Near Space Arduino Course

Marc Cortés i Fargas

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

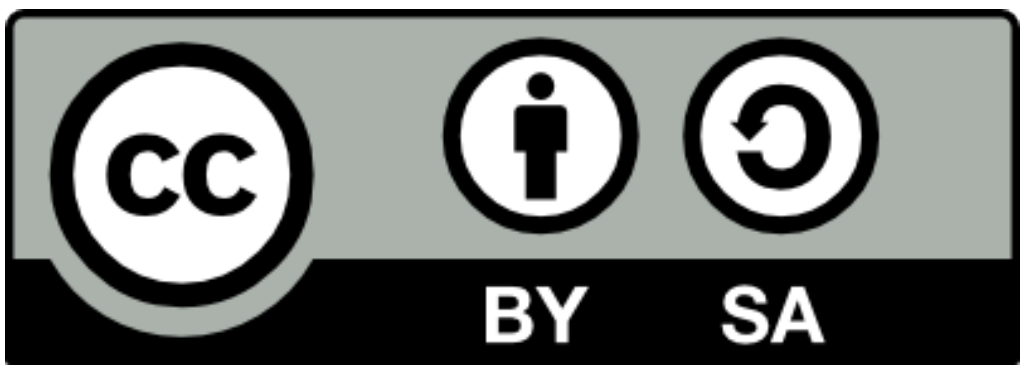
marc.cortes.fargas@upcprogram.space

April 22, 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.

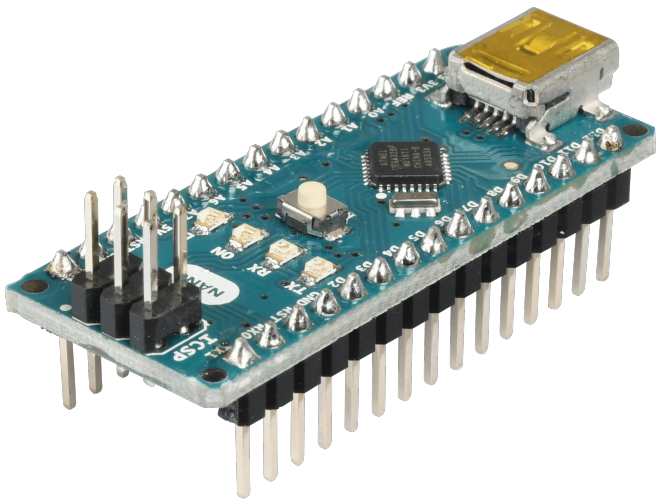
Overview

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),~~ ADC, Servomotors, Battery dimensioning. *26/04/2017*
3. *Bluetooth* issues (Serial comms.), Relay, Switch on a lamp using a mobile phone ~~(only on *Android*)~~ I2C comms. BMP180 *10/05/2017*
4. GPS main sentences. Filtering and main parameters. Communication through **Iridium** satellites. *24/05/2017*

Figure: Arduino nano.



Today's contents

1. ADC (Analog to Digital converters) Basics
2. ADC in Arduino
3. Example. LM35 sensor.
4. Exercise 1 and 2
5. Arduino's libraries importing
6. Servos basics
7. Servos controlling and PWM visualisation
8. Exercise 3
9. Battery dimensioning

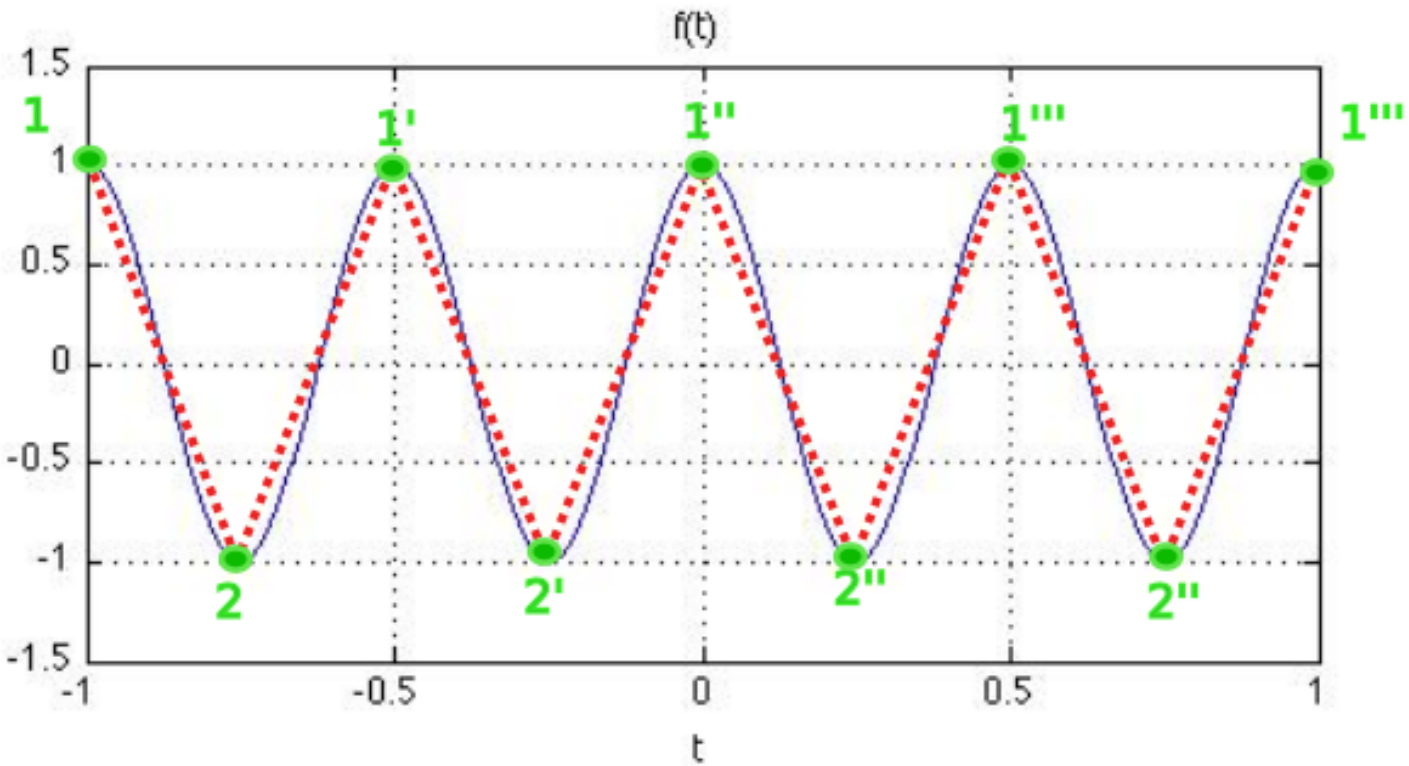
Analog to digital conversion

Constrain I - Sampling rate

Sampling rate must be at least twice greater than the bandwidth of the signal: *Nyquist law.*
If we want to reconstruct an analog signal, let's see if we have only a sample rate twice greater than the bandwidth.

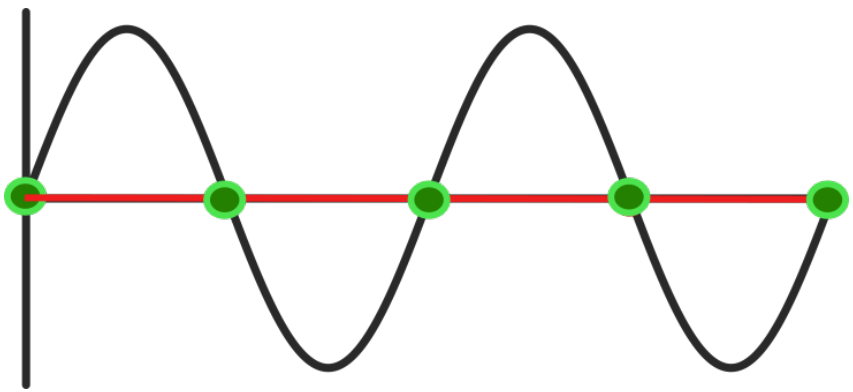
Analog to digital conversion

Figure: In green, number of samples per period. In blue, original signal. In red, reconstructed signal.



Analog to digital conversion

Figure: Reconstructed signal at same sampling conditions. As you can see, we will reconstruct 0VDC.



Constrains

As a rule of thumb and in Arduino applications, it is recommended a sampling frequency 10 times greater.

Resolution

Once we have a proper sampling rate, we have another big big concerning:

Constrain II - Resolution

Each sample corresponds to a voltage level. Resolution means having different voltage level. Mores resolution $\rightarrow$ more voltage levels $\rightarrow$ better sample.

Resolution

Resolution

I.e., we have a signal of 3V3. We have a reference voltage of 5V so we can have different voltage levels between 0V and 5V. If we have, for an instance 4 volts level that means we have 1.25 Volt step.
For our sample, we have 2 near values: 2.5V and 3.75V.
We measure the nearest value. Thus, we will measure 3.75V instead of 3V3.
As a conclusion: more resolution $\rightarrow$ more voltage values $\rightarrow$ more accuracy $\rightarrow$ better sample.

Resolution

Resolution in your Arduino

Resolution is measured with "bits". The number of voltage levels is defined by: 2^{bits}. Your Arduino has 10-bit resolution, that means 1024 voltage levels (0-1023) values.
Each value correspond to a 10 bits number: 1111111111 means 1023 (full reference voltage) and 0000000000 means 0 (GND).

I strongly recommend you to go **here** to know the ADC of your Arduino.

To sum up...

Two main parameters: *sample rate* and *resolution.*. Whereas sample rate is important in high-frequency applications (i.e., to measure magnitudes in a rocket) resolution is important depending on the application.
Usually, a high-velocity ADC (an ADC with high sampling rate) comes with low resolution **like this 500Msps 8-bit resolution ADC.** Both high resolution and sample rate means higher price.

Temperature sensor LM35

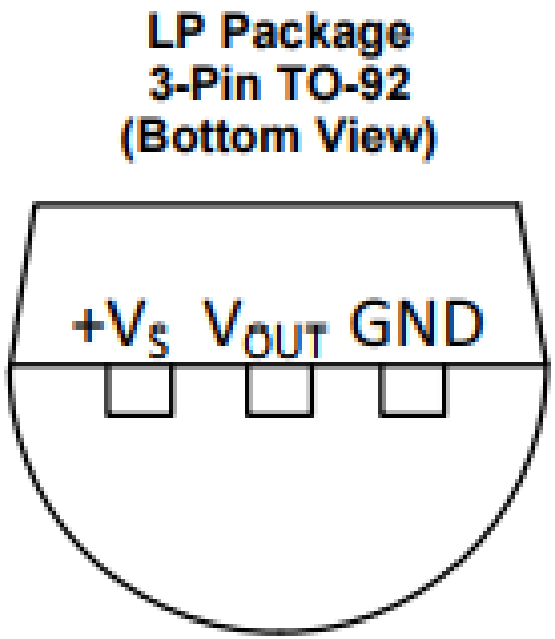
First at all, go **here** to download LM35 *datasheet*. Datasheet are documents where all the information concerning components are explained as well as common applications.

From LM35 datasheet

1. Uses: basic temperature sensor vs full-range temperatures sensor. We will use Basic temperature sensor.
2. Footprint: parts of the sensor identification.
3. Component information
4. LM35 Transfer function (7.3.1):
   $V_{out} = 10mV/degC \cdot Temp[degC]$

Temperature sensor LM35

Figure: Your LM35 footprint. Please notice LM35 comes with different configurations.



Temperature sensor LM35

From LM35 transfer function:

$$V_{out} = 10mV/degC \cdot Temp[degC]$$

We take Temp[degC]:

$$Temp[degC] = V_{out}/10mV/degC$$

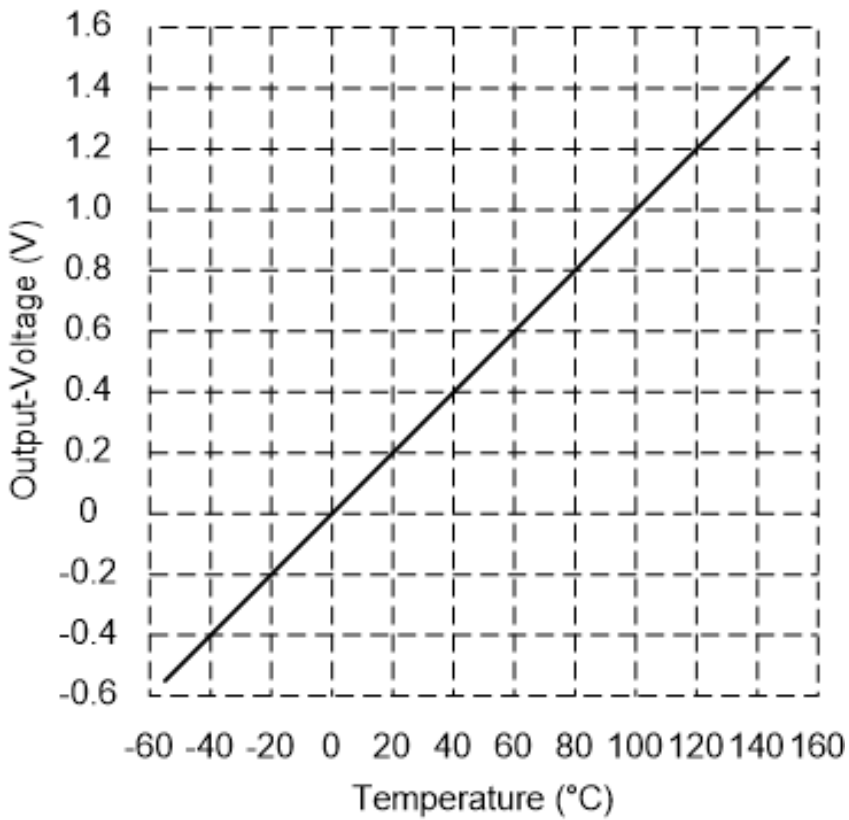
And

$$\frac{degC}{10mV}$$

Is the slope of our function and V_{out} is in mV.

Temperature sensor LM35

Figure: LM35 Graphic



Exercise 1

Measure temperature of your LM35.

Steps

1. Connect your LM35 following schematic from **its footprint**
2. Connect V_{out} to one of the analog pins of your Arduino.
3. Use function *AnalogRead(analogPin)* to read an analog value.
4. You will read a number between 1023 and 0 (according to your Arduino's resolution!)
5. Convert this number to voltage level in mV. Remember, 1023 means 5000mV!
6. Once you have output voltage in mV, use **transfer function** to get temperature in deg C.

Exercise 2

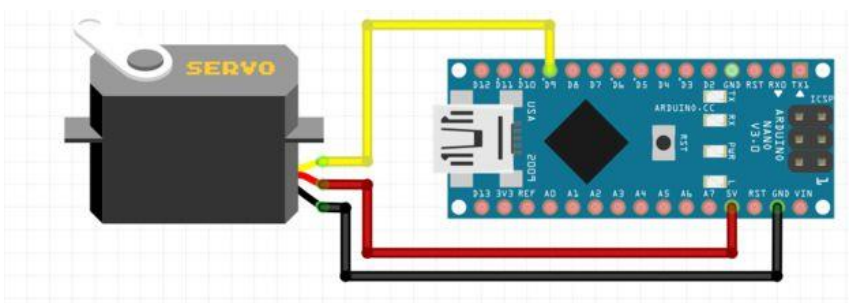
Use a PWM output and your RGB and your own body temperature to change colour from blue to red of a LED by using your LM35 sensor.

That's homework btw ☺

Servomotor controlling

PWM signals are widely used in electronics world. And in RC applications, PWM signal is used to control Servomotors. Servomotors are devices used to deflect control surfaces in RC planes and drones, such as flaps, elevator and ailerons.
Load ./Examples/Servo/sweep and play with delay (never lower than 5ms)

Figure: Connect your servo with the following wiring



Servomotor and PWM

Visualise the PWM signal of your servomotor in the oscilloscope.
Realise how by changing the duty cycle you will change the position of the servomotor.
Note: To use a PWM signal is not common the use of an Arduino: it is not necessary to have a controller to generate a PWM signal.
The vastly and famous PWM generator is a **NE555**

Generalities of batteries

Battery

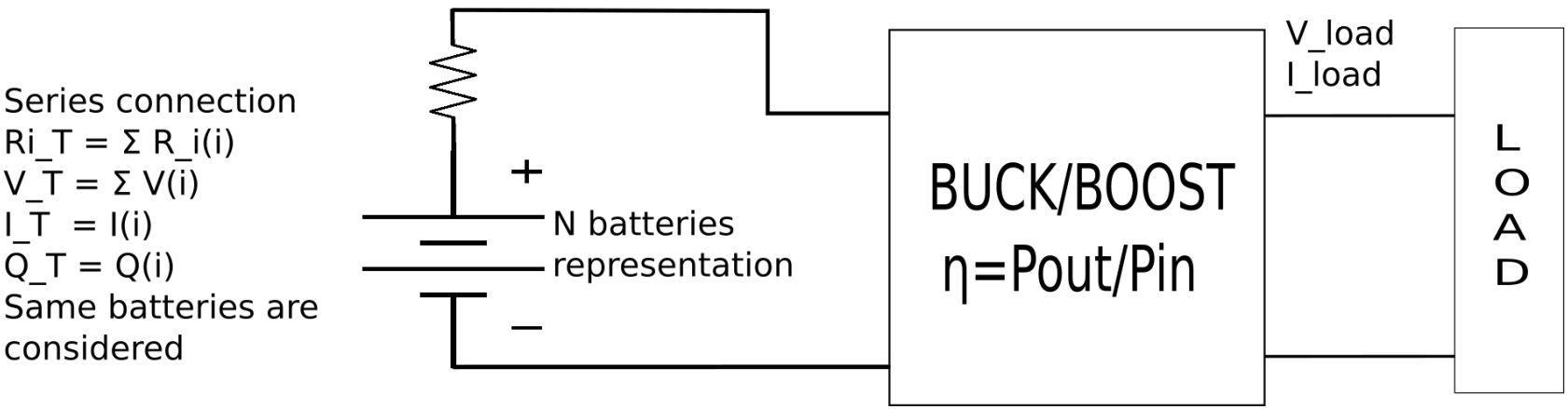
1. P_{load}: is the power consumed by the load. $P_{load} = V_{load} \cdot I_{load}$
2. Regulator (i.e., a Buck/Boost): power converter. It increase/decrease input voltage to a consign voltage. $\eta = \frac{P_{out}}{P_{in}}$. In this case $P_{load} = P_{out}$
3. $P_{battery}$: Power delivered by the batteries.
4. R_i: Internal resistance of the batteries.
5. Q: capacity of the batteries. It is usually given in [mA·h].
6. Time of operability is: $t[h] = \frac{Q[mAh]}{I[mA]}$
7. I_{bat} must not exceed maximum intensity given by the battery, I_{max}. Batteries have a fixed voltage and the intensity they delivery depends on the load they are feeding.

Generalities of batteries

We usually need a voltage converters because our load needs a certain voltage to work. I. e., Arduino 5V entry level needs, surprising, 5V. If we input different voltage the Arduino won't work, and if we go +7V we may burn the Arduino. V_{in} pin of the Arduino handles from 7 to 12 V, and uses a linear regulator to go to 5V to get the microcontroller working.

Battery schema

Figure: Battery scheme with batteries in series configuration.



Battery Considerations

Generally:

$$P_{load} = f(operation)$$

$$V_{bat} = f(t, T)$$

$$Q_{bat} = f(T)$$

$$\eta = f(V_{in})$$

In series: $Q_{total} = Q_{singlebattery}$ and $V_{total} = \sum V_{1bat}$ and considering n batteries of the same type $V_{total} = n \cdot V_{1bat}$.
In paralel: $V_{total} = V_{singlebattery}$ and $Q_{total} = \sum Q_{1bat}$ and considering n batteries of the same type $Q_{total} = n \cdot Q_{1bat}$.

Example

Having:
$P_{load} = 1W = cte$, $R_i = 0$, $\eta = 0.4 = cte$, $n = 4$, and
$Q = 2000mAh$, Battery tipe AA in series (1.5V), get running time.
Homework: do the same having a R_i from **L91 datasheet** at
$T = 0degC$.

References

Jeremy Blum

Exploring Arduino

Ed. Wiley

Robert Boylestad et al

Electronic devices and circuit theory

Ed. PHI, 1982

Margolis, Michael

Arduino cookbook

Sebastopol, CA : O'Reilly, cop. 2012, 2nd ed.

You can find this book online **here**, only for UPC community.