Arduino intervalometer for Sony a6000 / A7

by **PJBulls** on April 23, 2016

Table of Contents

Intro: Arduino intervalometer for Sony a6000 / A7

I wanted to make time lapses on my Sony mirrorless camera, but the built-in app only supports 999 shots at a time. So, I decided to build an simple Arduino-based intervalometer that can take a picture at a desired interval for as long as I want.

Step 1: What you'll need

This intervalometer allows you to set an interval between 2 and 90 seconds or minutes, and maintain itself in a low power state in between to save battery. I haven't been able to empty the battery yet, but it should run for a very, very long time on one charge. These are the parts:

Arduino Pro Mini: I got a 3.3V / 8 MHz version from which I removed the regulator and LEDs using small pliers to save power, but this isn't strictly necessary. The Arduino will run directly from a 1S LiPo.

Sony Multiport connector:this piece is crucial if you want to make a wired Sony remote, but is somewhat expensive and difficult to source. I got mine from Mobile X Copter, or you could scavenge a Multiport cable. If you are building this for another DSLR, you don't need the proprietary Sony connector and a standard 2.5mm jack can usually replace this part.

A pot:to set the desired time interval. The value isn't that important, I used 10k Ohm.

7-segment LED display: shows you what your interval will be. You should get at least two 7-segments (or one double), but I used a triple module to be able to display slightly more info. Common anode/cathode and color don't really matter.

Switches: to turn on and start the intervalometer, as well as one to select second/minute intervals. I used a rocker, a momentary push button and a SPDT toggle switch respectively.

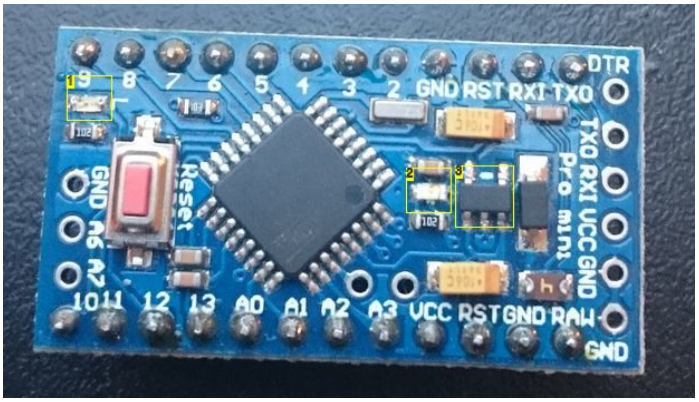
Resistors: prevent your 7-segment LEDs from burning out. 150 Ohm (7x) will do, also a few 20k Ohm for pull-up/pull-down.

1S LiPo: will power the project. I used two 680mAh cells in parallel as I had space in my enclosure.

LiPo charger board: allows you to charge the battery via 5V USB power.

Micro-USB breakout board: connects the intervalometer to the camera and a power source to recharge the battery.

Enclosure: a nice box to put everything into. I used a 3" x 2" x 1" enclosure which was a snug fit.



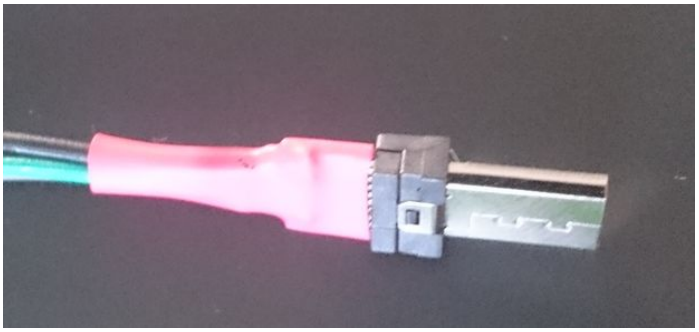
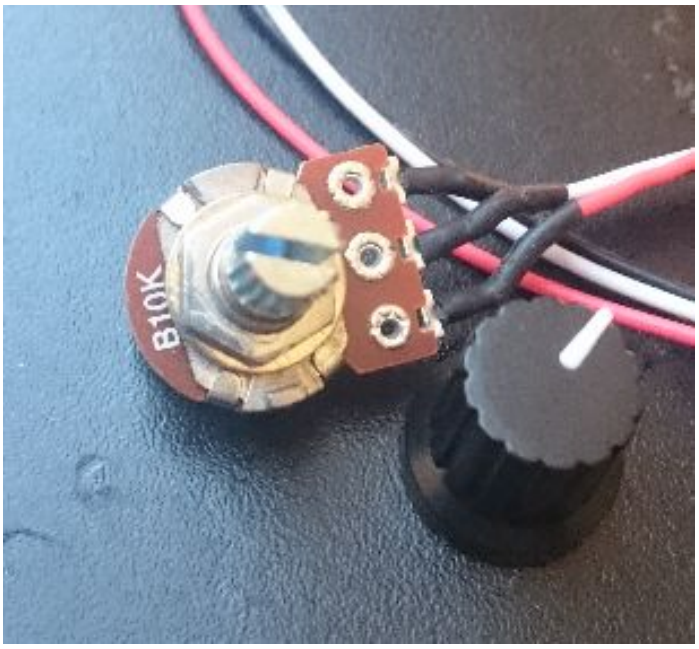
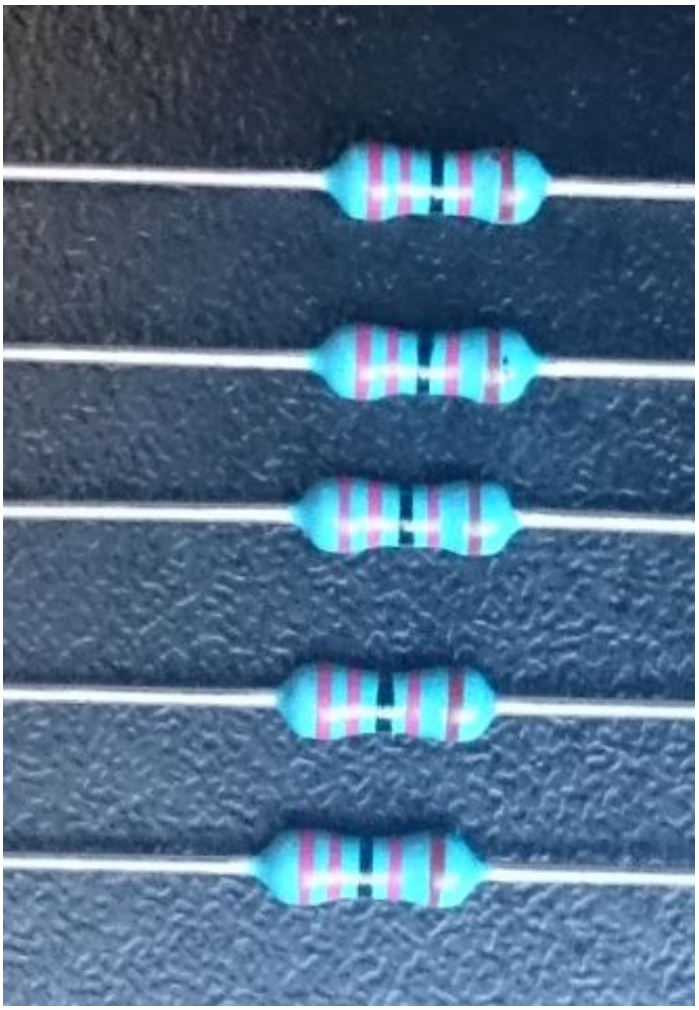


Image Notes
1. remove this LED
2. remove this LED
3. remove this regulator


B10K



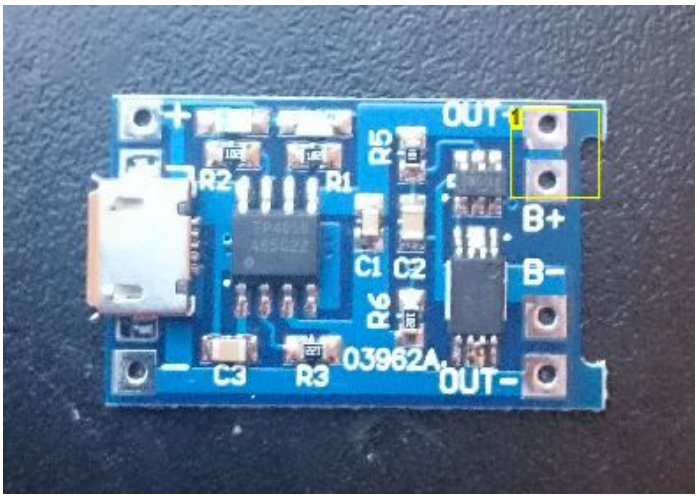
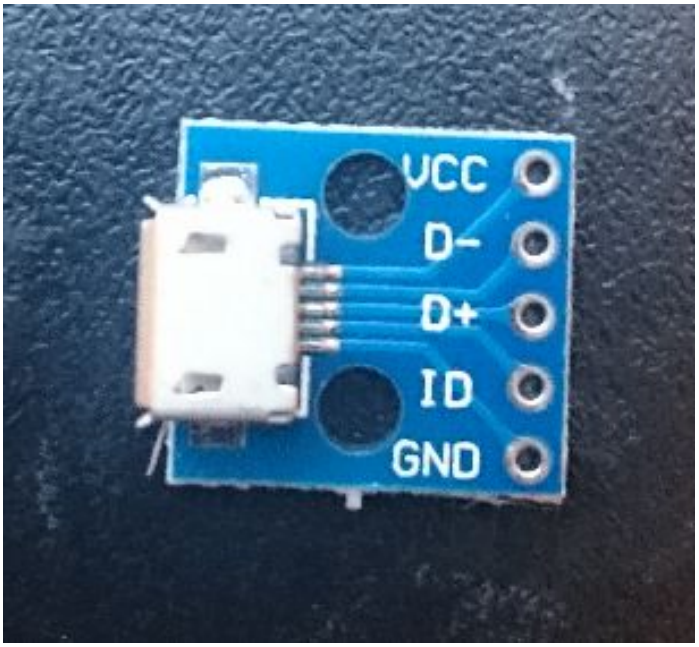
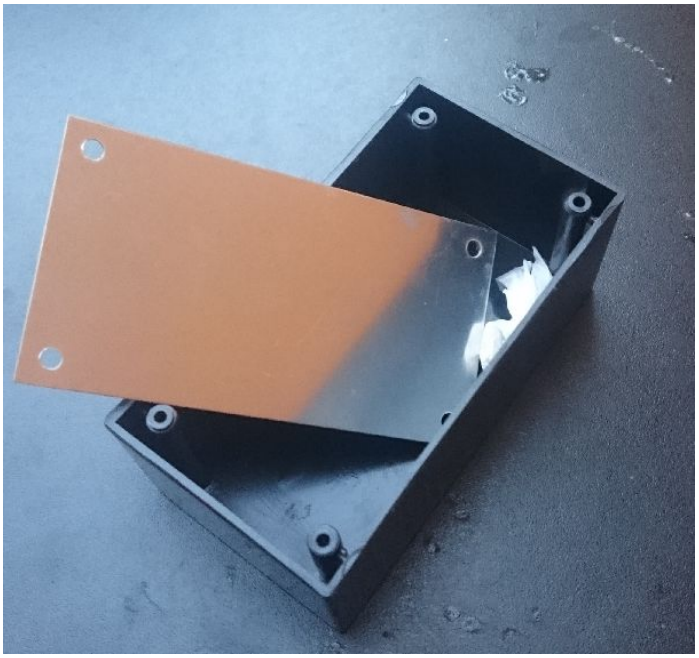


Image Notes
1. twin terminals = protected version (will prevent discharge below ~3V)



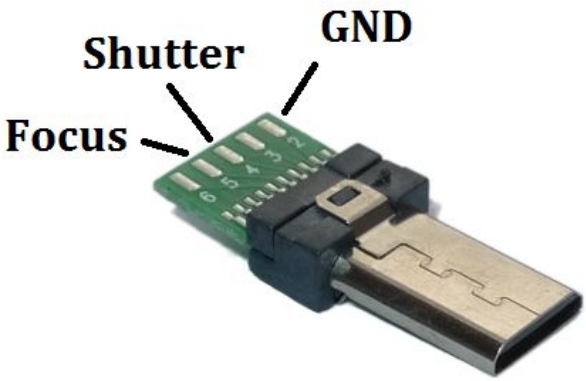


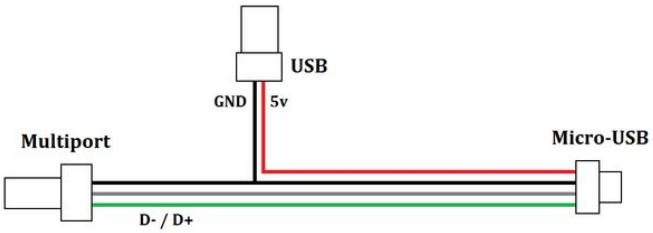
Step 2: Wire the Multiport connector

To make the camera take a picture, we'll have to pull both the focus and the shutter wire to ground. These are therefore the 3 most important connections, and they are on pin 5, pin 4 and pin 2 of the Multiport connector respectively.

I spliced these connections into a common USB-to-Micro-USB cable as shown above, where the power and ground wires remain connected to the USB plug, but the data lines (green and white) go into the Multiport pins 4 and 5 along with ground. The advantage of this setup is that you can use this cable both to control your camera and to charge the intervalometer (at the same time), but as long as you have a way of getting the 3 pins from your Multiport to the Arduino it will work.

As a side note, there are 2 more interesting connections on the Multiport plug but they are not broken out by default: pin 1 (the topmost little pin above ground) is 'ON/OFF' and will put your camera in and out of sleep when pulled to ground. Pin 15 (bottom small pin in the picture) is 'LANC_DC' and should be able to supply 3.1V power. This could make your meter battery-independent, although I haven't tested this.

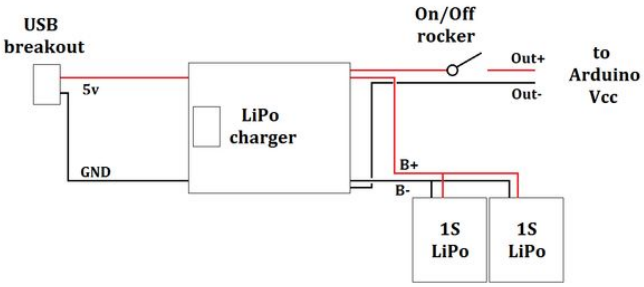




Step 3: Wire the battery and charger

Next, we'll take care of powering our project. Connect the USB breakouts for 5V (Vcc) and GND to the inputs of the charger board, and the battery to the B+ and B- terminals on the other side. If you don't have a board with protection circuit you'll only have one set of output terminals to which you connect both the battery and the Arduino. I wouldn't recommend using unprotected cells in that case, as there's no real way to check the battery level once everything is put together.

Finally, run some wires from the charger board output terminals towards your Arduino. I put a rocker switch here to switch the meter on/off.



Step 4: Wire the Arduino

Now we come to the heart of everything: hooking it all up to the Arduino. It doesn't matter that much what goes where, as long as you label them correctly in the code later on. The only thing you absolutely need an analog pin for is the potentiometer. Also, I found out the hard way that A6 and A7 can only be used as inputs, so you can only use these pins for the switches.

First, Vcc and ground should be connected to the corresponding LiPo charger board outputs.

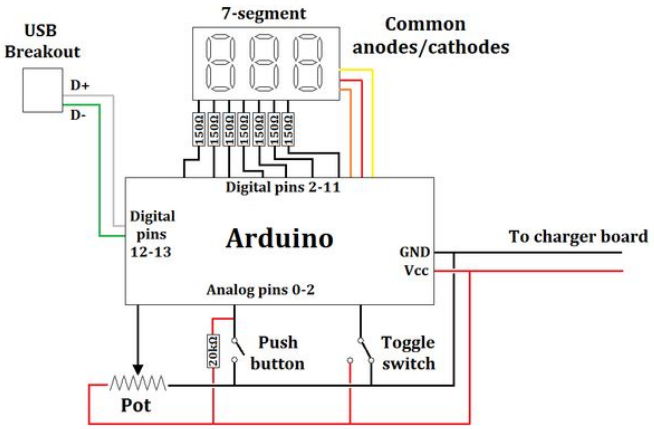
Connect the 7-segment LEDs through a 150 Ohm resistor each to some Arduino digital pins. Do the same for the common anodes (or cathodes).

Connect the USB breakout data lines to digital pins as well. These are the ones that go to the focus/shutter pins via your Multiport connector.

Then come the switches: the first one is to select second or minute intervals. In this build I used a SPDT switch with one leg tied to Vcc, one to GND, and the common one to an analog or digital pin.

The momentary pushbutton can be used to start the time lapse, you can connect an analog or digital pin to GND through this switch. I also added a 20k Ohm pull-up resistor here, but you could use the ones on the Arduino itself.

Finally, the potentiometer should be connected with one leg to Vcc, one leg to GND and its middle leg to an analog pin on the Arduino to create a variable voltage divider.



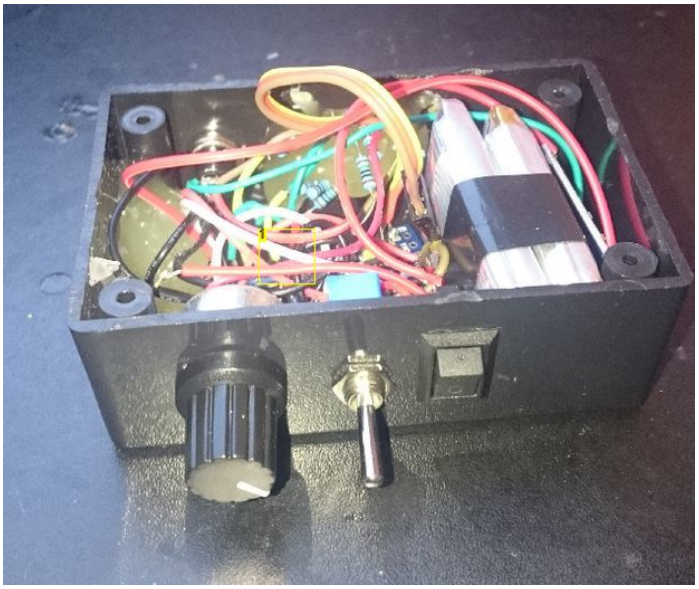


Image Notes
1. cable salad

Step 5: Code

Attached is a sketch that will run the intervalometer. Make sure to correctly declare the relevant pins at the top, you'll also need the LowPower library to put the Arduino to sleep in between pictures.

Keep in mind this sketch is written for common anode LED arrays, if you have common cathodes you'll need to drive the LED and cathode pins in the other direction (i.e. cathode LOW + led HIGH = on).

File Downloads

Sony_Timelapse.ino (5 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Sony_Timelapse.ino']

Step 6: Done!

And that's it! You should be able to turn on the meter, select an interval with the potentiometer and start it with the push button. This will put the Arduino to sleep, only to wake up momentarily when taking a picture.

By my not very accurate measurements the Arduino will draw <1 mA while asleep, so it should run for about two months on a 1200 mAh battery. You'll probably need some kind of power source for your camera though ;-)

Related Instructables

Time Lapse for Camera by samsungite

Arduino Time-Lapse Controller by hacker3455

How to Make Time Lapse Videos with Canon EOS DSLR by dndx

Time-Lapse Photography by randofo

How to Do Arduino-Controlled Intelligent Time-Lapse Photography by hlesliebole

DIY Time lapse dolly by ChiragShah

Comments

1 comments **Add Comment**

DIY Hacks and How Tos says:
Awesome Arduino project.

Apr 24, 2016. 8:29 PM **REPLY**