Agenda

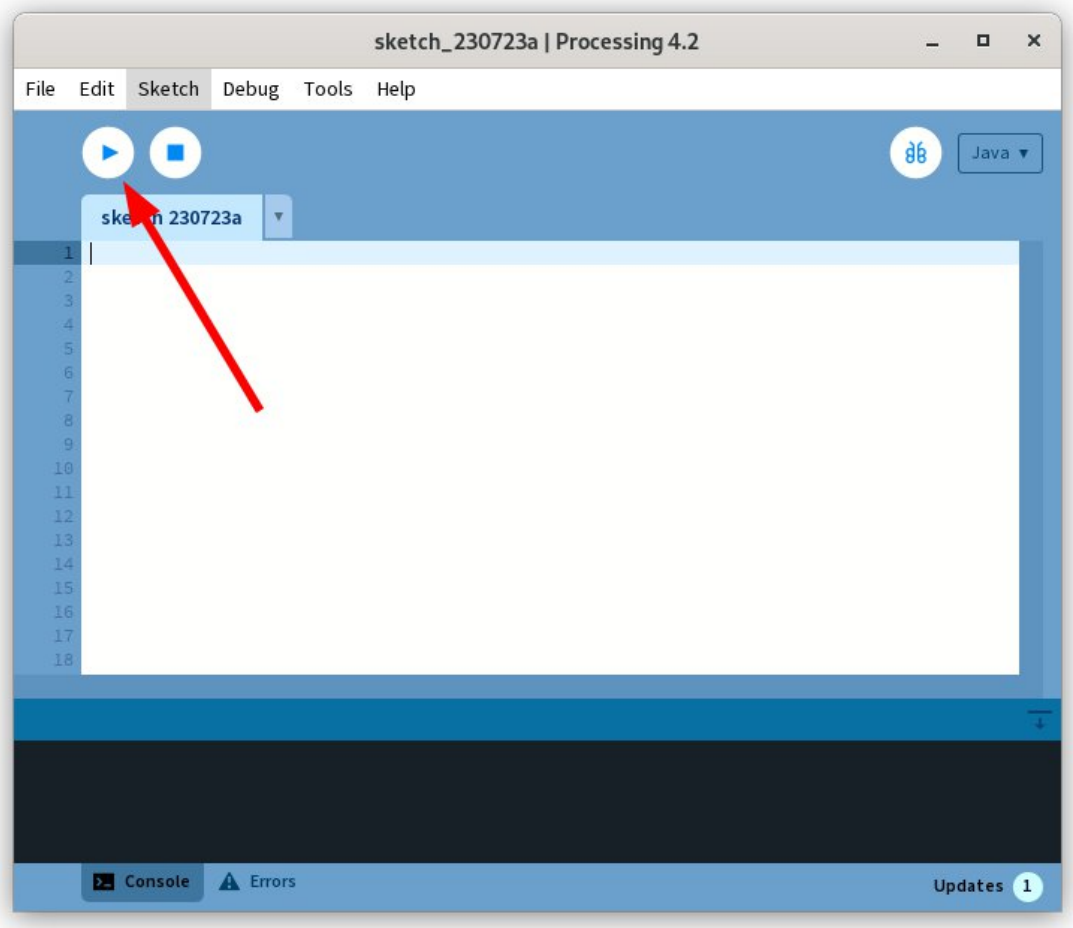
Day 1	Processing, the coding side of Arduino: Functions and Variables
	Processing, the coding side of Arduino: Flow control
Day 2	Arduino: Serial communication between board and PC, I/O ports
	Arduino: Digital Sensors and Digital Actuators
Day 3	Arduino: Analog Sensors
	Arduino: Analog Actuators
Day 4	Arduino: Protocol communication with sensors
	Arduino: Actuator control based on sensor feedback
Day 5	LoRa: Point to point communication
	LoRaWAN: Gateway and Server

Processing
the coding side of Arduino

IDE

Processing IDE

- Setup
- IDE Definition
- Just click RUN
- What is that?



So, what is Processing?

- CAD?
- Let’s write something
- Look at the colors: syntax!
- IDE : Text editor + Compiler + Debugger
- What’s the result?
  - A software application executable by anyone
  - Available exportation for Windows – MacOS - Linux – Android

IDE

- Introduction to the environment
  - An IDE is a text editor? Also. But not only.
  - It helps users with syntax and algorithms, error detection, warning and debug
  - Some quick examples:

```
background(<red>, <green>, <blue>);
background(<grey>);
//background(<grey>);
/*
background(<grey>);
*/
```

  - This was the very first function

Processing
the coding side of Arduino

Functions

Functions pt.1

- We distinguish them by color in the IDE
- They perform a job or return data
- They may require parameters to be executed

Functions pt.2

- The order is relevant
  - If multiple functions have the same “target”, only the latest is relevant
- Just a couple of examples:
  - size(<width>, <height>);
  - fullScreen();
- The same function could require different amount of parameters
  - background(<gray>)
  - background(<red>,<green>,<blue>)

Functions pt.3

Is there a list of functions? Let’s have a look at *Reference*.

Reference

https://processing.org/reference/

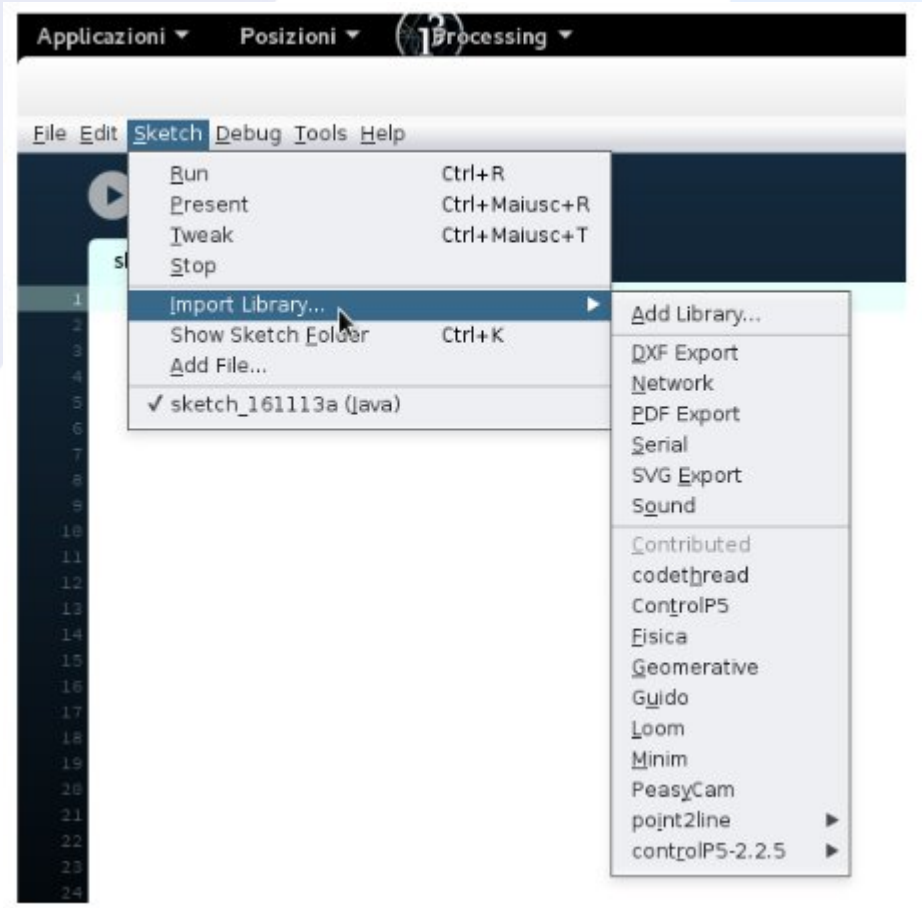
- Structure
- **Environment**
- Data
- Control
- Shape
- **Input**
- **Output**
- Transform
- Lights
- Camera
- Color
- Image
- Rendering
- Typography
- **Math**
- Constants

Core Libraries
https://processing.org/reference/libraries

- DXF export
- Hardware I/O
- Network
- PDF Export
- Serial
- Sound
- SVG Export
- Video

Contributed libraries

https://processing.org/reference/libraries



Sum up

- There are functions that perform many different tasks
- Some functions are already available by default
- Some are available within libraries, organized by topic, that user has to import into program
- Lots of other functions are available on the web, easily installable
- Many functions → Many features → Many possibilities

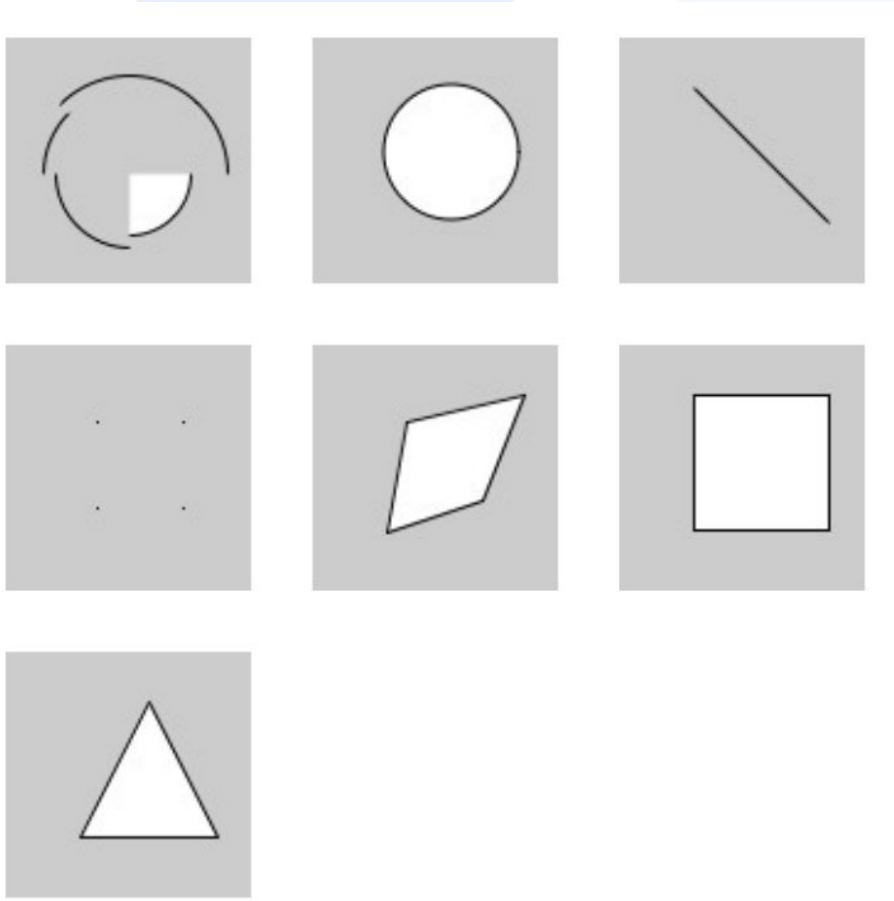
Let’s get started

- Have a look at graphical functions

- Our target is not to create a layout, but understanding the basics of coding

2D Primitives

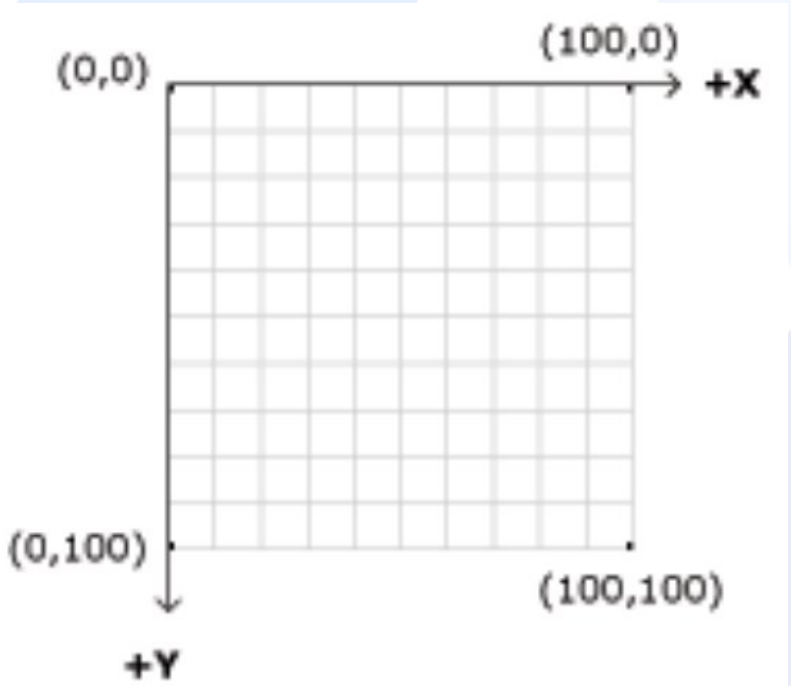
- arc()
- ellipse()
- line()
- point()
- quad()
- rect()
- triangle()
- beginShape()
  vertex()
  endShape()



point(x,y);

- size(400,400);
  point(100,100);

- Axis directions
- Size of lines and points
- By default strokeWeight(1);
- strokeWeight(10);
- Save project
- Auto_format: Ctrl+T



line(x1, y1, x2, y2);

- size(x,y);
  line(0, 0, 300,300);
- Increase strokeWeight
- Size of lines and points
- Change color
- stroke(red, green, blue);
- stroke(gray);
- Tip: Auto_format with *Ctrl+t*

rect(x, y, width, height);

- size(400,400);
- rect(50,100,200,50);

- stroke(R,G,B);
- rect(x,y,b,h);

- noStroke();
- rect(x,y,b,h);

ellipse(x, y, width, height);

- size(400,400);
- rect(50,100,200,50);

- stroke(R,G,B);
- rect(x,y,b,h);

- noStroke();
- rect(x,y,b,h);

Test#1

Draw a picture with a circle in the middle.

Draw a clearly visible point in the middle of the screen.

Also insert the diagonals of the view: don’t overlap the circumference!

Do not use fullScreen() but size().

Size parameters as desired, but less than (300,300).

Tip:

- size() with even and round numbers!
- You can use math into parameter field
  Ex: point(100 / 5 , 100 * 2);

Time:

15 minutes

Processing
the coding side of Arduino

Variables

System variables

Let’s imagine if we change the picture, making it bigger (x2).

Is there any smarter way than change every number? YES.
We have to introduce two system variables: **width** and **height**.

Test#2

Re-make test#1 in order to make it work correctly with any size parameters.

Time:

10 minutes

Test#3

Draw an horizontal line and place 3 **visible** points upon it.

Don’t use system variables.

Time:

15 minutes

User variables pt.1

What if we move the line?

What happens to the points?

Is there any way to create a relationship between line and points?

YES.

We have to introduce user variables.

```
int yPosition = 30;
```

User variables pt.2

Variable *definition*	`int position;`
Variable *assignment*	`position = 75;`
Variable *definition* and *assignment*	`int position = 75;`

Test#4

Draw an horizontal line and place 3 **visible** points upon it.

Don’t use system variables, but use a user variable to make the point always be upon the line if moved.

Time:

10 minutes

Test#5

Draw an horizontal line and place 4 points with same distance from each other.

Remember: you can use math inside the parameter field!

Time:

15 minutes

User variables pt.3

- boolean student_present;
  student_present = false;
- char student_mark;
  student_mark= 'b'; // single quote
- float student_height;
  student_height= 1.84;
- int student_age;
  student_age= 22;
- String student_name;
  student_name = "Tom"; // double quote

User variables pt.4

- How to read the content of a variable?
- Write content on screen:

  text(variable , x , y);
- Write content on console:

  print(variable);

  Not only variables:

  print(“Variable value is:”);

  println(“example”);

Processing
the coding side of Arduino

Flow control

IF condition pt.1

```
if (check) {
        statement;
    }

if (check) {
        statement_1;
    } else {
        statement_2;
    }
```

IF condition pt.2

Check can only give as result **true** or **false.**

- If (pos_x == 80) equal to
- If (pos_x != 80) different from
- If (pos_x > 50) bigger than
- If (pos_x >= 50) bigger than or equal to

Check can involve more than a variable:

- If (pos_x == pos_y)

IF condition pt.3

- Check can be a logical combination of several simple checks

  If (pos_x > 50 && pos_x < 80)

- Can include all the typical elements of boolean math
  - && AND
  - || OR
  - ! NOT

Test#6

- Draw an horizontal line and a point
  - fill the background with green if the point is above the line
  - fill the background with red if the point is under the line

Tip: create user variables for the y coordinates

Time 15 minutes

Test#7

- The y position of the point is no more assigned by the user but assigned from a function:

```
random(max_value);
```

Time 5 minutes

Functions pt.4

random() is the first function that returns data instead of executing a job (change color, draw pictures, etc…).

Let’s look at Reference and find other functions that return data

Recap

Where do data come from?

- System variable
- User variable
- System function
- User function

width, height,

int age, float distance

random(), sqrt()

“Butterfly” programs

All the programs written lived just for a very short amount of time.

Let’s draw on the screen what a new function returns: day()

```
size(200,100);
int value;
value = day();
text( value, 10, 10);
```

Now change the time function, step by step.

First complete program pt.1

```
void setup(){
        instruction#1;
        instruction#2;
        instruction#…;
        instruction#n;
                }

void draw(){
        instruction#1;
        instruction#2;
        instruction#…;
        instruction#n;
        }
```

First complete program pt.2

Everything we have done so far was as if it had been written in setup().

```
void setup(){
        instruction#1;
        instruction#2;
        instruction#…;
        instruction#n;
        }
```

First complete program pt.3

So, what does it happen in draw()?

Let’s try.

```
void draw(){
                text( second(), x, y);
        }
```

And also

```
void draw(){
                ellipse( second(), y, 10, 10);
        }
```

What is happening?

background(255);

More global variables

- mouseX, mouseY
- mousePressed
- key
- keyPressed

Test#8

- Draw an horizontal line and a point in mouse position
- fill the background with green if the point is above the line
- fill the background with red if the point is under the line

Time 10 minutes

Arduino

Download and Setup IDE

Arduino IDE

It originated from the IDE for the language Processing.

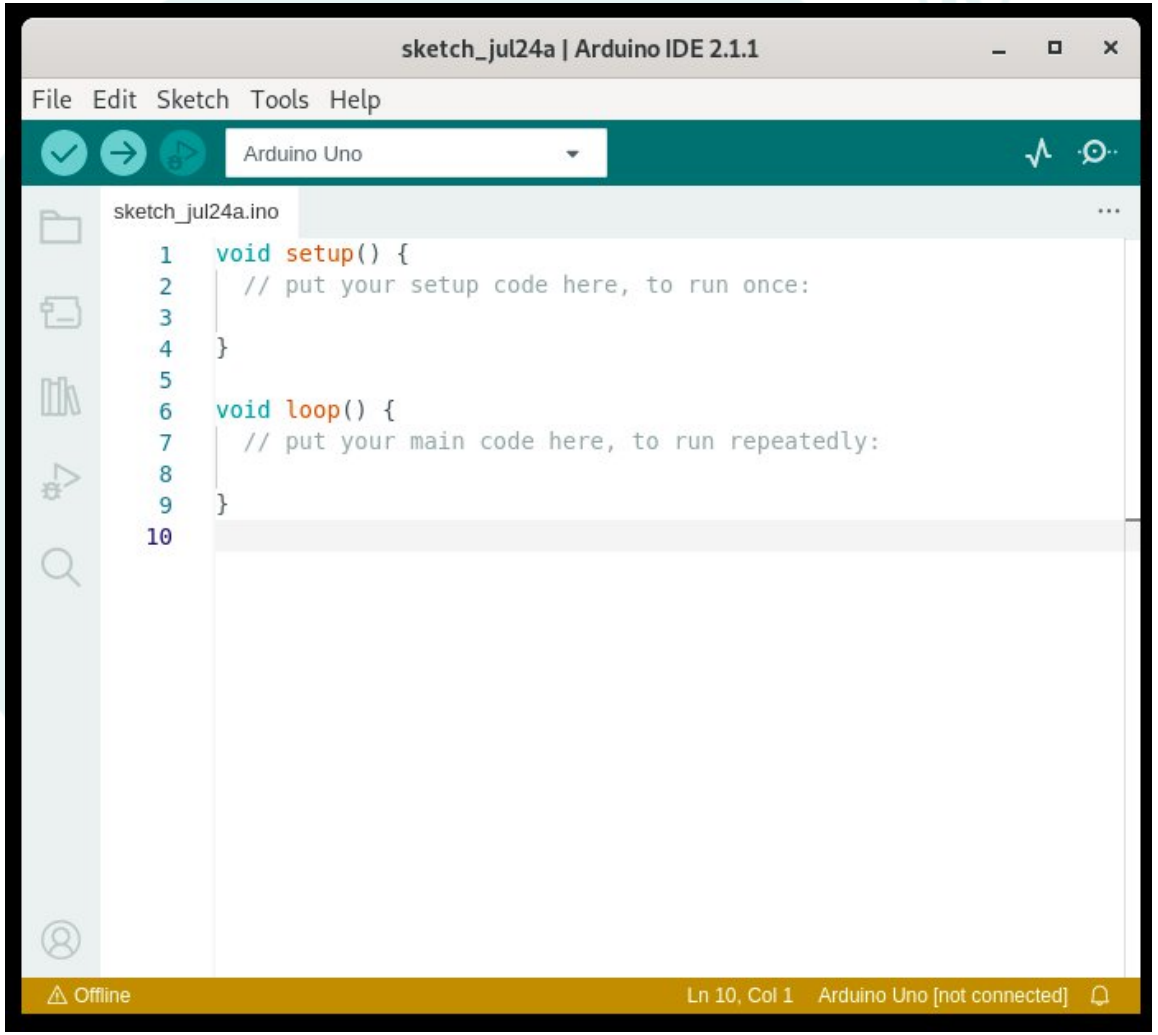
The coding rules are exactly the same.

The system variables and system functions are totally different:

It can't exist mouseX, size(), background(), etc.

Most of the variables and the functions in Arduino IDE are related to the physical environment.

Where Processing uses the function draw(), Arduino uses the function loop(). Both have the same functionality.



Arduino

Serial communication between board and PC

Serial pt.1

```
void setup(){
        Serial.begin(9600);
        Serial.print(“hello”);
}

void loop(){
}
```

Serial pt.2

As we said Processing and Arduino have many things in common.

Write a program that uses five different datatypes and show their value to the user.

Just once.

Time 15 minutes

Serial pt.3

Write a program that sends millis() on the serial terminal.

Continuously.

Time 10 minutes

Serial pt.4

Write a code that shows to the user, on the serial terminal, a multiple selection.

The selection has to be made pressing different letters (a, b, c, d)

The menu should be clear and readable.

Time 10 minutes

Serial pt.5

Let’s write together the code that receive the desired selection and gives a feedback to the user.

Serial pt.6

Use two options of selector to enable / disable the printing of millis().

Time 15 min

Tx / Rx

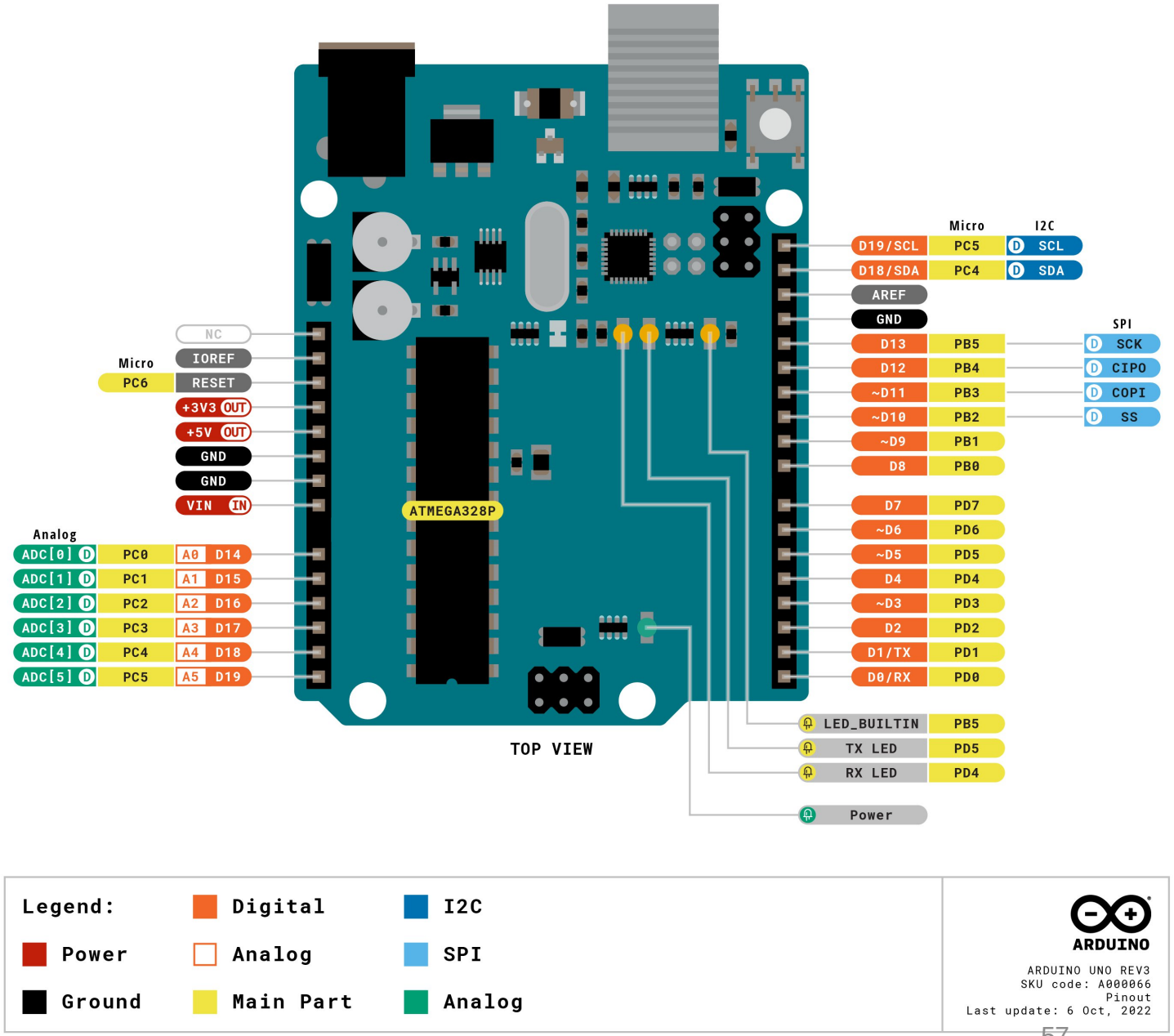
Let’s have a look at the Tx and Rx LEDs on the board.

Arduino

I/O ports

Arduino board

- Let’s have a closer look at the board using Fritzing



Arduino
Digital Input

Arduino board

How does informations go to and come into the boards?

Incoming informations:

- Boolean
- Numerical

Outgoing informations:

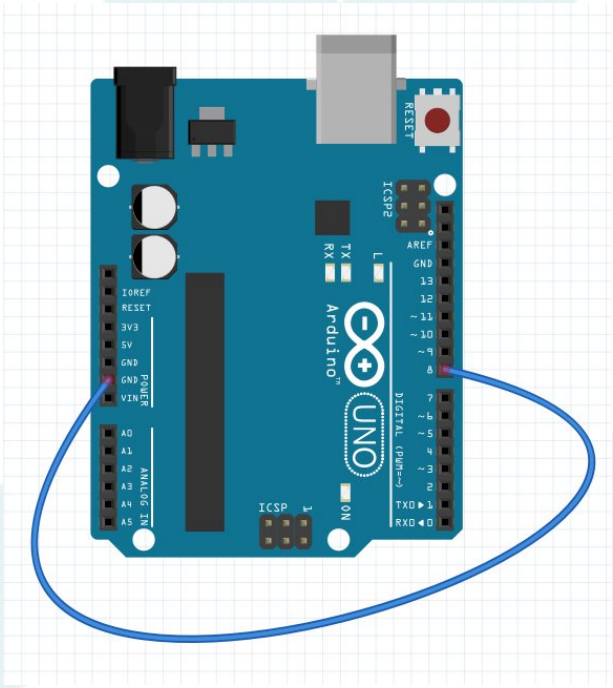
- Boolean
- Numerical

Boolean incoming informations pt1

- Arduino logic level high is 5V
- Is there 5V or 0V signal connected to a pin?

Boolean incoming informations pt2

- Is there 5V or 0V signal connected to a pin?



Boolean incoming informations pt3

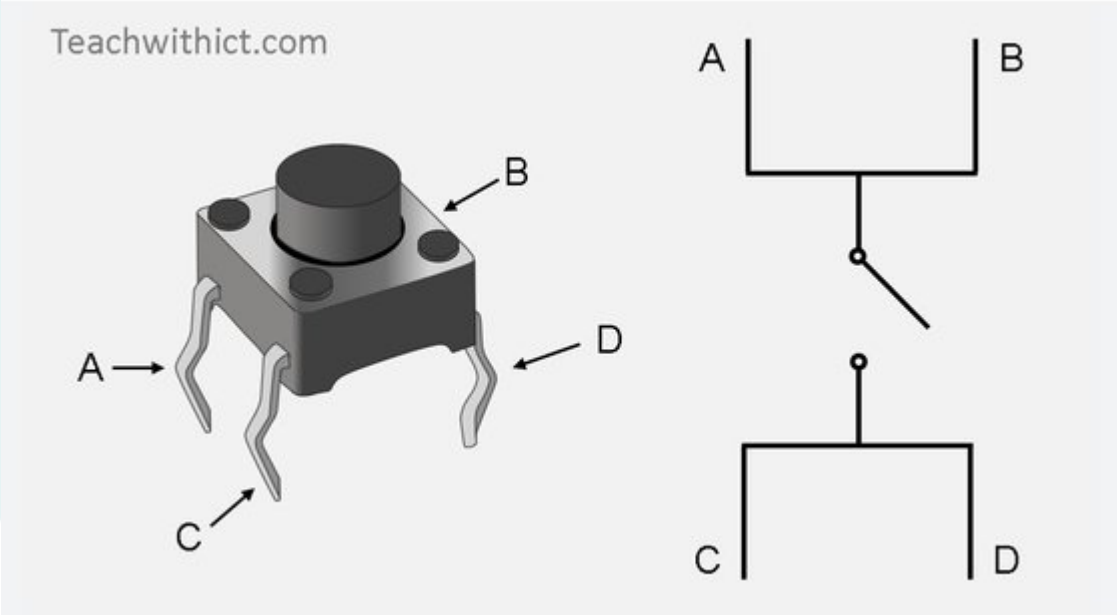
- Let’s write our first Arduino code to interact with hardware

```
void setup(){
                pinMode(8, INPUT);
                Serial.begin(9600);
        }

void loop(){
                bool myDI = digitalRead(8);
                Serial.println(myDI );
        }
```

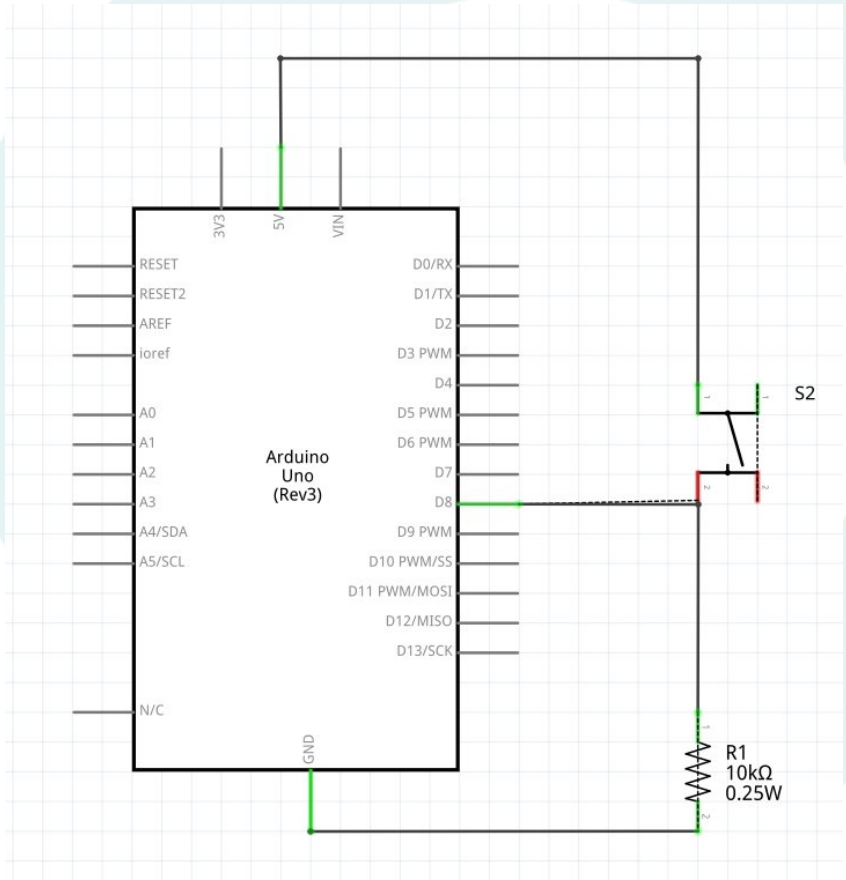
Boolean incoming informations pt3

First device: the button



Boolean incoming informations pt4

- External
  pull down resistor

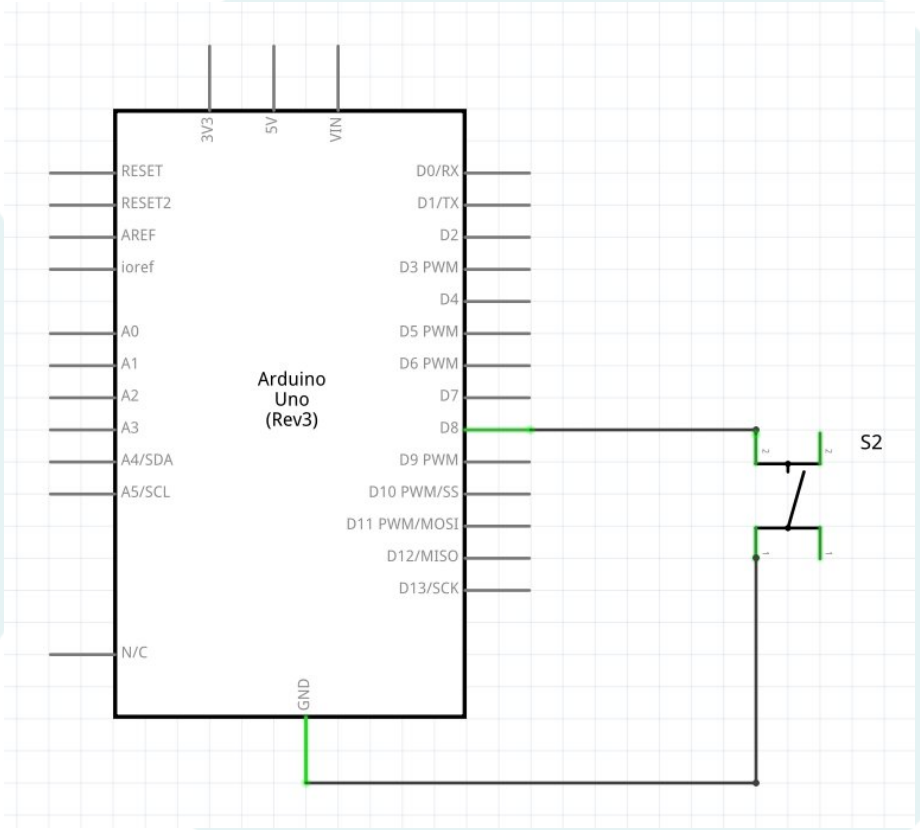


Test it
10 min

Boolean incoming informations pt5

- Internal
  pull up resistor

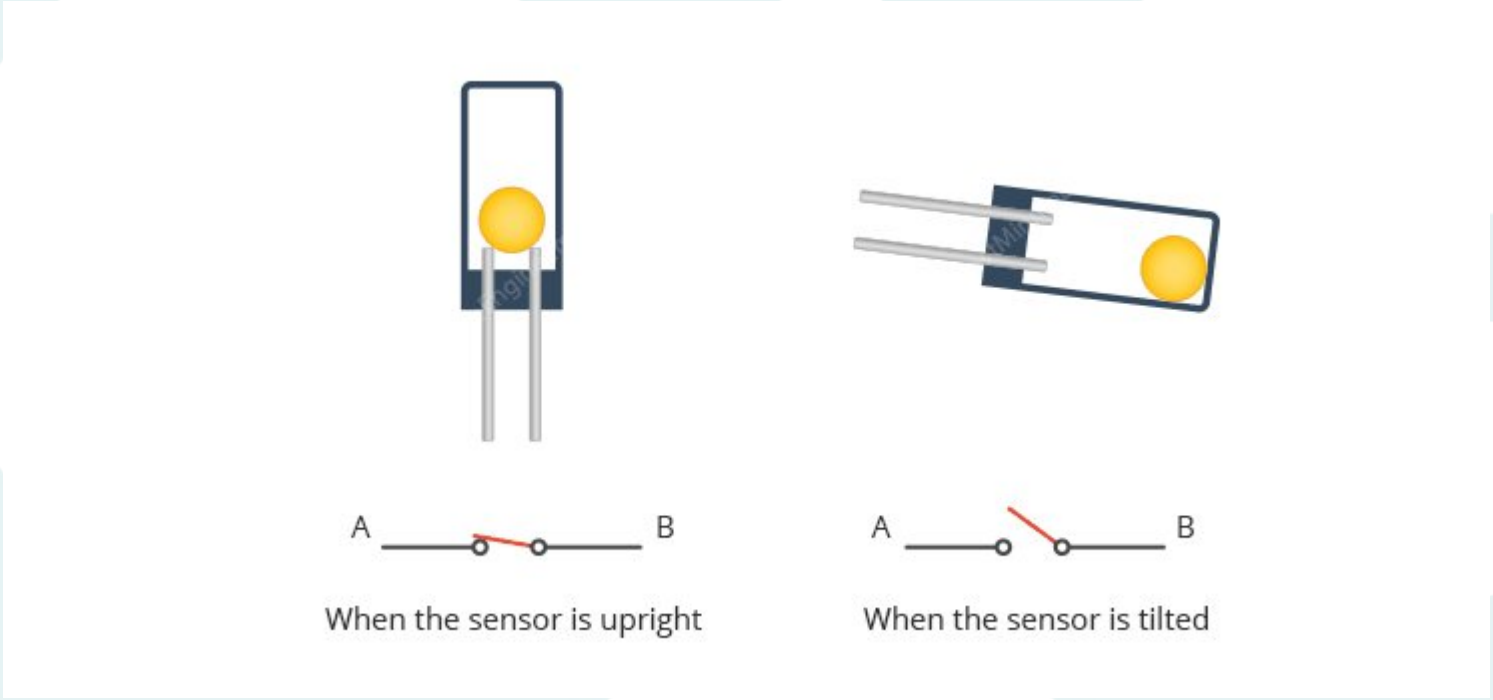
pinMode(8, INPUT_PULLUP);



Test it
10 min

Boolean incoming informations pt6

- Tilt switch



Test

- Create a code that shows that button is pressed, but only if the tilt switch is not tilted
    - Tip: there are at least 2 ways to do that!

Time 15 minutes

Arduino
Digital Output

Boolean outgoing informations pt1

```
void setup(){
                pinMode(13, OUTPUT);
        }

void loop(){
                bool myDO = true;
                digitalWrite(13, myDO);
        }
```

Try with **false** and find what changed.

Boolean outgoing informations pt2

```
void setup(){
                pinMode(8, OUTPUT);
                // Serial.begin(9600);
        }

void loop(){
                // Serial.println(8);
                bool myDO = true;
                digitalWrite(8, myDO);
        }
```

Try with **false** and find what changed.

Boolean outgoing informations pt3
LED

Connecting a LED requires a specific circuit, for different reasons:

- LED has a polarity, the current flow is possible in a single direction
- LED has very low resistance, so it would let flow “high” current (the maximum available from Arduino Digital Output is 40 mA). Current has to be limited, so it’s needed a resistor: which resistor?
  Every kind of LED has a different voltage drop, so it’s needed a different resistor.
- It can provide a maximum current of 40mA, but only 20mA continuously, that is enough to drive a single-color LED.

Boolean outgoing informations pt4 Blink

Let’s have a look at the Example code.

Now let’s try to make two different LED blinking at different timing.

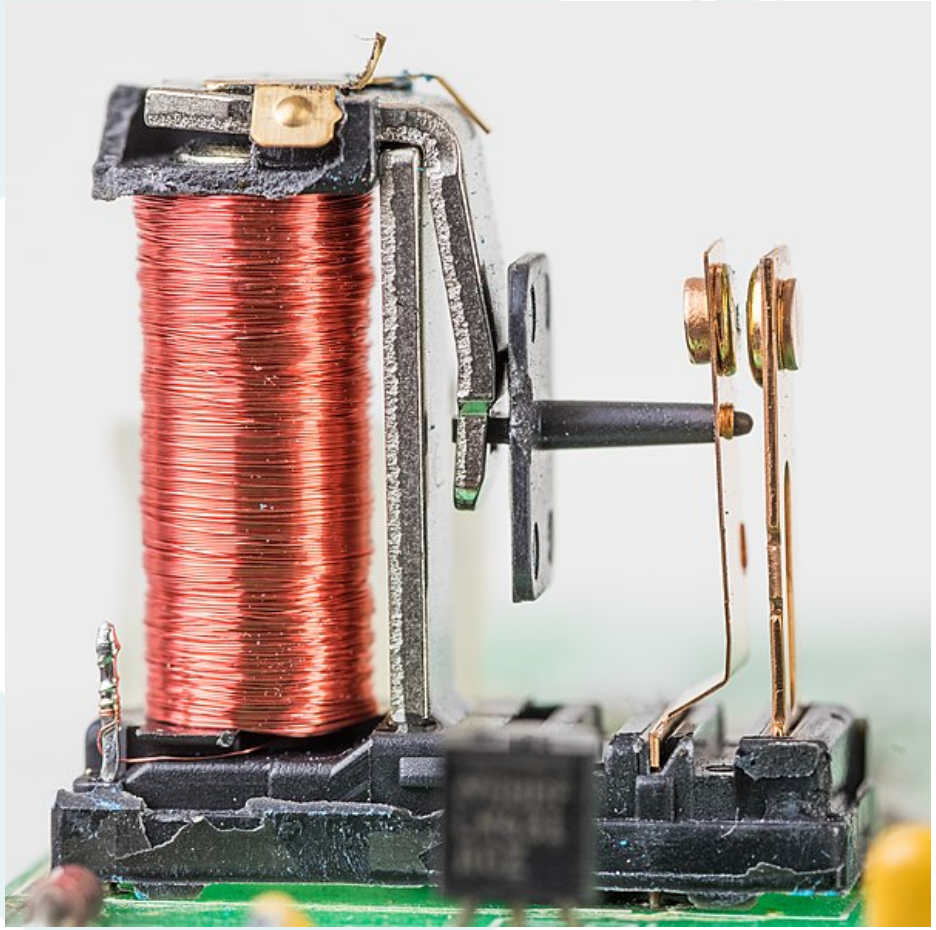
delay() is a very bad way to make things work.

The correct way is to use millis().
Even if it’s not so easy.

Let’s start from re-writing the blink example.

Boolean outgoing informations pt5 Relay

- Relay is probably the most important output device, because it’s like a bridge between the electronic and electrical systems.
- it’s like a switch button, but instead of being switched by finger, it’s switched by a low voltage signal
- It creates a separation between the control side and the actuator side
- A device powered by higher voltage can be controlled by low voltage signal
- https://en.wikipedia.org/wiki/Relay#/media/File:Relay_animation_without_flyback_diode_.gif



Input and output

Let’s use two relays to communicate between two Arduino.

Arduino
Analog Sensors

Analog Input pt.1

On the physical side the analog value has to be a voltage between 0V and 5V.

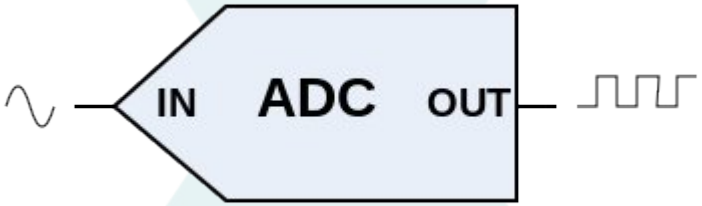
On the coding side the analog value is managed as a number between 0 and 1023.

1024 different levels.

Why 1024?

Analog Input pt.2

10 bit ADC : 2^10 = 1024



First test using GND, 5V and 3.3V with 1k Ohm.

Analog Input pt.3

```
int analogInput_pin = A0;
int analogInput_value;

void setup() {

  Serial.begin(9600);

}

void loop() {

  analogInput_value = analogRead(analogInput_pin);

  Serial.println(analogInput_value);

}
```

Analog Input pt.4

Are we able to print the value as a voltage instead of a number.

We have to calculate a conversion:

InputVoltage : (5V ‒ 0V) = AnalogInput : (1023‒0)

Arduino IDE has a function for doing this: map()

Analog Input pt.5

Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Parameters

`value`: the number to map.

`fromLow`: the lower bound of the value's current range.

`fromHigh`: the upper bound of the value's current range.

`toLow`: the lower bound of the value's target range.

`toHigh`: the upper bound of the value's target range.

Returns

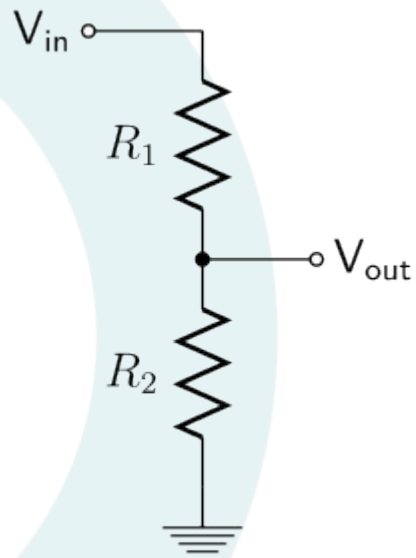
The mapped value. Data type: `long`.

Example

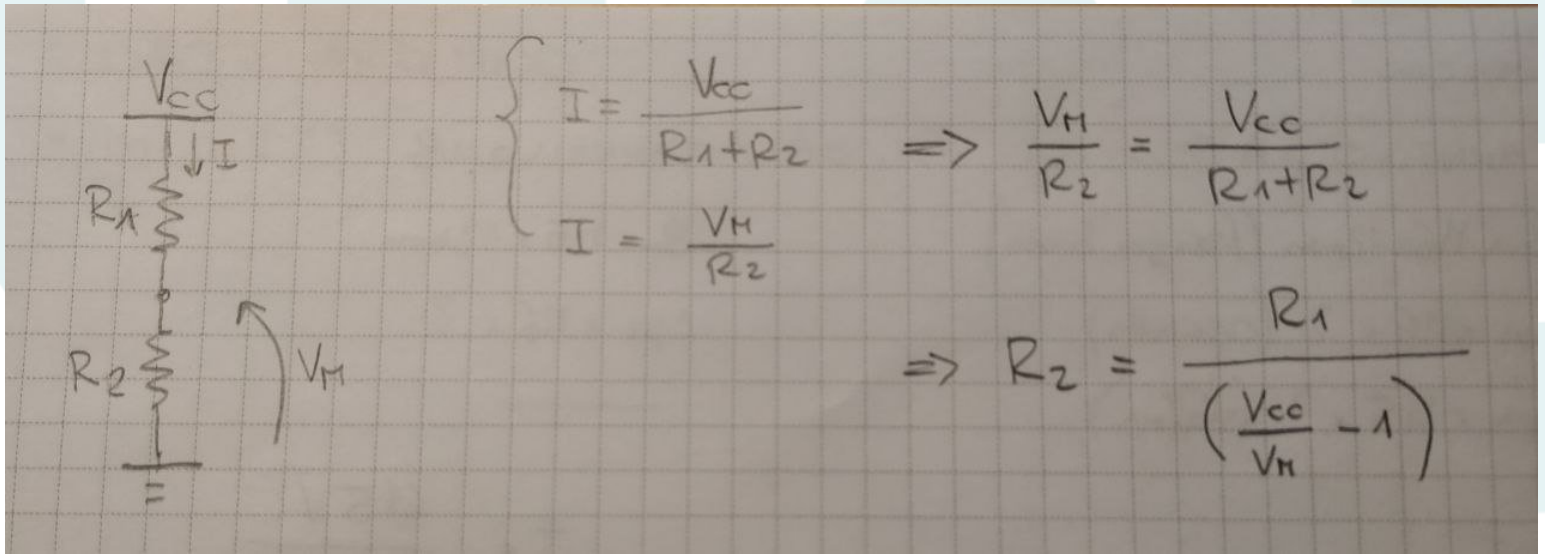
Let’s write together a code that shows the input value as voltage

Analog Input pt.6

https://en.wikipedia.org/wiki/Voltage_divider



Can I detect the value of the resistor from the voltage?



Analog Input pt.7
Photocell

Let’s see how the photocell resistance value changes and select a minimum value to turn on an digital output.

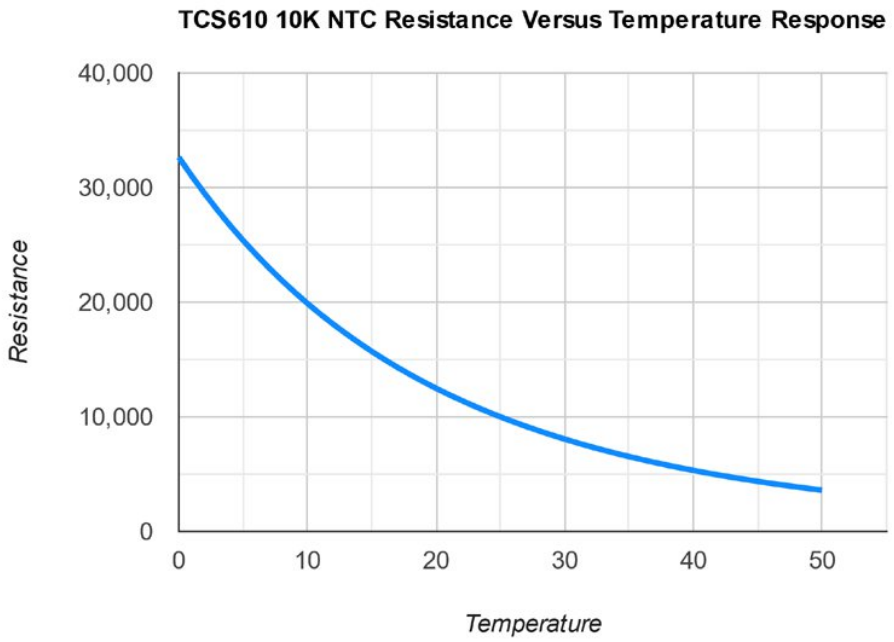
Tip: use pin 13

Analog Input pt.8 Thermistor

NTC : Negative Temperature Coefficient.

Its resistance will decrease when the temperature increases.

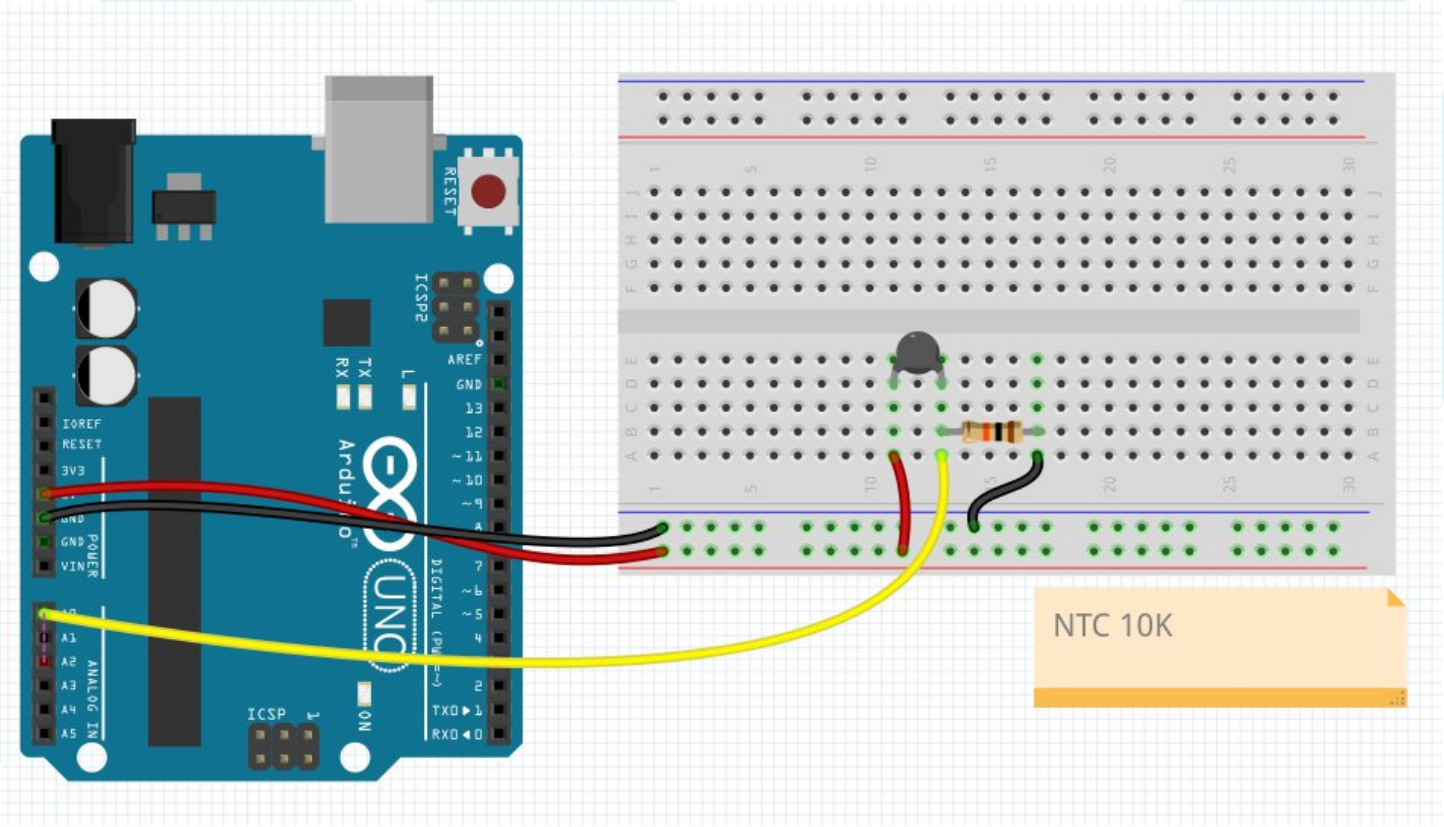
Let’s detect the correct temperature and send alarm messages on serial if temperature is too low or temperature is too high.



Analog Input pt.9 Thermistor

The known resistor has to be equal to the value of the resistor at 25°C, so 10kOhm.

NB: Voltage measured on the known resistor

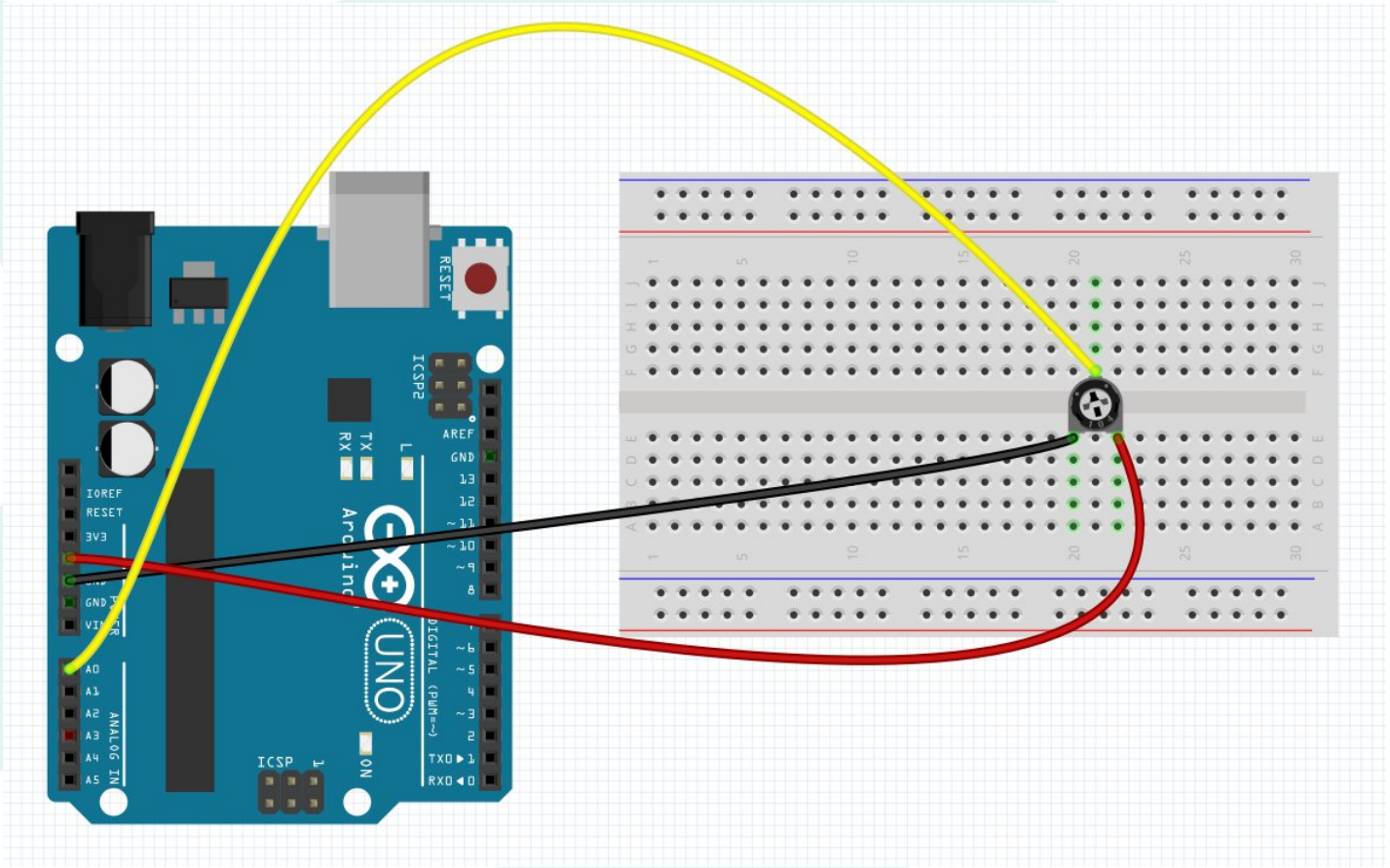


Analog Input pt.9 Potentiometer

Potentiometer doesn’t need any known resistor, and its physical value could be not relevant.

It could just be used as a human interface to the application.

Let’s see Serial Plotter tool.



Arduino
Analog Actuators

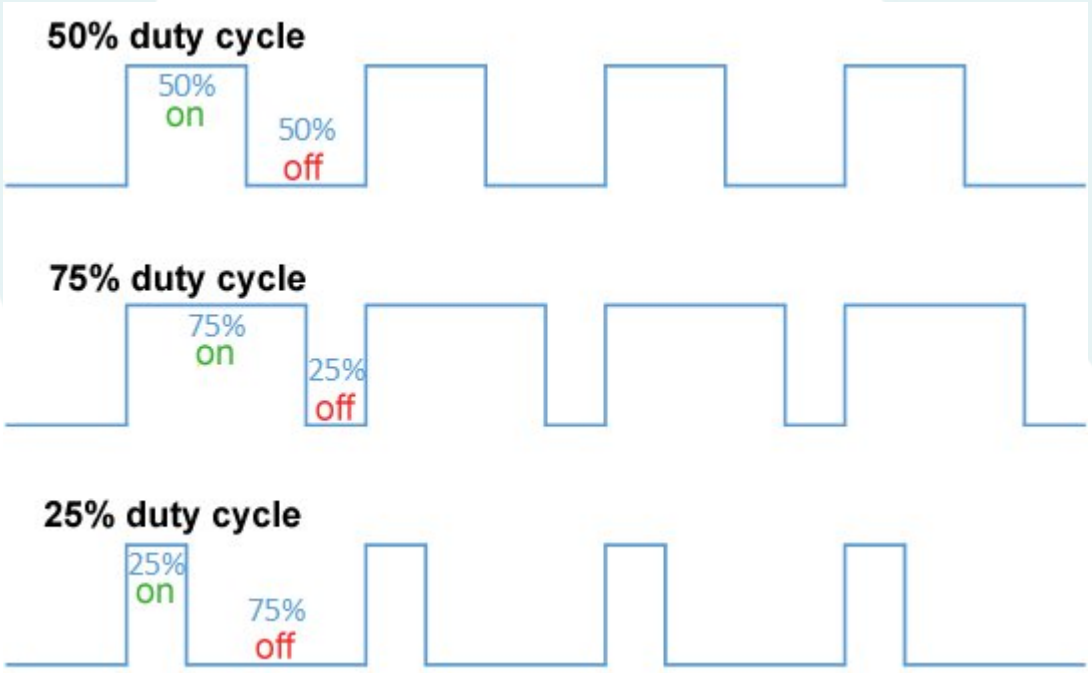
Analog Output pt.1
No DAC available!

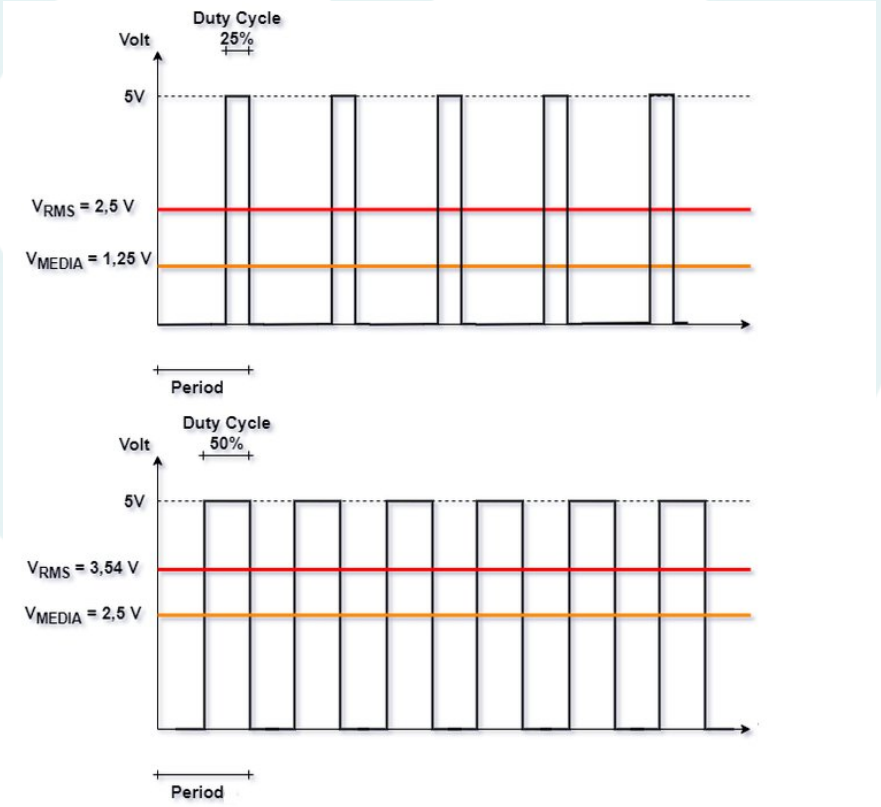
We could expect that as we have seen with Analog Input and ADC, for Analog Output there are some DAC, but it’s not true.

Analog Output pt.2
PWM – Pulse Width Modulation

PWM frequency = 490Hz (980Hz on pin 5 and 6)

User can set duty cycle percentage with a value from 0 to 255





Analog Output pt.3

```
int analogOutput_pin = 9;
int analogOutput_value = 127;

void setup() {
  Serial.begin(9600);
  pinMode(analogOutput_pin, OUTPUT);
}

void loop() {
  analogWrite(analogOutput_pin, analogOutput_value);
}
```

Analog Output pt.4

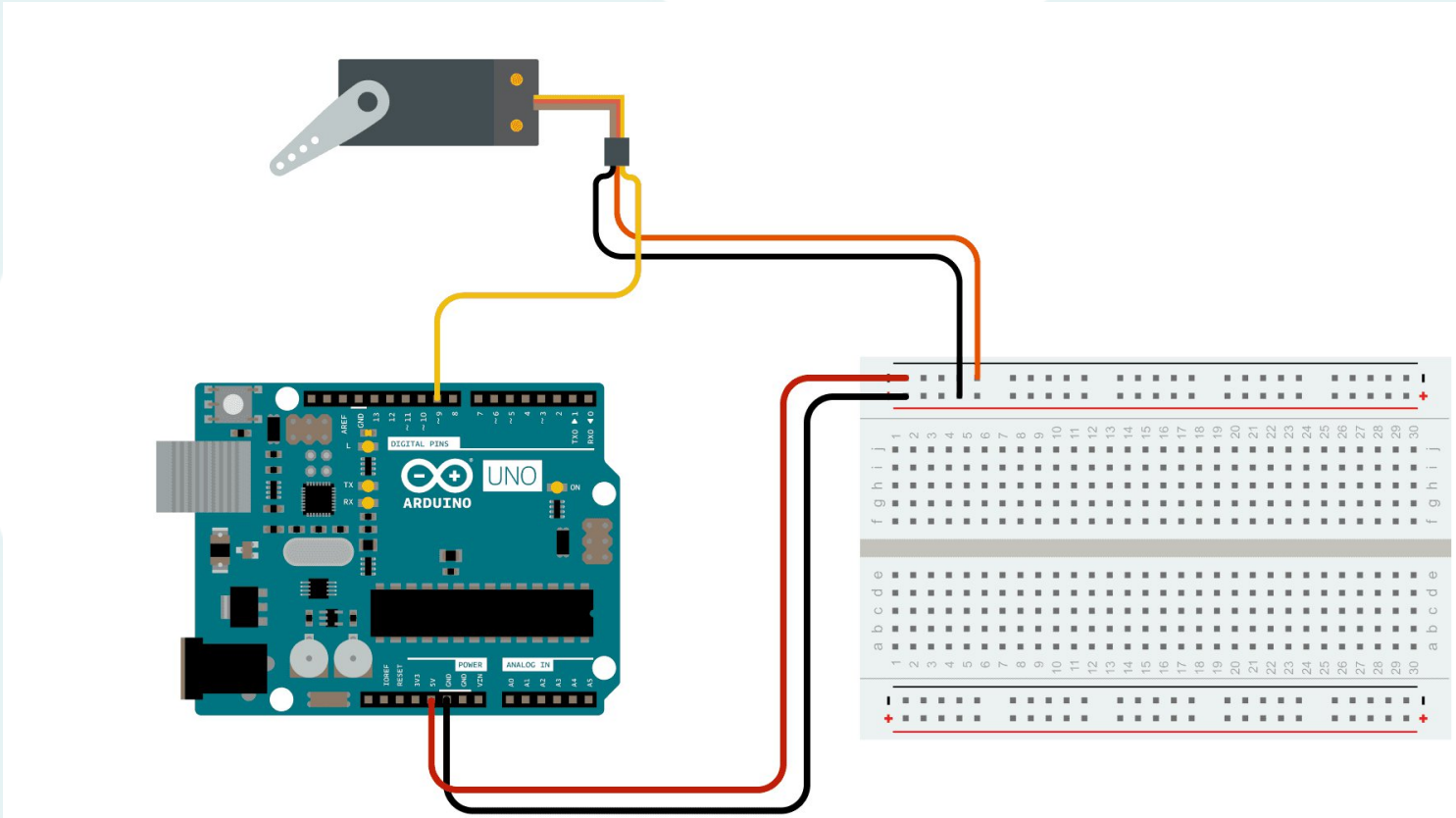
An examples of actuators where PWM is similar to a real Analog output: LED

Analog Output pt.5

Servo motor control in examples of actuator where PWM is not used as substitute for a missing analog output, but where is required a square wave generator, that we control with PWM.

Passive buzzer instead, has to receive an incoming signal with specific frequency for every desired tone.

Analog Output pt.6
Servo Motor



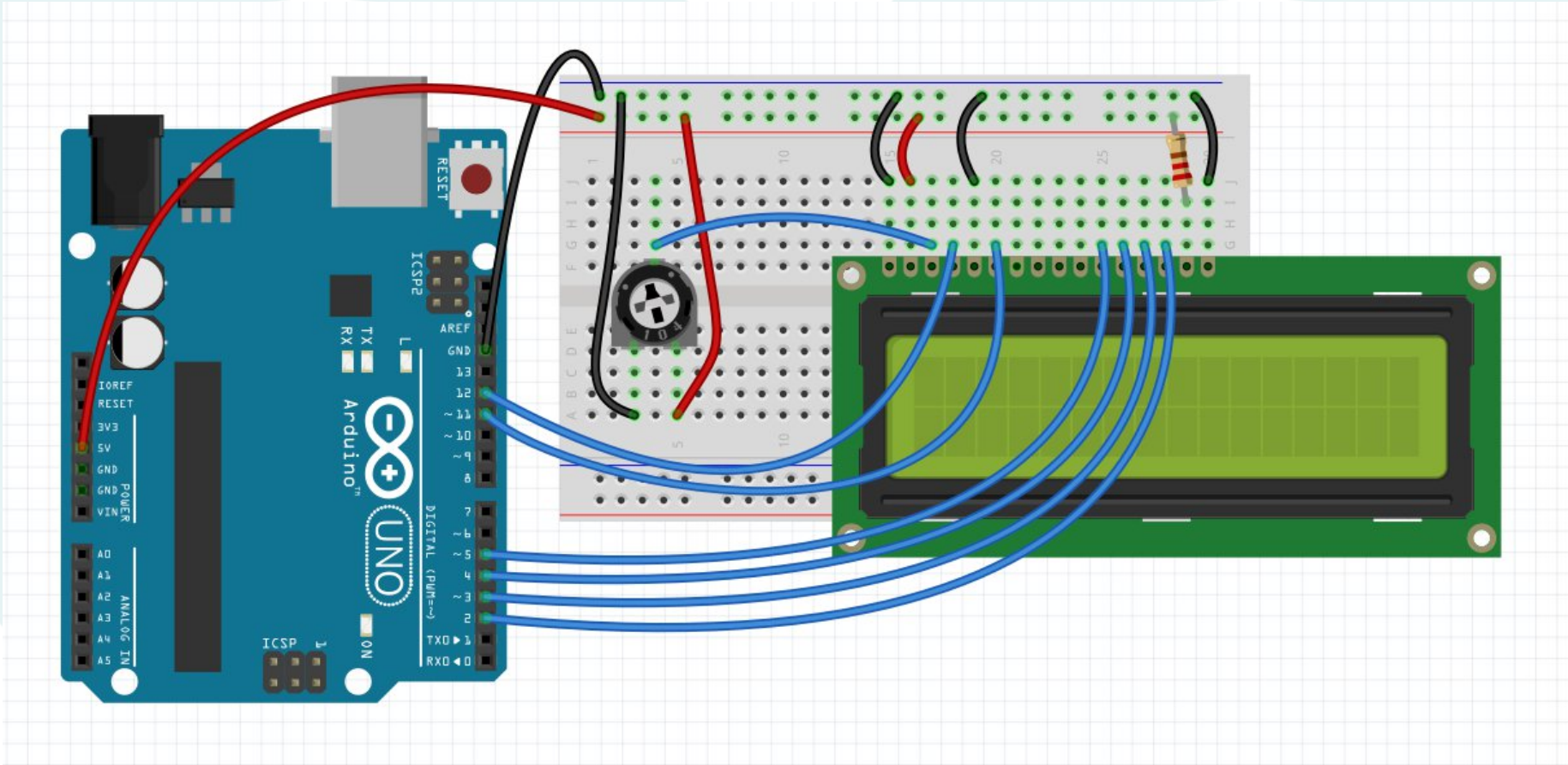
Analog Output pt.7

Move the servo using a potentiometer as input.

Time 20 min

Arduino
Protocol communication with sensors

Liquid Crystral Display
Parallel Interface



Protocols for sensor and actuators

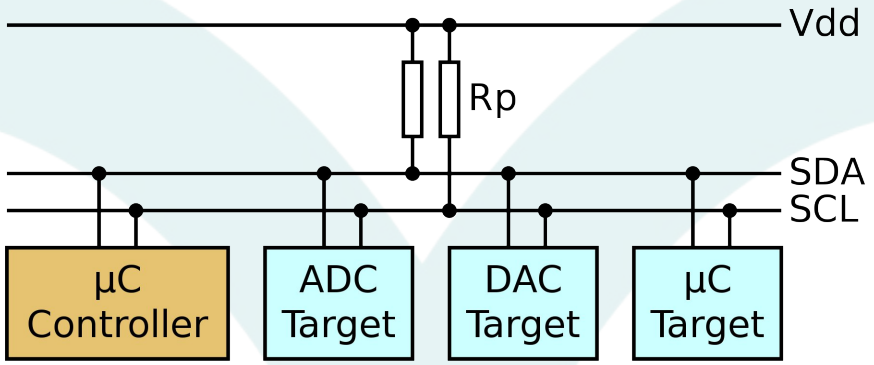
LCD Display

OLED Display

DS18B20 temperature sensor

DHT11 temperature and humidity sensor

I^2C – Inter Integrated Circuit

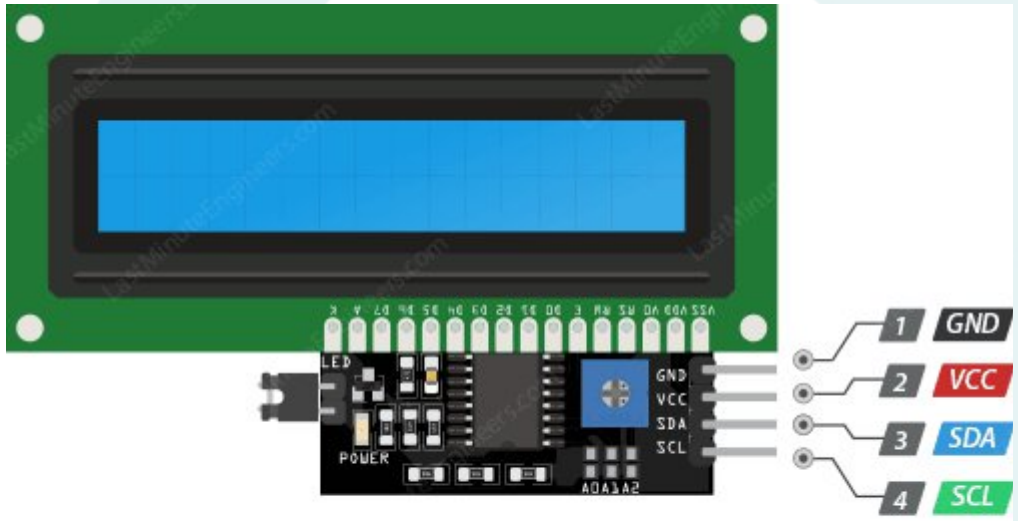


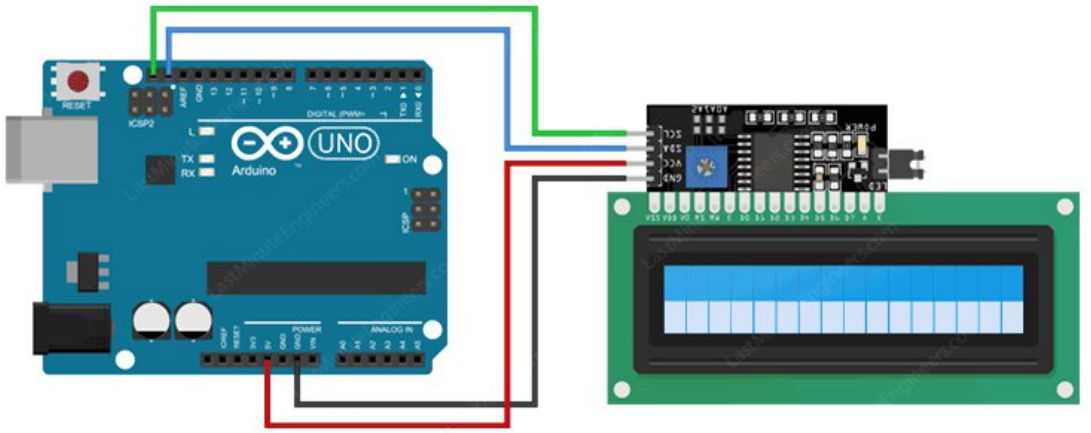
2 wires for communication

SCL : Serial Clock
SDA : Serial Data

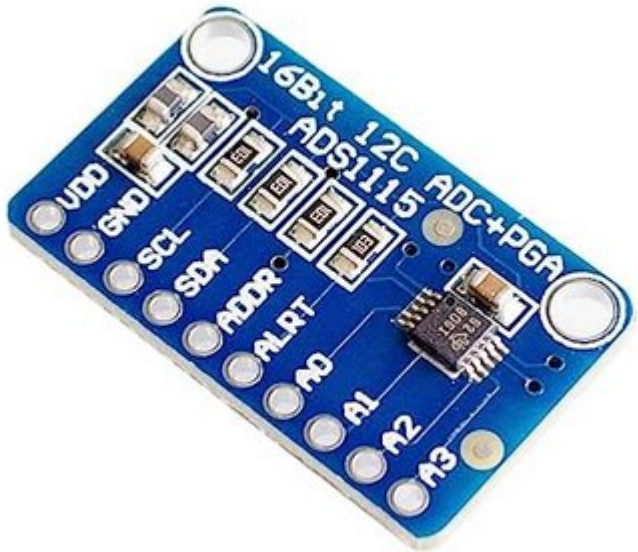
2 wires for power

I^2C – Inter Integrated Circuit
I^2C adapter for LCD



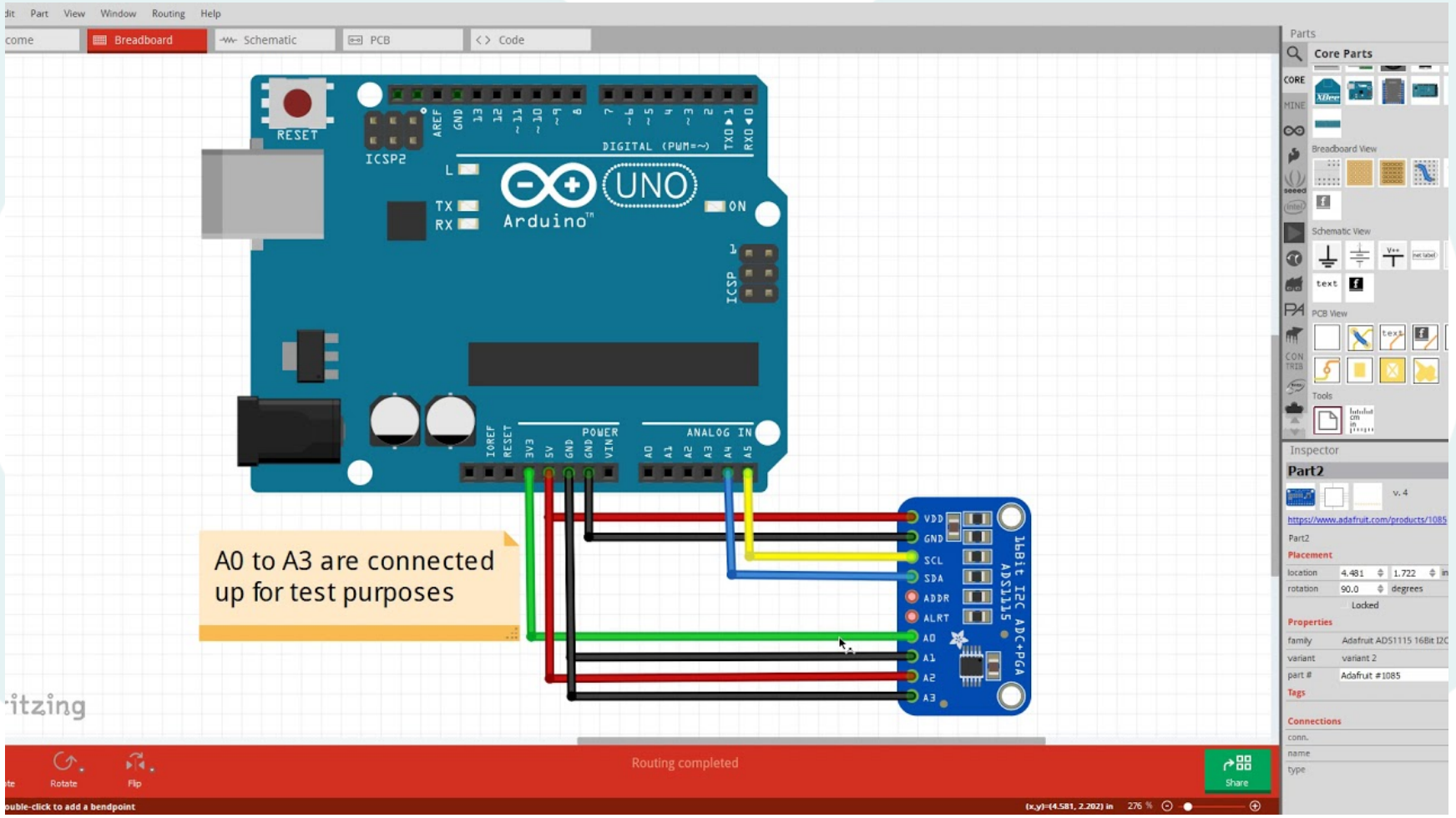


I^2C – Inter Integrated Circuit
AI expansion



ADS1115 Module ADC Module 16Bit

I²C – Inter Integrated Circuit
AI expansion

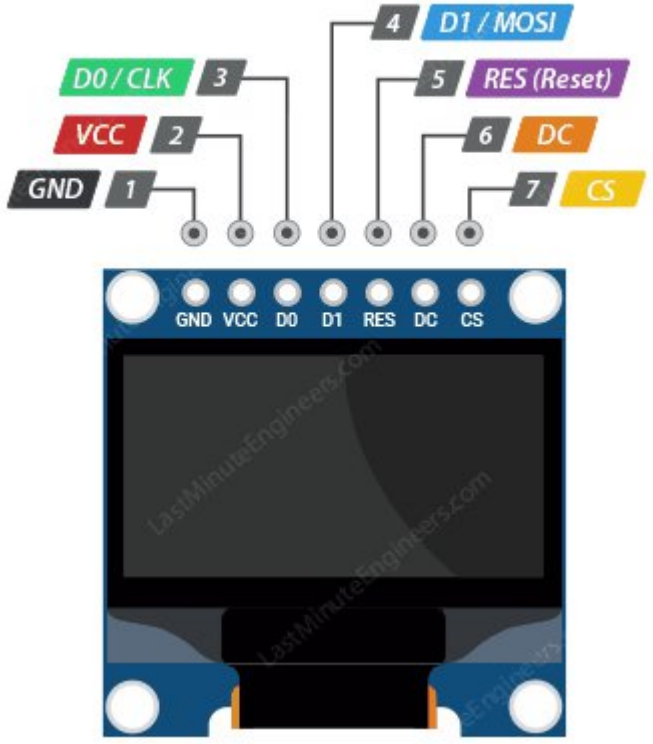


I²C – Inter Integrated Circuit SSD1306 OLED Display

I2C Display

Fixed I2C Address

Pin order not always the same!



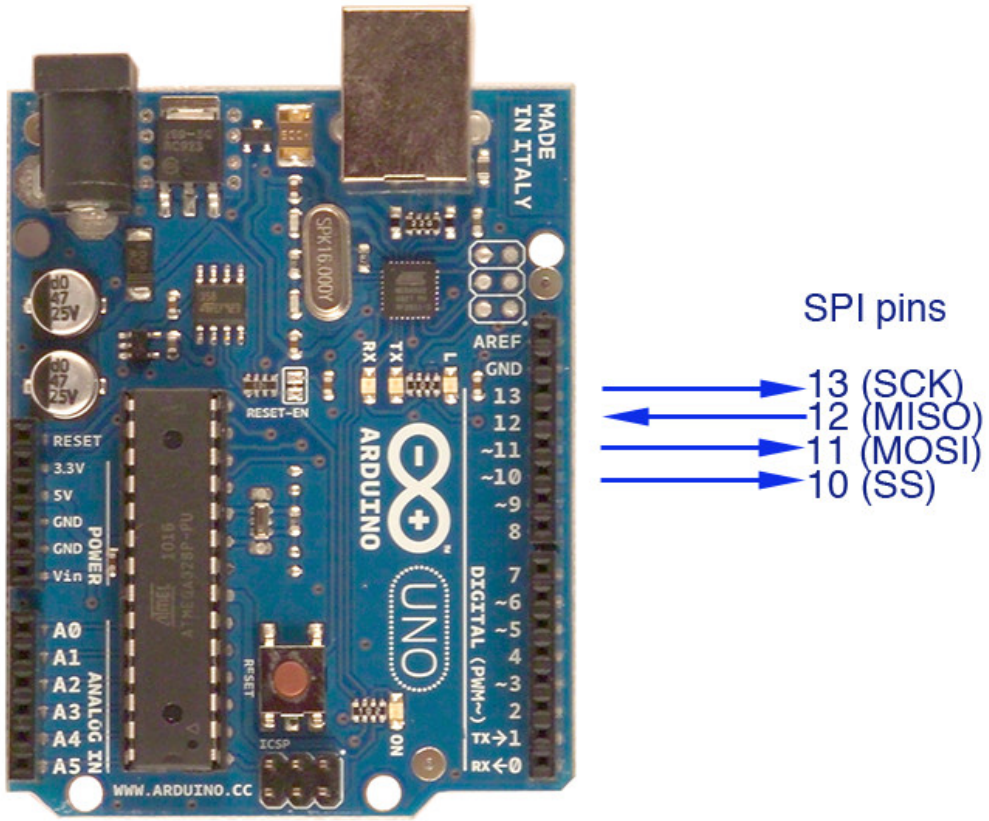
SPI – Serial Peripheral Interface

4 wires for communication

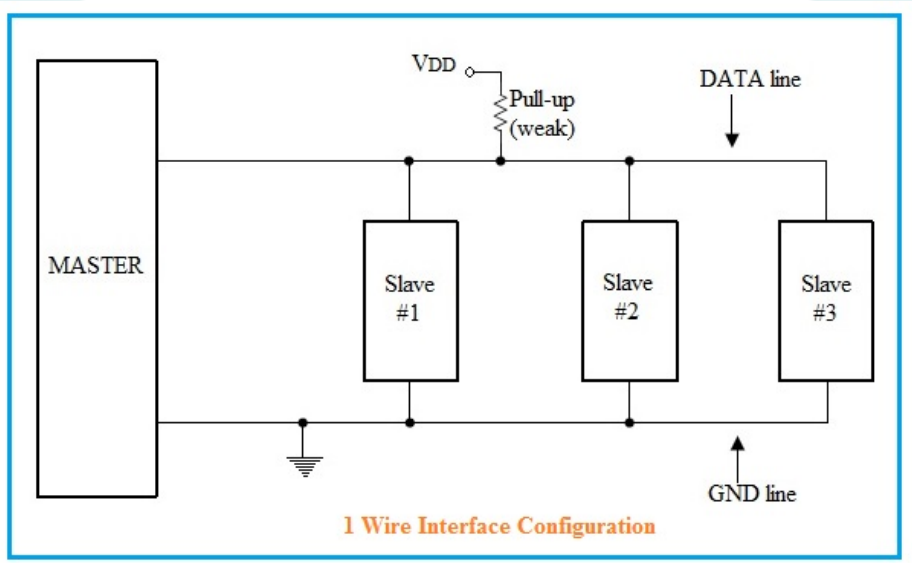
SCLK	:	Serial Clock (output from master)
MOSI	:	Master Out Slave In (data output from master)
MISO	:	Master In Slave Out (data output from slave)
SS	:	Slave Select (often active low, output from master to indicate that data is being sent)

2 wires for power

SPI – Serial Peripheral Interface



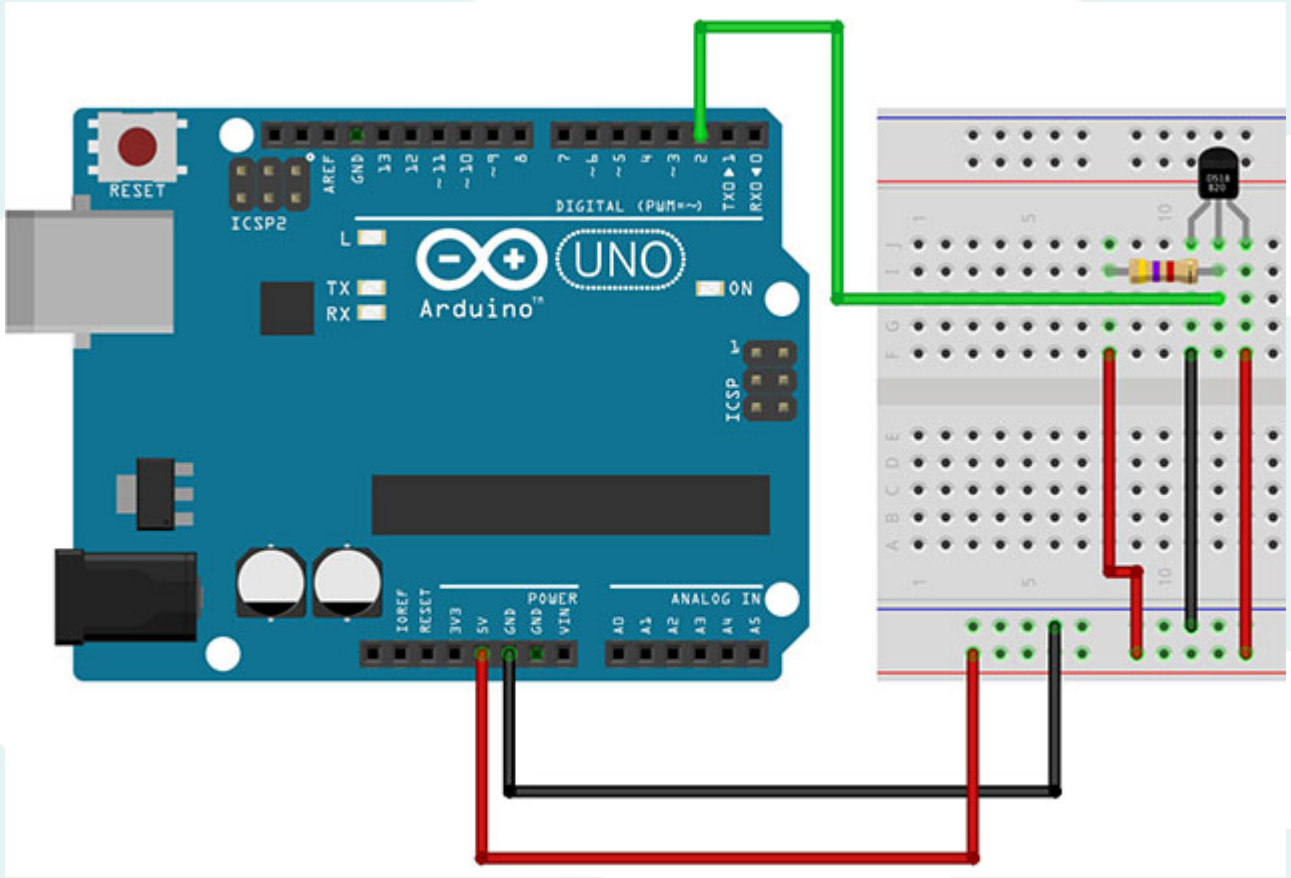
1Wire – One Wire



1 wire for data

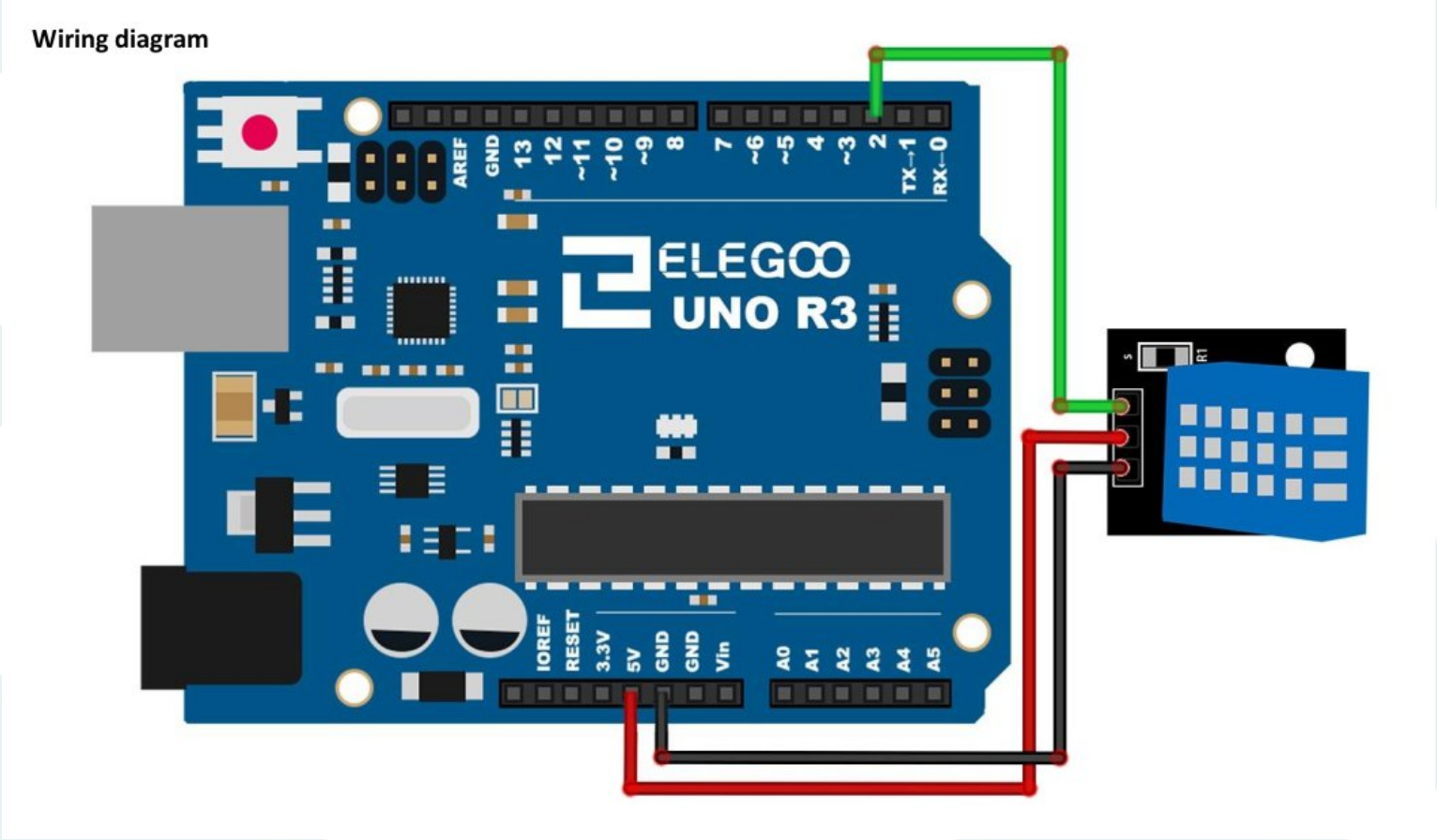
2 wires for power (sometimes used only 1 for GND)

1Wire – One Wire
DS18B20 Temperature Sensor



4,7k resistor between DATA and +5V

Example: DHT11



Arduino
Actuator control based on sensor feedback

Test

Turn on heater or cooler depending on temperature.

Show temperature and which actuator is on.

Test

Use a single button, to turn on multiple LEDs.

One click, one LED.

Two clicks, two LEDs.

Three clicks, three LEDs.

Four clicks, no LED.

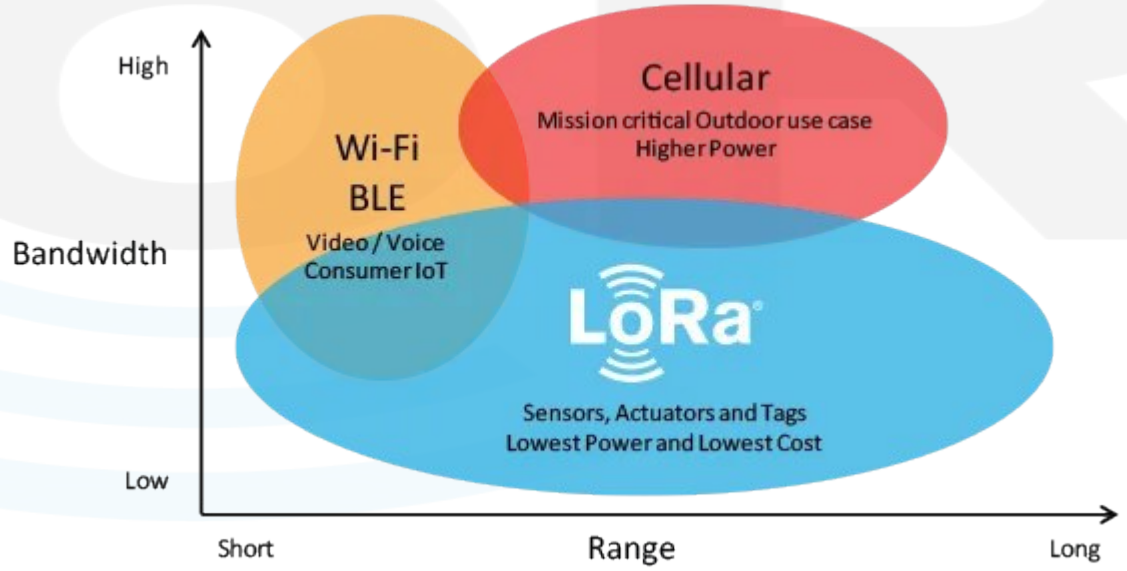
Test

Make multiple LEDs blinking at different time.

Agenda

Day 1	Processing, the coding side of Arduino: Functions and Variables
	Processing, the coding side of Arduino: Flow control
Day 2	Arduino: Serial communication between board and PC, I/O ports
	Arduino: Digital Sensors and Digital Actuators
Day 3	Arduino: Analog Sensors
	Arduino: Analog Actuators
Day 4	Arduino: Protocol communication with sensors
	Arduino: Actuator control based on sensor feedback
Day 5	LoRa: Point to point communication
	LoRaWAN: Gateway and Server

LoRa


High
Bandwidth
Low
Short
Range
Long
Wi-Fi
BLE
Video / Voice
Consumer IoT
Cellular
Mission critical Outdoor use case
Higher Power
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost

LoRa pt.1

LoRa : **Lo**ng **Ra**nge

It’s a physical **radio** communication technique.

It’s **proprietary** of Semtech Corporation.

When you buy a LoRa module, you have to consider **where** it will be used.

LoRa uses **license-free** sub-gigahertz radio frequency bands:

EU868 (863–870/873 MHz) in Europe

US915 (902–928 MHz) in North America

Worldwide 2,4 GHz (but already used by WiFi, Bluetooth, Zigbee)

LoRa enables **long-range** transmissions with **low power** consumption.

LoRa pt.2

The **spreading factor** (SF) is a selectable radio parameter from 5 to 12 that determines how much the information is spread over time.

A **lower SF** corresponds to a **higher data rate** but a **worse sensitivity**,

A **higher SF** implies a **better sensitivity** but a **lower data rate**.

Compared to lower SF, sending the same amount of data with higher SF needs more transmission time, known as time_on_air.

More time_on_air means that the modem is transmitting for a longer time and consuming more energy.

LoRa pt.3

COVERAGE

According to the LoRa Development Portal, the range provided by LoRa can be up to **4.8 km in urban areas**, and up to **16 km or more in rural areas** (line of sight).

COVERAGE EXTENSIONS

Range extenders for LoRa are called LoRaX.

LoRa pt.4

LoRa is **not** LoRaWAN.

LoRa defines the **lower physical layer.**

LoRaWAN defines the **upper layers of the network**.

LoRa pt.5
Best practices

https://www.thethingsindustries.com/docs/devices/best_practices/

Limit the frequency of your transmissions to the minimum possible.

Optimize message encoding for size. Shorter messages mean shorter transmission time.

The duty cycle of radio devices is often regulated by government.

If this is the case, the duty cycle is commonly set to 1%, but make sure to check the regulations of your local government to be sure.

For example, in Europe, duty cycles are regulated by section 7.2.3 of the ETSI EN300.220 standard.

LoRa pt.6

Expect Packet Loss

You should expect packet loss up to 10%.

LoRa pt.7 Hardware

RFM95 LoRa transceiver

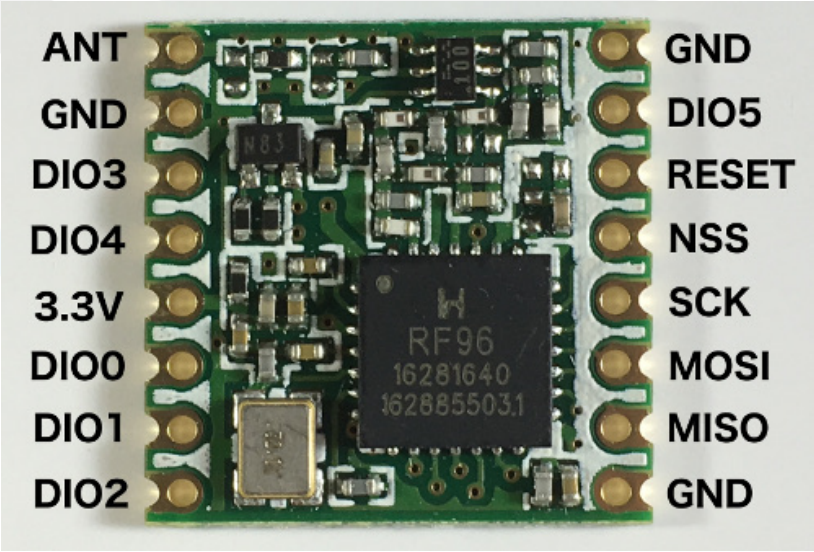
Module is marked RF96 which means the chip is using the SX1276 chip.

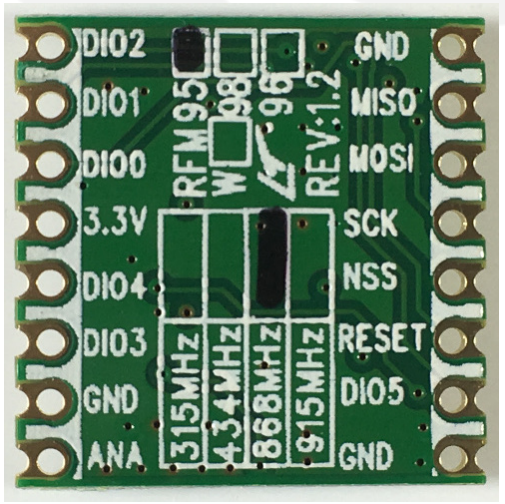
The RFM95 and RFM95W are the same.

W=CE/FCC version for worldwide, without W=internal Chinese market only.

Coverage

2 km line of sight using simple wire antennas, or up to 20 km with directional antennas and right settings

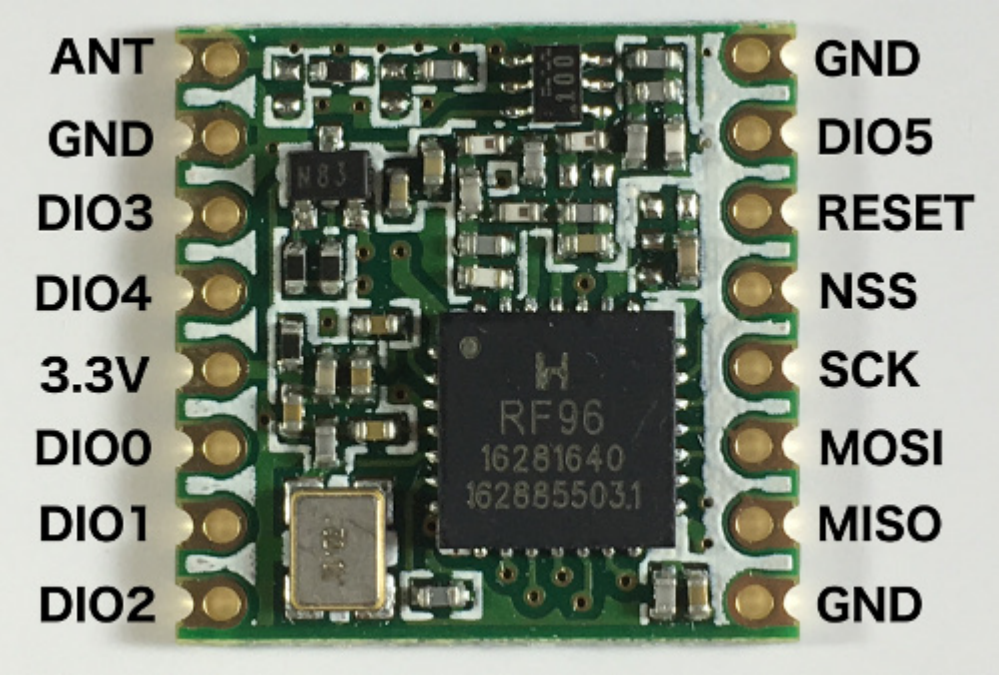


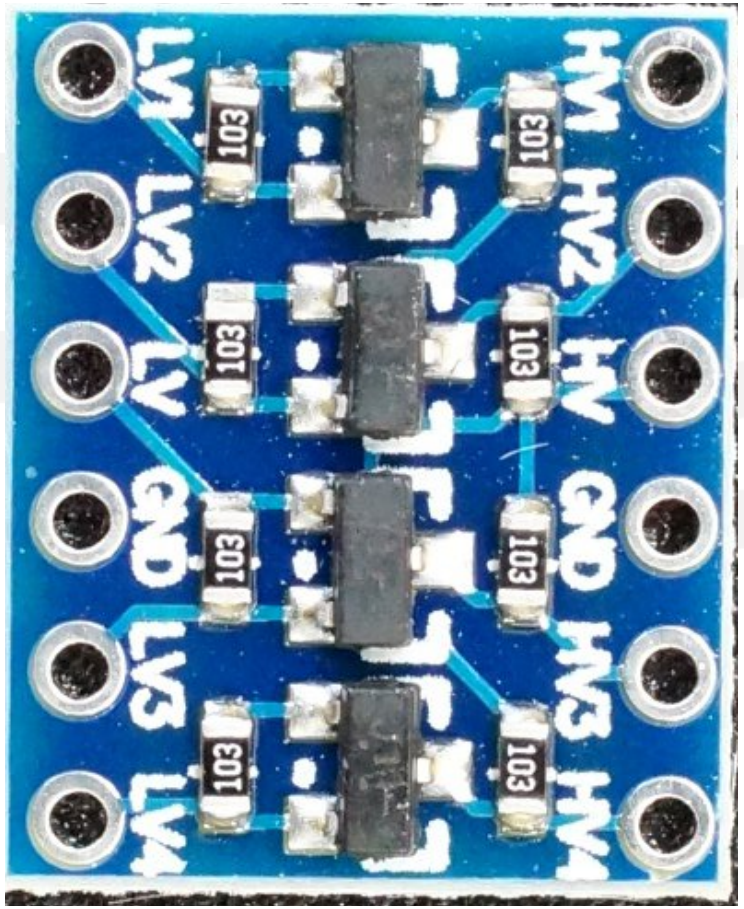


LoRa pt.8

SPI Protocol

3.3V Power and Logic

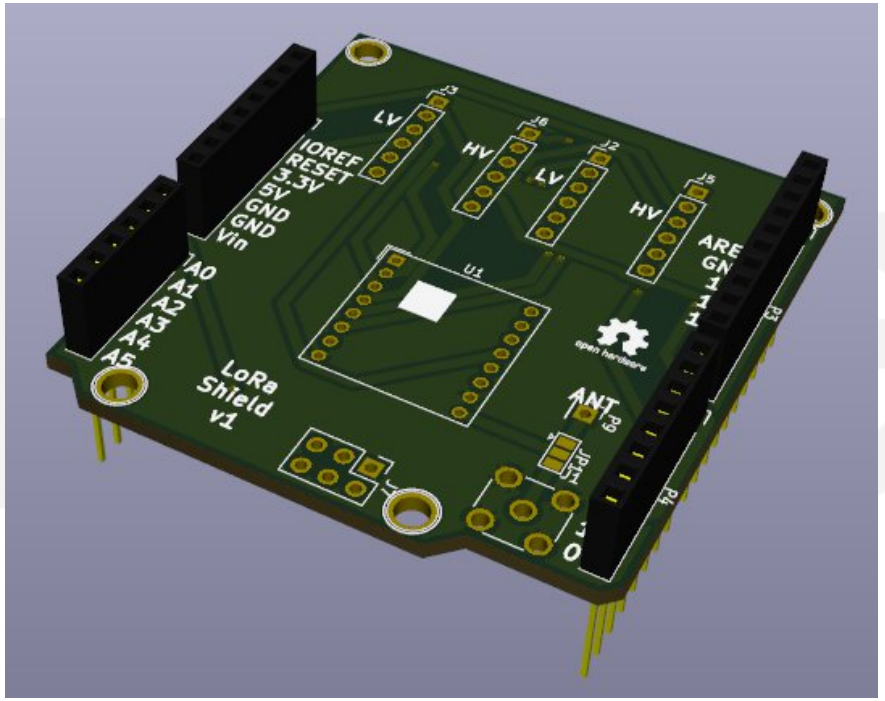




LoRa pt.9
Arduino Uno Shield

kicad Project

https://github.com/EcceRobot/kicad-uno-lora



LoRa Example#1

Point to point communication: simple message.

One sender, one receiver

Pay attention to Sync Word

LoRa Example#2

Point to point communication: Sensor and actuator.

One board is sending level measurement,
other board controls a digital output.

LoRa Example#3

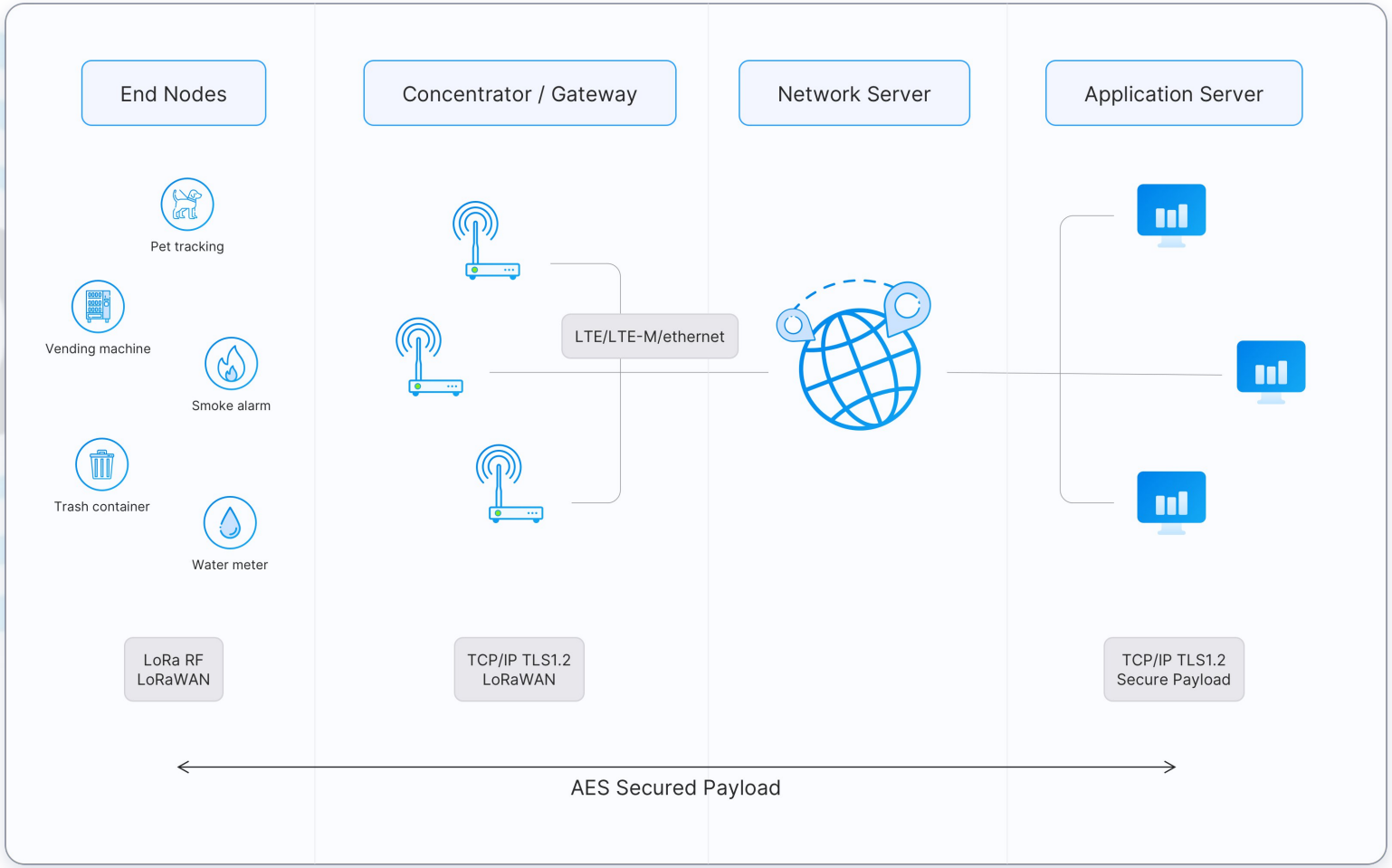
Many sensors and many nodes.

Every node sends its own values. Including a counter.

Just one node receives all the informations.

LoRaWAN
Gateway and Server

LoRaWAN pt.1 Architecture



LoRaWAN pt.2

Limitations

LoRaWAN is not suitable for every use-case, so it is important that you understand the limitations. Here's a quick overview:

Suitable use-cases for LoRaWAN:

- **Long range** - multiple kilometers
- **Low power** - can last years on a battery
- **Low cost** - less than 20€ CAPEX per node, almost no OPEX
- **Low bandwidth** - between 250bit/s and 11kbit/s in Europe using LoRa modulation (depending on the spreading factor)
- **Coverage everywhere** - you are the network! Just install your own gateways
- **Secure** - 128bit end-to-end encrypted

Not Suitable for LoRaWAN:

- **Realtime data** - you can only send small packets every couple of minutes
- **Phone calls** - you can do that with GPRS/3G/LTE
- **Controlling lights in your house** - check out ZigBee or BlueTooth
- **Sending photos, watching Netflix** - check out WiFi

LoRaWAN pt.3

Fair Use Policy

On The Things Network's public community network a **Fair Use Policy** applies which limits the **uplink airtime** to **30 seconds per day (24 hours) per node** and the **downlink messages** to **10 messages per day (24 hours) per node**. If you use a private network, these limits do not apply, but you still have to be compliant with the governmental and LoRaWAN limits.

LoRaWAN pt.4
Uplink

Sending data from a Node to your Application (uplink)

We want you to create products that are as efficient as possible. This will get the most out of your battery, and doesn't require you to buy many gateways. If you follow these recommendations, you'll definitely build an amazing product!

- **Payload** should be as small as possible. This means that you **should not send JSON or plain (ASCII) text**, but instead encode your data as binary data. This is made really easy with the Cayenne Low Power Payload format which is fully supported by The Things Network.
- **Interval** between messages should be in the range of **several minutes**, so be smart with your data. You could for example transmit a `min|avg|max` every 5 minutes, or you could only transmit when you sensor value changed more than a certain threshold or have it triggered by motion or another event.
- **Data Rate** should be as fast as possible to minimize your airtime. `SF7BW125` is usually a good place to start, as it consumes the least power and airtime. If you need more range, you can slowly increase until you have enough. You can also enable adaptive data rate (ADR), the network will then be able to automatically optimize your data rate.

LoRaWAN pt.5
Downlink

Sending responses from your Application to your Node (downlink)

We want to be able to handle as many Nodes as possible per Gateway. But as full-duplex radios are not widely available yet, a Gateway is not able to receive transmissions from Nodes while it is transmitting. This means that if a gateway is transmitting 10% of the time, it's not able to receive anything for that 10% of the time. This is even worse when you realize that a gateway can receive at 8 channels simultaneously. Except when it's transmitting. So while an idle gateway can receive transmissions from 8 devices, those 8 devices are worthless when the gateway is transmitting.

We want to build a network that offers high reliability. If your device transmits, the gateway should receive it. In order to keep the gateway availability as high as we can, we ask you to follow these recommendations.

- **Data Rate** should be, just as with uplink, as efficient as possible. The downlink data rate is based on the uplink data rate, so if you send efficient uplinks, the network will respond with efficient downlinks.
- **Downlink** messages should be avoided if possible, and if you send downlink, keep the payload small.
- **Confirmed Uplink** is often not necessary. Try to make your application work without confirmations.

LoRaWAN pt.6
Coverage

Ground breaking world record! LoRaWAN® packet received at 702 km (436 miles) distance

TheThings Network
The Things Network Global Team
Posted on 08-09-2017

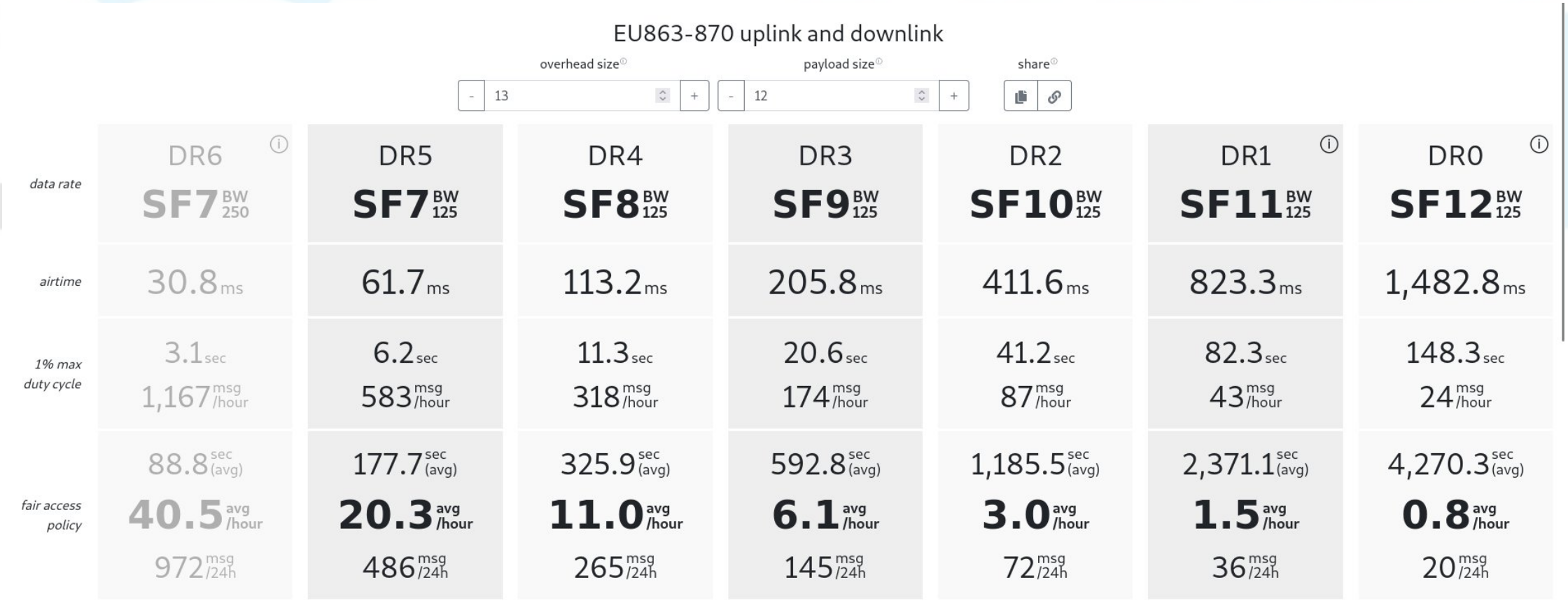
With the rise of novel wireless technologies, we surprise ourselves over and over again of what these technologies are capable of. LoRa has been around now for more than 2 years and people all over the planet are excited about its immense distance it can bridge while consuming extremely little energy.

During the past two years, many have tried to set world records on the maximum distance a data packet can travel. Ideal circumstances are found by climbing high buildings, mountains, or by releasing heliums balloon up in the air reaching for higher altitudes. Andreas Spiess from Switzerland became famous for his ground to ground connection of 212 km and the Dutch company SODAQ started with releasing helium balloons, resulting in a stunning 354 km from an altitude of almost 15 km.

On Saturday 26th of August, a weather balloon was launched during the Koppelting event, an annual grassroots festival about peer production and free/libre alternatives for society, in Amersfoort, The Netherlands. In connection with the citizen science project Meet je stad!, we wanted to investigate what we could measure higher up in the atmosphere using simple and cheap sensors. Additionally, we tested a new bridge between The Things Network and habhub.org - a high-altitude balloon tracking site - written by Bertrik Sikken (TTNHABBridge).

LoRaWAN pt.7
Airtime calculation

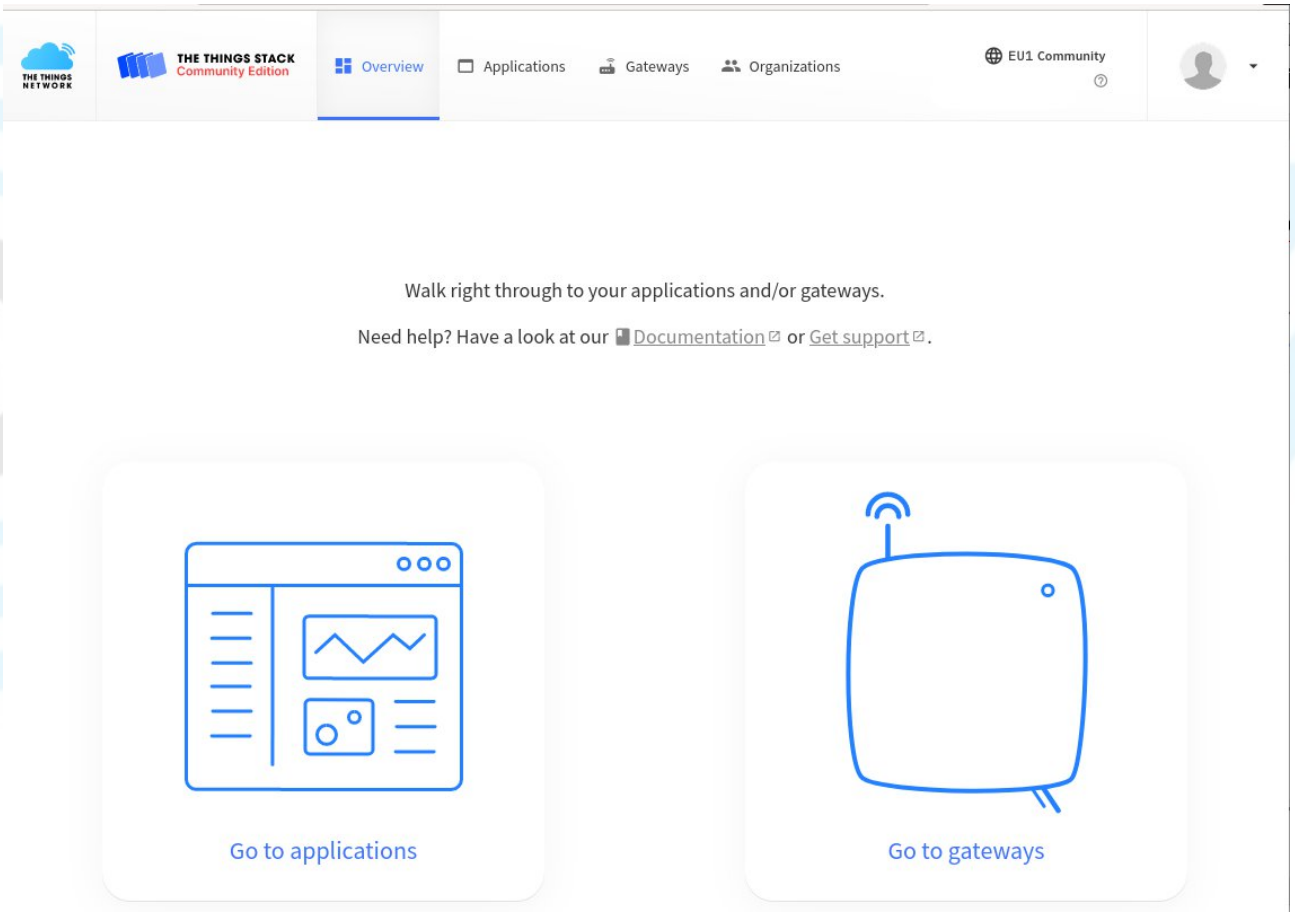
https://avbentem.github.io/airtime-calculator/ttn/eu868



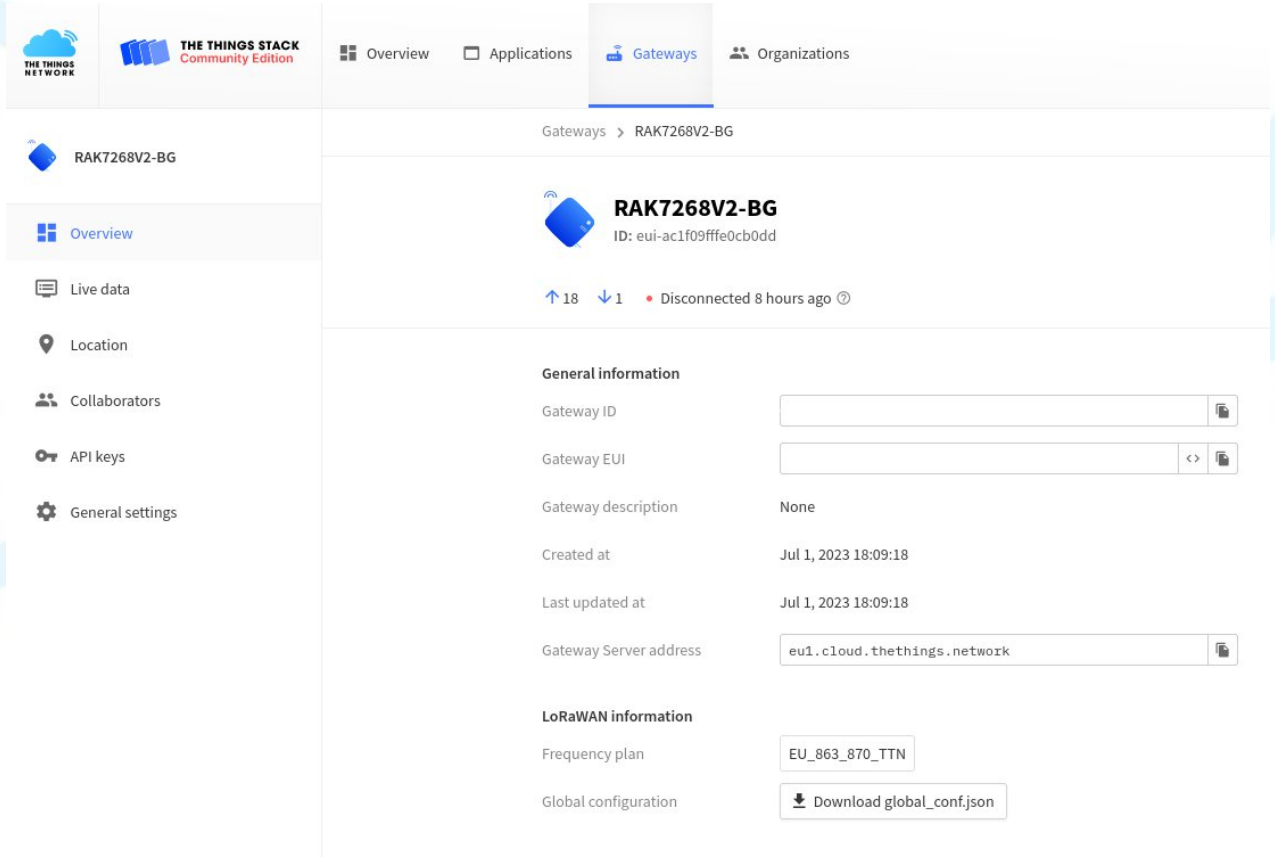
	DR6	DR5	DR4	DR3	DR2	DR1	DR0
data rate	SF7 BW 250	SF7 BW 125	SF8 BW 125	SF9 BW 125	SF10 BW 125	SF11 BW 125	SF12 BW 125
airtime	30.8 ms	61.7 ms	113.2 ms	205.8 ms	411.6 ms	823.3 ms	1,482.8 ms
1% max duty cycle	3.1 sec 1,167 msg /hour	6.2 sec 583 msg /hour	11.3 sec 318 msg /hour	20.6 sec 174 msg /hour	41.2 sec 87 msg /hour	82.3 sec 43 msg /hour	148.3 sec 24 msg /hour
fair access policy	88.8 sec (avg) **40.5 avg /hour** 972 msg /24h	177.7 sec (avg) **20.3 avg /hour** 486 msg /24h	325.9 sec (avg) **11.0 avg /hour** 265 msg /24h	592.8 sec (avg) **6.1 avg /hour** 145 msg /24h	1,185.5 sec (avg) **3.0 avg /hour** 72 msg /24h	2,371.1 sec (avg) **1.5 avg /hour** 36 msg /24h	4,270.3 sec (avg) **0.8 avg /hour** 20 msg /24h

LoRaWAN Hardware

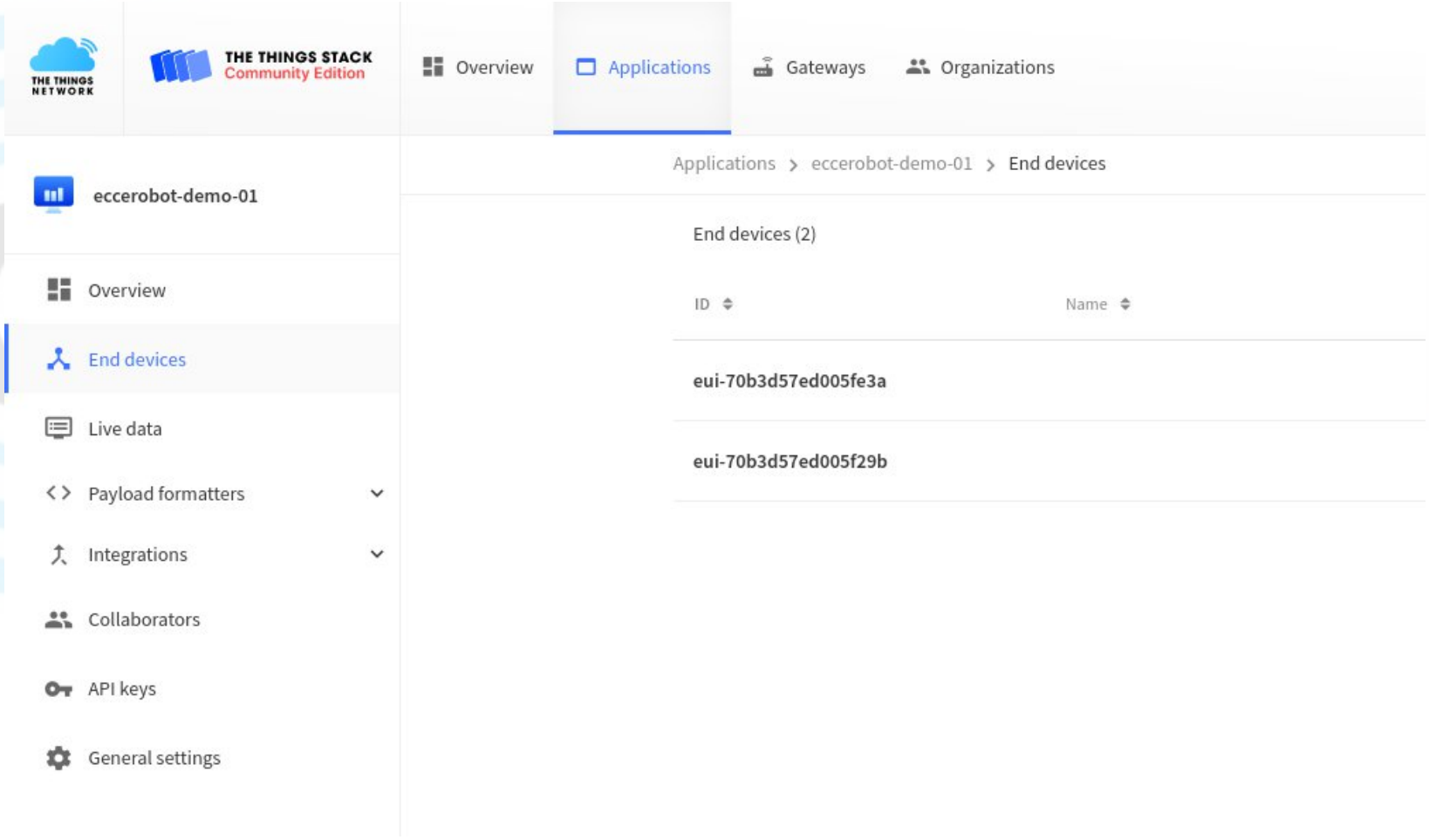
LoRaWAN Network Server



LoRaWAN
Network Server - Gateway



LoRaWAN
Network Server – Application



LoRaWAN Application Server

The Application Server hosts an MQTT broker.

End user can send MQTT subscriber requests.

Do you really need LoRaWAN?

Is there any kind of alternative?

What if the RX device could also forward received information to the web?

End of Arduino Training