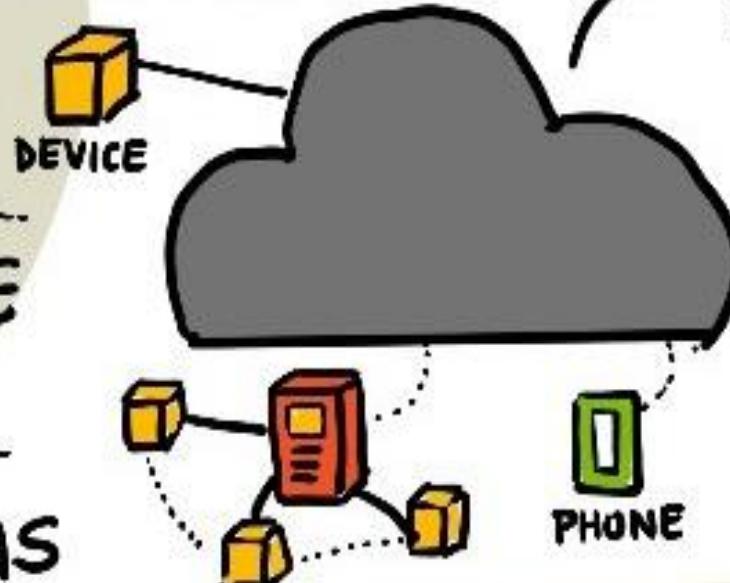
- What other things use sensor data to make decisions?



# Internet

CONSISTS OF APPS THAT

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



TYPICAL SETUP

CLOUD SERVICE  
INTERMEDIARY

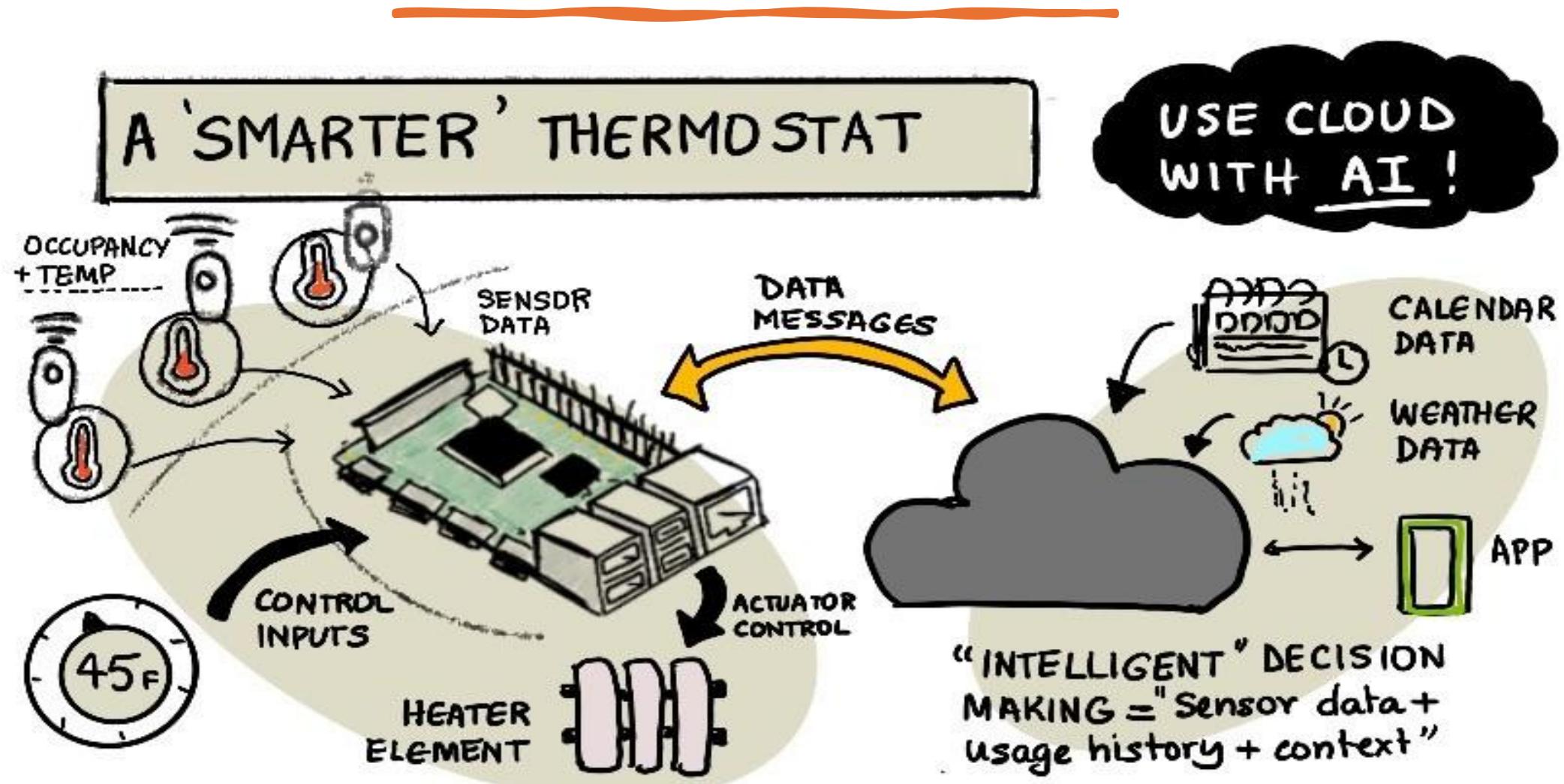
- ✓ HANDLE SECURITY
- ✓ INTERACT WITH SENSORS ON ONE SIDE
- ✓ CONNECT TO APPS ON THE OTHER

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

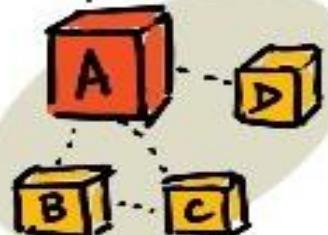


## IOT ON THE EDGE

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)

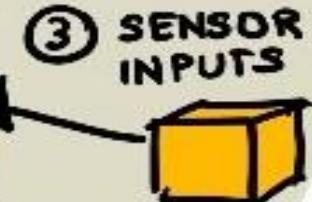
B, C, D = IOT DEVICES  
WITH NO DIRECT INTERNET

① AI MODEL TRAINED  
IN THE CLOUD

EIB



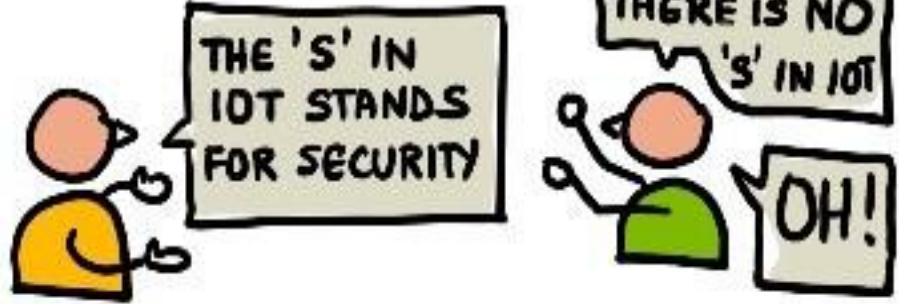
② MODEL DOWNLOADED  
TO EDGE DEVICE



### EXAMPLE

GOOGLE HOME  
AMAZON ALEXA  
APPLE HOMEPOD

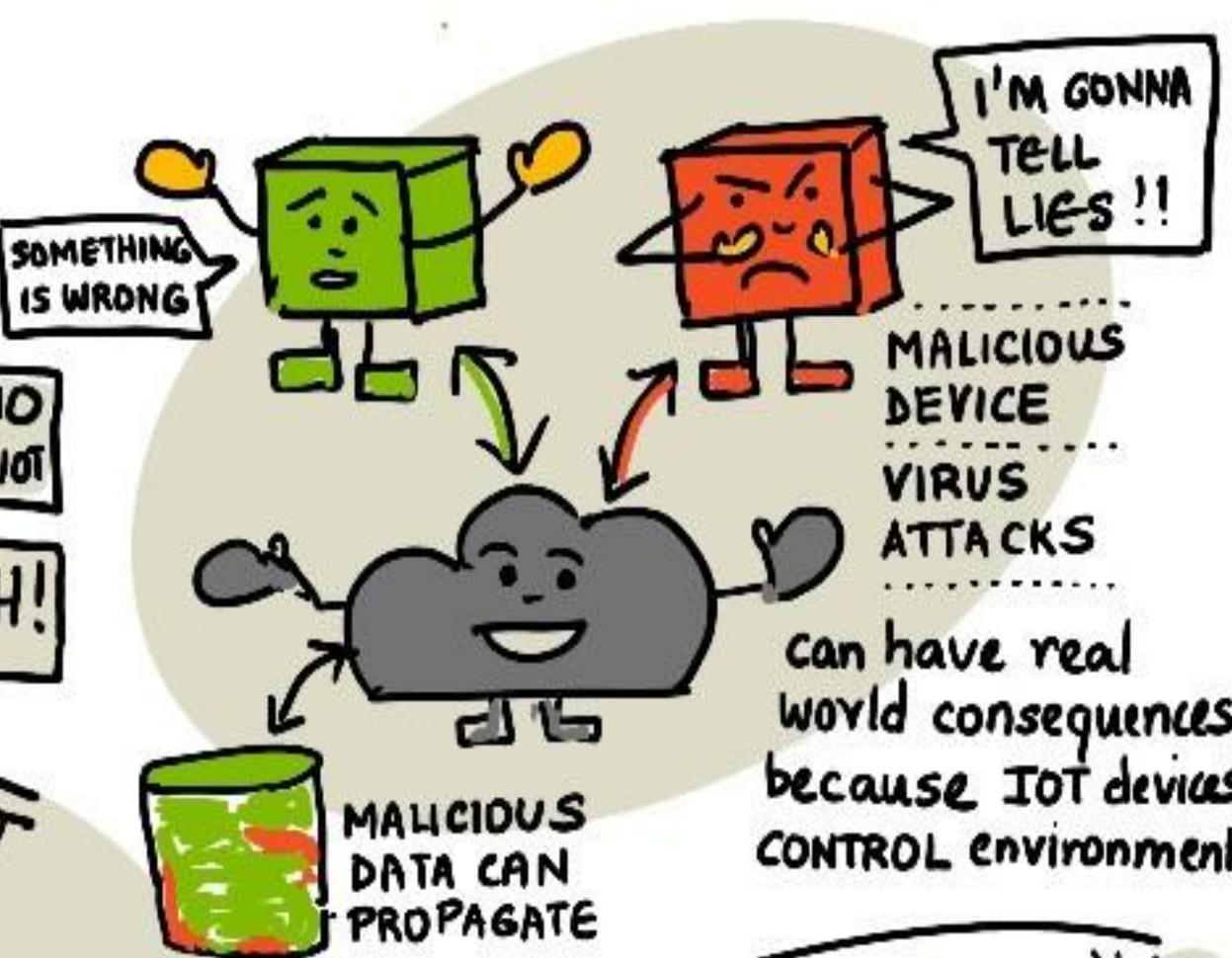
# IOT SECURITY



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



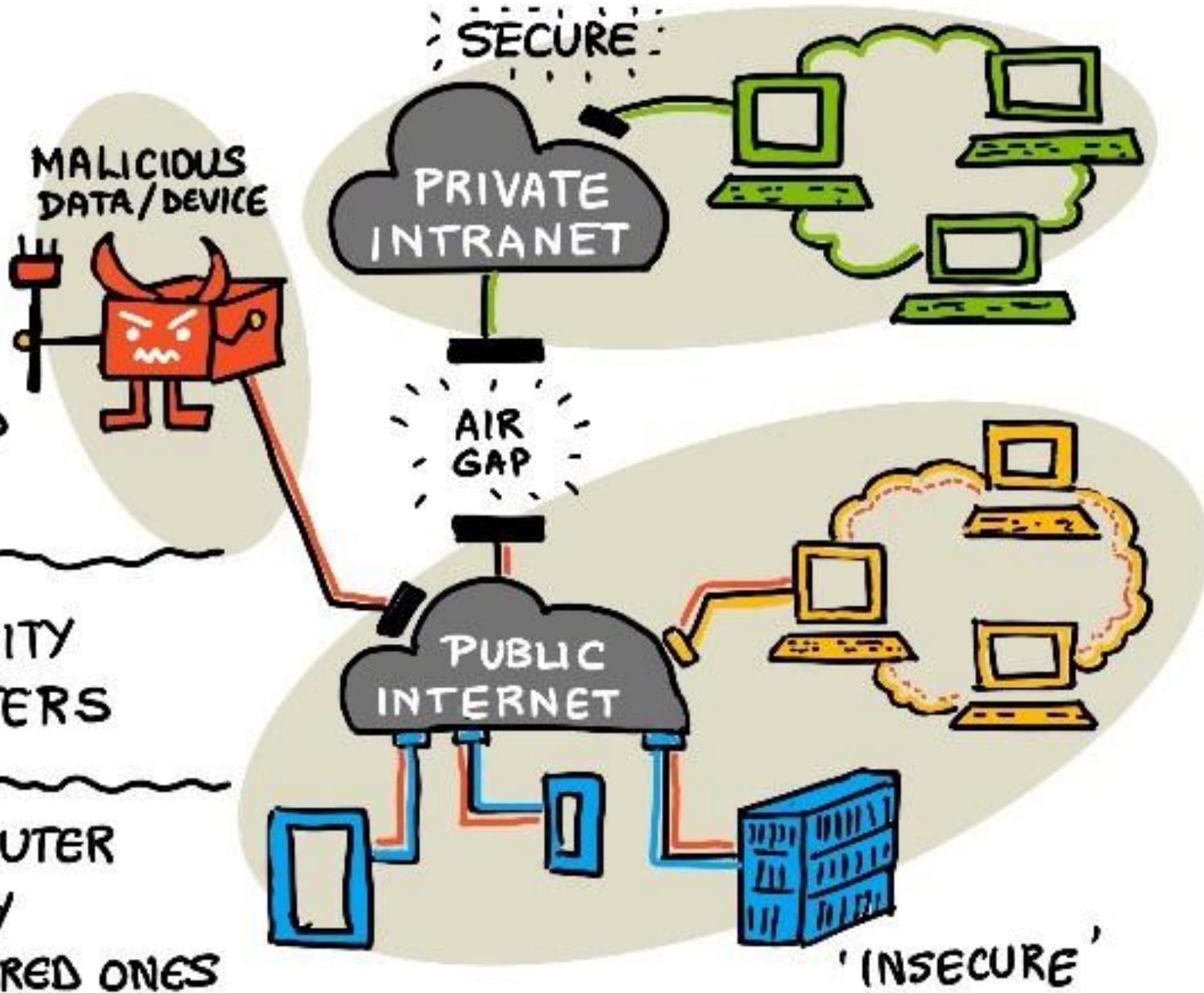
# AIR GAPPING

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

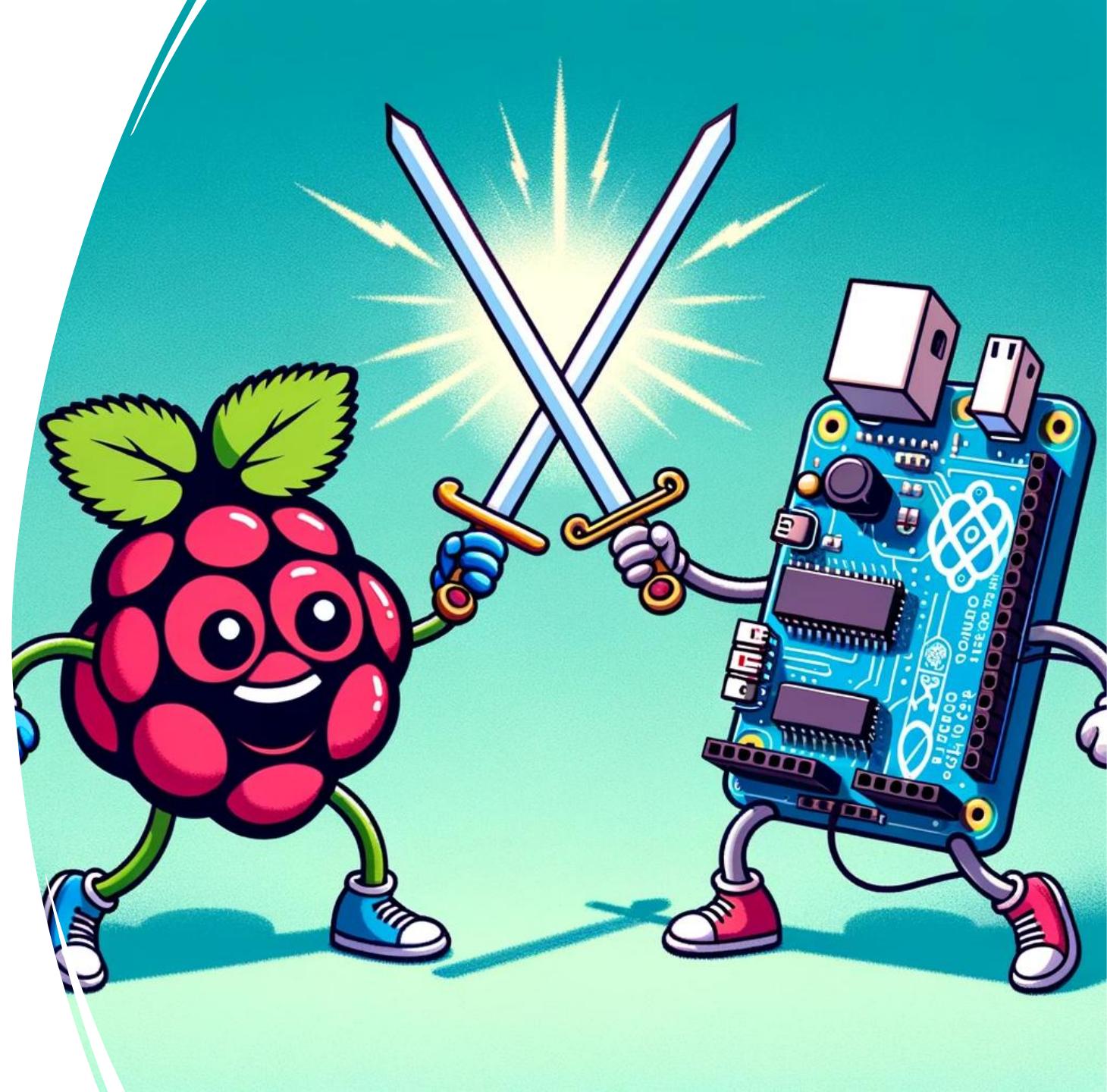
- What other security challenges do you think IoT Systems have?



# Single Board Computers

vs

# Microcontrollers



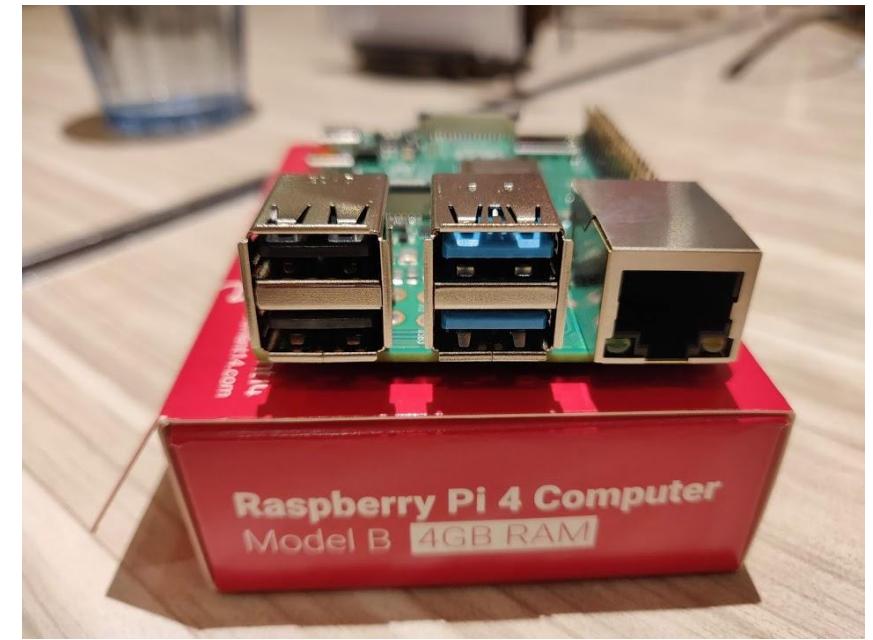
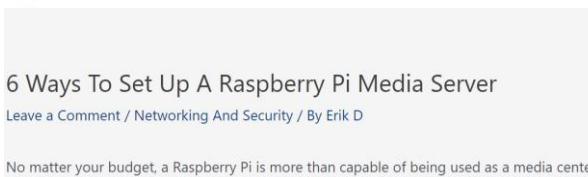
# SBC vs µC

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



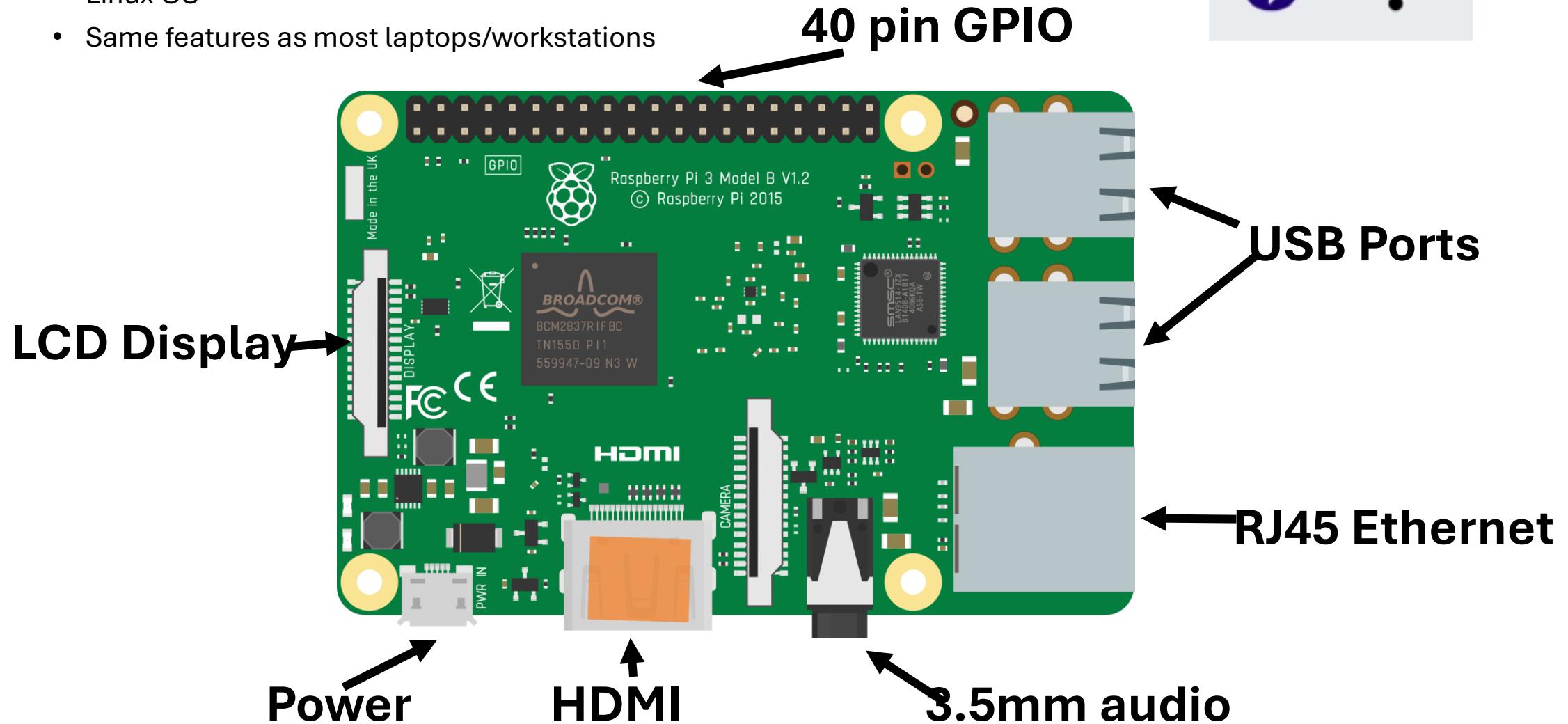
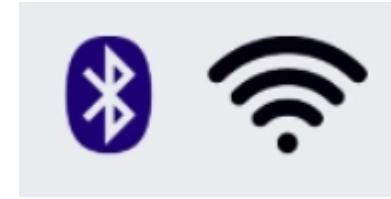
# What is a Single Board Computer(SBC)?

- 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



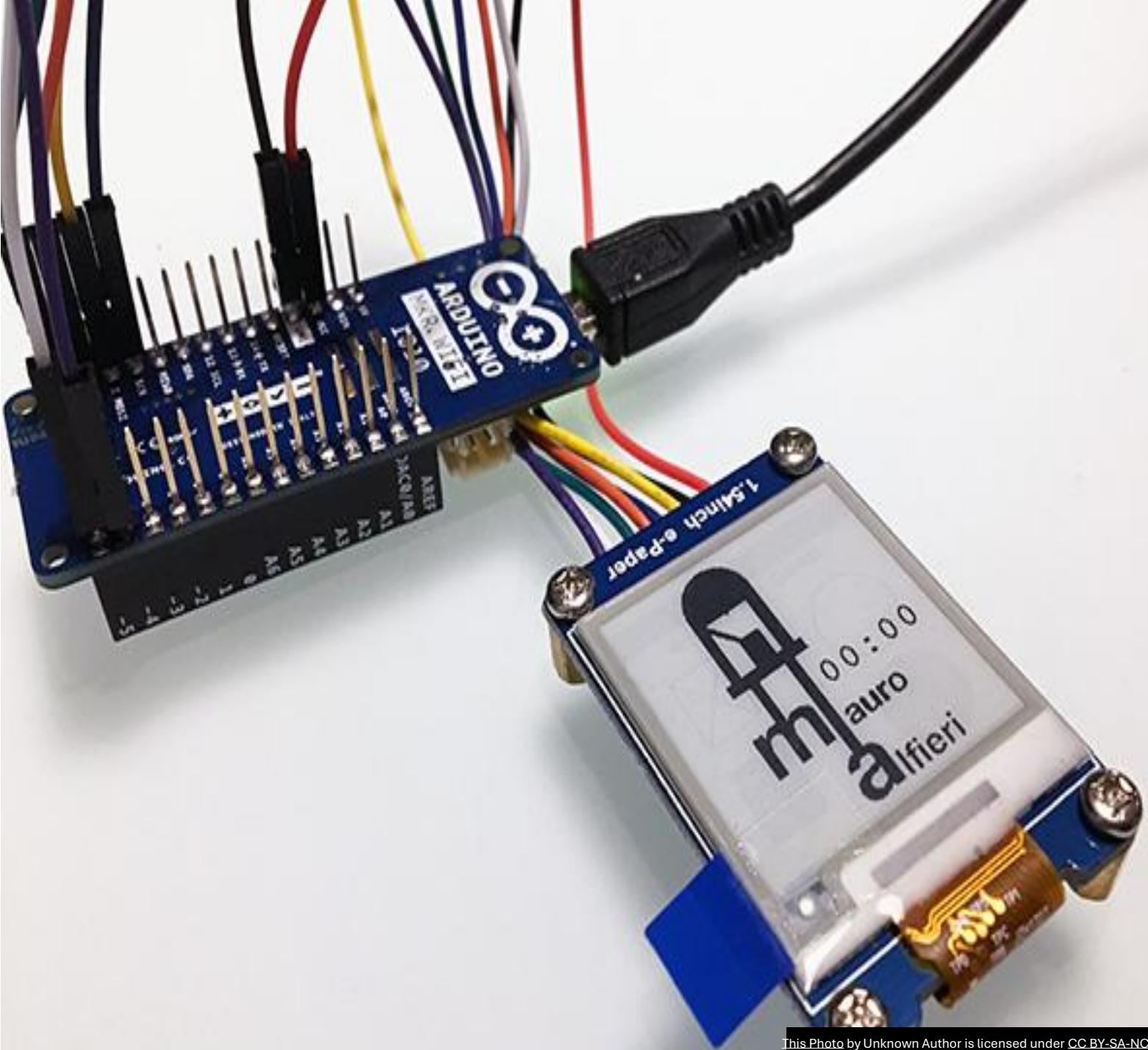
# What is a Microcontroller?

- 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

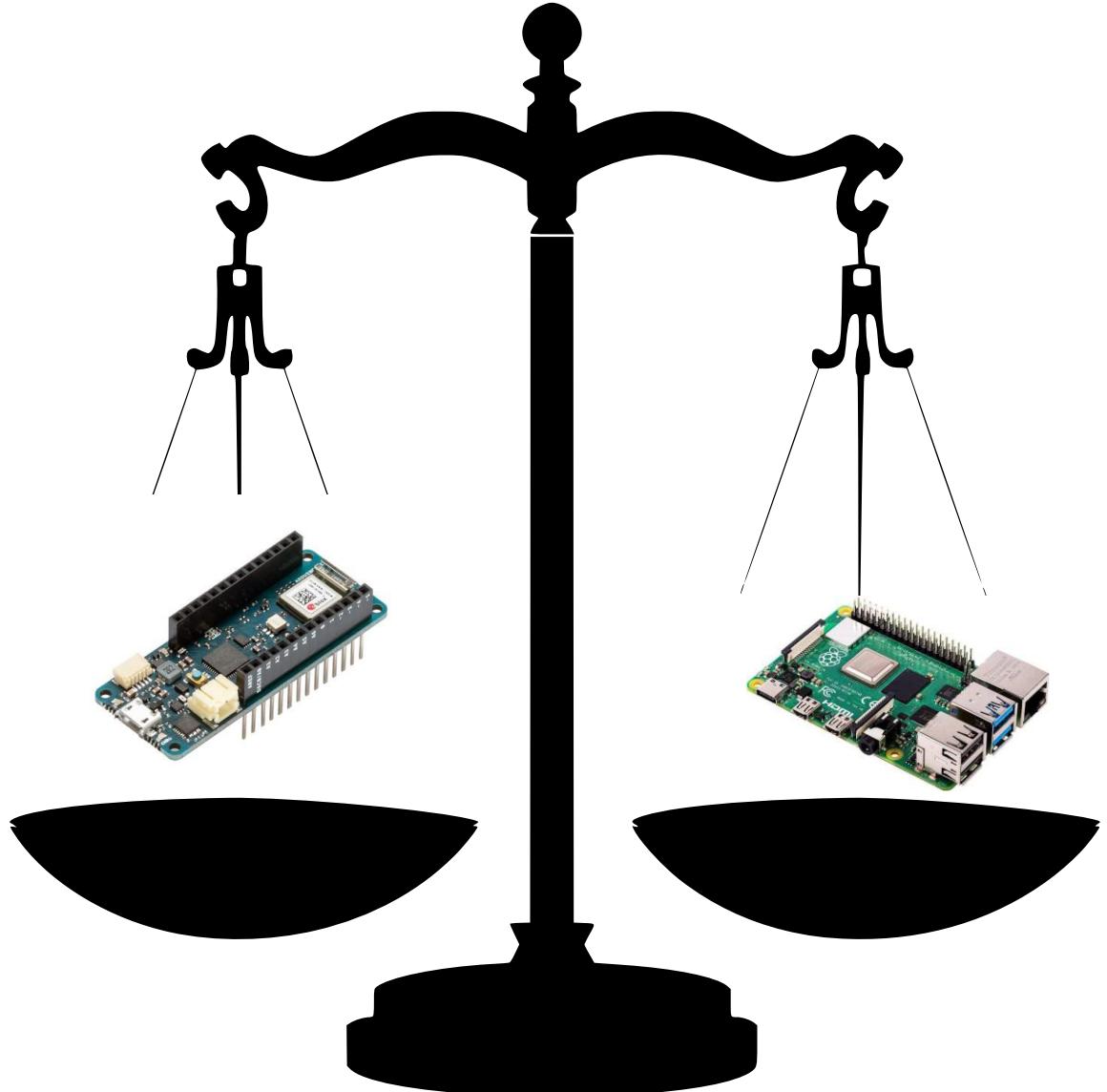
€0.92 Off  
Store Coupon

# 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.**

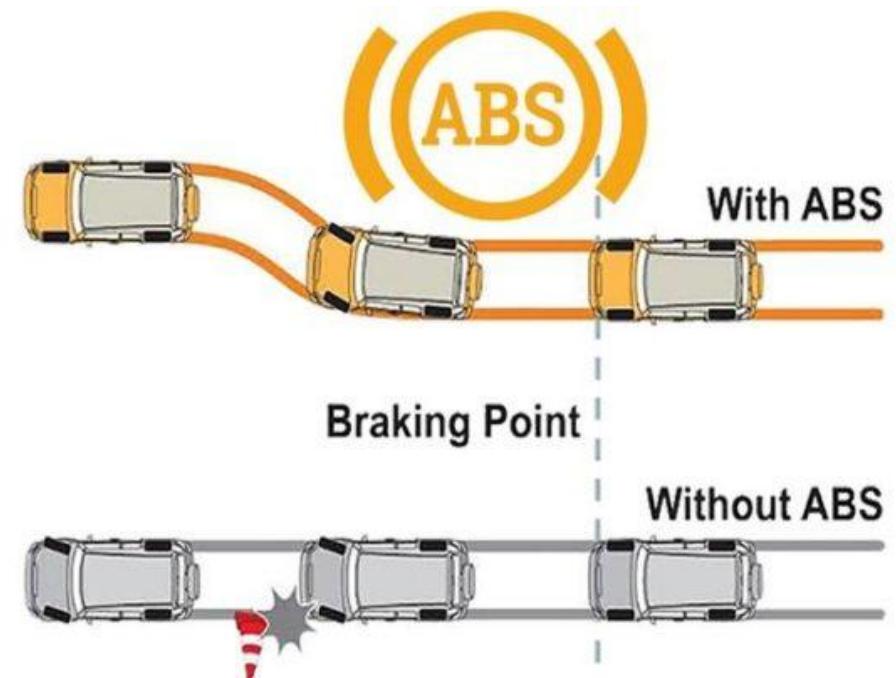
# Which one???

- **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

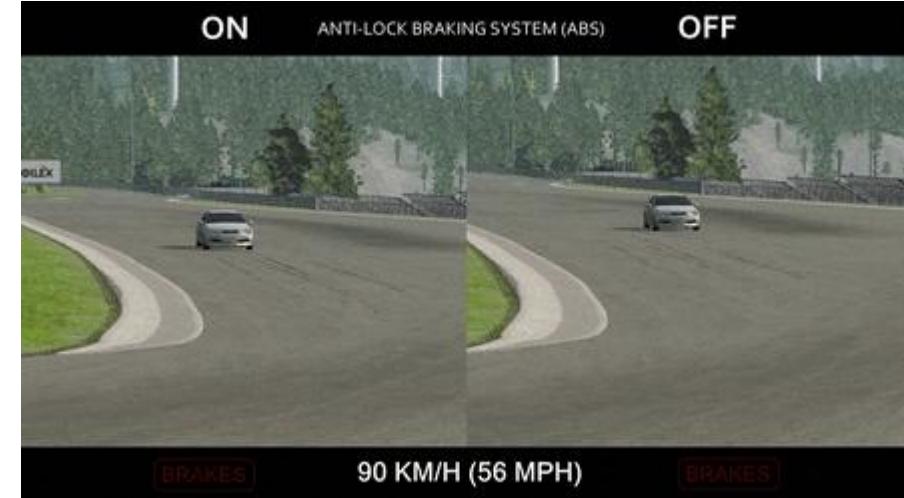
- **Anti-lock Braking System (ABS)**
  - ABS is a safety anti-skid braking system that prevents the wheels from “locking up” during braking, which helps the driver maintain control.
- **How it works:**
  - Wheel Speed Sensors constantly measure the speed of each wheel and send this data to the ABS controller.
  - 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))



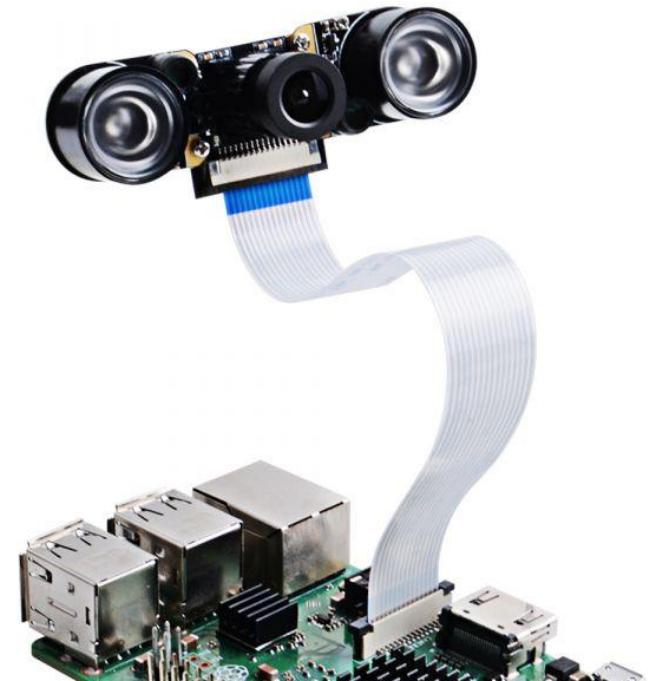
## Example Use Case2: Licence Plate Recognition (LPR)

- 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.

