ELTE | FACULTY OF INFORMATICS

Embodied Intelligence – L05
Control units

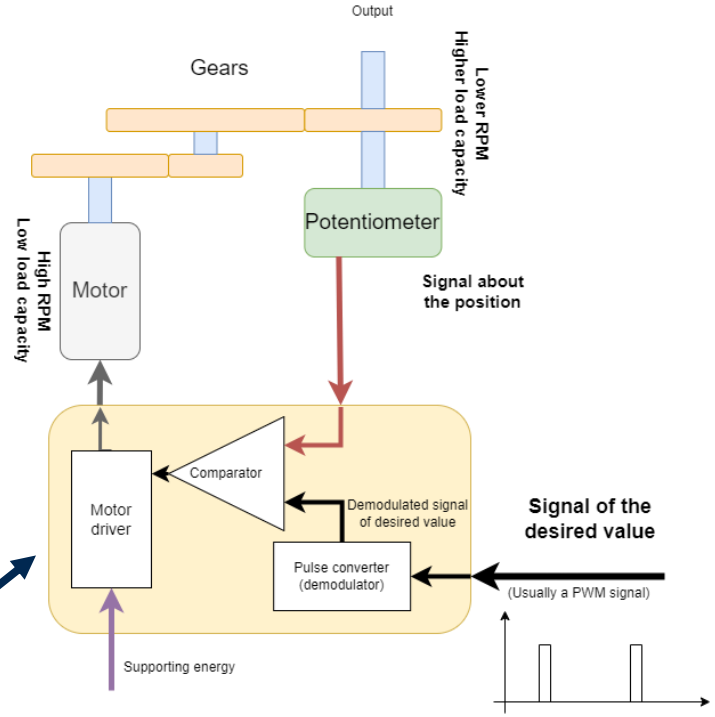
by Márk Domonkos

ELTE | IK
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Introduction

What is control?

- „to order, limit, or rule something, or someone's actions or behaviour” – cambridge dict.
- The control we will learn about is similar
- We need for control:
  - A process
  - An actuation
  - A contol unit
  - (optionally) Sensing
  - Desired outcome

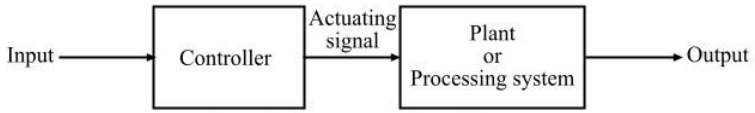


Remember this?

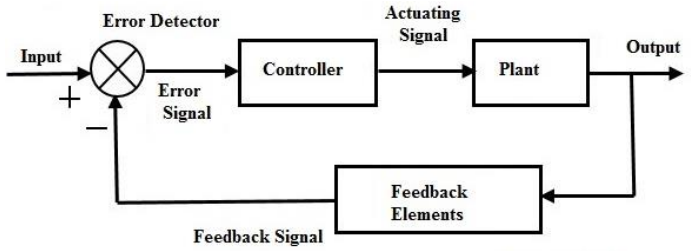
Closed and Open loops

- Based on that the proces has an active feedback on the control itself we distinguish:
- Open loop control
- Closed loop control

We have no sensors attached to the control in the system.
e.g.: traditional toasters (if the slice is thicker it will probably burn)

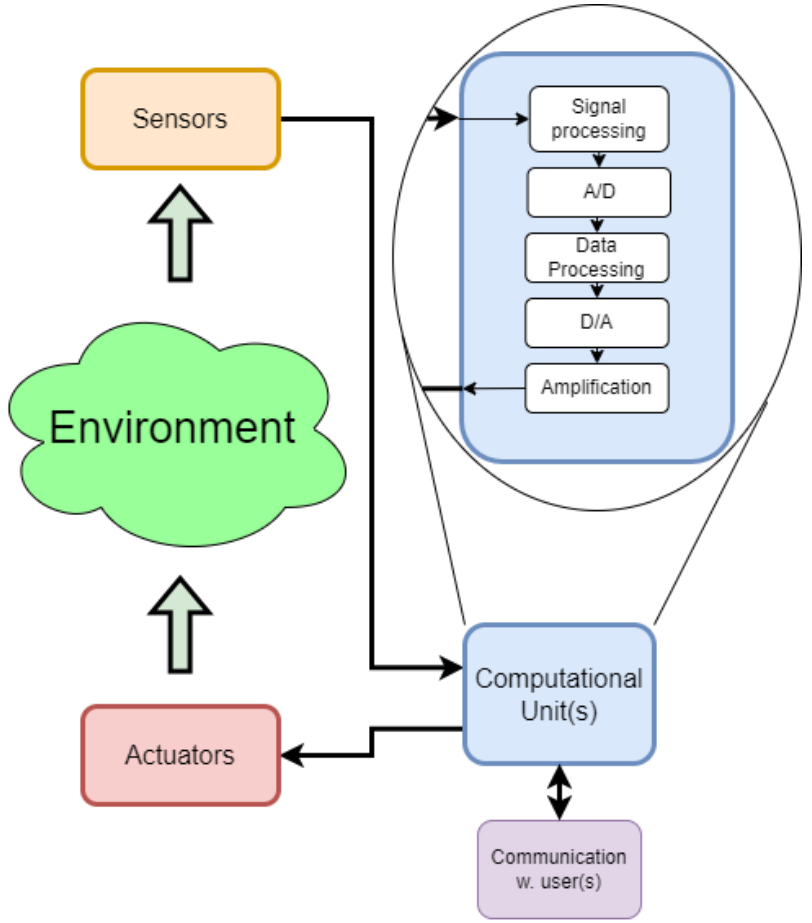


We constantly have a feedback from a sensor in the control loop.
e.g.: control of a drone (within a limit, the small wind will not affect the drones operation)



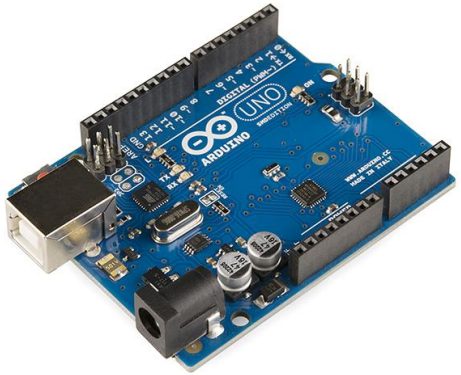
We saw already that some modern sensors/ actuators have built in controllers as well

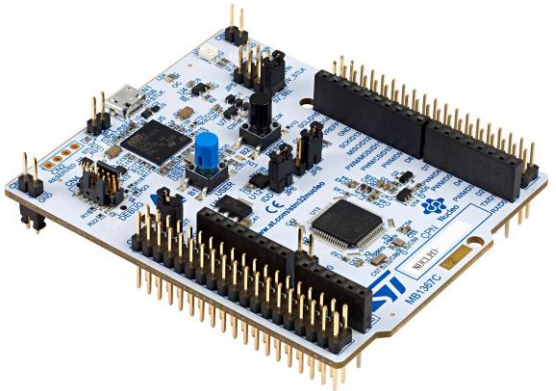
- Some have built in:
  - Signal preprocessing
  - A/D conversion
  - Communication protocols

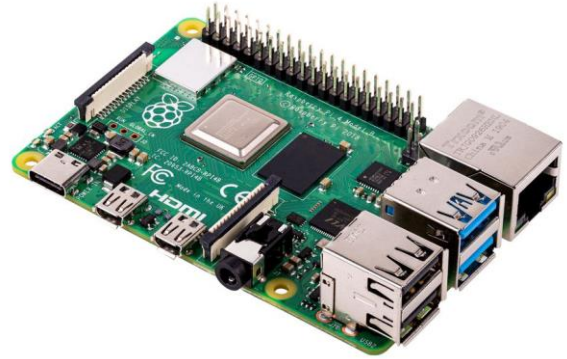


Implementation of the functionality

- Controller units
  - Make calculations based on the inputs
  - Decision making
  - Initiate events based on the calculations
  - Timing of those events
  - Controll the events if needed
  - Communicate with other devices / user







An argument against Arduino

Arduino

STM32

Disclaimer: I had found similar picture as a meme on a social media workgoup, but I could not find the original.

From an another perspecivet

What can we say about the current status?

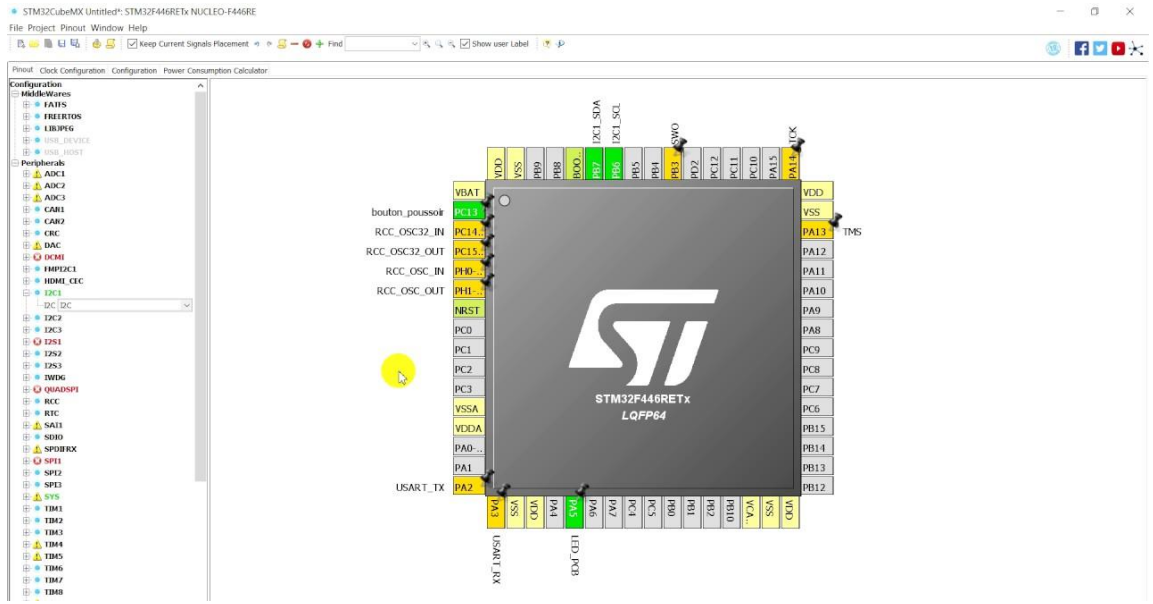
- Microcontrollers are (usually):
  - Cheap
  - Flexible controller solutions (easy reprogramming if new/other functionalities are needed)
  - Hundreds of configurations (with different peripherals)

Microprocessor or Microcontroller

- Usually in everyday conversations the two words are used as synonims
- BUT they are acutally not
- **Microprocessor:** A processor
- **Microcontroller:** A processor + pheripherals (I/O, A/D converters, communication, memory, timers, D/A converters etc.)

How to select one exact model?

- Depends on the application:
  - What performance is needed?
  - Which kind of peripherals do the task needs?
  - How many from specific peripherals are needed during the task?
  - What is the specification of those peripherals?



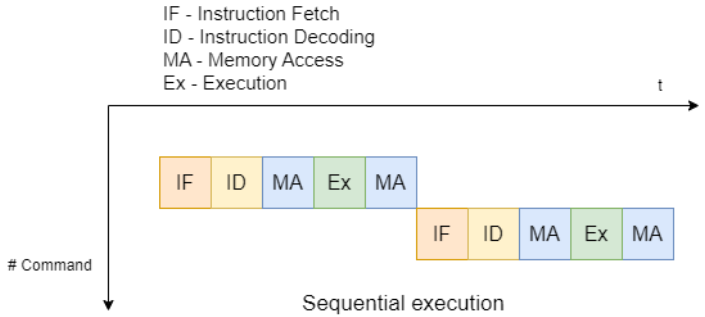
Usual peripherals

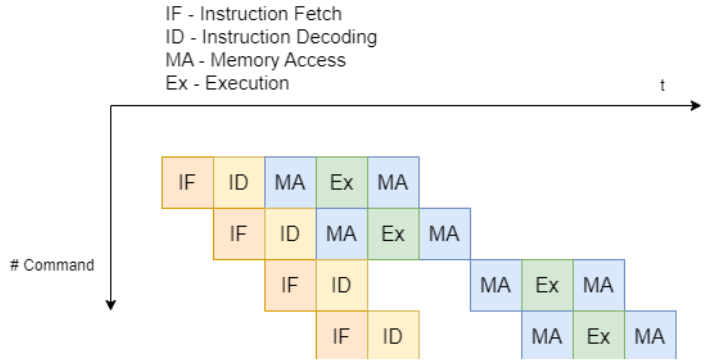
- Input / Output pins (I/O)
  - Digital / Analog / Interrupt
- Timers
- Analog Digital converters and Digital Analog converters
- Comparators
- Communication protocols
  - UART / I2C / SPI / MODBUS etc.

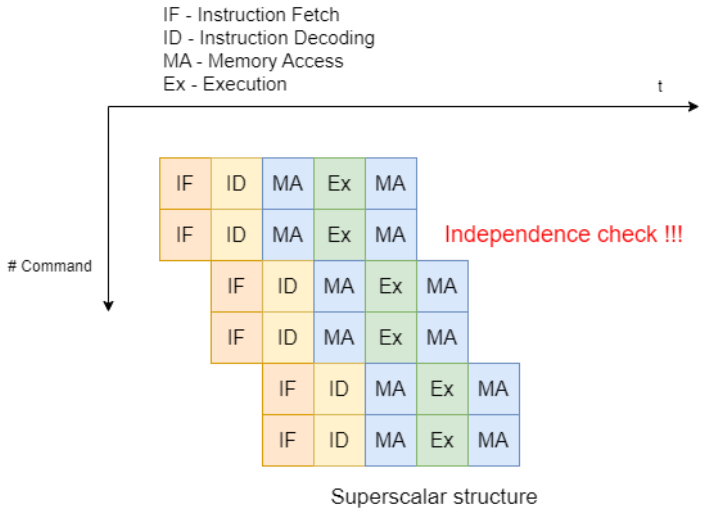
CPU – Central Processing Unit

- Program counter – counts during the run
- Instruction fetcher & decoder
- Data memory
- Program memory
- Registers
- ALU – satisfies commands

Operation of the CPU







Moving towards Microcontrollers

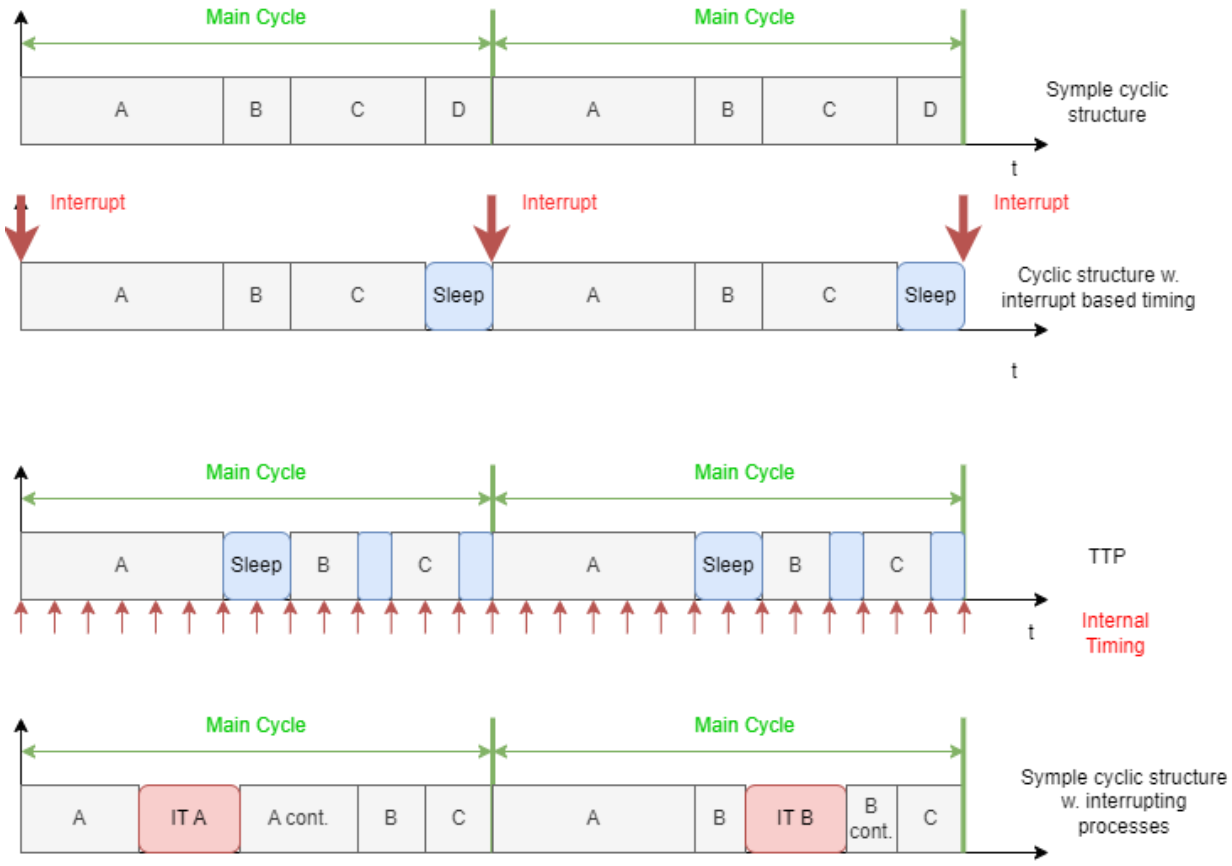
- A microcontroller is basically a microprocessor with peripherals integrated for complete standalone operational capacity (ADC, timers, etc.).

Real-timeness

- A real-time system means that the system is subjected to real-time, i.e., its response should be guaranteed within a specified timing constraint or the system should meet a specified deadline.

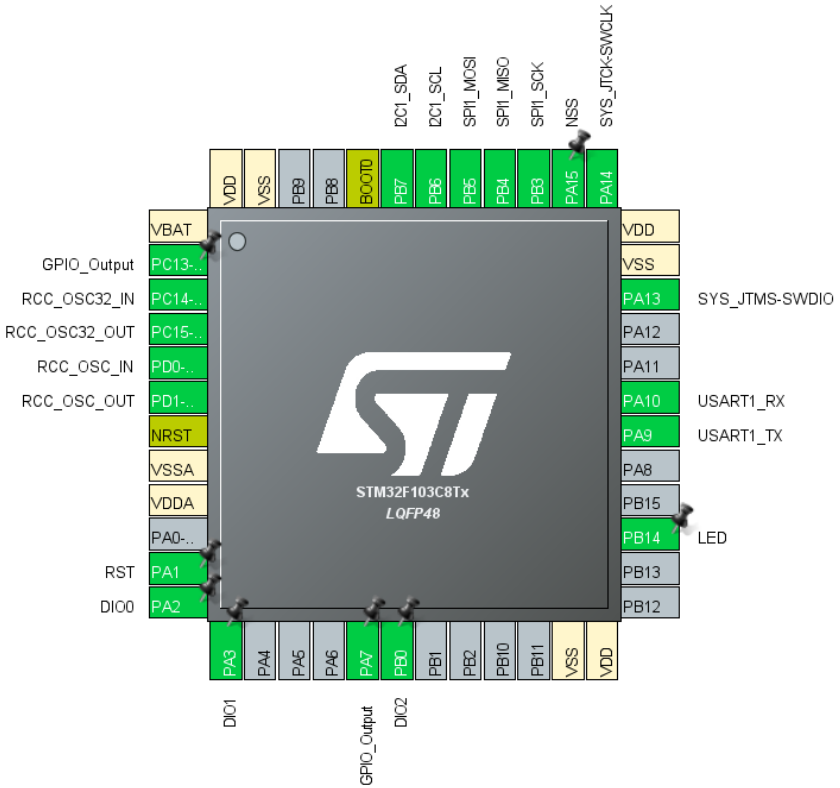
Some program organisation methods

- Cyclical structure
- Time Triggered Protocol
- Earliest deadline first timing
- Least Laxity First timing



Input/Output pins

- Input and Output – usually on a physical pin
- Can be analog or digital
- Can initiate interrput based on external events
- Can be used with other peripherals
- Usually organised in PORTs



Case Study 1 – Input Multiplexing

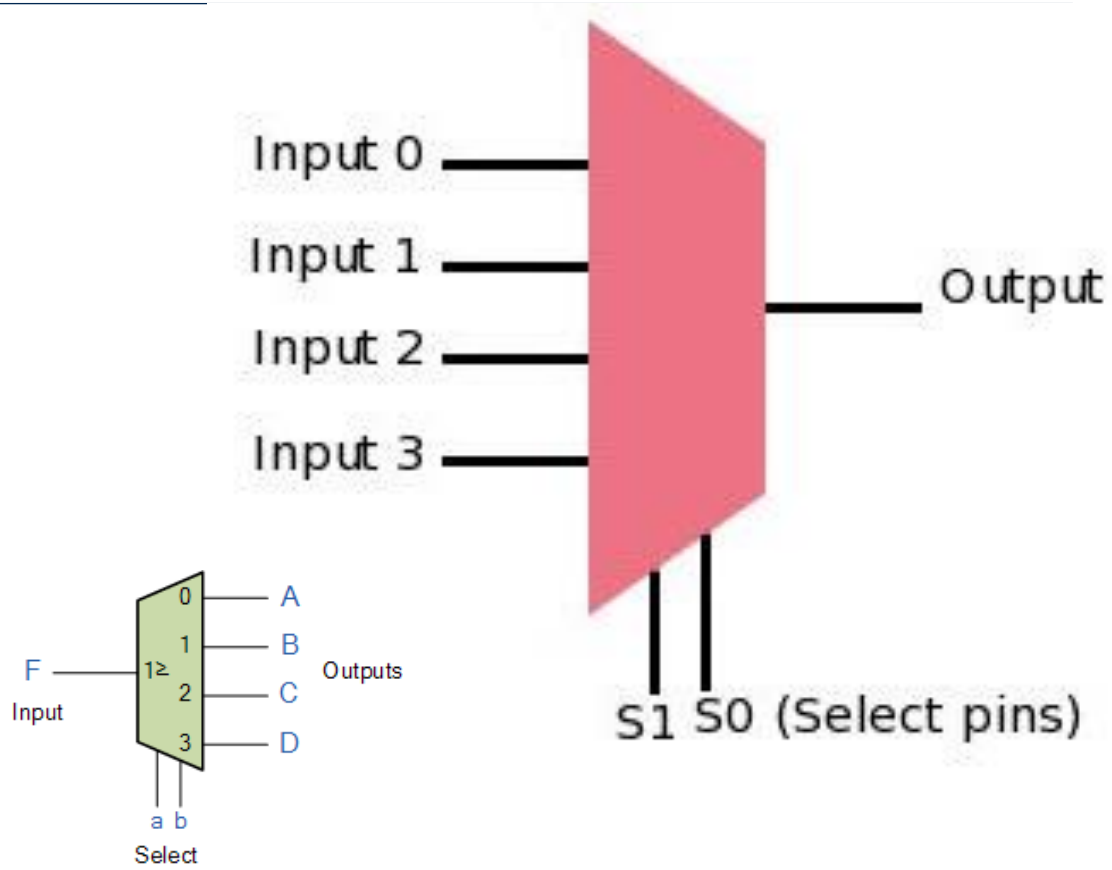
Story: You have 8 sensors and You would like to forward their signals (digital signal) to an another processing unit based on some manually set inputs.

Task Description: Multiplexing 8 signal, digital switching inputs needed, the input signal needs to be directed to the output

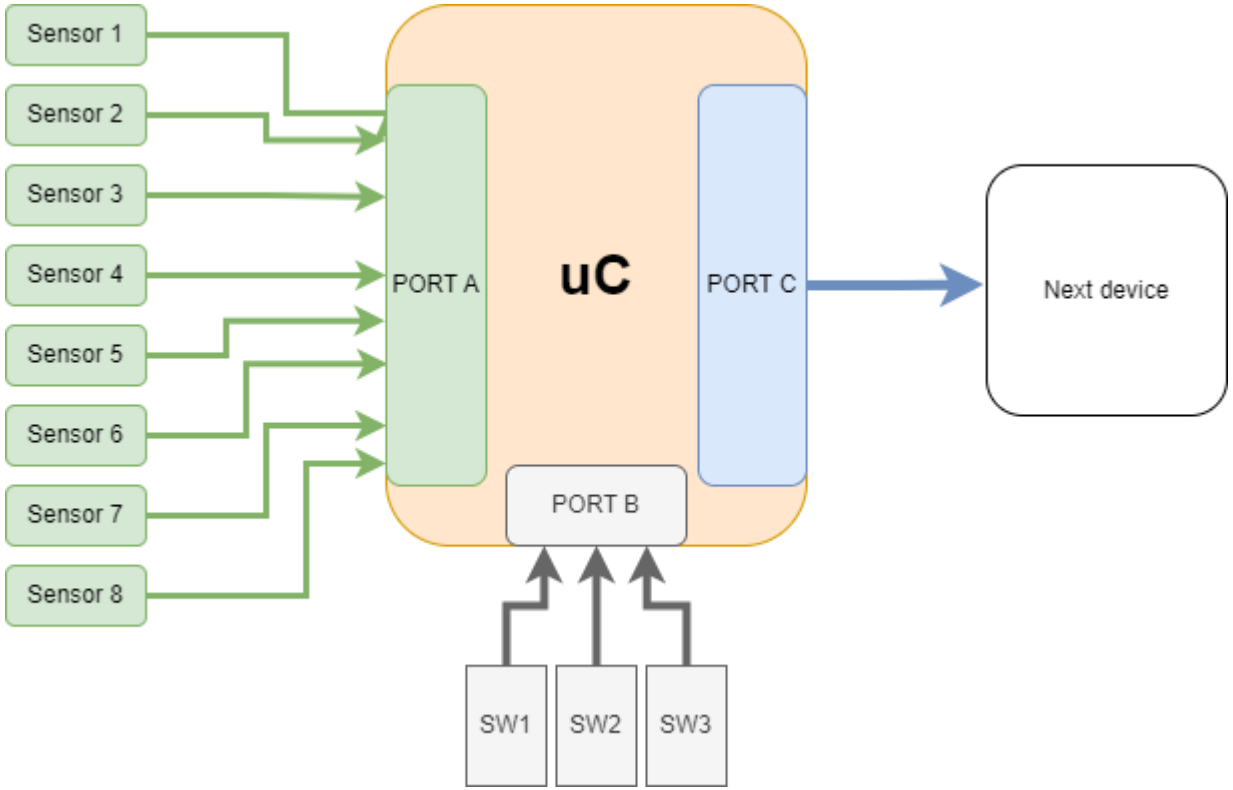
Multiplexer

- Digital electronics component
- Main function:
  - forwarding signals on input pins
  - to one output pin
  - based on the selector pins input

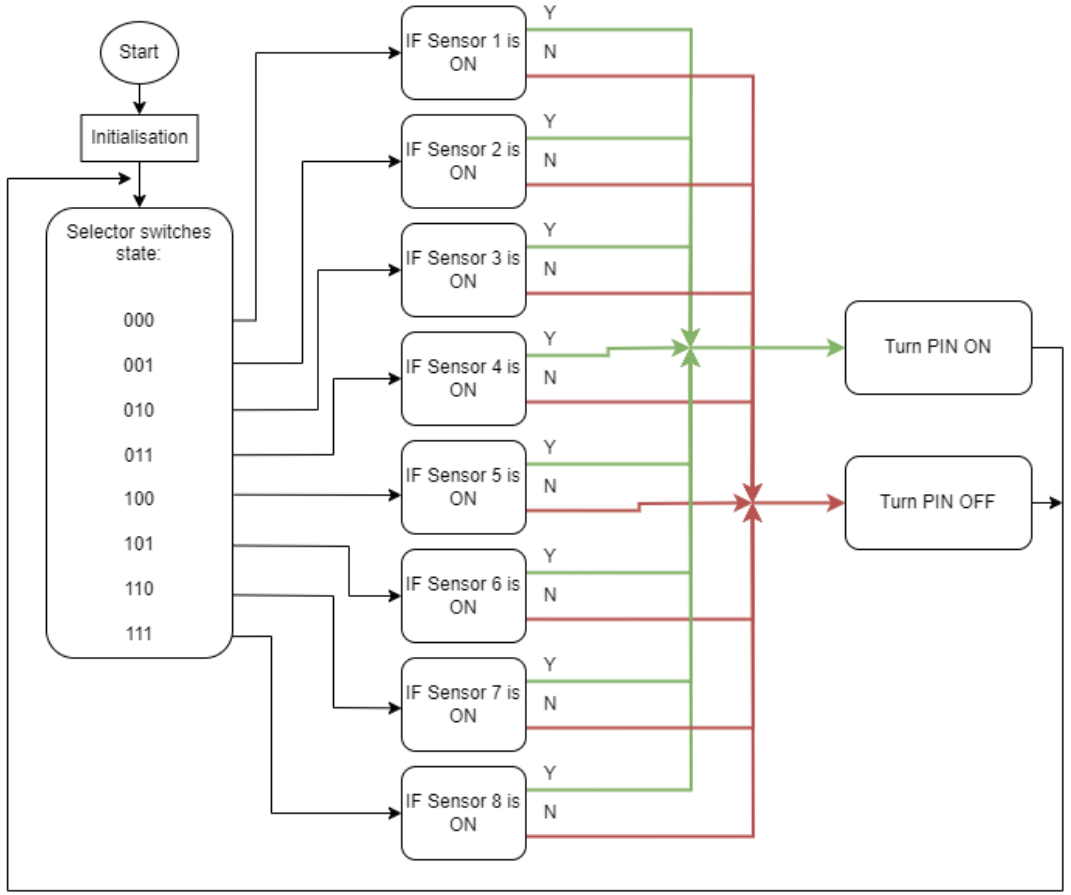
- Dual pair the demultiplexer
- Used: old telephone centers etc.



Electronics logic plan



Program flow

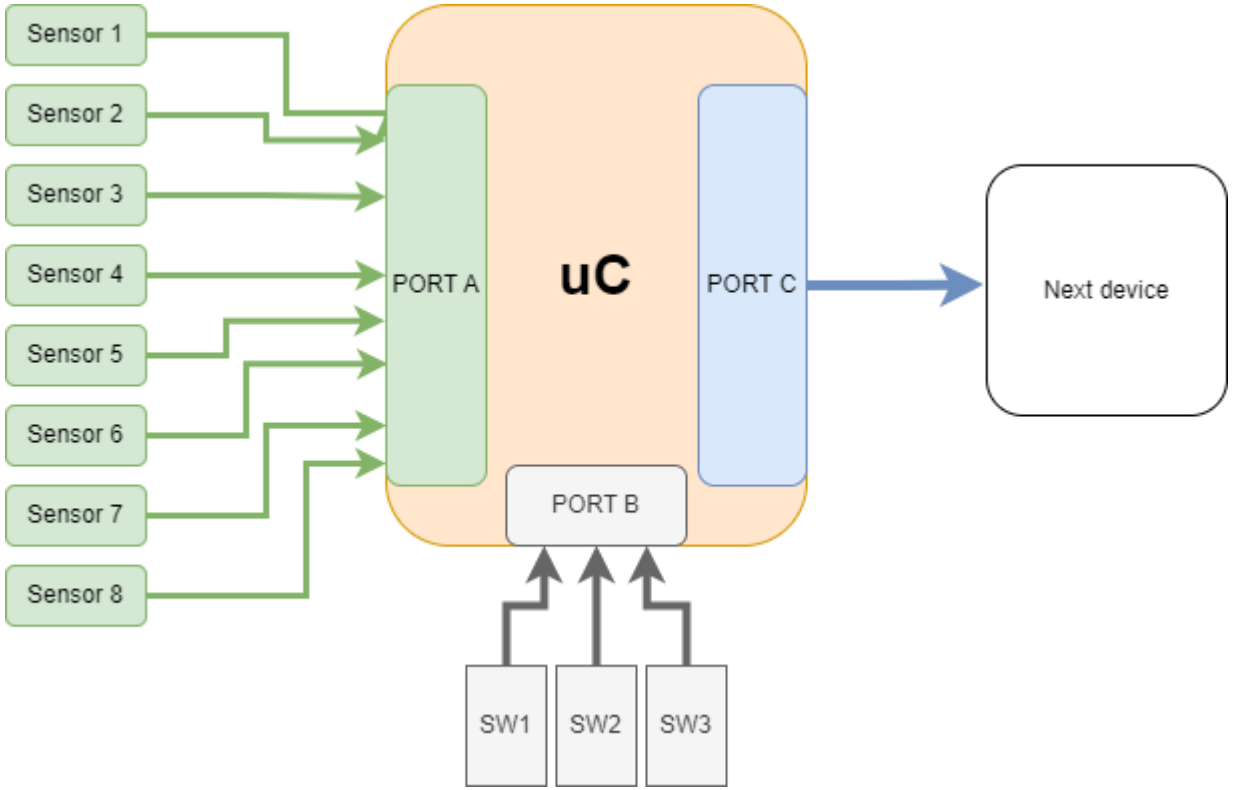


Interrupt

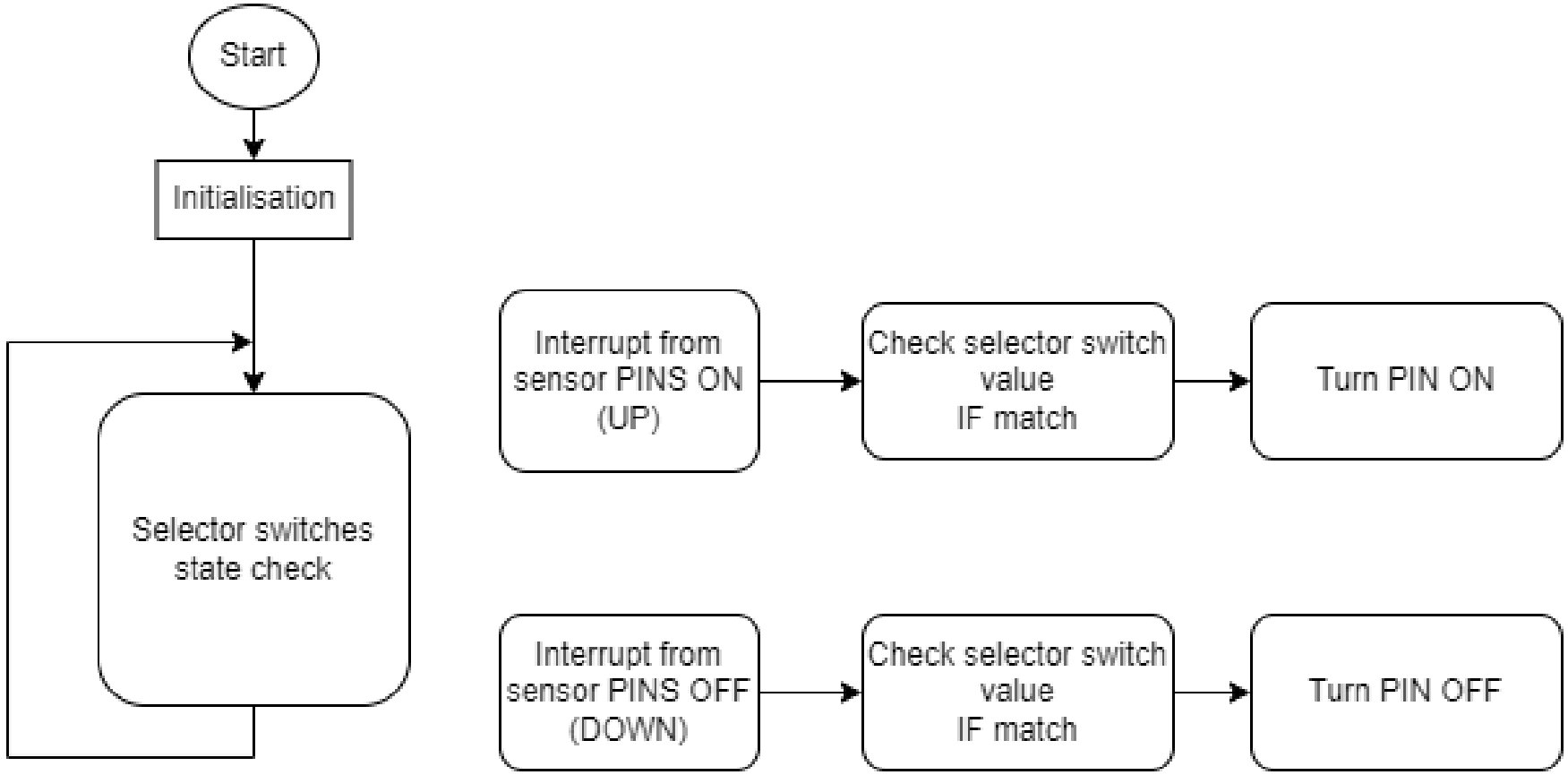
- On some external (or internal) event stops the main program to complete the coresponding subroutine
- Usually all pheriperals can initiate an interrupt
- In some microcontrollers hierarchy can be added to the interrupts

- Examples: interrupt on a down-to-high change on a pin

Electronics logic plan

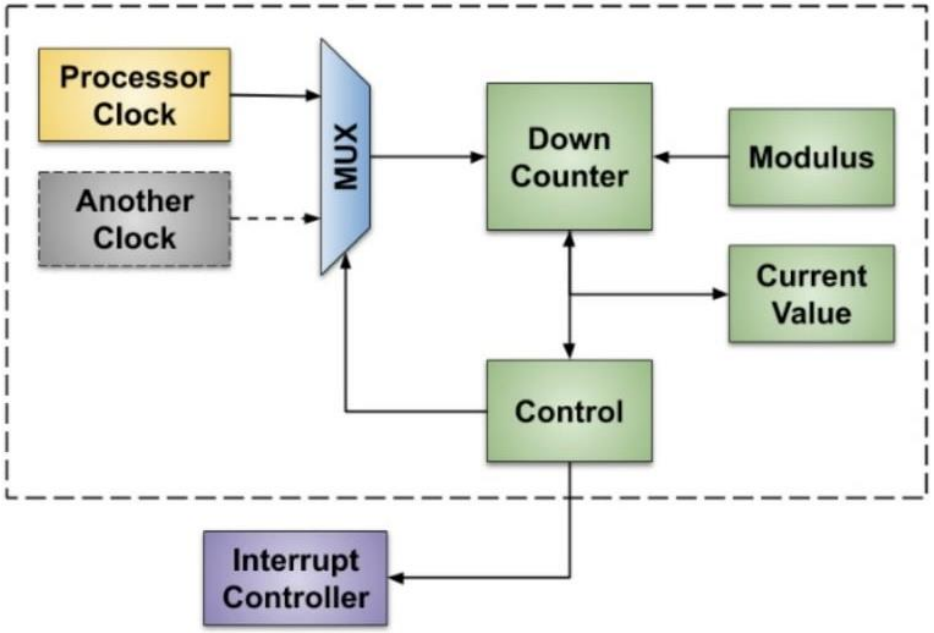


Program flow with interrupt



Timers

- One of the most used functionalities:
  - Timing events (scheduling)
  - Generation of specific signals (PWM)
  - Event synchronisation (internal or external)
  - Timeouts
  - Etc.
- Usually consists of:
  - Clock signal generator
  - Prescaler
  - Controller

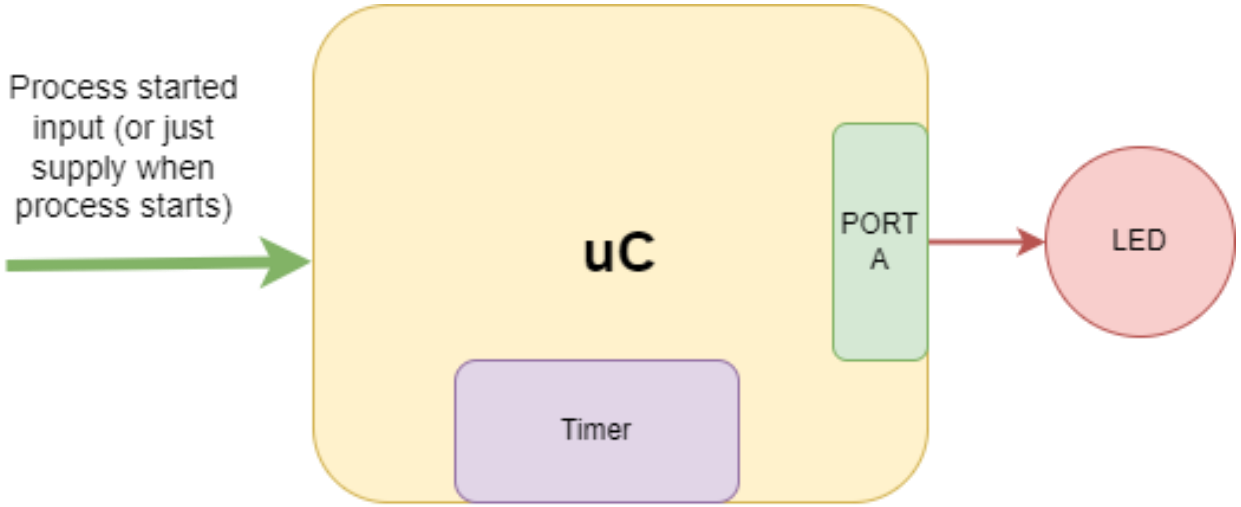


Case study 3 – Blinker (Hello Microcontroller world)

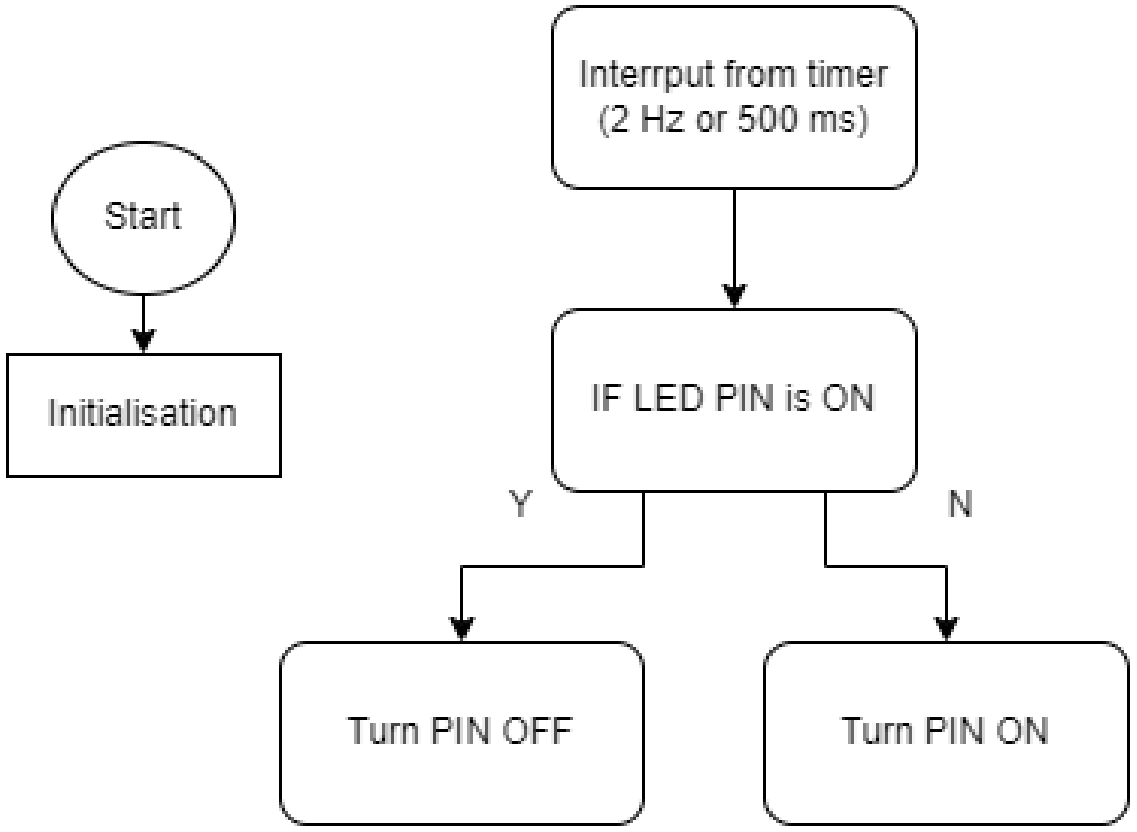
Story: You have a machine that has potential threat on human near to it during operation. You decide to add a blinking light source to warn bystanders.

Taks description: A square wave generator is needed with the given frequency (1 Hz with 50% duty cicle can do its job probably).

Electornics logic plan



Program flow

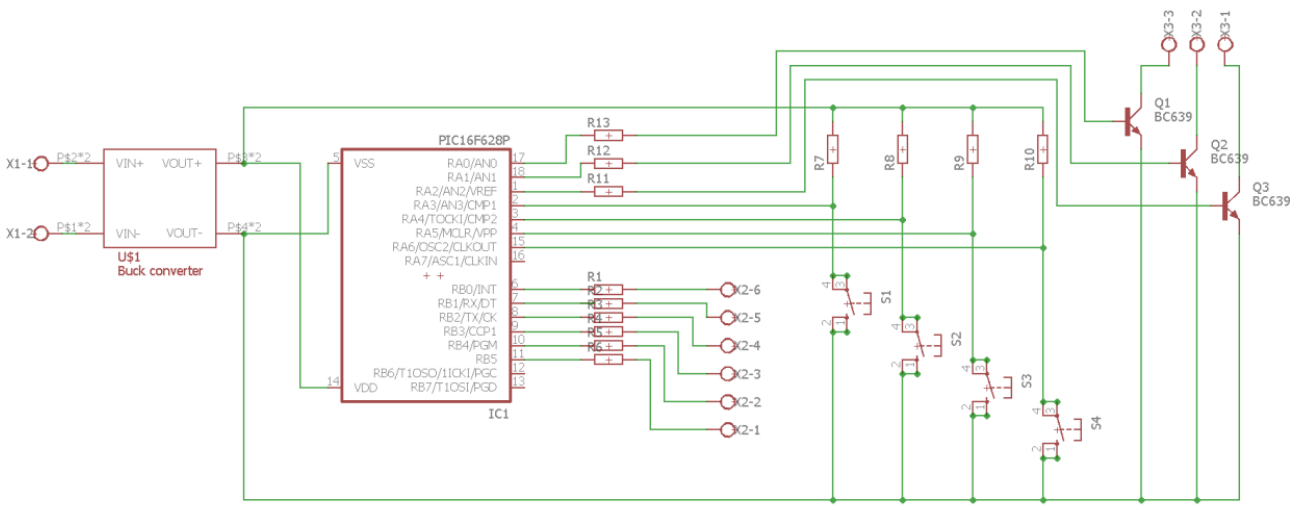


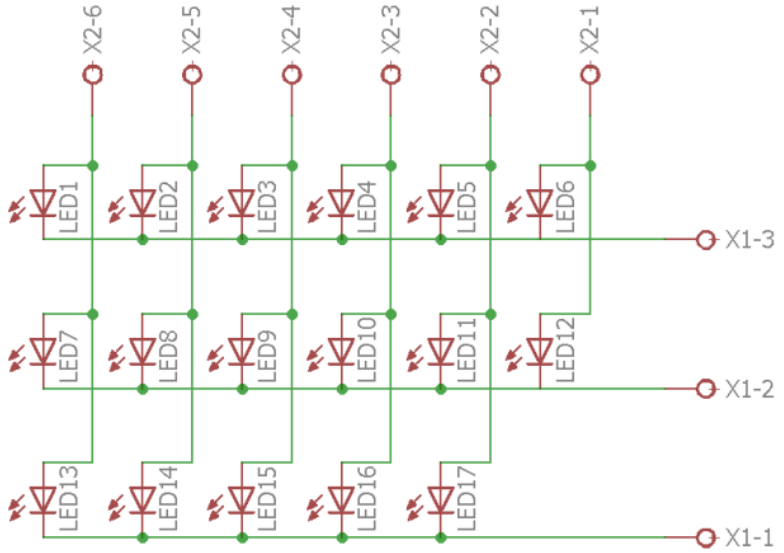
Case Study 5 – Binary Clock

Story: You want to make some puns on Your friends when they ask You the time.

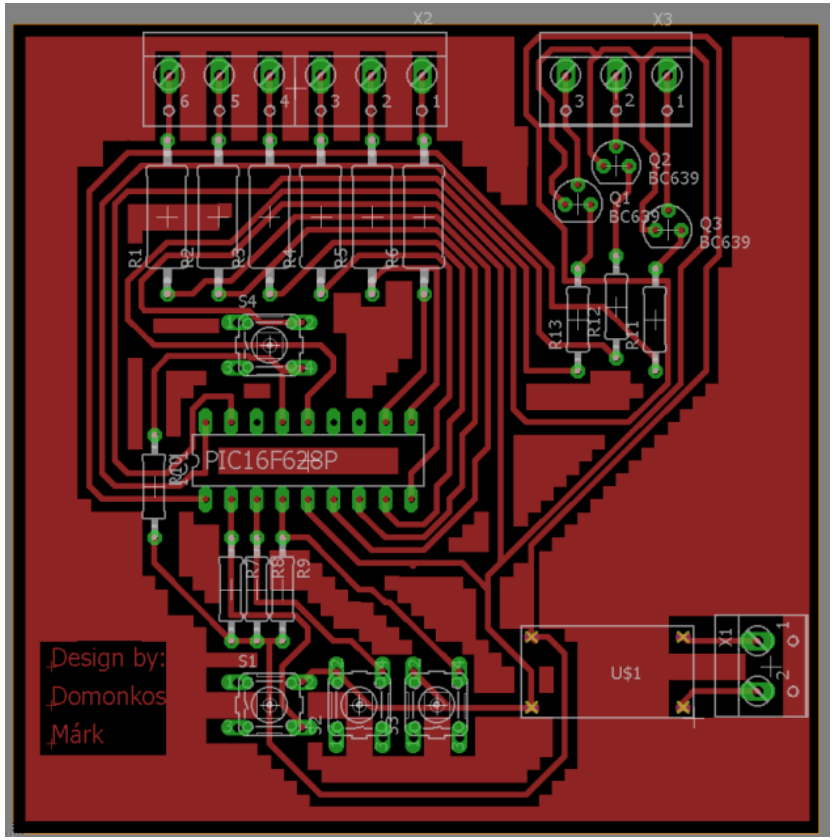
Taks description: You need a matrix of LEDs as the display of the clock, some buttons timers, and supply electronics.

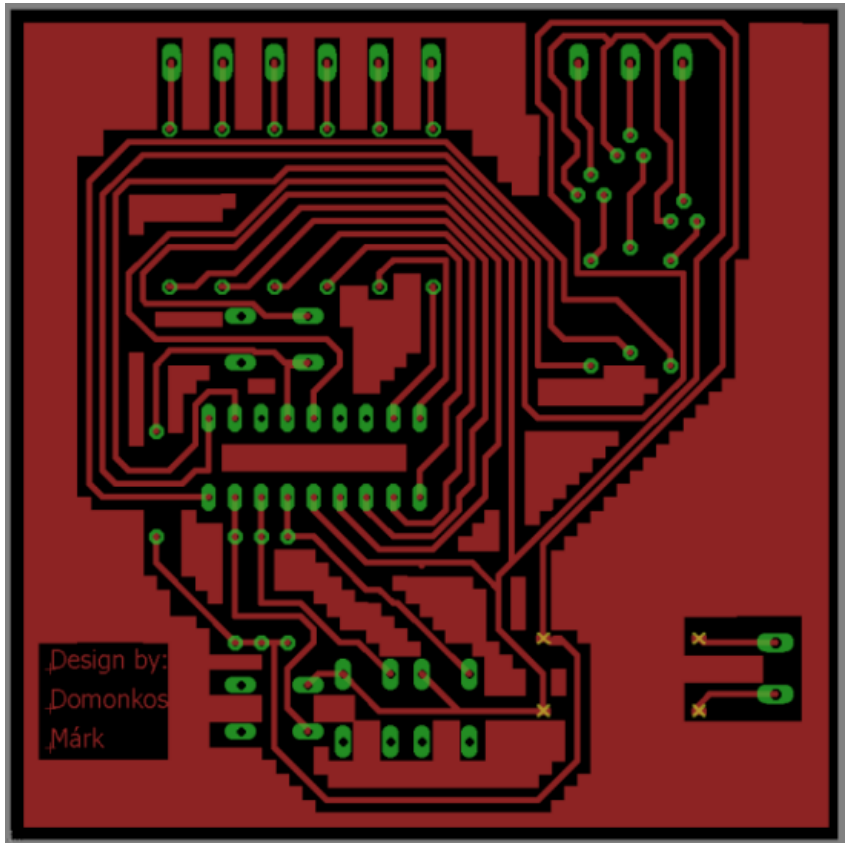
Electronics Schematic



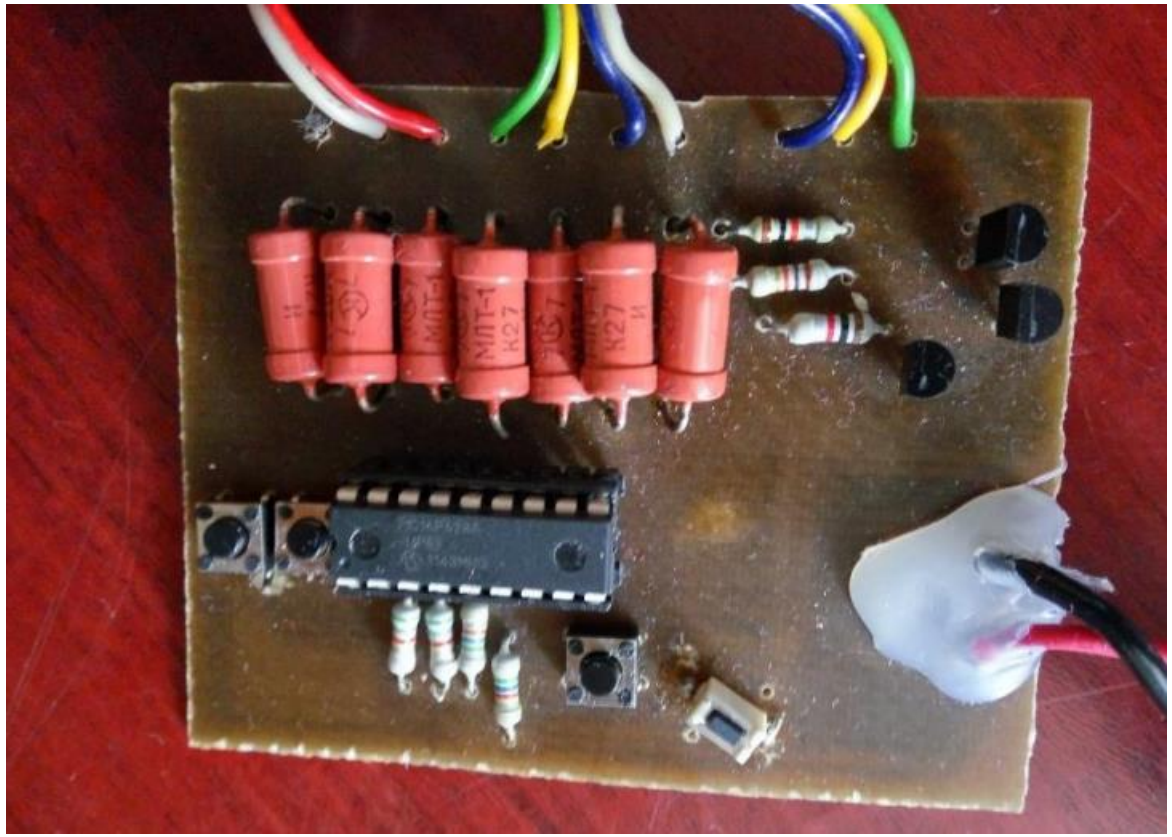


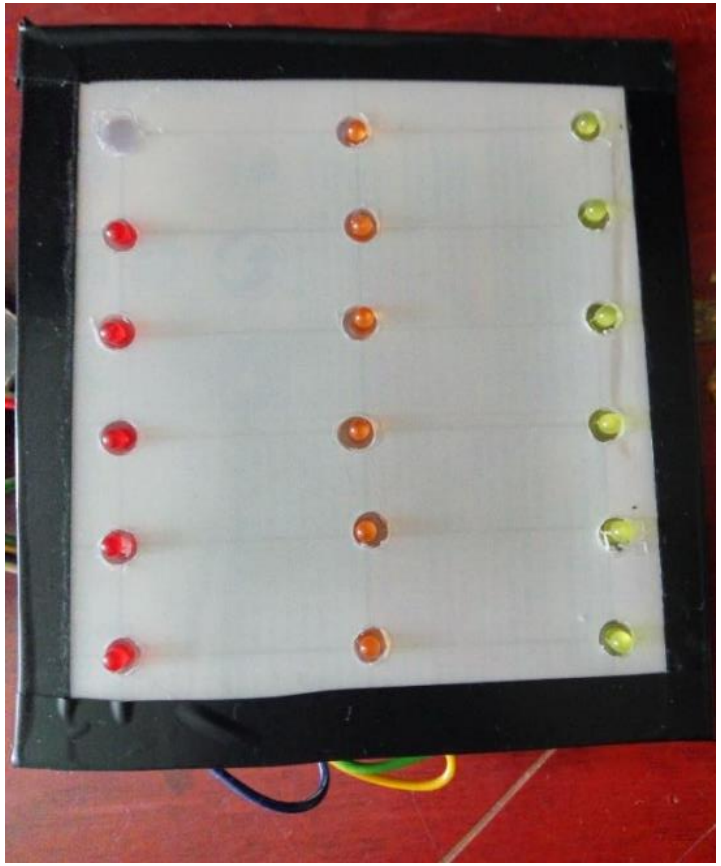
Electronics layout





The real board (not beautiful but works)

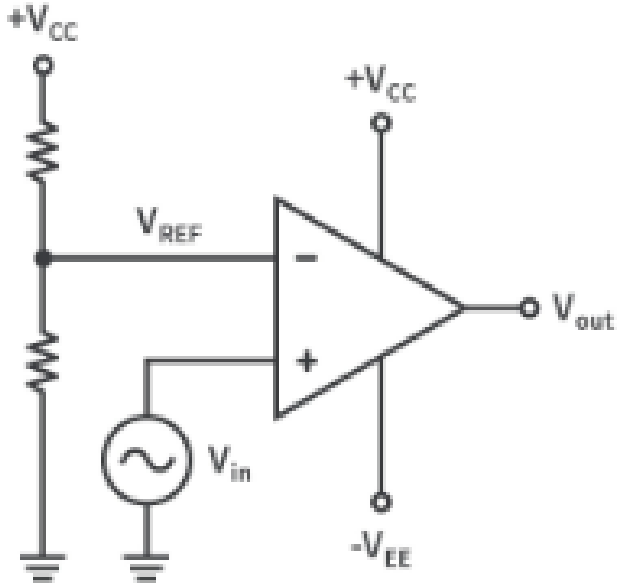


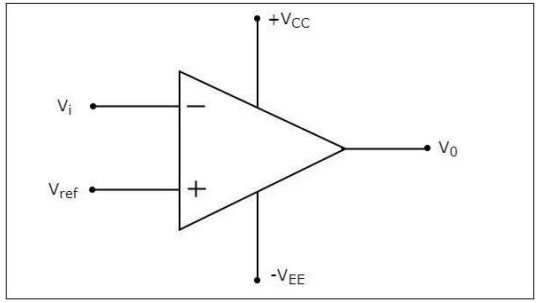


Demo of the binary clock

Comparators

- Comparison of a reference sygnal (or value) and a usually external input
  (like: if (ext_sing > ref): then….; else….;)
- Can be also implemented by a program
- Can initiate interrupt
- It is usually implemented with operational amplifiers



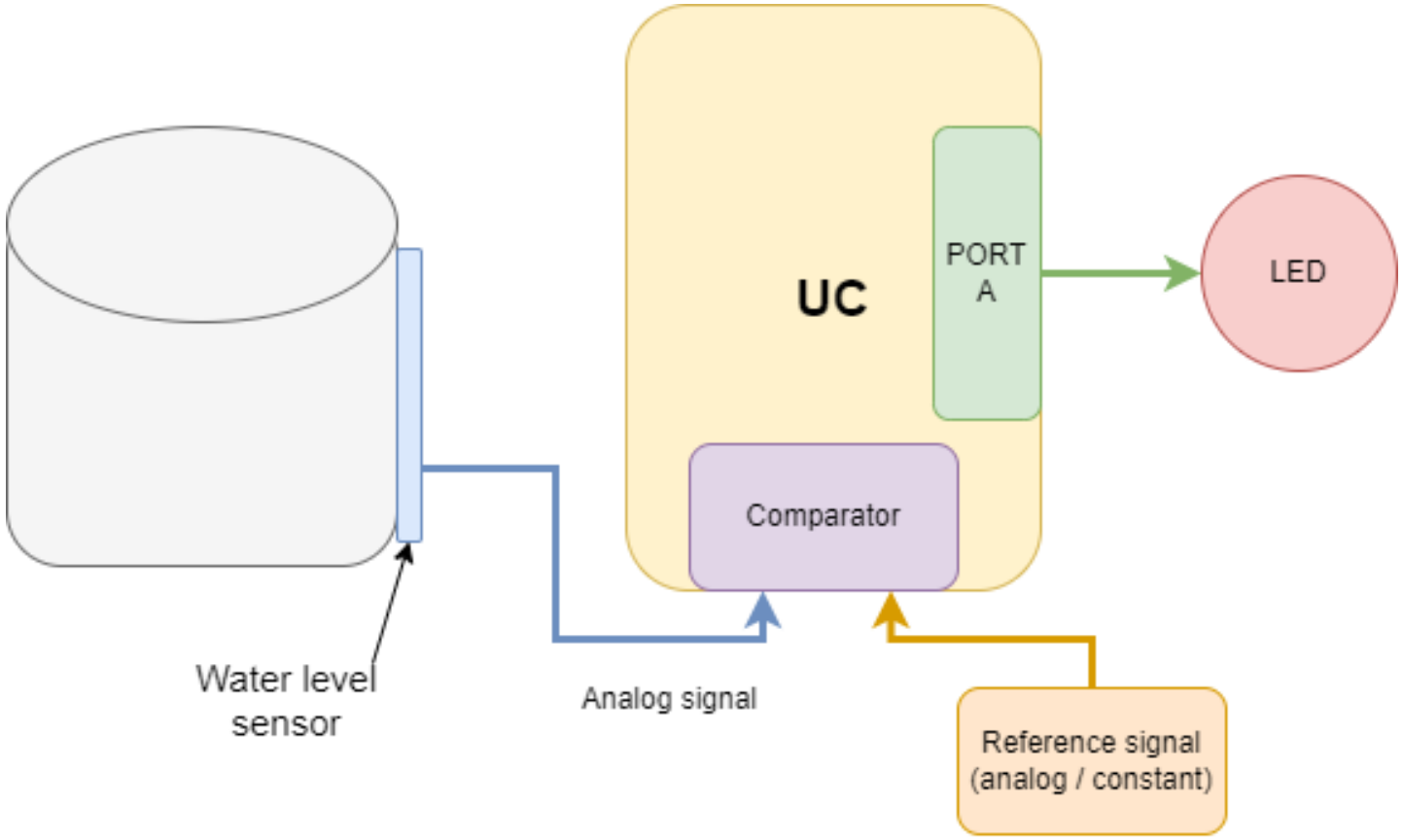


Case study 6 – Water dump control ☺

Story: You have an airconditioner that has to dump its water into a bucket. You are too lazy to check it from time to time, so You decide to make a simlple dumping mechanism that dumps all the water from the bucket when it is filled (or tells You to do so. ;)).

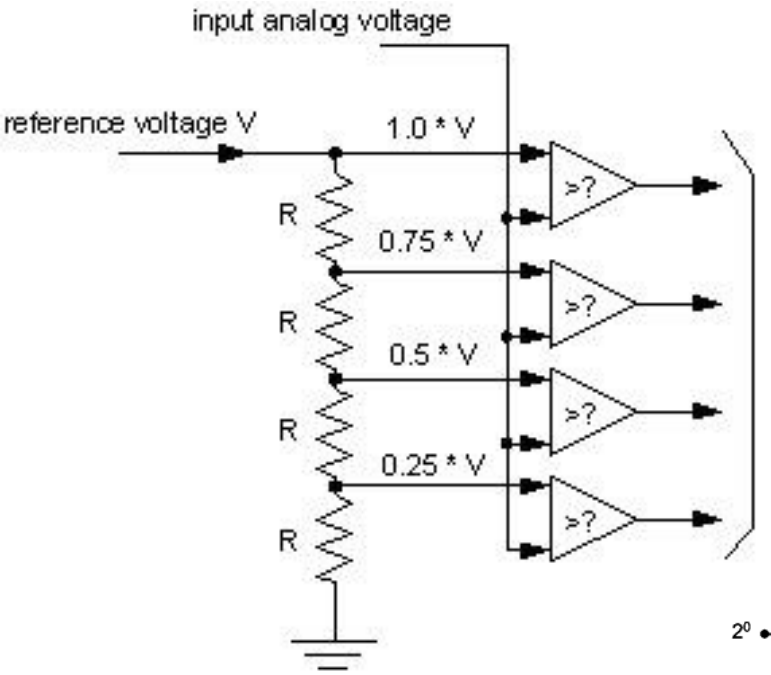
Task Description: You have a sensor in the bucket that sends You the level of the water. You need to compare it to a certain level and take an action (turn ON a PIN).

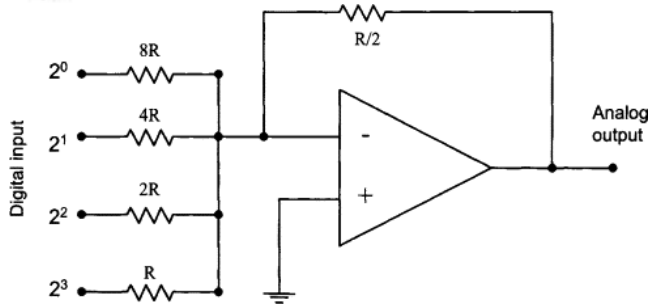
Logic schematic



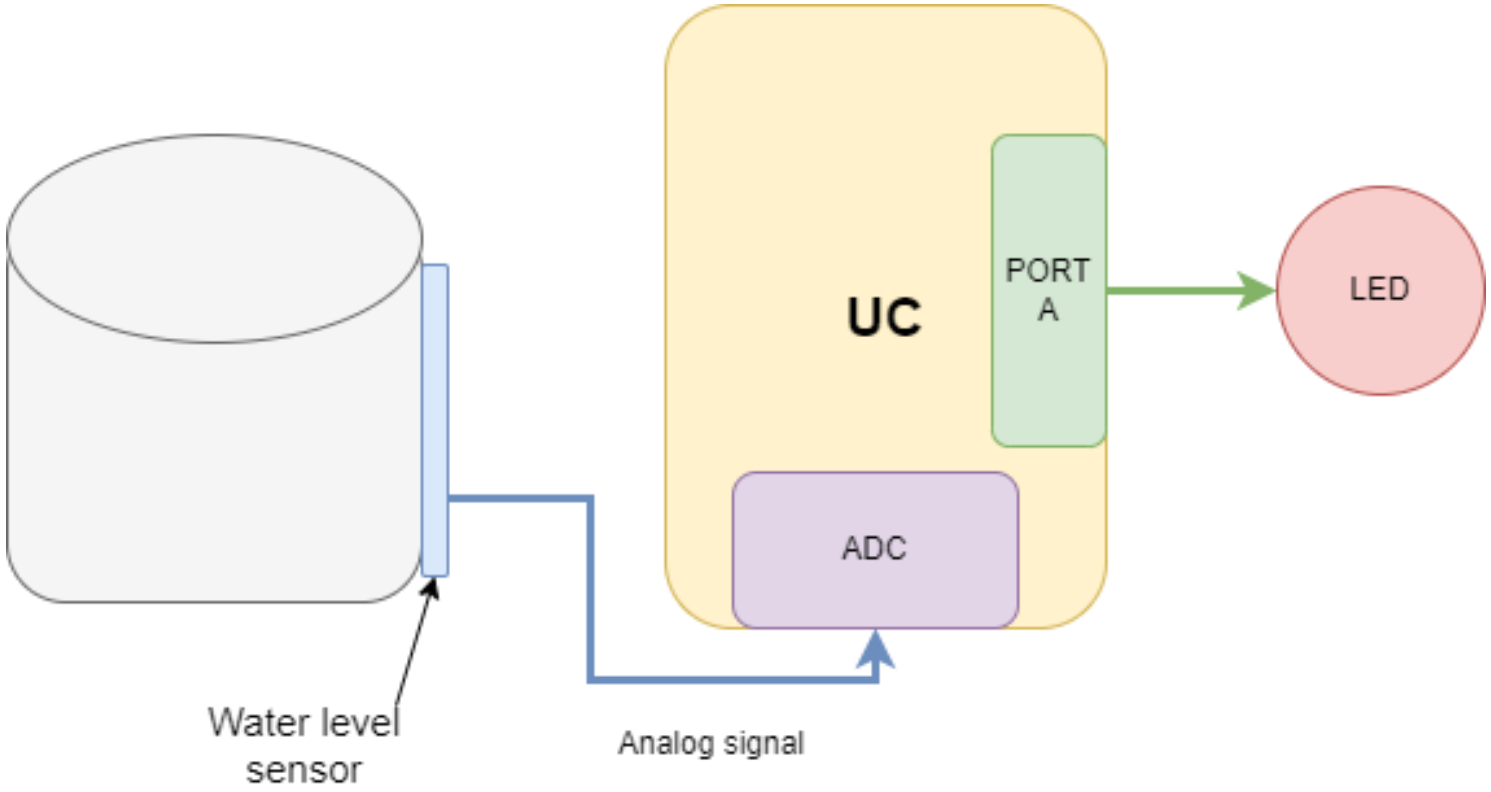
A/D converters and D/A converters

- Conversion of analog signals to digital ones (ADC) and digital signals to analog ones (DAC)
- Commonly used functionality in embedded systems

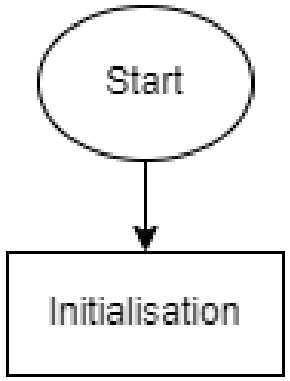


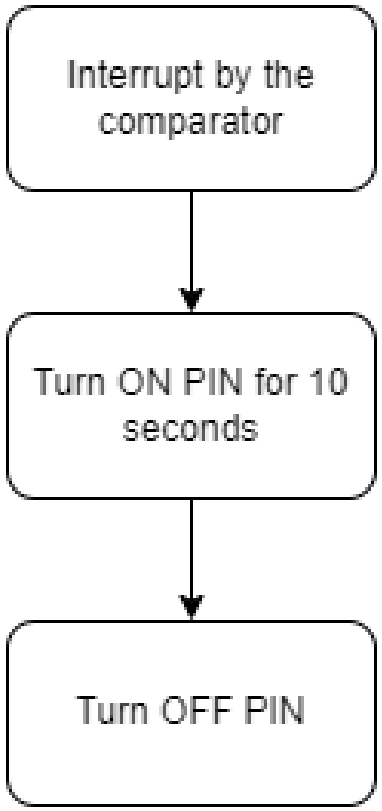


Logical schematic – with ADC



Program flow



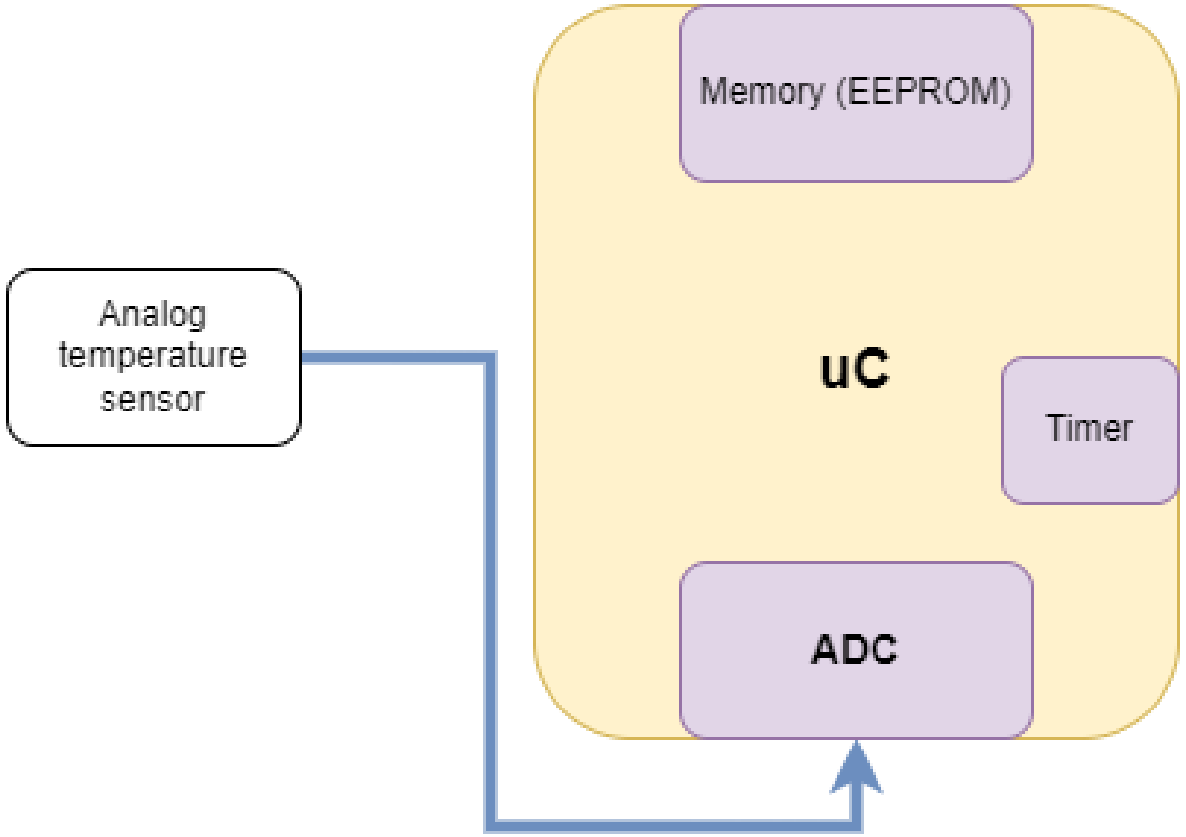


Case study 7 – Recording temerature

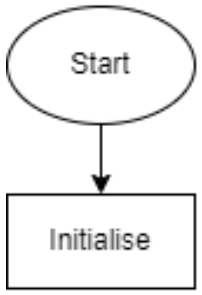
Story: You want to have a database of Your rooms temperature troughtout the Year.

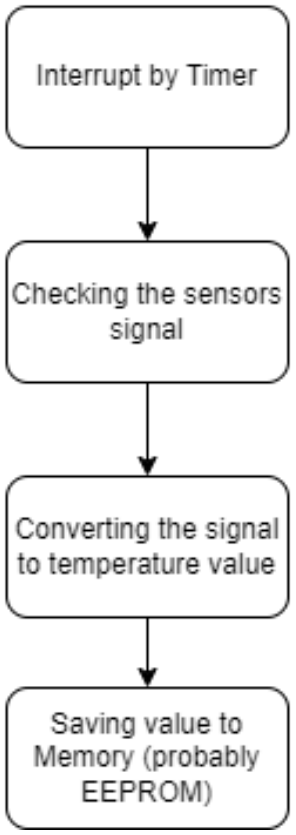
Task description: You have a teperature sensor that sends analog signals. You will download the data regularly from the controller.

Logic Schematic



Program flow



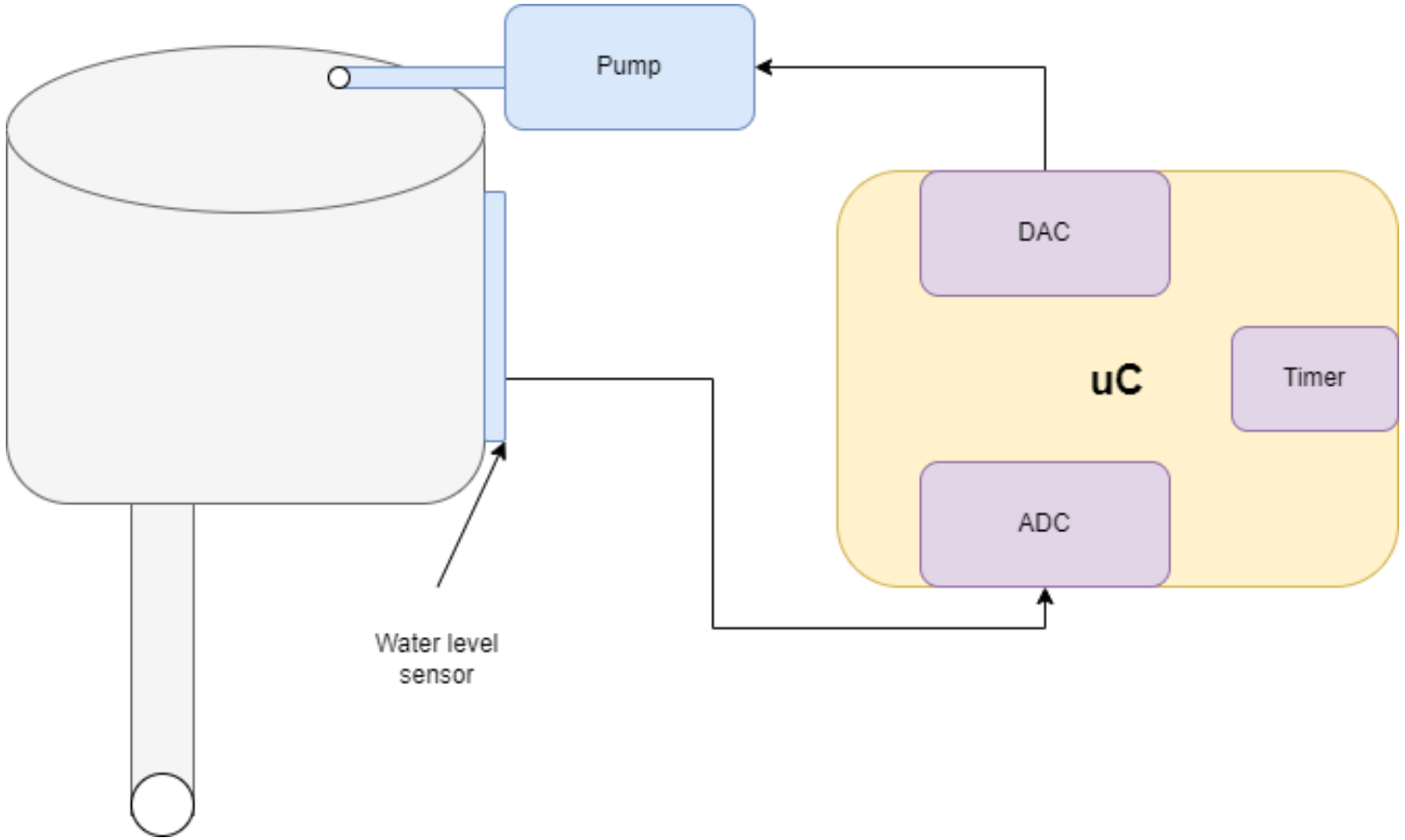


Case study 8 – Simple water level control

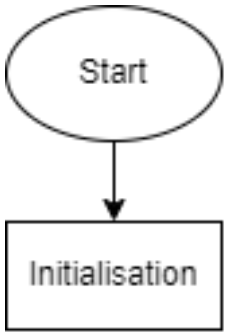
Story: You have a tree irrigation system that has a big barrel of water. In the barrel the water level always needs to be nearly constant (that way the waterflow to the trees will be good, and also the water will have enought time to heat up to be optimal for the trees).

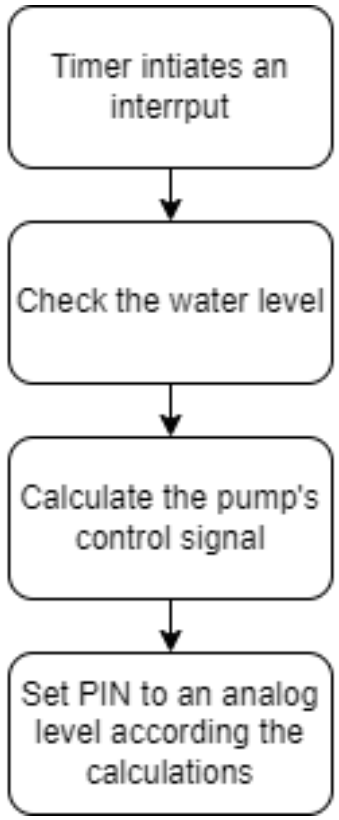
Task description: You have a barrel with a water level meter + pump attached to the barrel for water supply. ADC conversion will be needed from the sensor part and DAC for the pumps DC motor control. You also figured out an equation that maps between the water level and the DC motors power for a good enought control of the water level.

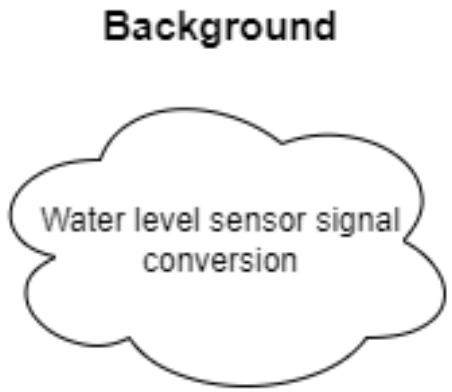
Logic schematic



Program flow





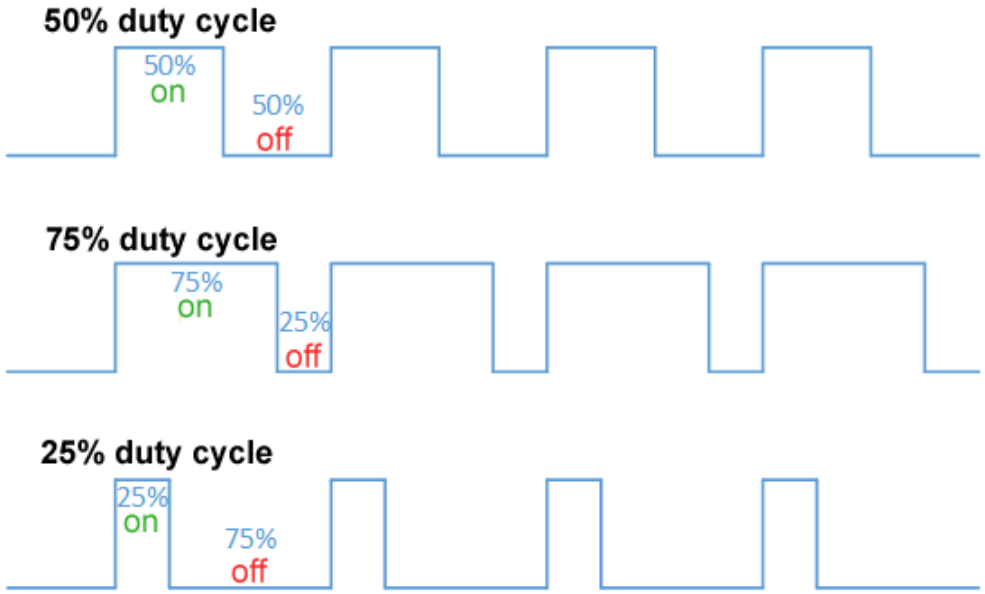


Alternative solution

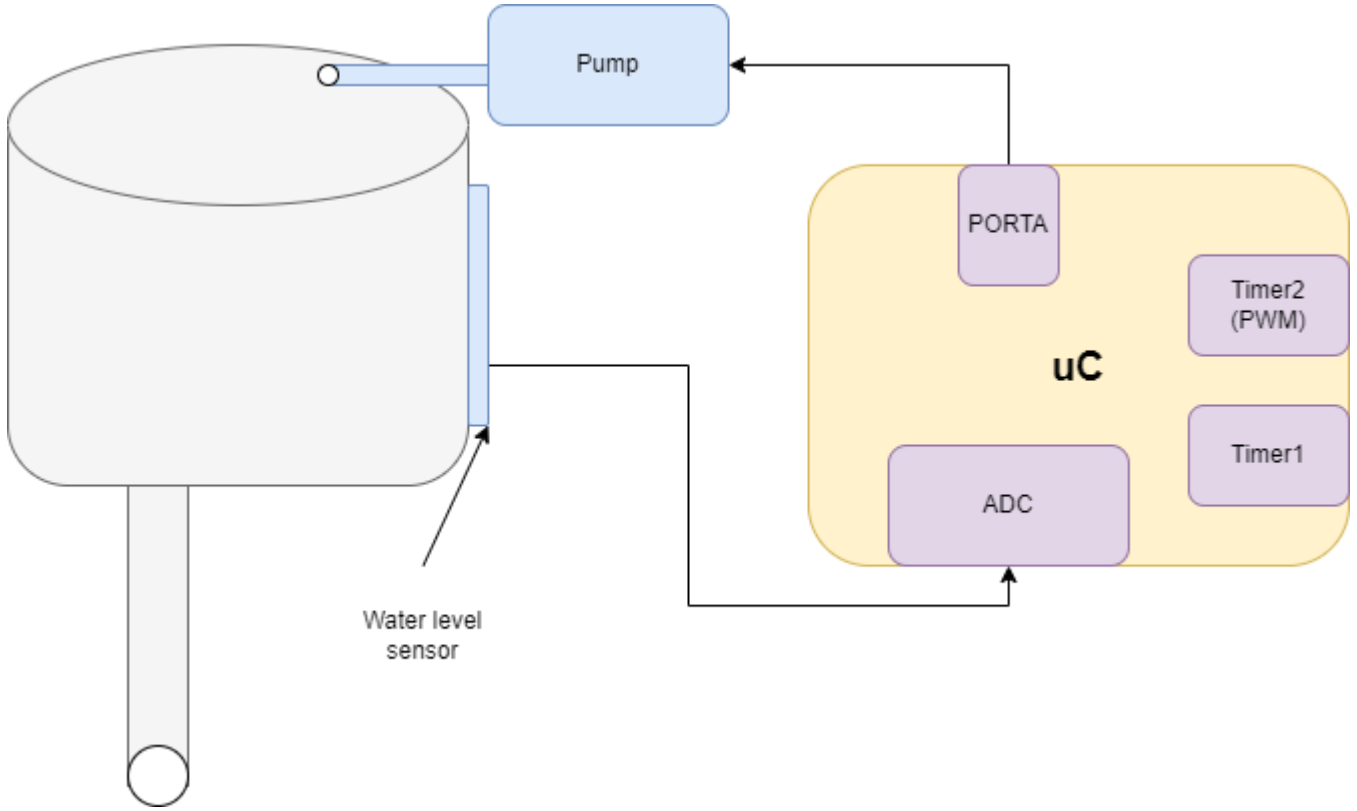
- Usage of PWM signal instead of analog signal for the motor control.

PWM (recap)

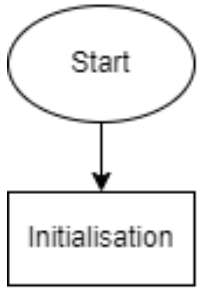
- Pulse Width Modulation
- Parameters:
  - Frequency [Hz]
  - Duty cicly [%]
- Used for imitation of analog signals with alternating ON-OFF cicles
- Uses the inertia of the controlled system
- Examples: Motor control (KHz), Light dimming (~100Hz)

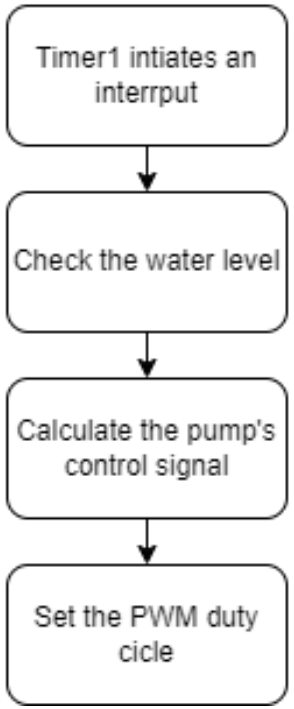


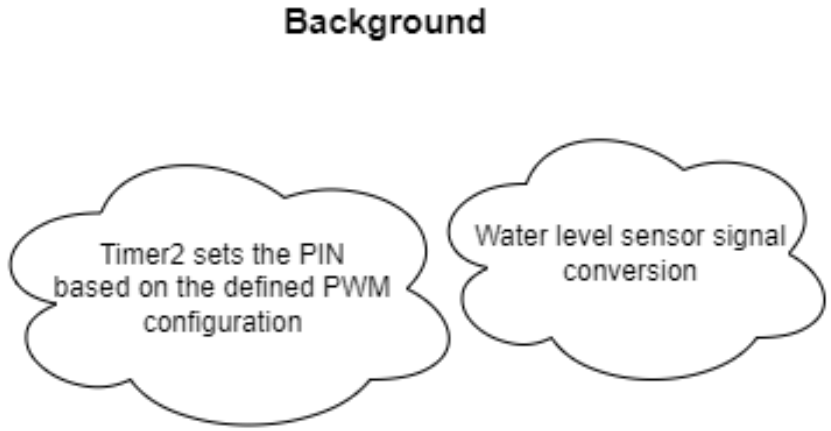
Logic Schematic



Program flow







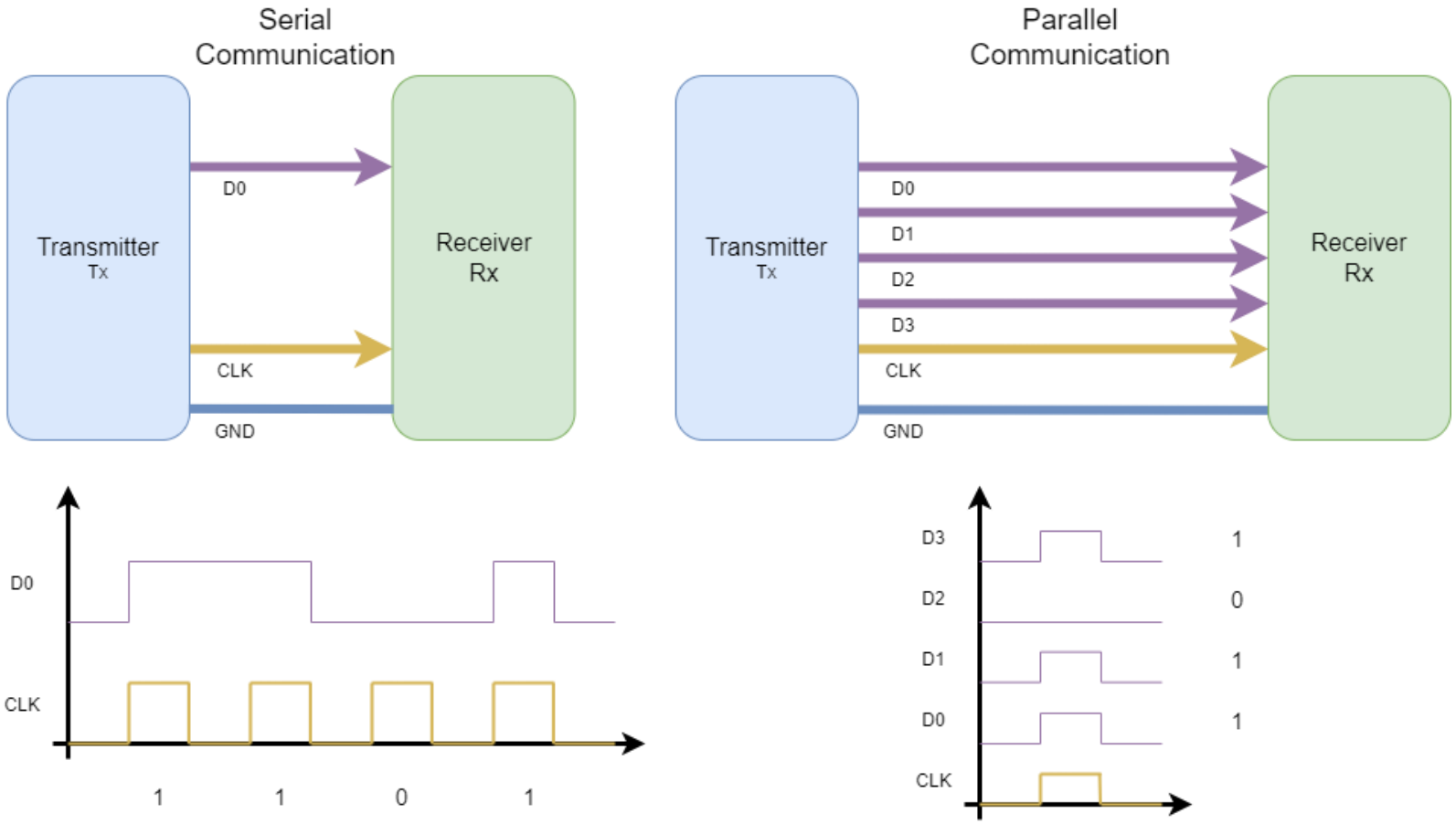
Communication protocols

- Most widely used protocols in embedded systems:
  - UART
  - SPI
  - I2C
  - CAN bus
  - ModBus
  - FLEXRAY
- Needed a Transmitter (Tx) a Receiver (Rx) and a broadcast medium
- Also needed a protocol which based on the information inside a message can be coded/decoded

Communication can be

- Based on the data transmission (channels involved):
  - Serial
  - Parallel
- Based on the direction of the communication:
  - Unidirectional (one way communication)
  - Bidirectional (back-and-forth communication)
- Based on the simultaneous communication ability:
  - Simplex
  - Half duplex (walkie-talkies)
  - Duplex (mobile phones)
- Based on hierarchy
  - Master – Slave
  - Non-MS type

Serial and parellel communication



Case Study 10 – Custom Communication on 16 bits

Story: You have a module that is sensing an environment with high precision. You decided that 16 bit for one message will be enought precise for the process. Unfortunatelly the module can’t communicate with serial communication, but has 9 free pins on it can communicate.

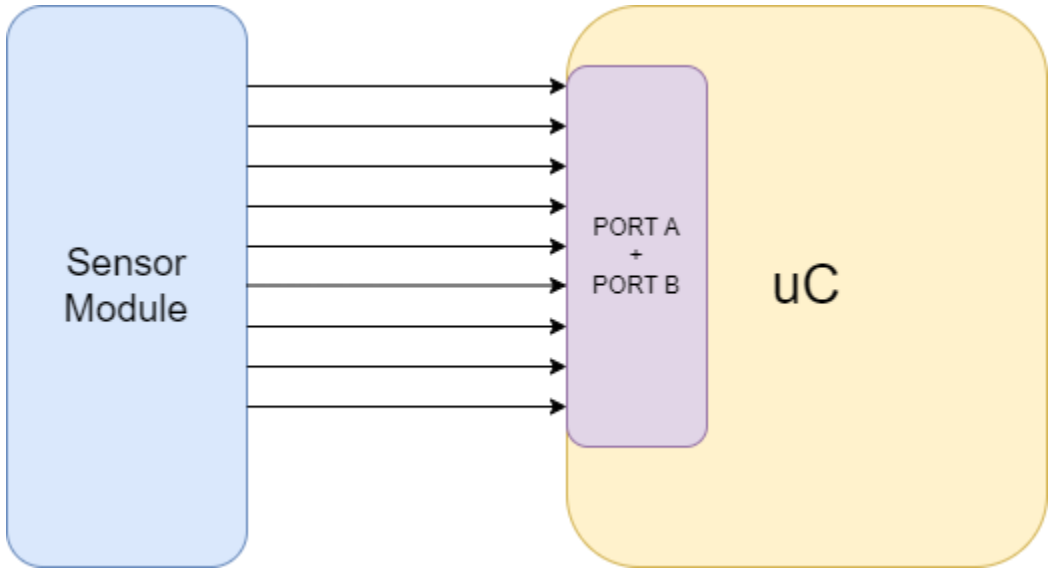
Task description: You will need a microcontroller that has at least 9 free pins.

Schematic logic

- Remark: To the communication to be understandable the modul always sends data in the following pattern:

255-0-0-0-data_L-data_H

- Let PORT A pins receive the data
- Let PORT B pin0 receive clock signal for sincronisation



Program flow draft