Universität Stuttgart
Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen

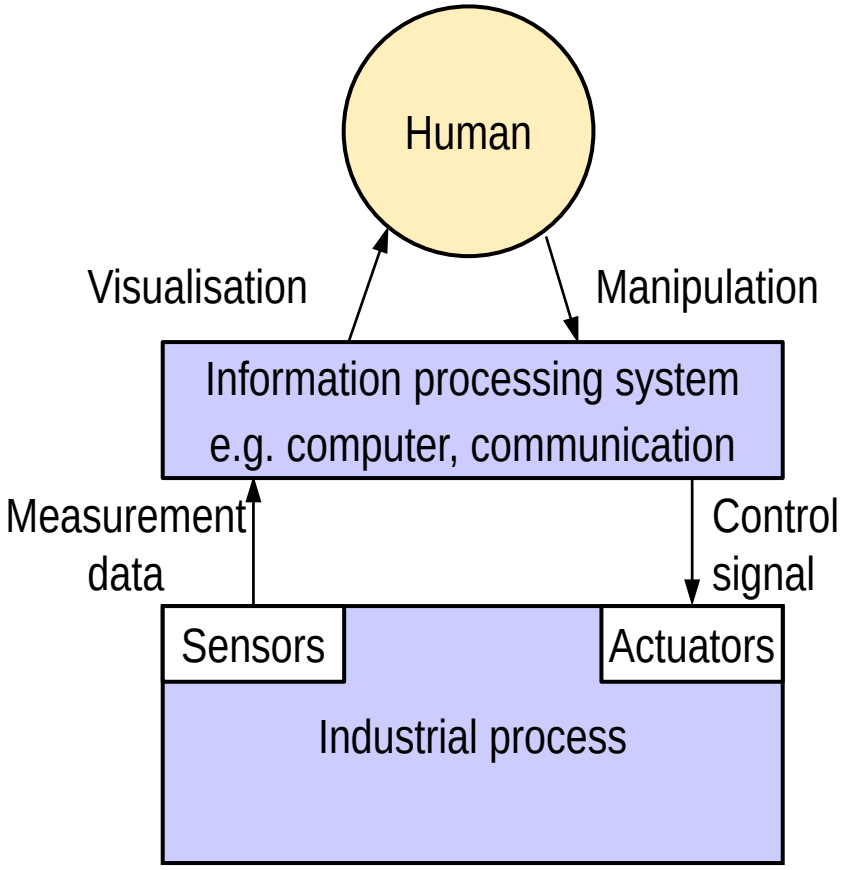
MEDIS – Module 2

Microcontroller based systems for controlling industrial processes

Chapter 3: Input-/Output-system of microcontrollers

M. Seyfarth, Version 0.1

Interfaces to the industrial process



Input - / Output-interfaces

There is a 2-way-communication between industrial process and information processing system

- Signal input: Measurement data
- Signal output: Control signals, Control of actuators

Input in direction to microcontroller

Output in direction to process

Realisation of signal input

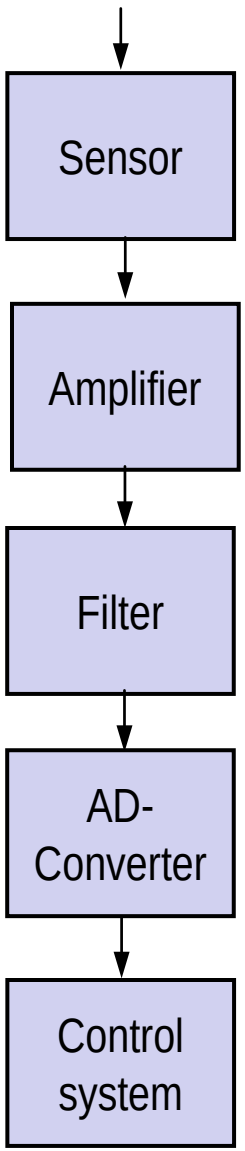
Universität Stuttgart

- Direct connection wire by wire
  - Each process signal is directly connected from the sensor to the information processing system by wires as a star
  - Signal processing is done in the information processing system (microcontroller)
- Connection via a fieldbus
  - Connection by a bus-coupler unit
  - Signal processing is done in the information processing system (microcontroller)
- Connection via a sensor-actuator-bus
  - Direct connection of sensor to bus-system
  - Signal preprocessing is done in the sensor
  - Intelligent sensors

Types of electrical process signals

- Analog signals
  - Analog in amplitude
  - Analog in frequency
  - Analgo in phase
- Binary signals (TRUE, FALSE)
- Digital signals
- Pulsed signals
- Sloped signals

Information flow at signal input



Physical value

Electrical or optical signal

Amplified signal (on higher voltage level)

Scaling, Filtering, Linearization, Temperature compensation, …

Filtered signal (e.g. low-pass-filter against noise)

Sampled signal, time discrete, value discrete

Normally missing if binary or digital signal

Sensors

- Basic Tasks of sensors
  - Capture a physical process value, e.g. temperature, position, …
  - Convert physical value to an adequate, processible value, e.g. voltage, current, …
- Possible physical effects:
  - Change of resistance
  - Change of inductance or capacity
  - Ppiezo-electrical effect
  - Thermo-electrical effect
  - Photo-electrical effect
- Additional Tasks of intelligent sensors
  - Supervision of boundary values
  - Self testing
  - Self calibration
  - Adaption to special bussystems

Types of Sensors

- Classification by sensor signals
  - Binary sensor: TRUE-/FALSE-information, e.g. limit switch
  - Digital sensor: e.g. multi-switch, encoder
  - Analog sensor: signal is value continous and proportional to physical value, e.g. temperature-sensor, distance-sensor
- Possible physical effects:
  - Change of resistance
  - Change of inductance or capacity
  - Piezo-electrical effect
  - Thermo-electrical effect
  - Photo-electrical effect
- Additional Tasks of intelligent sensors
  - Supervision of boundary values
  - Self testing
  - Self calibration
  - Adaption to special bussystems

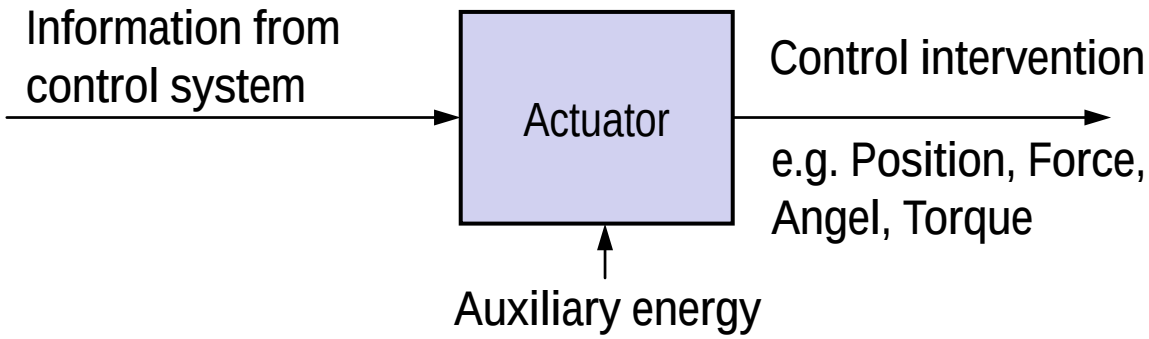
Types of sensors

Universität Stuttgart

Physical value	Sensor	Output value
Temperature	Thermal element Metallic Resistors Semiconductor-Resistors (negative temperature coefficent) Ceramics Resistors (positive temperature coefficient)	Voltage (mV) Change of resistance Change of resistance Change of resistance
Pressure	Pressure cell with diaphragm and strain gauges Pressure cell with silicium diaphragm (piezo)	Change of resistance Change of resistance
Force	Strain gauges Inductive force sensors	Change of resistance Change of inductance
Revolution speed	Tachometer generator Counting of pulses	Voltage (V) Sequence of pulses
Acceleration	Silicium-piezo resistor Ferraris sensor	Change of resistance Induced Voltage (mV)
Flow Rate	Cylindrical piston meter Inductive flow meter	Sequence of pulses Voltage (mV)
Distance	Hall-element Ultrasonic sound reflection	Voltage (mV) Voltage / pulselength
Angel	Resolver Impulse generator	Digital value Sequence of pulses
Light intensity	Photodiode Photoresistor	Current (μA) Change of resistance

Actuators

- Task of actuators is the transformation of the information of the control system to control interventions in the technical process



- Actuators for different actuating variables
  - Optical variables
  - Mechanical variables
  - Thermic variables
  - Flow rates

Types of actuators

Universität Stuttgart

- Generate movement
  - Electromechanical: Motor, stepper motor, magnet, linear drive, …
  - Hydraulic: cylinder, hydraulic motor
  - Pneumatic: cylinder, pneumatic motor
  - Piezoelectric actuators
  - Thermometalls
- Damping / braking of movement:
  - Electromechanical: magnet with brake shoe
  - Electrorheologic or magnetorheologic actuator
- Manipulate flow:
  - Electromechanical: valve with magnet, valve with motor

Universität Stuttgart

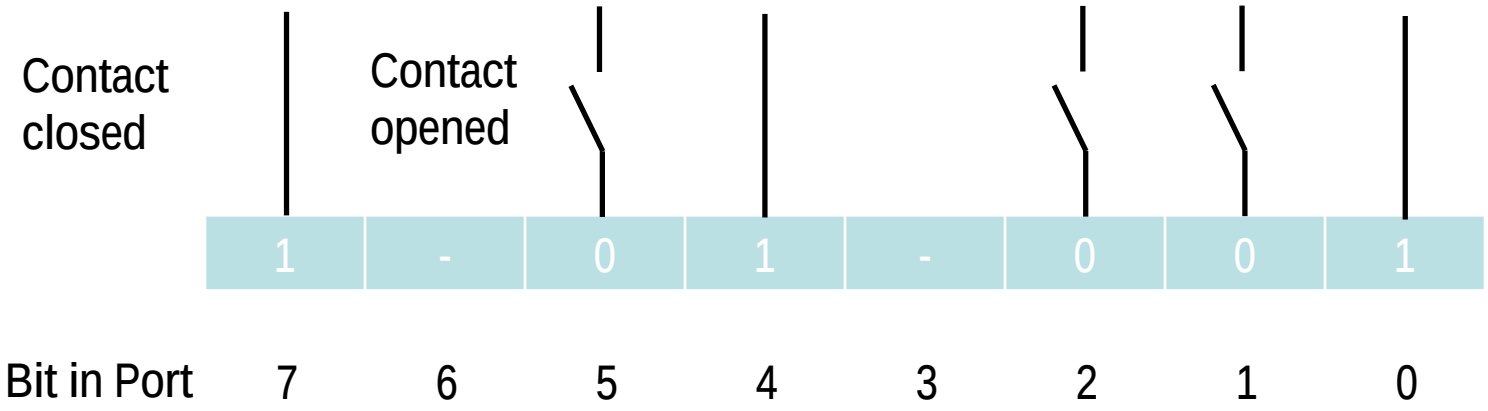
Digital input signals

Universität Stuttgart

- Input of single binary process signals
  - E.g. limit switch, manual switch
- Input of group of binary process signals
  - E.g. multistage coding switch
- Types of binary input signals:
  - Binary voltage levels (e.g. LOW = 0V, HIGH = 5V)
  - Binary current levels (e.g. LOW = 0mA, HIGH = 20mA)
- Temporal behaviour of signals
  - Static signal: HIGH and LOW are two detectable states of the signal
  - Dynamic signal: only slopes (falling or rising) of the signal are detected
  - Random signal invokes an interrupt

Represenation of digital signals

- Combination of single binary signals (Bits) to Ports (8/16/32 Bits); each bit of a port is accesible independently



- Representation by dual numbers or hexadecimal numbers

Digital output signals

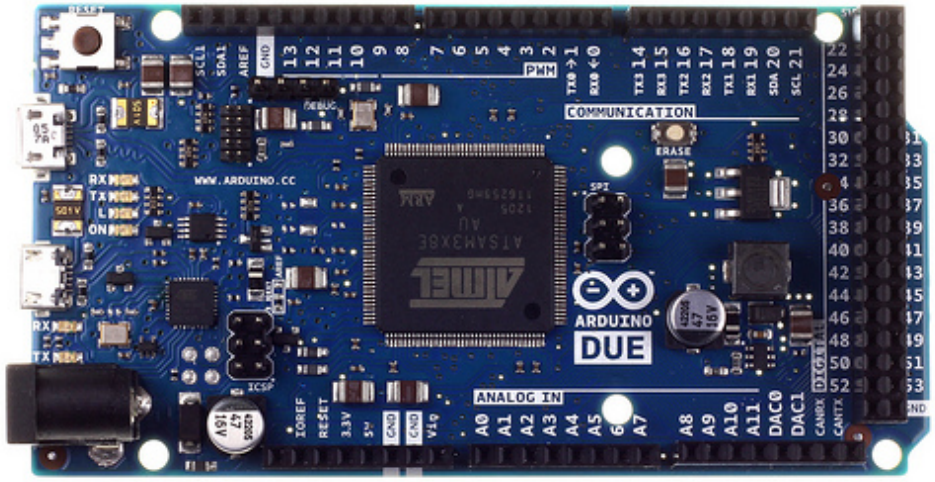
Universität Stuttgart

- Output of single binary signals
  - E.g. LED, relay
- Types of binary output signals:
  - Binary voltage levels (e.g. LOW = 0V, HIGH = 5V)
  - Binary current levels (e.g. LOW = 0mA, HIGH = 20mA)
- Temporal behaviour of signals
  - Normally static signal: HIGH and LOW are two detectable states of the signal
- Output signals are also organized in ports
- Output ports can only deliver very low power (mW !!!) → amplifier circuit

Digital I/O-system of microcontroller Arduino DUE

Universität Stuttgart

- A digital pin can be used as input or output pin (general purpose input output GPIO)
- Up to 103 I/O lines, with external interrupt capability (AT91SAM3X8E), only 54 digital I/O pins on Arduino board; rest for special functions (bus, communication)
- Up to six 32-bit parallel Input/Outputs (PIO)
- Digital I/O pins work with 3.3V; maximum output current 3 – 15 mA (depending on pin); maximum input current 6 – 9 mA.



Access of digital I/O-system of Arduino DUE

- Easy access to digital pins by core library functions

- Selection of a single pin as input or output:
  - `pinMode(pin, mode)`
  - pin: number of the pin whose mode should be set (integer)
  - mode: INPUT (default), OUTPUT, INPUT_PULLUP
    - INPUT: pin is in high-impedance state; very little current necessary for changing state
    - INPUT_PULLUP: an internal pullup resistor (100 kΩ) is activated
    - OUTPUT: pin is in low-impedance state; can provide current to other circuits (! Limited to 3–15 mA!)

- Write information to a digital pin:
  - `digitalWrite(pin, value)`
  - pin: number of the pin where the information should be sent (integer)
  - value: LOW, HIGH
    - LOW: pin is set to 0 V
    - HIGH: pin is set to 3.3 V; drives a current up to 3 – 15 mA.

Access of digital I/O-system of Arduino DUE

Universität Stuttgart

- Read state of a digital pin:
  - `Value = digitalRead(pin)`
  - pin: number of the pin which state should be read (integer)
  - value: return value of function: LOW, HIGH
    - LOW: pin senses 0 V
    - HIGH: pin senses 3.3 V
- Example: set pin 3 as input pin, read the information on pin 3 and write it to pin 13.

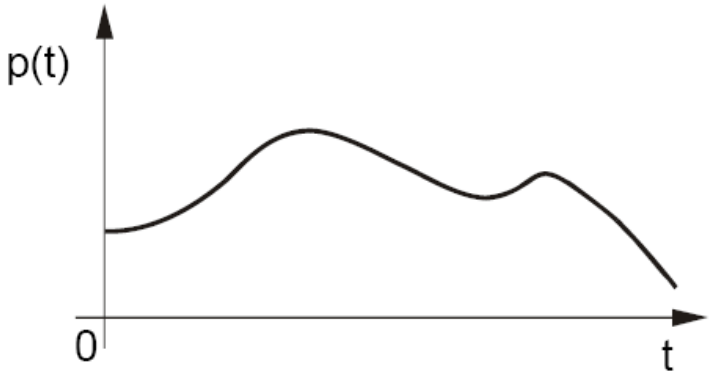
```
void setup()
{  pinMode(3, INPUT);
   pinMode(13, OUTPUT);
}
void loop()
{  int value;
   value = digitalRead(3);
   digitalWrite(13, value);
}
```

Universität Stuttgart

To be done

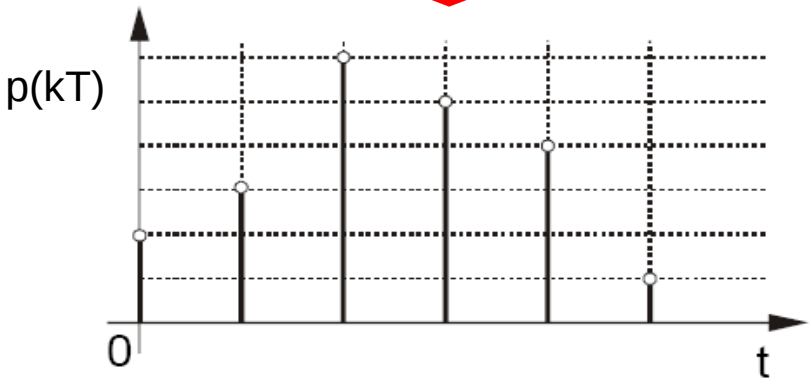
Universität Stuttgart

Analog process signals



time- and value continuous

ADC

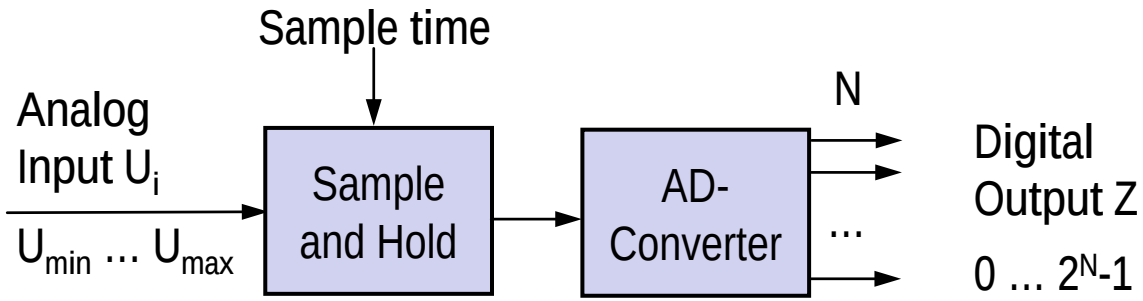


time- and valuediscret

- E.g. temperature, pressure, distance, revolution speed, …
- Analog process signals must be converted to a digital value before their use in a computersystem
- Use of analog-digital-converters (ADC)

Analog Digital Conversion

- The conversion from the analog to the digital signal takes some time: sampling time
- Sample time is limited → maximum sampling rate (1 MHz for Arduino Due)



$$Z = \frac{U_i - U_{min}}{U_{max} - U_{min}} \cdot (2^N - 1)$$

- The conversion accuracy is limited by the number of bits of the analog digital converter (12 bits for Arduino Due) → quantization

Quantization and sampling rate

- Sampling rate must be greater than twice the highest frequency of the signal (Shannon sampling theorem). Otherwise no reconstruction of signal from sampled values possible.

$$f_{Sampling} \geq 2 \cdot f_{Signal}$$

- Quantization: one quantization step is the size of the least significant bit of the analog digital converter:

$$U_{LSB} = \frac{U_{max} - U_{min}}{2^N - 1}$$

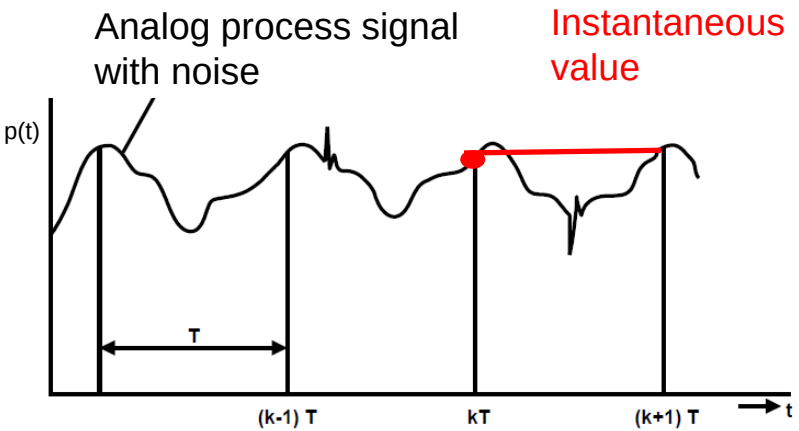
- Quantization error:

$$\Delta U_{error,max} = \frac{1}{2} \cdot U_{LSB}$$

Analog input signals

- Input of single analog process signals
  - E.g. temperature, pressure, revolution speed, …
- Types of analog input signals:
  - Analog voltage signals
  - Analog current signals
- Types of converters
  - Slow analog to digital converter: > 100 µs
  - Fast analog to digital converter: < 100 µs
  - Conversion of instantaneous value
  - Conversion of average value

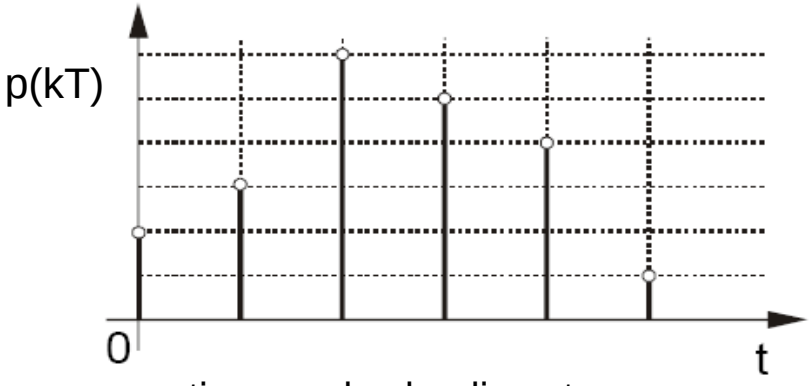
$$p(kT) = \frac{1}{T}\int_{(k-1)T}^{kT} p(t)dt$$



Types of converters

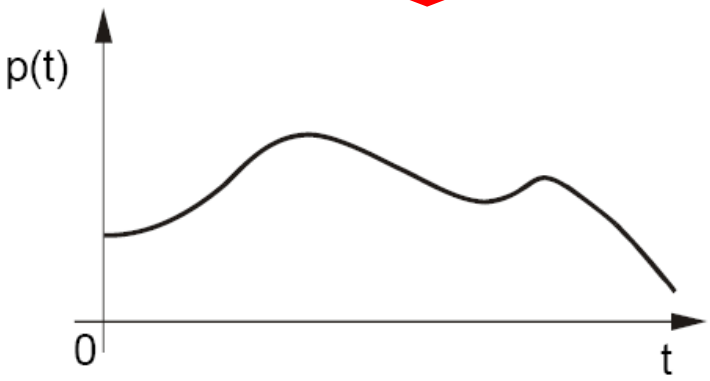
	Conversion of instantaneous value	Conversion of average value
Advantage	High conversion speed (10^4 … 10^8 values/s	High interference supression
Disadvantage	Noise falsifies digital values	Low conversion speed
Method of conversion	Counting, steps, direct	Dual-slope, voltage-frequency

Analog output signals



time- and valuediscret

DAC

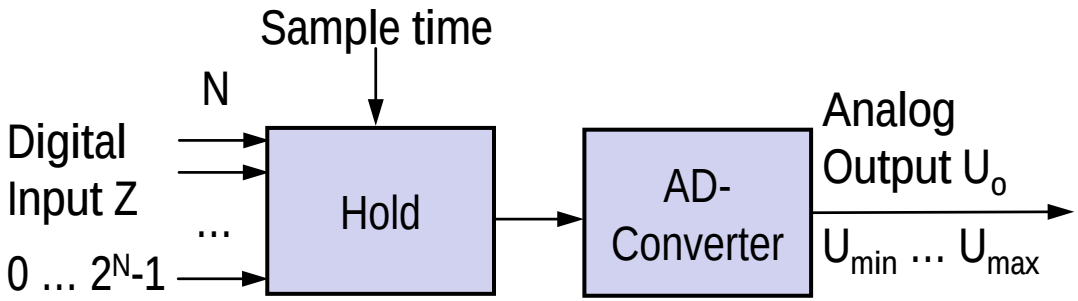


time- and value continuous

- E.g. speed, revolution speed of a motor, …
- Digital representation must be converted to an analog signal
- Use of digital-analog-converters (DAC)
- Not every microcontroller has an onboard DAC → pulse width modulation

Digital Analog Conversion

- The digital analog conversion works opposite to the digital analog conversion

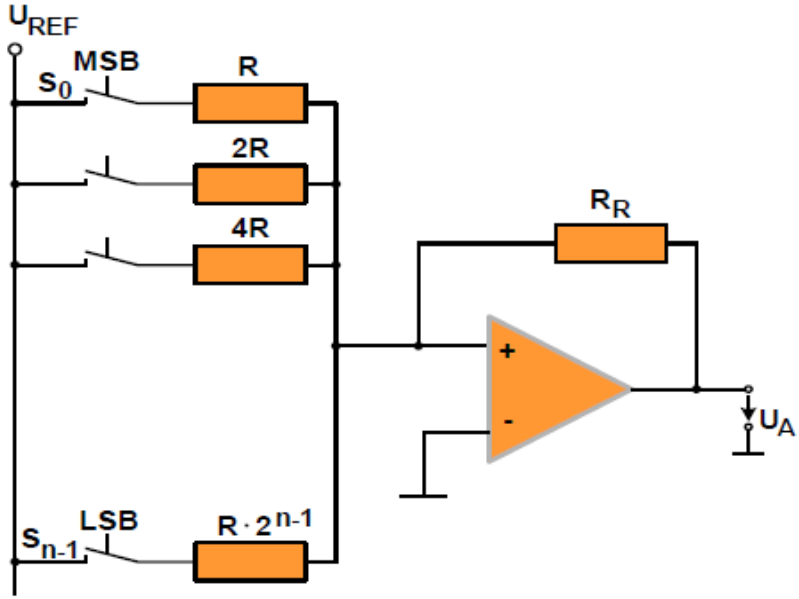


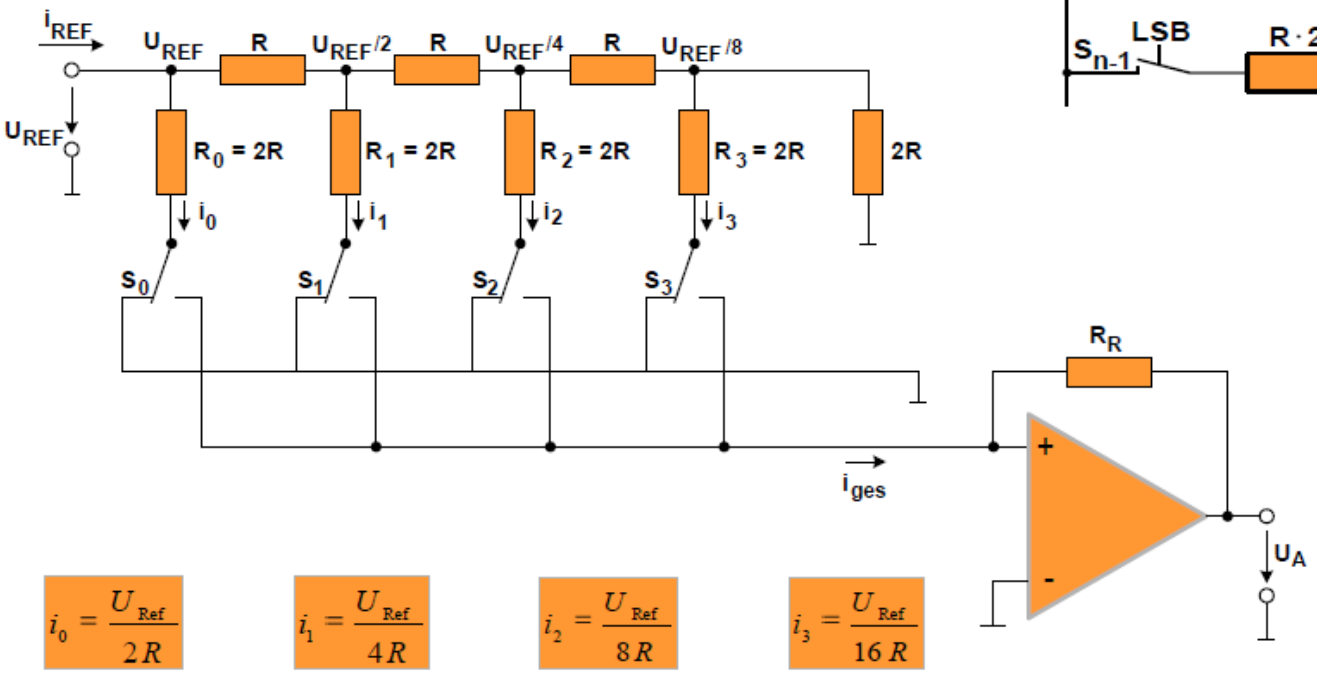
$$U_O = \frac{Z}{2^N - 1} \cdot (U_{max} - U_{min}) + U_{min}$$

- The conversion accuracy is limited by the number of bits of the digital analog converter (12 bits for Arduino Due)

Method of Digital Analog Conversion

- Use of networks of resistors
  - Sum of weighted current
  - Ladder network of resistors



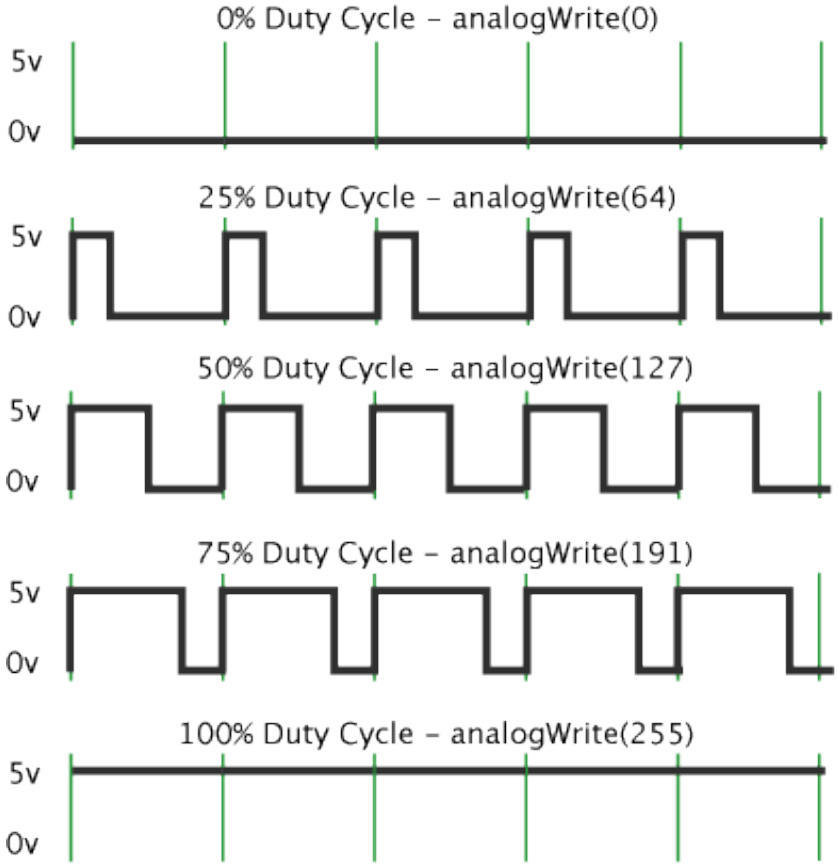


$$i_0 = \frac{U_{Ref}}{2R} \qquad i_1 = \frac{U_{Ref}}{4R} \qquad i_2 = \frac{U_{Ref}}{8R} \qquad i_3 = \frac{U_{Ref}}{16R}$$

Source: IAS, University Stuttgart

Analog output by pulse-width-modulation

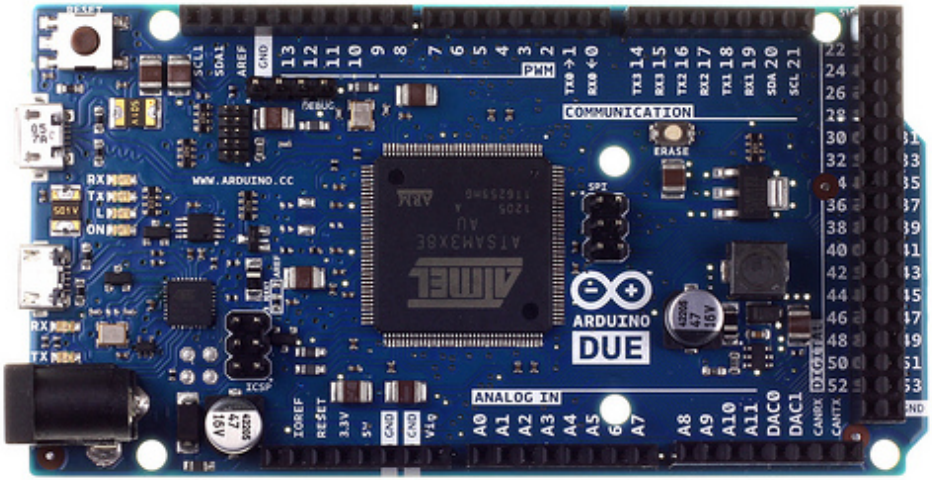
- Only few microcontroller have a real DAC onboard
- Virtual generation of an „analog signal“ by pulse-width-modulation



- Smoothing of signal by a low-pass-filter (RC) or directly by a motor or LED (mechanical/optical inertia)

Analog I/O-system of microcontroller Arduino DUE

- Three types of pins for analog signals:
  - Analog input pins A0 – A11
  - Analog output, real analog values (DAC, 12 bit), pins DAC0 – DAC1
  - Analog output, 12 bit pulse-width-modulated, digital pins 2 - 13
- Analog I/O pins work with 3.3V; maximum output current 3 – 15 mA (depending on pin); maximum input current 6 – 9 mA.



Access of analog I/O-system of Arduino DUE

Universität Stuttgart

- Easy access to analog pins by core library functions

- No input/output selection for the „real“ analog pins. Pinmode must be set to output for the PWM-pins:
  - `pinMode(pin, OUTPUT)`
  - pin: number of the pin whose mode should be set (integer)

- Write information to an analog pin:
  - `analogWrite(pin, value)`
  - pin: number of the pin where the information should be sent (integer)
  - value: 0…255 (4095)
    - 0: duty cycle is 0%
    - 255 (4095): duty cycle is 100%; drives a current up to 3 – 15 mA.

- Set analog write resolution:
  - `analogWriteResolution(bit)`
  - bit: number of bits for conversion (8 or 12)

Access of analog I/O-system of Arduino DUE

Universität Stuttgart

- Read value of an analog pin:
  - `Value = analogRead(pin)`
  - pin: number of the pin which state should be read (integer)
  - value: return value of function: 0 … 1023 (4095)
    - 0: pin senses 0 V
    - 1023 (4095): pin senses 3.3 V
- Set analog read resolution:
  - `analogReadResolution(bit)`
  - bit: number of bits for conversion (10 or 12), 10 is default setting

- Example: read analog pin 0 and print the value to the Serial Monitor.

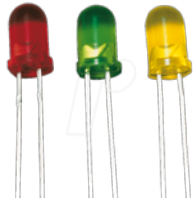
```
void setup()
{  Serial.begin(9600);
}
void loop()
{  int value;
   value = analogRead(0);
   Serial.println(value);
}
```

Universität Stuttgart

Conncection to IO-pins of microcontrollers

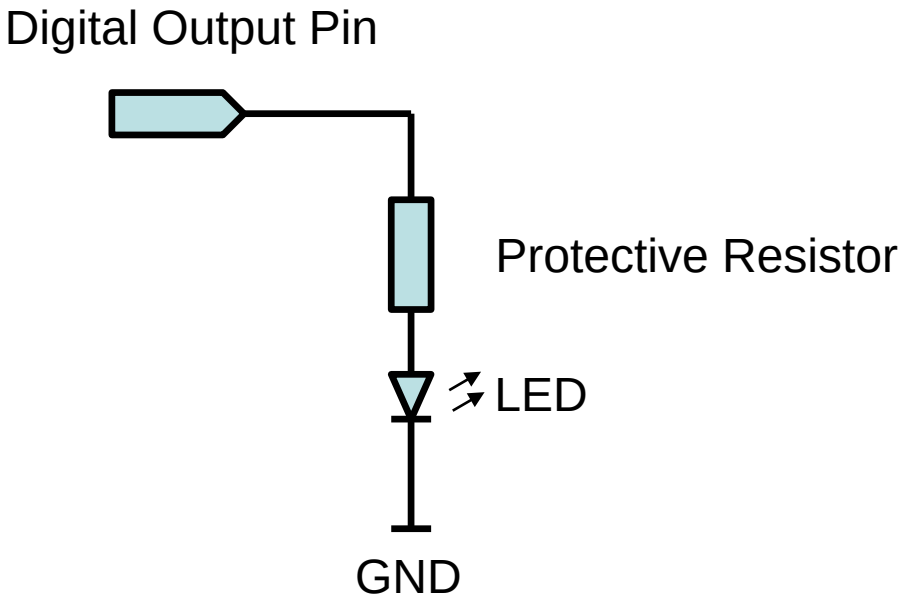
- Digital and Analog I/O pins work with 3.3V; maximum output current 3 – 15 mA (depending on pin); maximum input current 6 – 9 mA → no consumer with considerable power possible
  !! Microprocessor will be damaged if overloaded !!
- Amplifier- and protection circuits are necessary
- Types of IO-pins
  - Digital Input-Output pin
  - Analog Input pin
  - Analog Output pin with pulse width modulation
  - Analog Output pin with digital analog converter

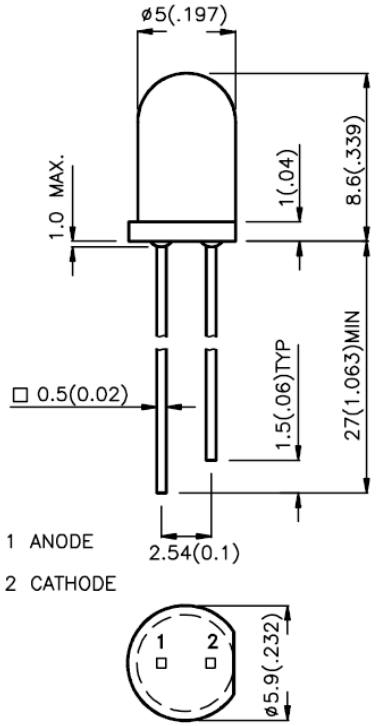
Direct connection to digital output pin



- Maximum output current 3 – 15 mA (depending on pin) at 3.3 V
- Direct connection of a low current LED (e.g. Kingbright L-53L)

- From datasheet: I_F = 2mA, V_F = 1.7 V (red), 1.8 V (yellow), 1.9 V (green)
- Protection resistor necessary.

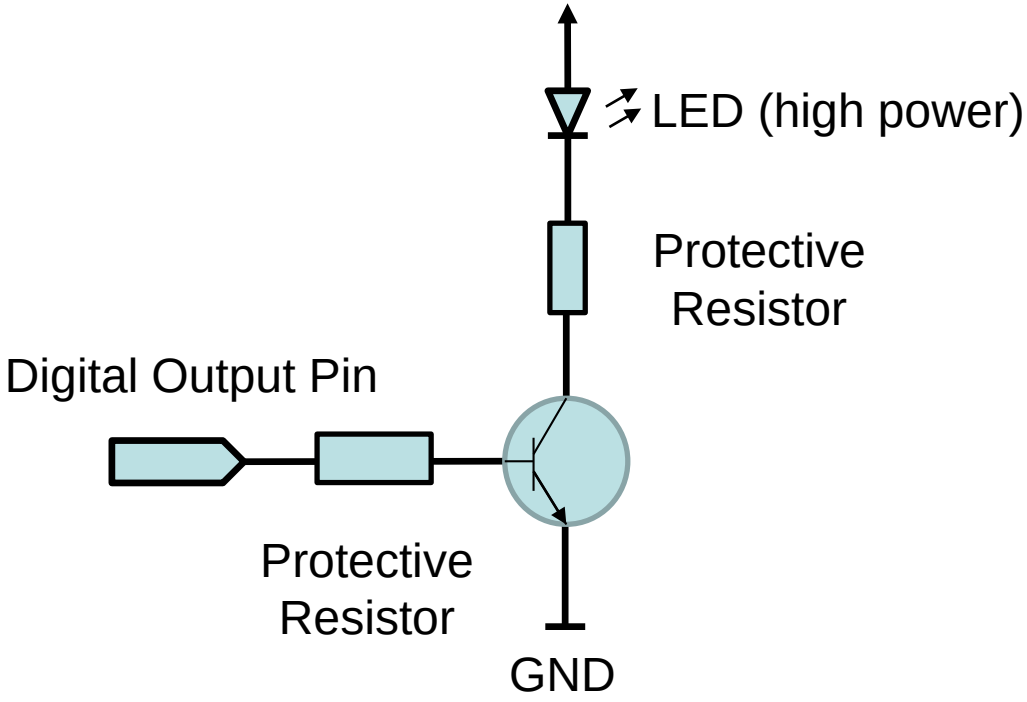


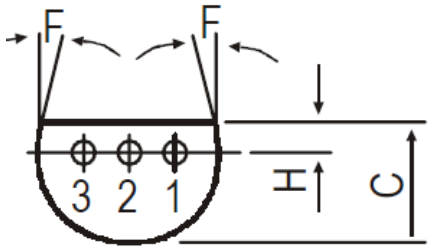


Connection to digital output pin by transistor

- To gain current / voltage use a low power transistor (e.g. 2N3904)

- Transistor used as electronic switch
- Protection resistor for digital pin and transisotor (I_{Cmax}) necessary.

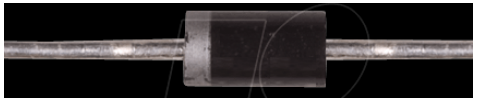




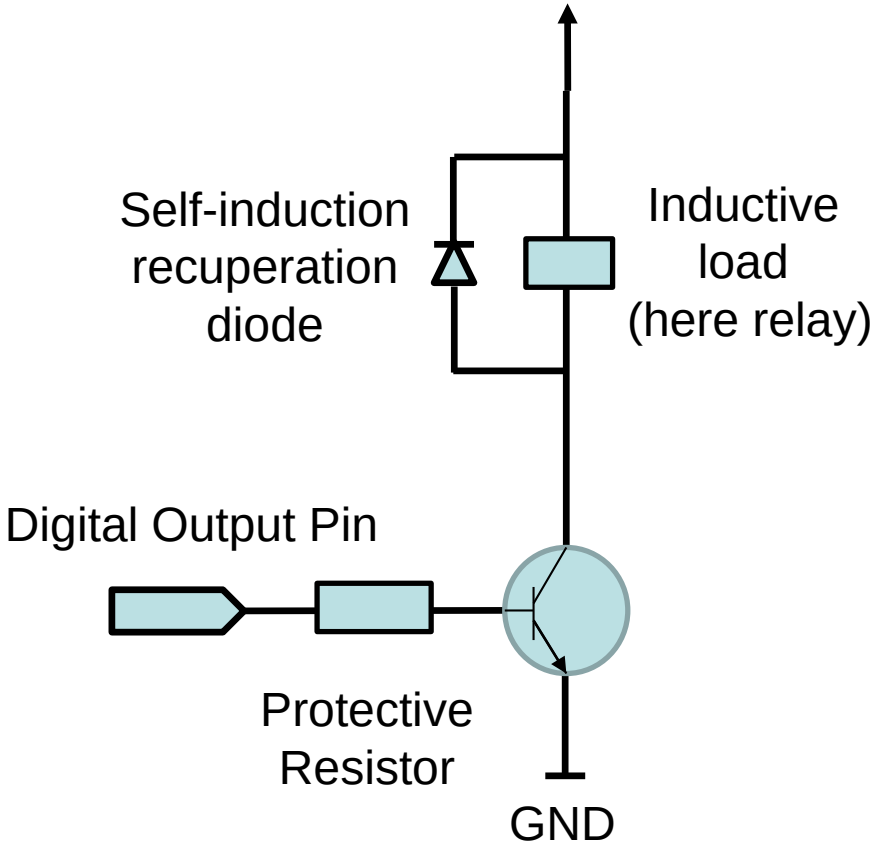
PIN CONFIGURATION
1. EMITTER
2. BASE
3. COLLECTOR

Connection to digital output – inductive load

- If an inductive load (motor, relay) is switched off an induced voltage appears → danger of spark generation if energy is not dissipated



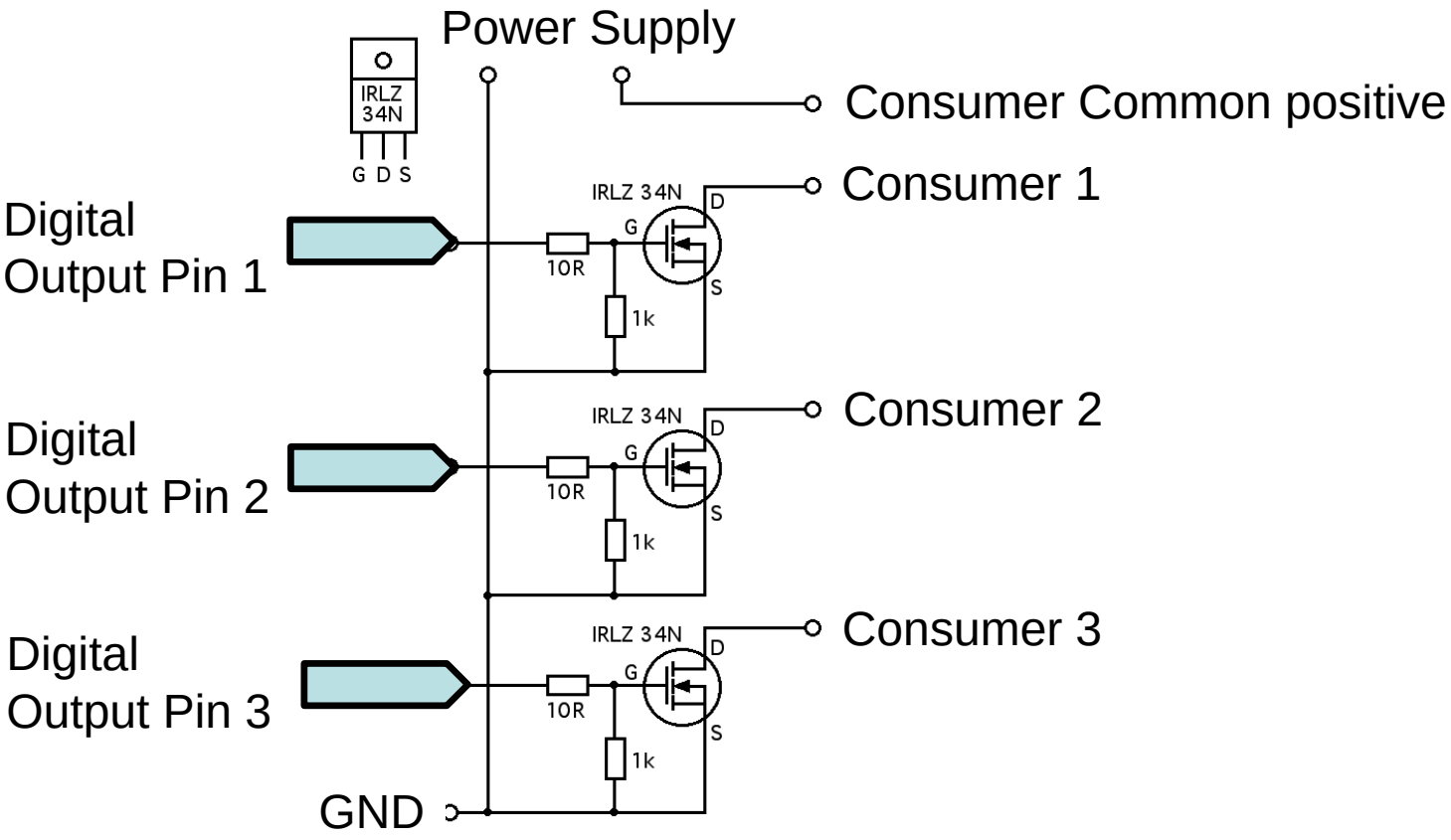
- Use of a self-induction recuperation diode (e.g. 1N4001)

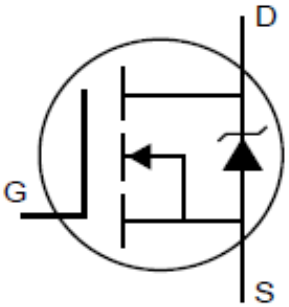


Connection to digital output pin by power mosfet

Universität Stuttgart

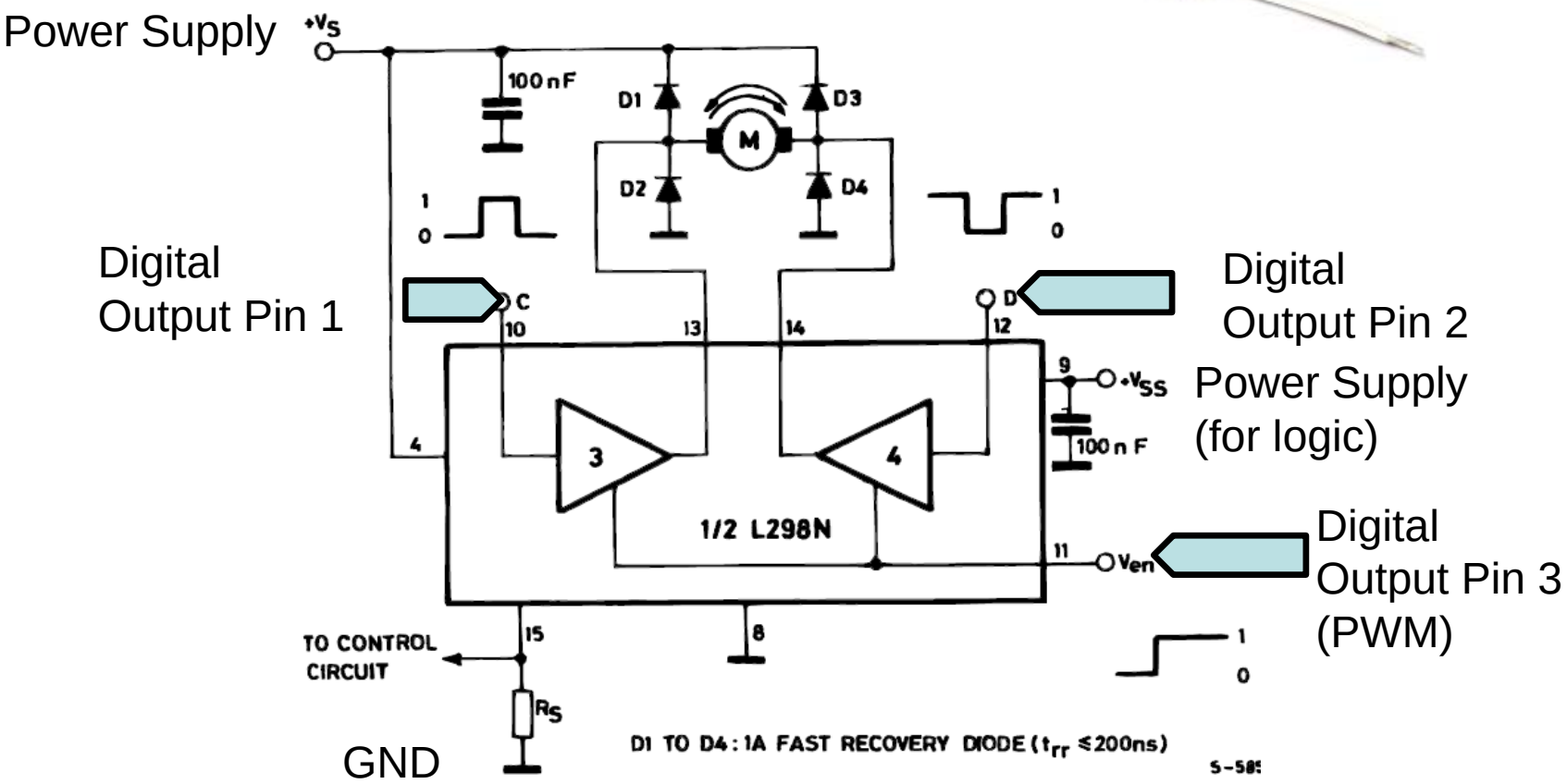
- To gain current (>1A) use a power mosfet (e.g. IRLZ34N)
- Power mosfet used as electronic switch
- Protection resistor for digital pin and pull down resistor for safe switch off.





Connection to digital output pin – DC motor

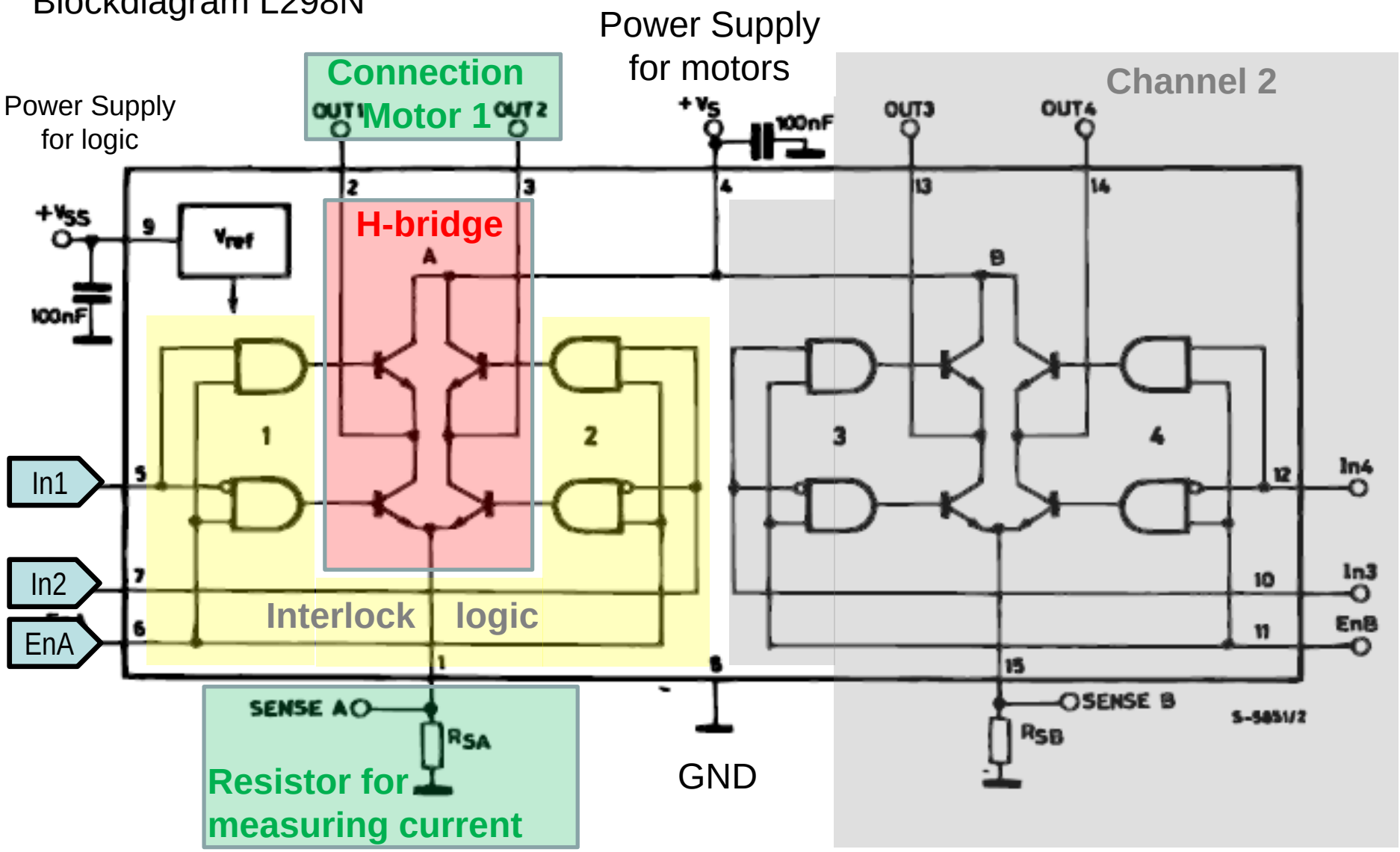
- DC-motor can drive in two directions with variable revolution
- Connection to IO-system by motorcontroller (e.g. L298N, two motors, up to 4A)



Connection to digital output pin – DC motor

Universität Stuttgart

Blockdiagram L298N



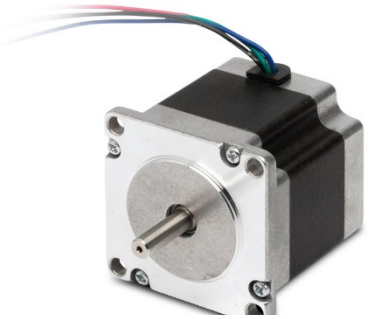
Connection to digital output pin – DC motor

- Example for controlling DC-motor

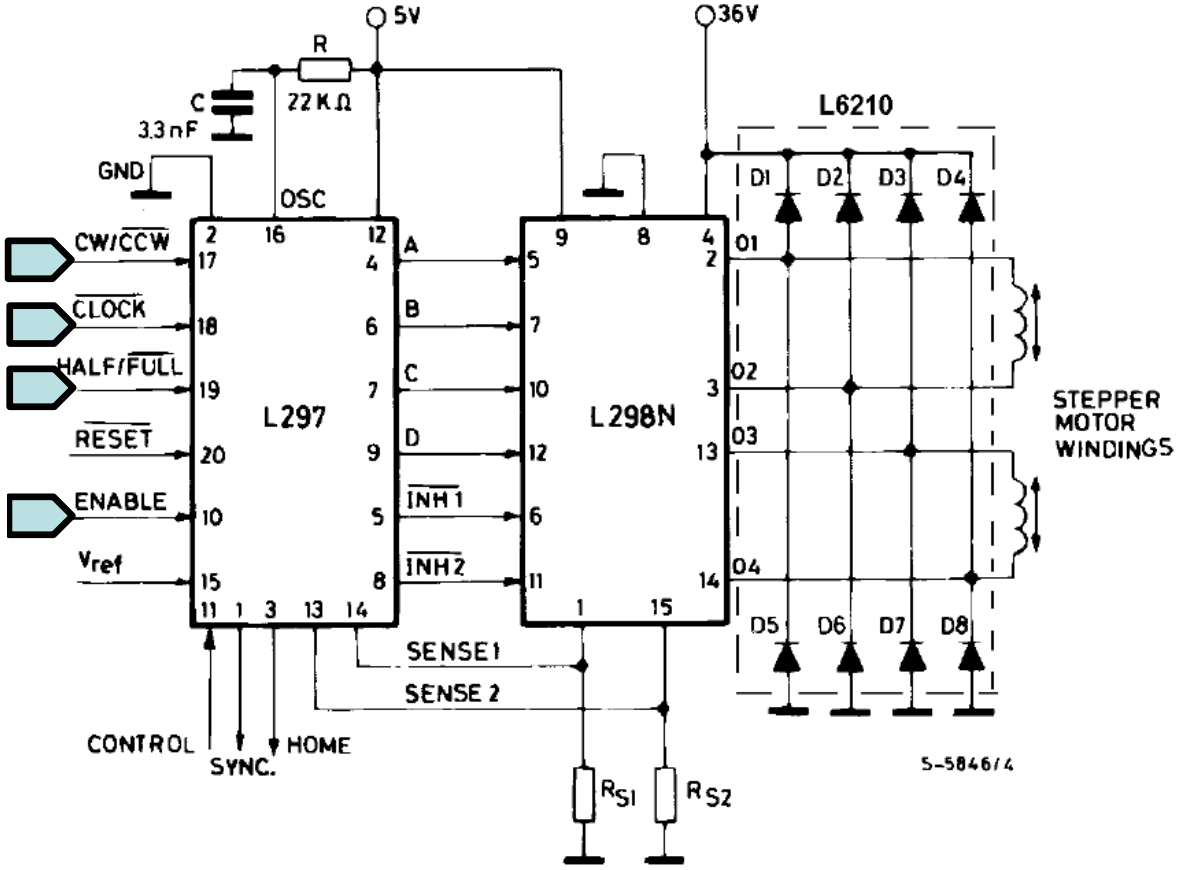
```
void setup()
{  pinMode(3, OUTPUT);  //EnA signal
   pinMode(30, OUTPUT); //In1 signal
   pinMode(31, OUTPUT); //In2 signal
}
void loop()
{  int value;
   digitalWrite(30, HIGH);
   digitalwrite(31, LOW); //H,L : Motor forward
   analogWrite(3, 122);   //PWM 50%
   value = analogRead(3);
   if (value > 150)
   {   analogwrite(3, 0); // switch of motor if overcurrent
       delay(1000);       // wait for 1 second
   }
}
```

Connection to digital output pin – stepper motor

- stepper-motor moves step by step with each slope
- Easy method for positioning an axis (e.g. conveyor belt)
- Connection to IO-system by motorcontroller combination (e.g. L297 and L298N, motor up to 2A, 2 phase bipolar stepper)

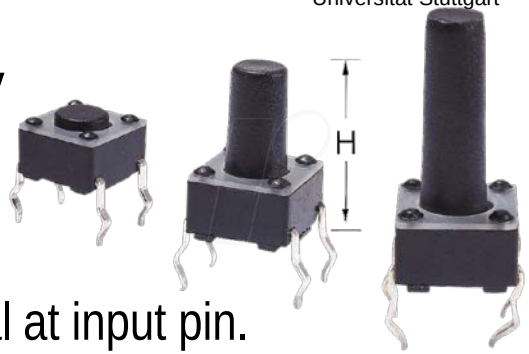


Enable: must be high for stepping operation
Vref: defines peak load current by voltage divider
CW/CCW: Clockwise/counter-clockwise direction of rotation
CLOCK: an active low pulse advances one increment. The step occurs on the rising edge
HALF/FULL: when high a half step is done; when low a full step is performed

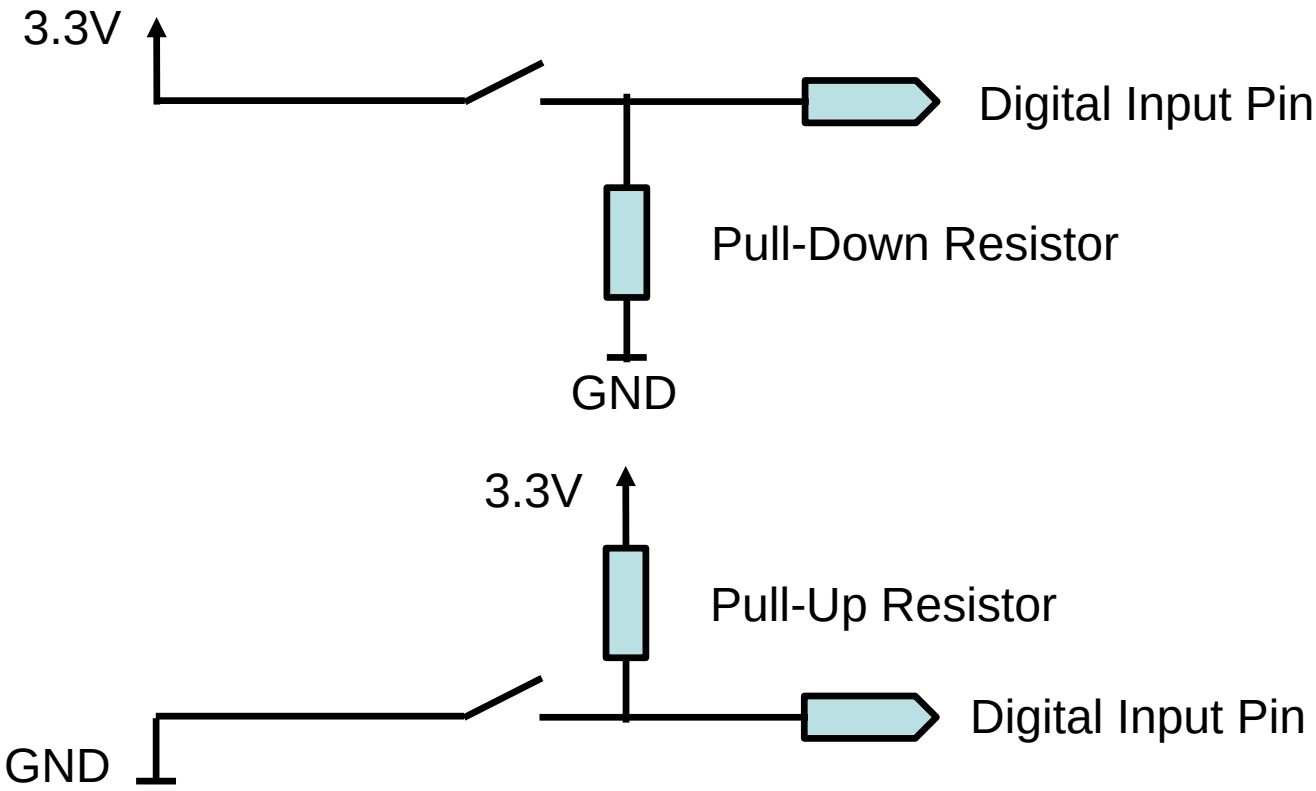


Direct connection to digital input pin

- Maximum input current 6 – 9 mA (depending on pin) at 3.3 V
- Direct connection of a push button (e.g. Wentronic TS 6)

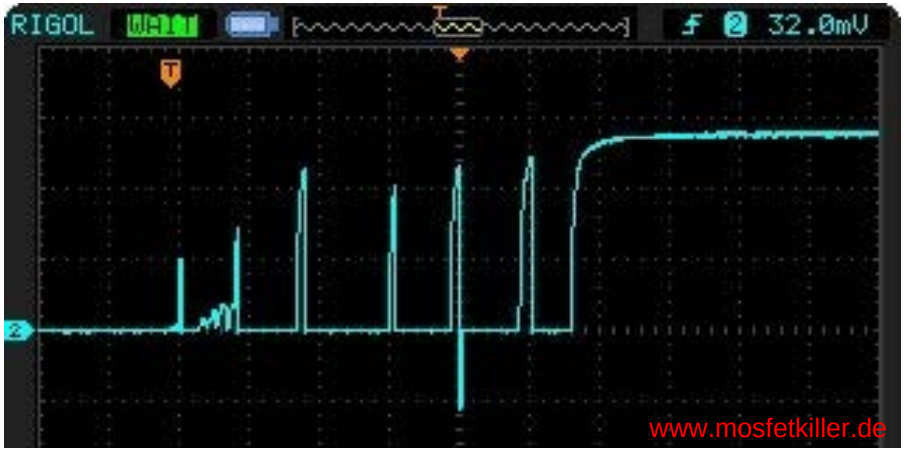


- Pull-up- or pull-down resistor necessary, for defined potential at input pin.

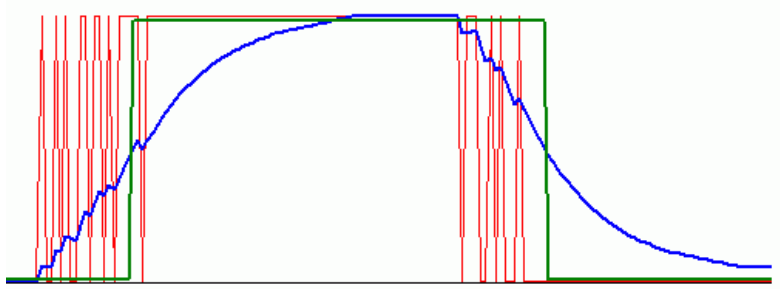


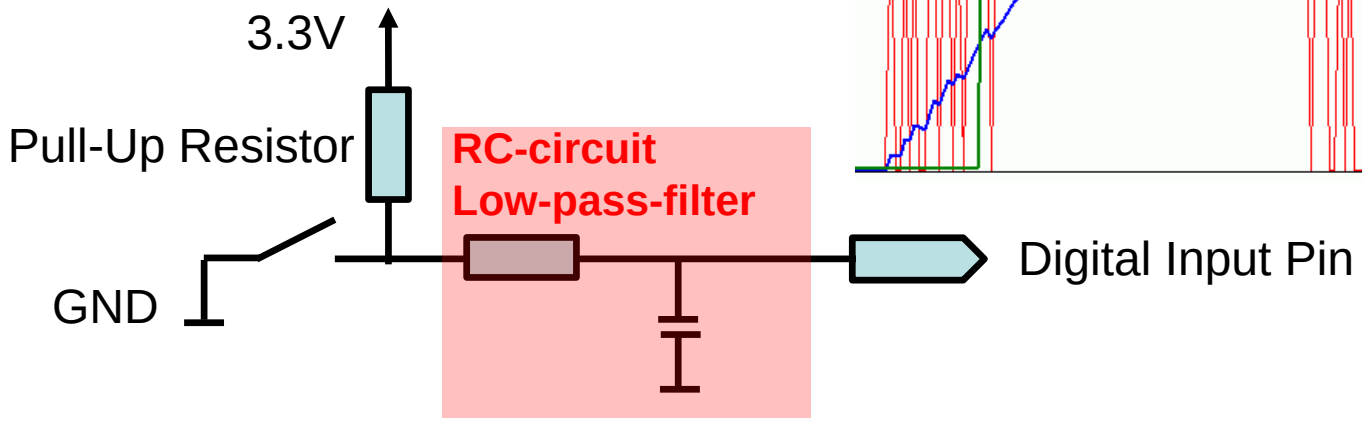
Direct connection to digital input pin

- Mechanical buttons bounce when pushed → multiple detection of ‚1‘- and ‚0‘-signals



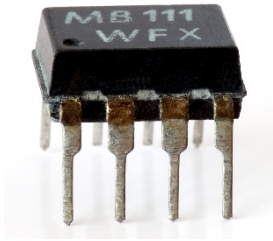
- Hardware or software filtering

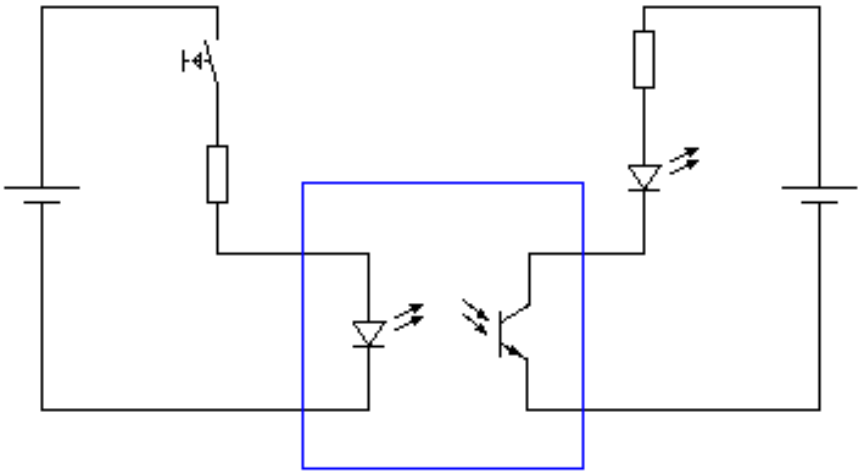


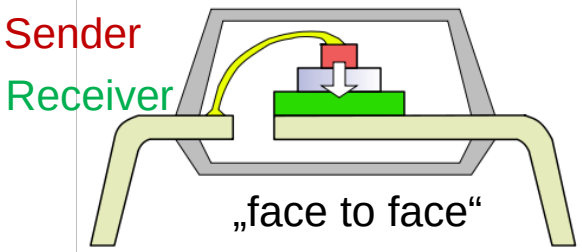


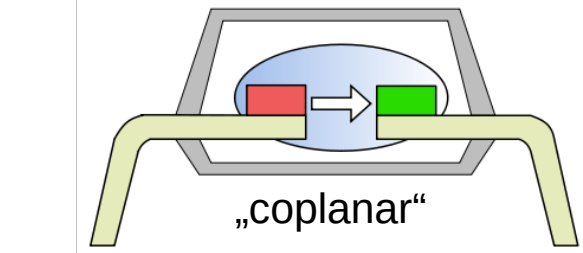
Galvanic isolation of digital pins

- Use of optoelectronic couplers (e.g. 4N38A, SFH618A-3)
  - To protect the digital pins of the microcontroller
  - To shift voltage niveaus
  - For galvanic isolation of two circuits
- Structure: sending LED, receiving photo-transistor → signal is transmitted by light





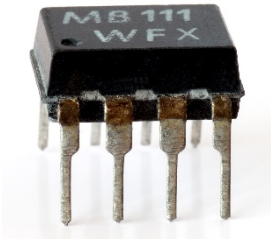


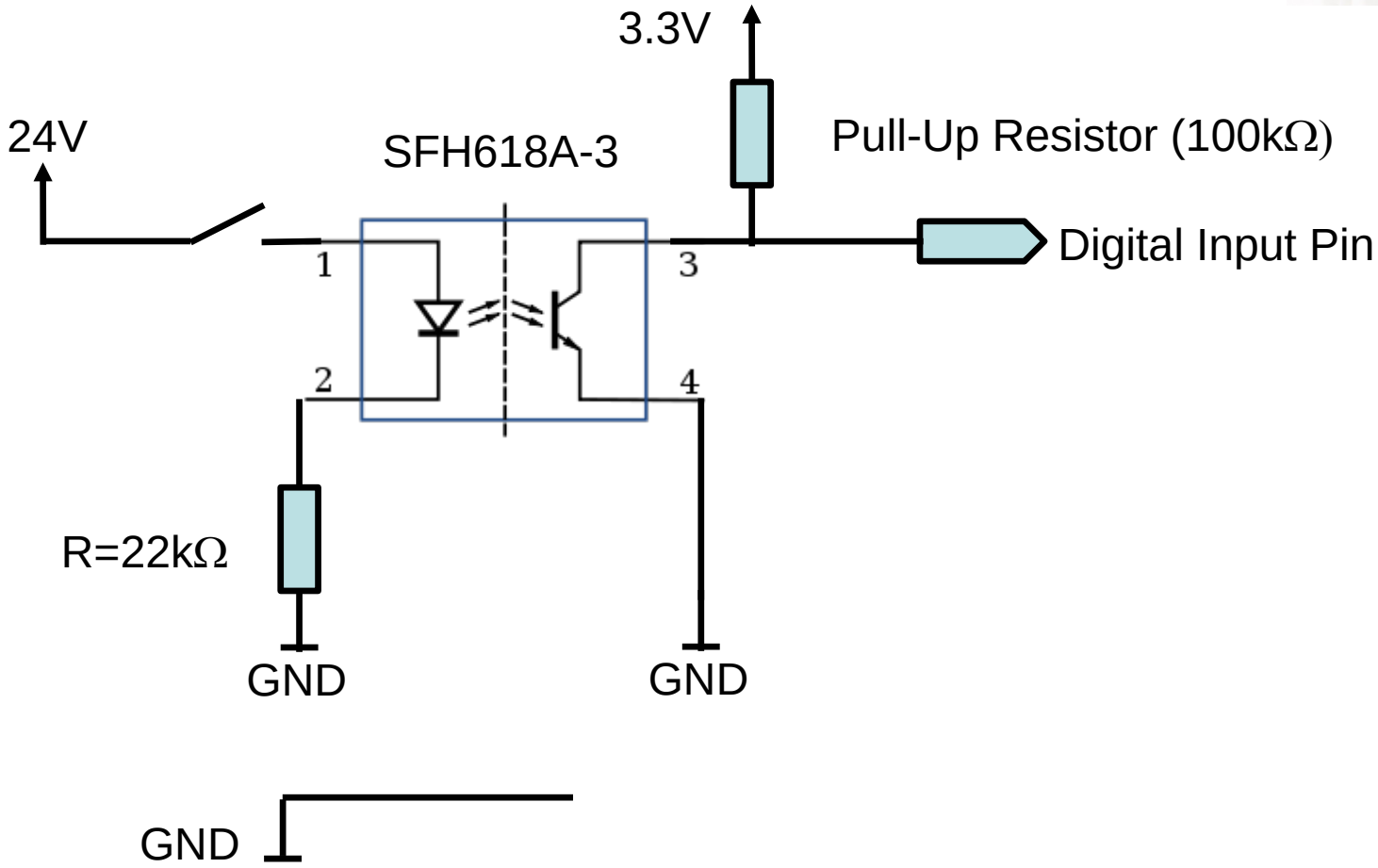


Wikipedia.org

Galvanic isolation of digital pins - example

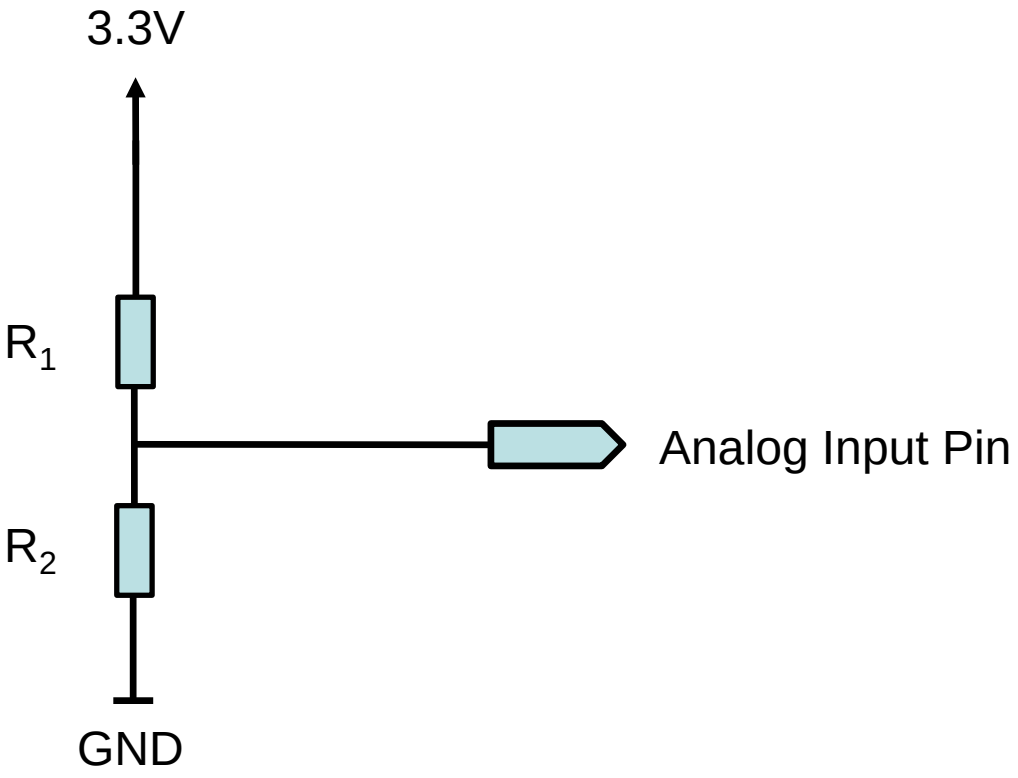
- Use of optoelectronic coupler SFH618A-3
- Read a 24V-signal



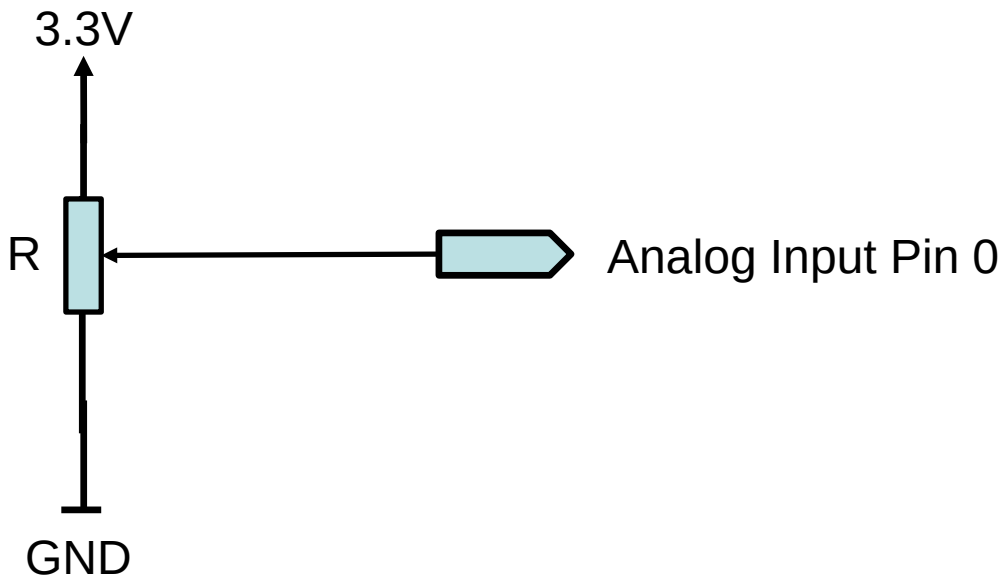


Connection to analog input pin

- Analog input pins sense a voltage between 0 – 3.3 volts.
- Use principle of a voltage divider



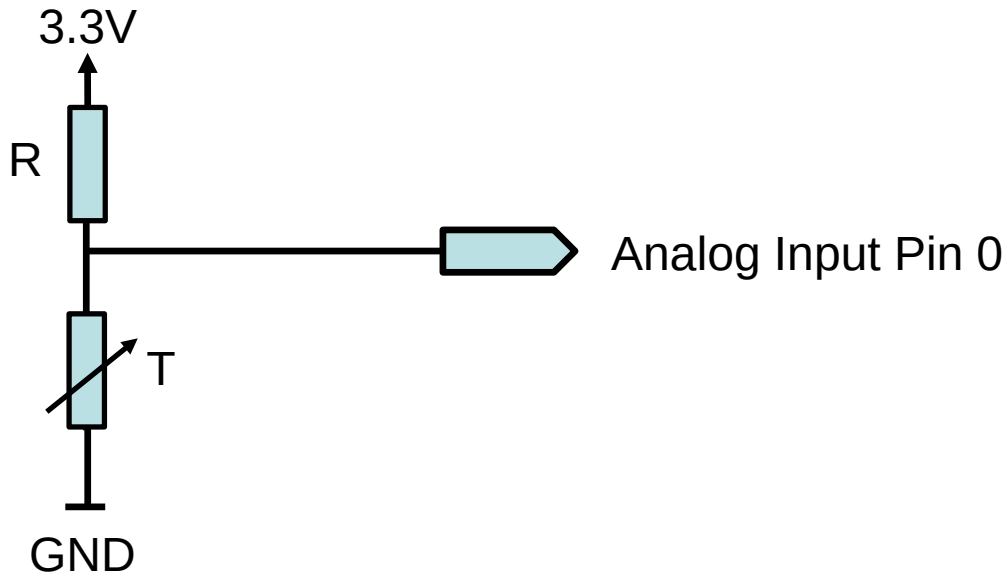
Connection to analog input pin – potentiometer example




```
void setup()
{  Serial.begin(9600);
}
void loop()
{  int value;
   value = analogRead(0);
   Serial.println(value);
   delay(200); // wait 200ms
}
```

Connection to analog input pin – temperature sensor example

- Sensor KTY81-110 – cheap (~0,50€), easy to use
- Positive Temperature Coefficient (PTC)
- From datasheet: I_{cont} = 1mA, R_{10} = 886Ω, R_{50}=1209Ω, R_{90}=1591Ω
- Calculation of R



Connection to analog input pin – light dependent resistor

- Sensor LDR03 – cheap (~1,50€), easy to use
- From datasheet: R_{01Lux} = 180kΩ, R_{10Lux}=20kΩ, R_{100Lux}=5kΩ

- Calculate the value of the resistor R serial to the LDR.
- Draw the electrical circuit: The darker the light, the higher the analog value

Connection to analog output pin (DAC) – DC-Motor

Universität Stuttgart

- Use an analog output pin (DAC) to control the rotation speed of a DC-motor. Only one direction of revolution is necessary. The DC-motor is driven by 24V and a maximum current of 10A

- Draw the electrical circuit! Consider the motor as inductive load, ist high currents and voltages and a galvanic isolation. To control the motor speed a potentiometer is used.
- Write a sketch for the control task.

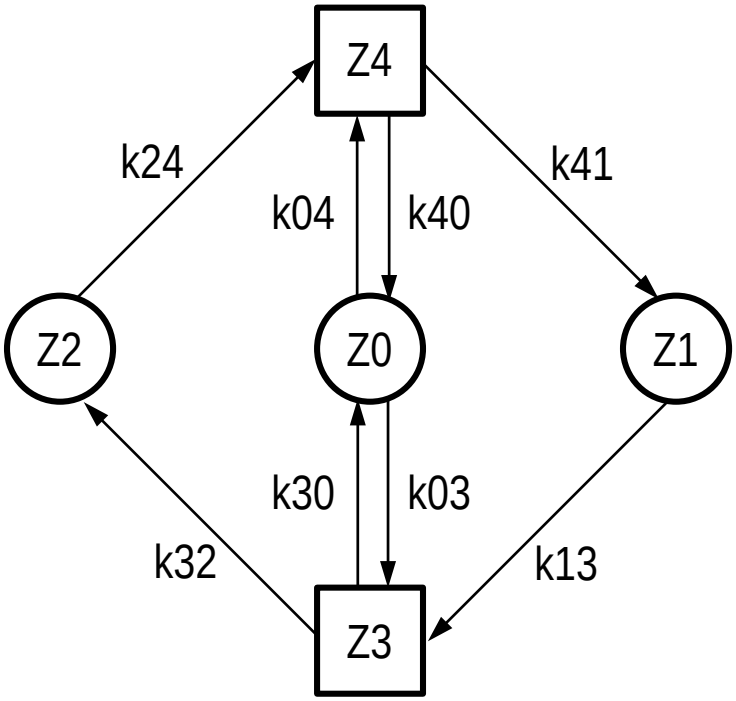
Universität Stuttgart

State Machines

Universität Stuttgart

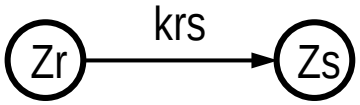
- State machines are an integrated way of describing control tasks (beginning with raw specification down to the programmers level)
- Description of functions based on a modularization of machines
- Step by step detailling, top down.
  - Functional groups (e.g. turning machine)
  - Functional unit (e.g. tool clamping)
- State machines support in
  - Testing, Ramp up
  - Diagnosis
  - Visualisation
- Modelling of parallel sequences by coordination of multiple state machines with messages
- Good for modelling of mechanical movements and processes, not the best for mathematical calculations and closed loop control

State Machines



Variable of state machine:
statevariable Z (0 ≤ Z ≤ n)

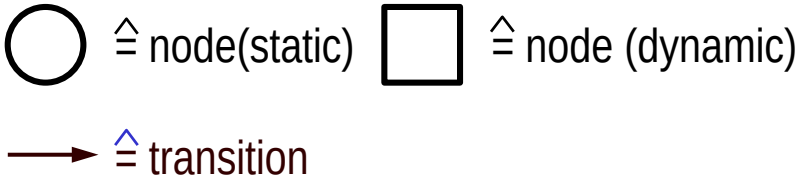
Transition condition krs
- binary variable
- logical combination of binary variables



Requirements:

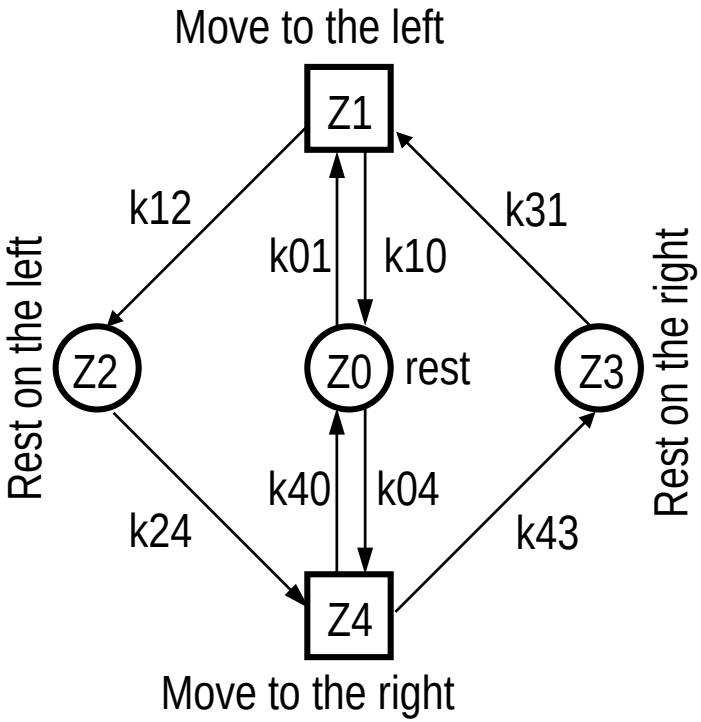
For each state only one transition condition can be true at the same time.
For each state machine only one state can be active at the same time.

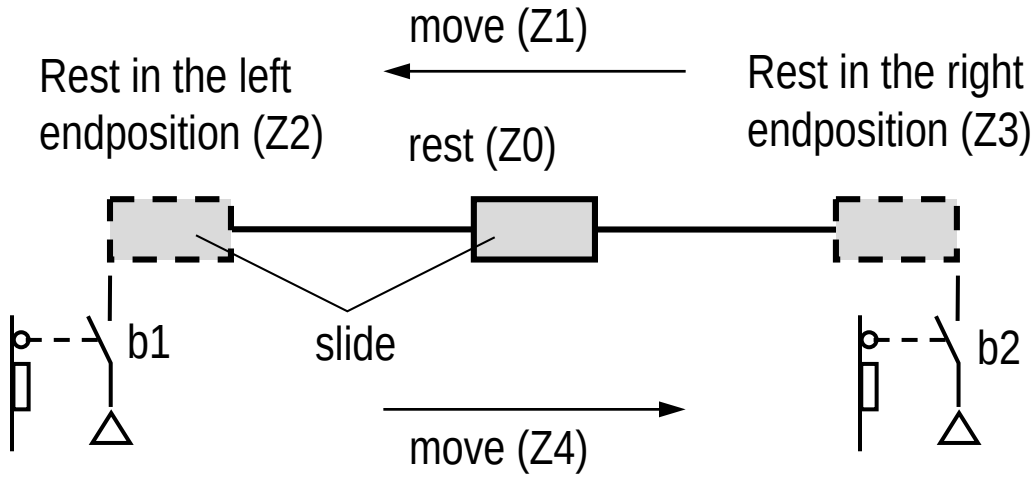
Graphical elements of a state machine:



source: Pritschow – Einführung in die Steuerungstechnik

Example of a state machine





Z ... state
k ... transition condition

Example of state machine – transition conditions and actions

Universität Stuttgart

Name of variable	Type	Datatype	Description
Drive_left	INPUT	BOOL	Input signal „Drive left“, e.g. with a push button
Drive_right	INPUT	BOOL	Input signal „Drive right“, e.g. with a push button
End_left	INPUT	BOOL	Signal „at the left end“, e.g. by a mechanical button
End_right	INPUT	BOOL	Signal „at the right end“, e.g. by a mechanical button
Motor_left	OUTPUT	BOOL	Output signal „Motor drive left“
Motor right	OUTPUT	BOOL	Output signal „Motor drive right“

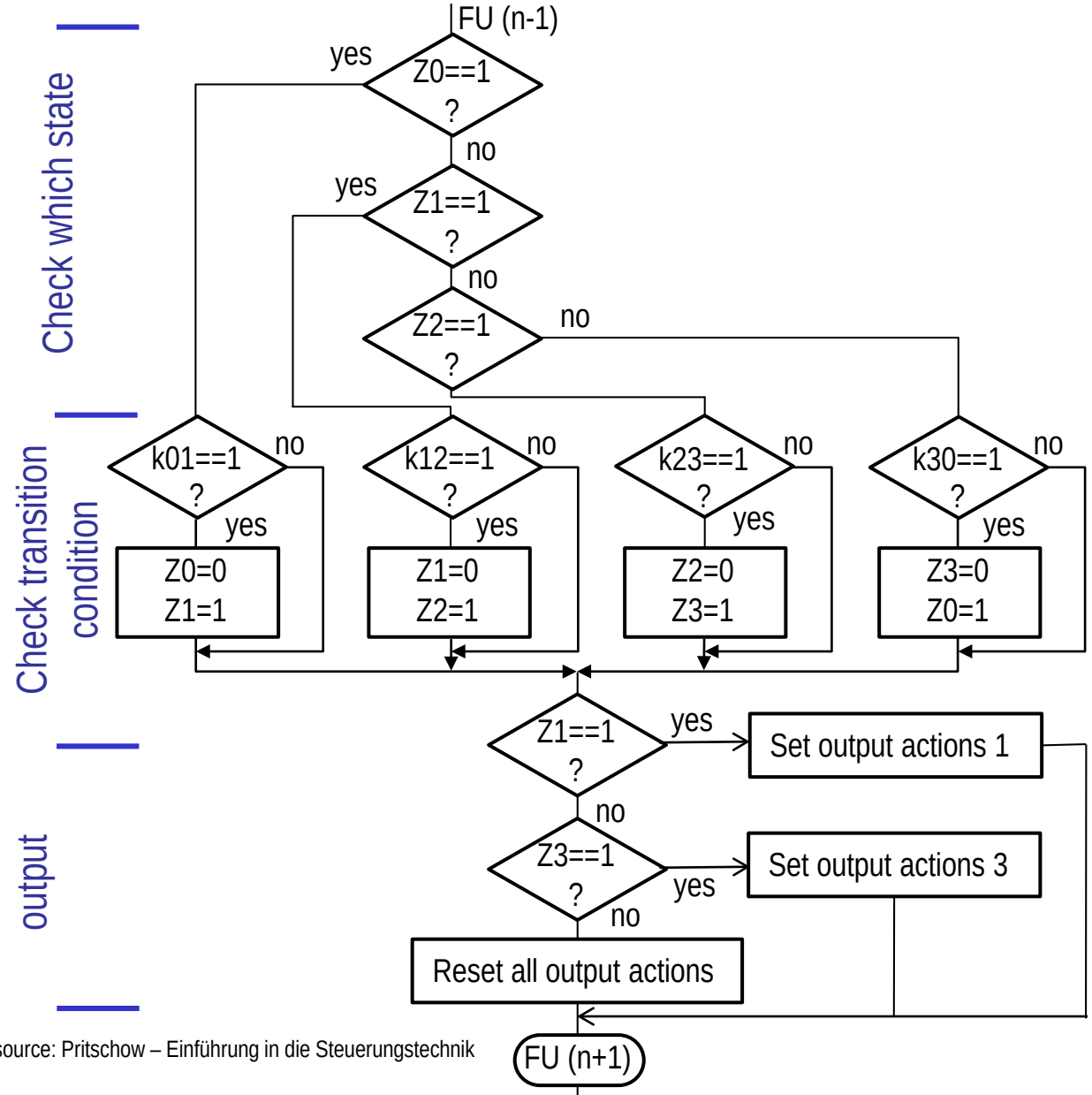
Transition conditions:

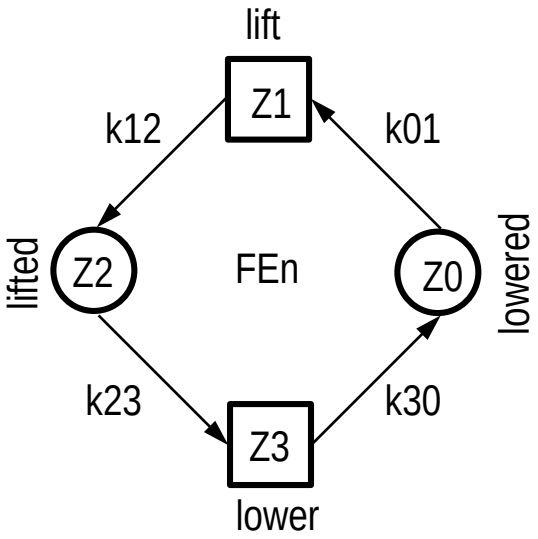
k31:	Drive_left == TRUE
k01:	Drive_left == TRUE
k10:	Drive_left == TRUE
k12:	End_left == TRUE
k24:	Drive_right == TRUE
k04:	Drive_right == TRUE
k40:	Drive_right == TRUE
k43:	End_right ==TRUE

Actions in single states:

Z1:	Action when entering	Motor_left = TRUE
	Action when leaving	Motor_left = FALSE
Z4:	Action when entering	Motor_right = TRUE
	Action when leaving	Motor_right = FALSE

Translation of a state machine to a programming language





Explanation:

FU … Functional unit

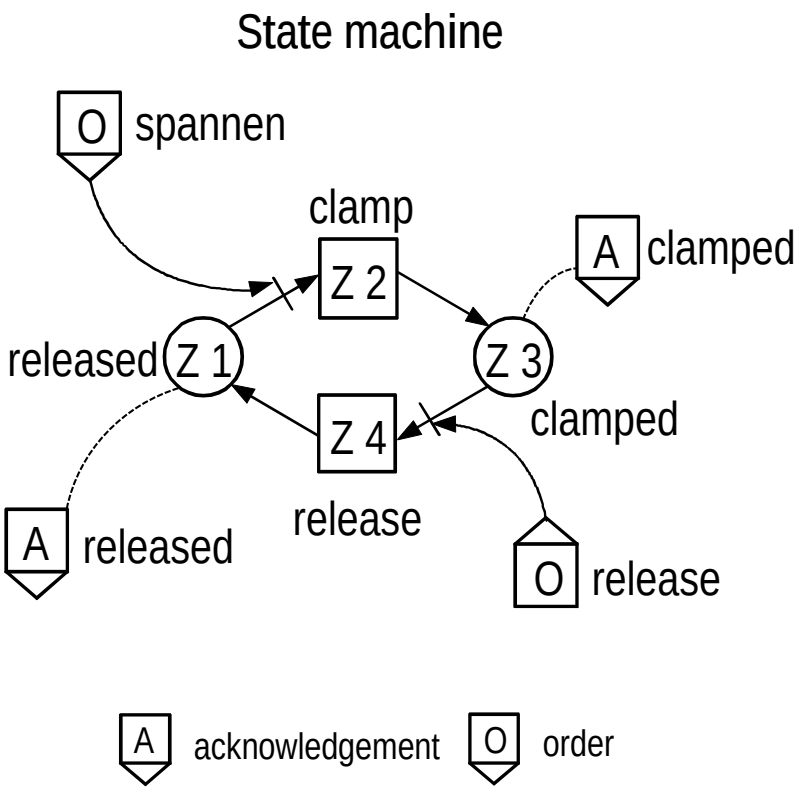
Z … State

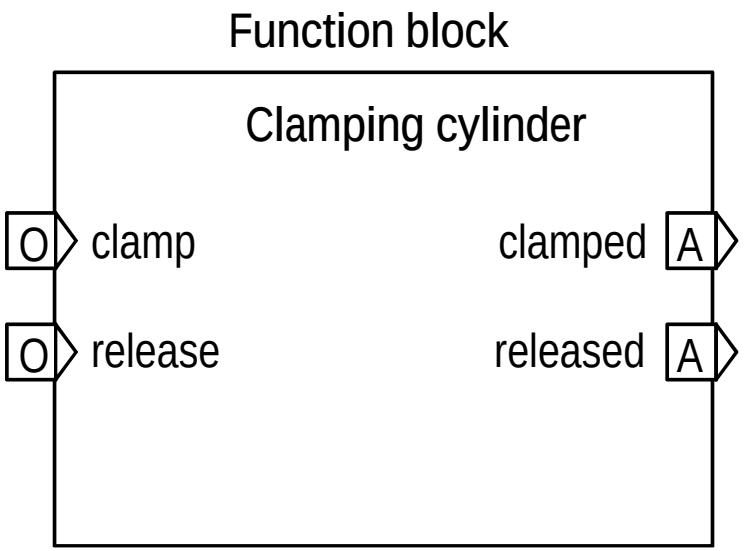
k … Transition condition

source: Pritschow – Einführung in die Steuerungstechnik

State machines and function blocks

- Encapsulation of completed subfunctionalities of a machine / system to functional groups, functional units.
- Assignment of associated sensors and actuators to functional units; Use of symbolic signal addresses.
- Implement a state graph in a function block.



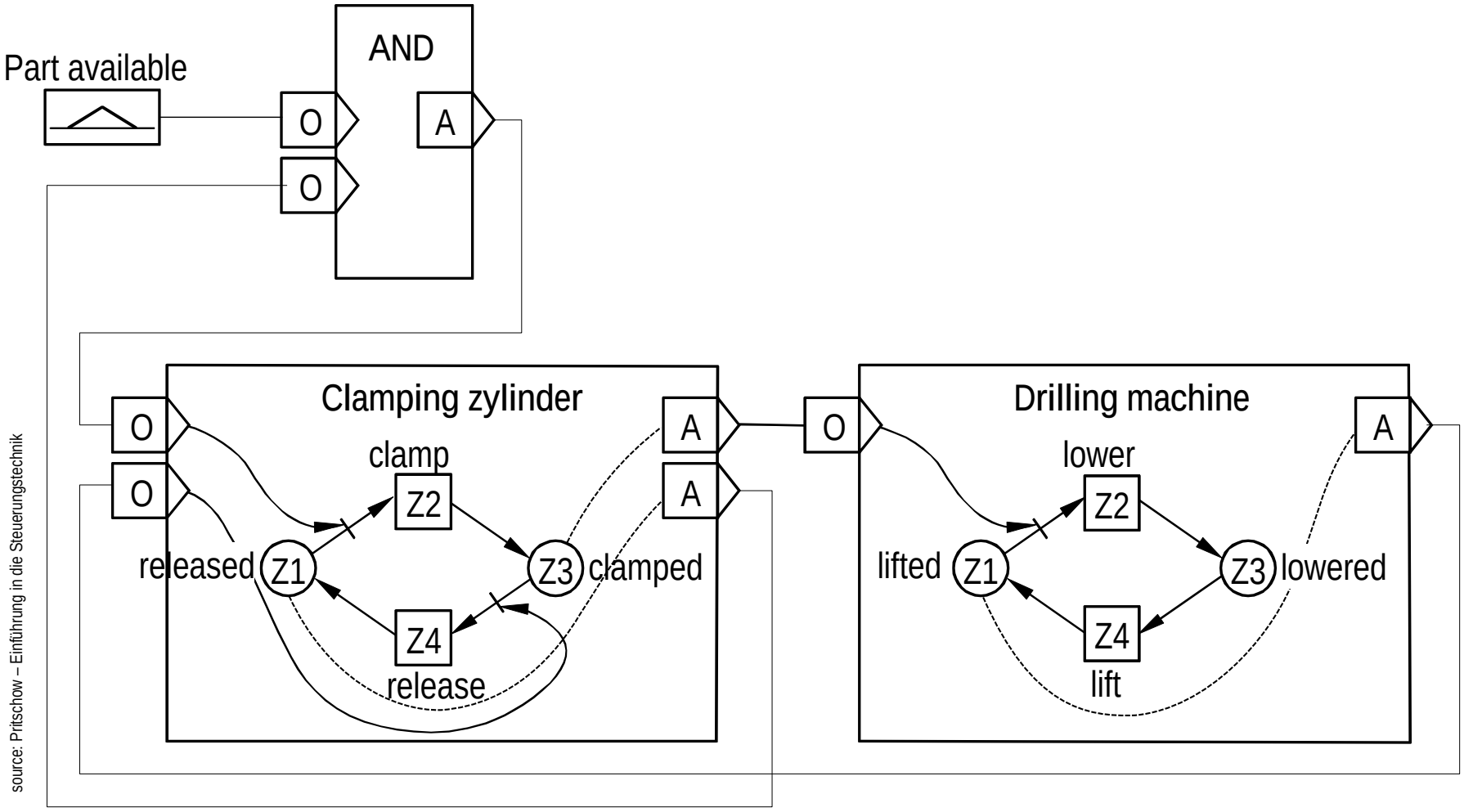


source: Pritschow – Einführung in die Steuerungstechnik

Connection of several state machines

Universität Stuttgart

Direct connection of state machines



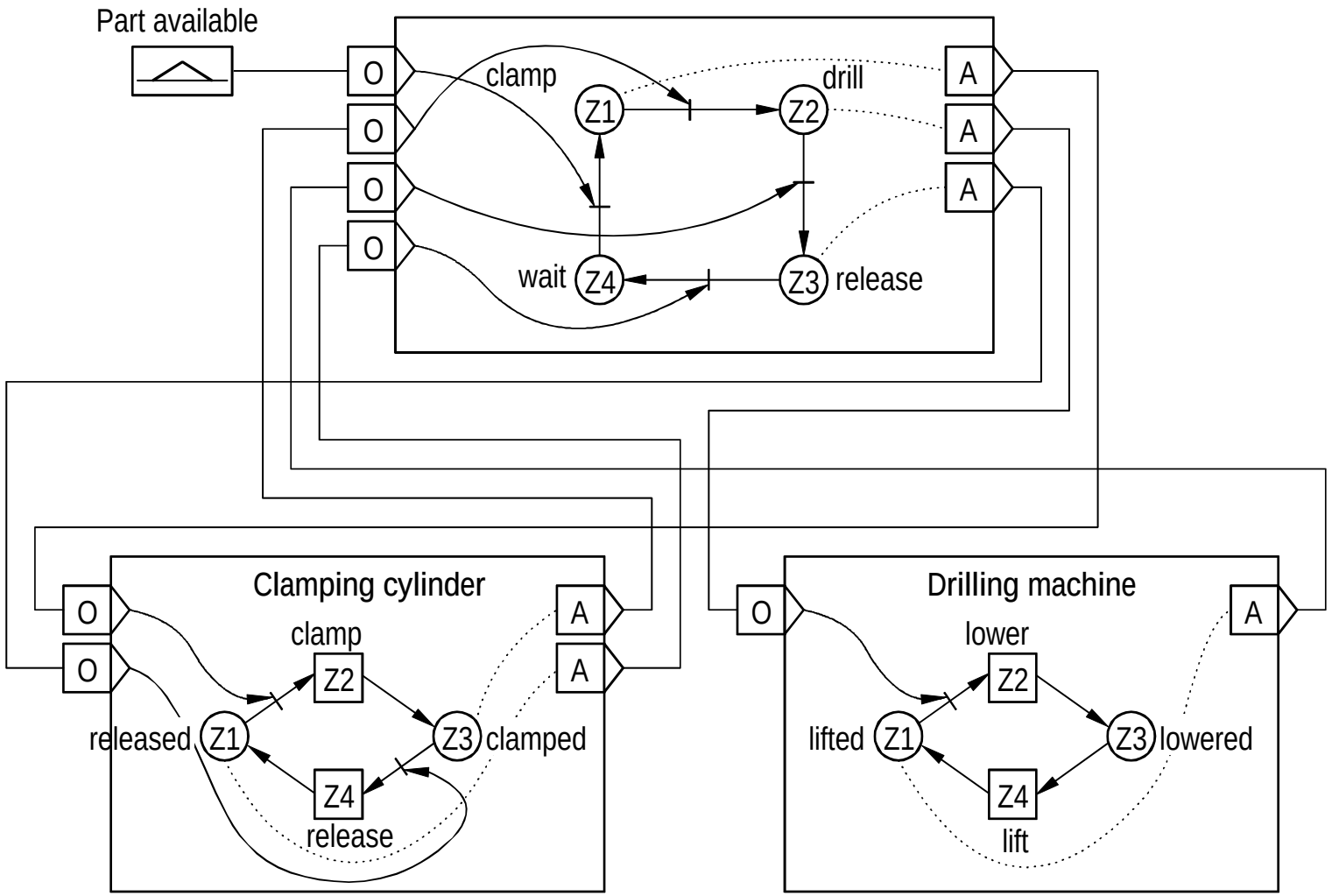
source: Pritschow – Einführung in die Steuerungstechnik

The inputs and outputs of function blocks (with encapsulated state machines) are connected directly.

Connection of several state machines

Universität Stuttgart

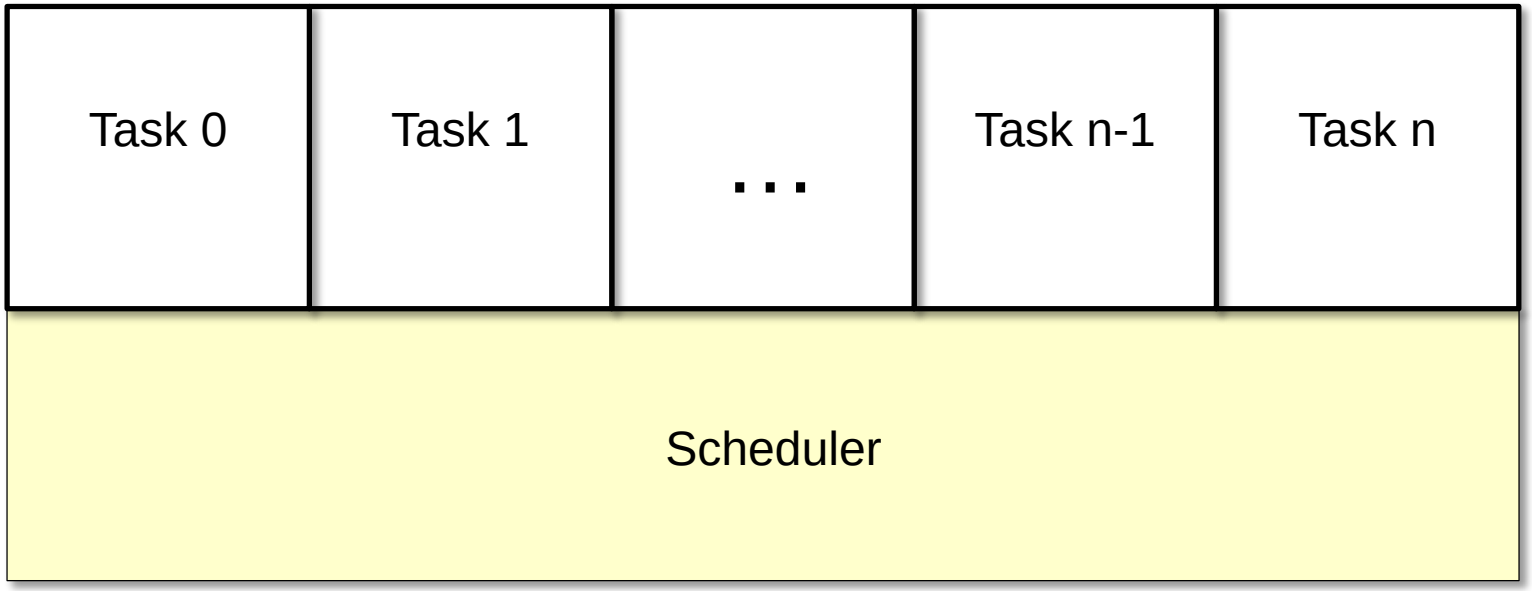
Indirect connection of state machines



Quelle: Pritschow – Einführung in die Steuerungstechnik

The inputs and outputs of function blocks (with encapsulated state machines) are connected by a high-level flow graph.

Scheduling



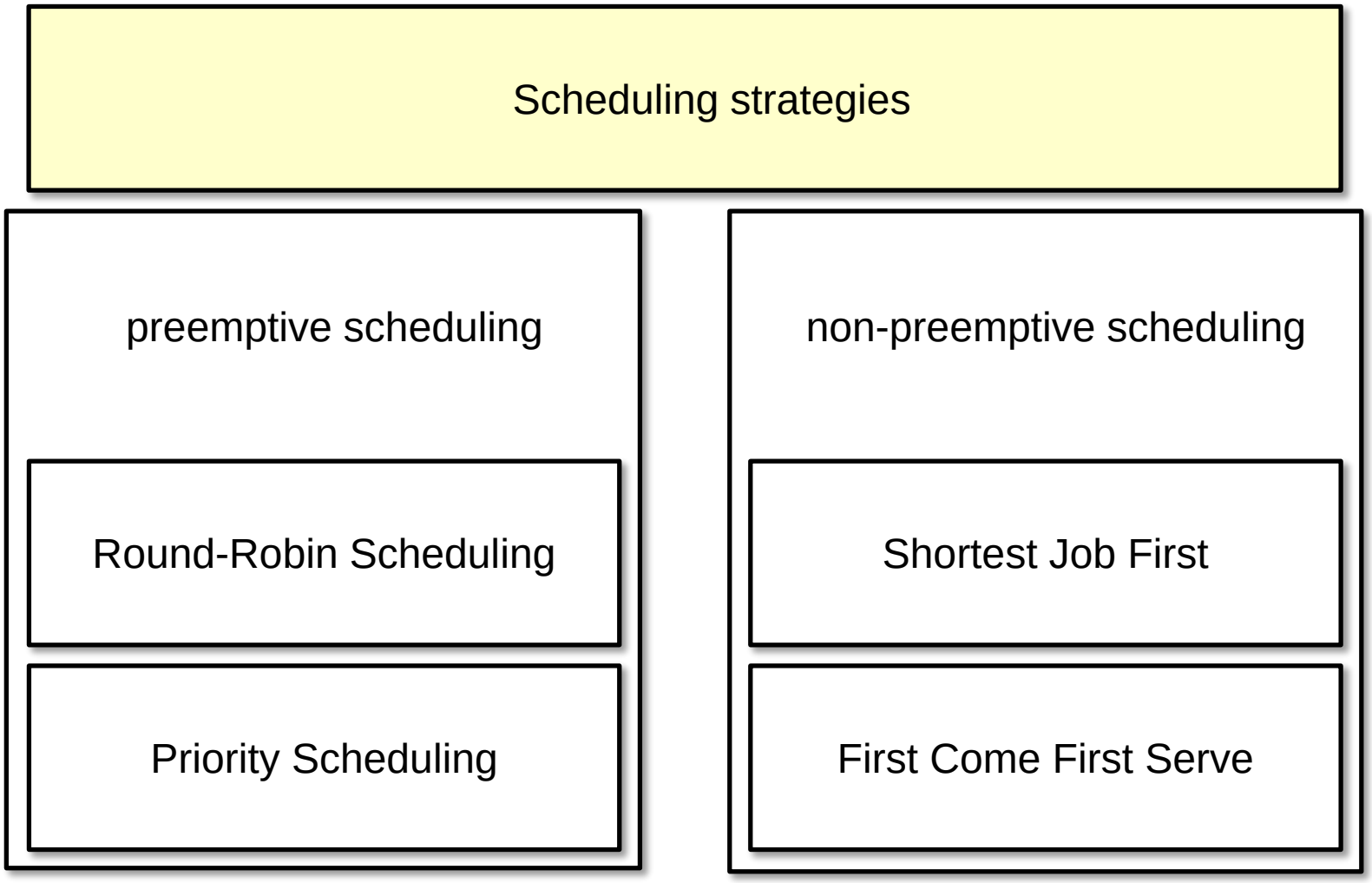
- Allocation of tasks to the processor by a given strategy
- Scheduling criteria:
  - fairness
  - efficiency
  - response time
  - residence time
  - Delivery rate
- Some criteria are inconsistent with one another!

Scheduling methods

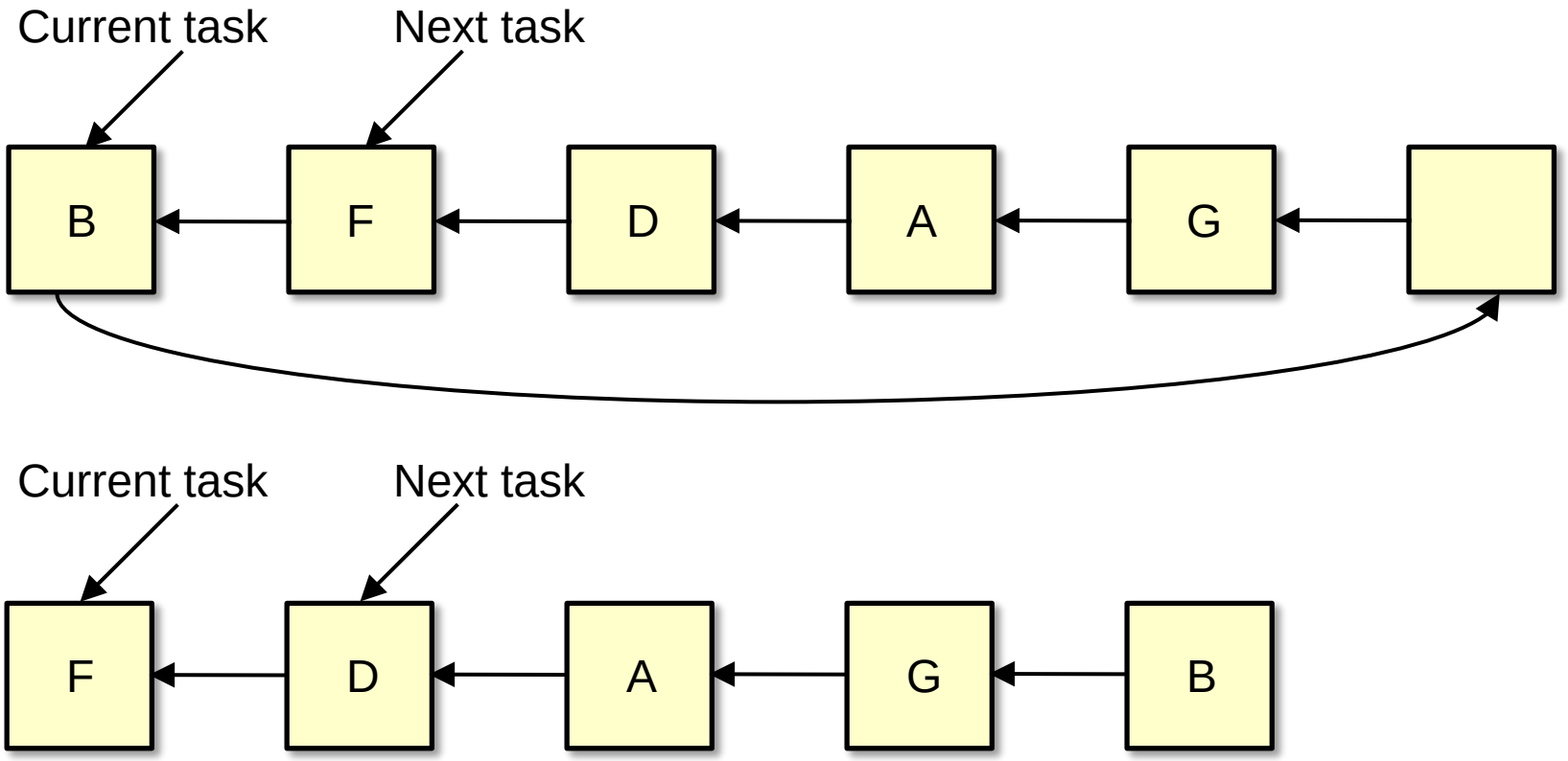
- **Non-preemptive scheduling** (run-to-completion method): a process is executed to its end, before the next process gets assigned to the processor. There is no interruption of a running process. This method was very suitable for earlier batch processing systems, but today will be used very limited.

- **Preemptive scheduling**: a process that currently has the processor (that means is processed) can be interrupted by an other process. The selection of the process which will be allocated to the processor is done by one of the following scheduling strategies.

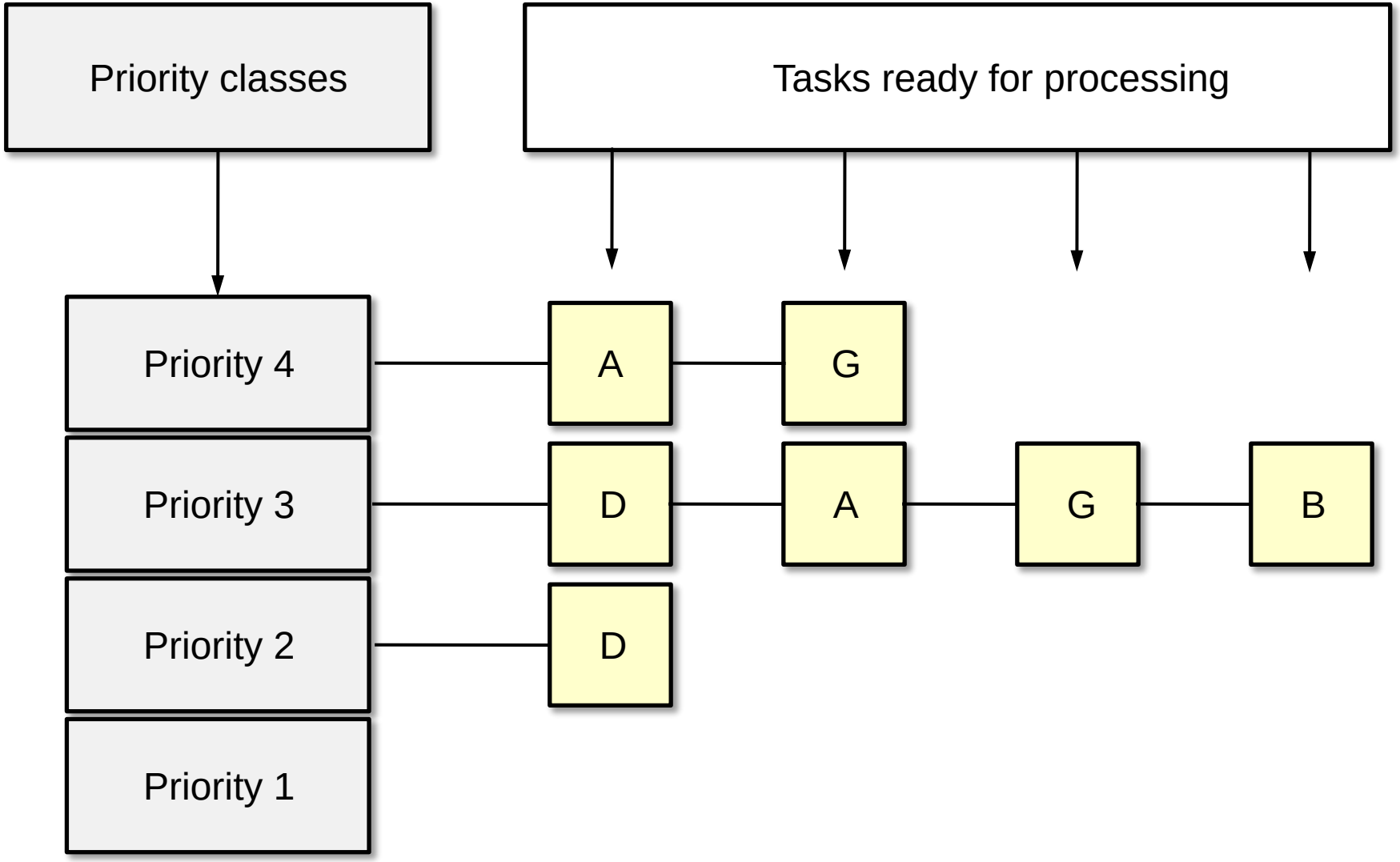
Scheduling strategies



Scheduling strategy: Round-Robin



Scheduling strategy: prioritiy classes



Scheduler on Microcontroller

- Normally there is no operating system on the microcontroller
- No direct support for several tasks, no scheduler available

- There are real-time operating systems especially for microcontrollers, but: Use of limited processor time by overhead functions, not suitable for time critical applications
- Example of real-time operating systems:
  - FreeRTOS (www.freertos.org)
  - TNKernel (www.tnkernel.com)
  - MicroC/OS-II (www.micrium.com/rtos/ucosii/overview)