# MA2012 Introduction to Mechatronics System Design Notes

Hankertrix

April 29, 2025

## Contents

**1 Definitions** **6**
- 1.1 Mechatronics . . . 6
- 1.2 System . . . 6
- 1.3 System science . . . 6
- 1.4 System engineering . . . 6
- 1.5 Measurement system . . . 6
- 1.6 Actuation system . . . 7
- 1.7 Control system . . . 7
- 1.8 Microprocessor (MPU) . . . 7
- 1.9 Microcomputer . . . 7
- 1.10 Microcontroller (MCU) . . . 7
- 1.11 Embedded systems . . . 8
- 1.12 General purpose computers . . . 8
- 1.13 Central Processing Unit (CPU) . . . 9
- 1.14 Memory . . . 9
- 1.15 Microprogram . . . 12
- 1.16 Firmware . . . 12
- 1.17 Program (noun) . . . 13
- 1.18 Program (verb) . . . 13
- 1.19 Write . . . 13
- 1.20 System or frontside bus . . . 13
- 1.21 Double Data Rate (DDR) . . . 13
- 1.22 Bootloader program . . . 14
- 1.23 Transducer . . . 14
- 1.24 Sensor . . . 14
- 1.25 Analogue quantity . . . 14

1.26 Digital quantity . . . 15
1.27 Slotted opto switch . . . 15
1.28 Data acquisition (DAQ) . . . 15
1.29 Digitised signal . . . 16
1.30 Shannon sampling theorem . . . 16
1.31 Aliasing . . . 17
1.32 Quantising theory . . . 18
1.33 Pulse width modulation . . . 19
1.34 Inductive kickback . . . 19
1.35 Servo motor . . . 20
1.36 Stator . . . 21
1.37 Rotor . . . 21
1.38 Solenoid . . . 21
1.39 Stepper motor . . . 22
1.40 Handshaking . . . 26
1.41 Interrupt service routine (ISR) . . . 27
1.42 Parallel data communications . . . 28
1.43 Serial data communications . . . 29
1.44 Parallel-serial interface . . . 29
1.45 Bit time interval ($T_B$) . . . 29
1.46 Baud rate . . . 29
1.47 Universal asynchronous receiver and transmitter (UART) . . . 30
1.48 Bus . . . 31
1.49 Asynchronous communication . . . 31
1.50 Synchronous communication . . . 32
1.51 Inter-integrated Circuit (I2C) Bus . . . 32
1.52 Full-duplex . . . 34
1.53 Serial Peripheral Interface (SPI) bus . . . 35
1.54 16-key hexadecimal keypad . . . 39
1.55 74C922 keypad encoder . . . 40
1.56 Input signal conditioning . . . 42
1.57 Output signal conditioning . . . 42
1.58 Noise . . . 42
1.59 Signal-to-noise ratio . . . 43
1.60 Arcing . . . 43
1.61 Logic level converter . . . 43
1.62 Bitwise OR . . . 43

**2 Mechatronics system components** **44**
2.1 Mechanical system . . . . . 44
2.2 Input signal conditional and interfacing . . . . . 49
2.3 Output signal conditioning and interfacing . . . . . 49
2.4 Graphical displays . . . . . 49
2.5 Digital control architectures . . . . . 50

**3 Basic computer structure** **51**
3.1 Central Processing Unit (CPU) . . . . . 51
3.2 Memory Unit . . . . . 51
3.3 Input/Output Unit (I/O Unit) . . . . . 51

**4 Design tips for mechatronics systems** **52**

**5 Examples of mechatronics systems** **52**
5.1 Car . . . . . 52
5.2 Inkjet printer . . . . . 53
5.3 Robots . . . . . 53

**6 Sensors** **54**
6.1 Digital sensors . . . . . 54
6.2 Interfacing with digital sensors . . . . . 57
6.3 Analogue sensors . . . . . 58

**7 Analogue to digital converters** **60**
7.1 Successive approximation . . . . . 60
7.2 Flash converter . . . . . 62

**8 Data acquisition (DAQ) process** **63**
8.1 Sensing element . . . . . 63
8.2 Signal conditioning element . . . . . 63
8.3 Signal processing element . . . . . 63
8.4 Data presentation element . . . . . 64
8.5 Examples of the DAQ process . . . . . 64

**9 Direct current motor** **65**
9.1 Permanent magnet or brushed DC motors . . . . . 65
9.2 Brushless DC motor . . . . . 66
9.3 Brushless vs brushed DC motors . . . . . 66
9.4 Controlling DC motors . . . . . 67

**10 Process and instrument control I/O** **69**

**11 Keyboard entry and display I/O** **70**

**12 Deterministic and random signals** **71**
12.1 Deterministic signals . . . . . 71
12.2 Random signals . . . . . 71
12.3 Randomness . . . . . 71

**13 Sources of noise** **73**
13.1 Internal noise sources . . . . . 73
13.2 External noise and interference sources . . . . . 74

**14 Input signal conditioning elements** **74**
14.1 Deflection bridges . . . . . 74
14.2 Operation amplifiers (Op-Amp) . . . . . 77
14.3 Filters . . . . . 81
14.4 Averaging . . . . . 87

**15 Output signal conditioning elements** **88**
15.1 Digital-to-Analogue converter (DAC) . . . . . 88

**16 Arduino Uno MCU** **90**
16.1 Specifications . . . . . 90
16.2 Arduino Uno and ATmega328 . . . . . 90
16.3 Number of bits . . . . . 90
16.4 Program execution speed . . . . . 91
16.5 Memory . . . . . 91
16.6 Components . . . . . 92
16.7 Hardware interrupts . . . . . 93
16.8 Pinout . . . . . 93

**17 Input and output interfacing** **94**
17.1 MCU-initiated transfer . . . . . 94
17.2 Device-initiated transfer . . . . . 97
17.3 Polling vs interrupting . . . . . 97

**18 Determining the output of switches** **98**

**19 Breadboard wiring** **98**

**20 Resistor colour code chart** **99**

**21 Calculating voltage resolution** **100**
21.1 Example . . . . . 100

**22 Parsing multiple bytes into a single integer** **101**

# 1 Definitions

## 1.1 Mechatronics

- A combination of the word "Mecha-nical" and "Elec-tronics". It is a termed coined in Japan in the 1960s.
- It is an interdisciplinary field of engineering dealing with the design of products whose function relies on the integration of mechanical and electronic components coordinated by a control architecture.

## 1.2 System

A combination of related parts combined into a complex whole.

## 1.3 System science

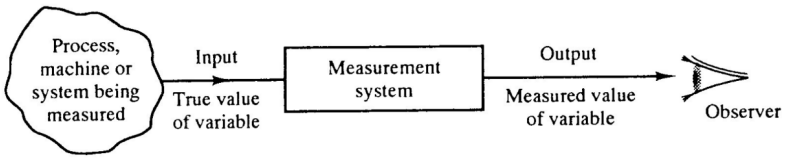
An interdisciplinary field of science that studies the nature of complex systems.

## 1.4 System engineering

An interdisciplinary field of engineering that focuses on how complex systems should be designed, engineered and managed.

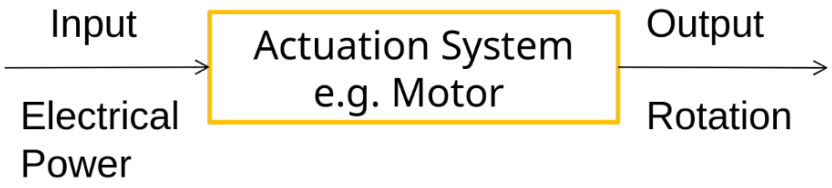
## 1.5 Measurement system

A system that presents an observer with a numerical value that represents the state or change of a variable of a process, machine or system being measured.




## 1.6 Actuation system

A system that converts one form of energy (e.g. electrical, chemical, etc.) to another to perform mechanical work (e.g. movement, heat, etc.).




## 1.7 Control system

A system to modify one or more measurable parameters of a plant or process to achieve desired outputs. For example, a central heating system:

- Input: Electrical power
- Actuation system: Heating coil and blower
- Plant: Environment or room
- Measurement system: Temperature sensor
- Output: Desired temperature

## 1.8 Microprocessor (MPU)

A microprocessor refers to the Central Processing Unit (CPU) on a single integrated-circuit (IC) chip.

## 1.9 Microcomputer

A microcomputer refers to the MPU combined with the memory (RAM) and input and output unit (I/O unit).

## 1.10 Microcontroller (MCU)

A microcontroller (MCU) refers to the microcomputer (CPU + Memory + I/O) in a single integrated-circuit (IC) chip.

## 1.11 Embedded systems

Embedded systems are microcontrollers for a single purpose (not single function).

### 1.11.1 Examples

- Consumer products: Mobile phones, washers, fridge, air-conditioner, etc.
- Healthcare products: Treadmills, blood glucose metre, etc.
- Automobile system controller
- And much more...

### 1.11.2 Advantages

Embedded systems are cheap, small and have low power consumption.

### 1.11.3 Disadvantages

Embedded systems are inflexible and have limited or even no expandability.

## 1.12 General purpose computers

General purpose computers are computers that are for general use and can do a wide variety of tasks. They are outnumbered by embedded systems in a ratio of $> 100 : 1$

### 1.12.1 Examples

Personal computer (PC), Macintosh, Workstations, Personal Digital Assistant (PDA), Smartphone, etc.

### 1.12.2 Advantages

- Flexibility, as general purpose computers are capable of running many kinds of software and interface with lots of devices.
- Expandability and scalability, as the software and the hardware of general purpose computers are upgradeable.

### 1.12.3 Disadvantages

- Complexity, to cater for flexibility and expandability
- Cost, as general purpose computers cost from $100 to upwards of $10,000
- High power consumption

## 1.13 Central Processing Unit (CPU)

- The CPU consists of the arithmetic and logic unit and the control unit
- Arithmetic and Logic Unit (ALU)
  - Arithmetic: $+, -, \times, \div$, etc.
  - Logic: OR, AND, NOT, NOR, NAND, XOR, etc.
  - Result of ALU is usually stored in the accumulator register
- Control Unit
  - Directs the operation of all parts of a computer by providing timing and control signals
  - Fetches, decodes, and executes the instructions in memory.

## 1.14 Memory

- A memory cell is a device or circuit used to store a single bit ("1" or "0"), like flash file system (FFS), magnetic core memory, disks.
- A memory word is a group of memory cells.
- Internal memory, also known as main or working memory, is the highest speed memory.
- Memory is used to store programs for sequenced execution.
- Memory is also used to store data for output at required time.
- Types of memory
  - Random Access Memory (RAM)
  - Read-Only Memory (ROM)
  - Electrically Erasable Programmable Read-Only Memory (EEPROM)
  - Flash memory

### 1.14.1 Random access memory (RAM)

- Any memory device that can go directly to an address without having to sequence through other locations.
- Random access memory is volatile, which means the data is lost when the system loses power.
- Variables and I/O data are stored in random access memory when arithmetic and logic operations are performed on them.
- The Arduino Uno has static random access memory with a capacity of 2 kB

### 1.14.2 Static RAM (SRAM)

- A semiconductor RAM device that consists essentially of flip-flop registers and the necessary circuitry for decoding.
- Information will remain valid as long as power is on.
- It has the advantages of fast access speed and simple circuitry.

### 1.14.3 Dynamic RAM (DRAM)

- Memory devices that store binary data by charging the gate capacitances of Metal Oxide Semiconductor (MOS) transistors.
- Refreshing or periodically recharging (roughly ever 4 ms) is required
- It has the advantages of lower cost for higher memory density, as well as low power consumption.

### 1.14.4 Synchronous dynamic RAM (SDRAM)

- Synchronous dynamic RAM is DRAM which is synchronous with the system bus.
- Use DRAM is when capacity and power consumption are valued more than access speed.

### 1.14.5 Read-Only memory (ROM)

- A semiconductor memory device designed primarily for having data read from them.
- Data cannot be changed during operation.
- The memory is non-volatile, which means the data persists even if the power is turned off.

### 1.14.6 Erasable programmable ROM (EPROM)

- A semiconductor ROM device with which the user can erase and reprogram the contents of memory as many times as desired.
- Exposure to UV lights erases information on the memory chip. The UV light sets all the cells to 1s.

### 1.14.7 Electrically erasable programmable ROM (EEPROM)

- A semiconductor ROM device with which the user can erase and reprogram individual locations by the application of appropriate voltage levels and pulse durations.
- The size of the EEPROM in the Arduino Uno is 1 kB.

### 1.14.8 Flash memory

- Flash memory is a type of EEPROM, which means it is non-volatile.
- It is erasable and programmable, and it works with a block of data instead singular bits.
- It also can be used as read-only memory (ROM), such as in flash Basic Input/Output System (BIOS), as a disk drive (USB drive, SD card, solid state drive, etc.) or random access memory (RAM), like flash RAM.
- Flash memory has advantages over:
  - EEPROM in versatility, as it can be used as RAM instead of just ROM
  - RAM in being non-volatile, as it can persist data even when the device is powered off, and it is also as fast as dynamic RAM (DRAM).
  - Disk drive in speed, as it has no moving parts, being a solid state drive.
- The Arduino Uno contains 32 kB of flash memory.

## 1.15 Microprogram

- A collection of instructions that control the timing and control logic of all components of the microprocessor.
- It is built into the microprocessor by the chipmaker and cannot be modified by programmers.

## 1.16 Firmware

- Firmware is software programmed into the Read-Only Memory (ROM) of a microprocessor for the execution of functions of a device.
- Programmed by device manufacturer and cannot be modified by users.

## 1.17 Program (noun)

- A program is software loaded into the Random Access Memory (RAM) of a general purpose computer.
- It is programmed by software developers (not necessarily device manufacturers) and cannot be modified by users.

## 1.18 Program (verb)

Programming refers to a procedure of putting data into memory for non-volatile storage.

## 1.19 Write

Writing means an instruction that sends data to an address in RAM or external storage.

## 1.20 System or frontside bus

- A system or frontside bus is the parallel conductor lines which connects the CPU to the other components.
- A 64-bit processor needs the bus to be 64-bit wide, which is 1 bit of information per line.

## 1.21 Double Data Rate (DDR)

- Double data rate refers to data transfer at both the positive and negative transitions.
- DDR$n$, where $n$ is a number, stands for DDR generation $n$.

### 1.21.1 Lower Power DDR (LPDDR)

Lower Power DDR is just DDR with lower power consumption to be used in mobile devices.

### 1.21.2 Graphics DDR (GDDR)

Graphics DDR is just DDR for graphics cards.

## 1.22 Bootloader program

- A bootloader program is a program to "boot-up" or "bootstrap" the computer.
- For a general purpose computer, it loads the operating system (Windows, iOS, Android, etc.) from hard drive or flash memory on to the RAM.
- For the Arduino Uno, it does the following:
  - It checks for new programming command from the serial connection to the PC.
  - If there is a new programming command, it downloads the new program from the PC into the Arduino.
  - Otherwise, it loads the latest programme stored in the flash memory.
- The Arduino Uno bootloader program is activated when it is powered on or reset.

## 1.23 Transducer

A device which converts one form of energy into another.

## 1.24 Sensor

- A sensor is a transducer which performs measurement.
- It is a device which produces an output signal (typically electrical) for the purpose of sensing physical phenomena.

## 1.25 Analogue quantity

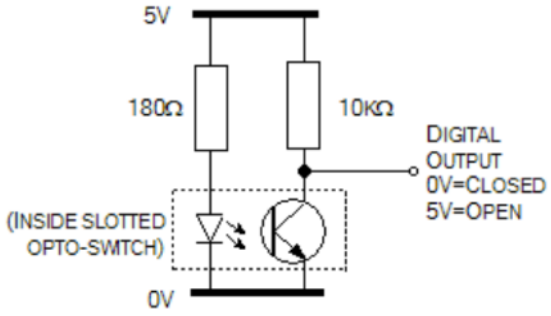
- An analogue quantity refers to a sensed quantity (voltage, current, etc.) that is continuous.
- It usually (not always) needs an analogue-to-digital converter (ADC) to interface with the MCU.

## 1.26 Digital quantity

- A digital quantity refers to a sensed quantity that is discrete.
- One example of digital quantities are two-state devices, such as switches and proximity sensors, which only have on or off, or high (1) or low (0).
- Another example are multiple state devices such as encoders, which have two types, one incremental, and another absolute.

## 1.27 Slotted opto switch

A slotted opto switch is a switch that returns an output, like high (1) or low (0), when there is something in the slot. It works by using an LED to shine a light into the sensor right beside it, and when the light is blocked, the slotted opto switch returns an output.




## 1.28 Data acquisition (DAQ)

Data acquisition is the process of sensing a physical state and converting it into a digital form for processing, presentation and storage.

## 1.29 Digitised signal

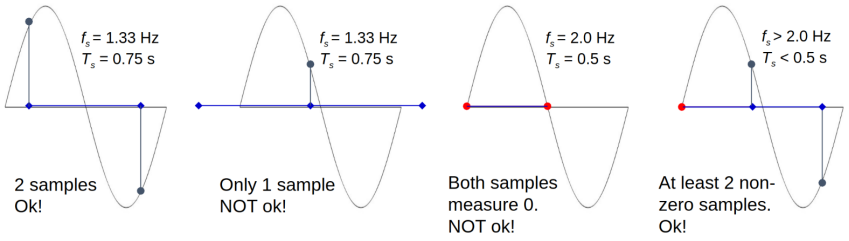
A digitised signal is a series of numbers which approximates an analogue signal at the sampling time instances. There are two important parameters, sampling rate (x-axis) and digitisation resolution (y-axis).

## 1.30 Shannon sampling theorem

The Shannon sampling theorem states that there is no maximum sampling frequency, higher sampling frequency also means more noise and higher costs. It also states that the sampling frequency ($f_s$) must be at least 2 times higher than the highest frequency component ($f_{max}$) in the signal, i.e.

$$f_s > 2f_{max} \iff T_s < \frac{1}{2}T_{min} \quad \because \text{ Period } T = \frac{1}{f}$$

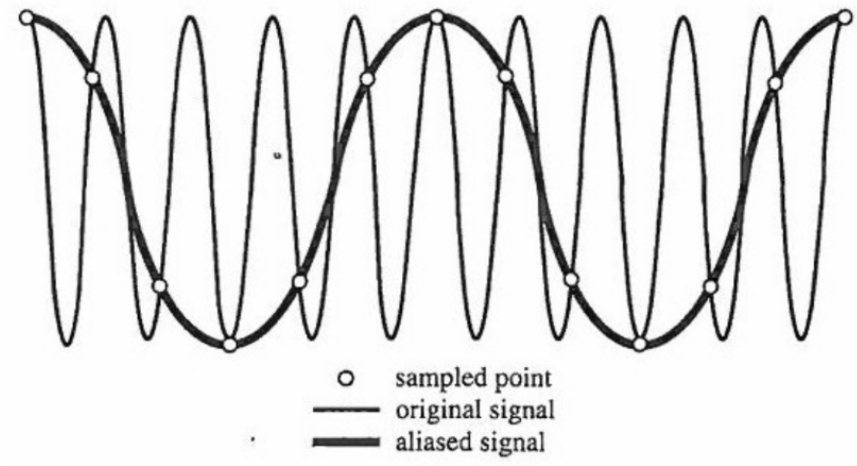
The Nyquist frequency refers to this frequency, which is $2f_{max}$.
Consider the following 1 Hz signals:



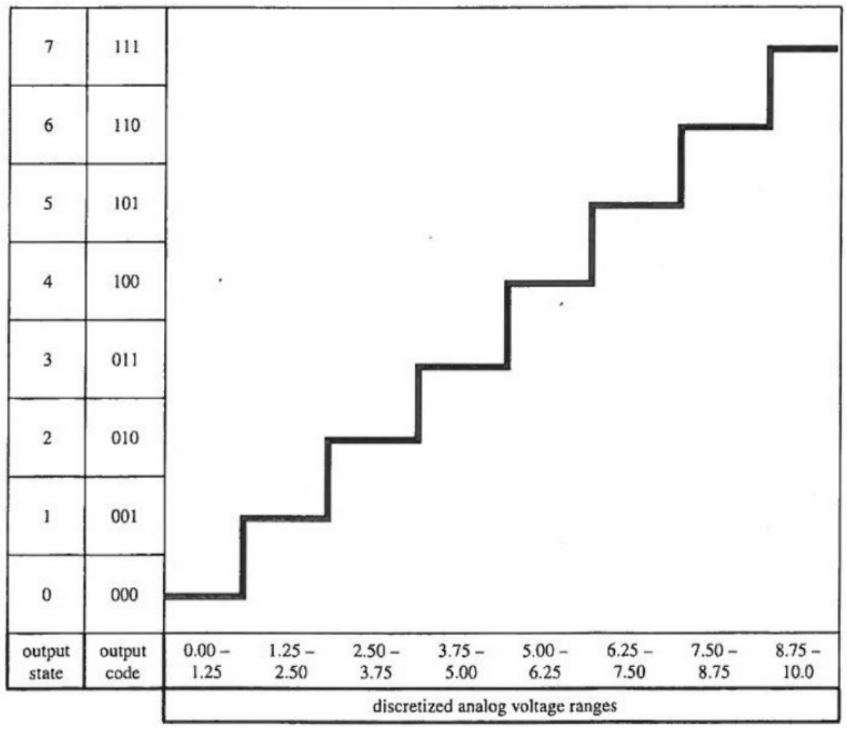
## 1.31 Aliasing

Aliasing is an effect that causes different signals to become indistinguishable (or aliases of one another) as a result of undersampling.
Example:

- The original signal is 10 cycles at $f_0$.
- The sampling frequency $f_s$ is $1.2f_0$.
- There is a total of 12 sampling points.
- The resulting sampled signal appears to be $0.2f_0$.



## 1.32 Quantising theory




### 1.32.1 Quantising

Quantising refers to the conversion of a continuous analogue signal to a series of discrete states.

### 1.32.2 Coding

Coding refers to the assignment of a digital code number to each state.

### 1.32.3 Number of states

The number of states is given by:

$$N = 2^n \text{ for the states } 0 \dots N-1$$

Where:

- $N$ is the number states
- $n$ is the number of bits of the analogue to digital conversion

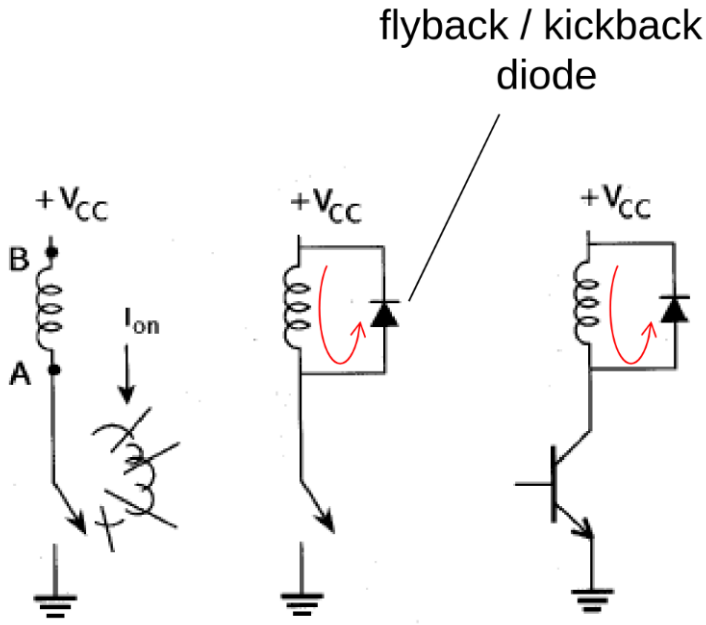
## 1.33 Pulse width modulation

Pulse width modulation varies the duty cycle or the duration of a pulse to control the average power or amplitude delivered by an electrical signal.

### 1.33.1 Example

- If a switch is always on, the lamp receives 9 V and lights up to the rated brightness.
- If a switch is 50% on and 50% off very quickly (1 - 200 kHz), the lamp receives and equivalent of 4.5 V, and lights up to only 50% of the rated brightness.

## 1.34 Inductive kickback

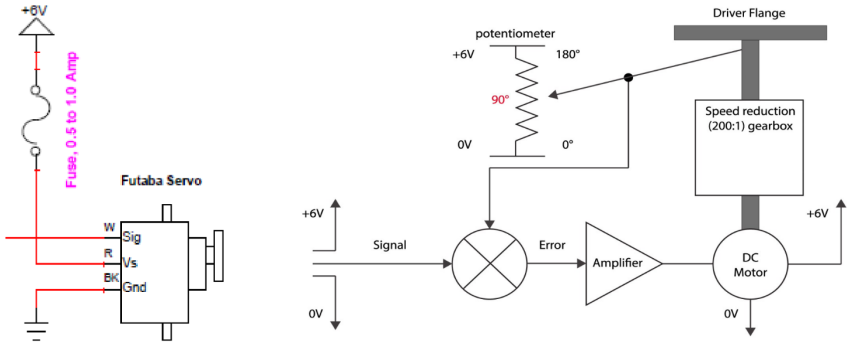



- The steady-state current through an inductor, $I_{on}$, cannot immediately go to zero when the switch is opened. The changing current induces a voltage across the inductor, making the potential $A$ greater than $B$, causing the switch to "blow up".
- A kickback or flyback diode protects the switch (physical or transistor) from blowing up.

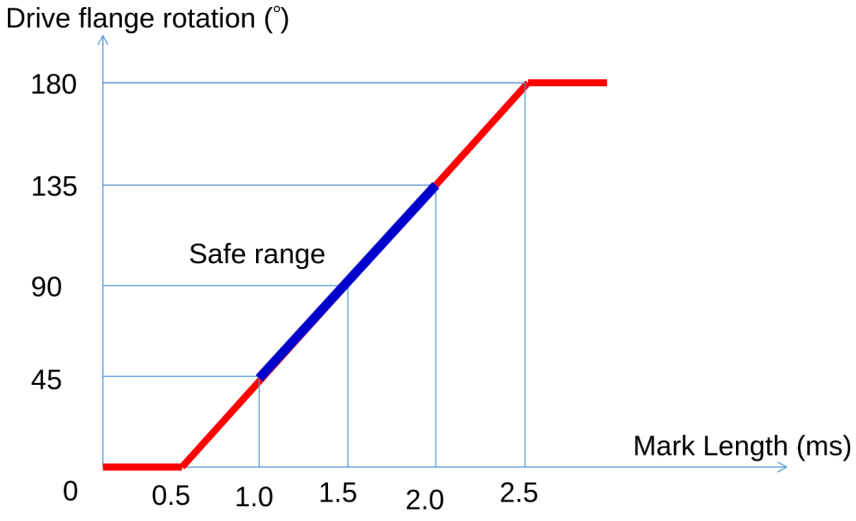
## 1.35 Servo motor

The drive flange of the servo motor can rotate 180°. It is driven by width of high pulse (Logic 1) called "Mark" length.


### 1.35.1 Working principle




### 1.35.2 Driving a servo motor



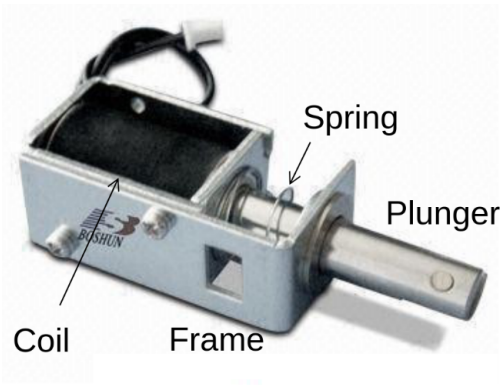
## 1.36 Stator

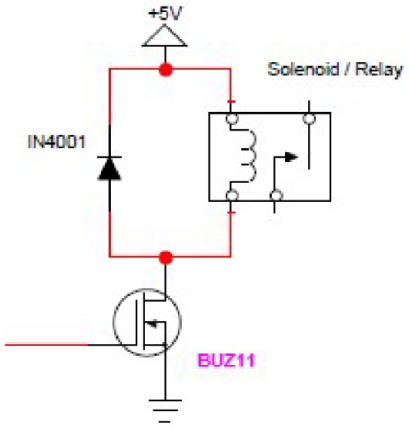
Stator just means an element or part that is static, or doesn't move.

## 1.37 Rotor

Rotor just means an element or part that rotates.

## 1.38 Solenoid




### 1.38.1 Construction

- Stationary iron frame (stator)
- Coil (solenoid)
- Ferromagnetic plunger (armature)

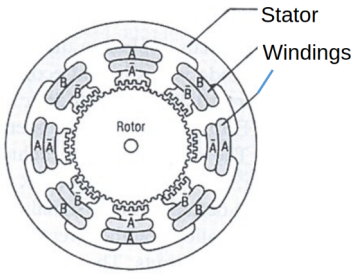
### 1.38.2 Types

A solenoid can be one of two types, a push solenoid or a pull solenoid.

### 1.38.3 Control

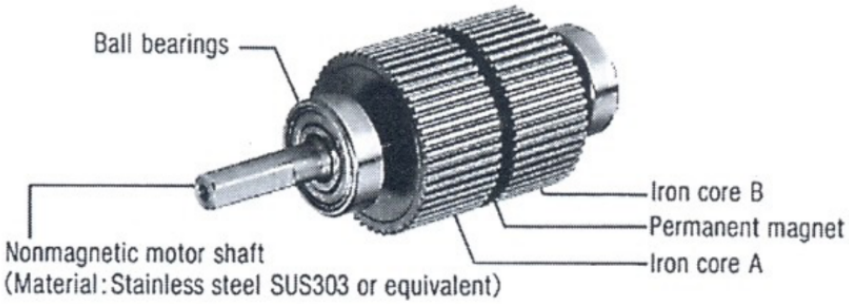
A pulse is used to turn the solenoid on and off.

## 1.39 Stepper motor



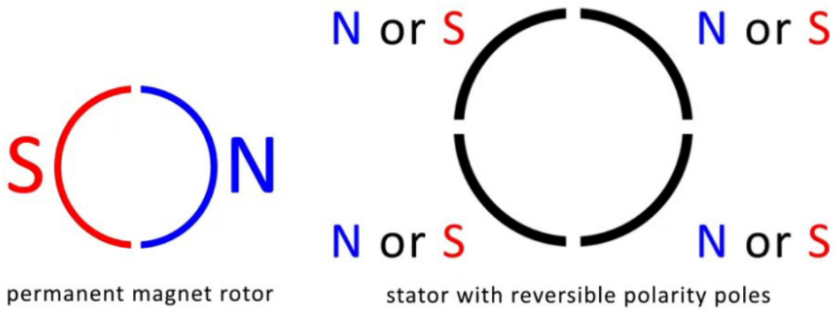

Rotor & Stator Configuration

### 1.39.1 Construction




- Permanent magnet rotor
- Stator with rotating magnetic field

### 1.39.2 Working principle




- If just one winding of the motor is energized, the rotor will snap (rotate) to a fixed angle and then hold that angle until the torque exceeds the holding torque of the motor.
- As the polarity of the stator changes, the permanent magnet motor will rotate to the next fixed position.
- Doubling the number of poles will half the step size, as each step is now smaller.

### 1.39.3 Wave drive or single phase stepping method

- In the wave drive or single phase stepping method, only one magnet in the stator is turned on at a time.
- This means the rotor inside the stepper motor will be attracted to one magnet of the stator of the stepper motor each time it takes a step.
- The rotor will therefore be aligned to the one magnet each step.

### 1.39.4 Two phase full step stepping method

- In the two phase full step stepping method, two magnets beside each other in the stator are turned on at a time.
- This means the rotor inside the stepper motor will be attracted to both magnets of the stator of the stepper motor each time it takes a step.
- The rotor will therefore be aligned in between the two magnets of the stator.
- These two magnets being activated allows the stepper motor to have more holding torque as the magnetic force holding the rotor in place is much higher compared to the wave drive stepper motor.
- As such, the motor can take a higher payload (more weight can be attached to the stepper motor).
- However, the motor also consumes more power.

### 1.39.5 Two phase half step stepping method

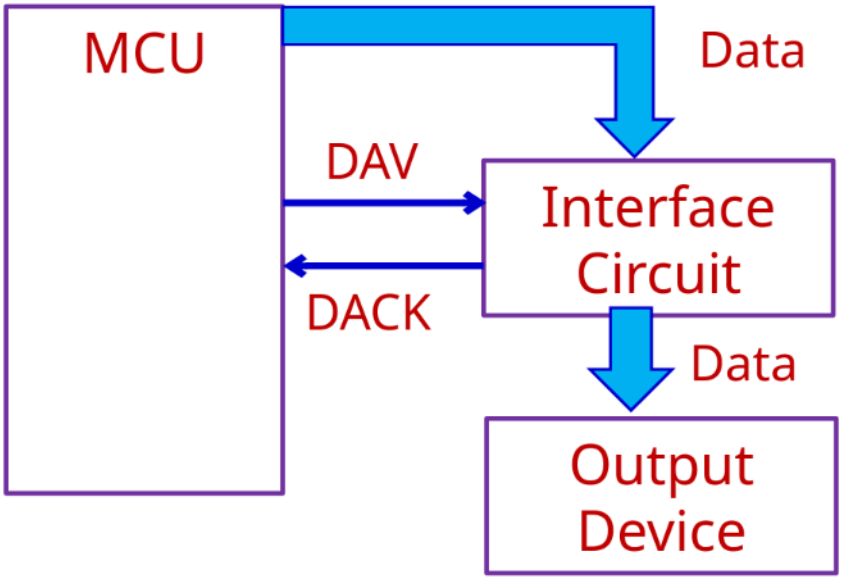
- In the two phase half step stepping method, the stepper motor alternates between the wave drive stepping method and the two phase full step stepping method.
- On the first step, the two phase full step stepping method is used, so the rotor will be attracted to two magnets of the stepper motor on the first step, and it will be in between both magnets.
- On the second step, the wave drive stepping method is used, so the rotor will be attracted to one magnet of the stepper motor on the second step, and it will be aligned to the one activated magnet on the stator.
- On the third step, the two phase full step stepping method is used, so the rotor will be attracted to two magnets of the stepper motor on the first step, and it will be in between both magnets.
- On the fourth step, the wave drive stepping method is used, so the rotor will be attracted to one magnet of the stepper motor on the second step, and it will be aligned to the one activated magnet on the stator.
- This pattern continues for the rest of the steps, with the motor alternating between the two phase full step stepping method and the wave drive stepping method.
- This stepping method has the advantage of a smaller step resolution than the two phase full step stepping method, as each step of this stepping method is half a step of the two phase full step stepping method.

### 1.39.6 Micro-stepping

- Micro-stepping is a way to achieve smaller step resolutions by controlling the fractions of the current flowing into the two magnets being activated in the two phase stepping method individually.
- This allows the motor to step in fractions of a full step at a time instead of a full step.
- For example, the second magnet can be powered with one-eighth the current of the first magnet, which will cause the rotor to be much closer to the first magnet than the second magnet, allowing the rotor to rotate one-eighth of a step.

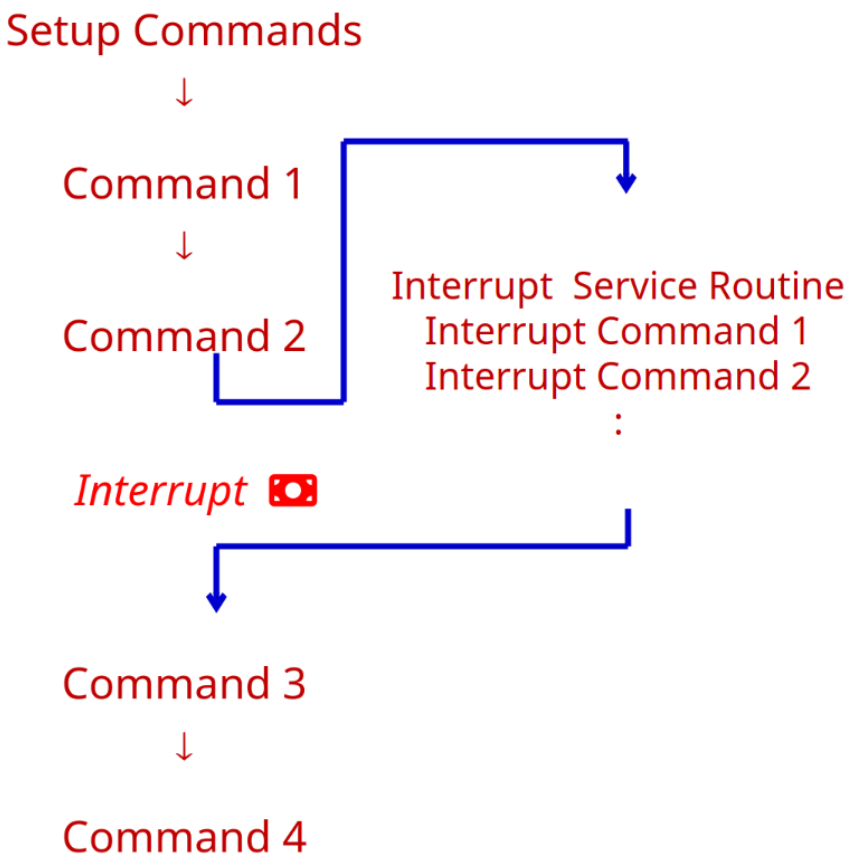
## 1.40 Handshaking

Handshaking is the process when the MCU sends a data available (DAV) control signal to an output device, and the output device accepts the data upon receiving the DAV signal, and sends a data accepted (DACK) control signal back to the MCU.



### 1.41 Interrupt service routine (ISR)

Interrupt service routine contains the commands for transferring to and from the interrupting I/O devices.



### 1.41.1 In an Arduino

- The interrupt service routine cannot have parameters and returns nothing.
- Only 1 interrupt service routine can run at any one time, other interrupt service routines (if any) will be turned off until the current one is executed.
- Functions which rely on timer interrupts will not work while an interrupt service routine is running, like `delay()` and `millis()`.
- Global variables are used to pass parameters between the main program and the interrupt service routine, so declare these variables as `volatile`.

Example:

```
int pin = 1;
volatile int state = LOW;

void setup() {
    pinMode(pin, OUTPUT);
    attachInterrupt(0, blink, CHANGE);
}

void loop() {
    digitalWrite(pin, state);
}

void blink() {
    state = !state;
}
```

## 1.42 Parallel data communications

- Multiple bits of data are transmitted all at one time.
- One data line or pin per bit is needed.
- The advantage of parallel data communications is faster data transfer rate.

## 1.43 Serial data communications

- Data is transmitted one bit after another
- Only one data line or pin is needed
- Advantages:
  - Cheaper to implement as physical pins and data lines are costly.
  - Easier to integrate into IC & PCB design as fewer physical pins and lines result in a smaller footprint.

## 1.44 Parallel-serial interface

A parallel-serial interface is for serial or parallel conversions for communications between the MPU and a serial I/O device.

- It converts an N bit parallel word from the MPU data bus to a serial data word.
- It also converts a serial data signal from a serial device to an N bit parallel data word.

## 1.45 Bit time interval ($T_B$)

The bit time interval is the period of time between one bit to another, when transferring data.

## 1.46 Baud rate

Baud rate is the data rate, or the rate of data transmission, which is measured in bits per second or Mbps.

$$\text{Baud rate} = \frac{1}{T_B}$$

Common baud rates are 19200, 14400, 9600, etc.

### 1.46.1 Example

If the data rate (baud rate) is 9600 baud, what is the time duration of 1 bit?

- The data rate is 9600 bits per second
- The bit time is $\frac{1}{9600} = 104.17\,\mu s$

## 1.47 Universal asynchronous receiver and transmitter (UART)

- It uses one to one communication, which means a pin receives the data and another pin transmits the data.
- For the Arduino Uno, digital pin 0 is the Rx pin and digital pin 1 is the Tx pin.
- For other pins that don't have hardware serial, use the `SoftwareSerial` library to use them for communication.
- It consists of the following components:

### 1.47.1 Serial receiver (Rx)

A serial receiver converts a serial input to a parallel format, and stores it in the receiver data register (RxDR) for eventual transmission to the MPU. For the Arduino Uno, this is digital pin 0.

### 1.47.2 Serial transmitter (Tx)

A serial transmitter takes a parallel word from the transmitter data register (TxDR) and converts it to a serial format for transmission. For the Arduino Uno, this is digital pin 1.

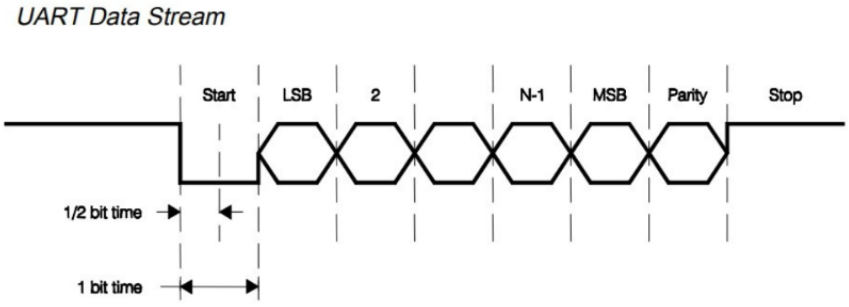
### 1.47.3 Bidirectional data bus buffer

A bidirectional data bus buffer is to pass data from the MPU to the transmitter data register (TxDR) or from the receiver data register (RxDR) to the MPU over the system bus.

### 1.47.4 Baud rate generator

Sometimes an external clock is used instead of the timer on the Arduino.

### 1.47.5 How UART works




1. It first organises data into packets, and each packet contains:
    - 1 start bit (LOW)
    - 5 - 9 data bits
    - 1 optional parity bit, to validate that the data is transferred correctly
    - 1 - 2 stop bits (HIGH)
2. Read the data according to the baud rate.

## 1.48 Bus

A bus is a group of wires used as a common path connecting all the inputs and outputs of several registers and devices so that the data can be easily transferred from any one register or device to any other using various control signals.

## 1.49 Asynchronous communication

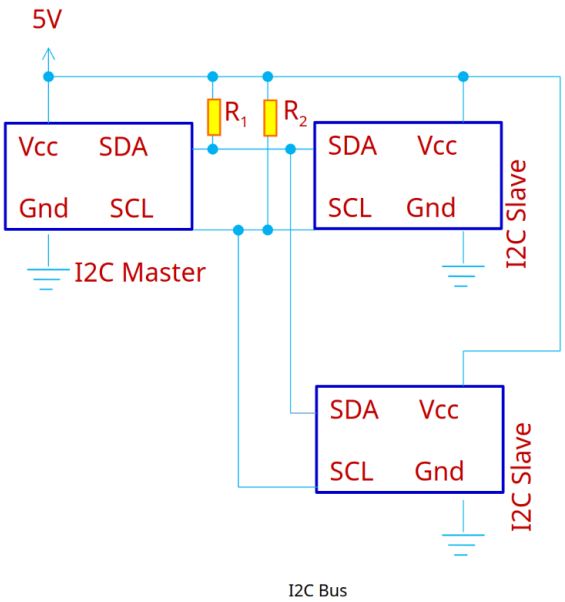
- The transmitter can send data to the receiver at any time.
- The time delay between the transmission of two words may be indeterminate.
- Transmitter clock need not synchronise with the receiver clock.

## 1.50 Synchronous communication

- Transmitting and receiving are synchronised by common clock pulses.
- Transmitter sends data to receiver continually.
- Transmitter sends meaningless data, like the ASCII sync character $16_{16}$ continually when there is no data to send.

## 1.51 Inter-integrated Circuit (I2C) Bus

- I2C bus is controlled by a master device, which is usually the MCU.
- One or more slave devices, usually an I/O device, receives the control signal from the master device.
- All devices share the same clock signal (SCL) and a bidirectional data line (SDA), and hence there are only 2 signal lines.
- Only the master device can initiate communications between master and slaves to avoid bus contention.
- Each slave device has its own unique 7-bit address or ID number. This address may be fixed or selectable, and is dependent on the manufacturer.
- Each signal line requires a pull-up resistor to restore the signal to HIGH when no device is set to LOW.
- When a master device initiates a communication, a device address is transmitted.
- Only the slave device with the correct address shall respond to the master.




Note: pull-up resistors are needed to maintain HIGH state when all devices are disconnected

### 1.51.1 Communication protocol

Steps to communicate with different I2C slave devices need to follow protocol defined by manufacturers in the provided data sheets.
Basic steps:

1. Master sends a Start bit.
2. Master sends a 7-bit slave address of intended device.
3. Master sends a Read (1) or Write (0) bit depending on application.
4. Slaves responds with an "acknowledge", i.e. ACK bit (0).
5. In Write mode, master sends 1 byte of information (command or data) at a time, and slaves respond with ACKs. In Read mode, master receives 1 byte of information at a time and sends an ACK to the slave after each byte.
6. When communication has been completed, master sends a Stop bit.

### 1.51.2 On the Arduino

- Use the `Wire.h` library to use I2C communication.
- Analogue pin A4 is the bidirectional data line (SDA).
- Analogue pin A5 is the line for the same clock signal (SCL).
- The data is transferred in 8 bits.
- To read data, a request for data must be sent first using `Wire.requestFrom(address, quantity, stop)`.
  - Address refers to the 7-bit address of the device to request data from.
  - Quantity refers to the number of bytes to request.
  - Stop is a boolean. When set to `true`, the protocol will send a stop message after the request, releasing the bus. When set to `false`, it will continually send a repeated start message after the request, keeping the connection active.

### 1.51.3 I2C vs SPI

| I2C advantages | SPI advantages |
|---|---|
| Requires only 2 communication lines | Higher data transmission rate |
| | Easier to implement |
| | No pull-up resistors needed |

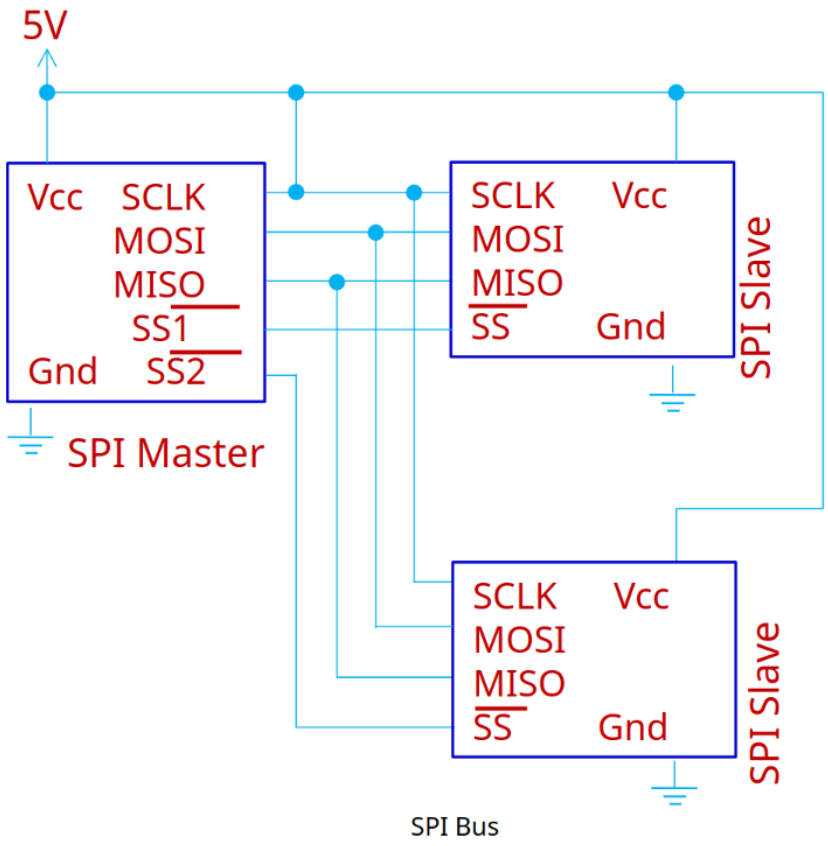
## 1.52 Full-duplex

Full-duplex means simultaneous bidirectional transmission of information.

### 1.53 Serial Peripheral Interface (SPI) bus

SPI bus is a full-duplex serial communication protocol between master and one or more slaves. It is an interface bus used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards.

- There are 3 pins for communications between master and all slaves
  - Shared or Serial Clock (SCLK)
  - Master Out Slave In (MOSI) or Slave Data In (SDI)
  - Master In Slave Out (MISO) or Slave Data Out (SDO)
- Each slave device requires an additional slave select (SS) or chip select (CS) pin, so 1 slave select (SS) or chip select (CS) line is needed for each slave device.
  - This line is normally held HIGH to disconnect the slave.
  - When it is brought to LOW, communication is initiated with the slave.
  - It is brought to HIGH again after communication has completed.
- The total number of I/O pins required is $3 + n$, where $n$ is the number of slave devices.
- All slave devices share MOSI, MISO and SCLK lines, hence all commands from the master are sent to each slave device.
- Every clock cycle a bit must be sent and received (i.e. synchronous), but that bit may be meaningless.




### 1.53.1 Modes of communication

SPI devices are synchronous, i.e. data is transmitted in sync with a SCLK. There are 4 modes of communication.

| Mode | Clock polarity | Clock phase (data capture on . . . ) |
|---|---|---|
| 0 | Low at idle | Rising edge |
| 1 | Low at idle | Falling edge |
| 2 | High at idle | Falling edge |
| 3 | High at idle | Rising edge |

### 1.53.2 Communication protocol

Basic process:

1. Set the SS pin to LOW for the targeted slave device
2. Toggle SCLK (square wave) at a speed that is less than or equal to the transmission speed supported by the slave device.
3. For each clock cycle, master sends 1 bit on MOSI and receives 1 bit on MISO.
4. Continue until data transmission is complete, and stop toggling the clock line.
5. Set SS pin to HIGH.

### 1.53.3 On the Arduino Uno

- Use the `SPI.h` library to use SPI communication.
- Digital pin 11 is the MOSI pin.
- Digital pin 12 is the MISO pin.
- Digital pin 13 is the SCK pin.
- Digital pin 10 is the SS pin.
- `SPISettings(speed, MSB/LSB, mode)`
  - Speed is the rated data transfer speed of the slave in bits per second.
  - Most Significant Bit (MSB) or Least Significant Bit (LSB) first.
  - Mode:

| Mode | Clock Polarity (CPOL) | Output Edge | Data Capture |
|---|---|---|---|
| `SPI_MODE0` | Low at idle | Falling | Rising |
| `SPI_MODE1` | Low at idle | Rising | Falling |
| `SPI_MODE2` | High at idle | Rising | Falling |
| `SPI_MODE3` | High at idle | Falling | Rising |

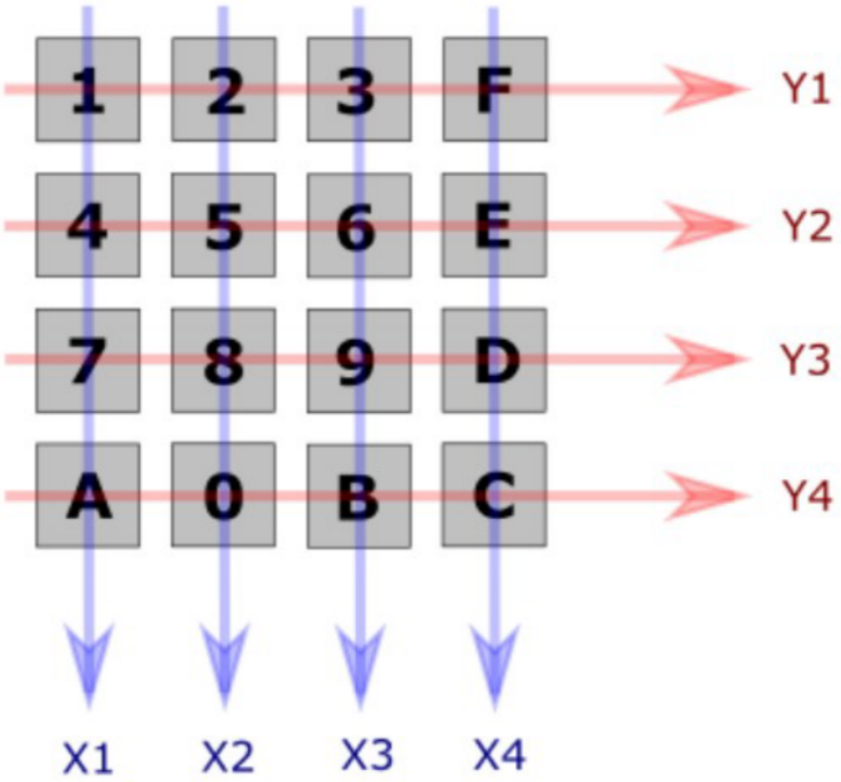
### 1.53.4 Communication using SPI in the Arduino

- The slave select or chip select pin must be set to LOW to enable communication first, i.e. `digitalWrite(10, LOW)`
- SPI transfer works in two steps, and the first step is to send a byte to the slave device.
- The first bit of this byte (8 bits) is the read or write bit. Set this bit to 1 to read, and 0 to write.
- The second bit is the multiple byte (MB) bit, which when set to 1, allows the SPI communication protocol to transfer multiple bytes at once, instead o just one byte at a time. The next byte transferred from the slave device is the register address that is 1 more than the current register address.
- When this second bit is set to 0, the SPI communication must be terminated, i.e. slave select or chip select pin set to HIGH, `digitalWrite(10, HIGH)`, after the data is read or written.
- The third bit to the eighth bit holds the register address to read or write from, which is determined from the data sheet of the slave device.
- The second step is to read or write the data.
- For a read operation, transfer a 0 to the slave device, i.e. `SPI.transfer(0)`.
- For a write operation, send a byte that corresponds to the operation to perform on the address sent in the first step, which can be obtained from the slave device's data sheet.

### 1.53.5 I2C vs SPI

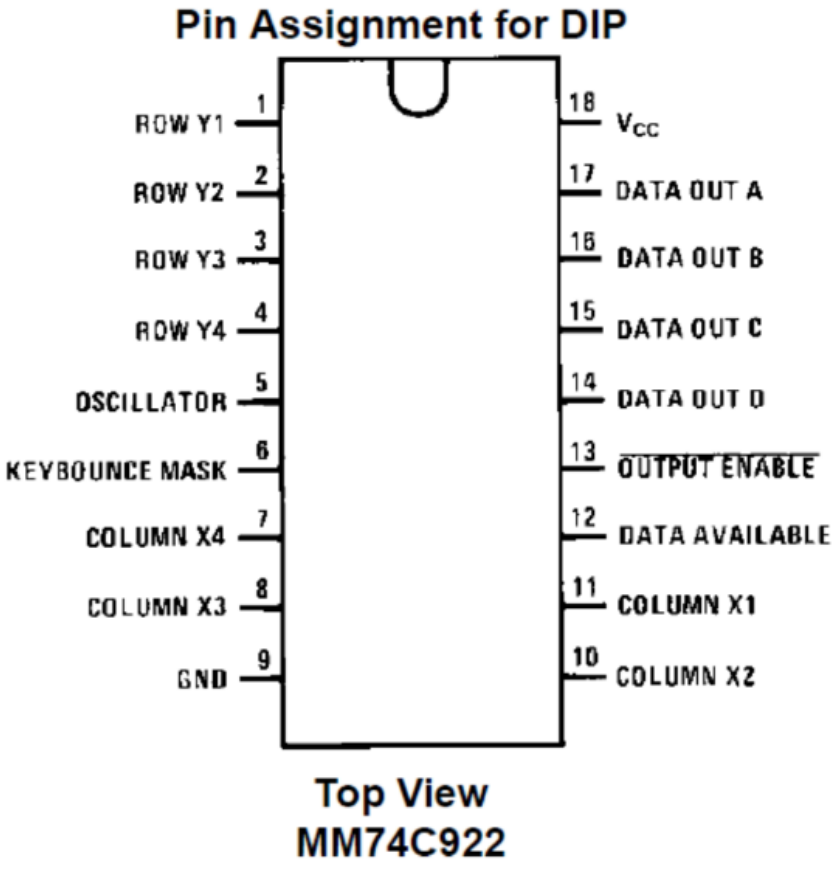
| I2C advantages | SPI advantages |
|---|---|
| Requires only 2 communication lines | Higher data transmission rate |
| | Easier to implement |
| | No pull-up resistors needed |

### 1.54 16-key hexadecimal keypad




The 16-key hexadecimal keypad has 8 outputs, and hence requires 8 pins.

### 1.55 74C922 keypad encoder




The 74C922 keypad encoder has 8 inputs for the keypad and 4 outputs to the Arduino. The data available pin tells the Arduino that the keypad has been pressed.

### 1.55.1 Position of the key pressed

**Truth Tables**

(Pins 0 through 11)

| Switch Position | | 0 Y1, X1 | 1 Y1, X2 | 2 Y1, X3 | 3 Y1, X4 | 4 Y2, X1 | 5 Y2, X2 | 6 Y2, X3 | 7 Y2, X4 | 8 Y3, X1 | 9 Y3, X2 | 10 Y3, X3 | 11 Y3, X4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA OUT | A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | E (Note 1) | | | | | | | | | | | | |

(Pins 12 through 19)

| Switch Position | | 12 Y4, X1 | 13 Y4, X2 | 14 Y4, X3 | 15 Y4, X4 |
|---|---|---|---|---|---|
| DATA OUT | A | 0 | 1 | 0 | 1 |
| | B | 0 | 0 | 1 | 1 |
| | C | 1 | 1 | 1 | 1 |
| | D | 1 | 1 | 1 | 1 |
| | E (Note 1) | | | | |

**Note 1:** Omit for MM74C922

$$\text{Position} = A + 2 \cdot B + 4 \cdot C + 8 \cdot D$$

## 1.56 Input signal conditioning

Input signal conditioning is to convert the output of sensing elements into a form suitable for further processing.

### 1.56.1 Types

- Analogue-to-Digital conversion
- Reducing noise level
- Enhancing a signal's power
- Improving noise immunisation
- Eliminate non-linearity

## 1.57 Output signal conditioning

Output signal conditioning is to convert the output of digital control systems, like the MCU or the PC, into a form suitable for interfacing with output elements.

### 1.57.1 Types

- Digital-to-Analogue conversion
- Amplifying signals
- Improving noise immunisation

## 1.58 Noise

- Noise is unwanted signal.
- White noise is a signal with equal power in all frequencies and zero mean, i.e. a totally random signal.
- Coloured noise is unwanted signal with certain bias or distinctive frequencies. An example of coloured noise is leaves falling from a tree when the wind is blowing. The mean of the leaves spread is no longer at the foot of the tree, i.e. the mean or bias is non-zero.

## 1.59 Signal-to-noise ratio

- Signal-to-noise ratio is defined as the ratio of the power of a signal (meaningful information) and the power of the background noise (unwanted signal).

$$SNR = \frac{\text{Total signal power}}{\text{Total noise power}} = \frac{w_S}{w_N}$$

- Expressed in decibels:

$$SNR = 10 \log_{10} \left( \frac{w_S}{w_N} \right) \text{dB}$$

## 1.60 Arcing

Arcing is the process of raising the potential to cause electrical current to flow between the anode and cathode through the inert gases inside the tube of a fluorescent light.

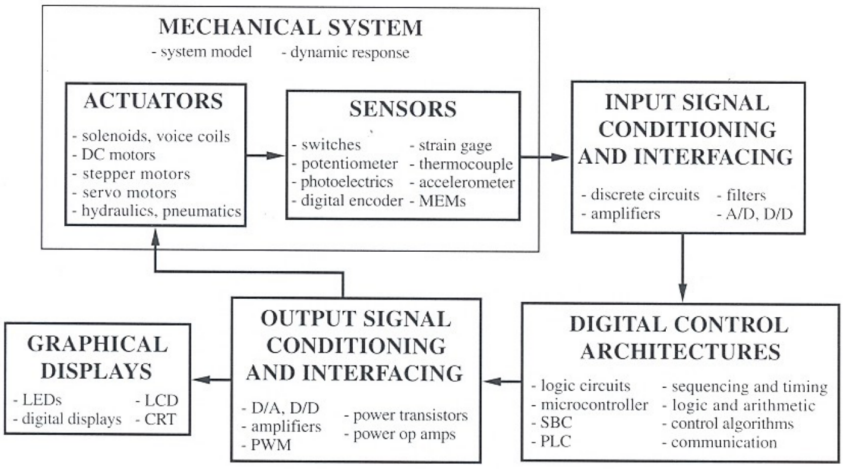
## 1.61 Logic level converter

A logic level converter converts from 5 V to 3.3 V.

## 1.62 Bitwise OR

Bitwise OR returns 0 only if both inputs are 0, otherwise it returns a 1. Using bitwise OR with 0 is the identity operation for binary, like how multiplication by 1 just gives you back the same number, the bitwise OR operation with 0 returns the same bit back.

# 2 Mechatronics system components




## 2.1 Mechanical system

A mechanical system is when you put the actuators and sensors together.

### 2.1.1 Actuators

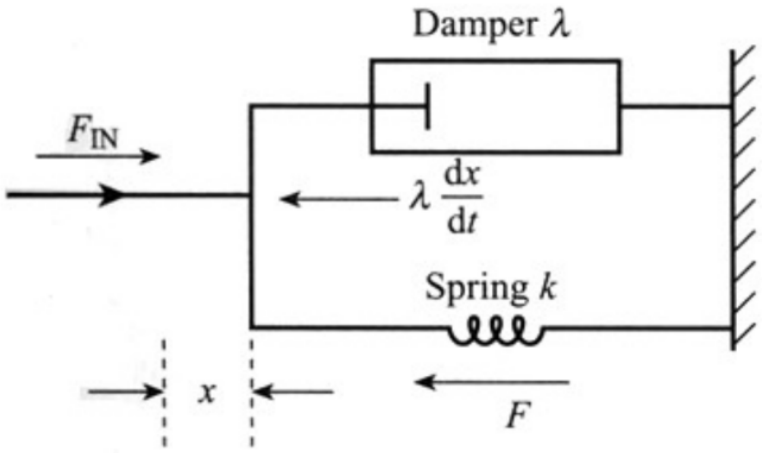
- Motors (AC motors, DC motors, Stepper motors, Servo motors, Voice coil, etc.)
- Hydraulics, pneumatics, solenoids
- Piezoelectrics and other smart materials
- Others elements like heating and cooling elements

### 2.1.2 Sensors

- Switches for turning things on and off
- Potentiometer and encoder for position or displacement
- Inertial sensors:
  - Accelerometer for acceleration
  - Rate gyroscope for angular velocity
- Thermocouple for temperature
- Strain Gage for deflection
- Photoelectrics for light intensity

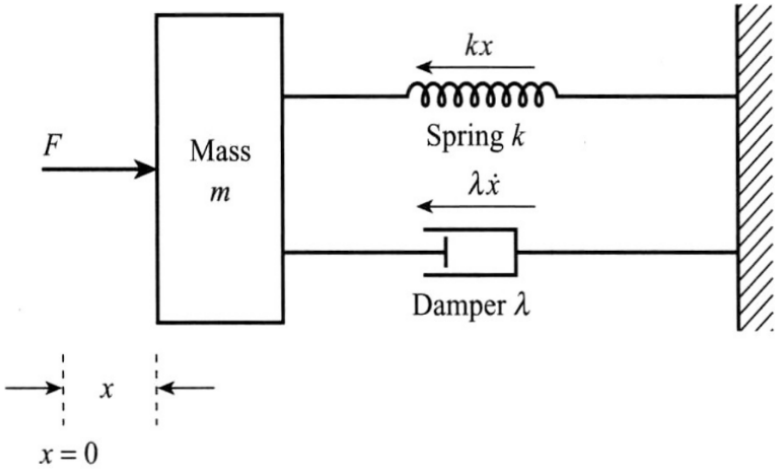
### 2.1.3 System model

- First order system, which consists of a damper and a spring.




$$F_{in} - F = \lambda \frac{dx}{dt}$$

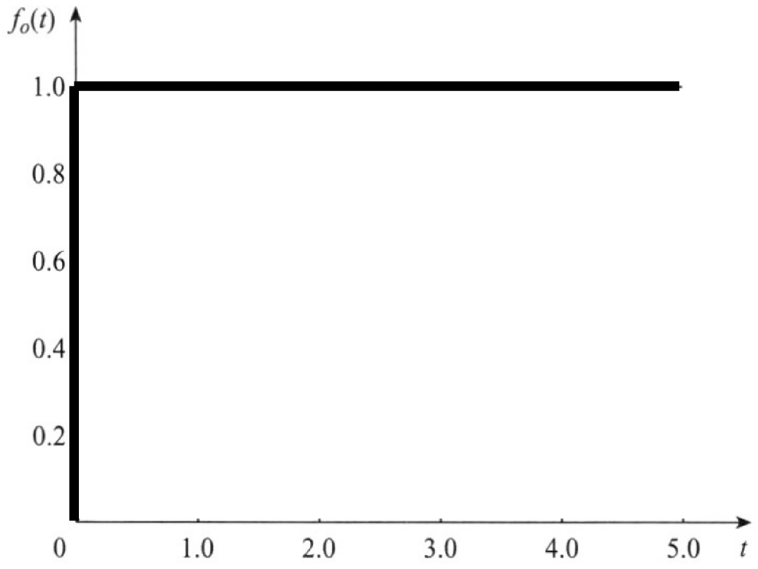
- Second order system, which also consists of a damper and a spring, but the mass is no longer negligible.



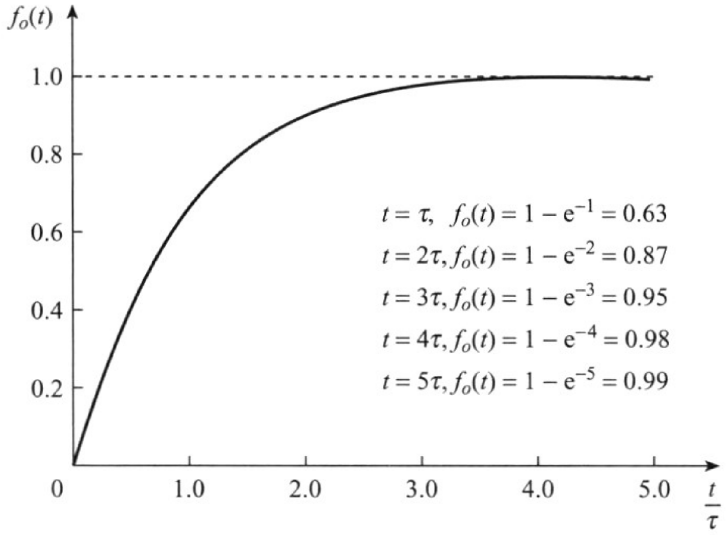

$$m\ddot{x} + \lambda\dot{x} + kx = F$$

### 2.1.4 Static and dynamic responses

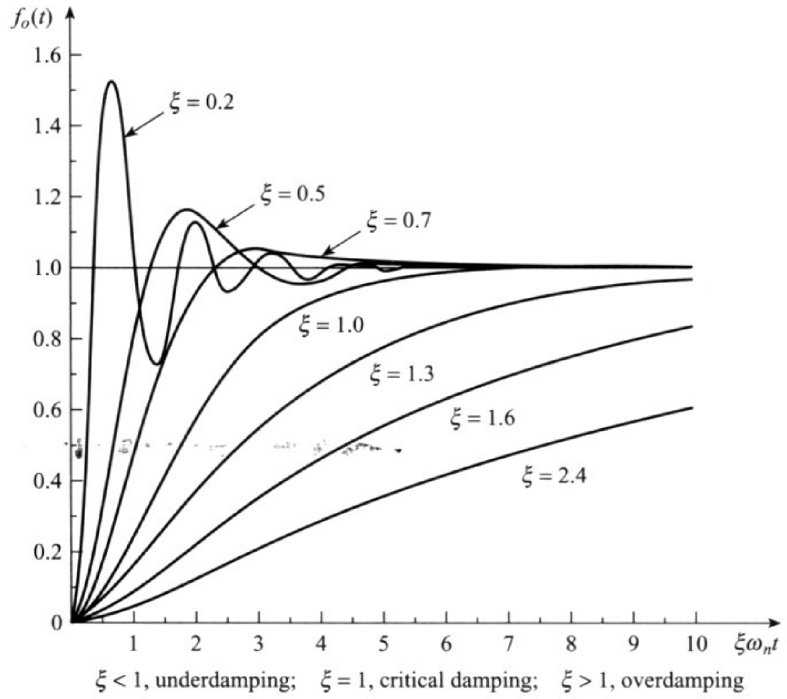
- Step response A step response is like dropping an accelerometer to induce a sudden acceleration. This step response is essentially the input into the mechanical system.




- 1st order A 1st order system will behave in response to a step response as shown in the graph below:



- Second order A second order system will behave in response to a step response as shown in the graph below:




$\xi < 1$, underdamping; $\xi = 1$, critical damping; $\xi > 1$, overdamping

## 2.2 Input signal conditional and interfacing

### 2.2.1 Discrete circuits

Discrete circuits are circuits made up of discrete components, like resistors, capacitors, etc.

### 2.2.2 Amplifiers

Amplifiers amplify the signal to improve signal-to-noise ratio (SNR).

### 2.2.3 Filters

Filters get rid of unwanted signal contents, e.g. high-pass, low-pass, band-pass, band-stop, etc.

### 2.2.4 Analogue to digital converter

Analogue to digital converters convert analogue signals like the turning of a knob, into a digital signal.

## 2.3 Output signal conditioning and interfacing

- Filters
- Output amplifiers
- Digital to analogue converter (DAC)
- Etc.

## 2.4 Graphical displays

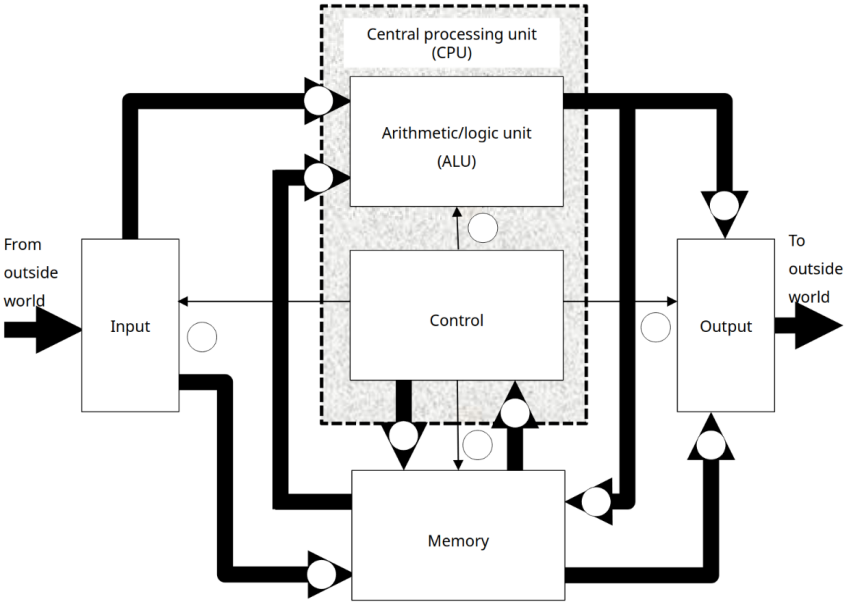
- Light Emitting Diodes (LED)
- Liquid Crystal Display (LCD)

## 2.5 Digital control architectures

- The digital control architecture is the "brain" of the system, and is usually a microcomputer.
- Microprocessor (MPU) refers to the Central Processing Unit (CPU) on a single integrated-circuit (IC) chip.
- Microcomputer refers to the MPU combined with the memory (RAM) and input and output unit (I/O unit)
- Microcontroller (MCU) refers to the microcomputer in a single integrated-circuit (IC) chip.
- Function: Logic, Arithmetic, Control, Timing and Sequencing, Communications, etc.

# 3 Basic computer structure

All computers contain 3 basic units.




## 3.1 Central Processing Unit (CPU)

- Arithmetic and Logic Unit (ALU)
- Control Unit

## 3.2 Memory Unit

- Internal memory, like Random Access Memory (RAM), Read-only Memory (ROM).
- External memory, like Hard Disk Drives (HDD), Cassette Tape, etc.

## 3.3 Input/Output Unit (I/O Unit)

- Input unit - Mouse, keyboard (K/B), Analogue to Digital Converter (ADC)
- Output unit - Display, Digital to Analogue Converter (DAC)

# 4 Design tips for mechatronics systems

1. Understand the task and define the problem
2. Sketch a function block diagram
3. Decide and select mechatronics components (type, number, communication protocol, etc.):
    - Digital control architecture
    - Sensors and input interfacing
    - Actuators and output interfacing
    - Display
4. Look for the components with suitable specifications and ensure compatibility of components when they interface with each other
5. Construct hardware prototypes
6. Program software or firmware

# 5 Examples of mechatronics systems

## 5.1 Car

A typical car contains over 50 microprocessors for the systems listed below:

- Fuel and fluid alert system
- Airbag system
- Entertainment and navigation systems
- Speed control system
- Combustion engine control
- Automatic gearbox
- Active stabilisation
- Anti-lock braking system (ABS)
- Climate control (air-conditioning)
- Seatbelt alert system
- Door security system

## 5.2 Inkjet printer

### 5.2.1 Actuators

- Stepper motor to drive print head mechanism
- Stepper motor to drive paper feeding mechanism
- Stepper motor to park print head mechanism (some models)
- Ink firing at nozzles
  - Piezoelectric (Epson) or Thermal resistor (Canon and HP)

### 5.2.2 Sensors in paper feeding mechanism

- Proximity sensors or limit switch to detect presence of paper
- Proximity sensors to detect paper loaded in position at the start of a print

### 5.2.3 Sensors in print head mechanism

- Linear encoder for precision positioning of print head
- Limit switch to detect print head in parked position

### 5.2.4 Digital control architecture

- Microcontroller based
- Communications
  - USB (parallel) port
  - Bluetooth
  - Local Area Network (LAN) port
  - Wi-Fi
- Graphics display is a simple LCD to high definition LED

## 5.3 Robots

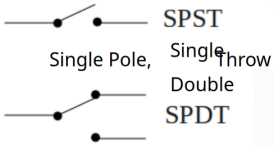
- Serial robot
- Parallel robot

# 6 Sensors

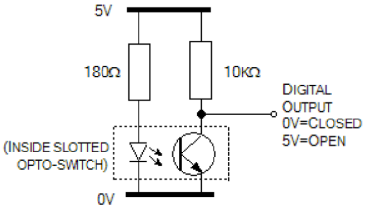
## 6.1 Digital sensors

### 6.1.1 Switches

Toggle Switch

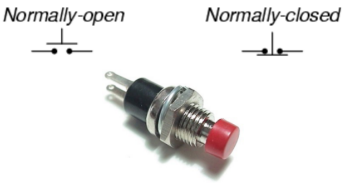



Slotted Opto Switch

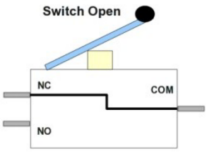


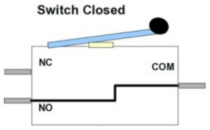

Limit Switch


Pushbutton switch




| LIMIT SWITCHES | |
|---|---|
| Normally Open | Normally Closed |
| | |
| Held Closed | Held Open |



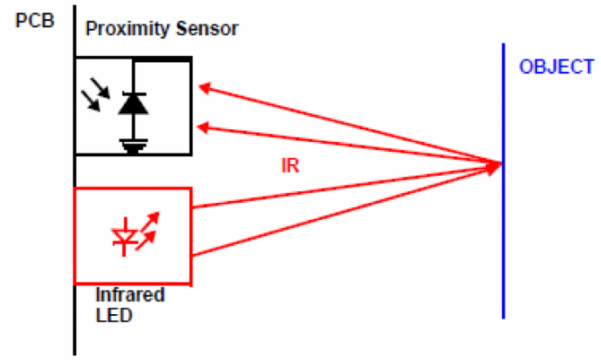



### 6.1.2 Proximity sensor

A proximity sensor detects the presence of an object. Its output is either high, or low upon detection.

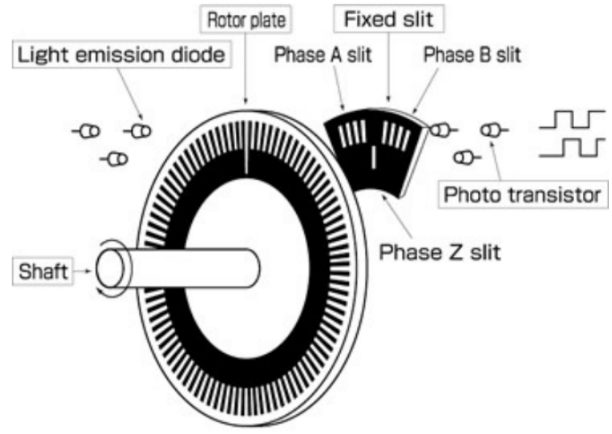
- Infrared and ultrasound sensors detect the incoming waves reflected by the object. An example is a car reverse sensor.
- Inductive sensors change the induced current upon detection of ferrous or magnetic materials. An example is a metal detector for security.
- Capacitive sensors change their capacitance upon detection of conductive materials. An example is a touch screen.

Infrared proximity sensor

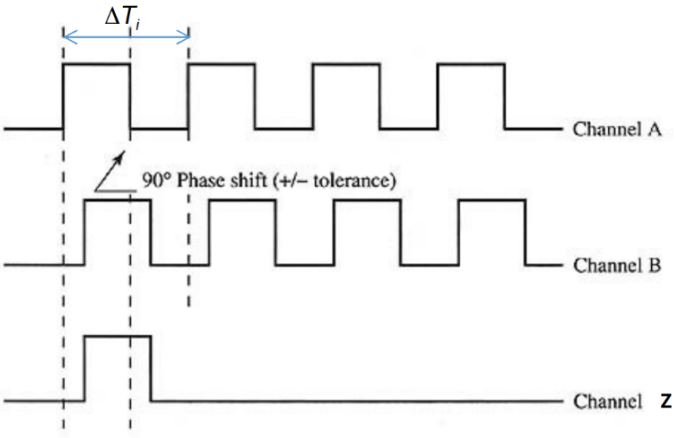



Inductive / Capacitive proximity sensor

### 6.1.3 Incremental encoders



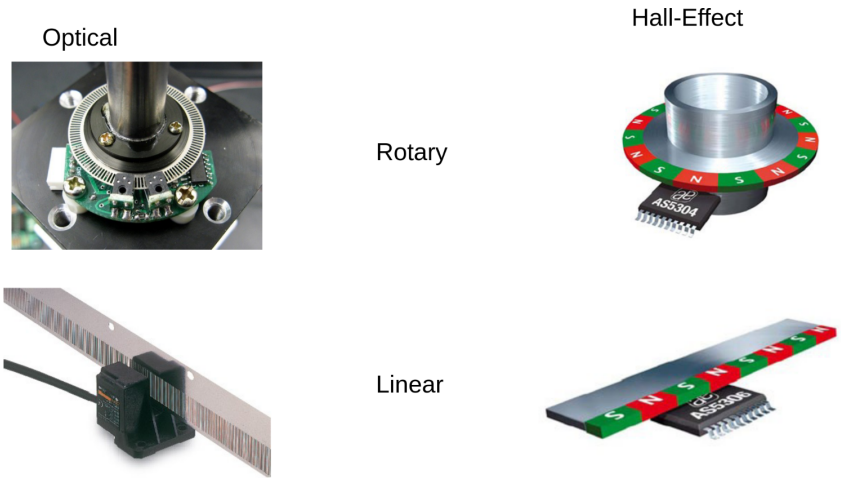

Incremental Encoder Simplified Structure




Incremental encoders have the same working principle as the slotted opto switch.

- When channel A **leads** channel B by 90°, the direction is anti-clockwise.
- When channel A **lags** channel B by 90°, the direction is clockwise.
- The time $\Delta T_i$ between 2 pulses determines the rotation speed, as:

### 6.1.4 Digital encoders




$$\omega_i = \frac{2\pi}{\Delta T_i}$$

## 6.2 Interfacing with digital sensors

### 6.2.1 Sensing modalities

- One modality is to sense a state, like high (1) or low (0).
- Another modality is to measure the time duration of a state.

### 6.2.2 Communication protocols

- Serial, where data is fed one after another.
  - There are two types of serial communication, synchronous and asynchronous.
  - Serial is cheaper but slower
- Parallel, where multiple streams of data is fed via multiple I/O pins.
  - It is more expensive, but also faster.

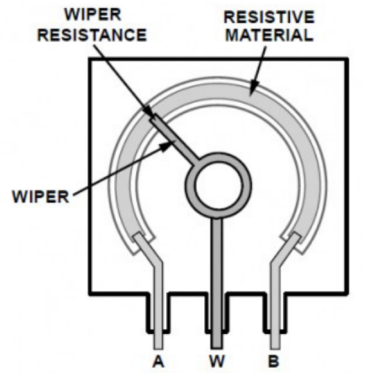
## 6.3 Analogue sensors

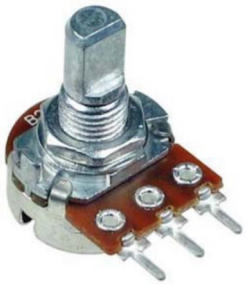
### 6.3.1 Potentiometer

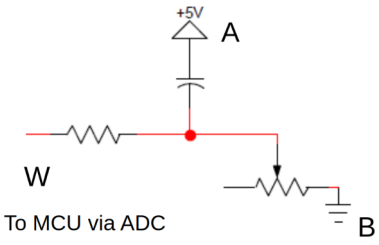
- A potentiometer is used to measure angular or linear displacement.
- It works based on the voltage divider principle where:

$$\text{Variable resistance} \propto \text{Variable voltage}$$

- Applications of potentiometers include light dimmers and the knob to adjust the audio volume.

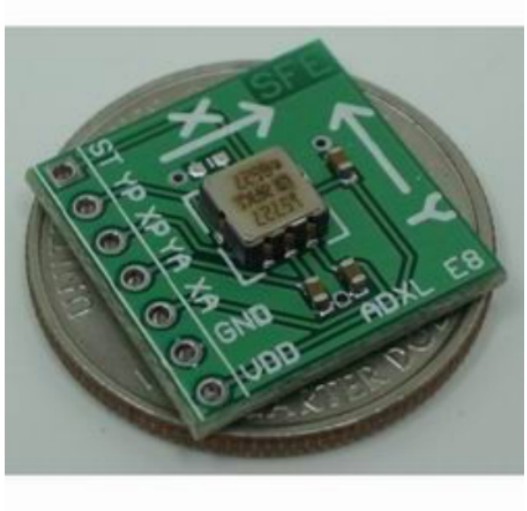


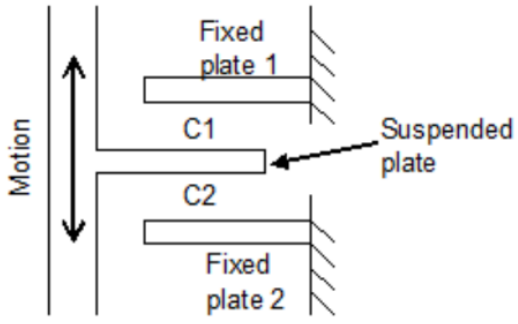





### 6.3.2 Accelerometer

- An accelerometer is used to measure linear acceleration from movement and gravity.
- For an analogue accelerometer, its output is a voltage value.
- For a digital accelerometer, its output is a duty cycle.
- It works because motion causes a change in the distance between the plates and hence a change in the capacitance of the accelerometer.
- Applications of accelerometers include mobile devices and inclinometer, which is an instrument used for measuring angles of slope, elevation, or depression with respect to gravity's direction.



# 7 Analogue to digital converters

## 7.1 Successive approximation

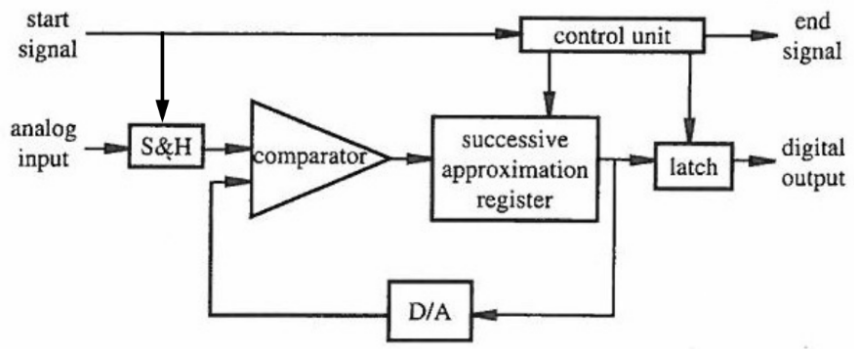
Successive is a fast, cheap and the most widely used method to convert analogue signals to digital signals.
The steps to convert an analogue signal to a digital signal using successive approximation:

1. With a start signal, analogue input is temporarily stored at the sample and hold (S&H) latch.

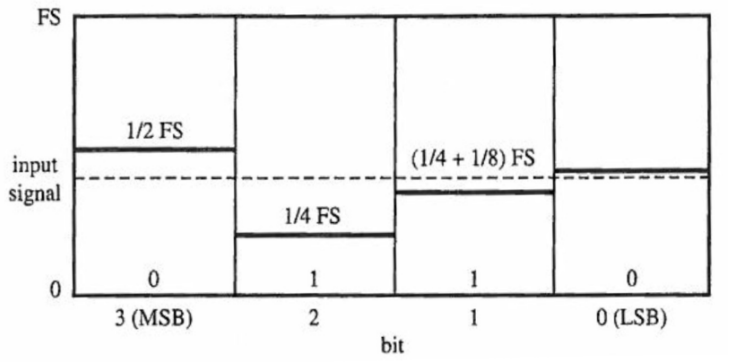
2. The control unit makes an approximation of a digital equivalent of the analogue input and hold the result at an output latch.

3. The digital to analogue convert converts the digital approximation to analogue signal compares it with the analogue input.

4. If both are equal, the result held at the latch is the output. Otherwise, it goes back to step 2 to make the next successive approximation iteration.



### 7.1.1 Example



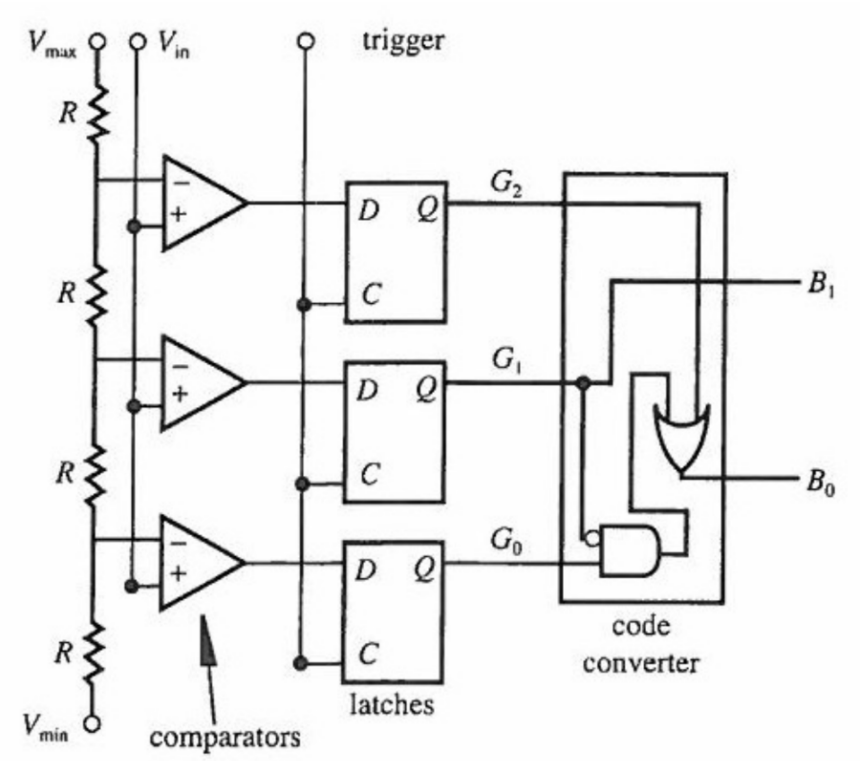

- Full Scale ($FS$) is 1.0 V
- Resolution ($n$) is 4 bits
- Analogue input is 0.4 V

Steps:

1. Control unit turns on only the most significant bit (MSB), which is Bit 3, that represents $\frac{1}{2}FS = 0.5\,\text{V}$. The first approximation is 1000 in binary.
2. It is larger than the analogue input given (0.4 V), hence Bit 3 is turned off.
3. Control unit turns on Bit 2 $\frac{1}{4}FS = 0.25\,\text{V}$. The second approximation is 0100 in binary.
4. Current value of $0.25\,\text{V} < 0.4\,\text{V}$, hence Bit 2 remains on.
5. Control unit turns on Bit 1 $\frac{1}{8}FS = 0.125\,\text{V}$. The third approximation is 0110 in binary.
6. Current value of $0.25\,\text{V} + 0.125\,\text{V} = 0.375\,\text{V} < 0.4\,\text{V}$. Bit 1 remains on.
7. Control unit turns on Bit 0 $\frac{1}{16}FS = 0.0625\,\text{V}$. The fourth approximation is 0111 in binary.
8. Current value of $0.375\,\text{V} + 0.0625\,\text{V} = 0.4375\,\text{V} > 0.4375\,\text{V}$. Bit 0 is turned off.
9. Final output is 0110 in binary, which is 0.375 V.

## 7.2 Flash converter

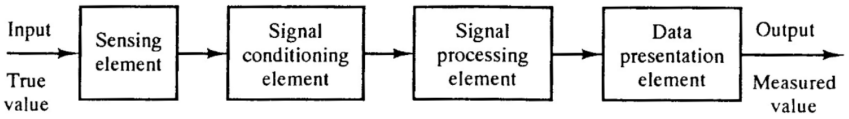
- A flash converter is a series of comparators acting in parallel to approximate an analogue input.
- Fastest type of analogue-to-digital converter.
- Increasing resolution does not increase conversion time.
- Flip-flop latches produce 3-bit code.
- AND & OR gates to convert 3-bit code to binary output.




| State | Code ($G_2$ $G_1$ $G_0$) | Binary ($B_1$ $B_0$) | Voltage Range (V) |
|---|---|---|---|
| 0 | 000 | 00 | 0 to 1 |
| 1 | 001 | 01 | 1 to 2 |
| 2 | 011 | 10 | 2 to 3 |
| 3 | 111 | 11 | 3 to 4 |

$$B_0 = G_0 \cdot G_1 + G_2$$
$$B_1 = G_1$$

# 8 Data acquisition (DAQ) process




## 8.1 Sensing element

The sensing element is the element in contact with the process and gives an output which depends in some ways on the variables to be measured.
Examples:

- Thermocouple: Measured E.M.F in mV depends on temperature.
- Strain gauge: Its resistance depends on mechanical strain.

## 8.2 Signal conditioning element

The signal conditioning element takes the output of the sensing element and converts it into a form more suitable for further processing, usually a DC voltage, a DC current or frequency signal.
Examples:

- An amplifier which amplifies mV to V.
- Wheatstone bridge that converts impedance change into voltage change.

## 8.3 Signal processing element

The signal processing element takes the output of the signal conditioning element and converts it into a form more suitable for presentation.
Examples:

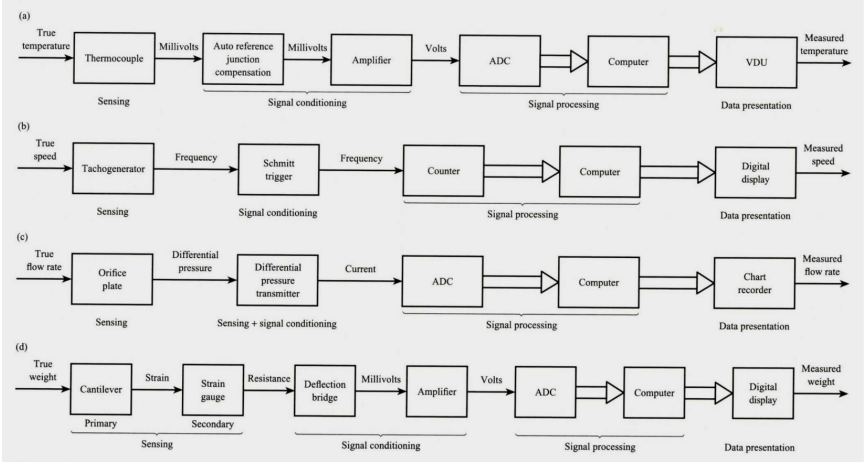
- Analogue-to-Digital converter which converts a voltage into a digital form.
- Computer which calculates the desirable measurement from raw data, like:
  - The mass of gas from flow rate and density.
  - Correction for non-linearity.

## 8.4 Data presentation element

The data presentation element presents the measured value in a form which can be easily recognised by the observer.
Examples:

- Visual Display Unit (VDU)
- Dial or pointer-scale indicator

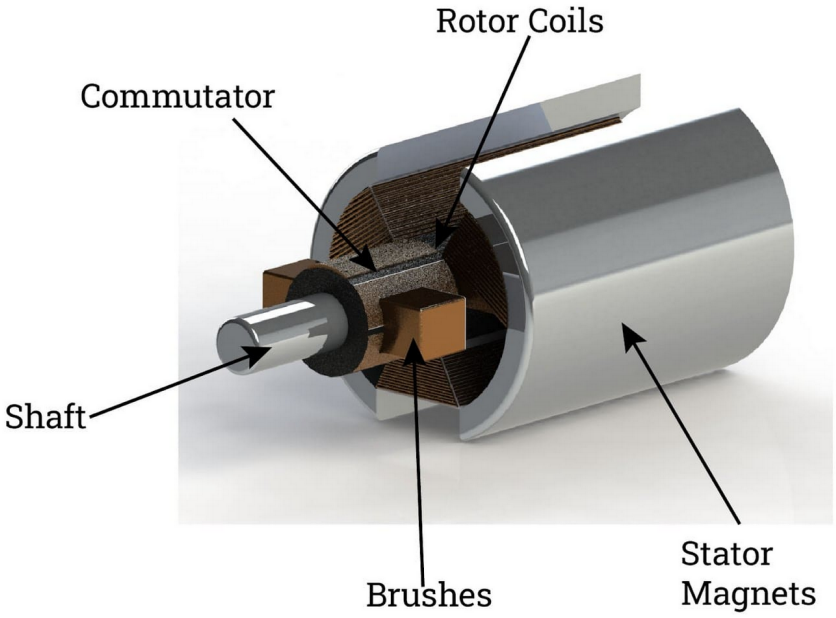
## 8.5 Examples of the DAQ process



# 9 Direct current motor

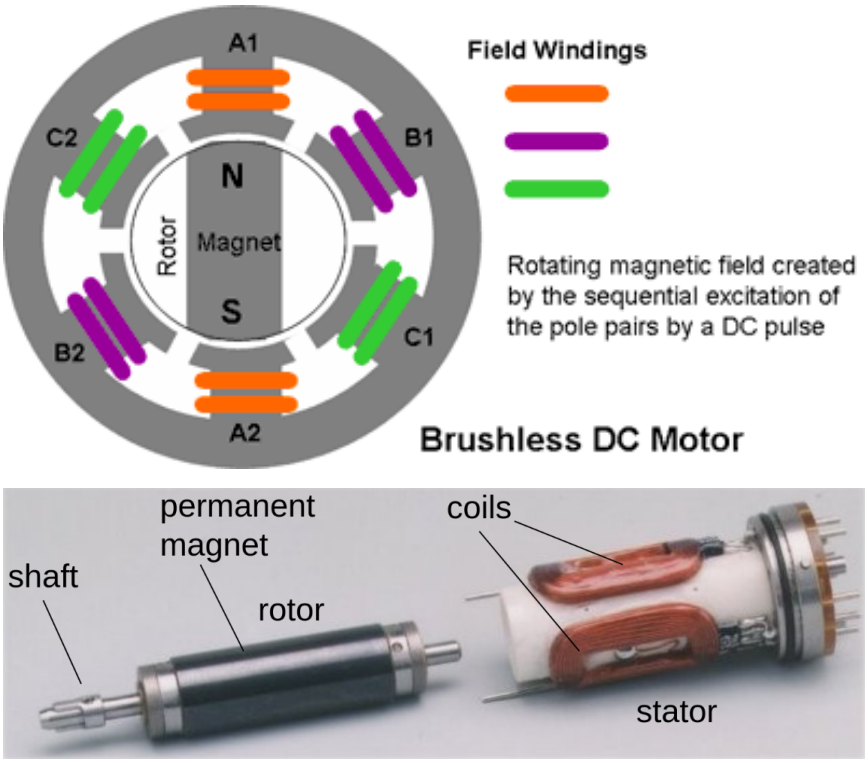
## 9.1 Permanent magnet or brushed DC motors

- The stator, which is the external shell containing the components of the DC motor, is fixed in place.
- The rotor, which is the rotating internal part inside the stator, rotates to produce the movement.
- The motor works based on Fleming's left-hand rule.



## 9.2 Brushless DC motor

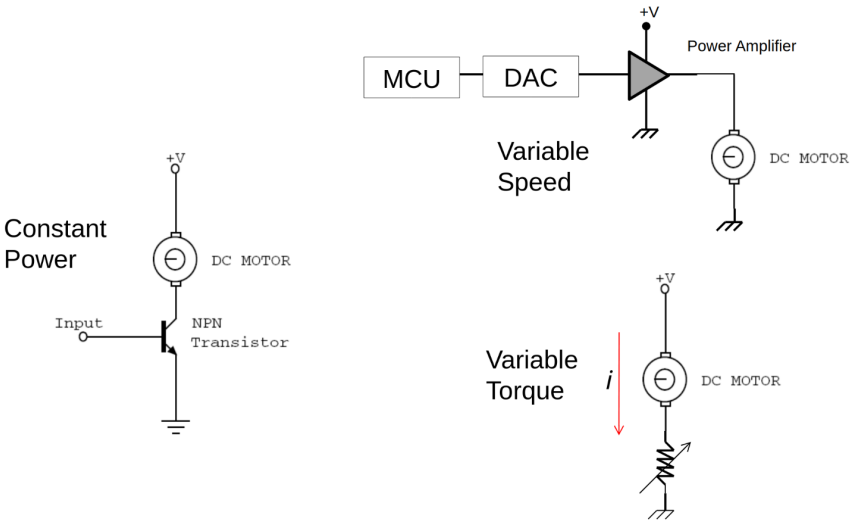
- It has the same body as a brushed DC motor, with a permanent magnet rotor on the inside and a fixed stator with rotating magnetic fields.
- It needs to know the exact angular position of the rotor to excite the correct coils. This is usually done with Hall effect sensors.




## 9.3 Brushless vs brushed DC motors

| Brushless | Brushed |
|---|---|
| Simple maintenance | Low cost thanks to the simple construction and control, and only two wires are needed. |
| High efficiency as there is no drop in voltage across the brush, and has low electrical noise | More robust in harsh environments as there are no electronic components |
| Higher speed range | |
| Reduced size | |

## 9.4 Controlling DC motors




### 9.4.1 Motion control fundamentals

- Power $= VI = \tau\omega$
- DC motor control
  - Voltage controls velocity: $V \propto \omega$
  - Current controls torque: $I \propto \tau$

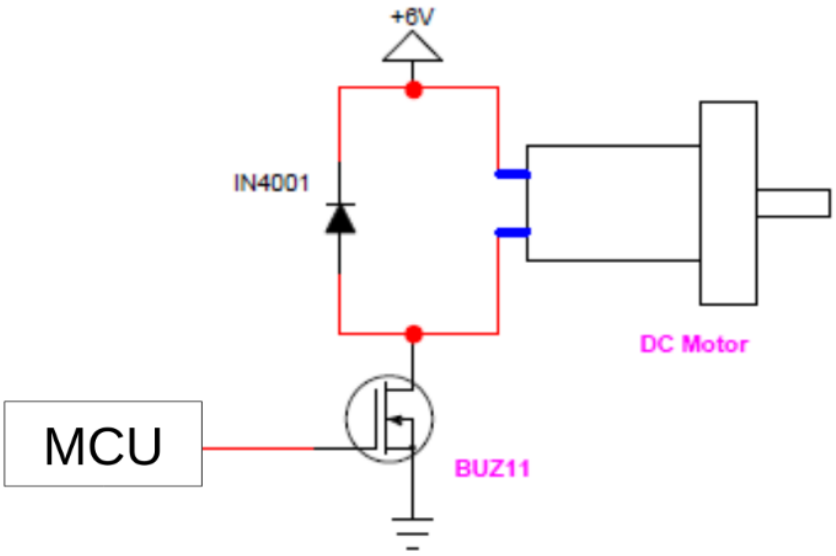
### 9.4.2 Power amplifiers

Using power amplifiers is possible but is typically avoided. There is usually large power dissipation, and the amplifier usually overheats.

### 9.4.3 Digital-to-analogue converter

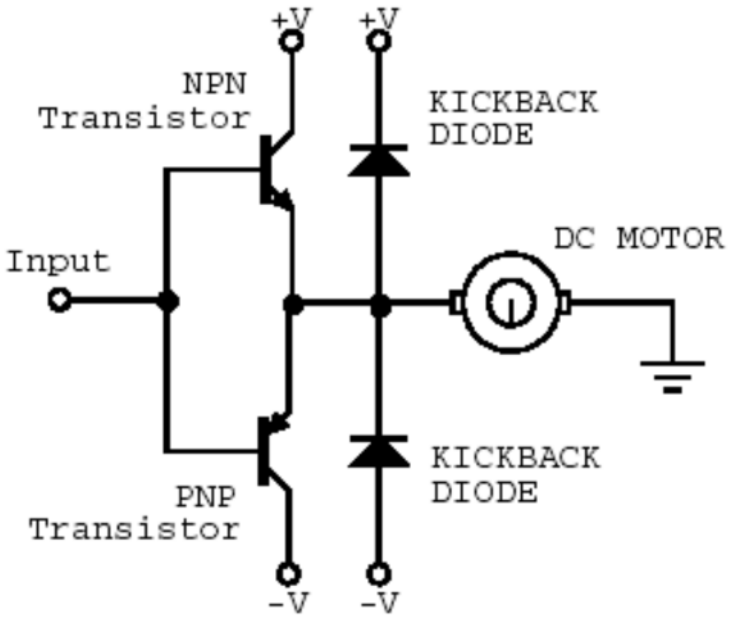
Digital to analogue converters (DACs) are expensive, so most MCUs are not equipped with a DAC.

### 9.4.4 Uni-directional PWM control

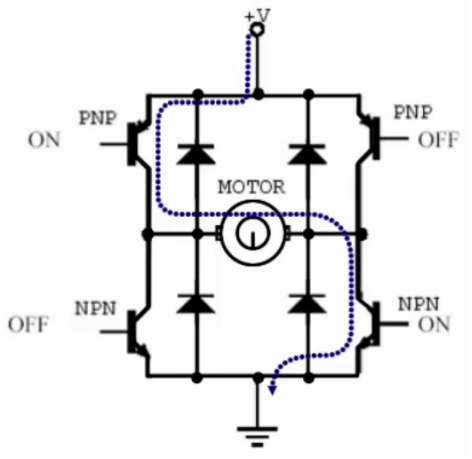



### 9.4.5 Bidirectional PWM control

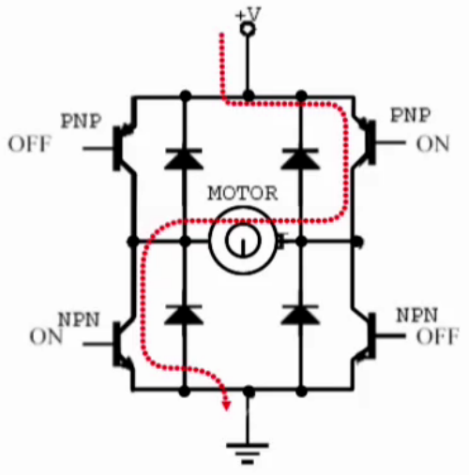
Below is a bidirectional DC motor control using a dual power supply.



### 9.4.6 Bidirectional PWM control with H-bridge circuit




Clockwise direction




Anti-clockwise direction

# 10 Process and instrument control I/O

- The microcontroller operates automatically and continuously under the control of the program stored in the ROM, no human intervention is required.
  - The operation is changed only by changing the content of the ROM.
- During execution, the microcontroller receives data from devices monitoring some physical states (temperature, speed, etc.), operates on the data and sends data or control signals to the process or instrument via output devices.

### 10.0.1 Examples

- Traffic red-light camera
- Car airbag system
- Fire and burglar alarm systems

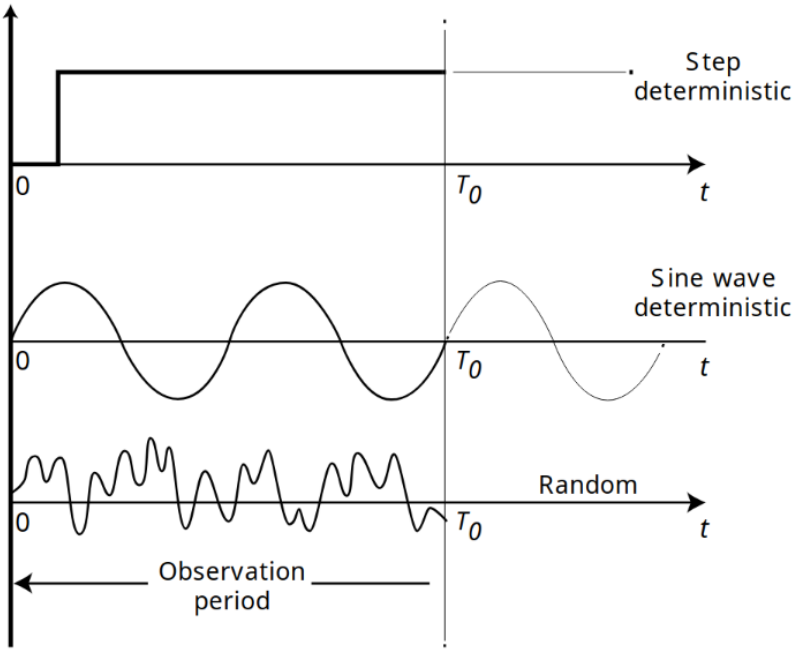
# 11 Keyboard entry and display I/O

- Communication with human operators
- The microcontroller executes a keyboard monitoring program stored in ROM.
  - It reads the keyboard continuously until a key is actuated, then determines the actuated key, and executes the appropriate instructions.
- Once the instructions are executed, it gets back to the keyboard monitoring program.

### 11.0.1 Examples

- Point-of-sale machines
- Lift
- Television

# 12 Deterministic and random signals




## 12.1 Deterministic signals

Deterministic signals are signals with values that can be predicted exactly, after an observation period $T_0$.

## 12.2 Random signals

Random signals are signals that cannot be predicted exactly, after an observation period $T_0$. Essentially, the signal cannot be represented by a continuous algebraic equation $y(t)$ for the signal $y$ at time $t$.

## 12.3 Randomness

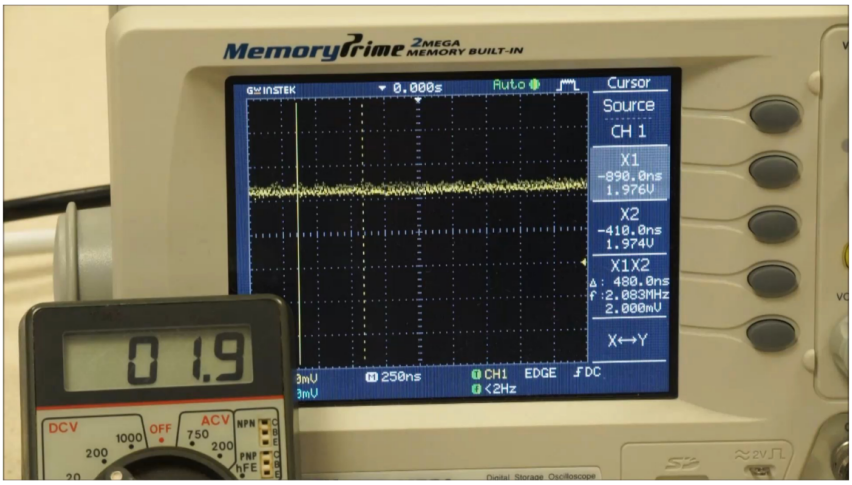
- A real process has many parameters that cannot be exactly known, because of the randomness of nature.
- An absolutely clean signal does not exist if the resolution is allowed to be infinitesimally small.
- Observed randomness is dependent on resolution.

### 12.3.1 Leaves falling from a tree example

With a resolution of:

- $1\,\mathrm{m}^2$, the outcome is noisy
- $100\,\mathrm{m}^2$, the outcome is clean

### 12.3.2 Voltage source example




With a resolution of:

- 0.1 V, the outcome is clean
- 0.001 V, the outcome is noisy

# 13 Sources of noise

## 13.1 Internal noise sources

- Johnson or thermal noise is random, temperature-induced motion of electrons and other charge carriers in resistors and semiconductors that give rise to a corresponding random voltage. The white noise is proportional to the absolute temperature in Kelvin (K).
- Shot noise is the random fluctuations in the rate at which charge carriers diffuse across a junction of a transistor. It is another source of white noise.

### 13.2 External noise and interference sources

- AC power circuits operating at $220\,\mathrm{V}, 50\,\mathrm{Hz}$ (US: $110\,\mathrm{V}, 60\,\mathrm{Hz}$) produce "mains pick-up" or "hum" which is a corresponding sinusoidal interference signal in the measurement circuit.
- Fluorescent lighting arcing at 2 times per cycle of the AC power. Arcing is the process of raising the potential to cause electrical current to flow between the anode and cathode through the inert gases inside the tube of a fluorescent light.
- Radio frequency (RF) interference. Transmitters, welding equipment and electric arc furnaces can produce interference at frequencies of several MHz.

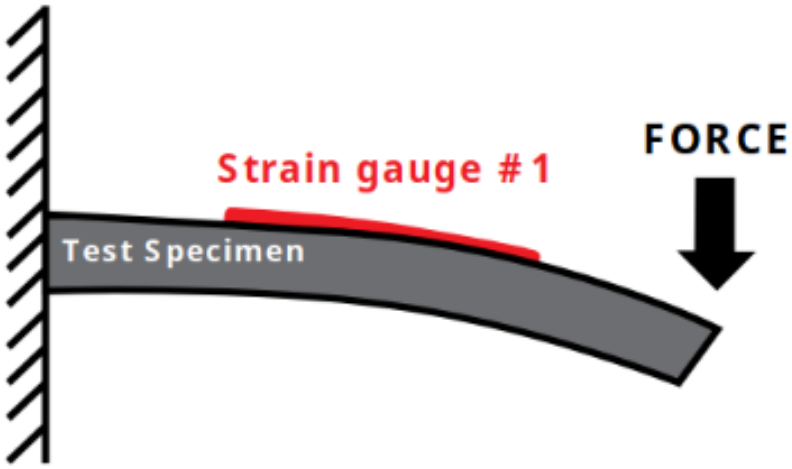
## 14 Input signal conditioning elements

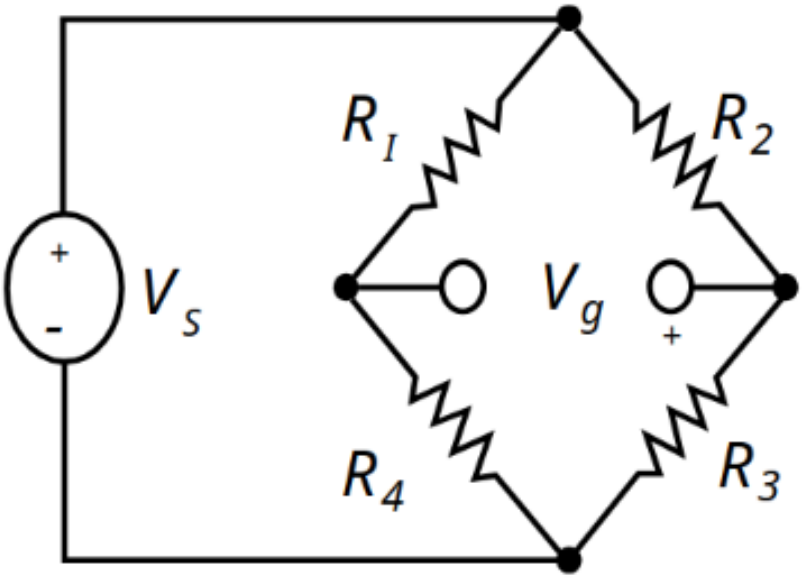
- Types of input signals that can be conditioned:
  - DC voltage
  - DC current
  - Variable frequency AC voltage
- Examples:
  - Deflection bridges
  - Operational amplifiers (Op-Amp)
  - Filters

### 14.1 Deflection bridges

- Deflection bridges measure the deflection of a material by using strain gauges that change their resistance when there is strain on the gauge.

### 14.1.1 Quarter bridge



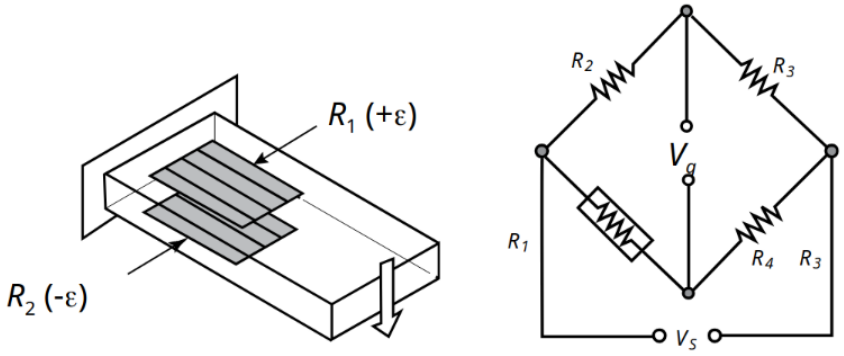




The voltage output is given by $V_g$:

$$V_g = V_s \left( \frac{1}{1 + \frac{R_4}{R_1}} - \frac{1}{1 + \frac{R_3}{R_2}} \right)$$

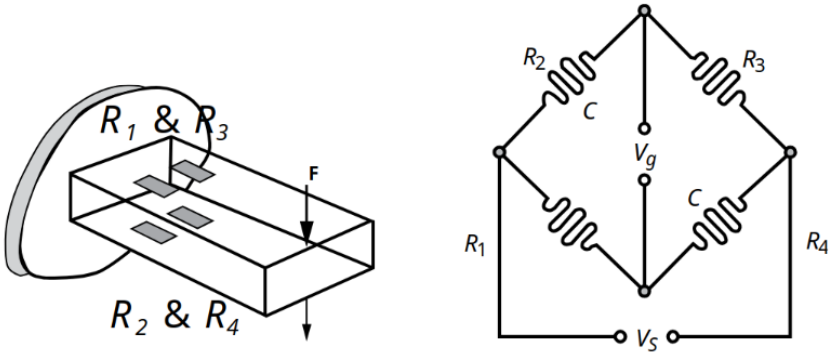
### 14.1.2 Half bridge

Half bridges have double the sensitivity of the quarter bridges.




### 14.1.3 Full bridge

Compared to half bridges, full bridges simplify the equation used to calculate the voltage output, and their output is more linear.

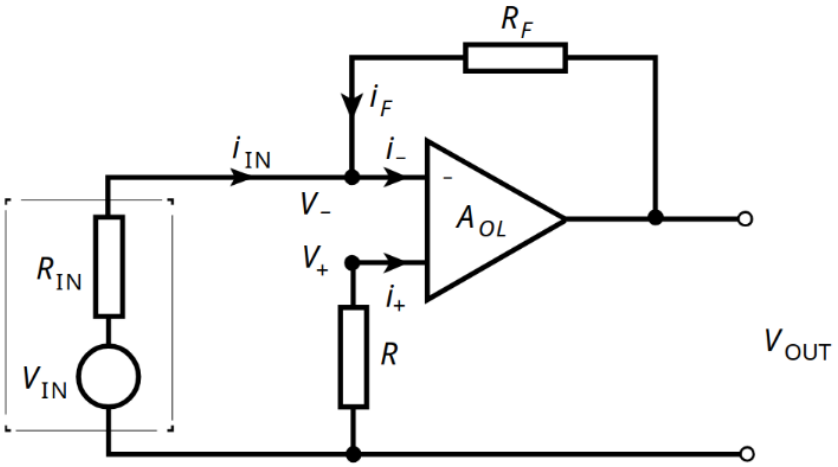


## 14.2 Operation amplifiers (Op-Amp)

- Operational amplifiers (op-amps) are used as the basic building blocks for instrumentation and power amplifiers.
- Types of op-amps:
  - Inverting
  - Non-inverting
  - Voltage follower
  - Voltage adder
  - Differential

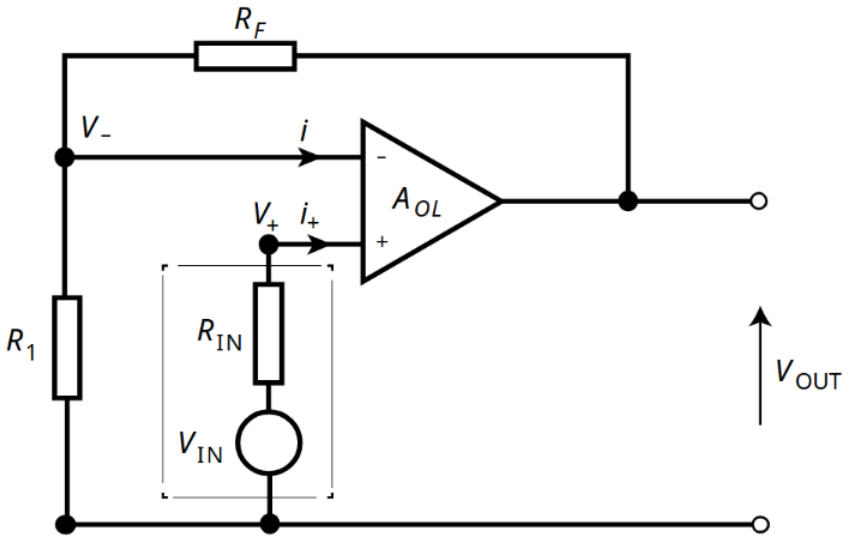
### 14.2.1 Inverting amplifier

The voltage input is being amplified by the ratio $\frac{R_F}{R_{IN}}$ and is inverted.




$$V_{OUT} = \frac{-R_F V_{IN}}{R_{IN}}$$

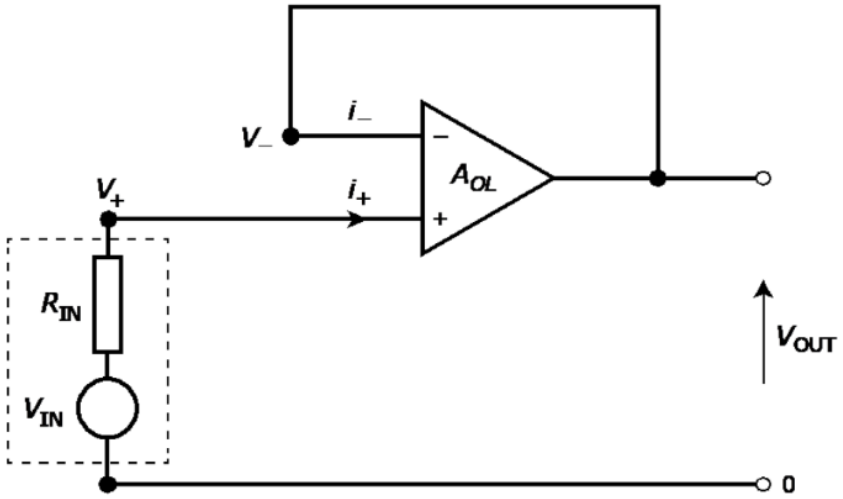
### 14.2.2 Non-inverting amplifier




$$V_{OUT} = \left(1 + \frac{R_F}{R_{IN}}\right)$$

### 14.2.3 Voltage follower

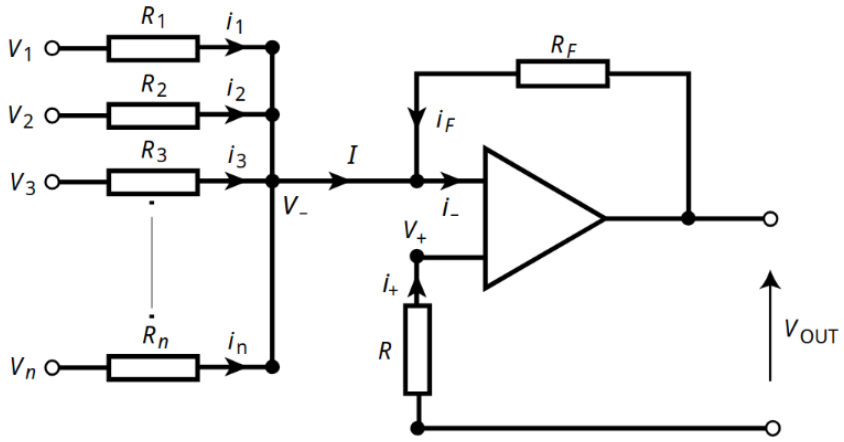
Voltage followers increase the current passing through them while keeping the voltage from the input.




$$V_{OUT} = V_{IN}$$

### 14.2.4 Voltage adder

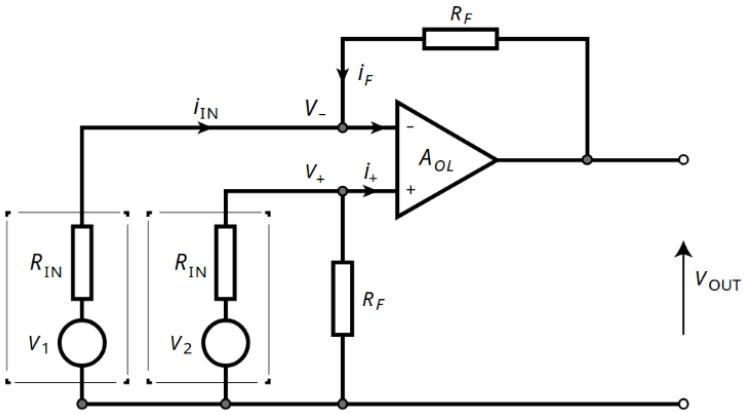
Voltage adders add up all the input voltages and amplify the sum of the input voltage.

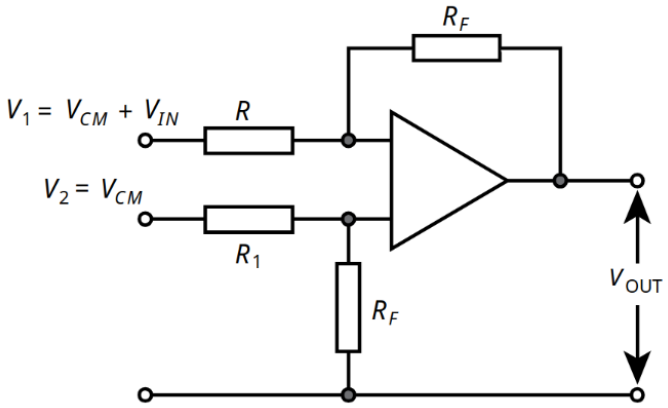



$$V_{OUT} = -R_F \left( \frac{V_1}{R_1} + \frac{V_2}{R_2} + \cdots + \frac{V_n}{R_n} \right)$$

### 14.2.5 Differential amplifier

- Differential amplifiers amplify the difference between two voltage inputs.
- A useful application of differential amplifiers is in removing background noise from a person speaking, which is a type of interference called common mode interference.
- The microphone picks up both the person's voice and the background noise, then another microphone picks up only the background noise.
- The differential amplifier is then used to amplify the difference between the two signals from the microphones, resulting in a clean signal of the person's voice.

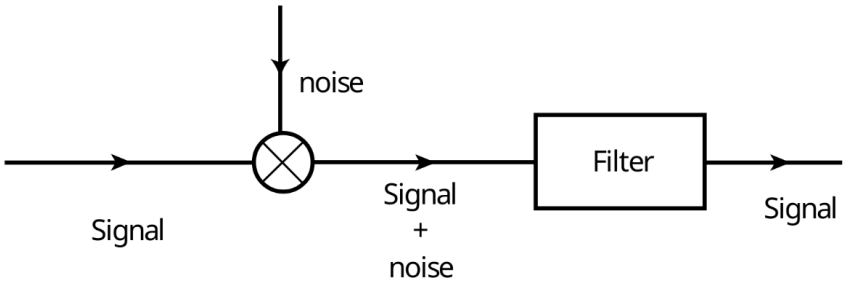


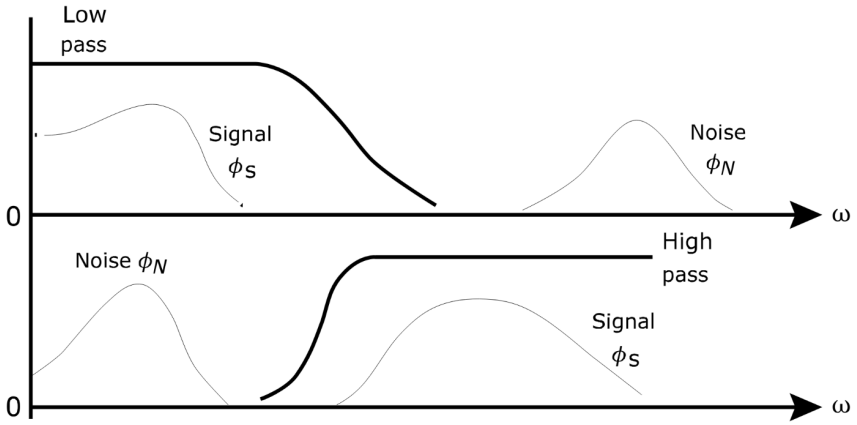

$$V_{OUT} = \frac{R_F}{R_{IN}} (V_2 - V_1)$$

## 14.3 Filters

- A frequency selective filter is an element which transmit a certain selected range of frequencies and rejects all others.
- An analogue filter is a network of resistors, capacitors and op amps to process continuous signals.
- A digital filter is a computer programmed to process sampled values of a signal.
- RC filters refer to filters that have both a resistor and a capacitor.

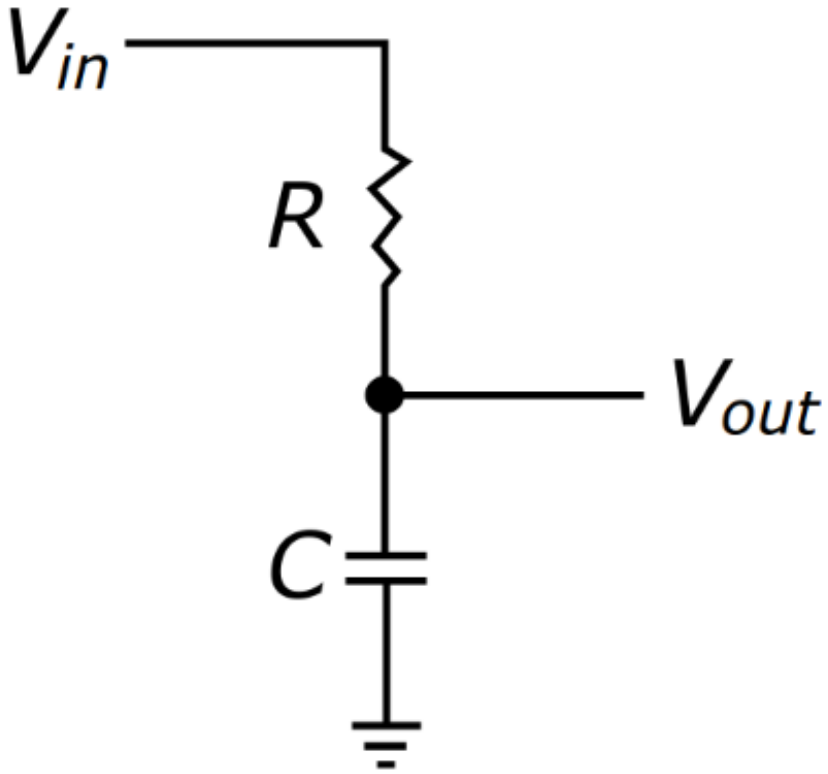



### 14.3.1 Low and high pass filters (RC filters)



### 14.3.2 Passive low pass filters

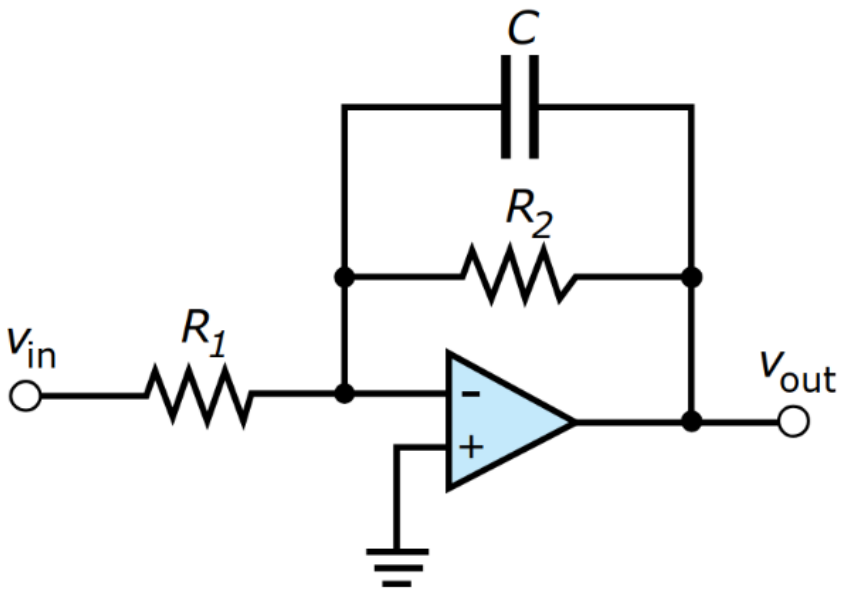
Passive low pass filters block all frequencies **above** the cut-off frequency $f_c$.




$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$

### 14.3.3 Active low pass filters

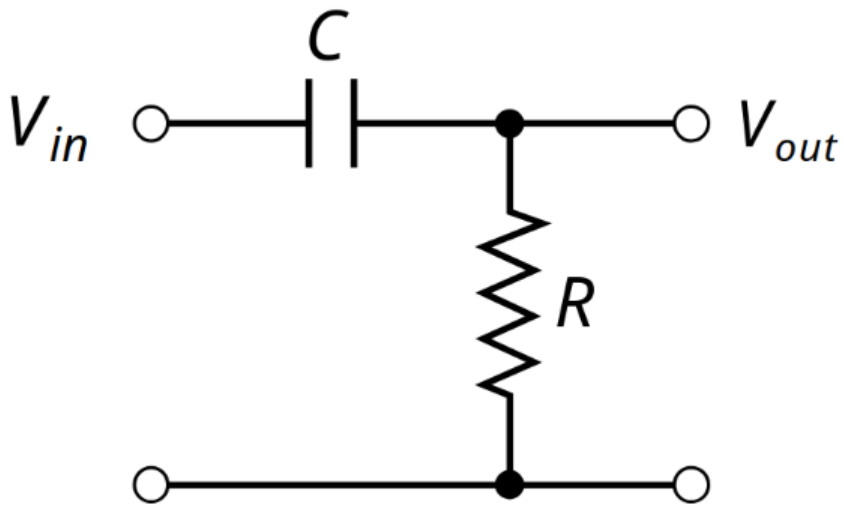
Active low pass filters behave exactly the same as passive ones, just that they increase the current passing through them as they have an op-amp in the circuit. They also block all frequencies **above** the cut-off frequency $f_c$.




$$f_c = \frac{1}{2\pi R_2 C}$$

### 14.3.4 Passive high pass filters

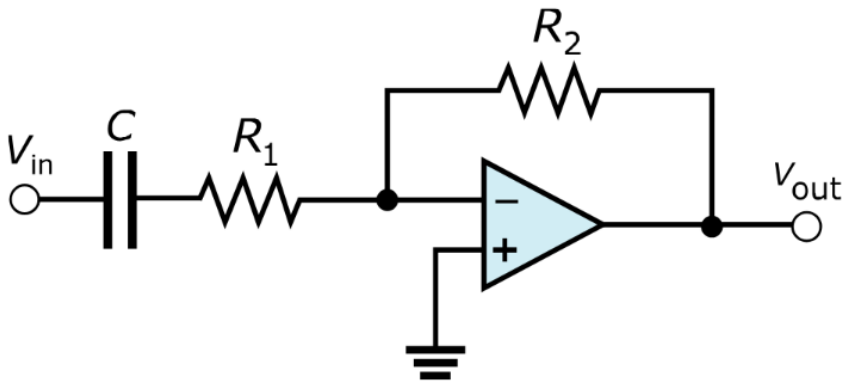
Passive high pass filters block all frequencies **below** the cut-off frequency $f_c$.




$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$

### 14.3.5 Active high pass filters

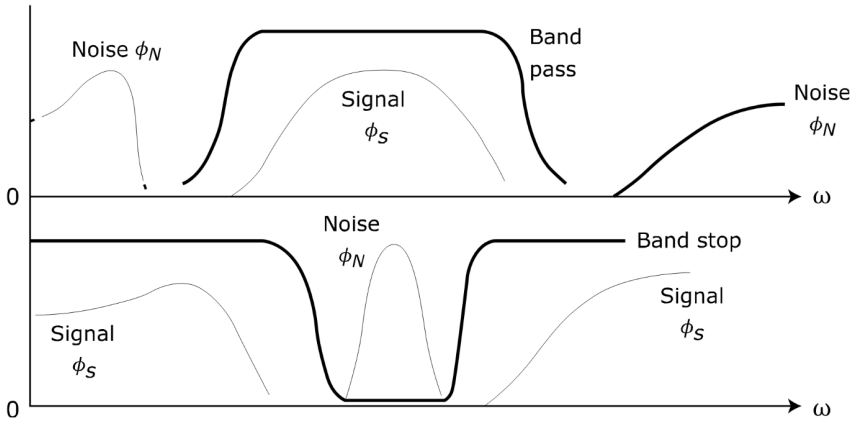
Active low pass filters behave exactly the same as passive ones, just that they increase the current passing through them as they have an op-amp in the circuit. They also block all frequencies **below** the cut-off frequency $f_c$.



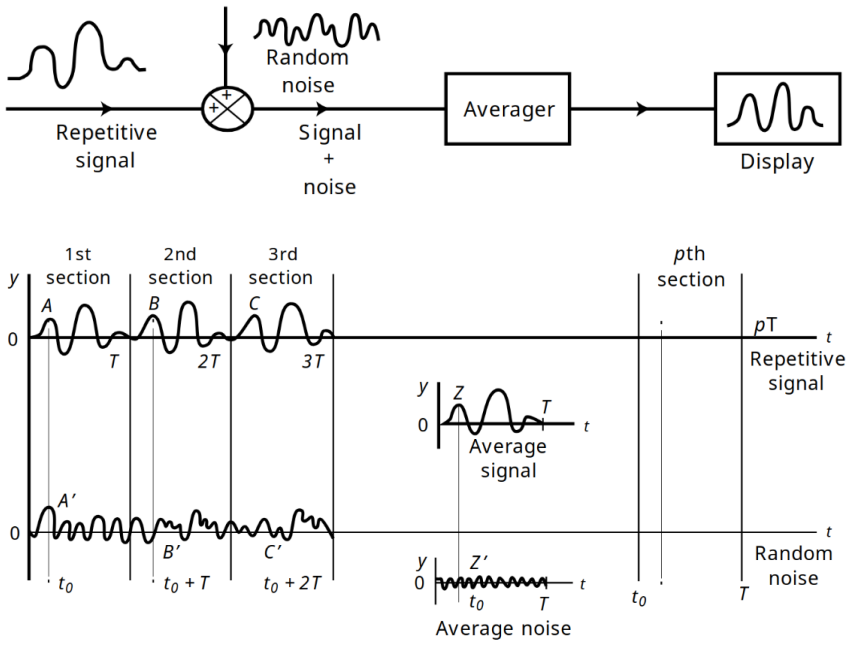

$$f_c = \frac{1}{2\pi R_1 C}$$

### 14.3.6 Band pass and band stop filters

- Band pass filters allow frequencies within the band to pass through, blocking out all other frequencies.
- Band stop filters block out frequencies within the band, allowing all other frequencies to pass through.



## 14.4 Averaging




For a repetitive measure signal affected by random noise, suppose that:

- It has a period $T$, a total of $p$ cycles and $N$ samples in each cycle, giving $pN$ samples in total.
- For each sample, there are $p$ number of corresponding samples from each cycle.
- The average value of the $i$-th sample is:

$$y_i^{AV} = \frac{1}{p} y_{i_1} + y_{i_2} + \ldots + y_{i_p}, i = 1, \ldots, N$$

Since the noise is random, that means it has zero mean, which means that taking the average of the signal over time will effectively remove the noise.

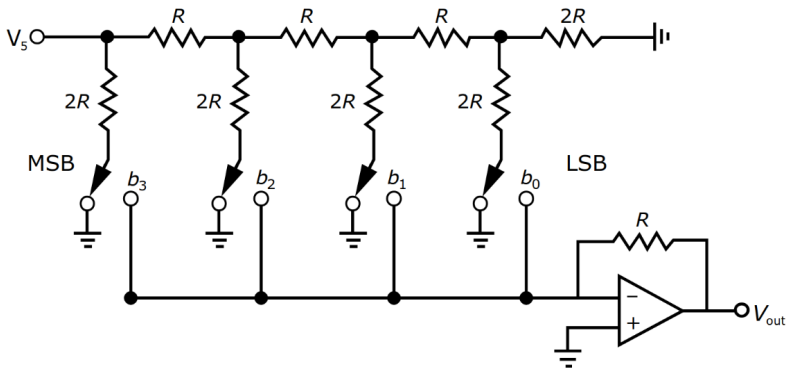
# 15 Output signal conditioning elements

Output signal conditioning elements are to convert the output of the MCU into a form suitable for interfacing with the output devices.
Some suitable forms are:

- Analogue signal: voltage or current
- Higher voltage
- Higher current
- Alternating current

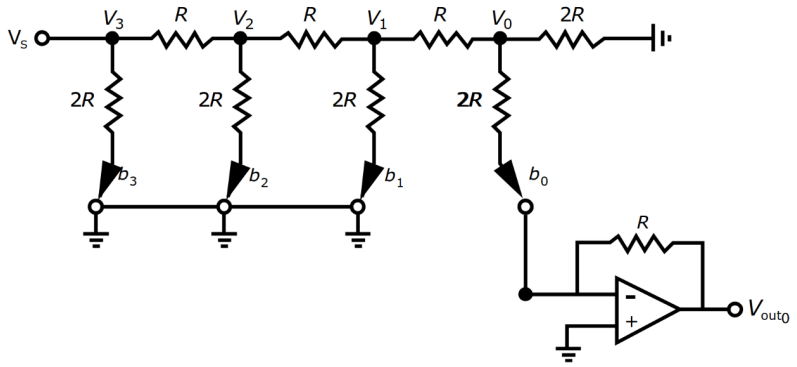
## 15.1 Digital-to-Analogue converter (DAC)

- The simplest type of digital-to-analogue converter is a resistor ladder network connected to an inverting op-amp circuit.



### 15.1.1 Example

- Consider a 4-bit output of 0001, the analogue circuit equivalent would be:




- Using voltage division:

$$V_0 = 0.5V_1 \quad V_1 = 0.6V_2 \quad V_2 = 0.5V_3$$

- Therefore, $V_0 = 0.125V_3 = 0.125V_s$.
- $V_0$ is the input to the inverting op-amp, which has a gain of $\frac{-R}{2R} = -0.5$.
- Therefore, the analogue output voltage of input 0001 is $V_{out_0} = -0.0625V_s$
- The analogue output voltage for the binary input is:
  - 0001 is $V_{out_1} = -0.0625V_s$
  - 0010 is $V_{out_1} = -0.125V_s$
  - 0100 is $V_{out_2} = -0.125V_s$
  - 1000 is $V_{out_3} = -0.125V_s$
- The output for any combination of the four bits is:

$$V_{out} = b_3V_{out_3} + b_2V_{out_2} + b_1V_{out_1} + b_0V_{out_0}$$

# 16 Arduino Uno MCU

## 16.1 Specifications

| | |
|---|---|
| Microcontroller | ATmega328 |
| Digital I/O | 14 (of which 6 provide PWM output) |
| Analogue Input | 6 |
| Microprocessor speed | 16 MHz |
| Flash (Memory) | 32 kB |
| SRAM (Memory) | 2 kB |
| EEPROM (Memory) | 1 kB |
| Operating voltage | 5 V |
| Input voltage | 7 - 12 V |
| Physical dimensions | 68.6 x 53.4 mm |
| Weight | 25 g |

## 16.2 Arduino Uno and ATmega328

Arduino Uno is an MCU and ATmega328 is also an MCU. Because Arduino uses OEM (Original Equipment Manufacturer) microcontrollers and re-engineers them into different architectures (i.e. memory, bus, I/O configuration, communication integrated chips (IC), etc.) for their own customised needs.

## 16.3 Number of bits

- ATmega328 is an 8-bit MCU, i.e. Arduino Uno is an 8-bit MCU
  - Amount of data processed at one time = 8 bits = 1 byte
- 16-bit MCUs would process twice the amount of data, while 32-bit MCUs would process 4 times the amount of data and 64-bit MCUs would process 8 times the amount of data
- N-bit MCU has:
  - N-bit word (variable) size, $2^N$ different possible values
  - N-bit instruction size, $2^N$ number of instructions or commands

## 16.4 Program execution speed

- Arduino Uno microprocessor speed is 16 MHz
- Program execution speed is not equal to the driven clock speed or microprocessor speed.
- The microprocessor runs at a higher frequency than all other components, i.e. memory, bus I/O, etc.
- Program execution speed is heavily influenced by the slowest components.

## 16.5 Memory

### 16.5.1 2 kB of RAM

This RAM is used to store runtime data.

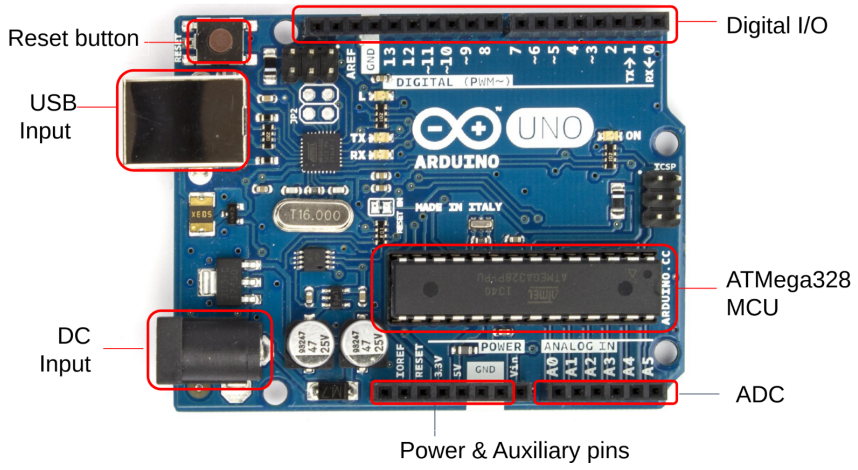
### 16.5.2 1 kB of EEPROM

This EEPROM is used to store settings and fixed parameters between resets.

### 16.5.3 32 kB of flash memory

- This flash memory acts like a hard drive to store programs and data.
- The programs uploaded into the Arduino are stored here.
- 0.5 kB is reserved for the bootloader program.

## 16.6 Components




### 16.6.1 Other components

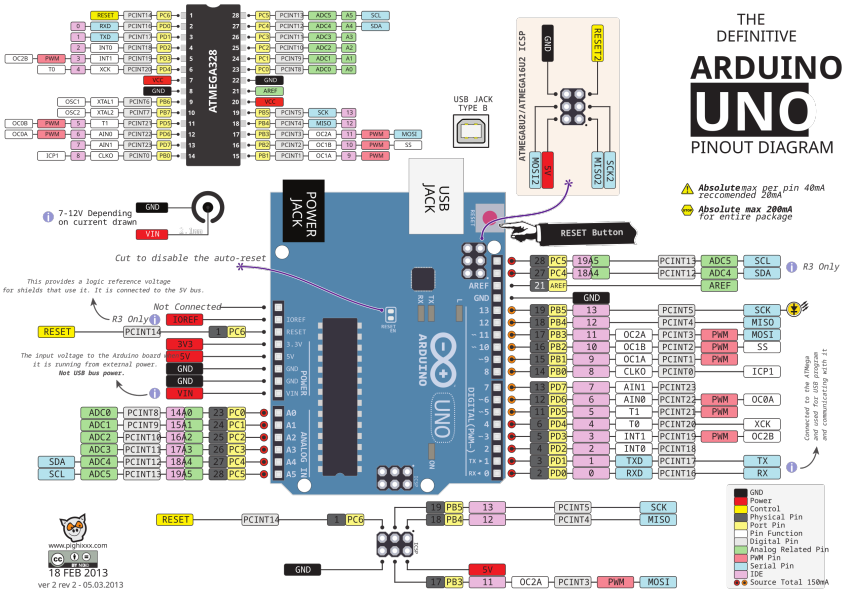
- General or Digital Input/Output pins
  - 14 pins in total
  - Digital signal has two states:
    - LOW, or 0, which refers to 0 V
    - HIGH, or 1, which refers to 5 V
- Analogue-to-Digital Converter (ADC) pins
  - 6 pins in total
  - These are used to convert analogue (continuous) signals into their digital equivalents.
- Power supply
  - The Arduino accepts 6 - 20 V of DC input, but 7 - 12 V DC is recommended.

## 16.7 Hardware interrupts

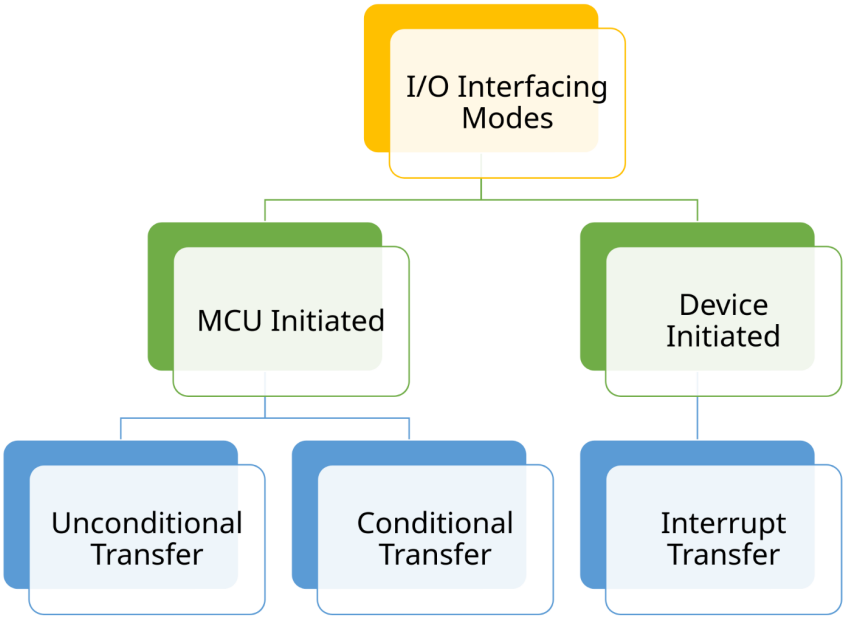
INT 0 pin (digital pin 2) and INT 1 (digital pin 3).
Function to attach the interrupt service routine:
`attachInterrupt(interrupt, ISR, mode)`

- `interrupt` is 0 (digital pin 2) or 1 (digital pin 3)
- `ISR` is the interrupt service routine, which is a function that takes no parameters and returns nothing
- Mode:
  - LOW means to trigger the interrupt when the interrupt pin is LOW
  - CHANGE means to trigger the interrupt when the pin changes state
  - RISING means to trigger the interrupt when the pin goes from LOW to HIGH
  - FALLING means to trigger the interrupt when the pin goes from HIGH to LOW

## 16.8 Pinout



# 17 Input and output interfacing

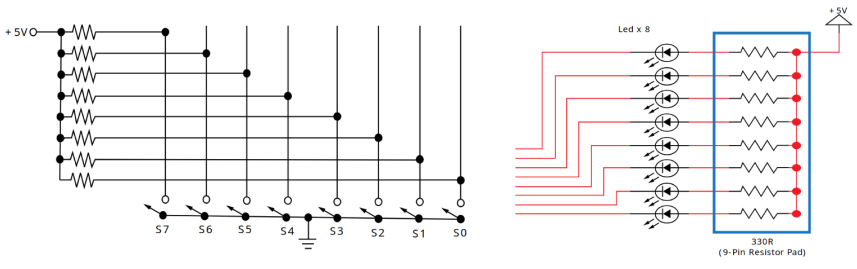



## 17.1 MCU-initiated transfer

### 17.1.1 Unconditional transfer

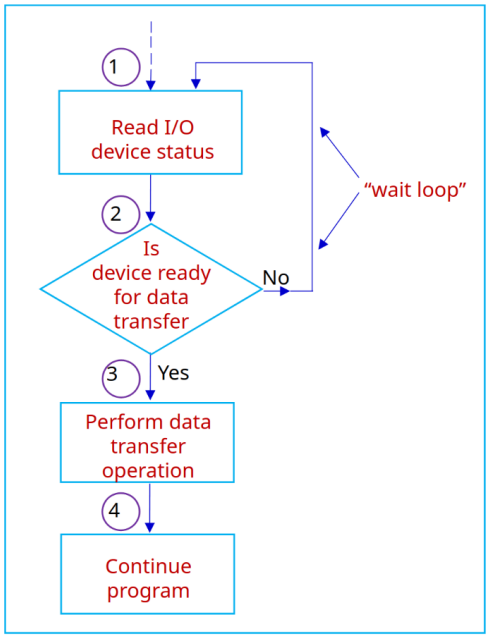
I/O device must always be ready for communication.
Examples:

- To input an 8-bit data word from a set of 8 switches
- Output data to LEDs



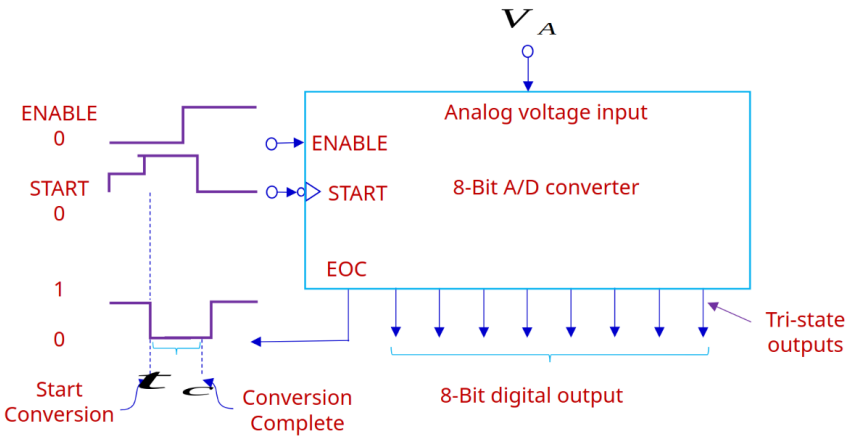

MCU-initiated transfer

### 17.1.2 Conditional transfer




- Communication takes place only when the I/O device is ready. The Arduino performs handshaking to transfer data to the I/O device.
- The MCU must read the status information from the I/O device (1).
- The MCU then tests the status to see if the device is ready for data transfer (2).
- If the device isn't ready for data transfer, remain in the "wait loop" until the device is ready.
- If the device is ready, perform the data transfer (3).
- Handshaking is needed for the data to be transferred.
- The data acquisition subroutine may be accessed at any point in the user's program.
- However, there are some disadvantages to conditional transfer:
  - Needing to wait for I/O devices to be ready.
  - MPU can do other things while waiting, especially when I/O devices are slow.

### 17.1.3 Example of conditional transfer




- The circuit converts analogue voltage input $V_A$ so an 8-bit output ($D_7$ - $D_0$)
- Conversion process is initiated by a pulse to START
- Conversion time, $t_c$ can be up to 100 µs
- End-of-conversion (EOC) is LOW during conversion
- EOC is HIGH when conversion is completed
- When ENABLE is HIGH, make the latched binary output available
- When ENABLE is LOW, the output is at the High-Z state (disconnected)

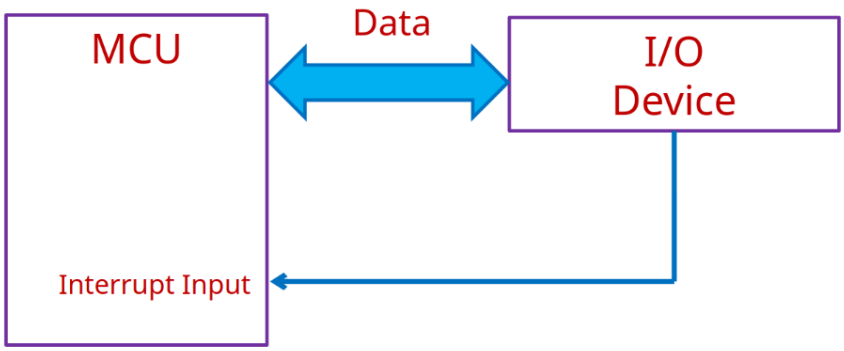
Steps to use the circuit above:

1. MCU issues a START pulse to the ADC to convert $V_A$ to its digital equivalent
2. MCU polls the status of EOC output until conversion is completed
3. MCU reads the ADC output into one of its internal registers

## 17.2 Device-initiated transfer

### 17.2.1 Interrupt transfer

- Handshaking is required for an interrupt transfer.
- The I/O device sends a signal to an interrupt input to inform the MCU it is ready for data transfer.
- Hardware interrupts are triggered by a state (HIGH or LOW) or a change in state (HIGH to LOW or LOW to HIGH).




- When an interrupt occurs, all the important registers content which define the current state of the MCU are immediately stored away in a dedicated memory location, before going to the interrupt service routine (ISR).
- Upon returning from the ISR, the MCU returns to the previous state by restoring the contents of the important registers.

## 17.3 Polling vs interrupting

### 17.3.1 Advantages of polling

- Ease of software implementation

### 17.3.2 Advantages of interrupting

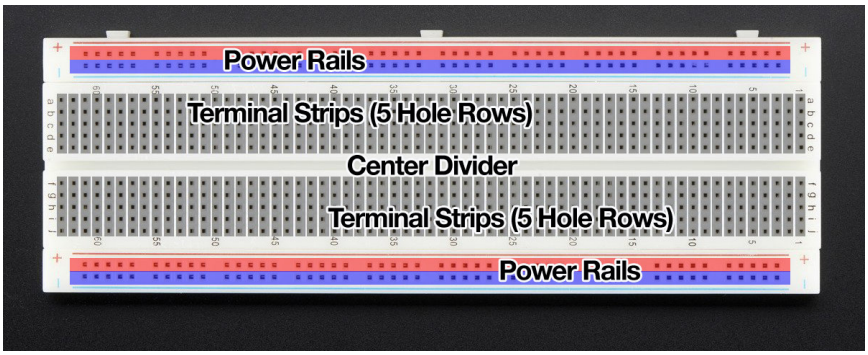
- Multitasking, as the MCU can process other commands while waiting for an I/O device to be ready.
- Acquisition accuracy for fast acquisition tasks. For example, like reading an encoder on a fast rotating motor shaft. These pulses are too short for the polling method to capture, resulting in missing pulses.

## 18 Determining the output of switches

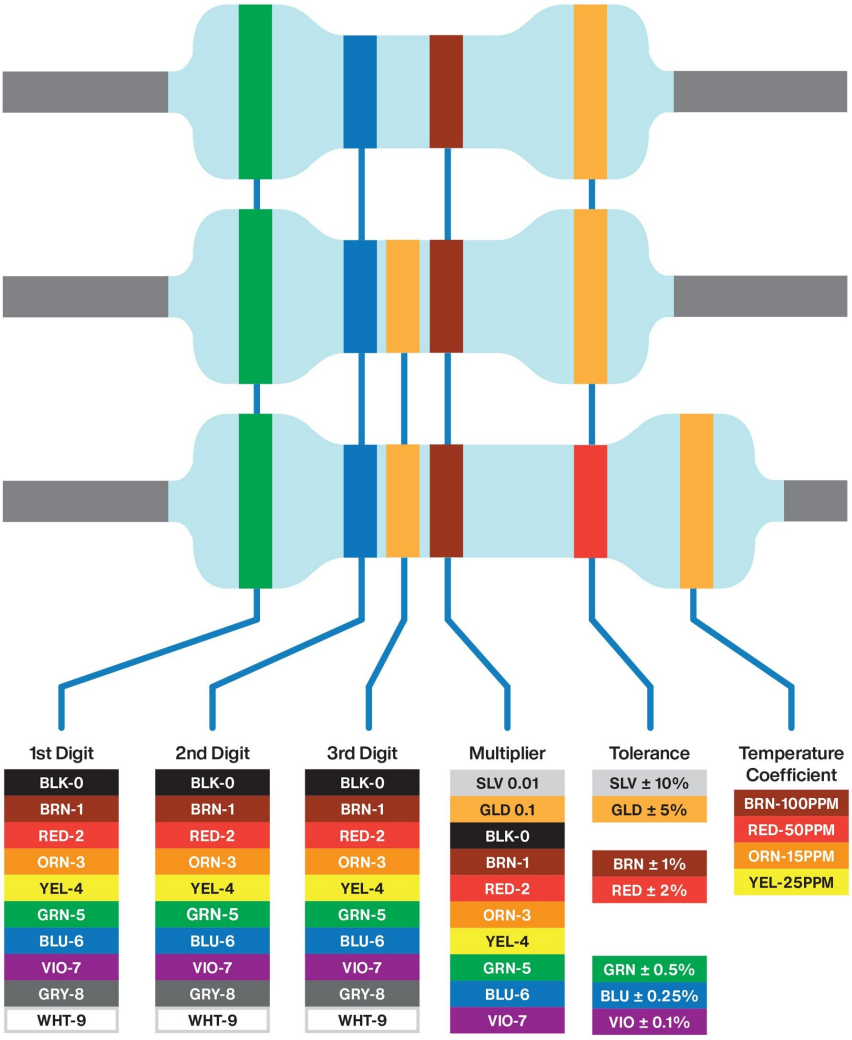
1. Look at what the ends of the switch is connected to.
2. If the switch has one end connected to ground, that means when it is closed, the electric potential will decrease to zero, which means the switch will output a "LOW".
3. If the switch has one end connected to a power output, like +5V or something similar, that means when it is closed, the electric potential will increase to the potential of the power output, which means the switch will output a "HIGH".
4. The voltage is considered high when it is above 3V, low if it is under 0.8V, and indeterminate if it is between 0.8V to 3V.

## 19 Breadboard wiring

Each line in the image below represents a line of slots that are connected.



# 20 Resistor colour code chart



# 21 Calculating voltage resolution

$$\text{Voltage resolution} = \frac{V_{max} - V_{min}}{2^n - 1}$$

Where:

- Voltage resolution is the change in voltage for a single step
- $V_{max}$ is the maximum voltage value
- $V_{min}$ is the minimum voltage value
- $n$ is the maximum resolution of the Arduino in bits. For the Arduino Uno, this value is 10 as the maximum resolution of the Arduino Uno is 10 bits.

## 21.1 Example

For a reference high voltage of 5V and a reference low voltage of 0V and using an Arduino Uno:

$$\text{Voltage resolution} = \frac{5 - 0}{2^{10} - 1} = 0.004\,882\,812\,5\,\text{V}$$

## 22 Parsing multiple bytes into a single integer

- Read the data from the first byte and the second byte.
- If the first byte is the lower byte, just leave it be.
- Otherwise, the first byte is the higher byte, so bitwise shift the first byte left by 8 bits, i.e. `first_byte << 8`, as a byte is 8 bits long.
- If the second byte is the higher byte, bitwise shift the second byte left by 8 bits, i.e. `second_byte << 8`, as a byte is 8 bits long.
- Otherwise, the second byte is the lower byte, so just leave it be.
- Do a bitwise OR operation on the first byte and second byte to get back the final 16-bit integer.
- The process is the same for parsing from larger amounts of data, like 2 bytes, just that the bit shift needs to be 16 bits as 2 bytes is 16 bits. The end result of this parsing would be a 32-bit integer.
- The process is also the same for parsing into larger integer representations, like 32-bit integers, just that the most significant byte will have to be shifted by 32 bits to the left, the second most significant byte will be shifted by 24 bits to the left, and so on, subtracting 8 bits from the number of bits to shift to the left every time.