Product development & engineering design

ME-320

PROF. JOSIE HUGHES

Lecture 6: (i) Control & Sensing (ii) Fabrication

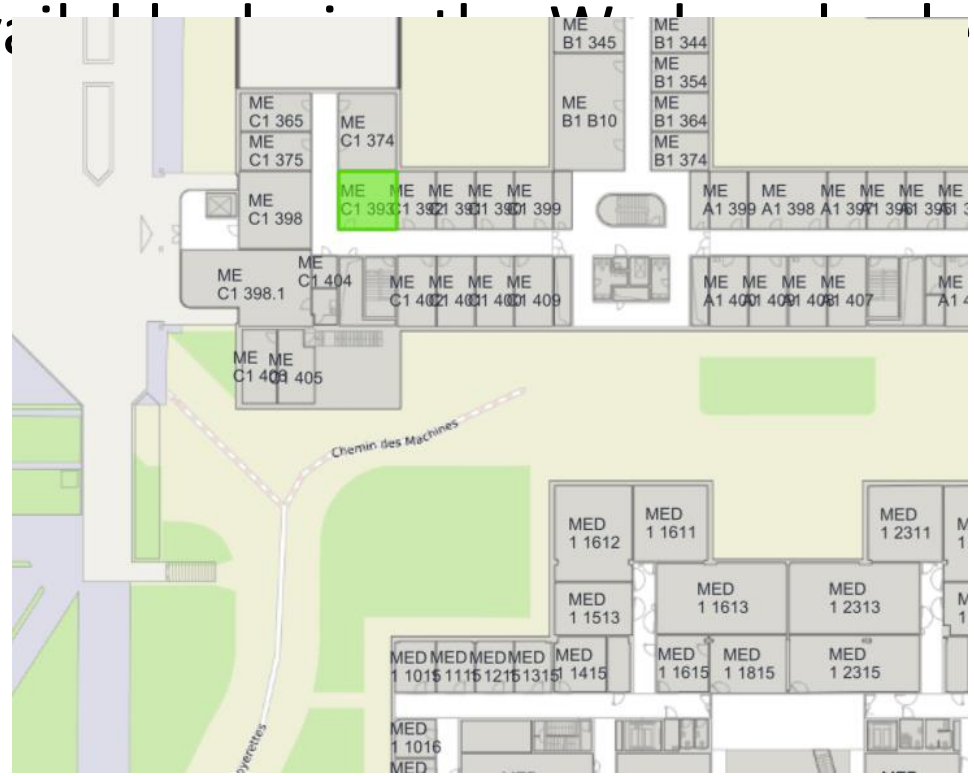
Drop in Sessions

Friday: 12-13

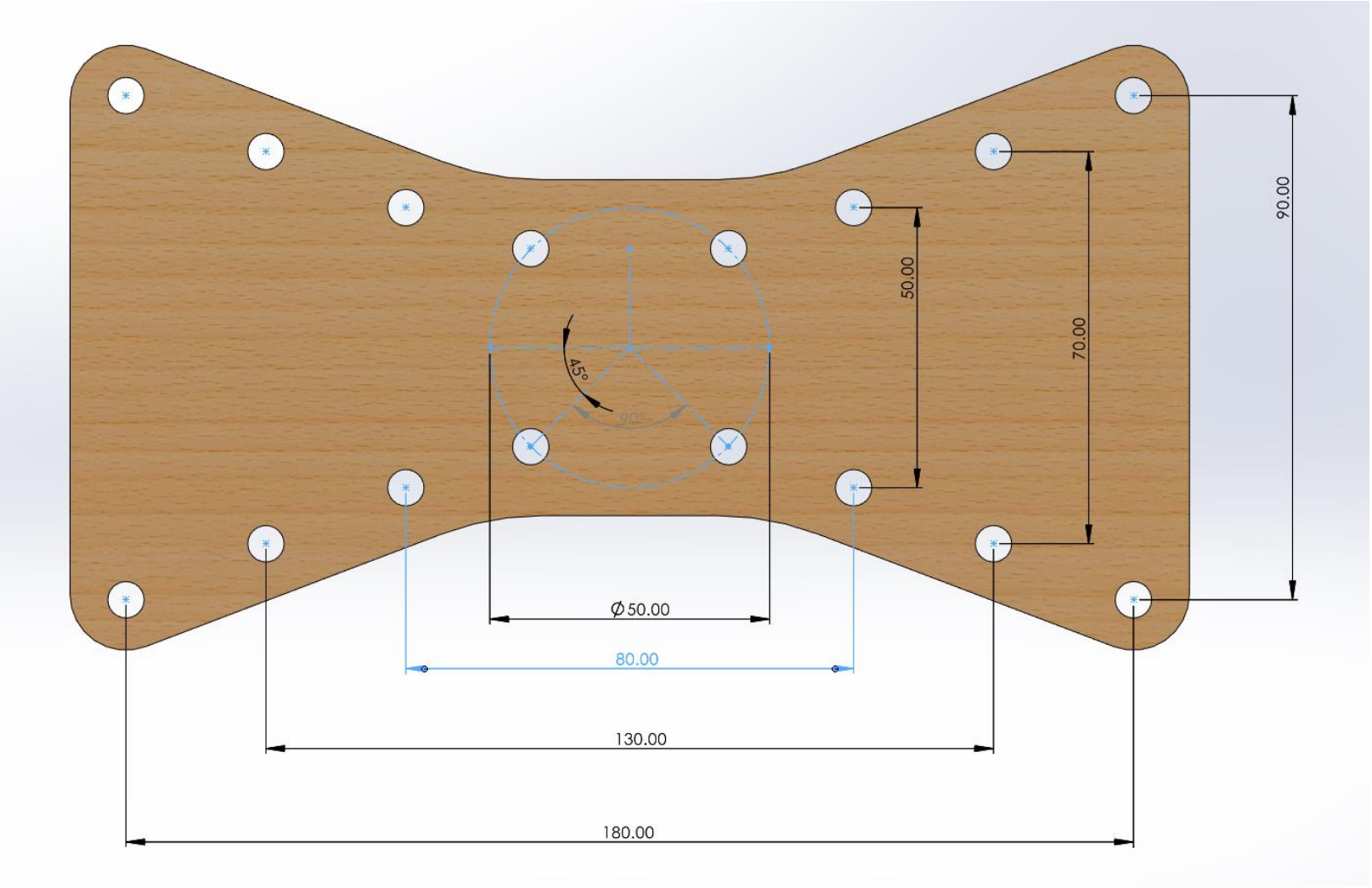
Monday 16-17

- Come to ask for advice/debugging/technical problems/advice
- Pick-up/exchange parts (this is ava[illegible]ctures and this Friday session)

ME C1 393



Attachment/mounting point



Raspberries

- Available for testing any time (9-17h) in the drop-in room area
- Conductive raspberry ~k Ohms
- Will be moved to Spot in ~2 weeks (when there is space)
- Robot testing will start in December

Next week: Design Review

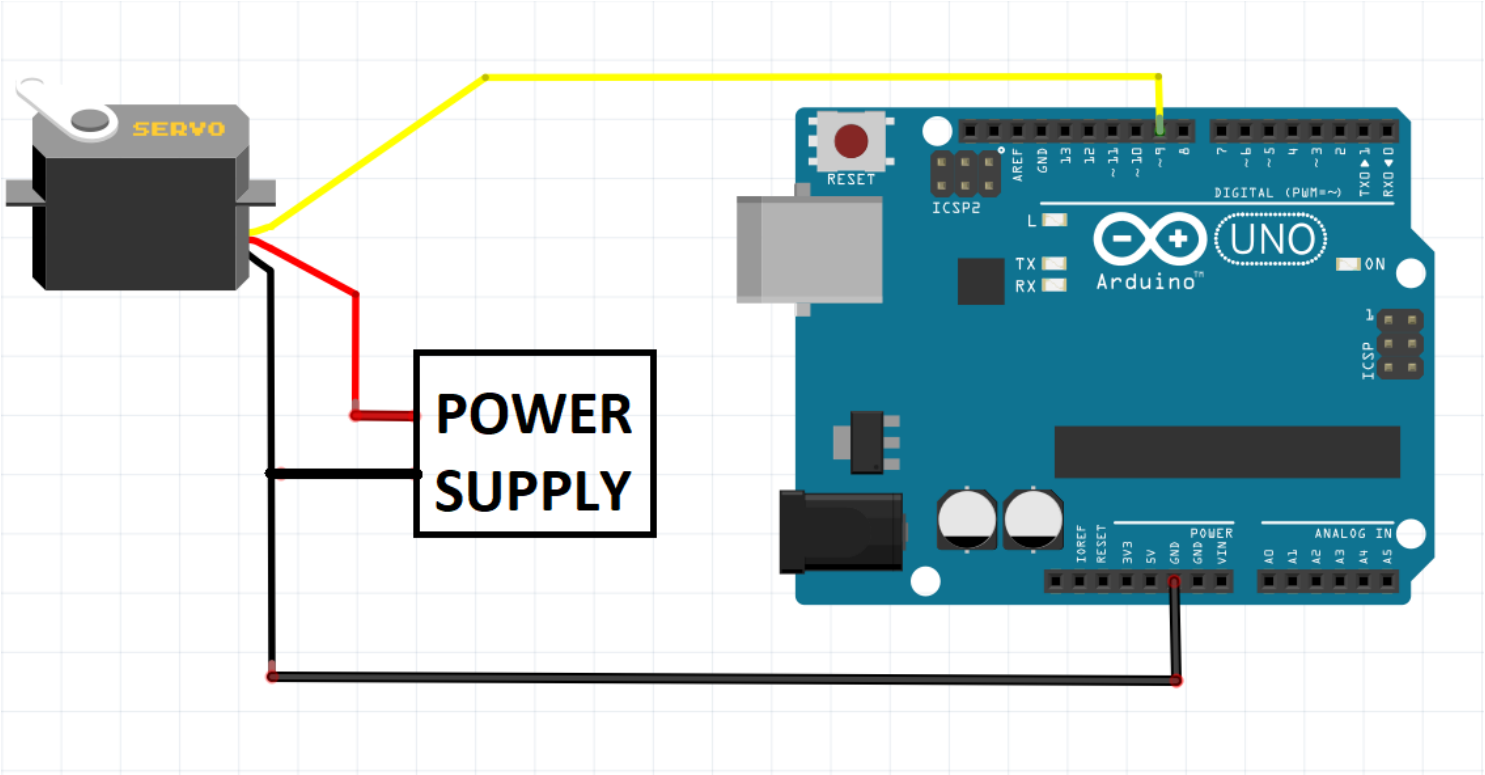
Come prepared to discuss your design, ask any questions, and get help!

Time	Table 1	Table 2	Table 3	Table 4	Table 5	Table 6
8:20	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
8:40	Group 7	Group 8	Group 9	Group 10	Group 11	Group 12
9:00	Group 13	Group 14	Group 15	Group 16	Group 17	Group 18
9:20	Group 19	Group 20	Group 21	Group 22	Group 23	Group 24
9:40	Group 25	Group 26	Group 27	Group 28	Group 29	Group 30
10:00	Group 31	Group 32	Group 33	Group 34	Group 35	Group 36
10:20	Group 37	Group 38	Group 39	Group 40	Group 41	Group 42
10:40	Group 43					

Actuator Types

Controlling Servos

How can we more than the 40mA the Arduino can supply!?



- Need a common ground to preventing floating of voltages, and so they are at a common potential
- Make sure the voltage does not exceed that allowed by the servo

Power supply – white line indicates +ve supply

Controlling Servos

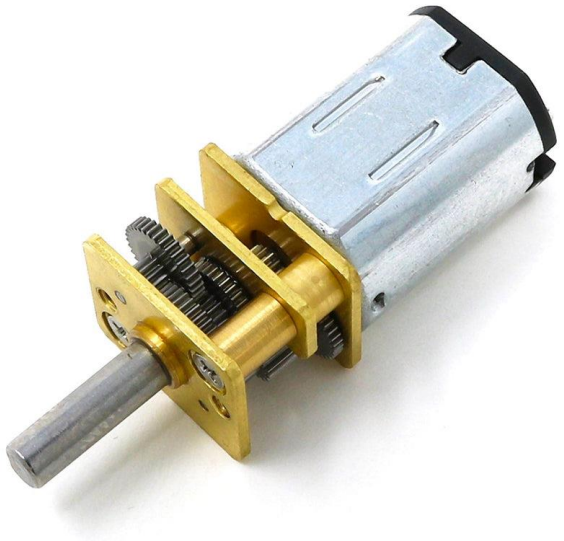
- Can use position control with the servo, and control the rate at which you vary the position
- If the servo can not apply enough torque to get to a given position, it will continue to ‘hunt’ and you can damage the servo
- There is no ‘feedback’ mechanism.

Physically Interfacing with Servos

Mounting points

Servo ‘horns’ – can screw on in variable position

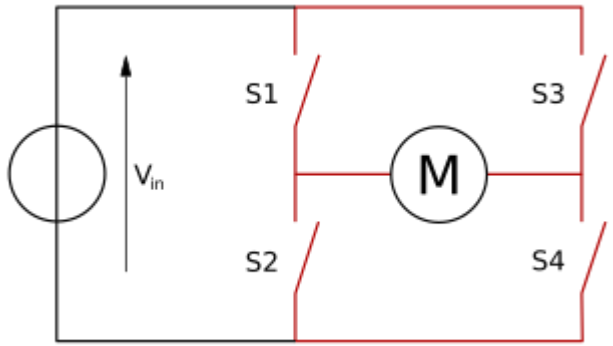
Controlling Motors

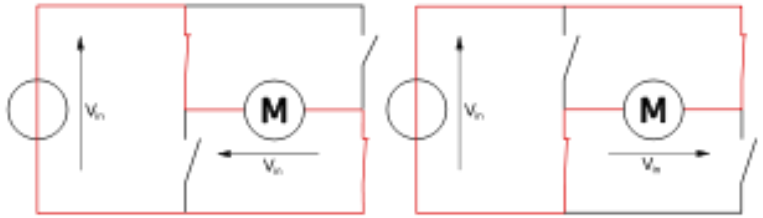


- Speed is proportional to the voltage
- The polarity determine the direction

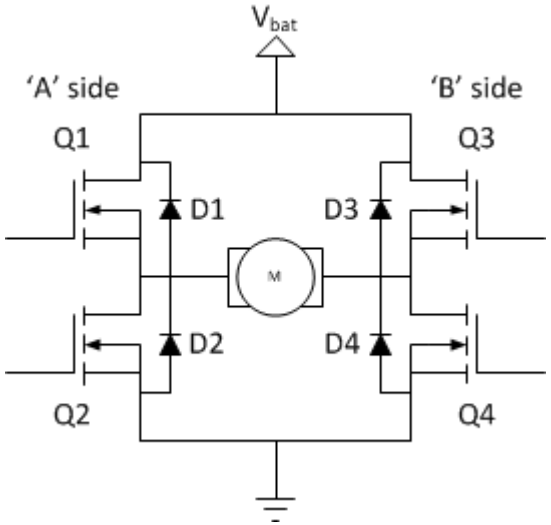
→ Need to be able to vary the polarity and the voltage from the micro-controller

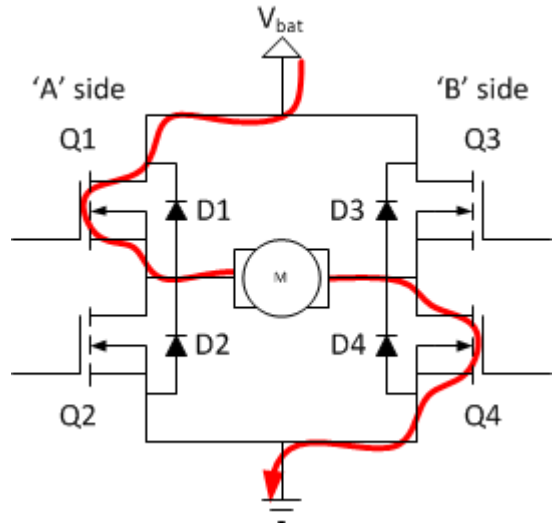
Controlling Motors: H-Bridges

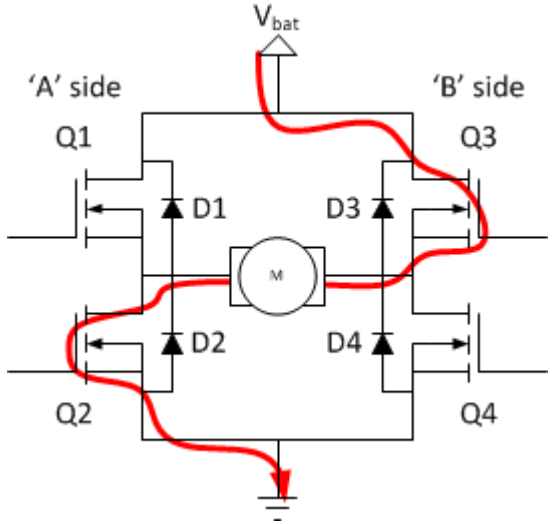




Controlling Motors: H-Bridges



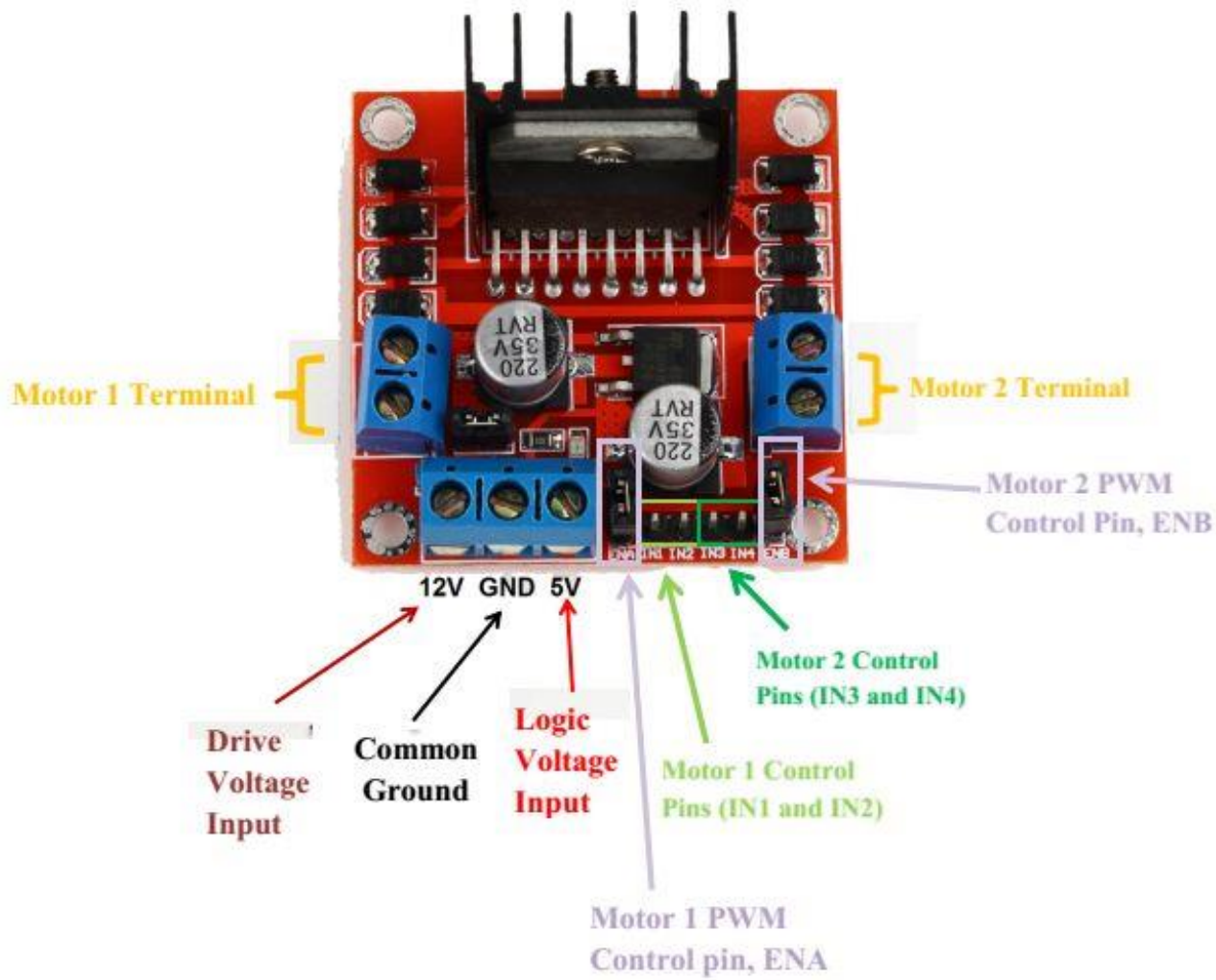




- The switching elements (Q1..Q4) are usually bi-polar or FET transistors, in some high-voltage applications IGBTs.
- Integrated solutions also exist but whether the switching elements are integrated with their control circuits or not is not relevant for the most part for this discussion.
- The diodes (D1..D4) are called catch diodes and are usually of a Schottky type.

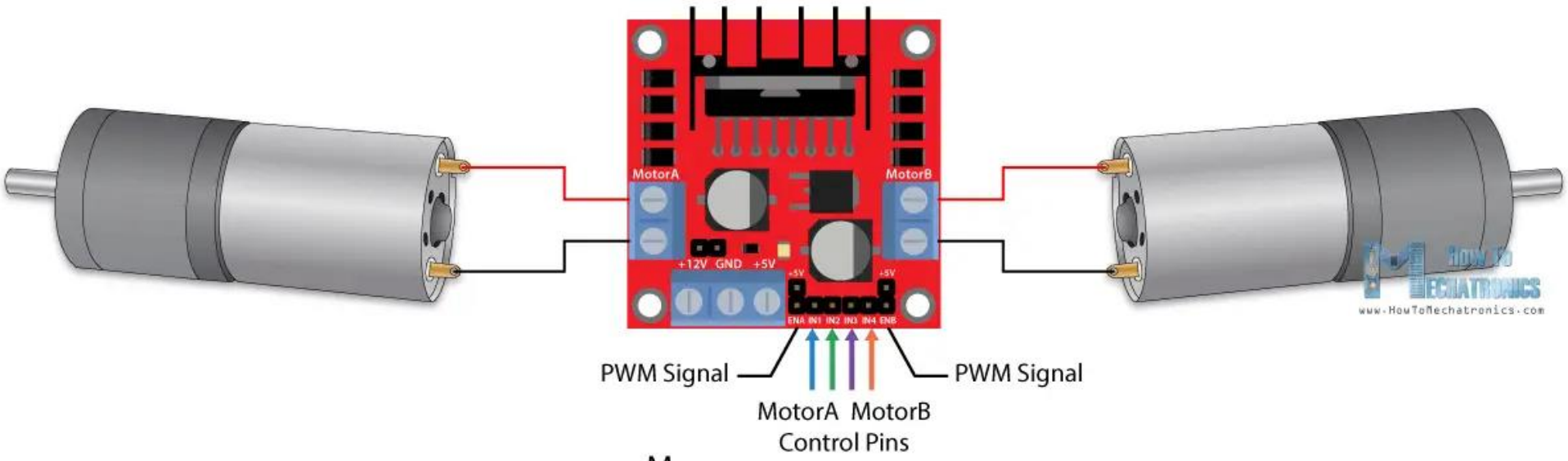
Controlling Motors: H-Bridges

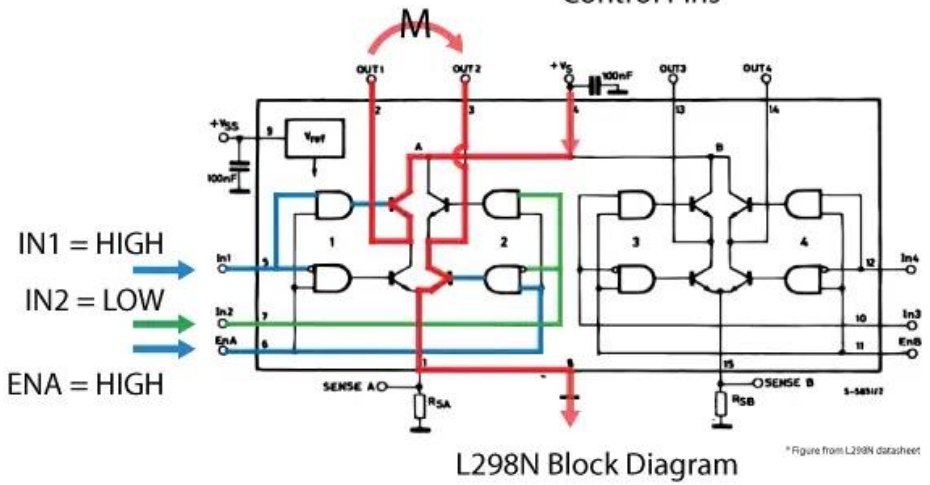
L298N



Remember to check out the data-sheet!

Controlling Motors: H-Bridges

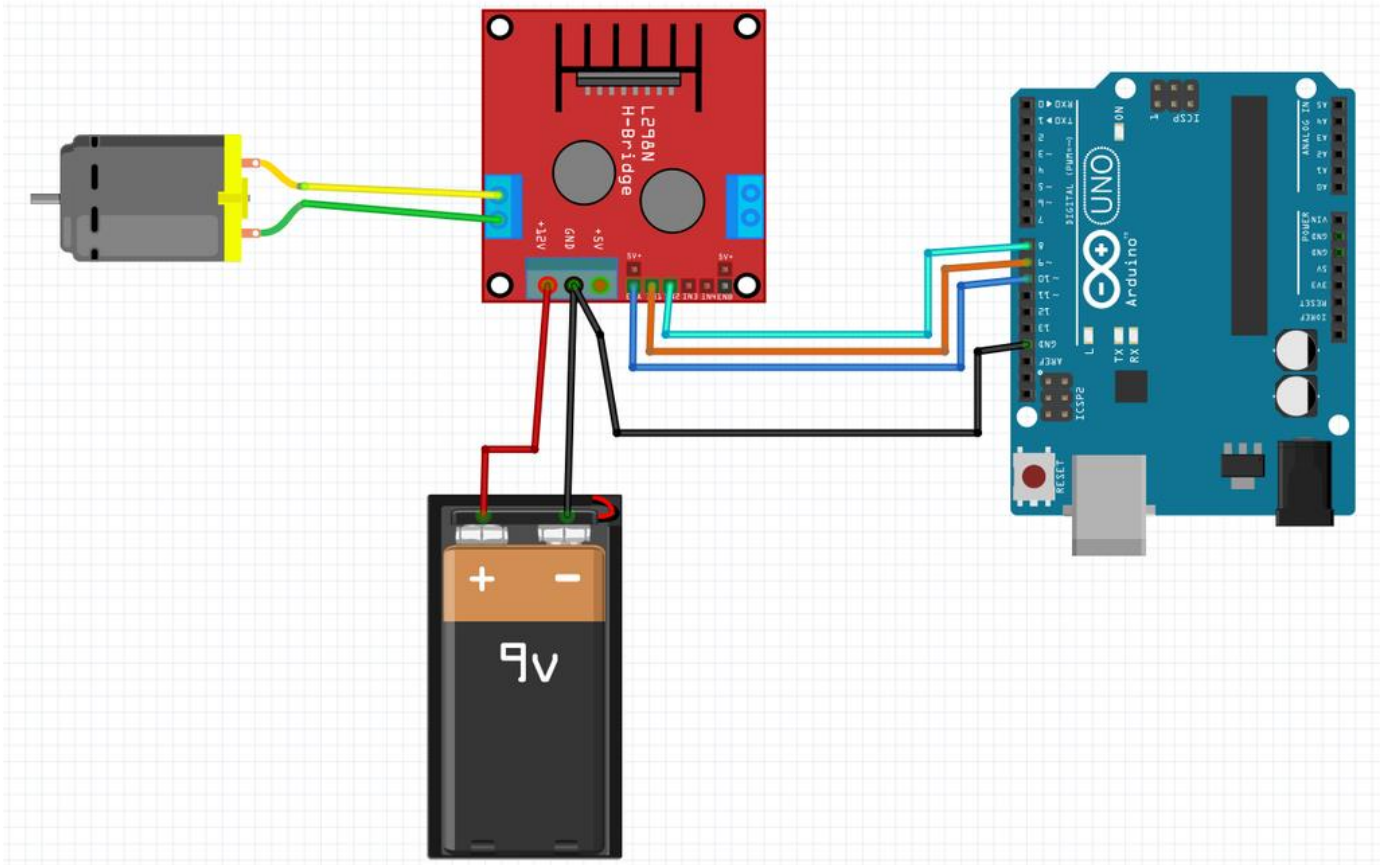


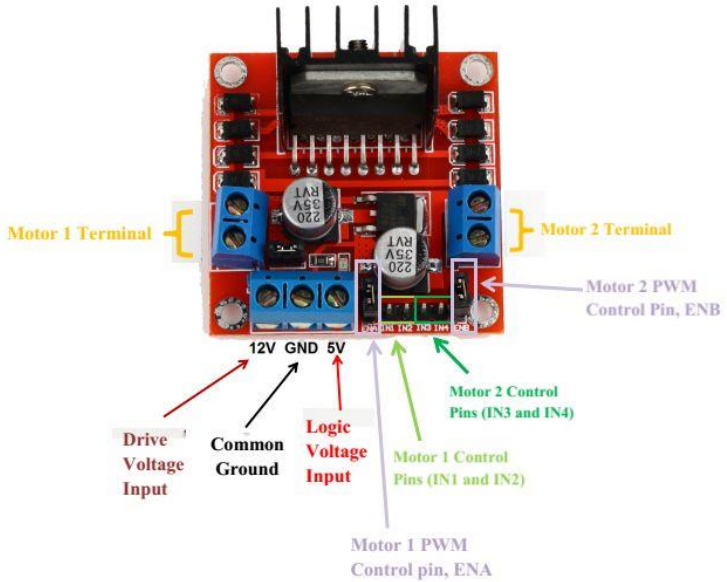


L298N Block Diagram

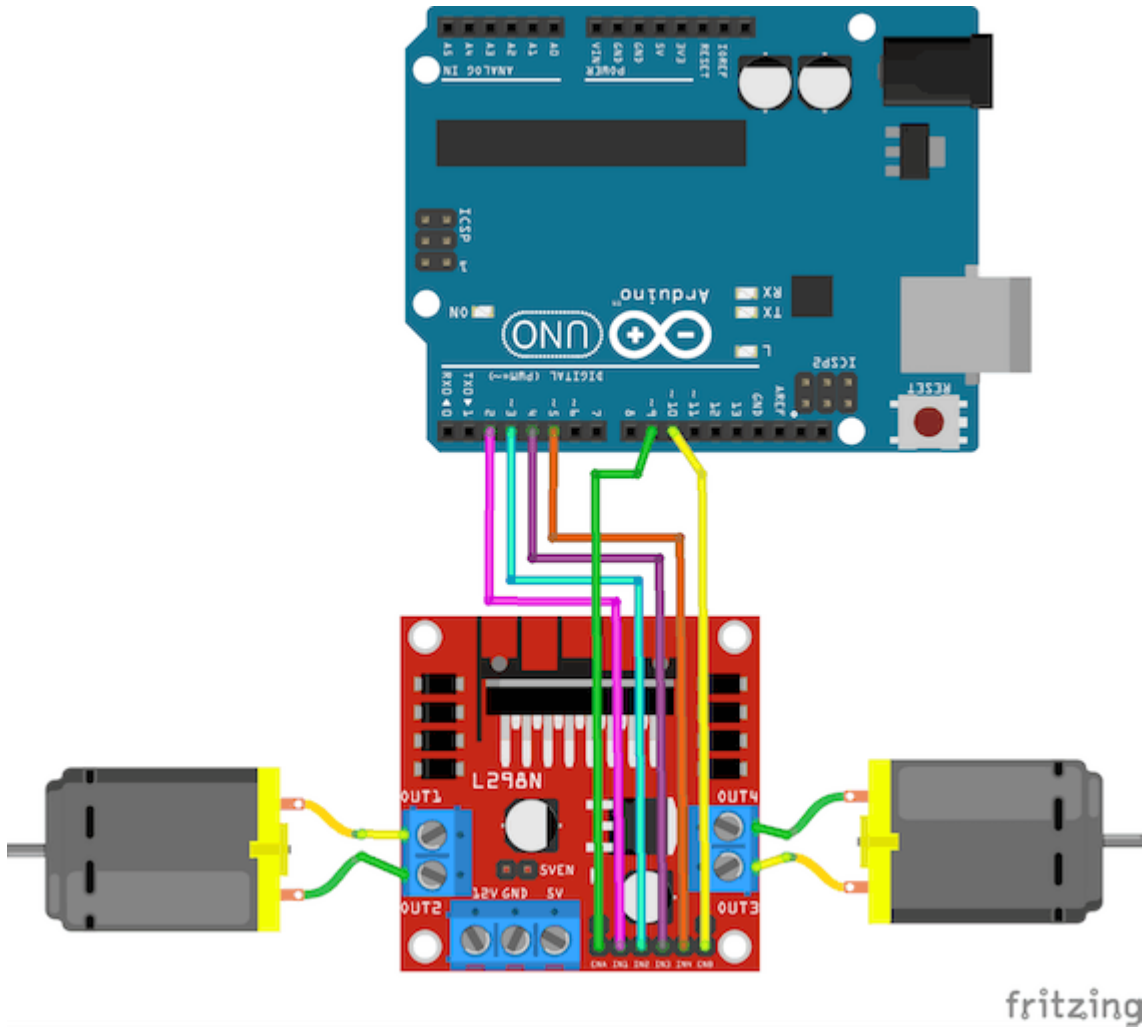
* Figure from L298N datasheet

Controlling Motors: H-Bridges





- Speed control via analog output from microcontroller (PWM). ENA controls the speed of the left motor and ENB controls the speed of the right motor.
- Setting IN1 to HIGH and IN2 to LOW will cause the left motor to turn a direction.
- Setting IN1 to LOW and IN2 to HIGH will cause the left motor to spin the other direction.



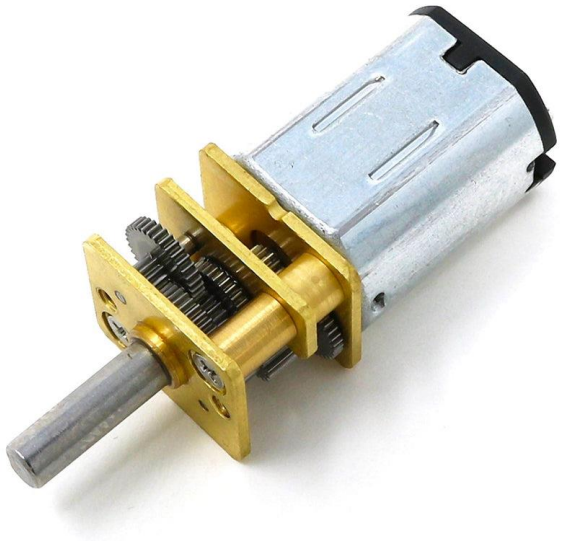

```ino
int motor1pin1 = 2;
int motor1pin2 = 3;

int motor2pin1 = 4;
int motor2pin2 = 5;

void setup() {
  // put your setup code here, to run once:
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(motor2pin1, OUTPUT);
  pinMode(motor2pin2, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);

  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(1000);

  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);

  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
  delay(1000);
}
```

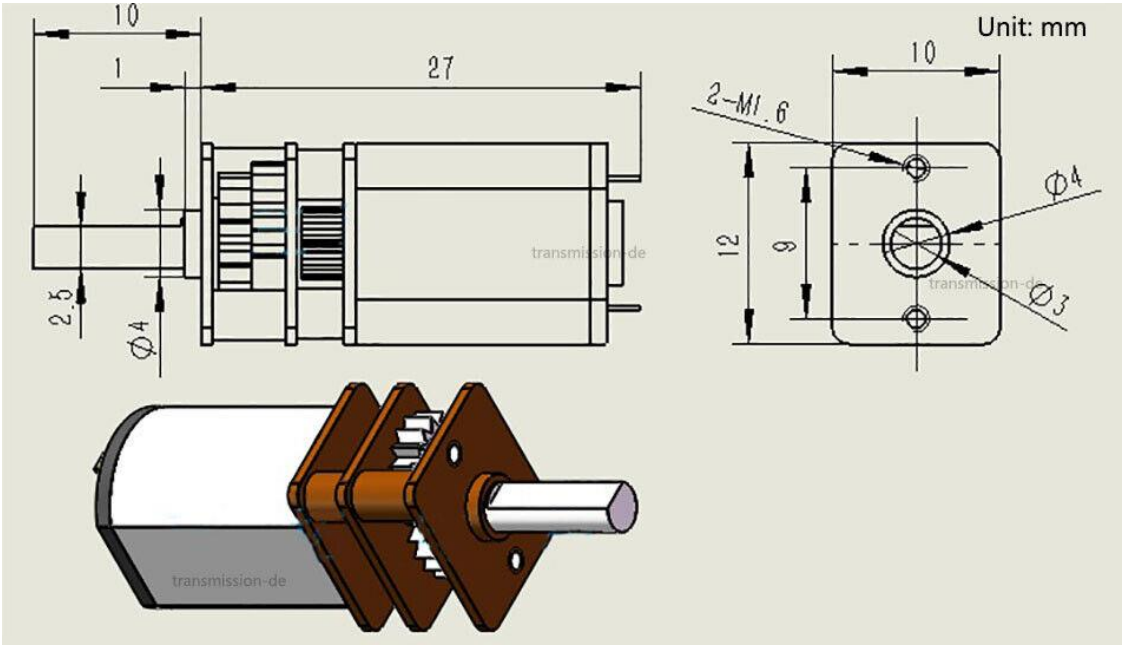
```
int motor1pin1 = 2;
int motor1pin2 = 3;

int motor2pin1 = 4;
int motor2pin2 = 5;

void setup() {
  // put your setup code here, to run once:
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(motor2pin1, OUTPUT);
  pinMode(motor2pin2, OUTPUT);

  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
}

void loop() {
```

Adding in speed control…

```
  //Controlling speed (0 = off and 255 = max speed):
  analogWrite(9, 100); //ENA pin
  analogWrite(10, 200); //ENB pin

  //Controlling spin direction of motors:
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);

  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(1000);

  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);

  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
  delay(1000);
}
```


Physically Connecting to Motors





- Use the flat on the shaft
- Create a brace/bracket to hold the motor

Try using the different actuators

Micro servo:

- Move to different position
- For loop to vary the speed

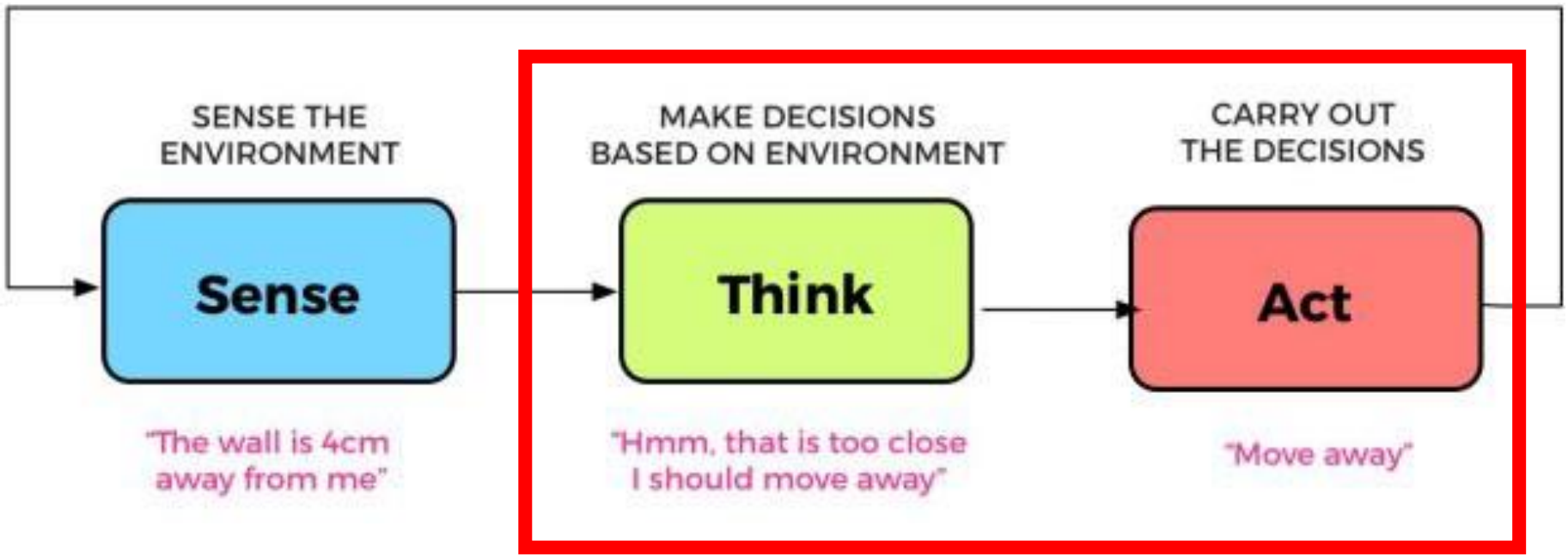
Larger servo (external power)

- Position control + speed control

DC motor

- Speed + direction control

Engineering Design



You are going to need some amount of control!

Servo Control

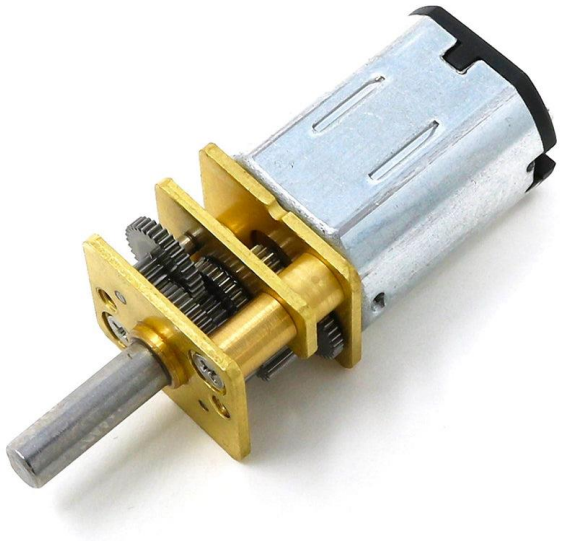
Inherently has ‘position’ control, so what can we control:

- **Absolute position**. Set a fixed position which you move between.
- **Move between positions.** Vary the speed/rate of increment of position.

How could we vary the position for different objects/tasks:

1. Combine with sensing. Detect and then choose the ‘correct’ position control
2. Run the same control (maximum change in position), and exploit the compliance in the system, and the current limit in the servo.

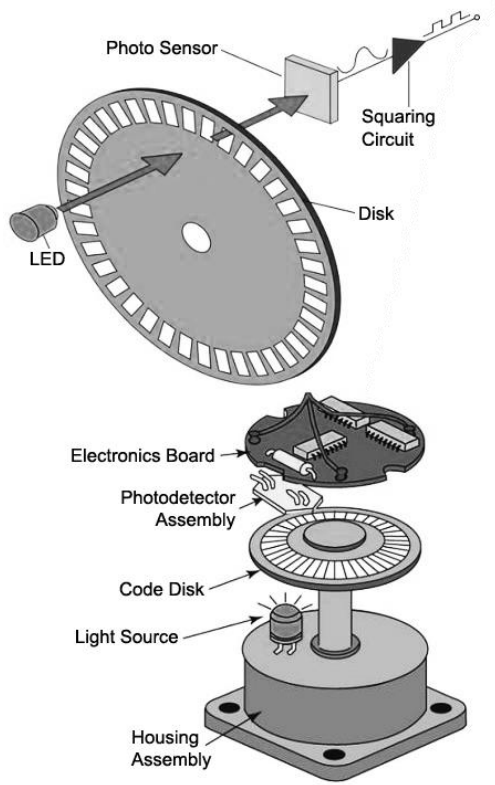
Motor Control



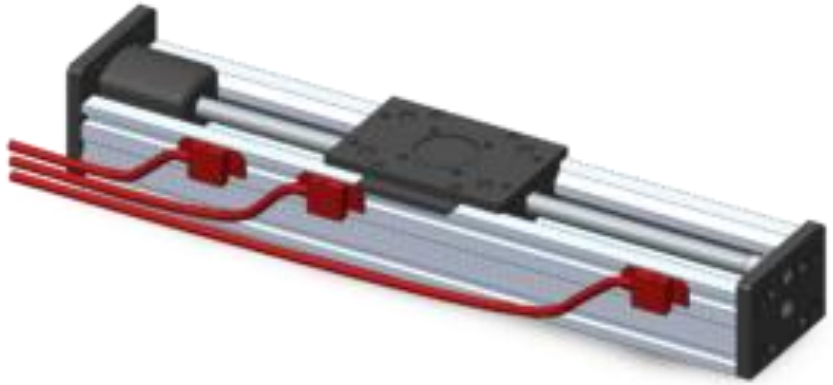
Has speed control and also you can set the current limit (essential with the speed)…

Approach #1: Add some position control

1. Make encoders



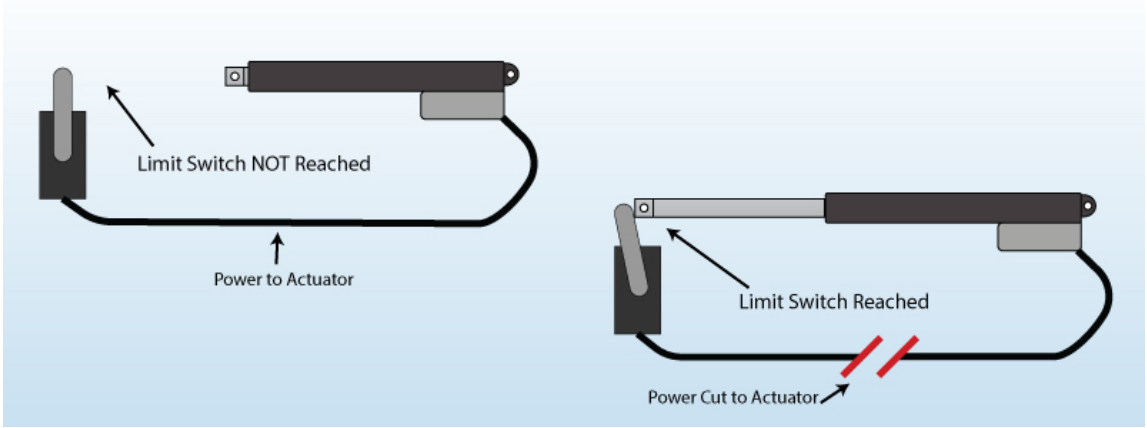
Motor Control



Has speed control and also you can set the current limit (essential with the speed)…

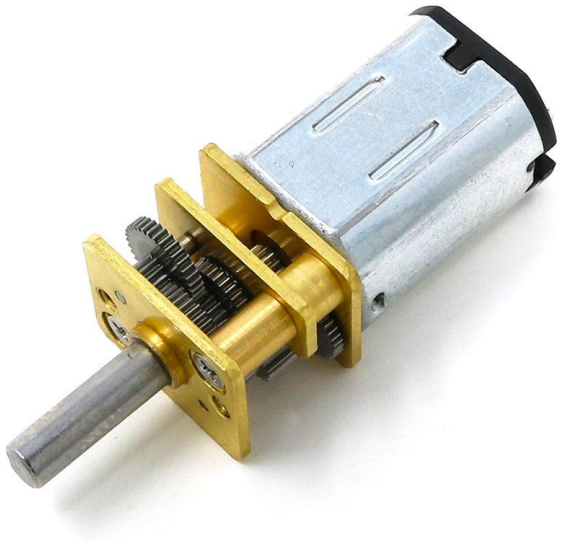
Approach #1: Add some position control, ‘homing’

2. Use limit switches or buttons



Not good if you have lots of different states/positions

Motor Control



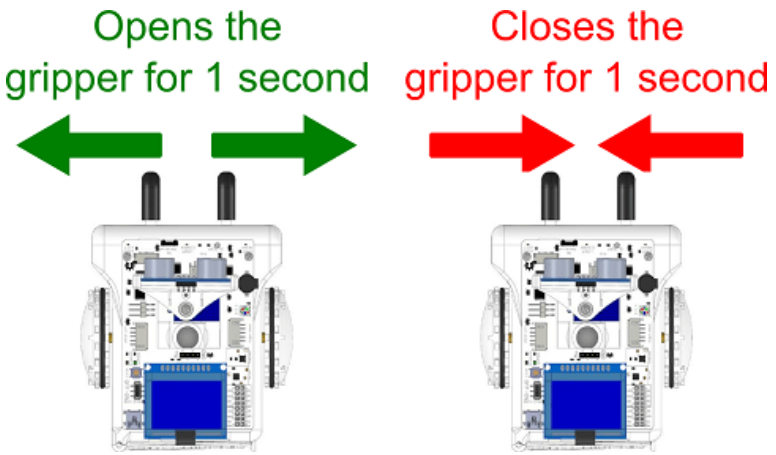
Has speed control and also you can set the current limit (essential with the speed)…

Approach #1: Add some position control

3. Create other ingenious means of detecting position!

- Light dependent resistors
- conductivity (metallic contact)
- distance sensing

Motor Control



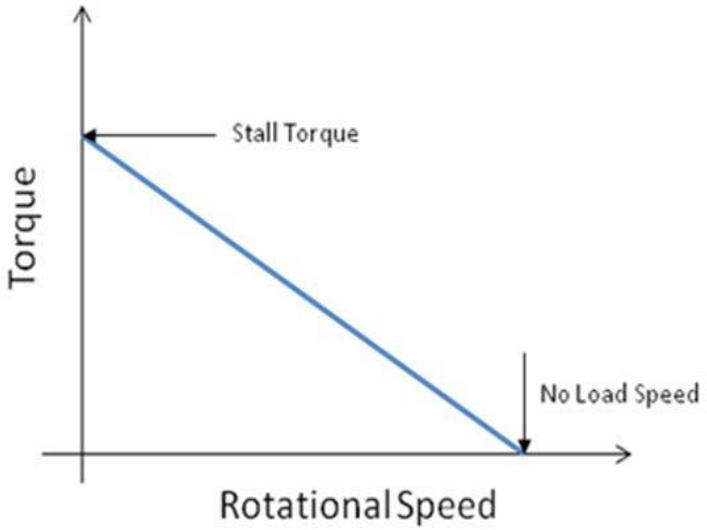
Has speed control and also you can set the current limit (essential with the speed)…

Approach #2: Go purely open loop

Rely on timing only:
e.g. Forwards, 0.5 speed for 2 seconds
Stop
Backwards, 0.5 speed for 2 seconds

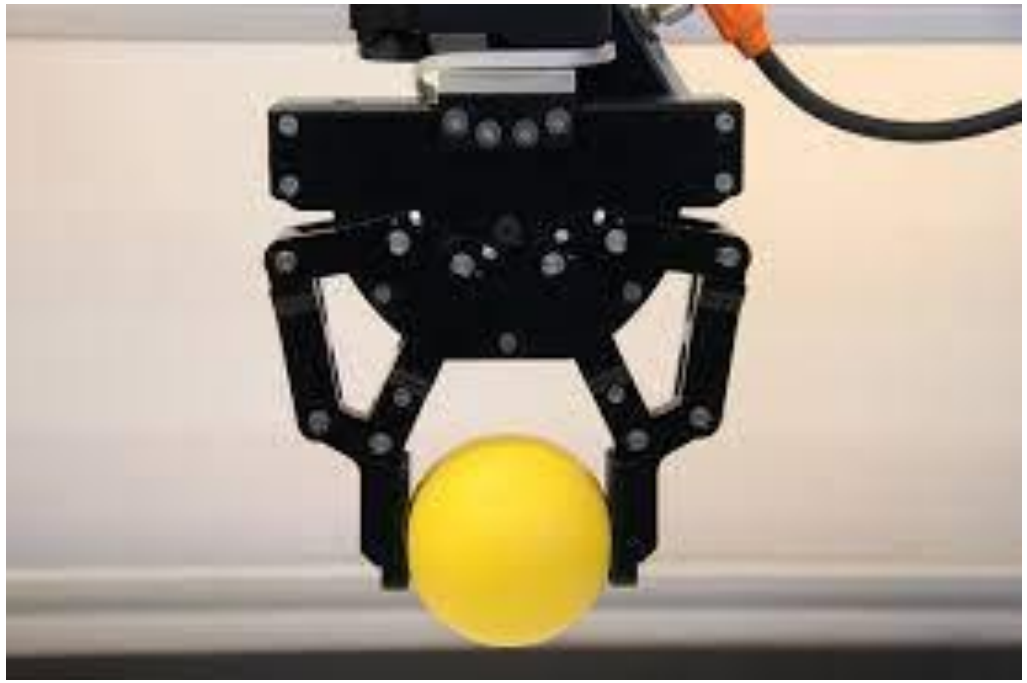
Why could this be a problem?

Motor Control



Has speed control and also you can set the current limit (essential with the speed)…

Approach #3: Open loop, but with ‘physical’ reference points or ‘hard stops’



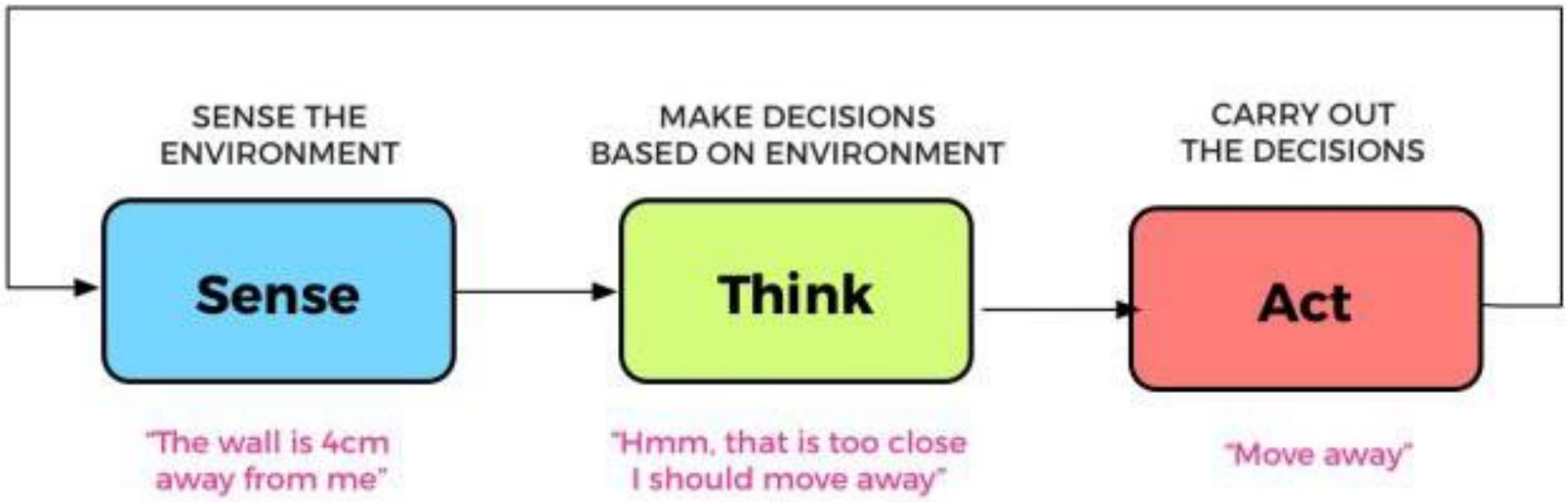
Why could this be a problem?

Control Strategies

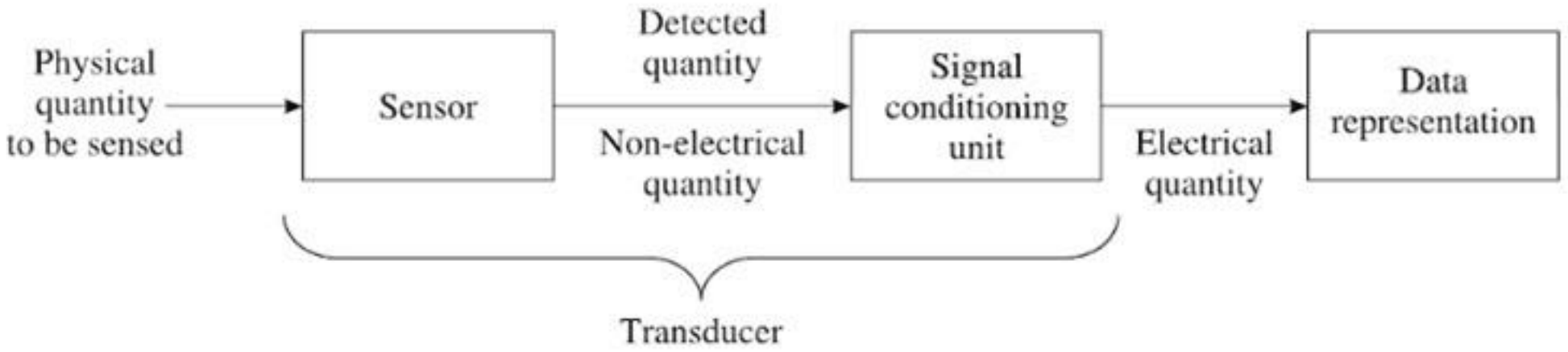
- Actuator changes which control strategy you deploy
- Various trade-offs in terms of the control strategy
- Depends on your mechanism and how you integrated the actuator

https://ttpoll.eu/p/datadriven

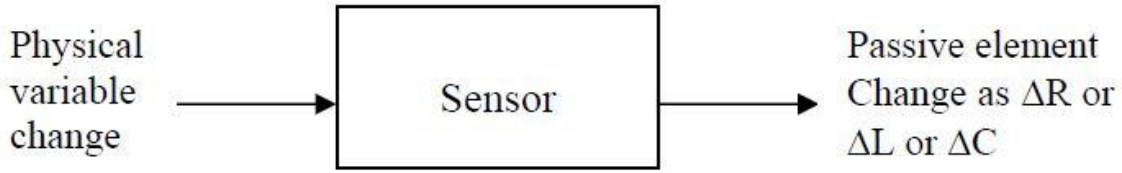
Engineering Design

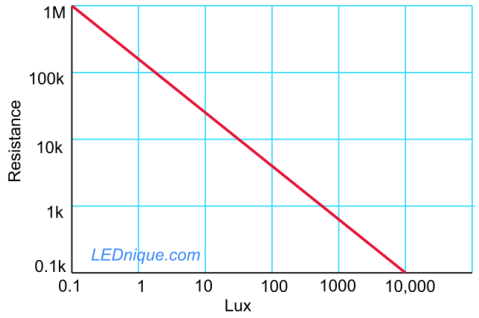


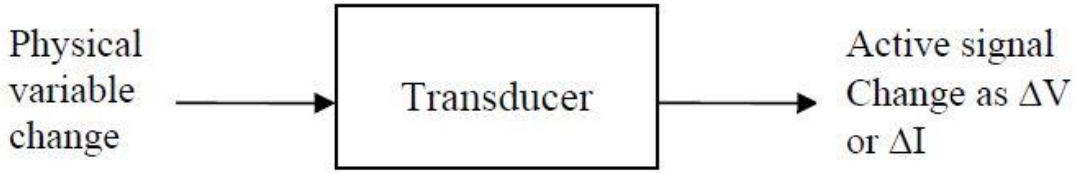
What is a sensor? What is a transducer?

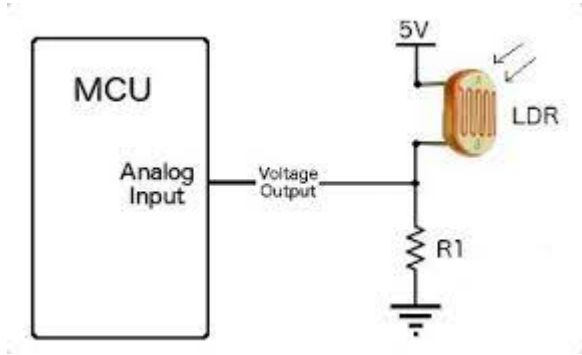


What is a sensor? What is a transducer?

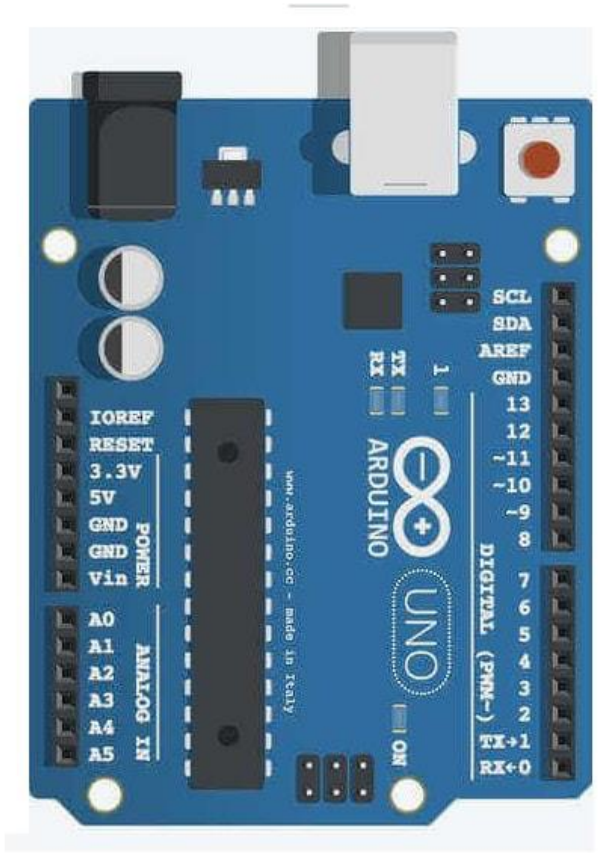






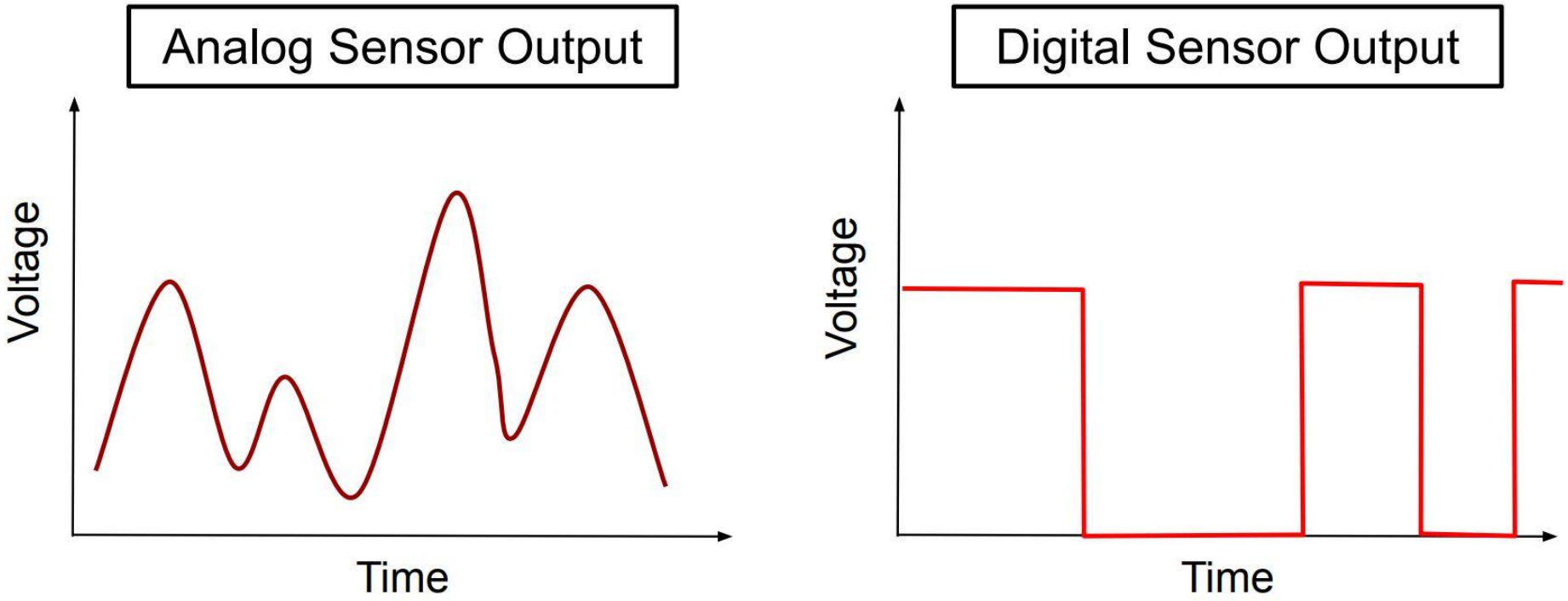


Sensing with a microcontroller



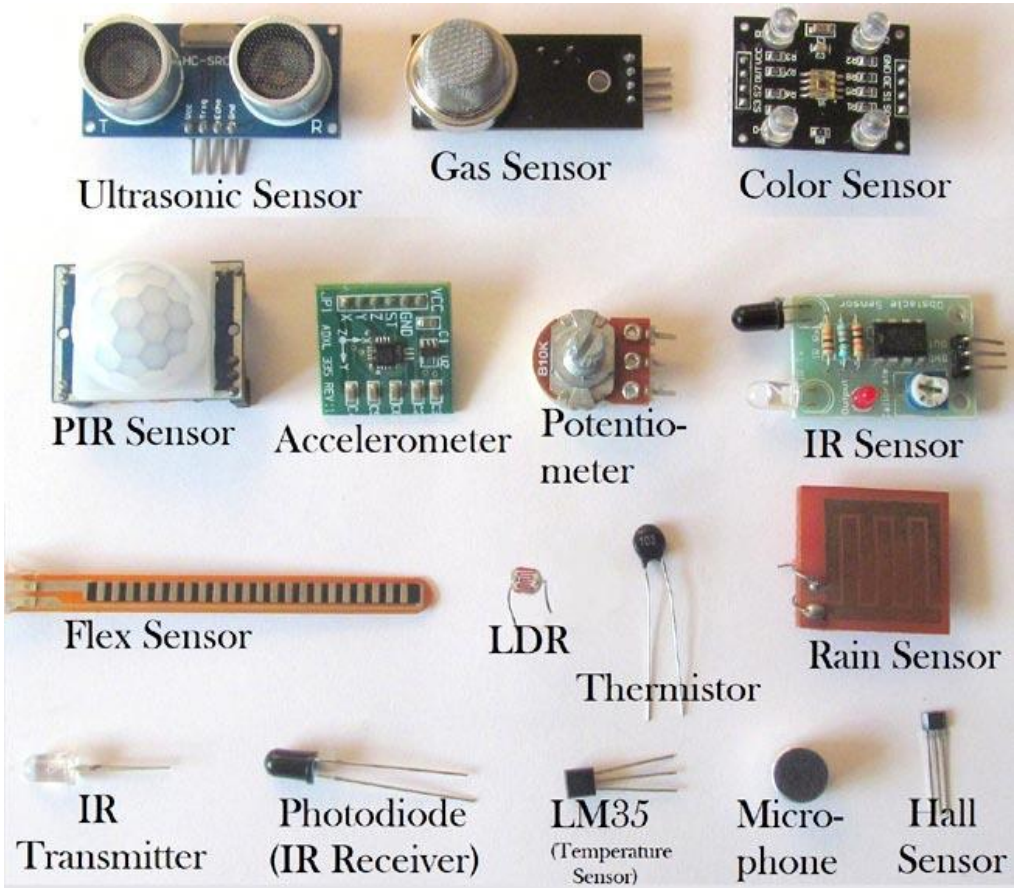
- Many different sensors
- Different input types
- Different transduction circuits

Analog vs. Digital



Analog Inputs

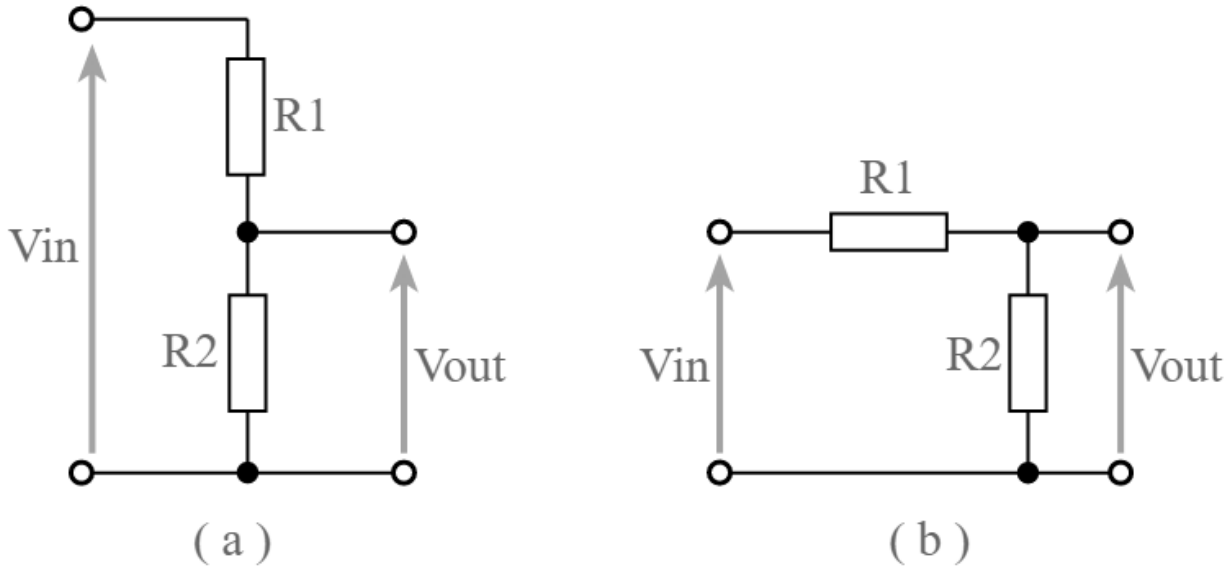
What types of sensors are analog?



- Many sensors can provide ADC outputs
- Universal method of sensing
- Can require additional amplification or transduction

Analog Inputs

Resistive sensors



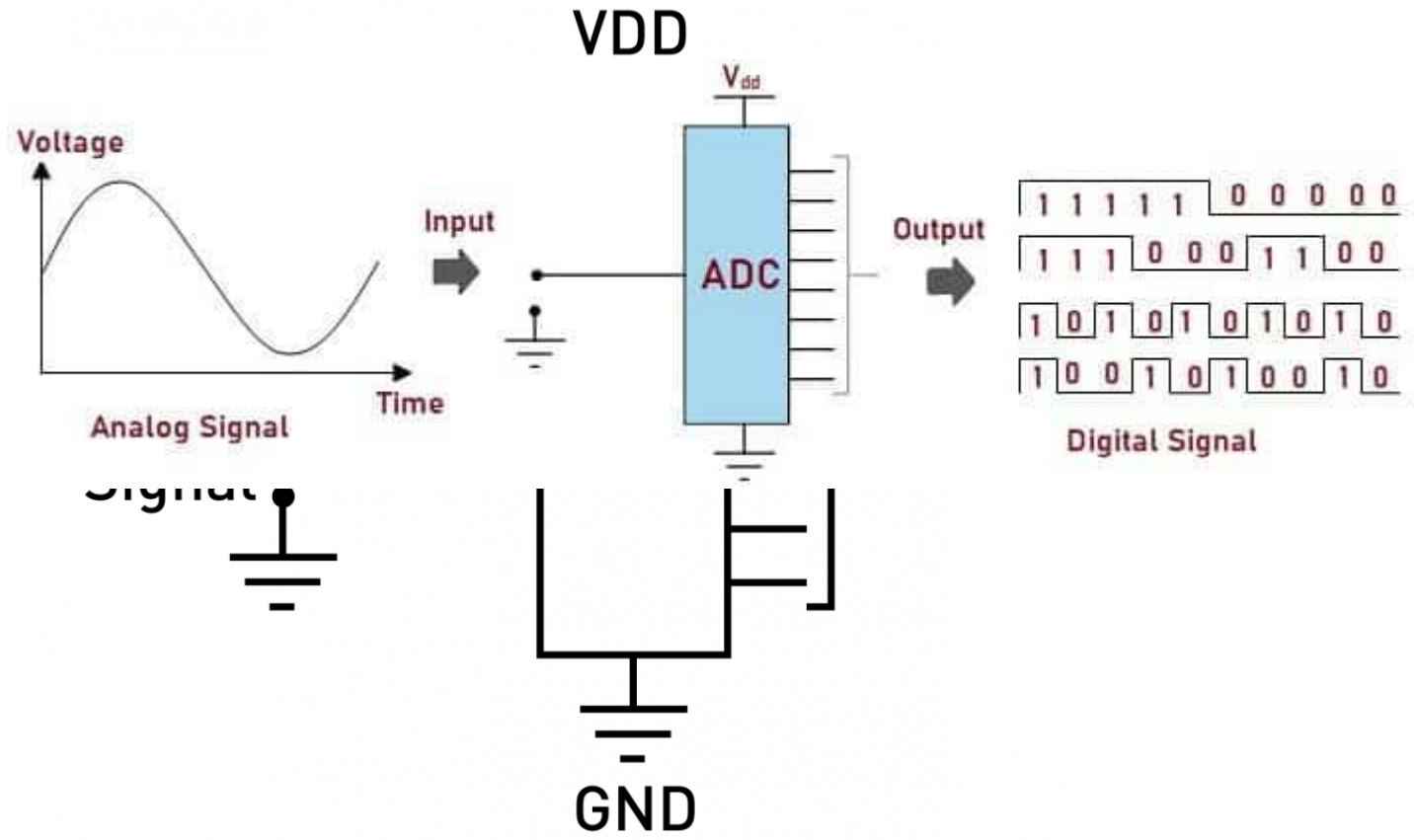
$$V_{out} = V_{in}\left(\frac{R_t}{R_1 + R_t}\right)$$

But how can binary machines (e.g. micro-controllers) deal with these analog signals?

Analog Inputs

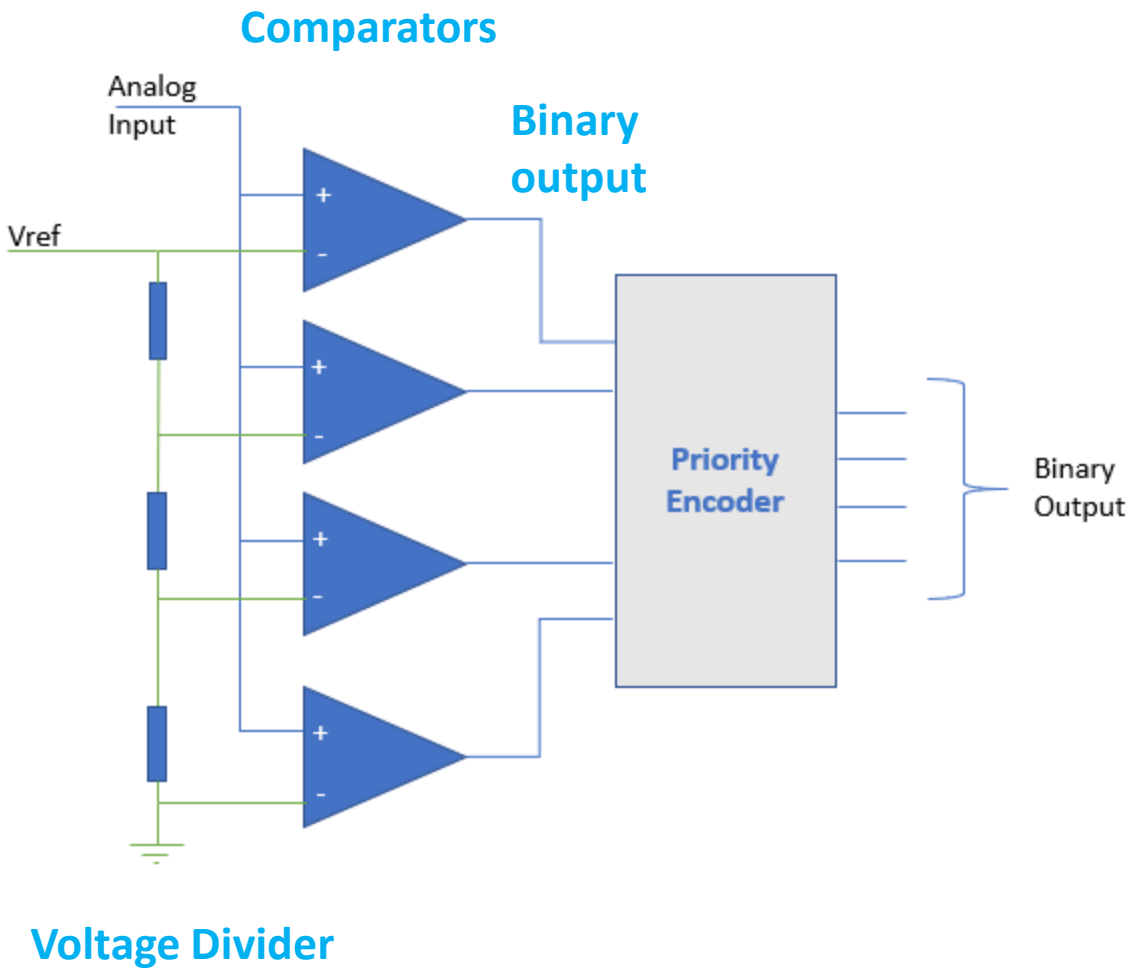
How do we read analog sensors?

Machines work in binary (1s and 0s) - how do we obtain a binary output?



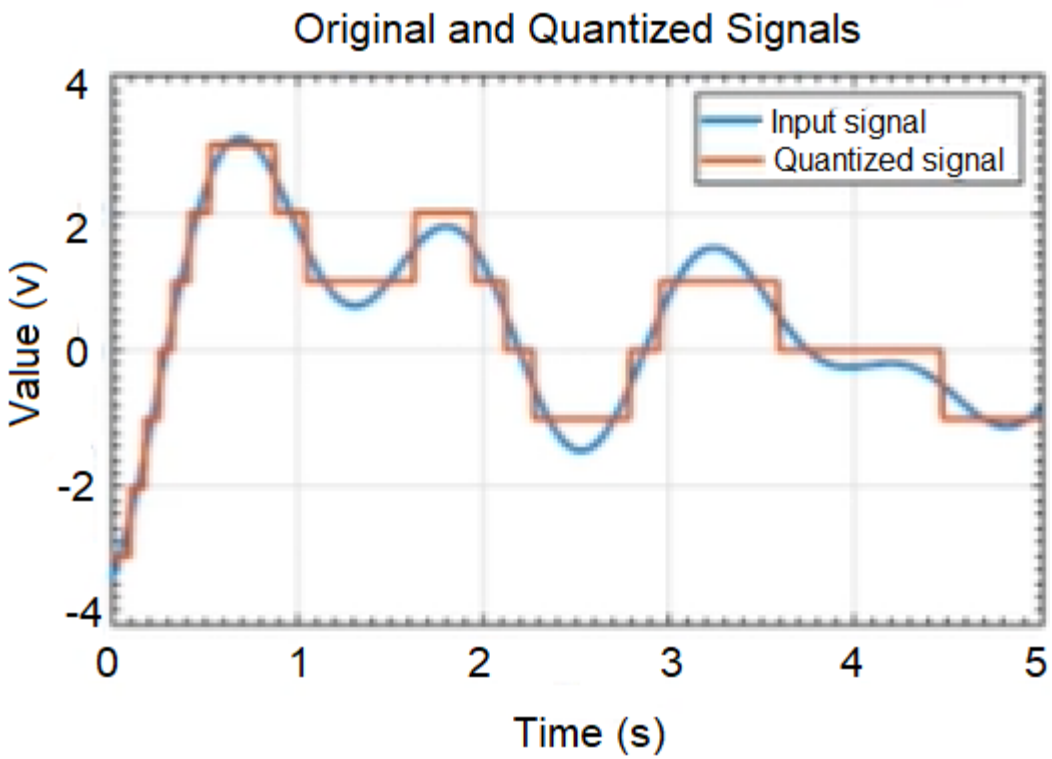
Analog Inputs

How do we read analog sensors?



Analog Inputs

Quantization



$$\text{Maximum Quantization Error} = \frac{(V_H - V_L)}{2\,(2^n)}$$

where n is the number of bits of resolution of the ADC

Arduino Uno → 10-bit ADC means it will give digital value in the range of 0 – 1023 (2^10)

Analog Inputs: Summary

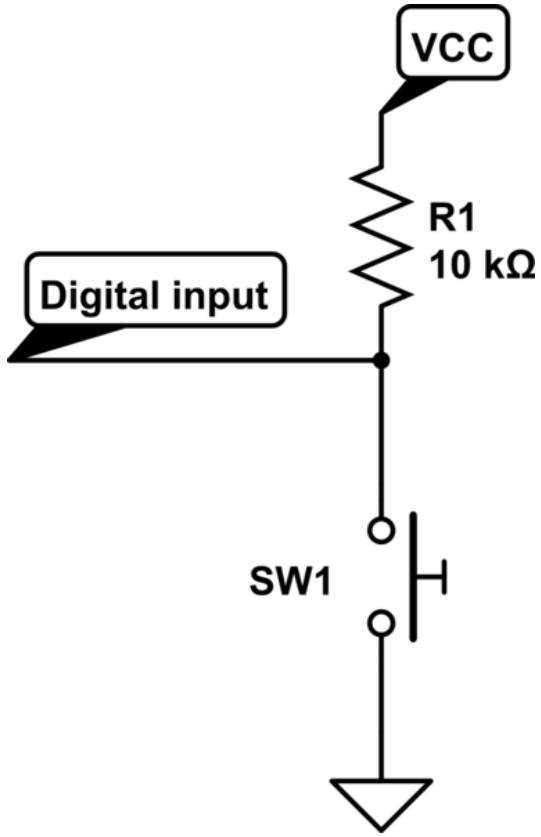
- Analogue inputs are really useful to acquire data
- Quantization error depends on ADC resolution
- Not a scalable approach – for each sensor you need ad additional ADC port
- ADCs can be expensive (for high resolution)
- Sampling rate can be limited
- Sensitive to external noise

How can we deal with more complex inputs, and achieve higher scaleability?

Digital Inputs

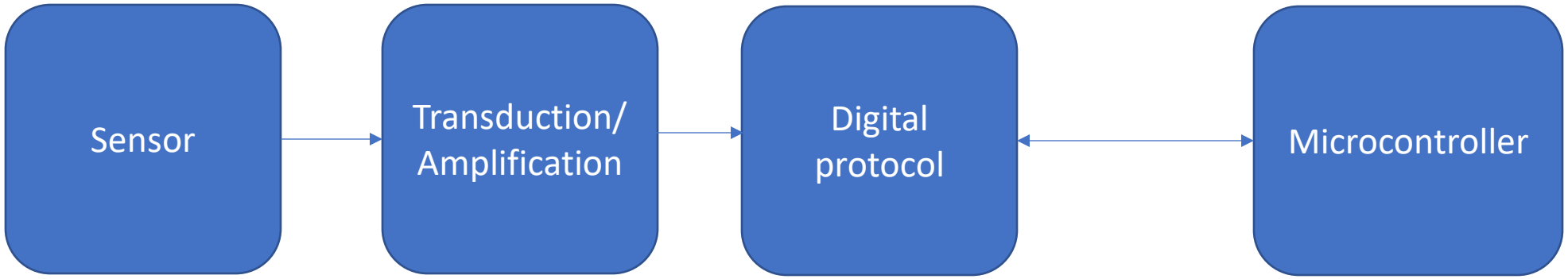
What would be an example of a digital sensor?

Digital Inputs



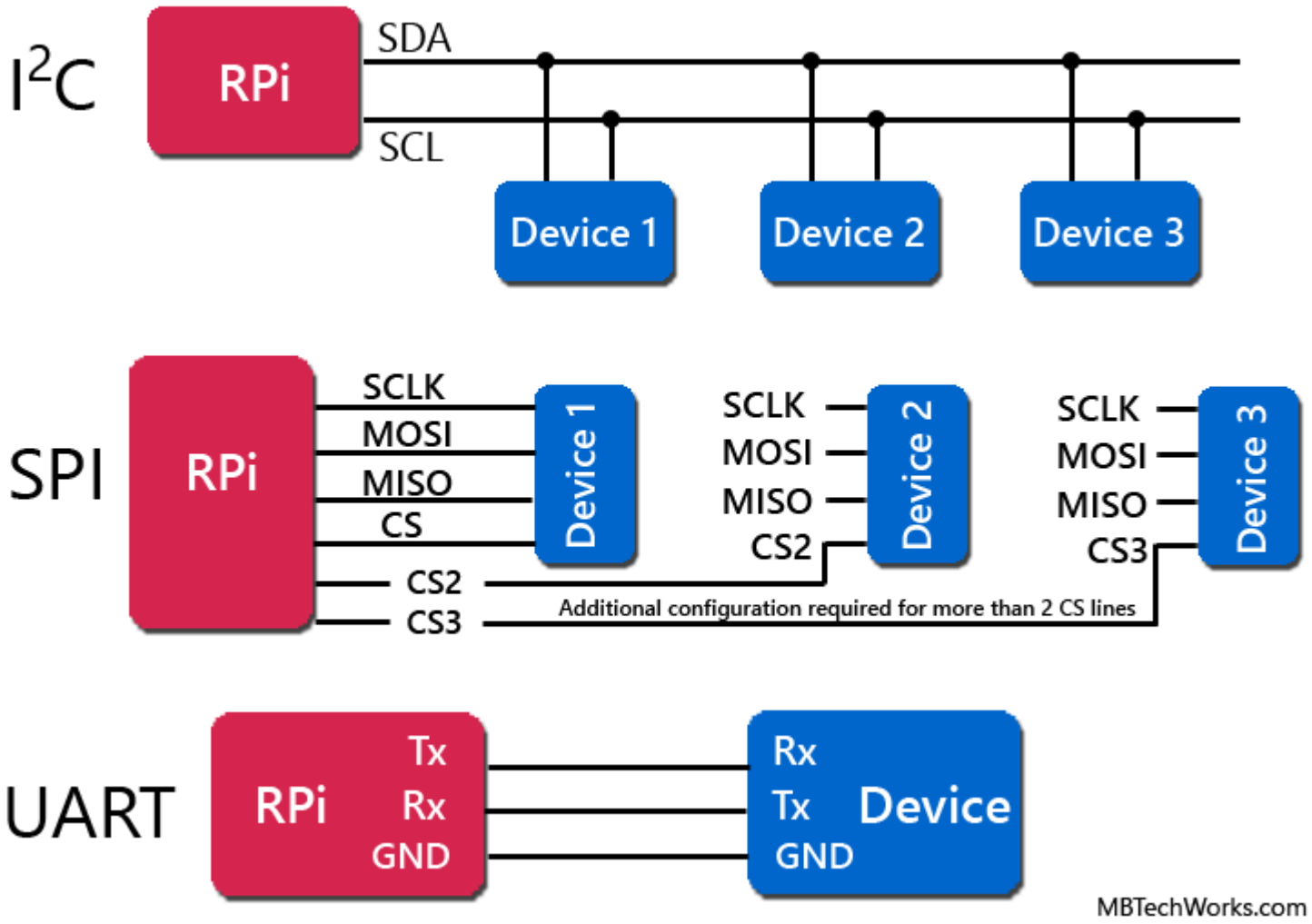
Can we use temporal/digital transmission of data?

Digital Protocols

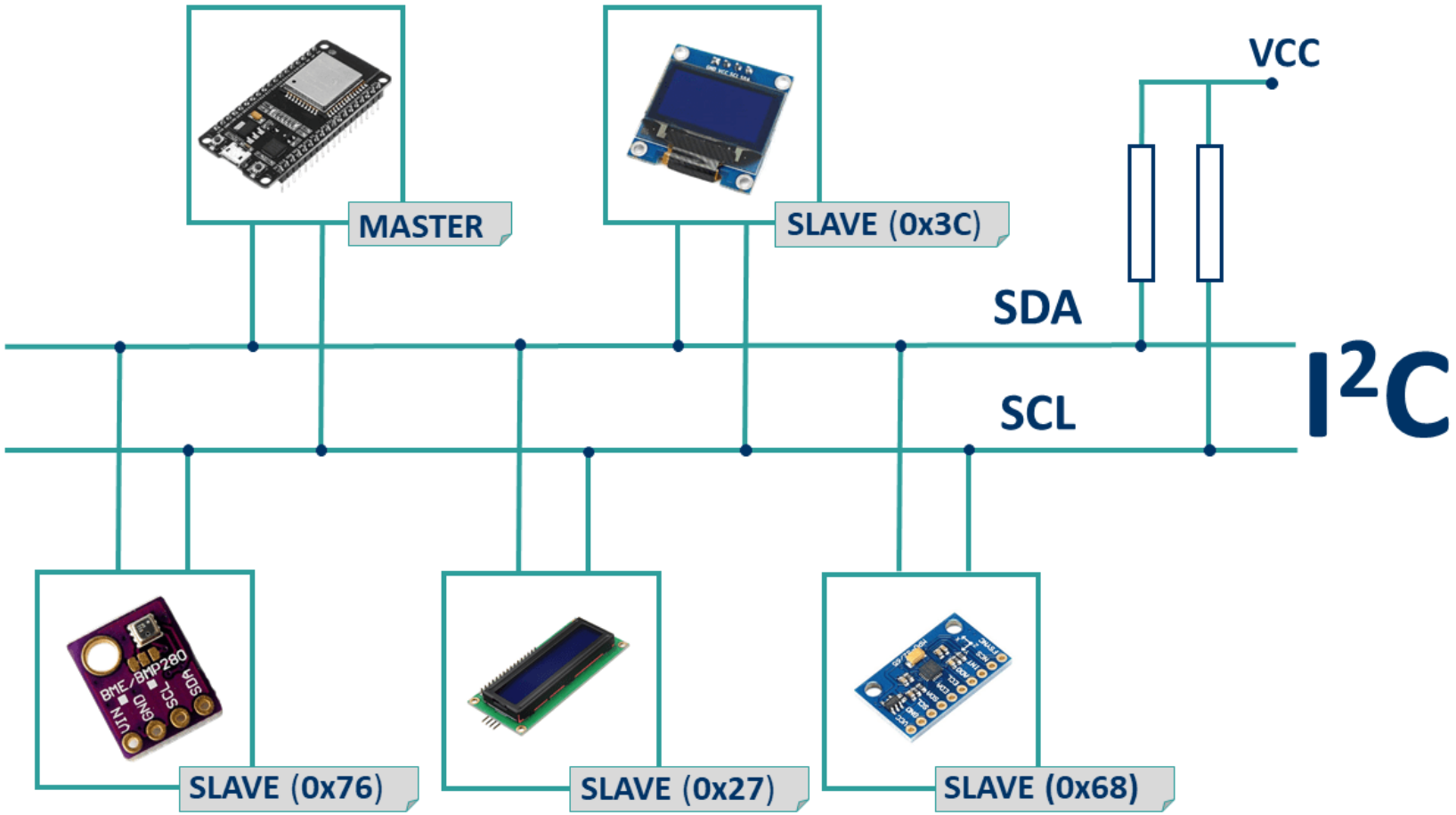


Digital Protocols

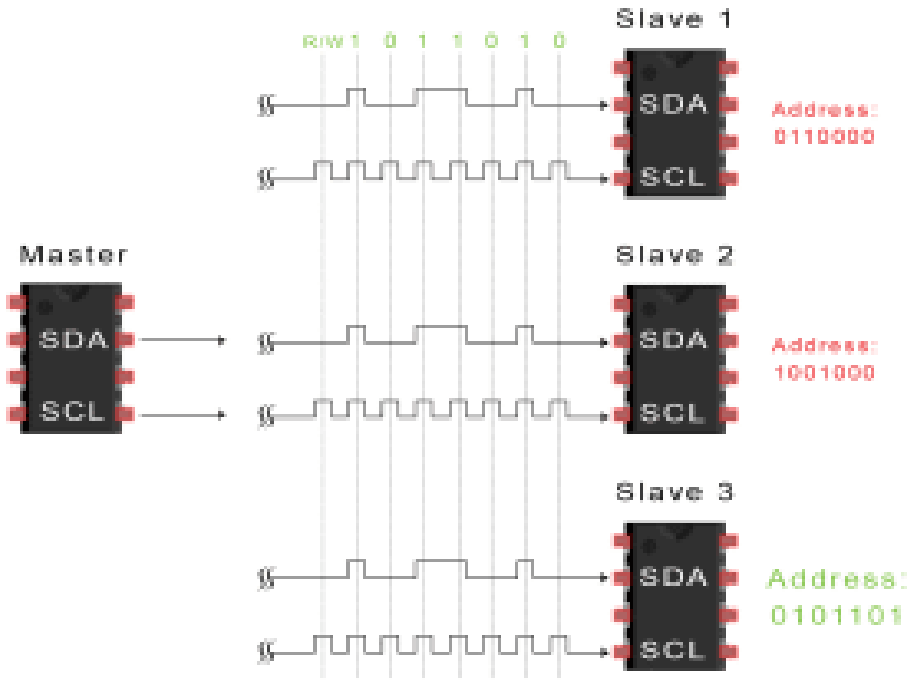
- I2C
- SPI
- UART
- CAN Bus

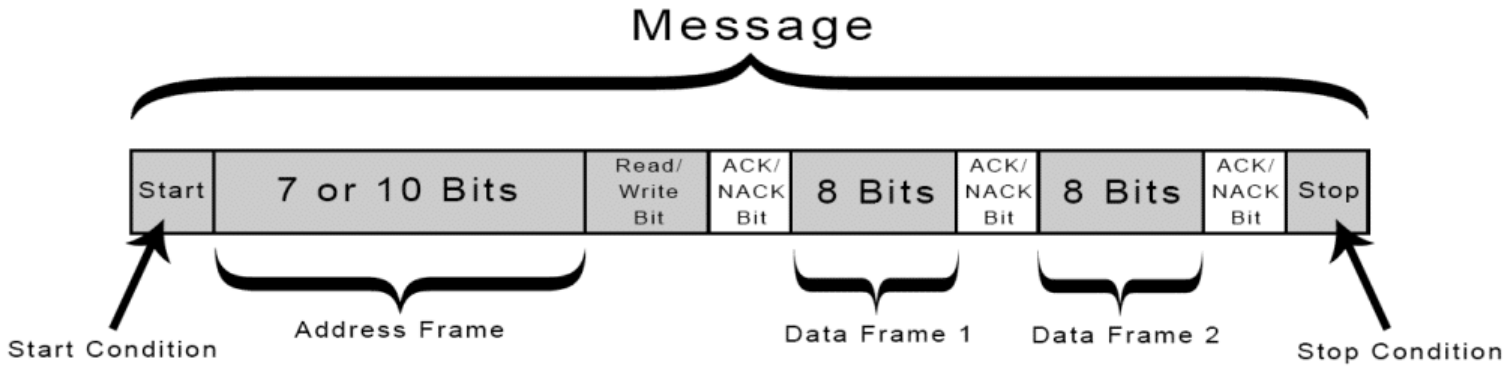


I2C


MASTER
SLAVE (0x3C)
VCC
SDA
SCL
I²C
SLAVE (0x76)
SLAVE (0x27)
SLAVE (0x68)

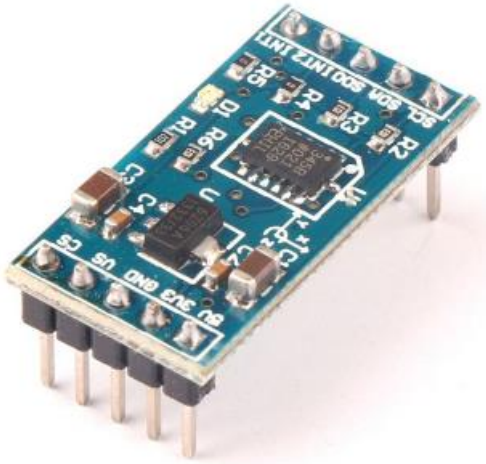
I2C


Slave 1
R/W 1 0 1 1 0 1 0
SDA
SCL
Address:
0110000
Master
SDA
SCL
Slave 2
SDA
SCL
Address:
1001000
Slave 3
SDA
SCL
Address:
0101101


Message
Start
7 or 10 Bits
Read/ Write Bit
ACK/ NACK Bit
8 Bits
ACK/ NACK Bit
8 Bits
ACK/ NACK Bit
Stop
Start Condition
Address Frame
Data Frame 1
Data Frame 2
Stop Condition

I2C

HMC5883L
Magnetometer

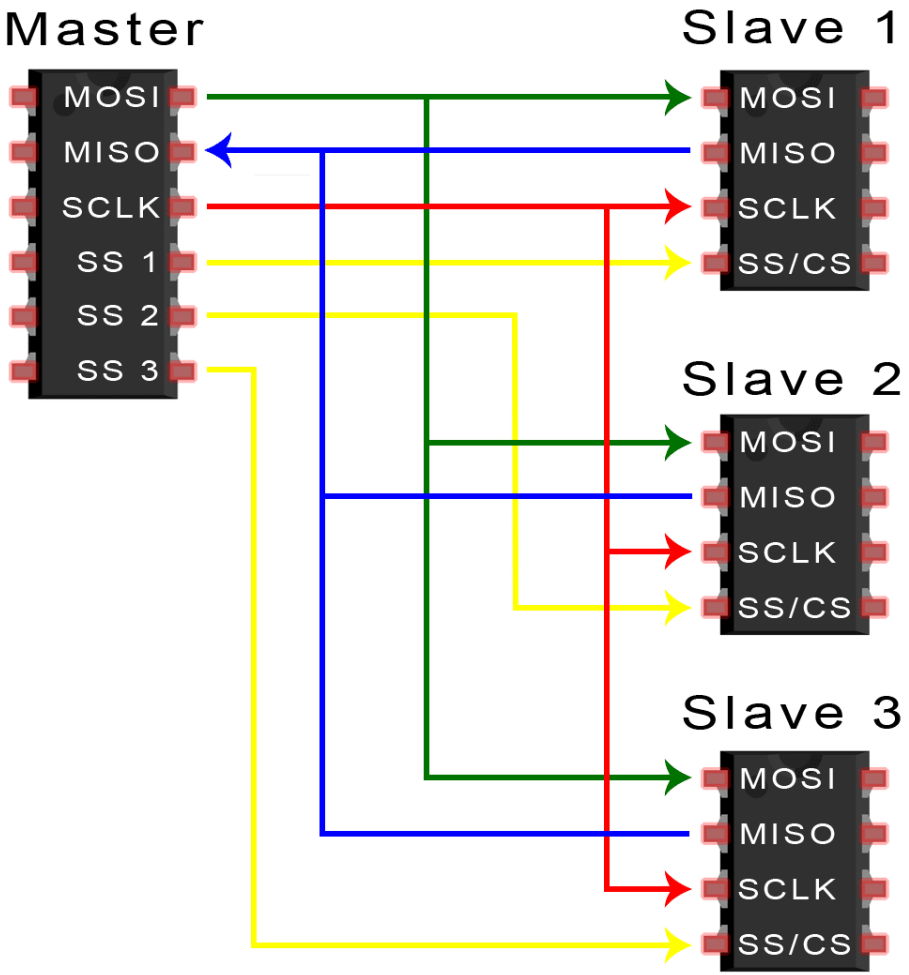


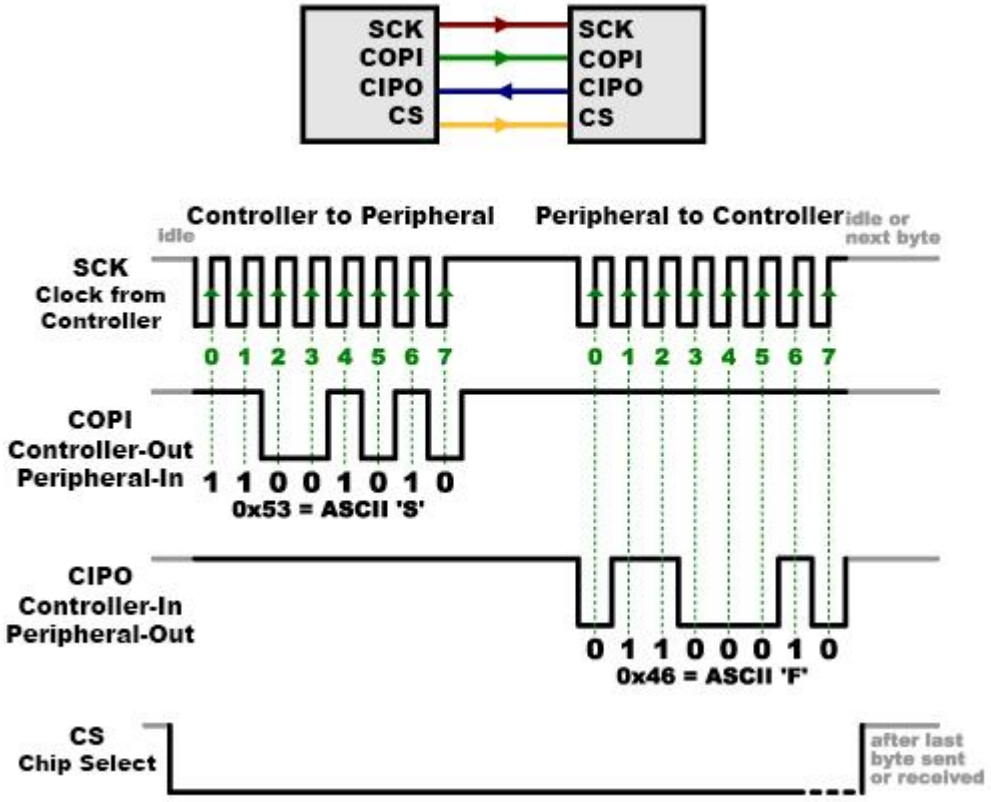
ADXL345
Accelerometer

MMA7660FC
Accelerometer

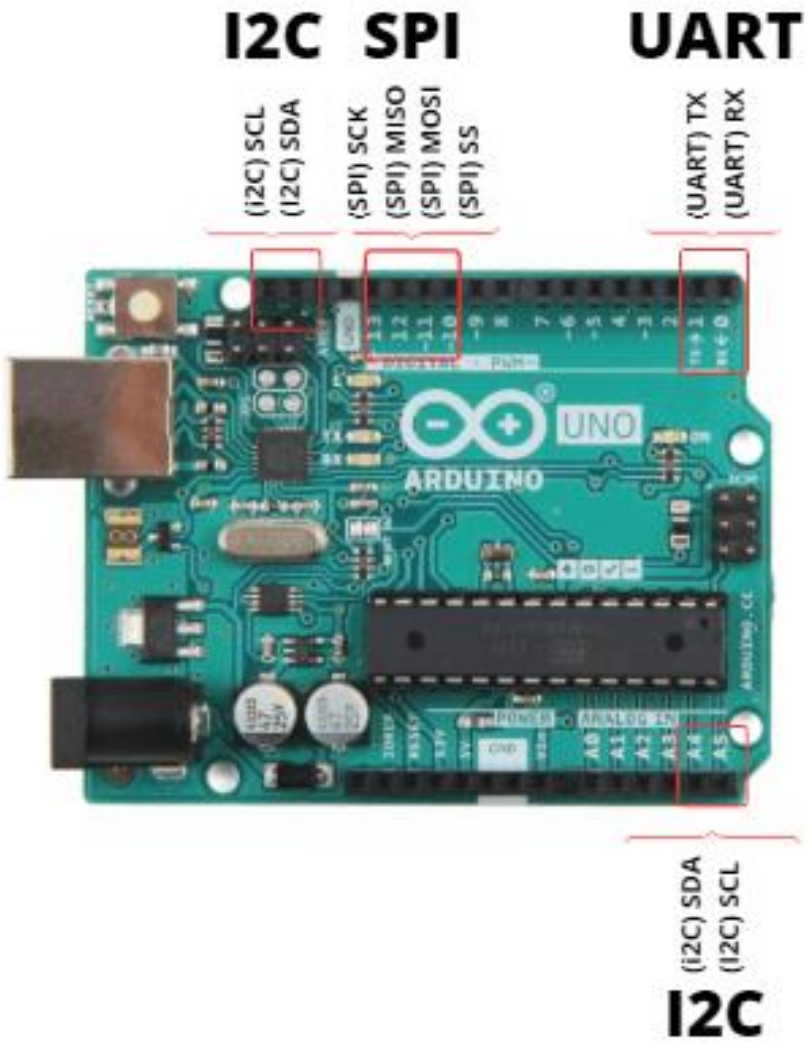
MPU6050
Accelerometer + Gyroscope

SPI (Serial Peripheral Interface)





Digital Protocols



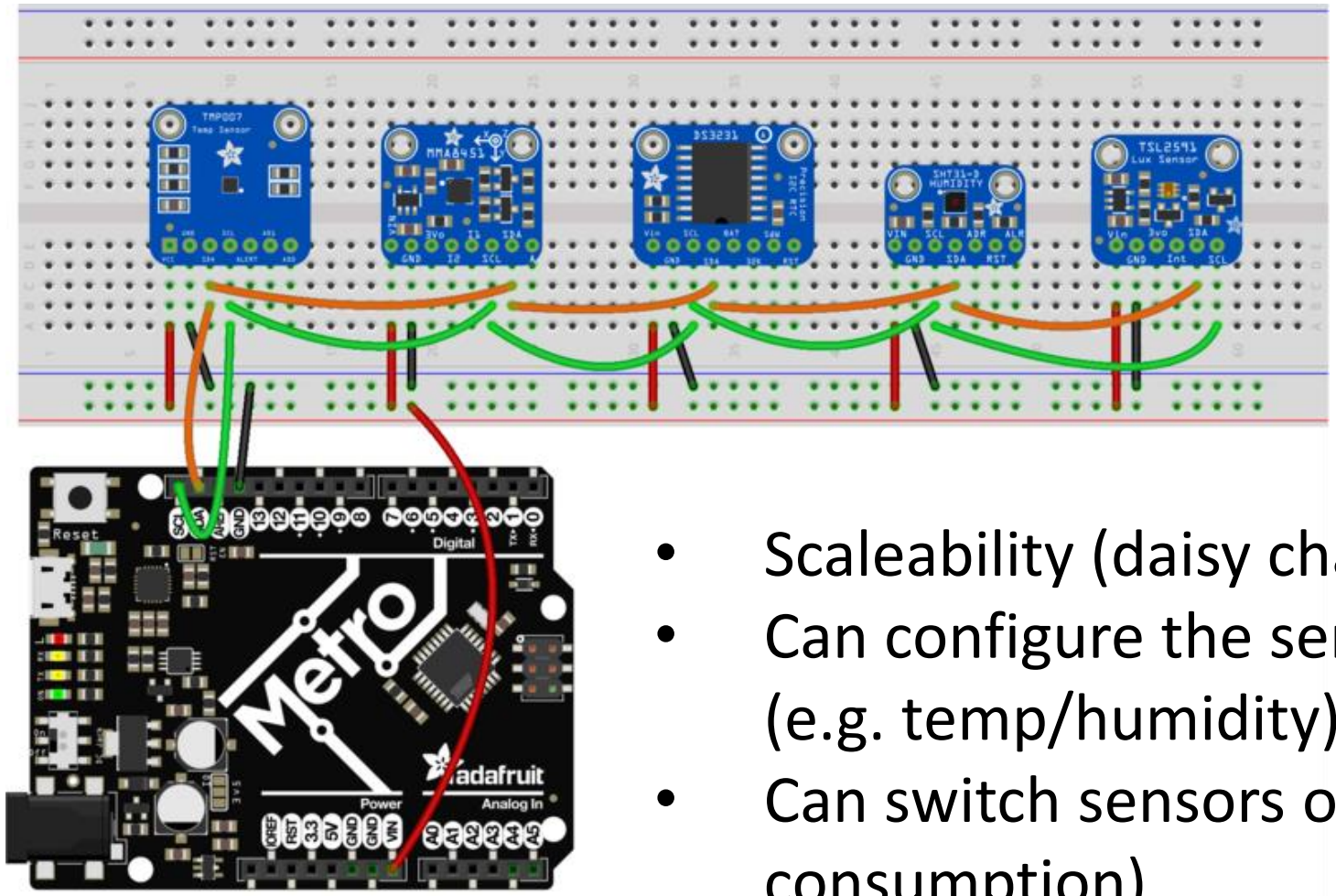
- Microcontrollers have dedicated digital pins for I2C, SPI, UART
- Can have multiple I2C ports
- Many sensors offer SPI or I2C connectivity

Digital Protocols

Characteristic	I2C	SPI	Bottom line
Speed	(100k, 400k, 1M, 3.4M) [Hz]	up to 25 [MHz]	SPI is much faster than I2C.
Scalability	Up to 127, 255, 1024 devices on the single bus depending on addressing. Note that one bus is only 2 wires (SDA, SCL)! Many sensors allow you to connect multiple numbers of the same device to the single bus with different addresses.	Hardly scalable. Every new device on the bus requires additional CS line to control this slave device.	I2C is much more scalable than SPI.
Bus length	Up to 1 meter long for 100kHz, shorter for higher speeds.	Best for short distances.	I2C is more suitable when distance between master and slave is significant.

The appropriate sensor + protocol should be selected for the application

Why are these digital data protocols useful?



- Scaleability (daisy chain)
- Can configure the sensors (data-rate), what to send (e.g. temp/humidity)
- Can switch sensors on/off (reduce power consumption)

What do we need to think about when incorporating sensors into products?

Sensor Design Selection…

- Form factor
- Sensitivity
- Protocol (analog, digital protocol?)
- Robustness
- Power draw
- Signal-to-noise ratio
- Cost
- Amount of signal processing required

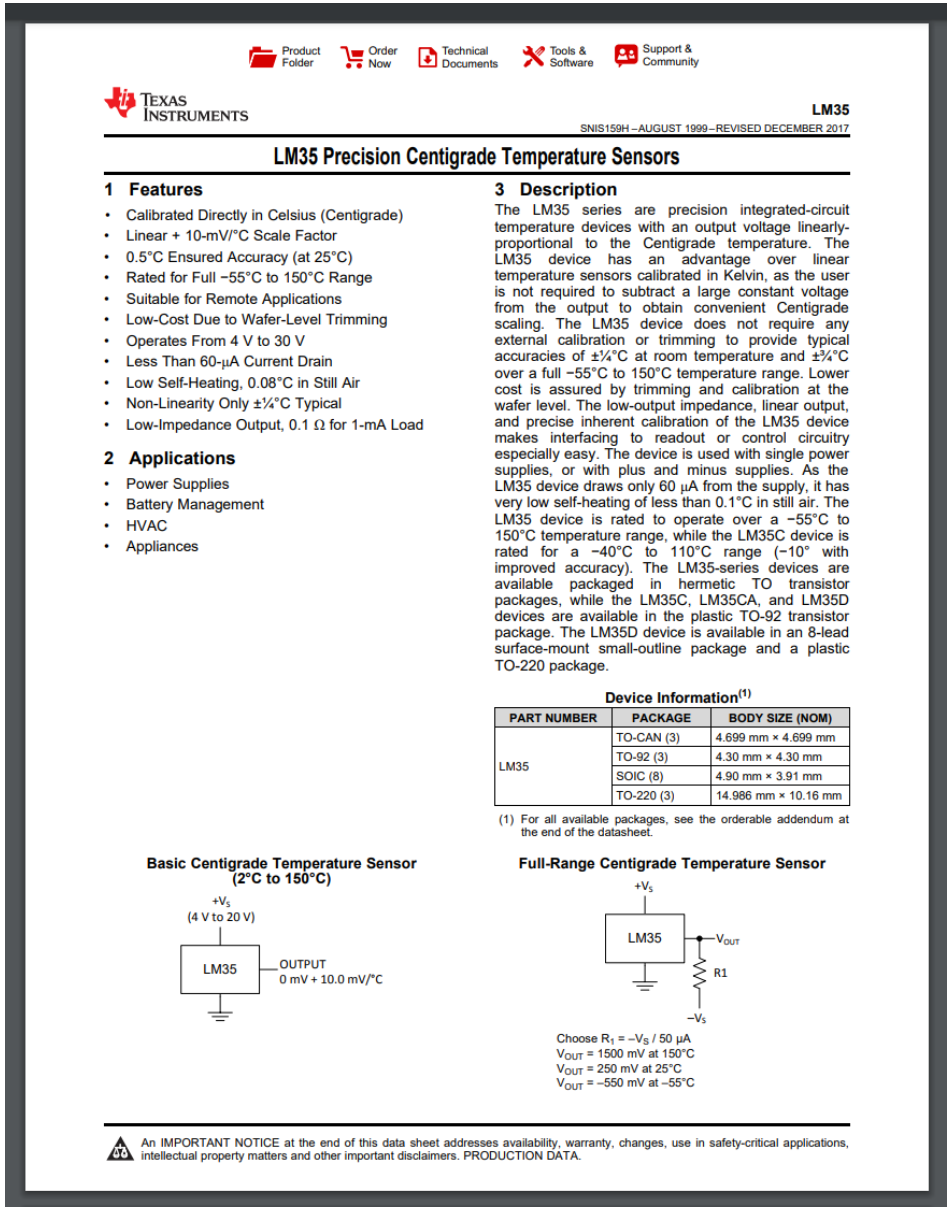
Sensor Design Selection…

- Form factor
- Sensitivity
- Robustness
- Power draw
- Signal-to-noise ratio
- Cost
- Amount of signal processing required

Can be found on data-sheets
- Component websites (mouser, digikey, RS, Farnell etc.)

Sensors: Data-sheets

LM35



Product Folder | Order Now | Technical Documents | Tools & Software | Support & Community

TEXAS INSTRUMENTS

LM35

SNIS159H – AUGUST 1999 – REVISED DECEMBER 2017

LM35 Precision Centigrade Temperature Sensors

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-µA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

3 Description

The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 µA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM35	TO-CAN (3)	4.699 mm × 4.699 mm
	TO-92 (3)	4.30 mm × 4.30 mm
	SOIC (8)	4.90 mm × 3.91 mm
	TO-220 (3)	14.986 mm × 10.16 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Basic Centigrade Temperature Sensor (2°C to 150°C)

Full-Range Centigrade Temperature Sensor

Choose $R_1 = -V_S / 50\ \mu A$
$V_{OUT} = 1500$ mV at 150°C
$V_{OUT} = 250$ mV at 25°C
$V_{OUT} = -550$ mV at −55°C

An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

https://www.ti.com/lit/ds/symlink/lm35.pdf

Sensors: Data-sheets

8.2.1 Basic Centigrade Temperature Sensor

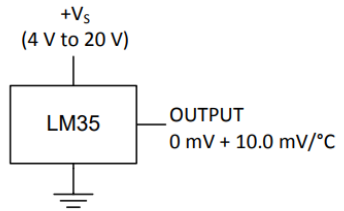


Figure 14. Basic Centigrade Temperature Sensor (2 °C to 150 °C)

8.2.1.1 Design Requirements

Table 1. Design Parameters

PARAMETER	VALUE
Accuracy at 25°C	±0.5°C
Accuracy from –55 °C to 150°C	±1°C
Temperature Slope	10 mV/°C

8.2.1.3 Application Curve

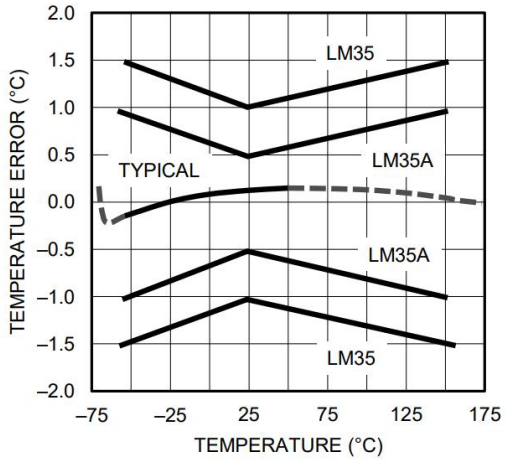


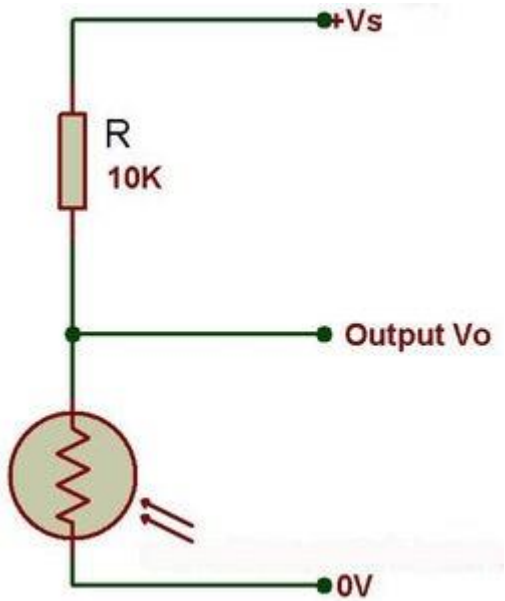
Figure 15. Accuracy vs Temperature (Ensured)

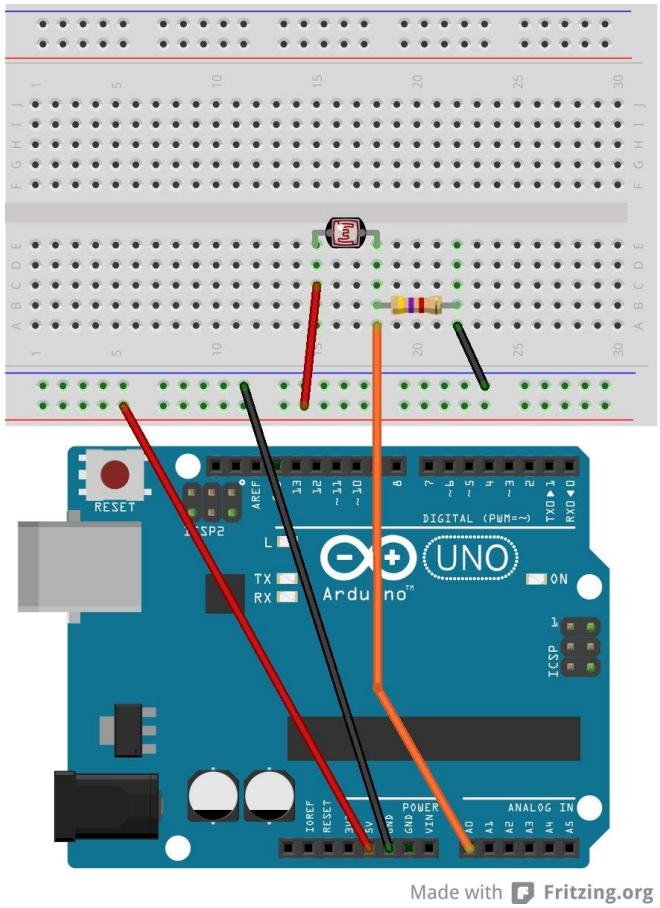
https://www.ti.com/lit/ds/symlink/lm35.pdf

Sensor for you to explore

- Load Cell
- Color Sensor (coming soon)
- Distance Sensor
- LDR (analog)
- Switch

LDR: Light Dependent Resistor

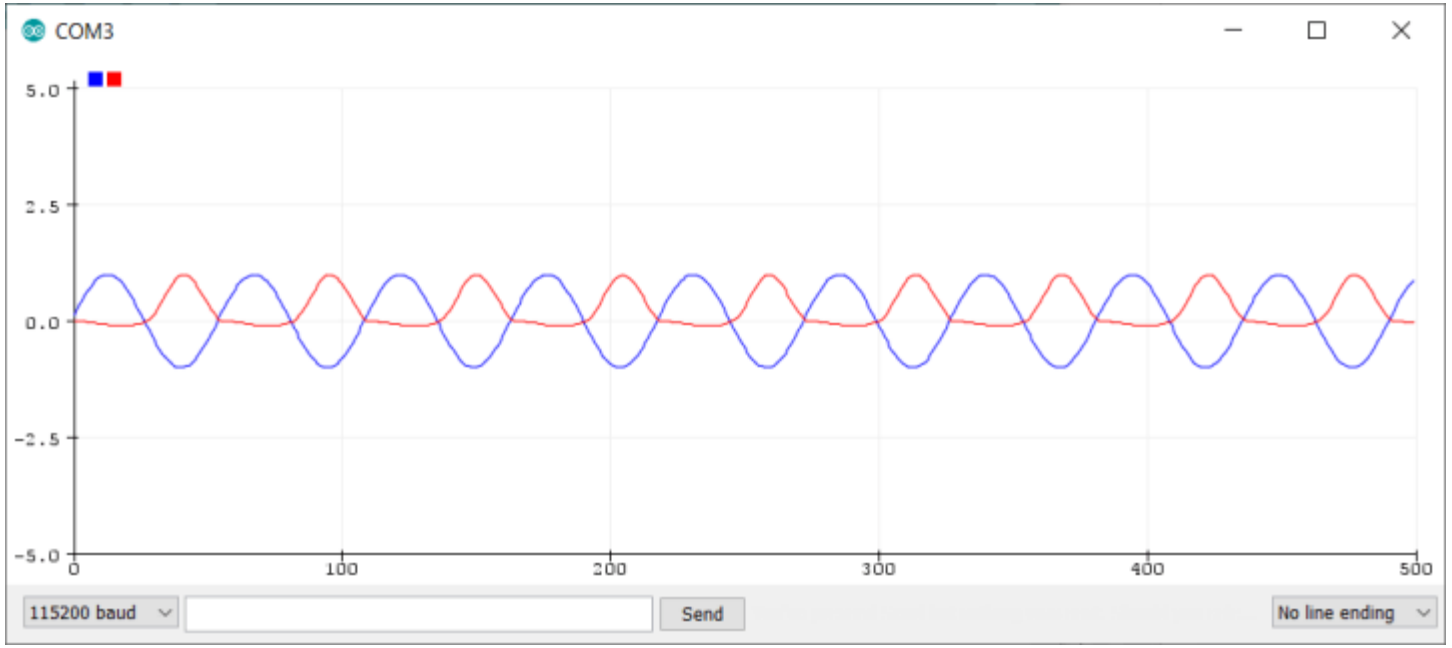




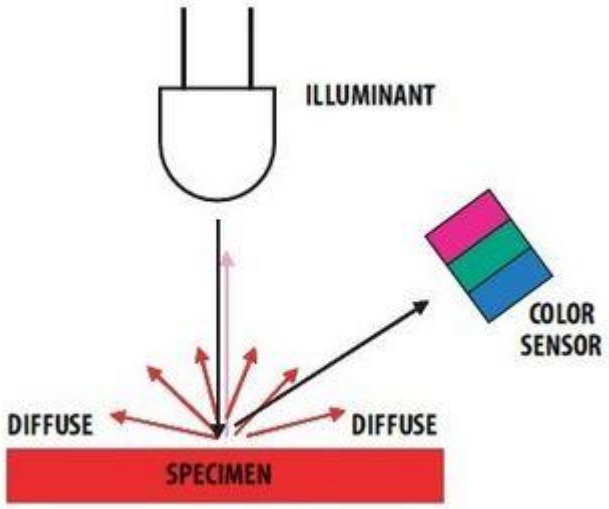
- What resistor value should I choose?
- What happens if you switch the LDR + base resistor around?

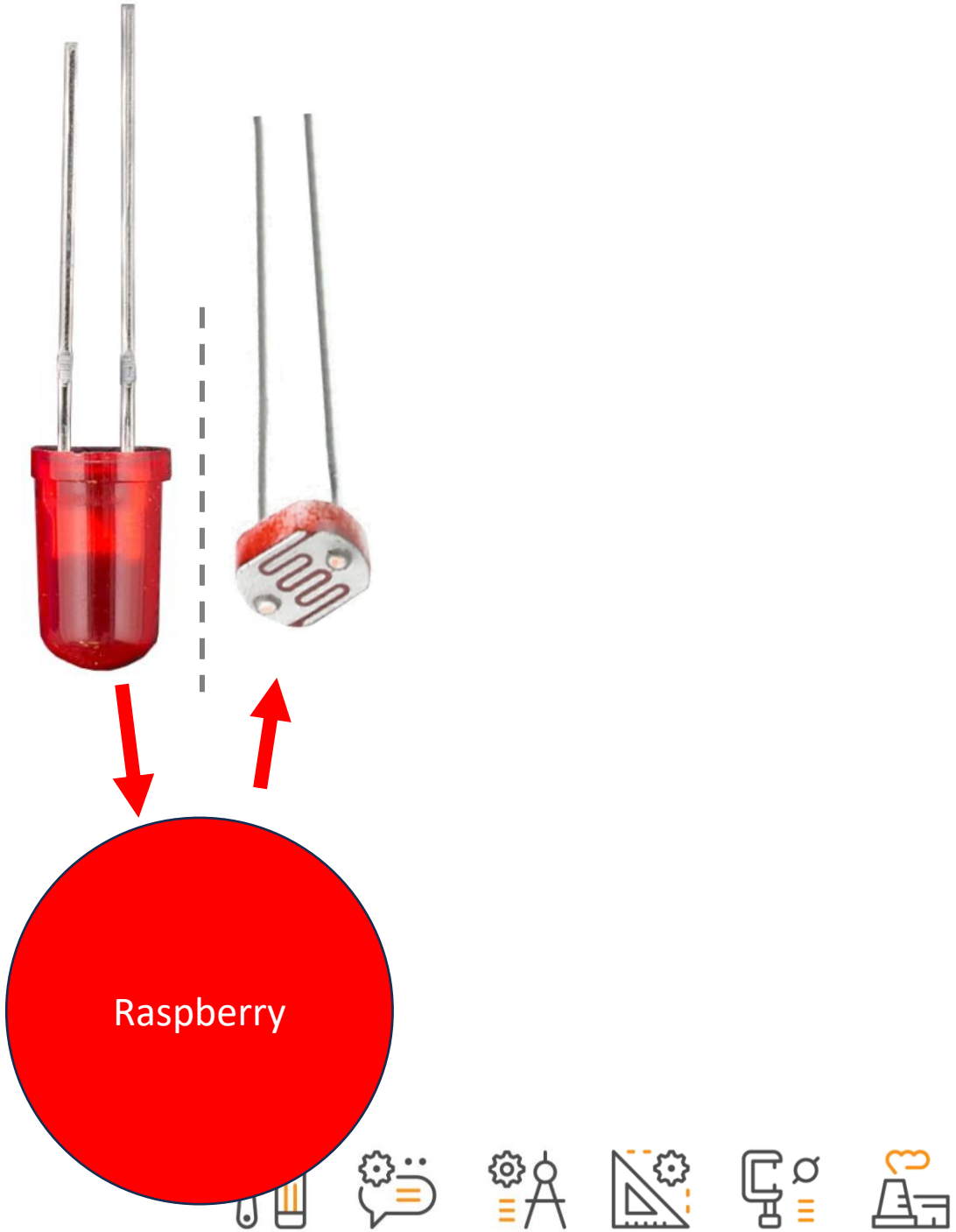
LDR: Light Dependent Resistor

```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0;  // variable to store the value read

void setup() {
  Serial.begin(9600);           //  setup serial
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
  Serial.println(val);          // debug value
}
```

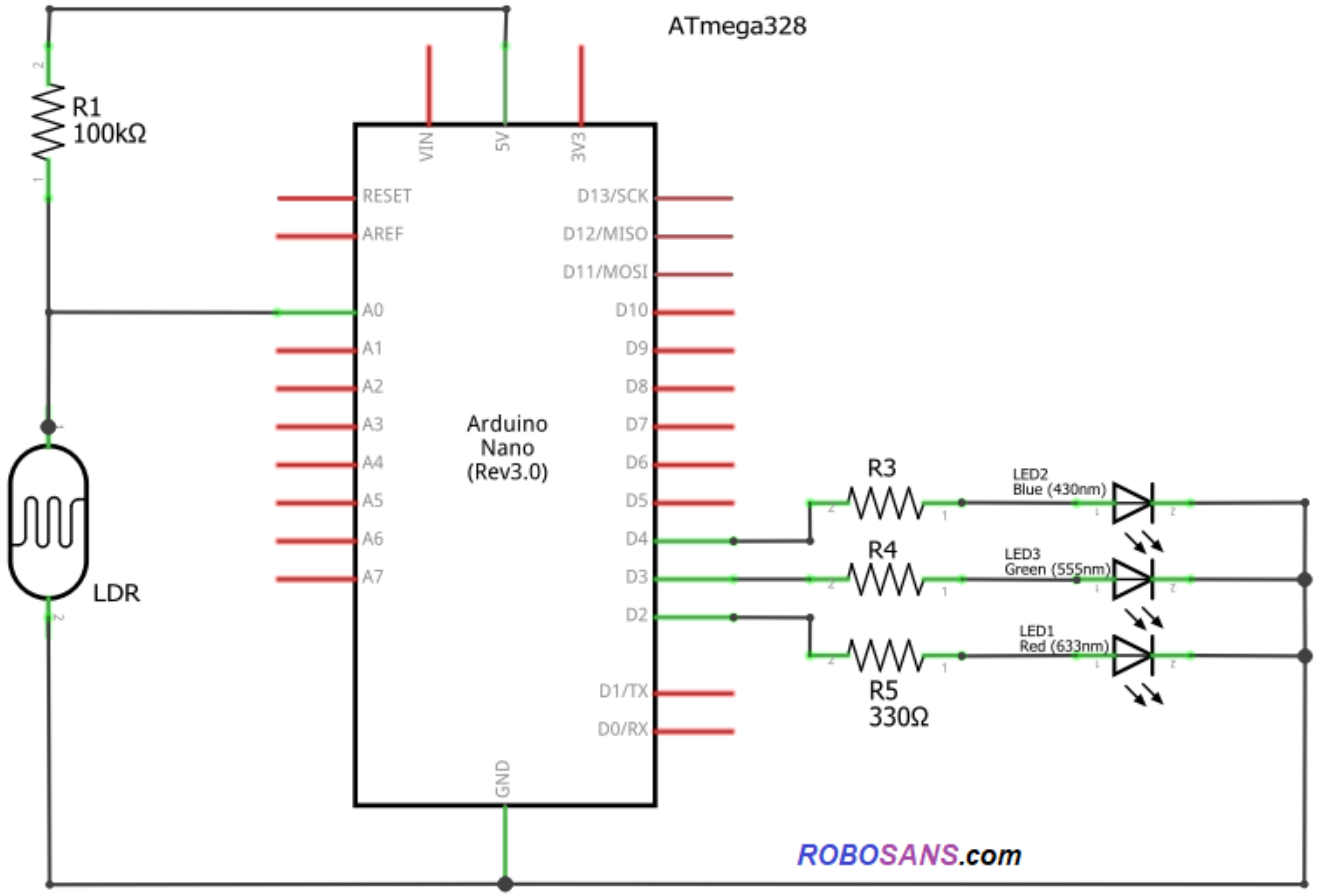


Color Sensor

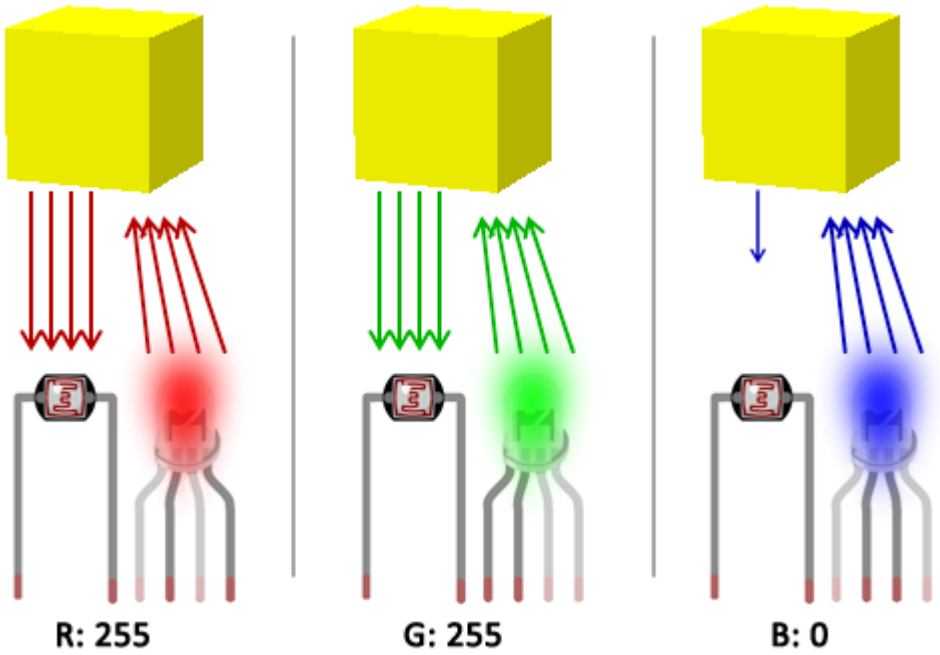




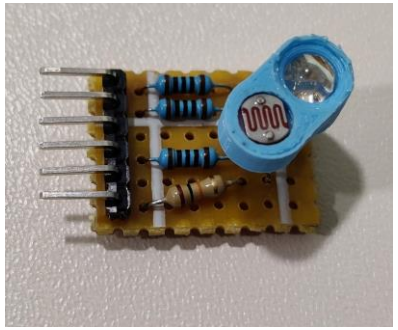
Color Sensor



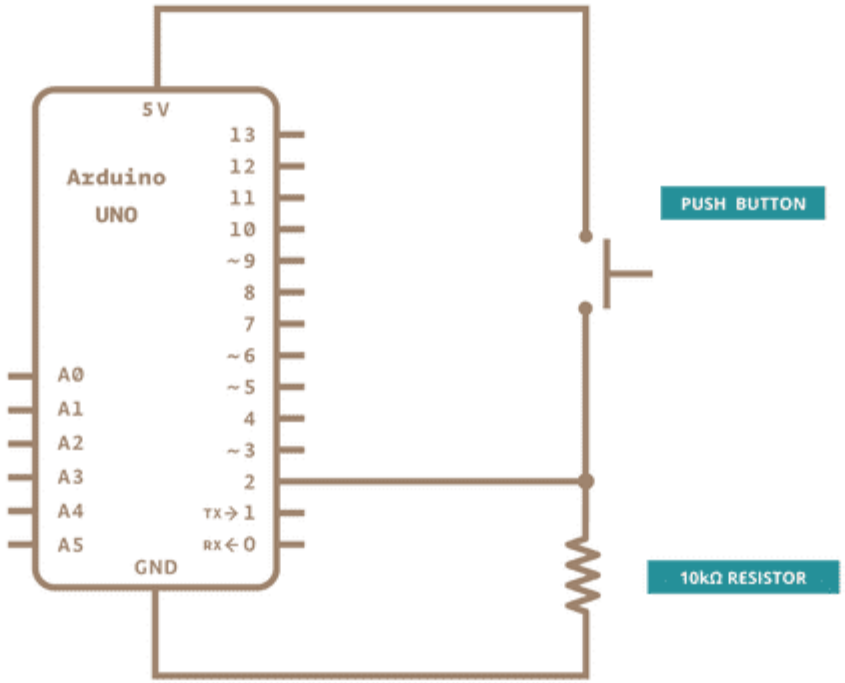
Building your sensor…



- How to shield your LDR from Diode
- What color(s) to use
- What distance to have the sensor from the raspberry
- How to package and integrated this into your gripper.



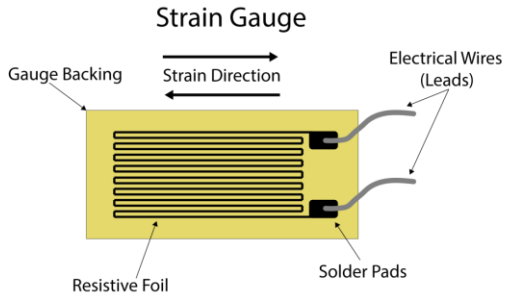
Switch (digital input)

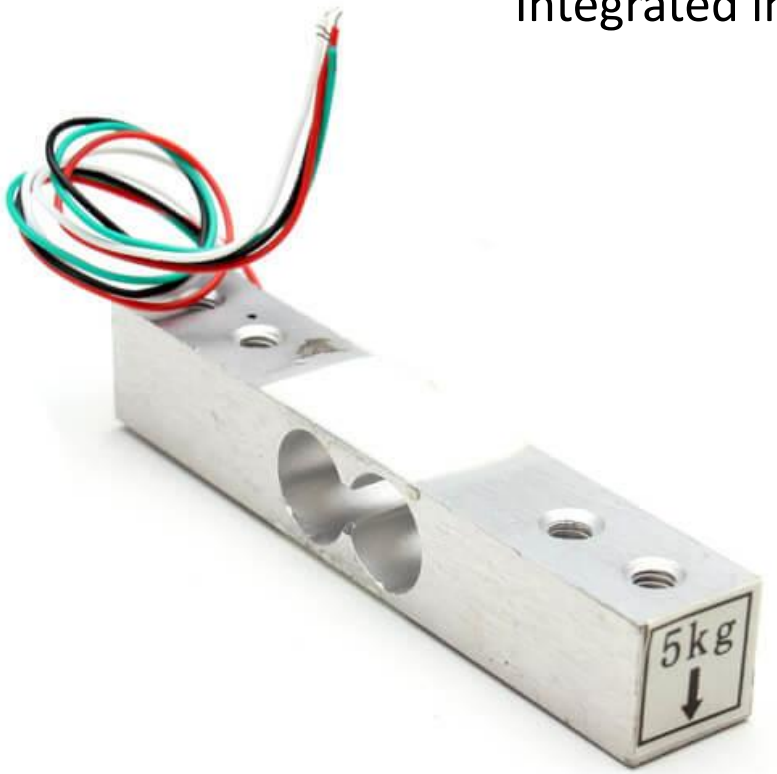



```
1  const int DIN_PIN = 7;
2
3  void setup(){
4      pinMode( DIN_PIN, INPUT );
5      Serial.begin( 9600 );
6  }
7
8  void loop(){
9      int value;
10
11     value = digitalRead( DIN_PIN );
12     Serial.println( value );
13
14     delay( 1000 );
15 }
```

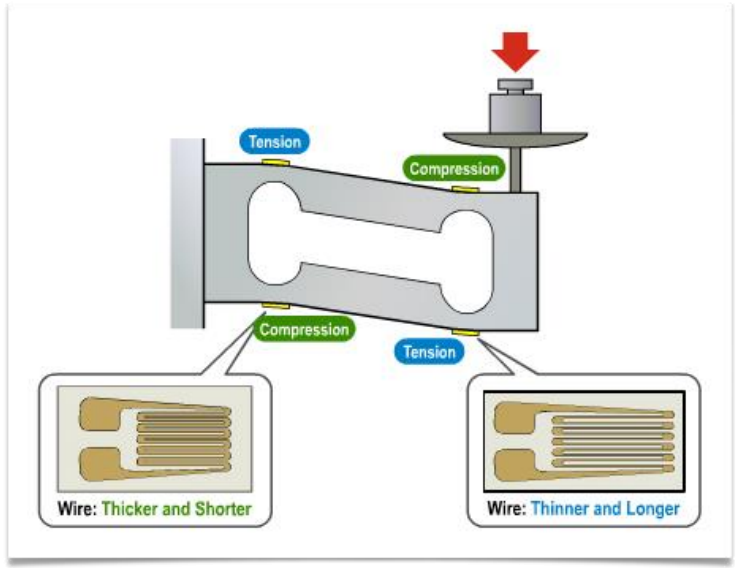
Load Cell

Strain gauge Sensors





Integrated into a Load Cell

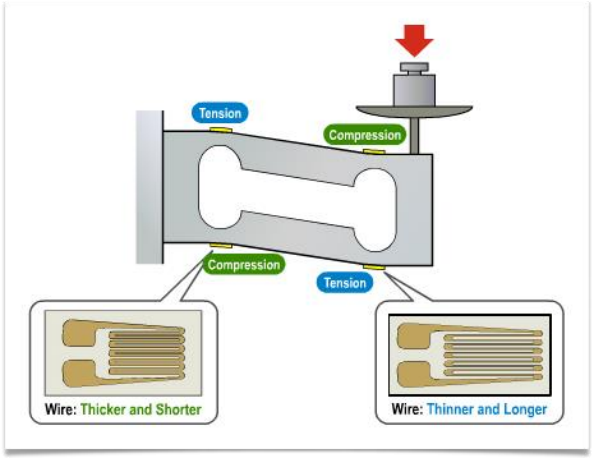


→ Resistance changes with strain

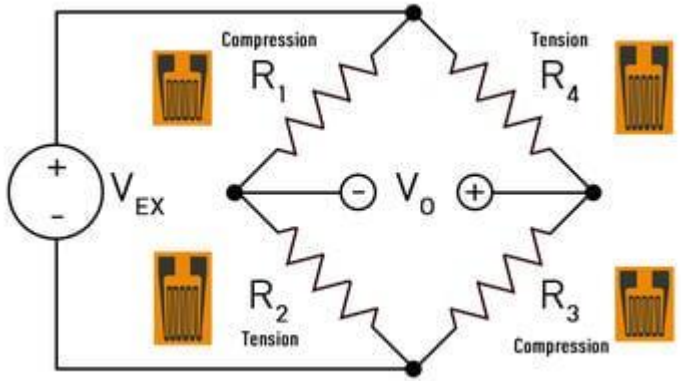
The placement amplifies the tension/compression measured

Load Cell

Load Cell



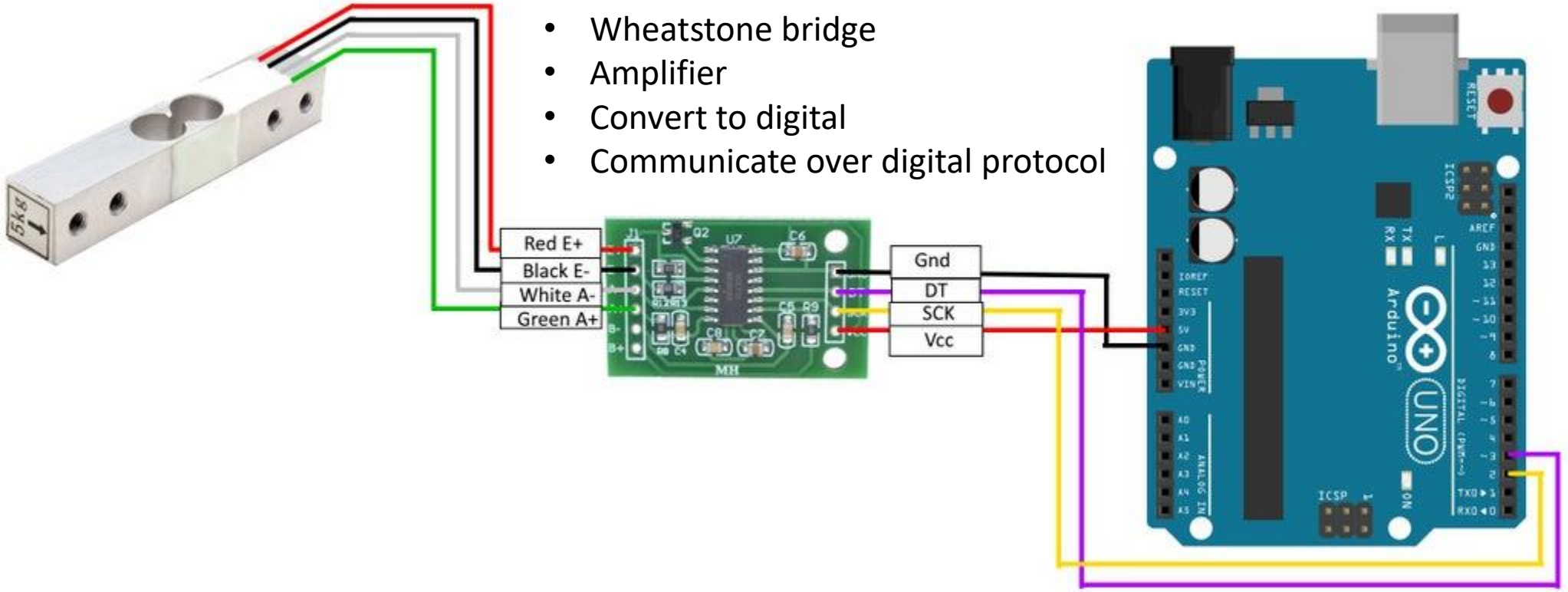
Wheatstone Bridge →
Subtract differences



Amplify differences

Load Cell Amplifier HX711

Load Cell

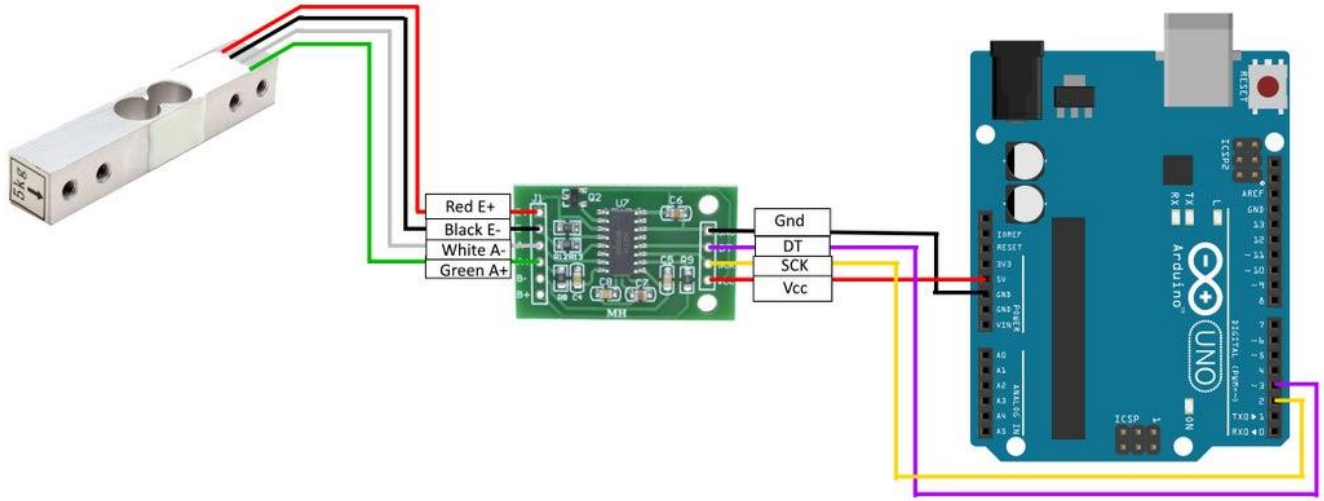


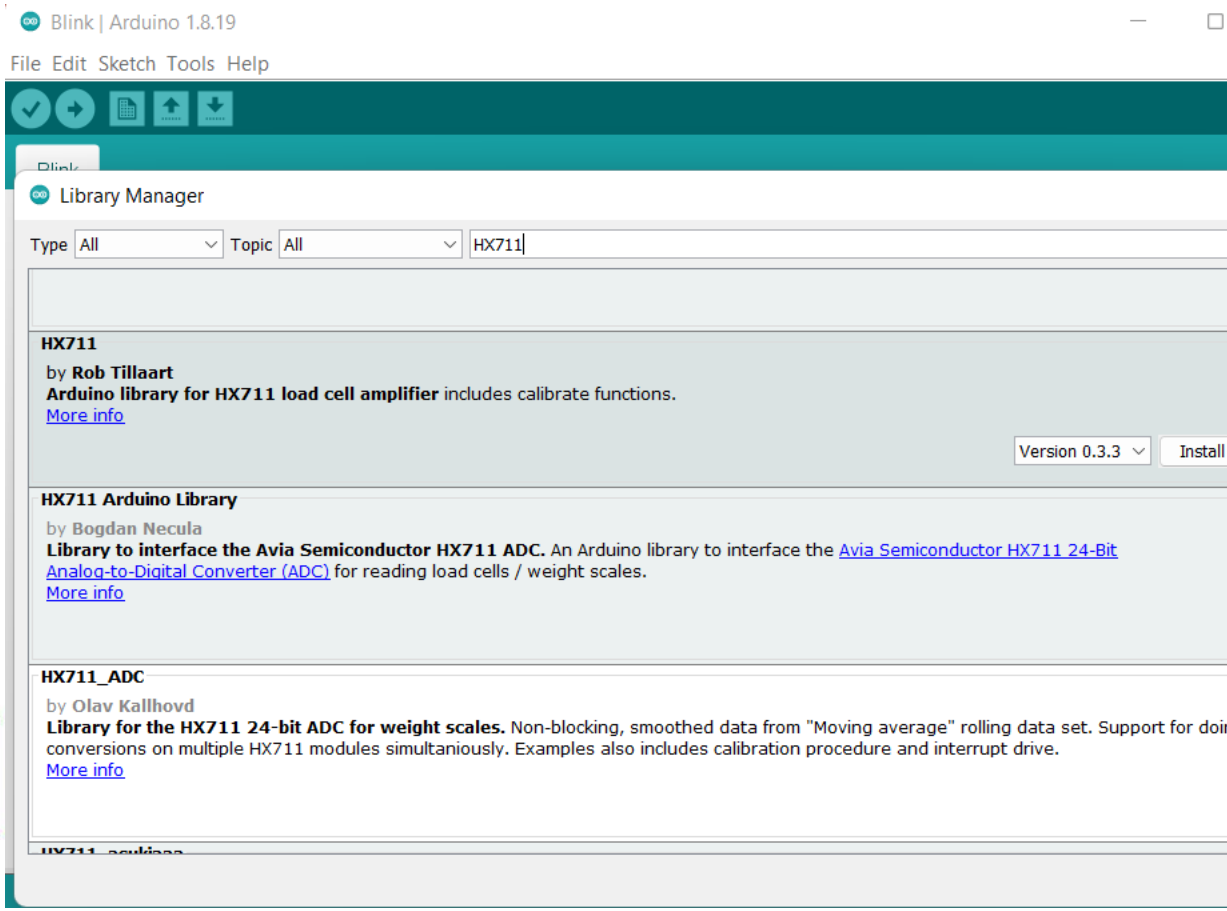
- Wheatstone bridge
- Amplifier
- Convert to digital
- Communicate over digital protocol

Use the Arduino HX111 library – provides the communication

Load Cell

Use the correct Library

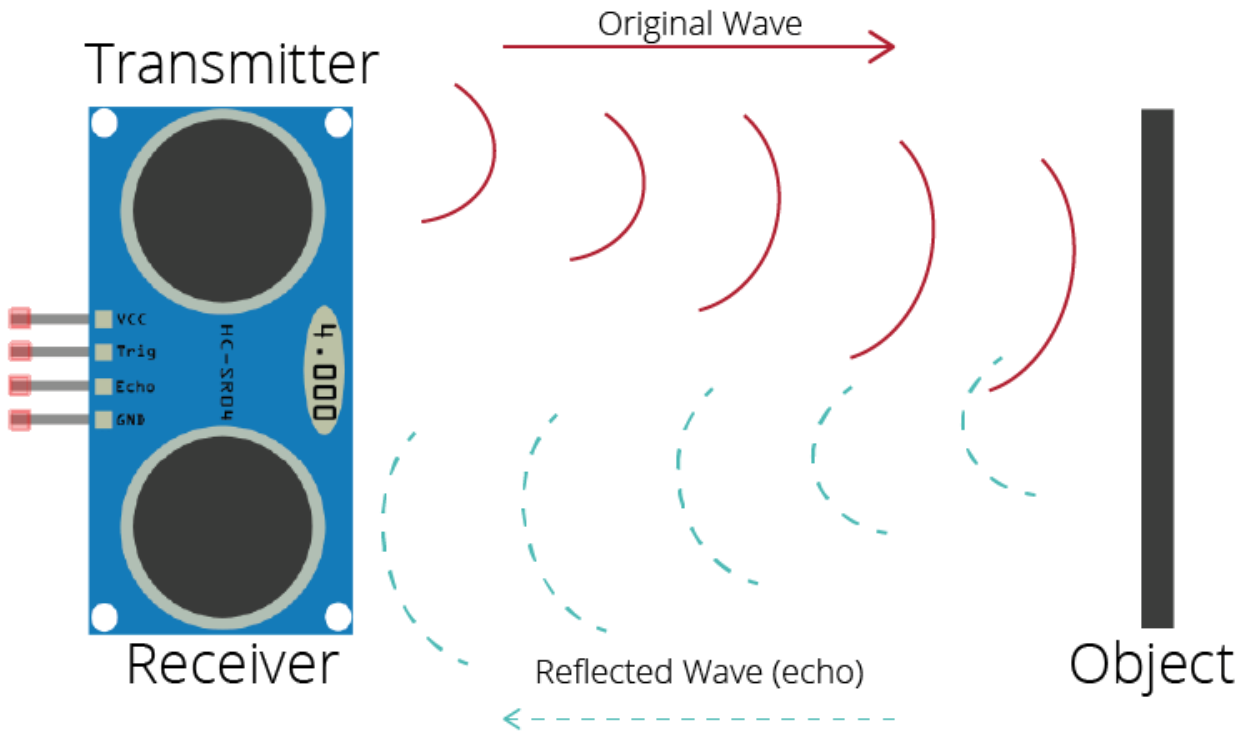


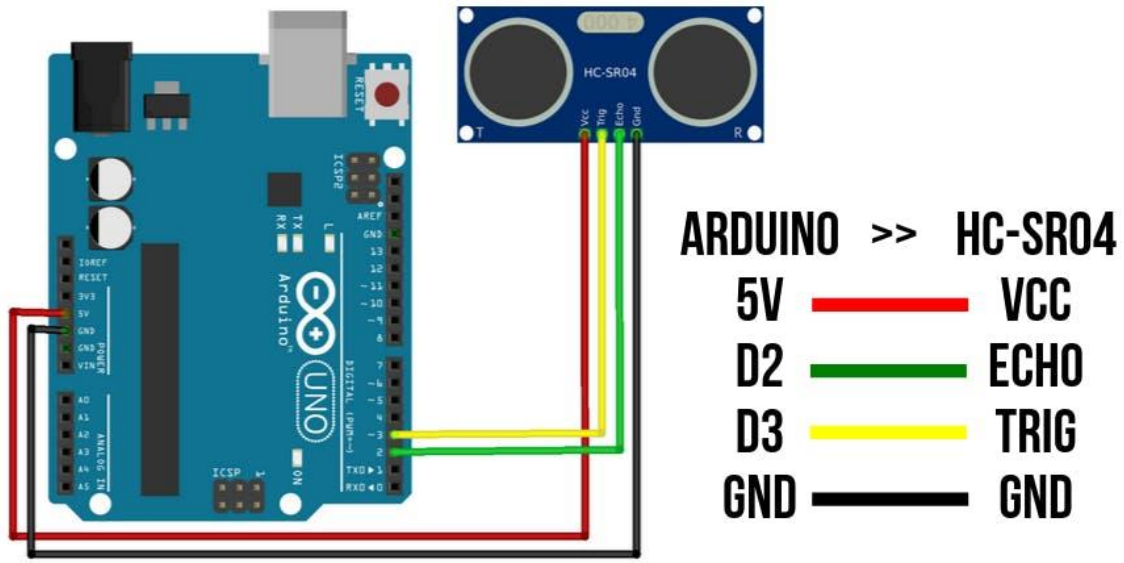


Add library (HX711)

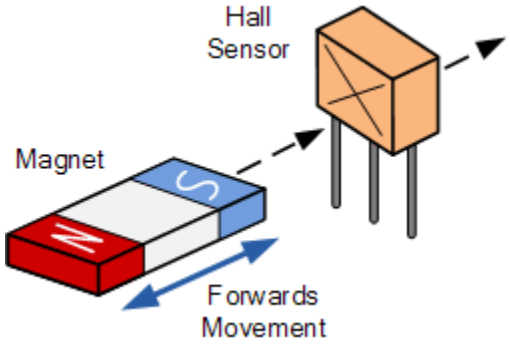
Nice tutorial:
https://randomnerdtutorials.com/arduino-load-cell-hx711/

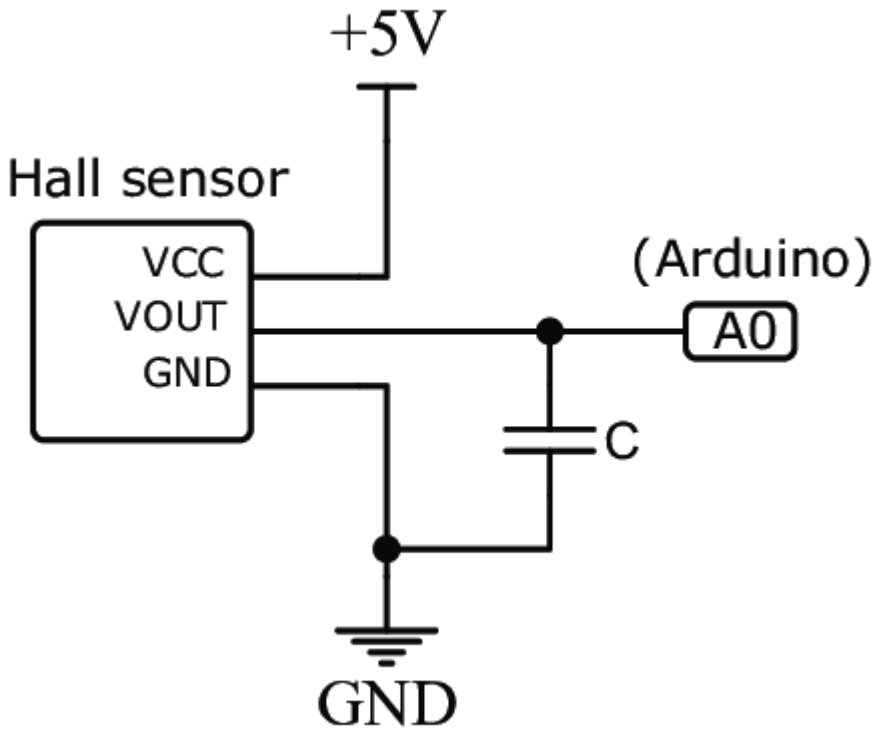
Distance Sensor





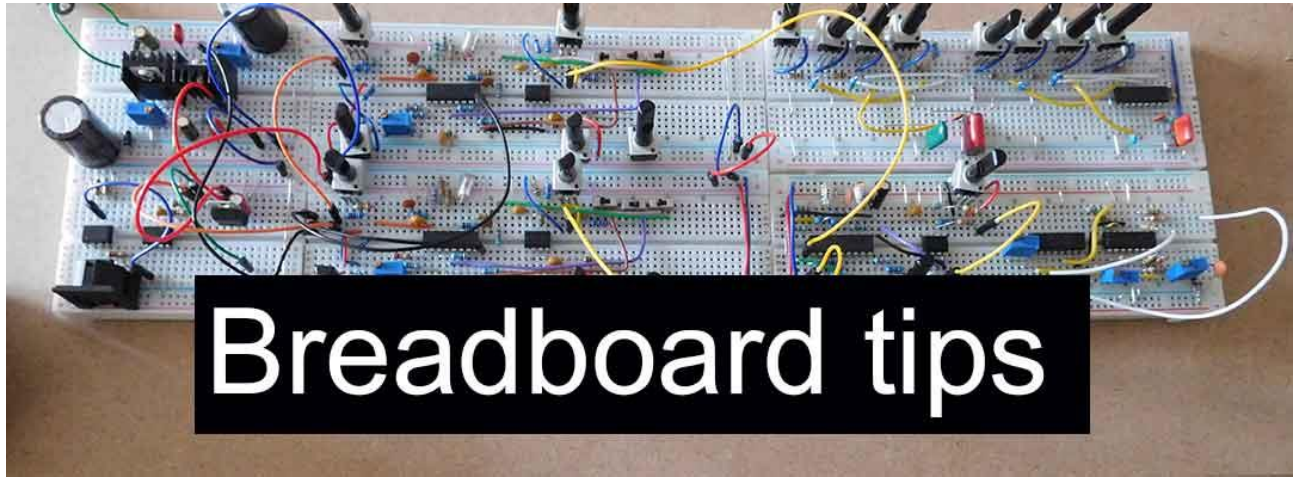
Hall Effect Sensors



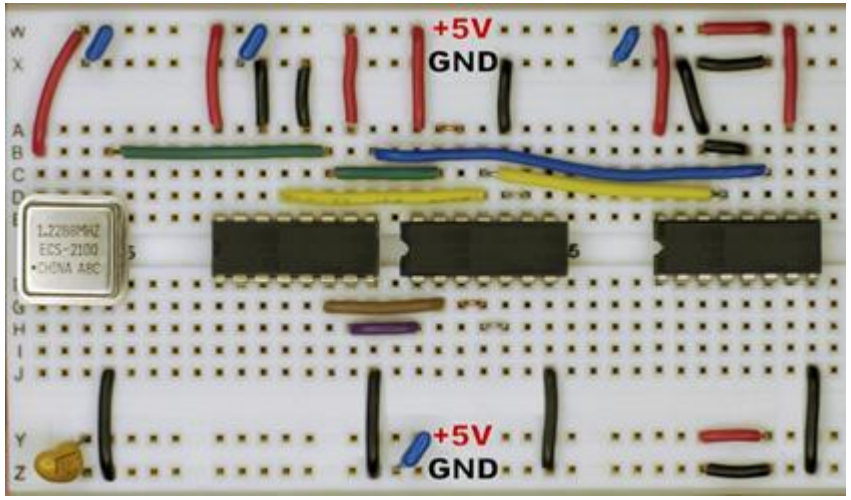


Prototyping Electronics: Breadboard

No



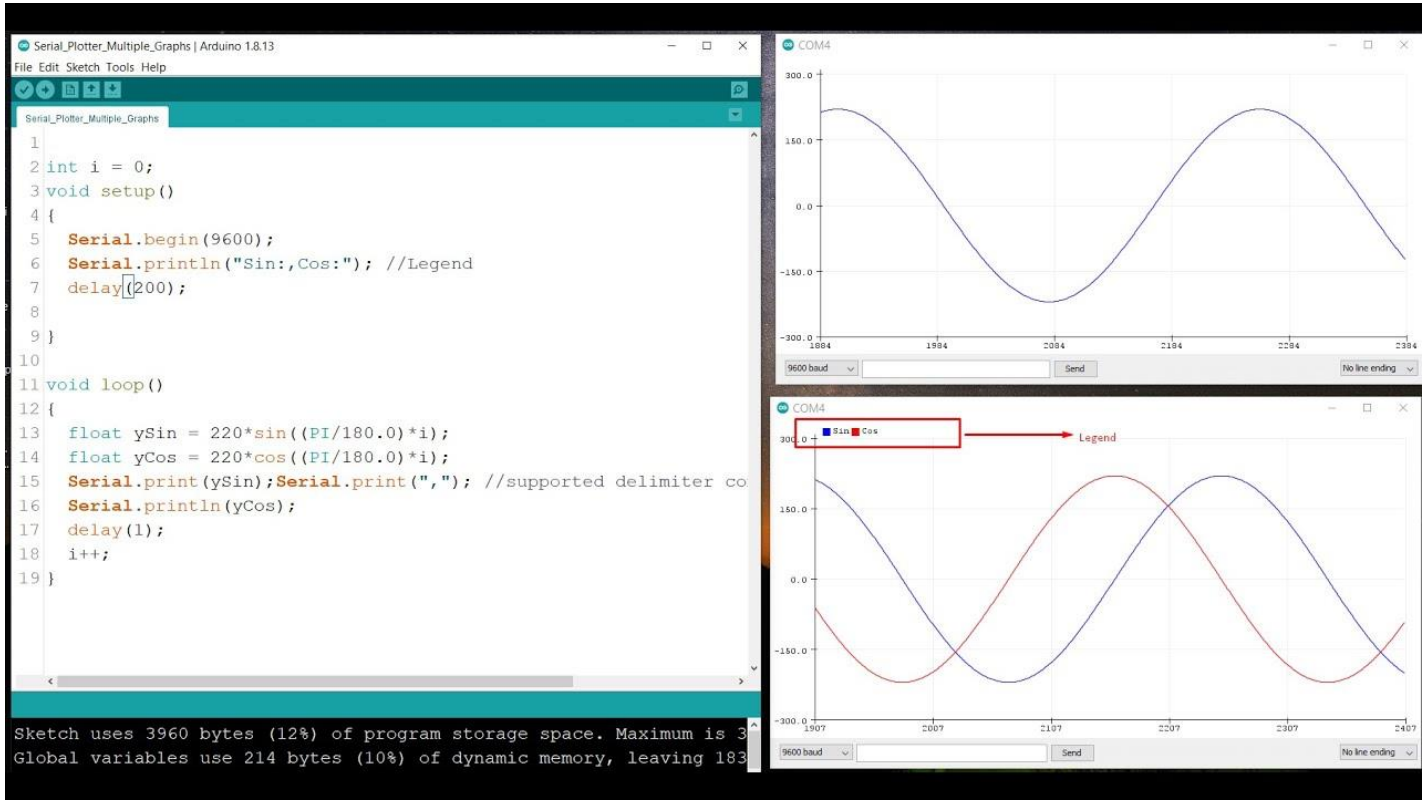
Yes



How to de-bug!

- Remember, problems can be a mixture of electronics/software, or both!

- Check electronics connections (pin numbers), good contact on breadboard
- Before using power supply – check you have the GND and Power the correct way around
- Use the Serial plotter on Arduino to check how the sensor is working
- Build up complexity in software (unit test)

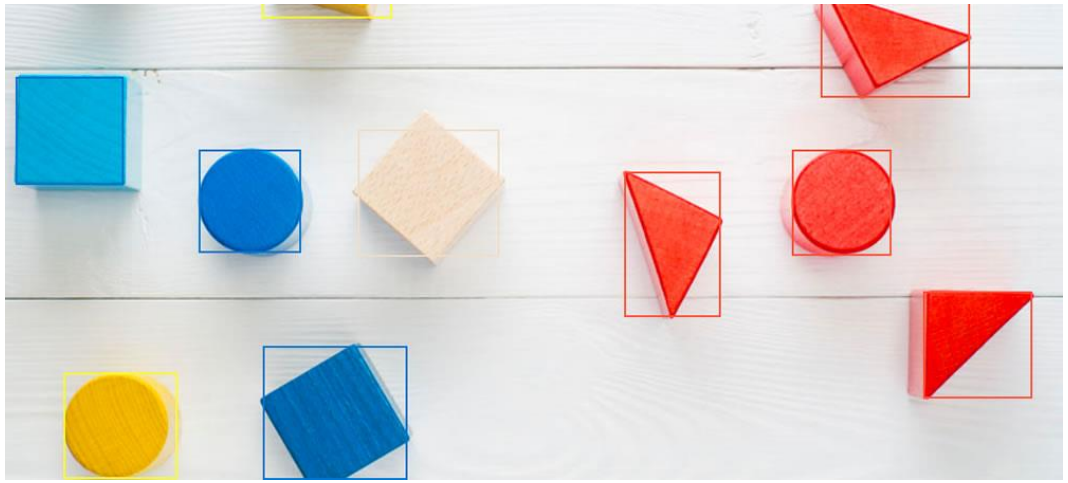


Webcams

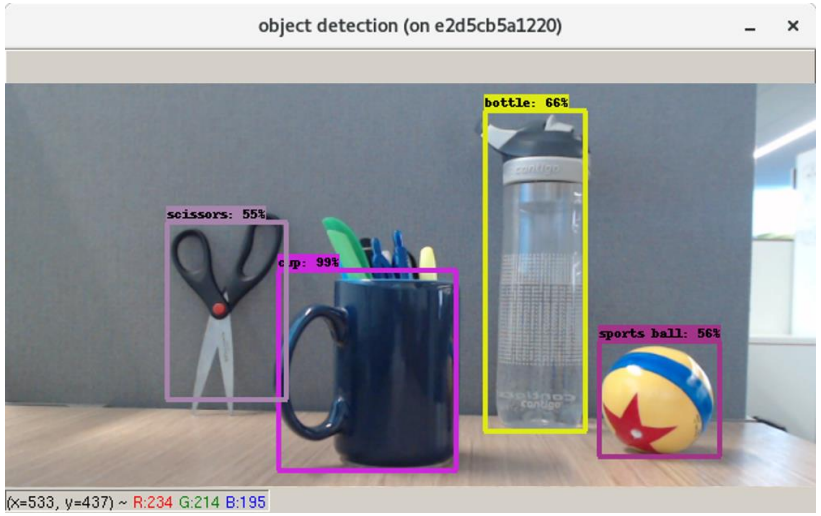
- Very powerful (high information content)
- Low cost, high useage
- Can be used ‘classically’ e.g. detect color/shape
- Can be used with learning based approaches. Many Neural Networks are particularly suited for image data

Webcams

Threshold/identify colors/shapes



Learning based object detection & classification



e.g. YoLo,

Although visual data is very useful for humans, it is challenging for machines

- Many different object orientations
- Variability in background light/conditions

→ Either need large training data-sets
→ Or need robust classifiers decision making algorithms

Major CAD Software

- **AutoCAD (Autodesk) → mainly for PC**
- **Pro Engineer (PTC)**
- **SolidWorks (Dassault Systems)**
- **CATIA (IBM/Dassault Systems)**
- **Unigraphics (UGS)**
- **I-DEAS (SDRC)**

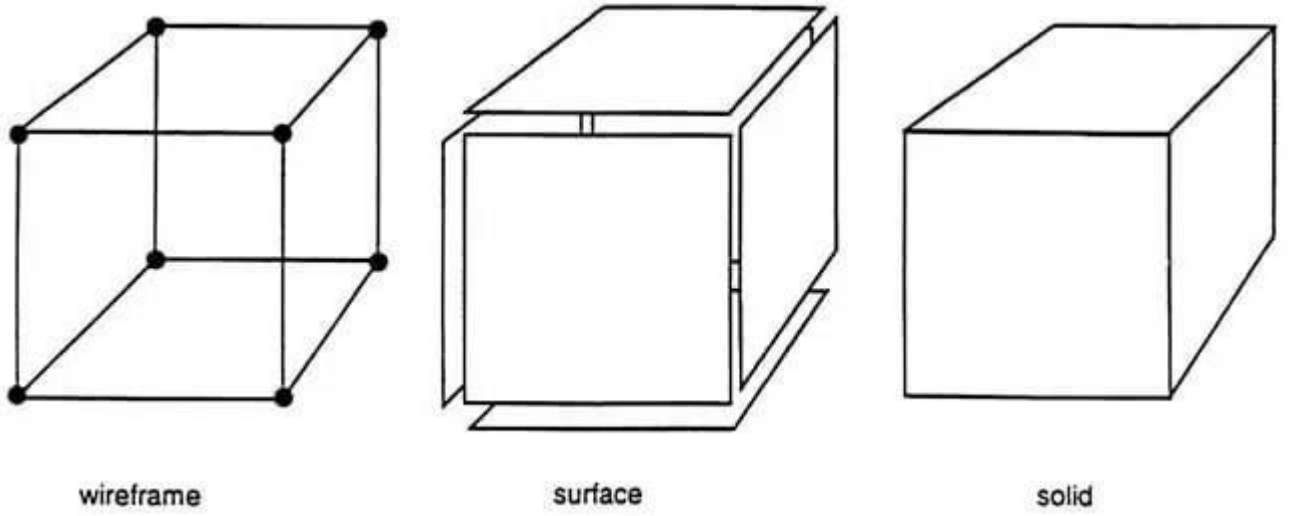
- Various disadvantages
- Expensive

- Catia → EPFL licence
- SolidWorks → On some EPFL PCs
- Autodesk Fusion 360 → Free student edition

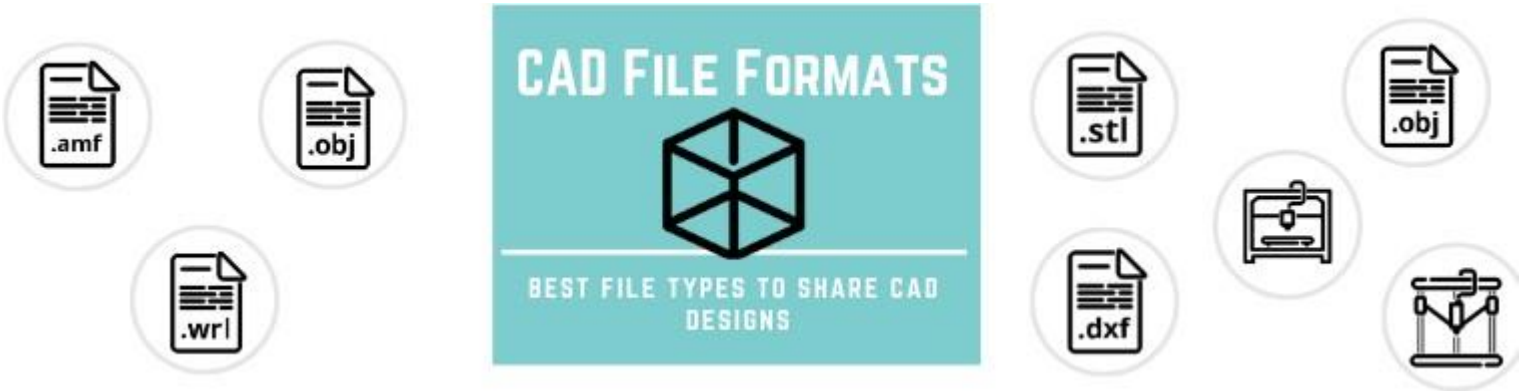
Many companies have their preferred CAD software

3D object geometric modelling

The three principal classifications:



What is the output from CAD?



Different features

- Proprietary or neutral (interoperable)
- Precise or tessellated
- Type of Assembly (multi-part) or one single file
- Part listing (BOM/flat list)

Top File Formats

STEP (.step files)

- Most widely used – maps to ISO standard (for product manufacturing info)
- Interoperable
- 3D CAD drawings

IGES (.iges files) Initial Graphics Exchange Specification

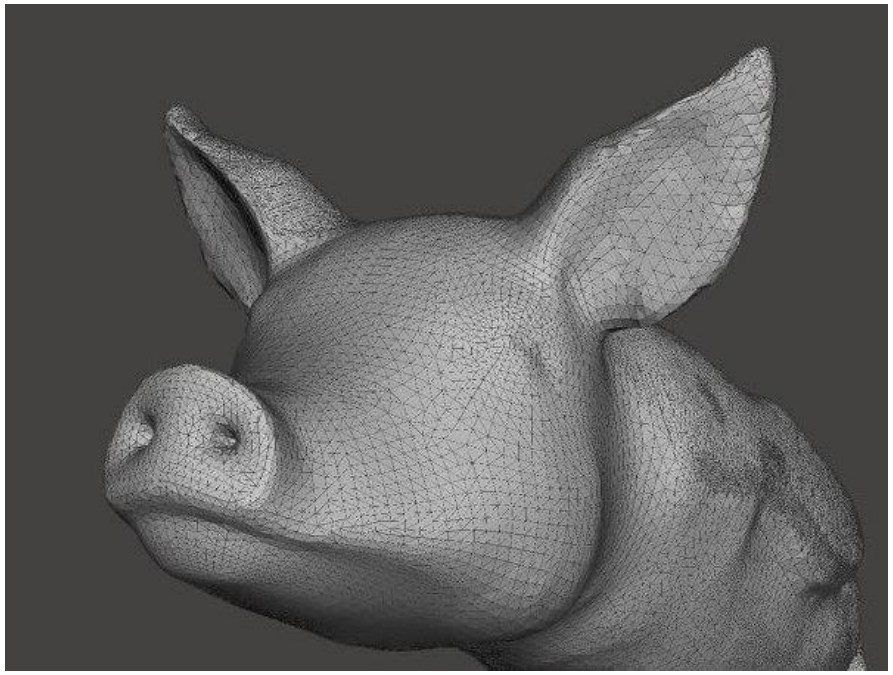
- File format specifically for wire frames/circuit diagrams
- Used for design work, not sharing

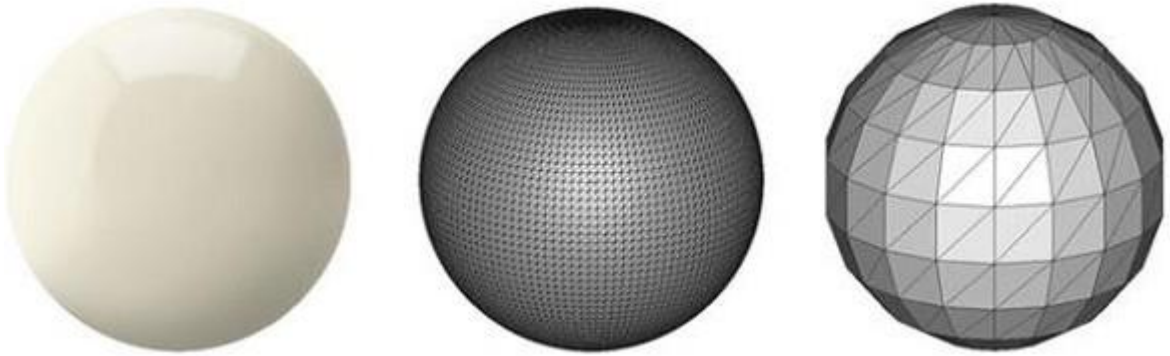
- **STL (.stl files) Standard Triangle Library**
  - STL is a universal format
  - Format for pure 3D information created by 3D programs
  - Stores surface geometry and shape, cannot
  - represent colour or texture
  - Good for simpler scenarios (and 3D printing)

- **DXF (.dxf files) Initial Graphics Exchange**
- **Specification**
- 2D file format – used by AutoCad
- Although proprietary used by many
- **DWG file format**

STL File Format

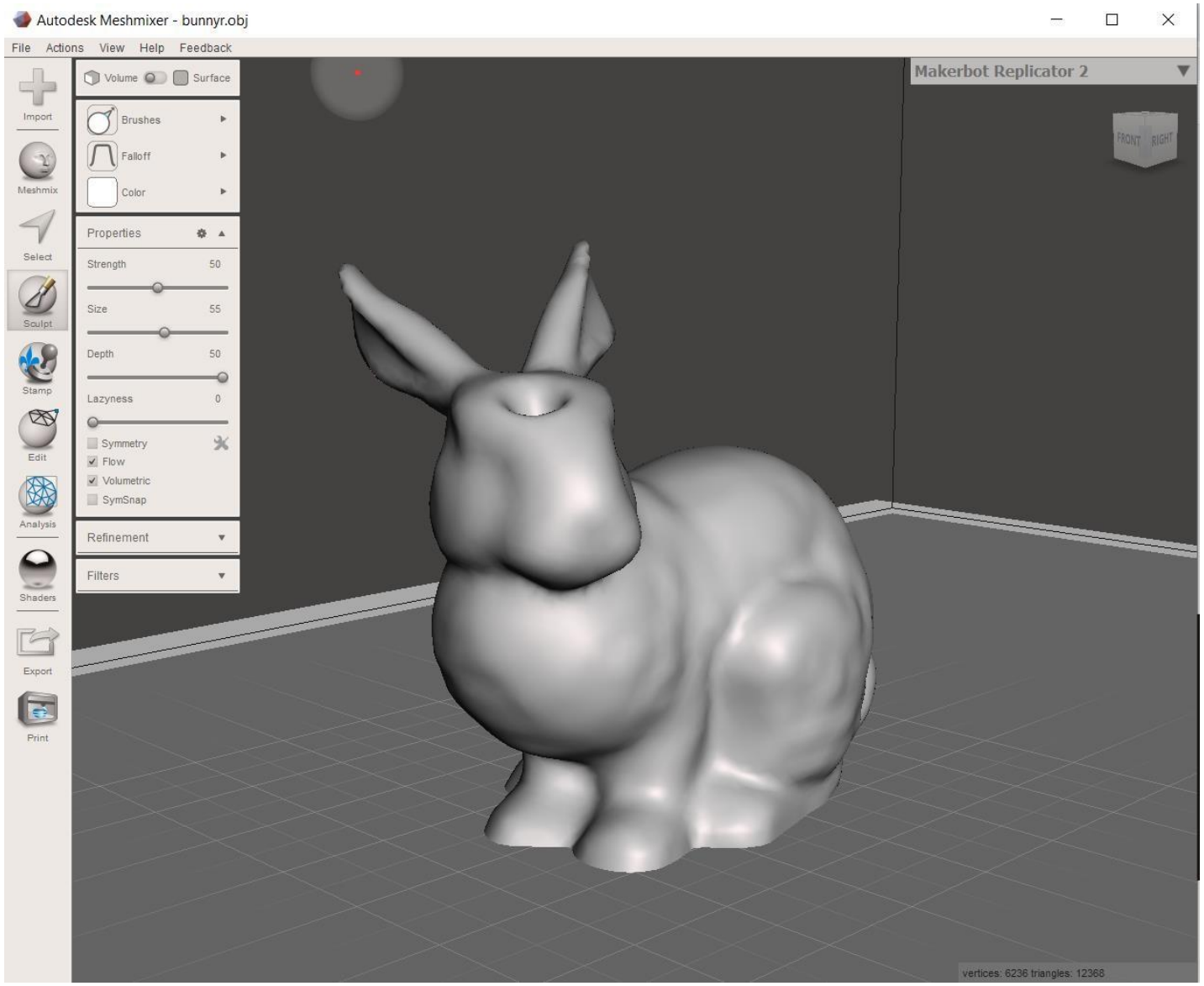
The main puípose of the SI'L file format is to encode the suíface geometíy of a 3D object. It encodes this infoímation using a simple concept called "tessellation".





- By making the triangles smaller and smaller the approximation can be made betteí and better, resulting in good quality píints.
- However, as you decrease the size of the triangle, the number of triangles needed to cover the suíface also increases increasing the file size.

Mesh Mixer

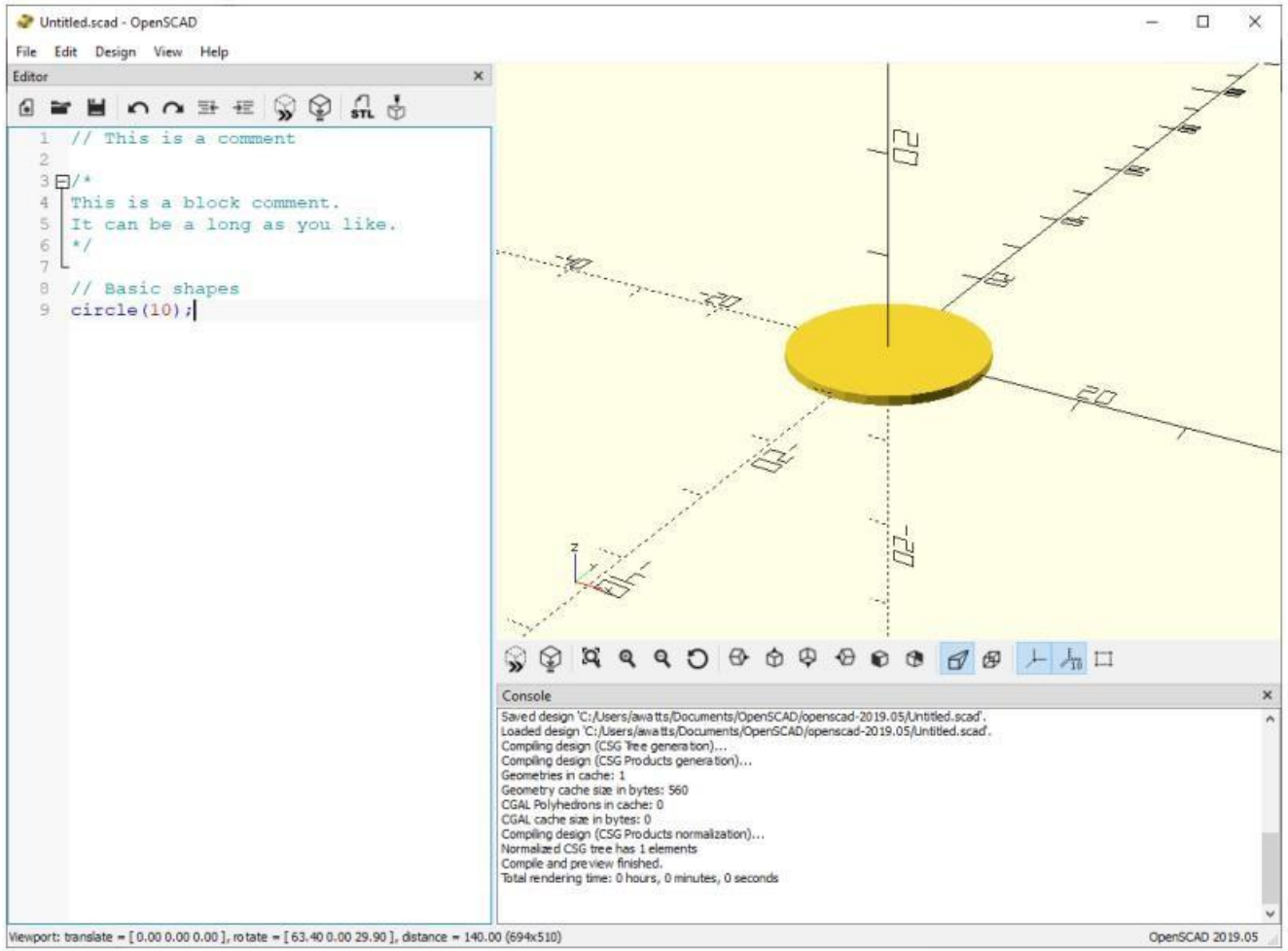


https://www.meshmixer.com/

Free – equivalent tools exist for other CAD packages

- Drag-and-Drop Mesh Mixing
- 3D Sculpting and Surface Stamping
- Robust Convert-to-Solid for 3D printing
- Hole Filling/fixing/editing
- Splitting bodies/merging bodies

Open SCAD



- Elements are created like calling functions in C: don't forget the semicolon!
- 2D shapes have fixed nonzero height, which I don't find useful. Instead of circle (shown above), I'd use a cylinder so I can specify the height as well.

Prototyping

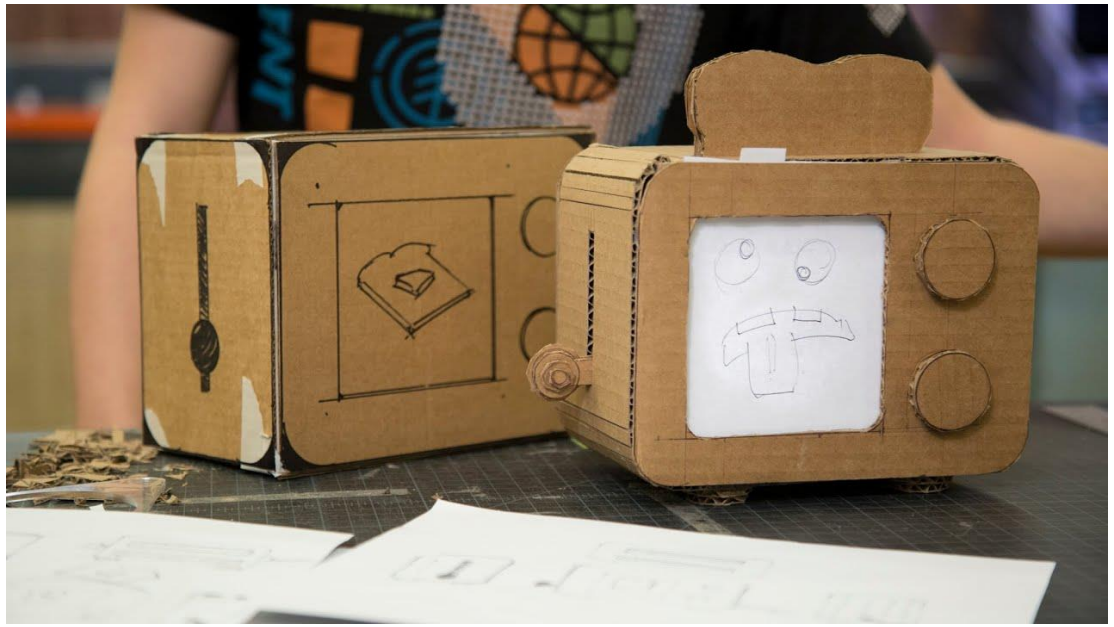
- Great hardware products don’t happen by chance.
- They are the result of a rigorous product development process that starts with concept development and moves through multiple cycles of design, engineering, and validation before the product enters full production.

Why do we use prototypes?

Why do we use prototypes?

A **prototype** is a preliminary model or mock-up of a product.

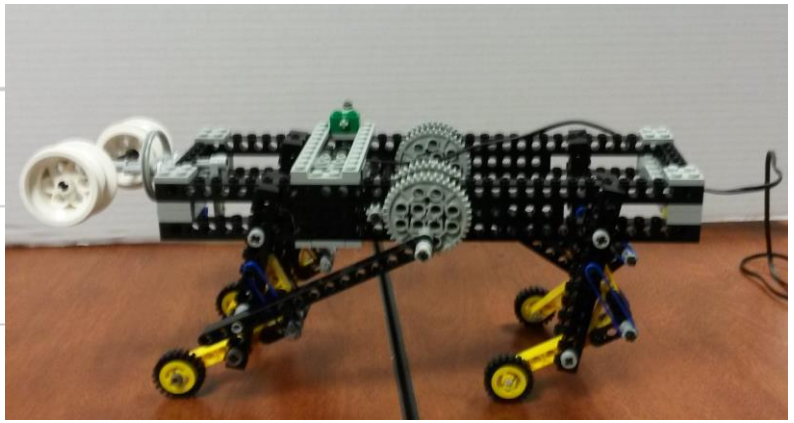
Some prototypes are created to get a better idea of what the final look or feel of a product might be, while others help to prove the functionality of a design.



- A company doesn't want to go through all the time, money, and effort to create a product and ship it, only to find out it fails to meet user needs or functional requirements.
- Design changes become increasingly costly as a product moves further along in the development process. Prototypes reduce the need for costly, late-stage design changes to the product.

Prototyping Tools and Methods for Hardware and Product Design

Basic Model Making



Cost	$
Production or Lead time	A few hours
Accuracy	★☆☆☆☆
Materials	Clay, cardboard, LEGO, foam
Tools required	Basic cutting tools, adhesives
Pros	Affordable Quick turnaround time Materials and tools are easy to access and work with Time and cost to revise a prototype or produce multiples is very low
Cons	Difficult or impossible to test meaningful functional aspects of the design Limited in ability to create complex shapes, parts, and assemblies Higher skill required to make presentation-quality models Low precision

Prototyping Tools and Methods for Hardware and Product Design

Fabrication

Cost	$$
Production or Lead time	A few hours to days
Accuracy	★★★☆☆
Materials	80/20 aluminum extrusion, sheet metal, plastics, wood, mechanical fasteners
Tools required	Variety of power tools for cutting, forming, welding, and assembly
Pros	Affordable materials Quick turnaround time Tools and materials are moderately easy to access and source Relatively easy to revise prototypes
Cons	May be difficult to produce small features and more complex shapes like splines Requires a wide selection of tools Requires moderate to high skill level to produce quality prototypes

Prototyping Tools and Methods for Hardware and Product Design

3D printing

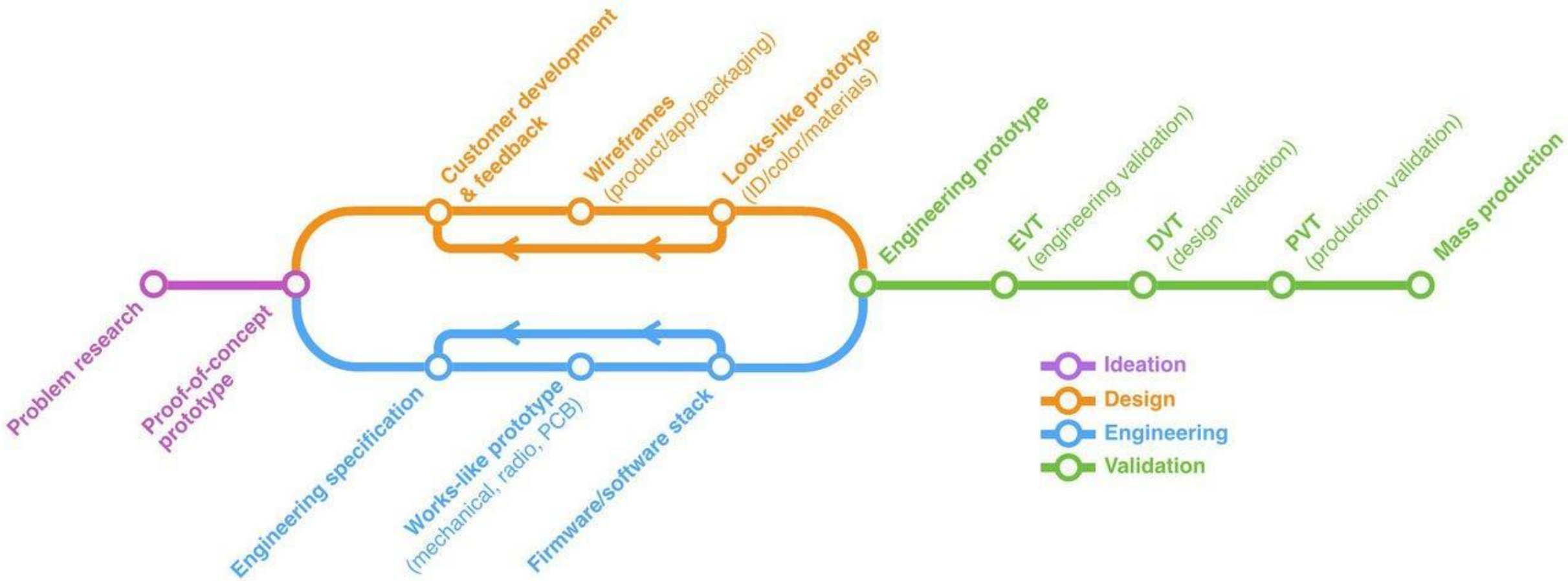
Cost	$$$
Production or Lead time	Less than 24 hours
Accuracy	★★★★★
Materials	Plastics, metals
Tools required	3D printer and finishing tools
Pros	Affordable (if using plastic) Fast turnaround time Easy to operate in-house, requires no specialized training or skills Easy integration with CAD software True form, fit and function can be tested Complex shapes and assemblies can be created
Cons	Part size restrictions based on the type of 3D printer Fewer material options available than for other processes (like injection molding) Post-processing can be labor-intensive depending on the process and part geometry Metal 3D printing is typically too costly for in-house use

Prototyping Tools and Methods for Hardware and Product Design

Machining

Cost	$$$$
Production or Lead time	A few days to weeks
Accuracy	★★★★★
Materials	Metals, plastics, composites
Tools required	CNC or manual machinery, CAD to CAM software
Pros	Very high precision with repeatable results Works with a wide assortment of metals, plastics, and composites Complex shapes and assemblies can be created
Cons	Expensive Machining in-house requires high investment in machinery, dedicated space, skilled labor to operate Outsourcing machining adds to lead time, slows design cycles Design restrictions mean certain geometries are very expensive—or completely impossible—to produce

Prototyping Tools for Different Prototype Stages



Proof-of-Concept (PoC) Prototype

- Proof-of-concept prototypes aim to prove that an idea is feasible and that there is a potential market for it.
- PoC prototyping happens at the earliest stages of the product development process, and these prototypes include the minimum functionality needed to validate assumptions before moving the product into subsequent stages of development.

Recommended prototyping tools:

•Basic model making
•Fabrication
•3D printing

Looks Like prototypes

- Looks-like prototypes represent the final product at an abstract level but may lack many of its functional aspects.
- Give a better idea of what an end product for end user testing
- Ergonomics, user interfaces, and overall user experience can be validated
- Looks-like prototype development usually starts with sketches, foam or clay models, then moves into CAD modeling.
- Iterative process

Recommended prototyping tools:

•Basic model making
•Fabrication
•3D printing

Functional or works like prototype

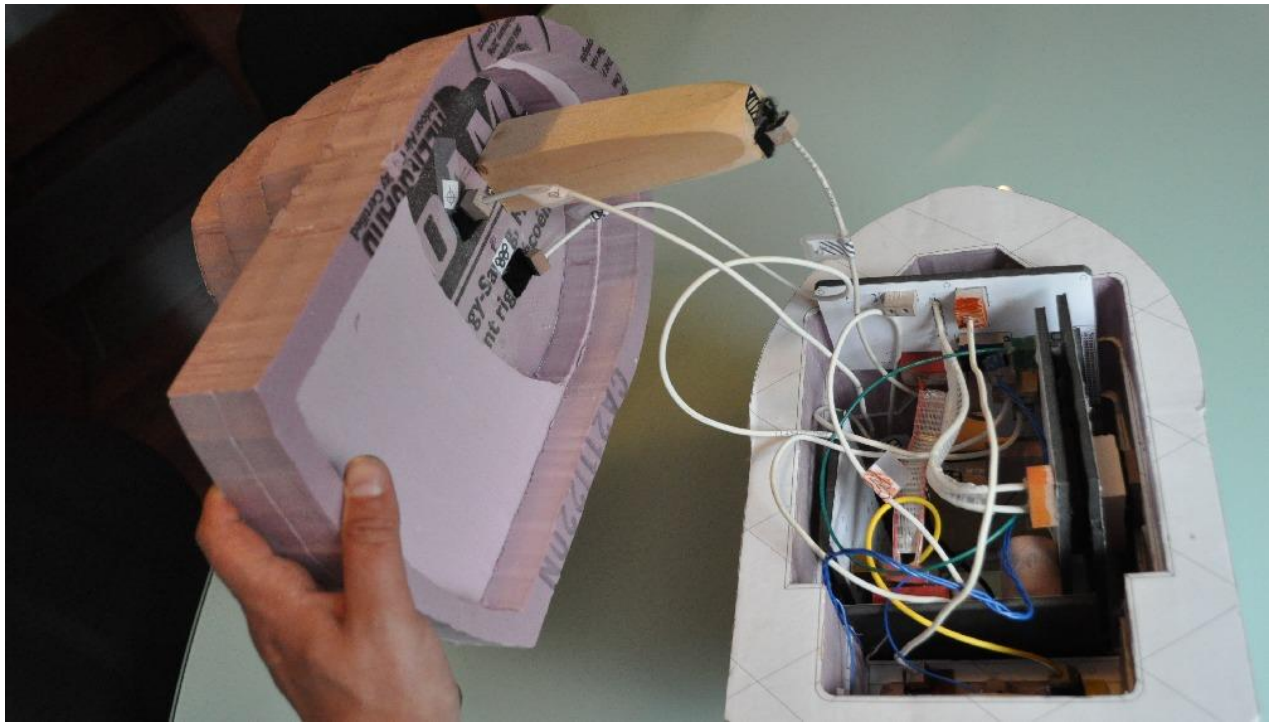
- Parallel to the industrial design process, engineering teams work on another set of prototypes to refine the engineering.
- Might look different but they include the core technologies and functions that need to be developed and tested.
- Often, these critical core functions are developed and tested in separate sub-units before being integrated into a single product prototype.
- This subsystem approach isolates variables, making it easier for teams to split up responsibilities and ensure reliability on a more granular level before folding all of the elements together.

Recommended prototyping tools:

•Fabrication
•3D printing
•Machining

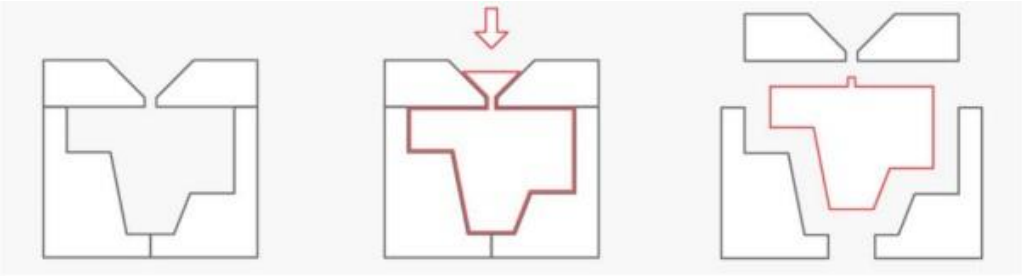
Engineering prototype

- Engineering prototype (EP) is where design and engineering prototypes meet, which often requires concessions from both sides.
- EP builds are usually the last prototypes built in-house before validation builds begin at the manufacturer.
- These prototypes should be made using the final materials, parts, and processes wherever possible, but without investing in costly tooling prematurely.
- For example, 3d printing might still be used in place of generating custom tooling/moulds

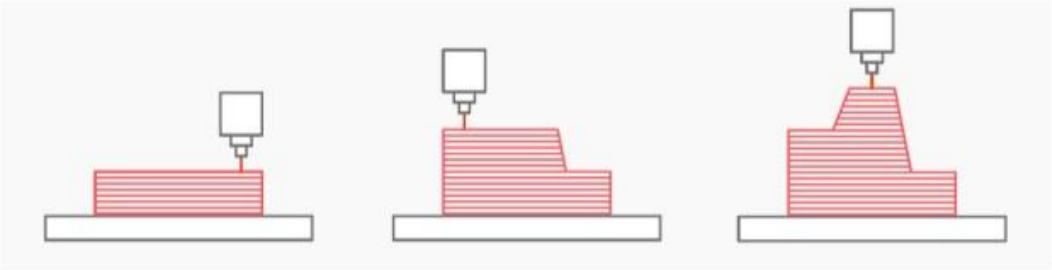


Recommended prototyping tools:
•3D printing
•Machining

Fabrication Methods



Formative manufacturing: best suited for high-volume production of the same part, requiring a large initial investment in tooling (molds) but then being able to produce parts quickly and at a very low unit price.

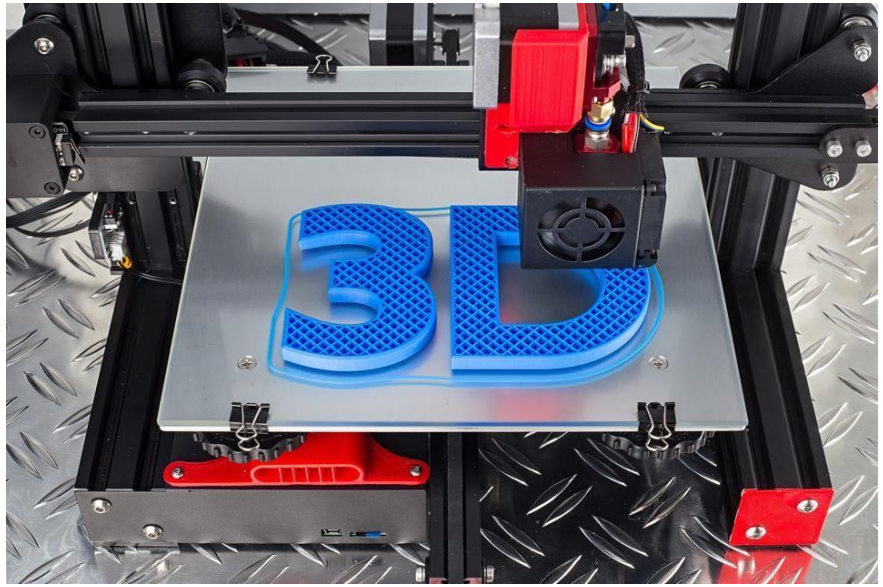


Additive manufacturing: best suited for low-volume, complex designs that other methods are unable to produce or when a unique, one-off rapid prototype is required.

Subtractive manufacturing: lies in between formative and additive, being best suited for parts with relatively simple geometries, produced at low-to-mid volumes, and where materials like wood or metal are necessary

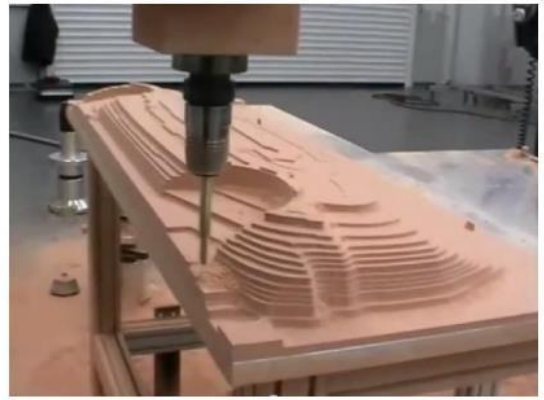
Fabrication Methods

Additive Manufacturing
= 3D printing



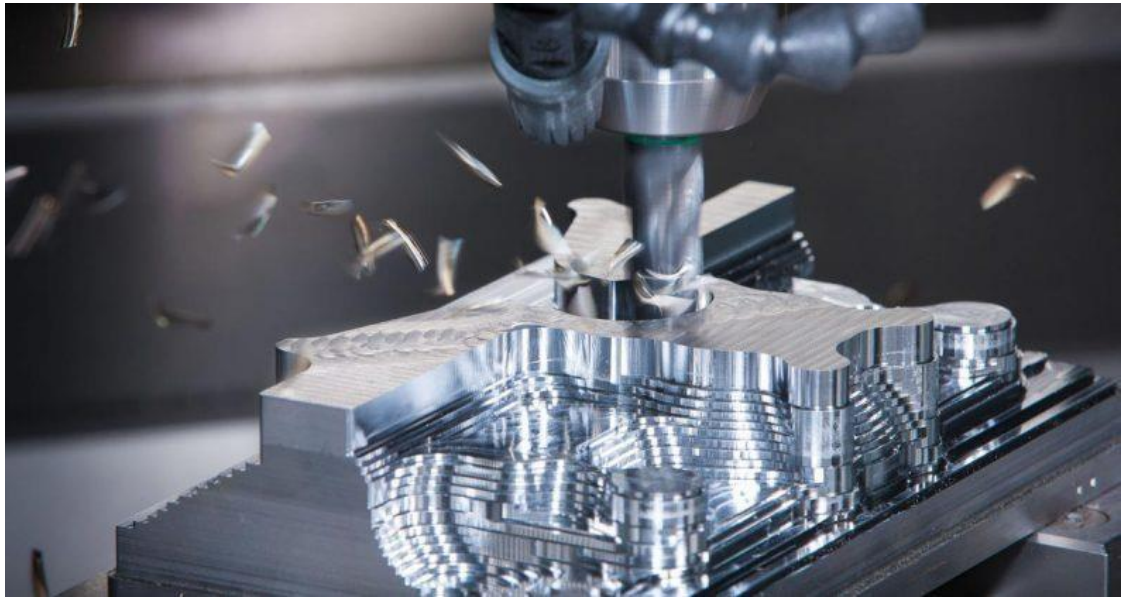
Subtractive Manufacturing
= CNC Milling, Laser Cutting

- Start with a block of material
- Remove material to obtain a given 3D shape

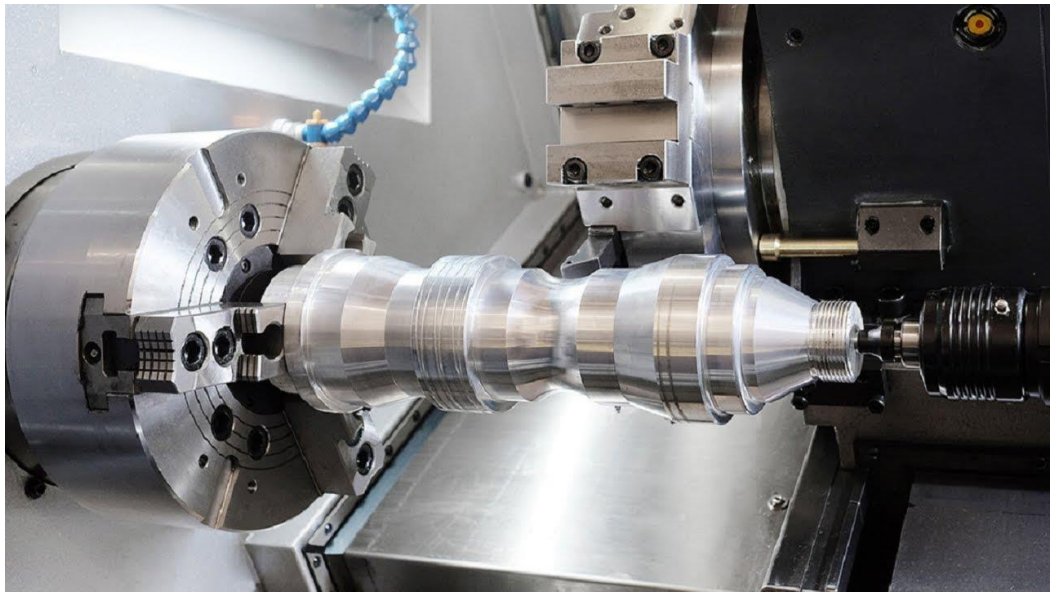


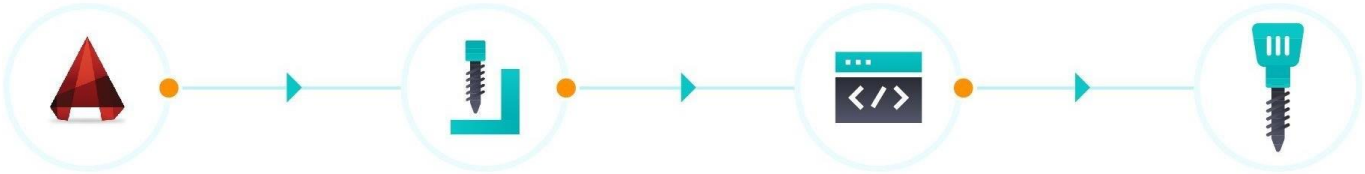
CNC (Computer Numerical Control)

CNC Mill



CNC Lathe





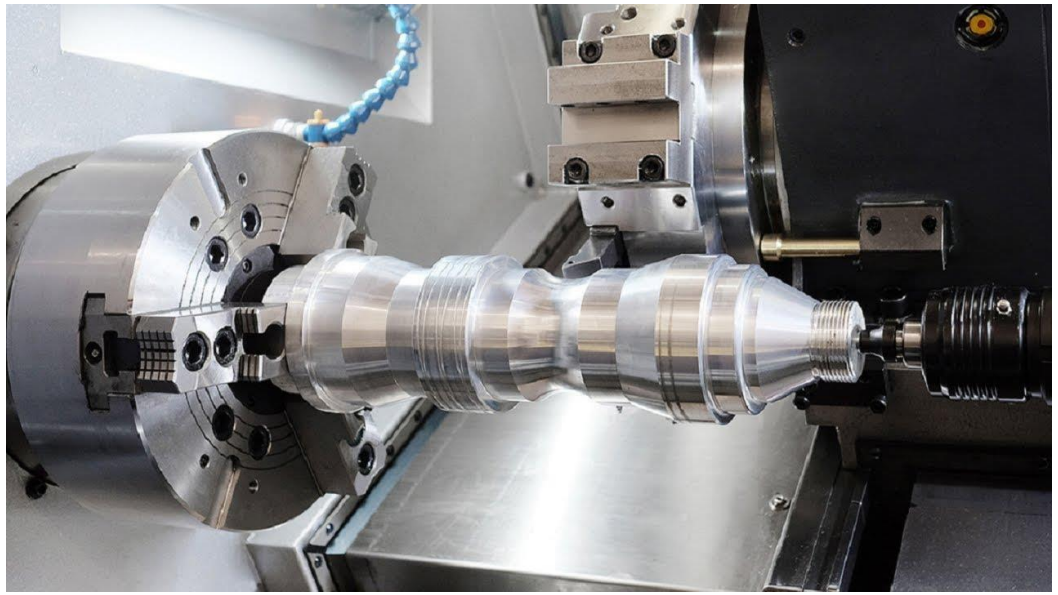
Designing the CAD model

Preparing the CNC machine

Converting the CAD file to a CNC program

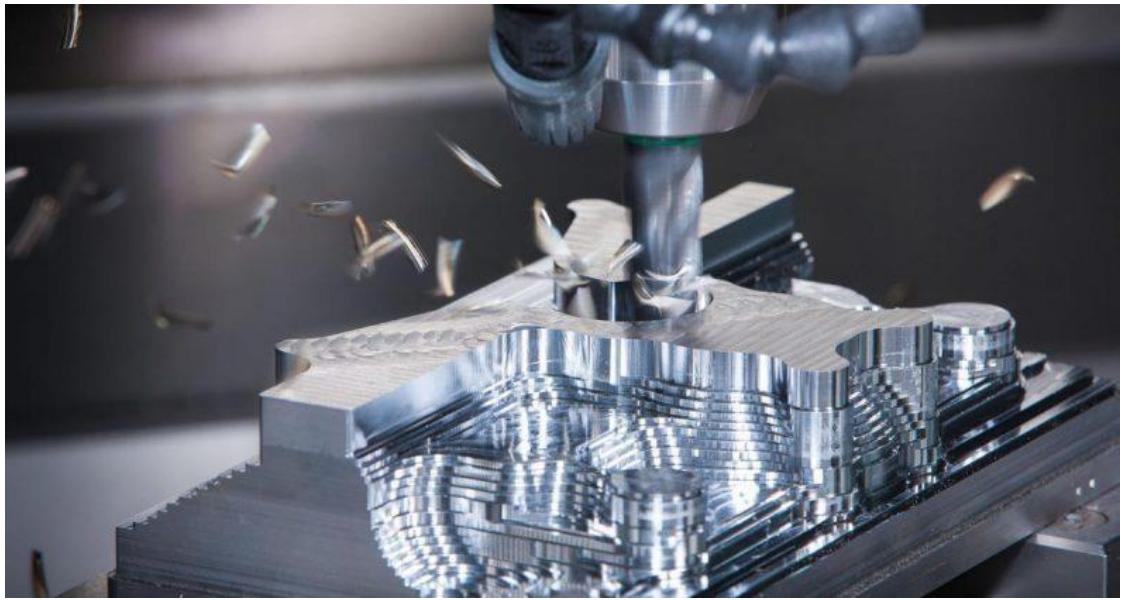
Executing the machining operation

CNC Lathe



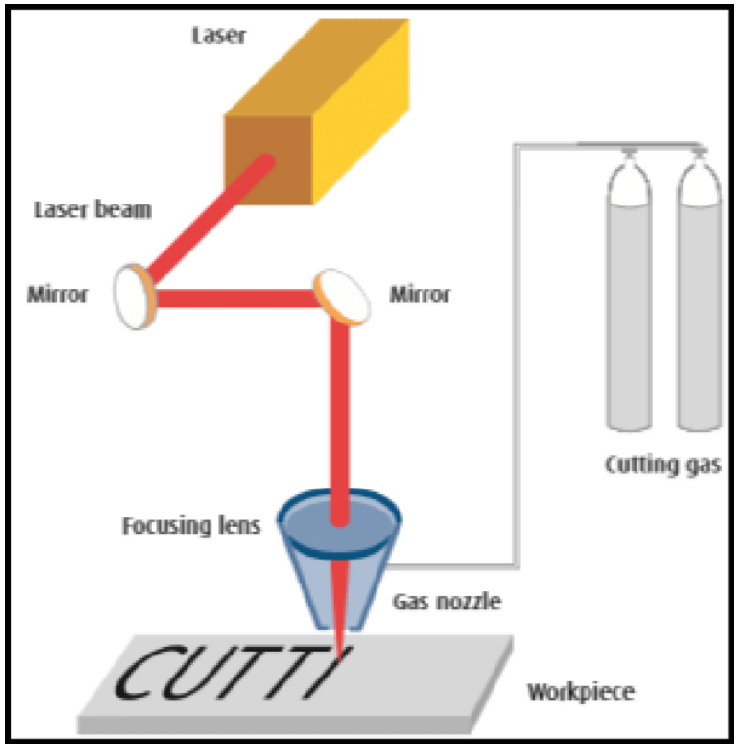
https://www.youtube.com/shorts/wbcmR97DxHs

CNC Mill



https://www.youtube.com/watch?v=AxHexqN0Hr0

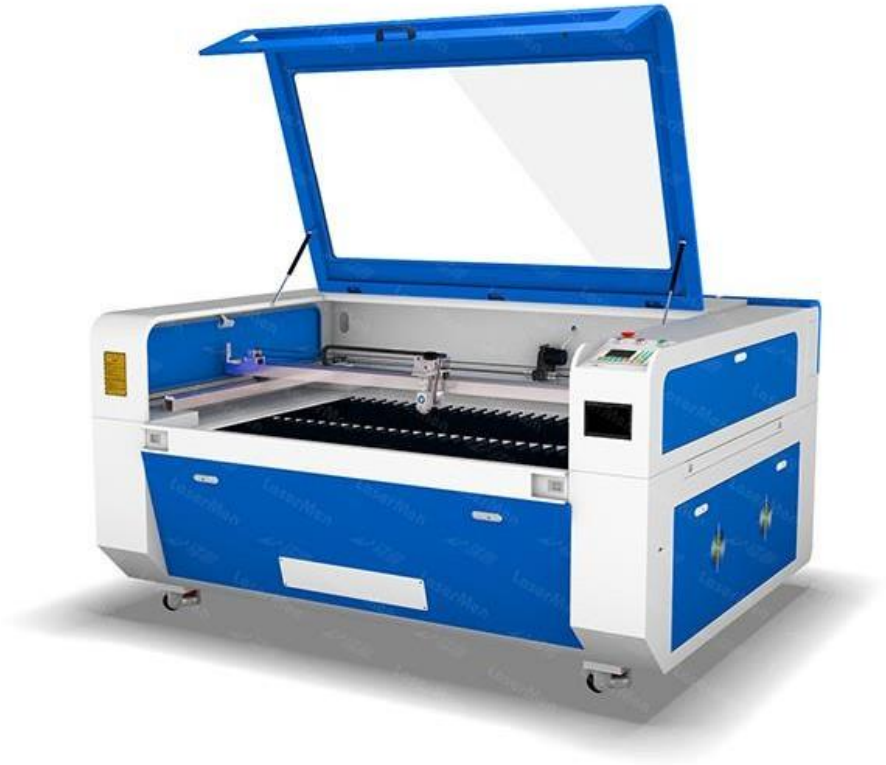
Laser Cutting

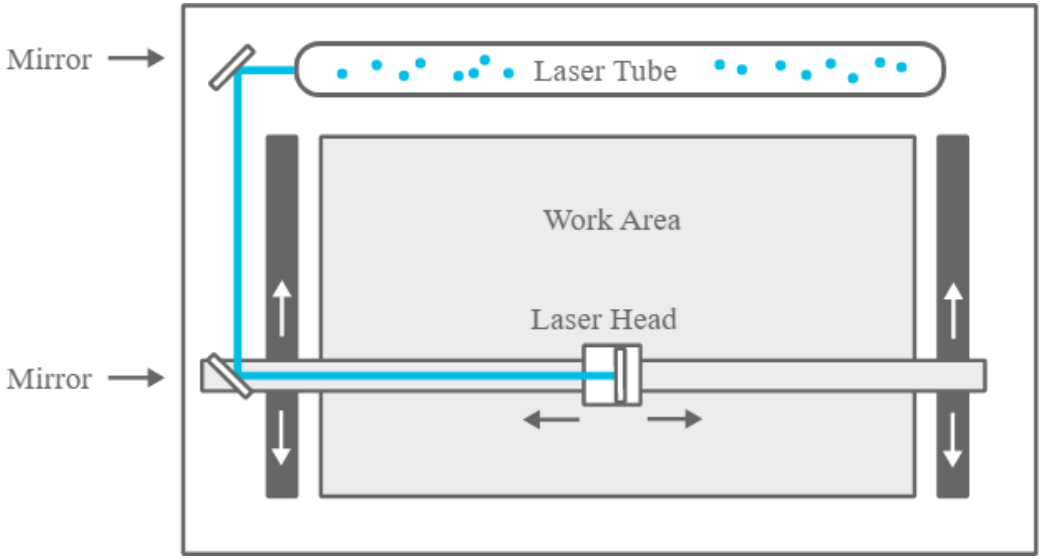


Different types of laser source:

- CO2
- Solid state (Nd:YAG)

Laser Cutting

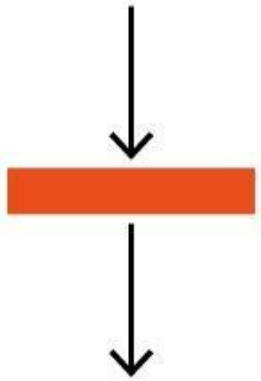




Laser Cutting: Materials

TRANSMISSION

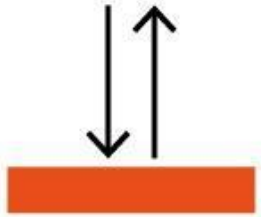
Laser beam goes **through** the material, (e.g. glass)



no effect on the material

REFLEXION

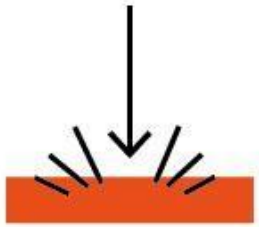
Laser beam is **reflected** by the material (e.g. the front side of the mirror)



no effect on the material

ABSORPTION

Laser beam is **absorbed** by the material (e.g. wood, felt, etc.)



desired effect: material is being laser-cut or engraved

Typical materials

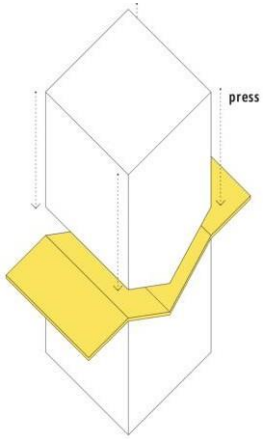
- Wood
- MDF
- Paper (as long as it doesn't catch fire)
- Acrylic
- Plastics (that don't melt/burn)
- Leather
- Ceramic
- Metal

Can't process:

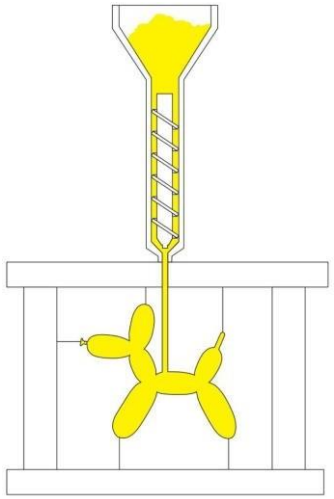
- Reflective materials (e.g. glass, fiberglass)
- Highly conductive
- Those that release harmful gases/melt (Polystyrene and Polypropylene ABS)

Different materials require different speeds and laser powers for cutting

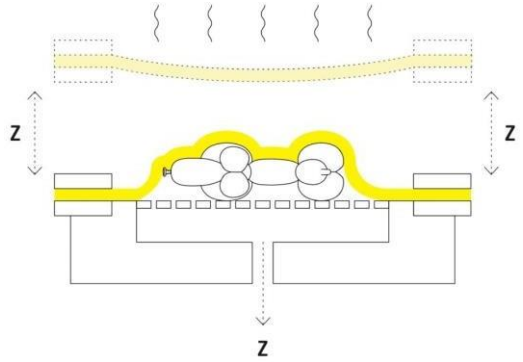
Formative Manufacturing



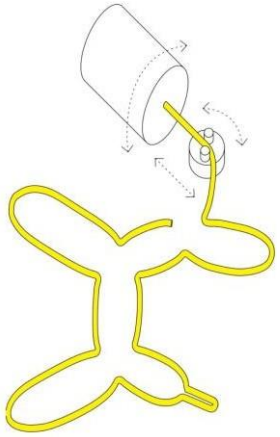
Press Moulding



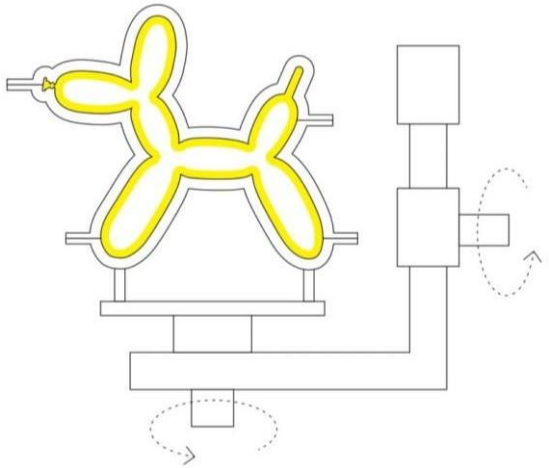
Injection Moulding



Vacuum Forming

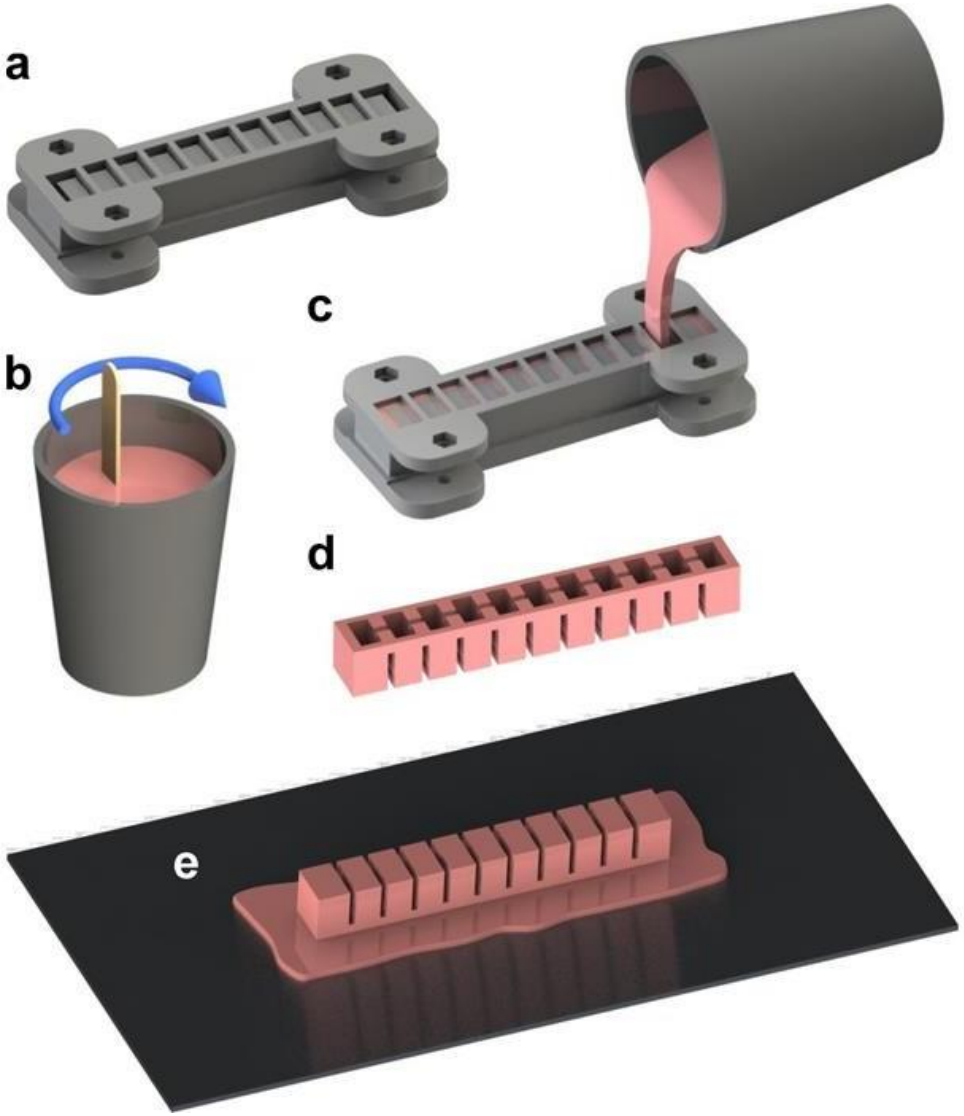


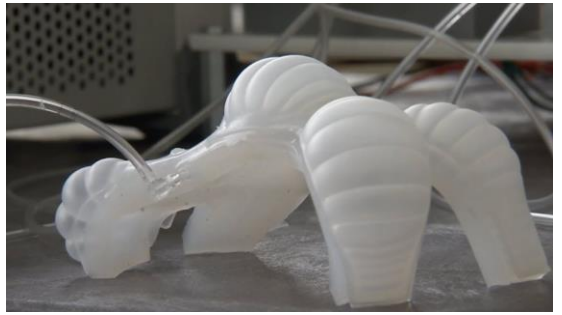
CNC Tube Bending

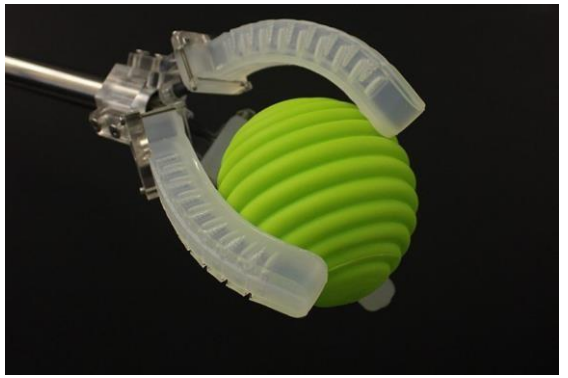


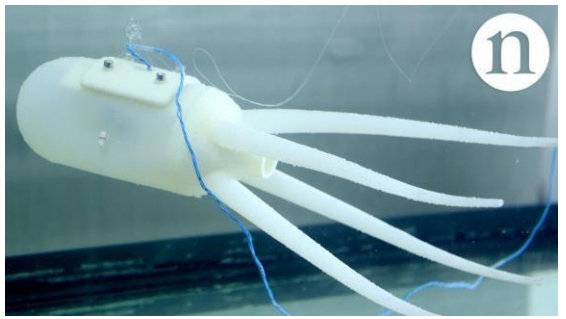
Rotoation Moulding

Formative Manufacturing









3D printing for products

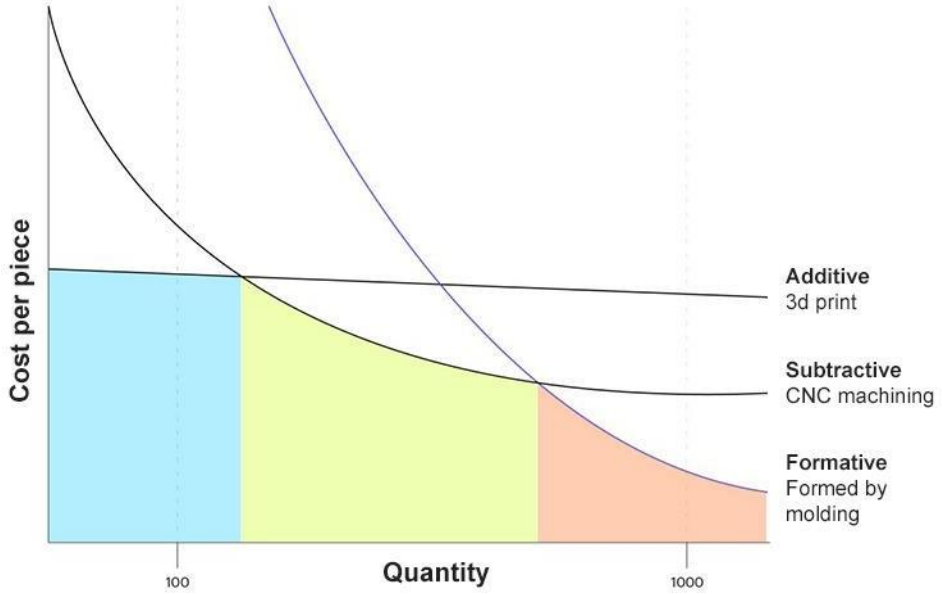
Custom Hearing Aids

Honeycomb
Tires

3D Printed Ventilation Prototype
(High Temperature 3D Printing Material)

Prosthetics

Scaling up…



- As the quantity increases the fabrication method may need to change

Fabrication Quiz

https://participant.turningtechnologies.eu/en/join

https://participant.turningtechnologies.eu

SessionID: datadriven

https://ttpoll.eu/p/datadriven