Digital Potentiometer Arduino Shield

One step ahead: in this article we will review the project for an Arduino shield compatible with *Arduino Duemilanove*, *Diecimila* and *Arduino UNO R1* and *R2*.

This shield is based on **AD5206** digital potentiometer. We can find some very basic applications of the component reviewed in many blogs and sites, including the arduino.cc website "**Controlling a digital potentiometer using SPI**". Unfortunately these tutorials focus on the way you can control this chip: the **SPI** (= Serial Peripheral Interface). Also the sample Sketches – limited to very few lines of code – are very simple but also almost useless.

Also the usage suggestions are very poor: you can control digitally the audio levels of some devices. But then? It's up to you. Nothing else.

You can find the shield only at **BalearicDynamics**.
Don't forget to use the discount code **ELEKTRO1008** to get 20% off discount, the special price for Electroschematics.com users.

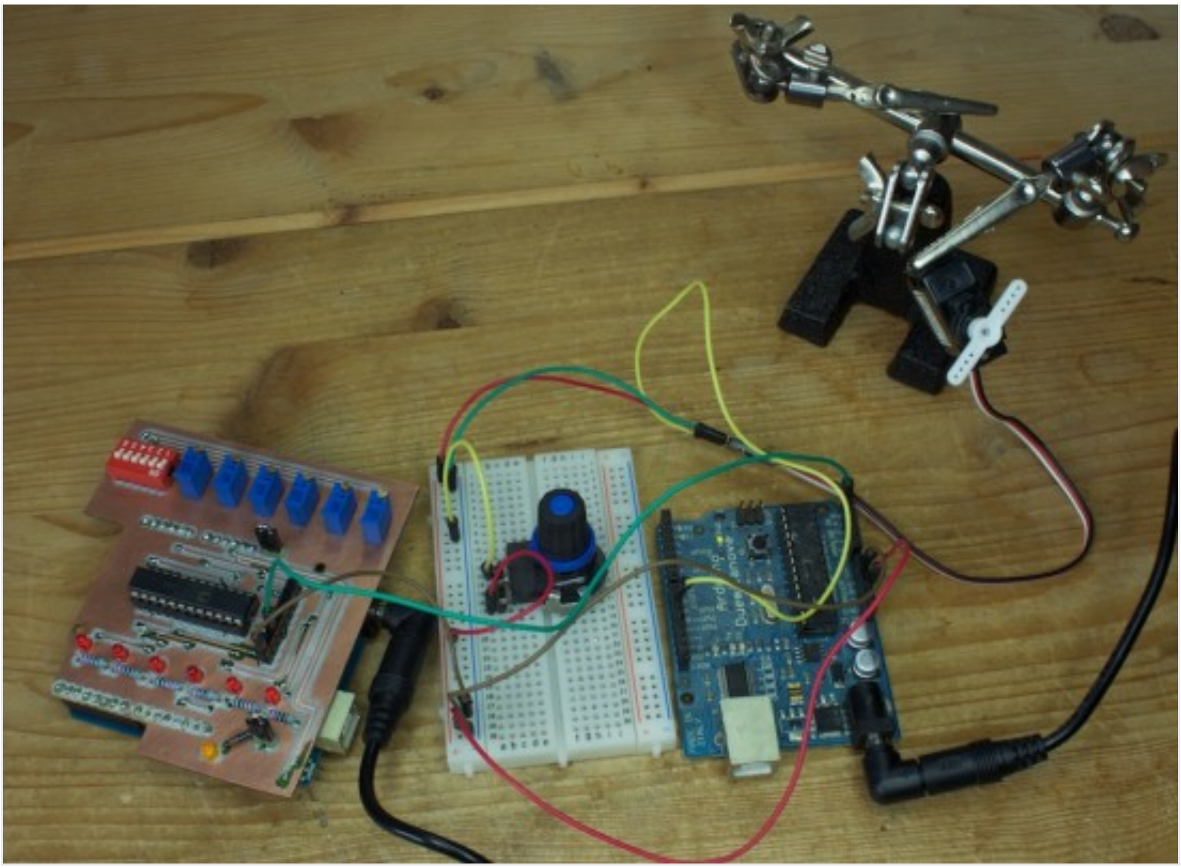
When I created *Digital Potentiometer Shield* I thought in the possible range of uses aiming to a sort of cheap "digital potentiometer lab" to carry out actual experiments and applications. So I added more components in the hope that this project will really be an inspiring source. Not necessarily for audio levels control only.

Potentiometer Controls: Analog vs Digital

Why we should use – in a real scenario – a digital potentiometer? As a matter of fact, we should use a digital potentiometer in every case we need *a variable resistance controlled by another event*. For example, one of the most common ways is changing the volume of an amplified sound depending on an external condition, e.g. the audio volume: with a sound sensor in a room we can adjust the music output level in order to compensate the room noise.

Another example may be a sensor following the movement of a subject to increase the sound level of the neartest speakers. By continuing the exploration of the world of audio controls we will discover almost unlimited possibilities.

The above examples are only feasable with one or more *digital potentiometers*.



There is also an interesting class of application cases where the *analog* potentiometer is filtered by a digital potentiometer.

Let’s suppose, for example, we are using the RC control of a car race: there are two jogs, one for the acceleration and one for the direction both moved manually in the x-y axis directions (i.e. a very simple RC radio control). We should consider many parameters while managing the two jogs, such as the limits of the RC car motors, the maximum supported acceleration, the maximum speed depending on the ground characteristics and so on.

The more experienced is the driver, better he will drive the RC car without accidents, and possibly win the race!

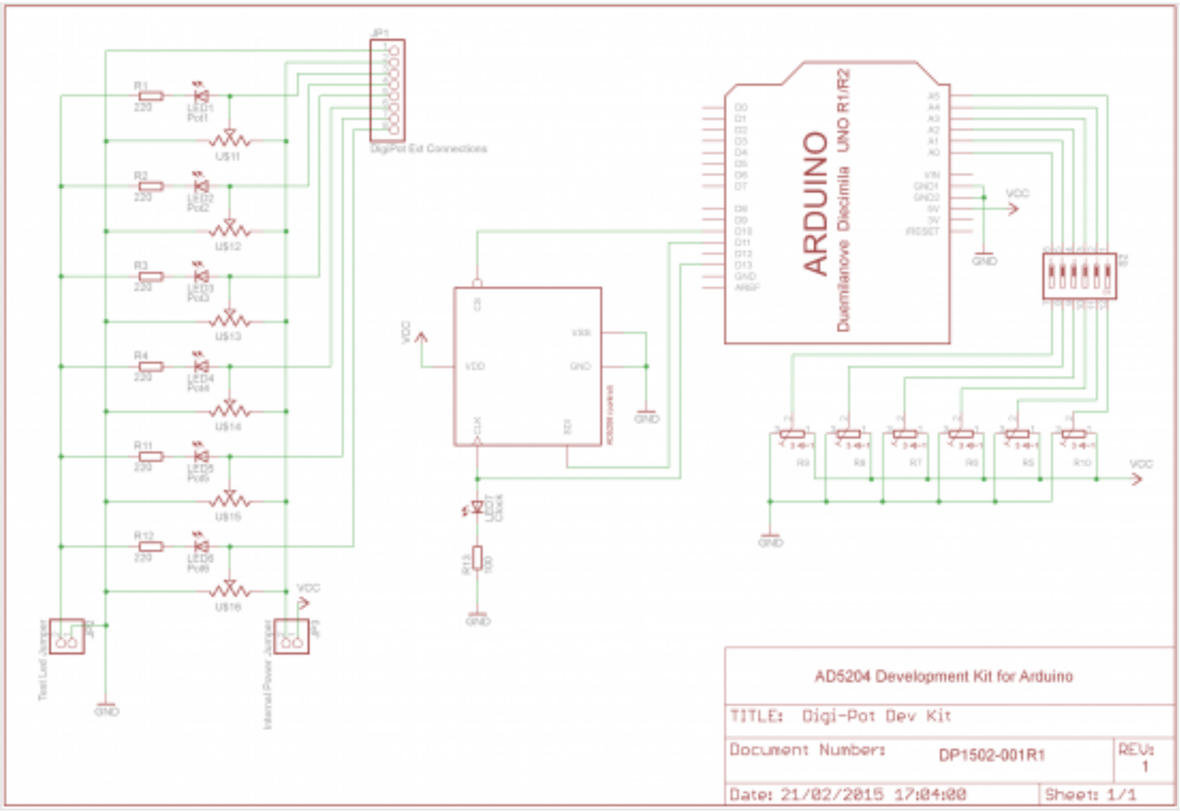
Hacking the RC Controller

Connecting the manually-controlled analog potentiometer to the analog input of the Arduino board and replacing the analog signals with digital ones on the RC board, we can dramatically change the play roles; correctly configuring the Arduino Sketch every manual input will be filtered by the microcontroller and the RC car will be definitely driven avoiding many of the the above mentioned risks.

But this is only one of the hundred cases which can benefit by a similar circuit.

The Shield Circuit

The schematics circuit connects the *AD5206 Digital Potentiomer* device to the SPI signals of the Arduino board with the Digital PIN 10 used as Chip Select.



There are 6 trimmers connected to the 6 analog PINs on the Arduino board to be used for testing purposes or to generate an analog predefined value (e.g. as a preset level for the microcontroller Sketch). The triggers can be excluded one by one through the dip switch freeing the corresponding Analog PIN for other usages.

That the SPI Clock Signal is also connected to a yellow LED: when it is blinking the communication between Arduino and the AD5206 device is working and the digital potentiometer is receiving data.

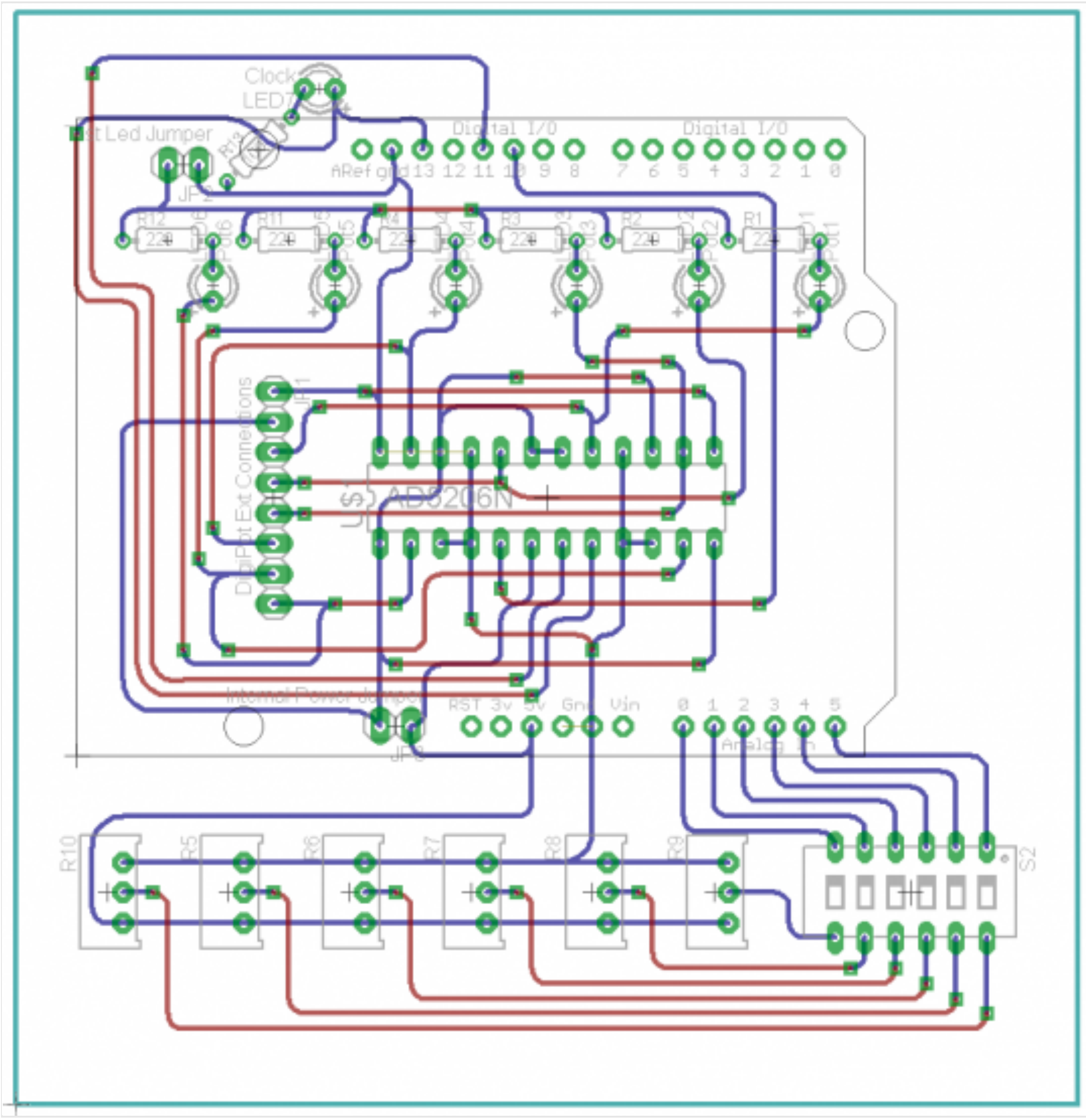
> Note that the digital potentiometer is internally latched: this means that when the circuit is powered, the device retains the last settings sent.

As the device can work at a reference VCC between 2,7 and 5,5V the jumper JP3 uses the Arduino power supply. When unplugged, an external reference votage can be used. In this case the external reference voltage should be connected to the pin 2 of the connector JP1.

> Warning: when connecting an external circuit, the shield circuit GND on pin 1 of the connector JP1 should be connected to the external circuit GND.

Connected in parallel to each potentiometer output of the SD5206 there is a red LED for testing purposes (LED1 … LED6). When connecting an external device, it is preferable to disable the test LEDs disconnecting the jumper JP2. The 6 digital potentiometers outputs are available on the pins 3-8 of the external connector JP1.

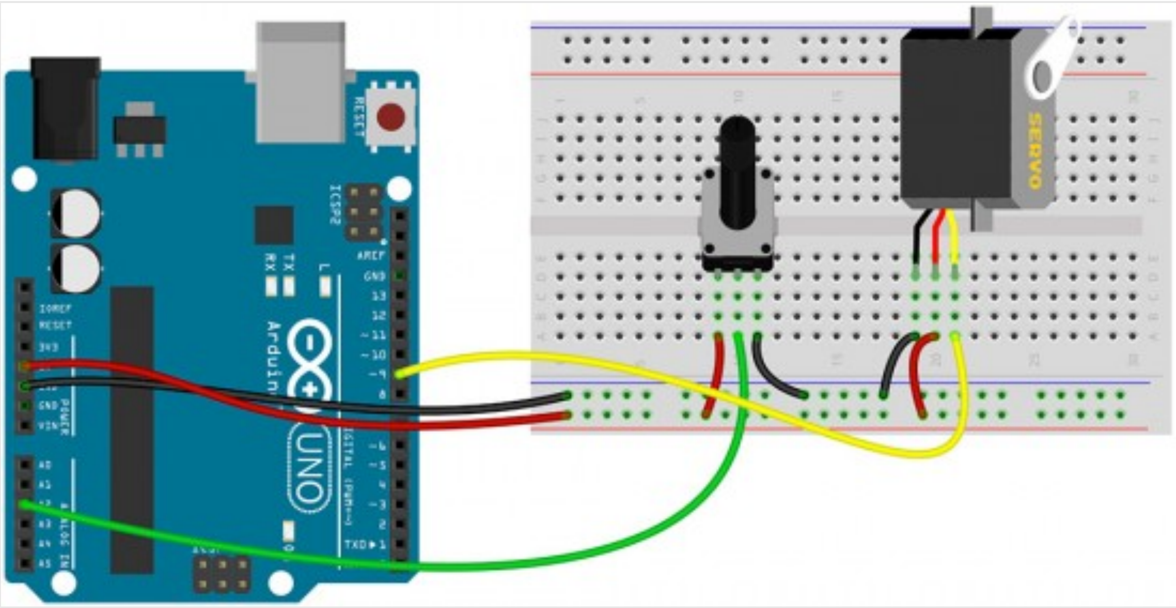
The following picture is the PCB layout of the circuit shield.



Filtering the Analog Potentiometer

The following video shows a simple Arduino Duemilanove moving a servo with an analog potentiometer. The stability is dramatically improved as we connect the digital potentiometer Shield.

The simple circuit used to control the servo with the Arduino Duemilanove does not need special explanations.



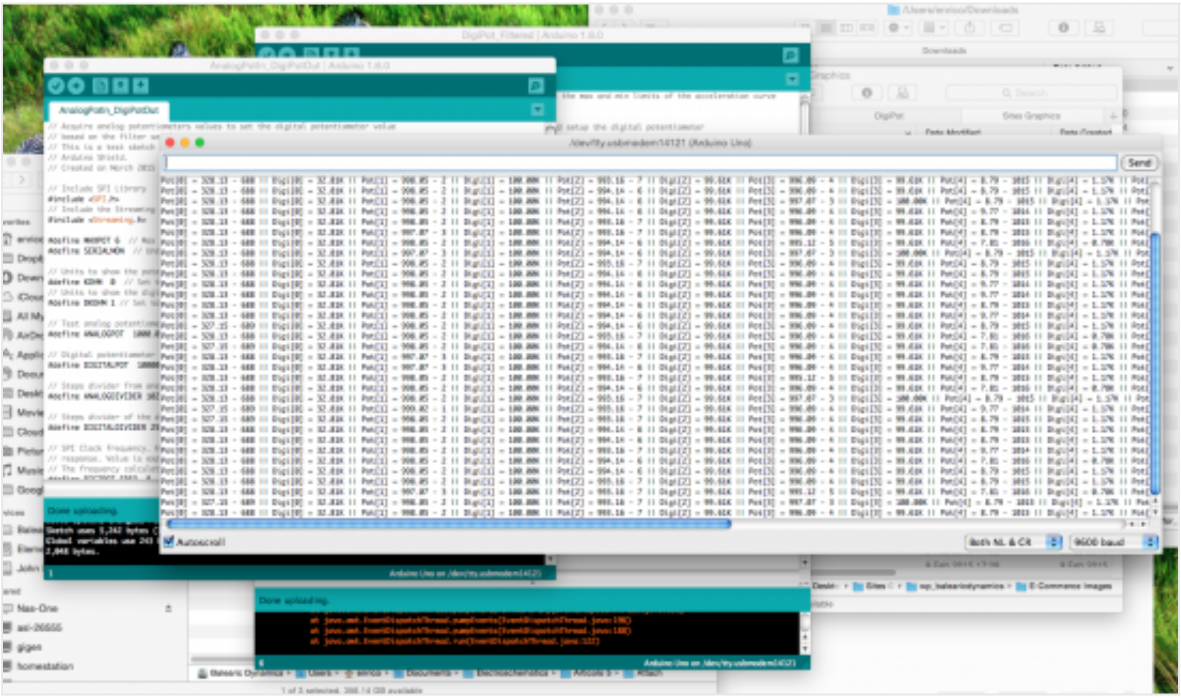
The test sketch can be downloaded **here**.

The sample sketch

The **first sketch** is useful to test the features of the Shield. It simply replicates the input trigger values generating a proportional value on the corresponding output.

Note: disabling the trigger analog inputs setting the dip switch to OFF, the analog reads of these PINs are random values and the resulting output is unstable.

Opening the serial monitor with the sketch running you will see a loop of input/output values as shown in this screenshot.



Every analog potentiomer (the triggers) is shown with the corresponding resistance value and analog read values, followed to the converted digital output values.

On top of the sketch there are the definitions of the program constants.

```
#define MAXPOT 6  // Max number of potentiometers
#define SERIALMON  // Undef SERIALMON to avoid serial output. For testing purposes only

// Units to show the potentiometer resistance value.
#define KOHM  0  // Set to 0 if the resistance should be in Ohm instead of K
// Units to show the digital potentiometer resisance value.
#define DKOHM 1 // Set to 0 if the resistance should be in Ohm instead of K

// Test analog potentiometers values in Ohm
#define ANALOGPOT  1000.00

// Digital potentiometer value in Ohm
```

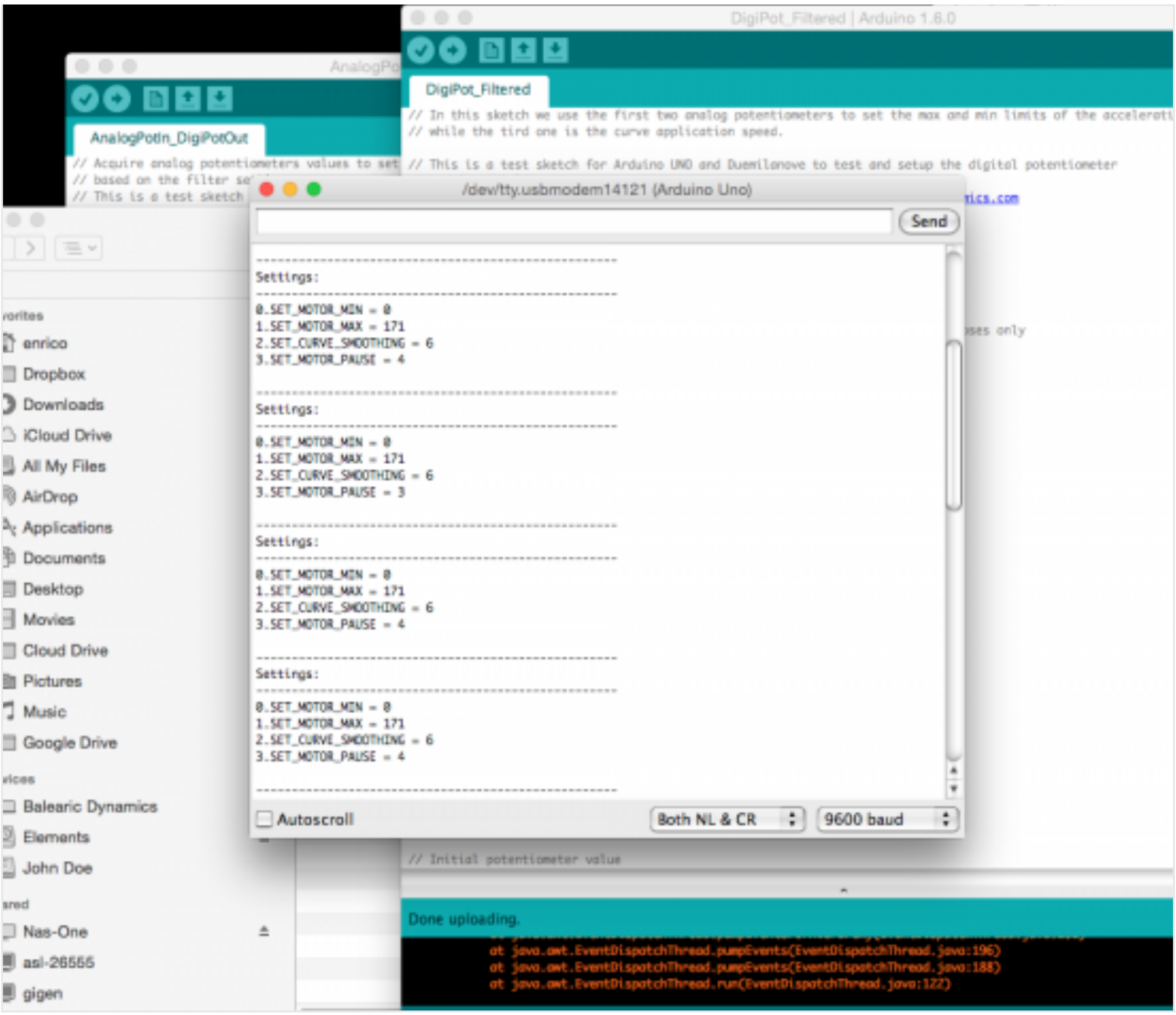
```
#define DIGITALPOT  100000.00

// Steps divider from analog converter
#define ANALOGDIVIDER 1024.00

// Steps divider of the digital potentiometer
#define DIGITALDIVIDER 256

// SPI Clock frequency. Higher is the clock frequency, higher is the digital potentiometer
// response. Value is expressed in reading-per-second
// The frequency calculation is an integer rounded.
#define DIGIPOT_FREQ  8
```

Changing the **DIGIPOT_FREQ** you can see how the clock LED changes the blinking frequency. You can disable the serial monitor that forces the program running slow by replacing the definition #**define SERIALMON** with #**undef SERIALMON**

Every step of the Sketch is documented in the code.

The video sketch

The **second sketch** is the one you see in action in the above video. In this case a simple linear filter is applied to the acceleration behavior. It is important to note as four triggers of the analog input are used to control the behavior of the digital output potentiometer.



Running the program with the serial monitor you get the output settings as shown in the screenshot above.

You can find the shield only at **BalearicDynamics**. Don't forget to use the discount code **ELEKTRO1008** to get 20% off discount, the special price for Electroschematics.com users.