Near Space Arduino Course

Marc Cortés i Fargas

ESEIAAT, Technical University of Catalonia – **BarcelonaTech**

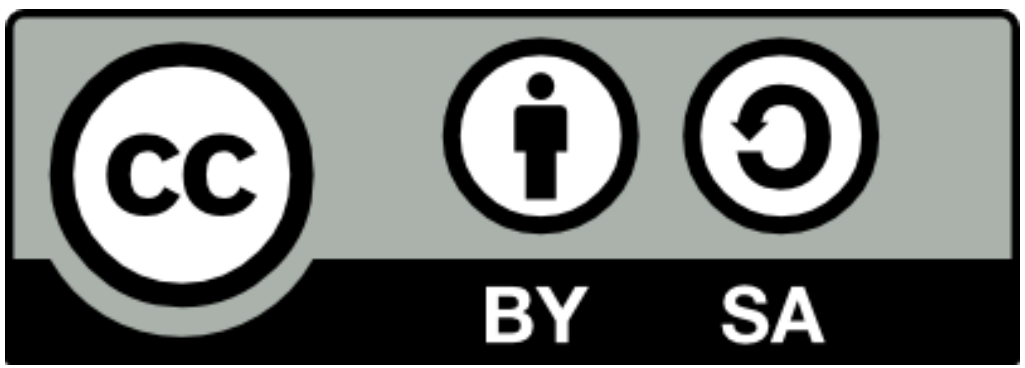
marc.cortes.fargas@upcprogram.space

March 19, 2017

Navigation

You can use sections at the top of the pages to navigate though this document. Also, everything in **bolt text** is a link to an external page.

License



Images of this course

All schemes of this course are drawn by using **Fritzing**.
Also, all the third-images and videos attached are used in a non-commercial way.

Overview

Lecturer background

Work experience

- Current – **Zero2Infinity** - Space Electronics Department
- Jun '15 - Sep '16 – **Bound4Blue SL**. Mechanic and electronics integration

Academics

- Current – MSc in Aeronautical Engineering
- Sep '16 – BSc in Aerospace Vehicle Design

Lecturer background

Projects

- '14-Current – **UPC Space Program** – HAB missions
- Specialized in Electronics and programming

Projects

- Cluster server with microcomputers – **Projecte Hermes**

Motivation

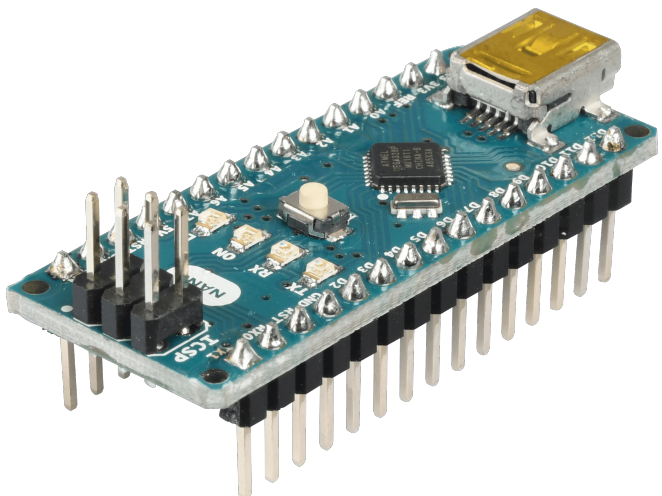
Figure: Photo taken in a HAB mission past May '16. Altitude: 30km (100000 ft) *Arduino* was the main platform to develop telemetry used in this kind of missions.

Contents of the course

Contents

1. Digital signals (IN and OUT) – Pulldowns, Arduino basics, Analog signals – A/D, PWM. *22/03/2017*
2. Barometer/Termometer (I2C comms), Servomotors, Battery dimensioning. *26/04/2017*
3. *Bluetooth* issues (Serial comms.), Relay, Switch on a lamp using a mobile phone ~~(only on *Android*)~~ *10/05/2017*
4. GPS main sentences. Filtering and main parameters. Communication through **Iridium** satellites. *24/05/2017*

Figure: Arduino nano.

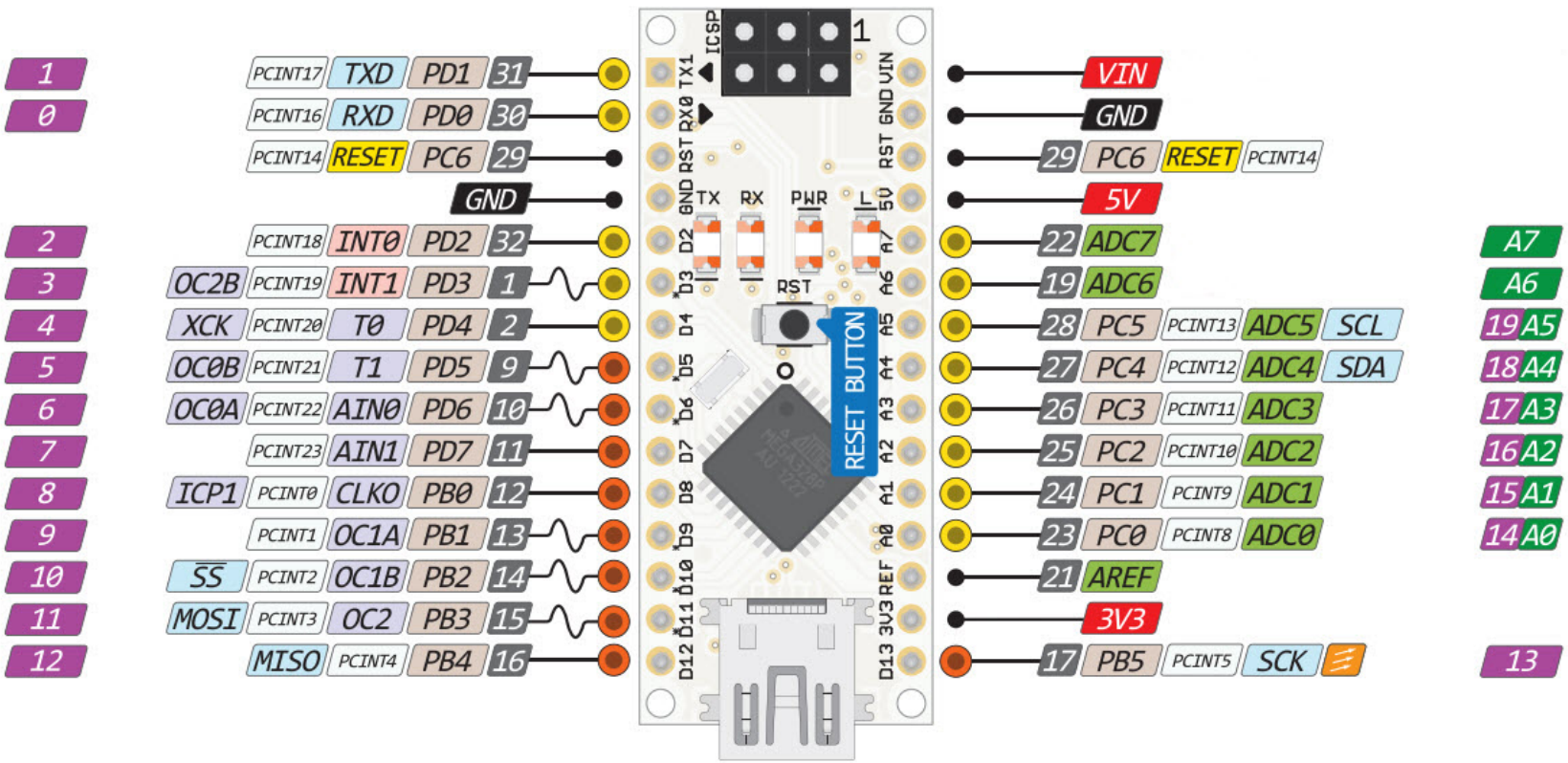


What is an Arduino?

- Microcontroller development platform.
- Open Source mind.
- Low cost (2EUR from China)
- Different versions
  1. Nano
  2. Mega
  3. Pro Mini
  4. Uno
  5. Etc.
- User friendly programming

Arduino NANO pin layout

Figure: Arduino Nano with pin Layout.



Arduino characteristics

- Read/Write Digital Values
- Read Analog Values
- PWM (Pulse Width Modulation)
- I2C
- SPI
- Serial

Arduino IDE (Integrated Development Environment)

Debian based GNU/Linux distributions (such as *Ubuntu)*

- sudo apt install arduino arduino-core

Windows and Mac

- Go **here** and download and install last Arduino IDE

Other Linux distributions

- If you use other GNU/Linux distributions you can compile source code from IDE's site.

Pro tips

What can I do with Arduino's IDE?

- Edit code.
- Compile code.
- Upload code to Arduino.
- Communicate with Arduino with Serial Monitor.

Arduino's IDE is not the only editor/compiler/uploader software. Feel free to use other text editors such as *Sublime Text* and *Atom* among others.

Driver problems in Windows and Mac

If you use Windows and Mac, you may find some problems when trying to communicate with your Arduino.
For Windows, install **this** (/CH341SER/SETUP.exe).
For MAC, I've been told **this** works.

Nevertheless, I strongly recommend you to use a GNU/Linux distribution if you will be longer involved in electronics and software development.

IDE characteristics

FIRST, connect your Arduino using a USB wire.

- *Target/Board:* The Arduino you are using.
- *Port:* Port of Serial communications Arduino is using.

Board

Go to "Tools", "Board" and select your Arduino

Port

Go to "Tools", "Port" and select Arduino's port. Usually only one port is available.

Digital electronics

Figure: On/Off international symbol

Could any of you explain me this symbol?

Blink a led

Go to */File/examples/1.Basics/Blink*. It will blink a LED in your Arduino.

Figure: Blink script in the Arduinos'IDE

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

First improvement

Use a handy external *integer (int)* value to change LED's blink velocity.

Figure: External delay time declaration

```
int led = 13;

int waitforit = 182;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(waitforit);              // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(waitforit);              // wait for a second
}
```

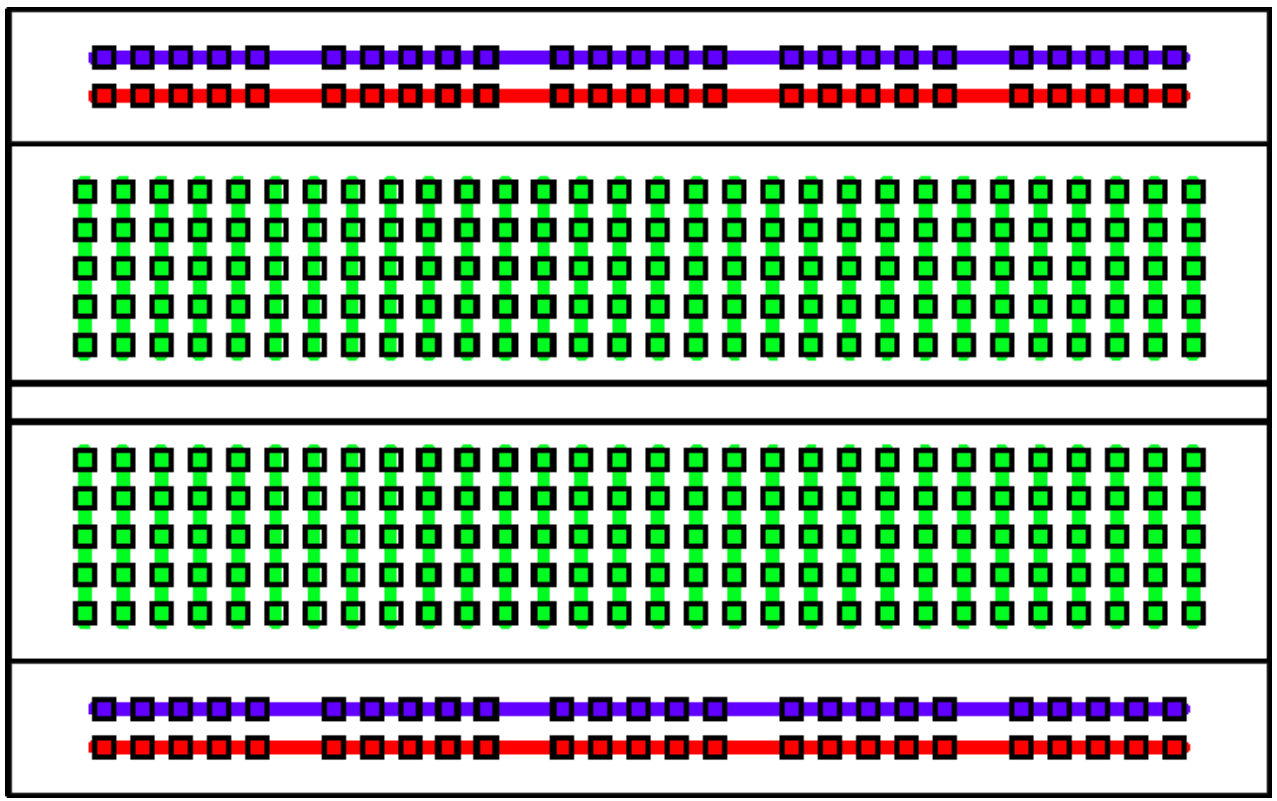
Second improvement

Figure: External function. Does it need to be a *function* or an *action?*

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  waitforit(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  waiforit(1000);               // wait for a second
}



int waitforit(int espera){
  delay(espera);
  }
```

Let's met your new best friend, the BreadBoard!

Figure: BreadBoard Schematic

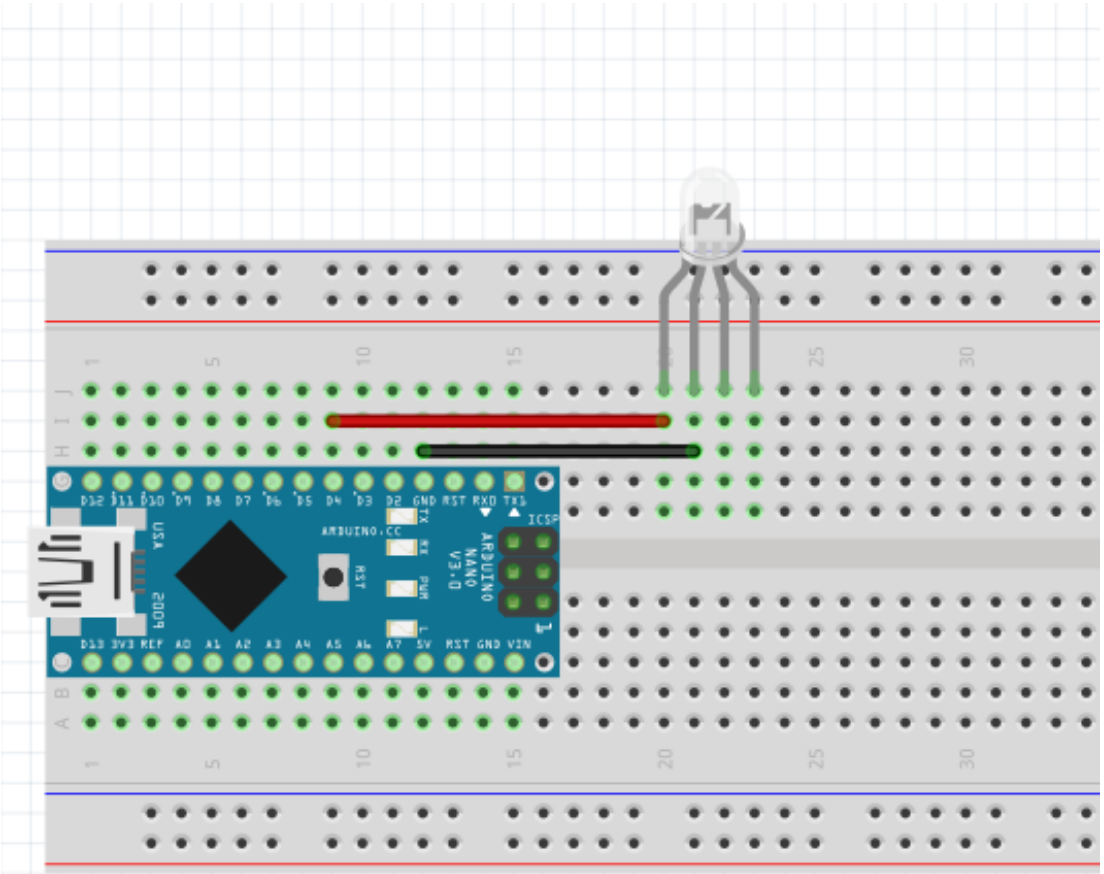


First exercise

Detect when you are not able to see differences when turning on and off a LED continuously. See difference between different colors (R, G, B).
OUTPUT: frequency which humans are not able to see highs and lows on LED.
PLEASE, use a sheet of paper to cover your LED in order to avoid headaches due to continuous lighting.

First exercise

Figure: First exercise schematic



First exercise

Let's see a cool application of that:

First exercise

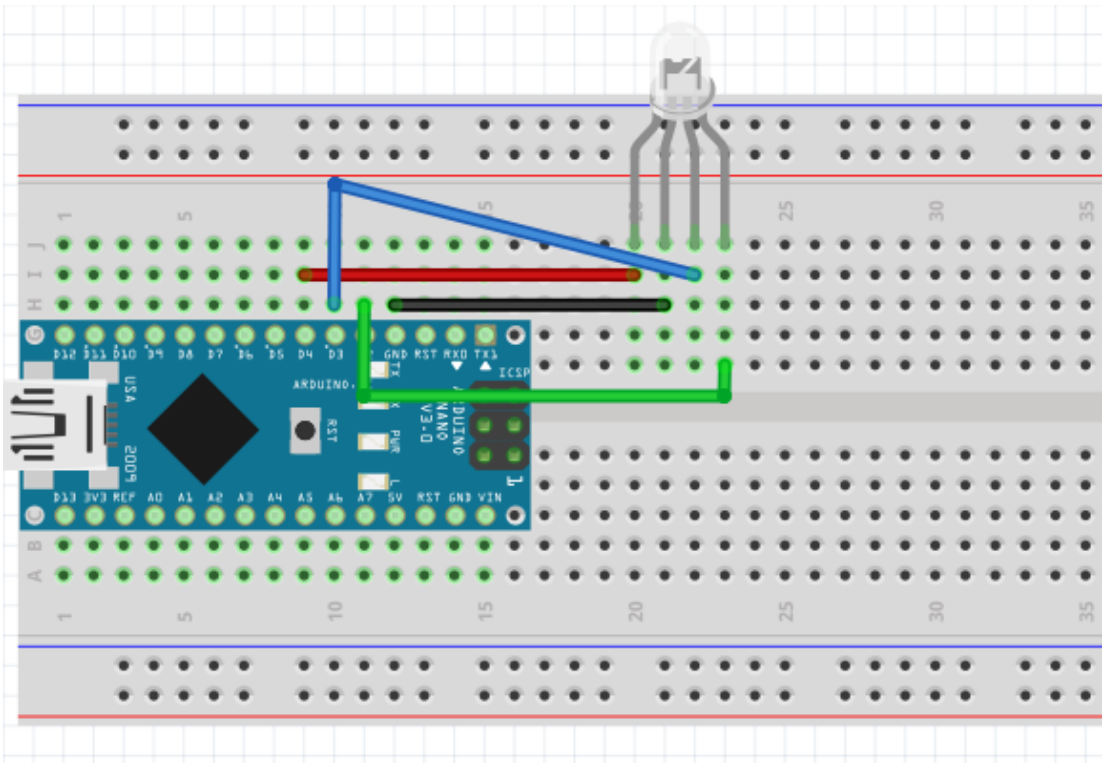
If you can't see this video, please go **here**. *To see the explanation go* **here**. *Of course, you can do this experiment with your Arduino! please ask me if you want to build a "levitating water" experiment.*

Second Exercise

Detect when you are not able to see differences when turning on and off a RGB LED: when you are not capable to see different colours.
OUTPUT: frequency which humans are not able to see highs and lows on LED.
PLEASE, use a sheet of paper to cover your LED in order to avoid headaches due to continuous lighting.

Second Exercise

Figure: Second exercise schematic



Reading digital values

New functions:

Declare INPUT pin

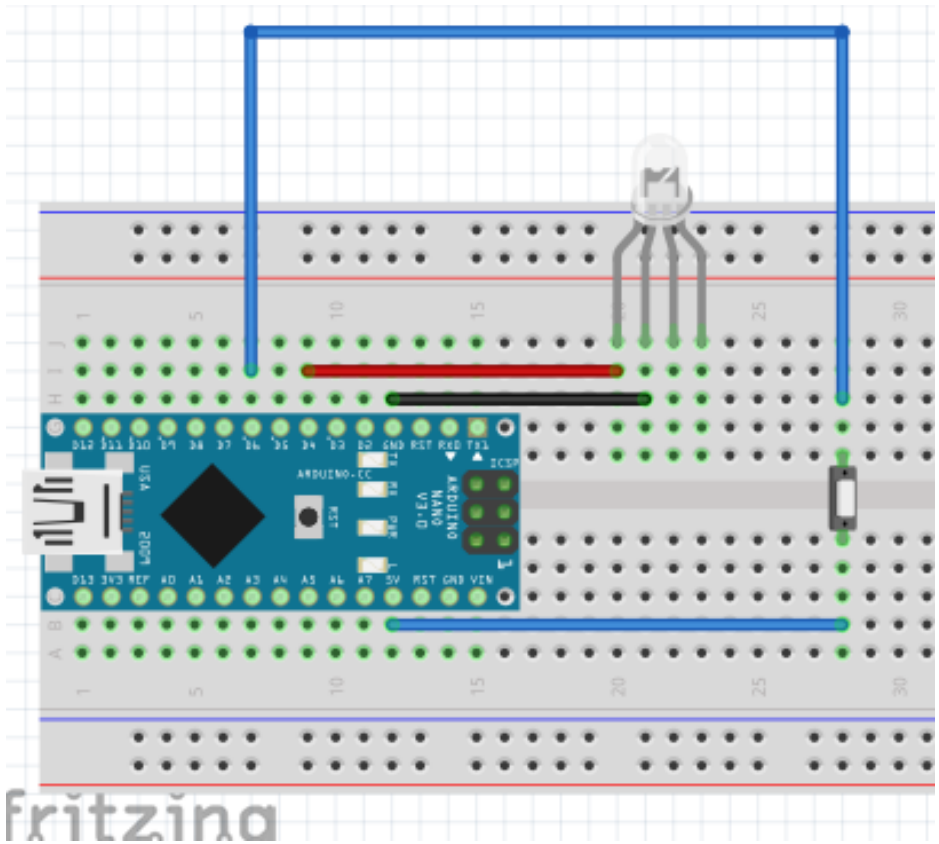
Use function *pinMode(pin,INPUT/OUTPUT)* to declare an input pin of your preference.

Read digital values with your declared pin

Use function *digitalRead(inputPin)* to know the status of a pin. This will return you an integer value (0 or 1, that means LOW or HIGH)

Read a digital value from a pin and Arduino's 5V. When you read a HIGH, switch on a LED. Use a I/0 button and the following schematic.

Figure: Third exercise schematic



Third Exercise code

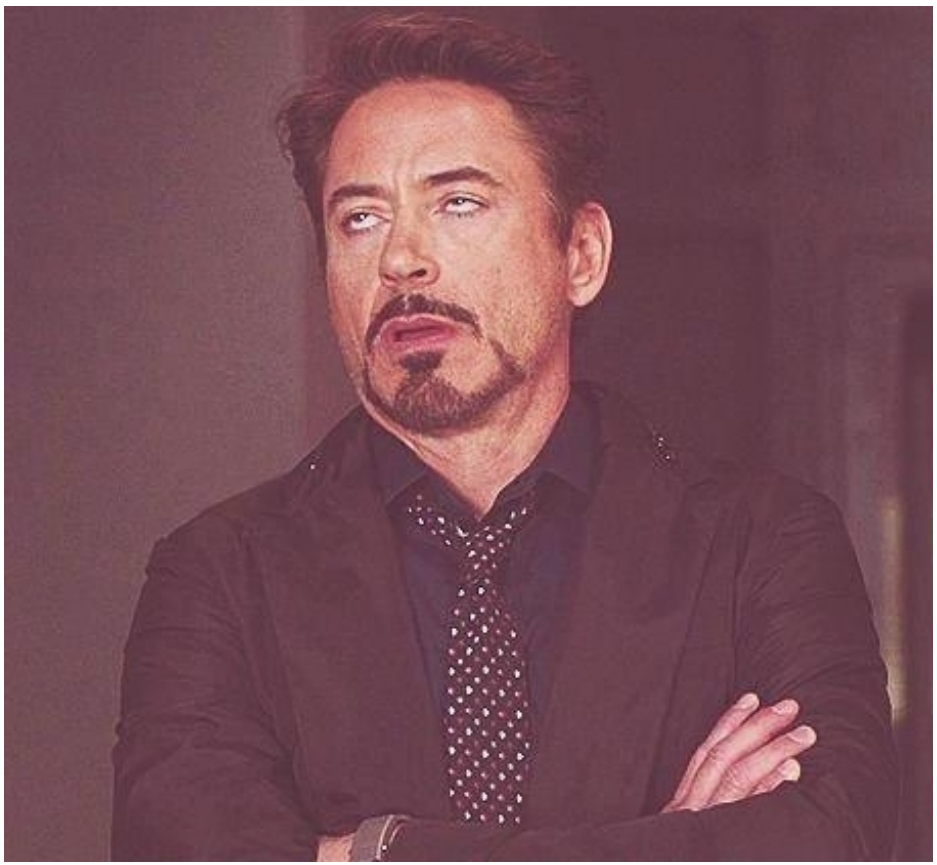
```
int ledPin = 4; // choose the pin for the LED
int inputPin = 6; // choose the input pin (for a pushbutton)
int val = 0; // variable for reading the pin status
void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin, INPUT); // declare pushbutton as input
}
```

Third Exercise code

```
void loop(){
val = digitalRead(inputPin); // read input value
if (val == HIGH) { // check if the input is HIGH
digitalWrite(ledPin, LOW); // turn LED ON
} else {
digitalWrite(ledPin, HIGH); // turn LED OFF
}
}
```

Why your experiment is not working properly?

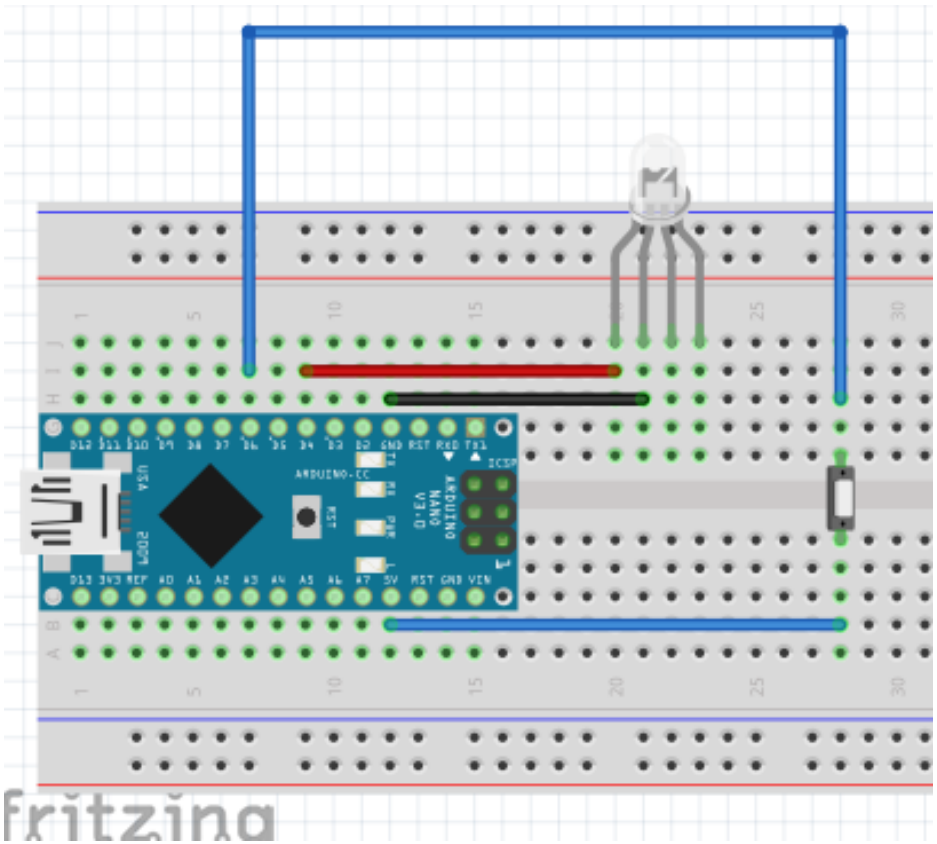
Figure: Even Tony Stark sometimes gets stuck with his code.



Why your experiment is not working properly?

Let's take a closer look...

Figure: Third exercise schematic

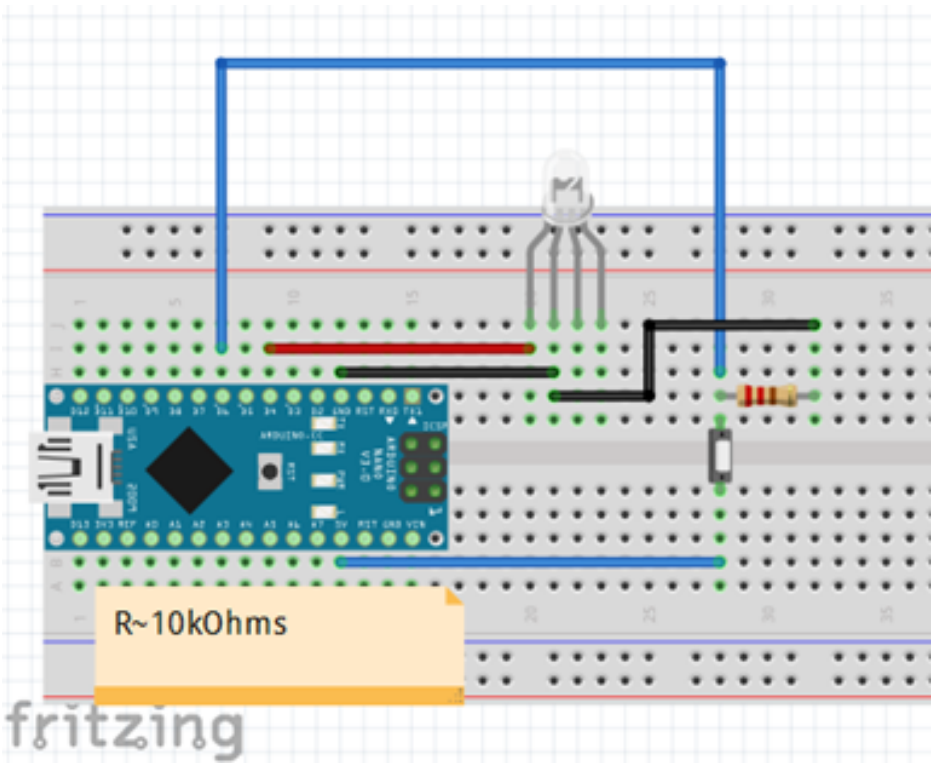


The importance of pulldowns

Your "reader" digital pin is not connected to any voltage level when button is not pressed. It is only connected to the breadboard, and that means it has not a clear level defined. In this condition any noise signal, such a hit on your breadboard, can make your Arduino to read a non-realistic voltage level.
We need to ensure 2 voltage levels: low and high, or ground (GND, 0VDC) and signal (5VDC in this case).
To do this, we use a Pulldown resistor.
Pulldown resistor must be used *always* in the case where we need to ensure 2 clear logic levels (i.e., at the gate of a MOSFET)

Pulldown

Figure: Third exercise schematic with a pulldown resistor



PWM

PWM

PWM aims for Pulse Width Modulation. It means we change voltage level of a pin between high and low in a certain frequency.

AnalogWrite

We use *AnalogWrite(pin, value)* function to write a PWM signal in a PWM. Please notice not all pins can be used for PWM. Pins with a sinus wave in subsection **Arduino's nano layout**

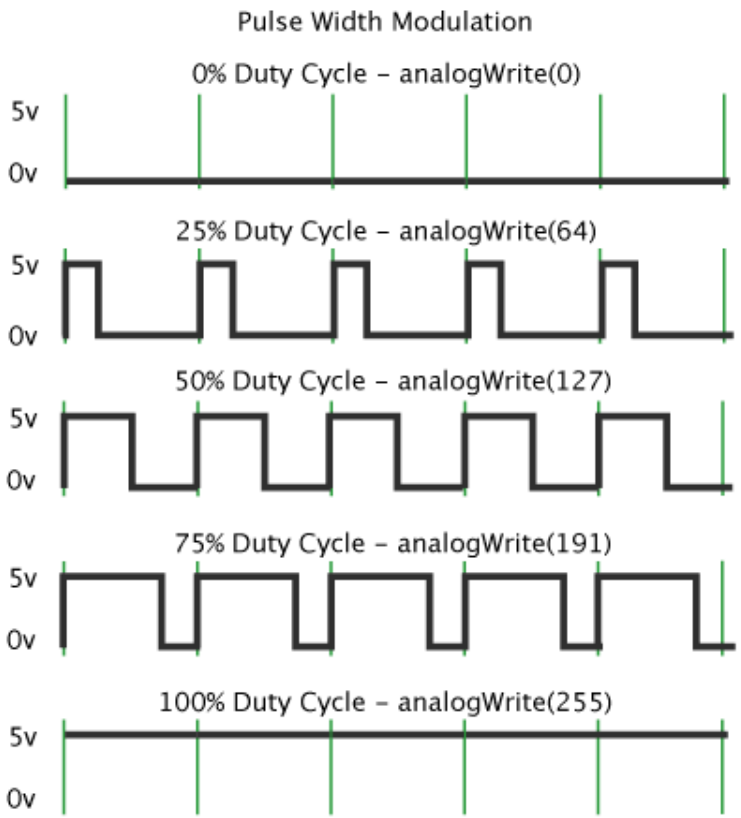
PWM

AnalogWite value

Arduino uses an PWM generator of 8-bit resolution. $2^8 = 256$, so we will have a value between 0-255. 255 means full-time high (aka 100% duty cycle) and 0 full-time low (aka 0% duty cycle). Following this fabulous linear concept, 127 means half-time on and half-time off, aka 50% duty cycle. Medium voltage will be $V_peak \cdot DC$.

PWM scheme

Figure: PWM scheme



PWM issues

PWM uses

PWM us used for:

- Dimming a LED
- Control a DC motor
- Drive servomotors
- Power Delivery (buck converter)
- Signal modulation

PWM issues

Although there is medium voltage of $V_{peack} \cdot DC$, this cannot be used to power other applications. There is not a real output of Medium Voltage, just a high-low signal that can emulate this analogue signal. To have a real medium voltage level, it is necessary a filter (RC filter) at the output of PWM to ensure our load will see only medium voltage level.

Dimming a led

Dimming a RGB LED in order to change brightness of a single colour.
Tip: you may need to use a "for" loop. if you want to change the brightness automatically.

Dimming a led and viewing a PWM signal

1. Use crocodile wire to connect PWM output to the oscilloscope. Red to PWM pin and black to arduino's GND.
2. Start Oscilloscope and press autoscale and autoset. Use different AnalogWrite values (from 0 to 255) to see different PWM signals.
3. You can use oscilloscope to get data as Duty Cycle. You can ensure that using a 127 argument input in your AnalogWrite Arduino's function it will be a 50% DC.

Potentiometer vs PWM

To dimming a led or controlling a fan, what would you use, a simply and reliable potentiometer or a PWM?

Figure: Homer does not know the difference.

Dimming a RGB led

Dimming a RGB LED in order to change brightness of all colours, intersecting each colour with its precedent (i.e., red, green blue). You will have full colour spectre!
Send your code *before* previous Sunday of following class. It is not mandatory but necessary to have your certificate with honours.

References

Jeremy Blum

Exploring Arduino

Ed. Wiley

Robert Boylestad et al

lectronic devices and circuit theory

Ed. PHI, 1982