Product development & engineering design

ME-320

PROF. JOSIE HUGHES

Lecture 3: Microcontrollers & Control

Today: Introduction & Content

- Project management & Gantt Charts
- Mechanism Design
- Introduction to microcontrollers
  - Getting started Exercises!!

Next lecture: Design Review

Next next lecture: Sensors, actuator selection and controlling actuators

Next Week: Design Review

Design Review Sessions

Join in the SPOT!

Time	Table 1	Table 2	Table 3	Table 4	Table 5	Table 6	Table 7
8:20	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
8:40	Group 8	Group 9	Group 10	Group 11	Group 12	Group 13	Group 14
9:00	Group 15	Group 16	Group 17	Group 18	Group 19	Group 20	Group 21
9:20	Group 22	Group 23	Group 24	Group 25	Group 26	Group 27	Group 28
9:40	Group 29	Group 30	Group 31	Group 32	Group 33	Group 34	Group 35
10:00	Group 36	Group 37	Group 38	Group 39	Group 40	Group 41	Group 42
10:20	Group 43	Group 44	Group 45	Group 46	Group 47	Group 48	Group 49
10:40	Group 50	Group 51					

- Schedule can be found on moodle.
- We be held in the SPOT
- Please be on time!

Next Week: Design Review Submission

- Functionality Diagram
- Concept Ideation (5, or 4 if a team of 4, designs)
- Concept Selection
- Gantt Chart

Bring in person (e.g. a print out)

→ Also to be submitted on Moodle. One pdf per team

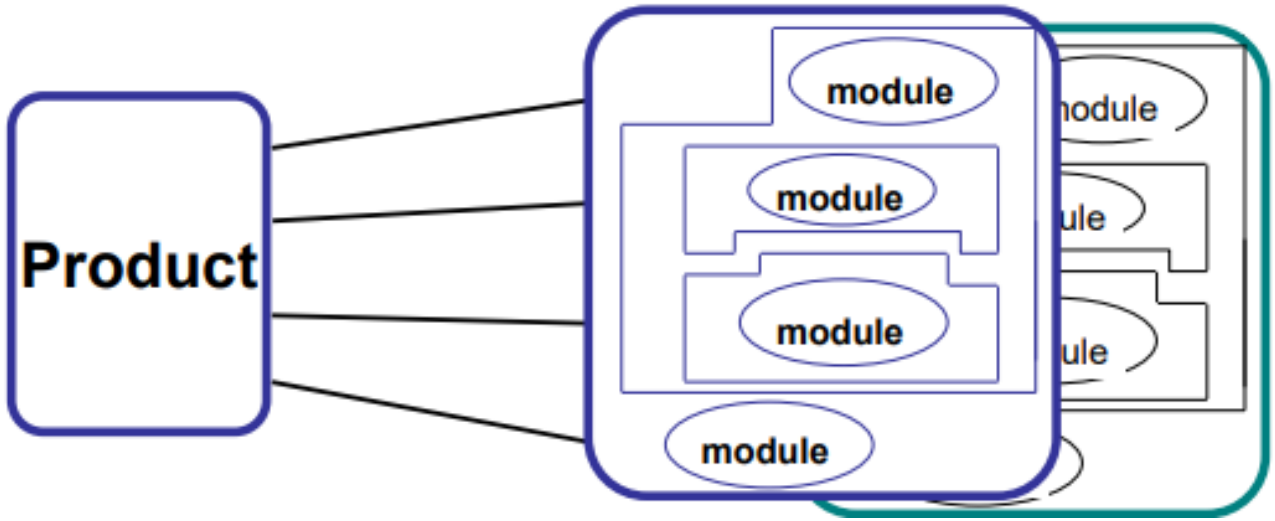
Project Management  Project Architecture

Product Architecture

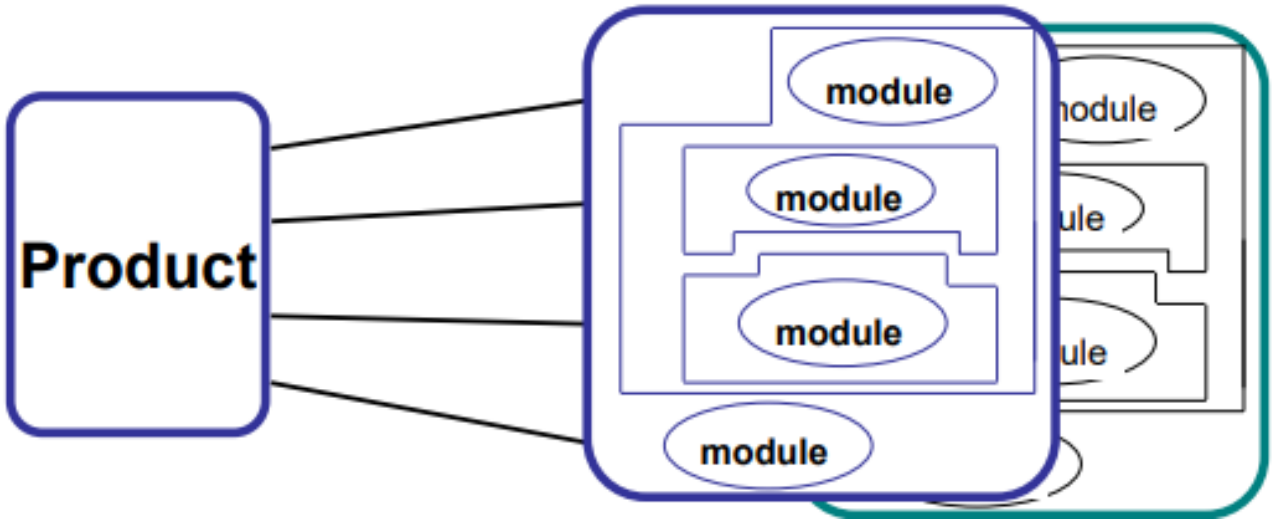
What is it?

Product architecture is the assignment of the product's functions to physical building blocks or "chunks“

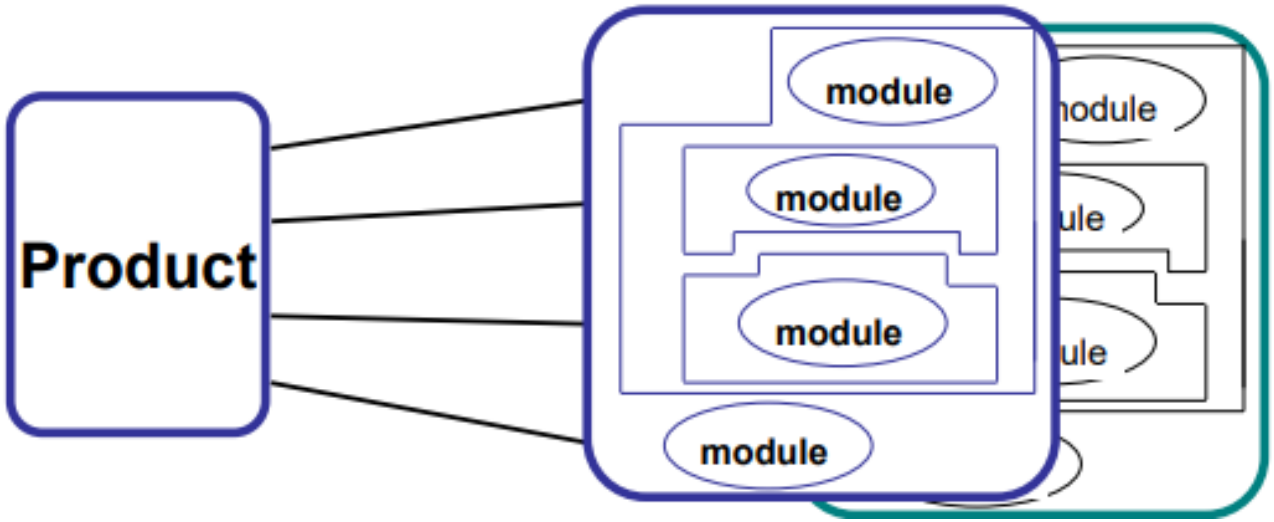
Physical elements are organized into several building blocks of chunks



What could it affect?



What could it affect?



- Could drive the initial design
- Manufacturing costs
- Product evolution
- Organization of design teams
- Repairability

Product Architecture

Integral or Modular

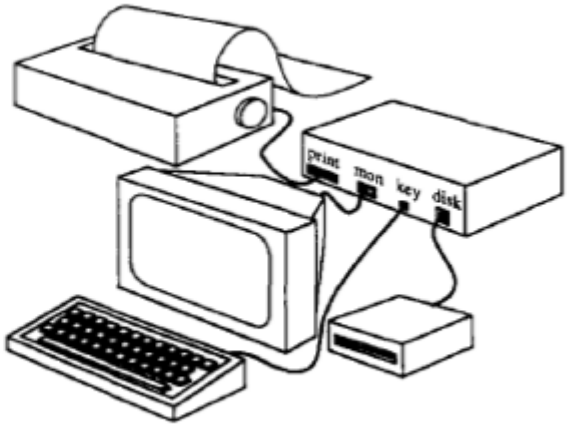
Example of a modular product?

Example of an integral product?

Product Architecture

Integral or Modular

Integral



Modular

- Video Games
  - Modular: gaming systems (e.g. GameCube)
  - Integrated: stand-alone arcade games
- Power Supplies
  - Modular: power bricks
  - Integrated: on-board power converter

Modular Architecture

- Each chunk fully embodies one or more product functions
- Interactions between chunks are well defined and fundamental to the primary function.
- Modular architecture has advantages in simplicity and reusability for a product family or platform.

Integral Architecture

- Typical functions involve more than one chunk
- Typical chunks implement more than one function
- Interactions between chunks are ill-defined and may be incidental to product's primary functions.
- **Integral architecture generally increases performance and reduces costs for any specific product model.**

Product Architecture

Integral or Modular

Integral

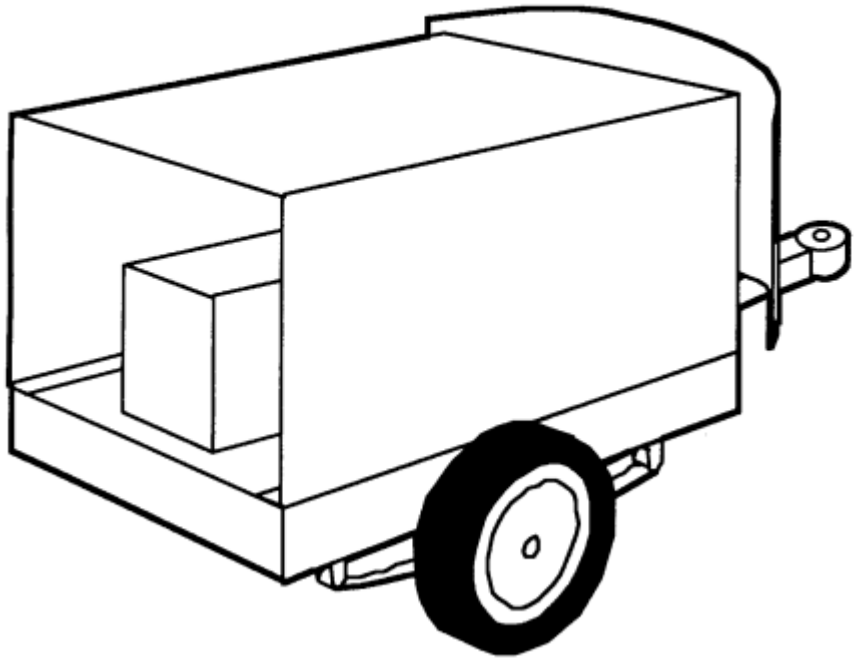
- Higher system performance
- Lower system cost (in large volume)
- Tightly coupled design teams
- Expensive Tooling
- Hard to change

Modular

- Changeability
- Decoupled design teams
- Reduced performance
- Requires flexible manufacturing
- Cheaper at low volumes

Integral or Modular?

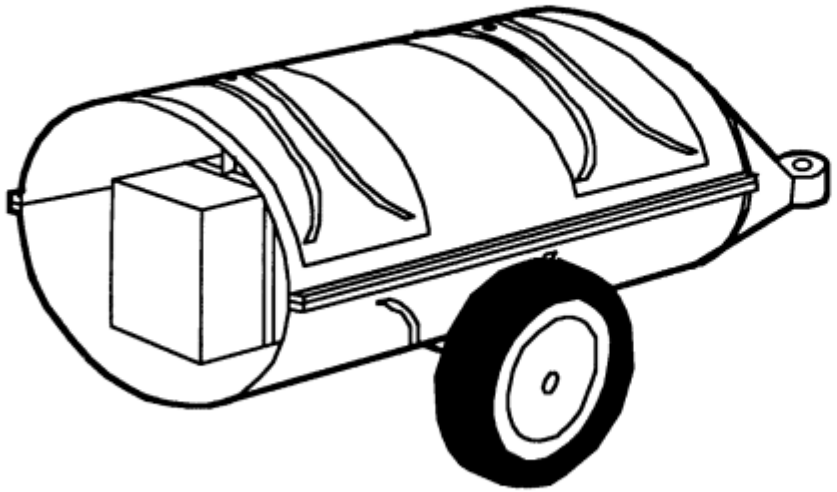
Modular Example of Trailer

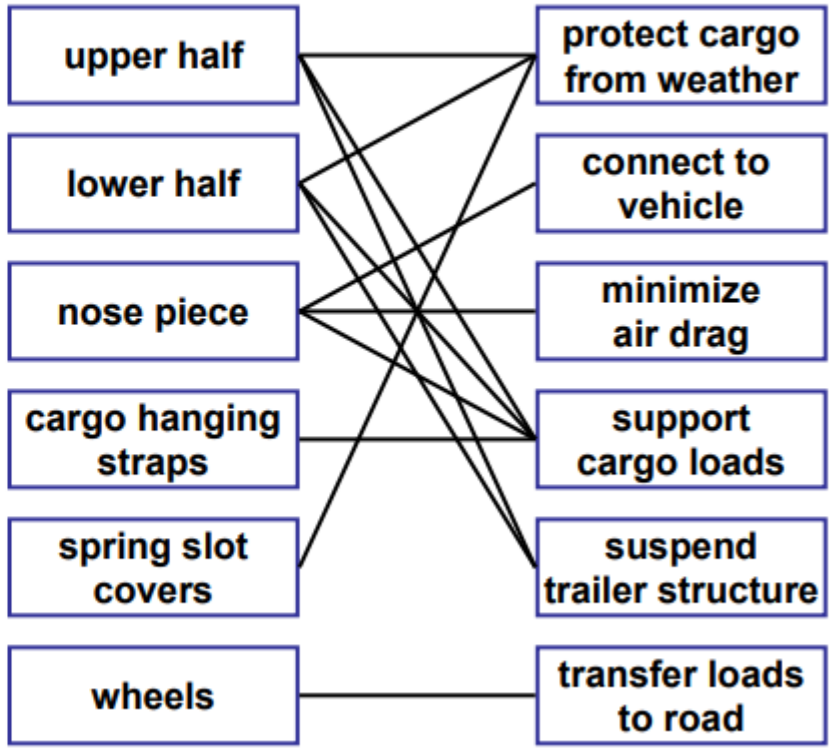


box	protect cargo from weather
hitch	connect to vehicle
fairing	minimize air drag
bed	support cargo loads
springs	suspend trailer structure
wheels	transfer loads to road

Component → Function 1:1 Mapping

Integral Example of a Trailer



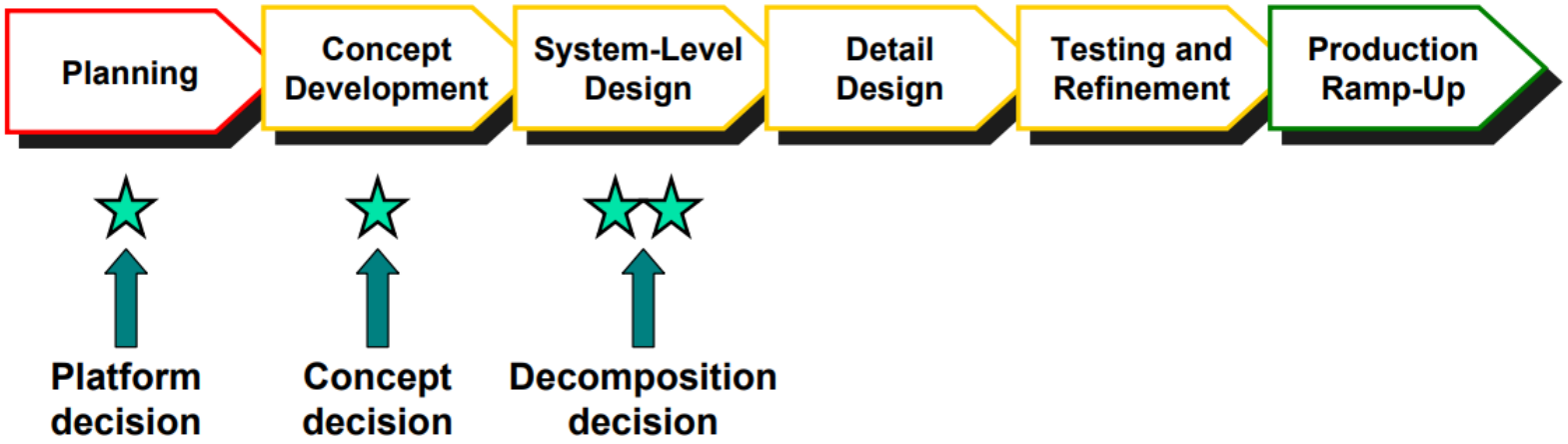


Component → Function 1:Many Mapping

Product Architecture

But, there is a reality…

- Modularity is a relative property
- Products are rarely strictly modular or integral.



Begins to emerge during concept development

Discussion

For your project….

What could be modular aspects of your robotic product?

What would be the advantage of having modular gripper design?

What would be advantages of having integral gripper design?

Selecting the product architecture?

Architecture decisions relate to product planning and concept development decisions:

- **Product Change** (copier toner, camera lenses)
  - Upgrades e.g. upgrade processor
  - Add ons
  - Adaptations to local conditions (110 or 220V PSU)
  - High-wear coponenets (e.g. tires)
  - Consumables (e.g. printer cartridges)
  - Reuse (consumer electronics)
- **Product Variety** (computers, automobiles)
  - Range of models a firm can produce in response to consumer demand
- **Standardization** (motors, bearings, fasteners)
  - Use standardized components/chunks (e.g. watch motion/battery)

Selecting the product architecture?

- **Performance** (racing bikes, fighter planes)
  - Function sharing between high-performance systems

- **Manufacturing Cost** (disk drives, razors)
  - Design for manufacturing
  - Minimization of parts

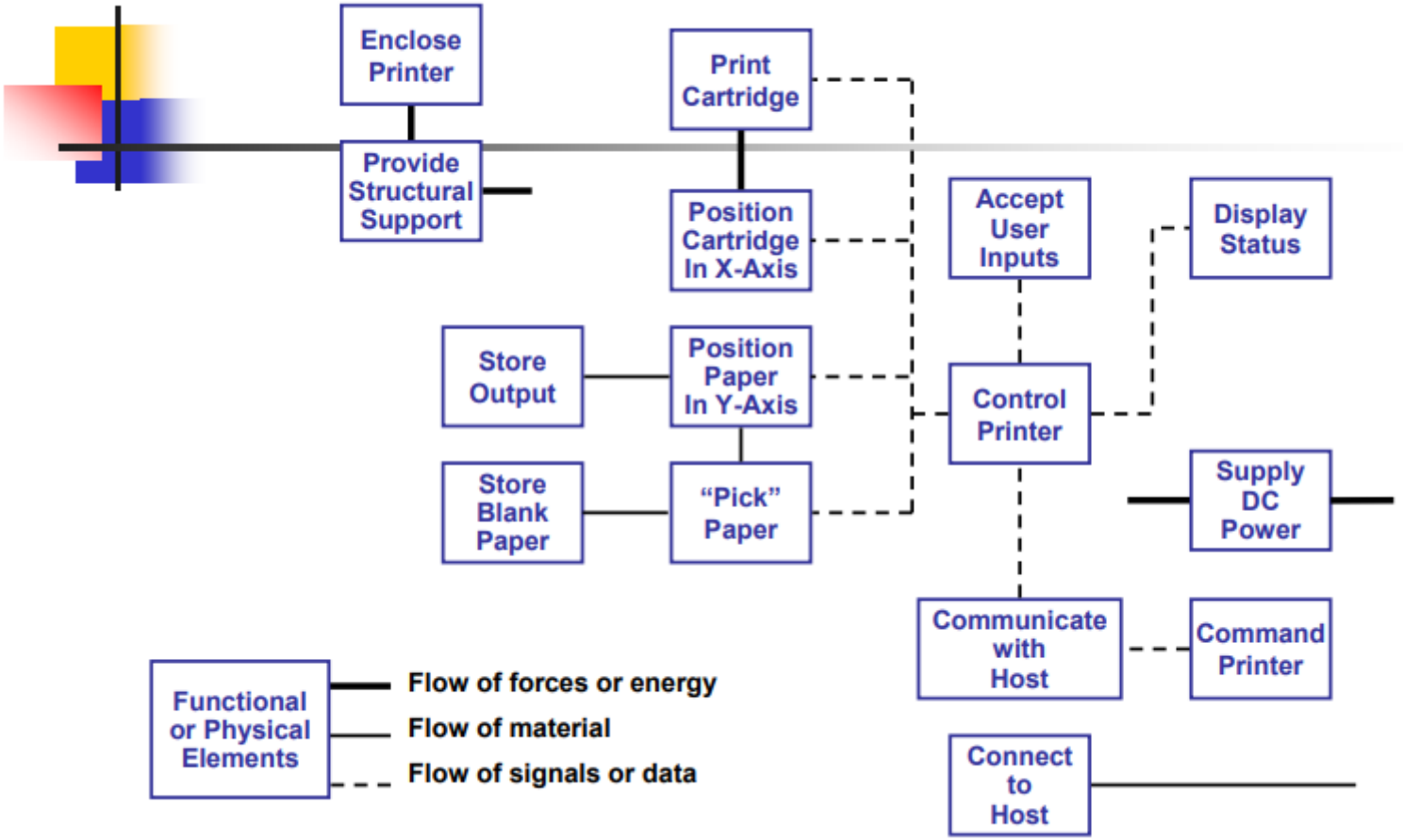
  →This can be best at the chunk level

- **Project Management** (team capacity, skills)
- System Engineering (decomposition, integration)
  - Can have teams for each chunk.

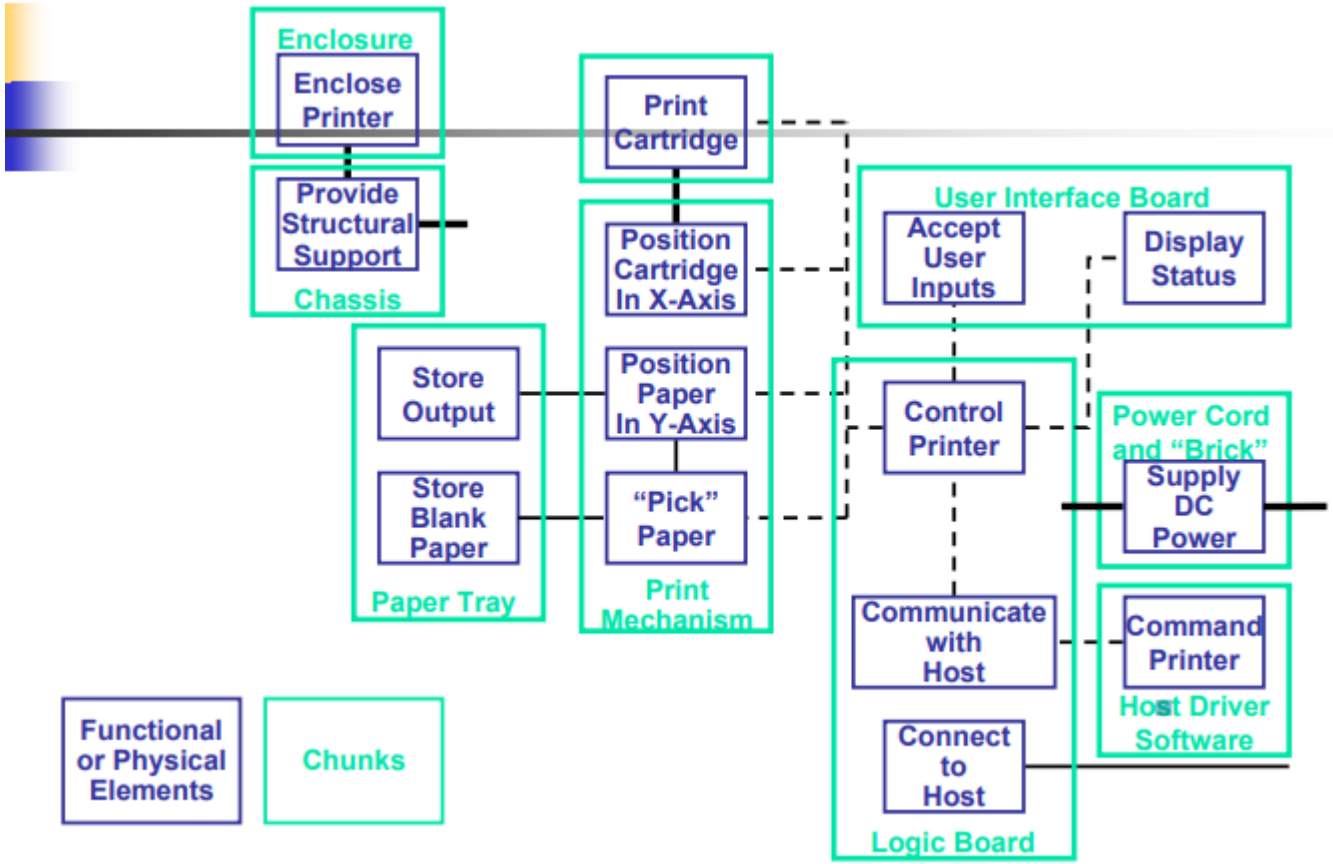
How can we determine appropriate chunks?

- Create rough product architecture (schematic)
- Cluster elements
- Create rough geometric layout
- Identify fundamental and incidental interactions

e.g. Desktop Printer: Identifying Chunks

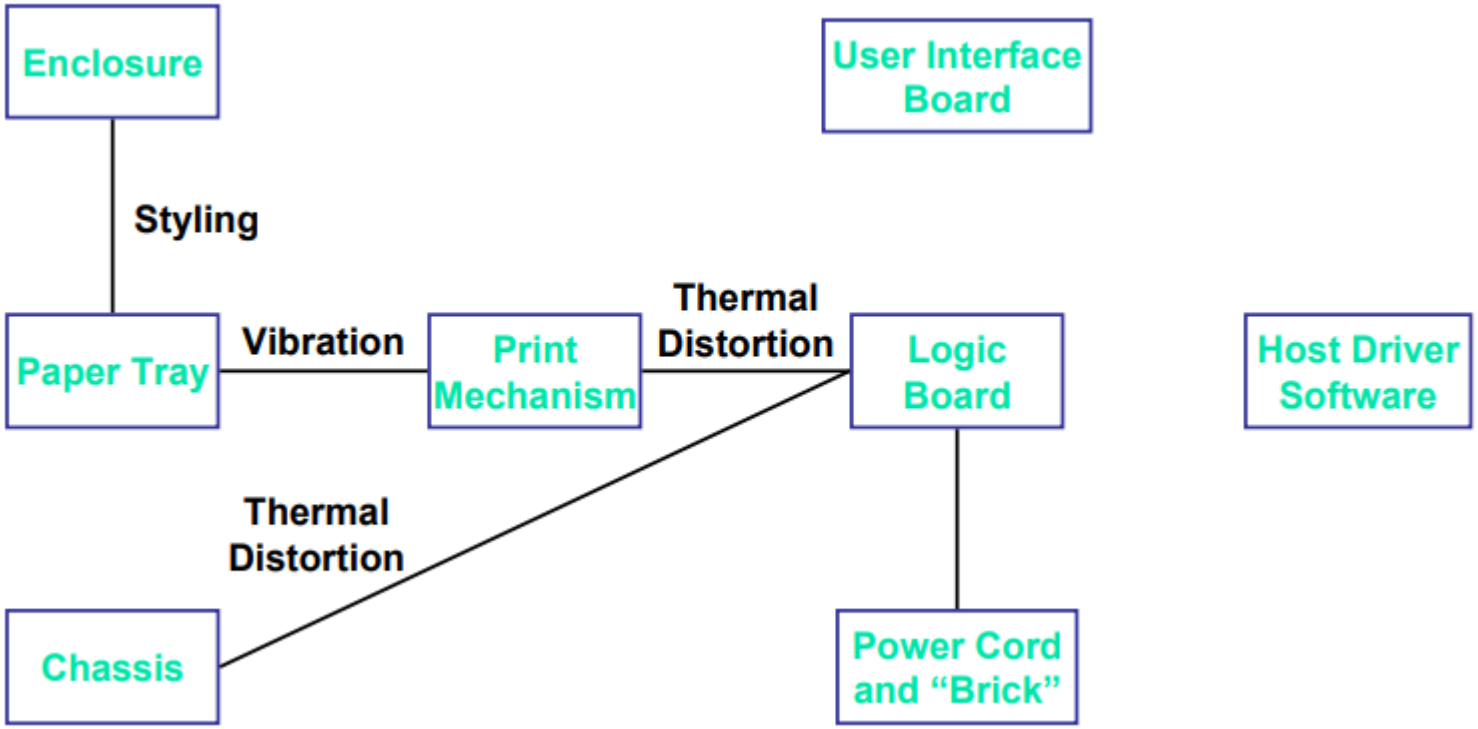


e.g. Desktop Printer: Identifying Chunks

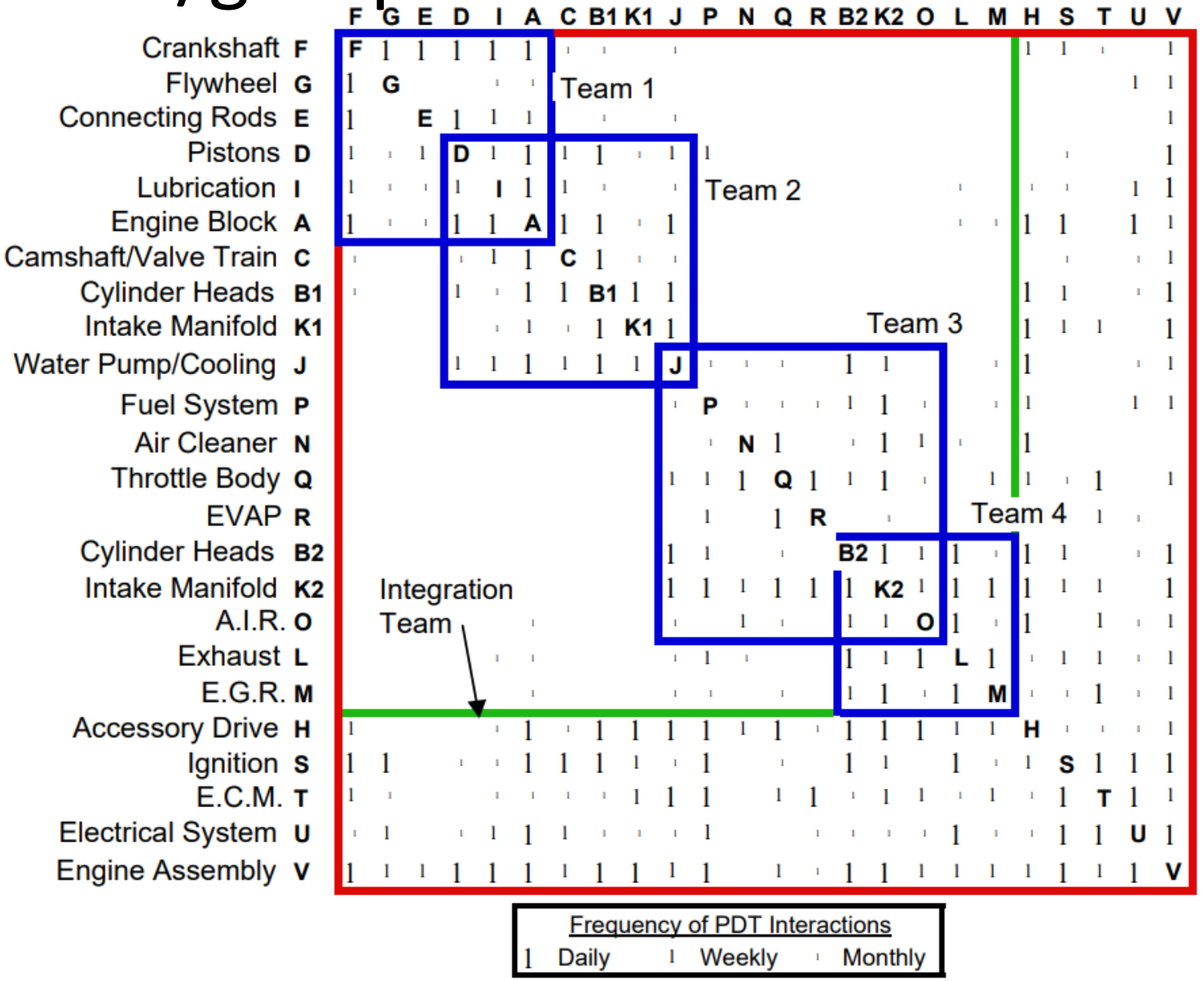


- Geometric integration and precision
- Function sharing

Identify incidental interactions



Map to teams/groups



Product Architecture: Summary

- Product architecture decisions affect product change, product variety, component standardization, product performance, manufacturability, and PD management.
- A key characteristic of a product architecture is the degree to which it is modular or integral
- Can be determined during or after concept selection

Engineering Design

Mechanism Design

- Gears
- Pulleys/belt
- Cams
- Worm gears
- Linear actuators
  - Rack and pinion
  - Linear Actuators

Rotary to Rotary Mechanism

Why are these useful?

What are some examples?

Gears

- Many different forms and design of gears (spur, etc.)
- Gear chains typically used to vary the speed or torque
- Can be used to translate/rotate motion


Spur Gears


Helical Gears


Gear Rack


Bevel Gears


Miter Gears

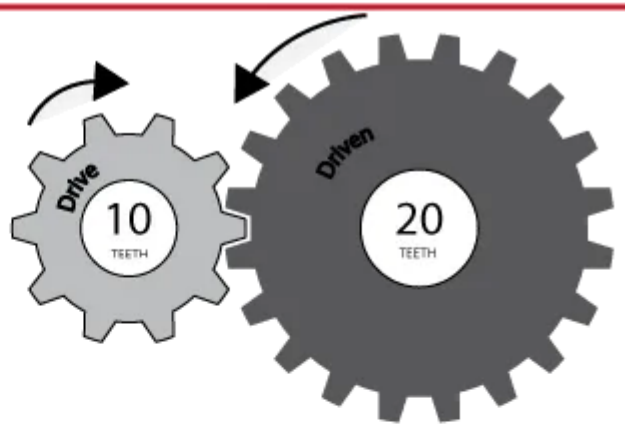

Worm Gears


Screw Gears

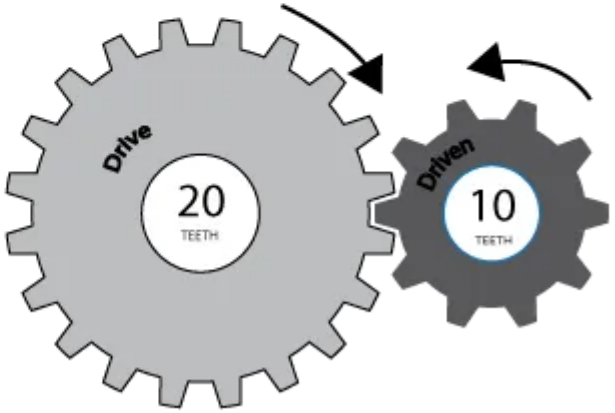

Internal Gears

Gears (Rotary to Rotary)

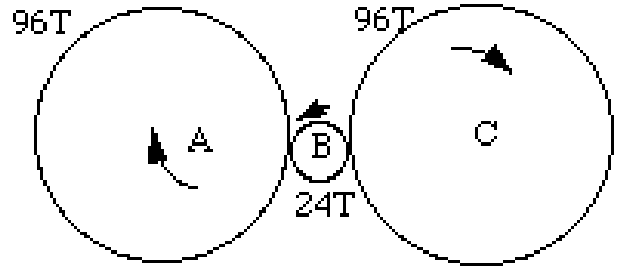
Gear reduction occurs when the drive gear is smaller or has fewer teeth than the driven gear.



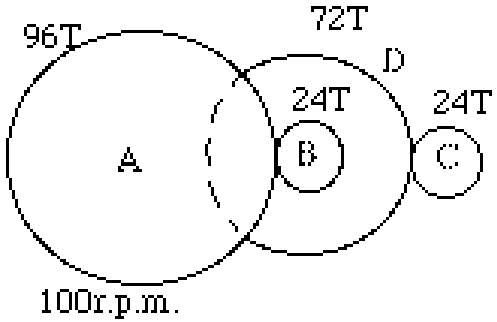
Overdrive occurs when the drive gear is larger or has more teeth than the driven gear.



Gear Chains

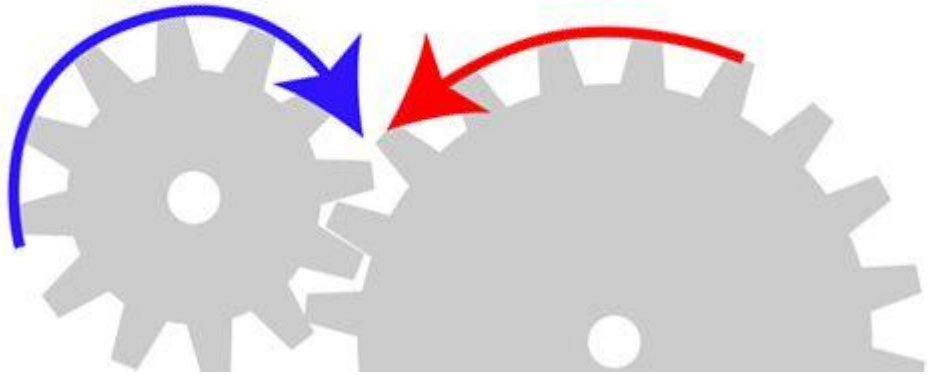


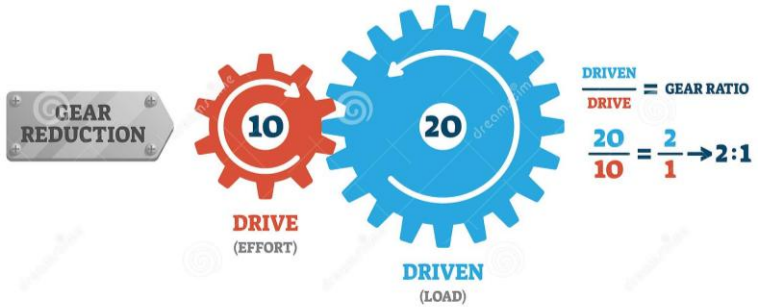
Compound Gears

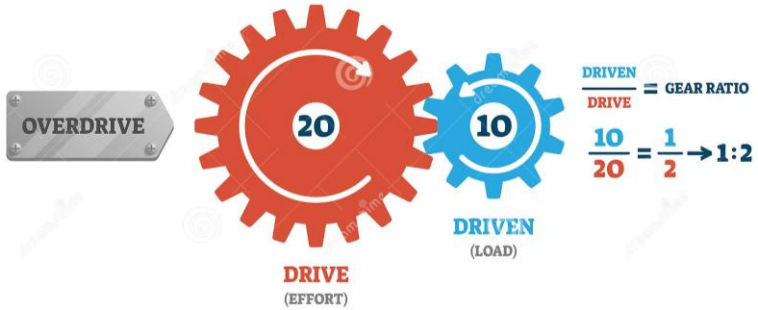


Gears

$$RPM_A \times Teeth_A = RPM_B \times Teeth_B$$



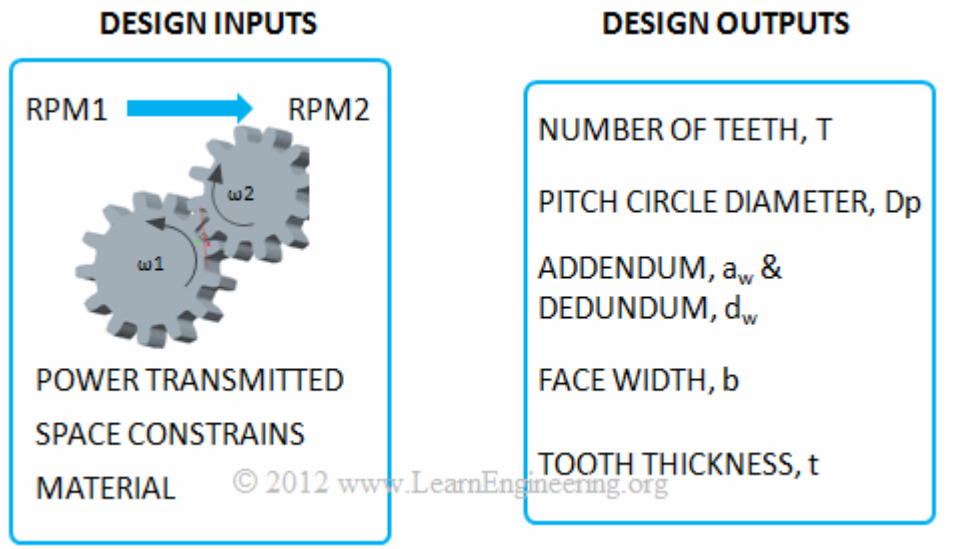


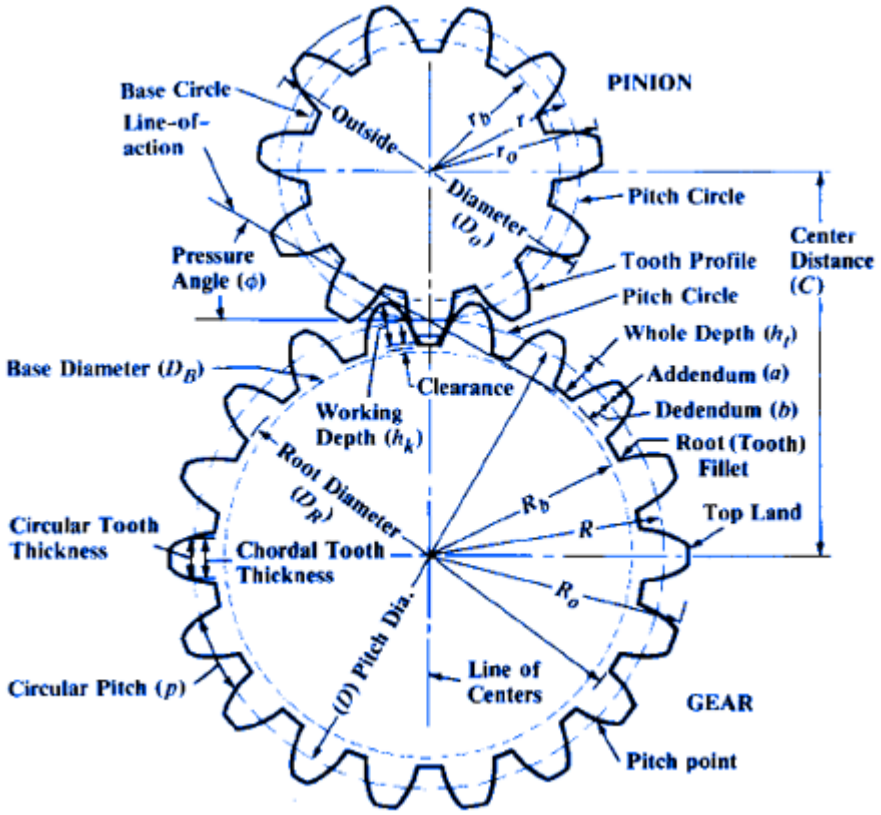


$$\text{Gear Ratio} = \frac{\omega_1}{\omega_2} = \frac{n_1}{n_2} = \frac{d_2}{d_1} = \frac{T_2}{T_1}$$

Practical Considerations of Designing Gears

- Materials (wear)
- What gear ratio required → Tooth size and radius
- Clearance

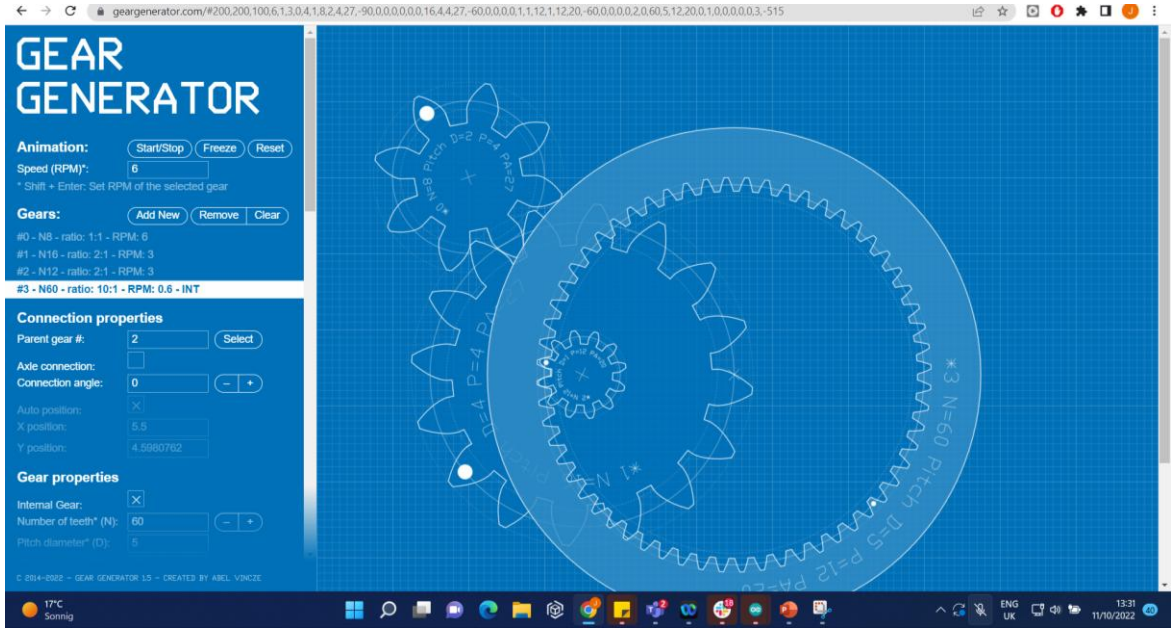


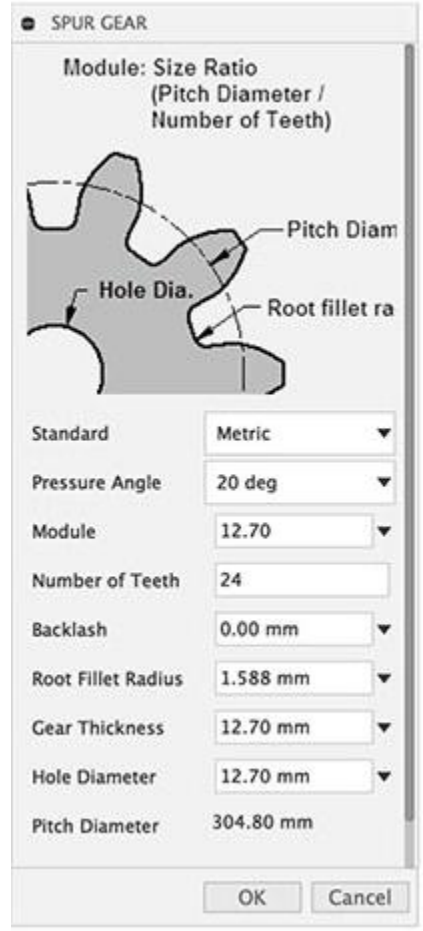


Gears

Tools for generating gears

- Dependent on your fabrication method
- Means of mounting the gears are required

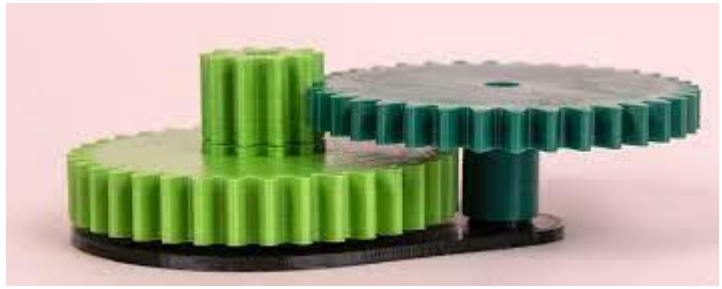




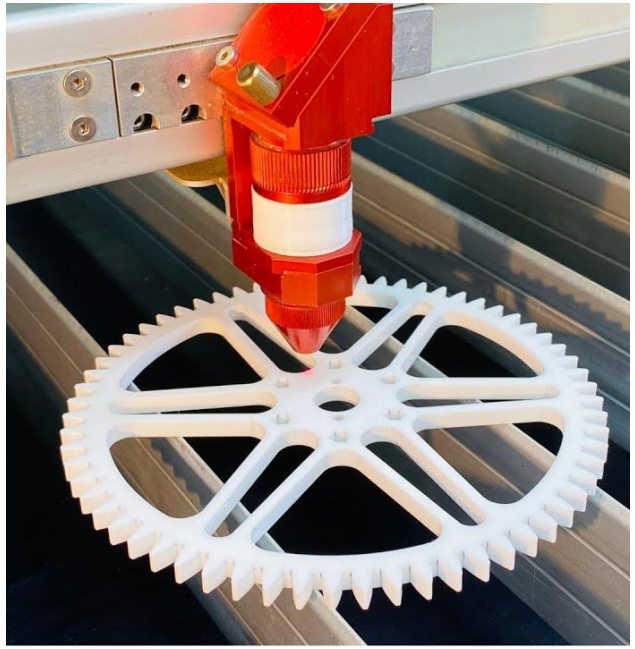
Fusion 360 Gear Tool

Gears

How should we fabricate?



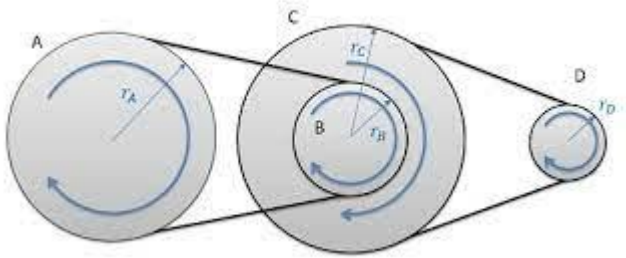
3D Printing



Laser Cutting

- The fabrication method can change the gear parameters
- To achieve reasonable clearance, may require prototyping or trial and error

Pulleys & Belt Driven Systems



$$\frac{d_{out}}{d_{in}} = \frac{\omega_{in}}{\omega_{out}} = \frac{\tau_{out}}{\tau_{in}}$$

- Follow same rules as gears (speed and torque)
- Relies on capstan friction (toothless), or timing belts
- No issues with clearance of teeth
- Requires tensioner
- Can easily transfer through angles, corners etc.
- Rotation direction the same

Rotary to Linear Mechanism

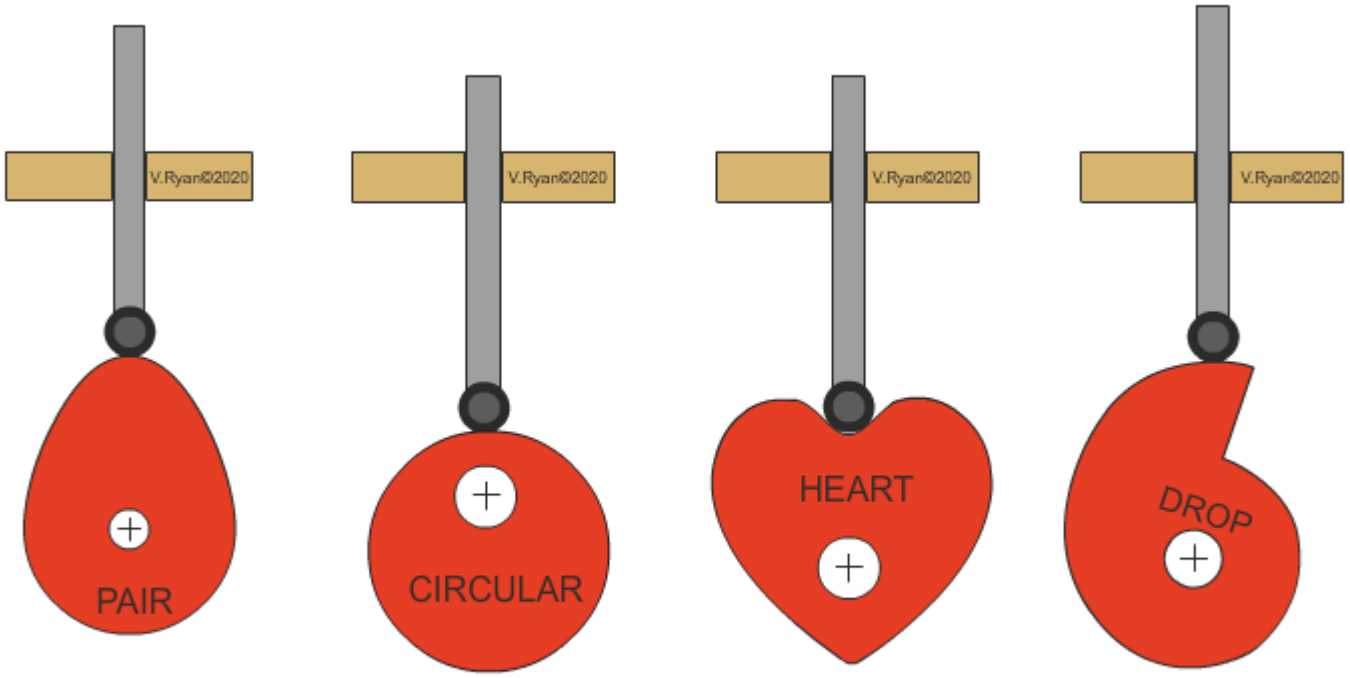
Why are these useful?

What are some examples of mechanisms?

Rotary → Linear

- CAMs
- Worm Gear Mechanism
- Lead Screw
- Rack & Pinion
- Tendon driven Structures

CAMs

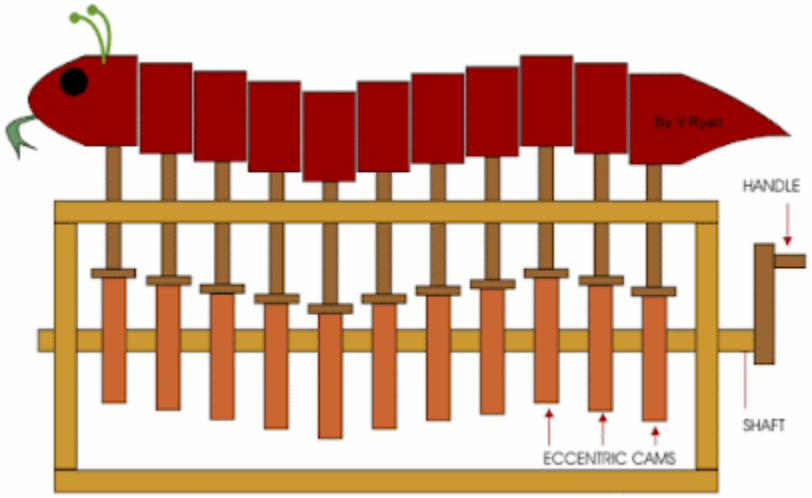


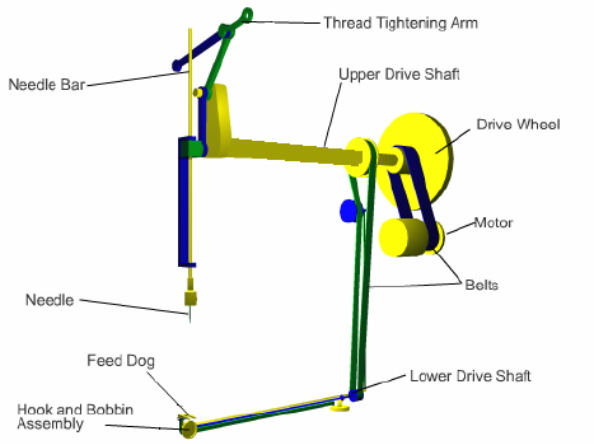
Cams can be shaped in any number of ways and this is determined by the way the follower is to move.

The shape of the cam is called the PROFILE.

The profile of the CAM sets the temporal motion of the FOLLOWER.

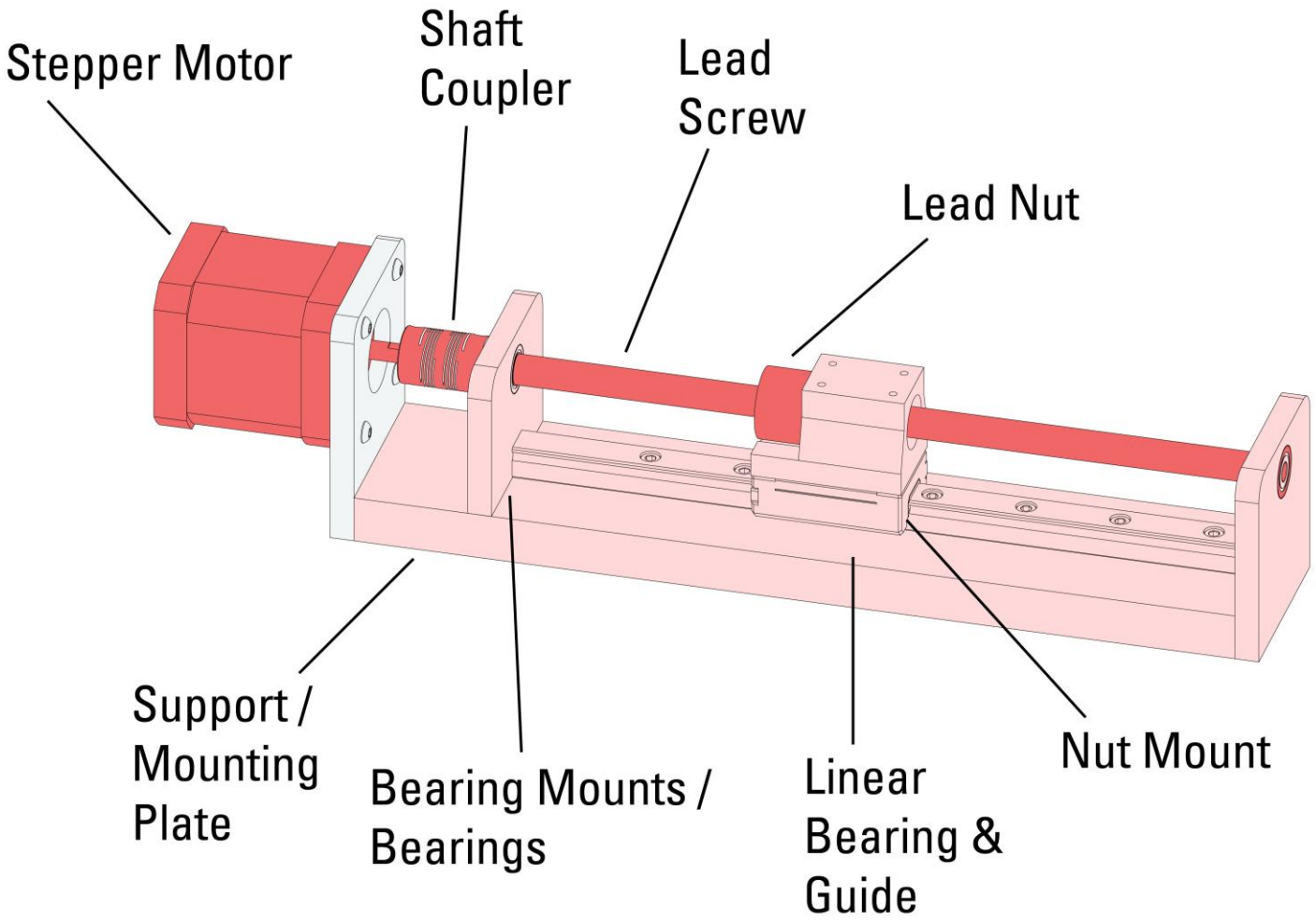
Relies on gravity/spring for the release.


By V.Ryan
HANDLE
SHAFT
ECCENTRIC CAMS


Thread Tightening Arm
Needle Bar
Upper Drive Shaft
Drive Wheel
Motor
Belts
Needle
Feed Dog
Lower Drive Shaft
Hook and Bobbin Assembly


Durabase

Lead Screw Mechanism



$T = Torque\ required\ from\ motor\ (Nm)$

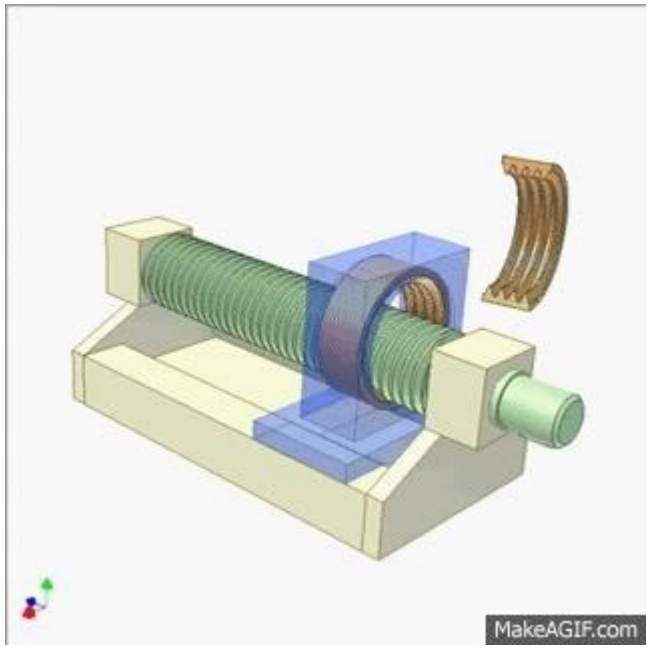
$L = Screw\ lead\ \left(\frac{mm}{rev}\right)$

$e = Lead\ screw\ efficiency$

$$T = F * \frac{L}{2\pi e}$$

- Very high torque
- High holding torque (not back-drivable)
- High precision
- Slow to move (depends on pitch of the screw)
- When assembly – require a linear approach to minimizing the torque

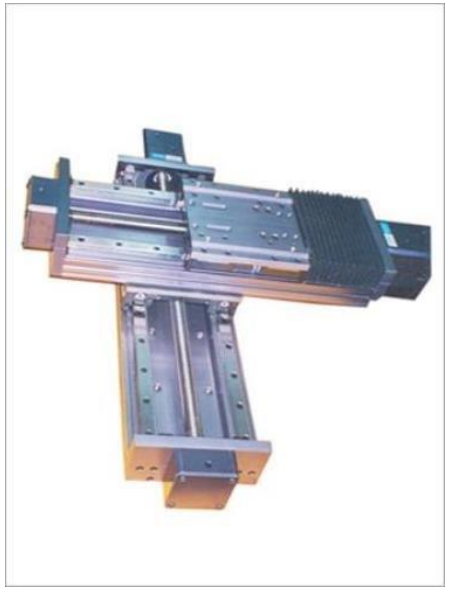
Lead Screw Mechanism



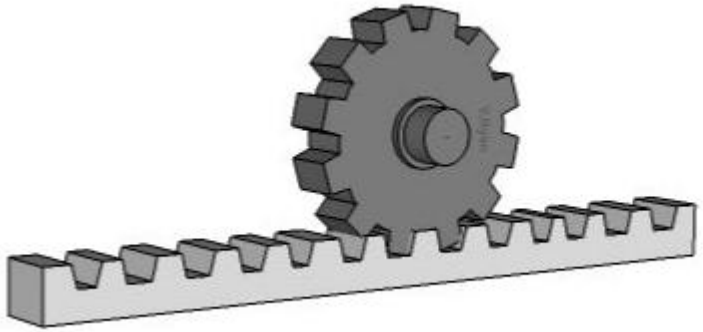
Typically utilizes a stepper motor to provide the necessary precision & control

Typical Uses:

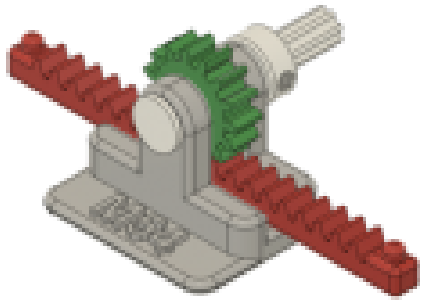
- High precision stages
- Syringe pumps
- Precision cartesian platforms
- High force grippers

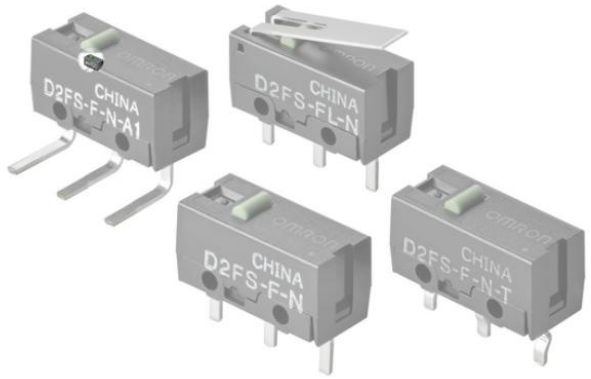


Rack & Pinion

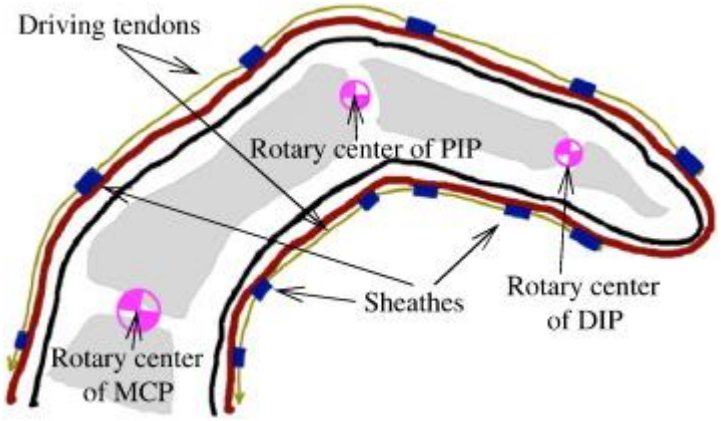


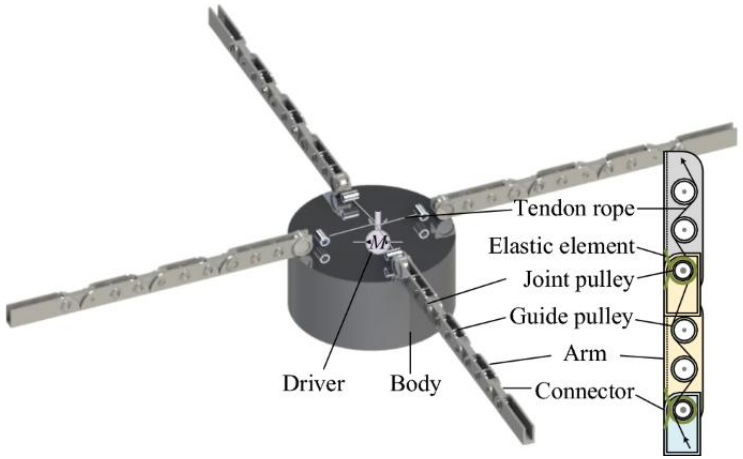
- Faster than a linear actuator
- Can set speed/torque through the size of the gear/teeth
- Few parts to design/fabricate
- Typically use a DC motor + encoder or limit switches

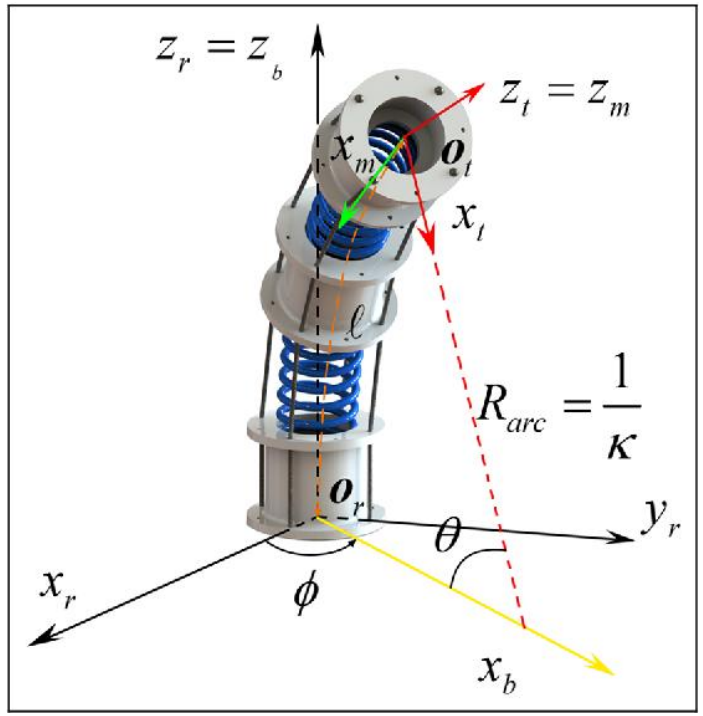


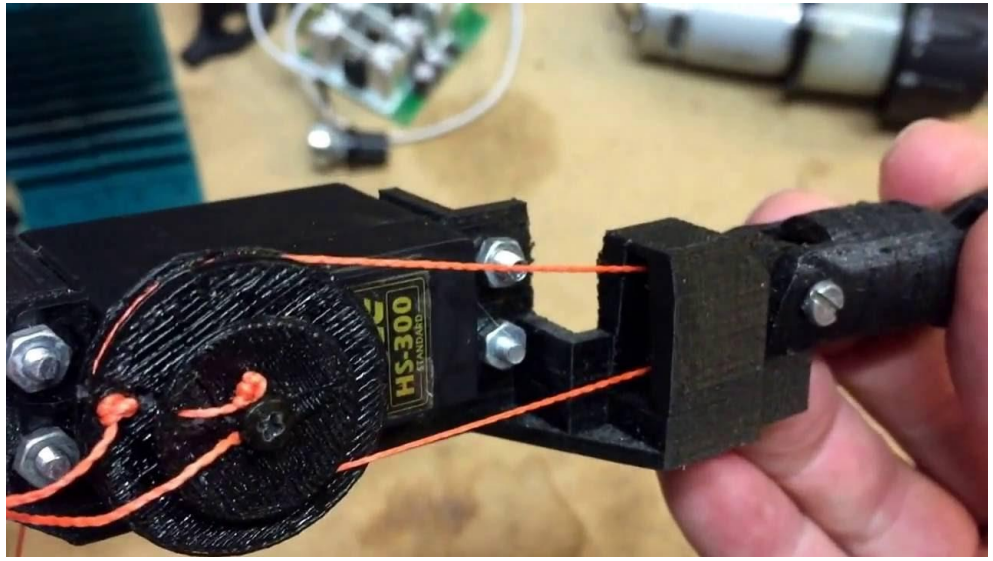


Tendon Driven Mechanism

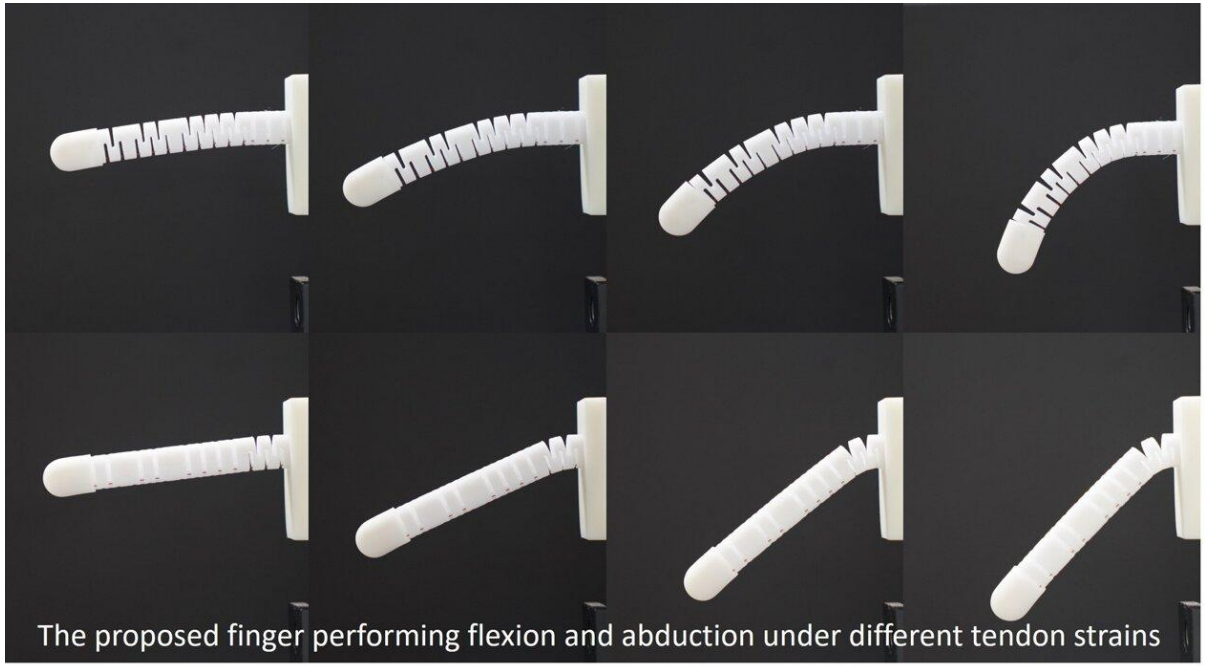


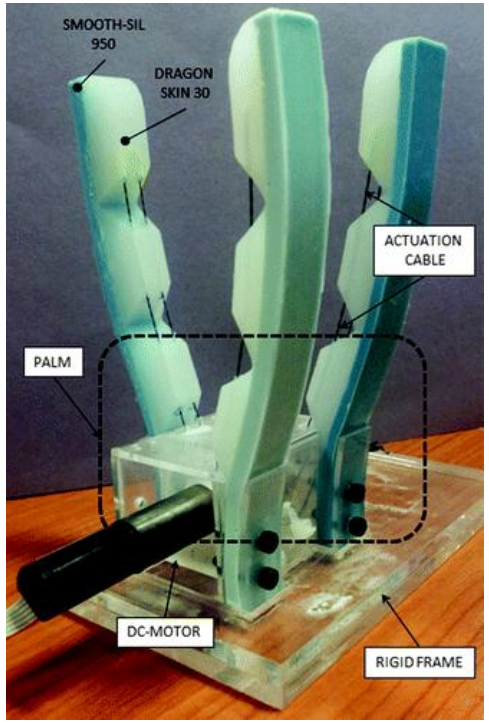


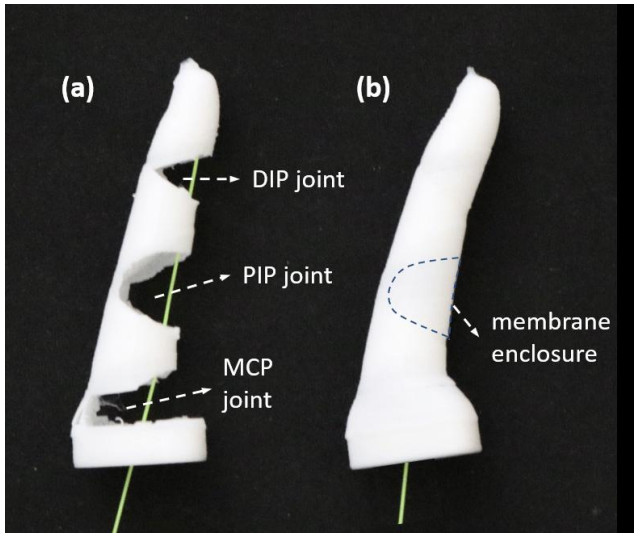




Soft Robotics Grippers

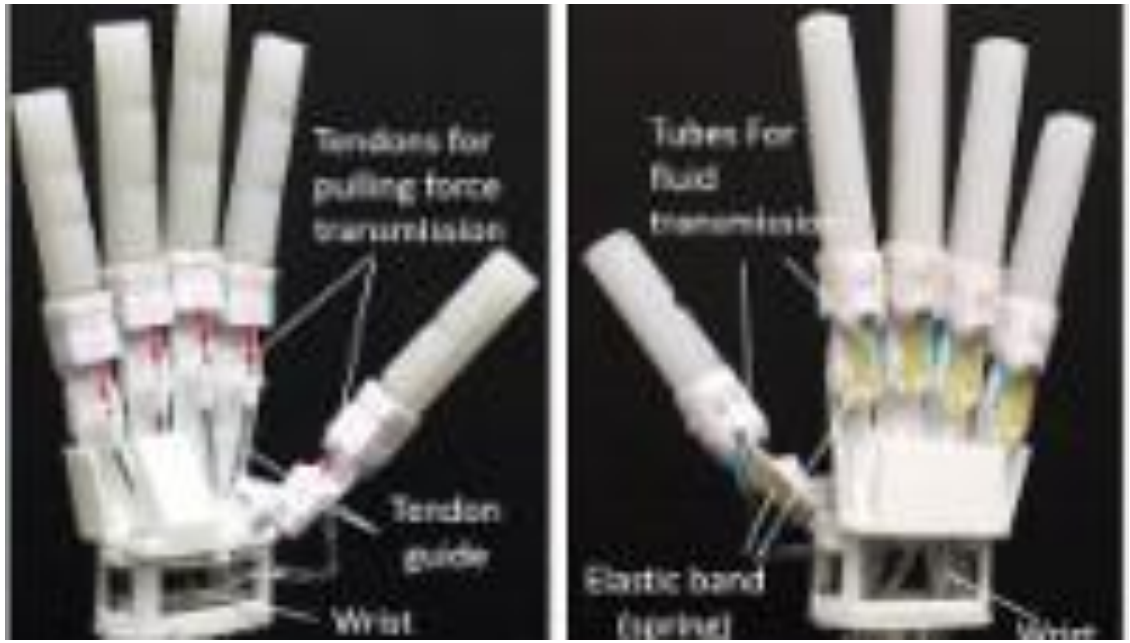

The proposed finger performing flexion and abduction under different tendon strains





- Compliance provides robustness and physical adaptability
- Compliance can be a material property but also a property of the geometry and design
- Tendons can be antagonistic (and with one motor)
- Tendons could have one actuated direction, and the other passive

Soft Robotics Grippers



Differential Drive Tendon Driven Systems

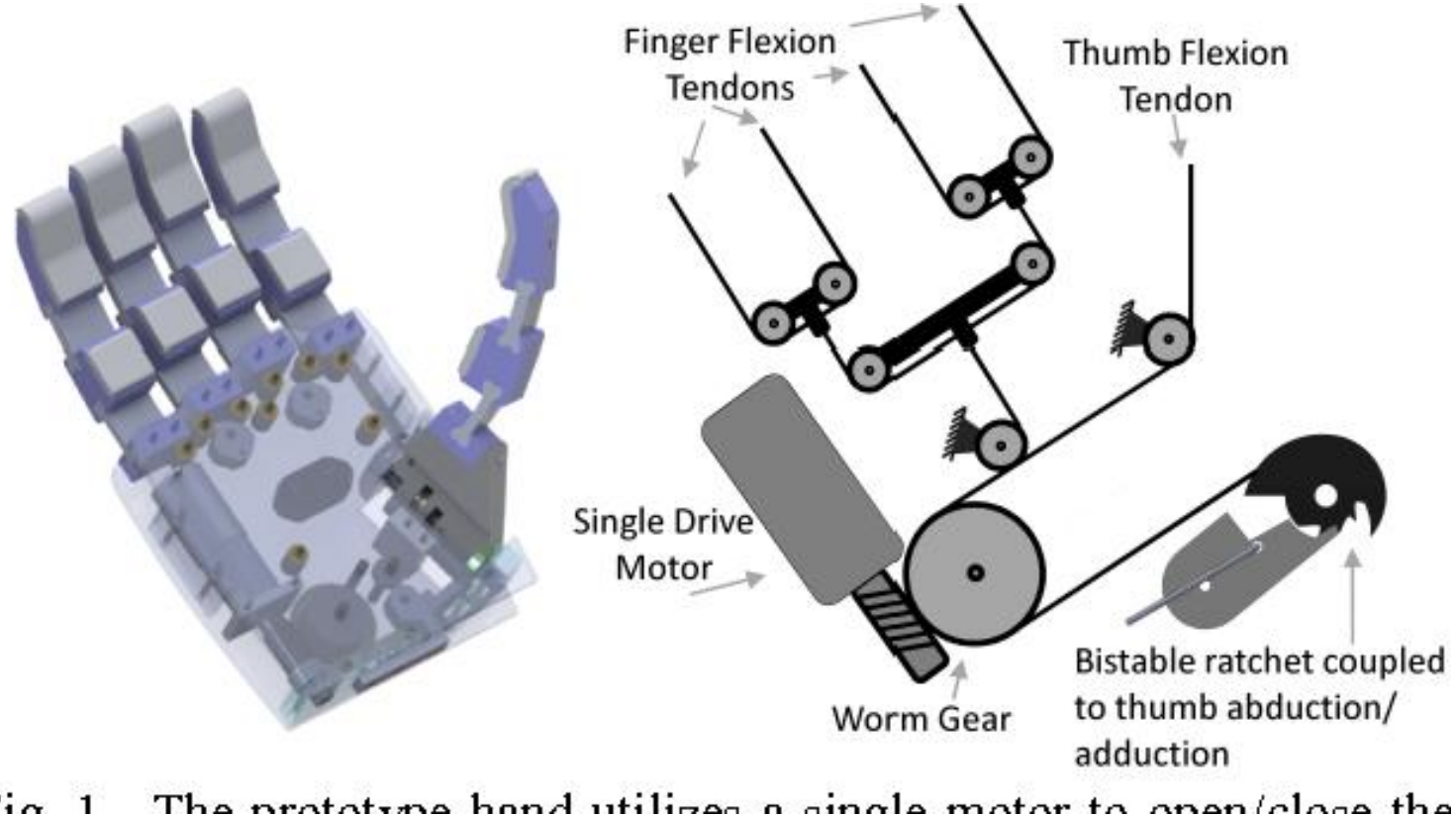
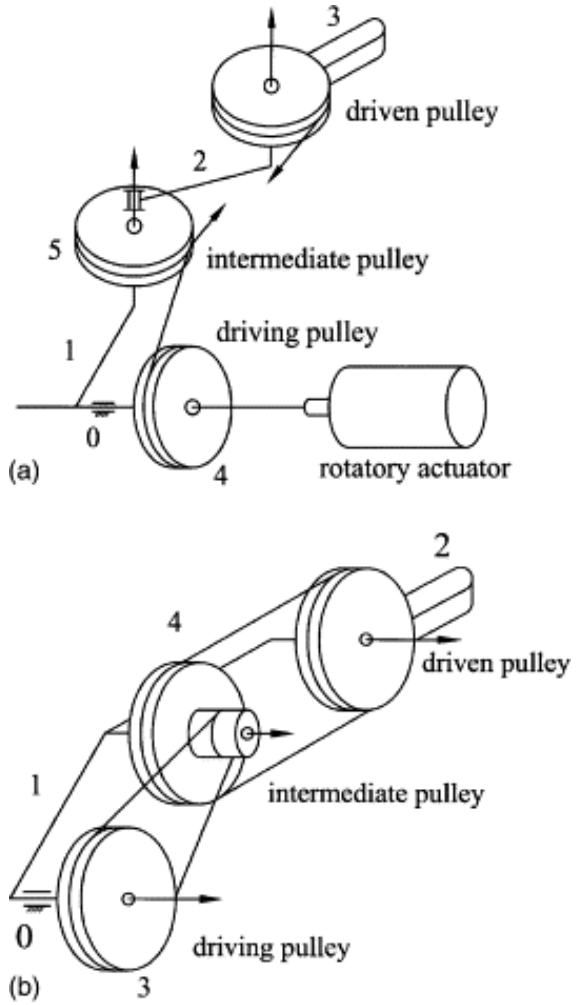


Fig. 1 The prototype hand utilizes a single motor to open/close the

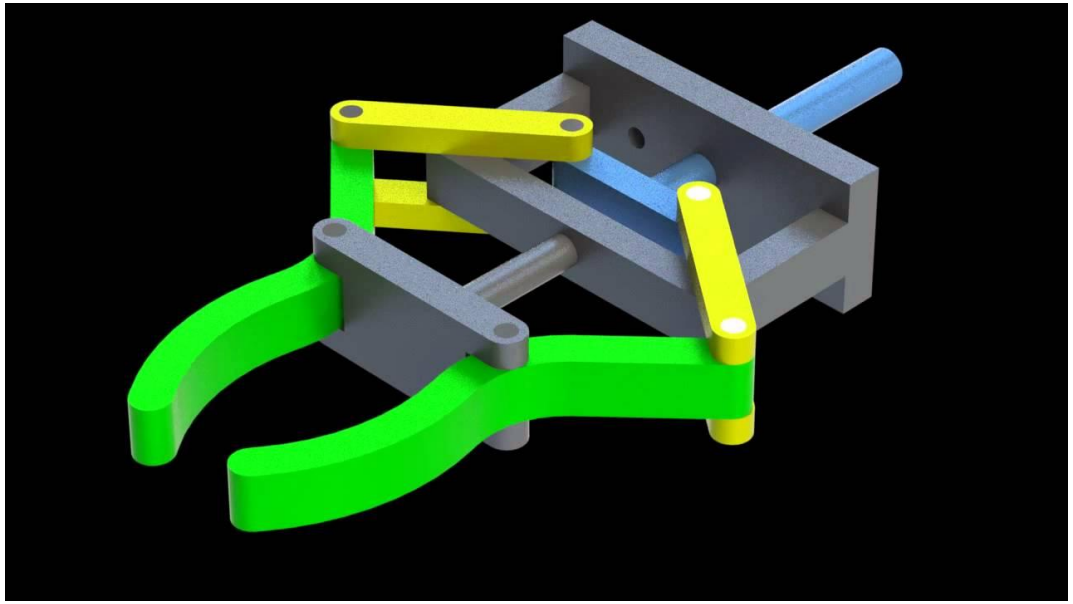


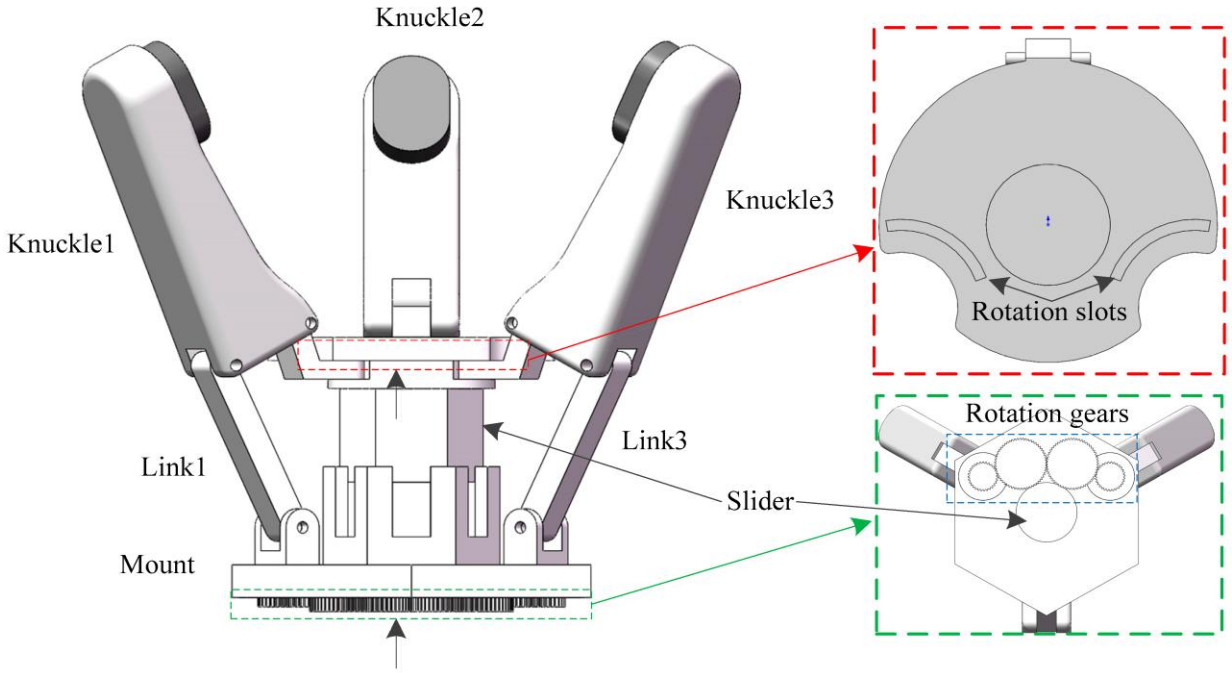
Linear to Linear Mechanism

Why are these useful?

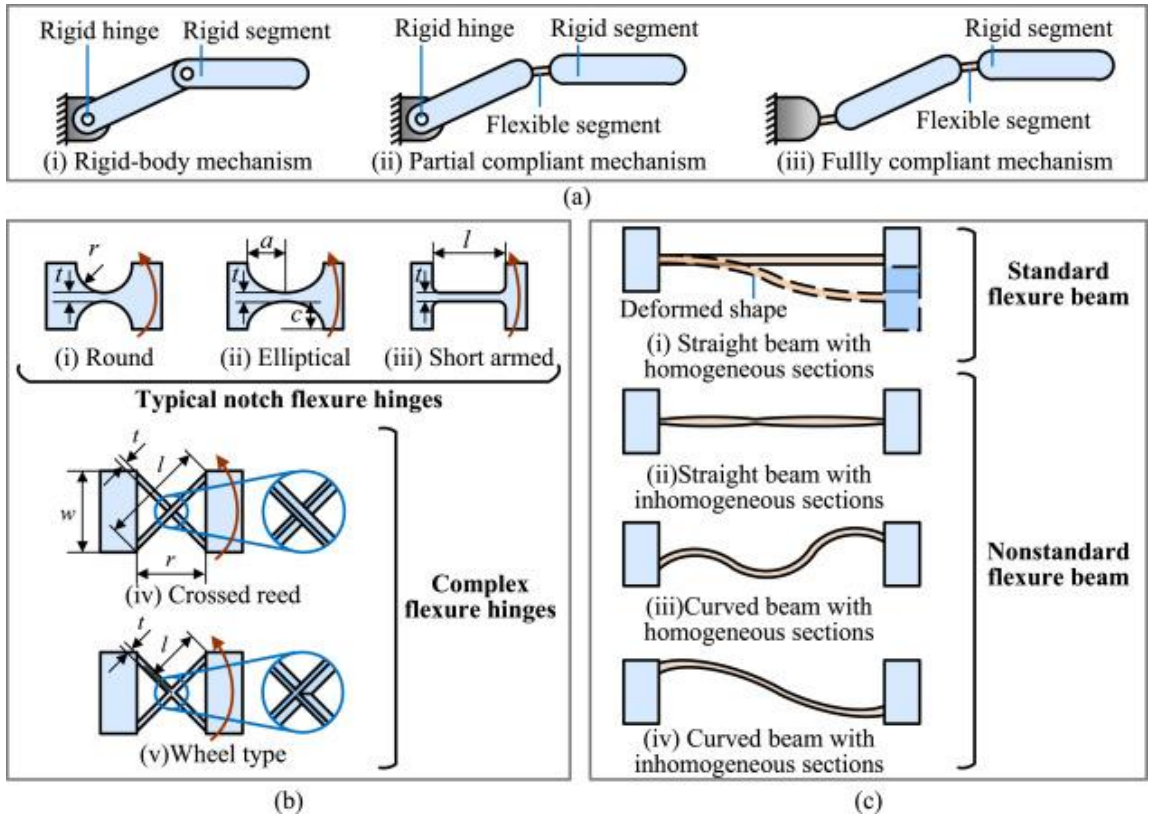
What are some examples of mechanisms?

Linkage Based Gripper

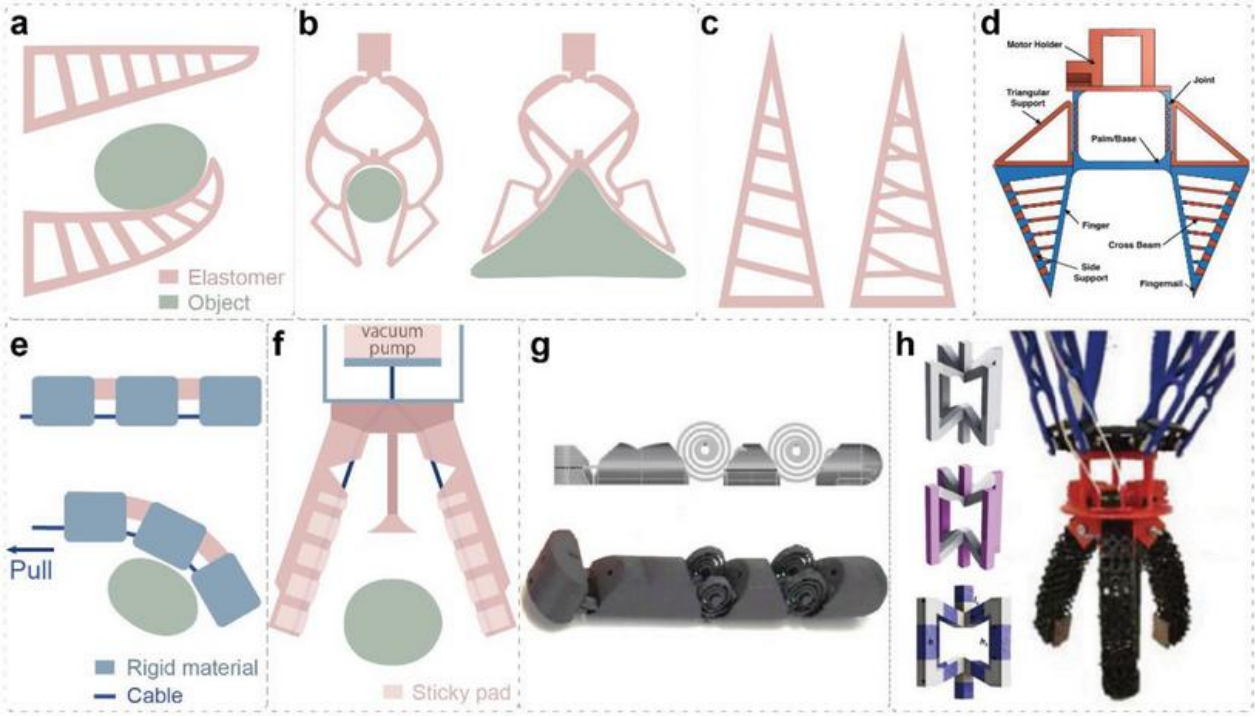




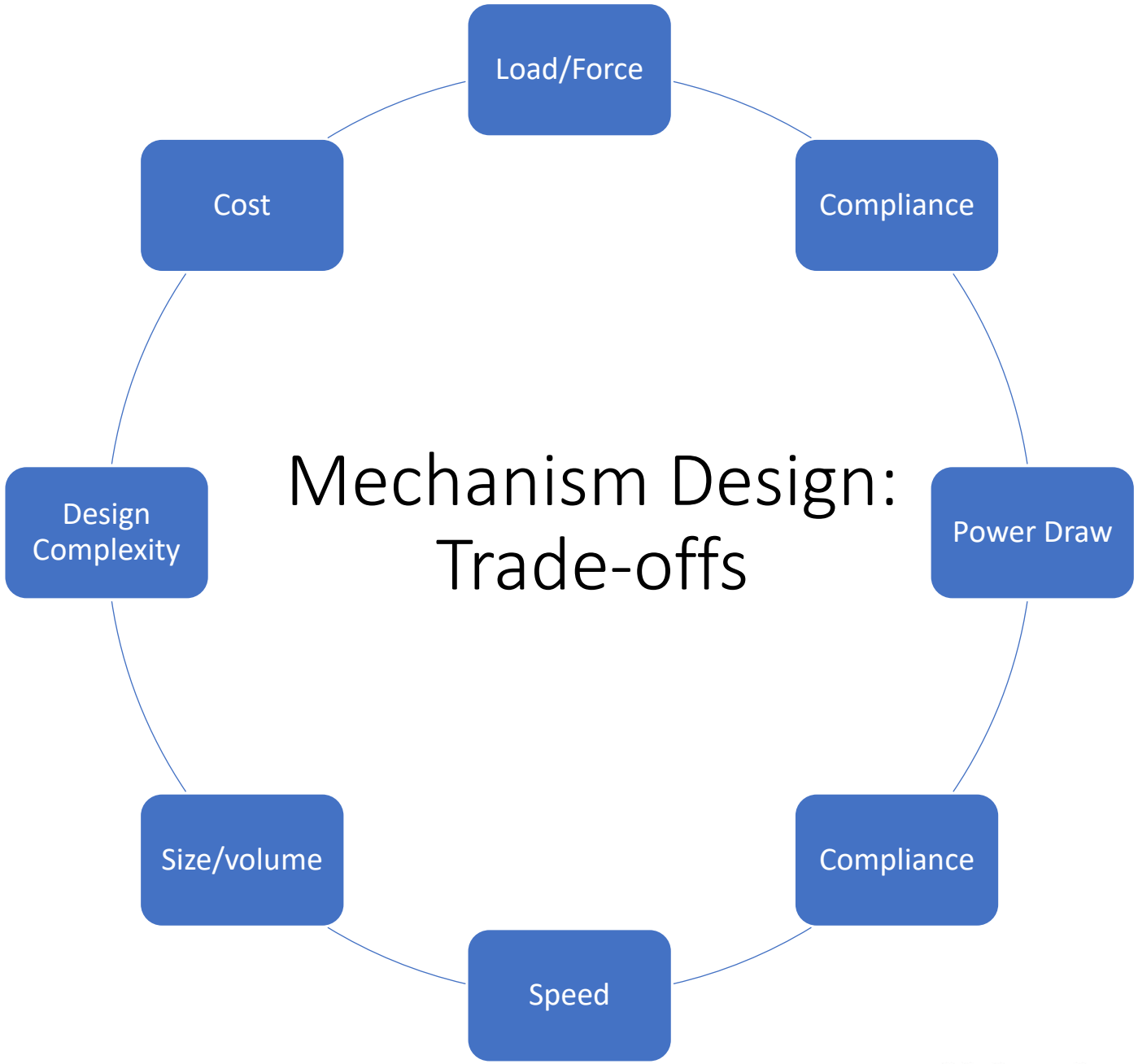
Compliant Mechanisms

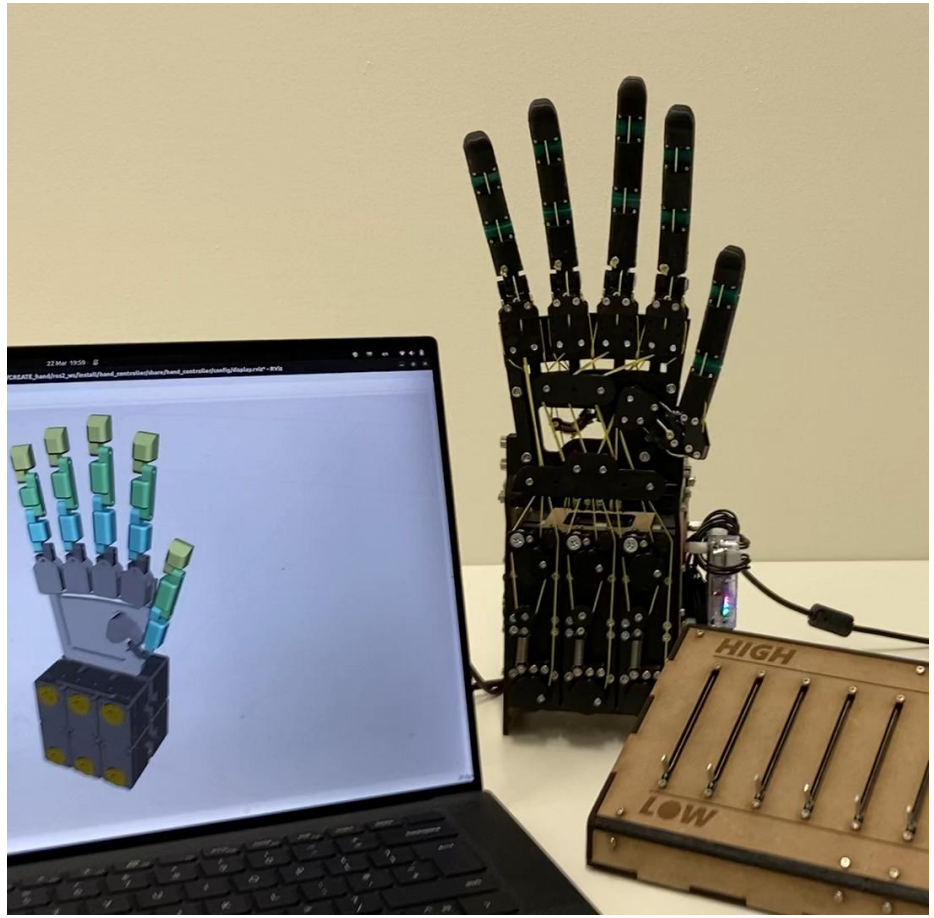


Passive Adaptive



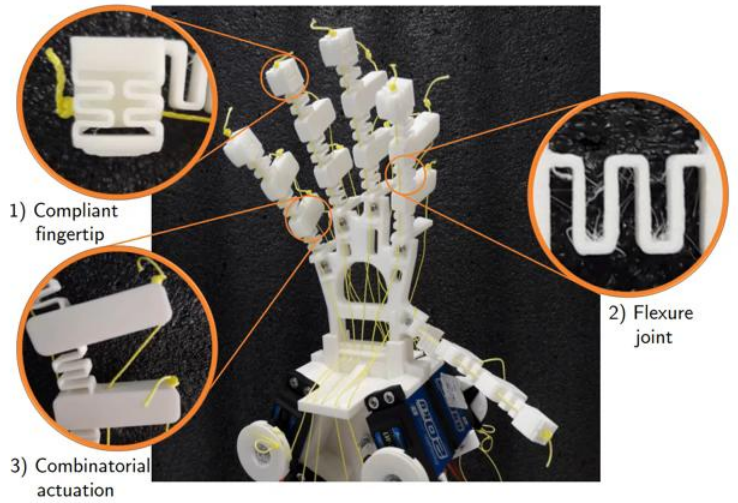
Mechanism Design: Trade-offs





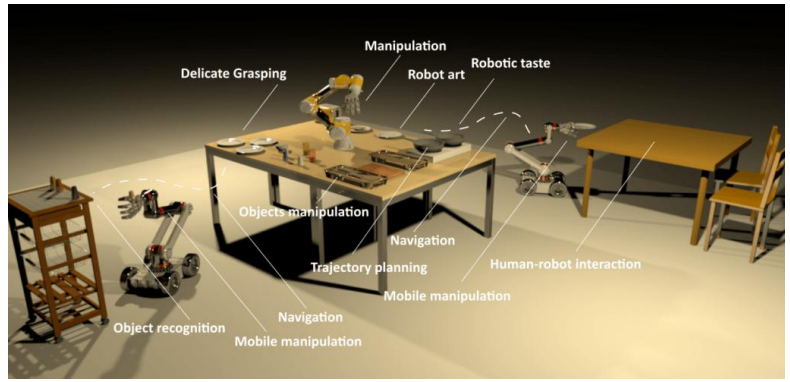
Goals:

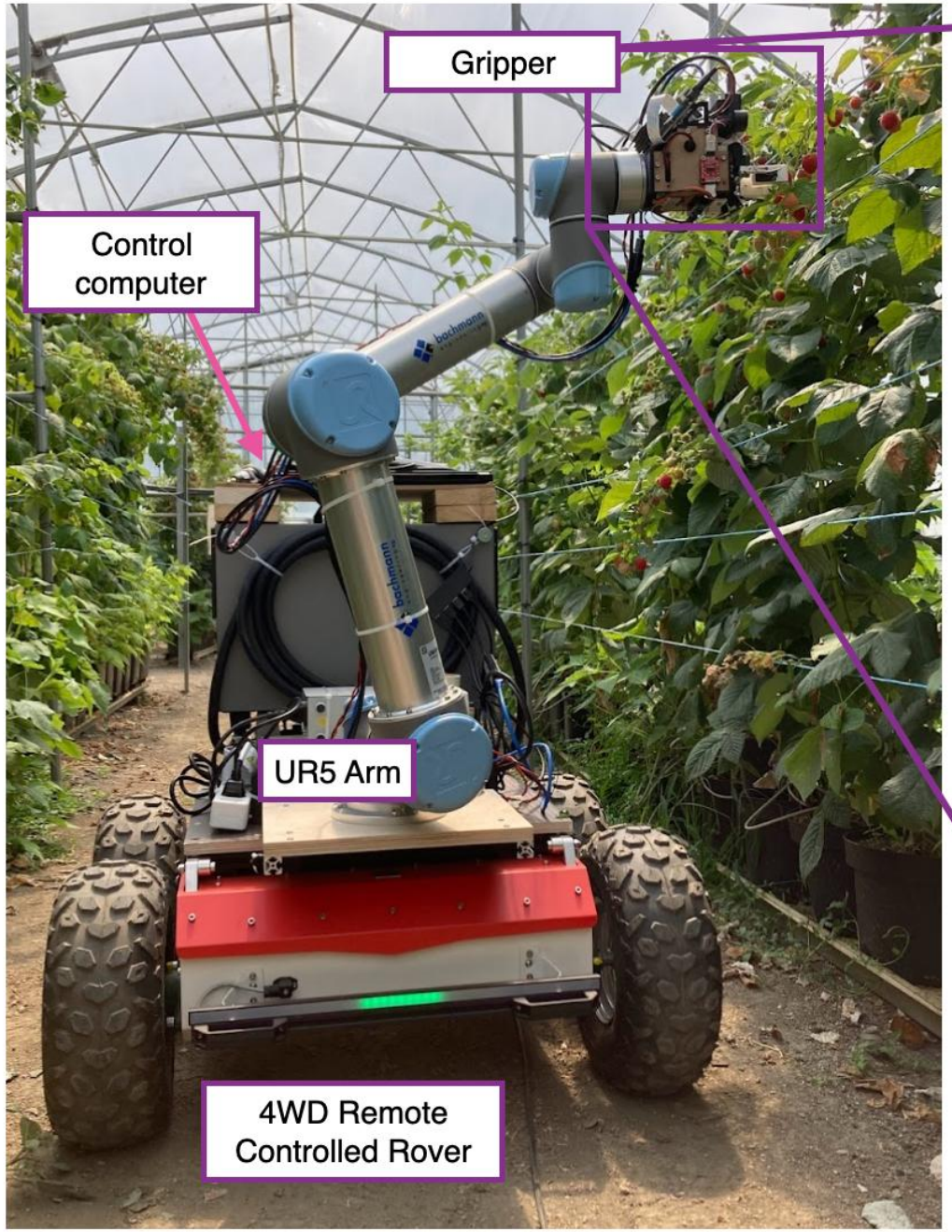
- Exploit 3D printing for fabrication of a bio-inspired robot hand
- Develop scaleable control strategy for flexure based joints

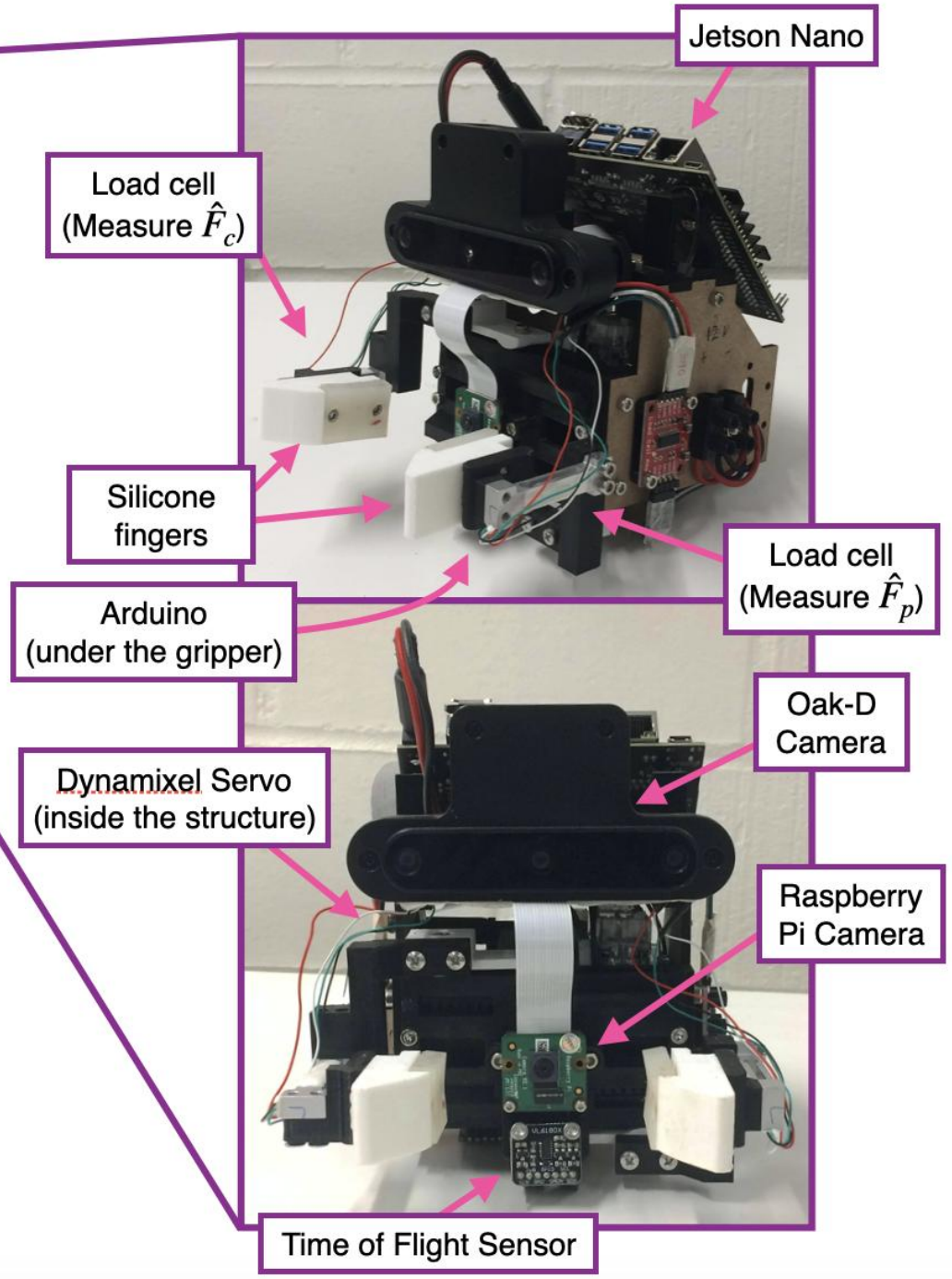


Preparation of Food

An ultimate ‘benchmark’ for robotic manipulation?




Gripper
Control computer
UR5 Arm
4WD Remote Controlled Rover


Jetson Nano
Load cell (Measure $\hat{F}_c$)
Silicone fingers
Load cell (Measure $\hat{F}_p$)
Arduino (under the gripper)
Oak-D Camera
Dynamixel Servo (inside the structure)
Raspberry Pi Camera
Time of Flight Sensor

PRODUC

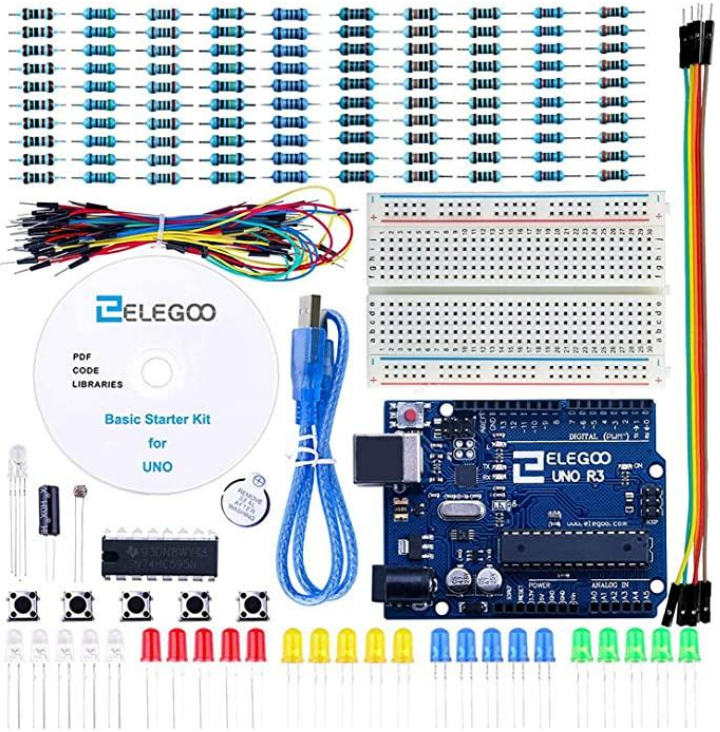
Raspberry Harvesting

Robotic harvesting allows exploration of new harvesting conditions (e.g. at night) when improved sugar content, or shelf life can be achieved.

https://vevox.app/#/m/121337220

Course Parts

Base microcontroller kit



+

Sensors and actuators/components from course list

+

Protoyping budget at SPOT

- MDF
- 3D printing materials
- Fasteners

30F Per team for the whole project! You must budget for this!

Project Groups

Please create projects groups of 5 students and sign up in the sheet below providing a group name and number, names, scipers and emails of everyone is each group. If you would be prefer to be randomly allocated, please add your name, sciper and email to the columns on the right hand side of the sign-up sheet.

Deadline for signing up Tuesday September 10th at mid-day.

Sign up sheet:

1GiQcrJ2ttffkiwstNVEjvSFxKUSYCRr5Wniq3QOz7_c
https://docs.google.com
Connect to Google Drive to update

Budgeting Spreadsheets

Any spending at the SPOT for consumables and protoyping must be recorded here. Each team has a budget of 30F. If for whatever reason you go over this, this must be approved with good reason by talking to the course leader.

It is your responsibility to make your own sheet and keep track of this. Failure to do so will mean you will not get access to other actuators/sensors.

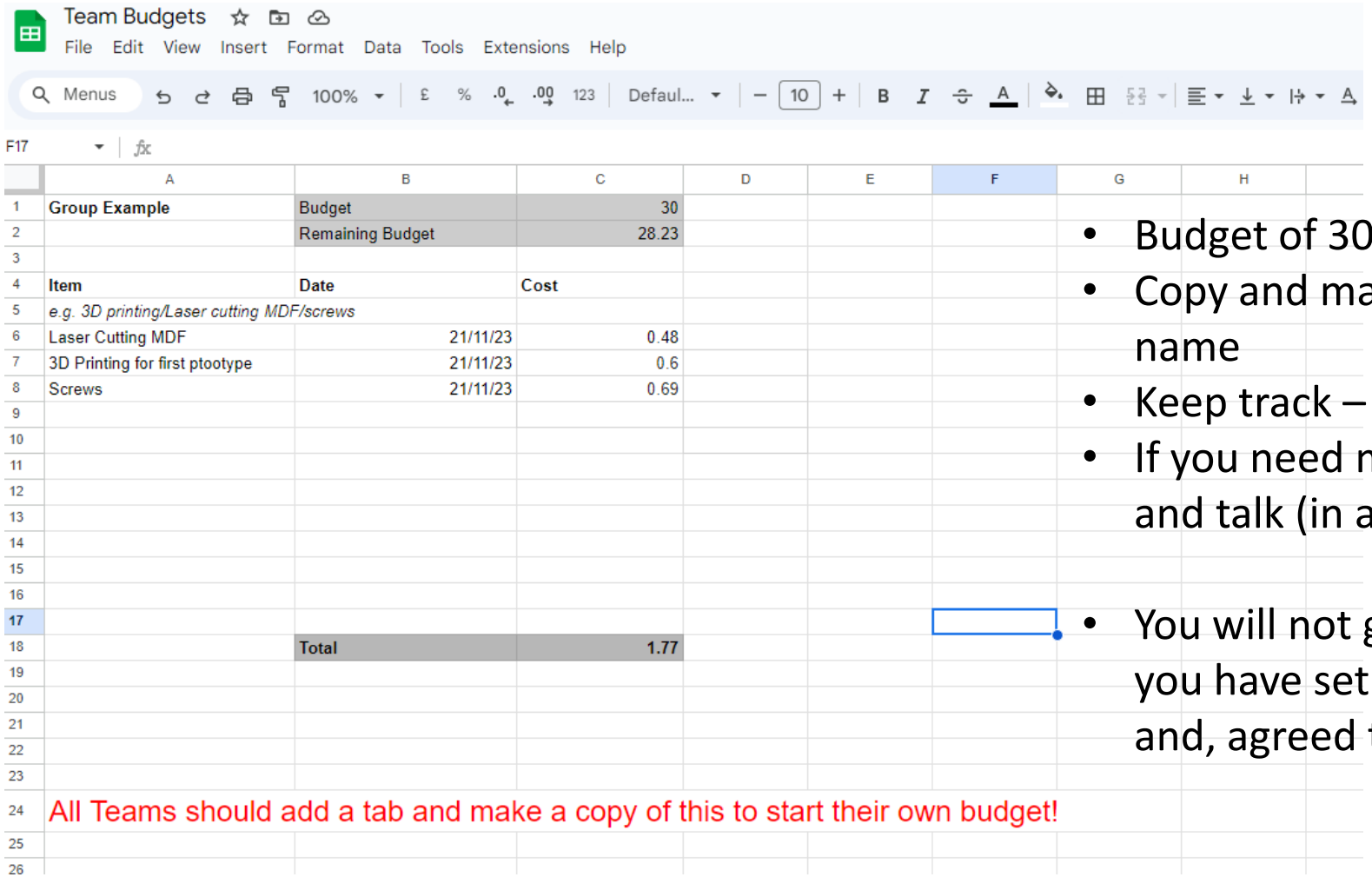
1y8IPTIhQII84bNVgiuXOqiP6o7OSgDmkJcPJf14MbnY
https://docs.google.com
Connect to Google Drive to update

Parts Signing Out

Parts signed out will be added to this document so we can keep track of who has what, and make sure it is retrieved at the end of the project

1OOZGpWJ_vclqx_tvdbxtPNI2Sy-u_fubYdsCq8XXplk
https://docs.google.com
Connect to Google Drive to update

Budgeting: Keeping Track



	A	B	C
1	**Group Example**	Budget	30
2		Remaining Budget	28.23
3			
4	**Item**	**Date**	**Cost**
5	*e.g. 3D printing/Laser cutting MDF/screws*		
6	Laser Cutting MDF	21/11/23	0.48
7	3D Printing for first ptootype	21/11/23	0.6
8	Screws	21/11/23	0.69
18		**Total**	**1.77**
24	All Teams should add a tab and make a copy of this to start their own budget!		

- Budget of 30F per team at SPOT
- Copy and make your own tab (Group X) as name
- Keep track – we will check
- If you need more budget/you go over come and talk (in advance!)

- You will not get any access to parts/kits until you have setup your teams budget sheet, and, agreed to tracking your budget.

Signing out of parts

A1 | Parts Taken

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Parts Taken**											
2												
3	**Group Number**	**Team has setup budget and agree to keep track (y/n)**	**Arduino Starter Kit**	**Parts given (add to the colum, and specific quantity)**								
4	Example	y	Yes	Motors x2	Webcam	1 pressure pad						
5	Group 1											
6	Group 2											
7	Group 3											
8	Group 4											
9	Group 5											
10	Group 6											
11	Group 7											
12	Group 8											
13	Group 9											
14	Group 10											
15	Group 11											
16	Group 12											
17	Group 13											
18	Group 14											
19	Group 15											
20	Group 16											
21	Group 17											
22	Group 18											
23	Group 19											
24	Group 20											
25	Group 21											

We must see this you have added a budget tab, and you must agree to this before getting access to parts

Parts will be available for pick up:

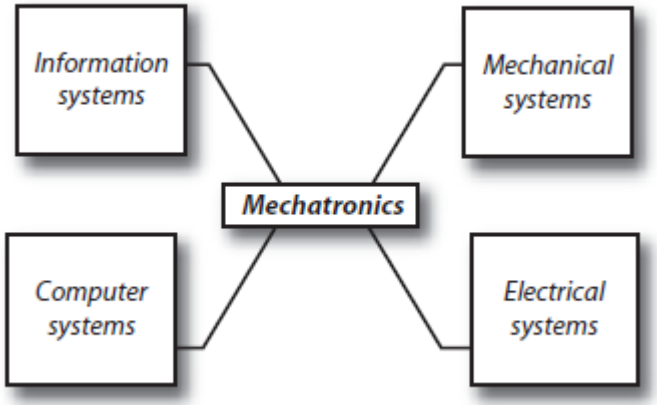
- During lecture sessions
- During drop in sessions
- You must have your design set following design review (or if earlier approval by TA) before you can get additional parts

Microcontrollers

Today: Parts & Practical Work!

- Get 1 Arduino kit per team
  - Your to keep for the project
  - Expected to return, it its box with parts
  - Resistors are consumables… but don’t lose!!
- Please sign out on the parts sheet
- This doesn’t count to your 30F budget

Mechatronic Product



Mechatronic Product

Control Strategy

Mechanical System & Actuation

Electronics & Sensing

Information System (Computation)

Mechatronics: What is available to you.

Mechanical System & Actuation

Actuators

<u>Rotary 0-180 degrees position control:</u>

- Servo motors (small)
- Servo motors (large)

<u>Rotary continuous, current control:</u>

- DC motor (continuous rotation)

Materials

- Fabrication of mecahnisms
- Plastic
- MDF
- Acrylic
- Adhesives
- Fishing wire, silicone etc.

Electronics & Sensing

Motor Controllers

Sensors

- Load cells
- Force sensitive resistors
- Hall-effect
- Ultrasound
- Resistive/current
- Buttons
- Limit switches

Misc.

LEDs

Resistors and other electronics

Control Strategy

Much flexibility in software (may be dependent on sensors)

Information System (Computation)

Arduino Microcontroller

Consult the parts list

Mechatronic Product

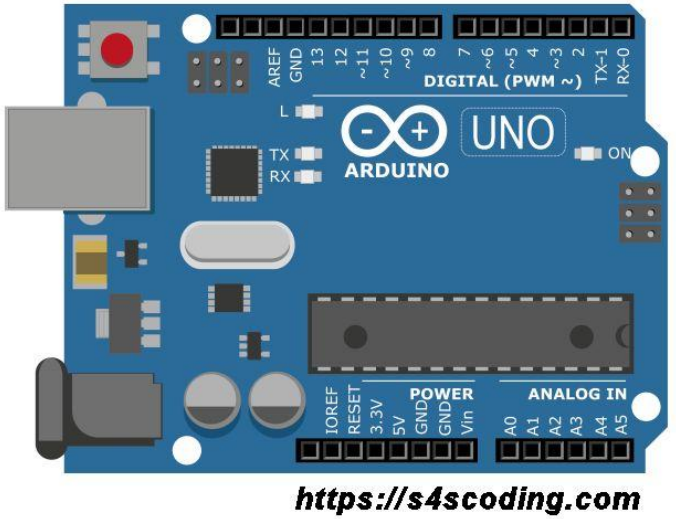
Control Strategy

Mechanical System & Actuation

Electronics & Sensing

Information System (Computation)

Single Board Computer vs Micro-controller



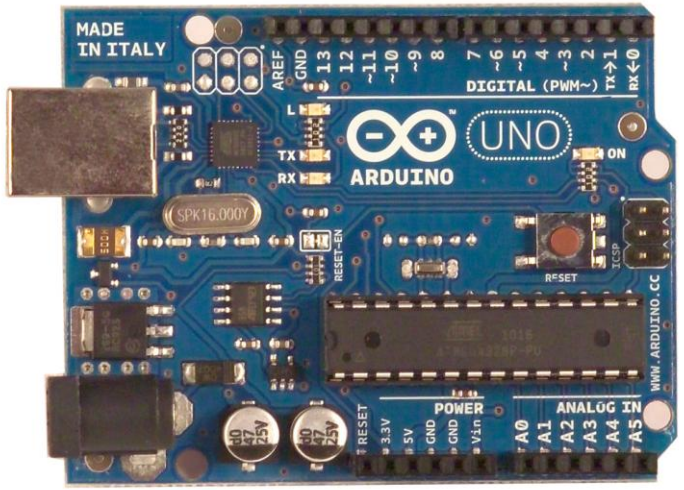
- The Raspberry Pi is a general purpose computer, more often than not running under a Linux Operating System (OS).
- Raspberry Pi's can run multiple complex programs.

- The Arduino Uno is a microcontroller board which is a simpler computer (compared to the Raspberry Pi) and can run one program at a time.

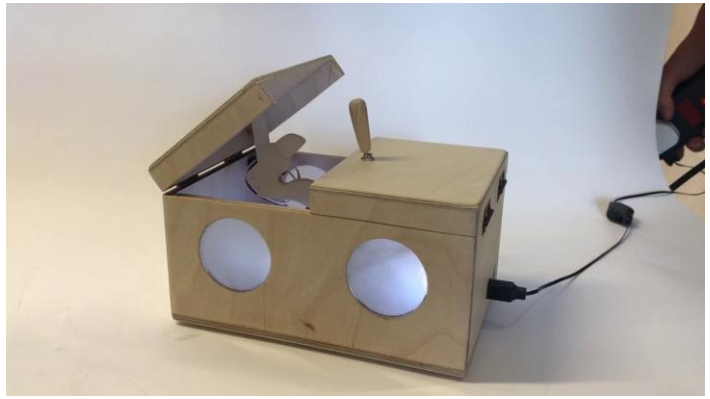
Micro-Controllers

Arduino

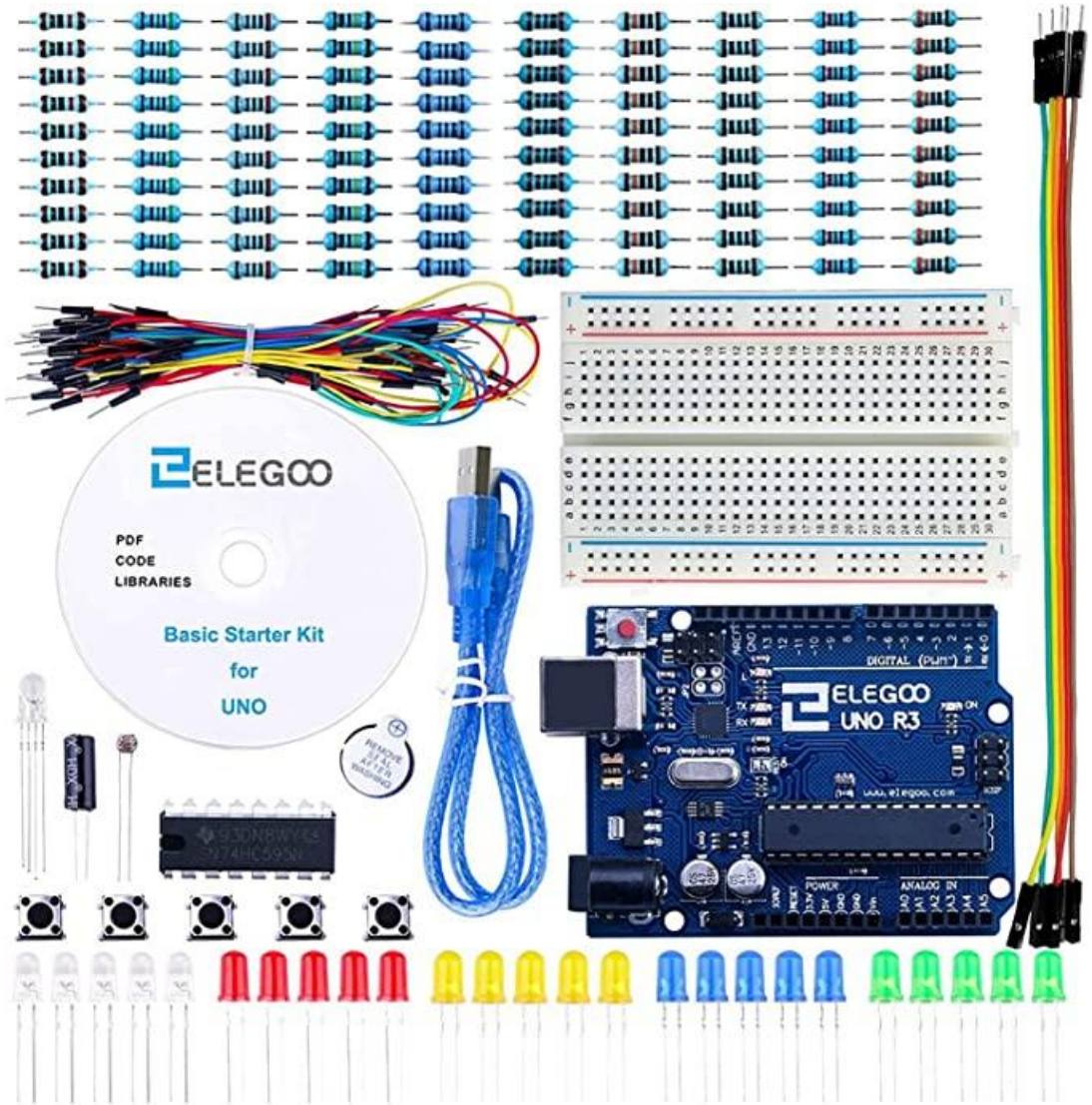
- **Open Source** electronic prototyping **platform** based on flexible **easy to use** hardware and software.



Widely used for prototyping…



Micro-Controllers



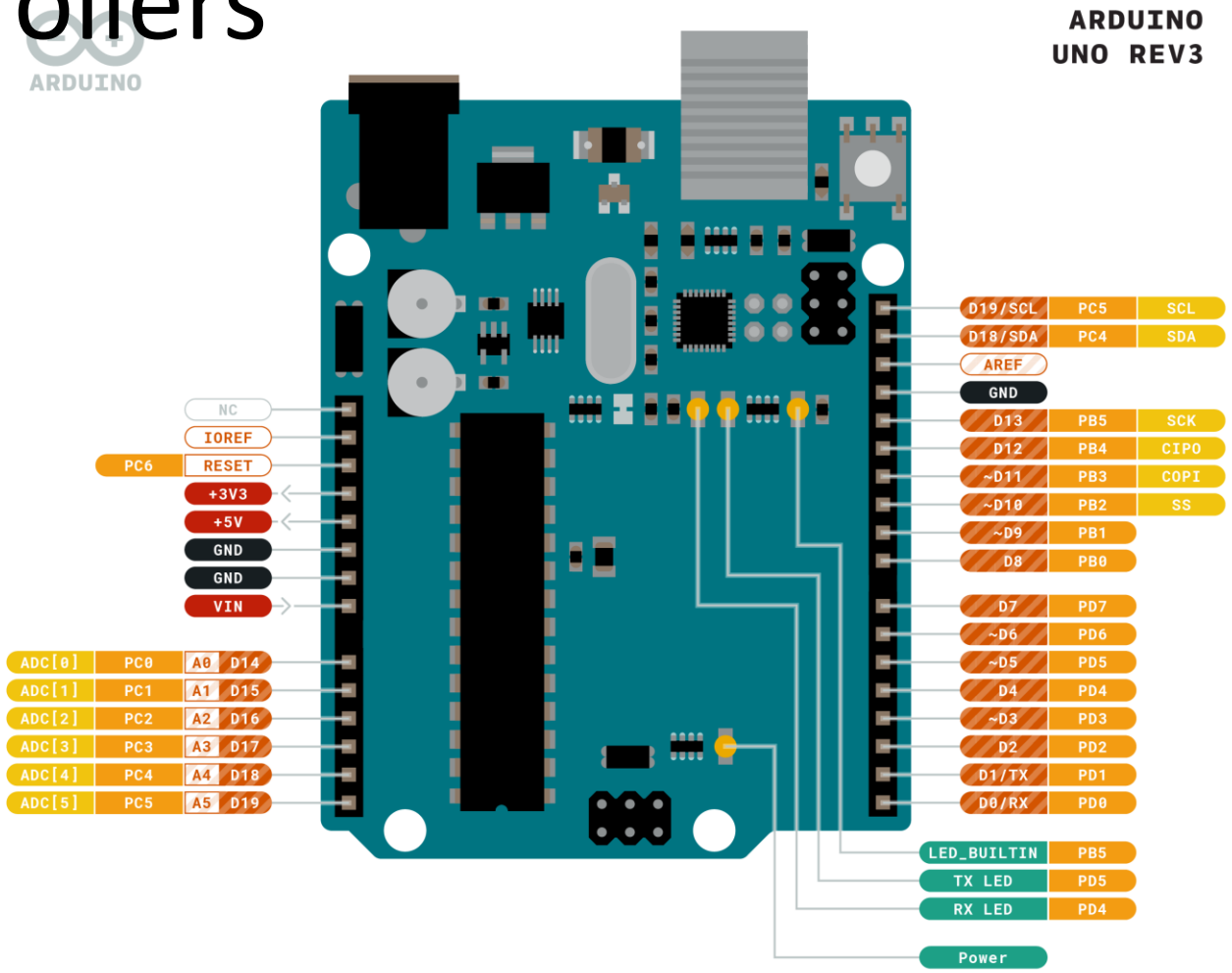
Starter Kit

- Can request 2 per team
- Only one should be on the robot
- It is expected that the main parts are returned in working order!

Micro-Controllers

Output voltages/GND

Analog Inputs – measures input voltages between 0-5V

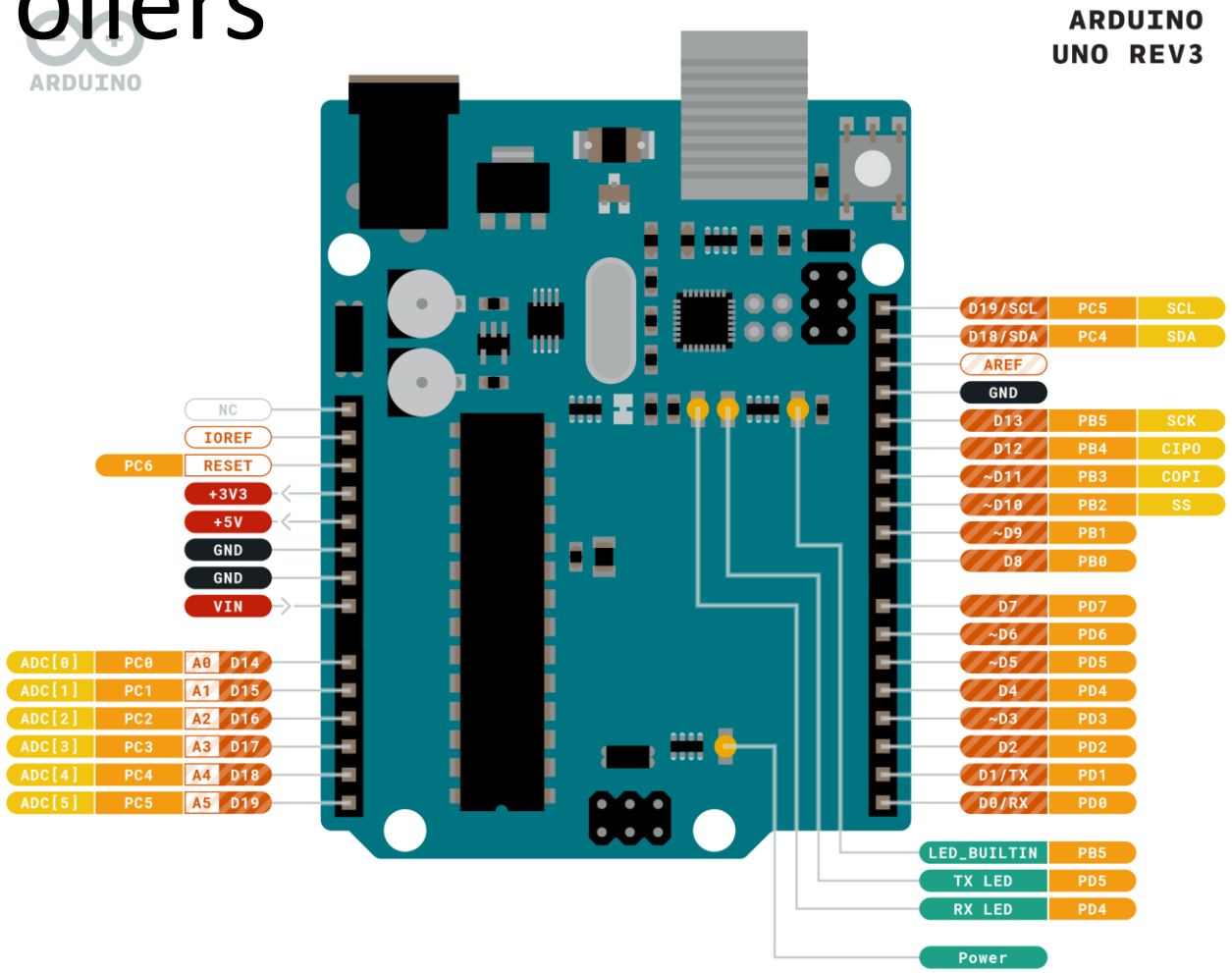


Digital Pins (can be inputs or outputs)

Micro-Controllers

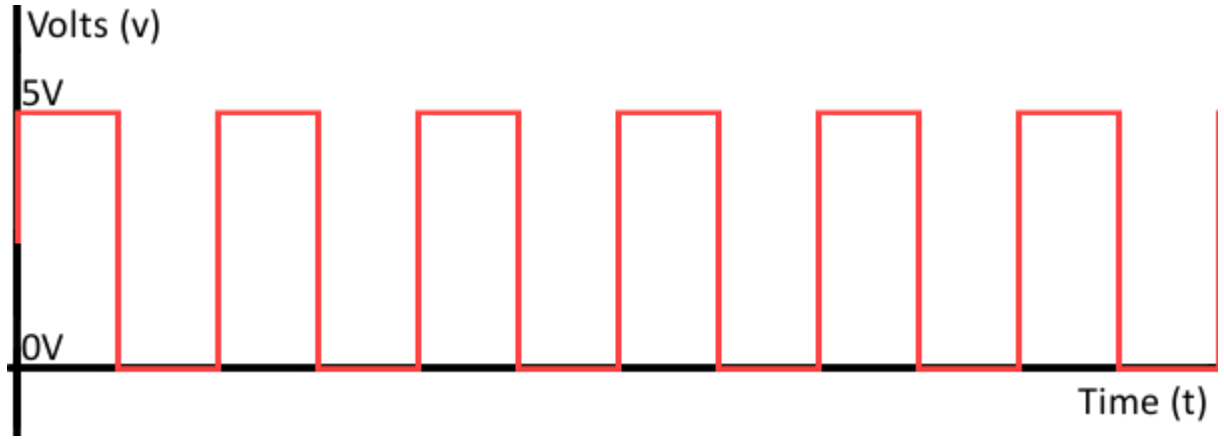
Output voltages/GND

Analog Inputs – measures input voltages between 0-5V



Digital Pins (can be inputs or outputs)

Digital Inputs & Outputs

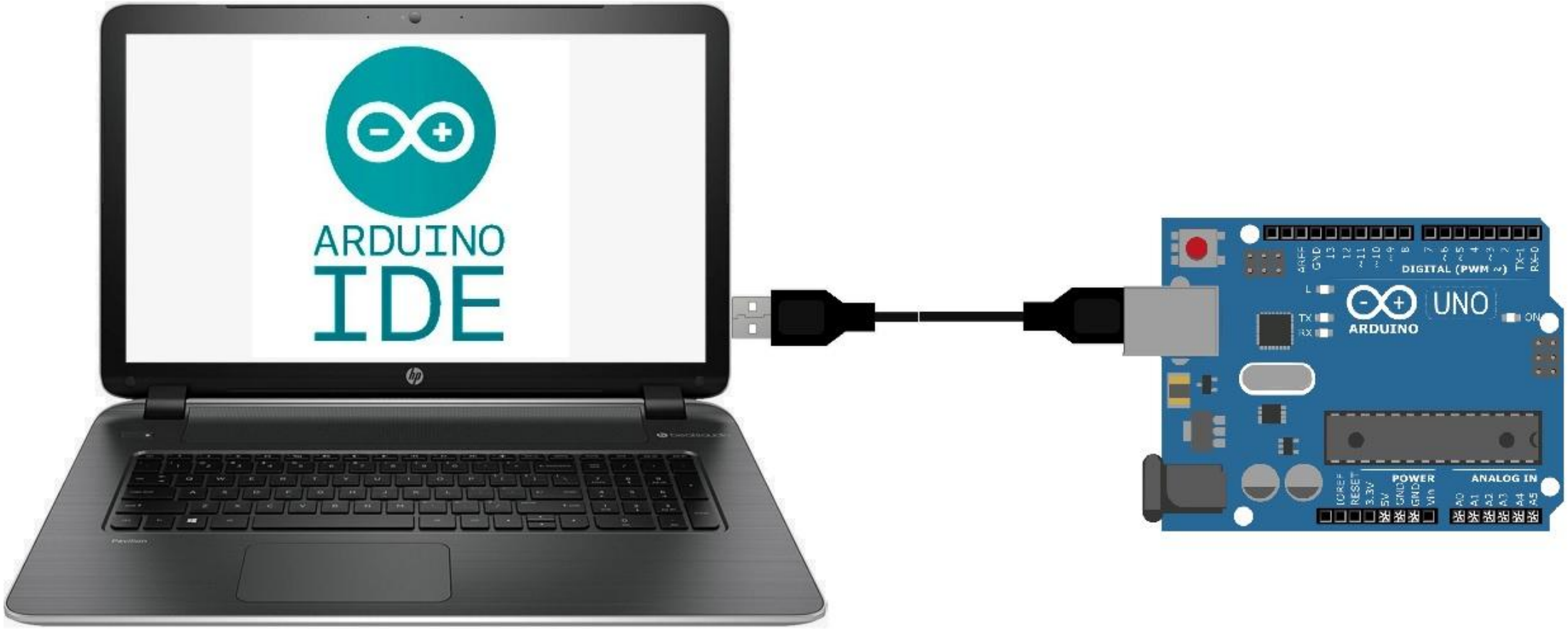


- Pins can be configured as an input/output
  - High is >3.3V – identified as a 1
  - Low is < 1.2V – identified as a 0
- The voltage (or potential) must be relative to ground you can't measure without a fixed ground signal
- Can have very high frequency switching

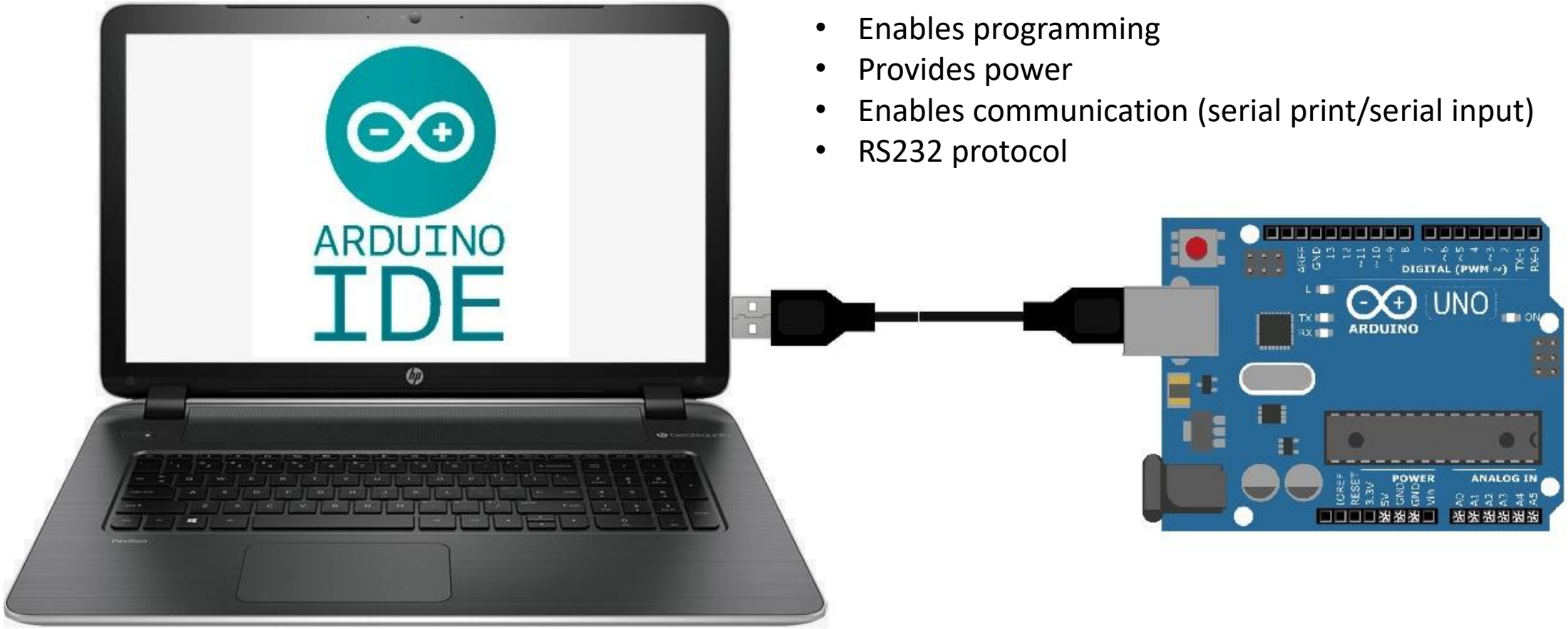
- Can only **source** low output current (can't drive high current devices directly)
- Can't **sink** lots of input current
  - Sink/source limit: 40mA

https://docs.arduino.cc/learn/microcontrollers/digital-pins

Connecting to your Arduino

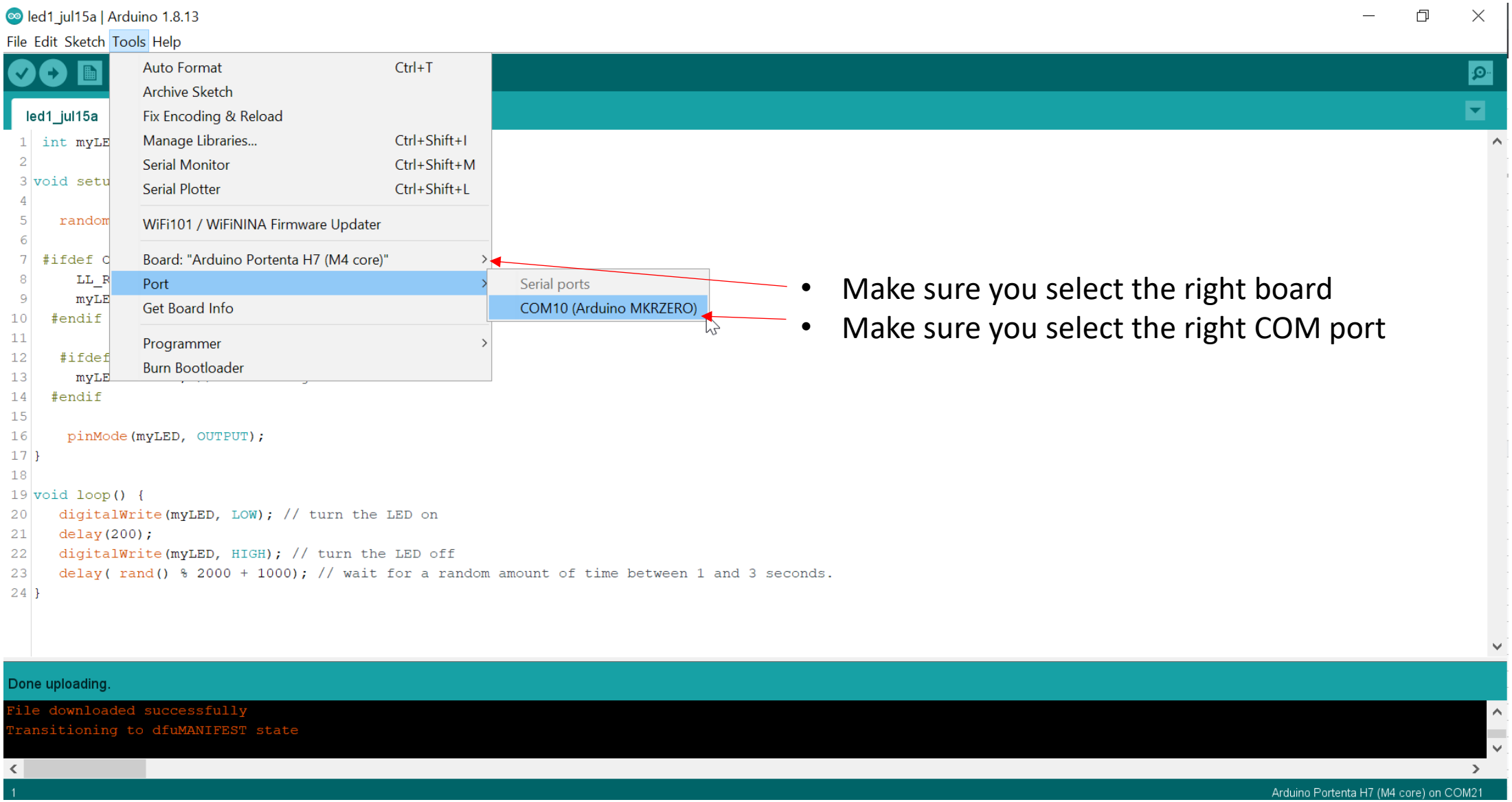


Connecting to your Arduino



USB-Serial Communication

- Enables programming
- Provides power
- Enables communication (serial print/serial input)
- RS232 protocol



- Make sure you select the right board
- Make sure you select the right COM port

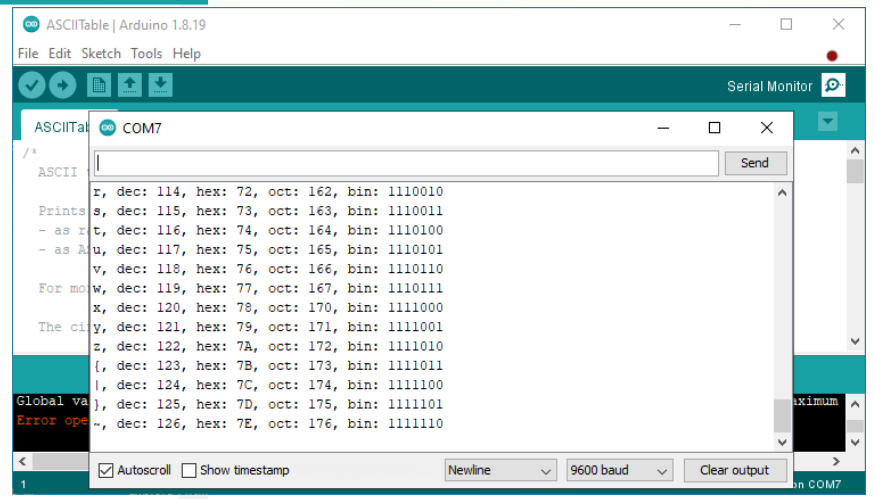
Bare minimum code

```
void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
}
```


Compile (check code!)

Download

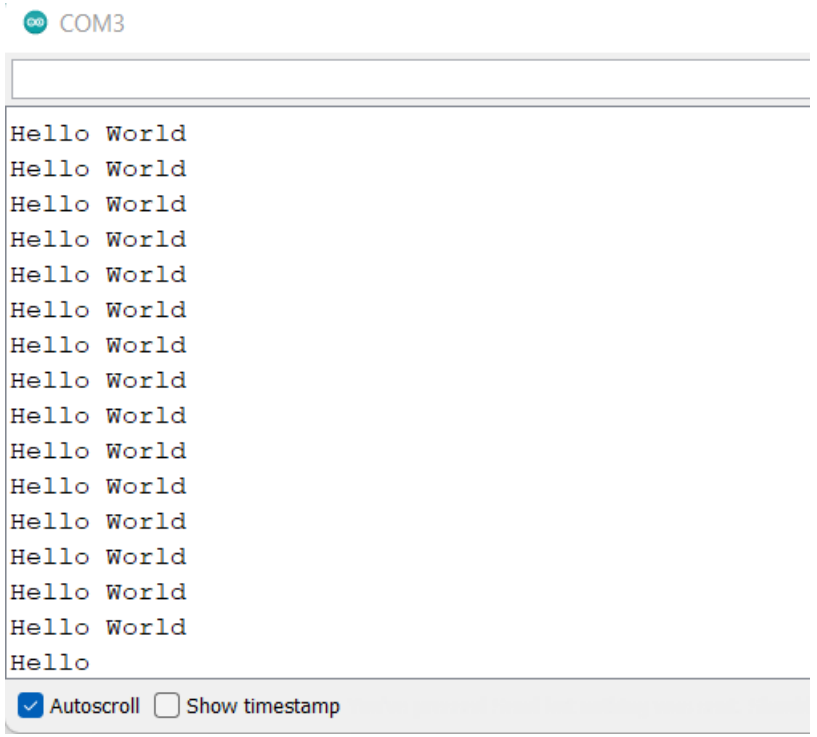
View serial monitor



Orange bugging/error messages
(Catch syntax errors or hardware errors)

Arduino: Serial Comms.

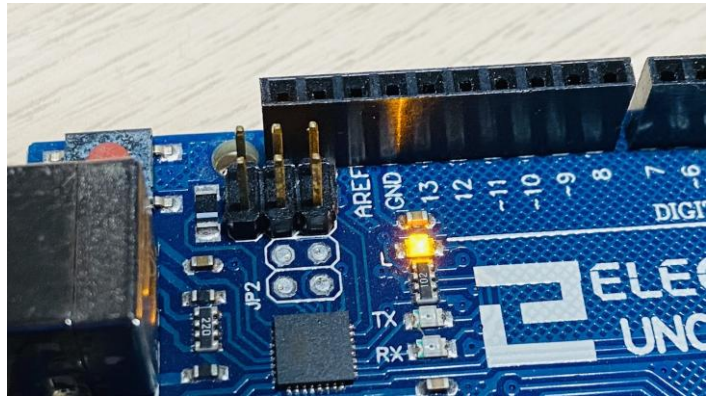
```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
}
```



- Useful for debugging
- Useful for printing and control (two ways) e.g. to state which object is detected
- What happens if I put the println in the setup()?

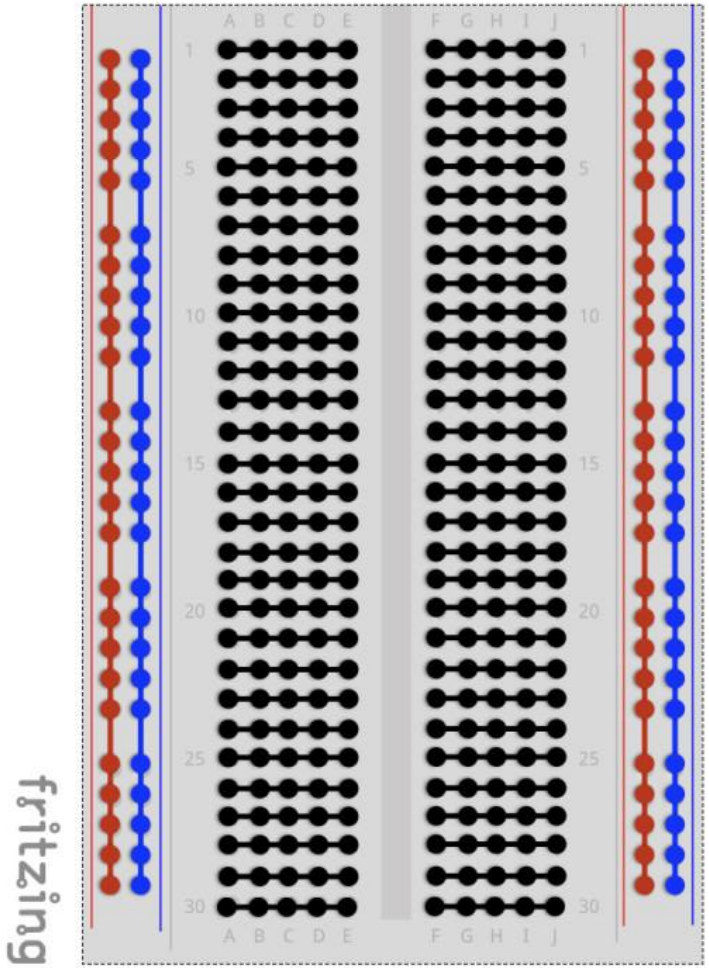
Arduino: LED (internal) switching

```
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                // wait for a second
}
```

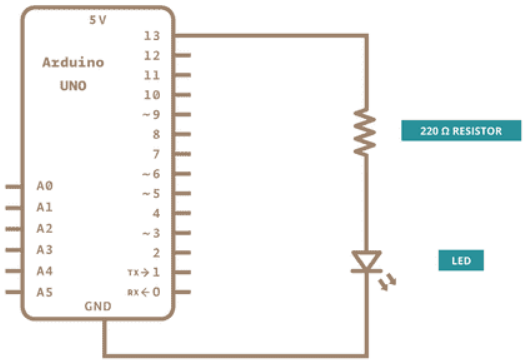


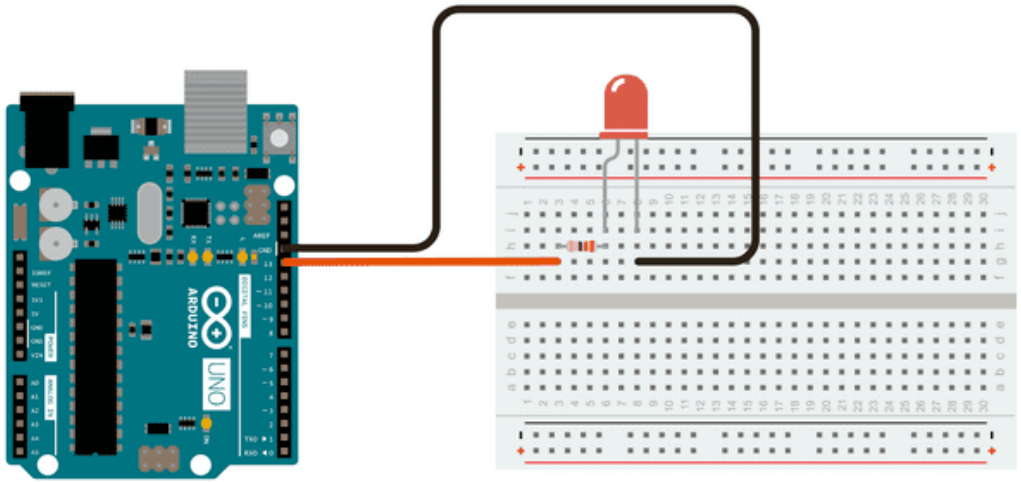
- Useful for debugging
- Example of a digital output

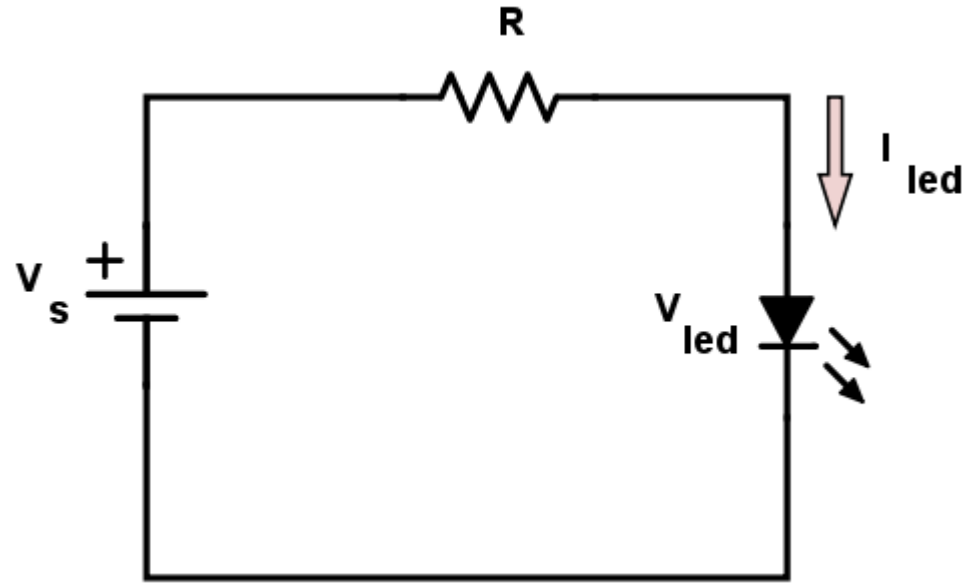
Prototyping Electronics: Breadboard



Control external LEDs

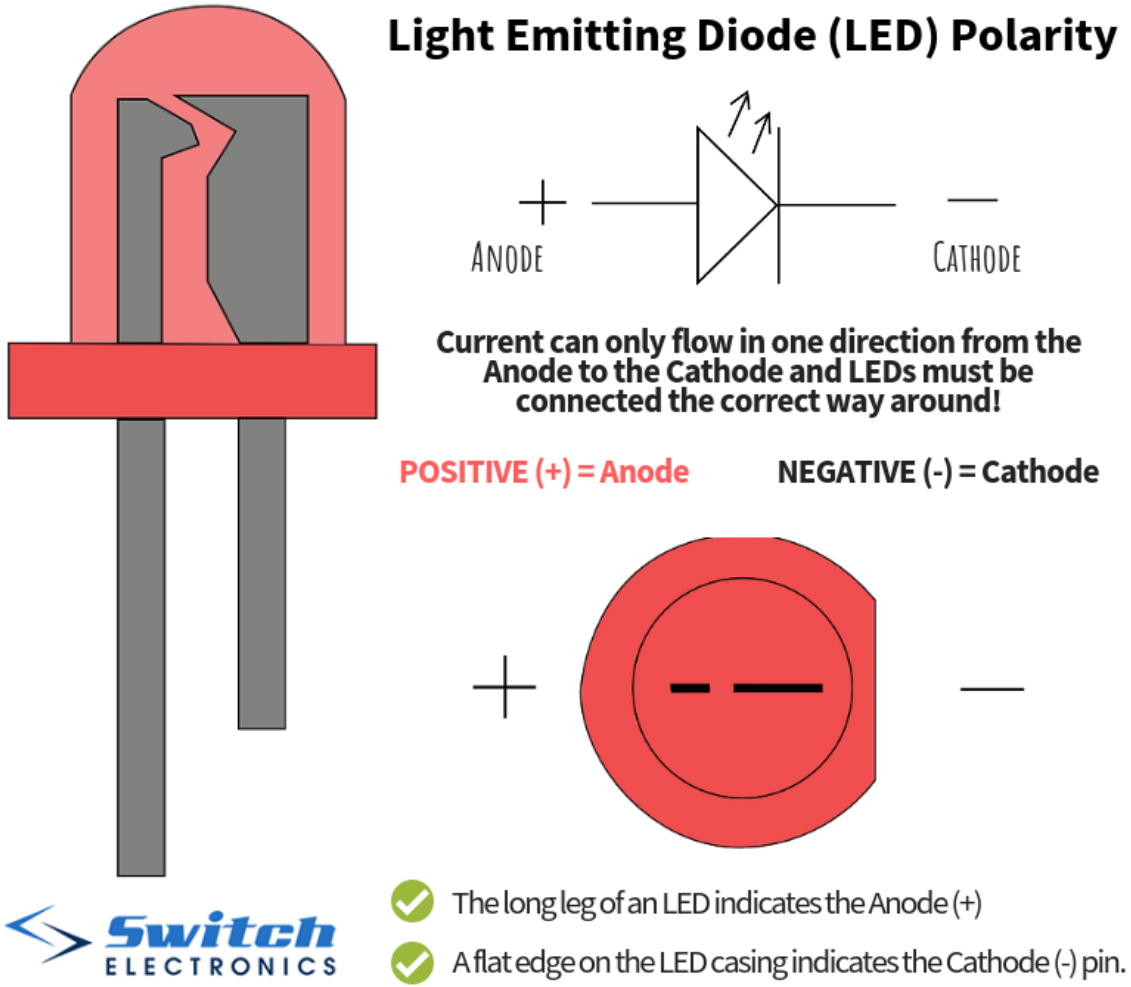






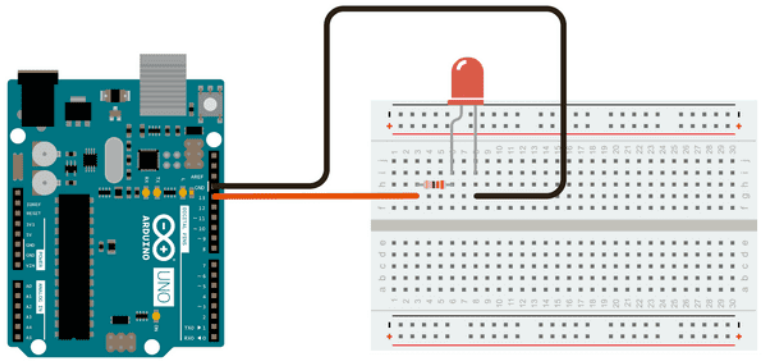
- Need resistor in series with LED for current limiting
  - V= IR (I =V/R) → max current that can be supplied is 40mA
- LED must go the correct way around! It has a polarity

Control external LEDs

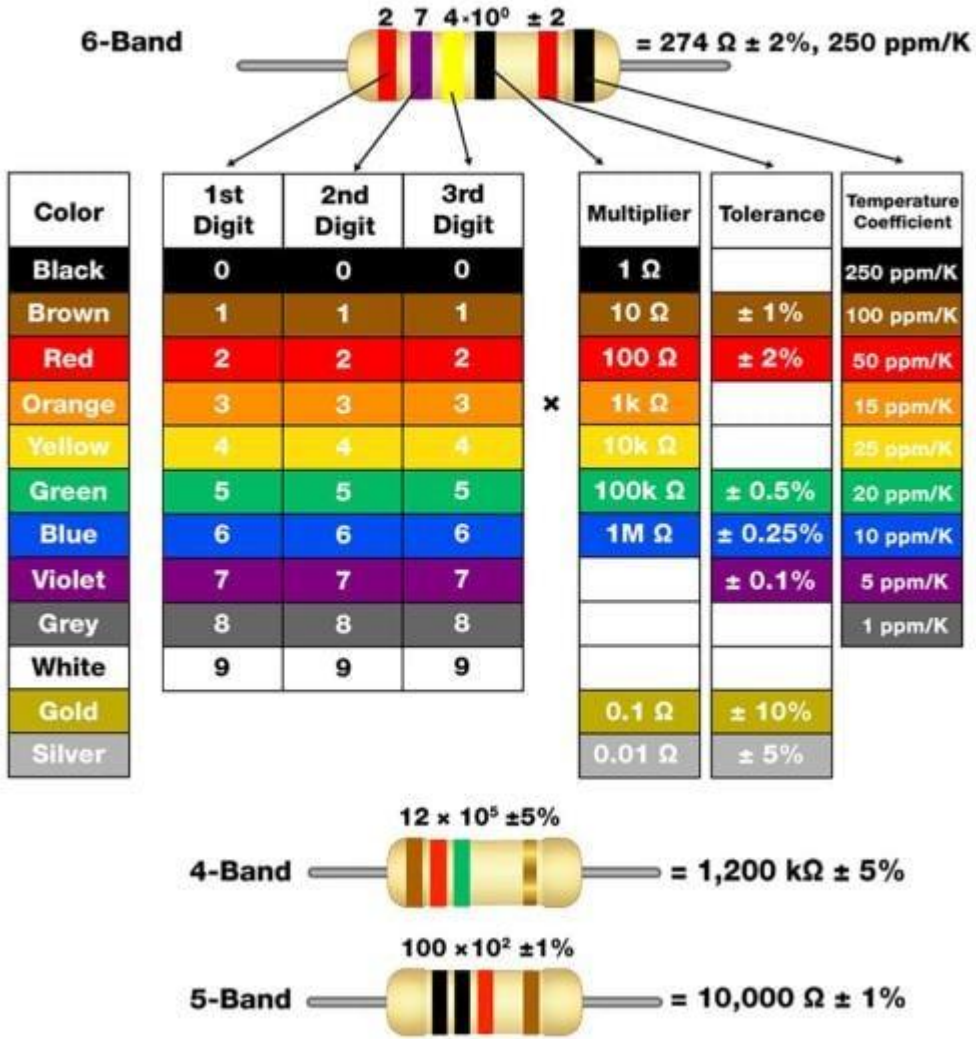


Control external LEDs

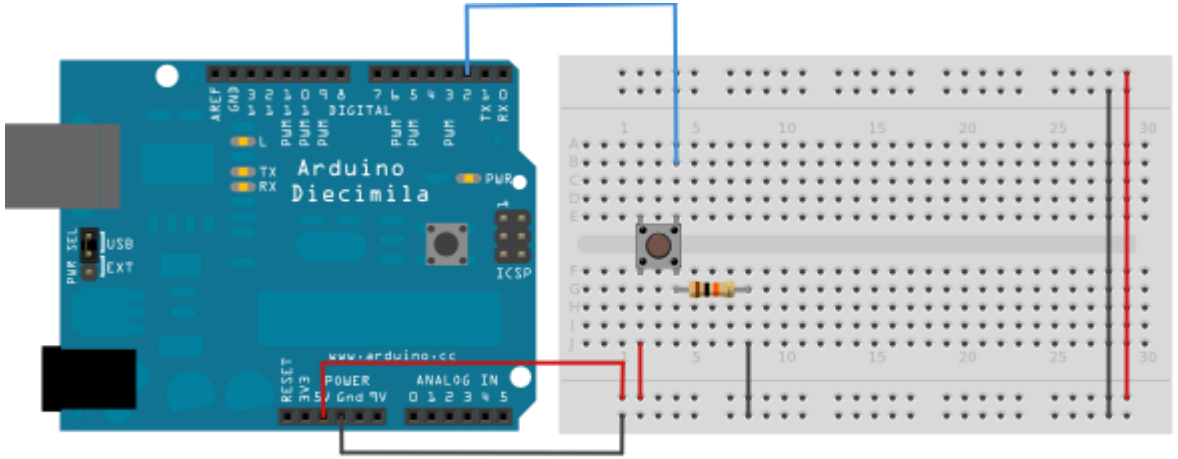
```
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                // wait for a second
}
```

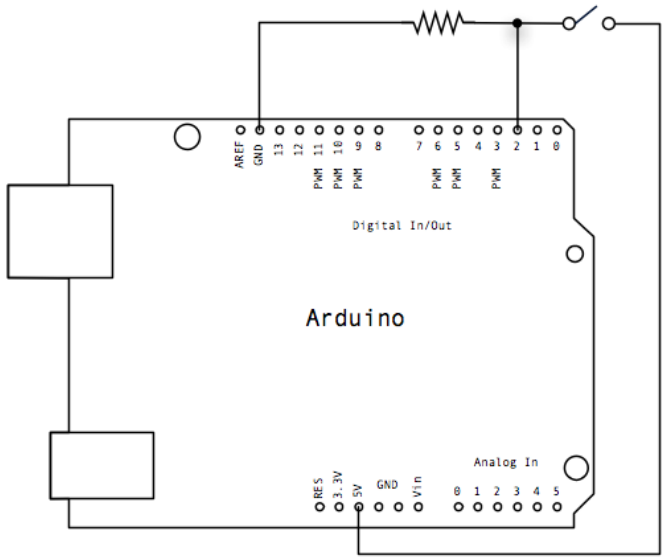


How to Read Resistor Color Codes



Read from Button





Pull down resistor

- Need resistor in series with the series so don't sink too much current
  - V= IR (I =V/R) → max current that can be sunk is 40mA
- The input can never be floating – i.e. you need the tie to ground even when switch is open.

```
const int buttonPin = 2;          // the number of the pushbutton pin
const int ledPin = LED_BUILTIN;   // the number of the LED pin
  // LED_BUILTIN is set to the correct LED pin independent of which board is used

// variables will change:
int buttonState = 0;          // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```


Exercise: Introduction to Arduino's

Task 1: Setup Arduino and perform 'hello world' with Serial Print

Task 2: Program LED internal to the board (change the rate of flashing – slow/fast)

Task 3: Setup 3 LEDs on the breadboard and set to flash at different times

Task 4: Detect button press and print to serial monitor

Task 5: Detect button press and switch on LED (Digital input and output)

<u>Resources:</u>

- These slides (example code + how to connect
- Arduino.cc
- Instructables and others

Next Week: Design Review

Design Review Sessions

Join in the SPOT!

Time	Table 1	Table 2	Table 3	Table 4	Table 5	Table 6	Table 7
8:20	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
8:40	Group 8	Group 9	Group 10	Group 11	Group 12	Group 13	Group 14
9:00	Group 15	Group 16	Group 17	Group 18	Group 19	Group 20	Group 21
9:20	Group 22	Group 23	Group 24	Group 25	Group 26	Group 27	Group 28
9:40	Group 29	Group 30	Group 31	Group 32	Group 33	Group 34	Group 35
10:00	Group 36	Group 37	Group 38	Group 39	Group 40	Group 41	Group 42
10:20	Group 43	Group 44	Group 45	Group 46	Group 47	Group 48	Group 49
10:40	Group 50	Group 51					

- Schedule can be found on moodle.
- We be held in the SPOT
- Please be on time!