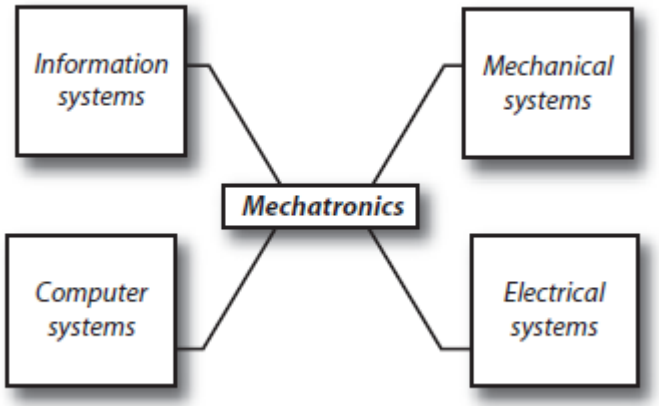
Product development & engineering design

ME-320

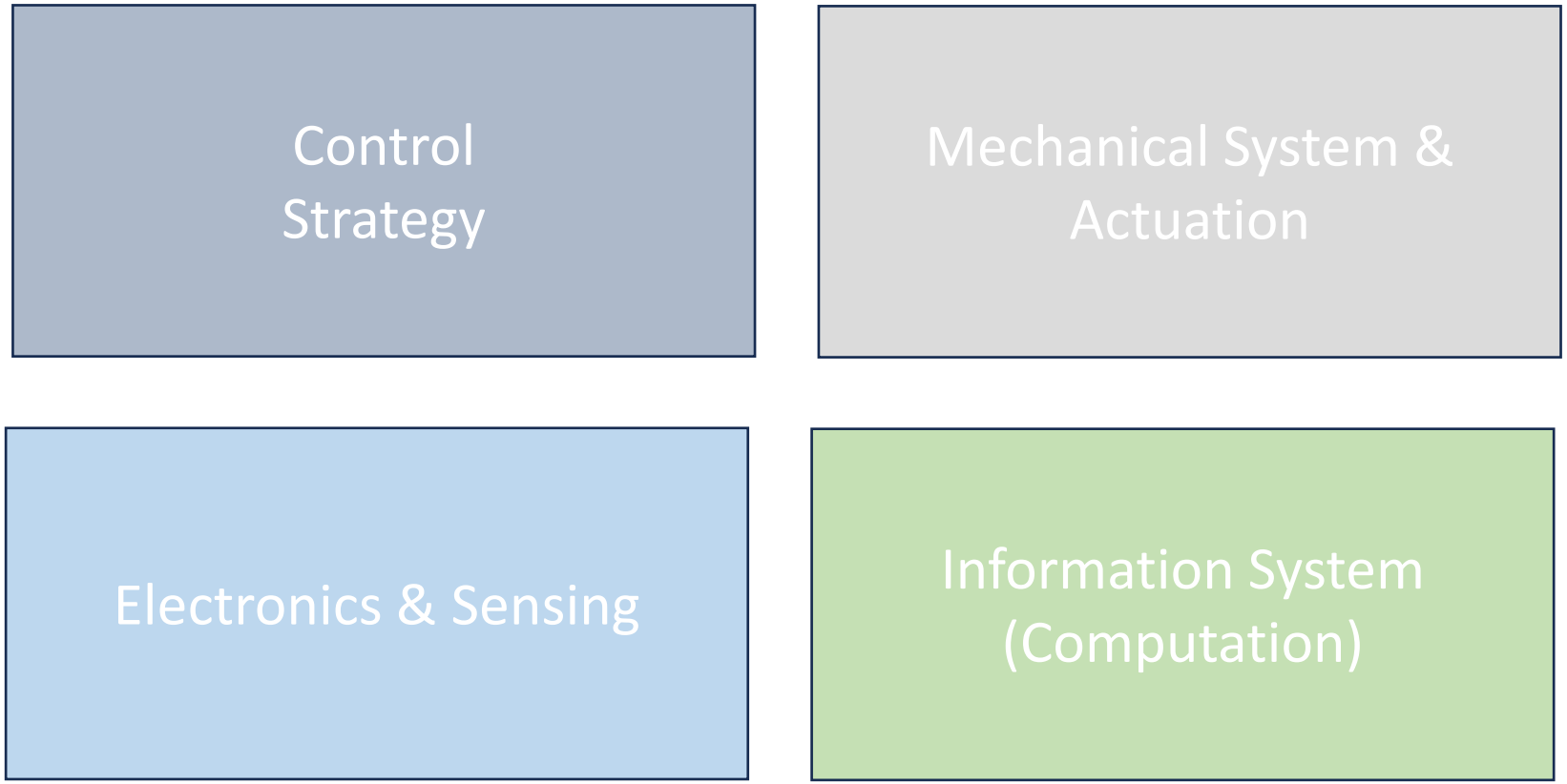
PROF. JOSIE HUGHES

Lecture 5: i) Microcontrollers
i) Motors
ii) Sensors

Mechatronic Product



Mechatronic Product



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

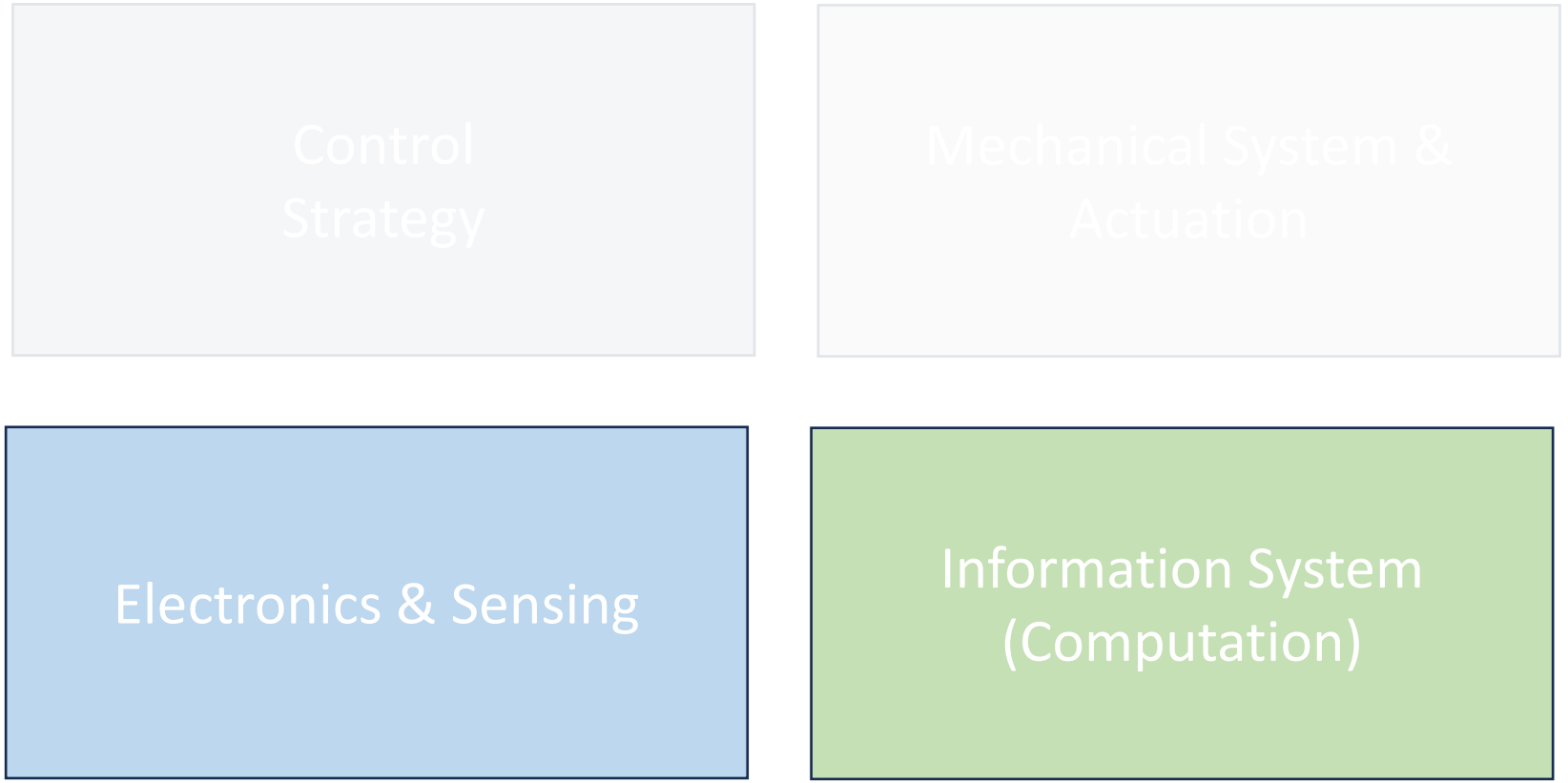
Much flexibility in software (may be dependent on sensors)

Information System (Computation)

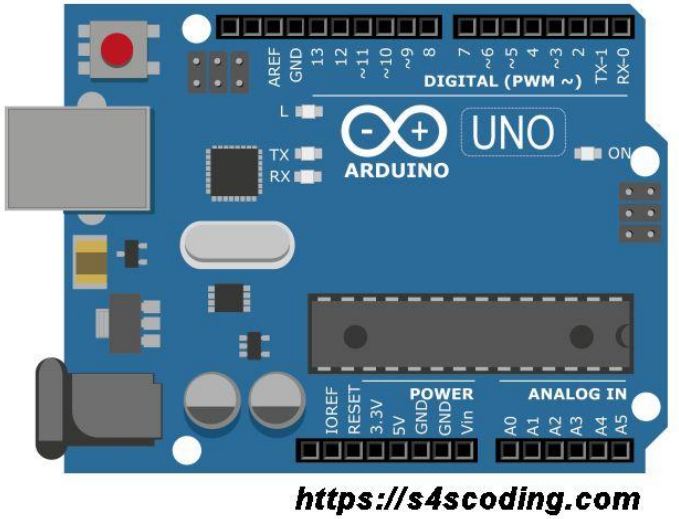
Arduino Microcontroller

Consult the parts list

Mechatronic Product



Single Board Computer vs Micro-controller



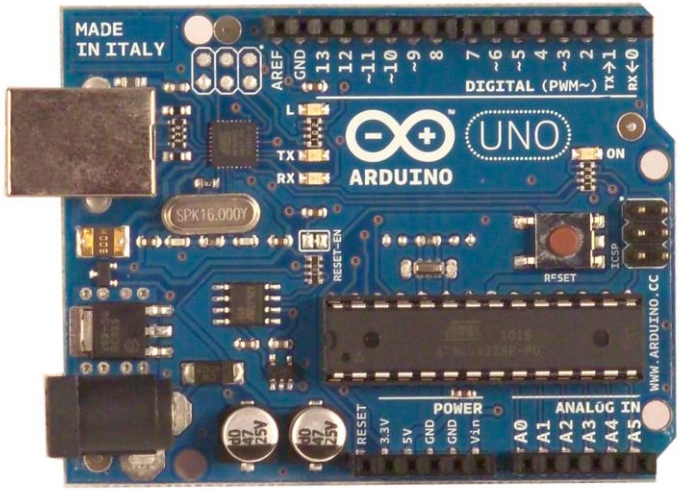
- The Raspberry Pi is a general purpose computer, more often than not running under a Linux Operating System (OS).
- Raspberry Pi's can run multiple complex programs.

- 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.

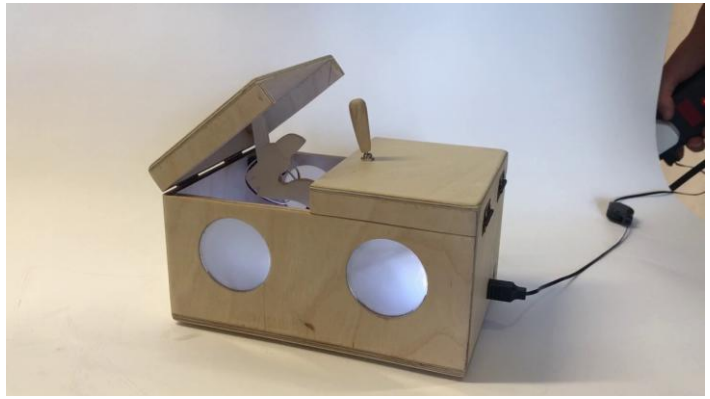
Micro-Controllers

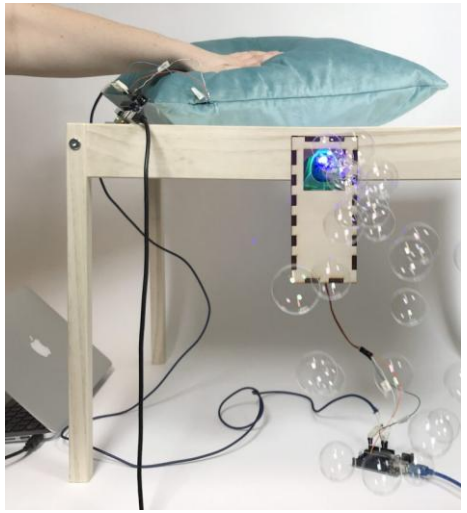
Arduino

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

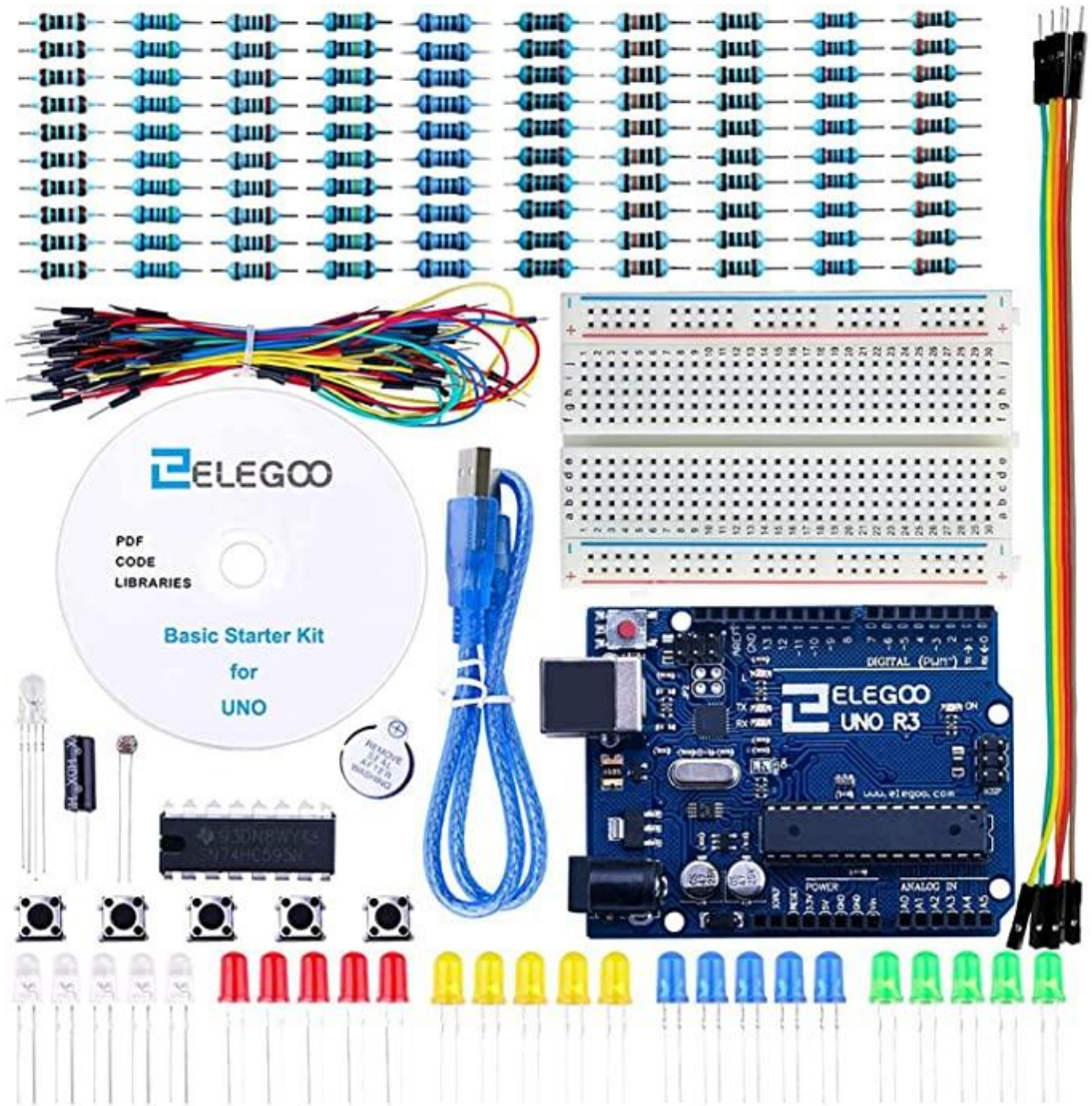


Widely used for prototyping…





Micro-Controllers

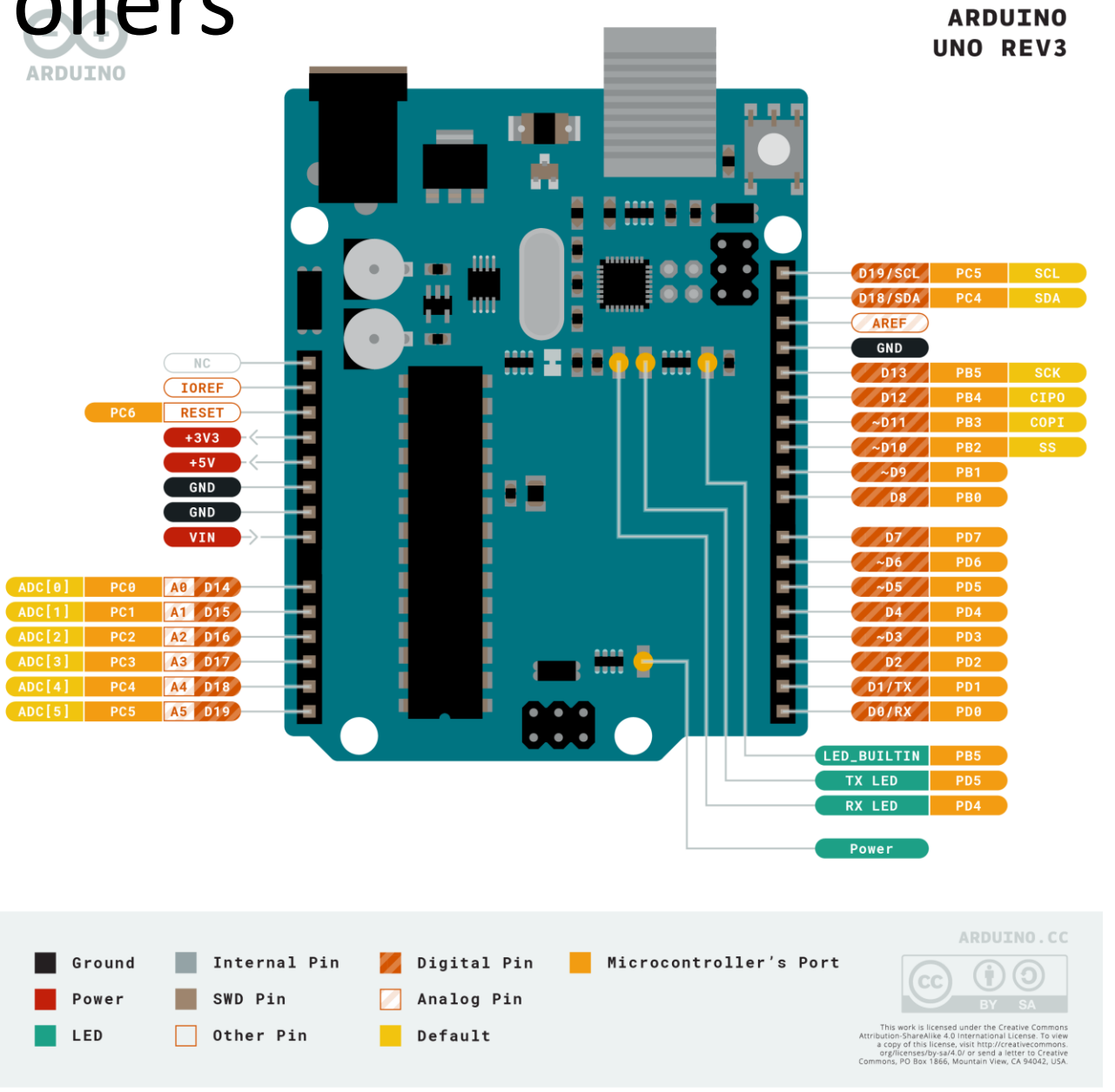


Starter Kit

Micro-Controllers

Output voltages/GND

Analog Inputs – measures input voltages between 0-5V

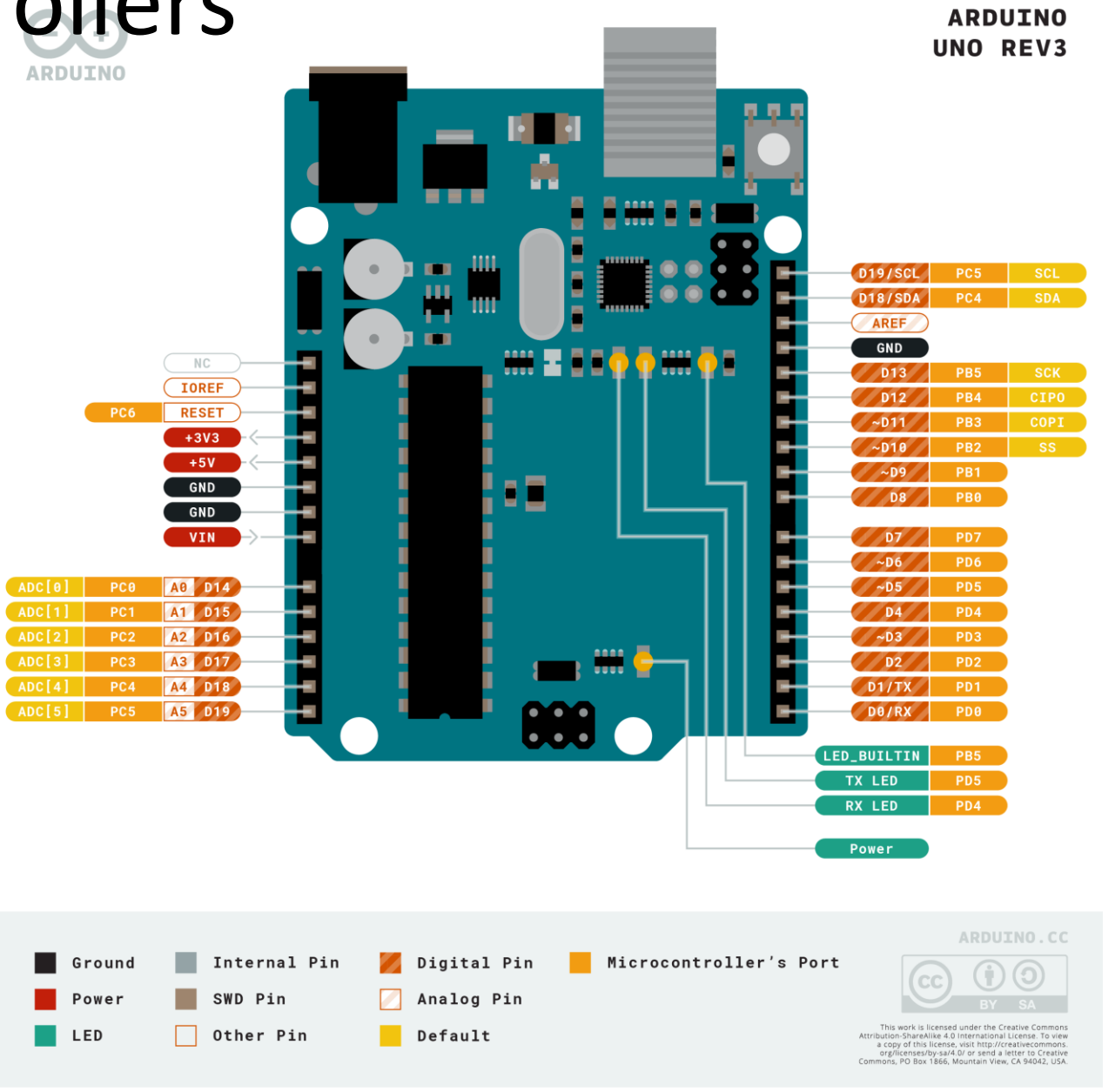


Digital Pins (can be inputs or outputs)

Micro-Controllers

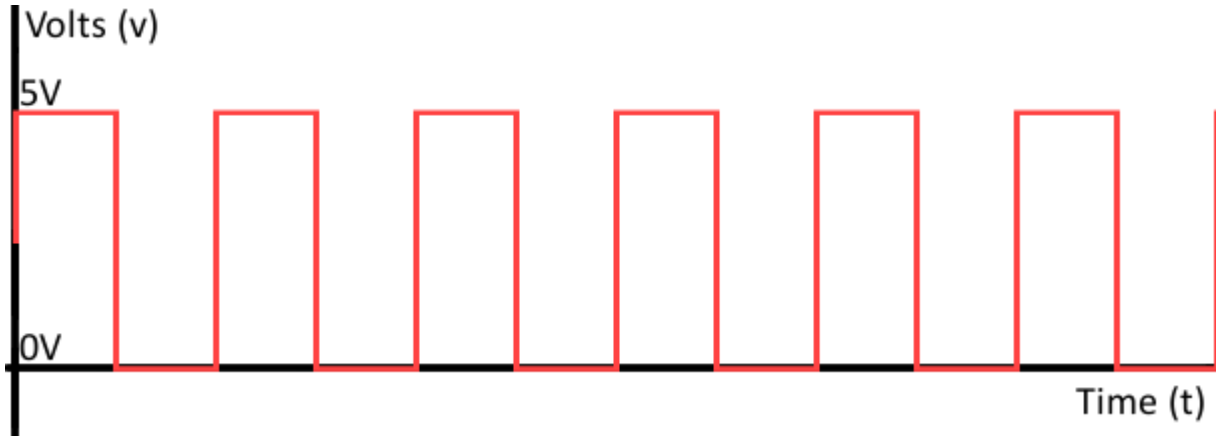
Output voltages/GND

Analog Inputs – measures input voltages between 0-5V



Digital Pins (can be inputs or outputs)

Digital Inputs & Outputs

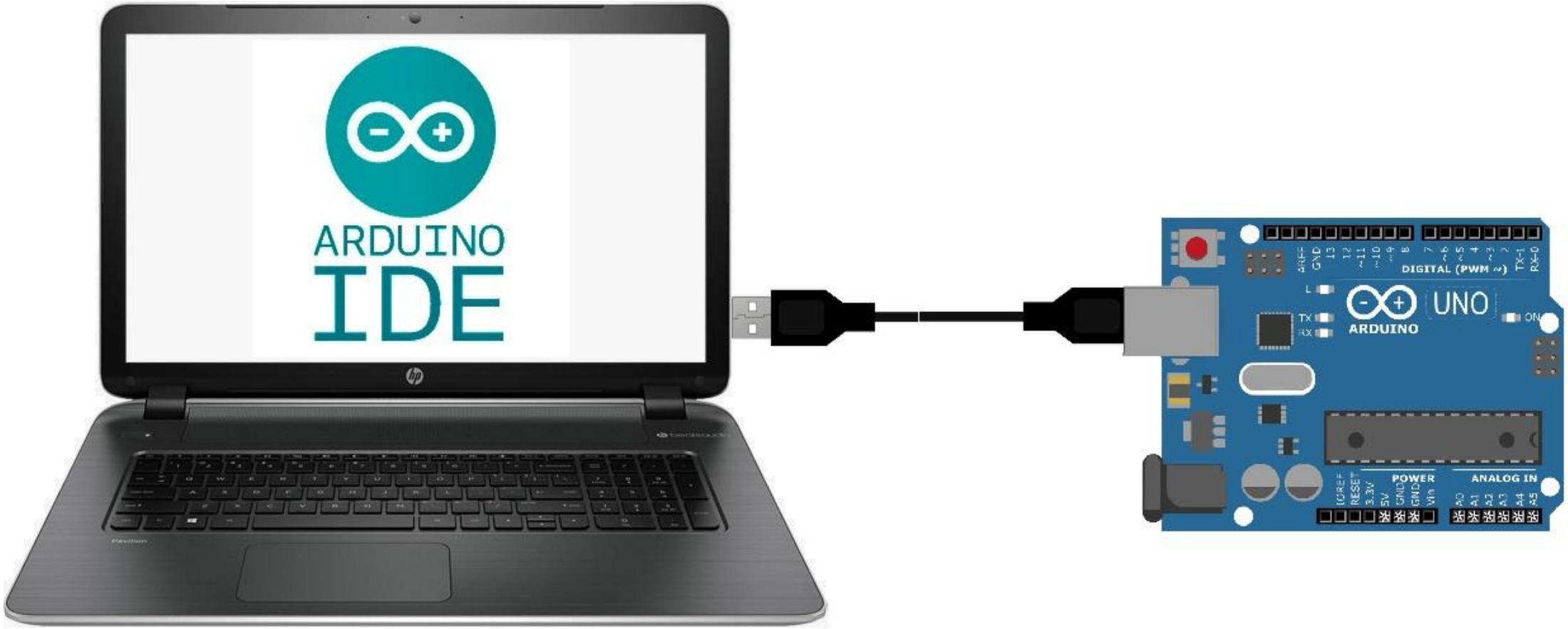


- Pins can be configured as an input/output
  - High is >3.3V – identified as a 1
  - Low is < 1.2V – identified as a 0
- The voltage (or potential) must be relative to ground you can't measure without a fixed ground signal
- Can have very high frequency switching

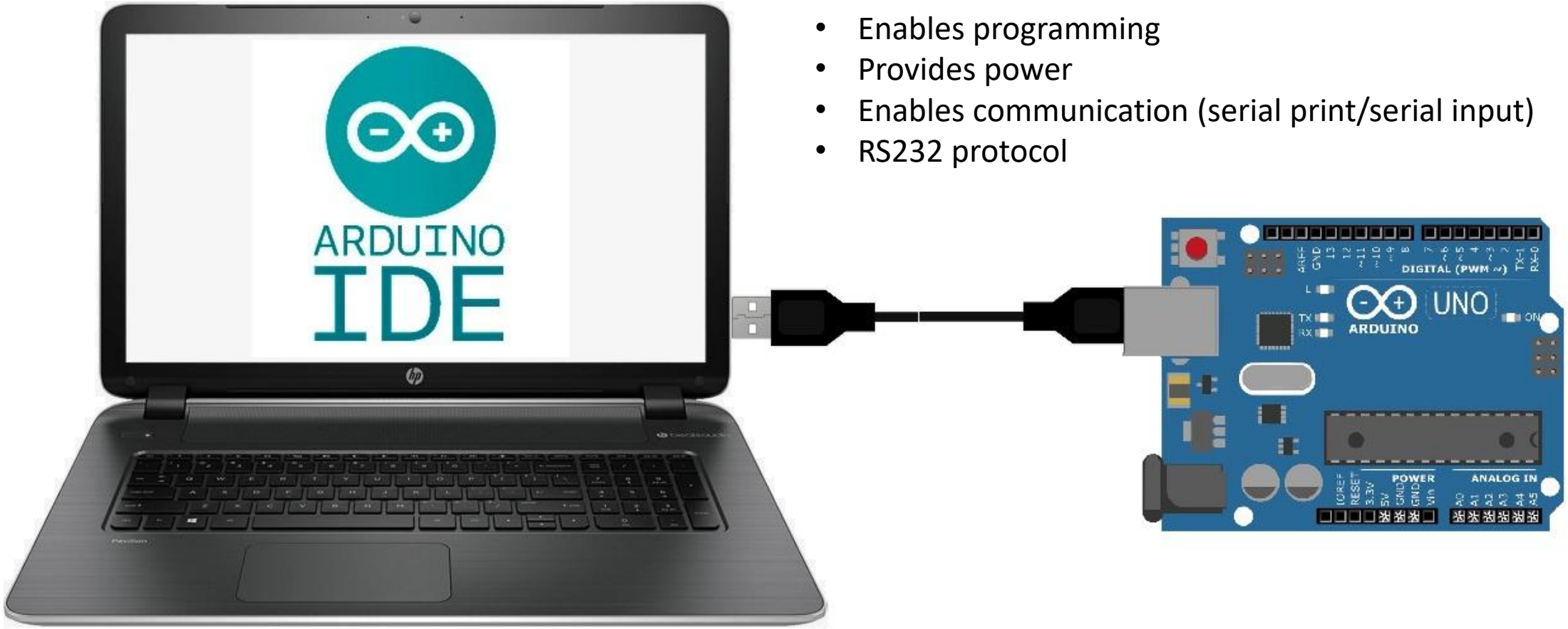
- Can only **source** low output current (can't drive high current devices directly)
- Can't **sink** lots of input current
  - Sink/source limit: 40mA

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

Connecting to your Arduino

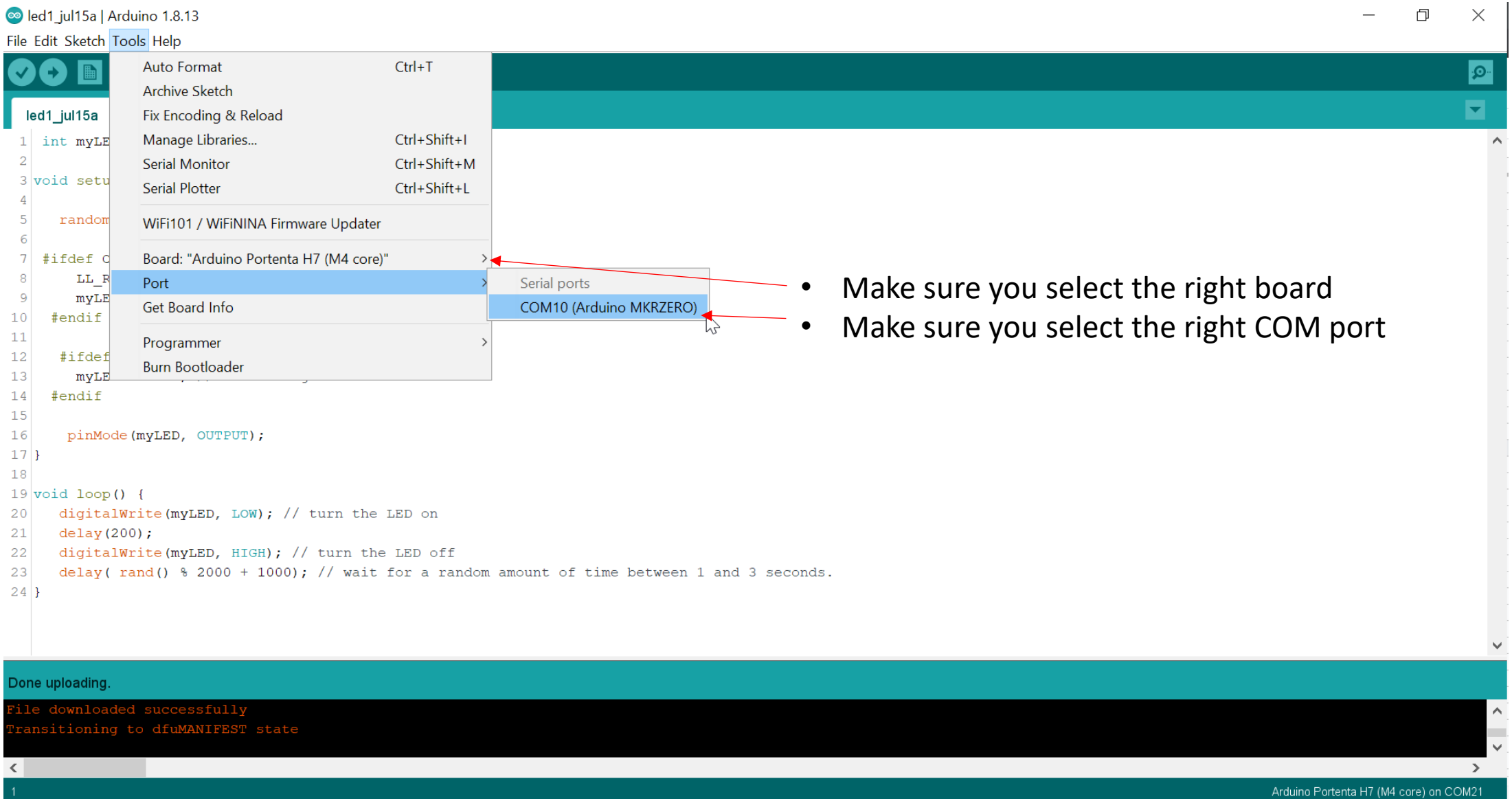


Connecting to your Arduino



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

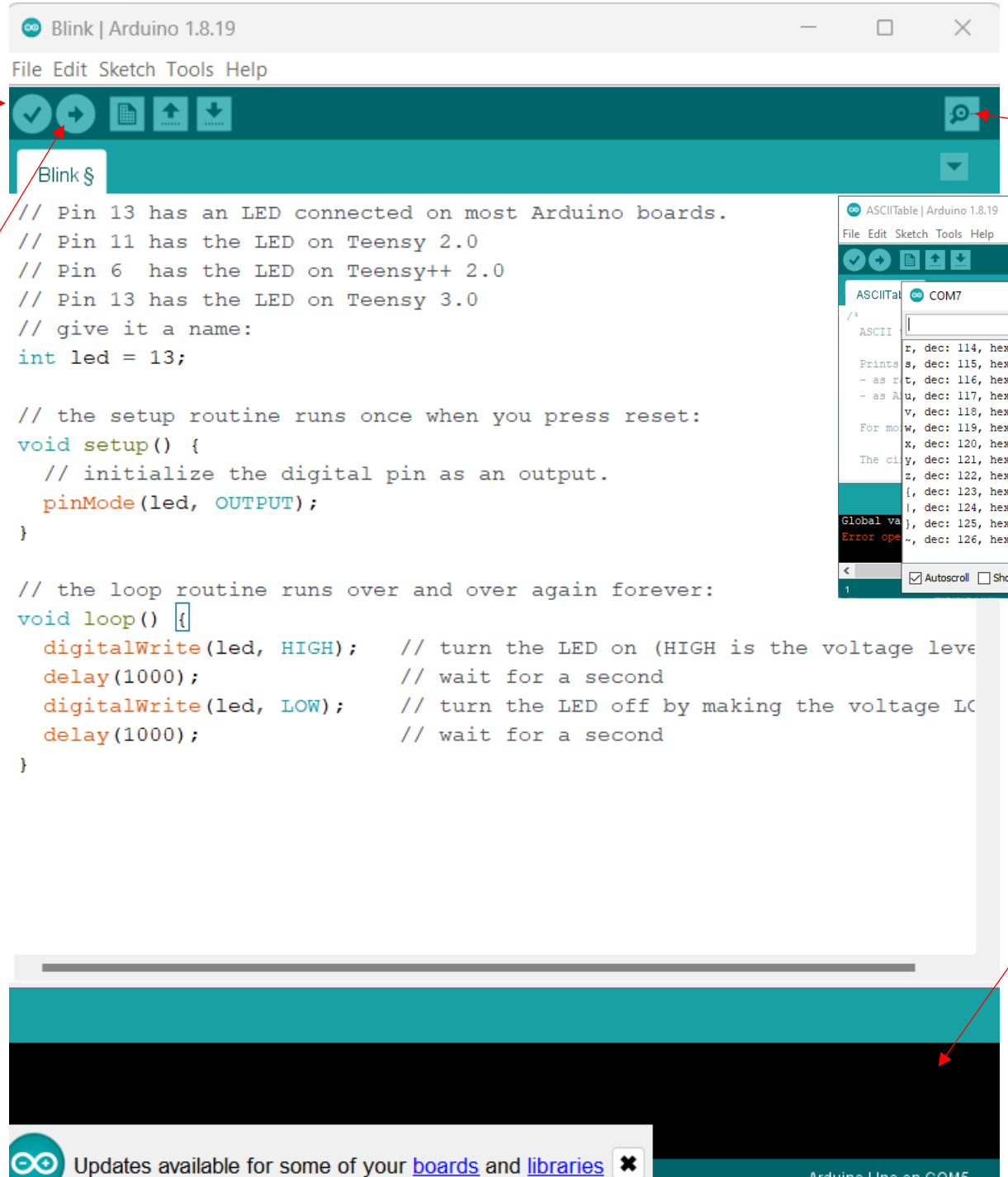
Bare minimum code

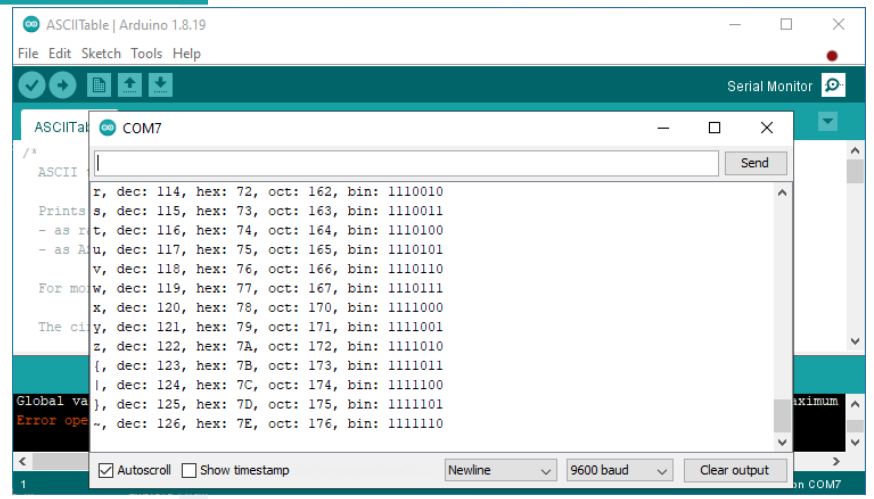
```
void setup() {
    // put your setup code here, to run once:
}

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)

Having problems/debug

1) Press the reset button the Arduio (restarts the code)
2) Use ‘serial print’ so you see where or what is being processed
3) Power down and then power up!

How to actuate Mechanisms: Actuator Selection

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

https://ttpoll.eu/p/datadriven

Actuators: Summary

"An actuator is **a component of a machine that is responsible for moving and controlling a mechanism or system**"

…many different actuator types

Electric actuators: motors, solenoids

Pneumatic actuators: Utilize compressed air for the driving force

Hydraulic actuators: Use incompressible fluid to amplify the control signal

We are going to focus on electrical ones

Electric Actuators

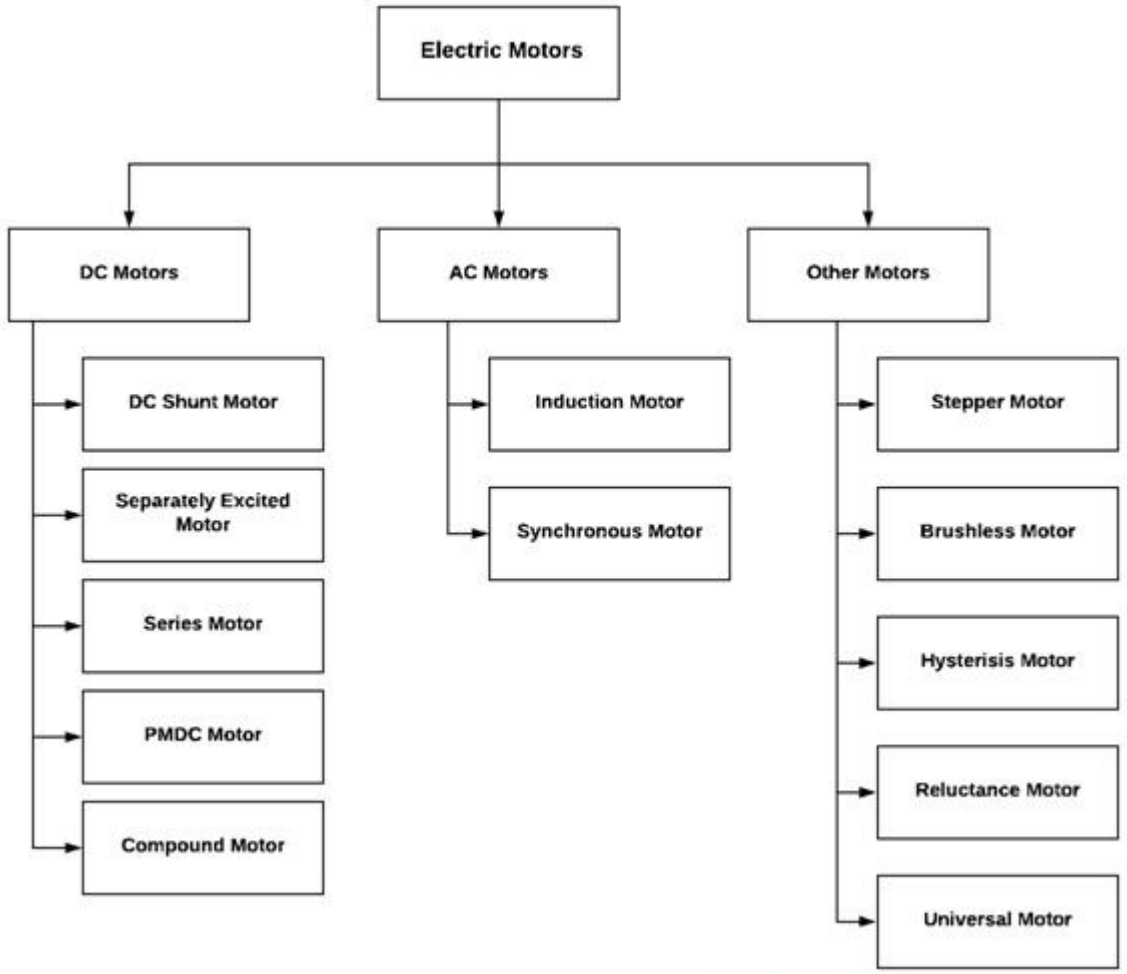
DC Motors

Servos

Stepper Motors

All provide rotary motion. Driven systems are then required to translate rotary motion to other forms of motion!

There are many different types of motor…



Motors

- How do they operate?
- Characteristics of the motor (torque, speed)
- Typical use cases
- Advantages/disadvantages
- How to control

End goal:

- Which actuator to choose for a given application
- How to find the properties of a motor

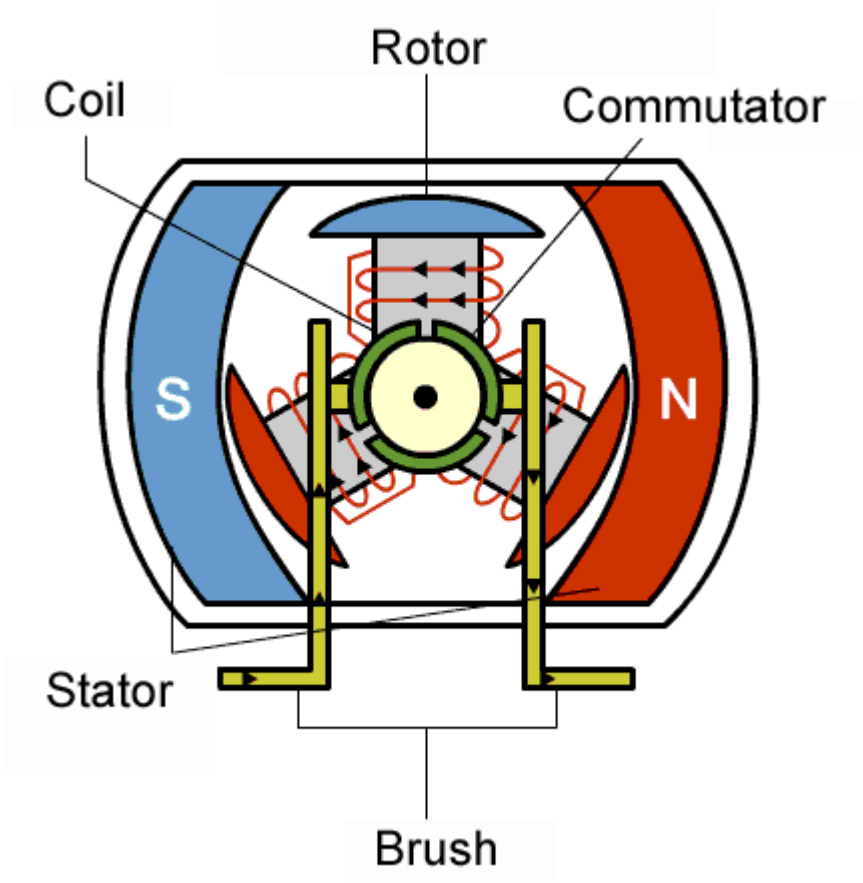
DC Motors: Introduction

- DC motors convert electrical energy to mechanical energy (rotation)
- Exploit the interaction of magnetic fields and conductors
- There are many types of DC motors The brushed and brushless motors are the most common DC motors.

Brushed DC Motors

- The brushed DC motor has been around for a long time, and its use can be traced back to the 1830s.
- Found in toys, household appliances, computer cooling fans
- Simple to construct & control
- Use brushes to make contact with the rotating segment
- Can lead to friction, heat, sparks

Brushed DC Motors: Operation



STATOR: A permanent magnet or electromagnetic windings. Provides a stationary magnetic field.

ARMATURE (Rotor): Electric windings around armature arms that generate a magnetic field when energized by the external current. The magnetic poles generated are attracted to the opposite poles from the stator causing rotation.

COMMUTATOR: Segmentation sleeve to switch the current to different armature segments. The current +/- is supplied to this commutator segments with the help of brushes.

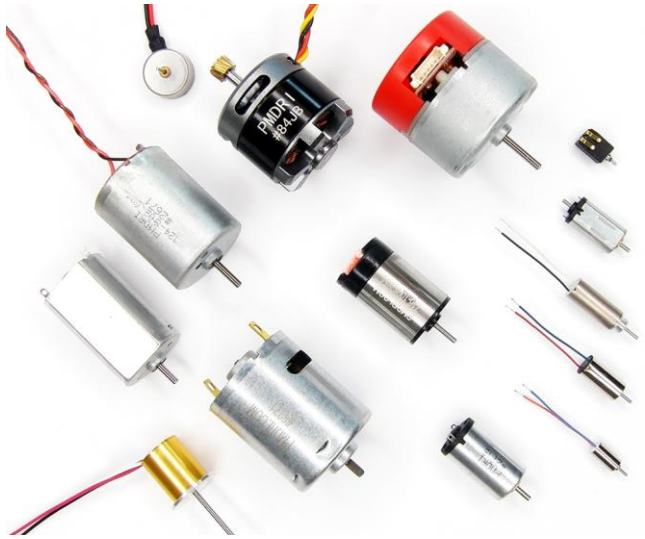
BRUSHES: slide over the commutator segments creating the variable magnetic field in different arms generating a dynamic magnetic field.

Brushed DC Motors: Pros & Cons

	Advantages
Simple to control	Controlling a brushed DC motor is as simple as a switch. Simply apply a voltage to start driving them. They slow down when the voltage is lowered, and spin in the other direction when the voltage is reversed.
Excellent torque at low speeds	High torque is achieved at low speeds.
Reasonably efficient	Brushed DC motors are about 75-80% efficient.
Inexpensive	DC motor costs can be as low as a few dollars

Brushed DC Motors: Pros & Cons

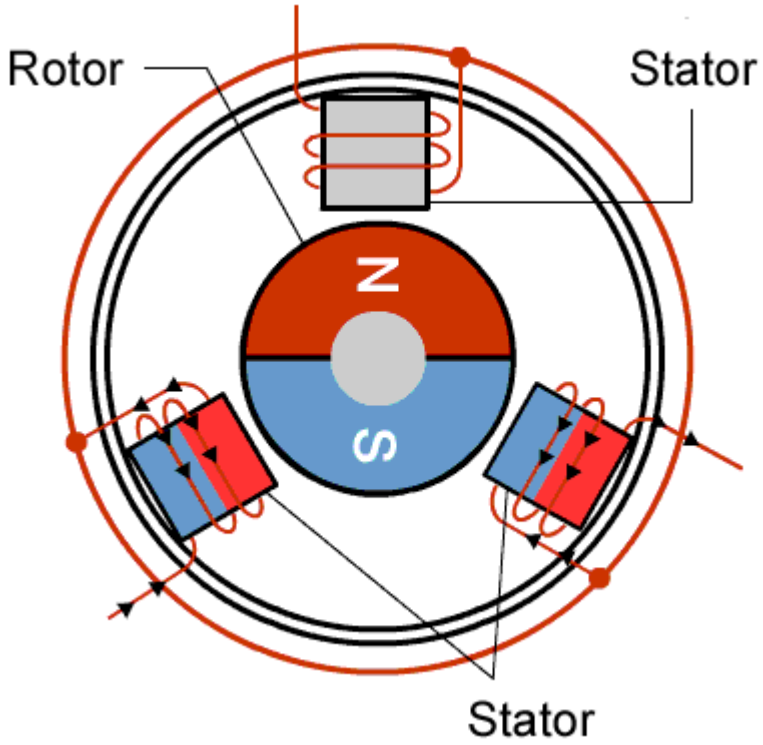
	Limitations
Noise	Aside from the audible noise from the rubbing parts, electromagnetic noise is also generated as a result of the strong sparks that occur at areas where the brushes pass over the gaps in the commutator. This can potentially cause interference in other parts of the system.
Constant Maintenance	Brushes could get easily worn out as a result of continuous moving contact and require constant maintenance. Speed could be limited due to brush heating.



Brushless DC Motors

- Brushless DC motors are mechanically simpler than brushed ones.
- Commutation is achieved electrically, the sparks and noise of brushed DC motors is eliminated,
- Electrical switching enables quiet switching
- These quiet motors find applications in computer fans, disk drives, drones, electric vehicles and high-precision servomechanisms.

Brushless DC Motors



The brushless DC motor only has one moving component – **the rotor.**

- The rotor consists of a ring of permanent magnets, whereas the coils are stationary. This set-up eliminates the need for brushes.
- The challenge comes in controlling the polarity of the current flowing through the coils and keeping this in sync with the speed of the rotor. This can be achieved by measuring back EMF or using Hall effect sensors to directly measure the position of the magnets.
- This can make DC motors more expensive and complex, despite their advantages.

Brushless DC Motors

	Advantages
Quiet	They generate less electrical noise compared to brushed motors as no brush is used. Hence, brushless DC motors are often preferred in applications where it is important to avoid electrical noise.
Efficient	Brushless DC motors are more efficient than brushed motor, as they are able to continuously achieve maximum rotational force/torque. Brushed motors in contrast, will reach maximum torque only at certain points during the rotation. For a brushed motor to achieve the same torque as a brushless motor, it would require a larger magnet.
Require less maintenance	Brushless DC motors offer high durability as there are no brushes to be replaced.

Brushless DC Motors: Applications

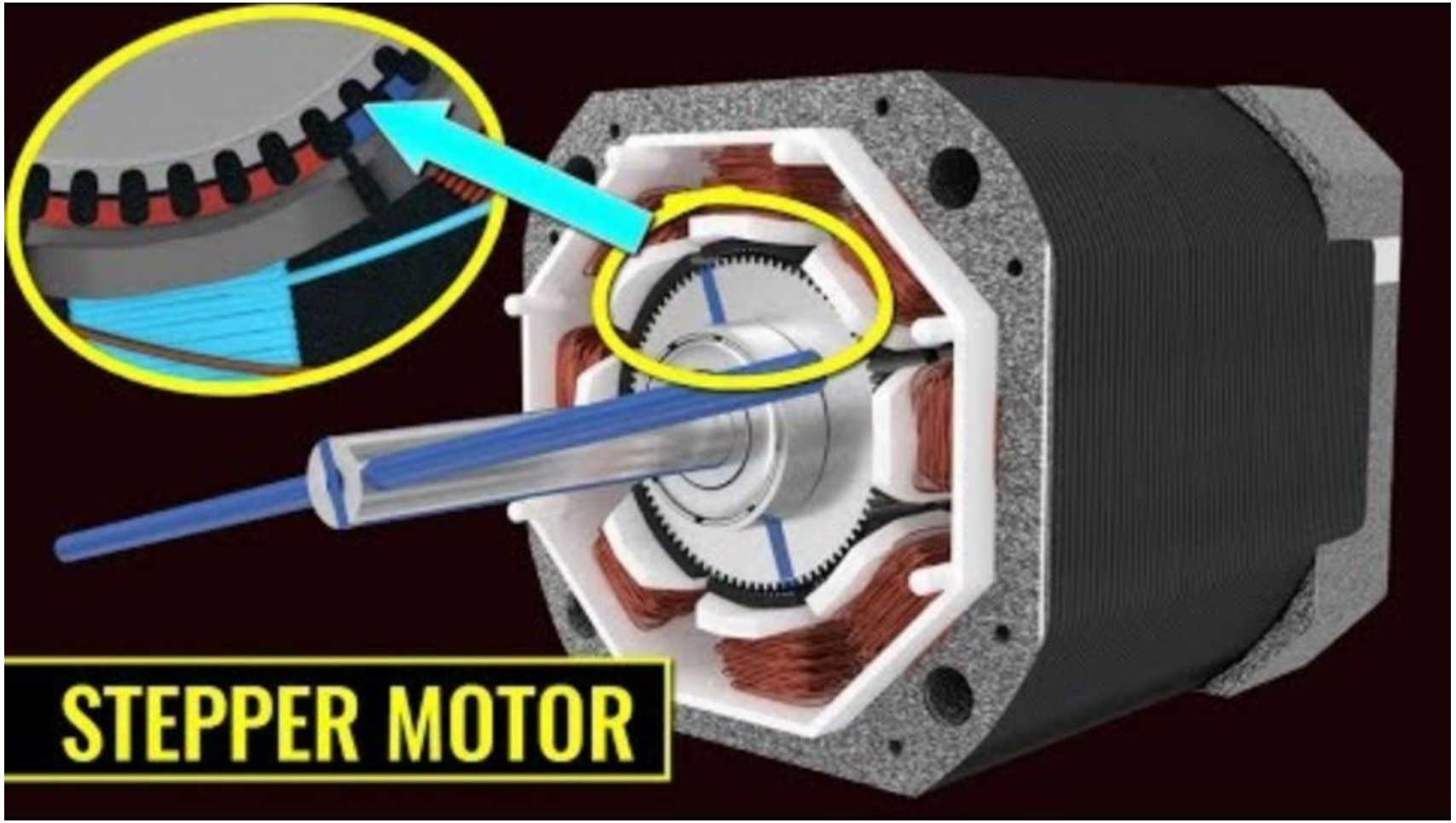
- Thanks to their efficiency and durability, the brushless DC motors have largely supplanted their brushed counterparts.
- They find a wide range of applications in devices that run continuously, such as **washing machines, air conditioners, and in consumer electronics like computer fans and disk drives**.
- More recently, they are used for **drones** as the rotational speed of each rotor can be precisely controlled.

In the near future, we can definitely expect more applications for brushless motors!

→ Do you know of any other advantages/use cases of Brushless DC motors?

Stepper Motors: Introduction

Stepper motors are motors that move in slow, precise and discrete steps. Valued for their precise position control, they find a myriad of applications such as desktop printers, security cameras, and CNC milling machines.



Stepper Motors: Advantages

	Advantages
Precise positioning	Stepper motors have a high pole count, usually from 50 to 100, and can accurately move between their many poles without the aid of a position encoder. As they move in precise steps, they excel in applications requiring precise positioning such as 3D printers, CNC, camera platforms and X, Y plotters.
Precise speed control	Precise increments in movement enables excellent speed control, making them a good choice in process automation and robotics.
Excellent torque characteristics at low speeds	Stepper motors have maximum torque at low speeds (less than 2000 rpm), making them suitable for applications that need low speed with high precision. Normal DC motors and servo motors do not have much torque at low speeds.
Excellent torque to maintain position	Suitable for applications with high holding torque.
Easy to control	Stepper motors can be easily controlled with microcontrollers such as the ATmega chips that are readily available on Arduino development boards.

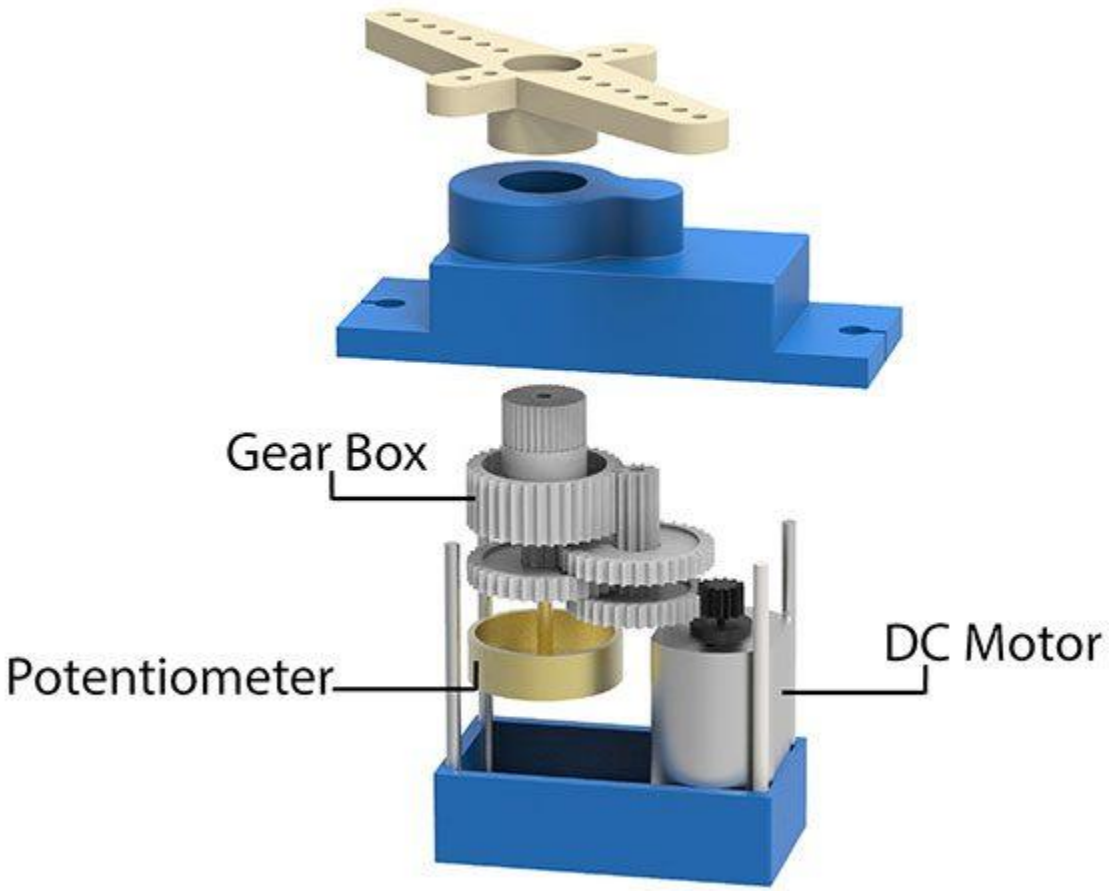
Stepper Motors: Disadvantages

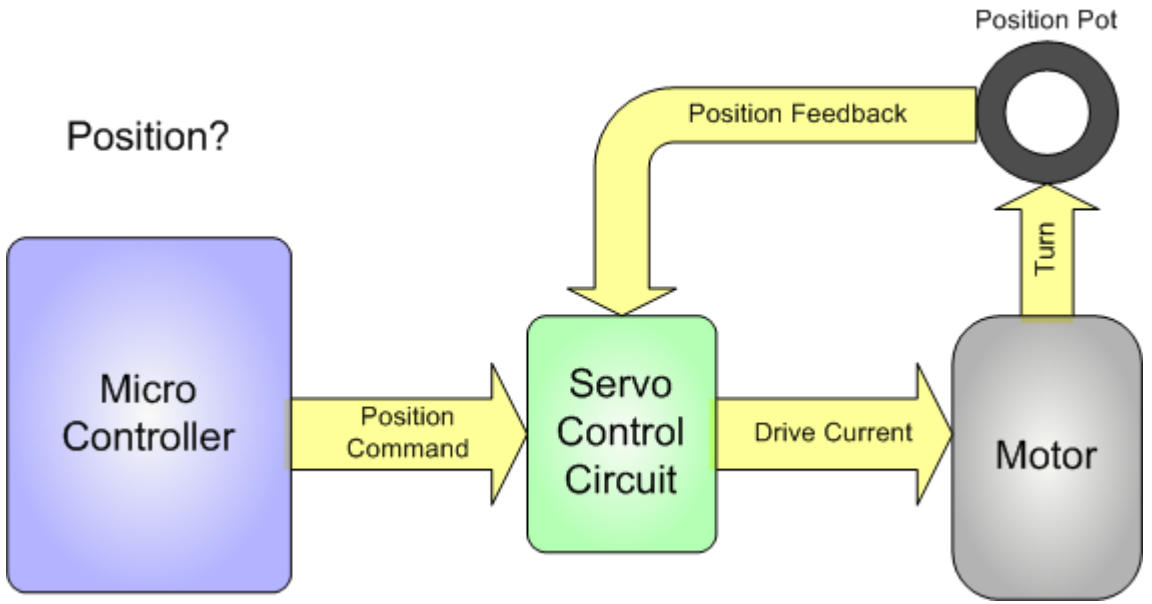
	Limitations
Noise	Stepper motors are known to generate some noise during operation. Thus, if your device needs to be quiet, accommodate a high range of speeds and torques and maintain a reasonable efficiency, then consider using a DC motor. But if your motion control application needs to be built quickly, does not need to be efficient, and a little noise is acceptable, then a stepper motor might be more suitable.
Limited high-speed torque	Generally, stepper motors have less torque at high speeds than at low speeds. Some steppers can be optimized for better torque at high speeds, but a driver would have to be paired with it to achieve that performance.
Low efficiency	Unlike DC motors, the current consumption of stepper motors is independent of load and they constantly draw maximum current. As such, they tend to become hot.
Might skip steps	As stepper motors have a low top-speeds, they might skip steps at high loads.

Servo Motors

- Can provide very precise motion control.
- The feedback in a servo motor system senses the difference between the actual and desired speed or position so that the controller can adjust the output to correct any drift from the target position.
- The positional rotation and continuous rotation are two basic types of servo motors

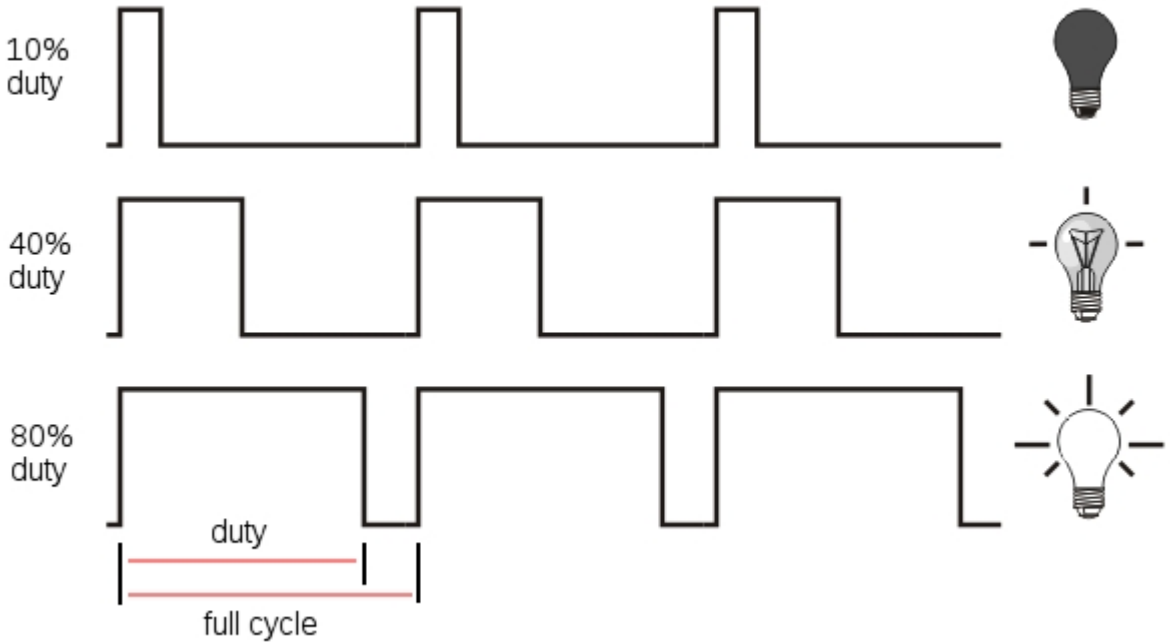
Servo Motors





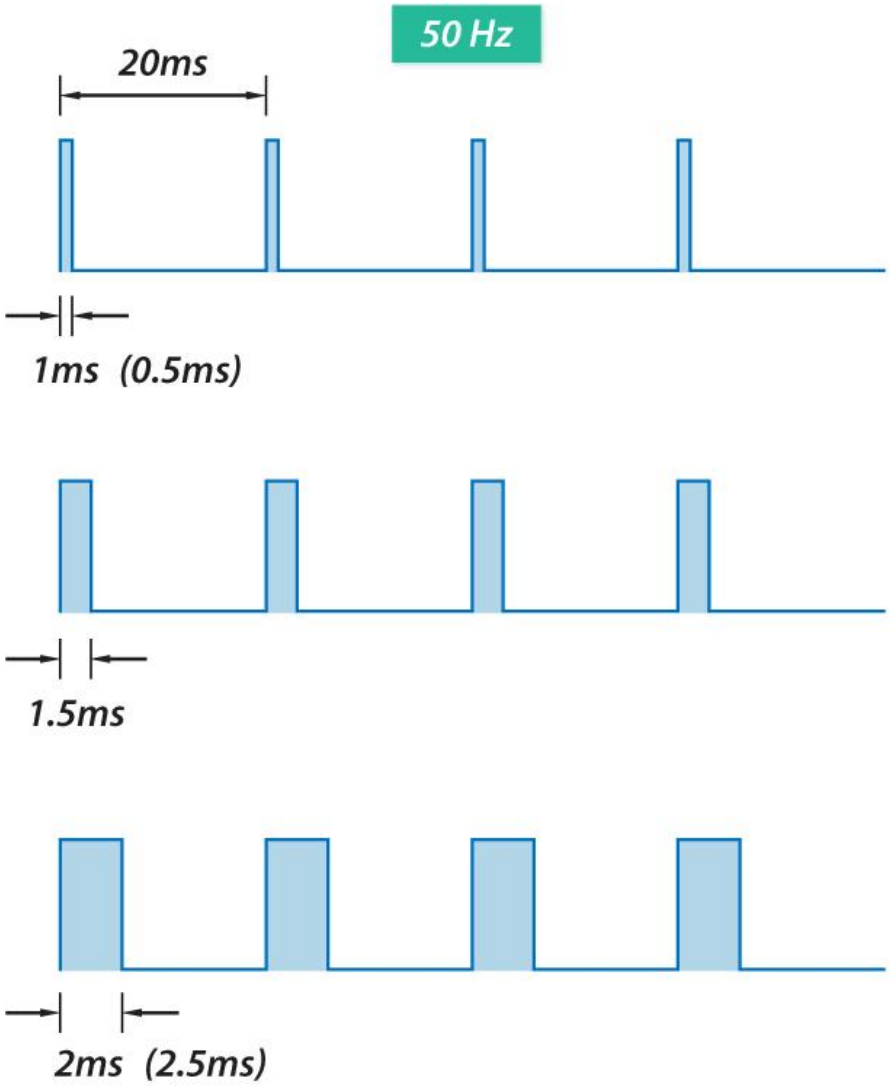
Servo Motors

Pulse width Modulation (PWM)



Servo Motors

Pulse width Modulation (PWM)



0 Degrees

90 Degrees

180 Degrees

Servo Motors

	Advantages
Excellent torque characteristics at high speeds	At speeds greater than 2000 rpm, servo motors have high torque and are best suited for applications with high speeds and high torque applications that involve dynamic load changes. Servo motors can generate a higher peak torque as they are able to operate at higher speeds. This is because servo motors operate under a constant closed-loop feedback mechanism as opposed to the open-loop system of a stepper motor, which allows it to reach higher speeds and generate higher peak torque.
Variety	They come in many different sizes and torque ratings.
Inexpensive	Small sized servos only costs a few dollars. Many servo motors have gears that are made of plastic to keep them light and at the same time, cheap.

Servo Motors

	Limitations
Limited range of motion	Positional rotation servos are limited to 180 degrees of motion.
Jitter	The feedback mechanism in the servo will constantly try to correct any drift from the desired position. This constant adjustment results in twitching while trying to hold a steady position. As such, a stepper motor could be considered instead if this is a problem for your application.

How to choose a motor/actuator?

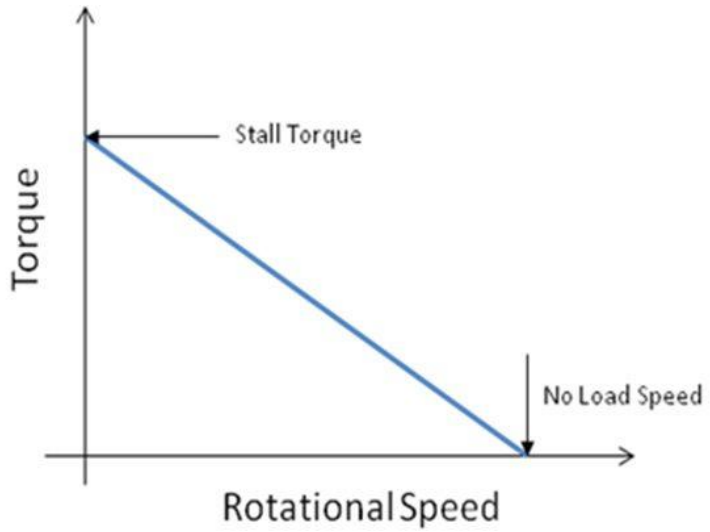
How can I choose what motor to use?

Explore

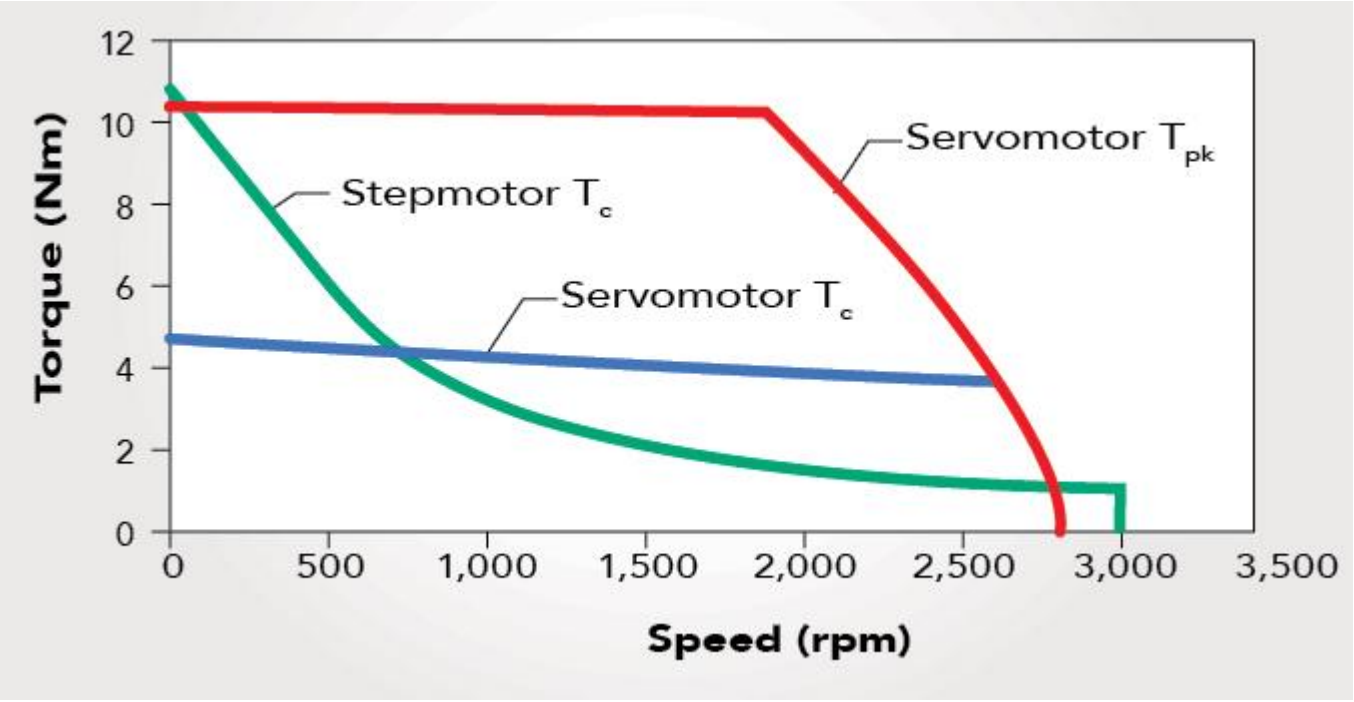
- Datasheet….
- Typical characteristics for a motor
- What existing designs and products use

Performance Curves: Torque-Speed Curves

DC Motors



Steppers & Servo Motors



What torque do you want at what speed?

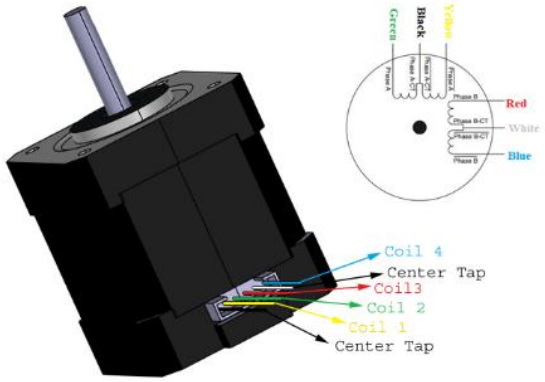
Data-sheet: DC Motor

Rated Voltage	Motor Type	Stall Current	No-Load Current	No-Load Speed (RPM)	Extrapolated Stall Torque (kg · cm)	Extrapolated Stall Torque (oz · in)	Max Power (W)	 Single-Shaft (Gearbox Only)	 Dual-Shaft (Gearbox & Motor)
12 V	high-power, carbon brushes **(HPCB)**	0.75 A	100 mA	6800	0.09	1.3	–	5:1 HPCB 12V	5:1 HPCB 12V dual-shaft
			80 mA	3400	0.17	2.4	1.5	10:1 HPCB 12V	10:1 HPCB 12V dual-shaft
				2200	0.25	3.5	1.4	15:1 HPCB 12V	15:1 HPCB 12V dual-shaft
				1100	0.39	5.4	1.1	30:1 HPCB 12V	30:1 HPCB 12V dual-shaft
				650	0.67	9.3	1.1	50:1 HPCB 12V	50:1 HPCB 12V dual-shaft
				450	1.0	14	1.1	75:1 HPCB 12V	75:1 HPCB 12V dual-shaft
				330	1.3	18	1.1	100:1 HPCB 12V	100:1 HPCB 12V dual-shaft
				220	1.8	25	1.0	150:1 HPCB 12V	150:1 HPCB 12V dual-shaft
				160	2.5	35	1.0	210:1 HPCB 12V	210:1 HPCB 12V dual-shaft
				130	3.0	42	1.1	250:1 HPCB 12V	250:1 HPCB 12V dual-shaft
				110	3.3	46	1.0	298:1 HPCB 12V	298:1 HPCB 12V dual-shaft
				85	5.0	69	1.1	380:1 HPCB 12V	380:1 HPCB 12V dual-shaft
				35	10	140	–	1000:1 HPCB 12V	1000:1 HPCB 12V dual-shaft

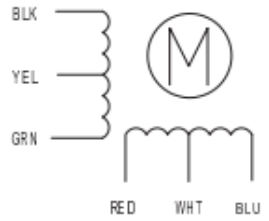
Data-sheets

HIGH TORQUE HYBRID STEPPING MOTOR SPECIFICATIONS

General specifications		Electrical specifications	
Step Angle (°)	1.8	Rated Voltage (V)	4
Temperature Rise (°C)	80 Max (rated current,2 phase on)	Rated Current (A)	1.2
Ambient temperature (°C)	-20~+50	Resistance Per Phase (±10%Ω)	3.3 (25°C)
Number of Phase	2	Inductance Per Phase (±20%mH)	2.8
Insulation Resistance	100MΩ, Min (500VDC)	Holding Torque (Kg.cm)	3.17
Insulation Class	Class B	Detent Torque (g.cm)	200
Max.radial force (N)	28 (20mm from the flange)	Rotor Inertia (g.cm²)	68
Max.axial force (N)	10	Weight (Kg)	0.365

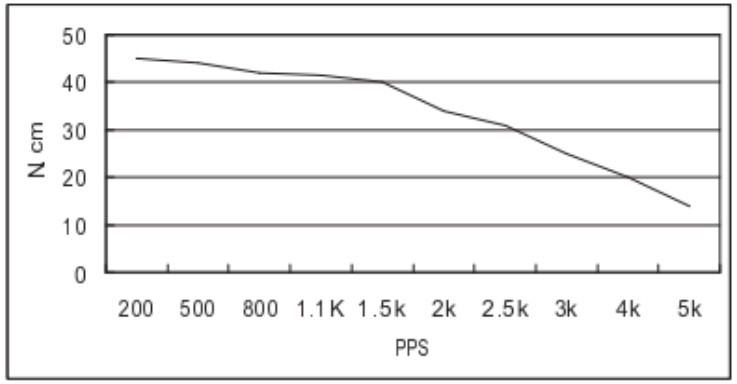


● Wiring Diagram:

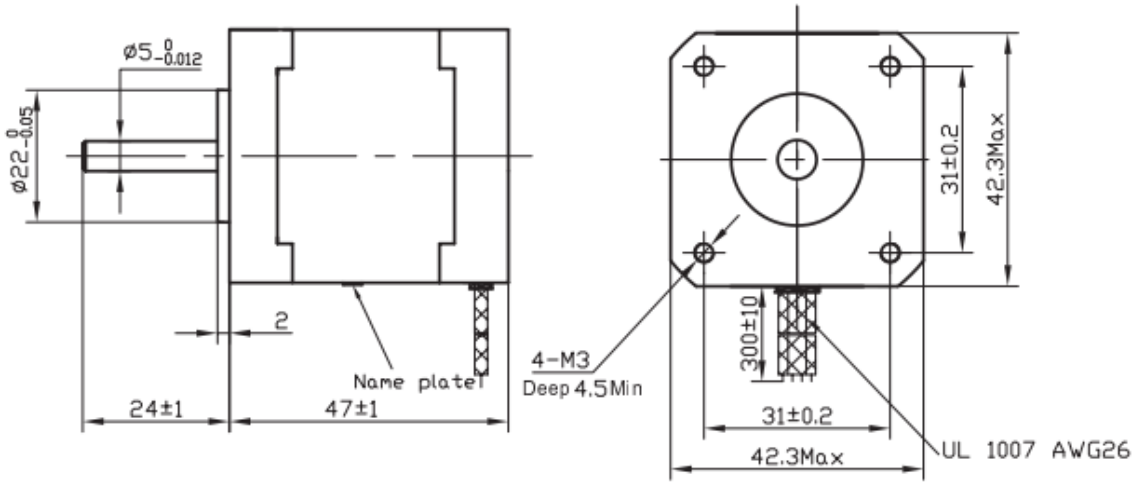


● Pull out torque curve:

VOLTAGE: 24VDC, CONSTANT CURRENT: 1.2A, HALF STEP



● Dimensions:
(unit=mm)



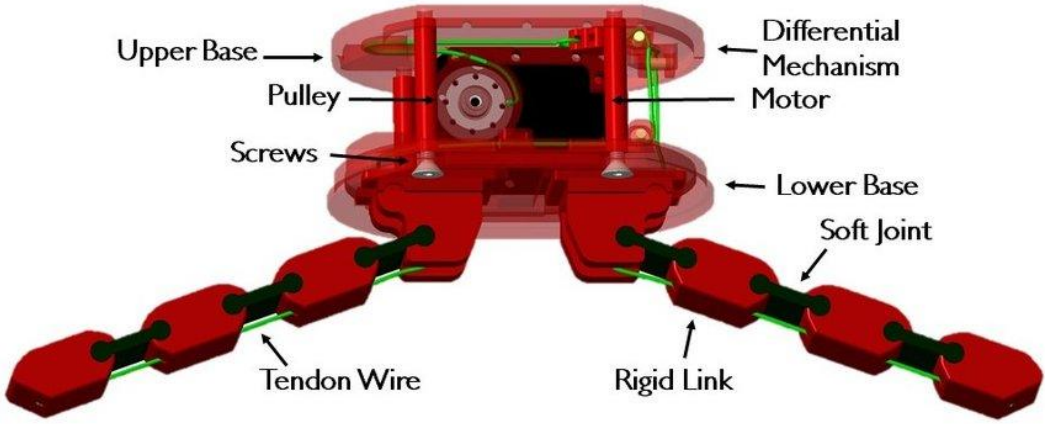
The actuator is only the starting point…

….connects to a mechanism or a drive chain

The actuator is only the starting point…

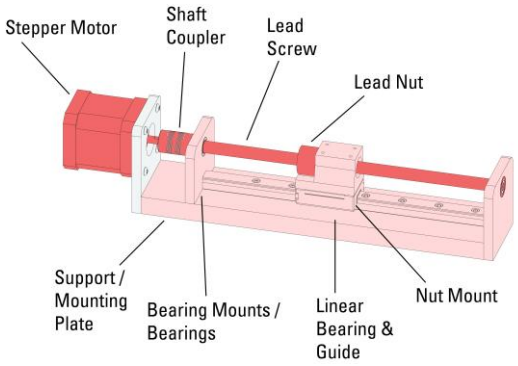
….connects to a mechanism or a drive chain

Motor + Pulley = Change in length
→ Tendon driven structures



Gear systems = change in speed/torque

Motor + lead screw = linear actuator



Rack & Pinion Mechanism

Controlling Actuators with a Microcontroller

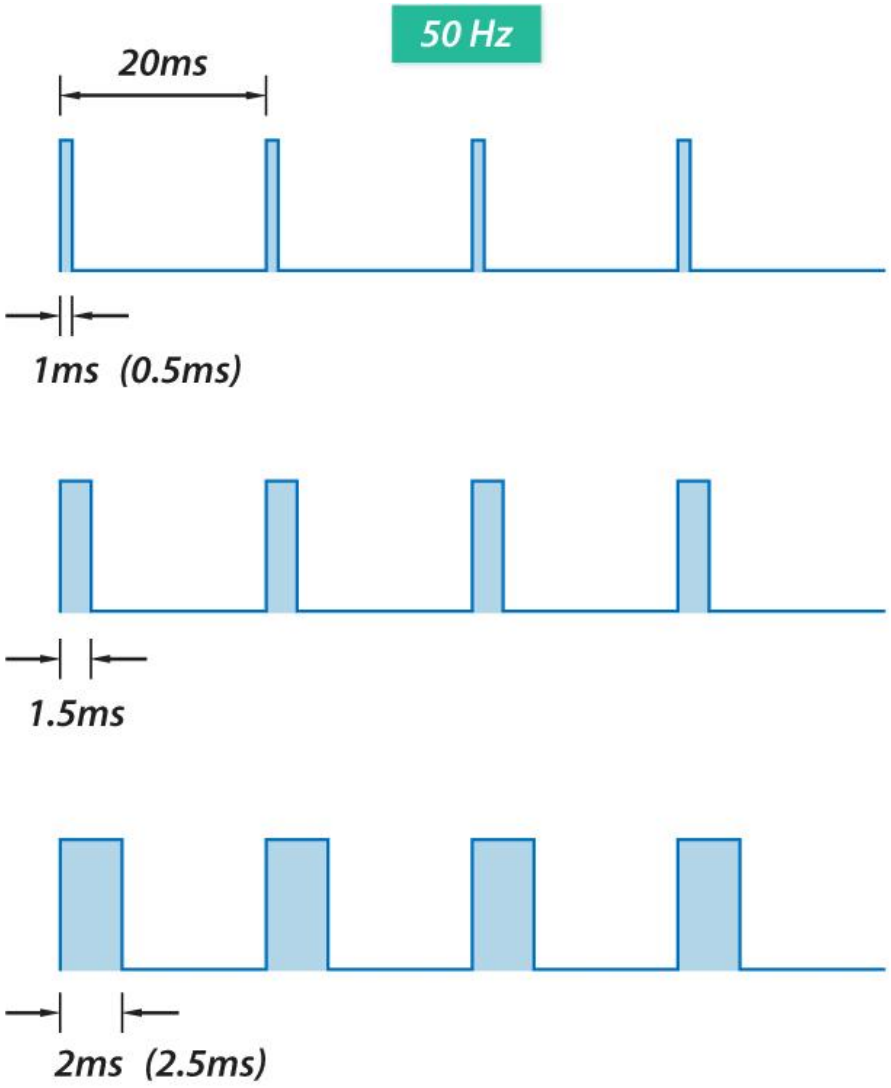
Controlling Servos

SG90 Micro Servo Motor 9G

MG995 RC Servo Metal Gear High Speed

Servo Motors

Pulse width Modulation (PWM)



0 Degrees

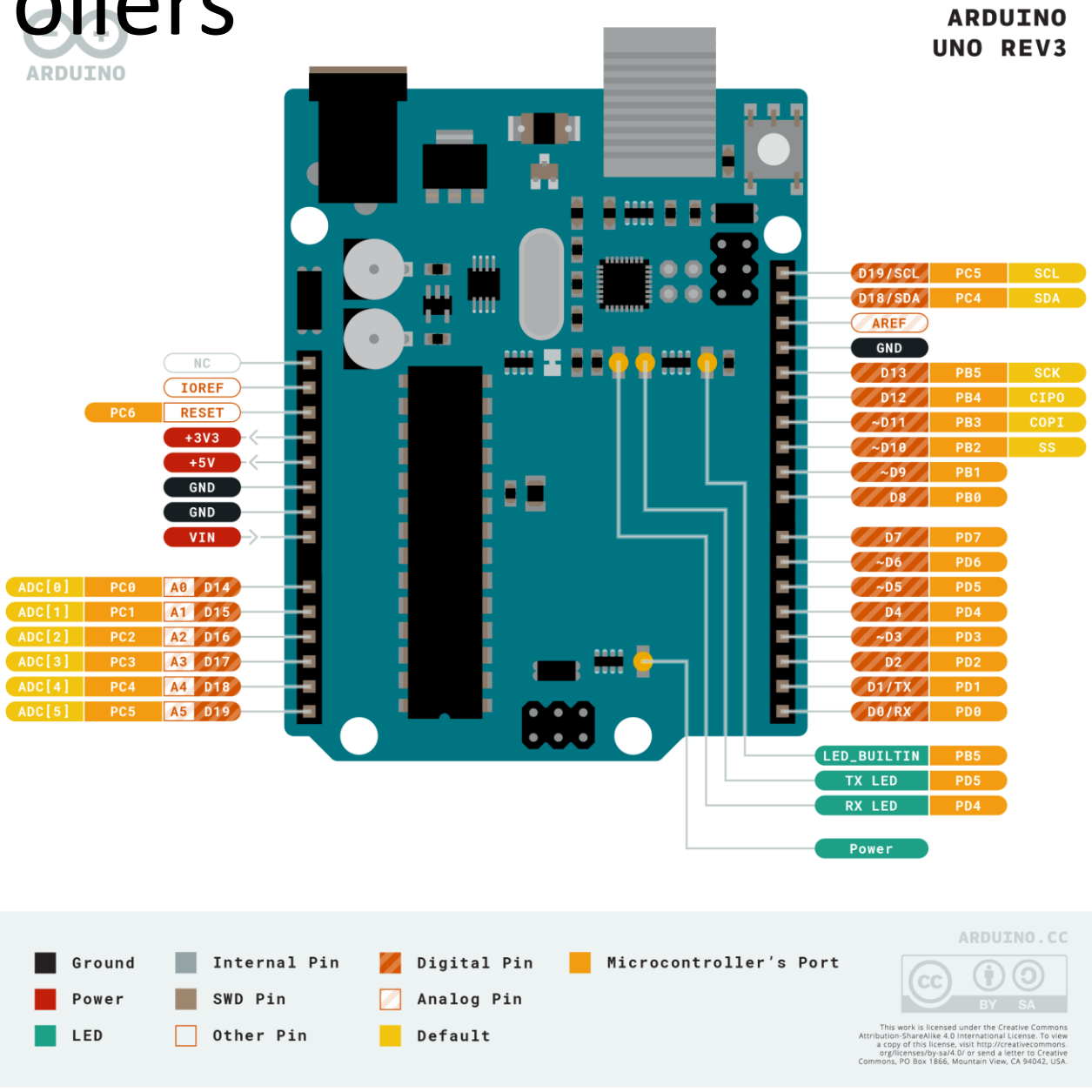
90 Degrees

180 Degrees

Micro-Controllers

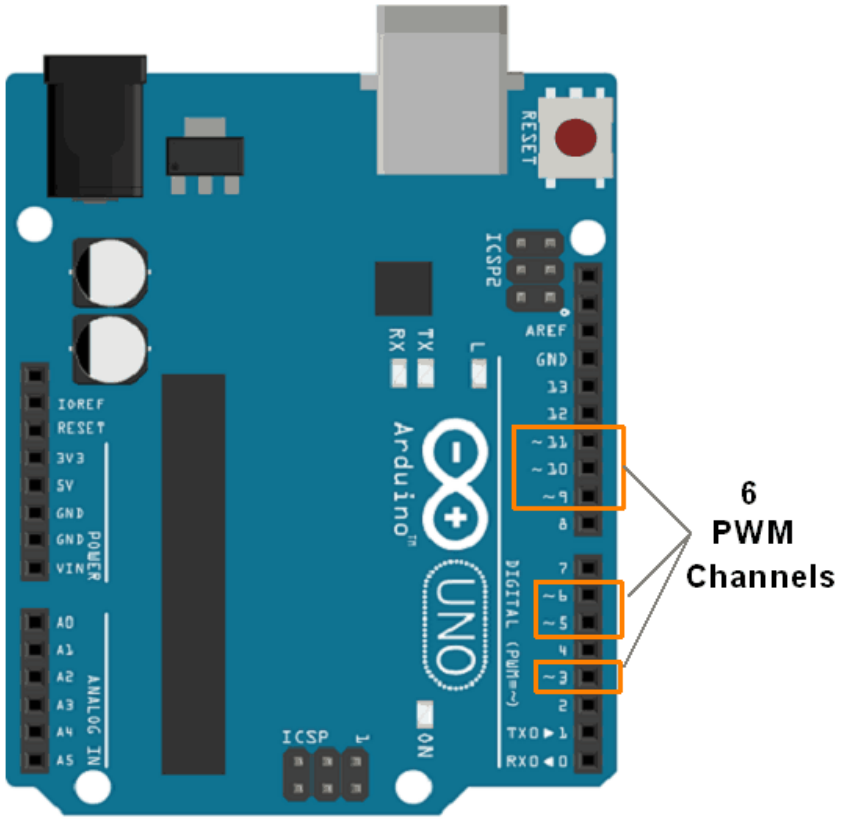
Output voltages/GND

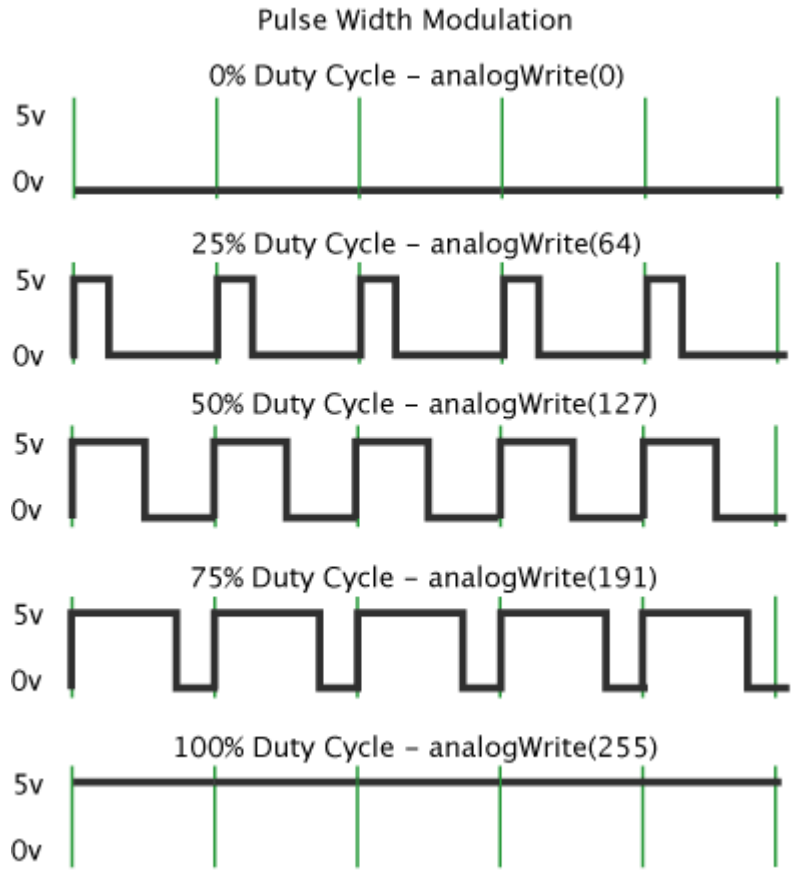
Analog Inputs – measures input voltages between 0-5V



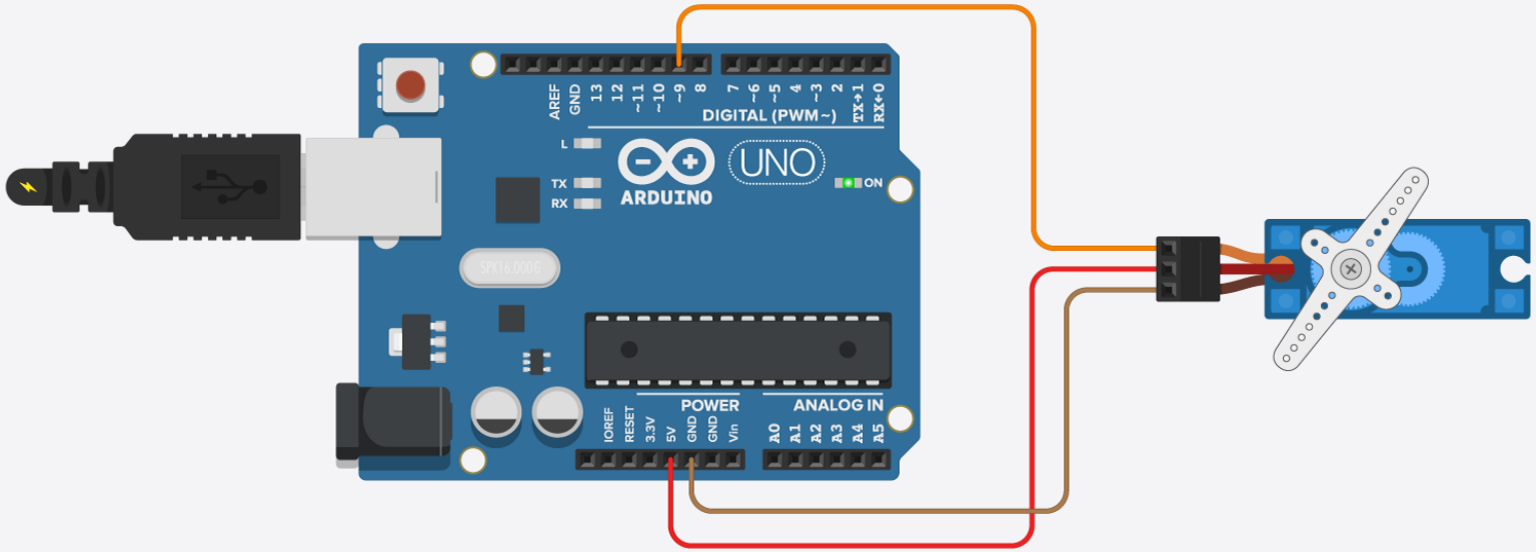
Digital Pins (can be inputs or outputs)

Controlling Servos





Controlling Servos



Limited to approx. 40-50mA draw from the Uno power supply…

Controlling Servos

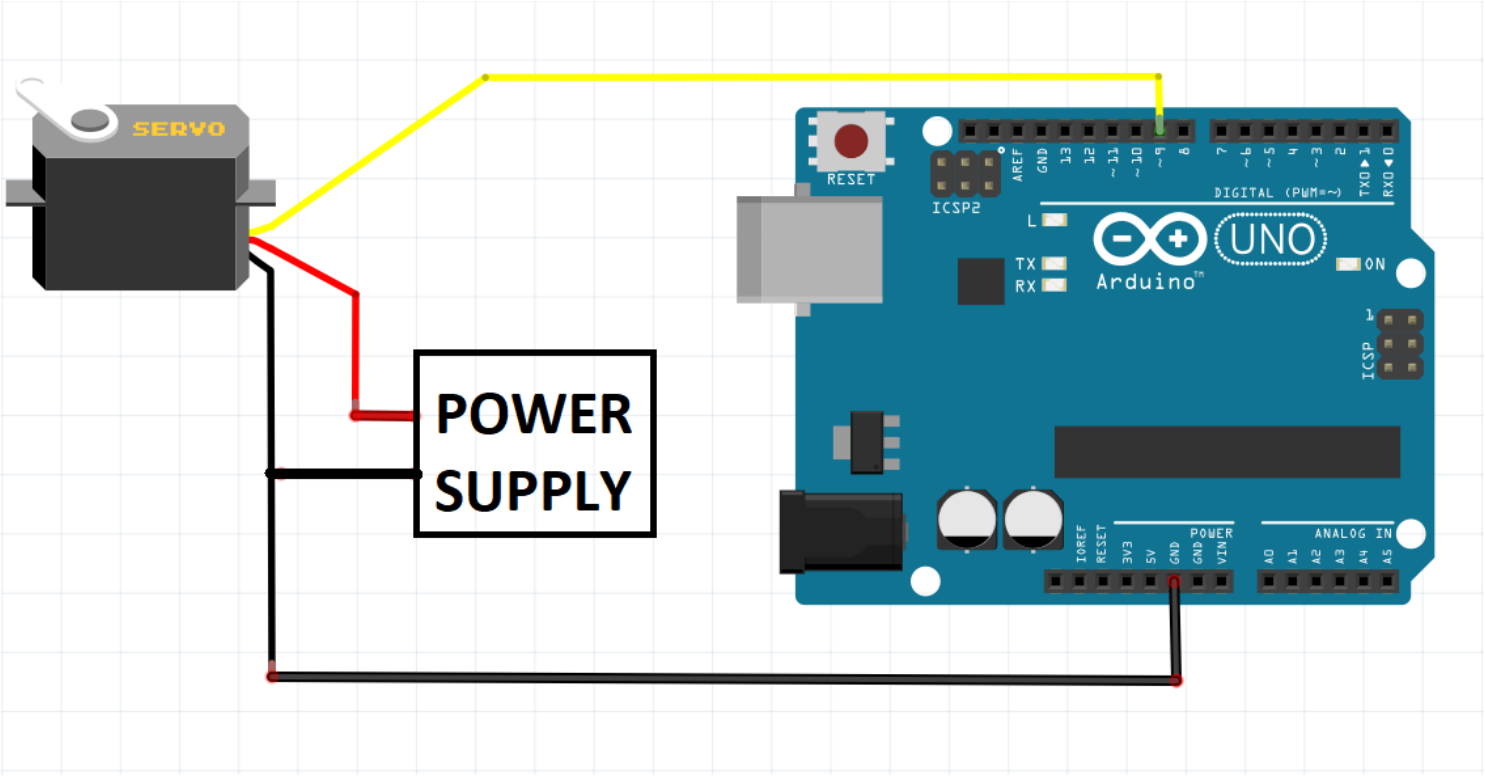
COPY

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in varia
    delay(15);                       // waits 15ms for the servo to reach the
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in varia
    delay(15);                       // waits 15ms for the servo to reach the
  }
}
```

Servo library!

Controlling Servos

How can we provide 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

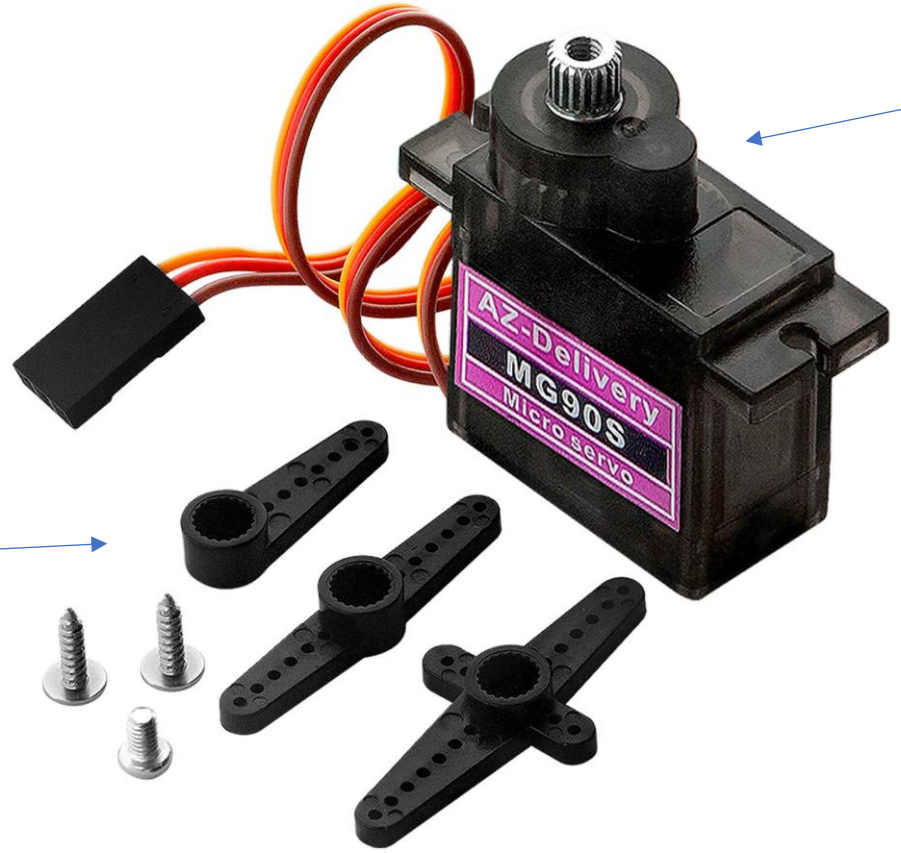
Power supply – white line indicates +ve supply

Be careful!!! The small micro servos will blow up if you apply more than 6V!

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.

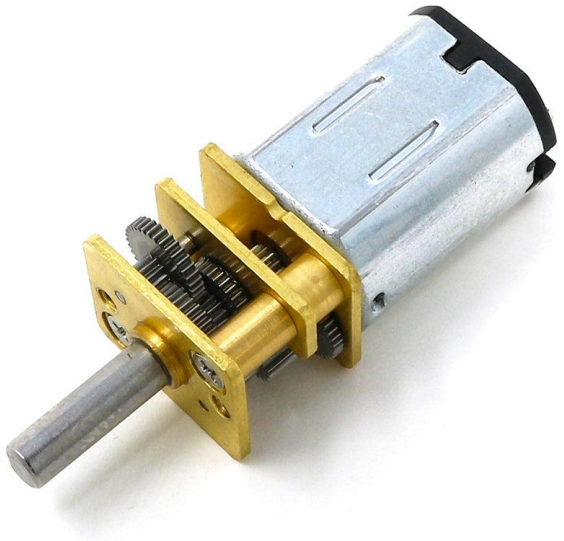
Physically Interfacing with Servos



Mounting points

Servo ‘horns’ – can screw on in variable position

Controlling Motors

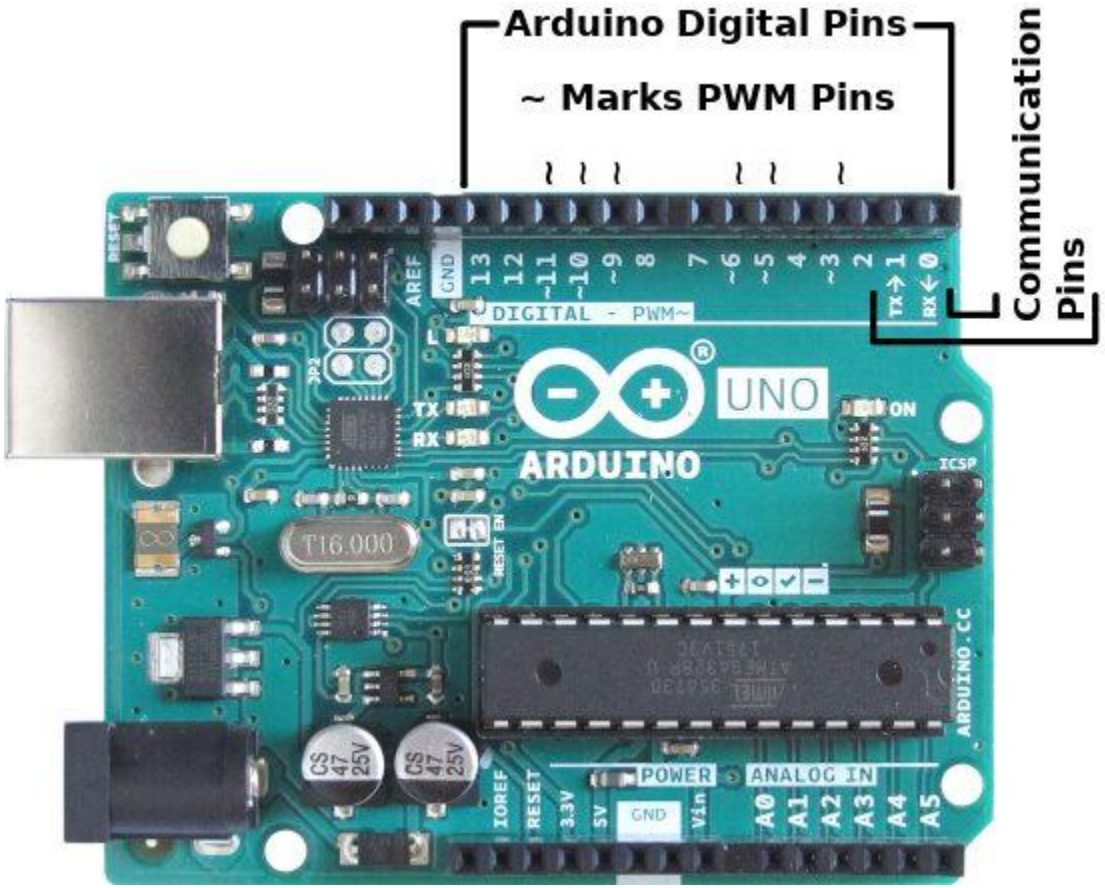


- Speed is proportional to the voltage
- The polarity determine the direction

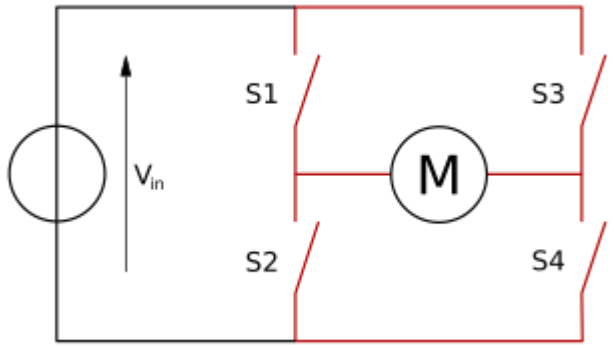
→ Need to be able to vary the polarity and the voltage from the micro-controller

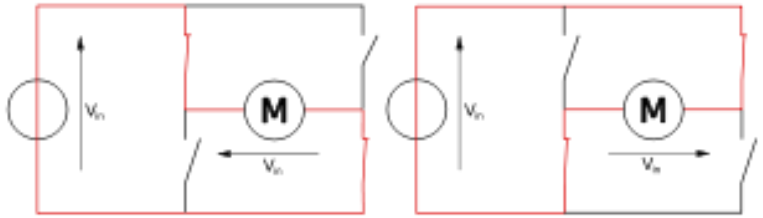
What kind of output should we use to control this?

Arduino Digital Outputs



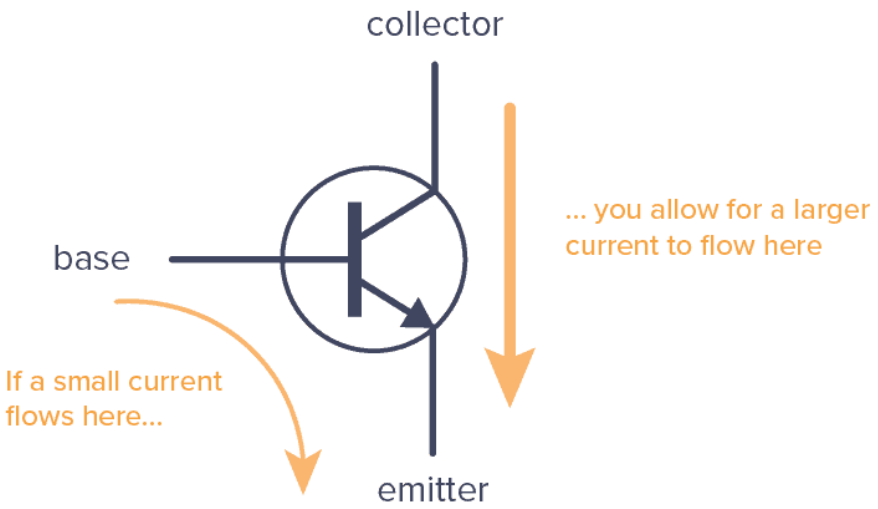
Controlling Motors: H-Bridges

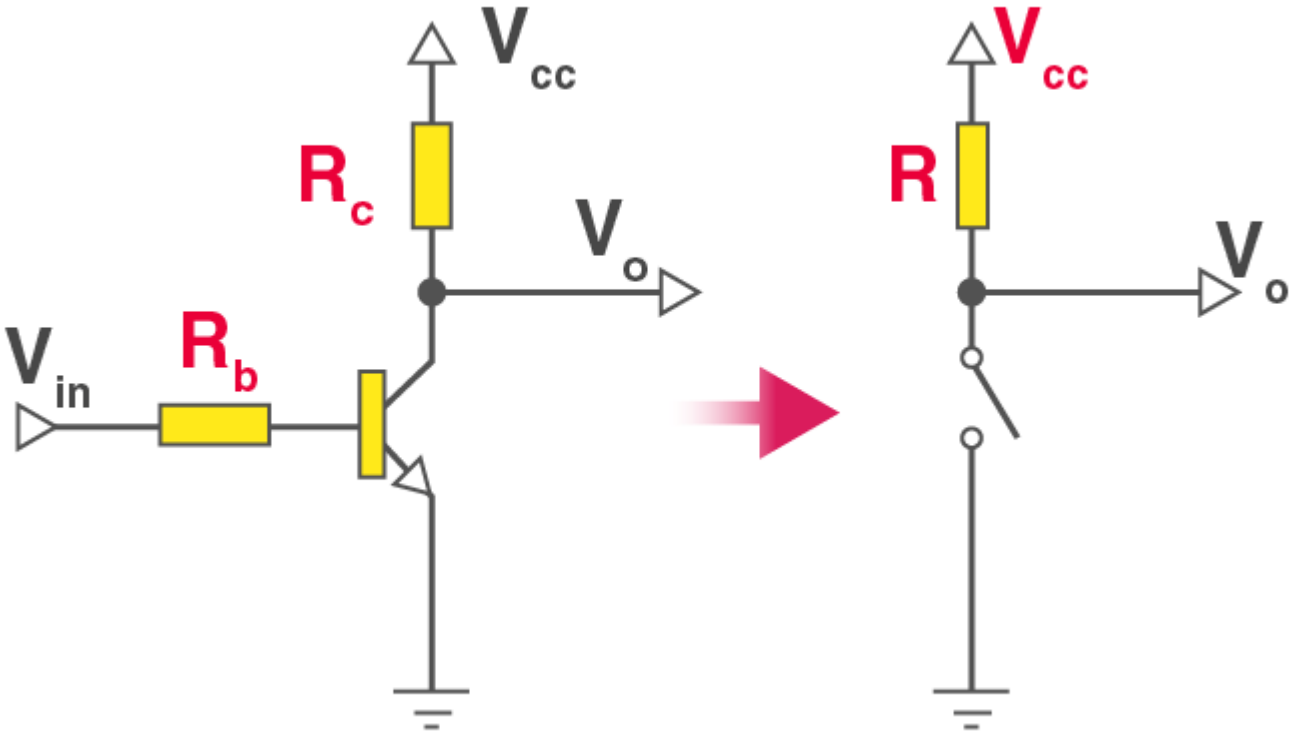




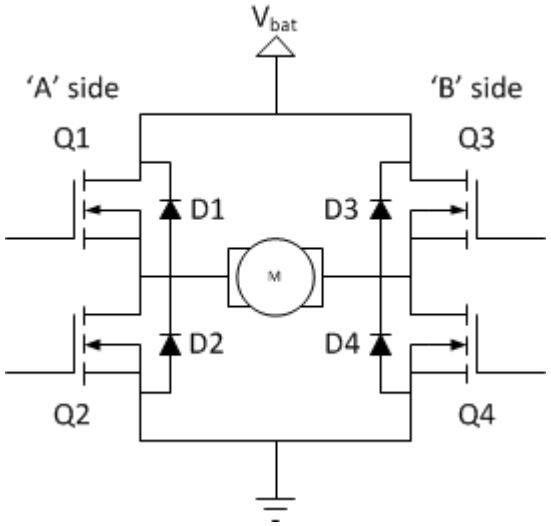
→ But we don’t want to use ‘mechanical’ switches, we want to use electrical ones…

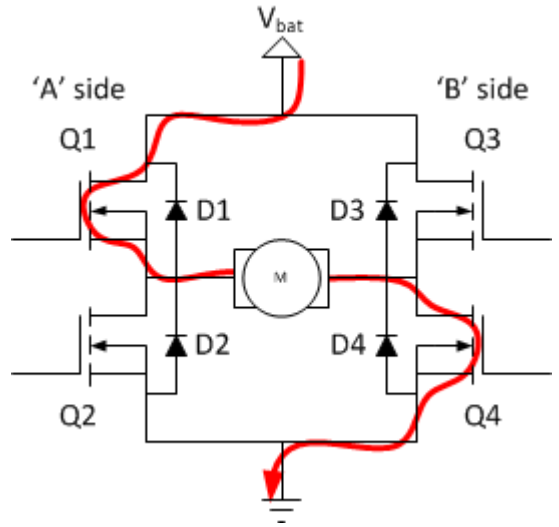
Introducing the Transistor….

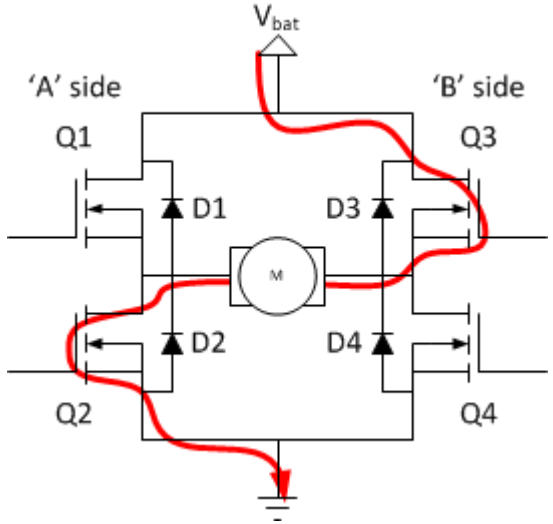




Controlling Motors: H-Bridges





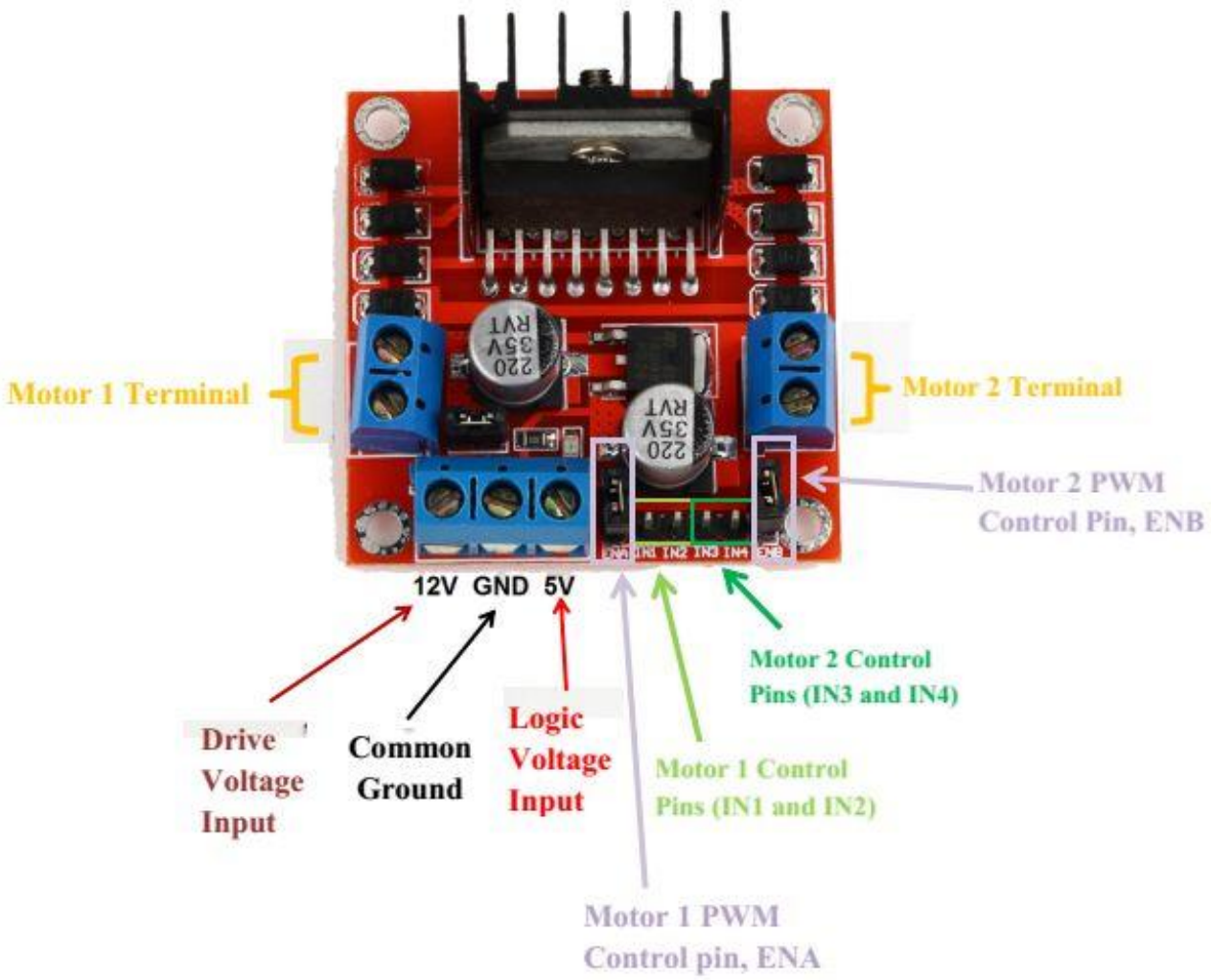


- The switching elements (Q1..Q4) are usually bi-polar or FET transistors, in some high-voltage applications IGBTs.
- Integrated solutions also exist but whether the switching elements are integrated with their control circuits or not is not relevant for the most part for this discussion.
- The diodes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

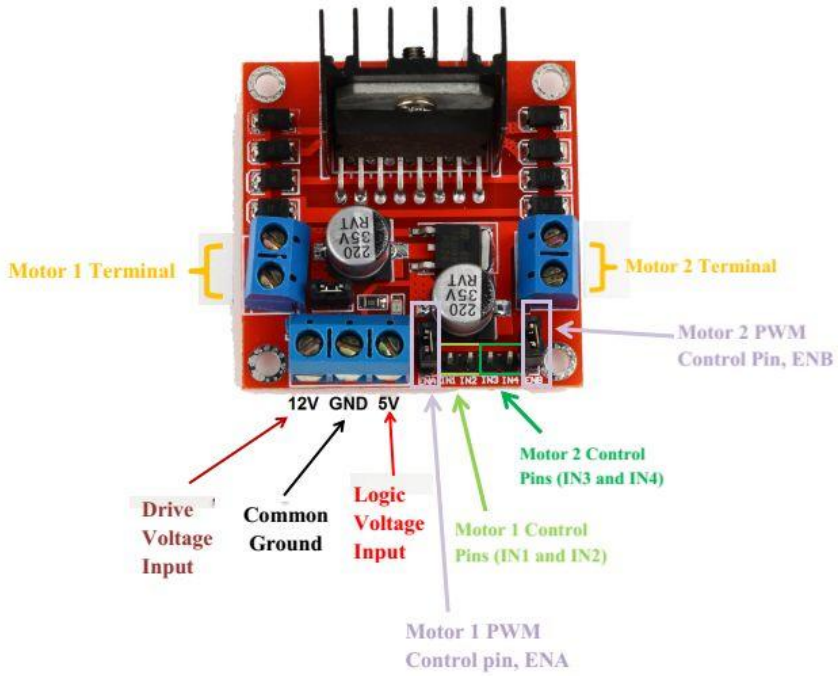
Controlling Motors: H-Bridges

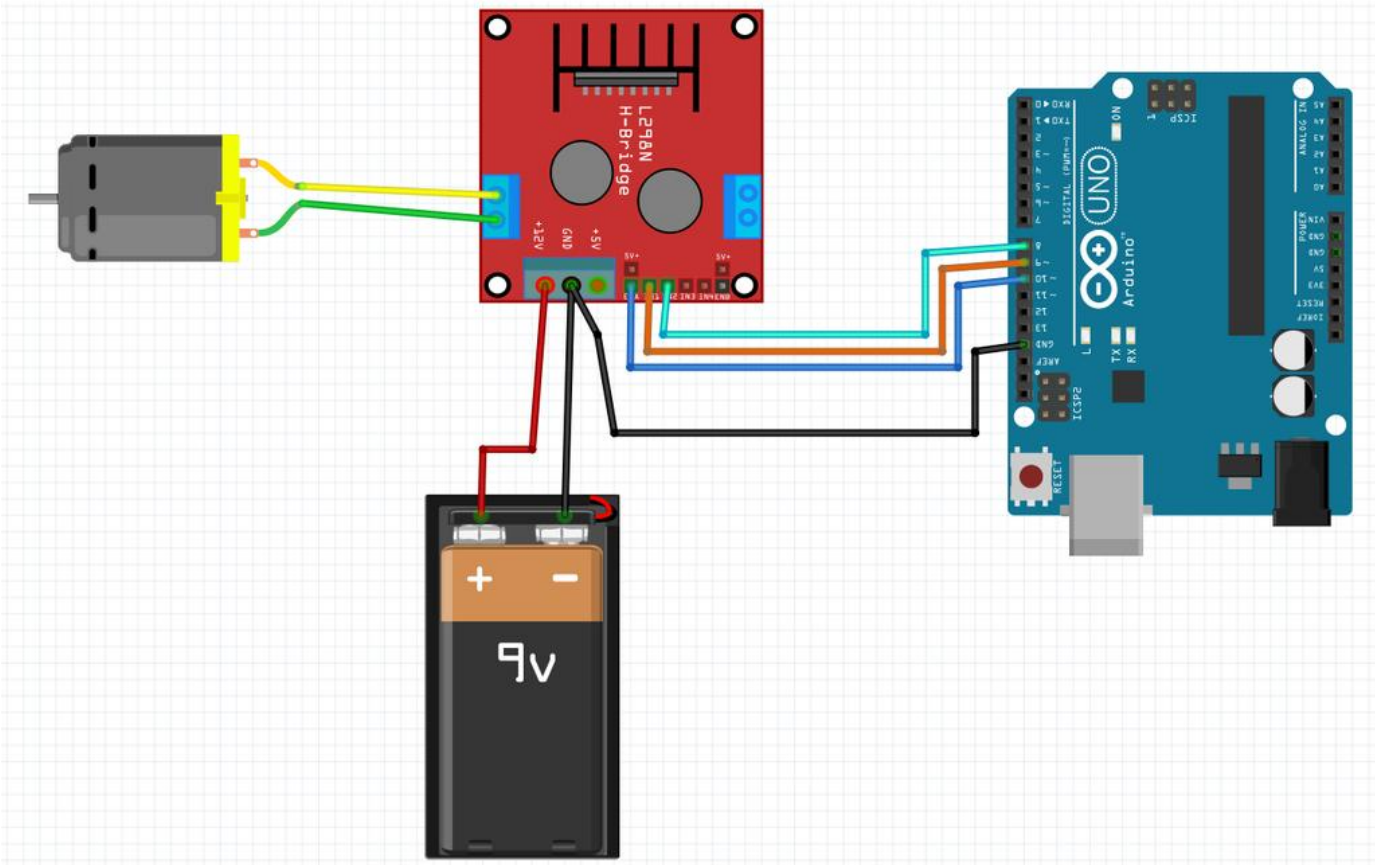
L298N

Remember to check out the data-sheet!

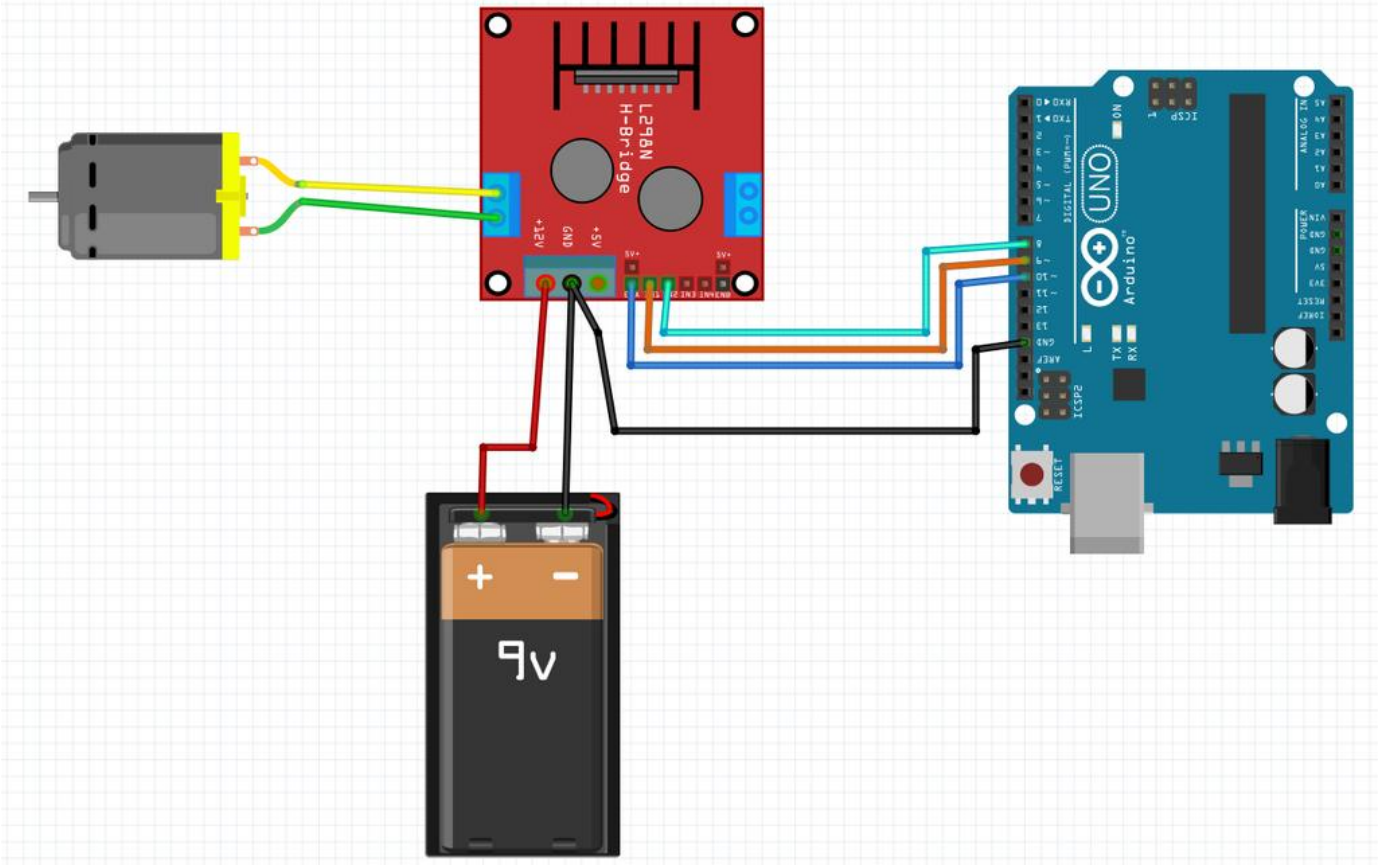


Controlling Motors: H-Bridges

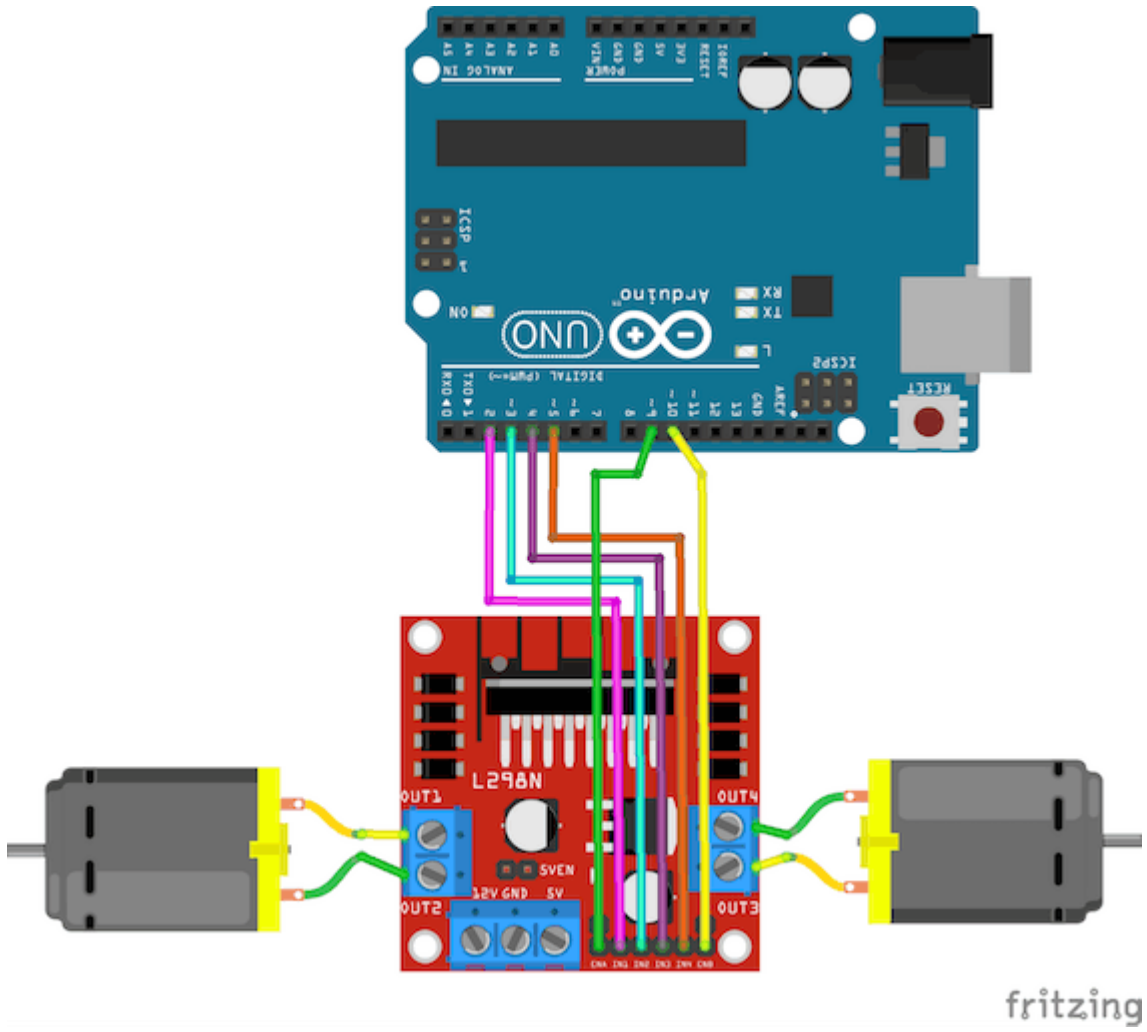




Controlling Motors: H-Bridges



- You can change the speed with the EN pins using 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);
}
```

Try using the different actuators

Micro servo:

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

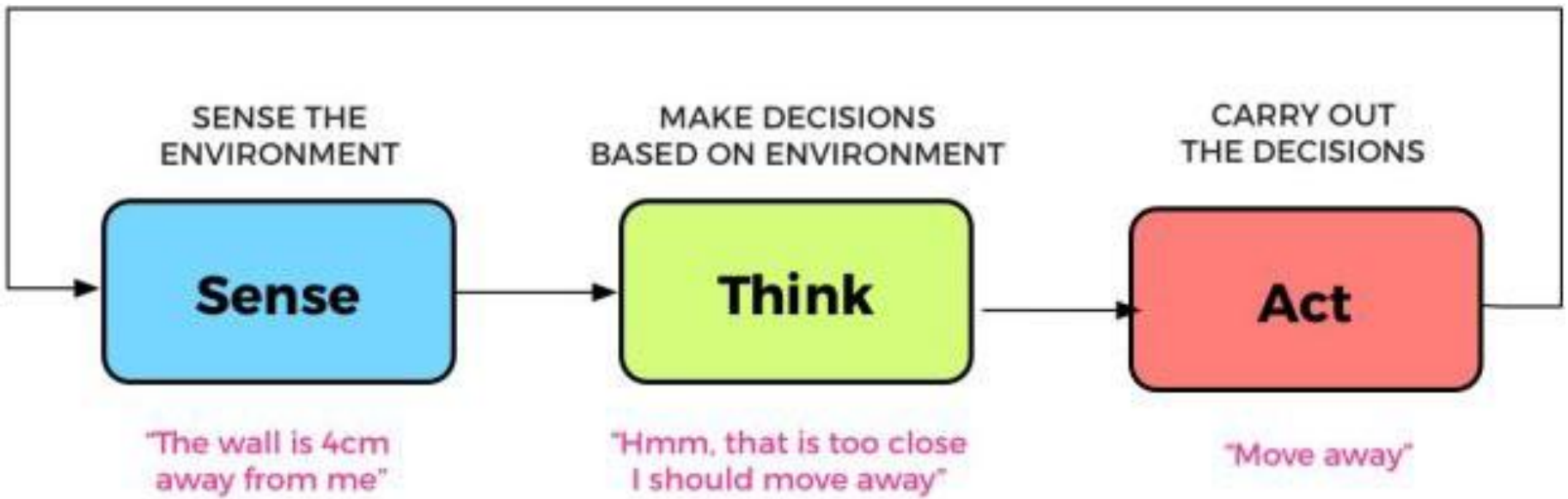
Larger servo (external power)

- Position control + speed control

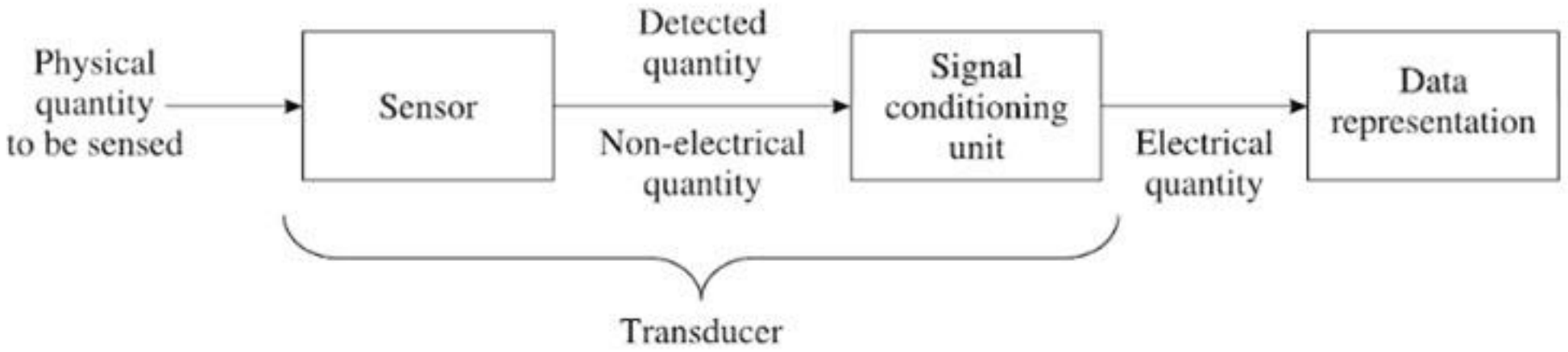
DC motor

- Speed + direction control

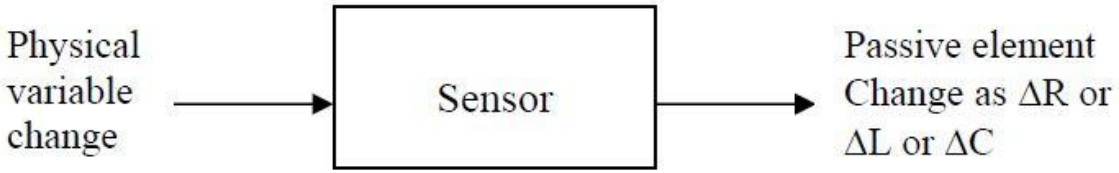
Engineering Design

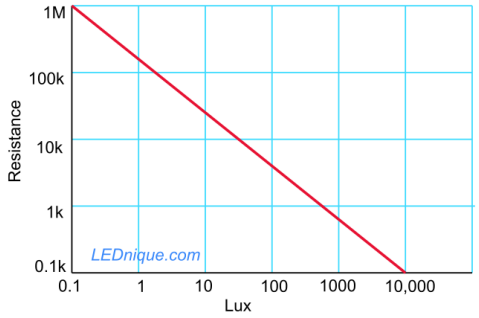


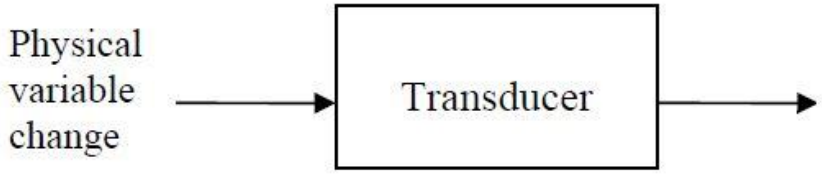
What is a sensor? What is a transducer?

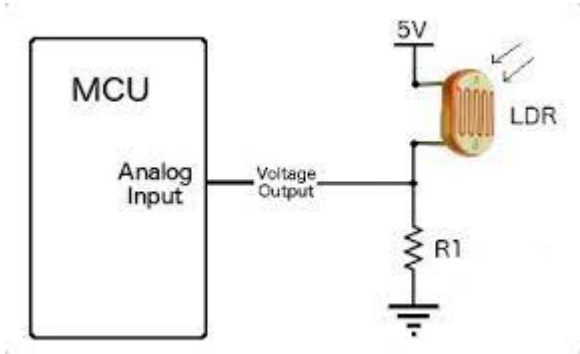


What is a sensor? What is a transducer?

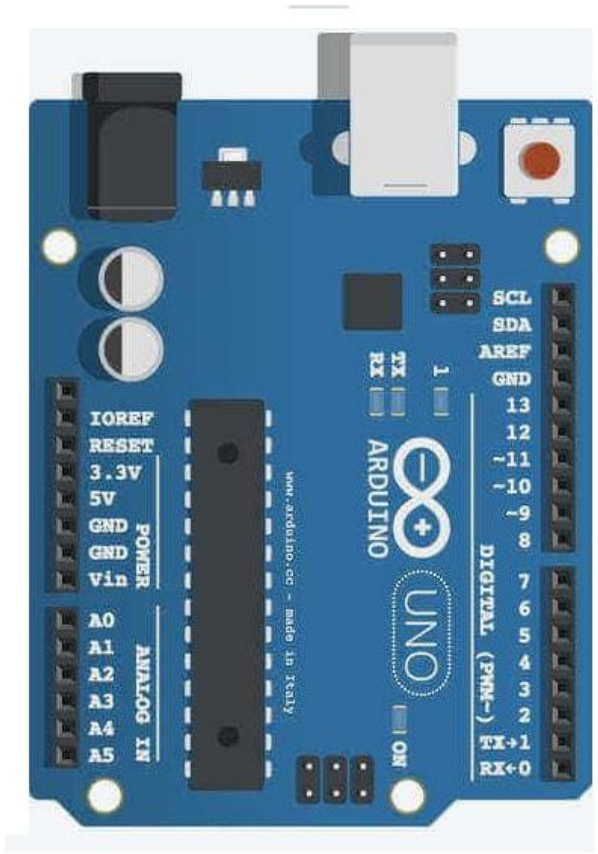






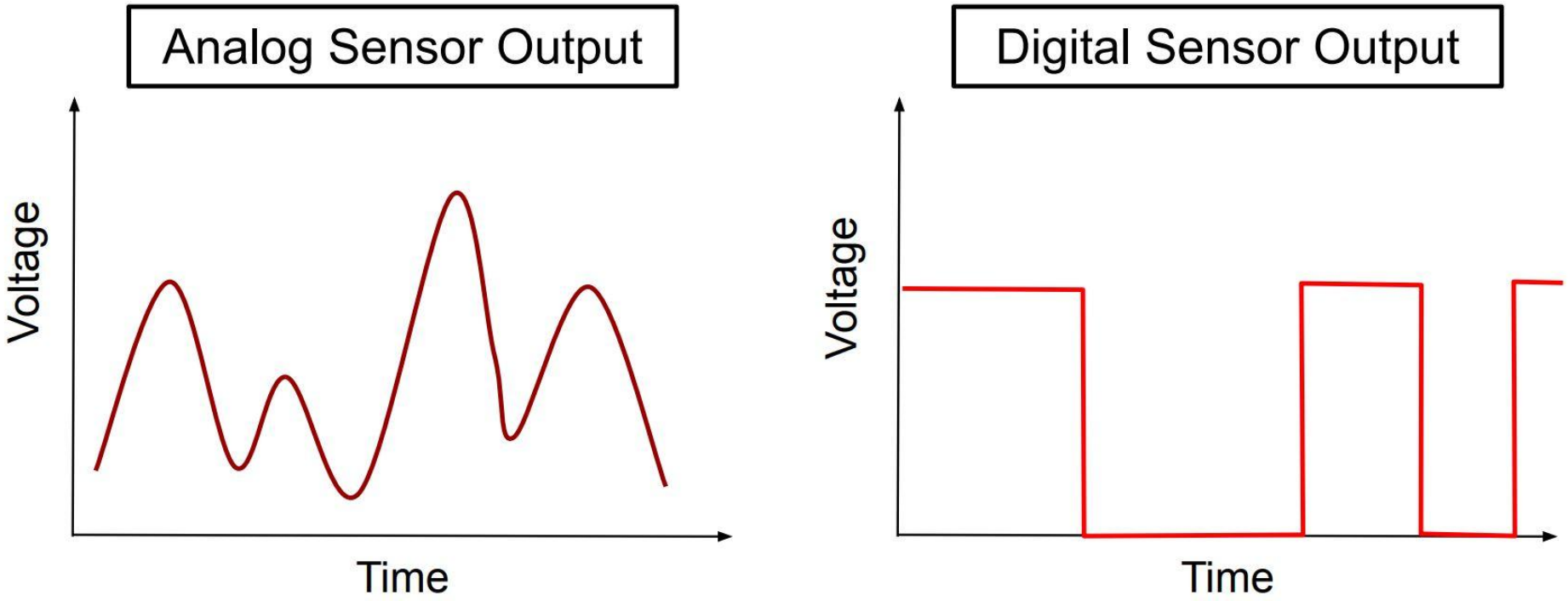


Sensing with a microcontroller



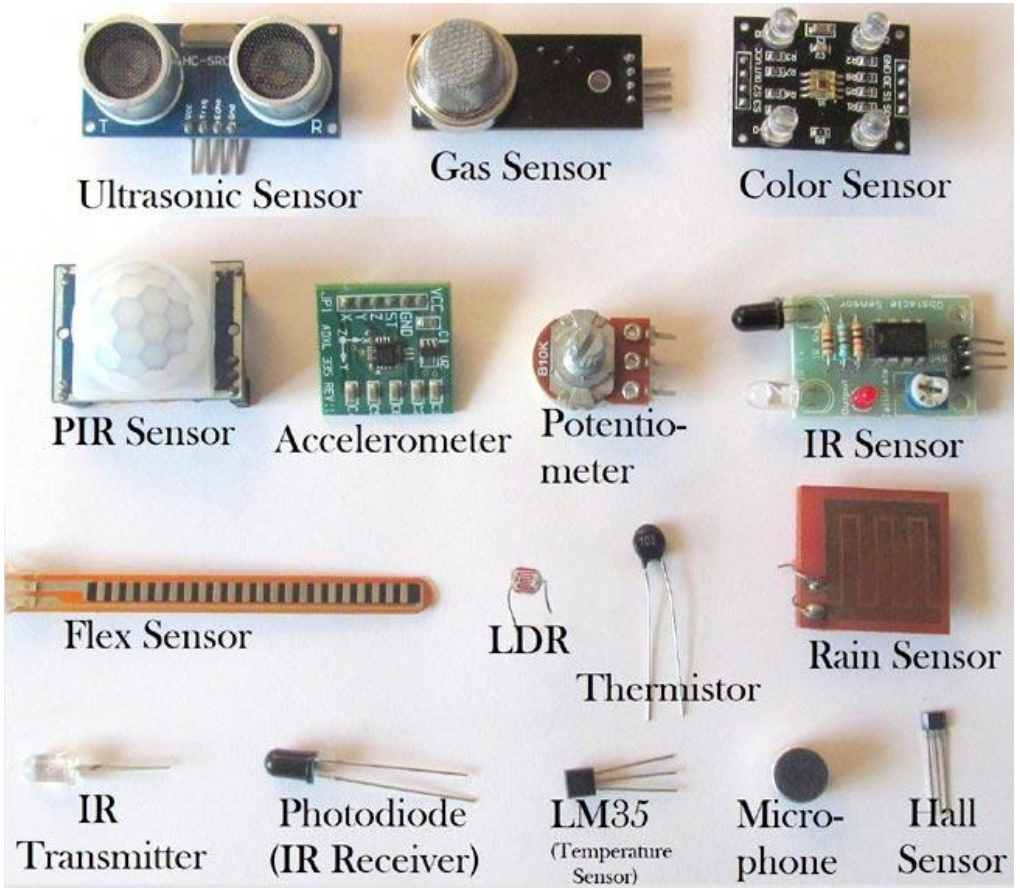
- Many different sensors
- Different input types
- Different transduction circuits

Analog vs. Digital



Analog Inputs

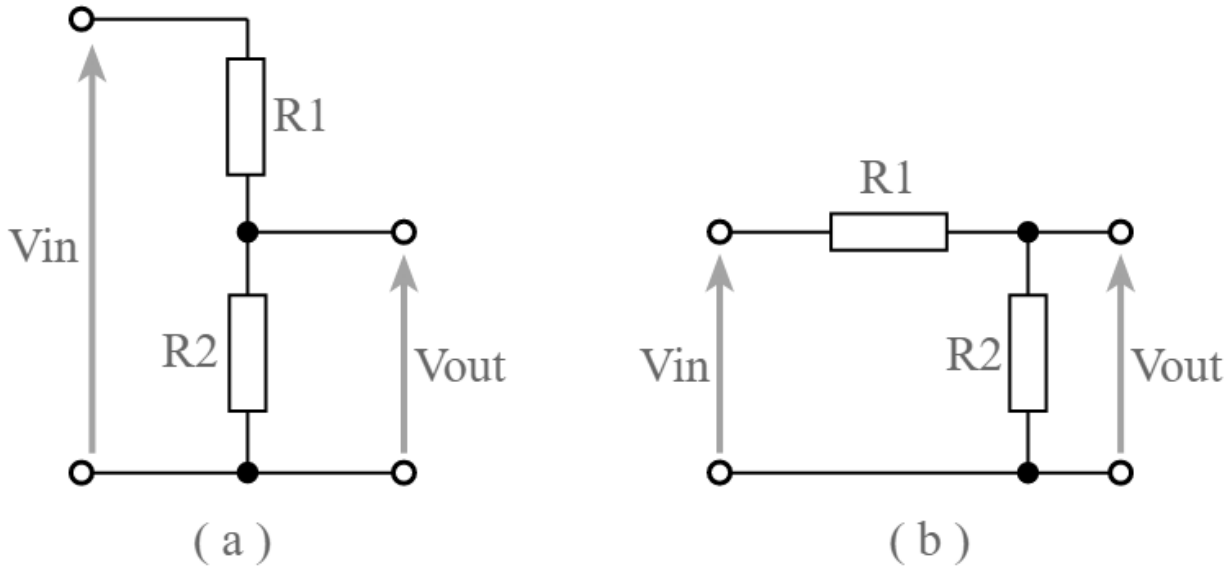
What types of sensors are analog?



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

Analog Inputs

Resistive sensors



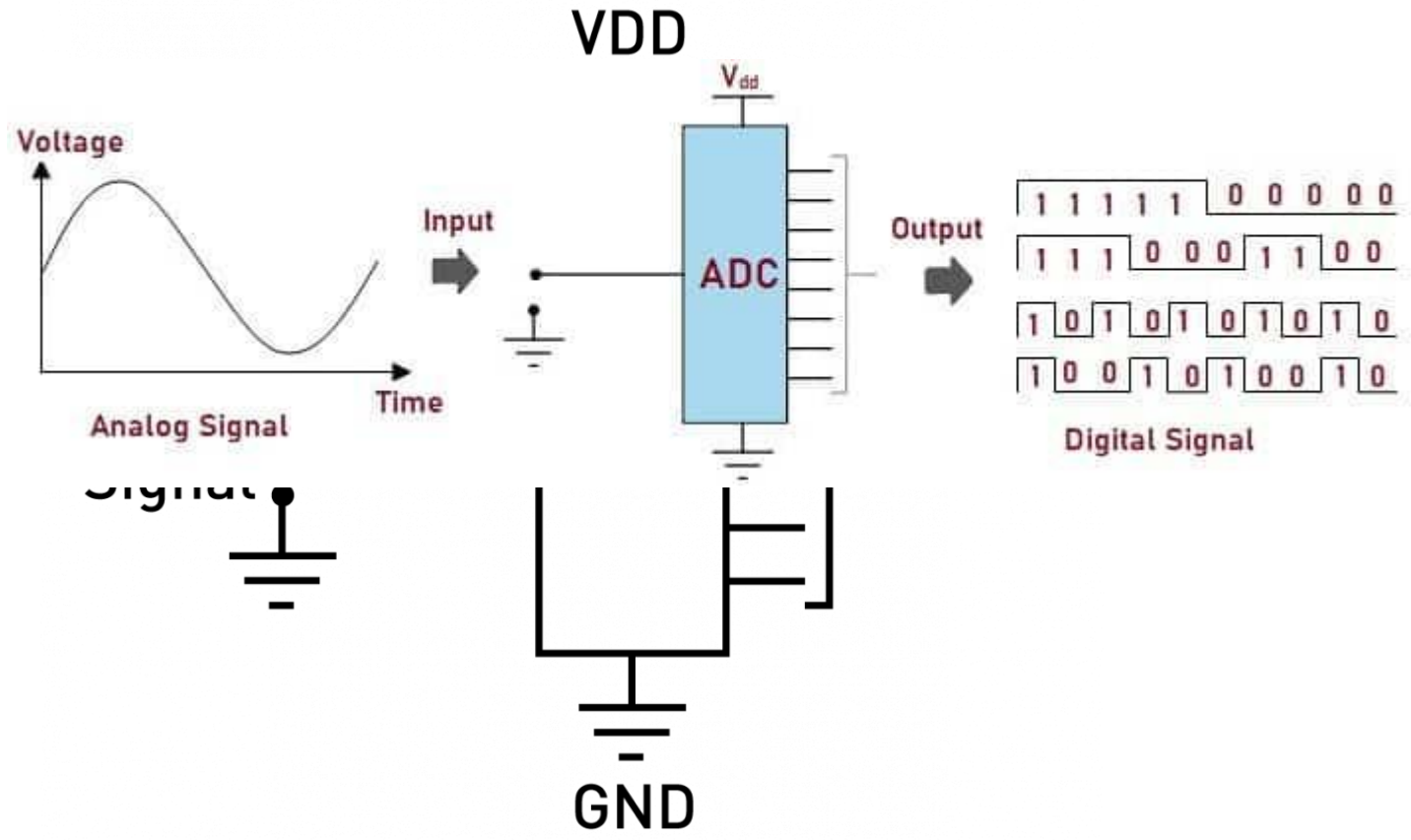
$$V_{out} = V_{in}\left(\frac{R_t}{R_1 + R_t}\right)$$

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

Analog Inputs

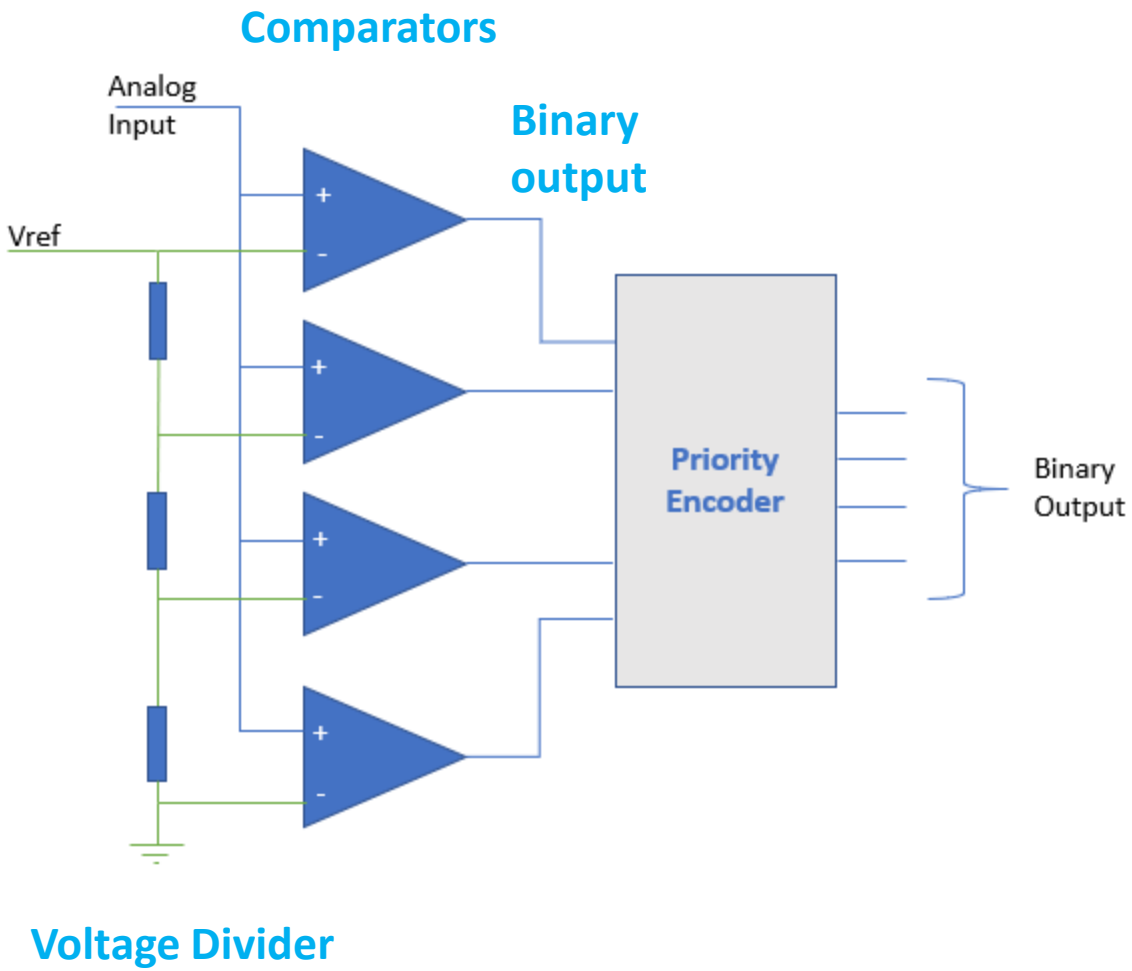
How do we read analog sensors?

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



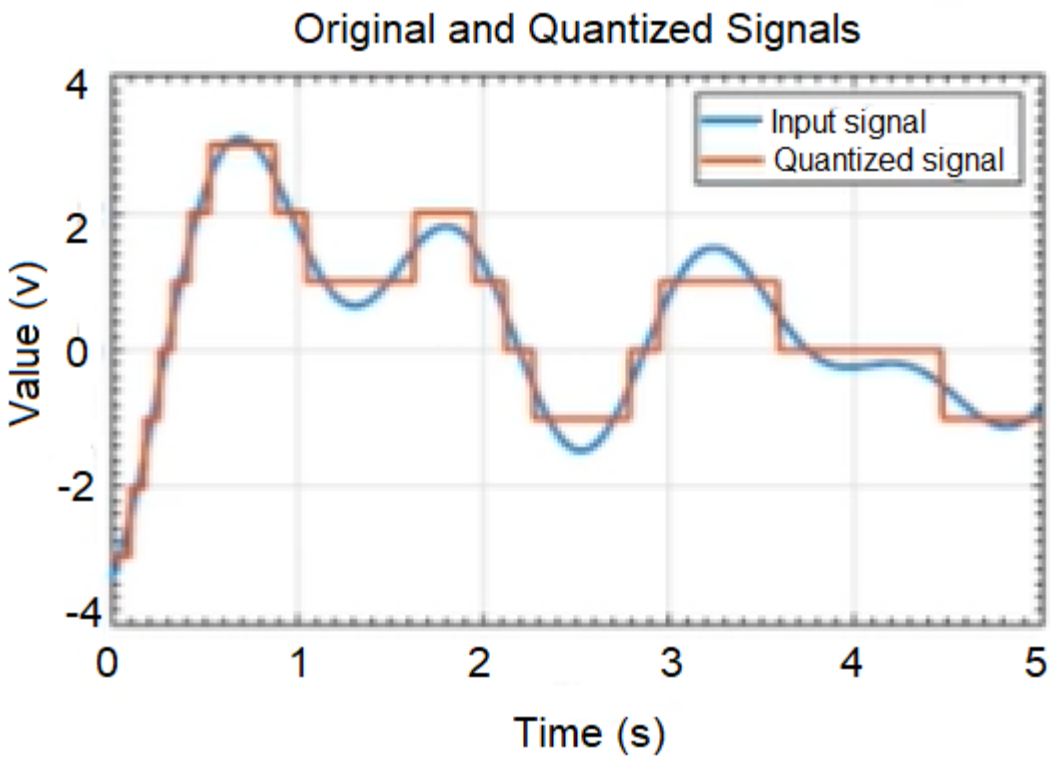
Analog Inputs

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

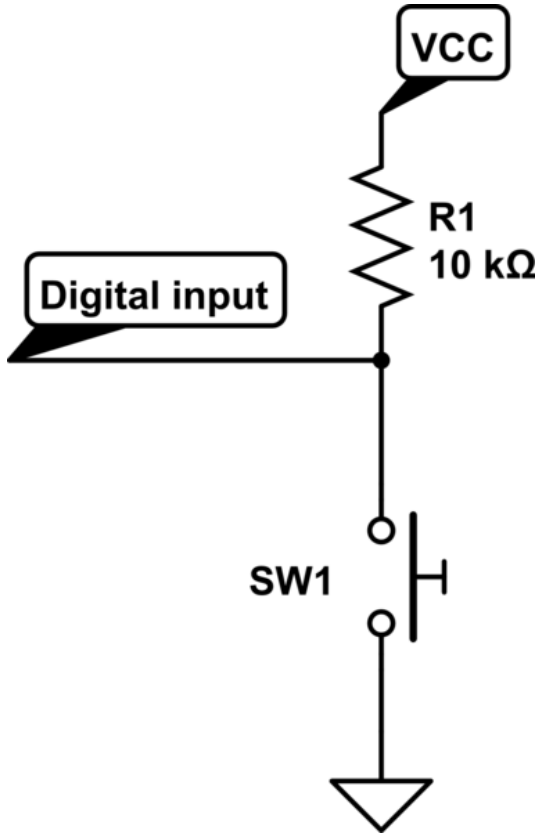
- Analogue inputs are really useful to acquire data
- Quantization error depends on ADC resolution
- Not a scalable approach – for each sensor you need ad additional ADC port
- ADCs can be expensive (for high resolution)
- Sampling rate can be limited
- Sensitive to external noise

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

Digital Inputs

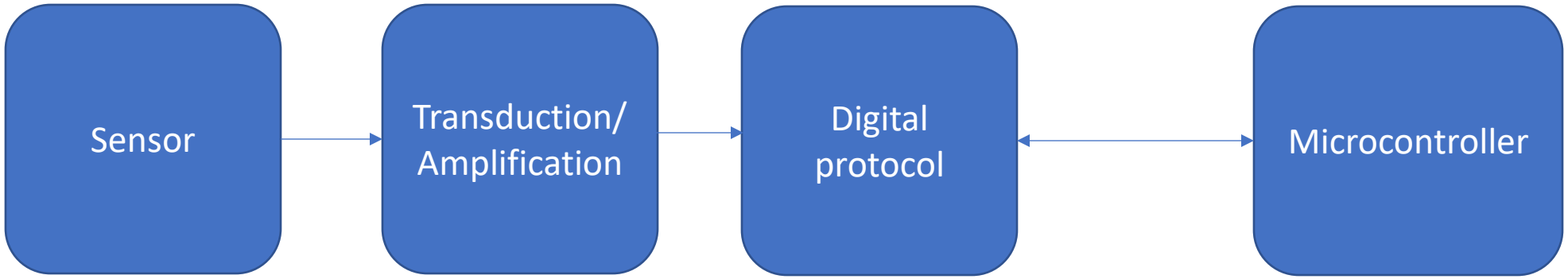
What would be an example of a digital sensor?

Digital Inputs



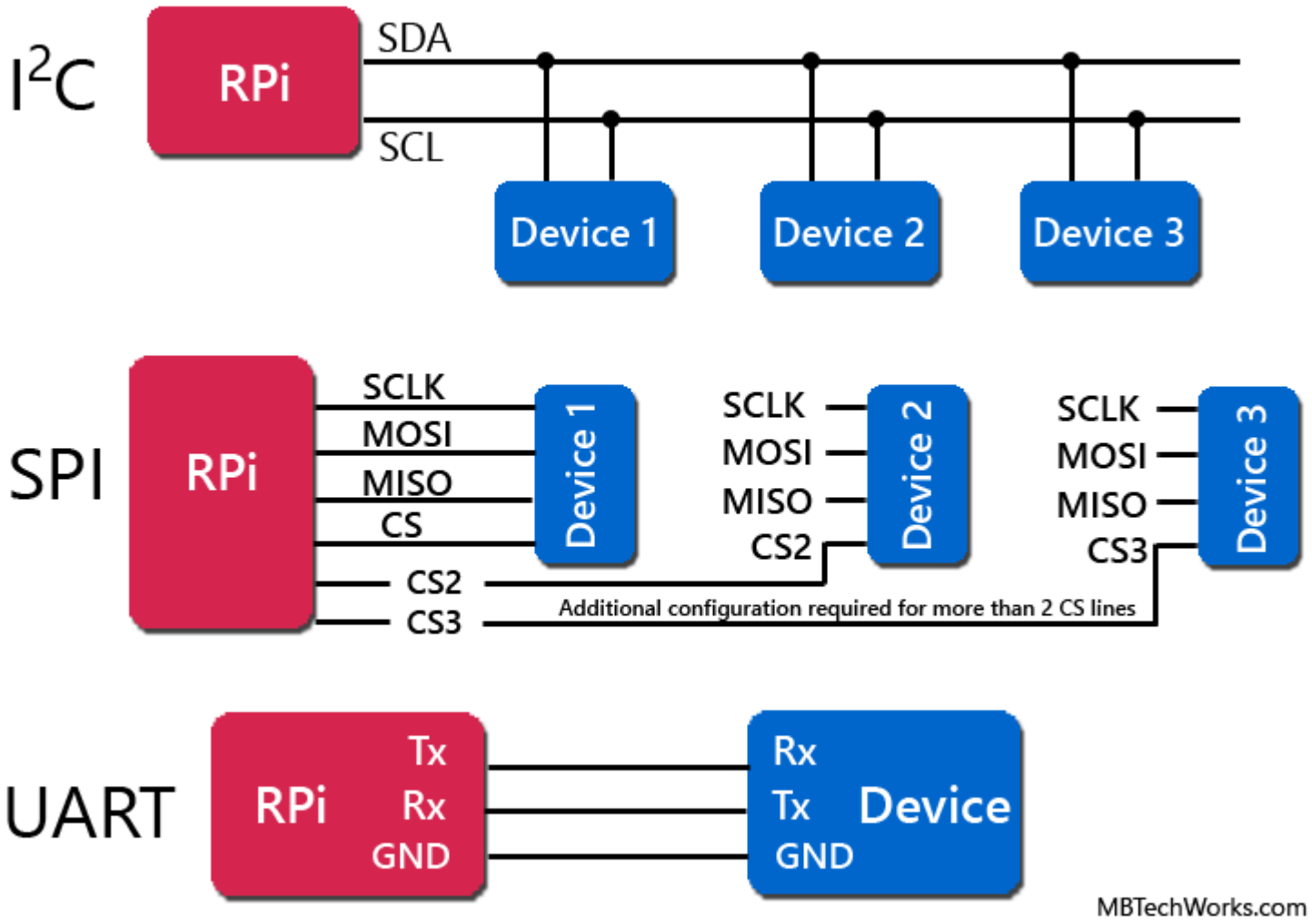
Can we use temporal/digital transmission of data?

Digital Protocols

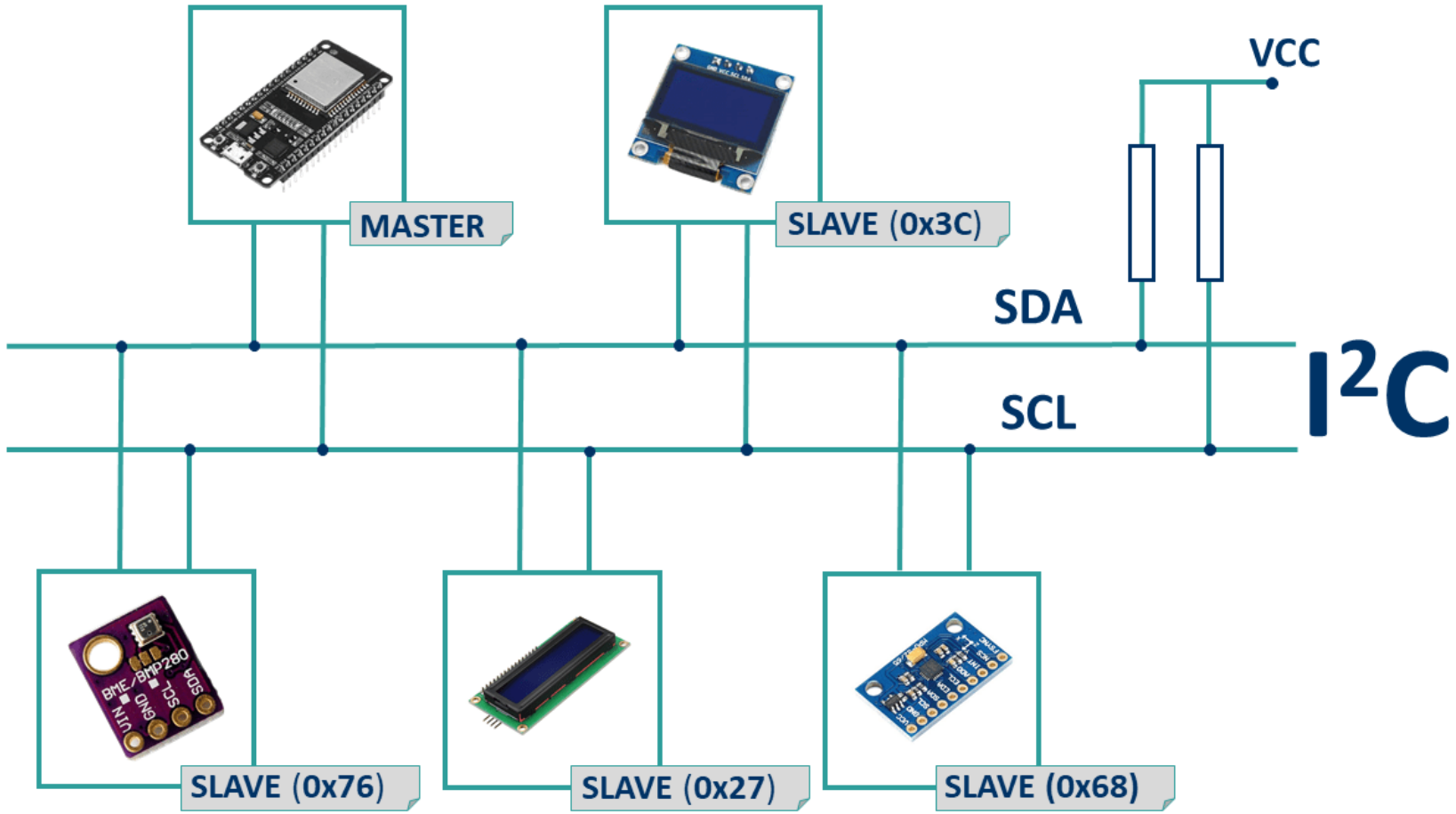


Digital Protocols

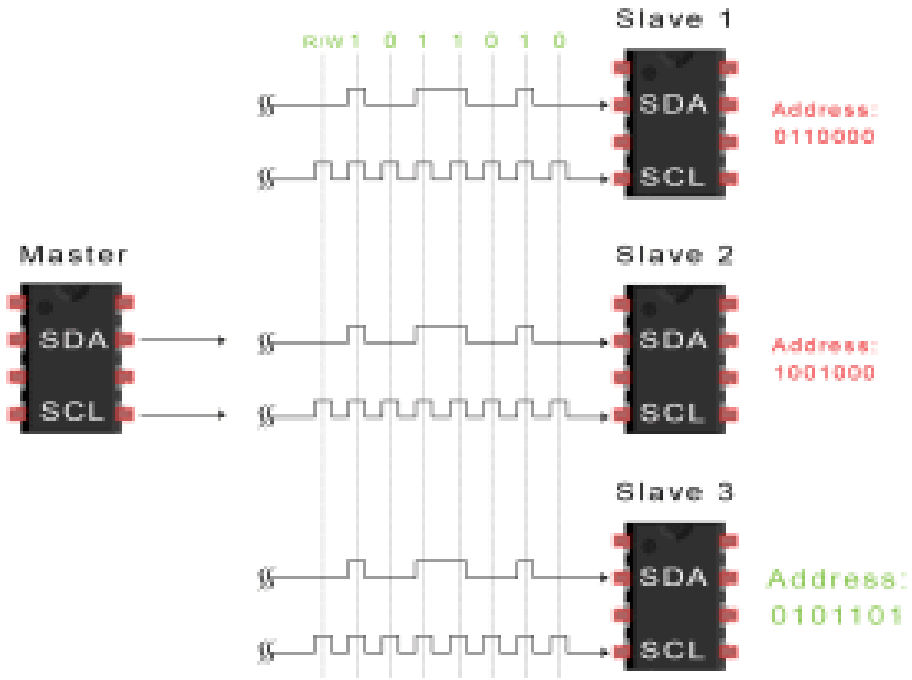
- I2C
- SPI
- UART
- CAN Bus

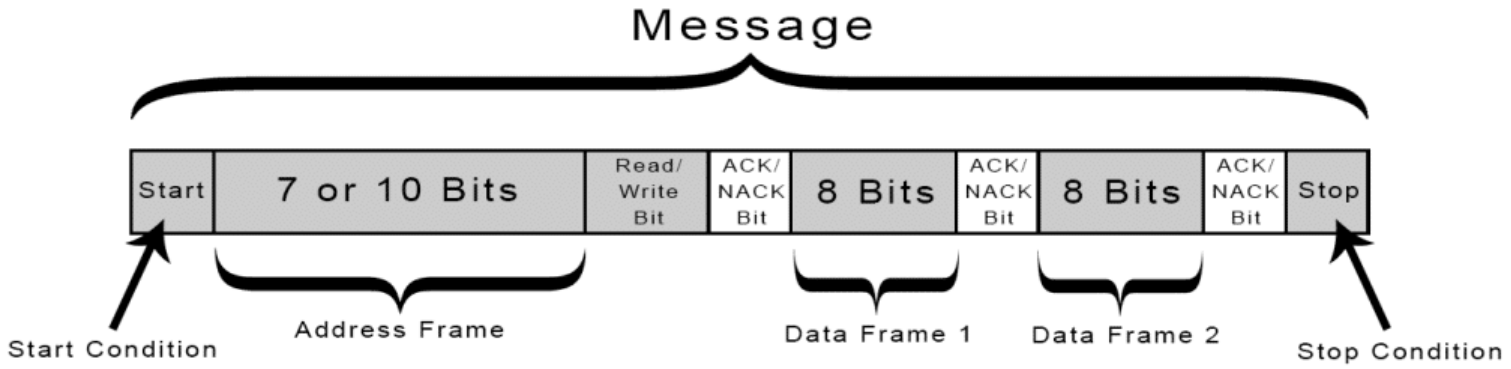


I2C


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

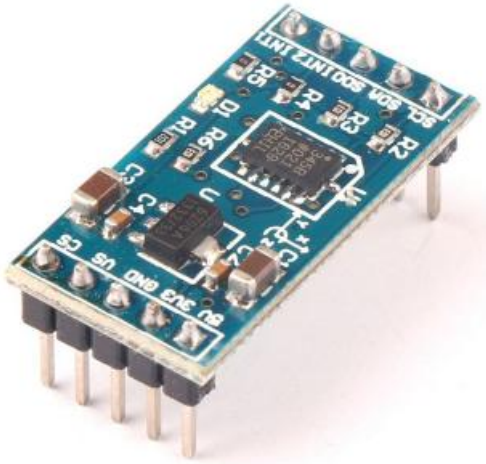
I2C


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

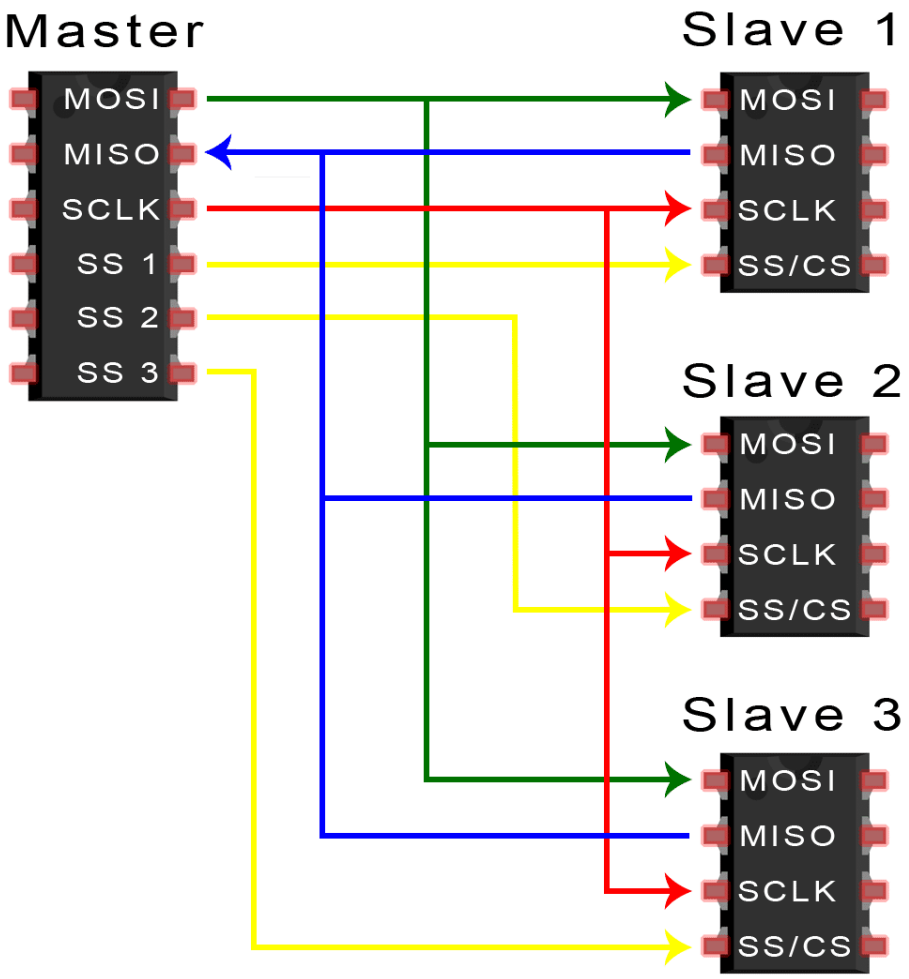


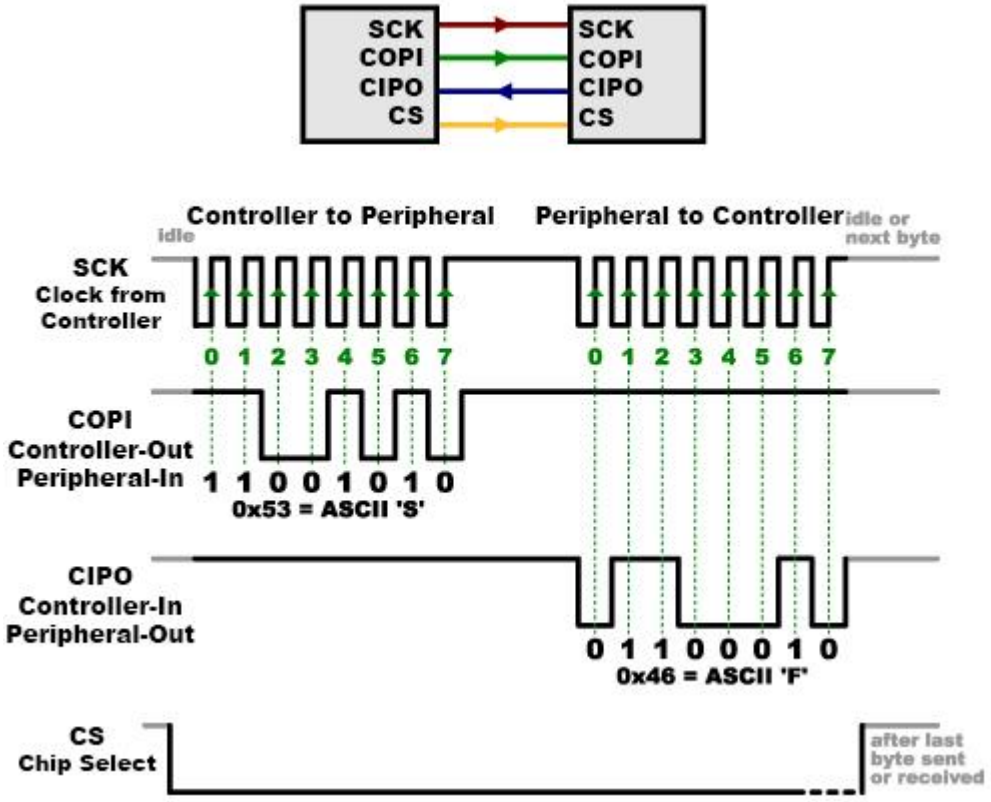
ADXL345
Accelerometer

MMA7660FC
Accelerometer

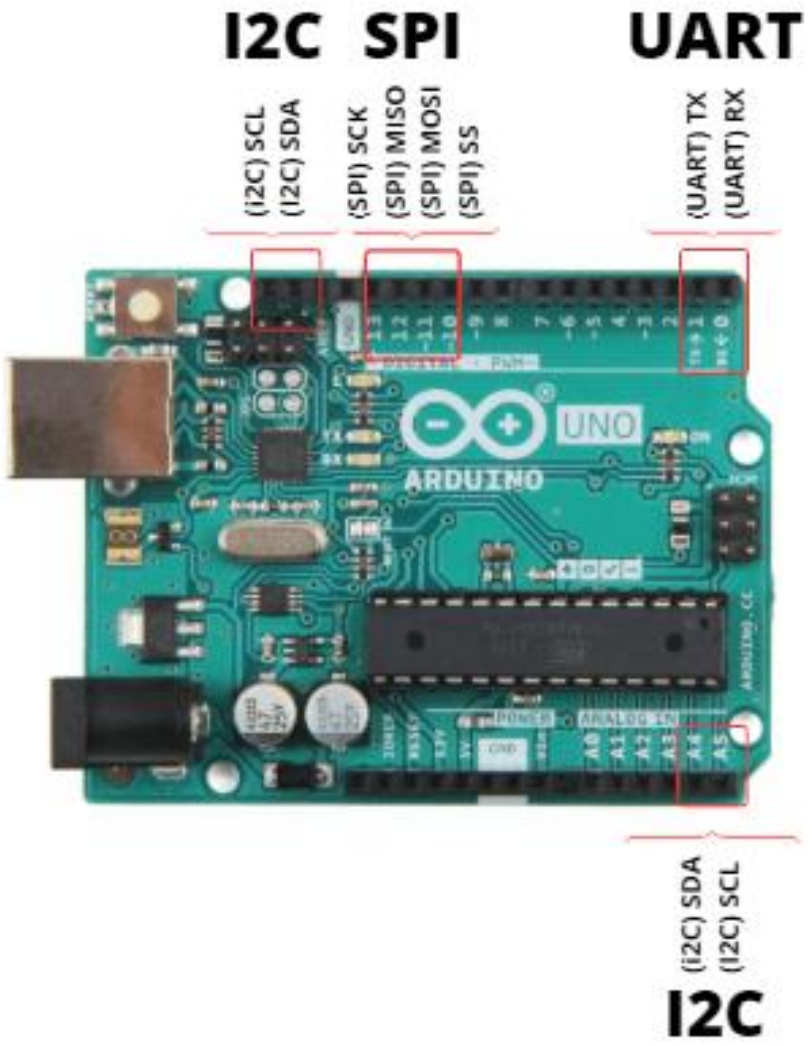
MPU6050
Accelerometer + Gyroscope

SPI (Serial Peripheral Interface)





Digital Protocols



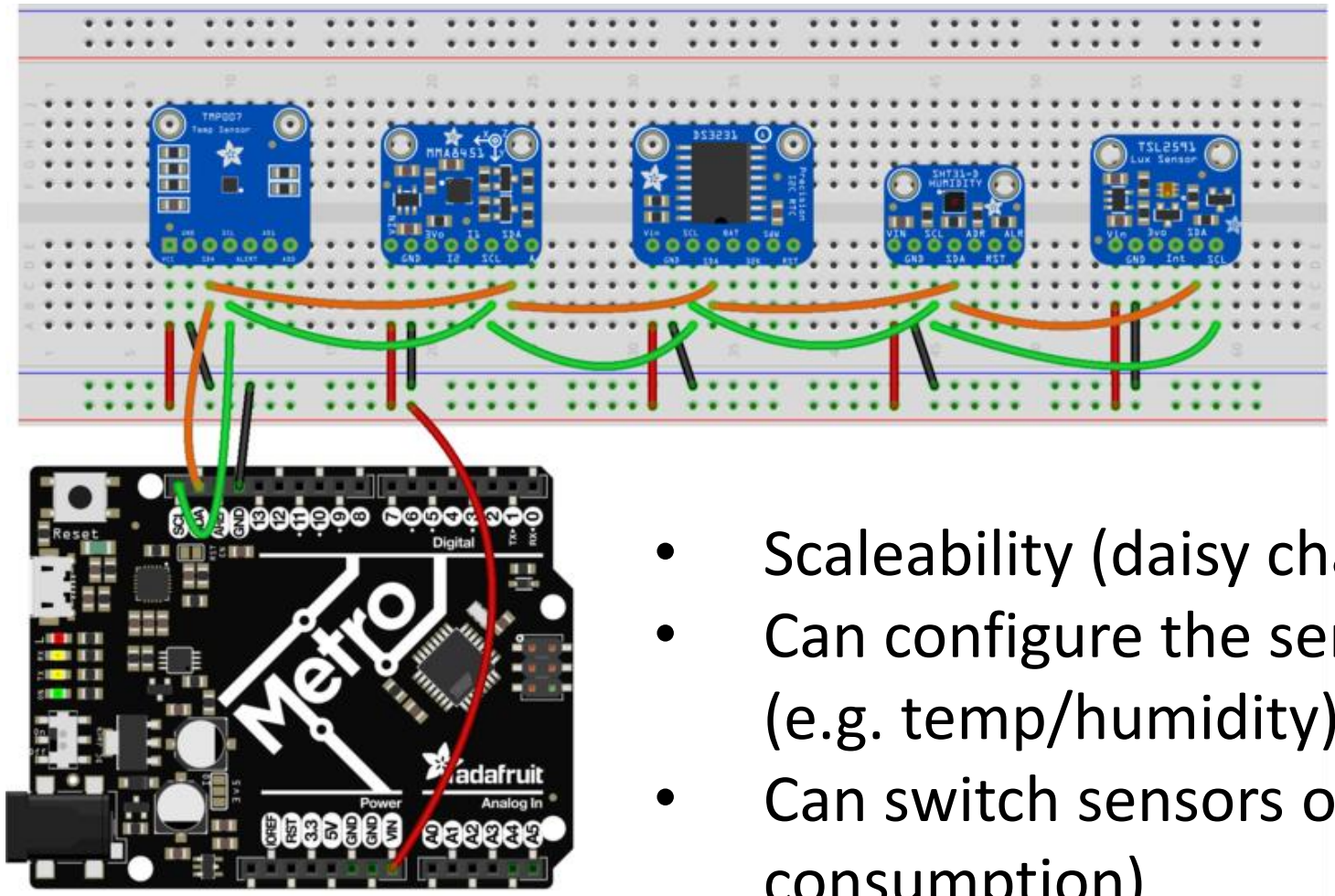
- Microcontrollers have dedicated digital pins for I2C, SPI, UART
- Can have multiple I2C ports
- Many sensors offer SPI or I2C connectivity

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)

fritzing

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

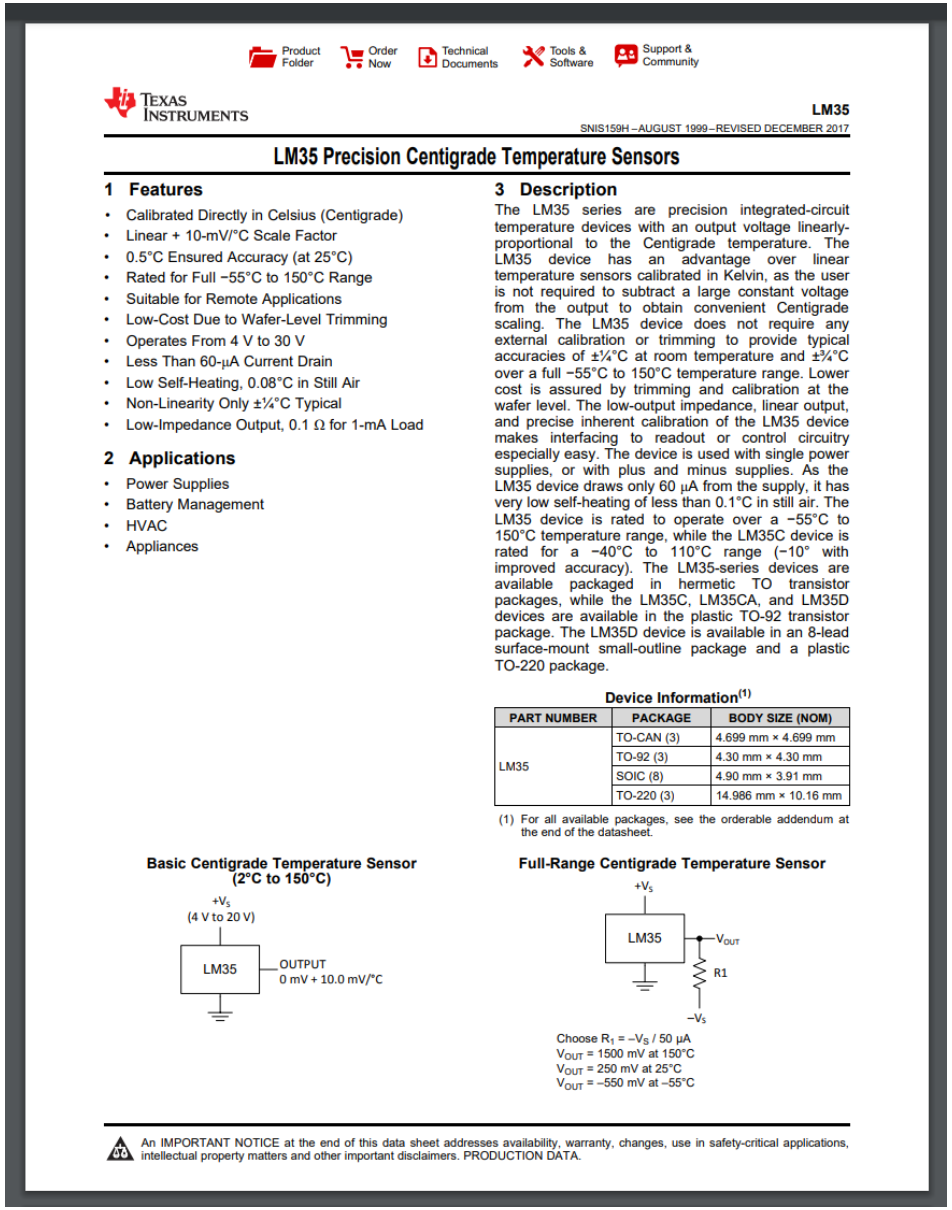
Sensor Design Selection…

- 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

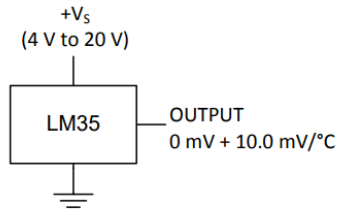


Figure 14. Basic Centigrade Temperature Sensor (2 °C to 150 °C)

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

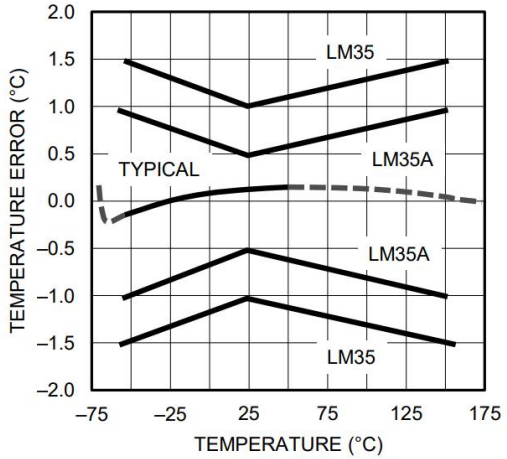


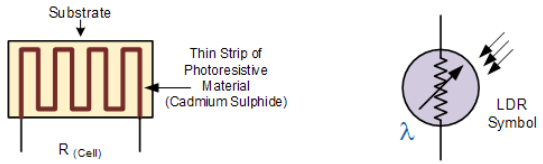
Figure 15. Accuracy vs Temperature (Ensured)

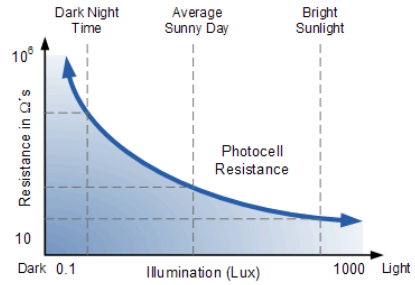
https://www.ti.com/lit/ds/symlink/lm35.pdf

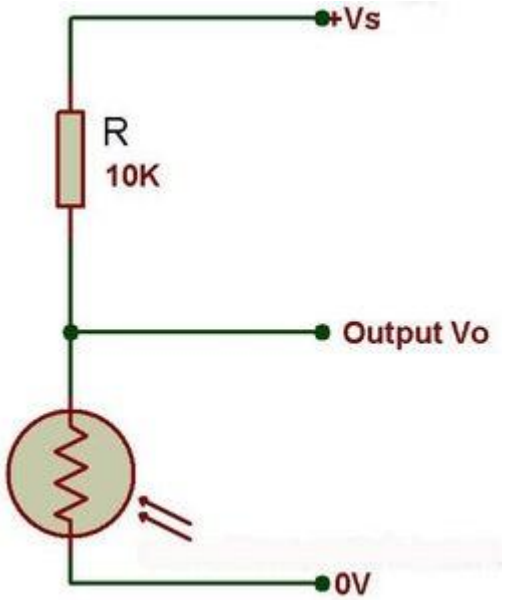
Sensors for you to design and explore

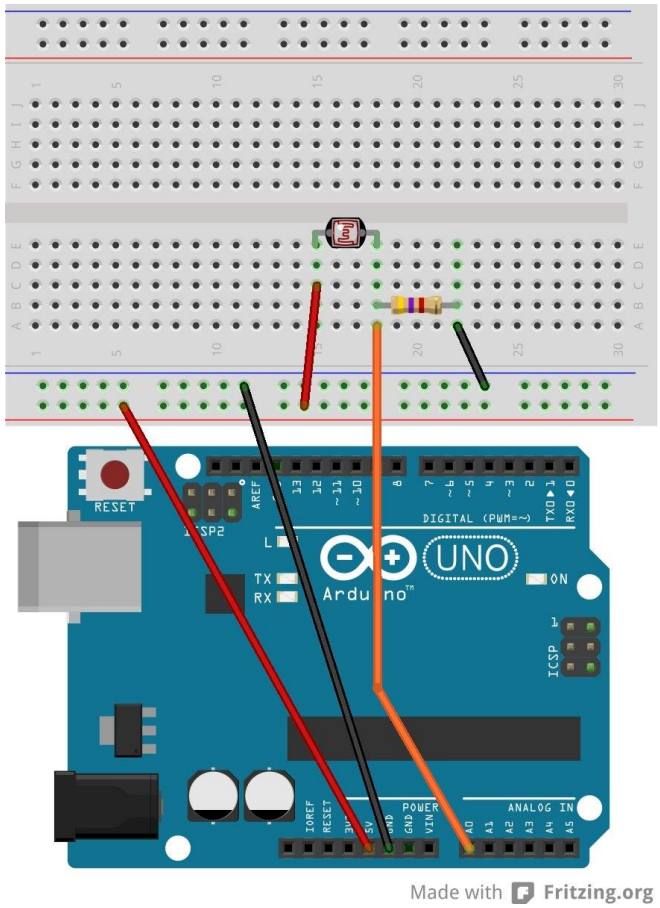
- Load Cell
- LDR Based color Sensors
- Photo-dioded based color Sensor
- Distance Sensor
- LDR (analog)
- Conductivity sensor for contact detection
- Switch

LDR: Light Dependent Resistor





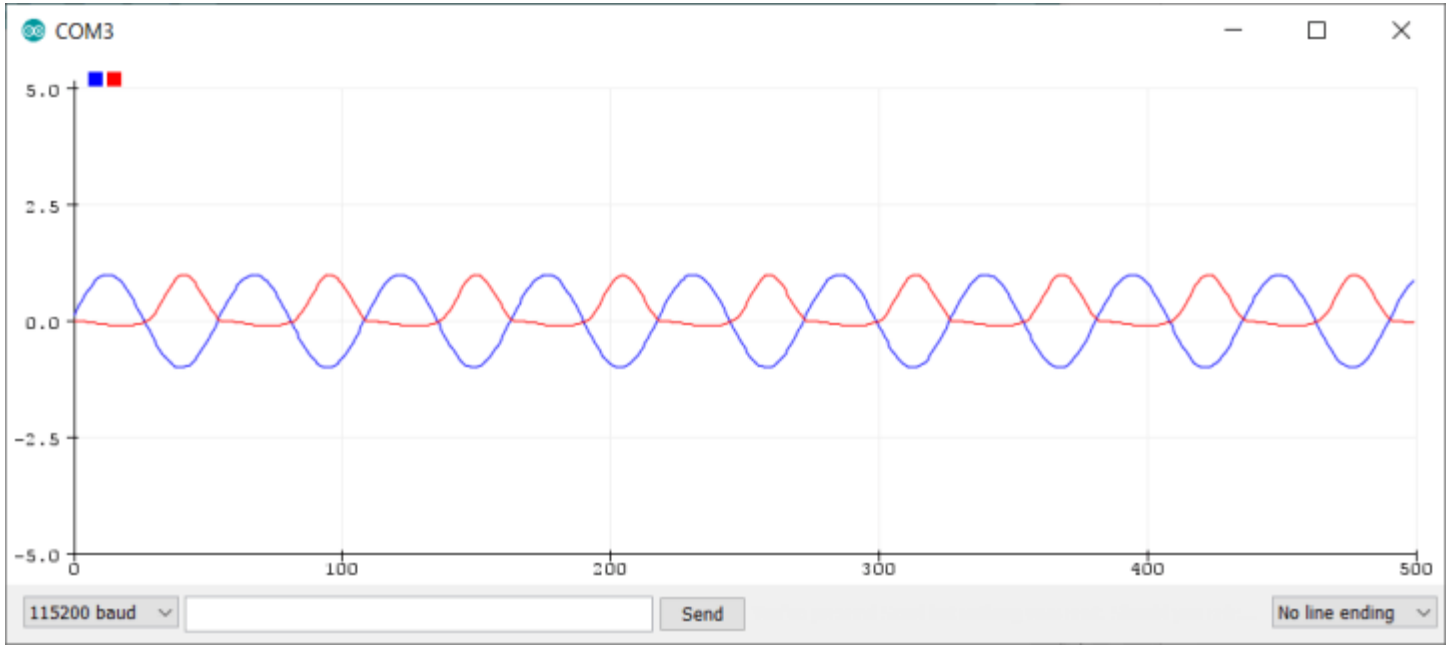




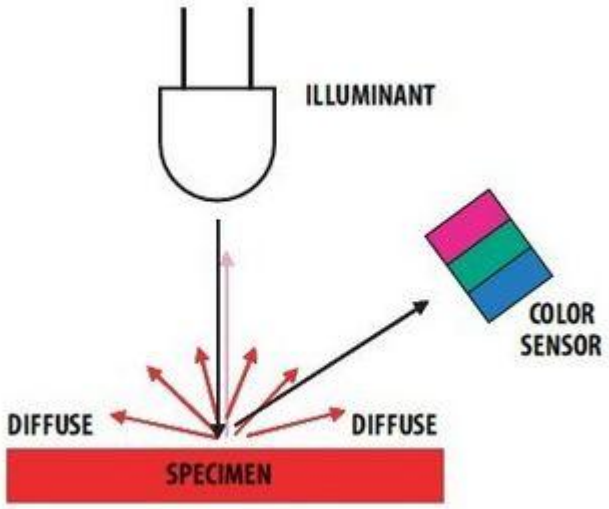
- What resistor value should I choose?
- What happens if you switch the LDR + base resistor around?

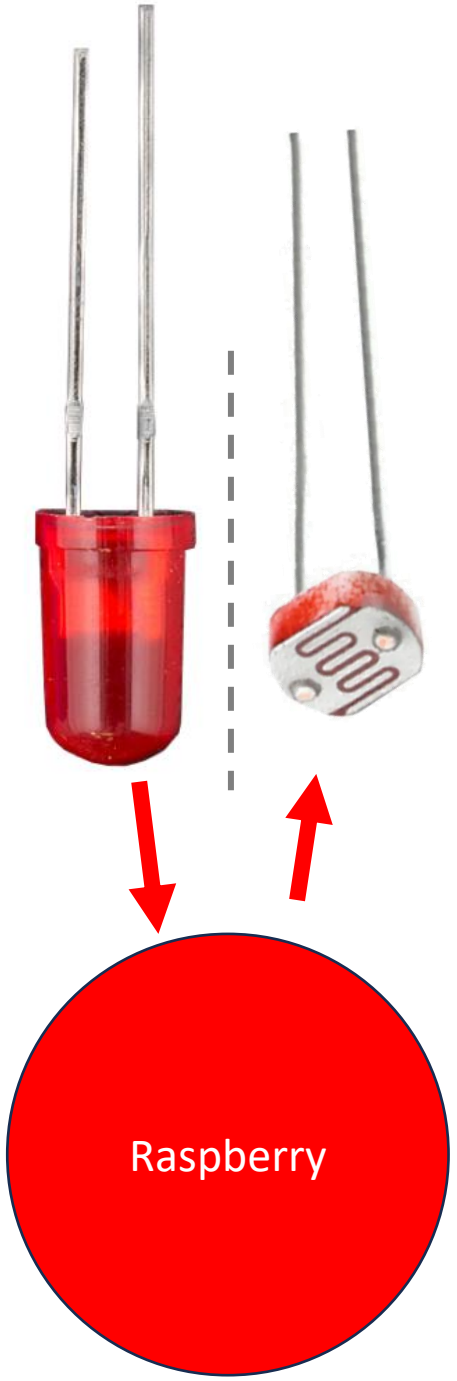
LDR: Light Dependent Resistor

```
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
}
```

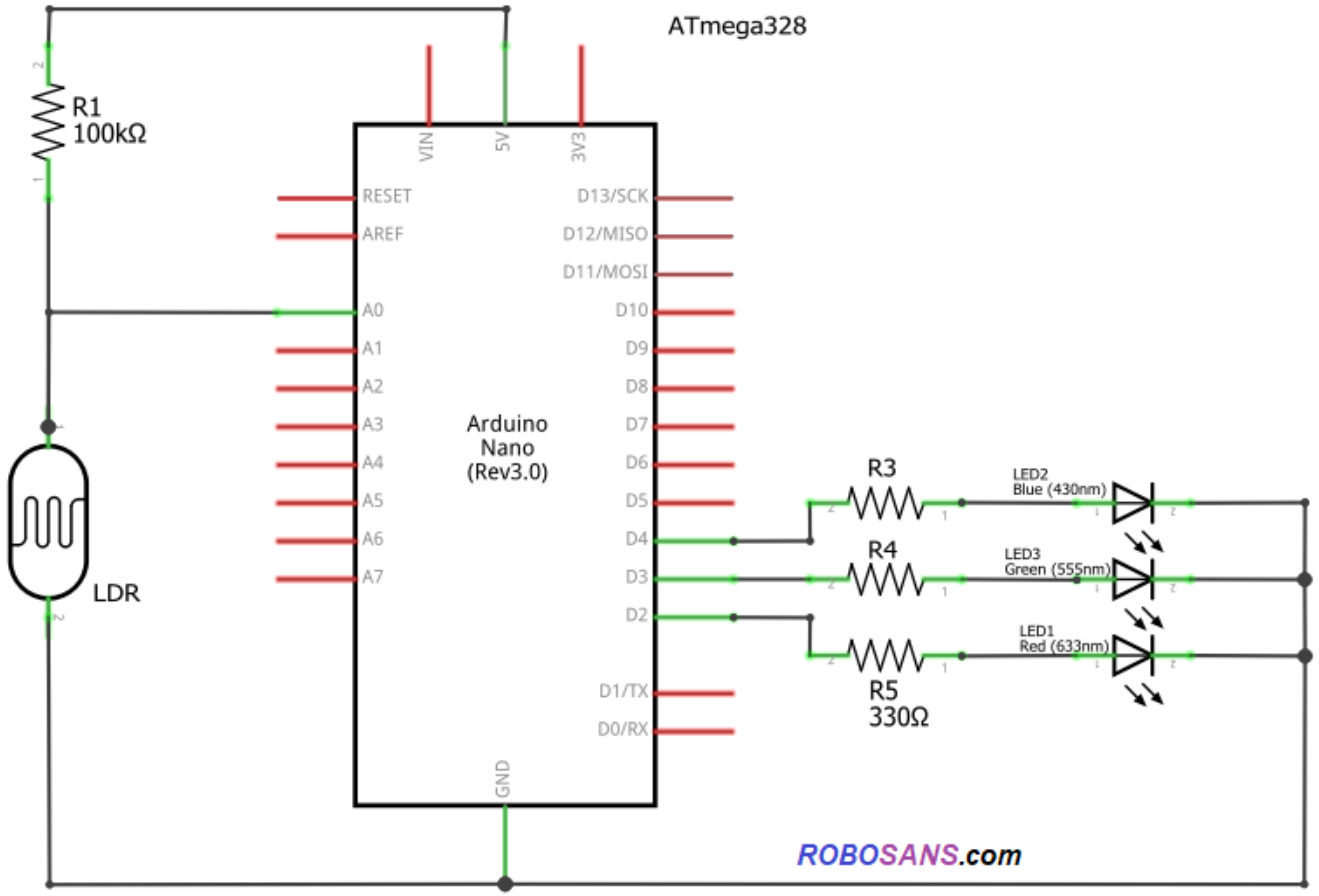


Color Sensor

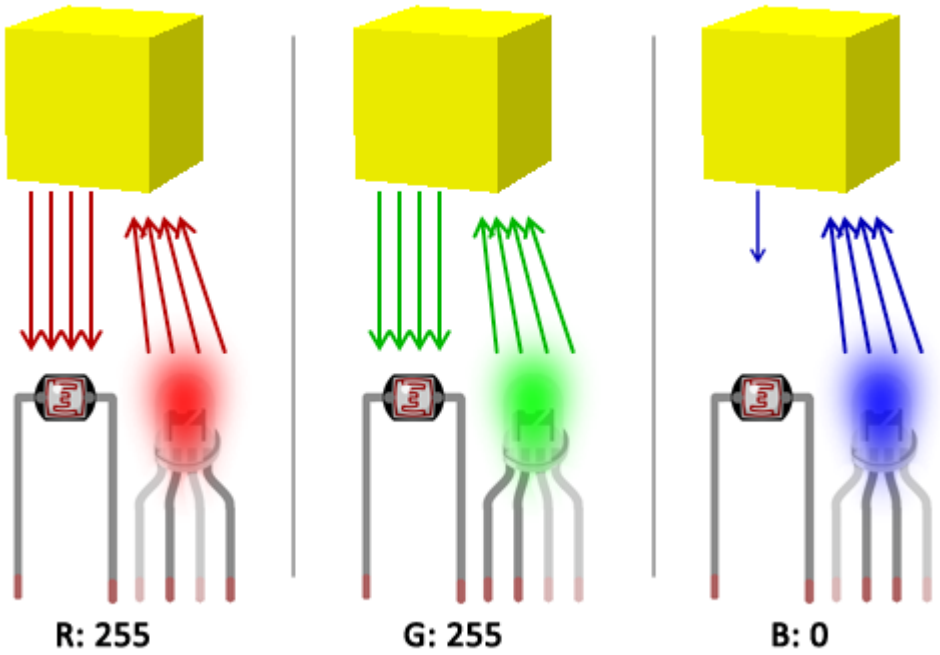




Color Sensor



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.

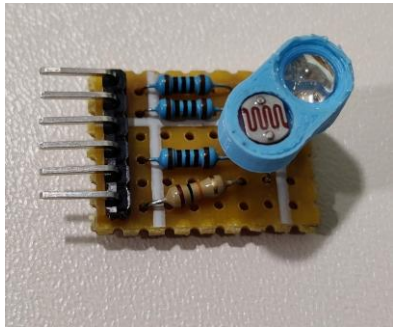
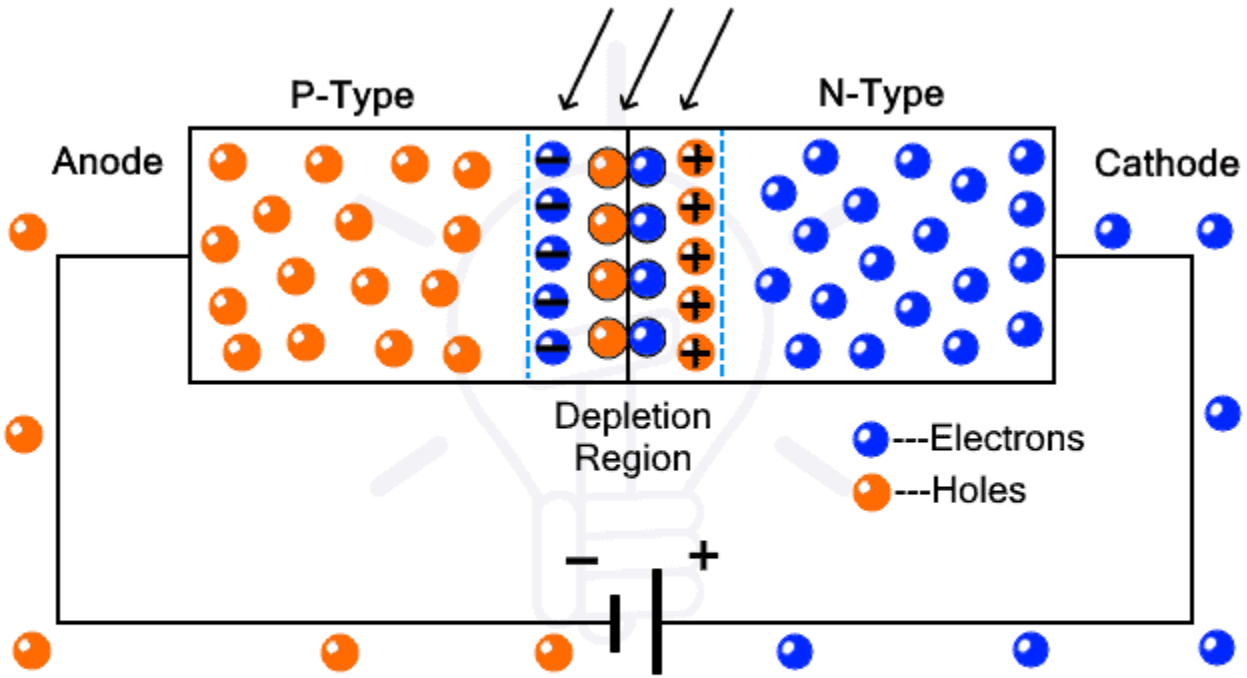
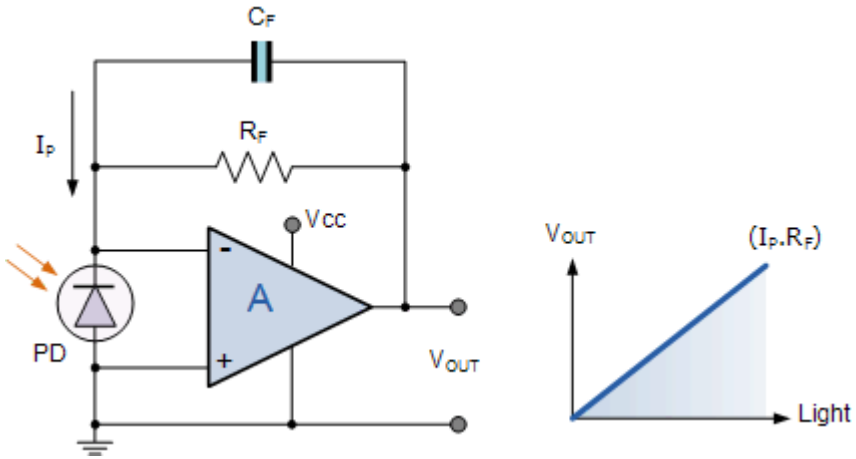


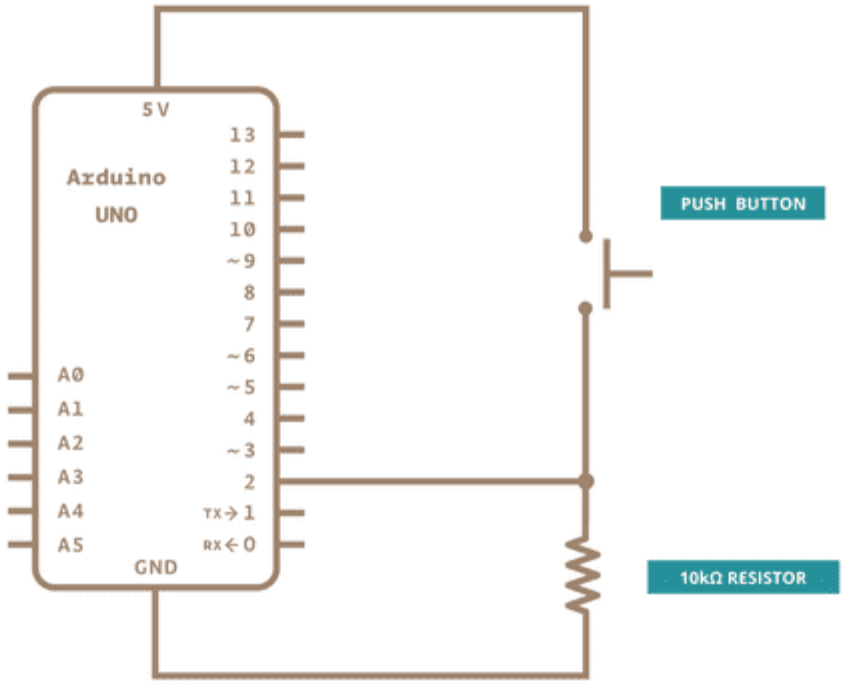
Photo-diode



Photodiode Working

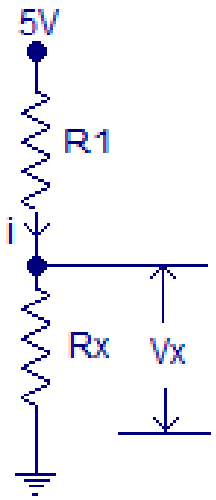


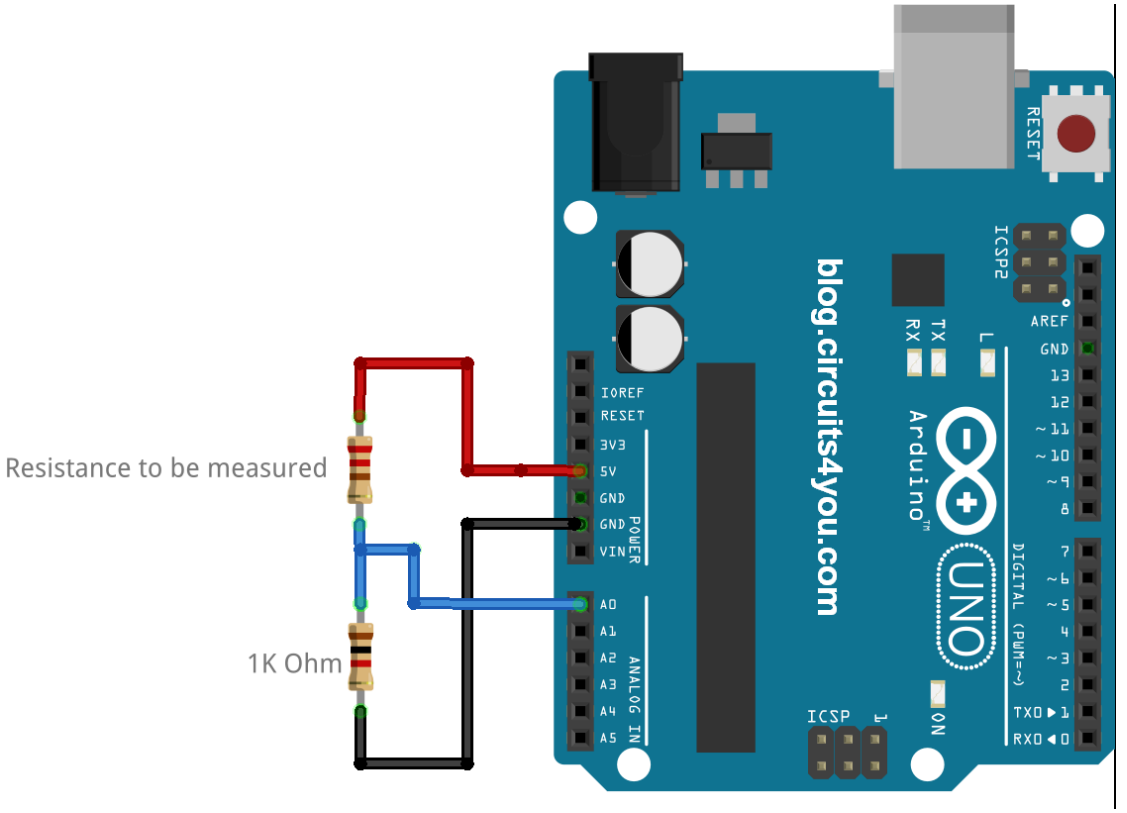
Switch (digital input)




```
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 }
```

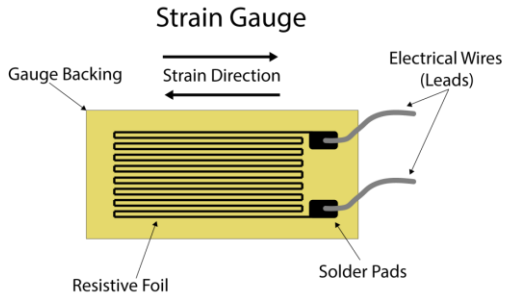
Conductivity Sensor

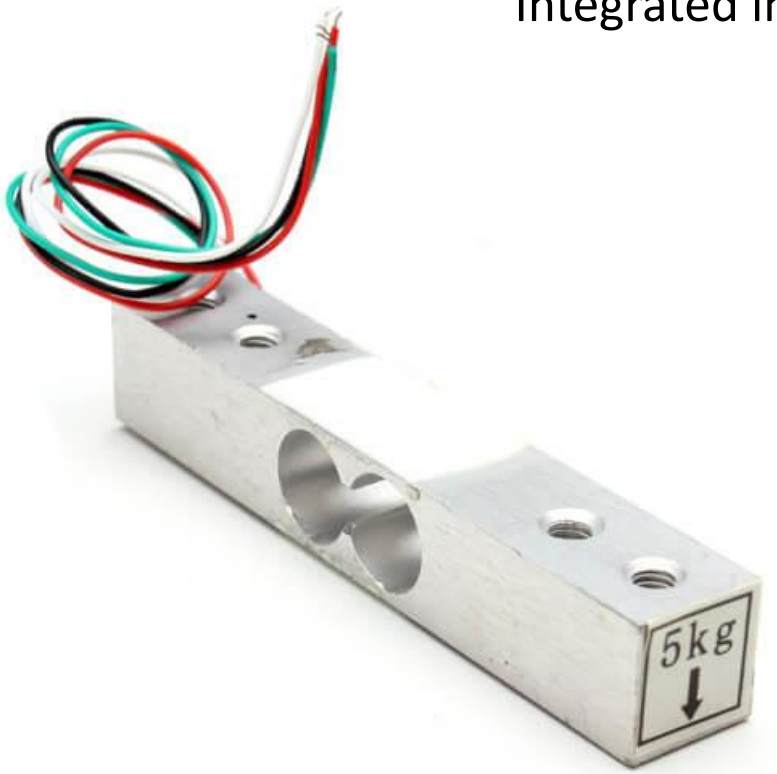




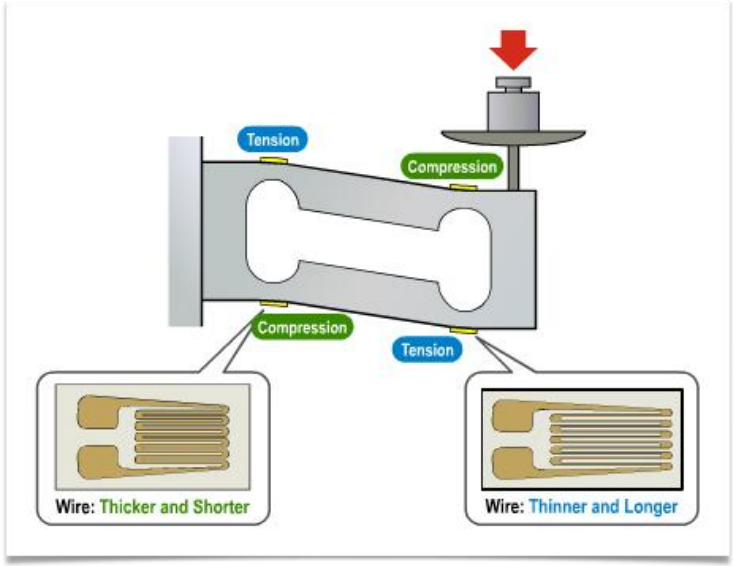
Load Cell

Strain gauge Sensors





Integrated into a Load Cell

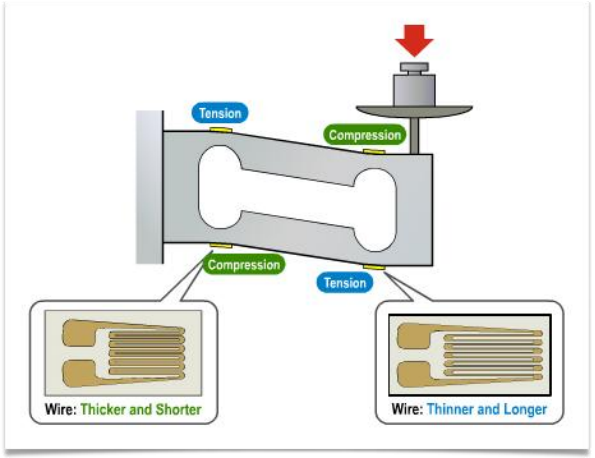


→ Resistance changes with strain

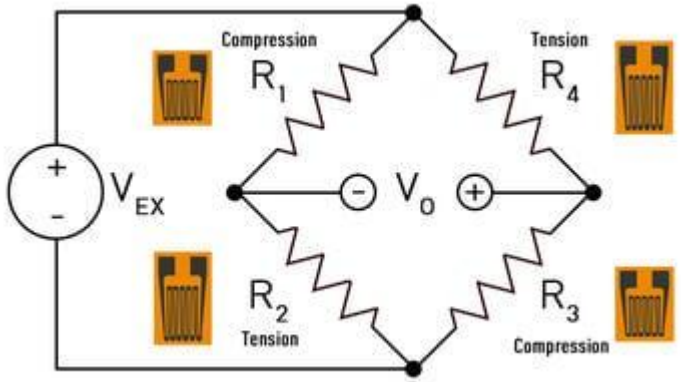
The placement amplifies the tension/compression measured

Load Cell

Load Cell



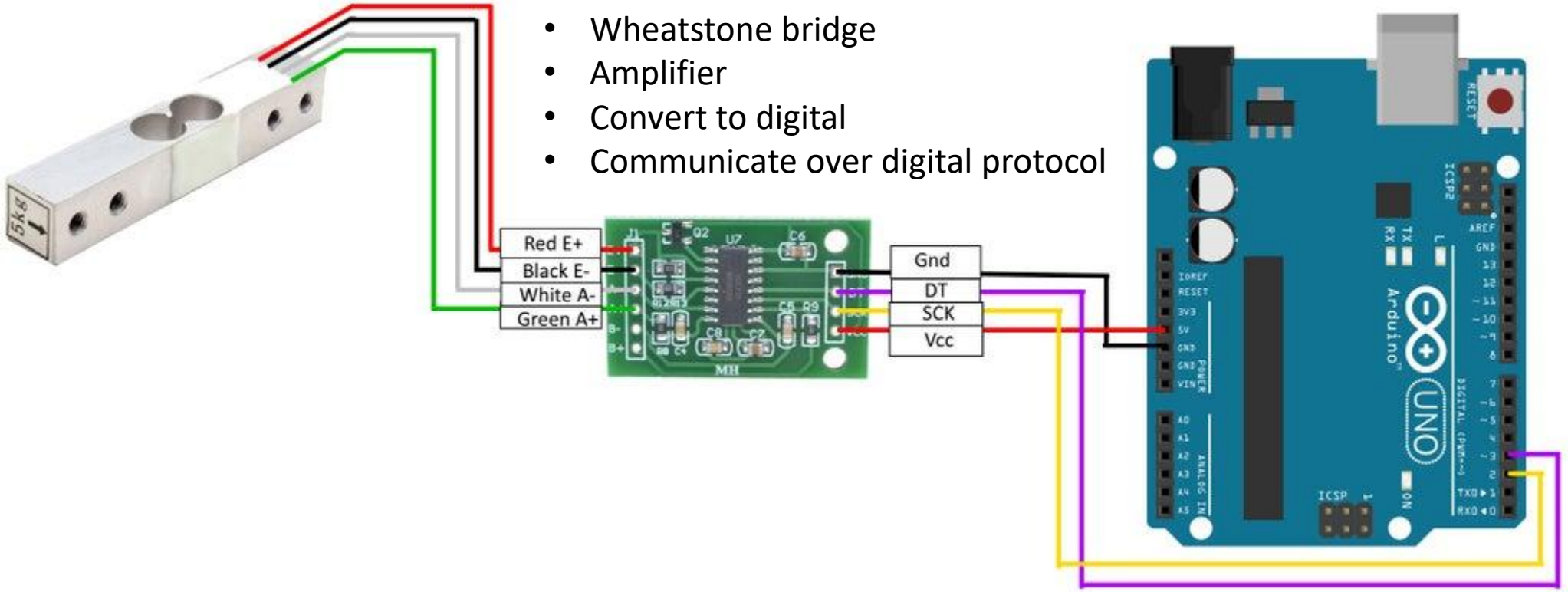
Wheatstone Bridge →
Subtract differences



Amplify differences

Load Cell Amplifier HX711

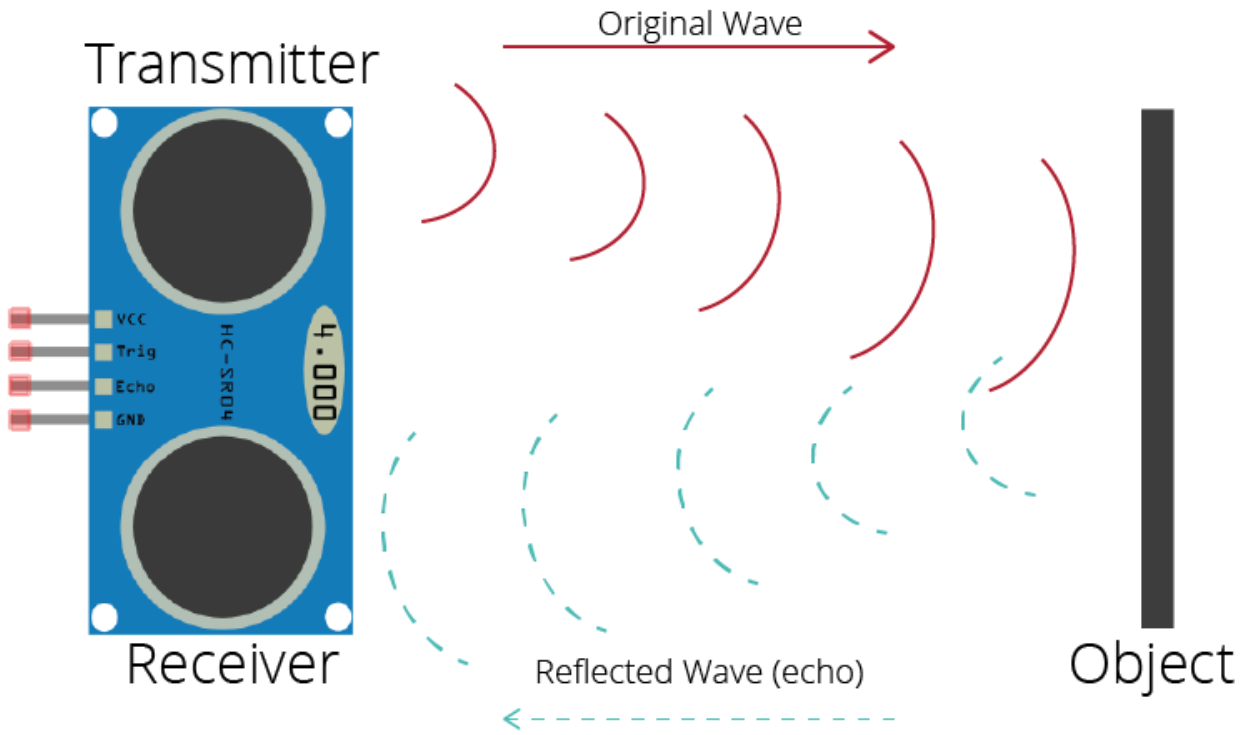
Load Cell

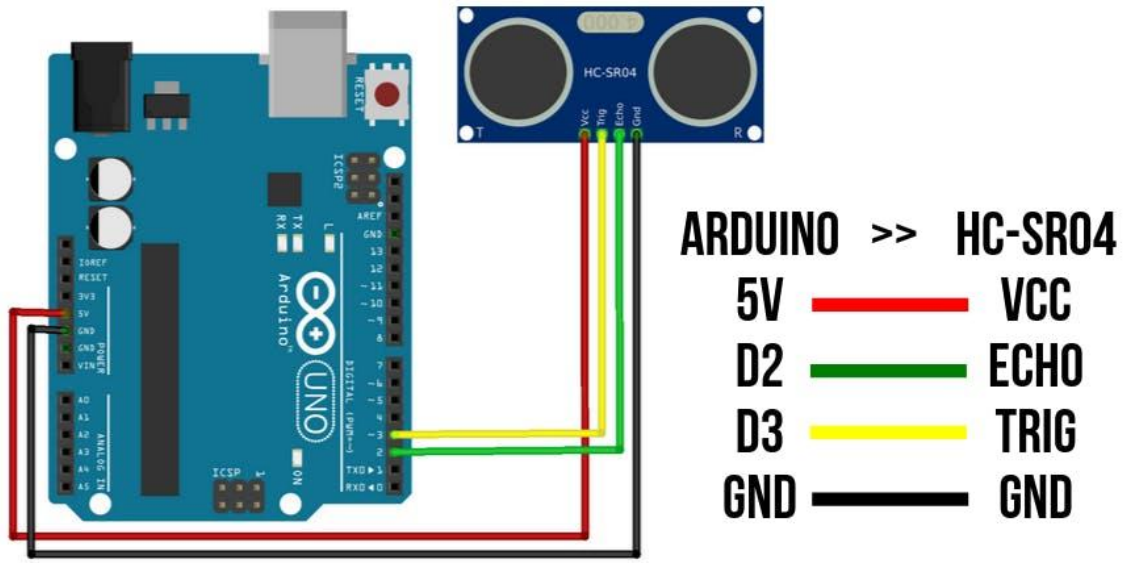


- Wheatstone bridge
- Amplifier
- Convert to digital
- Communicate over digital protocol

Use the Arduino HX111 library – provides the communication
Good tutorial:
https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide

Distance Sensor



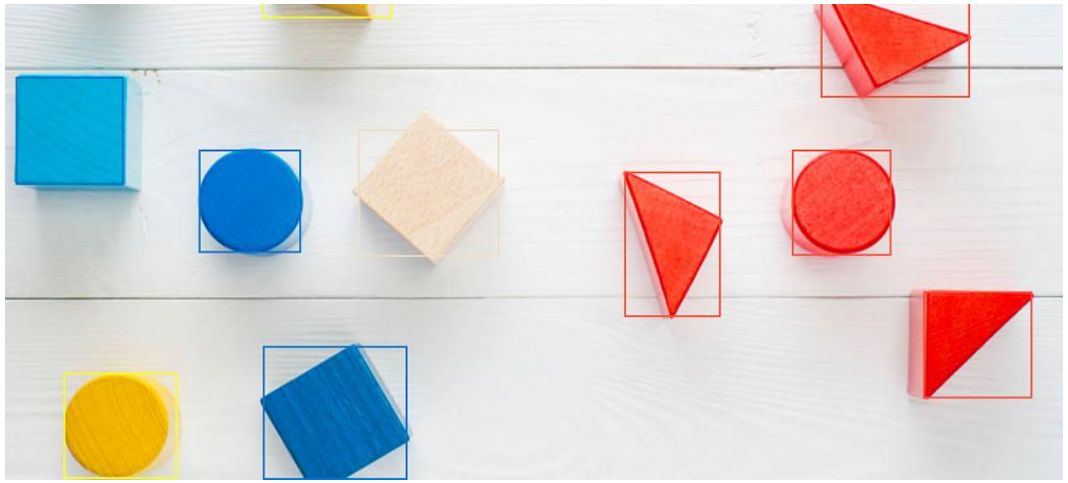


Webcams

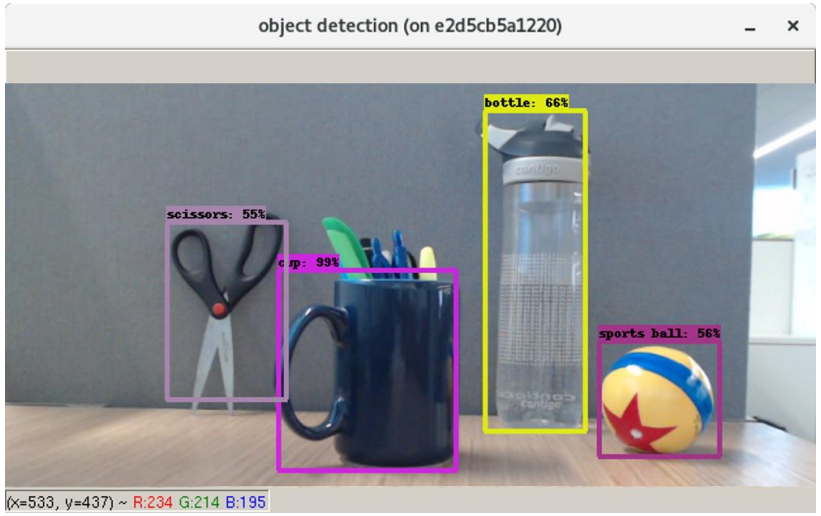
- Very powerful (high information content)
- Low cost, high useage
- Can be used ‘classically’ e.g. detect color/shape
- Can be used with learning based approaches. Many NN are particularly suited for image data

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