ARDUINO CRASHCOURSE – ZERO TO HERO

WHO AM I?

- Jacob Bechmann Pedersen
  - Electronics Engineer (AU, 2019)
  - Started using Arduino in 2014
  - Volunteer in Coding Pirates 2016-2018
  - Teaching at MakerCamp since 2018
    - 12-16 y/o ”Inventors”
  - Taught Arduino for IDA, StudyNow, Herning Municipality, etc.
  - Full-time embedded electronics engineer at DTU
    - Robots and automated systems

AGENDA

- 16:00-17:00 : Arduino Basics – Blinky
- 17:00-18:00 : Wireless communications with Arduino
- 18:00-18:45 : Dinner - Pizza and softdrinks
- 18:45-19:45 : Motor control and building robot
- 19:45-20:00 : Summary, evaluation, and thanks for now!

PURPOSE OF THIS WORKSHOP

- Give you a QUICK understanding of the basics and essentials of Arduino:
  - Hands-on successes
  - Experience some difficulties
- Try the possibilities with this platform
- Inspiration to how to use Arduino for yourself

TODAY'S CONTENTS

Part 1: Arduino Basics

- Introduction to Arduino
  - Blinky Example
- Sensors and actuators
- Digital and analog electronics
  - Button example
  - Analog example

Part 2: Wireless Communication

- Types of wireless
- ESP-01
  - Wireless example

Part 3: Motor Control

- About DC Motors
- L293D Driver
  - Motor driving example

Part 4: Robot Building

- Walkthrough of robot
- Freestyle building

BEFORE WE GET STARTED

- The code and examples throughout this workshop are all available on:
  - https://github.com/iakop/ArduinoCrashcourse
  - It's a good idea to keep this site open

PART 1:
ARDUINO BASICS

INTRO TO ARDUINO

WHAT IS A MICROCONTROLLER?

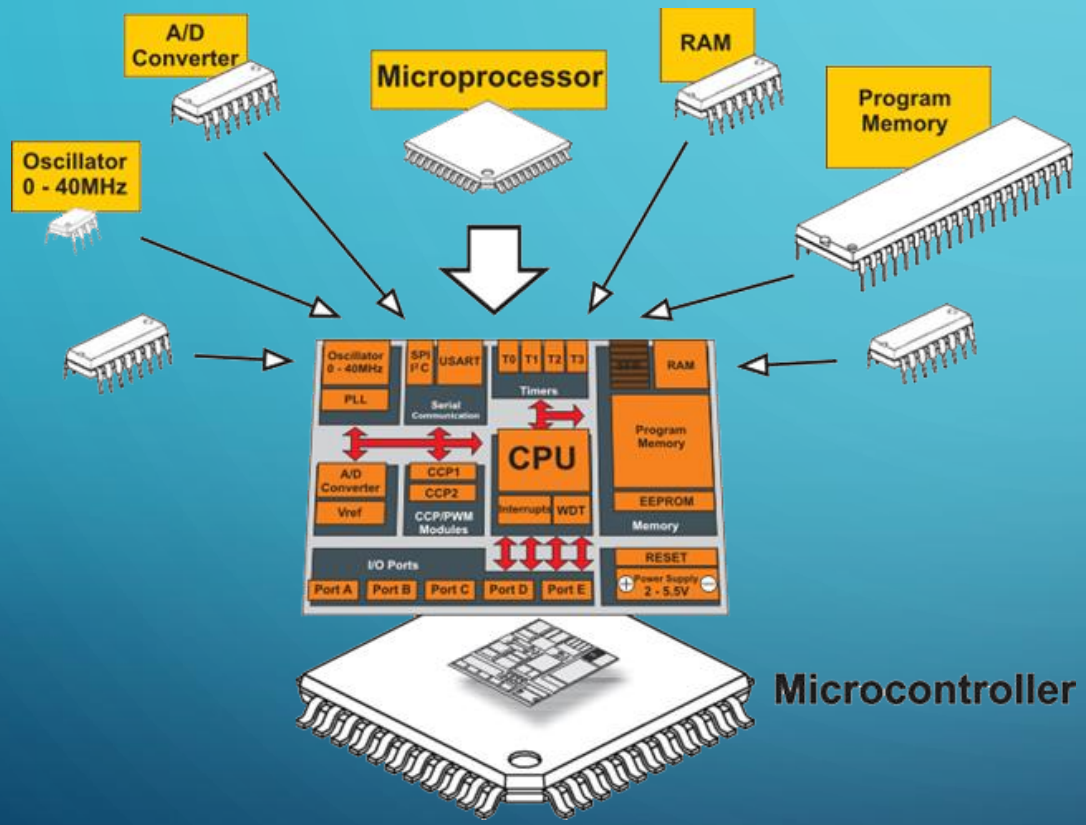
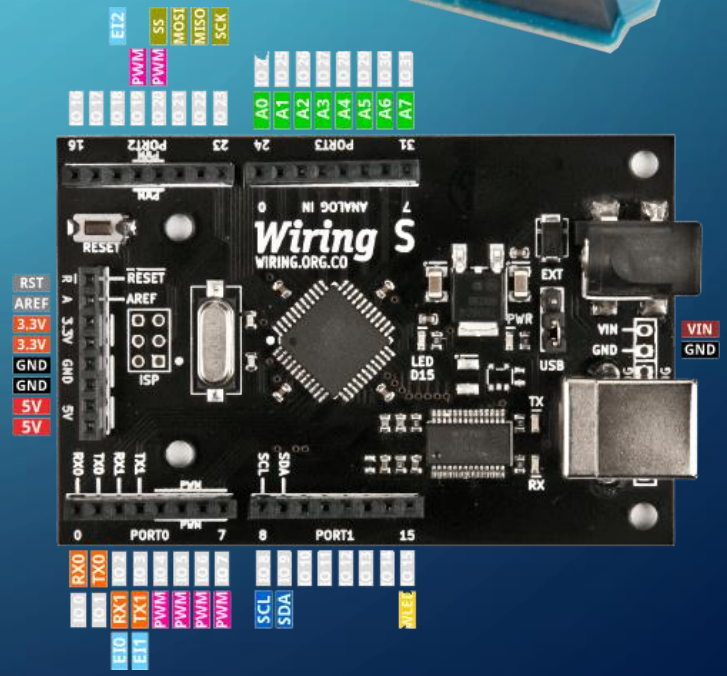


Figure from: Introduction to the World of MicroControllers – Mikroelektronika URL: https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c/introduction-to-the-world-of-microcontrollers

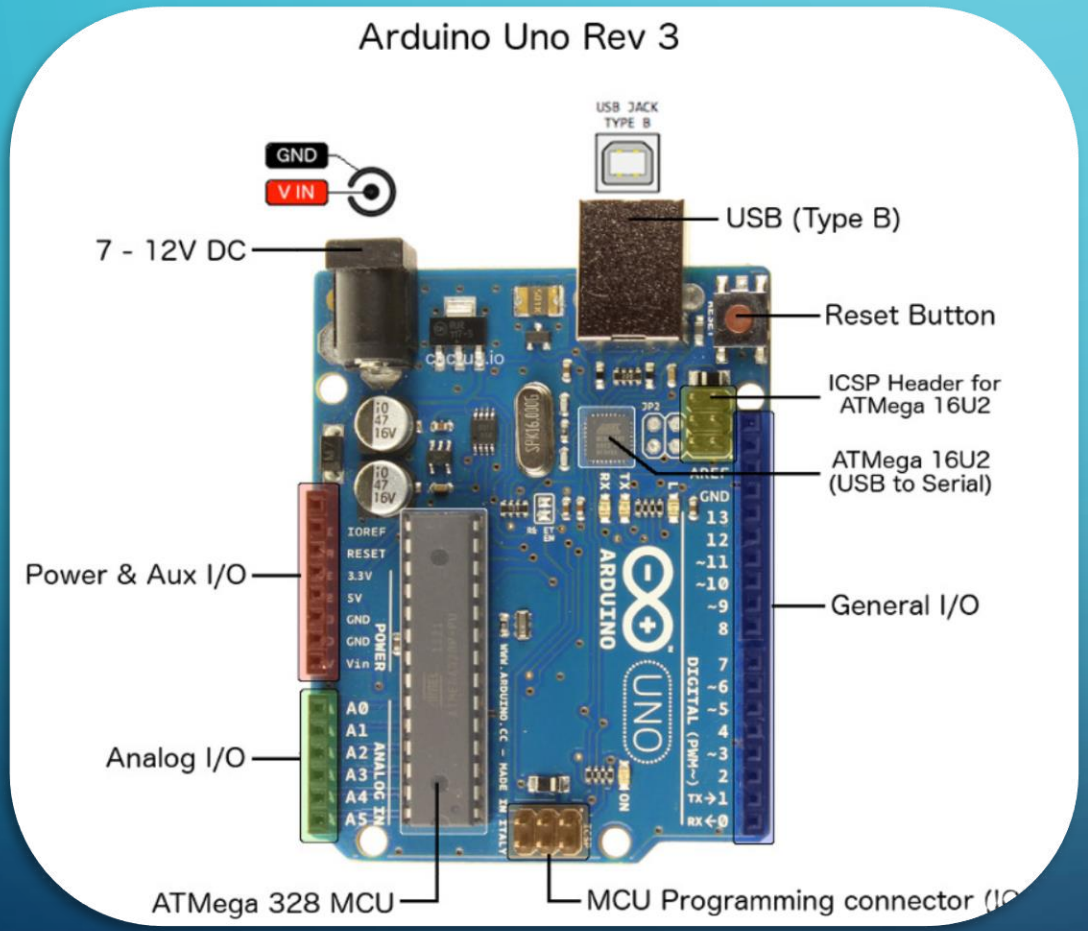
- MCU for short (MicroController Unit)
- They create our everyday magic
- Bind software and hardware together
- Not just processors:
  - Contain an entire system
  - Storage, memory, program memory, ADC mm.
- Easy to integrate in electronic circuits

THE ARDUINO PLATFORM

- Originally a fork of Wiring by Hernando Barragán
  - Master's Thesis in Interaction Design
  - Based on Processing (programming language)
  - Makes microcontroller development easier
- Most used platform for MCU prototyping
- Simple, intuitive programming
- Many code examples available
  - Many software libraries



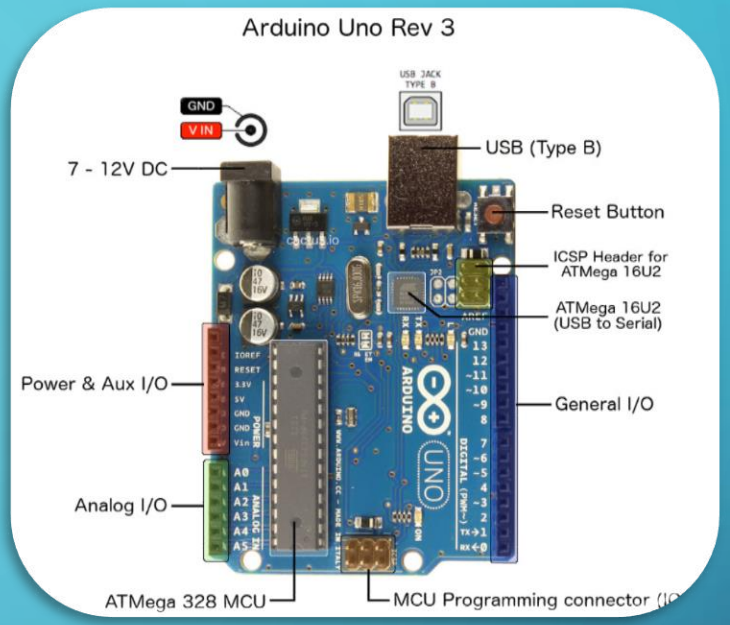
THE ARDUINO PLATFORM



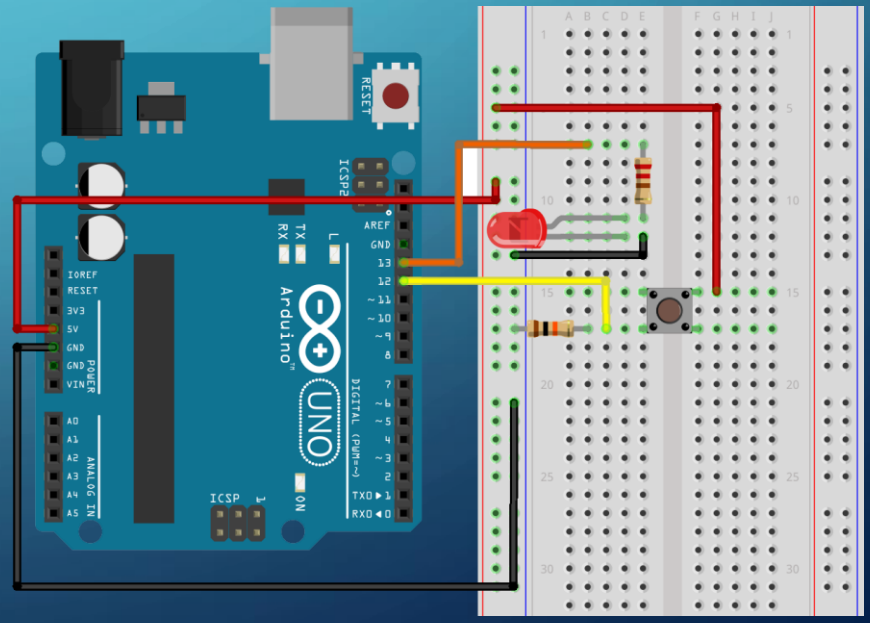
- The microcontroller (Atmega328) is the brain
- DC and USB B Port for powering and programming
- The pins:
  - RED Power supplies, reset etc.
  - GREEN Analog input/output
  - BLUE General Purpose input/output
- And some other neat features

THE ARDUINO PLATFORM

- We connect it to electronics through its pins
- Pin signals are controlled in programs (turn on/off, etc.)
- Mostly connects through a ”breadboard”



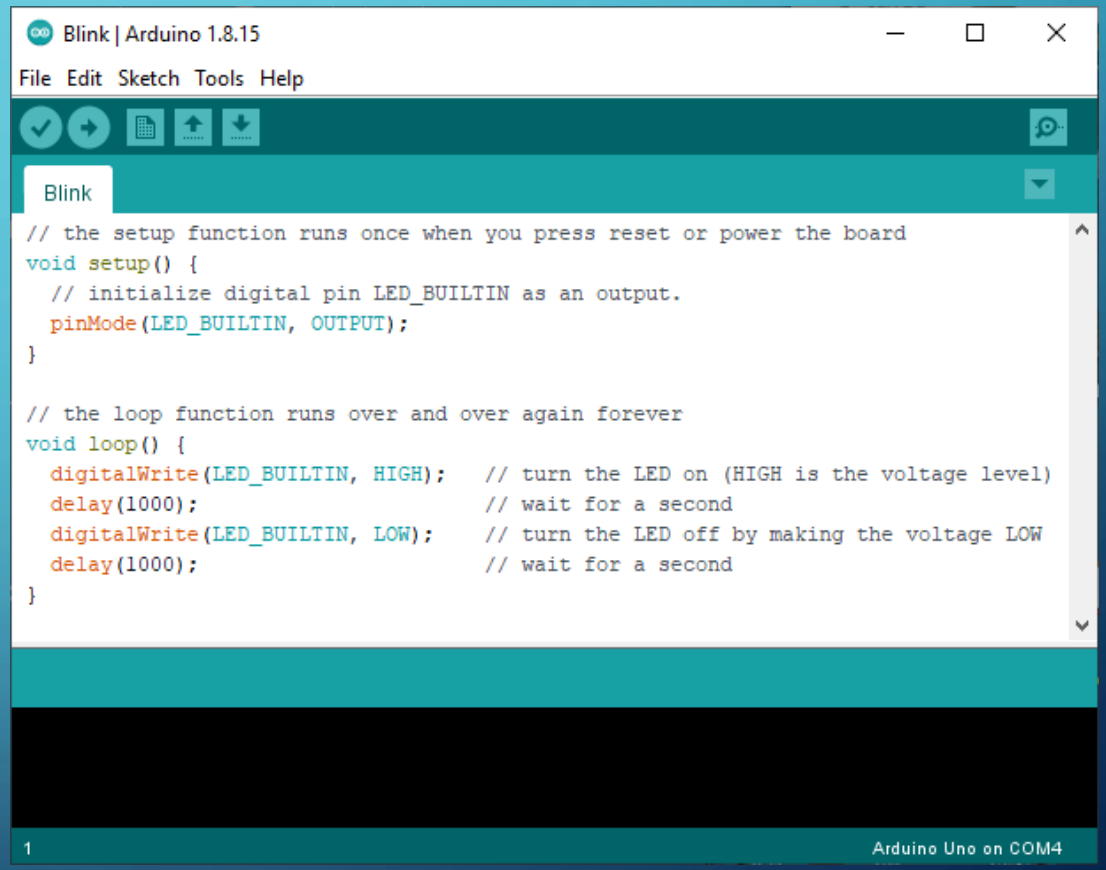
Arduino and its pins



Arduino connected to electronics on a solderfree breadboard

THE ARDUINO PLATFORM

- Is programmed through its ”integrated development environment” (IDE)
- Programs always contain:
  - setup() – commands to run only once
  - loop() – commands that will loop forever
- Is written in the C/C++ programming language



The official Blink program example in Arduino IDE

WHY ARDUINO?

- Observe this program:
  - Barebones AVR program
  - You need to know the processor, port and pin adresses
  - Is difficult to read
  - But it IS efficient on space

```
1  //Physical LED pin
2  #define LEDWRITEPORT PORTB
3  #define LEDPIN 5
4  //Physical button pin
5  #define BTNREADPORT PIND
6  #define BTNPIN 3
7
8  int main() {
9
10   //Initialize pins
11   DDRB |= (1 << LEDPIN); //ledPin set to output
12   DDRD &= ~(1 << BTNPIN); //btnPin set to input
13
14   while(1){
15     //If button is pressed LED on
16     if((PIND & (1 << BTNPIN)) >> BTNPIN == 1){
17       PORTB |= (1 << LEDPIN);
18     }
19     //Else, LED off
20     else{
21       PORTB &= ~(1 << LEDPIN);
22     }
23   }
24 }
```

```
Sketch uses 148 bytes (0%) of program storage space. Maximum is 32256 bytes.
Global variables use 0 bytes (0%) of dynamic memory, leaving 2048 bytes for local variables. Maximum is 2048 bytes.
```

WHY ARDUINO?

- Now observe this program:
  - Readable by humans
  - Simple operations
  - Obvious structure
- Takes up a little more space
  - Fair tradeoff for easier programming

```
1  const int ledPin = 13; //Physical LED pin
2  const int btnPin = 3; //Physical button pin
3
4  void setup() {
5    //Intialize pins
6    pinMode(ledPin, OUTPUT); //ledPin set to output
7    pinMode(btnPin, INPUT); //btnPin set to input
8  }
9
10 void loop() {
11   //If button is pressed LED on
12   if(digitalRead(btnPin) == HIGH){
13     digitalWrite(ledPin, HIGH);
14   }
15   //Else, LED off
16   else{
17     digitalWrite(ledPin, LOW);
18   }
19 }
```

TO GET STARTED:

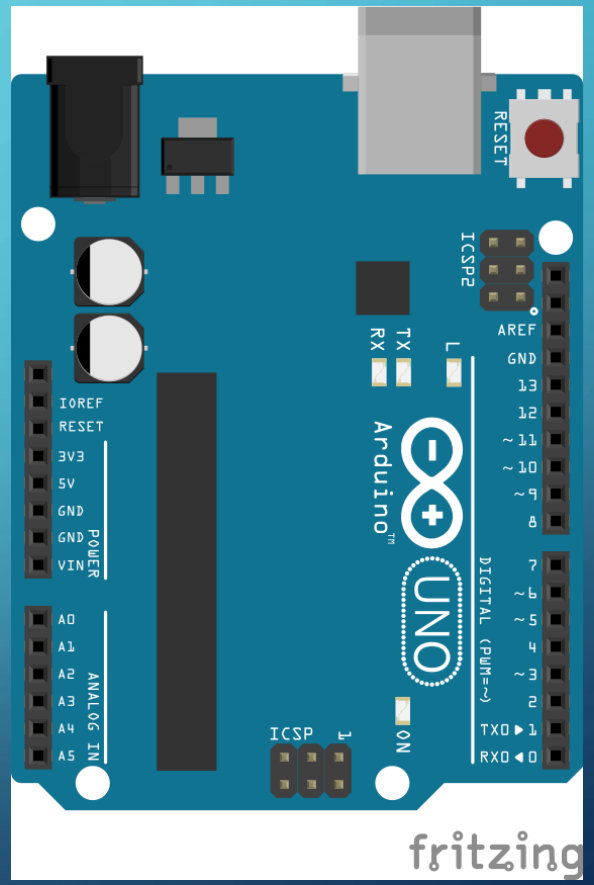
- Download and install the Arduino IDE
- Get the version from your platform as outlined
  - For the love of all that is good – do not get the Windows Store version!
- I have several examples hosted on Github: https://github.com/iakop/ArduinoCrashcourse

https://www.arduino.cc/en/software

BLINKY EXAMPLE

EXAMPLE 1: BLINKY

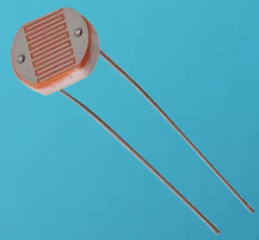
- To test if the Arduino works
- We will write a ”Blinky” program
- You will only need the Arduino itself
- We will write the code together



SENSORS AND ACTUATORS

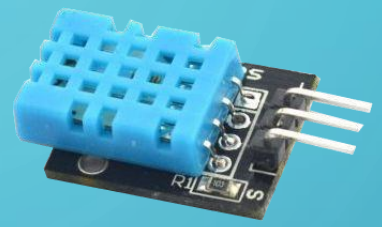
SENSORS

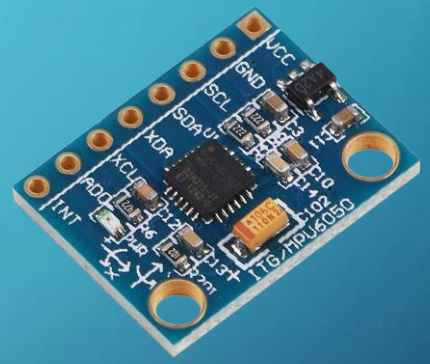
- Components that act as input
- They SENSE the outside world
  - E.g.:
    - Buttons
    - LDR's (Light Dependent Resistors)
    - Potentiometers
  - Simple to code
    - Electrical signals easily to translate to data.

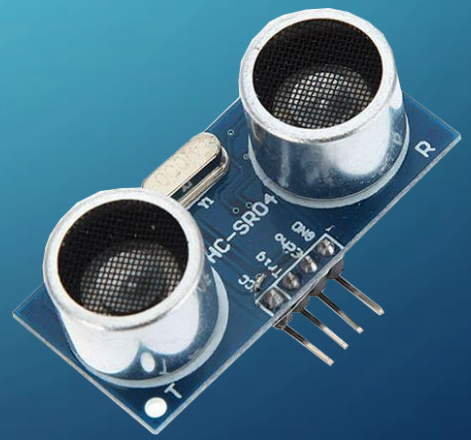


SENSORS

- More complicated sensors
  - Often made as modules
  - E.g.:
    - Temperature/humidity sensor
    - Accelerometer/gyro
    - Ultrasonic distance-sensor
  - Abstracts away a lot of ”glue electronics”
- Usually communicates through a standard protocol:
  - I2C
  - SPI
  - UART
- Although sometimes their own unique protocol





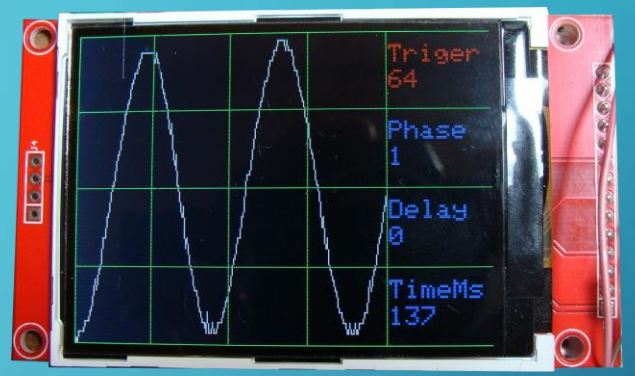


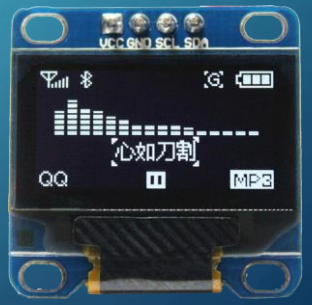
ACTUATORS

- Counterpart to sensors
- Actuators act as output
- They ACT on the outside world
- Most actuators are simple:
  - Motors
  - Speakers
  - Linear actuators
  - Pistons
- Some are a little more complicated
  - Servo motors
  - Stepper motors
- Most use no, or simple protocols

OTHER OUTPUTS

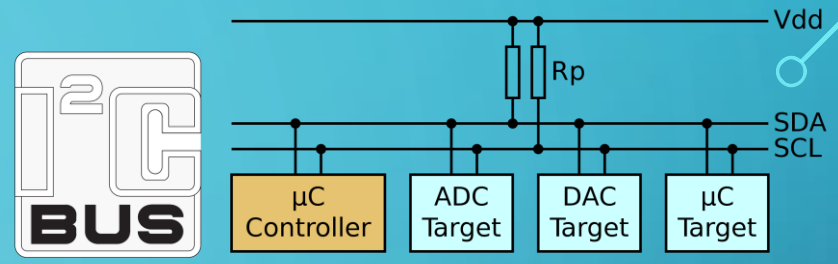
- Other outputs are usually indicators or interfaces:
  - LED's
  - Character Displays
  - OLED / TFT LCD
- Usually require a standard protocol:
  - SPI
  - Parallel Interface
  - I2C

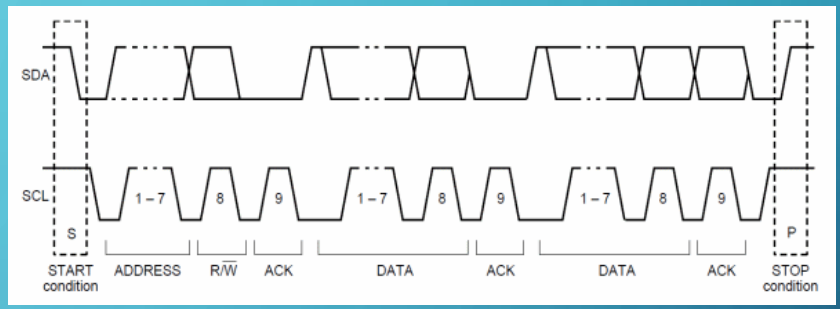




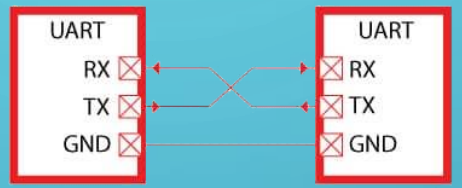
PROTOCOLS

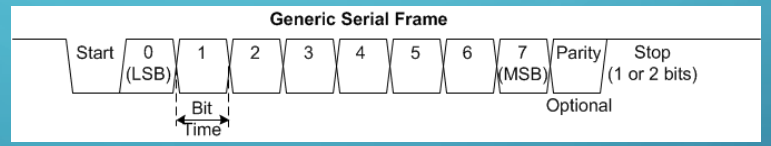
- Protocols cover the interfaces and language spoken between chips
  - E.g. between Arduino and a sensor
- The most widespread:
  - UART
  - I2C
  - SPI
- Arduinos libraries usually handle these!



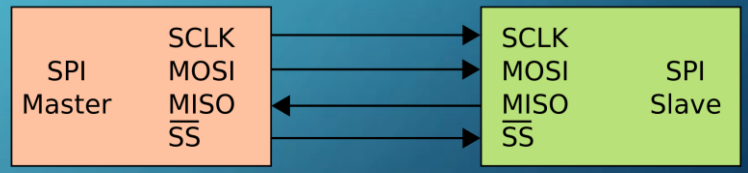


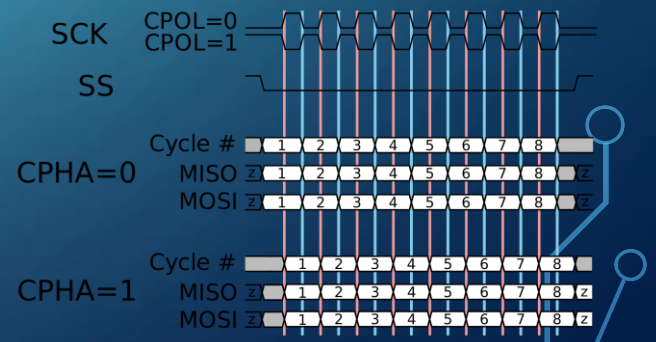
Pictured: I2C connections, and dataframe
URL: https://en.wikipedia.org/wiki/I%C2%B2C





Pictured: UART connections, and dataframe
URL: https://microcontrollerslab.com/uart-communication-working-applications/





Pictured: SPI connection, and dataframe
URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

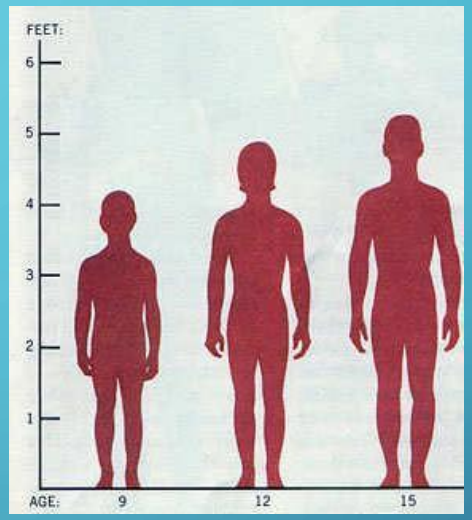
DIGITAL AND ANALOG ELECTRONICS

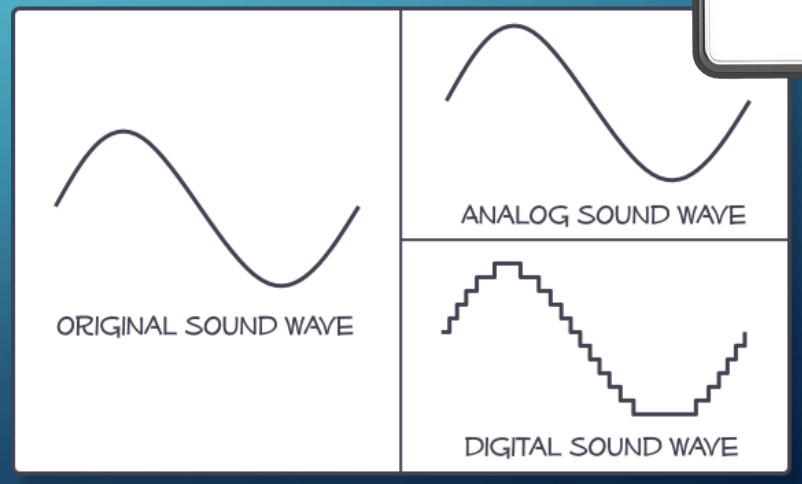
DIFFERENCE BETWEEN DIGITAL AND ANALOG?

- Digital values are either true/false:
  - Is the door open or closed?
  - Is the button pushed or not?
  - Is the coffee ready?
- Also known as boolean values

DIFFERENCE BETWEEN DIGITAL AND ANALOG?

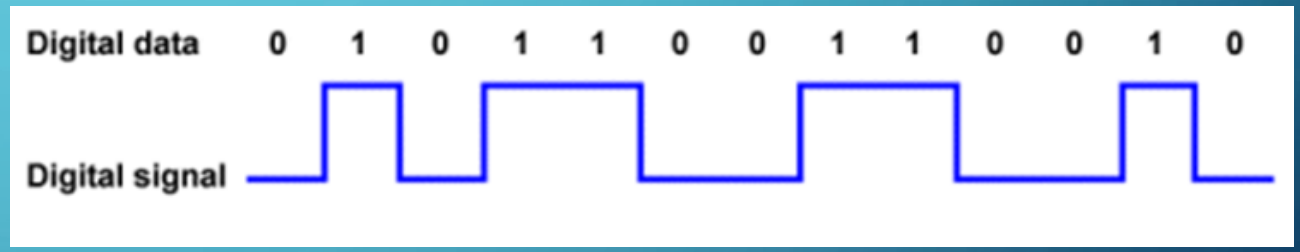
- But most of the world is analog:
  - People can vary in height
  - Time is measured in infinitesimal fractions
  - So can physical phenomena:
    - Within HiFi audio, minuscule variations in sound are fussed over





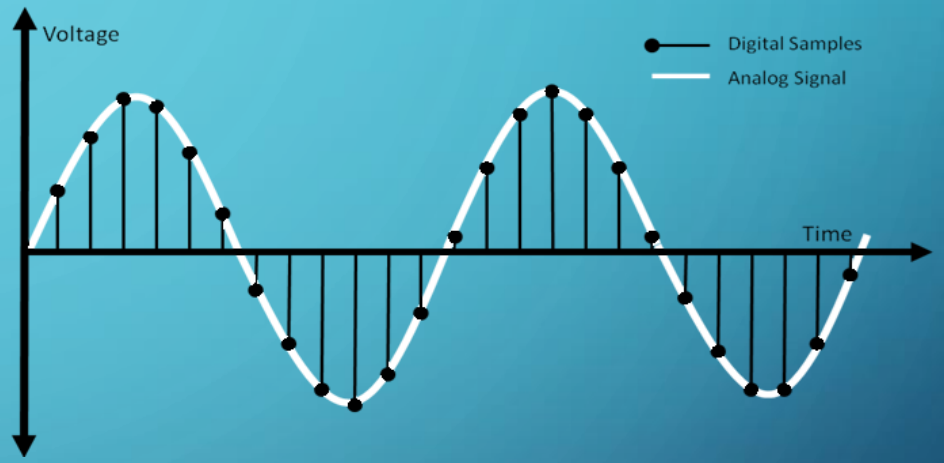
DIGITAL WITHIN ARDUINO

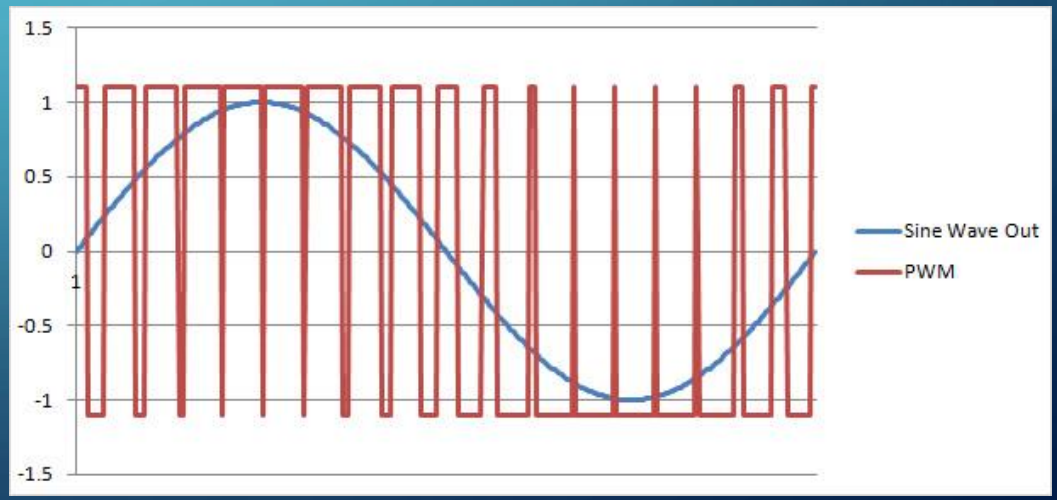
- Arduino can ”read” and ”write” digital data:
  - A voltage, either 0V or 5V
  - 0V = 0/false/LOW
  - 5V = 1/true/HIGH
- Represented on the Arduino as either: HIGH or LOW
- Voltages are interpreted from a threshhold, and rounded to either 1or 0



ANALOG WITHIN ARDUINO

- Arduino can also measure analog data
  - Done on the analog input pins
  - Called ”sampling”
- It can also output a ”semi” analog signal
  - Through PWM (Pulse Width Modulation)
    - Quickly turning on and off a digital pin
  - Same method as class D amplifier
    - Or voltage conversion (DC to DC)

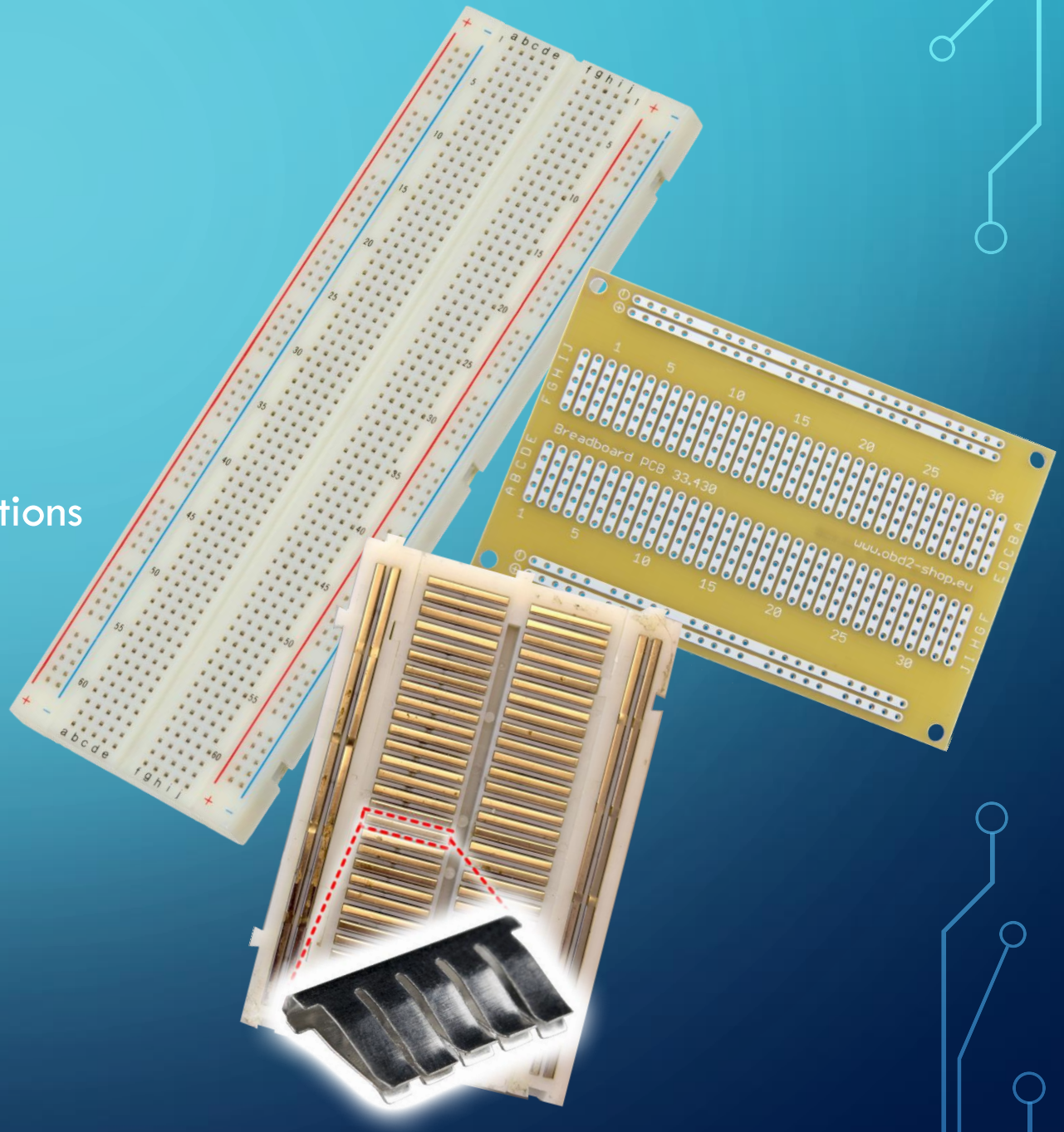




EXAMPLES

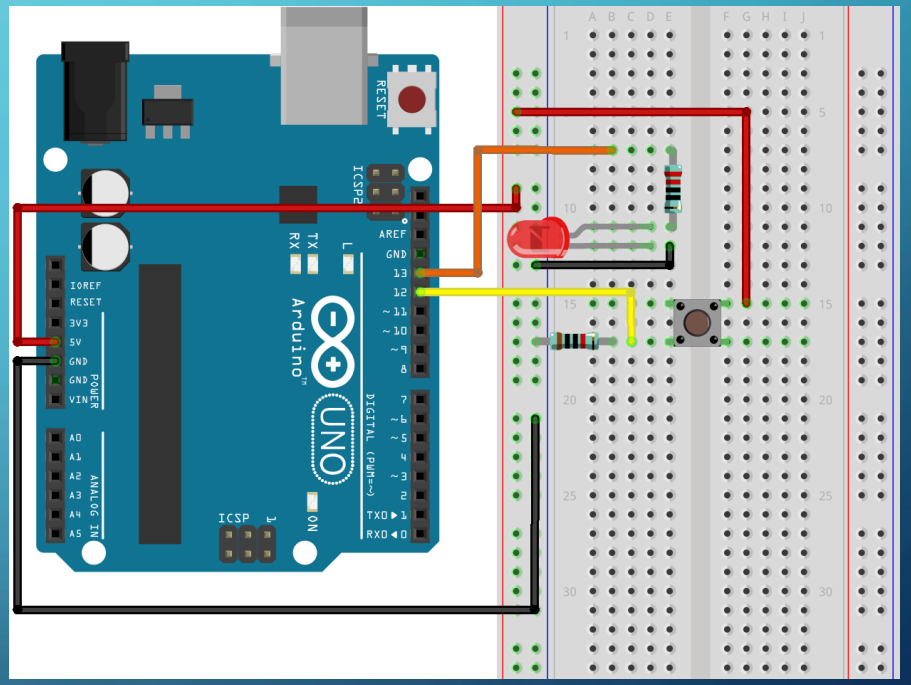
ABOUT BREADBOARDS

- Breadboards exist for easily prototyping electronics
- Wires go into the holes and make connections through clips on the inside
- Two types of lines exist:
  - Bus strips
    - Connects all the way through
  - Terminal strips
    - Connect ~5 pins
    - Interrupted in the middle



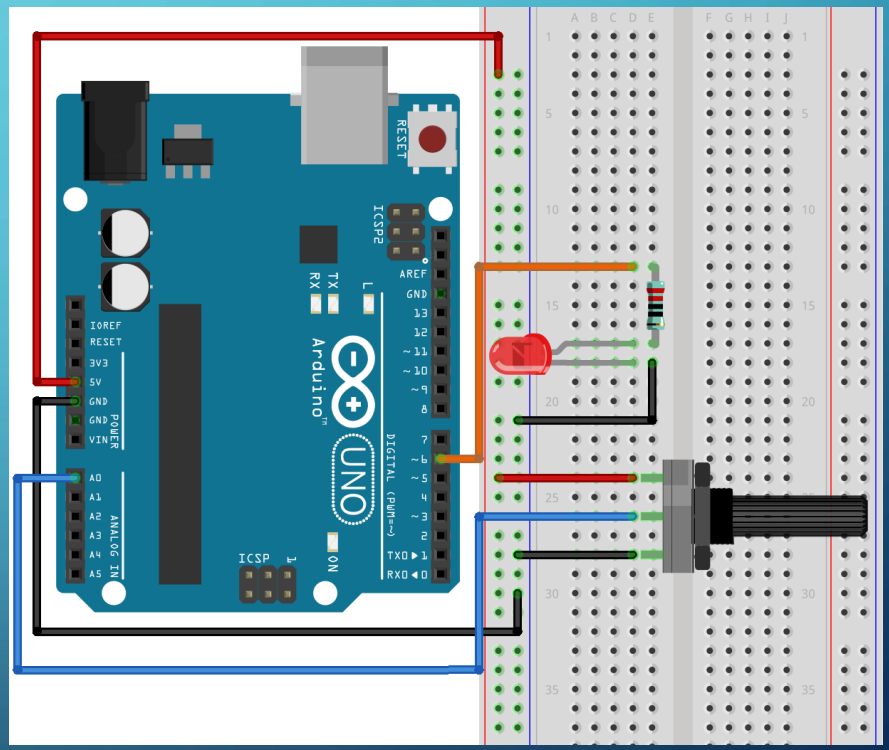
EXAMPLE 2: BUTTON

- Now we will add a circuit
- We try using an input – the button
- We'll control the LED using a boolean value



EXAMPLE 3: ANALOG

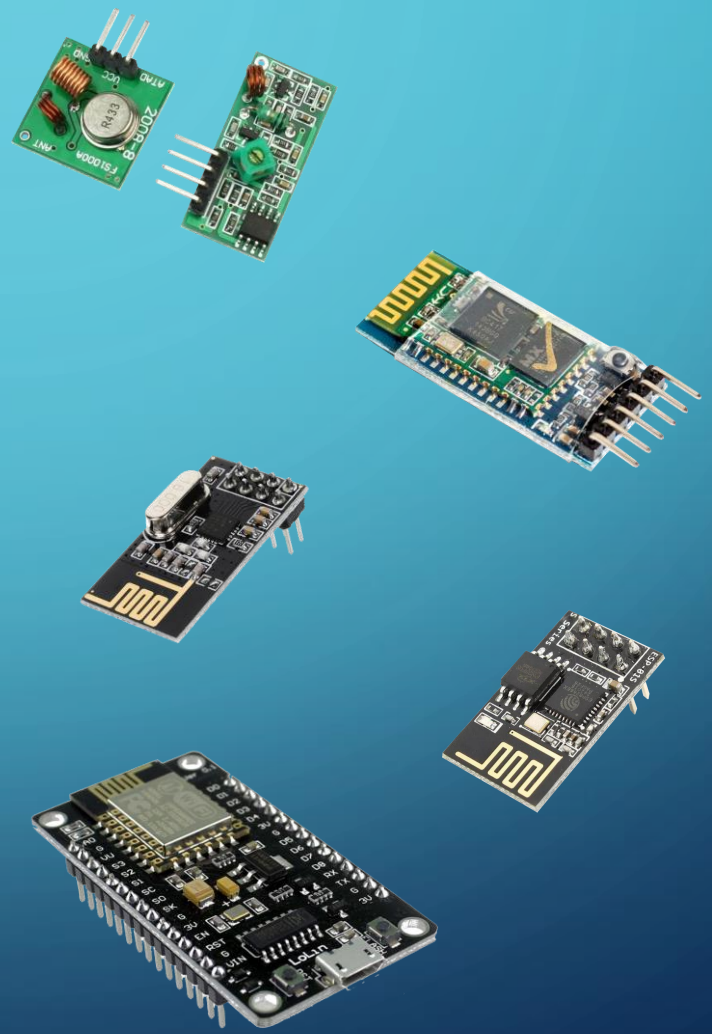
- Onto analog sampling!
- We'll build a new circuit
- We'll also make the Arduino communicate with the PC
- And use PWM ("analog output") to dim the LED



PART 2: WIRELESS COMMUNICATION

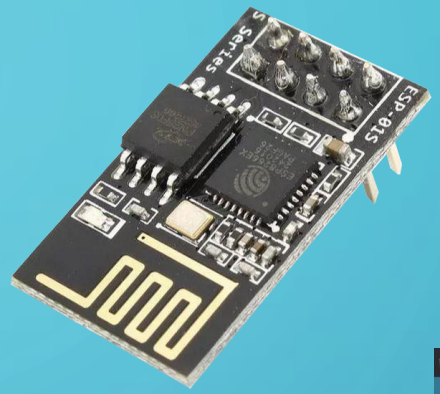
TYPES OF WIRELESS

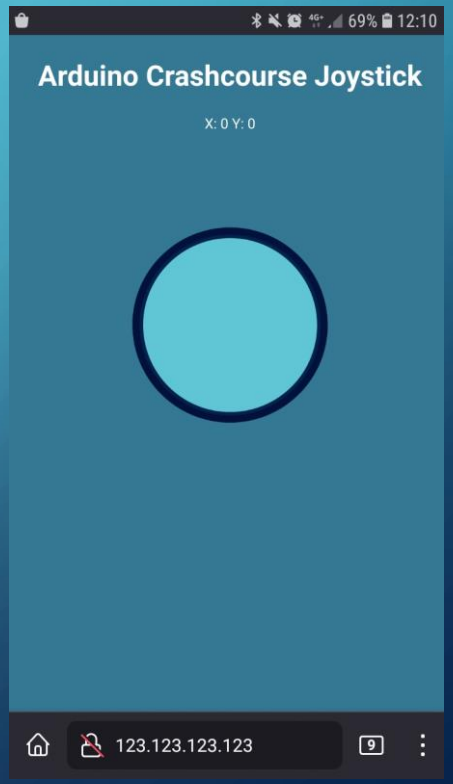
- FS1000A + XY-MK-5V
  - 433MHz – one way
  - Amplitude Shift Keying
    - Very susceptible to noise
- HC-05
  - 2.4GHz Bluetooth 2.0
  - Works as both master and slave
- NRF24L01
  - 2.4GHz
  - Gaussian Frequency Shift Keying
  - Supports BLE and "Enchanced Shockburst"
- ESP-01 / ESP-8266
  - WiFi -2.4GHz
  - Works as Access Point or Client
- ESP-32
  - Beefier ESP-8266 w/ BLE
- ESP's are actually MCU's themselves



ESP-01

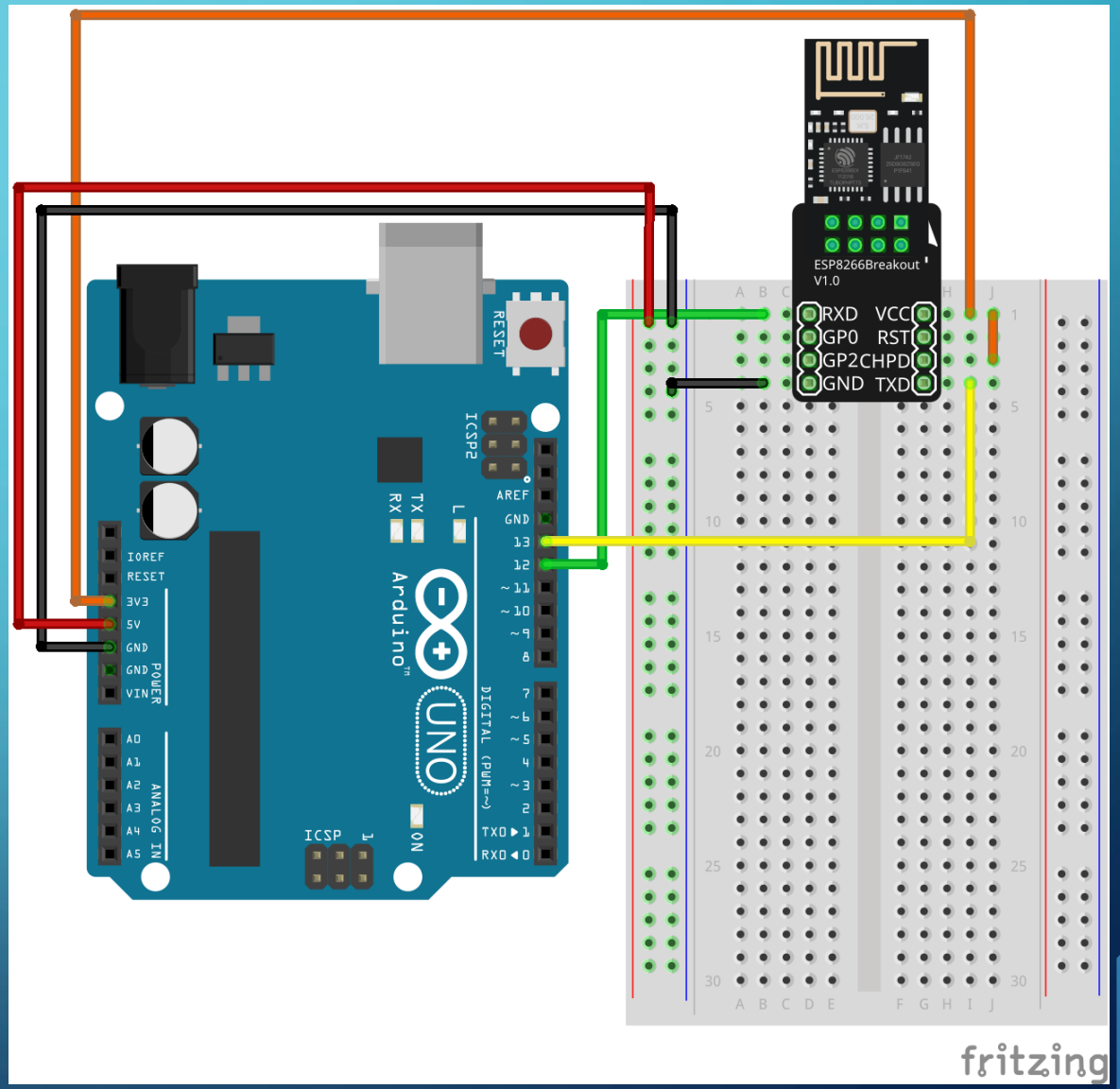
- ESP-01 contains a ESP8866 MCU
- Is set up as a Serial ”Modem”
  - Gives Arduino simple WiFi communication
- Best for simple IoT projects
- Ours has custom software (written by me)
  - Code on: https://github.com/iakop/ArduinoCrashcourse
- Serves a webpage on custom access point and IP
  - Sends X and Y coordinates over Serial
  - Can be used as remote control

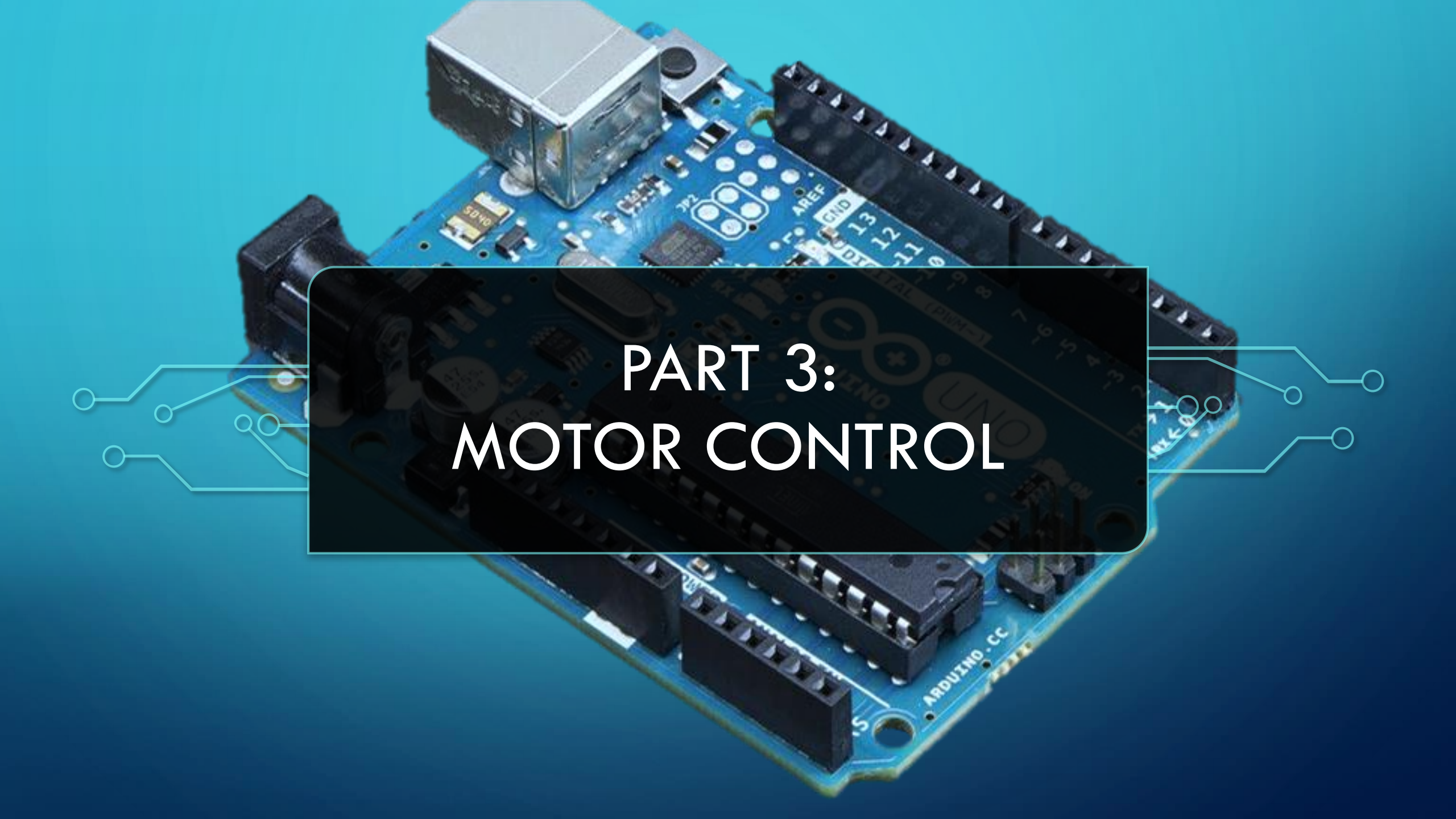




WIRELESS EXAMPLE

- The ESP-01 uses a Breadboard breakout
- We'll connect it to Arduino through Software Serial
- Arduino instructs it with an IP, SSID and password
- Our goal is to acces the webpage
  - And get X and Y over Serial





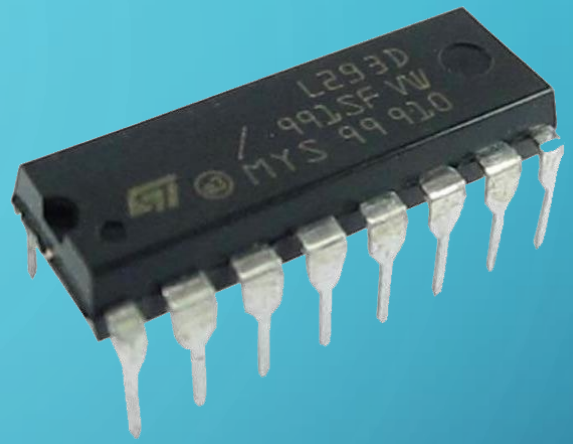
PART 3: MOTOR CONTROL

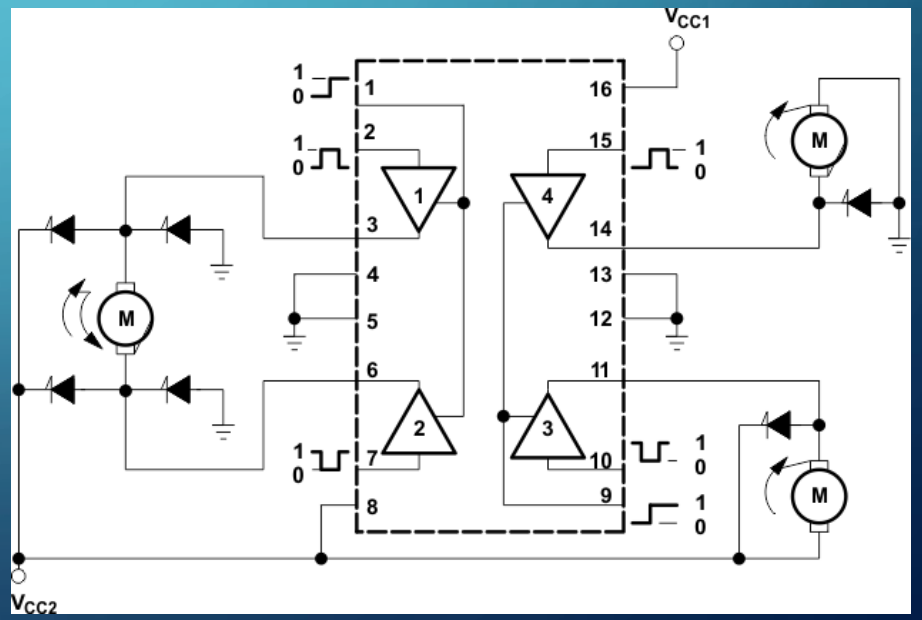
DC MOTORS

- DC motors are the simplest motors to control
- Only require a current to run
- Can vary speed depending on the current over time
  - Usually through PWM
  - Think ”analog speed”

L293D DRIVER

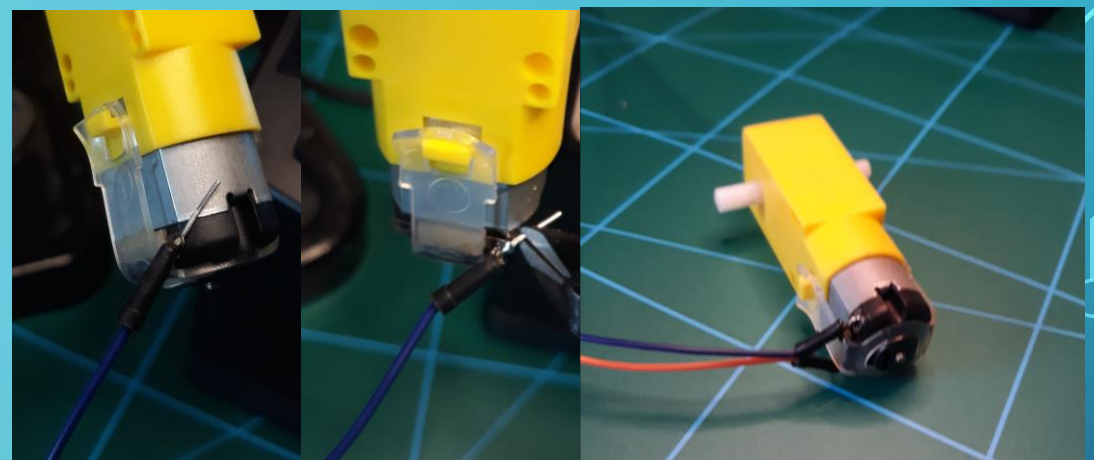
- Arduino does not supply powerful enough current for motors
  - We need an ”amplifier”, or driver
- L293D is a quad half H-bridge driver
- Can be used for:
  - 4 motors one way
  - 2 motors back/forth
  - We’ll use it for 2 back/forth

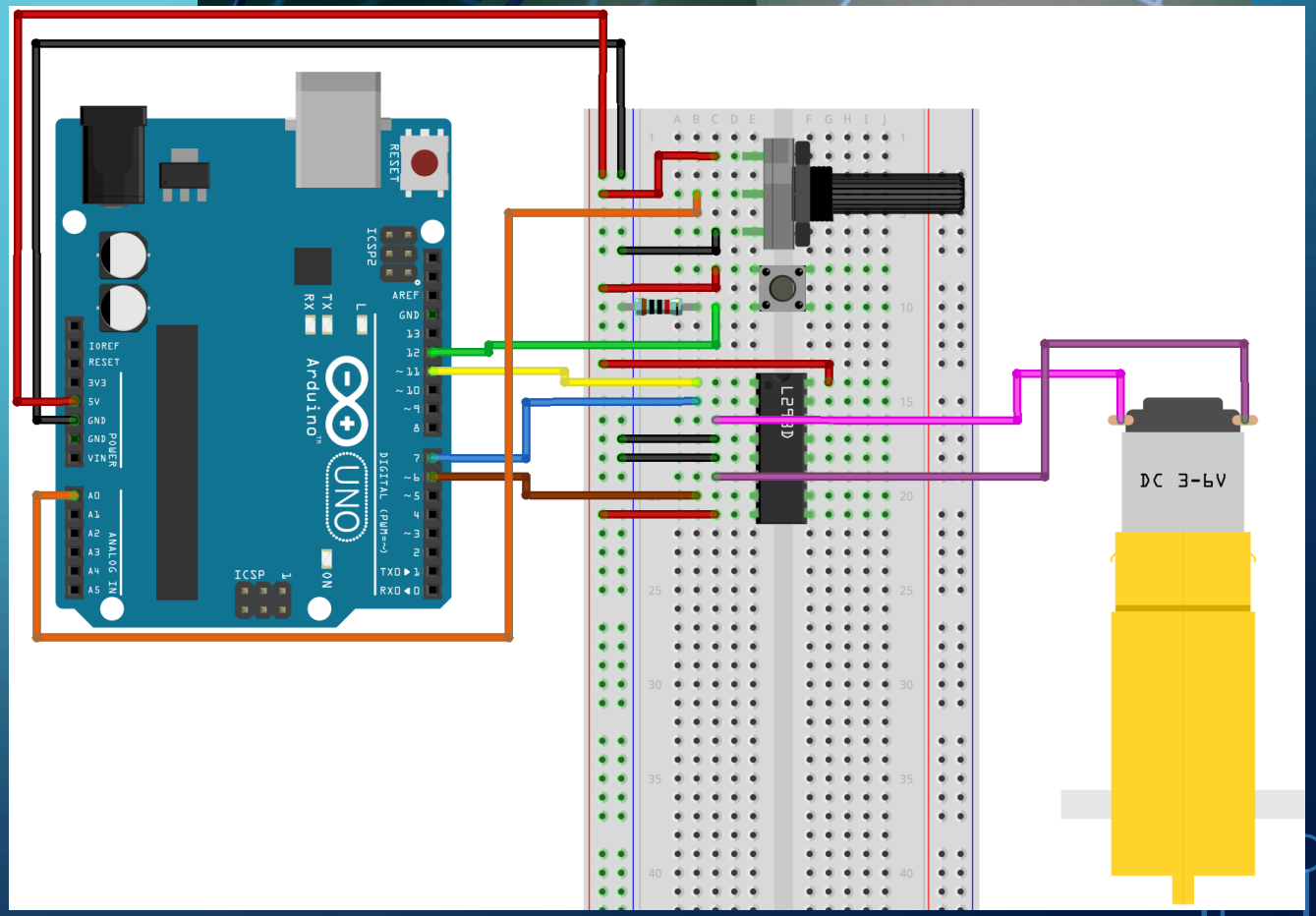


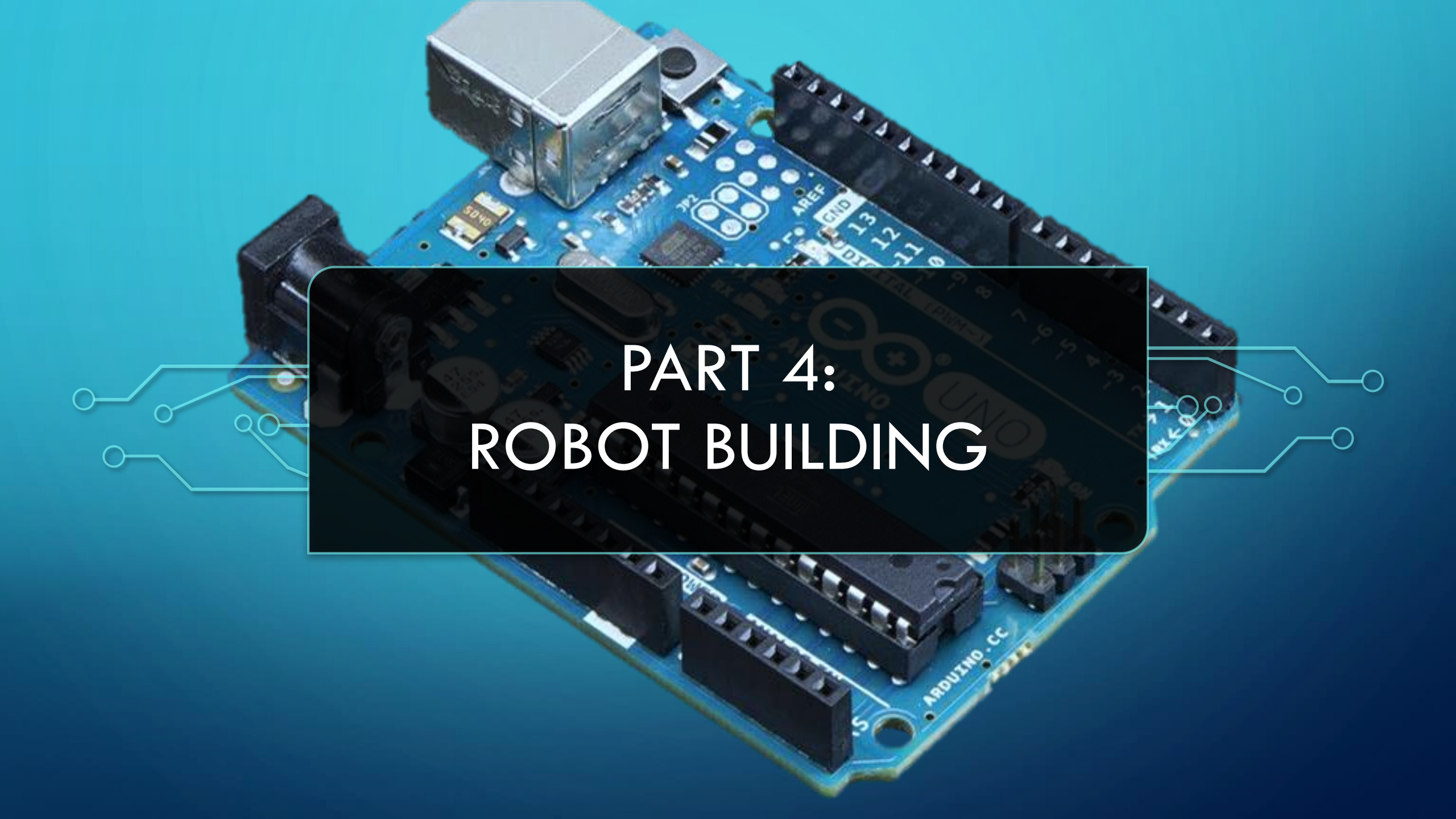


MOTOR DRIVING EXAMPLE

- We'll set up the control pins of the L293D
- Plug one of the DC-motors into it and test
  - We need to solder it first
  - Place a cable in the hold, solder, and cut to size
  - Repeat twice
  - Might as well solder both
- We instruct the Arduino to drive it back/forth
- We'll also try to vary the speed with analogWrite()



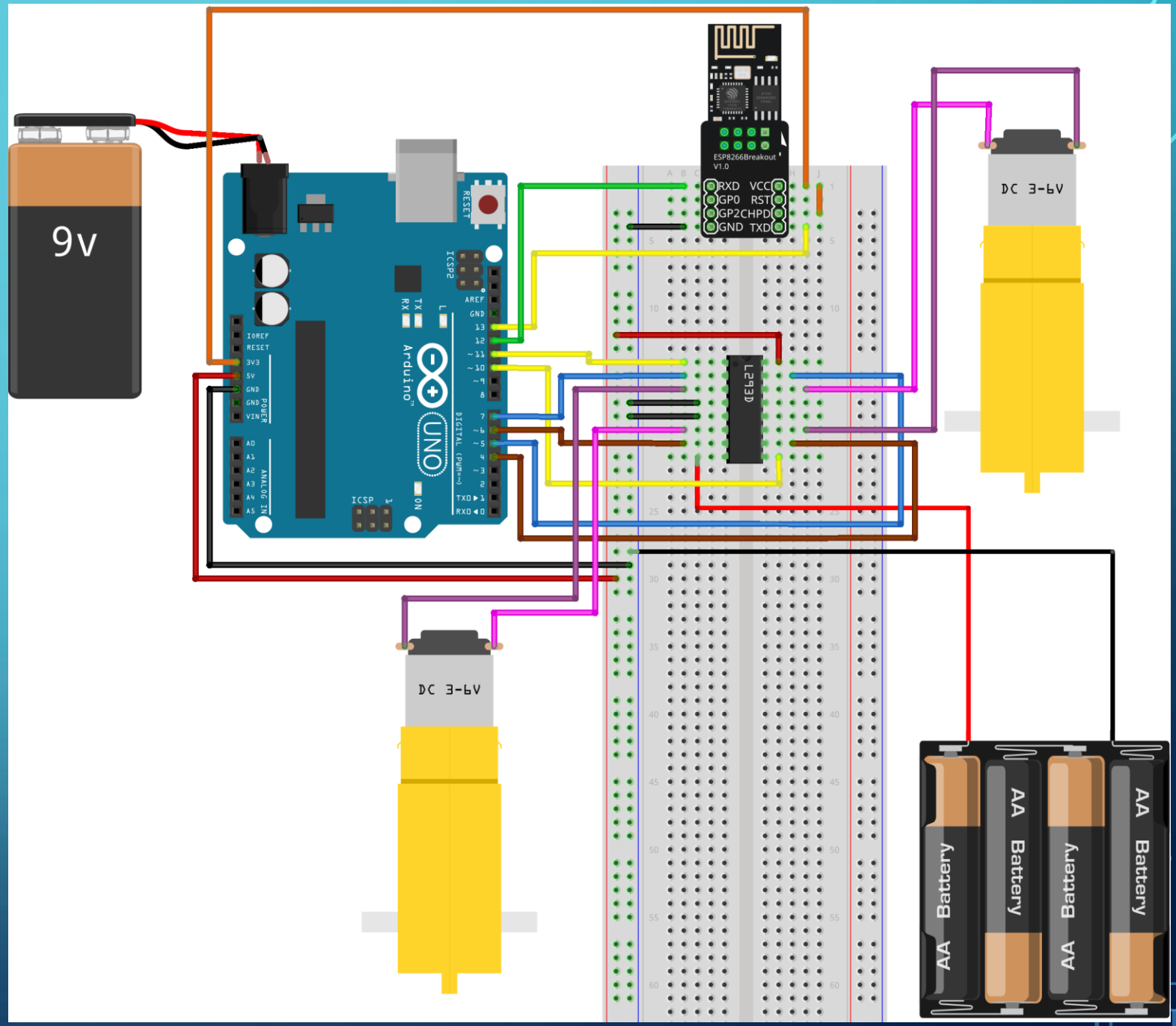




PART 4:
ROBOT BUILDING

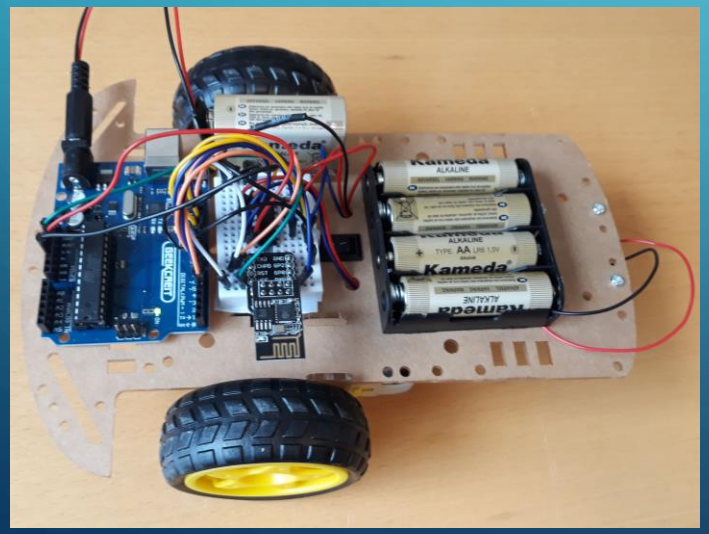
PUTTING IT ALL TOGETHER

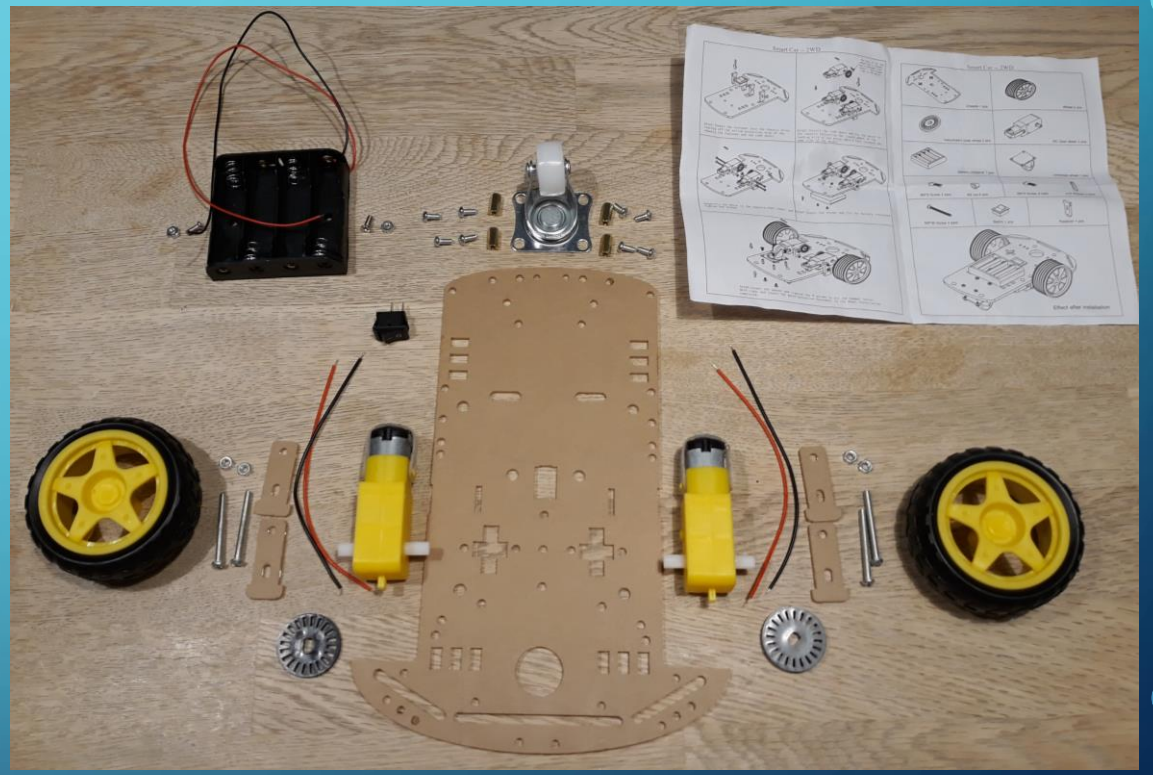
- Now we'll put all the electronics together
- Two DC Motors on the L293D
- ESP-01 connected to Arduino
- Arduino will interpret the coordinates
  - Drive the motors through L293D



WALKTHROUGH OF ROBOT

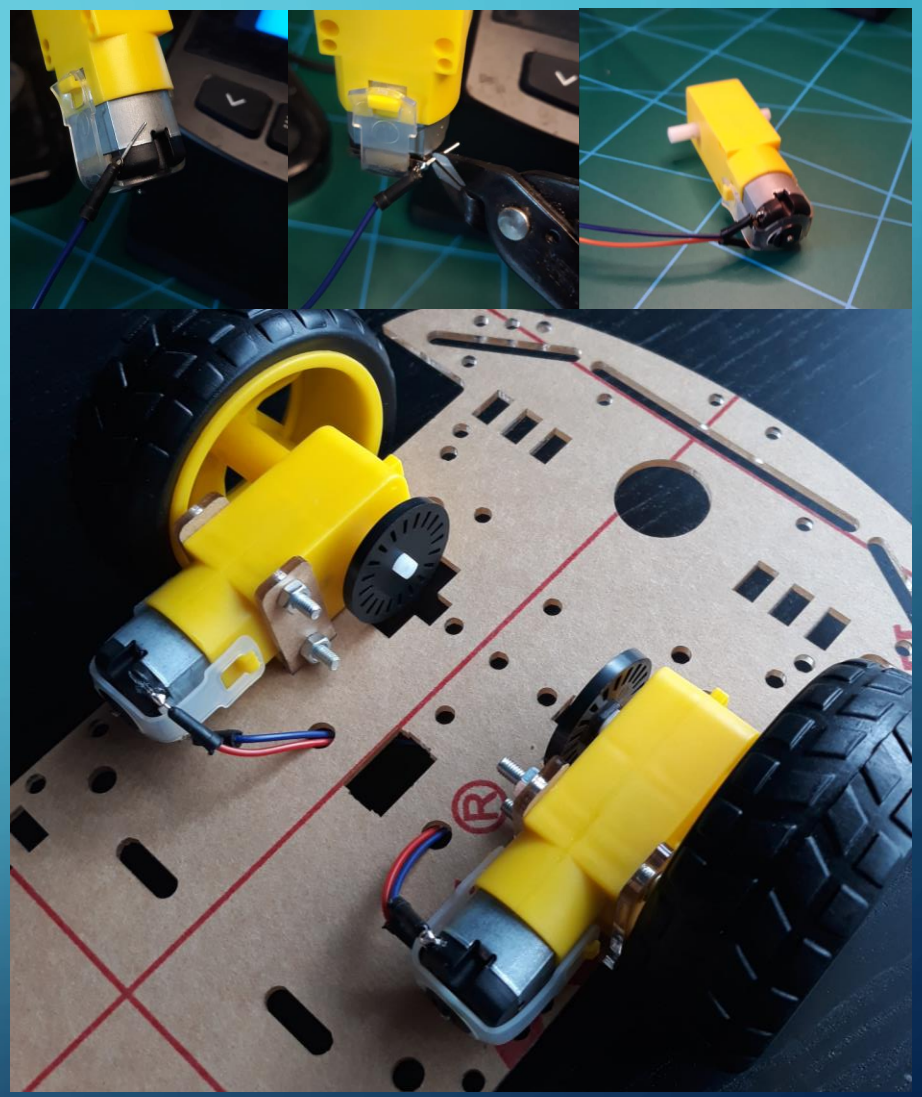
- With your kit came a robot set
  - Instructions for assembly are included
- When assembled, you circuits should be placed
  - An example can be seen here





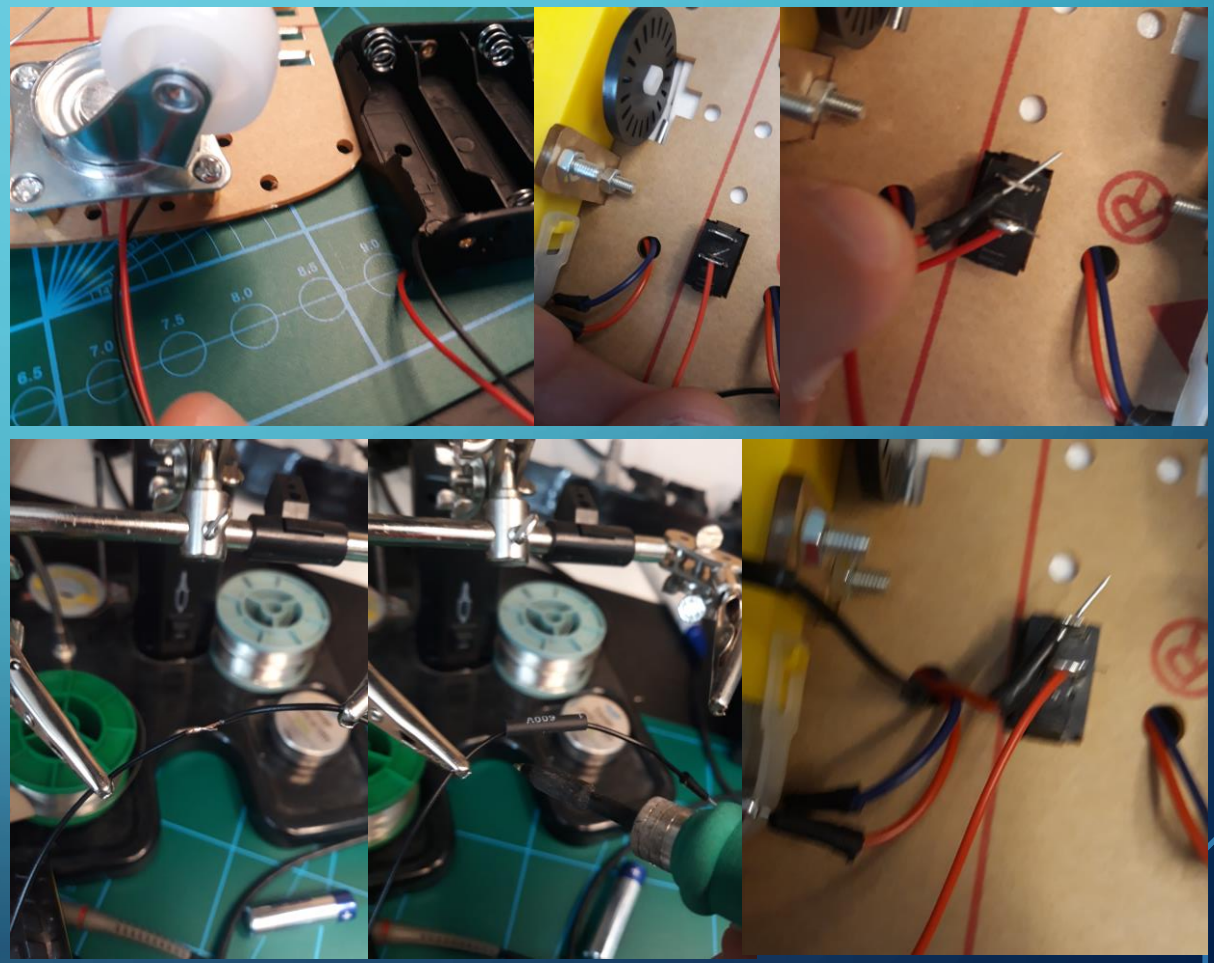
WALKTHROUGH OF ROBOT

- Some things must be soldered
  - Both motors
    - Place a cable in the hold, solder, and cut to size
    - Repeat twice
    - Place the wires in holes on back



WALKTHROUGH OF ROBOT

- Some things must be soldered
  - Battery holder
    - Run the cables through the castor on the back
    - Put + (RED) through on power button pin and solder
    - Place another + (RED) on the other pin
    - Extend the – (BLACK) with another wire
    - Place some heatshrink on it for safety
    - Run the red and black ends through the motor wire holes
    - Now they can power the motors



FREESTYLE BUILDING

- Now you can build at your own pace
- We have solder stations, wire cutters, screwdrivers etc.
  - Take turns, then everybody gets done
- I'll assist if any questions come up
  - I'll also assist if you want to add things to the circuit
    - But keep it simple

SUMMARY

- Now you should have an understanding of the basics
- You can build on robot, and add, for example:
  - Ultrasound sensor
  - Servo motor arm (servo included in kit)
  - More controls to the webinterface
  - Different control behaviour
- Many projects now work with connecting to the internet
  - ESP boards are not going away
  - Can be programmed through Arduino IDE

EVALUATION

- I'm open to direct feedback
  - What are some things you did/didn't like?
  - Could anything be made clearer?
  - How is the time alotted?
    - Is there enough?
- But most of all THANK YOU for participating!

RESOURCES

- A list of resources you can use:
  - Slides, examples, etc.
    - https://github.com/iakop/ArduinoCrashcourse
  - Arduino IDE Download
    - https://www.arduino.cc/en/software
  - Arduino Language Reference
    - https://www.arduino.cc/reference/en/