



This presentation is released under the terms of the  
**Creative Commons Attribution-Share Alike** license.

You are free to reuse it and modify it as much as you want as long as  
(1) you mention me as being the original author,  
(2) you re-share your presentation under the same terms.

You can download the sources of this presentation here:  
**[github.com/severin-lemaignan/module-introduction-sensors-actuators](https://github.com/severin-lemaignan/module-introduction-sensors-actuators)**

# **ROBOTICS WITH PLYMOUTH UNIVERSITY**

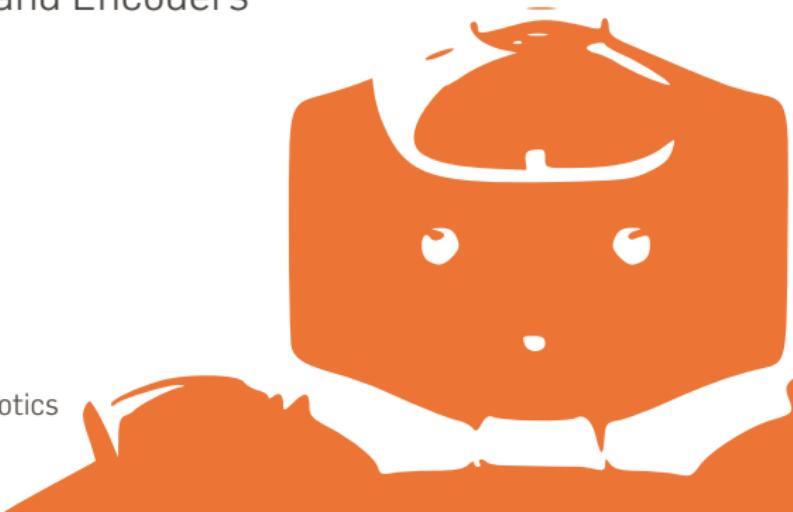
ROCO222

## Intro to Sensors and Actuators

Introduction, Arduinos and Encoders

Séverin Lemaignan

Centre for Neural Systems and Robotics  
**Plymouth University**



# WHO AM I?

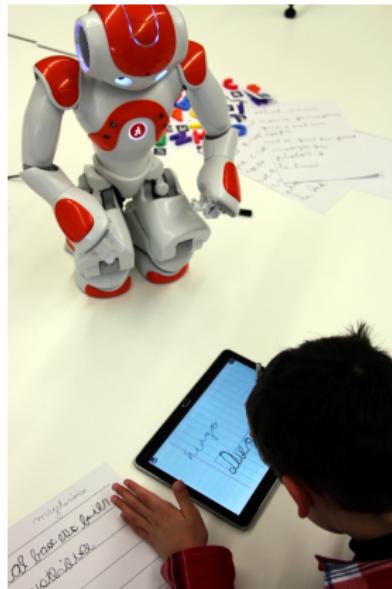
**Séverin Lemaignan**, B316 Portland Square

## Cognitive robotics

Building robots and their artificial intelligence inspired on natural systems, such as developing children

## Human-Robot Interaction

Building robots that can work alongside people, using social cues that people use to communicate



## AVAILABILITY

- I have lots of time (2 hours) to deal with your queries in the laboratory sessions
  - Otherwise, the best thing to do is to email me first and see if I can sort out your query. If not then I will arrange a time to meet up with you.
  - Don't forget – if you have any problems, or even if you are not sure about something - just email me

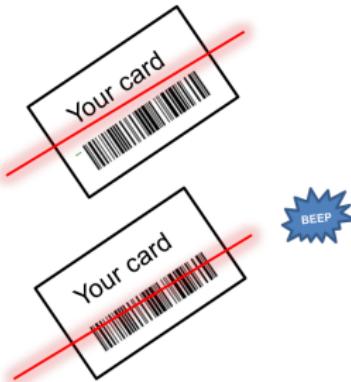
[severin.lemaignan@plymouth.ac.uk](mailto:severin.lemaignan@plymouth.ac.uk)

## REGISTRATION

## **STEP 1 – Activate the device**



**STEP 2 - Move red line across the barcode ...**



## SPECIAL NEEDS

- Anyone with dyslexia or any other special needs that requires any form of support during examinations and tests should email me immediately.
  - Please do not ASSUME that I know of your condition. It takes the University some weeks to process data and forward it to the correct members of staff.
  - Even if you are currently undergoing assessment, it is advisable to contact me and I will organize suitable support for you during your tests and exams.
  - Contact me if any issues relating to harassment, inclusivity or if language/cultural difficulties arise.



## Overview

Labs  
oooooooooooo

Arduino

## Encoders

LABS

- ROCO222 is also taught and supervised in a computer laboratory for TWO HOURS EACH WEEK.
  - There are 2 labs so please go to your timetabled group and pair up with someone

| Day?                        | Time?       | Where?  | Who?          |
|-----------------------------|-------------|---------|---------------|
| Monday – Lab 1/ <b>02</b>   | 09:00–11:00 | SMB 307 | Jake & myself |
| Thursday – Lab 1/ <b>01</b> | 11:00–13:00 | SMB 307 | Jake & myself |

# ASSESSMENT

# ASSESSMENT

## Formative vs Summative

**Formative:** during the term; **Summative:** at the end of the term.

- There will be practical assessments in the form of laboratory practical work (60%)
- **First practical needs to be submitted by 16:00, 13th October 2017**
- Feedback within 20 working days
- Complete lab journal submitted **Thursday 16:00, 11th January 2018**
- There will be a final written exam (40%) – **note that the exam questions may touch upon material discussed during the labs**

ROBOTICS: WHAT ARE YOUR  
EXPECTATIONS?

Go to [www.menti.com](http://www.menti.com) and use the code 60 34 45

# MODULE OVERVIEW



## MODULE AIMS

- Develop an in-depth understanding of **what electrical motors are, how they work** (hands-on!), how they are characterised

# MODULE AIMS

- Develop an in-depth understanding of **what electrical motors are, how they work** (hands-on!), how they are characterised
- Learn how to **control them**, and **program an embedded controller** for your own motor

## MODULE AIMS

- Develop an in-depth understanding of **what electrical motors are, how they work** (hands-on!), how they are characterised
- Learn how to **control them**, and **program an embedded controller** for your own motor
- Get a first **hand-on experience with a complete robotic system**: from the hardware, to the low-level closed-loop control, to system-level programming and visualisation

## MODULE AIMS

- Develop an in-depth understanding of **what electrical motors are, how they work** (hands-on!), how they are characterised
  - Learn how to **control them**, and **program an embedded controller** for your own motor
  - Get a first **hand-on experience with a complete robotic system**: from the hardware, to the low-level closed-loop control, to system-level programming and visualisation
  - **We have a bit of room to touch additional topics**

## LEARNING OUTCOMES

1. Demonstrate knowledge and critical understanding of the operating **characteristics of electrical motors** (brushed, brushless, servo, stepper motors...)

## LEARNING OUTCOMES

1. Demonstrate knowledge and critical understanding of the operating **characteristics of electrical motors** (brushed, brushless, servo, stepper motors...)
  2. Explore (some) **robot sensors**, integrate them into software

## LEARNING OUTCOMES

1. Demonstrate knowledge and critical understanding of the operating **characteristics of electrical motors** (brushed, brushless, servo, stepper motors...)
  2. Explore (some) **robot sensors**, integrate them into software
  3. Comprehend and characterise the effects of **closing the speed and current loops** in drive systems; demonstrate it practically by **creating a closed loop motor system**

## LEARNING OUTCOMES

1. Demonstrate knowledge and critical understanding of the operating **characteristics of electrical motors** (brushed, brushless, servo, stepper motors...)
  2. Explore (some) **robot sensors**, integrate them into software
  3. Comprehend and characterise the effects of **closing the speed and current loops** in drive systems; demonstrate it practically by **creating a closed loop motor system**
  4. Understand the fundamentals of the **ROS middleware**, and use it to control motors and run simple robot visualisations

## LEARNING OUTCOMES

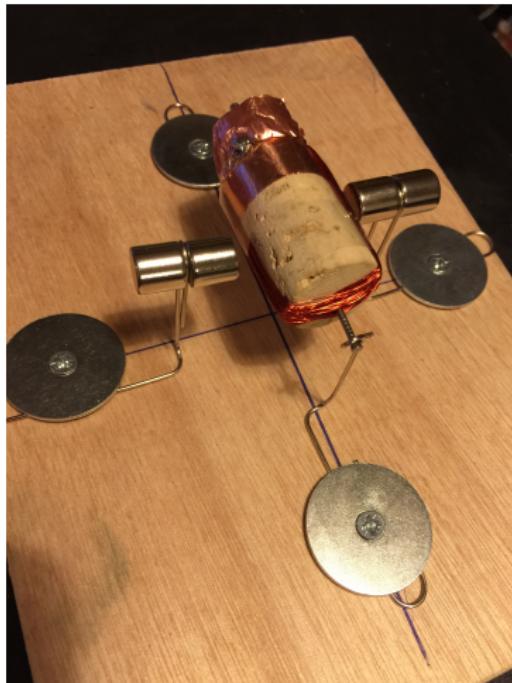
1. Demonstrate knowledge and critical understanding of the operating **characteristics of electrical motors** (brushed, brushless, servo, stepper motors...)
2. Explore (some) **robot sensors**, integrate them into software
3. Comprehend and characterise the effects of **closing the speed and current loops** in drive systems; demonstrate it practically by **creating a closed loop motor system**
4. Understand the fundamentals of the **ROS middleware**, and use it to control motors and run simple robot visualisations
5. Acquire the **basics of software engineering**; feel confident when using the **Linux command-line**; know how to use and reflect on **document/code versioning and sharing** (with git)

# IN YOUR CURRICULUM

- Last year: quite a lot of electronics, **ROCO103PP?**
- Next term: **ROCO224** (Martin Stoelen) – kinematics, mechanical engineering, robot control
- Next year: **ROCO318** – sensors, algorithms, AI (path planning, localisation)

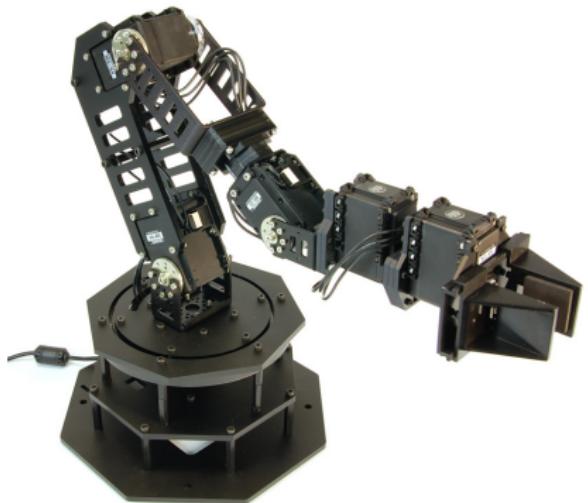
THIS YEAR: ROCO222 AND ROCO224

From...



# THIS YEAR: ROCO222 AND ROCO224

...to



# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless
- **Week 5** – Closed-loop motor control, PWM, PID, H-bridge

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless
- **Week 5** – Closed-loop motor control, PWM, PID, H-bridge
- **Week 6** – Induction motors

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless
- **Week 5** – Closed-loop motor control, PWM, PID, H-bridge
- **Week 6** – Induction motors
- **Week 7** – Servo-motors & stepper motors

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless
- **Week 5** – Closed-loop motor control, PWM, PID, H-bridge
- **Week 6** – Induction motors
- **Week 7** – Servo-motors & stepper motors
- **Week 9** – Software engineering for robotics

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless
- **Week 5** – Closed-loop motor control, PWM, PID, H-bridge
- **Week 6** – Induction motors
- **Week 7** – Servo-motors & stepper motors
- **Week 9** – Software engineering for robotics
- **Week 10** – ROS middleware, joint state, kinematics 101, visualisation

# THIS MODULE: SYLLABUS

- **Week 1** – Arduino + Encoders
- **Week 2** – The physics behind motors: electromagnetism, induction, magnetic force
- **Week 4** – DC motors, brushed & brushless
- **Week 5** – Closed-loop motor control, PWM, PID, H-bridge
- **Week 6** – Induction motors
- **Week 7** – Servo-motors & stepper motors
- **Week 9** – Software engineering for robotics
- **Week 10** – ROS middleware, joint state, kinematics 101, visualisation
- **Week 11** – Sensors

# LABORATORY COURSEWORK

# THIS WEEK: LAB JOURNAL & COMMAND-LINE 101

- Document versioning: what is GIT?
- Creating & managing a lab journal with GIT and markdown
- Linux command-line
- Remotely connecting to a robot

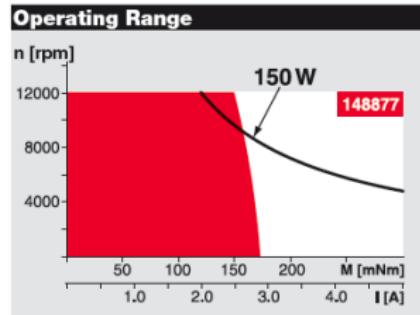
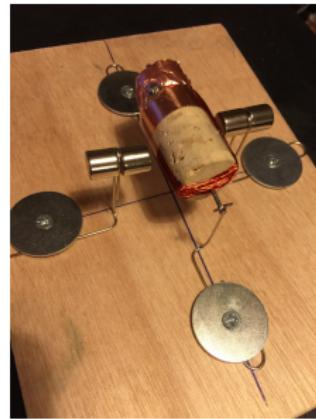
## Tomorrow's practical

If you want to, you can come from 11:00 to 13:00 to finish Monday's practical (including connecting to the robot)

# BUILD A DC MOTOR

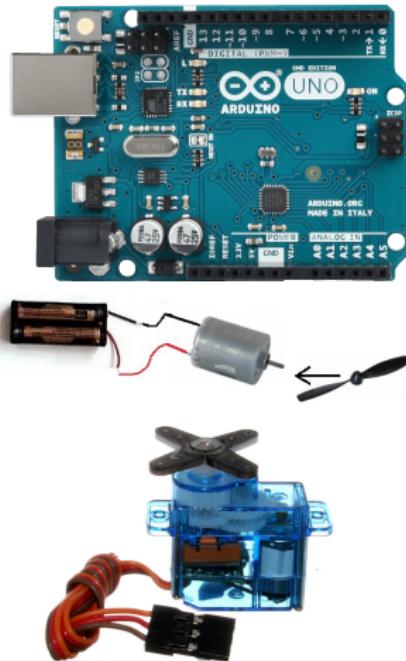
Build a DC brushed motor from first principles

- Wind armature
- Position field magnets
- Build commutator
- Test operation
- Measure characteristics



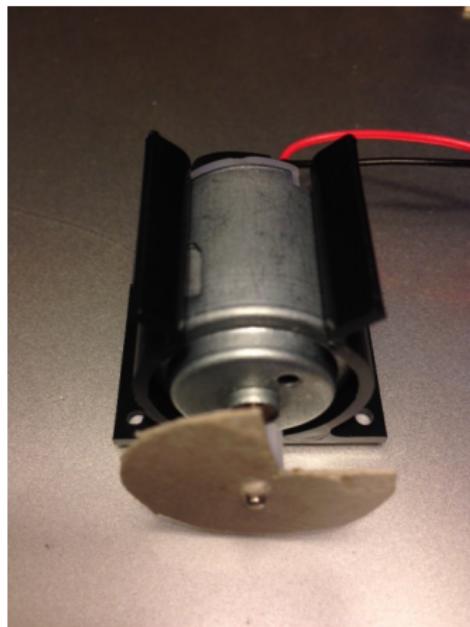
# INTRODUCTION TO ARDUINO AND SIMPLE MOTOR CONTROL

- Controlling your DC motor using an Arduino
  - Build a speed-controlled electric fan e.g. use temperature sensor
  - Control an RC servo using an Arduino

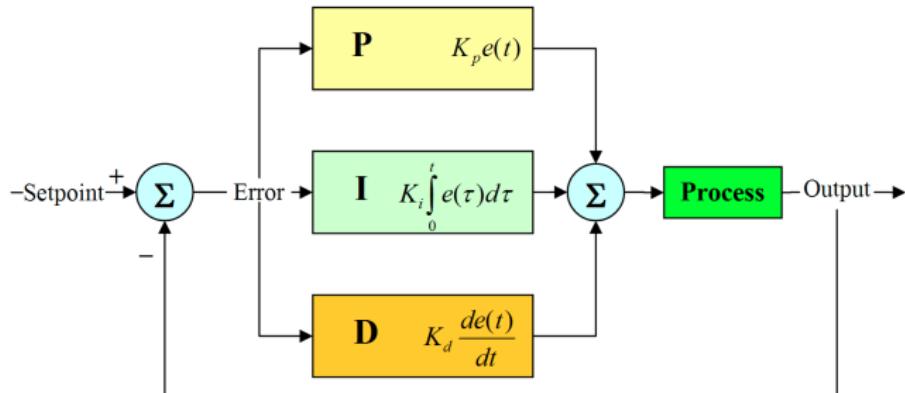


# BUILD AN INCREMENTAL ENCODER

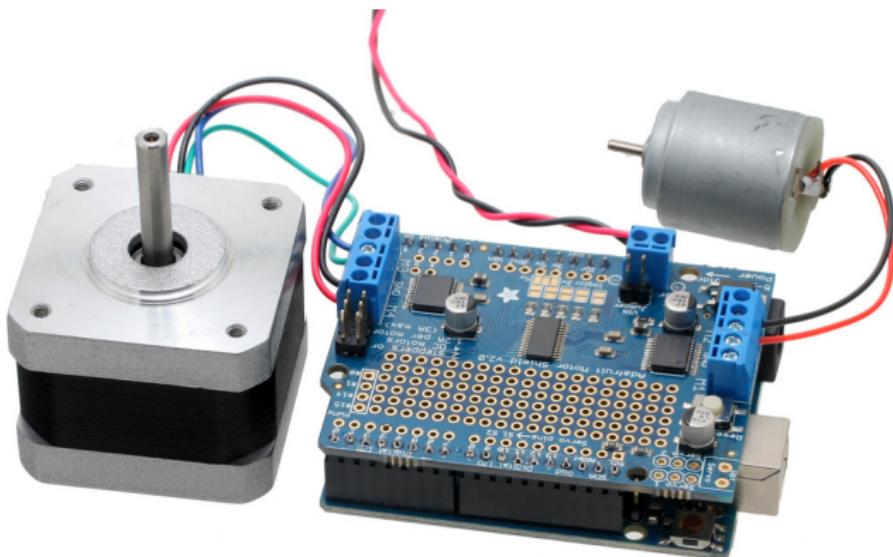
- Build a simple encoder from first principles
- Use a LED and phototransistor
- Measure rotational speed of a small DC motor
- Integrate it with the Arduino



# WRITE A PID CONTROLLER FOR YOUR DC MOTOR

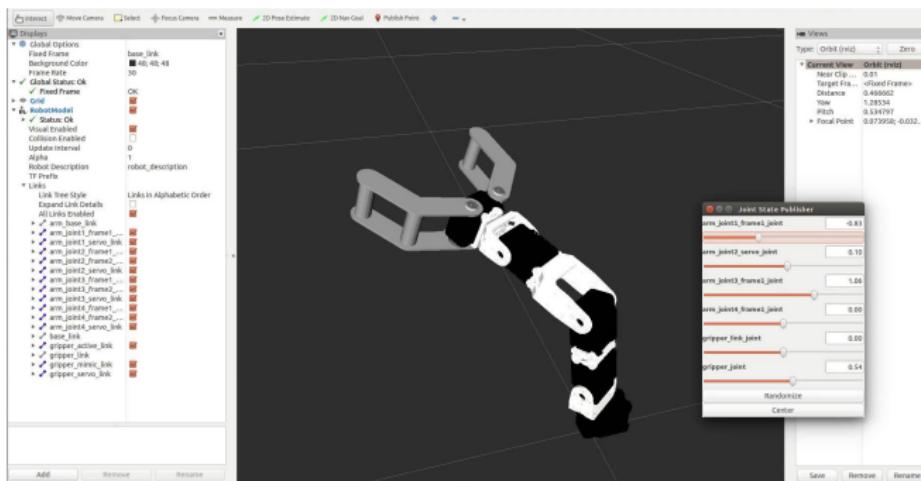


# CONTROL A STEPPER MOTOR WITH AN ARDUINO



- Control a stepper motor from the Arduino
- Implement stepper modes from first principles

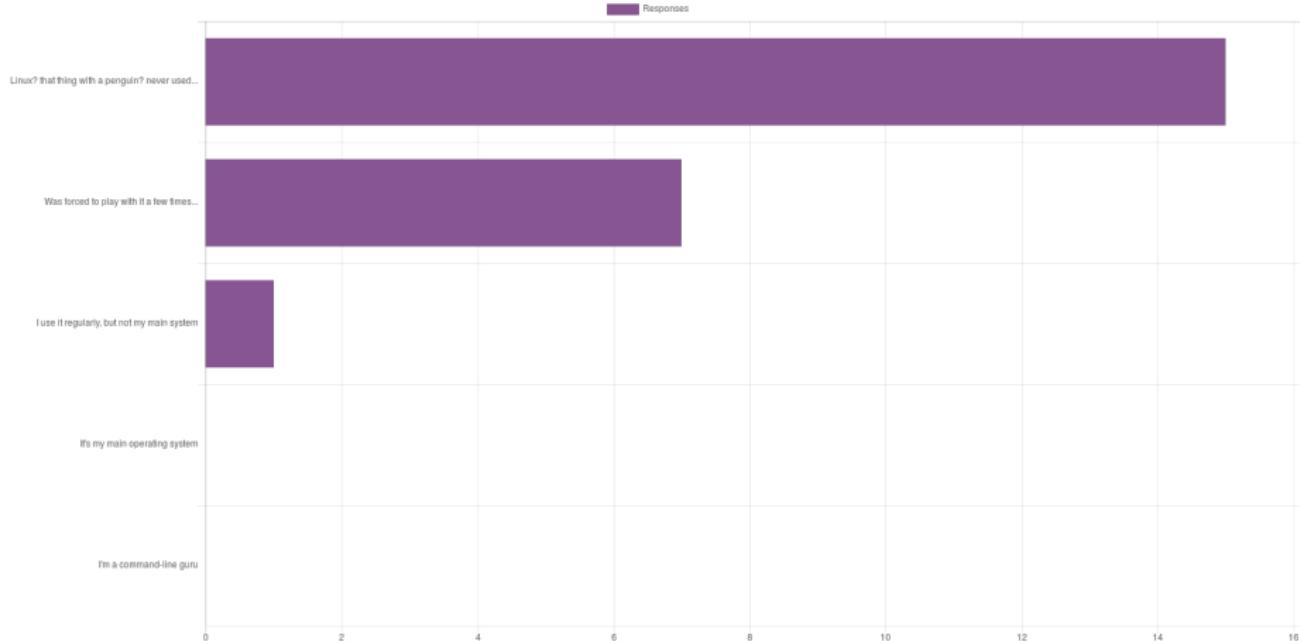
# MINI PROJECT: A 3DOF ROBOT ARM CONTROLLED FROM ROS



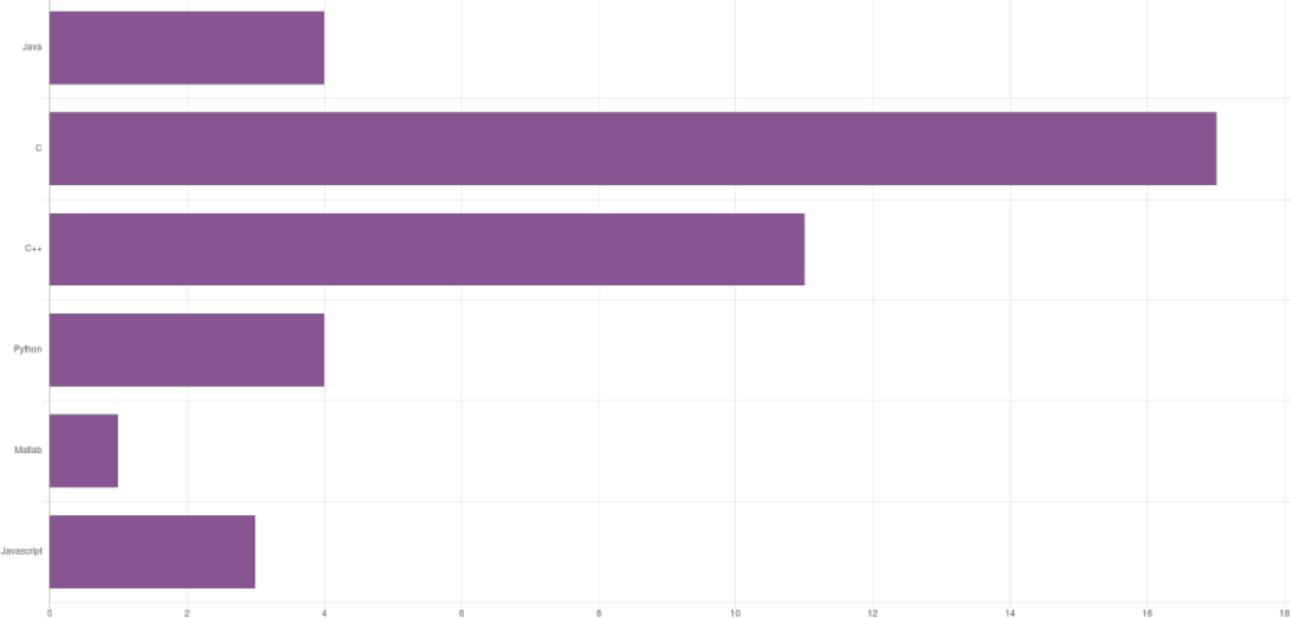
3 weeks to:

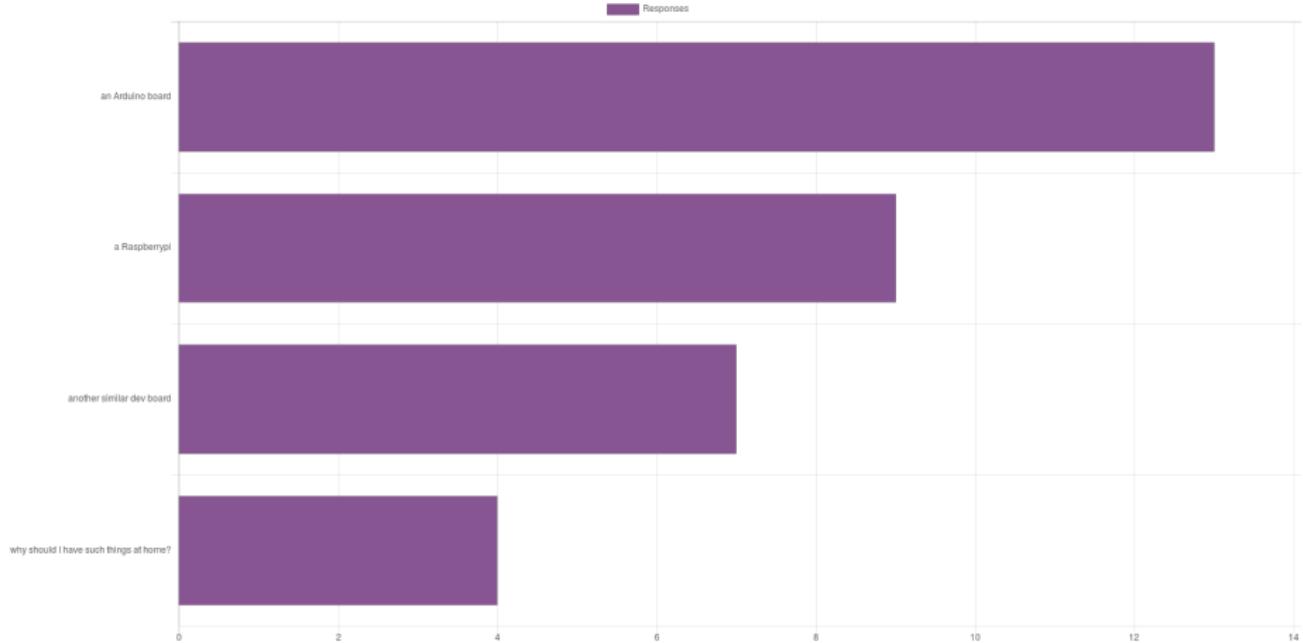
- Design and build a 3DoF (one stepper, 2 servos) robot arm
- Create a 3D model and visualise it
- Control it with ROS

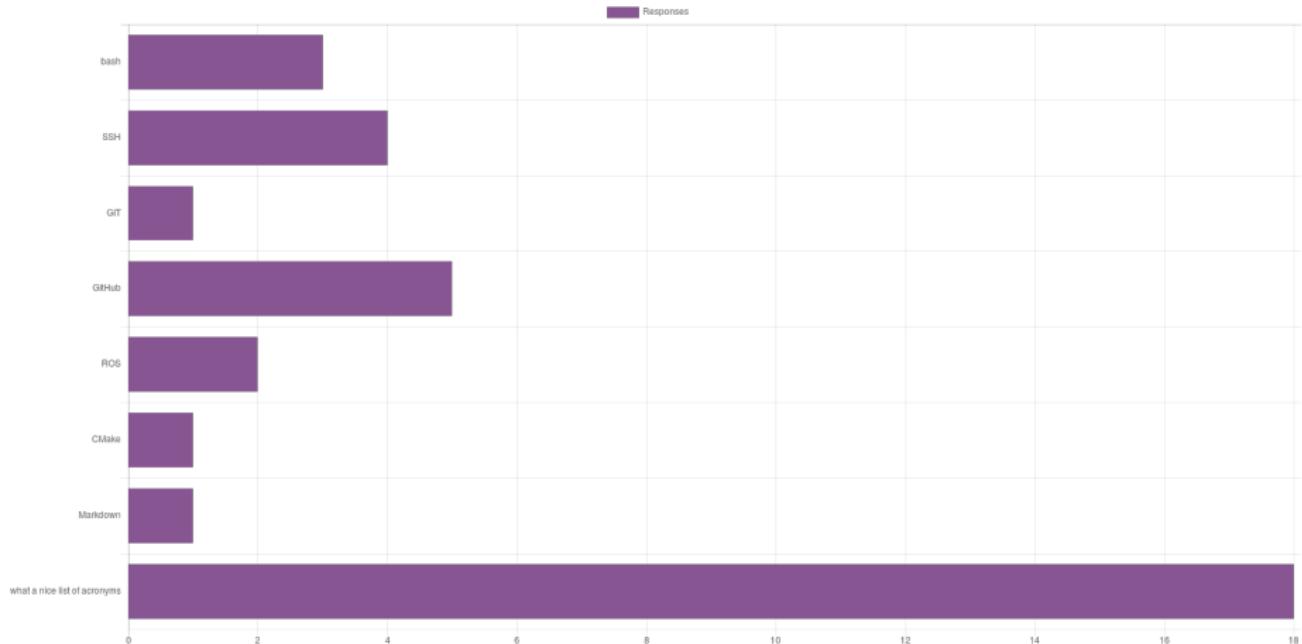
"HOMEWORK"



 Responses







# "HOMEWORK"

- Thinker with hardware and software!

# "HOMEWORK"

- Thinker with hardware and software!
- **Write code**, a lot of code – preferably Python and C++  
I am happy to answer code-related questions

# "HOMEWORK"

- Thinker with hardware and software!
- **Write code**, a lot of code – preferably Python and C++  
I am happy to answer code-related questions
- Invest £30 and get a Arduino/RaspberryPI starter kit (like the **Freenove ones**)

# "HOMEWORK"

- Thinker with hardware and software!
- **Write code**, a lot of code – preferably Python and C++  
I am happy to answer code-related questions
- Invest £30 and get a Arduino/RaspberryPI starter kit (like the **Freenove ones**)
- **Make it fun**: come up with **small, rewarding projects** –  
Tetris anyone?

# "HOMEWORK"

- Thinker with hardware and software!
- **Write code**, a lot of code – preferably Python and C++  
I am happy to answer code-related questions
- Invest £30 and get a Arduino/RaspberryPI starter kit (like the **Freenove ones**)
- **Make it fun**: come up with **small, rewarding projects** –  
Tetris anyone?
- **Install Linux** (and use it!) – alongside Windows as a  
dualboot, or inside a VM

# "HOMEWORK"

- Thinker with hardware and software!
- **Write code**, a lot of code – preferably Python and C++  
I am happy to answer code-related questions
- Invest £30 and get a Arduino/RaspberryPI starter kit (like the **Freenove ones**)
- **Make it fun**: come up with **small, rewarding projects** – Tetris anyone?
- **Install Linux** (and use it!) – alongside Windows as a dualboot, or inside a VM
- Get involved in an **open-source project**

# ATTENTION: I AM AWAY ON THE 11TH OF OCTOBER

- On 11th October: no lecture
- On 12th October: practical with Jake

10 min break, and off we start!

# INTRO TO ARDUINO

# THE ARDUINO MICROCONTROLLER

- What is an Arduino board?
- Developing programs
- Inputs and outputs
- Running control loops
- Using PWM to control motors



# THE ARDUINO MICROCONTROLLER

- What is an Arduino board?
- Developing programs
- Inputs and outputs
- Running control loops
- Using PWM to control motors
  - Arduino is an open-source electronics platform based on easy to use hardware and software
  - it is intended for anyone making interactive projects
  - very relevant to robotics, including professional-grade projects!



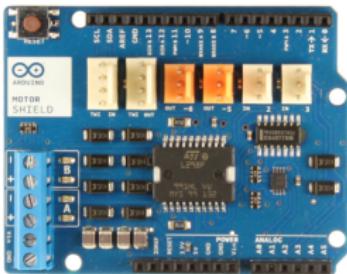
# THE ARDUINO MICROCONTROLLER ECOSYSTEM



Controller



Wifi shield



Motor shield



Ultrasonic  
sensor



Robot car

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments"

| Name      | Processor   | Operating Voltage/Input Voltage | CPU Speed | Analog In/Out | Digital IO/PWM | EEPROM [KB] | SRAM [KB] | Flash [KB] | USB     | UART |
|-----------|-------------|---------------------------------|-----------|---------------|----------------|-------------|-----------|------------|---------|------|
| Uno       | ATmega328   | 5 V/7-12 V                      | 16 Mhz    | 6/0           | 14/6           | 1           | 2         | 32         | Regular | 1    |
| Due       | AT91SAM3X8E | 3.3 V/7-12 V                    | 84 Mhz    | 12/2          | 54/12          | -           | 96        | 512        | 2 Micro | 4    |
| Leonardo  | ATmega32u4  | 5 V/7-12 V                      | 16 Mhz    | 12/0          | 20/7           | 1           | 2.5       | 32         | Micro   | 1    |
| Mega 2560 | ATmega2560  | 5 V/7-12 V                      | 16 Mhz    | 16/0          | 54/15          | 4           | 8         | 256        | Regular | 4    |
| Mega ADK  | ATmega2560  | 5 V/7-12 V                      | 16 Mhz    | 16/0          | 54/15          | 4           | 8         | 256        | Regular | 4    |

# ARDUINO UNO

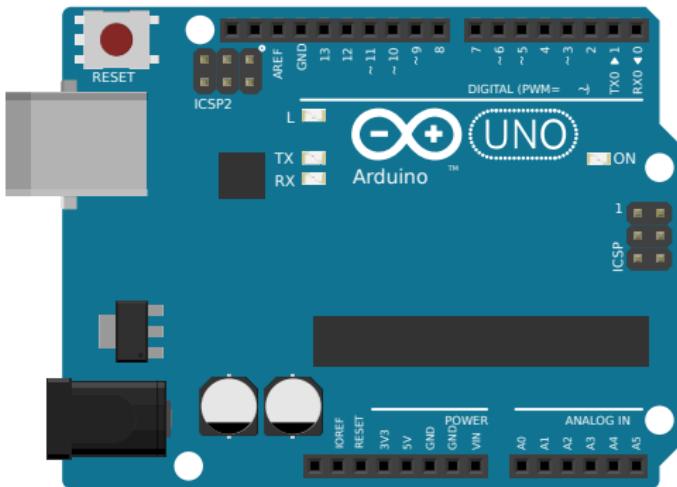
- Relatively robust board
- Microcontroller board based on the ATmega328
- 14 digital Input/Output pins
- 6 outputs can be used as PWM outputs
- 6 analog inputs
- 16MHz ceramic resonator
- USB connection
- power jack
- ICSP header
- reset button



# ARDUINO UNO

## Digital I/O

USB input  
9V external power supply



Power outputs

Analog inputs

# ARDUINO UNO

Digital I/O

USB input

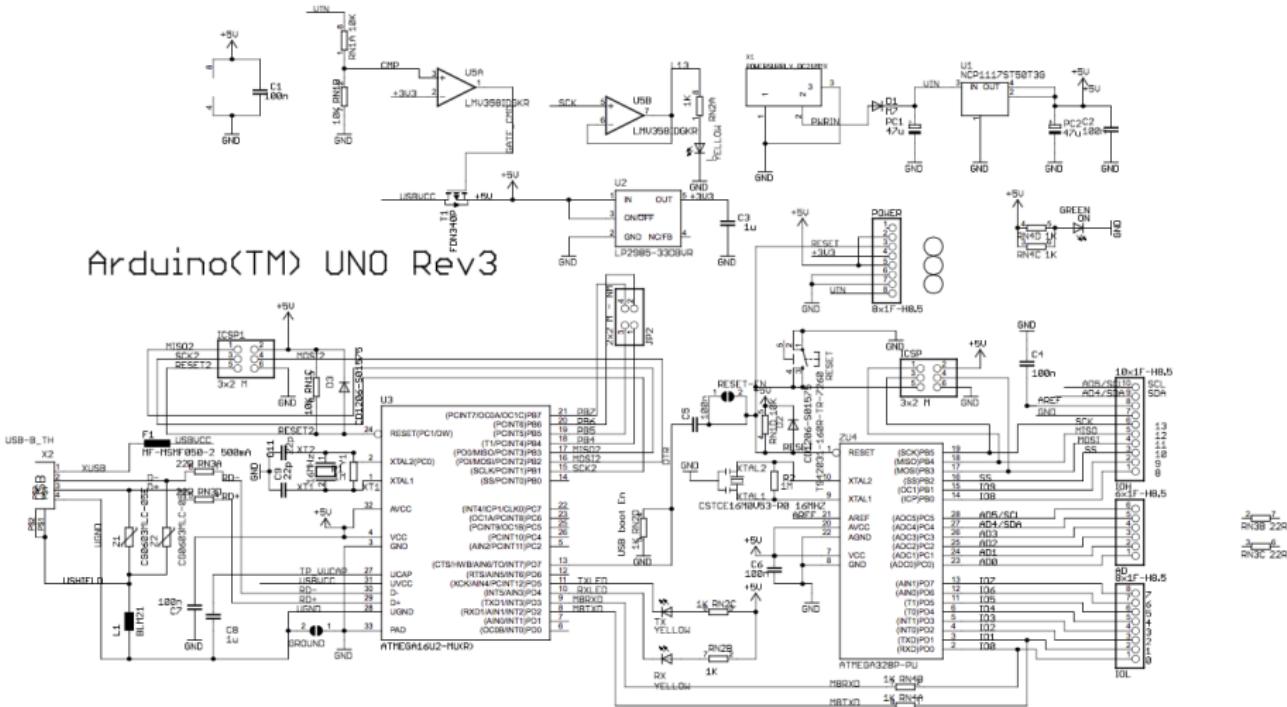


9V external  
power supply

Power outputs

Analog inputs

## Arduino(TM) UNO Rev3

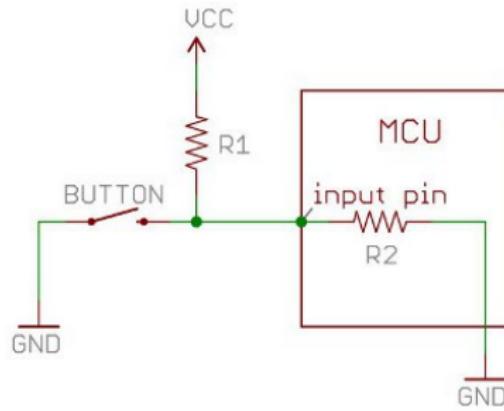


# ARDUINO UNO SUMMARY

- Microcontroller: **ATMega328**
- Operating voltage: **5V**
- External input voltage (recommended): 7-12V
- External input voltage (limits): **6-20V**
- Digital I/O pins: 14, of which 6 provide PWM output
- Analog input pins: 6
- DC current per I/O pin: 40 mA
- DC current for 3.3V pin: 50mA
- pins have internal pull-up resistors (off by default) of 20-50 k $\Omega$
- RAM: **2KB**
- Flash memory: **32KB**, of which 0.5KB used by boot loader
- EEPROM: 1KB
- Clock speed: **16MHz**

# WHAT IS A PULL-UP RESISTOR?

Prevent *floating* values on the input pins.



The ATmega328 microcontroller has internal pull-ups that can be enabled and disabled:

```
pinMode(5, INPUT_PULLUP); // Enable internal pull-up resistor on pin 5
```

## POWERING ARDUINO UNO

- via the USB connection
- external power supply
- the power source is selected automatically
- external (non-USB) 2.1mm centre-positive plug into the board's power jack
- leads from a battery can be inserted in the `Gnd` and `Vin` pin headers of the POWER connector
- recommended range is 7 to 12 volts

# POWER COMPARISON

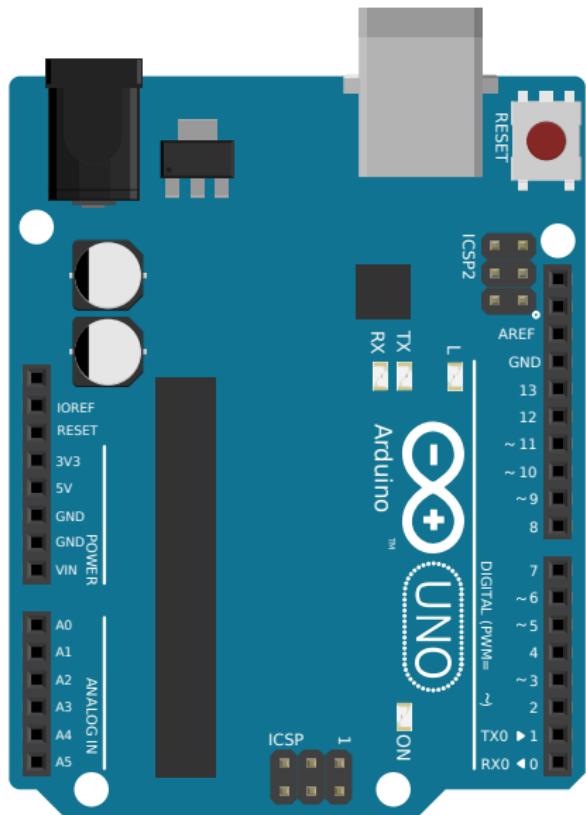
---

| Board            | Power Consumption              |
|------------------|--------------------------------|
| RaspberryPI A+   | 80 mA (0.4W)                   |
| RaspberryPI A+   | 160 mA (0.8W) with Wifi dongle |
| RaspberryPI B+   | 180 mA (0.9W)                  |
| RaspberryPI2 B   | 200 mA (1.0W)                  |
| RaspberryPI Zero | 80 mA (0.4W)                   |
| RaspberryPI Zero | 120 mA (0.7W)                  |
| Arduino Uno      | 30-50mA, can go down to <1mA   |

---

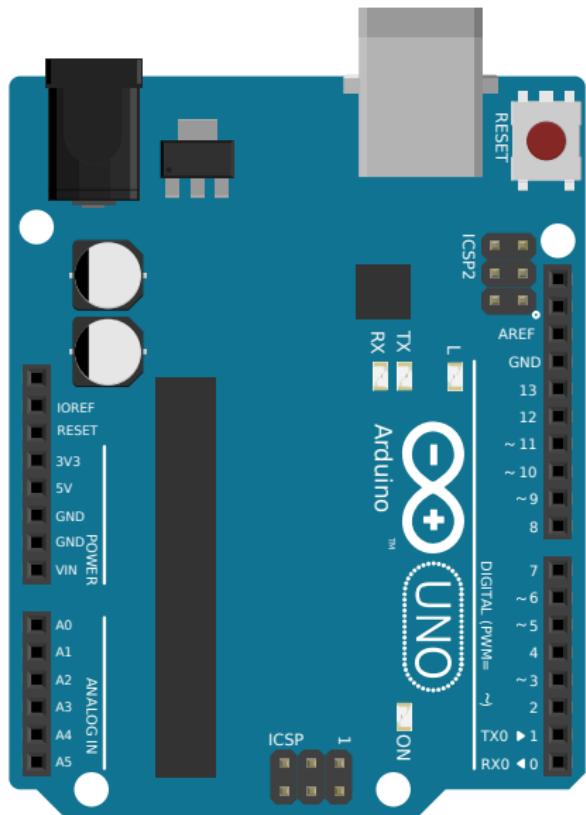
→ Arduino for a year on a coin cell

# SOME PINS HAVE SPECIALIZED FUNCTIONS



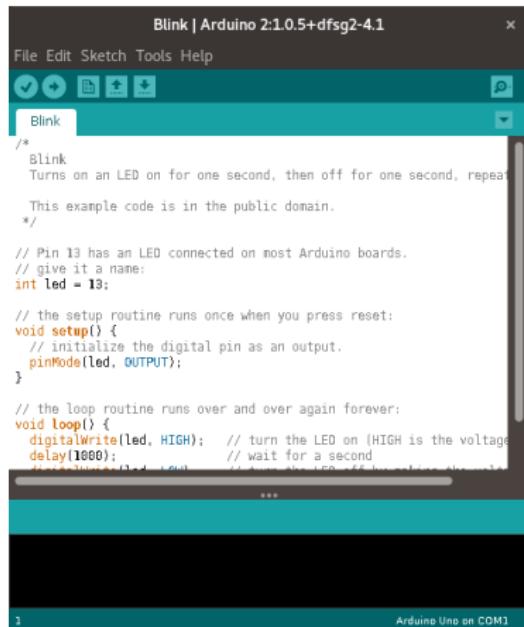
- Serial: 0 (RX) and 1 (TX)  
Used to receive (RX) and transmit (TX) TTL serial data
- external interrupts: 2 and 3  
Can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value
- PWM: 3, 5, 6, 9, 10 and 11  
Provide 8-bit PWM output with the `analogWrite()` function

# SOME PINS HAVE SPECIALIZED FUNCTIONS



- SPI: 10 (ss), 11 (MOSI), 12 (MISO), 13 (SCK)  
Support SPI communication using the SPI library
- LED: 13. Built-in LED connected to digital pin 13
- The Uno has 6 analog inputs, labeled A0 through A5
- I<sup>2</sup>C: A4 or SDA pin and A5 or SCL pin  
Support I<sup>2</sup>C communication using the Wire library

# HOW TO START CODING: SOFTWARE SETUP



- o install the IDE (download from [www.arduino.cc](http://www.arduino.cc) or `apt install arduino`)
- o open the Arduino IDE
- o use a USB A to B cable to plug the board into your computer
- o setup the board type and serial port from the *Tools* menu

# PROGRAMMING THE ARDUINO

- Arduino's programming language is a version of C++ with a bit of magic (pre-processing) to make it look simpler
- Arduino calls a program a **sketch**. Sketches are saved by default in your **sketchbook** (cf *File > Preferences*)
- Most C/C++ commands work in the sketch editor
- 4 basic components of an Arduino program:
  - initialization
  - setup
  - loop
  - user defined functions
- Arduino Uno is designed to allow it to be remotely reset by software

# PROGRAMMING THE ARDUINO: INITIALIZATION

- Include any extra libraries you are using
- All variables need to be initialized
- `a=4;` will not work, `int a=4;` will
- Declare variables you will assign later: `int a, b, c;`
- Remember datatypes: `bool, int, double, float, char, String...`
- Full reference: <https://www.arduino.cc/en/Reference>

# PROGRAMMING THE ARDUINO: SETUP

- Initialization for the Arduino board
- Usually initiate Serial communication
- Usually assign digital pins to input or output
- For example:

```
void setup() // 'void' means setup() does not return anything
{
    pinMode(13, OUTPUT); // pinMode sets the state of a digital pin,
    pinMode(12, INPUT); // either to OUTPUT or to INPUT

    Serial.begin(9600); // initialises a serial console at 9600 bauds
}
```

# PROGRAMMING THE ARDUINO: MAIN LOOP

- The loop is the main portion of your program
- The microcontroller runs it forever
- For instance:

```
void loop()
{
    digitalWrite(13, HIGH); // set the LED on
    delay(1000);           // wait for one second
    digitalWrite(13, LOW);  // set the LED off
    delay(1000);           // wait for one second
}
```

# PROGRAMMING THE ARDUINO: USER DEFINED FUNCTIONS

- These functions can be called in the loop and are usually responses to conditions that are met in the loop
- These functions must be declared with a *name*, a *return type* and optional *input parameters* with their types

The following function maps a user input *x* from one range to a new range:

```
float map(float x,
          float in_min, float in_max,
          float out_min, float out_max)
{
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

# PROGRAMMING THE ARDUINO: USER DEFINED FUNCTIONS

```
float map(float x,
          float in_min, float in_max,
          float out_min, float out_max)
{
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

The function is defined over *single precision floating numbers*, represented by the **float** C type.

The **return** statement defines what the function outputs (here, a floating number)

The curly brackets {} define what is inside the function (the function **body**). Curly brackets are generally required to separate blocks of code.

The indentation of the code is up to you and does not matter (contrary to other languages like Python!)

# PROGRAMMING THE ARDUINO: DIGITAL I/O

Each of the 14 digital pins on the Uno can be used as an input or output.

- `pinMode(<pin>, <mode>)`: set a digital pin as INPUT or OUTPUT
- `digitalWrite(<pin>, <state>)`: writes to a digital pin HIGH (5V) or LOW (GND)
- `digitalRead(<pin>)`: reads digital pin and returns HIGH or LOW

# PROGRAMMING THE ARDUINO: ANALOG I/O

- `analogReference(<type>)`  
    DEFAULT: 5V  
    EXTERNAL: voltage applied to Vref  
    **<http://arduino.cc/en/Reference/AnalogReference>**
- `analogRead(<pin>)`: reads voltage on given pin, 10 bits resolution (1024 values)

# PROGRAMMING THE ARDUINO: SERIAL COMMUNICATIONS

- The ATMega328 provides UART TTL serial communication
- Available on the digital pins 0 (**RX**) and 1 (**TX**)
- RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and the the USB connection to the computer
  
- The Arduino IDE includes a serial monitor
- It prints out whatever data is received on the serial port
- You can also use it to send data using the keyboard

# PROGRAMMING THE ARDUINO: SERIAL LIBRARY

- o `Serial.begin(<baudrate>)`: opens serial port at baudrate
- o `Serial.available()`: returns the nb of bytes available to read
- o `Serial.read()`: returns the first byte of incoming serial data available (or -1 if no data is available)
- o `Serial.print(<data>, <format>)`: prints data to the serial port as readable ASCII text
- o `Serial.println(<data>, <format>)`: prints data to the serial port as readable ASCII text and adds a line ending

**<http://arduino.cc/en/Reference/Serial>**

# PROGRAMMING THE ARDUINO: CONTROL FLOW

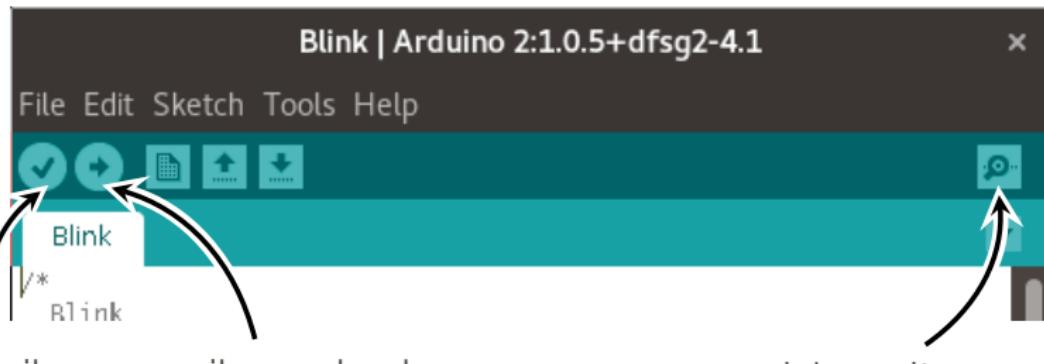
```
if(<variable> <comparison operator> <condition>)
{
    // do something
}
else
{
    // do something else
}
```

→ the MCU checks the condition and responds accordingly.

```
for(<initialization>; <condition>; <increment>)
{
    // do something
}
```

→ the MCU repeats something for a certain number of time. For example, read 500 chars on the serial port.

# COMPILING AND UPLOADING CODE



compile      compile + upload

serial monitor

Under tools, make sure Board matches the actual board you are using and Serial port is the correct one.

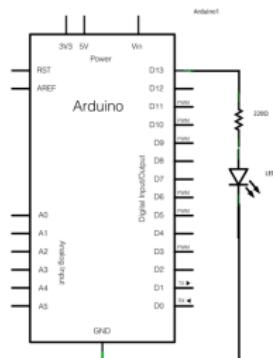
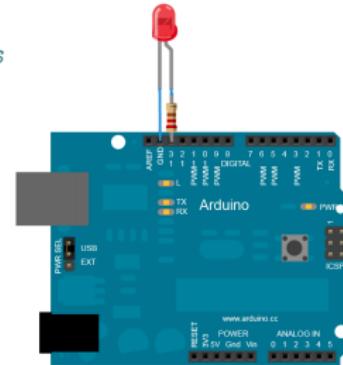
# PROGRAM EXAMPLES

# LED BLINKING

```
// Pin 13 has an LED connected on most Arduino boards
// give it a name:
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
  delay(1000);                // wait for a second
  digitalWrite(led, LOW);       // turn the LED off
  delay(1000);                // wait for a second
}
```



# ENCODER

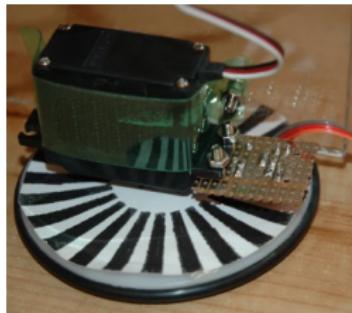
```
#include <Encoder.h>

// Change these two numbers to the pins connected to your encoder.
Encoder myEnc(5, 6);
// avoid using pins with LEDs attached

void setup() {
    Serial.begin(9600);
    Serial.println("Basic Encoder Test:");
}

long oldPosition = -999;

void loop() {
    long newPosition = myEnc.read();
    if (newPosition != oldPosition) {
        oldPosition = newPosition;
        Serial.println(newPosition);
    }
}
```



# MEASURING POSITION: ENCODERS

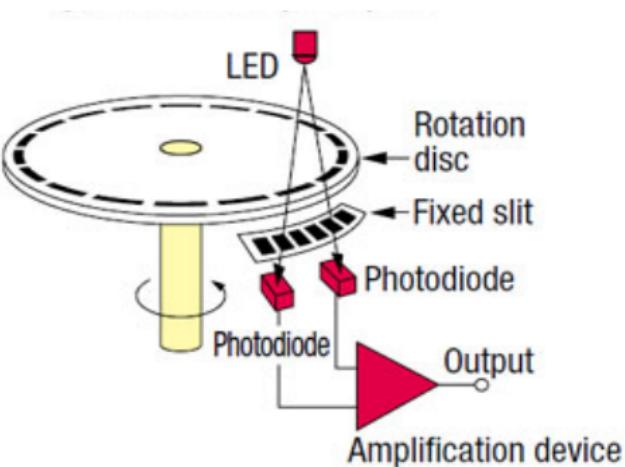
# DIGITAL INCREMENTAL ENCODER



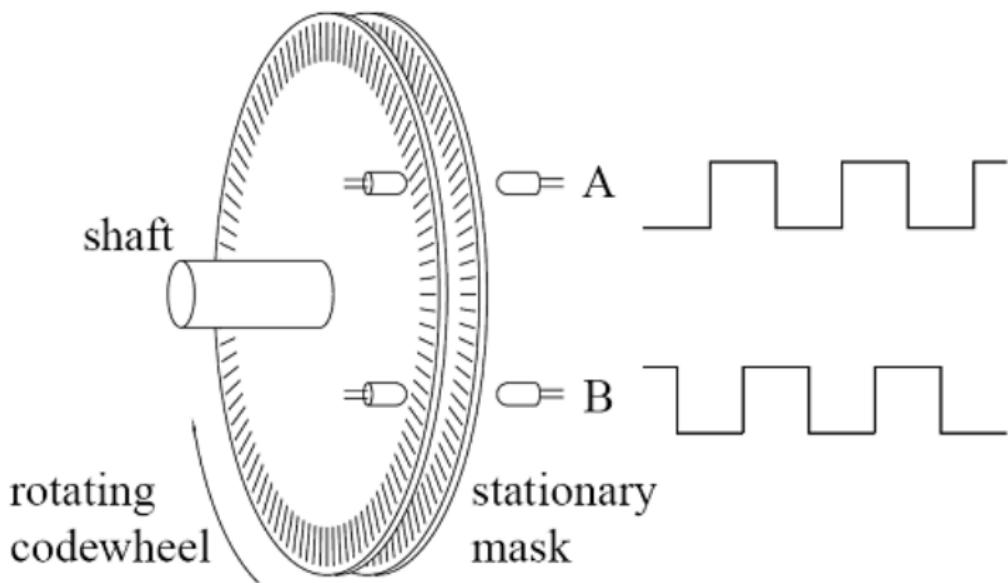
- Generates a relative position signal suitable for positioning tasks
- Rotation direction recognition
- Speed information from number of pulses per time unit
- Standard solution for many applications
- Single output types only used to determining the speed

# OPTICAL INCREMENTAL ENCODER

- Slotted discs interrupt the light beam
- Photo emitter-detector pair detect pulses of light
- Amplifier then generates output pulses as slot passes light path
- Pulses counted to provide a position estimate
- Secondary LED photo emitter-detector pair provide direction of rotation
- Quadrature encoder generates two square wave pulses 90 degrees apart

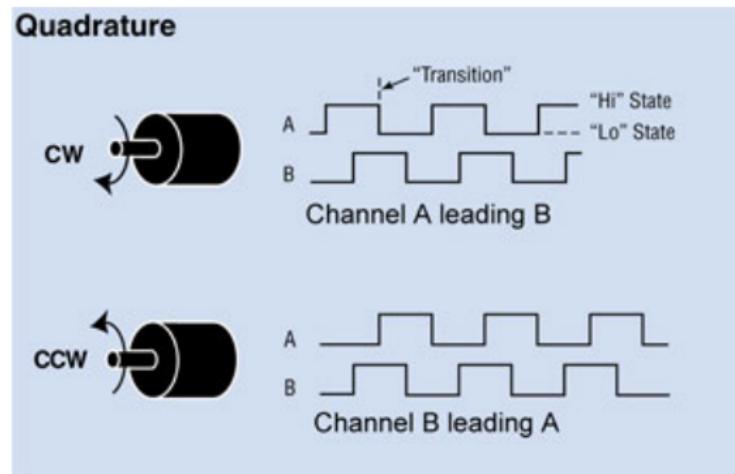


# QUADRATURE OUTPUT SIGNAL GENERATION



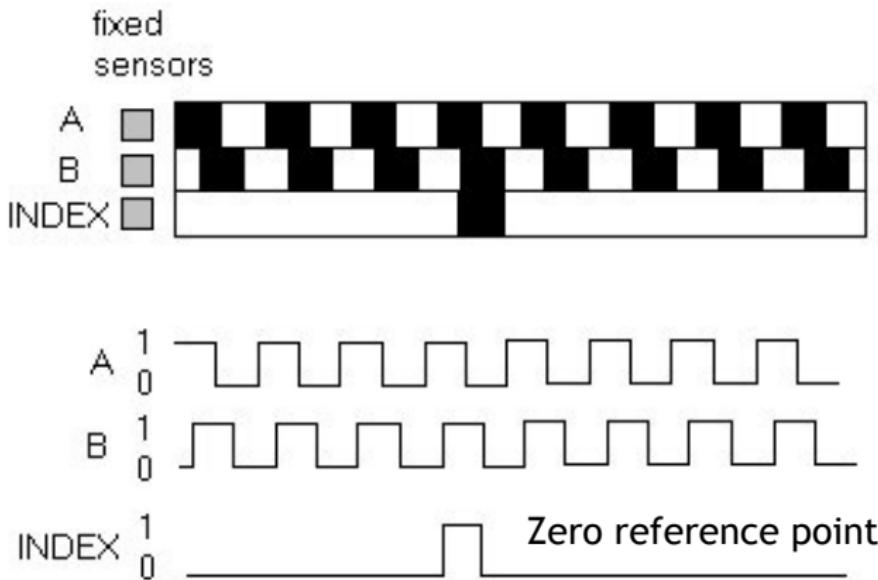
Output of two photo detectors are phase shifted by 90 degrees<sup>35</sup>

# QUADRATURE OUTPUT SIGNAL USAGE



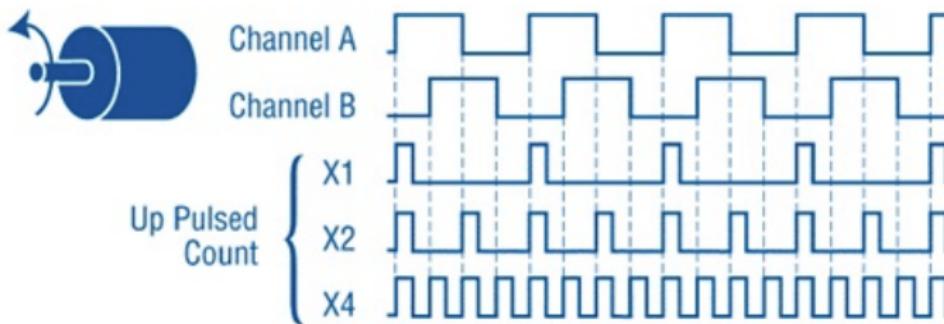
- A quadrature encoder generates two pulse streams that are  $90^\circ$  out of phase with one another
- It is possible to determine directionality by monitoring which channel leads in phase

# INCREMENTAL ENCODER INDEX SIGNAL



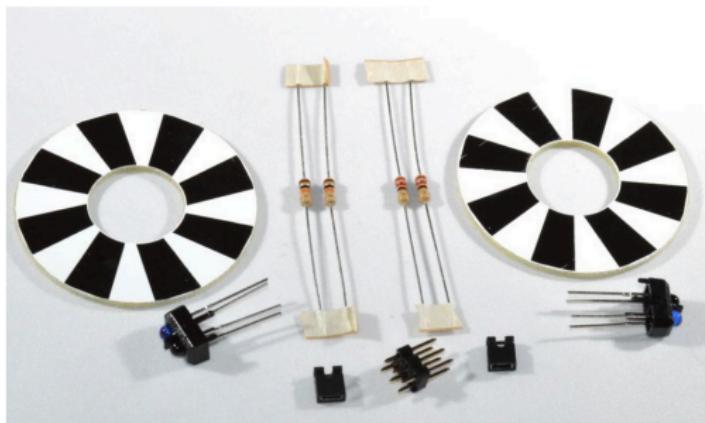
- Gives an absolute position signal at a single reference location
- Can be used to assist calibration

## USING ALL EDGES INCREASES RESOLUTION



- Triggering off of the leading and or trailing edges of the pulses can increase resolution by up to four times
- More transition to count if use all edges!
- But more sensitive to noise

## BUDGET WHEEL ENCODERS/SENSORS



- 2 wheel encoder discs for fitting inside the wheel drive
- 2 optical reflectance sensors
- Count 16 pulses per revolution
- very cheap - £5
- Use with Arduino systems

# TYPICAL DATASHEET FOR ENCODER

## CONNECTIONS

| FUNCTION  | CABLE  | PLUG  |
|-----------|--------|-------|
| 0 Volts   | Black  | Pin 1 |
| + Volts   | Red    | Pin 2 |
| A Channel | White  | Pin 3 |
| B Channel | Blue   | Pin 4 |
| Z Channel | Yellow | Pin 5 |
| A Channel | Green  | Pin 6 |
| B Channel | Violet | Pin 7 |
| Z Channel | Brown  | Pin 8 |

## ACCESSORIES AVAILABLE

-COUPLINGS

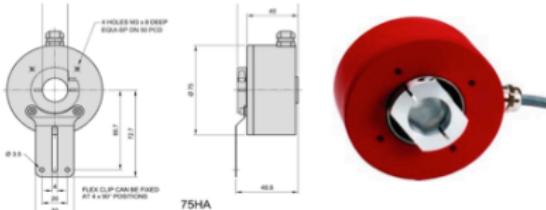
-BRACKETS & FLANGES

-MEASURING WHEELS

## SERIES



The 75HA Series is a precision, yet versatile hollow shaft encoder, machined from solid aluminium. Its choice of versatile spring mounting clip or front face mounting holes, makes the encoder compatible with many industrial formats.



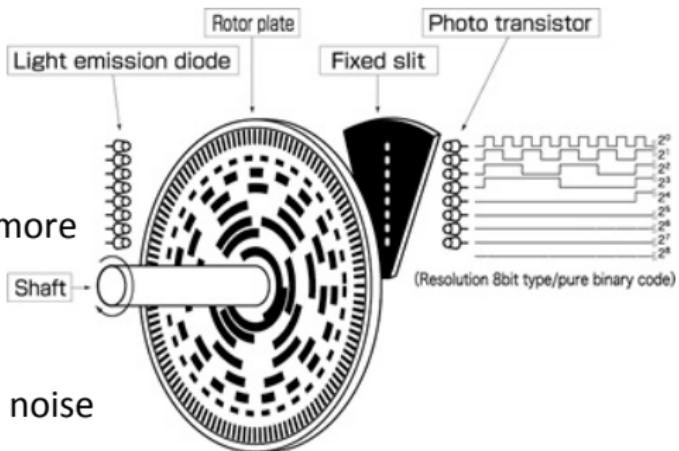
## SPECIFICATIONS

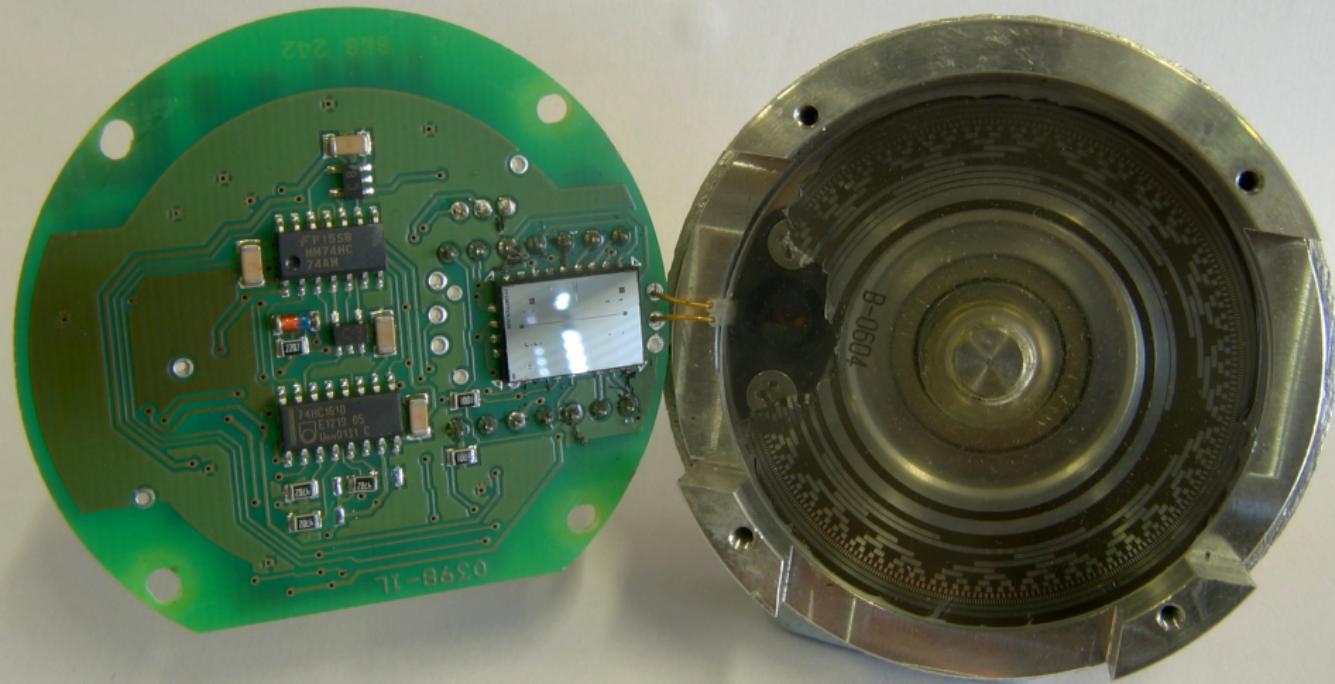
|                |   |                      |                                    |
|----------------|---|----------------------|------------------------------------|
| Body:          | Precision Machined Aluminium                                    | Inertia:             | 100gcm <sup>2</sup> (typical)      |
| Cover:         | Precision Machined Aluminium (Powder Coated)                    | Shaft Loading:       | Designed to support its own weight |
| Shaft:         | Stainless Steel   | Cable:               | Screened, oil and salt water proof |
| Bearings:      | Balbraces   | Impulse Frequency:   | Up to 300kHz (depending on model)  |
| Shaft Sealing: | Front: Sealed bearing<br>Rear: Single lip seal on rear of cover | Current Consumption: | 60mA (typical - without load)      |
|                |   | Torque:              | 2Nm (typical)                      |
|                |   | Operating Temp:      | -20 to +60°C (higher on request)   |
|                |   | Protection:          | IP65                               |
|                |   | Maximum Speed:       | 3000 RPM (Higher on request)       |

# ABSOLUTE OPTICAL ENCODERS

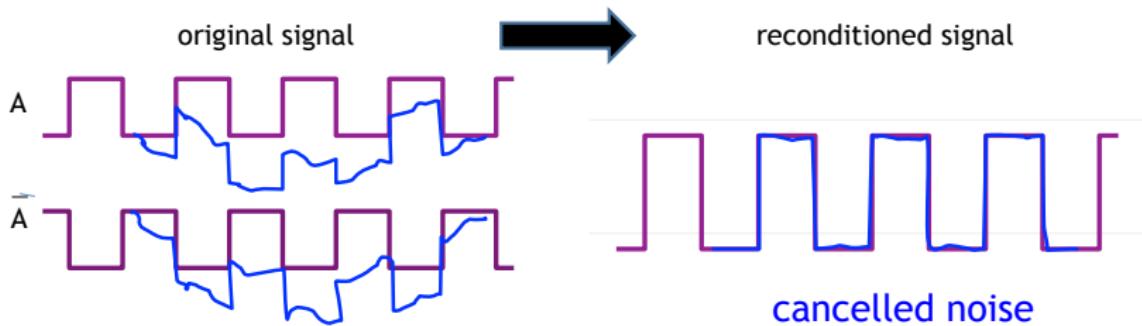
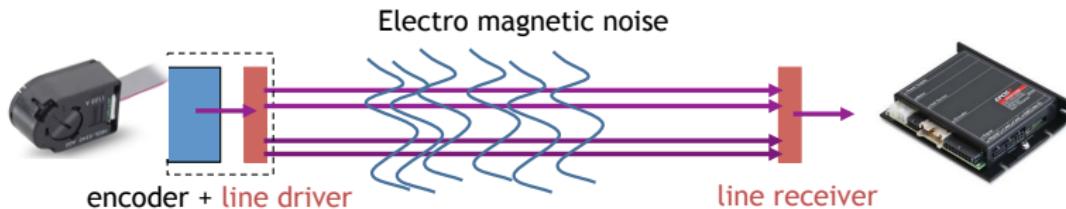
- Absolute encoders record the unique values of each shaft position
- Especially important in the event of a power failure
- Gray and binary codes most common

- Incremental rotary encoders more common
- Low cost
- Provides easy to use signals
- More susceptible to electrical noise





# ENCODER SIGNAL CONDITIONING



- Use differential output drivers
- Twisted pairs suppress interference better than single wires

# INTERFACE TO ENCODERS

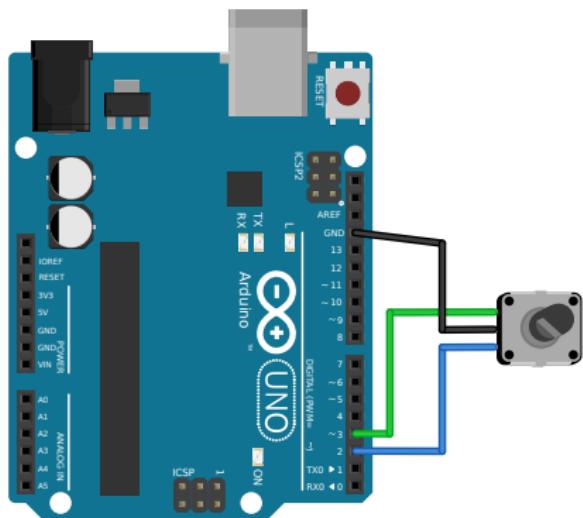
- E.g. Sensoray 626
- Interfacing encoder signals e.g. to a PC



| Table 24 Encoder Connectors |                 |                  |
|-----------------------------|-----------------|------------------|
| J5                          |                 |                  |
| 26 pin IDC ribbon connector |                 |                  |
| Pin                         | Function        | User Designation |
| 1                           | Encoder (0A) A- |                  |
| 3                           | GND             |                  |
| 5                           | Encoder (0A) B+ |                  |
| 7                           | Encoder (0A) I- |                  |
| 9                           | GND             |                  |
| 11                          | Encoder (1A) A+ |                  |
| 13                          | Encoder (1A) B- |                  |
| 15                          | GND             |                  |
| 17                          | Encoder (1A) I+ |                  |
| 19                          | Encoder (2A) A- |                  |
| 21                          | GND             |                  |
| 23                          | Encoder (2A) B+ |                  |
| 25                          | Encoder (2A) I- |                  |
| 2                           | Encoder (0A) A+ |                  |
| 4                           | Encoder (0A) B- |                  |
| 6                           | 5V              |                  |
| 8                           | Encoder (0A) I+ |                  |
| 10                          | Encoder (1A) A- |                  |
| 12                          | 5V              |                  |
| 14                          | Encoder (1A) B+ |                  |
| 16                          | Encoder (1A) I- |                  |
| 18                          | 5V              |                  |
| 20                          | Encoder (2A) A+ |                  |
| 22                          | Encoder (2A) B- |                  |
| 24                          | 5V              |                  |
| 26                          | Encoder (2A) I+ |                  |

# INTERFACE TO ARDUINO

- use interrupt inputs
- count the pulses



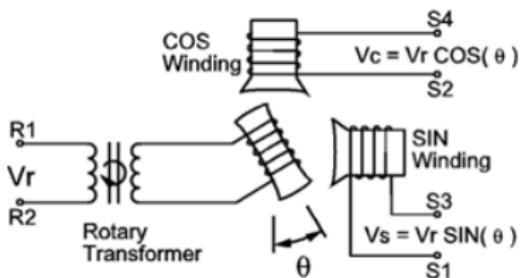
# RESOLVER POSITION MEASUREMENT



*Source: Maxon*

Electromechanical device with a mechanical design similar to a motor.

# RESOLVER POSITION MEASUREMENT

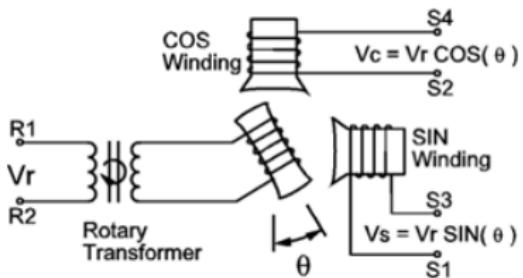


Source: Advanced Micro Controls, Inc.

Electromechanical device with a mechanical design similar to a motor.

- Analog rotor position signal
- Indicates absolute position within a single revolution
- Requires interfacing electronics
- Resolvers are “excited” by an AC reference sine wave

# RESOLVER POSITION MEASUREMENT

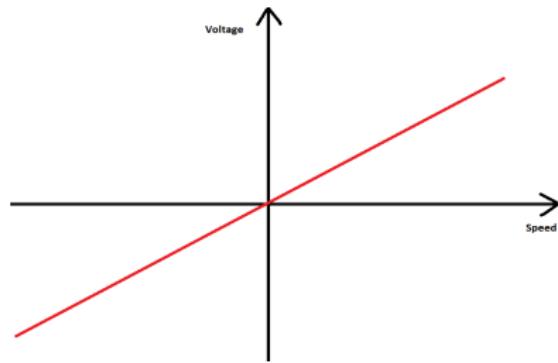


Source: Advanced Micro Controls, Inc.

Compared to an encoder:

- more rugged: no sensitive optics → withstand higher vibrations/shocks; resistant to electrical noise; withstand higher temperatures; higher durability
- encoders are cheaper; more accurate; smaller, lighter, less inertia

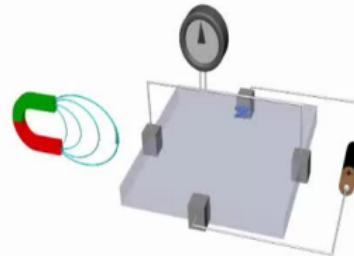
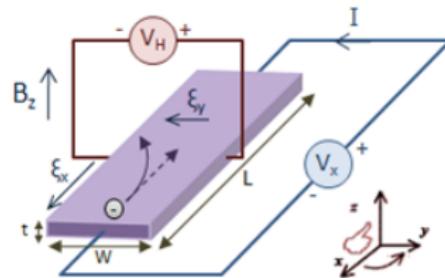
# TACHOMETER VELOCITY MEASUREMENT



- A tacho is a small generator
- A coil that revolves about a magnet.
- Much like the motor it too has a speed constant
- Measures velocity not position and therefore not an encoder
- Analog speed signal
- Rotation direction recognition
- Not suitable for positioning tasks

# HALL EFFECT MAGNETIC SENSOR

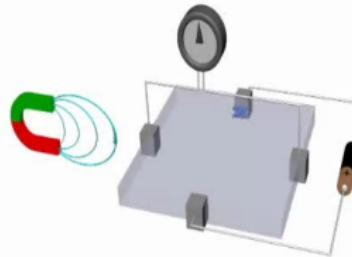
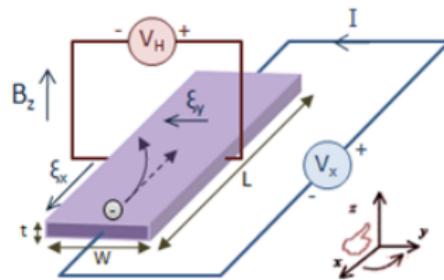
The **Hall effect** is the production of a potential difference across an electrical conductor when a magnetic field is applied perpendicular to the flow of current.



# HALL EFFECT MAGNETIC SENSOR

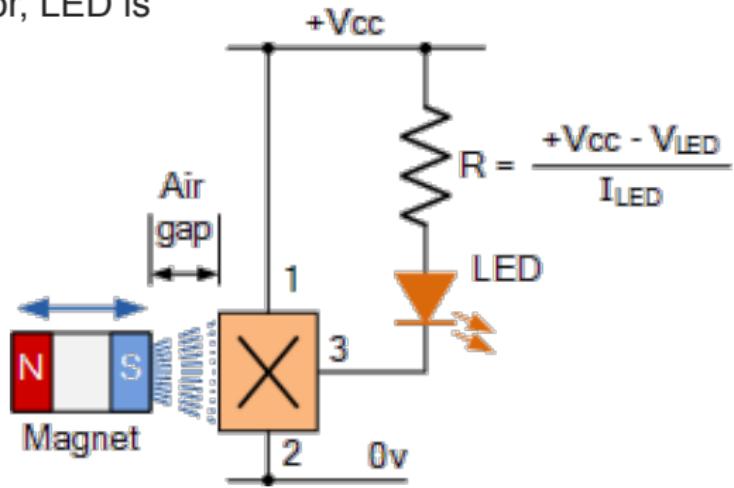
Can be taken advantage of in semiconductor devices to implement a switch activated by the application of a magnetic field.

→ Hall effect sensors can be used for proximity switching.

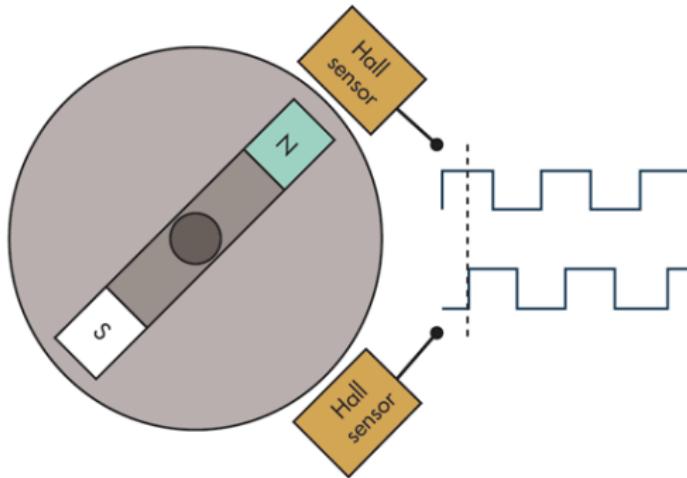


# HALL EFFECT MAGNETIC SENSOR CIRCUIT

- Simple switch activated by the application of a magnetic field
- When south pole of magnet brought up to this sensor, LED is switched on

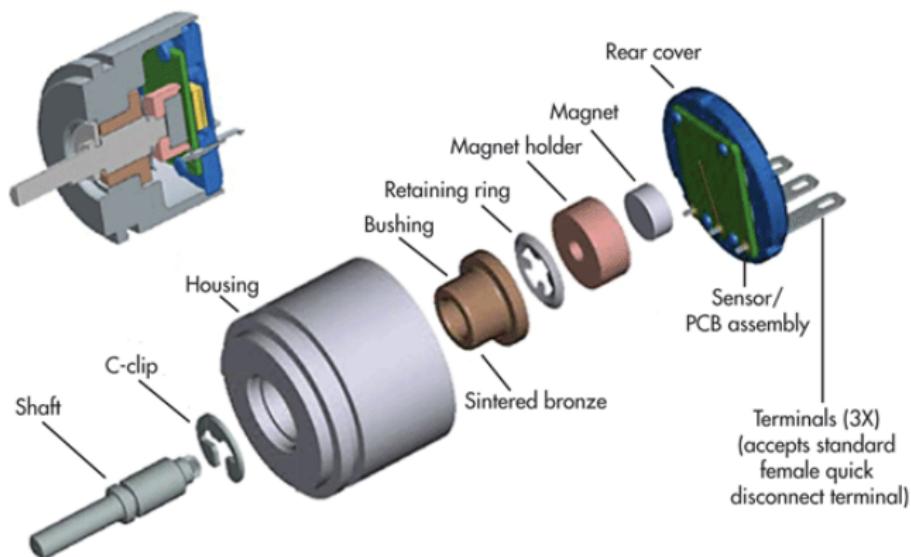


# HALL EFFECT MAGNETIC ROTARY ENCODER



- Magnetic rotary encoder that comprises two poles and two sensors
- The second sensor makes it possible to not only detect the direction of rotation but also to interpolate the absolute position of the shaft

# MAGNETIC ROTARY ENCODER



- Construction of a typical magnetic rotary encoder
- This type of encoder is often more robust than an optical encoder

That's all, folks!



**Questions:**

Portland Square B316 or **severin.lemaignan@plymouth.ac.uk**

**Slides:**

[github.com/severin-lemaignan/module-introduction-sensors-actuators](https://github.com/severin-lemaignan/module-introduction-sensors-actuators)  
...or the DLE!