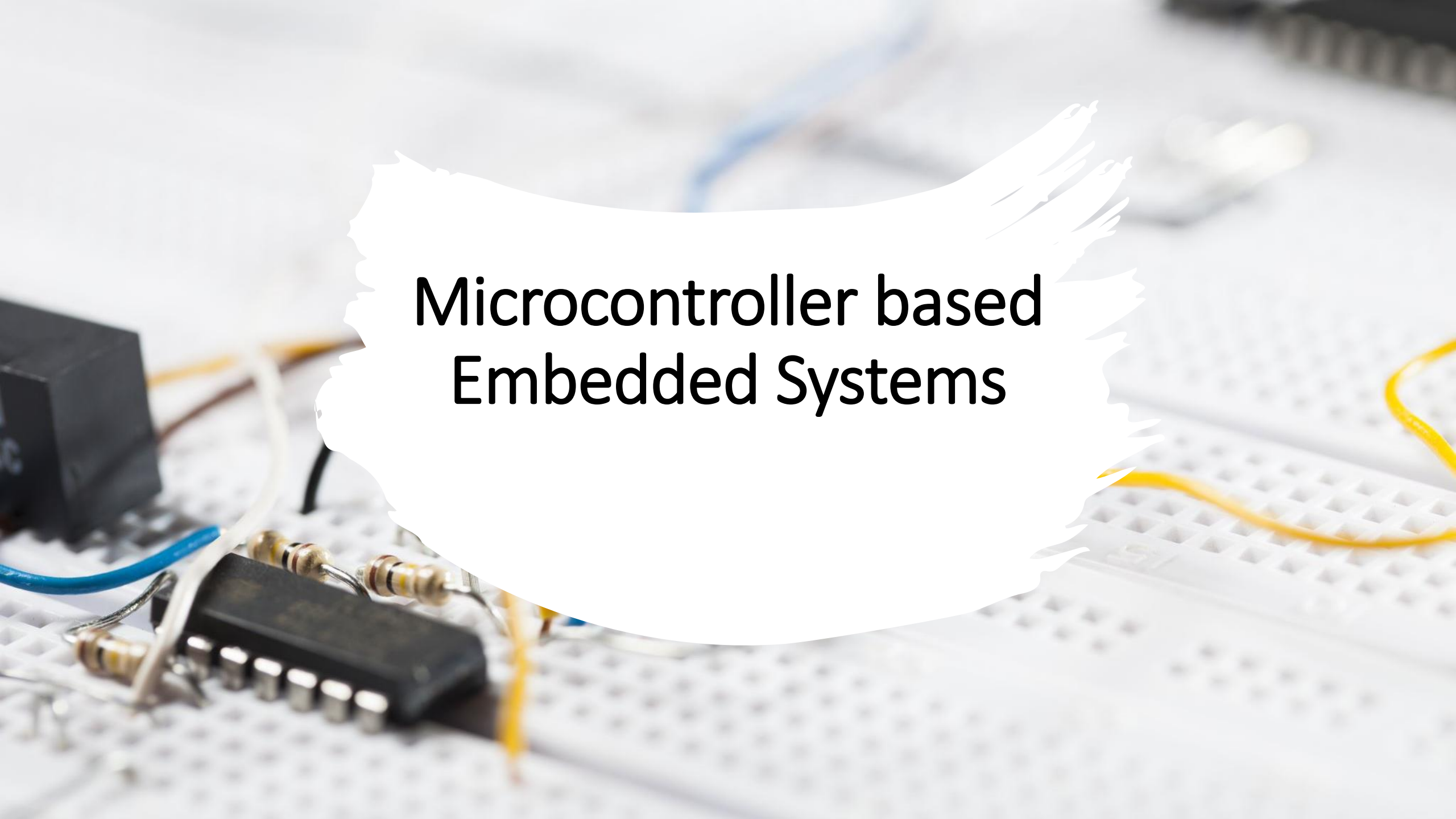


Microcontroller based Embedded Systems

General Purpose vs Embedded Computing

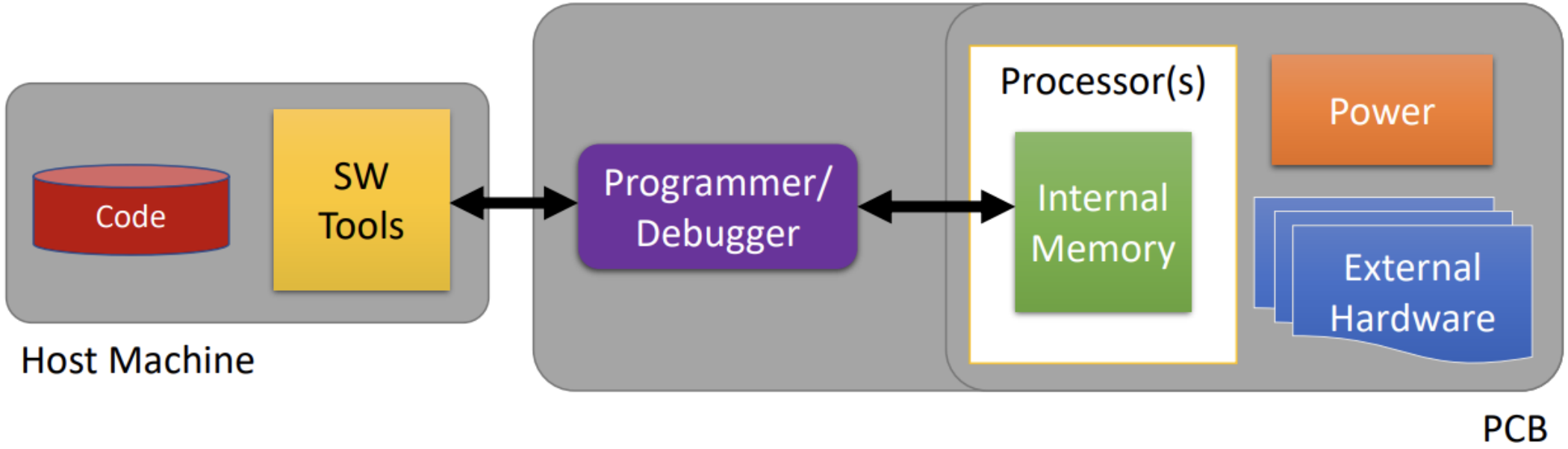
General Purpose Computing	Embedded Systems
Designed to run a broad class of applications that may not be known at design time	Run a single or few specialized applications often known at design time
Programmable by end user	Typically, not programmable by end user
Higher power, no real-time constraints (in general), runtime performance may not be fully predictable	Low power, real-time constraints, predictable runtime performance
Heavy weight, multi-tasking OS (Windows, Linux)	Lightweight, real-time OS or no OS
Faster (higher performance) is always better	Once the requirements are met, a faster processor is not desirable (overkill) due to increased cost/power.

General Purpose Computing

Embedded Systems

Embedded Systems

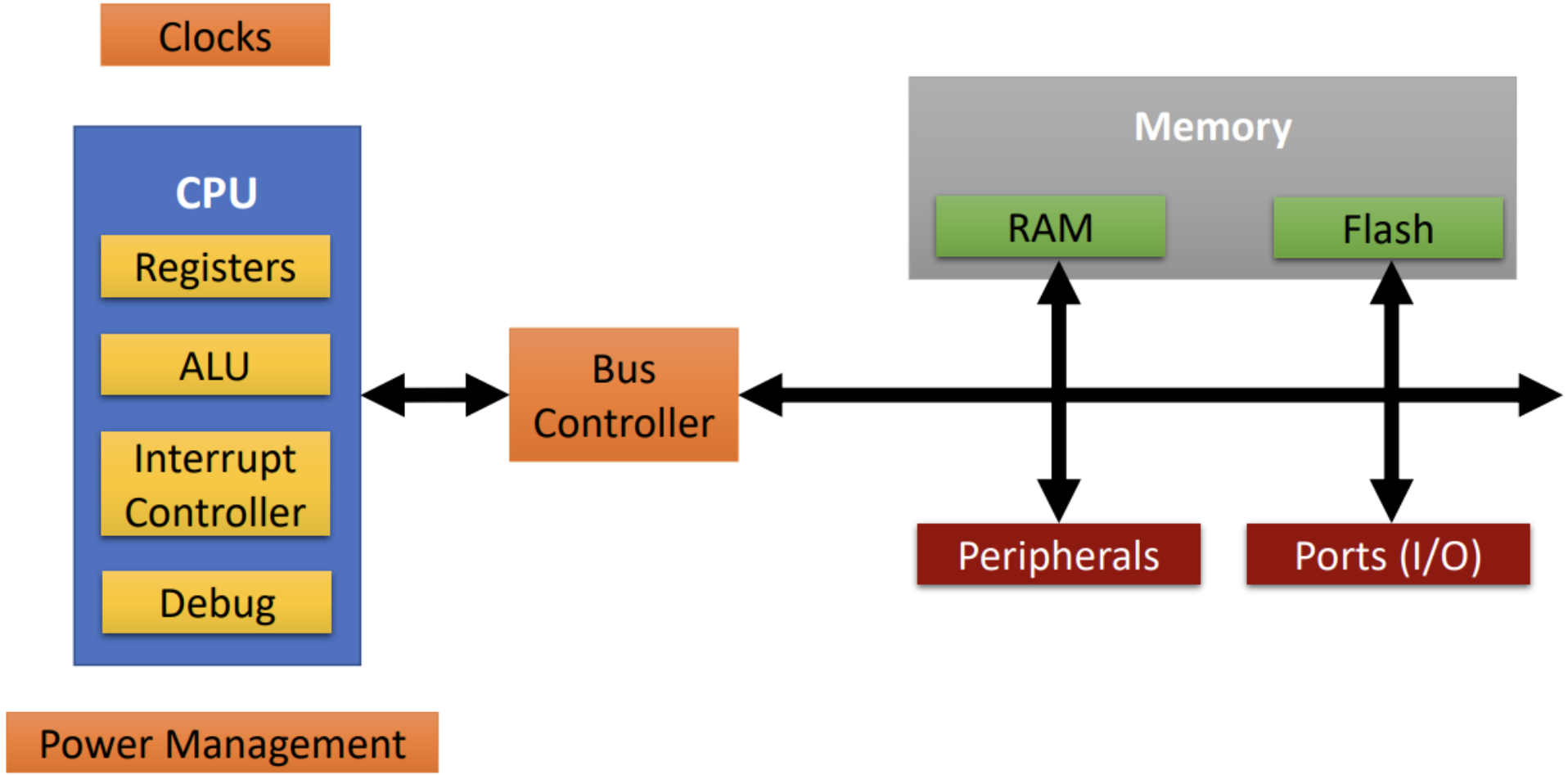
- Embedded systems are computer systems designed to perform a specific task, often with limited resources such as memory and processing power.



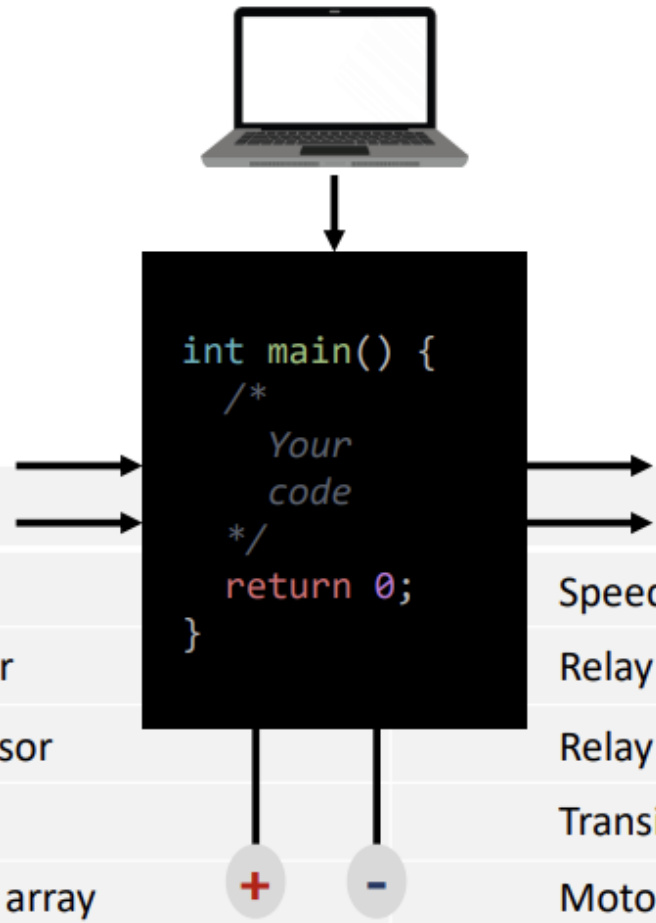
Microcontrollers

Microcontrollers

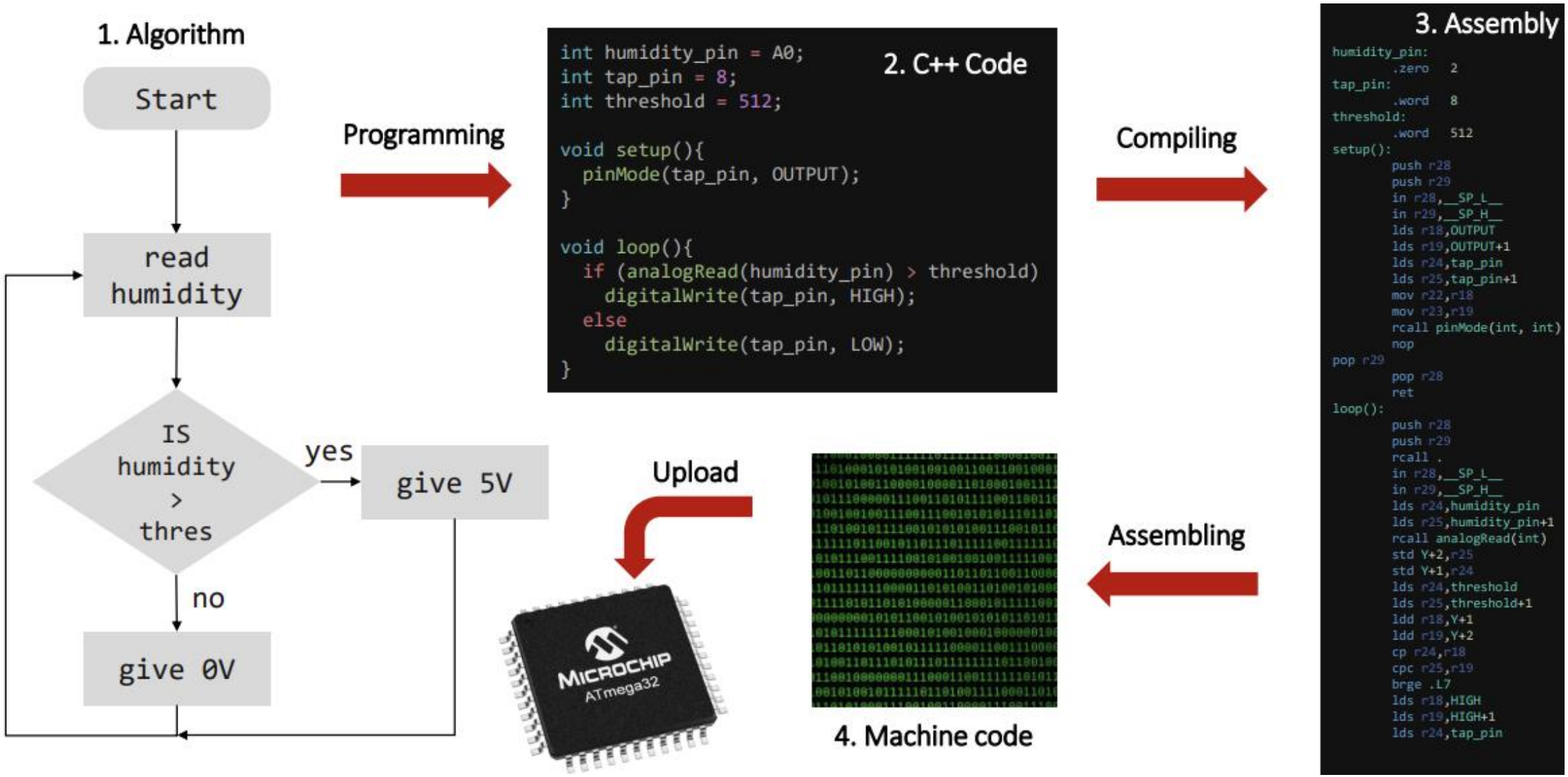
- A microcontroller is a small computer on a single integrated circuit chip that is designed to control specific devices or perform a particular set of functions.

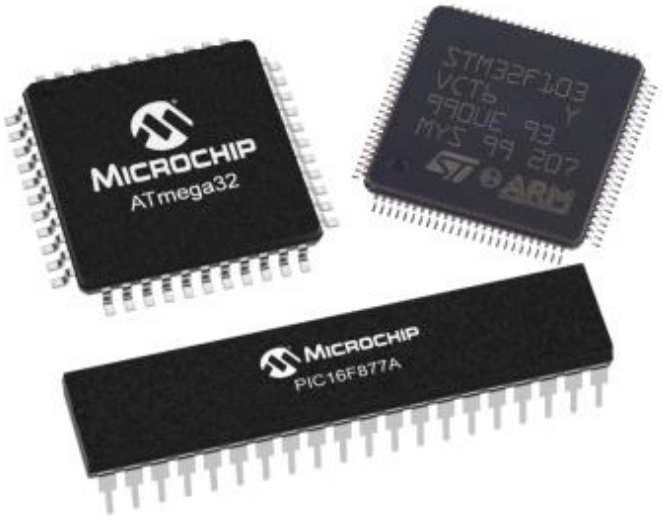


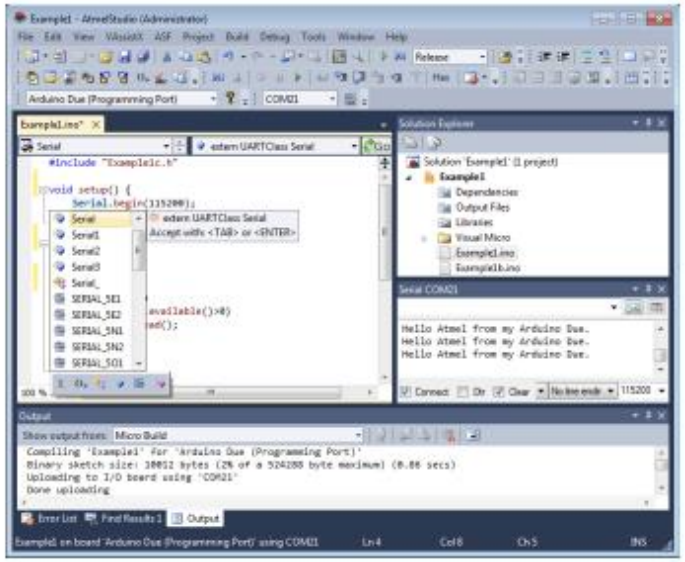
- An electronic machine
- Tiny, programmable computer
- Low power – can run for days / months on battery
- Cheap – 200 to 1500 LKR
- Automate tasks at home, build and sell products



	Inputs	Outputs
1. Adjust fan speed based on temperature	Temp sensor	Speed controller
2. Water plants based on humidity	Humidity sensor	Relay switch -> Solenoid valve
3. Fill water tank, based on water level	Water level sensor	Relay switch
4. Vesak decorations – lights in a pattern	Push button	Transistor amplifier -> Lights
5. Line following robot	Line (IR) sensor array	Motor controller -> Motor
6. Voice controlled AI system	Microphone	Any
7. Mini-self driving car (ARM)	Camera	Motor controller -> Motor


1. Algorithm
Start
read humidity
IS humidity > thres
yes
give 5V
no
give 0V
Programming
2. C++ Code
int humidity_pin = A0;
int tap_pin = 8;
int threshold = 512;
void setup(){
pinMode(tap_pin, OUTPUT);
}
void loop(){
if (analogRead(humidity_pin) > threshold)
digitalWrite(tap_pin, HIGH);
else
digitalWrite(tap_pin, LOW);
}
Compiling
3. Assembly
humidity_pin:
.zero 2
tap_pin:
.word 8
threshold:
.word 512
setup():
push r28
push r29
in r28,__SP_L__
in r29,__SP_H__
lds r18,OUTPUT
lds r19,OUTPUT+1
lds r24,tap_pin
lds r25,tap_pin+1
mov r22,r18
mov r23,r19
rcall pinMode(int, int)
nop
pop r29
pop r28
ret
loop():
push r28
push r29
rcall .
in r28,__SP_L__
in r29,__SP_H__
lds r24,humidity_pin
lds r25,humidity_pin+1
rcall analogRead(int)
std Y+2,r25
std Y+1,r24
lds r24,threshold
lds r25,threshold+1
ldd r18,Y+1
ldd r19,Y+2
cp r24,r18
cpc r25,r19
brge .L7
lds r18,HIGH
lds r19,HIGH+1
lds r24,tap_pin
Upload
MICROCHIP
ATmega32
Assembling
4. Machine code

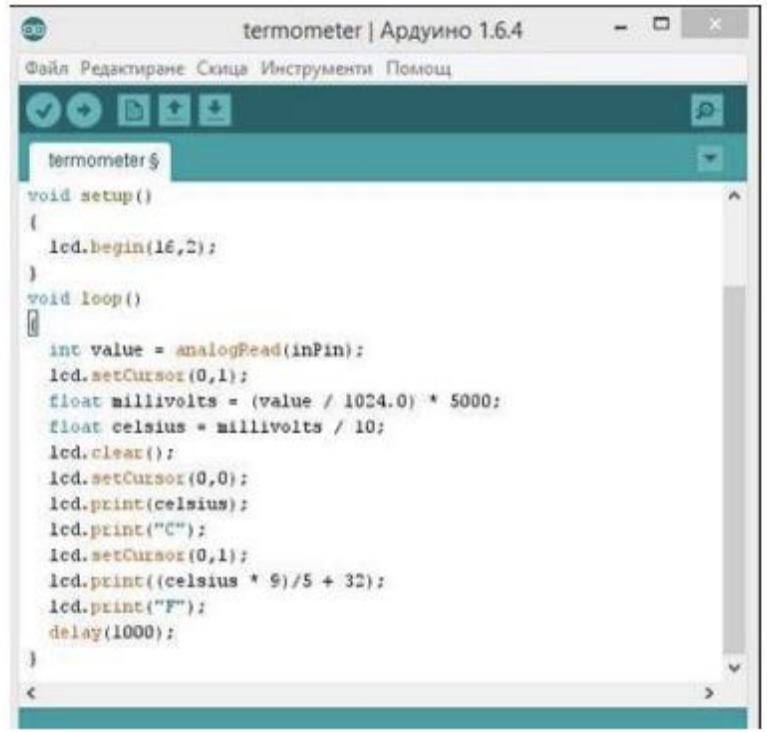




- AVR, PIC, STM-32
- 8-bit vs 32 bit

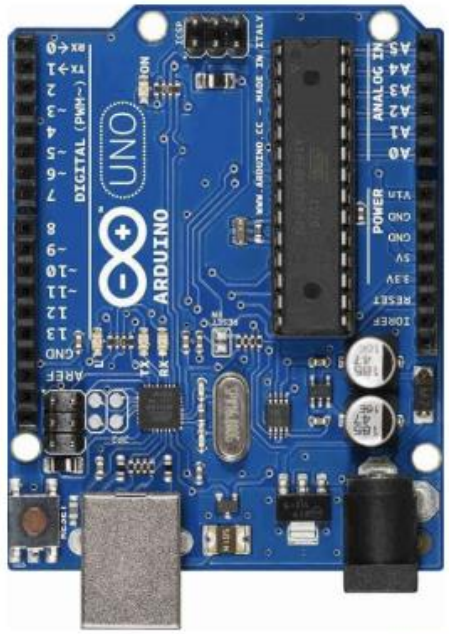
	Atmega 328p	Attiny 85
Bit	8	8
Speed (MIPS)	20	20
Program Memory (Flash)	32	8
Data Memory (SRAM)	2048	512
Data EEPROM	1024	512
Pins	32	8
PWM (Analog out)	6	5
Price	Rs 600	Rs 275

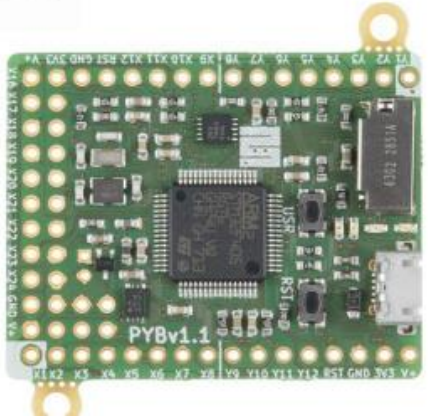
- Cheap, pre-built development board
- Plug and play modules



- C++ , made extremely simple
- Tons of easy-to-use libraries
- Limited flexibility

Microcontroller based Development Boards



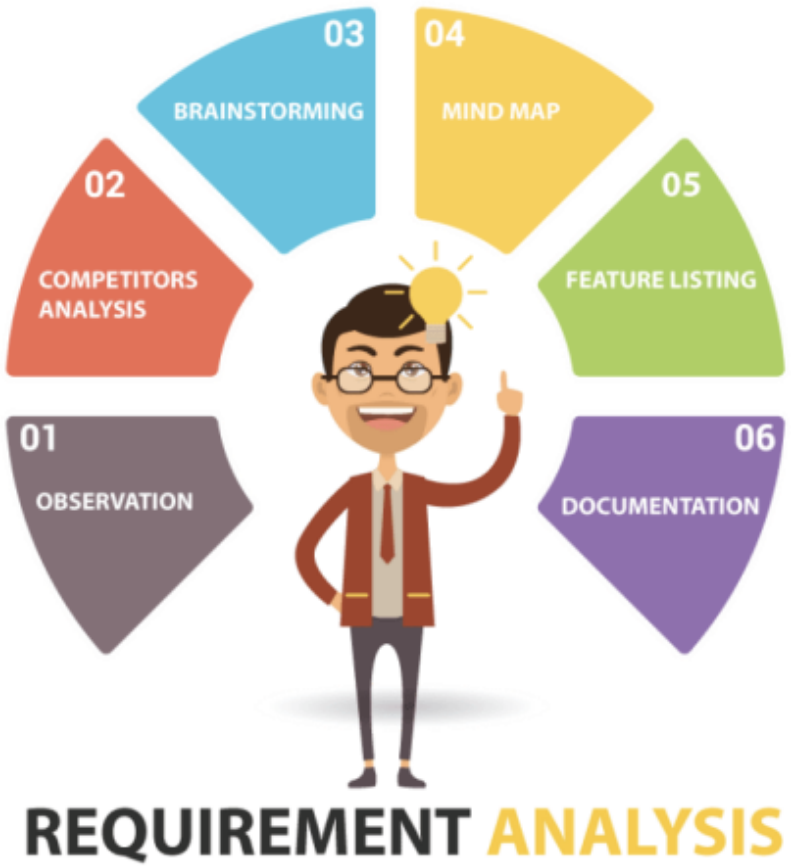


	Arduino UNO	Arduino Nano	Blue Pill STM32uino	PyBoard
Architecture	AVR		ARM	ARM
Language	C++			Python
Bit	8		32	32
Speed (MHz)	16		72	168
Program Memory (Flash)	32KB		64/128 KB	112 KB
Data Memory (SRAM)	2 KB		20 KB	192 KB
Data EEPROM	1 KB		-	
Size (cm)	6.8 x 5.3	4.5 x 1.8	5.3 x 2.3	4.3 x 4
PWM (Analog out)	6		15	20
SPI, I2C	1,1		2,2	2,2
Price	Rs 1500	Rs 960	Rs 1300	Rs 3631

Prototyping

- We see many Sri Lankan inventions & innovations – eg: exhibitions
- Only very few come to market – why?
- Prototypes are not product
- Step 1 – Prototype development

  High unit cost, low lifetime, may not be mass-manufacturable
  - Proof-of-concept
  - Working prototype
  - Visual prototype
  - User experience prototype
  - Functional prototype
- Step 2 – Product development



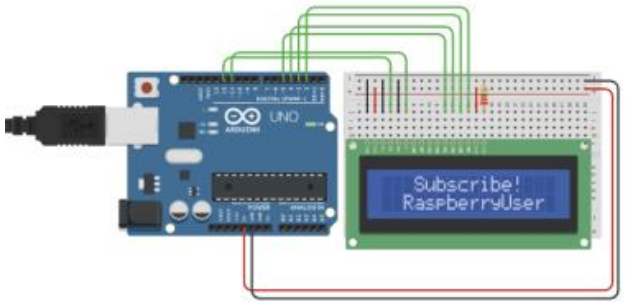
User-driven development

- Less risky
- Identify stakeholders, create user stories
- Extensive user surveys, market analysis
- To decide features, select requirements, identify constraints
- Brainstorm ideas, made design

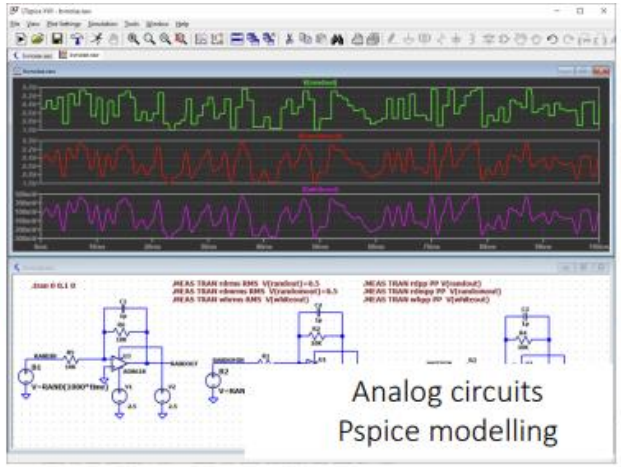
Design-driven development

- Risky, eg: Apple
- Still needs to be tested / cross-checked with users

Prototype Development - Simulation



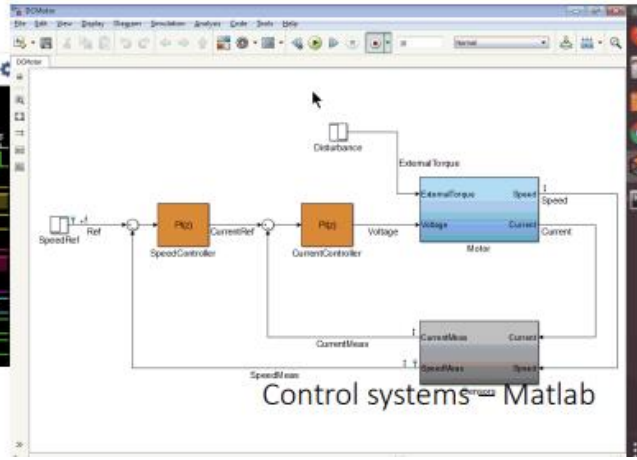
Microcontroller code & circuits
Tinkercad, Proteus



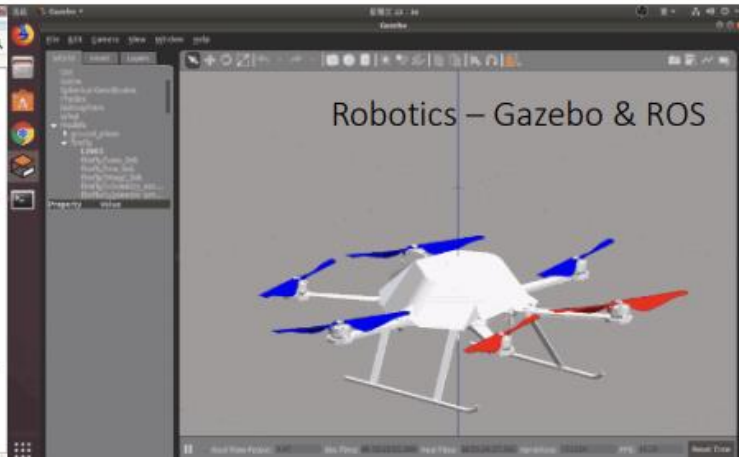
Analog circuits
Pspice modelling

Mechanical models - Solidworks

Digital circuits: Modelsim, Vivado simulator, verilator



Control systems – Matlab



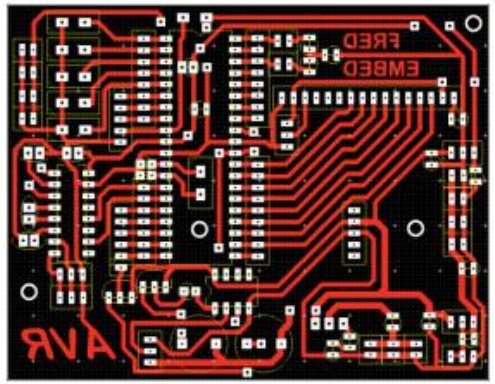
Robotics – Gazebo & ROS

Prototype Development

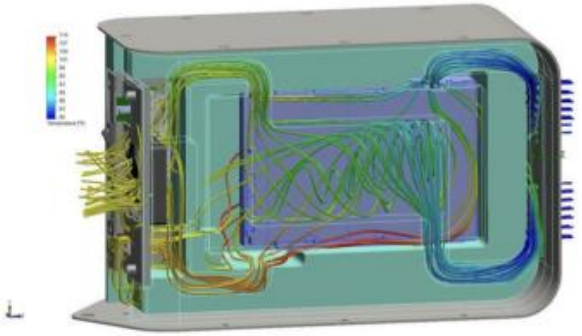
Development board-based prototypes

- Development boards (Arduino, Raspberry pi, STM) have well-tested support circuits: power, all peripherals
- Can quickly try different communication methods, designs
- Ideal for prototyping
- Not suitable for final product:
  - Cannot be mass manufactured
  - May not meet requirements: waterproof, dust proof, lifetime
  - High unit price

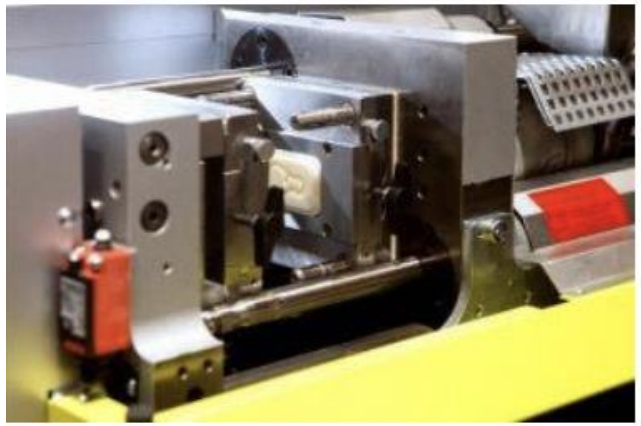
Final Prototype



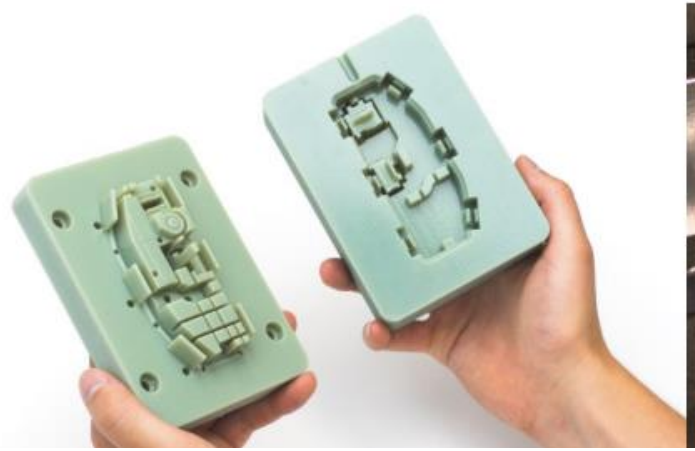
- PCB Design – Altium, Eagle, EasyEDA (free)
- PCB Manufacturing, assembling, testing
- Enclosure – Solidworks, Onshape (free)
- 3D printed enclosure
- Manufacturing concerns – draft angles, ease of assembly

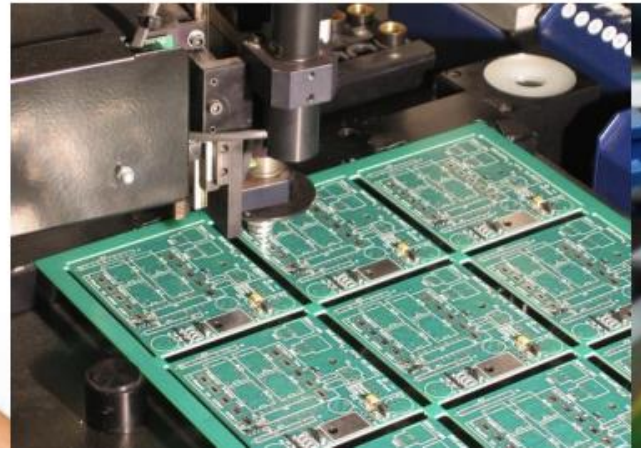


Manufacturing



- Enclosure – Injection molding, making dies
- PCB – outsourcing / in-house





2023 : ML is in almost every aspect of our lives

TinyML

Tiny Machine Learning (TinyML) is a field of study at the intersection of machine learning (ML) and embedded systems that **enables running ML models on devices with extremely low-power microcontrollers**.

TINY ML

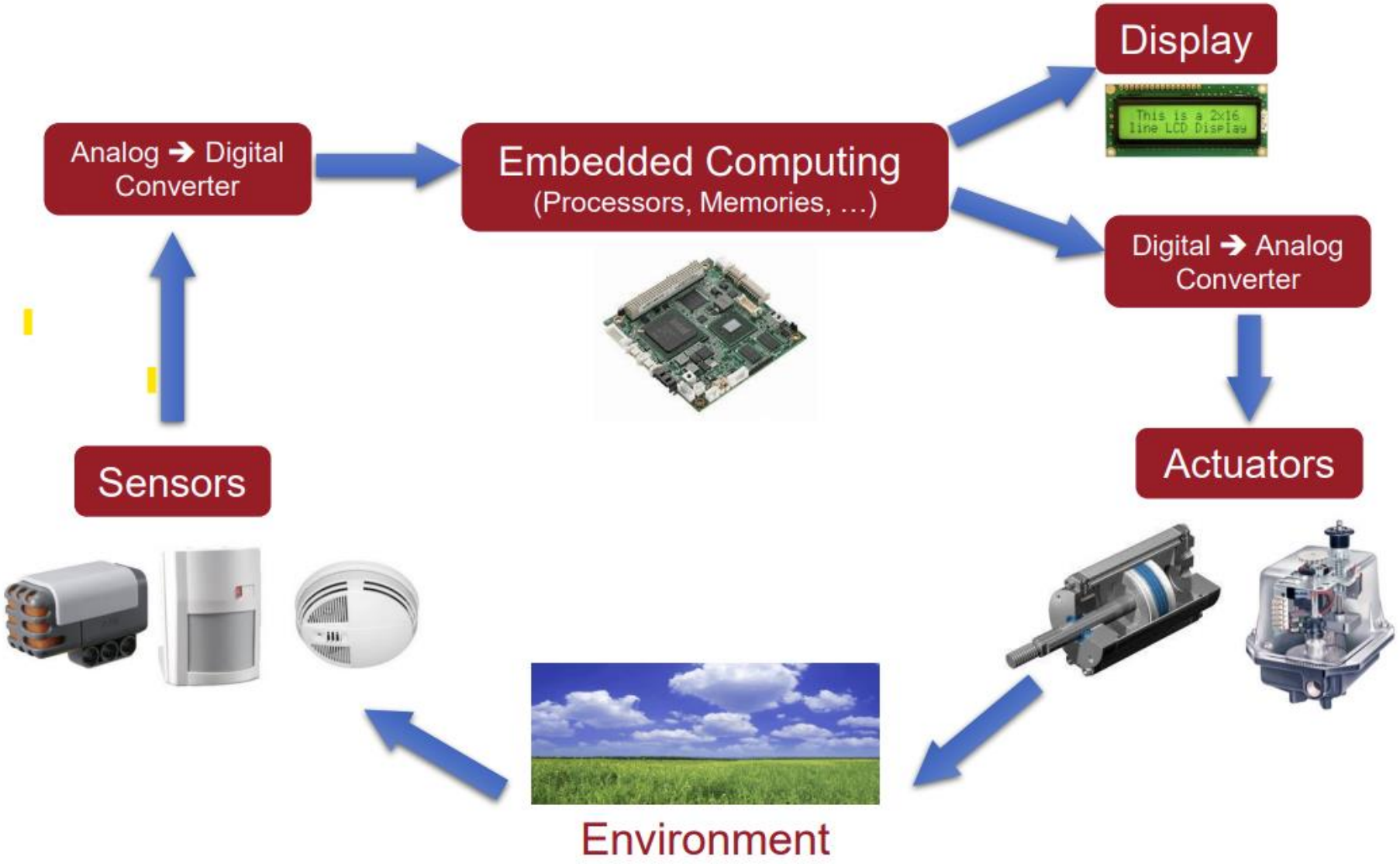
ADVANTAGES

- **Fast inference with low latency** – less communication with cloud
- **Data privacy** with data kept at edge
- **Doesn't depend on connectivity** – smart inference without internet connection

CHALLENGES

- **Limited memory** – puts restrictions on the size and the runtime of the machine learning models deployed on these devices.
- **Troubleshooting** – harder to determine and fix the performance issues.

Embedded Systems – Simplified Block Diagram

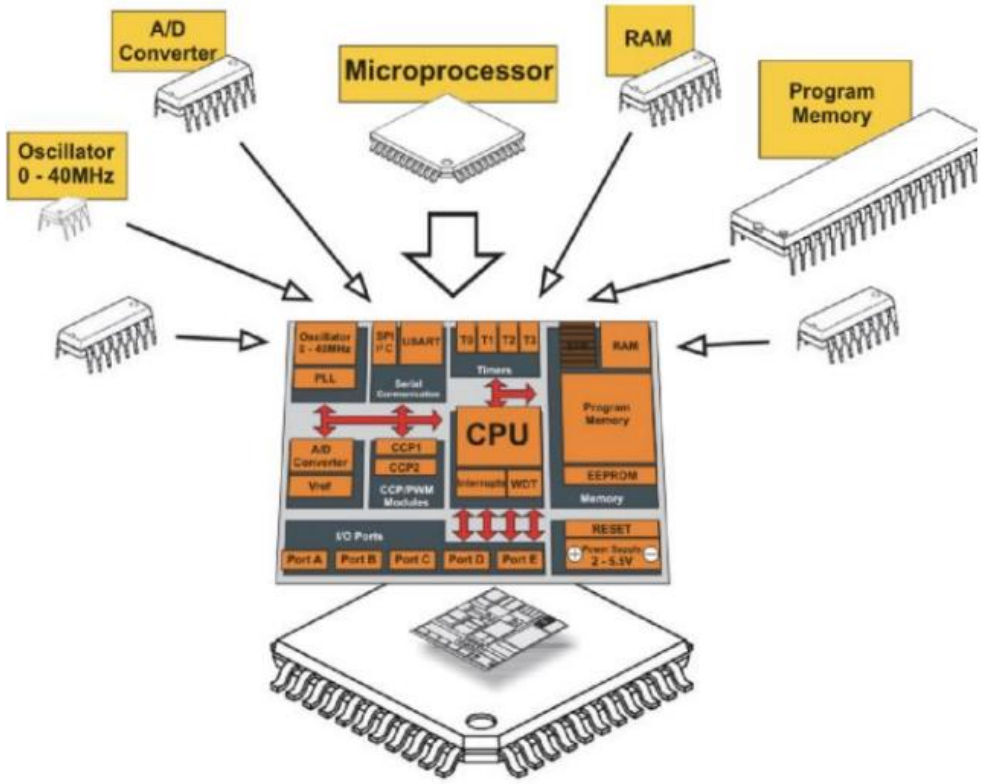


Microcontroller Vs Microprocessor

- A **microcontroller (µC or MCU)** is a small computer on a single integrated circuit containing a processor core, memory (RAM or ROM), Analog-to-Digital A/D converters and I/O ports.

- The MCU technically contains a simplified version of a CPU (a reduced power microprocessor).

- A **microprocessor** does not essentially contain any memory (RAM or ROM) or I/O ports and can only operate as part of wider embedded systems.



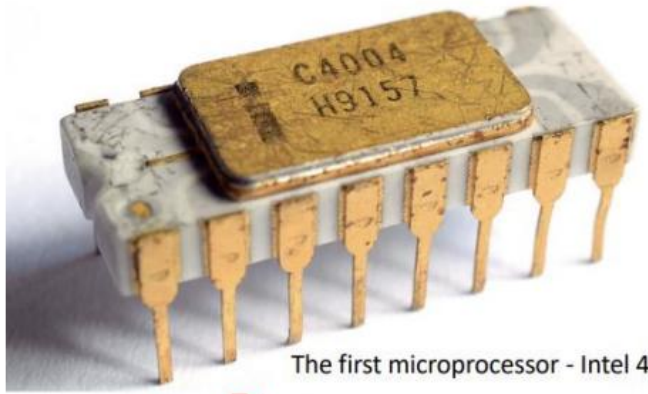
Microcontroller – Invention

- In 1970 and 1971, about the same time Intel was inventing its microprocessor, **Gary Boone**, an engineer at **Texas Instruments (TI)** and his team was working on a similar idea.

- At that time Texas Instruments was interested in making pocket calculators, and Boone designed a single integrated circuit chip that held nearly all of the necessary circuits to make a calculator (except for the keypad and display). It was named **TMS1802NC.**

- It has 5000 transistors providing 3000 bits of program memory, 128 bits of RAM, and could be programmed to perform different functions.

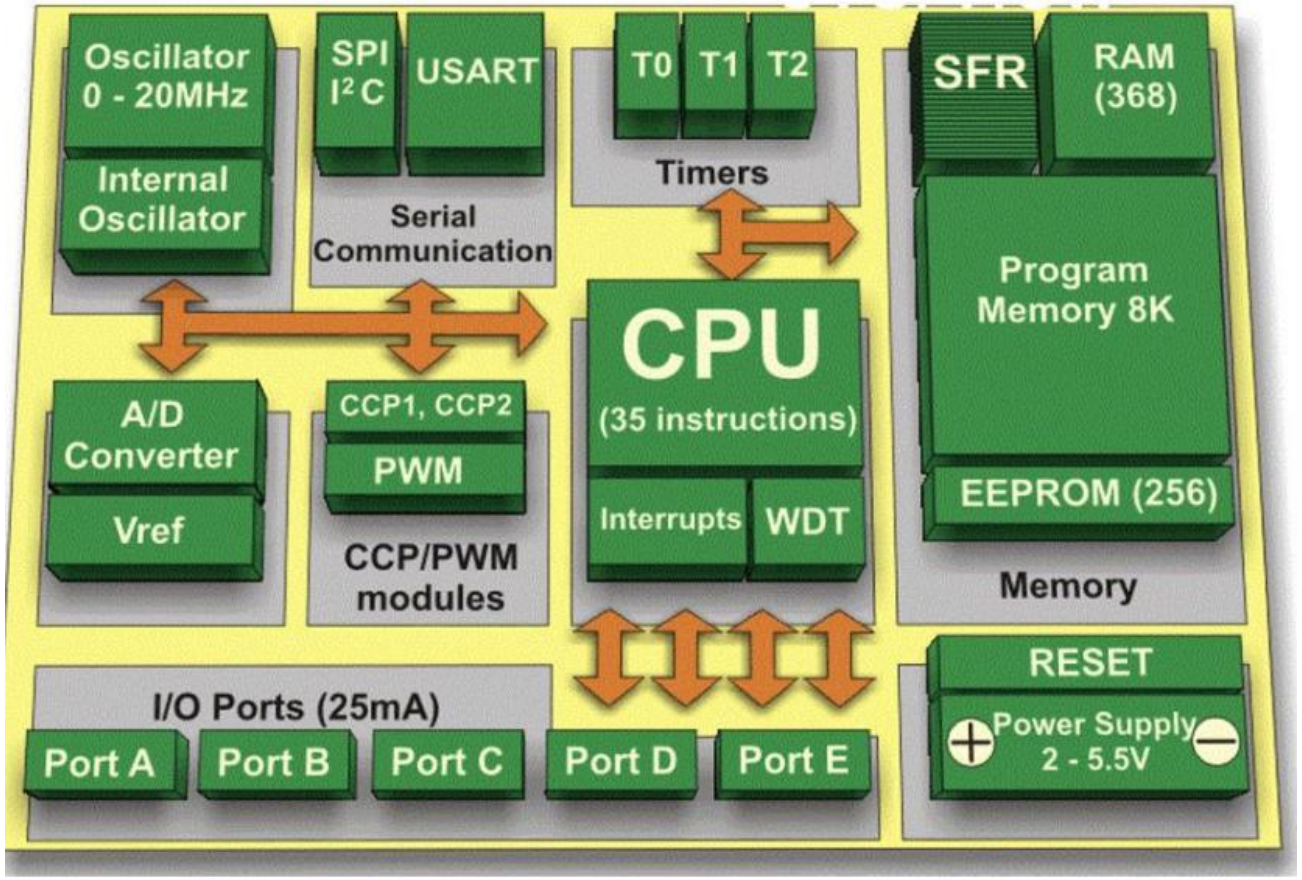
- The development of the microprocessor and microcontroller parted ways after this. Both were successful, but in terms of sheer numbers it was the **microcontroller** that sold the most.



The first microprocessor - Intel 4004 (197

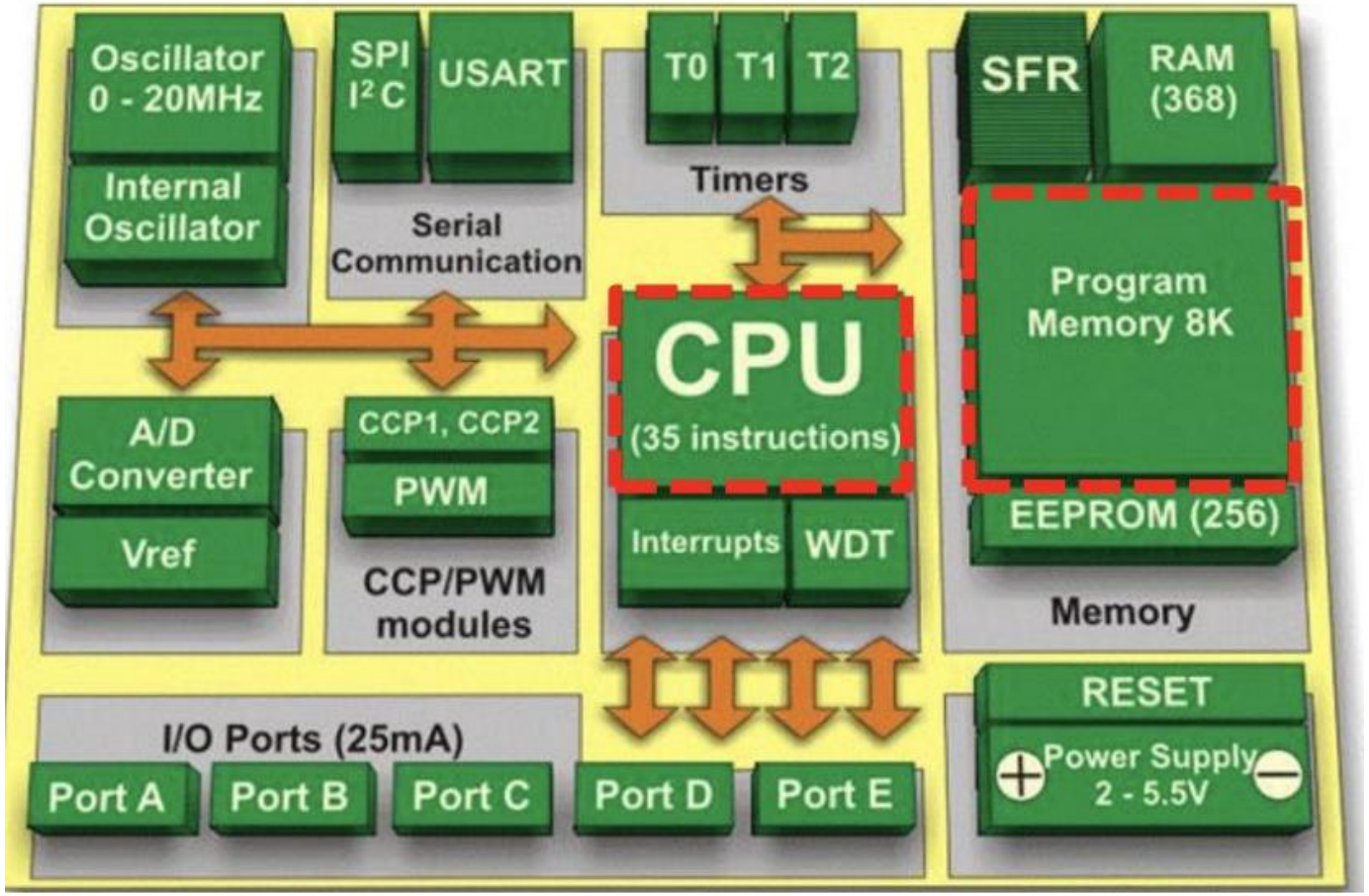
The first microcontroller - TMS1802NC (1970/71)

Components of a Microcontroller

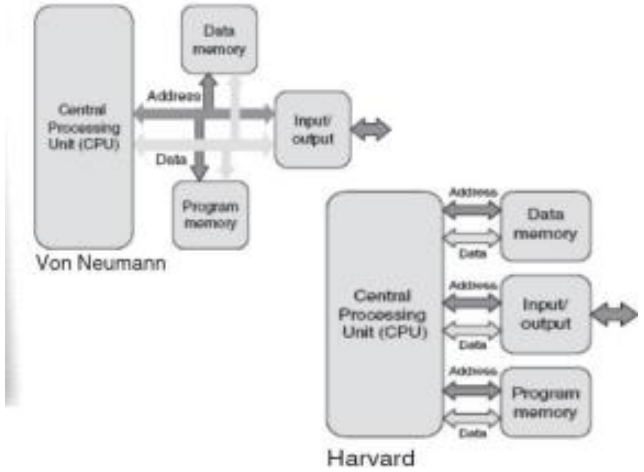


- **Central Processing Unit (CPU) -** ranging from small and simple 4-bit to complex 32 or 64-bit processors
- **Clock System and Timers**
- **RAM** - data storage
- **ROM, EPROM, EEPROM or Flash** - program and operation parameter storage
- **Serial I/O ports** - such as serial ports (USART)
- **Other serial** - I²C and Serial Peripheral Interface (SPI)
- **Peripherals** - timers, PWM generators, watchdog timers, internal oscillator
- **Analog I/O -** A/D converters

Executing the Code

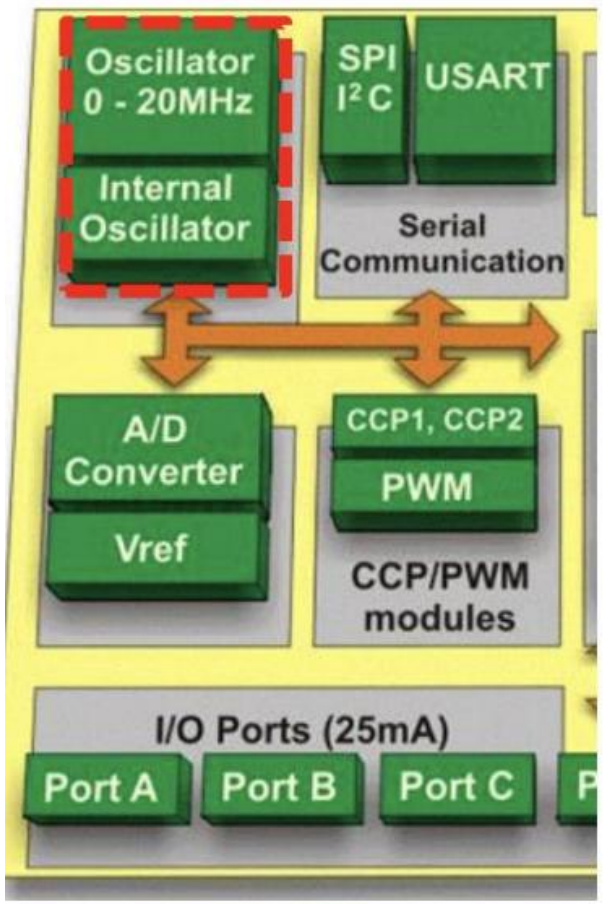


- The CPU executes the machine code, stored in the program memory, line by line (sequential).
- During the execution, data memory and I/O is addressed. The CPU is clocked by the clock system.
- Two main architectures: **Von Neumann (one bus)**, **Harvard (several busses).**

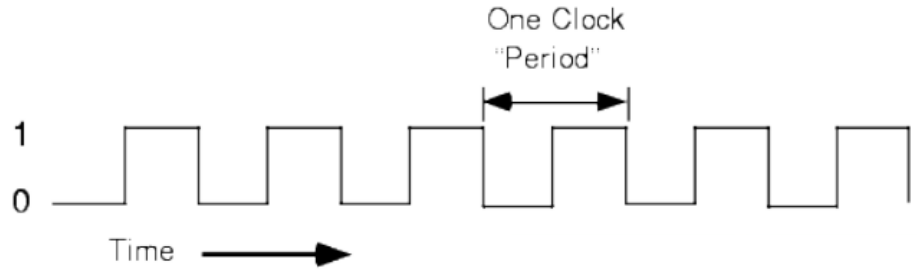


Components of a Microcontroller

01) Clock System

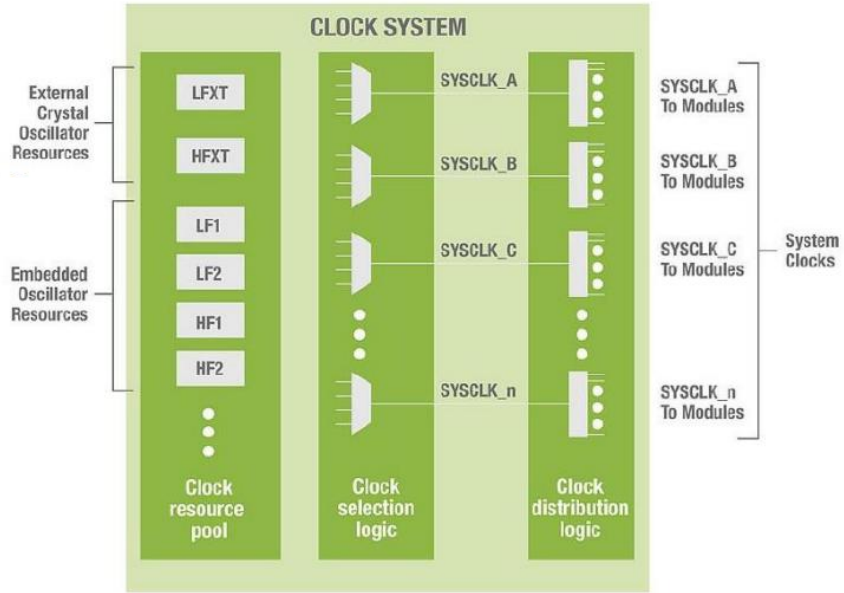


The clock system generates a clock signal, whose frequency is driving the CPU and other units of the microcontroller.
Most instructions are designed to execute in 1 or 2 clocks.



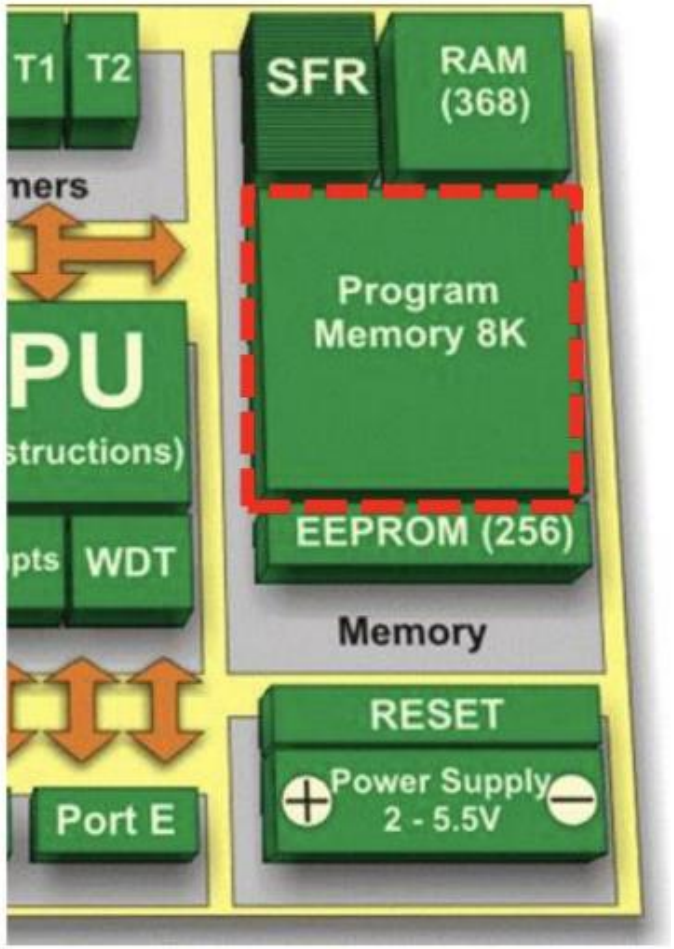
The typical clock frequency for modern microcontrollers is between 10 and 100 MHz.

- **Internal oscillator** -> not very precise (usually RC type)
- **External oscillator** -> usually high precision (often quartz crystal)



Components of a Microcontroller

02) ROM



ROM permanently saves the program (i.e., read only), corresponding to the program memory. We can only program the ROM using the µC programmer, not using the code you upload to the µC.

ROM can be built into the microcontroller or added as an external chip.

If External,

- The microcontroller is cheaper.
- The program can be considerably longer, since external memory typically are bigger.
- More pins are needed to interface the external ROM.

If Internal,

- Microcontroller is usually more expensive.
- Leaves more pins available (smaller PCB).

The typical size of a ROM ranges from 512B to 256KB

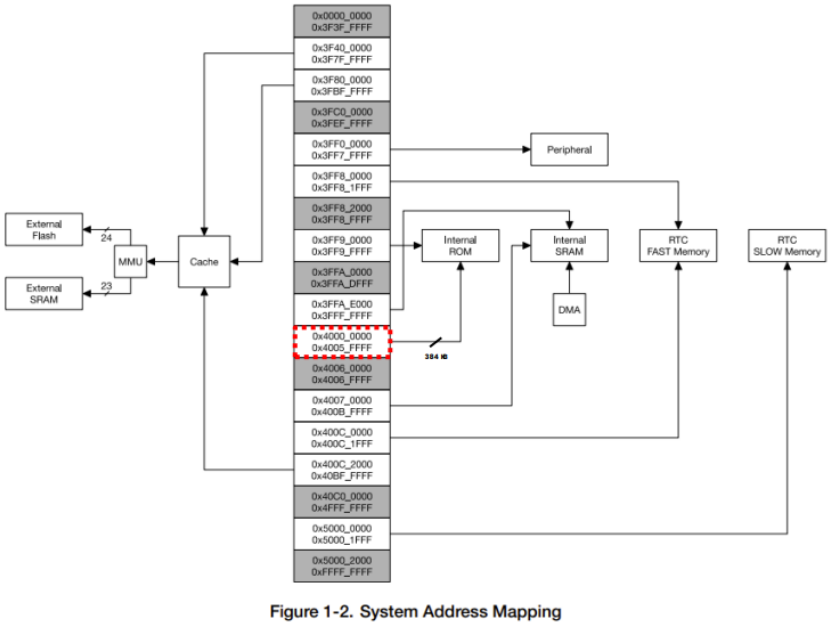
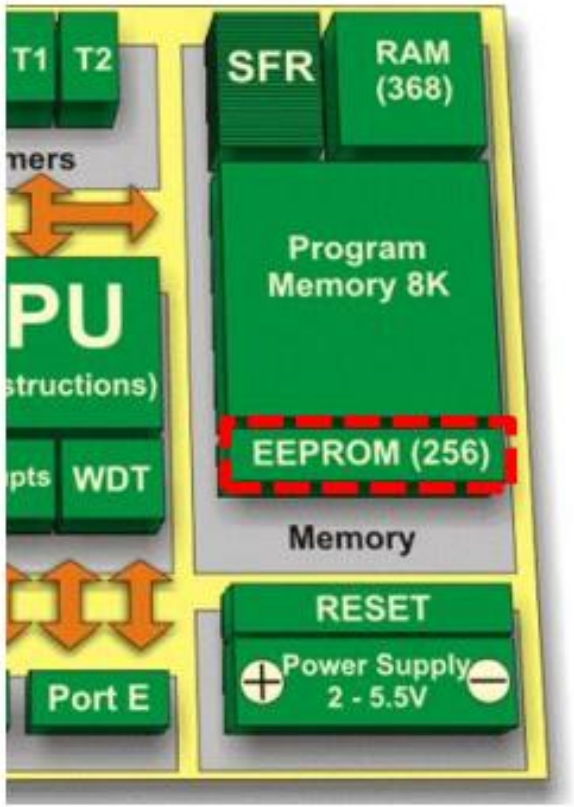


Figure 1-2. System Address Mapping

Components of a Microcontroller

03) EEPROM



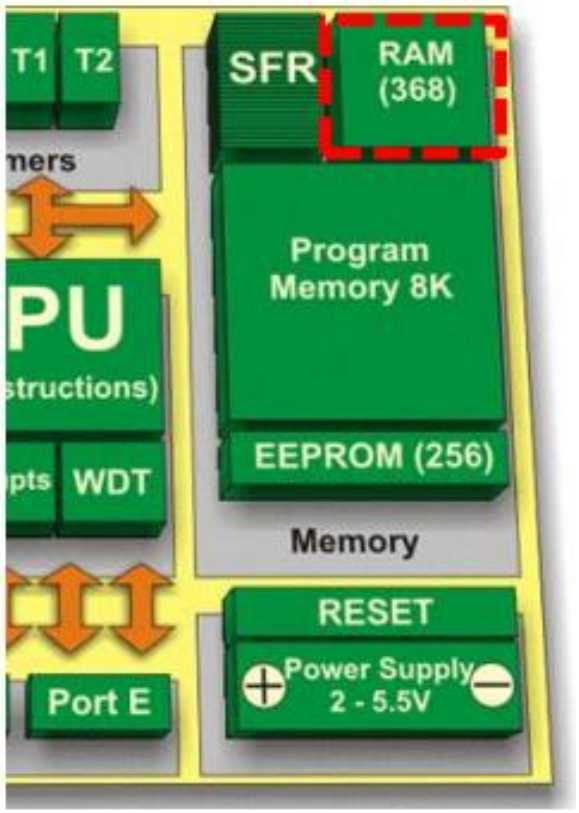
- **The content of an EEPROM can be changed during program execution (similar to RAM).**
- Content remains permanently saved even after power loss (similar to ROM).
- EEPROMs is often used to store values, created and used during operation.
  - Such as calibration values, codes, and counter values.
  - Generally, data that must be saved after turning off the power supply.

Disadvantage: the process of programming is relatively slow. EEPROM is not contained in all microcontrollers

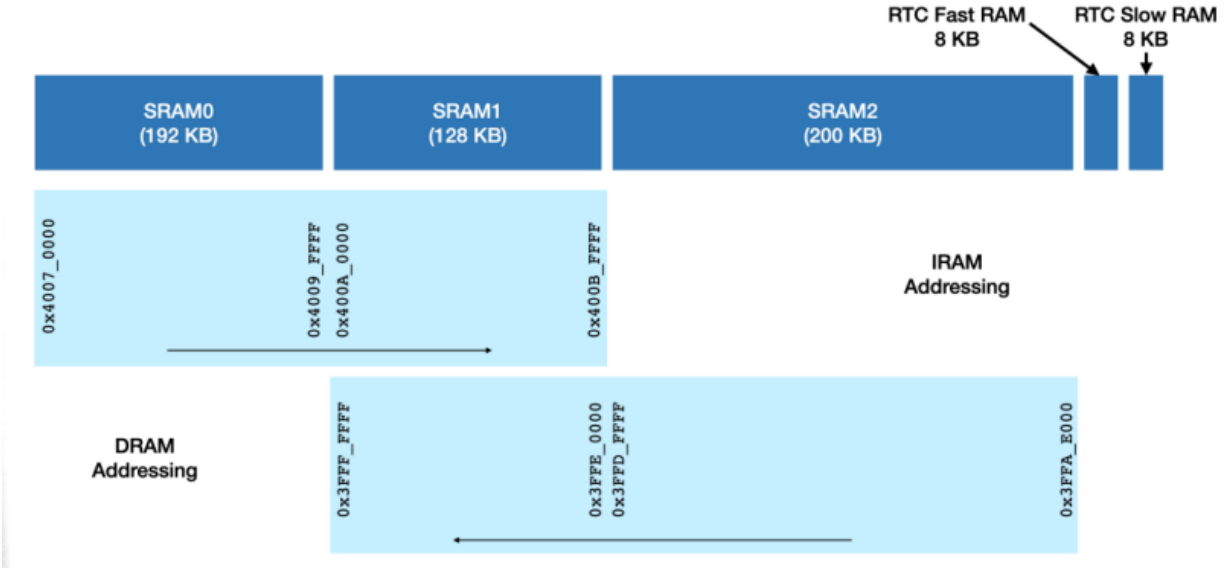
The ESP 32 contains 512 Byte of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles.

Components of a Microcontroller

04) RAM



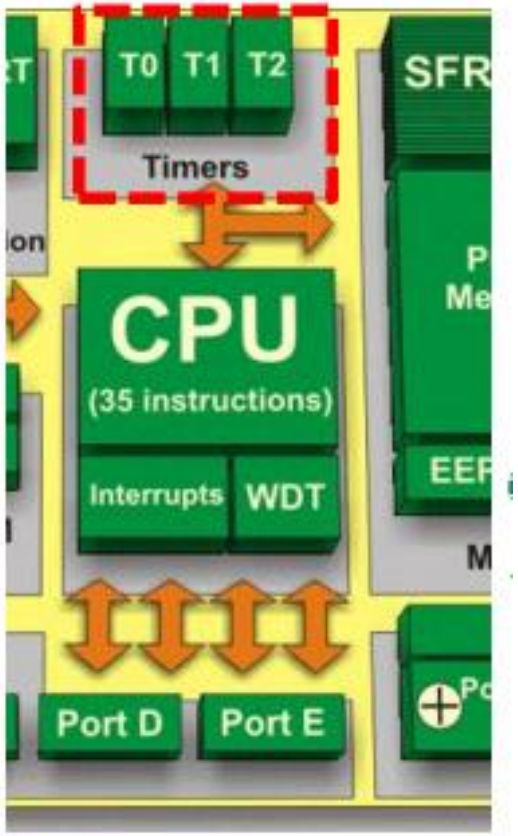
- RAM is used to **temporary store data.** This could be results created during execution of the program (variables, calculations, data inputs etc.)
- The content of the RAM is cleared at power off (working memory).
- The typical size of RAM integrated into the microcontroller is a few KBs



The 320 Kbytes of internal data SRAM in the ESP 32 is available, however, only 160 Kbytes can be accessed.

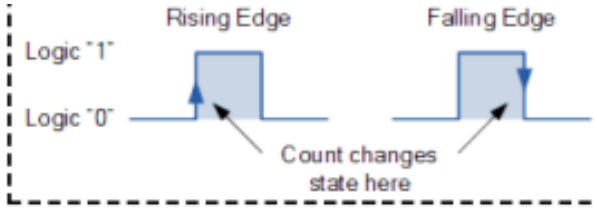
Components of a Microcontroller

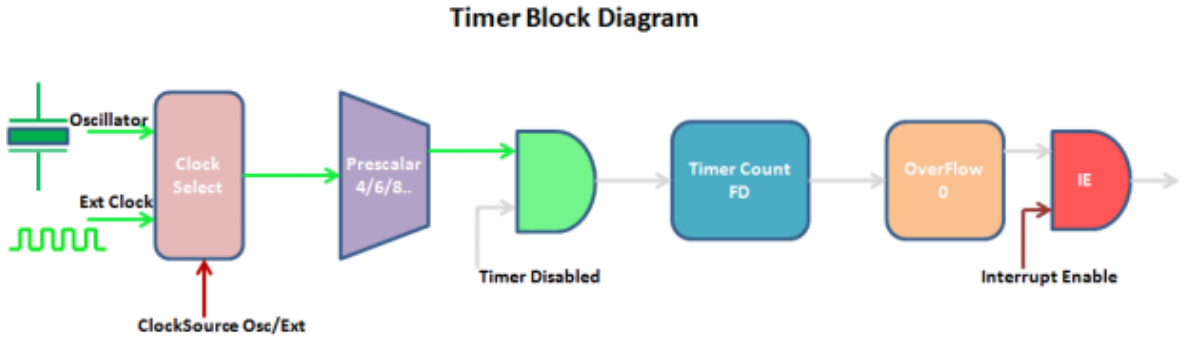
05) Timer

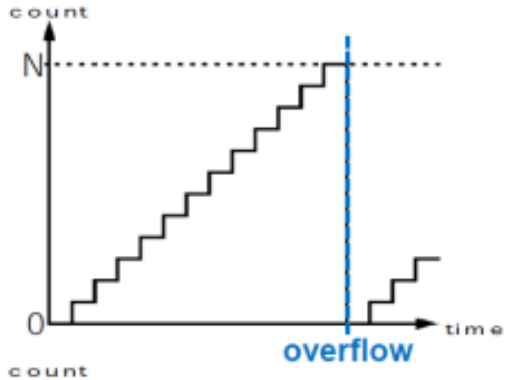


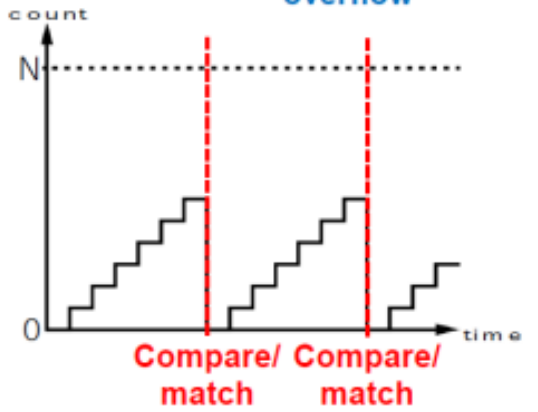
All microcontrollers have one or more built in hardware timers, designed to run independently of the CPU.

- All timers are fundamentally pulse counters.
- Sometimes referred to as a counter in the datasheet.
- The timer, counts changes at rising or falling edge









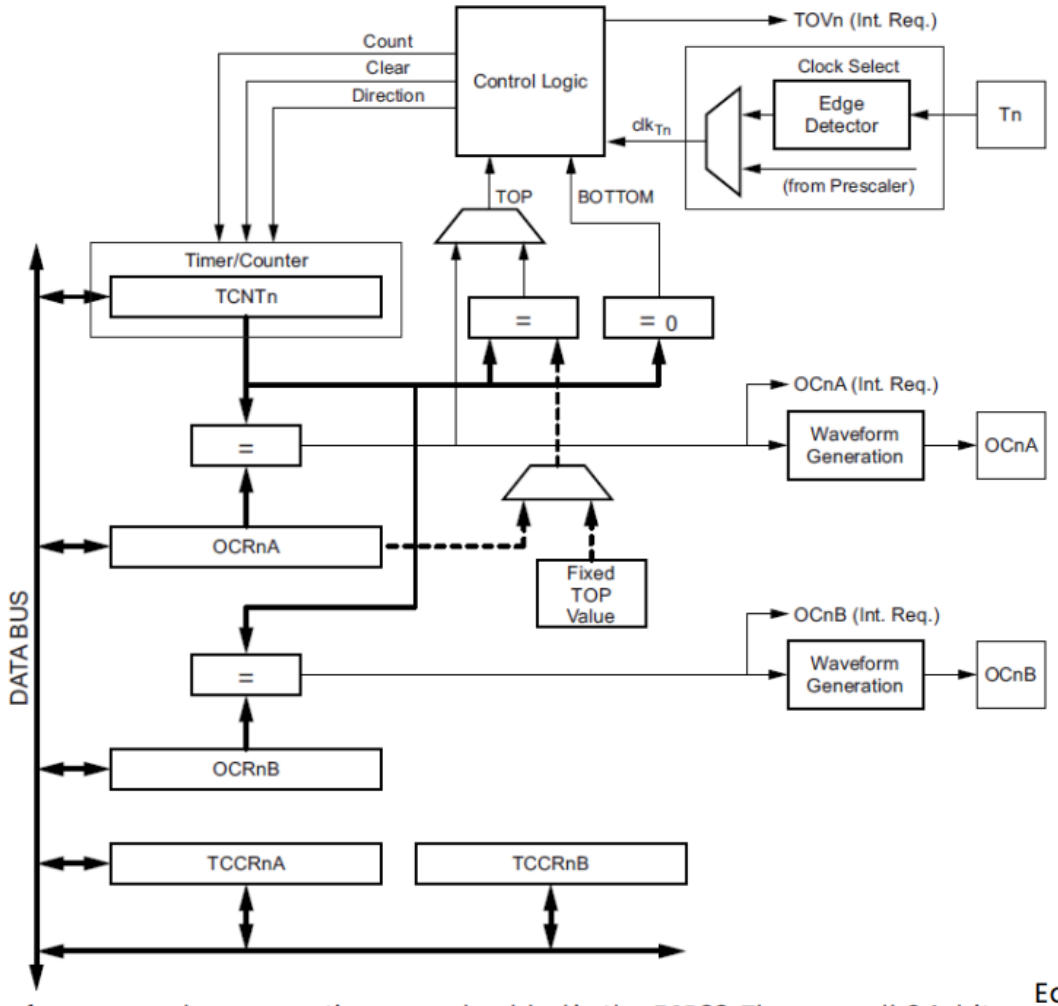
An Interrupt is given at overflow and/or when the counter value matches a predefined value (most suitable interrupt).

At the interrupt, relevant code can be executed.

Components of a Microcontroller

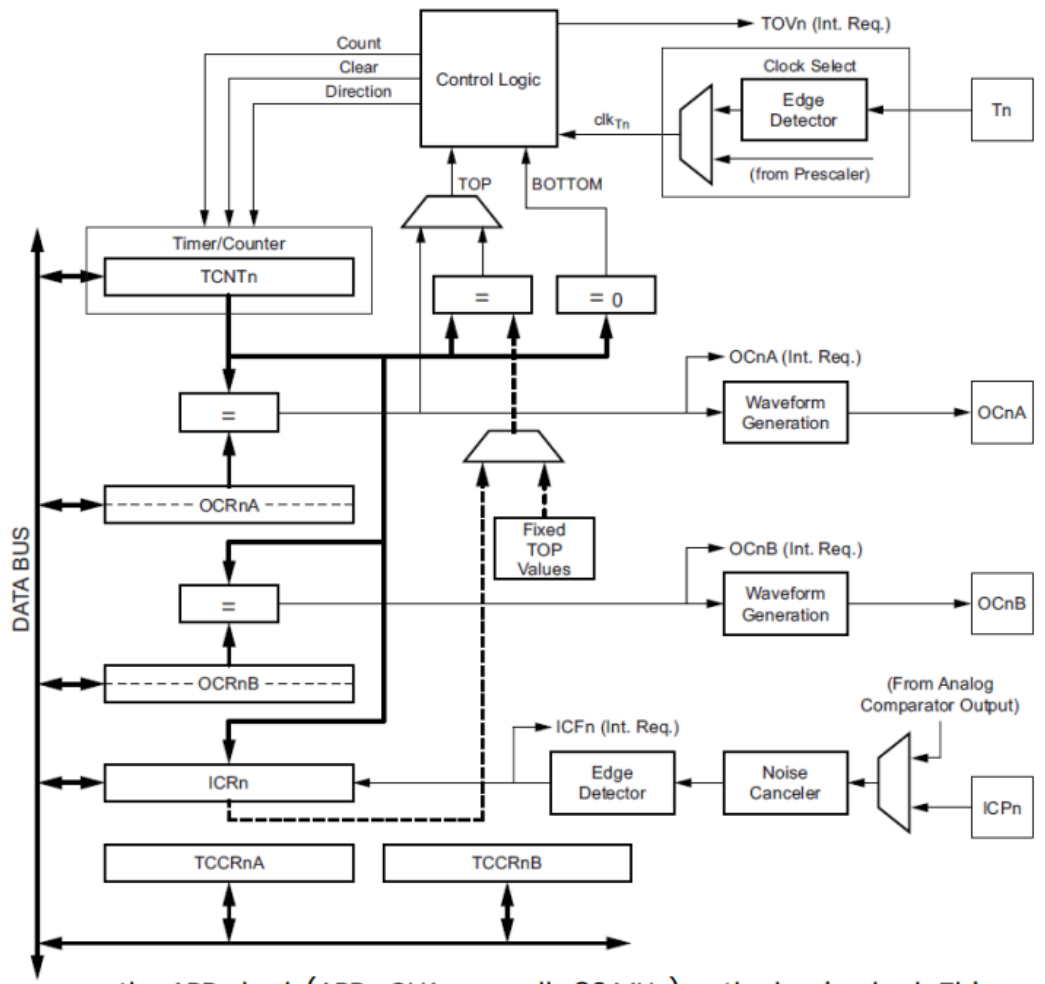
05) Timer

Figure 14-1. 8-bit Timer/Counter Block Diagram



There are four general-purpose timers embedded in the ESP32. They are all 64-bit generic timers based on 16-bit pre-scalers and 64-bit auto-reload-capable up/down counters.

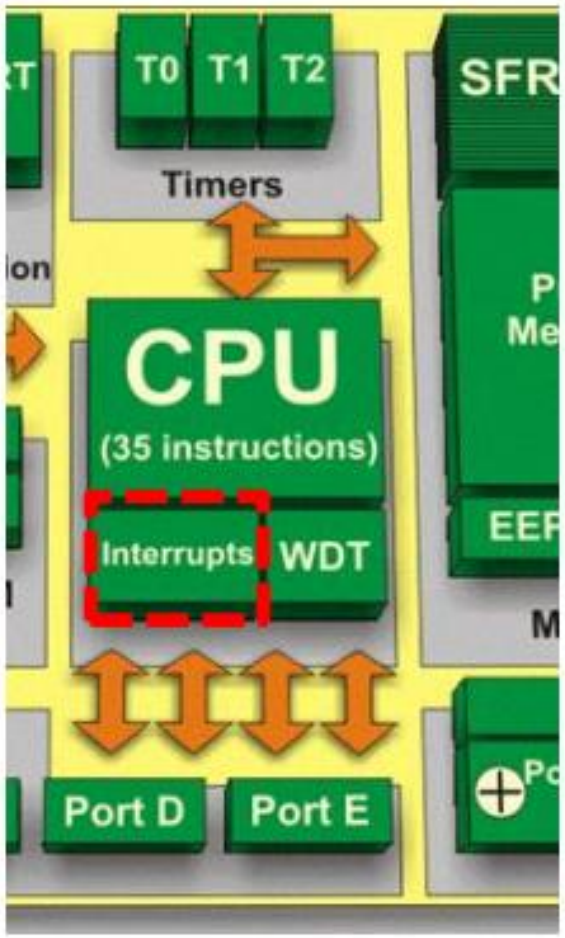
Figure 15-1. 16-bit Timer/Counter Block Diagram(1)



Each timer uses the APB clock (APB_CLK, normally 80 MHz) as the basic clock. This clock is then divided down by a 16-bit pre-scaler which generates the time-base counter clock (TB_clk).
The 64-bit time-base counter can be configured to count either up or down.

Components of a Microcontroller

05) Interrupt

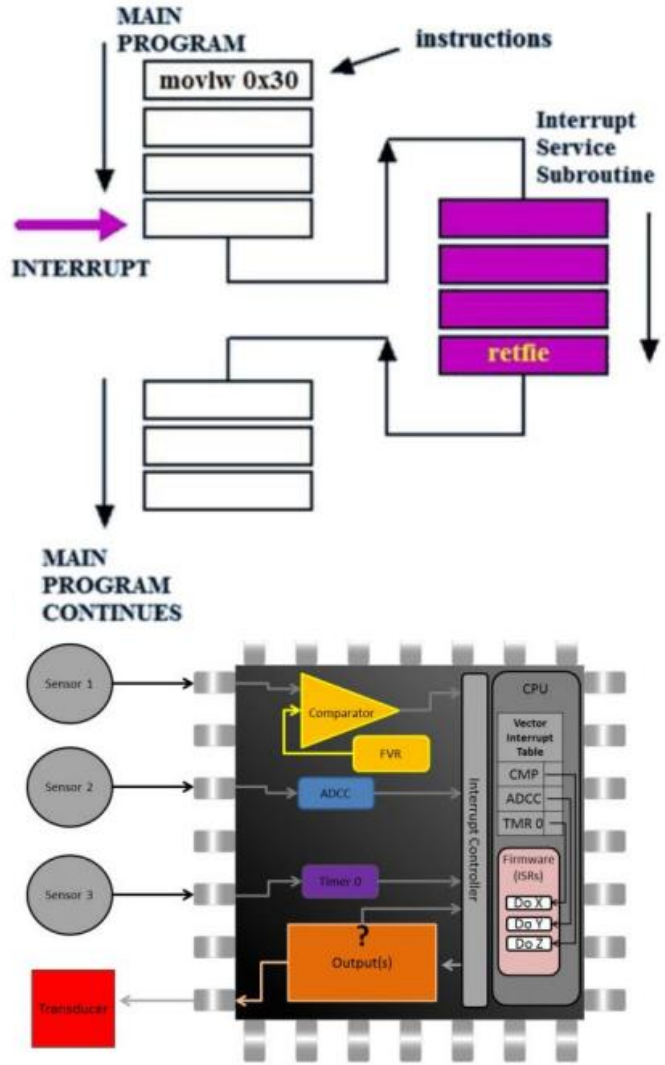


An Interrupt is an event that **temporarily suspends the main program,** passes the control to execution of the event related function, and resumes the main program where it had left off.

Procedure:

1. The main program is interrupted
2. The current program counter value is stored.
3. The program counter moves to a predefined interrupt position.
4. The code at the predefined position is executed
5. The program counter returns to execute the code at the stored value.

An interrupt can be caused by several events including, change of an input pin, counters, data ready.



Components of a Microcontroller

06) Digital I/O Ports

The digital input/outputs of a microcontroller can be **used to interface with other devices.**

- e.g., Button (input), LED (output), motor controller (output).
- Often the digital input/outputs are referred to as GPIOs (general purpose input/output)

Input or output functions can be configured.

- For inputs, **pull up** or **pull down** can usually be configured.

Usually, all input/outputs of the microcontroller can be configured as **digital ports.**

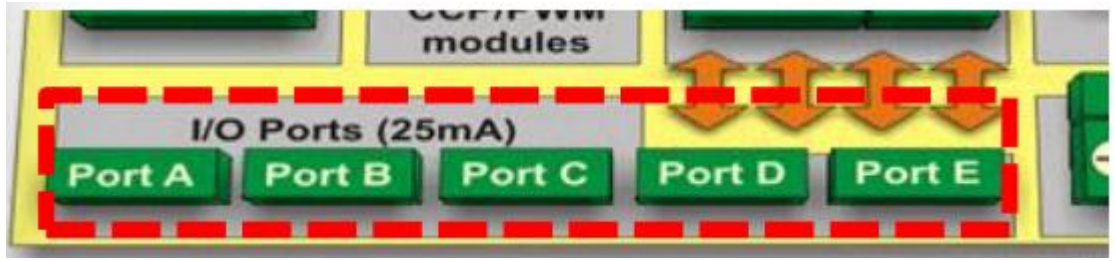


Figure 4-2 shows the logic for input selection via GPIO Matrix.

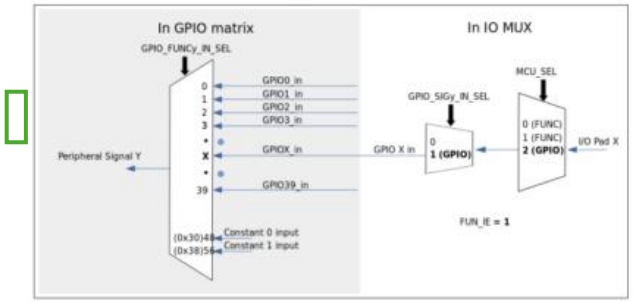


Figure 4-2. Peripheral Input via IO_MUX, GPIO Matrix

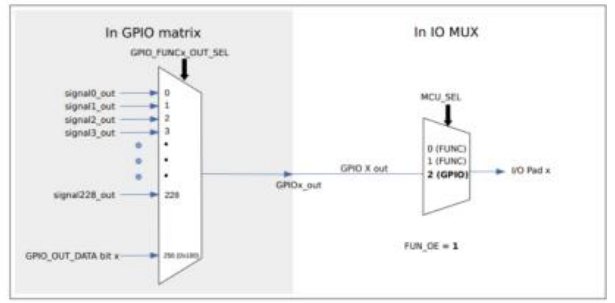
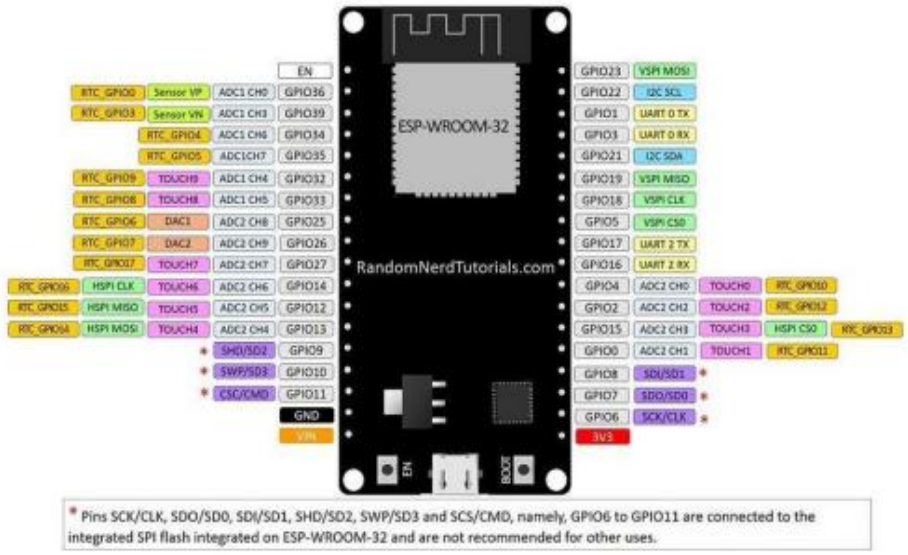


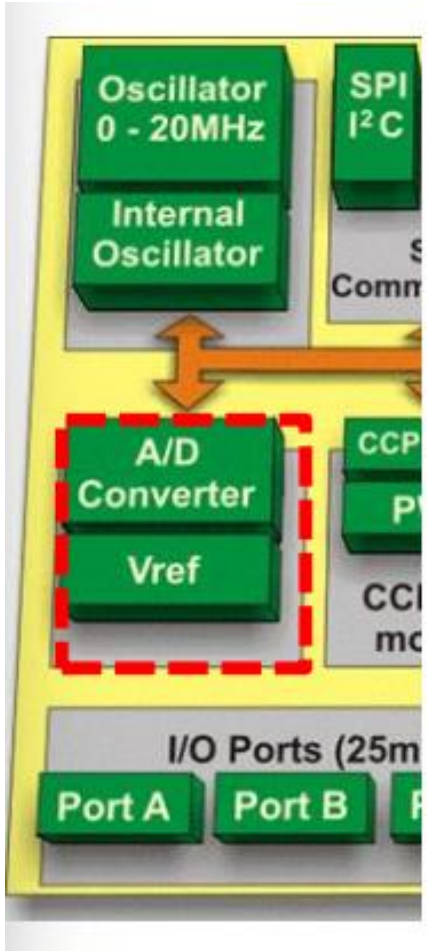
Figure 4-3. Output via GPIO Matrix



The GPIOs are typically organized in groups (sometimes referred to as **PORTs**).

Components of a Microcontroller

07) ADC and DAC

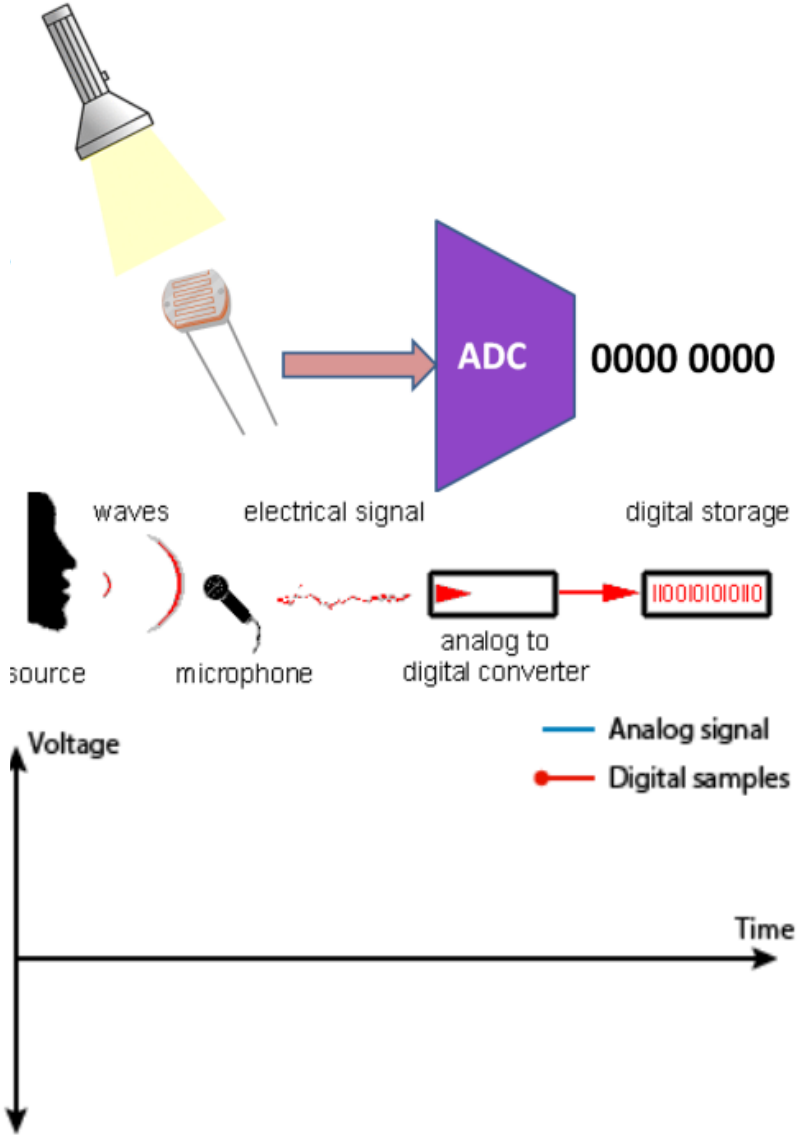


Analog to digital converters (ADC)

Often integrated into the microcontroller.

- Typical ADC resolutions are 8 to 16 bits.
- Usually, specific inputs must be used for the ADC

> The ESP 32 features **two 12-bit** (configurable) successive approximation **SAR ADC and supports up to 18 analog input channels**.
>
> The ADC-1 is connected to an 8-channel analog multiplexer which allows eight single-ended voltage from GPIOs 32 – 39. Similarly, ADC-2 is connected to a 10-channel analog multiplexer. The single-ended voltage inputs refer to 0V (GND).
>
> The ADC contains a sample and hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion.

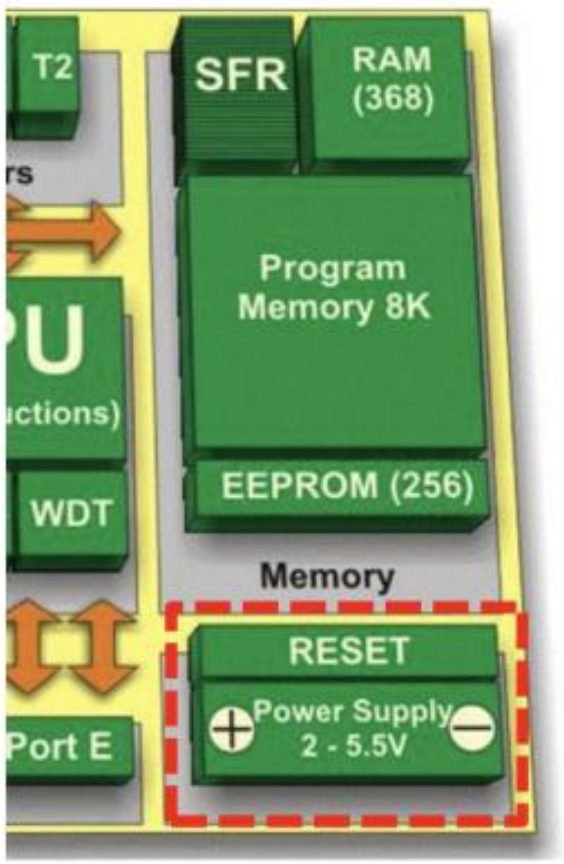


Digital-to-analog converters (DAC)

Converts a digital value (e.g., variable in your code) to an analog output. More rarely integrated into the microcontroller.

Components of a Microcontroller

08) Power Supply



5.3 DC Characteristics (3.3 V, 25 °C)

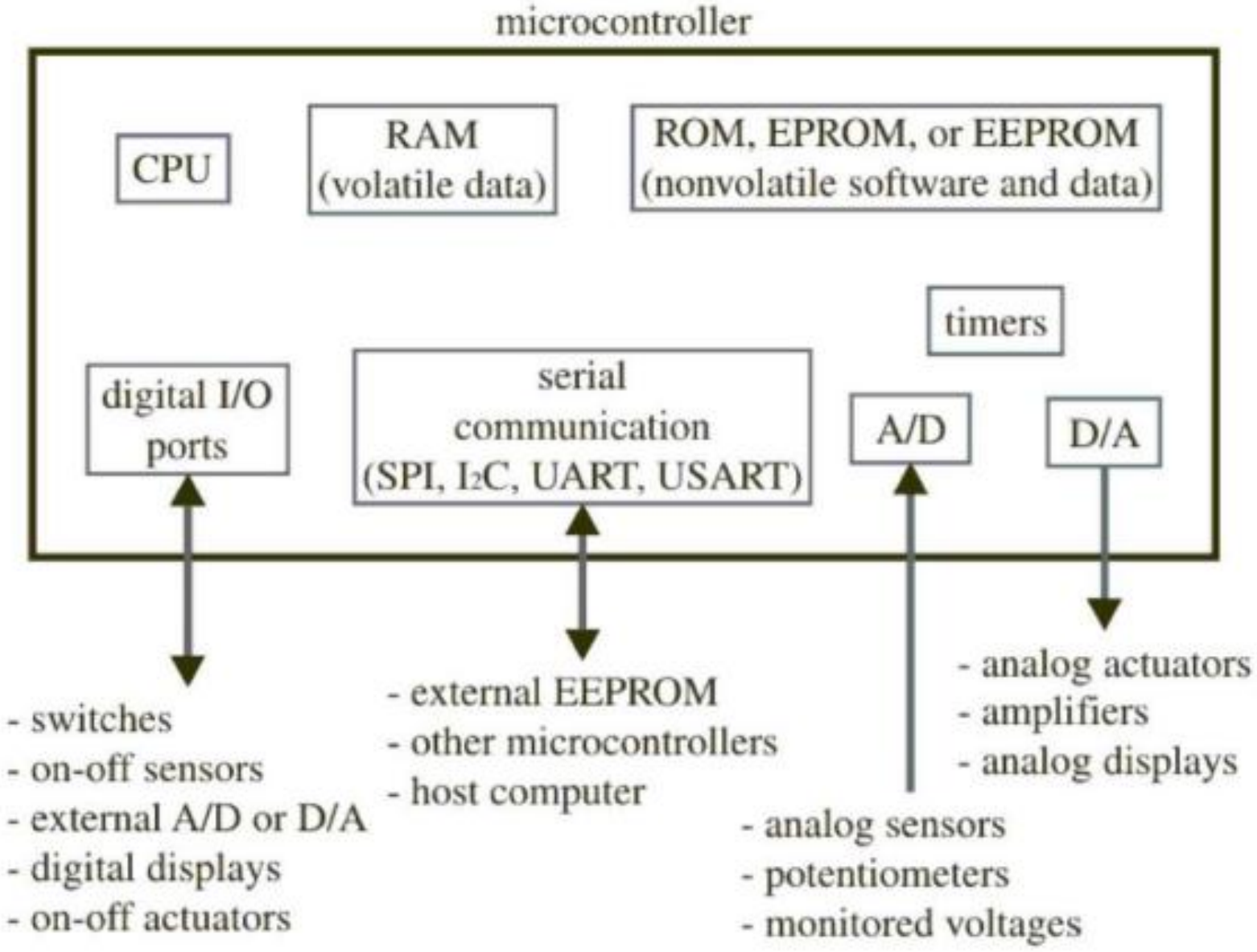
Table 15: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)		-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor		-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor		-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip		-	-	0.6	V

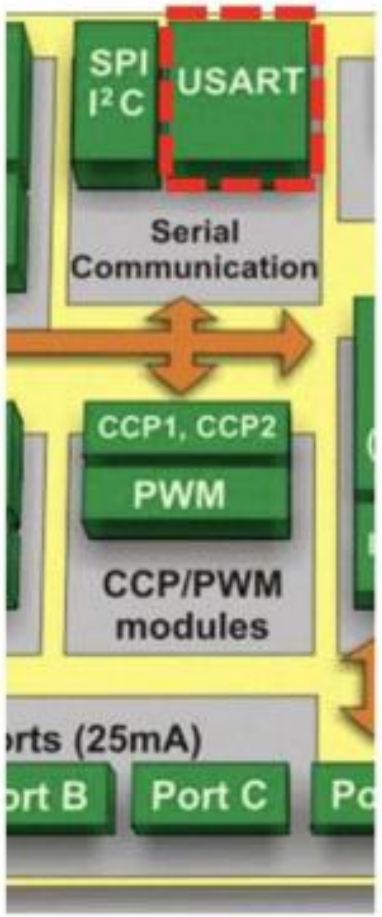
Notes:

1. Please see Table IO_MUX for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. For VDD_SDIO power domain, per-pin current sourced in the same domain is gradually reduced from around 30 mA to around 10 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.

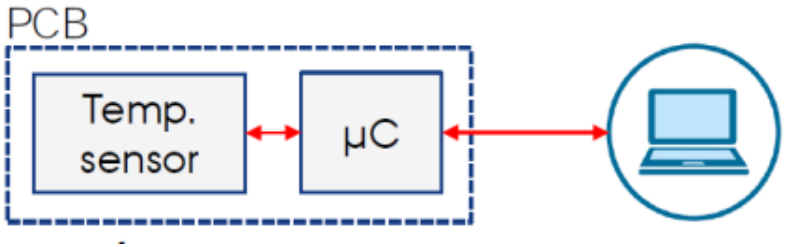
Components of a Microcontroller



USART Serial Communication

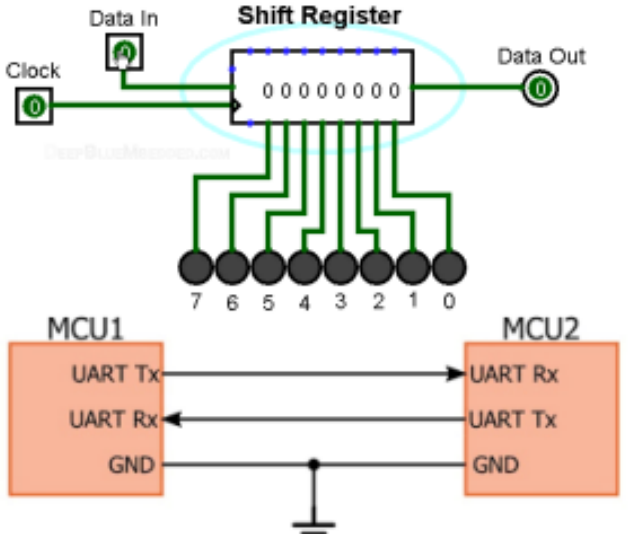


Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel (wire).



USART (Universal Synchronous and Asynchronous Receiver Transmitter) is used for on board (PCB) communication

- Often the UART is converted to a USB signal (USART will be handled as a virtual COM port by the connected device).
- The UART signal can be amplified to a RS232 signal, which can be used for communication with external devices.



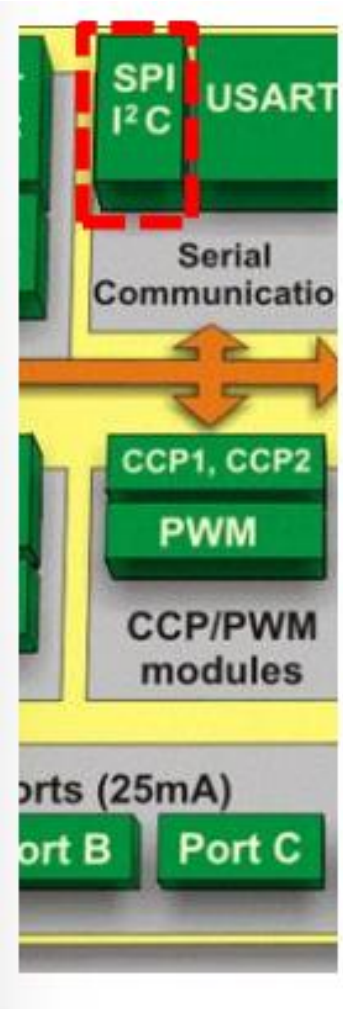
The data bits are transmitted with a **predefined baud rate (speed)** for Rx and Tx.

- The data is not synchronized with the clock signal of the microcontroller.

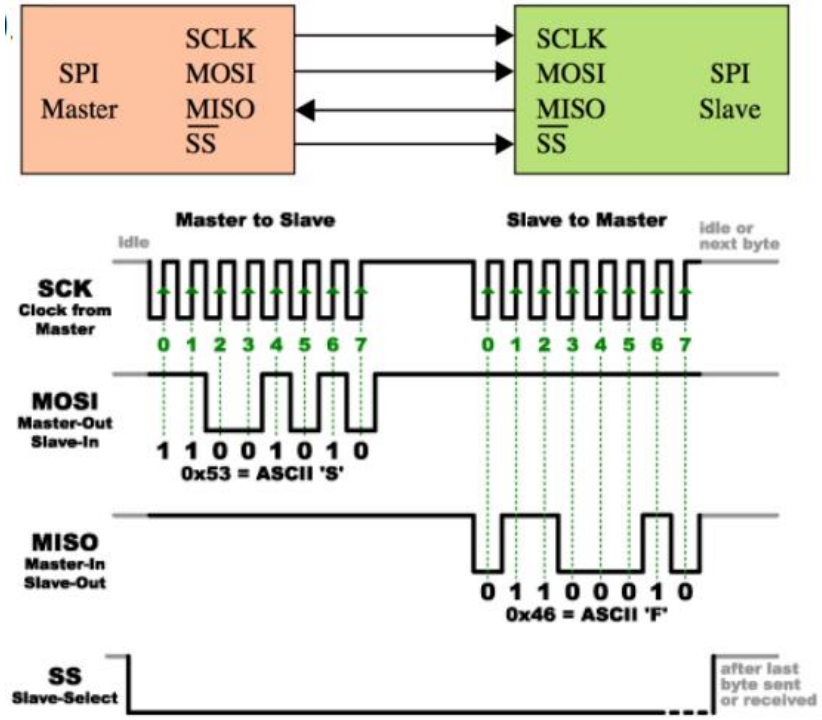

UART to USB conversion


RS232

SPI Serial Communication



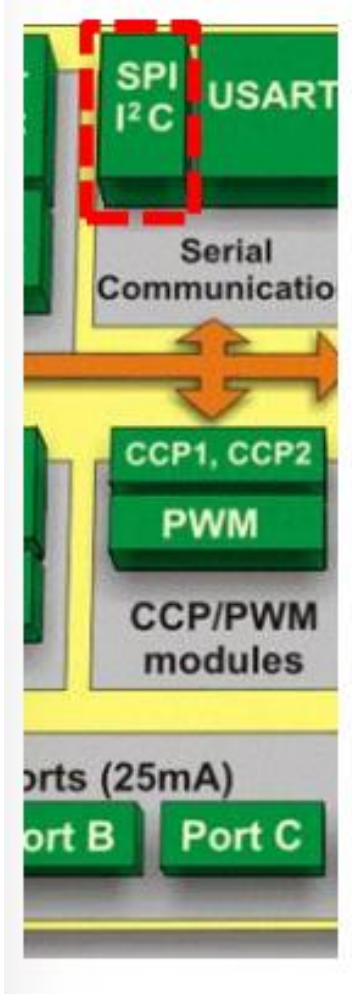
- Serial Peripheral Interface (Introduced by Motorola).
- Well suited for fast data rates (data streaming).
- Primitive (no error checks).
- Possible to address multiple slaves with only 1 bus.
- Commonly used inter component bus.
- Normally only one master (usually the µC).
- Synchronous : separate lines for data and clock.
- The data is sampled at rising or falling edge.
- No overhead and no stop/start bits.
- Data speed defined by clock.



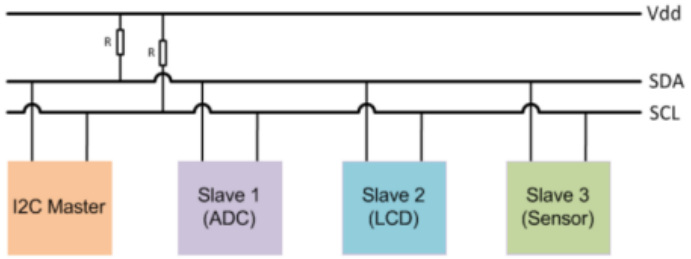
Connection to a SPI device requires 4 wires.

- **SCLK (CLK):** Clock signal.
- **Master Out Slave In (MOSI):** Transmits data from the master to a slave device.
- **Master In Slave Out (MISO):** Transmits data from a slave to the master device.
- **Slave Select (SS):** Tells the slave that it should wake up and receive / send data.
  - Sometimes referred to as chip select (CS).
  - When only one slave device, the SS is sometimes not needed (depends on the protocol).

I2C Serial Communication



- Inter IC bus (Introduced by Philips).
- Suited for medium data rates for short-distance, intra-board communication.
- Only requires two wires **(clock – SCL / data – SDA).** Inbuild addressing (minor HW demands).
- Each device (IC) model has a unique address.



E.g., used for audio streaming in MP3 players, displays, ADCs, and sensors.

Advantages

- Only uses two wires.
- Supports multiple masters and multiple slaves.
- ACK/NACK bit gives confirmation that each frame is transferred successfully.
- Hardware is less complicated than with UARTs.
- Well known and widely used protocol.

Disadvantages

- Slower data transfer rate than SPI
- More complicated hardware needed to implement than SPI

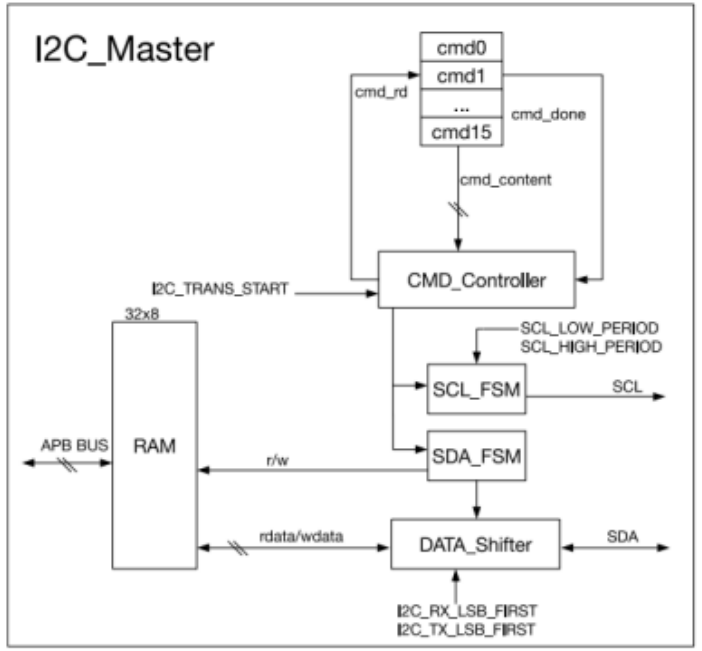


Figure 11-1. I2C Master Architecture

SPI/I2C Serial Communication

Feature	SPI	I2C
Number of Wires	4 Wires: MOSI, MISO, SCLK, SS (Chip Select)	2 Wires: SDA (Data), SCL (Clock)
Master-Slave Architecture	Supports multiple masters and multiple slaves	Supports multiple masters and multiple slaves
Clock Speed	Higher speed, typically up to 10+ MHz	Lower speed, typically 100 kHz, 400 kHz, and up to 3.4 MHz (High-Speed mode)
Communication Type	Full-Duplex (data can be transmitted and received simultaneously)	Half-Duplex (data transmitted in one direction at a time)
Power Consumption	Generally higher due to continuous clocking	Generally lower, especially in standby mode
Data Transmission	Faster, more suited for high-speed data transfer	Slower, more suited for control or sensor data
Configuration	Manual configuration for multiple devices	Easier configuration with address-based communication

Types of Microcontrollers

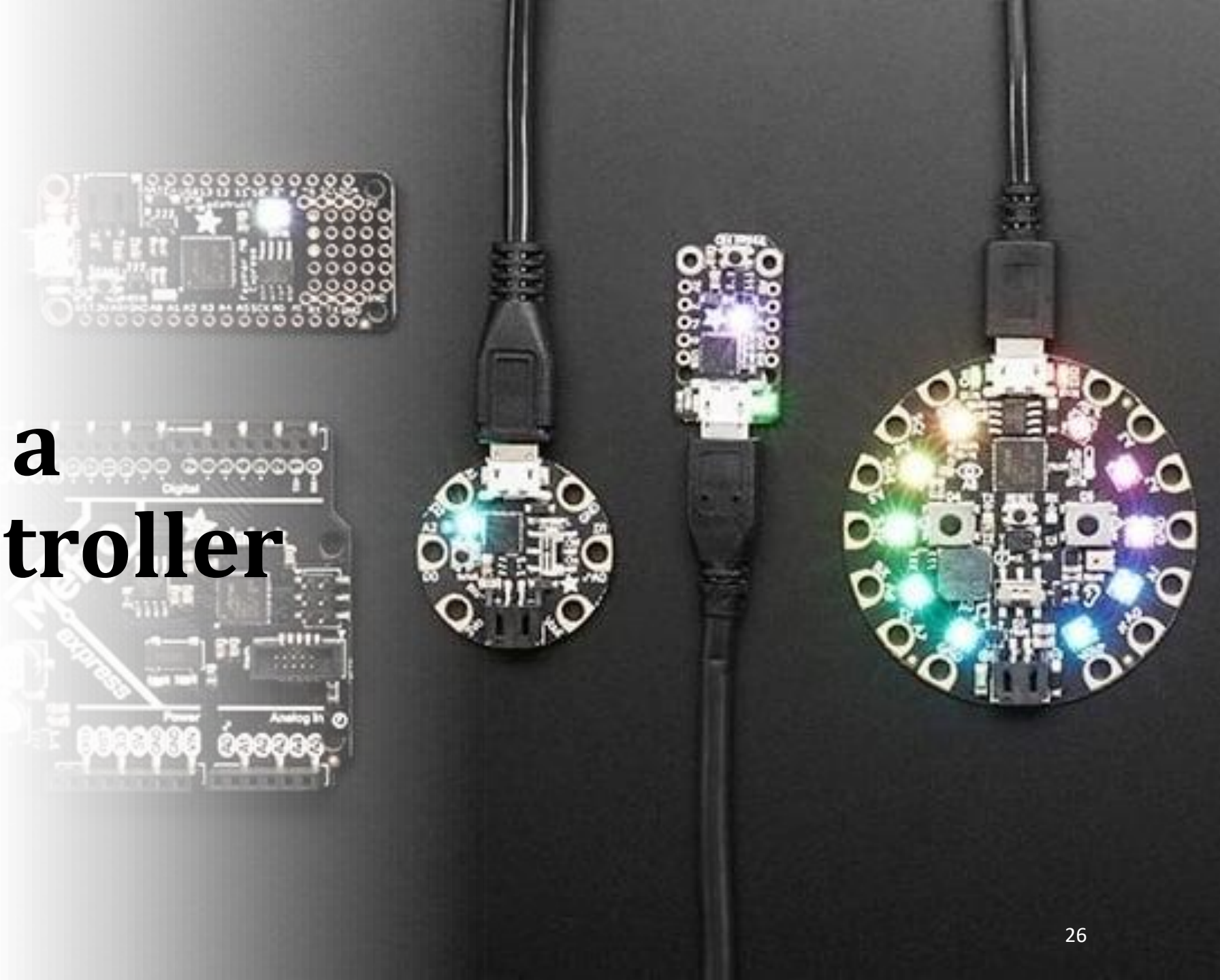
- Three most popular types of microcontrollers are:

PIC Microcontrollers
(OUTDATED)

AVR Microcontrollers

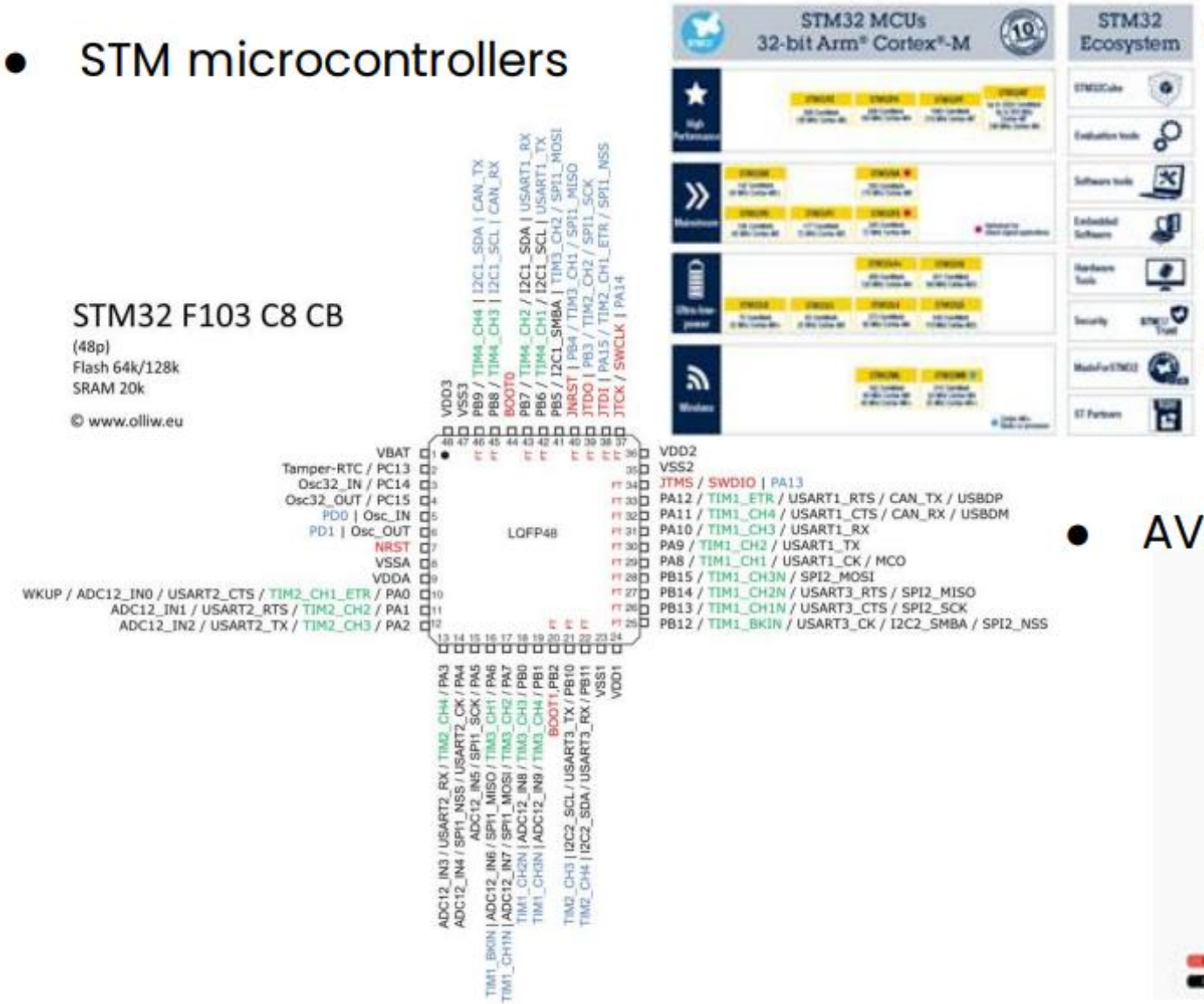
STM Microcontrollers

Selecting a Microcontroller



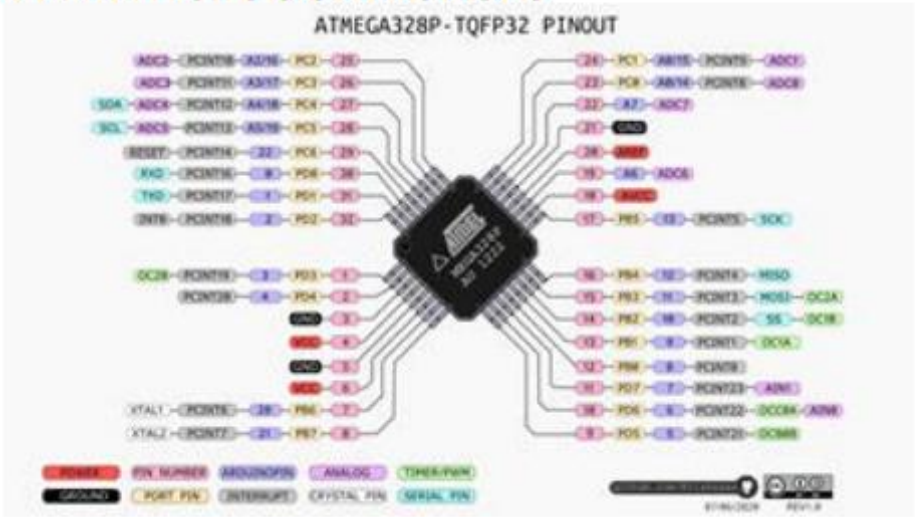
Selecting a Microcontroller

- STM microcontrollers

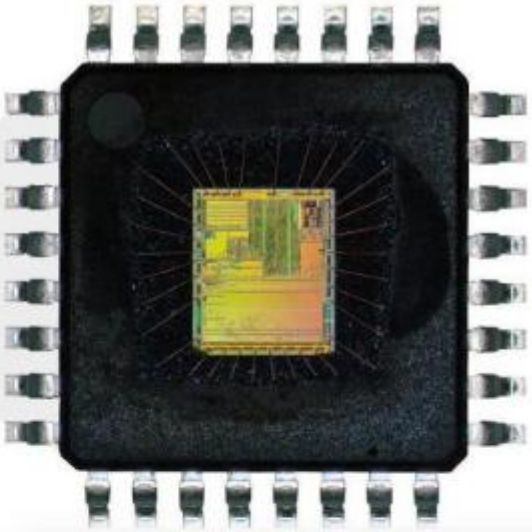


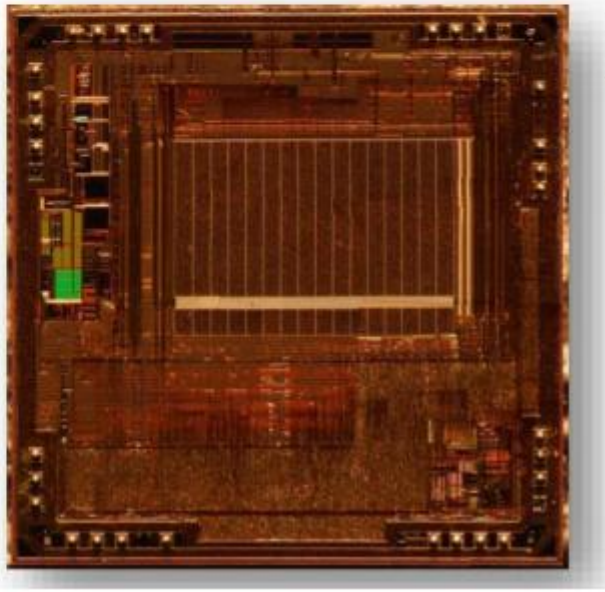
- There are many different types of µC, ranging from different factors. But, different µC's have many things in common.
- Microcontrollers are programmed in high-level languages (C, C++).
- If you can learn to handle one type of µC, then you can handle most of them.

- AVR microcontrollers

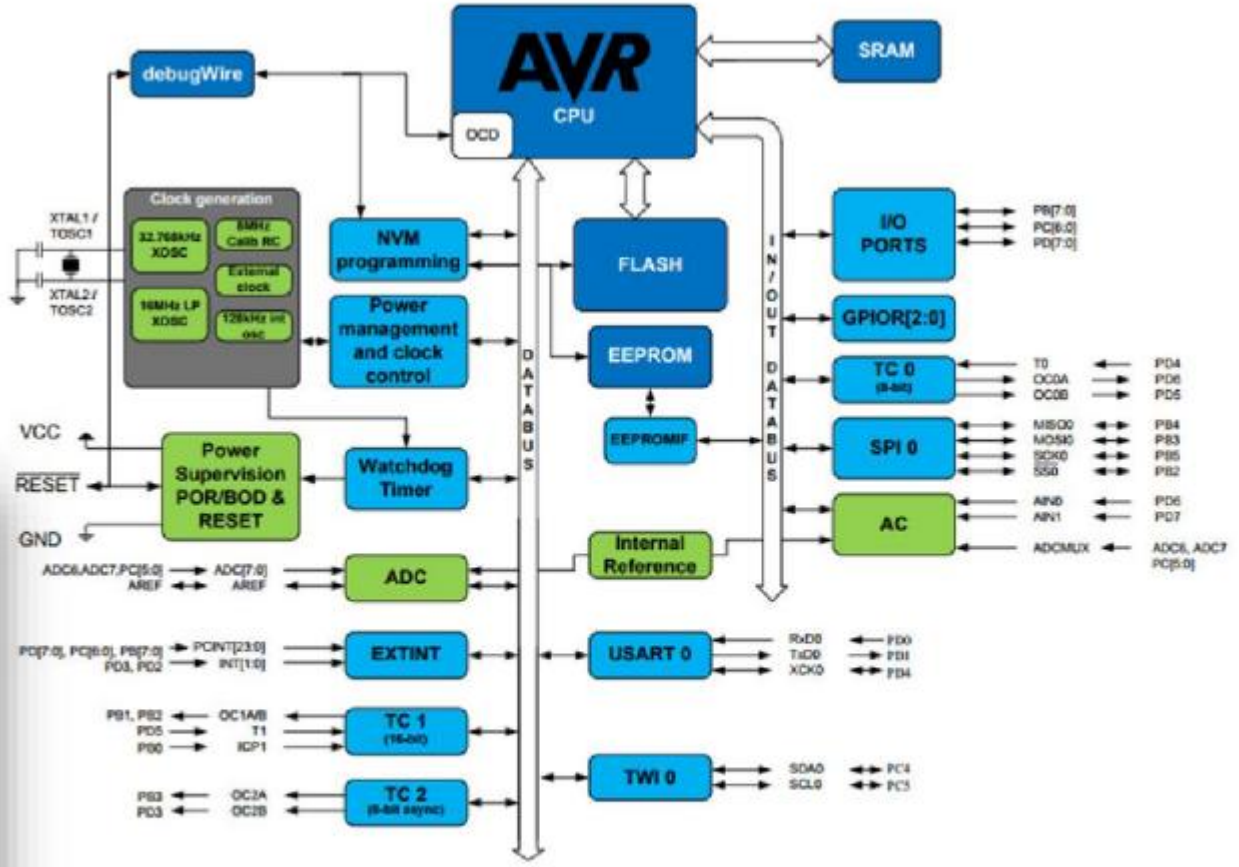


ATMEGA328P Microcontrollers





- Easy to learn
- Huge community
- Resource availability



Harward Architecture
(several busses)