
ARDUINO
WORKSHOP

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 courses and webinars on Arduino for IDA, StudyNow, Herning Municipality, etc.
  - Full-time embedded electronics engineer at DTU, working with robots and automated systems

OVERALL AGENDA

- Today is the general workshop
  - We will learn the basics
  - We will touch some advanced concepts
- Tomorrow is for sparring
  - I will be available for one-on-one sparring
  - Will help with ideas, parts searching, etc.

PURPOSE OF THIS WORKSHOP

- To give you an understanding of the basics of Arduino
- Take you through the essentials of Arduino
  - Give you some hands-on successes
  - Also some encounters with the difficulties
- Give you an appreciation of the possibilities with this platform
- Inspiration to how to use Arduino for your own needs

TODAY'S CONTENTS

Part 1: Arduino Basics

- Introduction to Arduino
- Sensors and actuators
- Digital and analog electronics
  - Examples

Part 2: Motors and displays

- Types of motors
  - Examples
- Types of displays
  - Examples

Part 3: Biometric sensors and IoT

- Re-visiting sensors
- Pulse-oximeter
  - Example
- ESP32-CAM
  - Example
- IoT Devices
  - Example

BEFORE WE GET STARTED

- The code and examples throughout this workshop are all available on:
  - https://github.com/iakop/BioMedical2022
  - It's a good idea to keep this site open

PART 1:
ARDUINO BASICS

ARDUINO BASICS

Intro to Arduino

- What is a microcontroller?
- The Arduino Platform
- Why Arduino?
  - Other platforms

Sensors and actuators

- Sensors
- Actuators
- Other Outputs
- Protocols

Digital and analog electronics

- Difference between digital and analog?
- Digital within Arduino
- Analog within Arduino

Examples

- 1 – Make an LED blink
- 2 – Toggle LED with a button
- 3 – Analog values from a potentiometer
- 4 – Replace potentiometer with LDR

INTRO TO ARDUINO

WHAT IS A MICROCONTROLLER?

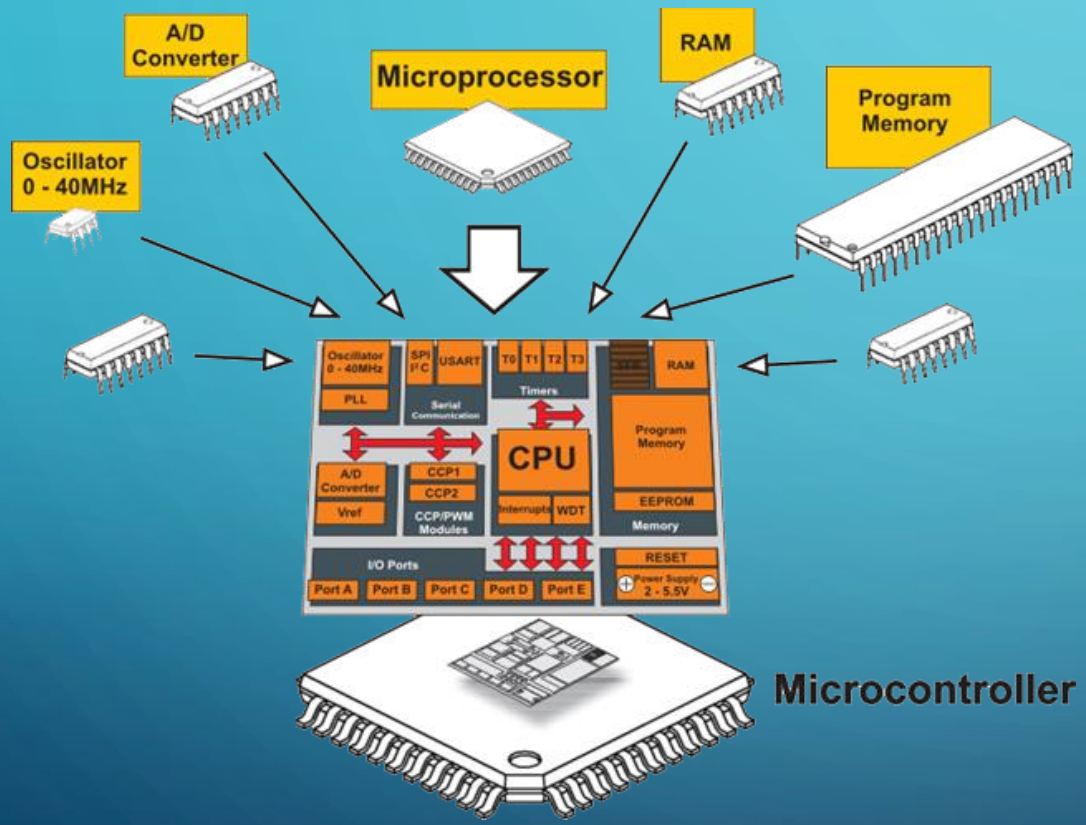
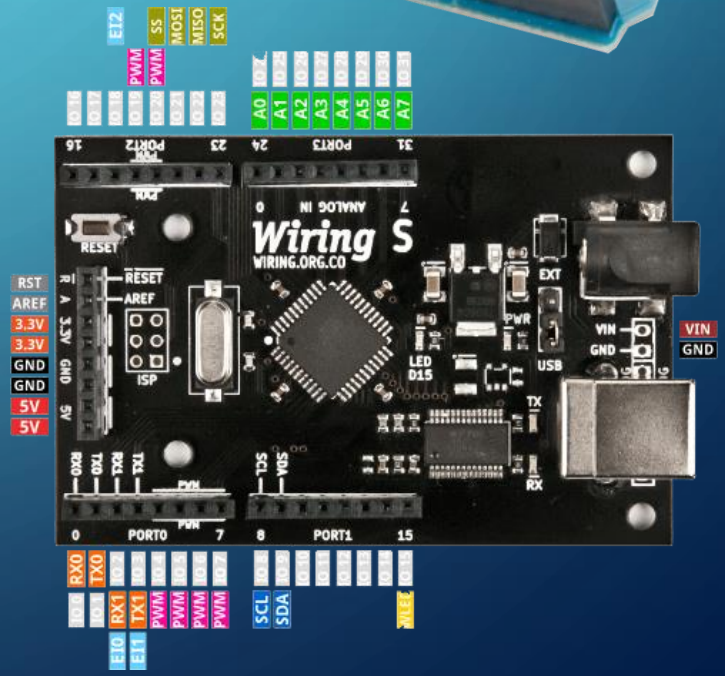


Figure from: Introduction to the World of MicroControllers – Mikroelektronika URL: https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c/introduction-to-the-world-of-microcontrollers

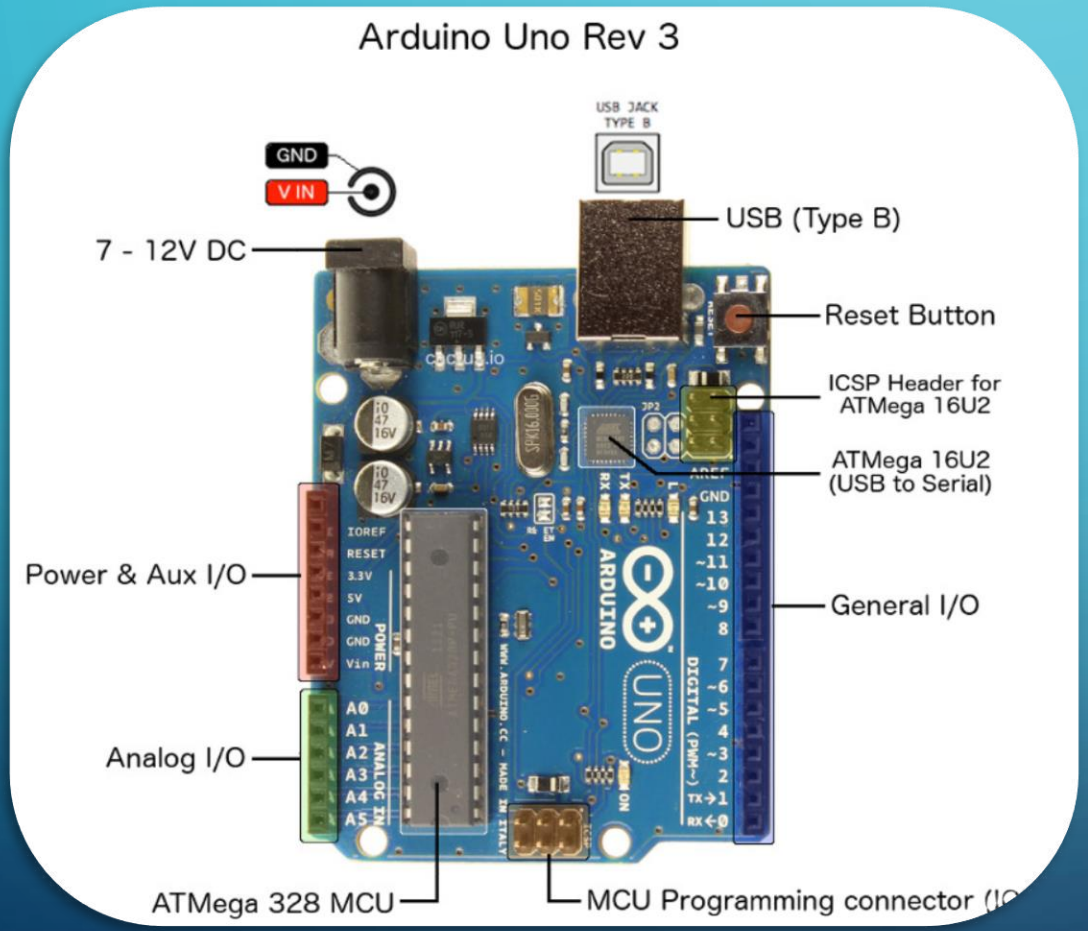
- MCU for short (MicroController Unit)
- They create our everyday magic
- Bind software and hardware together
- Not just processors:
  - Contain an entire system
  - Storage, memory, program memory, ADC mm.
- Easy to integrate in electronic circuits

THE ARDUINO PLATFORM

- The Arduino platform has an ”interesting” history
- Originally a fork of Wiring by Hernando Barragán
  - Master’s Thesis in Interaction Design
  - Based on the Procession programming language
  - Makes Microcontroller development easier, more approachable
- By far the most used platform for MCU prototyping
- Simple, intuitive programming
- Many code examples available
  - A lot of people have used it before us!
- Many available software libraries



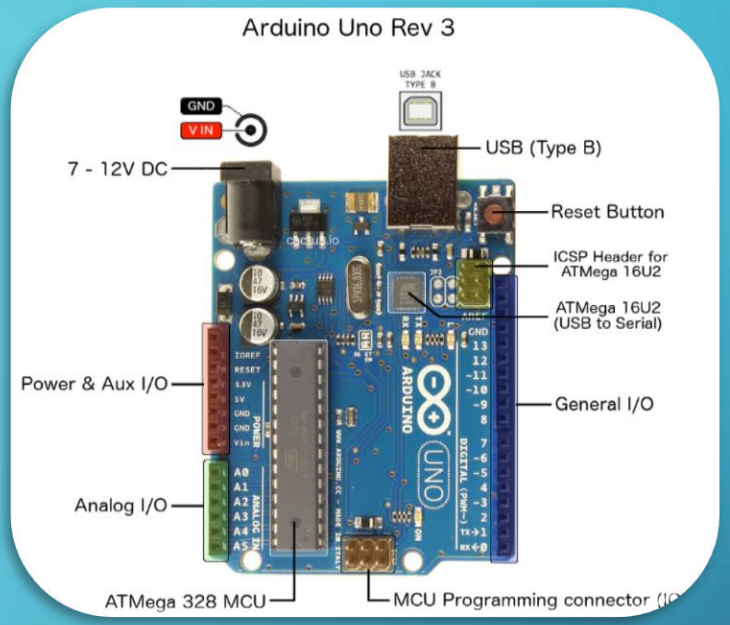
THE ARDUINO PLATFORM



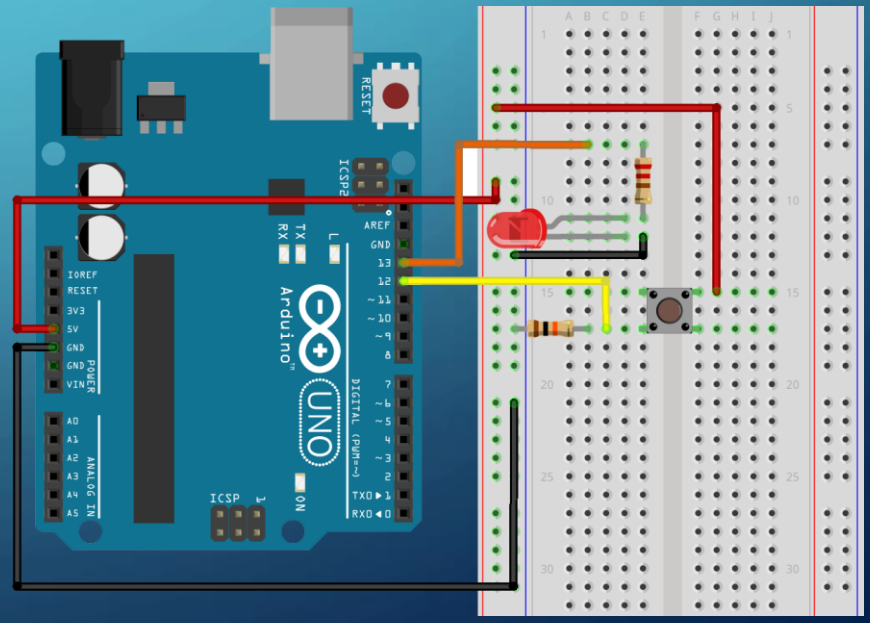
- The microcontroller (Atmega328) sits proudly on the board
- DC jack and USB type B for powering nd programming
- Most importantly the pins:
  - The most important:
    - RED Power supplies, reset etc.
    - GREEN Analog input/output
    - BLUE General Purpose input/output
- And other neat features

THE ARDUINO PLATFORM

- Arduino connects to electronics through its “pins”
- We can control their signals through our programs (turn on/off, etc.)
- Usually connects to through a ”breadboard”



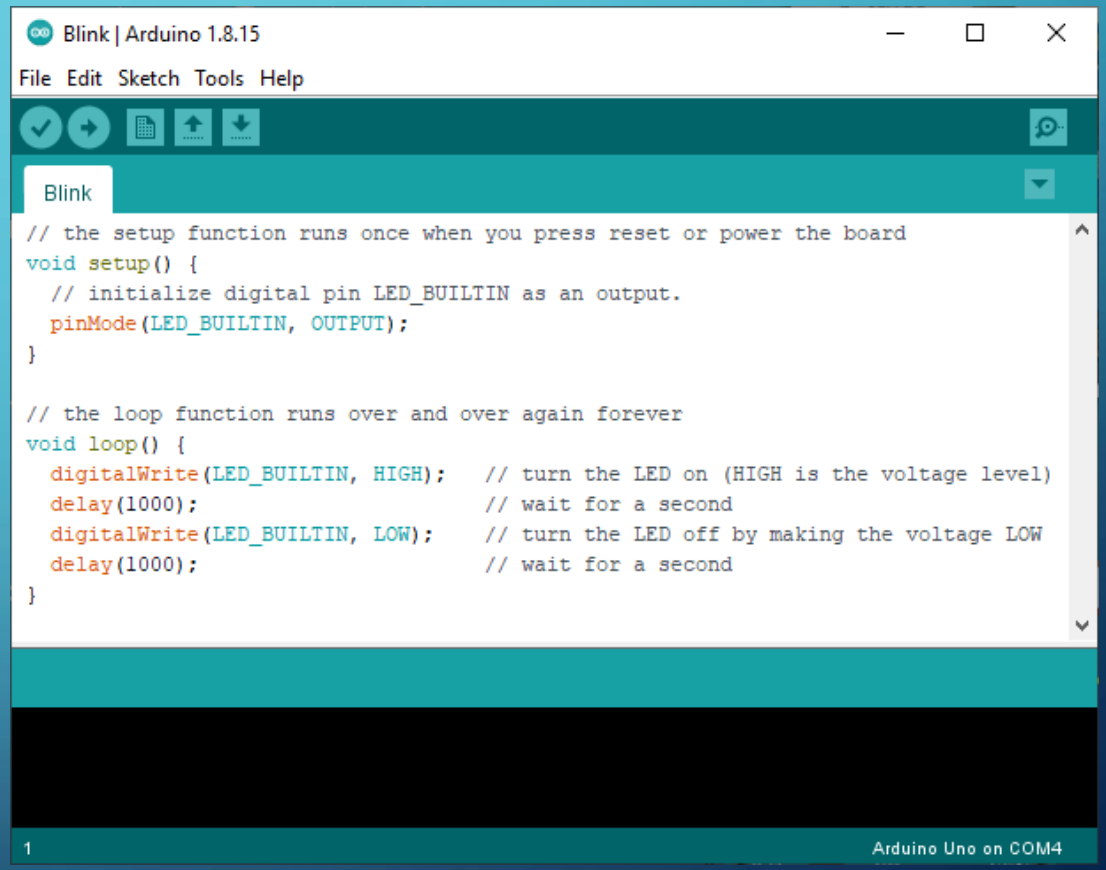
Arduino and its pins



Arduino connected to electronics on a solderfree breadboard

THE ARDUINO PLATFORM

- Is programmed through its integrated development environment
- Programs always consist of
  - 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:
  - Classic program for an AVR Microcontroller (The microcontroller in the Arduino)
  - You need to know the processor structure to understand it
  - Functionality is not easily discernible
  - But the program is actually rather 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:
- Obvious pros:
  - 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:

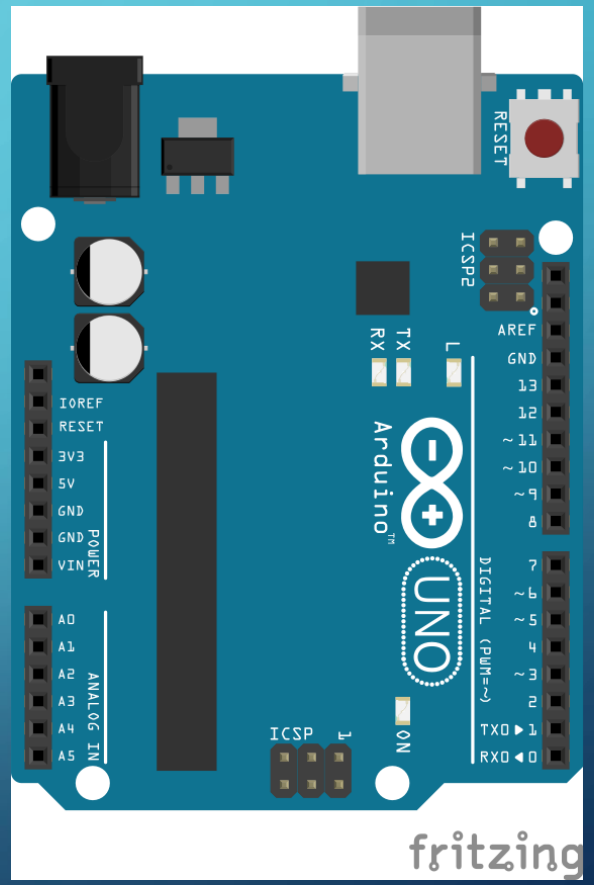
- Download and install the Arduino IDE
- Get the version from your platform as outlined
  - For the love of all that is good – do not get the Windows Store version!
- I have several examples hosted on Github: https://github.com/iakop/BioMedical2022/

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

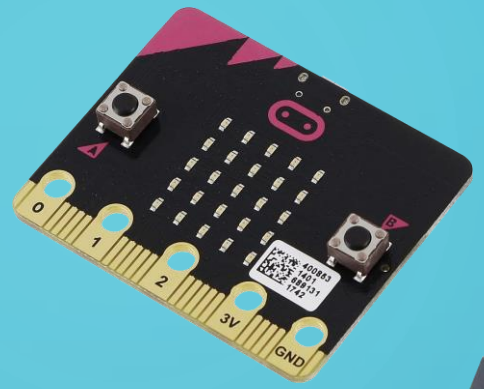
A QUICK EXAMPLE

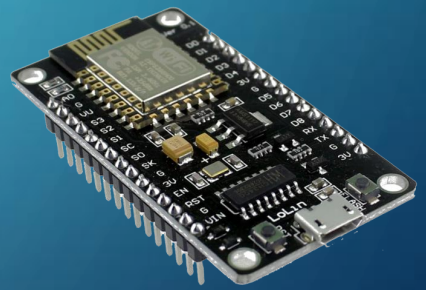
EXAMPLE 1: MAKE AN LED BLINK

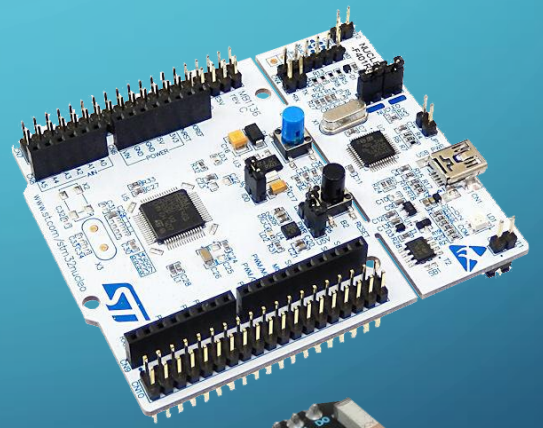
- To test if the Arduino works
- We will write a ”Blinky” program
- You will only need the Arduino itself
- We will write the code together

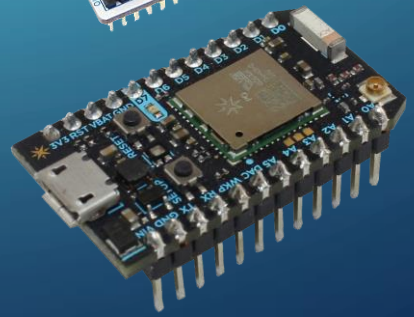


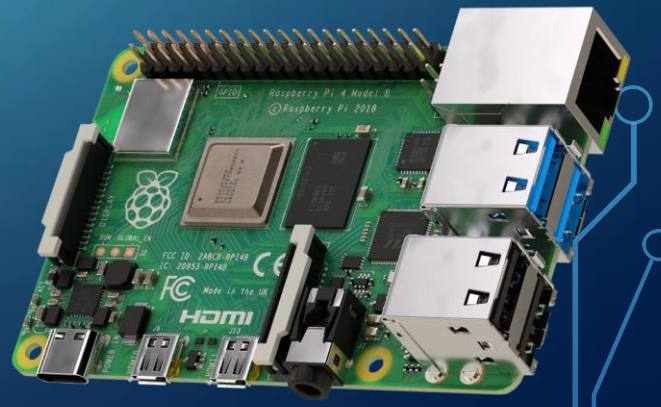
OTHER PLATFORMS









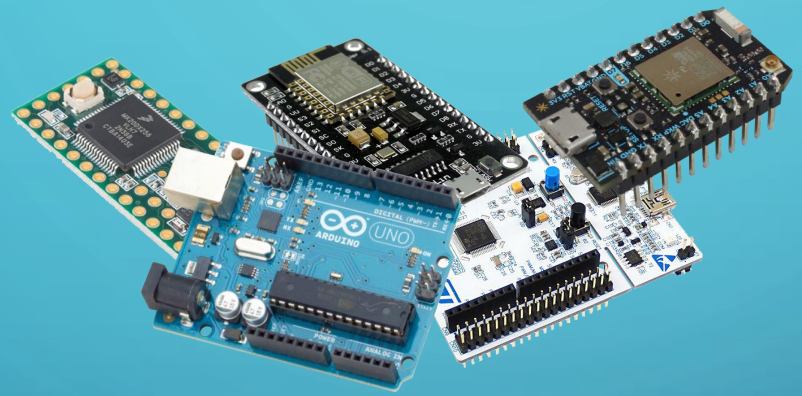


- Arduino is not the only prototyping platform
- Other honorable mentions are:
  - Micro:bit
  - Teensy
  - STM 32 Nucleo
  - NodeMCU
  - Electron
  - Raspberry Pi

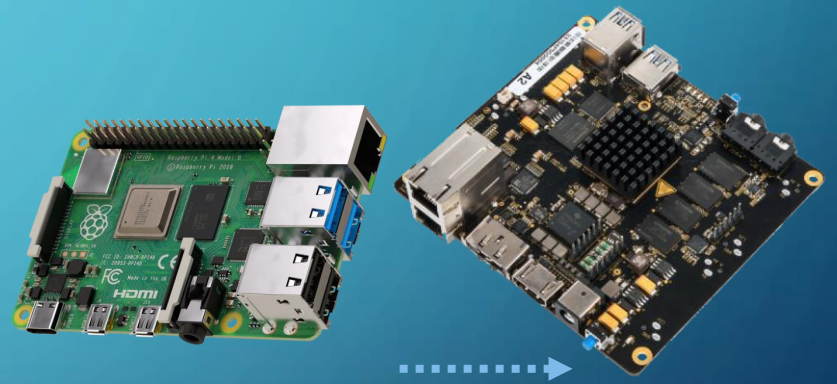
OTHER PLATFORMS

- By order of programming complexity:

- Block-based programming
- Built in display
- For grade-school use



- Simple control of electronics
- Big ecosystem with many available examples
- Easily controls electronic circuits
- Some even can connect to the internet



- Entire computer system, usually with Linux as OS
- Also a broad ecosystem
- For bigger computational loads, multimedia, IoT and web

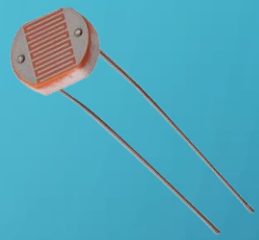
Simpler

More complex

SENSORS AND ACTUATORS

SENSORS

- Electronic components that act as input to the Arduino
- The SENSE the outside world
  - For example
    - Buttons
    - LDR's (Light Dependent Resistors)
    - Potentiometers
  - Simple enough to be manually handled in code
    - Electrical signals easy to translate to data.



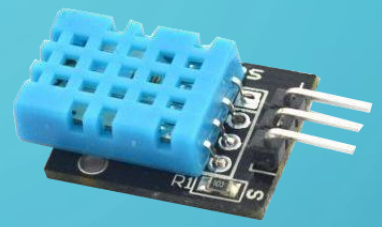
SENSORS

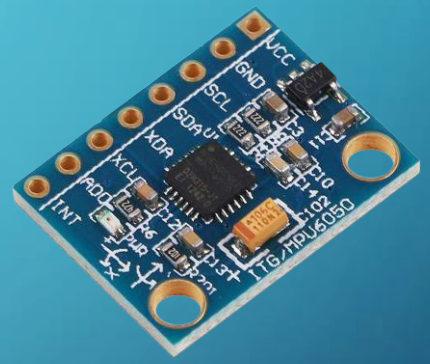
- More complicated sensors
  - Often made as modules
  - For example
    - Temperature/humidity sensor
    - Accelerometer/gyro
    - Ultrasonic distance-sensor
  - Abstracts away a lot of ”glue electronics”
  - Often their data can be directly requested in programs
    - Some of them even contain their own microcontroller (!)

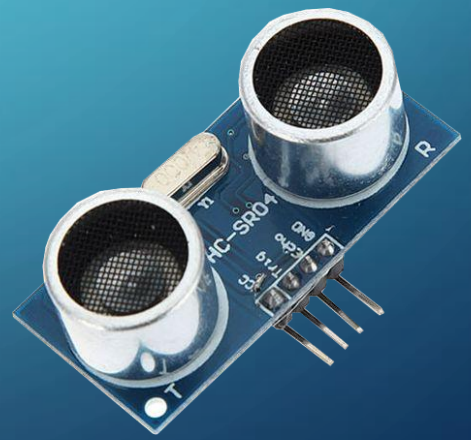
Usually communicates through a standardized protocol:

- I2C
- SPI
- UART

- Although sometimes their own unique protocol





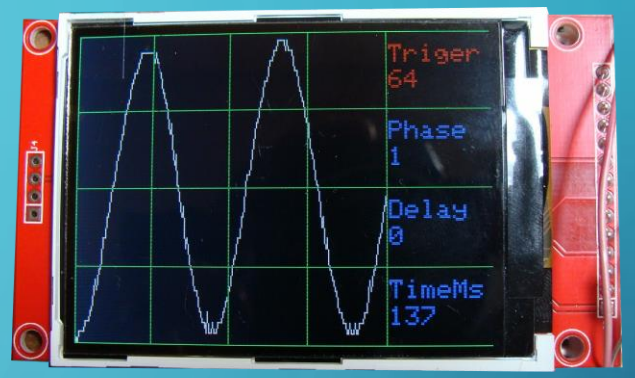


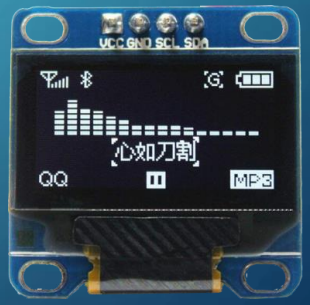
ACTUATORS

- Counterpart to sensors
  - Where sensors act input, actuators act as output
  - They ACT on the outside world
- Most actuators are simple:
- For example:
  - Motors
  - Speakers
  - Linear aktuators
  - Pistons
- Some are a little more complicated
  - Servo motors
  - Stepper motors
- However, most have a directly controlled interface
  - That means; no protocols

OTHER OUTPUTS

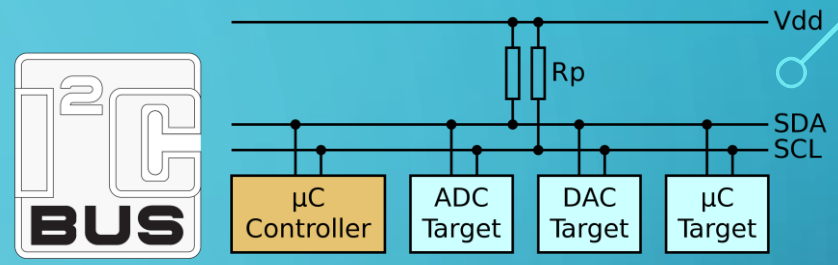
- Outputs do not need to move
- Other outputs are usually indicators or interfaces:
  - LEDs
  - Character Displays
  - OLED / TFT LCD
- Often visual
- Also usally require use of a communication protocol:
  - SPI
  - Parallel Interface
  - I2C

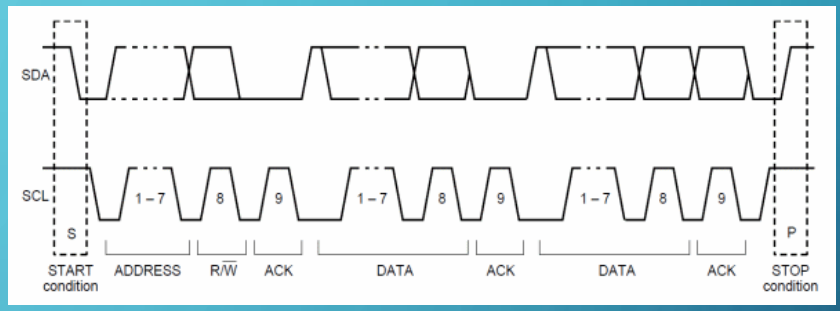




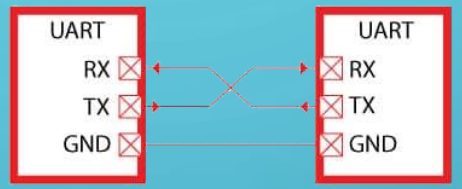
PROTOCOLS

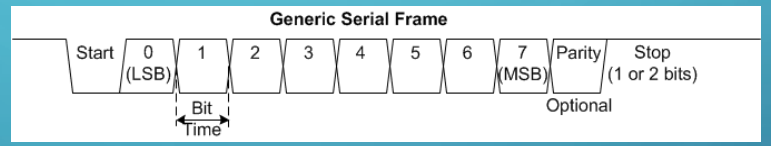
- Protocols cover the interfaces and language spoken between chips
- Usually between a microcontroller like Arduino and a specialized chip
  - E.g. a sensor, complicated actuator, or a display
- The most widespread you should recognize by now:
  - UART
  - I2C
  - SPI
- Arduinos built in libraries usually save us from writing these by hand!
- However, they are always available if necessary



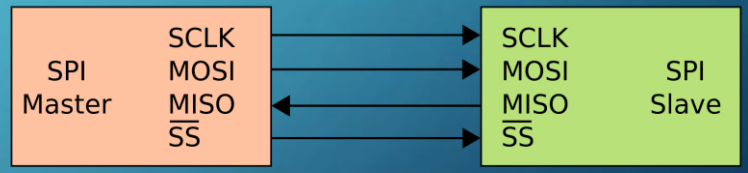


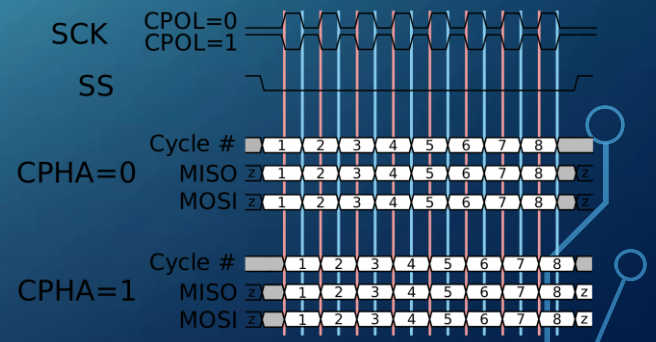
Pictured: I2C connections, and dataframe
URL: https://en.wikipedia.org/wiki/I%C2%B2C





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

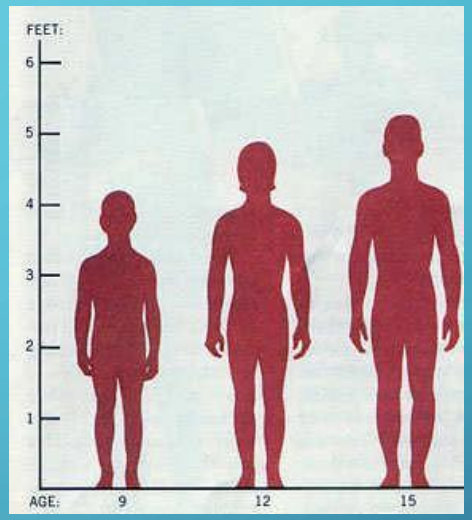
DIGITAL AND ANALOG ELECTRONICS

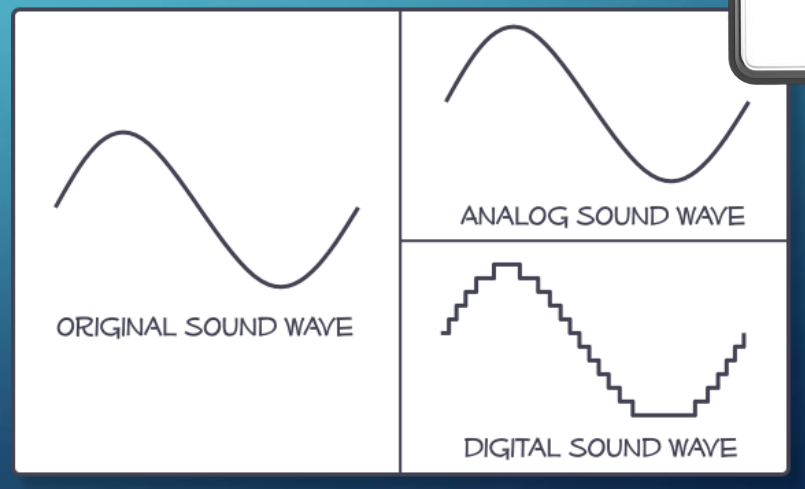
DIFFERENCE BETWEEN DIGITAL AND ANALOG?

- Some phenomena can be put as a ”truth value”, i.e.: (true/false)
  - Is the door open or closed?
  - Is the button pushed or not?
  - Is the coffee ready?
- These values are also known as boolean values

DIFFERENCE BETWEEN DIGITAL AND ANALOG?

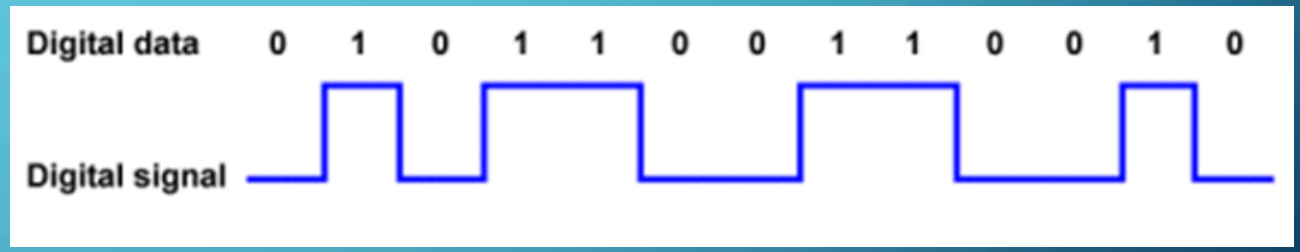
- Most of the world however, is analog:
  - People can have varying heights
  - Time can be measured in very small fractions
  - Within HiFi audio especially, these minuscule measurements are fussed over





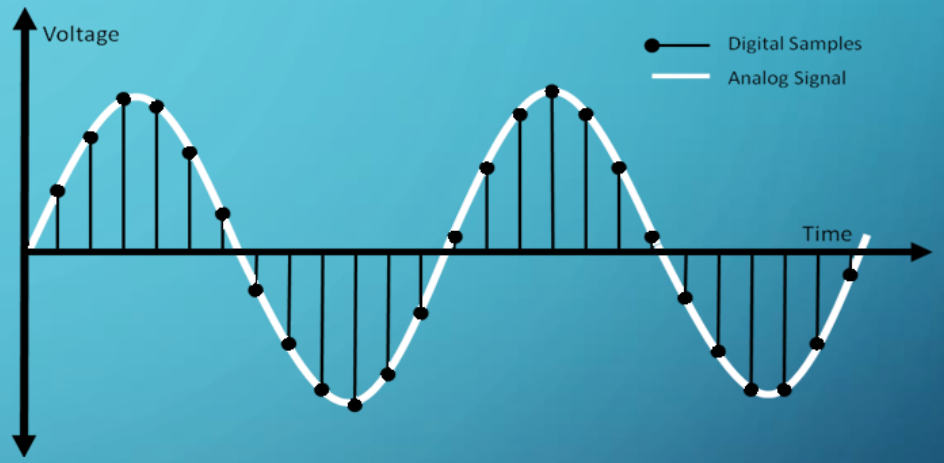
DIGITAL WITHIN ARDUINO

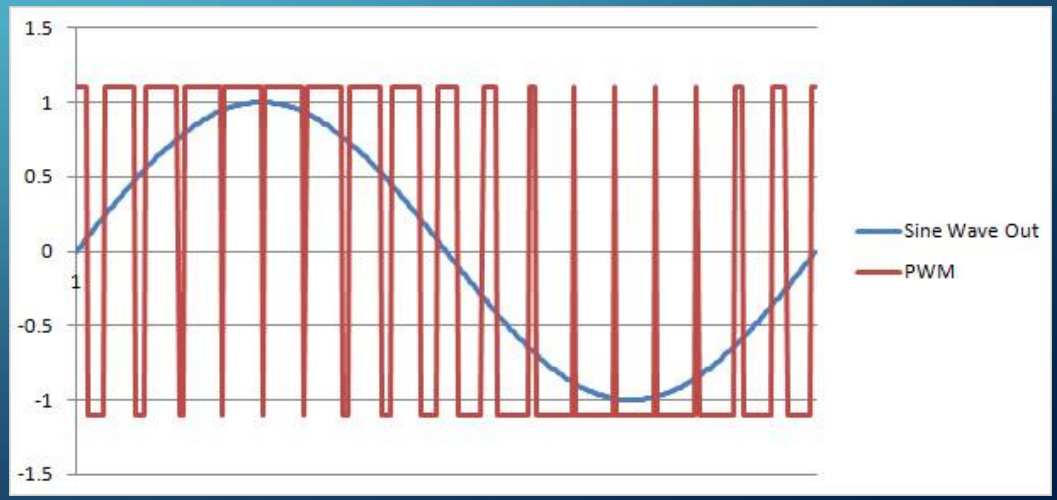
- Most Arduino can ”read” and ”write” digital data
  - An electric signal either 0V or 5V
  - 0V = 0/false/LOW
  - 5V = 1/true/HIGH
- Voltages of either 0V or 5V are represented data-wise on the Arduino as either: 0 or 1
- These values are interpreted from a threshhold, and rounded either up to 1, or down to 0



ANALOG WITHIN ARDUINO

- Arduino can also measurre analog data
  - Done one the analog input pins
  - This is known as ”sampling”
- It can also output a ”semi” analog signal
  - Through a technique known as PWM
  - Quickly turning on and off a digital pin
  - A method also used in class D amplifiers in audio
    - Or voltage conversion (DC to DC)

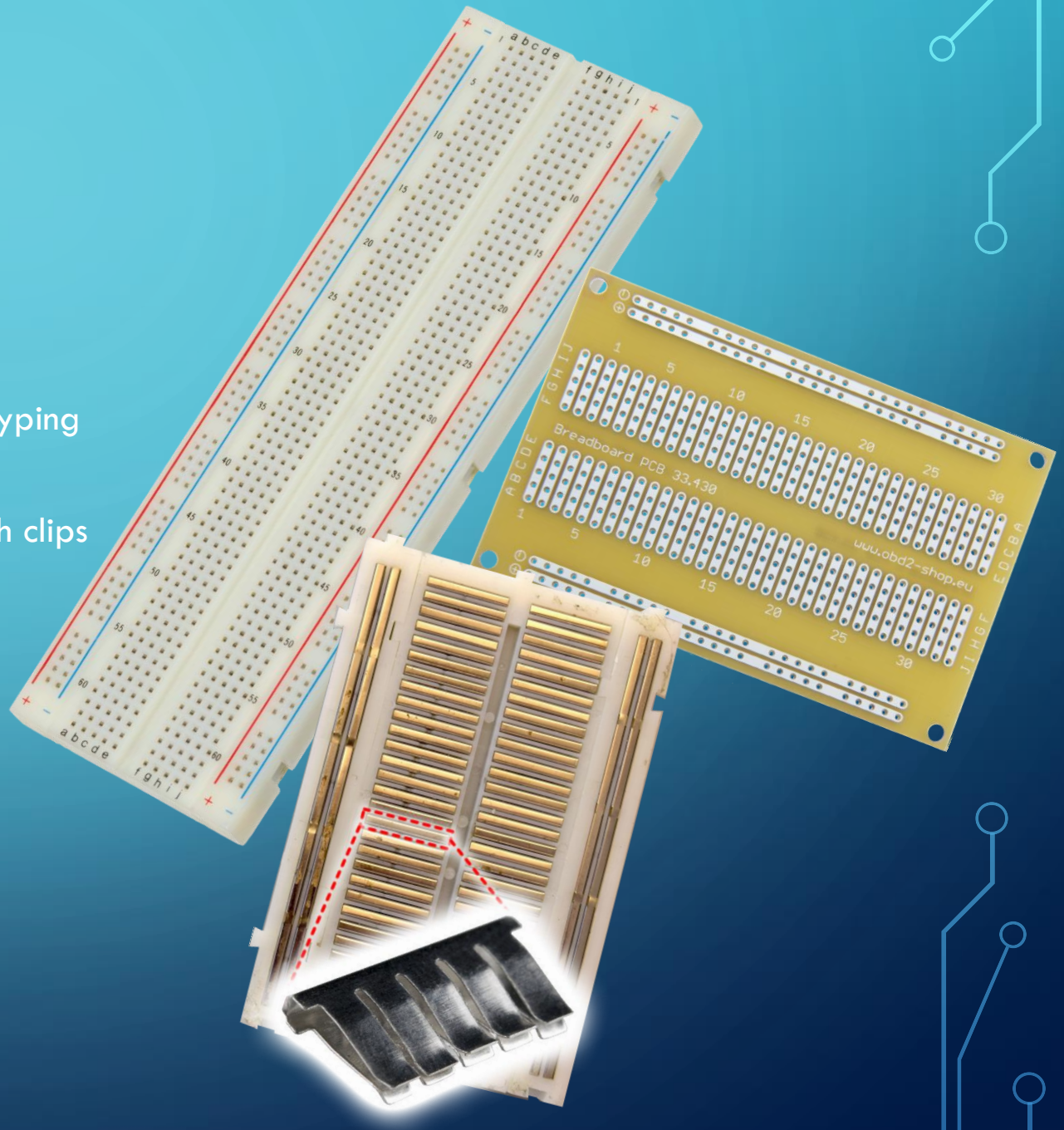




EXAMPLES

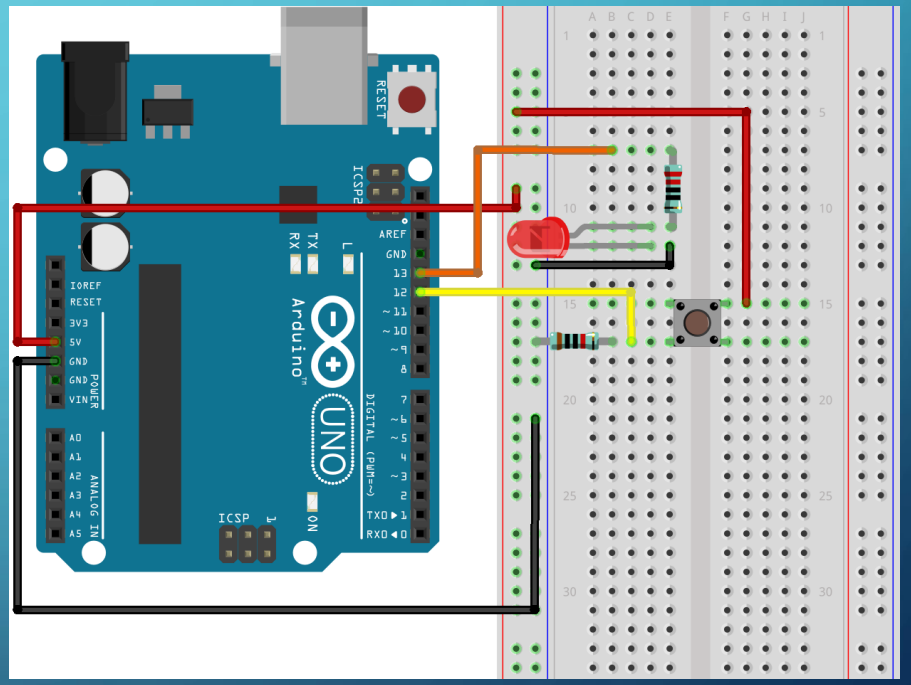
ABOUT BREADBOARDS

- Before we go further, let's look at breadboards
- The solderfree breadboard is made for easily prototyping electronics
- Wires go into the holes, and make connections through clips on the inside
- Usually have lines going vertical
  - Called Bus strips
  - Connects all the way through
- As well as many more going horizontal
  - Called Terminal strips
  - Connect ~5 pins
  - Interrupted in the middle



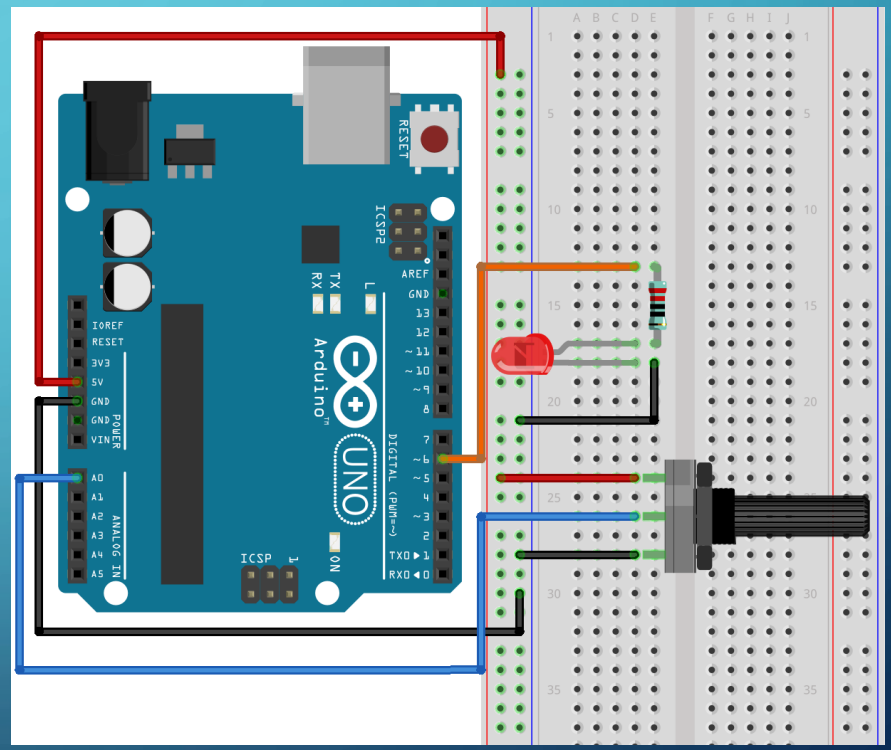
EXAMPLE 2: TOGGLE LED WITH A BUTTON

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



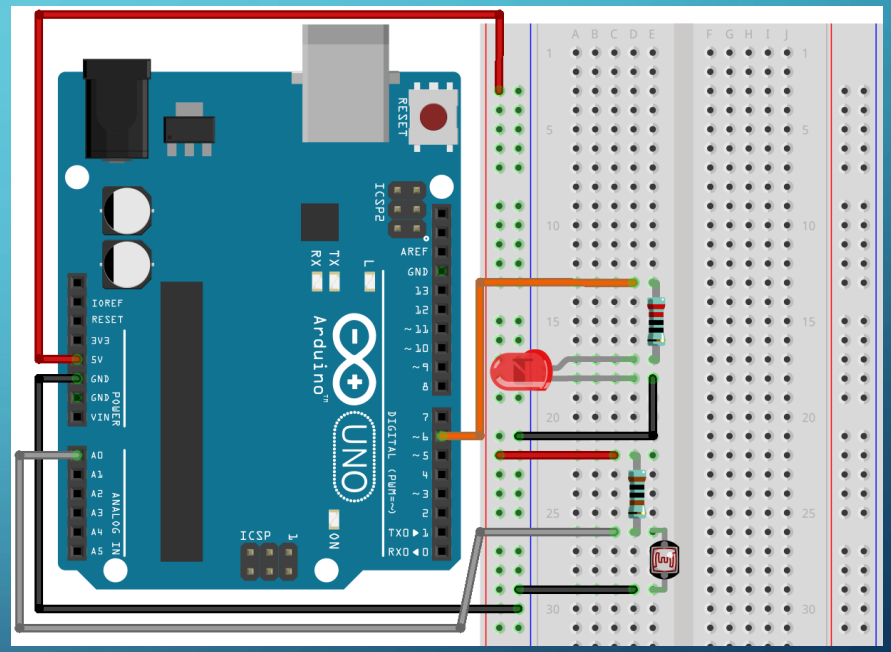
EXAMPLE 3: ANALOG VALUES FROM A POTENTIOMETER

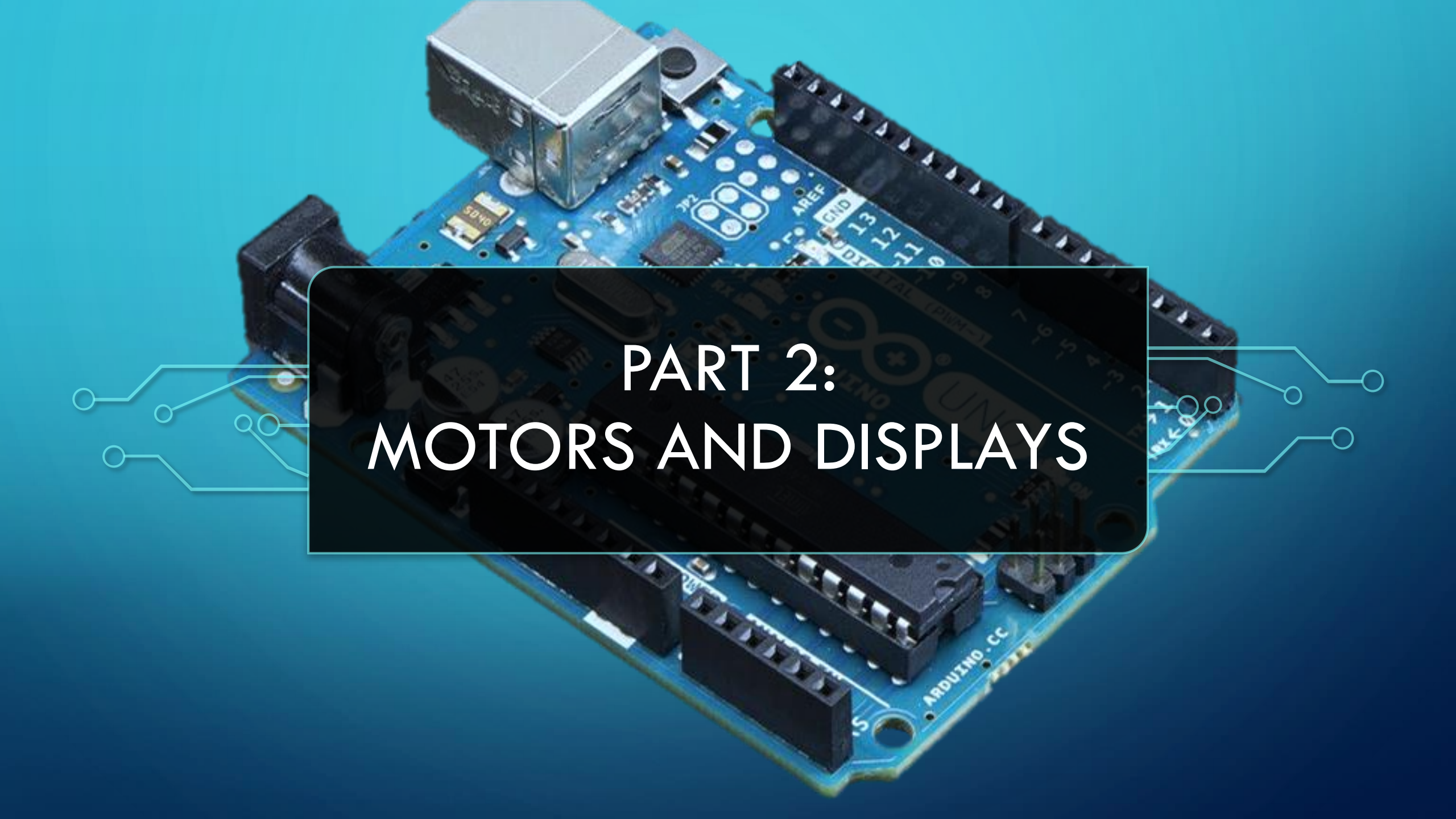
- Onto analog sampling!
- We'll build a new circuit
- We'll also make the Arduino communicate with the PC
- And use PWM ("analog output") to dim the LED



EXAMPLE 4: REPLACE POTENTIOMETER WITH LDR

- We'll replace the potentiometer with an LDR
  - Light Dependent Resistor
- Try to control the LED with it
- Everything that gives an analog value on the input can actually control the LED





PART 2:
MOTORS AND DISPLAYS

MOTORER AND DISPLAYS

Types of Motors	Examples	Types of displays	Examples
• DC motor • Servo motor • Stepper motor	• 1 – Servo motor driving • 2 – Stepper motor driving	• LED • 7-segment • LED-matrix • LCD and OLED	• Dot matrix LCD • TFT LCD

TYPES OF MOTORS

DC MOTORS

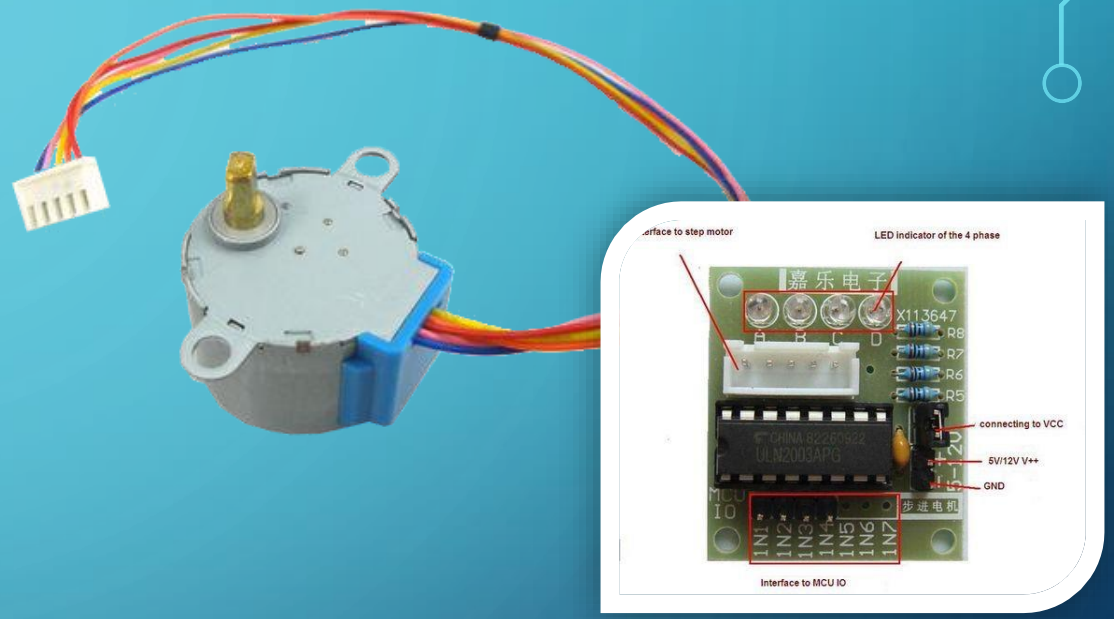
- DC motors are the simplest motors to control
- Only require a current to run
- Can vary speed depending on the current over time
  - Usually through PWM
  - Think ”analog speed”

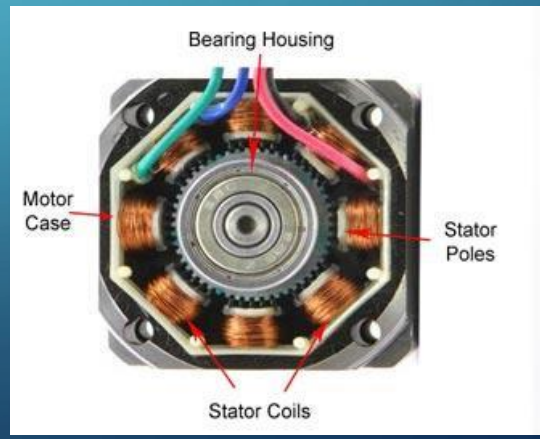
SERVO MOTORS

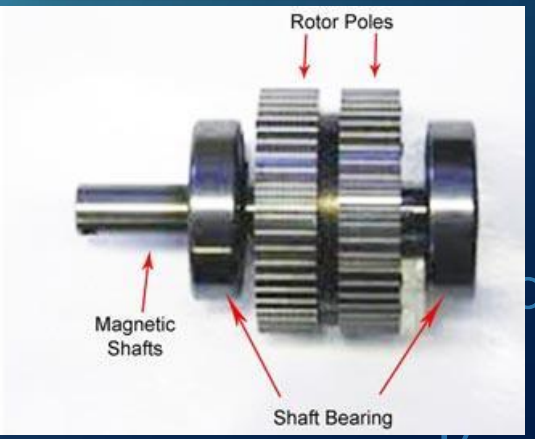
- Servomotorer are geared DC-motors with a control circuit inside
- The rotation feeds back into control curcuit
- Results in movement to set position
- The position is controlled with PWM

STEPPER MOTORS

- Steppermotors contain several coils
- Their magnetization through current determine the motorposition
  - In small increments, thus ”stepper”
- Enables very precise control of position
  - By running a sequence of steps
  - Usually current is supplied through a ”stepper driver” circuit
  - However, no feedback is obtained, so steps mudt be counted



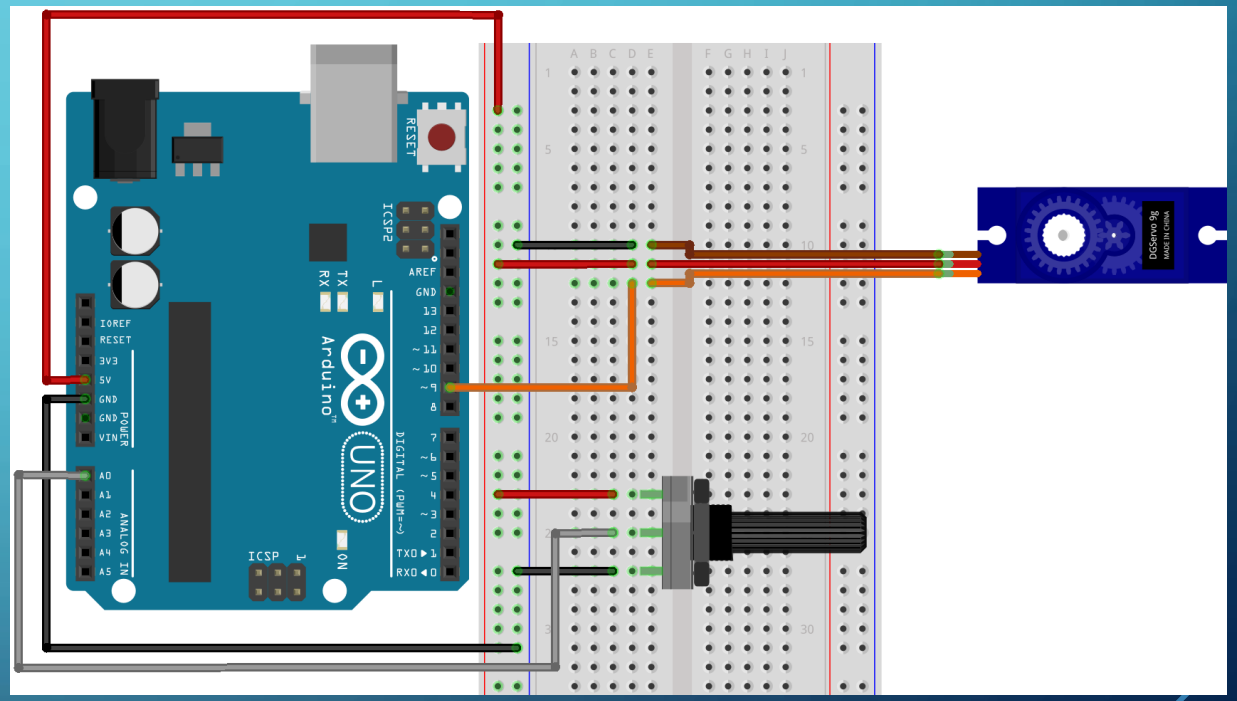




EXAMPLES

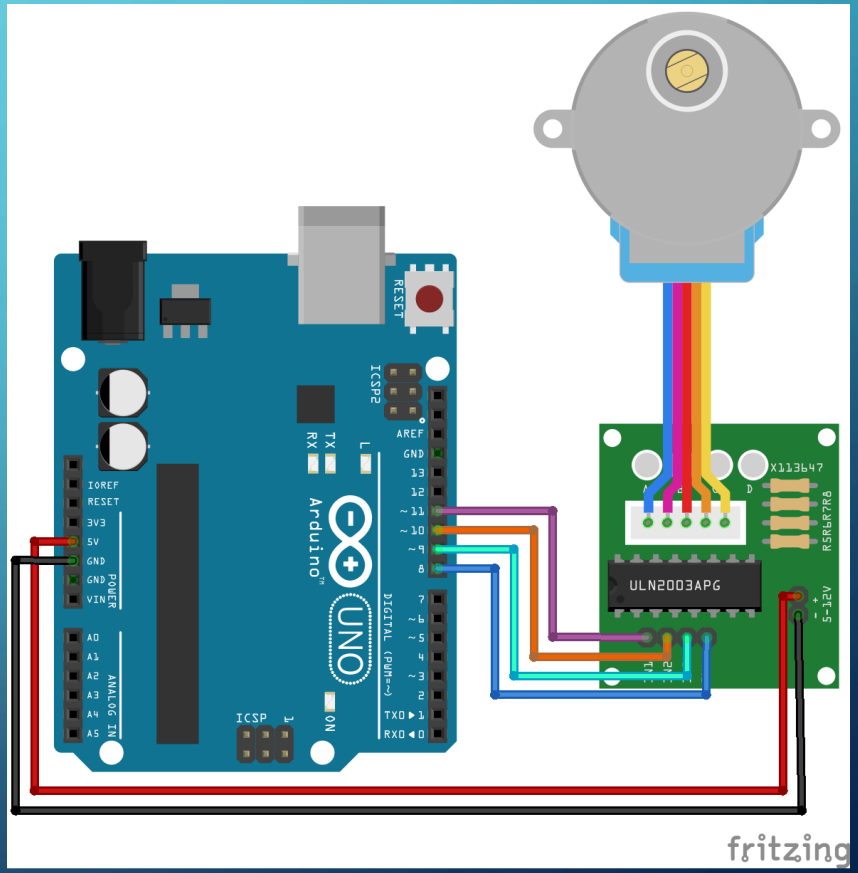
EXAMPLE 1: SERVO DRIVING

- Let's try controlling the servo
- Arduino uses the library Servo.h
- Here ve map the potentiometer values to Servo position



EXAMPLE 2: STEPPER DRIVING

- For this, the breadboard is not required
- But the Dupont cables are
- To drive the stepper we need the externallibrary, CheapStepper.h
- We'll try to make it turn
  - Keep track of, and read out the position



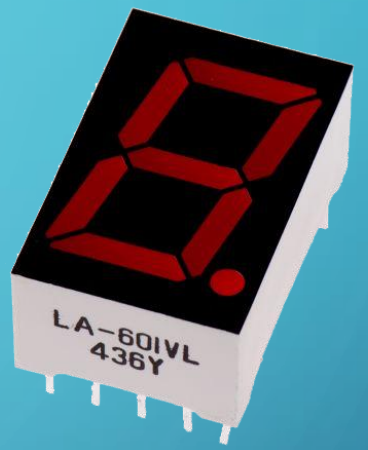
TYPES OF DISPLAYS

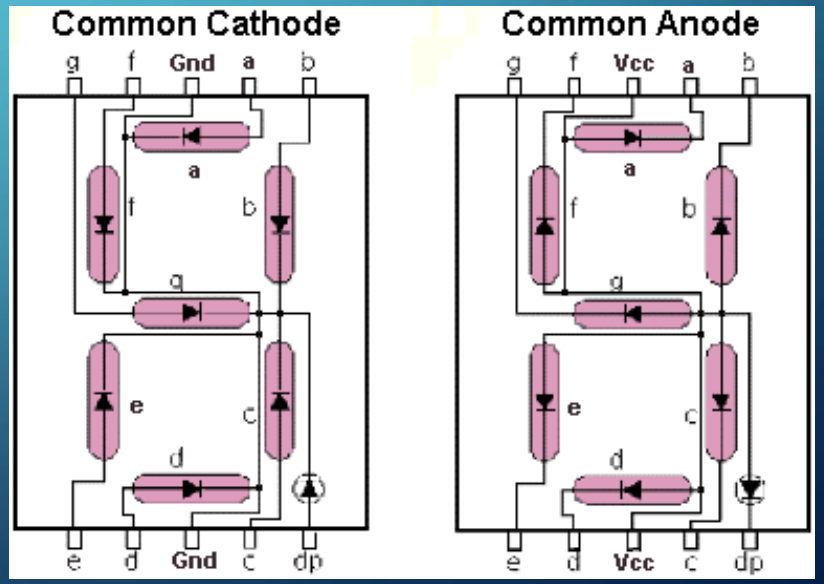
LED

- We know this fellow too well
- Simplest interface
  - Used for indicators, like on/off/standby
- Many displays consist of microscopic versions of them too!
- Some use the for backlighting

7-SEGMENT DISPLAYS

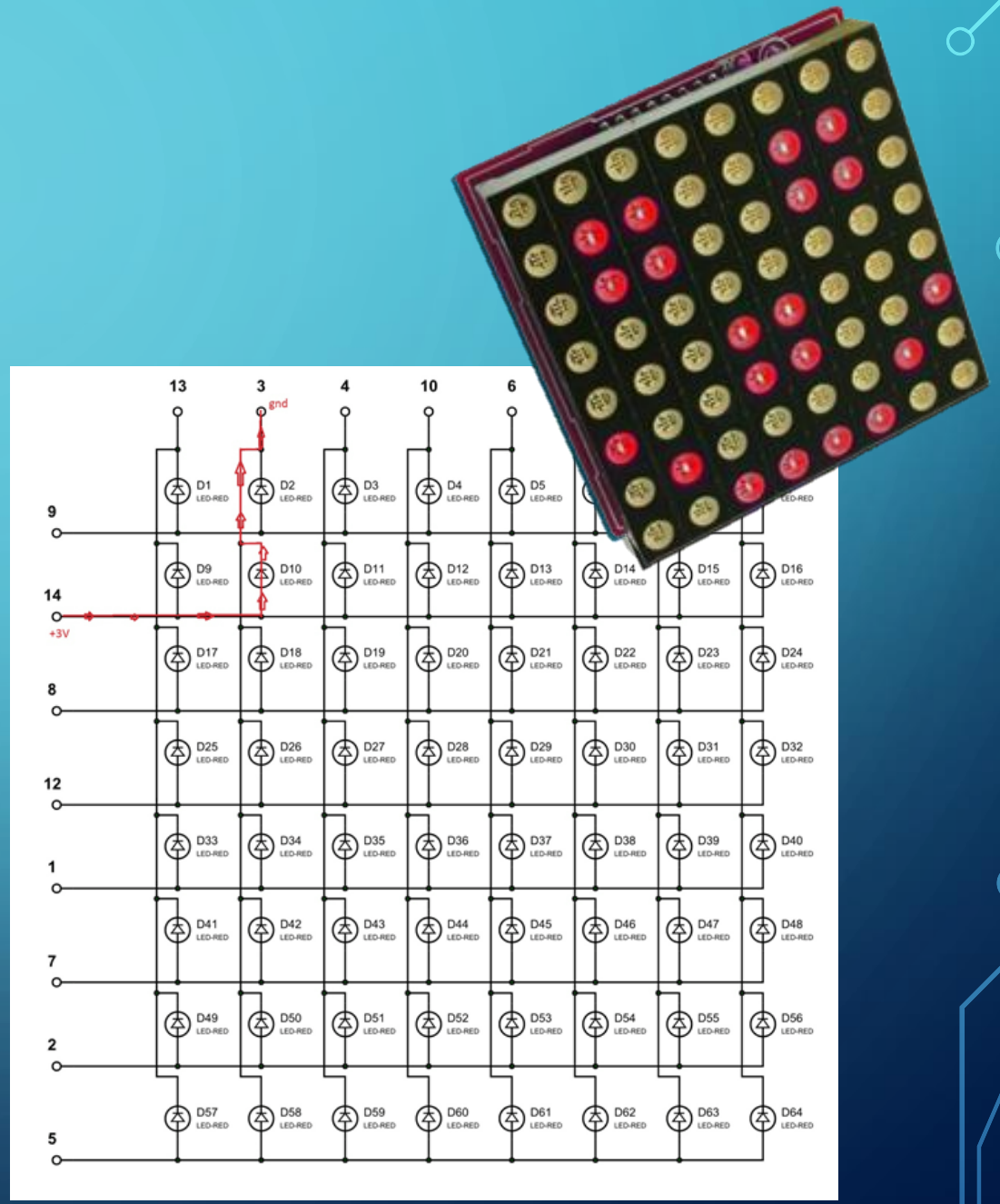
- 7-segment displays are 8 LEDs with common + or -
- Kan be driven directly from Arduinos pins
  - Through resistors, like normal LEDs
- Er with fewer wires through support chips
  - Commonly shiftregisters
  - You have a 74HC595 shift register, but we won't cover it





LED MATRIX

- An LED matrix lives up to its name
- It is literally LEDs in a matrix
  - Every LED is adressable electronically
  - Quite like in matrix notation
- The eye is tricked
  - The LEDs are adressed (individually toggled), in individually faster that we can see
    - Every cycle in 30-60 times per second
  - Called multiplexing



LCD DISPLAYS

- Liquid Crystal Displays
  - Material in a state between liquid and crystallized
  - Opacity is changed by exposing to voltage
- A matrix of dots (Dot matrix)
- Ours is 16x2 characters
  - Every character is in turn 7x5 dots

LCD DISPLAYS

- LCDs like this have their control chips on the back
- Instructed through their own protocol/interface
  - Can take long to program by hand
- Luckily Arduino has a built in library for that!
  - LiquidCrystal.h

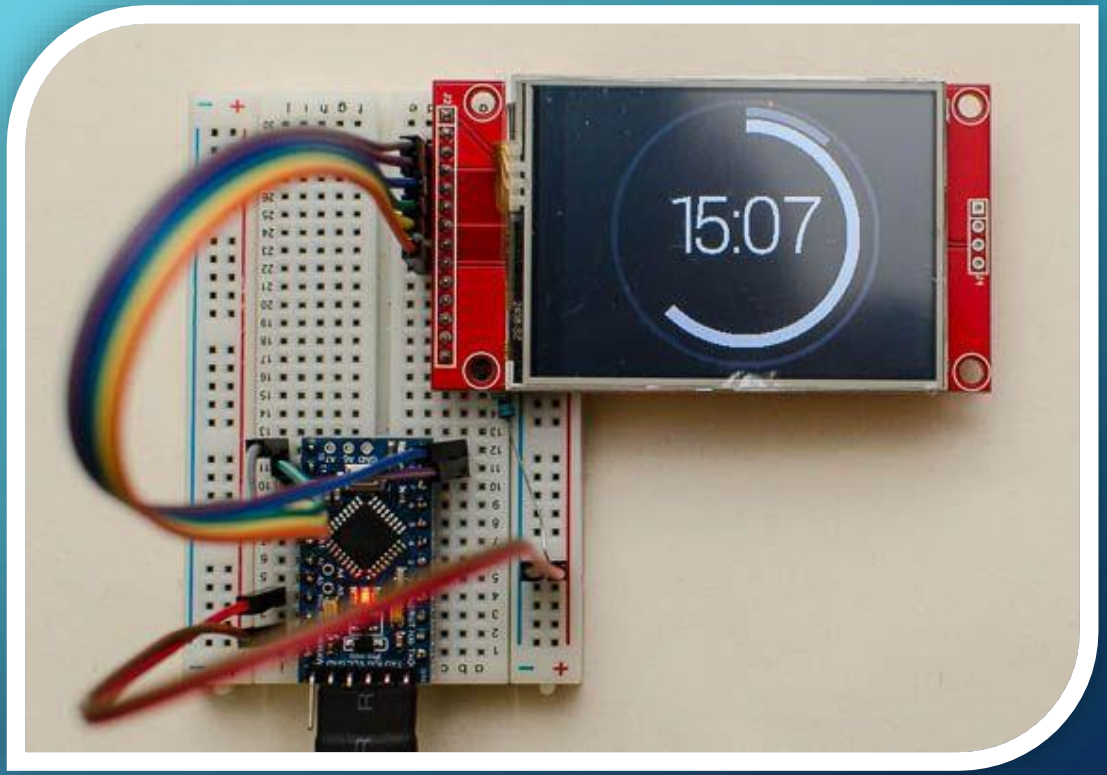
Instruction	Code									
	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0
Clear display	0	0	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	0	0	1	*
Entry mode set	0	0	0	0	0	0	0	1	I/D	S
Display on/off control	0	0	0	0	0	0	1	D	C	B
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*
Function set	0	0	0	0	1	DL	N	F	*	*
Set CGRAM address	0	0	0	1	CGRAM address					
Set DDRAM address	0	0	1	DDRAM address						
Read busy flag & address counter	0	1	BF	CGRAM/DDRAM address						
Write CGRAM or DDRAM	1	0	Write Data							
Read from CG/DDRAM	1	1	Read Data							

Instruction bit names —

I/D - 0 = decrement cursor position, 1 = increment cursor position; **S** - 0 = no dis
shift left, 1 = shift right; **DL** - 0 = 4-bit interface, 1 = 8-bit interface; **N** - 0 = 1/8 or

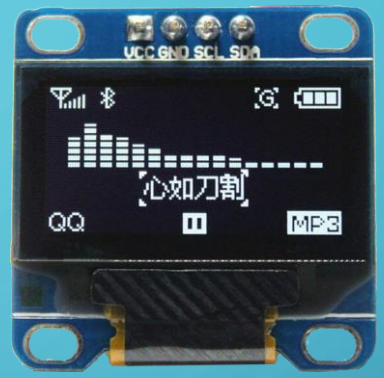
TFT LCD

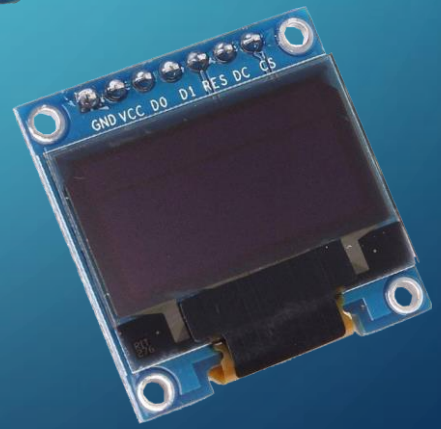
- Thin-Film-Transistor LCDs
- A further development on Dot-matrix LCD
  - Higher resolution
  - Most also have color
- Do require backlighting
- Some even have touch screens
- Controlled through common protocols:
  - SPI
  - Parallel Interface
  - I2C
- Mostly worked with through a library
  - Arduino has several of these



OLED

- Organic LED
- Display built from matrix of LED
- Like LED matrix, kan be turned on and off individually
  - No need for backlight
  - Very high contrast between black and color
- Also uses common protocols:
  - I2C
  - SPI

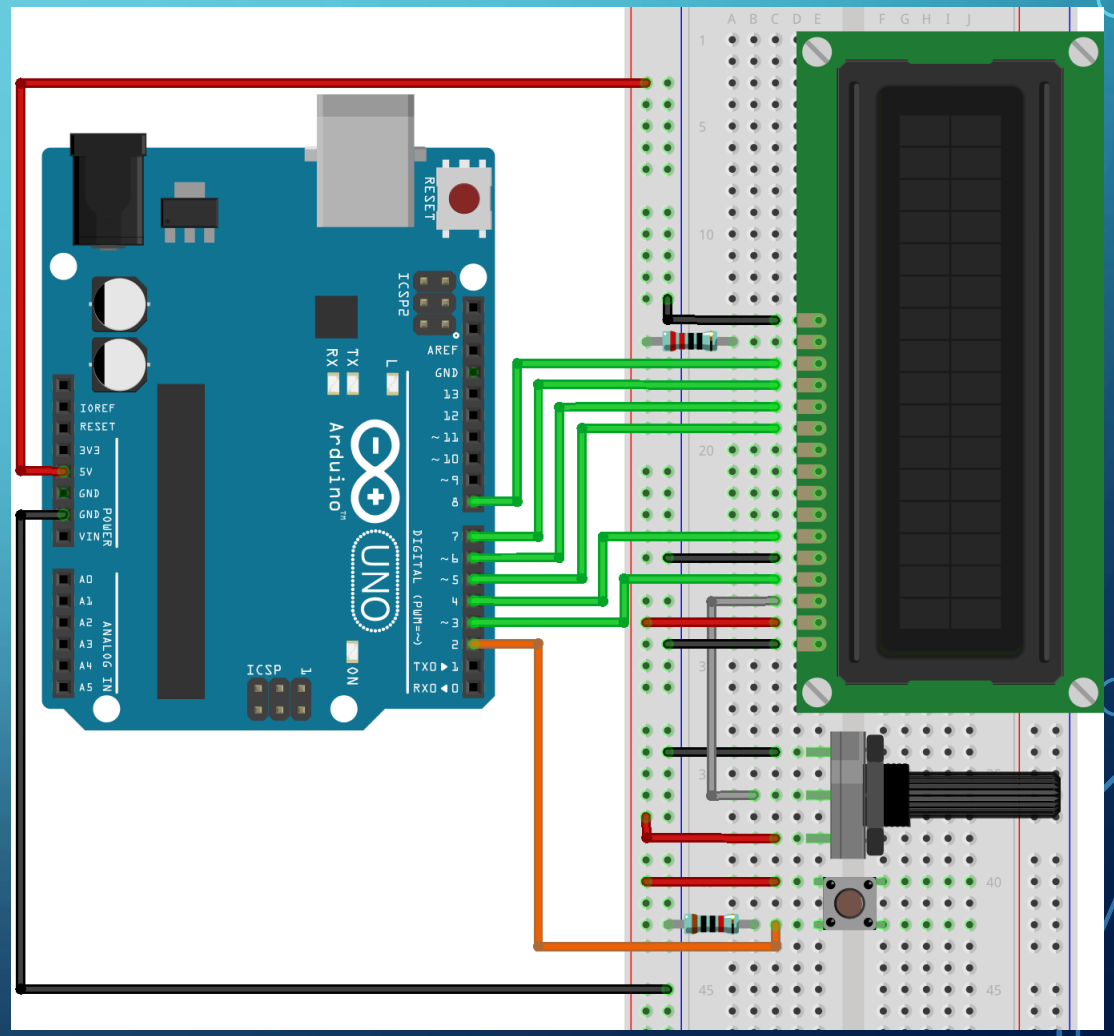




EXAMPLES

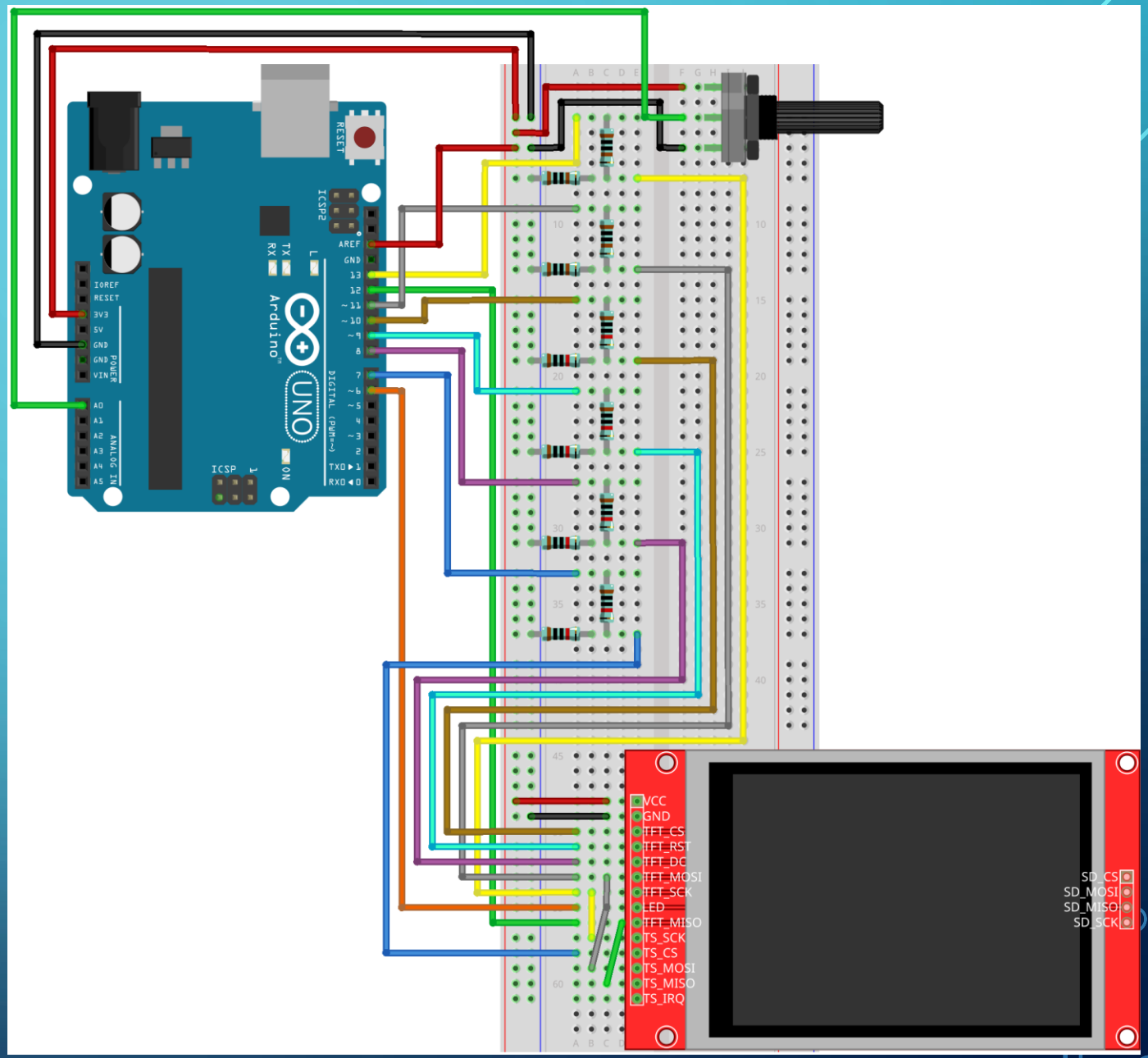
EXAMPLE 1: DOT MATRIX LCD PRINTING

- We need many wires here
- Potentiometer used to control contrast in hardware
- Button used for counter
- Arduino outputs data on the display
- First of all we require the LiquidCrystal.h library



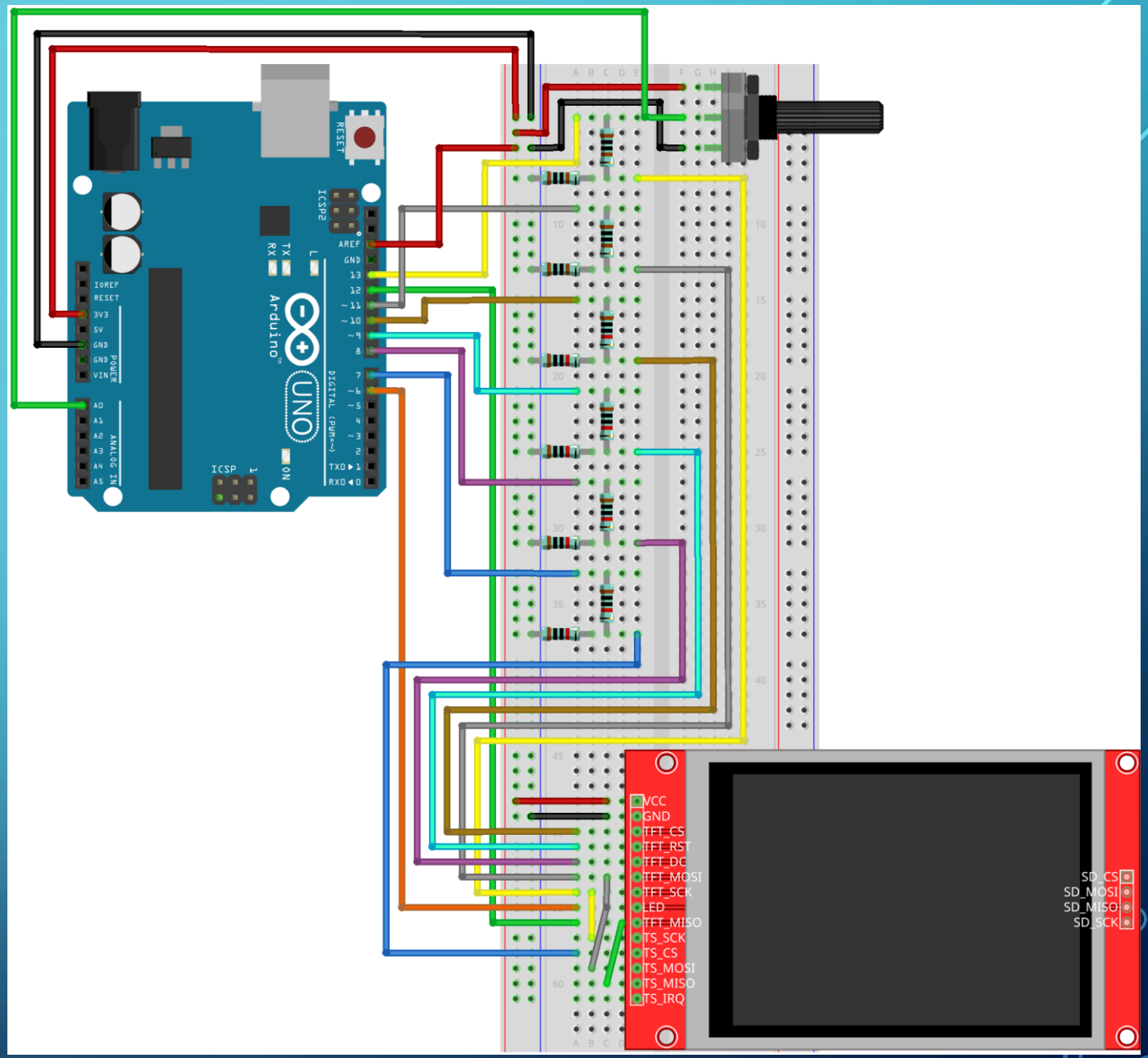
EXAMPLE 2: TFT LCD PRINTING

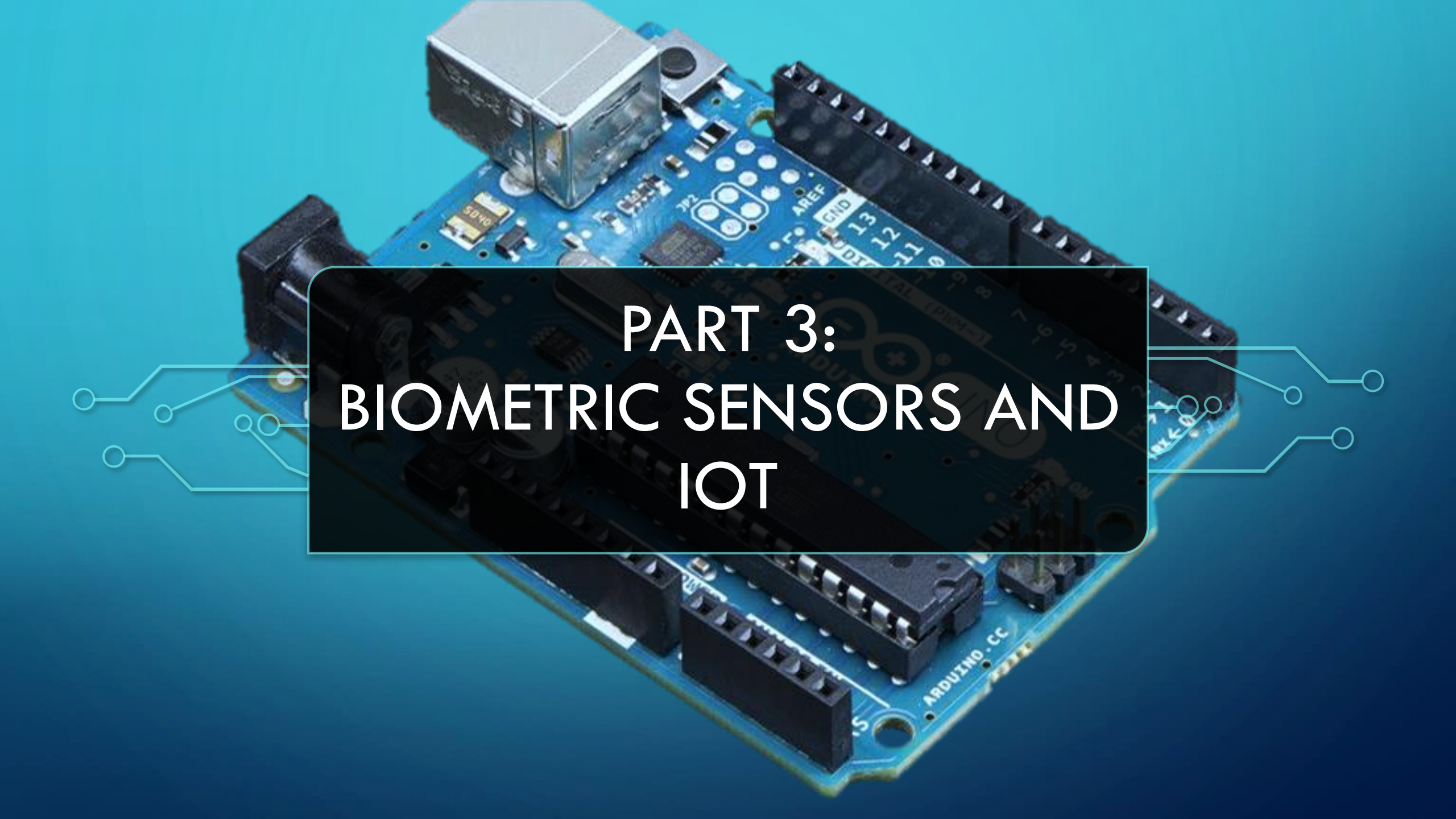
- About as many wires as last example
- We will make the display show a graph
- Potentiometer controls the value
- NOTICE: The TFT uses 3.3V as shown
- We'll need many libraries here:
  - SPI.h
  - Adafruit_GFX.h
  - Adafruit_ILI9341.h



EXAMPLE 3: TFT LCD TOUCH

- We'll keep what we have
- But we wil need another library:
  - XPT2046_Touchscreen.h
- (We'll also need to keep it afterwards)





PART 3: BIOMETRIC SENSORS AND IOT

BIOMETRIC SENSORS AND IOT

Pulse-oximeter

- About the pulse-oximeter
- Example

ESP32-CAM

- ESP32 in general
- Setup of ESP32-CAM / Example

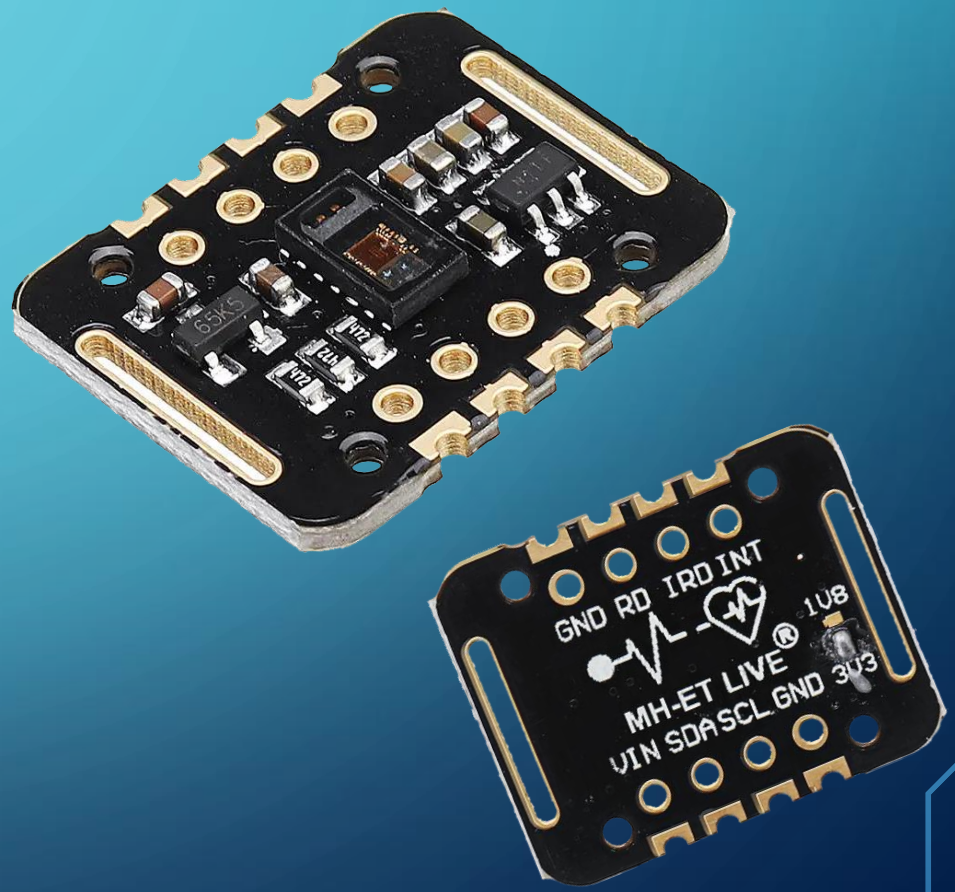
IoT Devices

- IoT gateway demo
- Example

PULSE-OXIMETER

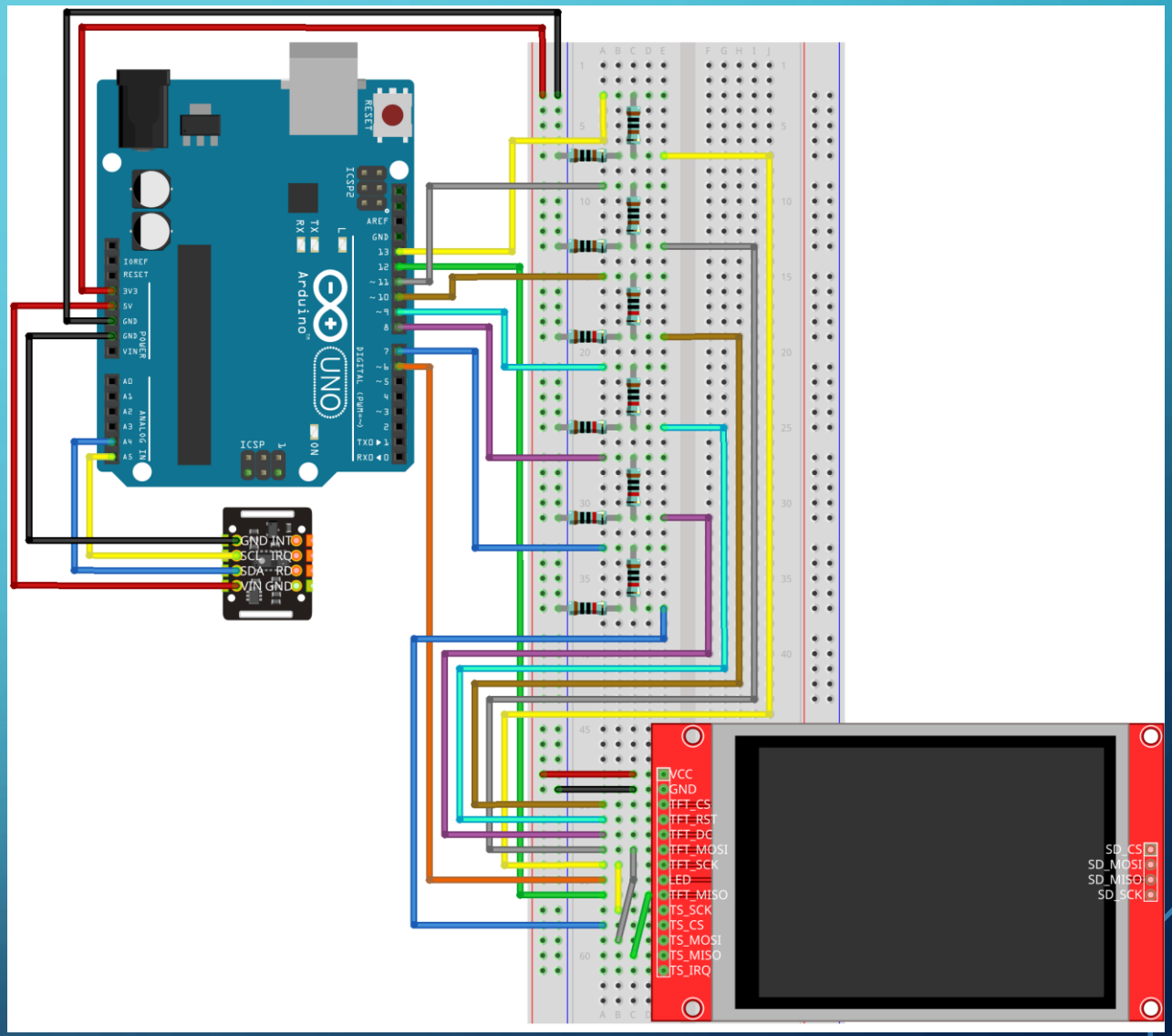
ABOUT THE PULSE-OXIMETER

- MAX30102
  - Consists of an LED that will shine unto your skin
  - Based on the light absorbtion (red or infrared) of hemoglobin, it detects oxygenation levels
  - Can be used for pulse measurement & SpO2.
- Is adressed through the I2C protcol
  - This is also abstracted through a library!



EXAMPLE

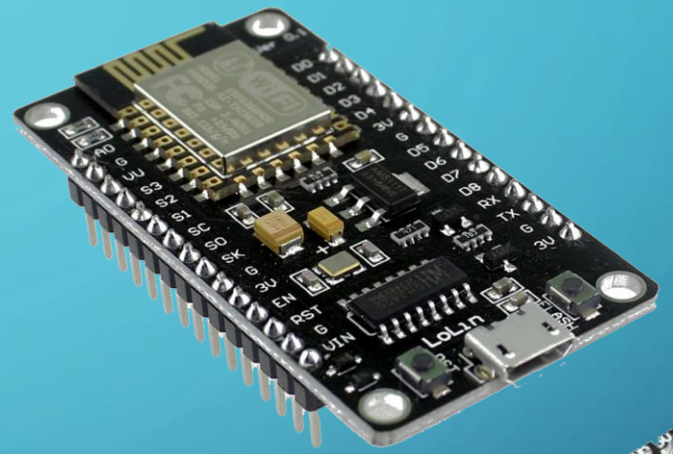
- Simple to connect
- Just use the Dupont cables
- We'll use it with our TFT graph
- The code is a little less simple
- But simply get it from Github

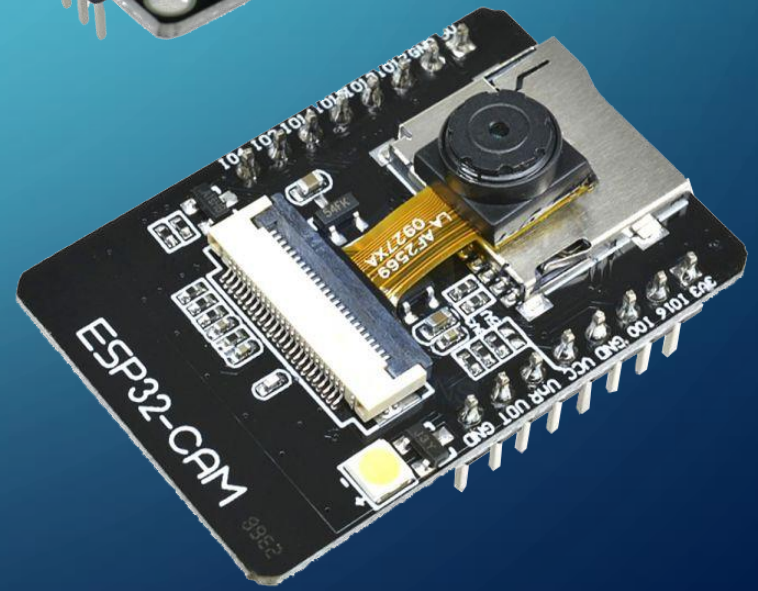


ESP32-CAM

ESP32 IN GENERAL

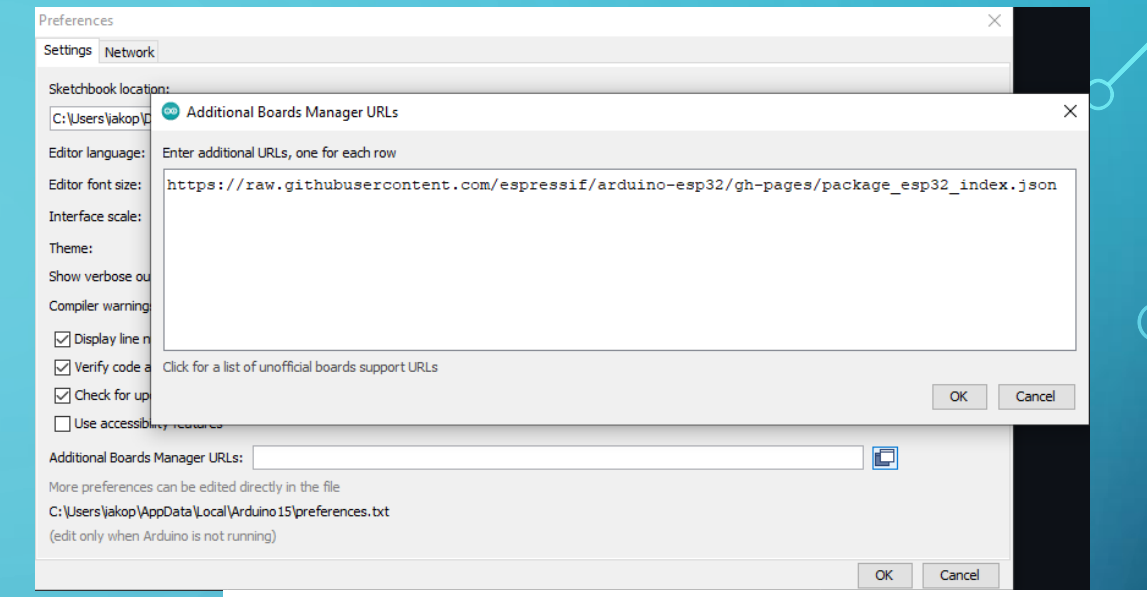
- ESP32 is a further development of ESP8266
  - Microcontroller chips with easy handling of WiFi
  - The go to platform for hobbyists and prototypers
  - Broadly supported by community
- Can be programmed from the Arduino IDE and with Arduino functions and language!
- ESP32-CAM is exactly what it says:
  - It's an ESP32 with a camera on it
  - It also has an SD card slot
  - Is used for monitoring and live image processing
    - It DOES have som creepy possible applications

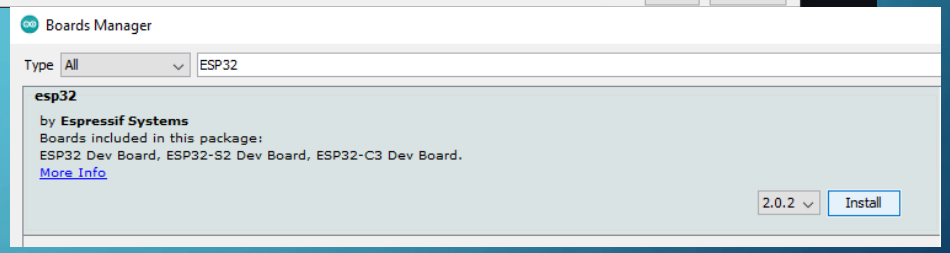


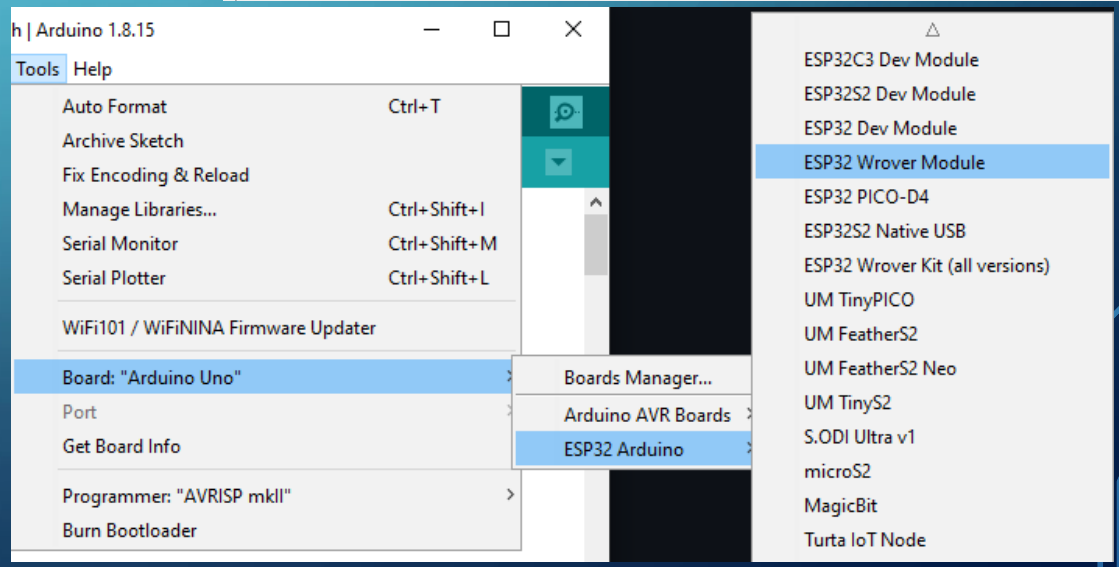


SETUP OF ESP32-CAM / EXAMPLE

- We need a board definition
  - Since ESP32-CAM is second party hardware
- Type in the URL for the board definitions
  - Then install esp32 from Espressif Systems
- It's added through the Board Manager
  - Tools -> Board -> Boards Manager
- Finally select ESP32 Wrover Kit from the list

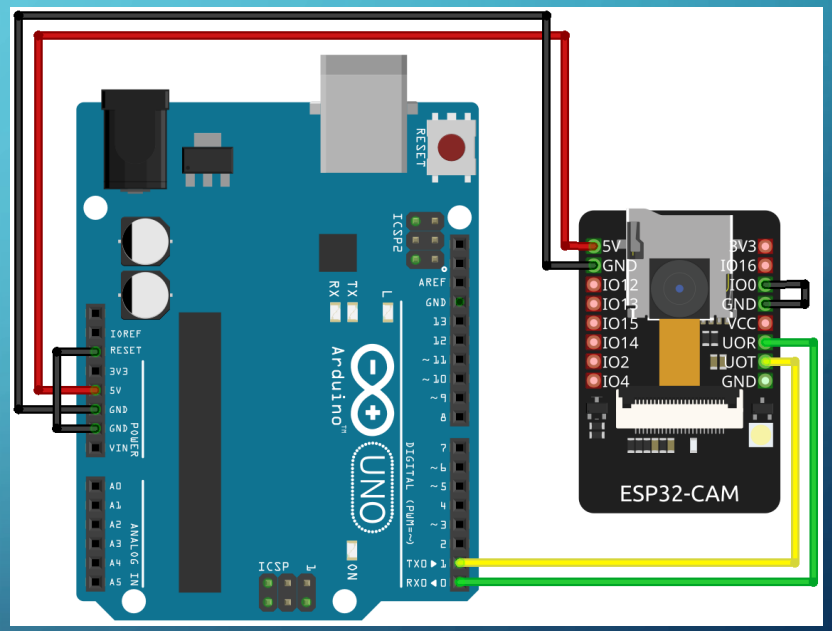






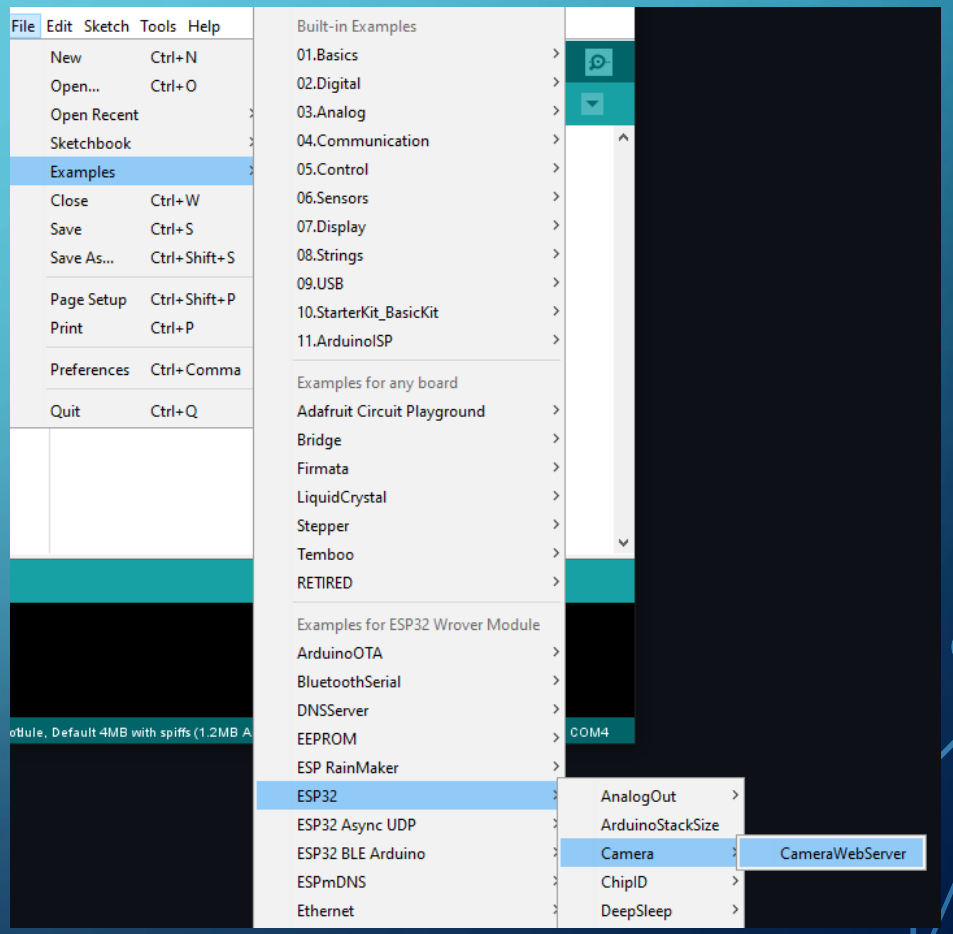
SETUP OF ESP32-CAM / EXAMPLE

- In order to program ESP32-CAM, we need a serial programmer
- Arduino has one built in
- We need to apply a small hack
  - To hi-jack the Arduino serial programmer
- The circuit hack is as shown
  - With the Arduino in perpetual reset, the ESP leeches the programmer pins
  - The IO 0 pin is set to GND to enable programming



SETUP OF ESP32-CAM / EXAMPLE

- The code is very complicated
- In interest of time and sanity we will demo a pre-made example
- It tests the camera over a website you can access from your PC!



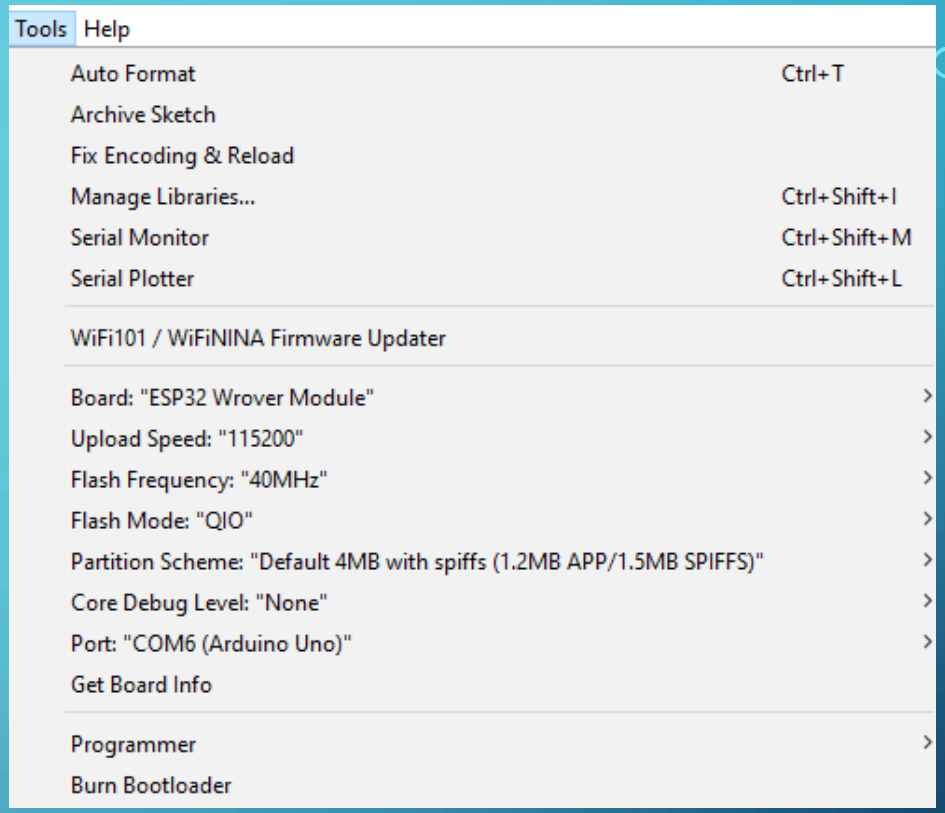
SETUP OF ESP32-CAM / EXAMPLE

- We'll need to change line 10 through 24 to this:

```
10 // Select camera model
11 //#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
12 //#define CAMERA_MODEL_ESP_EYE // Has PSRAM
13 //#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
14 //#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
15 //#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
16 //#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
17 //#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
18 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
19 //#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
20
21 #include "camera_pins.h"
22
23 const char* ssid = "workshop";
24 const char* password = "workshop";
```

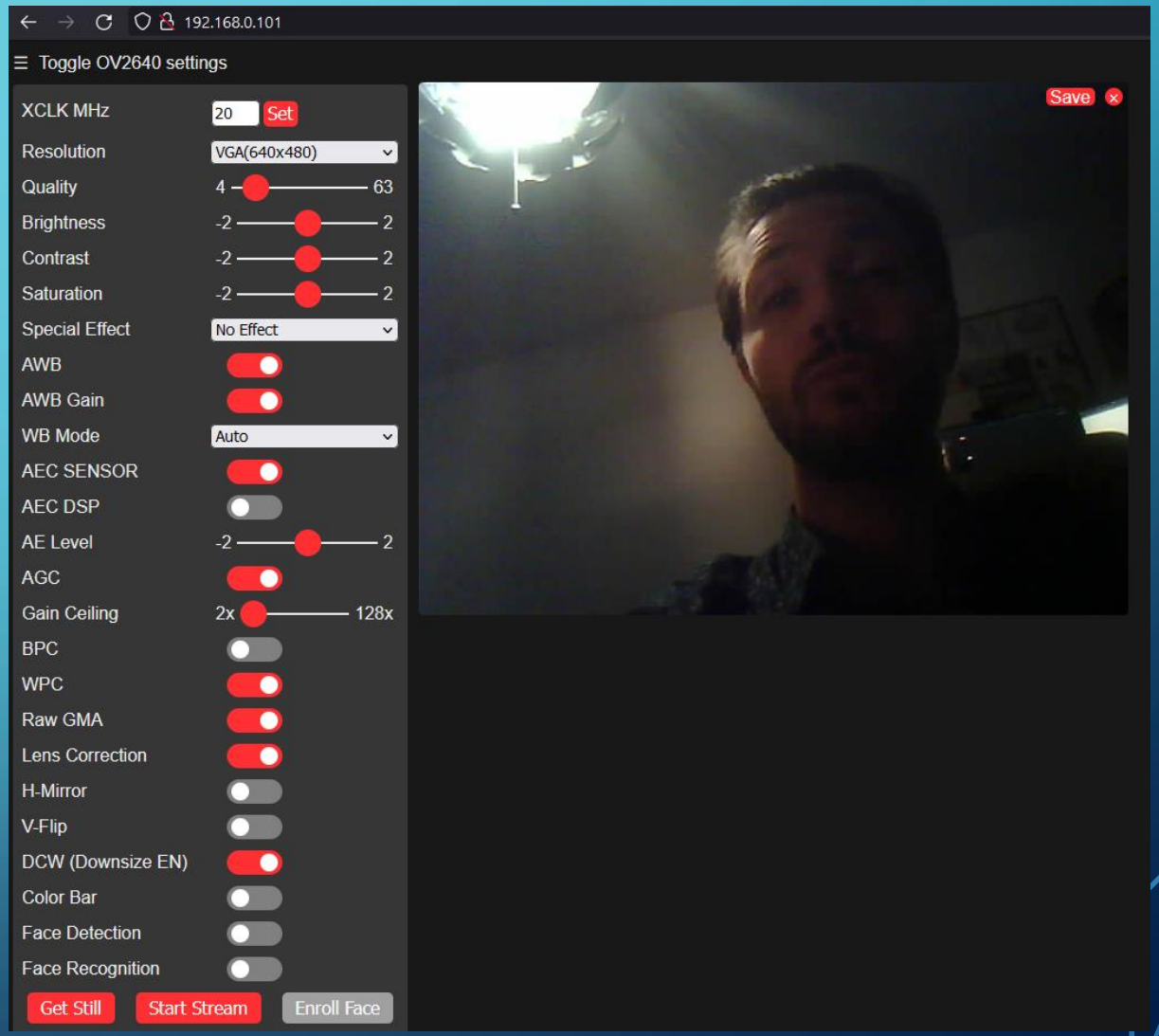
SETUP OF ESP32-CAM / EXAMPLE

- The settings we need to program it are as shown
- When uploading, the boot button must be pressed to accept the code
- Then the ESP32-CAM is flashed
  - Remove the connection between GND and IO 0
  - Press RST button
  - You'll see something like the serial output shown



SETUP OF ESP32-CAM / EXAMPLE

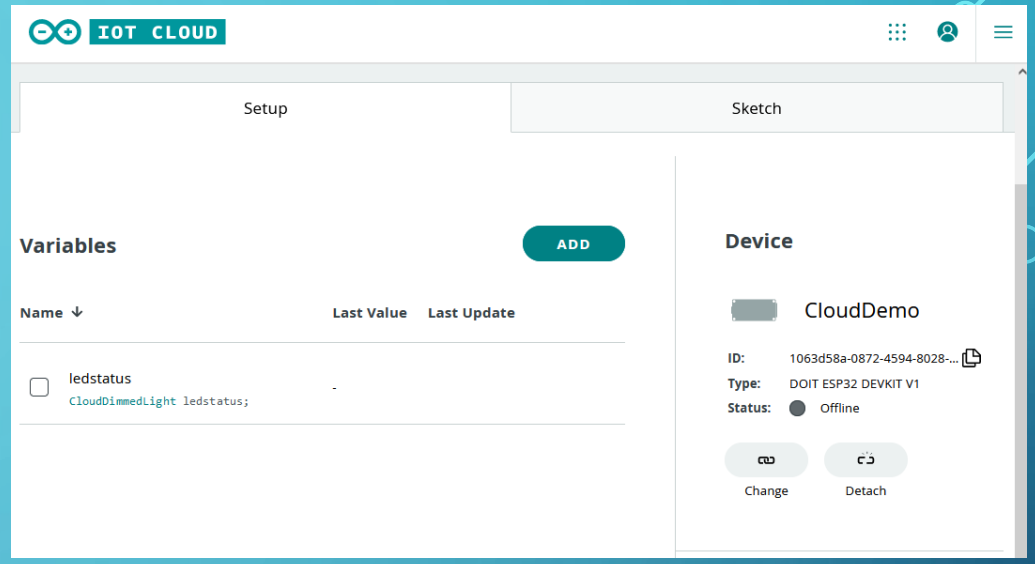
- Go to the address

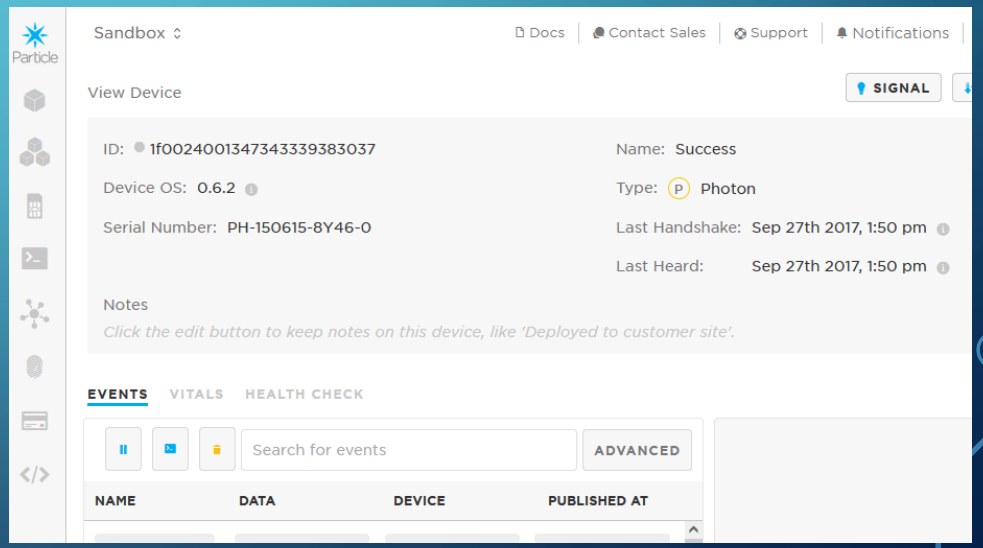


IOT DEVICES

IOT GATEWAY DEMO

- There are several ways to do IoT (Internet of Things) prototyping, the approachable ones
  - Arduino has an official offering
  - Particle has a platform
  - ARM has one too
  - And the classical providers can be set up for IoT





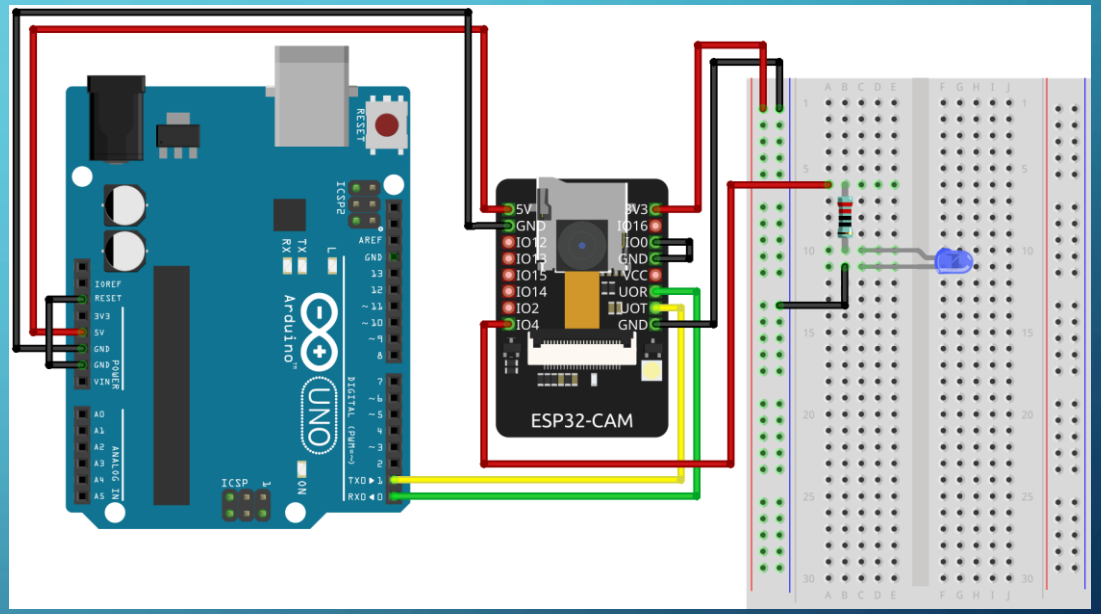
IOT GATEWAY DEMO

- We'll look at a more approachable one!
- Mozilla has a project called WebThings
- The Gateway can easily be self-served
  - And has relatively easy to use Arduino Libraries
- Only drawback is that it is primarily for Smart home applications
- I'll demonstrate the interface!

https://iot.mozilla.org/gateway/

EXAMPLE: IOT DEVICE

- We'll connect the ESP32-CAM to a simple LED
- Come full circle:
  - We'll toggle the LED
  - But over the internet!
- Use the WiFi
  - SSID: workshop
  - Password: workshop
- When we have programmed, note the ip-address
- We'll connect it to the gateway through this



THAT'S ALL FOLKS

- This was the last of the theory and examples
- So how do we go about using this in practice?
- When thinking about a possible idea
  - Simply search some of the constituent parts:
  - E.g.
    - "co2 sensor arduino"
    - "ekg arduino"
    - "blood pressure arduino"
    - "wearable arduino"
  - Often somebody has done the work for the simple implementation
  - This should make the prototype fast
- Usually an article or post will refer to a library
- Read the library documentation
- If all else fails, contact electronics suppliers, eg. Digikey, Arrow, RS, Mouser, etc.
  - If you reach out, sales engineers are happy to sell, and that includes advice on how to use their products.

RESOURCES

- A list of resources you can use:
  - Slides, examples, etc.
    - https://github.com/iakop/BioMedical2022
  - Arduino IDE Download
    - https://www.arduino.cc/en/software
  - Arduino Language Reference
    - https://www.arduino.cc/reference/en/
  - Arduino ESP32
    - https://github.com/espressif/arduino-esp32