Touchscreen Pi Timelapse Controller

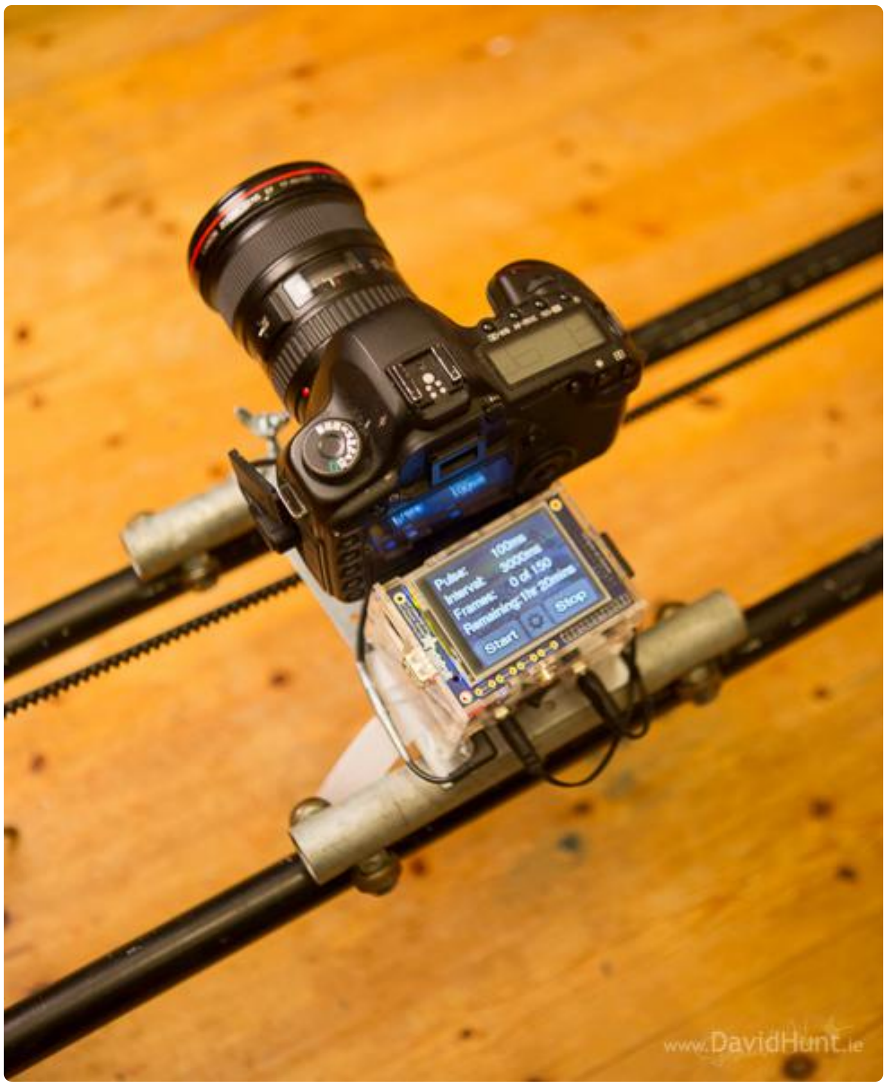
Created by David Hunt

https://learn.adafruit.com/touchscreen-pi-timelapse-controller

Last updated on 2023-08-29 02:28:38 PM EDT

Table of Contents

Overview



Turn your camera into an awesome high-tech time-lapse machine using a Raspberry Pi and a PiTFT for quick control! This is a great advanced project for camera + Pi geeks with motors, displays, and beautiful video output.

For example, here's a video I shot of the lovely Irish landscape - you can see how the motorized slider adds a dynamic motion to the timelapses!

Things You'll Need:

- Raspberry Pi computer, the Model B (http://adafru.it/998) is probably easier to set up and get going, but of you are comfortable with the Model A (http://adafru.it/1344), then that will work fine. You don't need any USB ports, and once set up, you don't need the Model B (http://adafru.it/998)'s ethernet port. The Model A (http://adafru.it/1344) save you on power, allowing you to run longer timelapses .:)

- PiTFT Mini Kit (http://adafru.it/1601) - 2.8" TFT touchscreen for Raspberry Pi
- SD memory card (http://adafru.it/102), 4GB or larger.
- A 10-14 volt battery pack (http://adafru.it/875).

If using WiFi and/or a Model B Pi, you'll want a robust battery pack that can provide 1 Amp (some are limited to 500 mA max).

- You can use a DC-DC converter (http://adafru.it/1385) to give us 5V for the Raspberry Pi. Or just go with a big USB battery pack such as this one. (http://adafru.it/1566)

If you want to use a Pi-controlled motor-drive slider, you'll also need:

- A Dual H-Bridge motor driver. I used a L298 Motor Driver from eBay () but your motor might also be OK with a 1.2A peak L293D (http://adafru.it/807)

You can also do without the motor driver, if you can live with switching the direction of the motor using a switch.

- A couple of NPN transistors and a couple of 1K resistors (for driving the shutter release of the camera.
- 2 DC Connectors, DC Switch, 3.5mm stereo jack socket

In some situations a USB to TTL Serial Cable (http://adafru.it/954) may be the preferred way to log in and configure the Raspberry Pi, if a spare keyboard and monitor are unavailable.

Some additional parts, tools and skills are also required: soldering iron and solder for connecting the header to the PiTFT display; some means of holding all the pieces together — could be as simple as a few rubber bands, to a drilled-out plastic electronics enclosure, to an elaborate custom 3D-printed case. This all depends on your available resources. Read through to see what's involved in the project and come up with ideas along the way.

This guide doesn't cover the building of the time-lapse rail or dolly, for that you can check out David Hunt's blog ().

Pi Setup

To ensure that all the software interdependencies can work, it’s easiest to start with a clean installation.

So, we need to set up the following:

- Fresh install of Raspbian
- PiTFT Setup
- Python development libraries
- Wiring Pi library
- Wiring Pi Python wrapper
- Lapse-Pi python script

Raspbian Setup

Format a 4GB or larger SD card (or microSD with adapter) and load it up with the 2013-09-25 version of the Raspbian Wheezy operating system. This guide explains how to prepare an SD card for the Raspberry Pi ().

On the PiTFT camera guide () (on which this project is based), it says that it’s vitally important to use the 2013-09-25 version of Raspbian, not the later 2013-12-20 release. However, the 2014-01-14 release does seem to work OK for this project, as we don't use the camera module.

Click here to Download Raspbian 2014-01-07 (780MB)

Connect a monitor and keyboard (or use a USB-to-serial console cable), power the Raspberry Pi from a USB phone charger or powered hub, and work through the usual first-time boot configuration.

The following options are required:

- Expand Filesystem

The following are very useful and recommended:

- Under Internationalization Options, select Change Timezone and Change Keyboard Layout to match your region.

The following are optional:

- Under Advanced Options, select Hostname to give this Pi a unique name (such as “picam”) to distinguish it from other Raspberry Pi’s on the network.
- Under Advanced Options, select SSH to enable command line access from the network (helpful for further configuration and troubleshooting).
- Other settings can be configured to your liking.

The following should not be used:

- Overclock. This is a portable, battery-operated project and an overclocked Pi will draw more current. Overclocked systems are also more likely to corrupt the SD card filesystem. Do notenable this option.

Once the basic system configuration is done, you can also set up wireless networking if you plan on using this capability. This guide may be of assistance ().

PiTFT Setup

Once the Pi is fully configured and on the network, work through the PiTFT tutorial:

()Adafruit PiTFT — 2.8" Touchscreen Display for Raspberry Pi ()

Work through the Assembly, Software Installation, Touchscreen Install & Calibrate and Using the Console pages, at least. The others (e.g. Playing Videos, adding the shutdown button) are not crucial to the time-lapse project but can be done if you’re also interested in exploring these capabilities. Once you have a Pi that boots to a login prompt on the 2.8" TFT, you’re in good shape.

Likewise, the optional tactile buttons on the PiTFT are not required for this project. You can install the buttons for other things if you like, but the time-lapse software is entirely touchscreen-based.

Python Development Libraries

This is simply getting a couple of packages so our packages below will build properly.

```
sudo apt-get install python-dev python-setuptools
```

Wiring Pi setup

Next we need to set up the python to use the Raspberry Pi's GPIO pins. This requires 2 packages, WiringPi and WiringPi2-Python.

First we get and build wiringPi

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
git pull origin
./build
```

Then we get WiringPi2-Python, a python wrapper for Wiring Pi.

```
git clone https://github.com/Gadgetoid/WiringPi2-Python.git
cd WiringPi2-Python/
sudo python setup.py install
```

LapsePi Touch Setup

Finally, we get the LapsePi python script and icons.

```
wget https://github.com/climberhunt/LapsePiTouch/archive/master.zip
mv master.zip LapsePiTouch-master.zip
unzip LapsePiTouch-master.zip
```

Now give it a try. The software must be run as root (using the sudo command) in order to access the TFT display

```
cd LapsePiTouch-master
sudo python lapse.py
```

If all goes well, after a few seconds' initialization you should see the Lapse Pi splash screen , Followed a couple of seconds later by the time-lapse information.

If this doesn't happen, an error message should give some sort of troubleshooting guidance; missing library or driver, etc. This is why we recommend working through the TFT tutorial first.

Once youve got that working, have the Pi boot straight into the time-lapse software by editing/etc/rc.local and adding the following lines before exit 0

```
cd /home/pi/LapsePiTouch-master
python lapse.py
```

Next time you reboot you should see the text console and then it will start the Lapse-Pi software.

Hardware

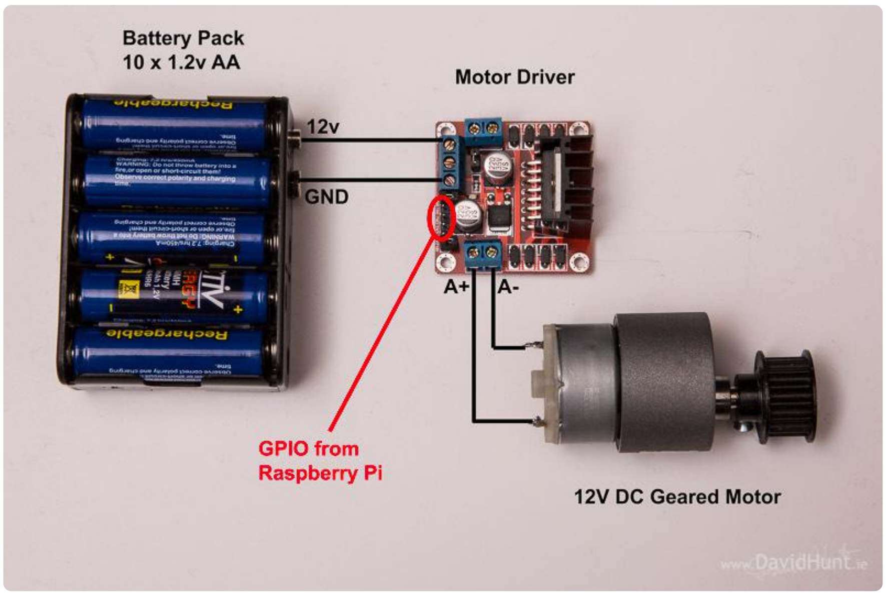
There's a bit of hardware setup in this project, which involves building a circuit with a couple of transistors and a couple of resistors, but nothing that can't be done in a couple of hours.

Also, there are a few pre-built modules used, namely a DC-DC converter and a Dual H-Bridge Motor Driver circuit.

If you're not going to drive a motor, then you can skip the Motor Driver section and if you use a 5v battery pack you can skip the DC-DC converter section. Jump straight down to the Camera Shutter Circuit section.

Motor Driver

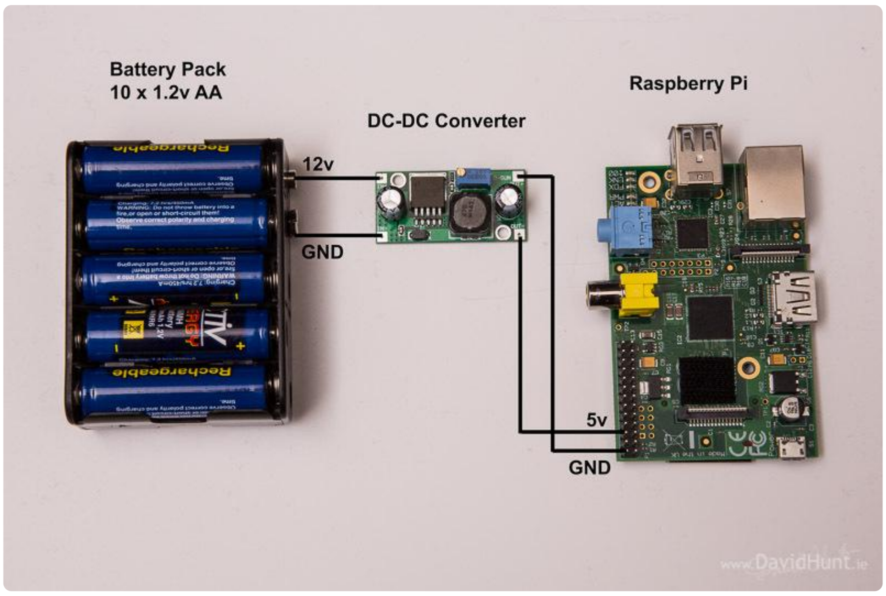
We'll start with the Dual H-Bridge Motor driver.



So, depending on which GPIO pin we activate, it will out out +12V or -12V on the A output or the B output. In the picture above, the motor is connected to the A output. If we wanted to use a stepper motor, we could use both A and B outputs, and change the code to pulse the GPIO pins accordingly. For now, we're just using one output to a plain 12V DC geared motor. You can also see in the image the timing pulley mounted on the motor which pulls the dolly along the rail.

DC-DC Converter

Next, we'll look at how we power the Raspberry Pi. Now we don't want to have a second battery just for the Pi, so we can use a DC-DC converter to give us 5v. A typical DC-DC converter is shown in the next image.



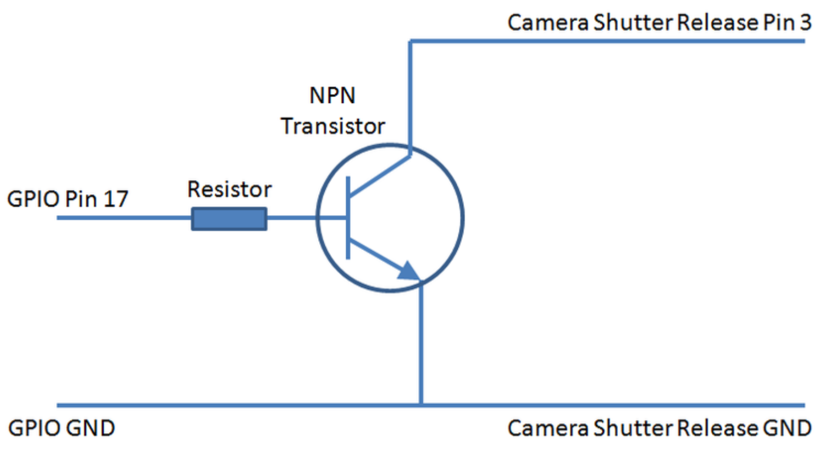
Notice that we feed the 5V in to the GPIO port, saving us having to have a connector sticking out of the mini-USB port, this saving us space.

Camera Shutter Circuit

Finally, we look at the circuit to drive the camera shutter. It's simpy a couple of NPN transistors, one for the focus and one for the shutter release.

The shutter release port in a DSLR will typically have 3 pins. One GND, one for focus, and one for the shutter. There's just like GPIO inputs, in that they will have a 3.3v voltage across them, and are activated by shorting them together. So, grounding the shutter pin will activate the shutter in the camera.

Some cameras need the focus pin actived before the shutter pin. Other cameras will trigger the camera with just the shutter pin shorted. On Canon DLSR's just the shutter pin is needed so one transistor and one resistor will suffice. Here's a diagram of the circuit. If you need to activate the focus pin on the camera, do a second one of these for the focus pin.

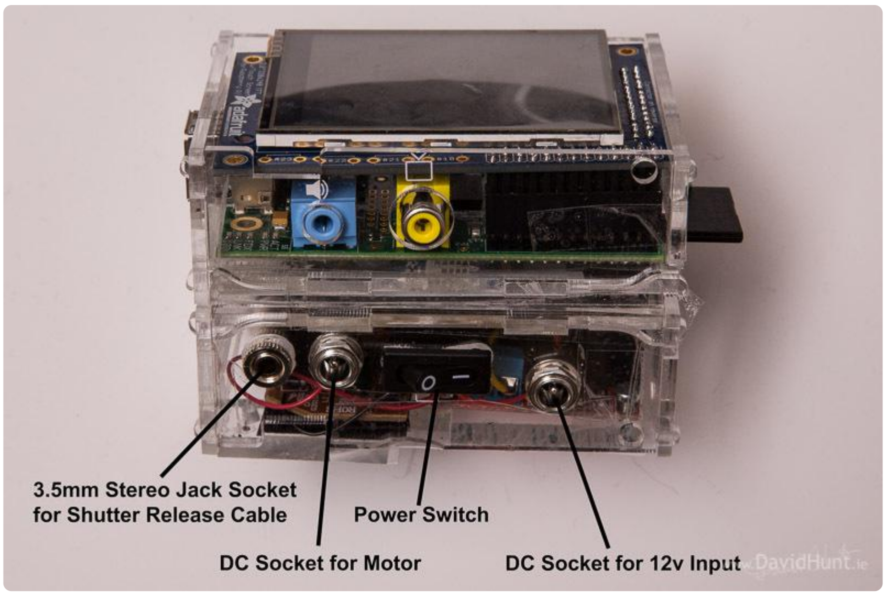


Here's the cable that's used to connect the project to a Canon 5D Mark III DSLR. It's made by replacing one end of a shutter release cable with a 3.5mm stereo jack plug.

Putting it all together

A couple of acrylic cases are ideal. You can leave the top off one of the cases for the PiTFT, as it's conveniently the same size as the Raspberry Pi. All the other circuitry can be put in another case below, and hidden away underneath the Raspberry Pi. You can also add suitable connectors for everything.

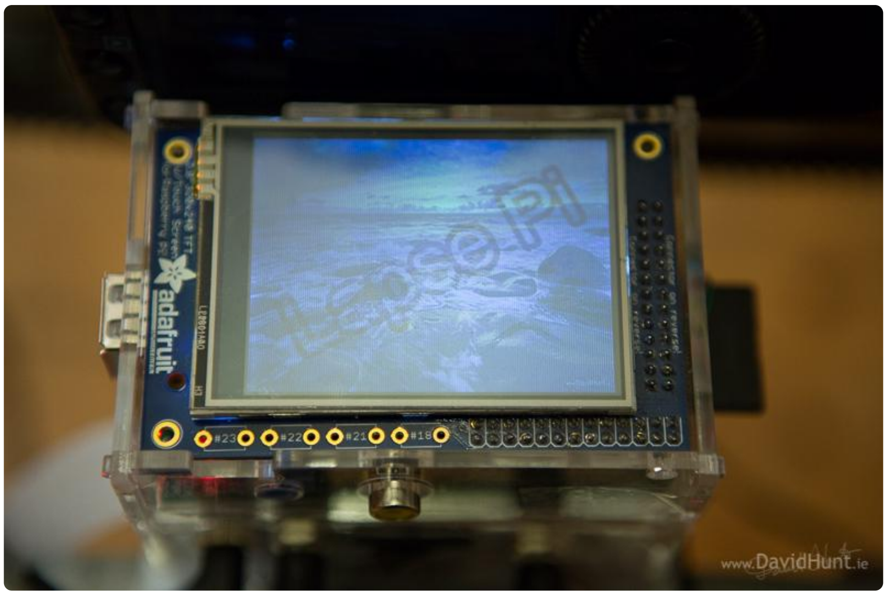
- DC jack socket for the 12V input from the battery pack
- DC jack socket for the 12V output to the motor
- Stereo 3.5mm headphone socket for shutter release cable to DLSR
- Power switch to switch everything off and on



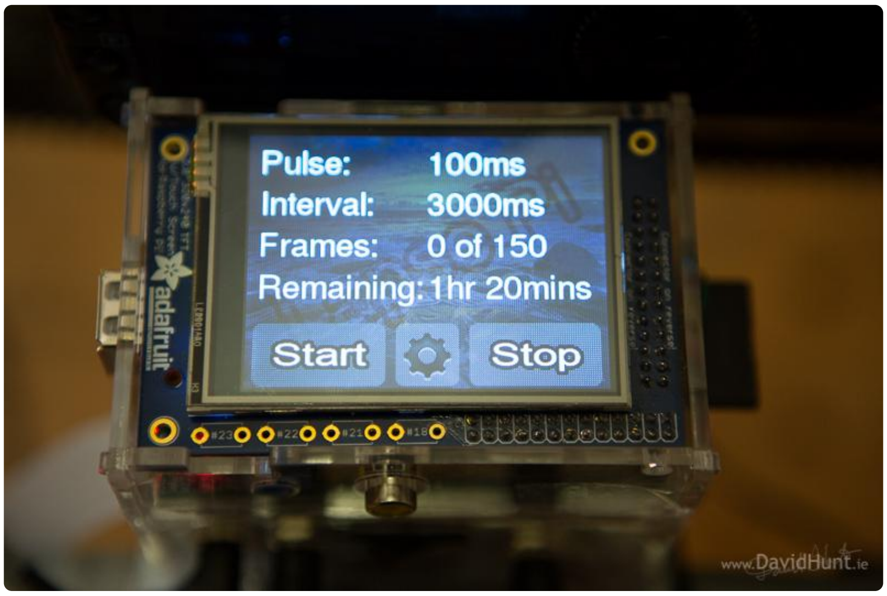
And here's the finished article.

User Guide

When you boot the Raspberry Pi, (as long as you've made the relevant changes to /etc/rc.local), you should observe the boot console followed by the lapse.py splash screen.

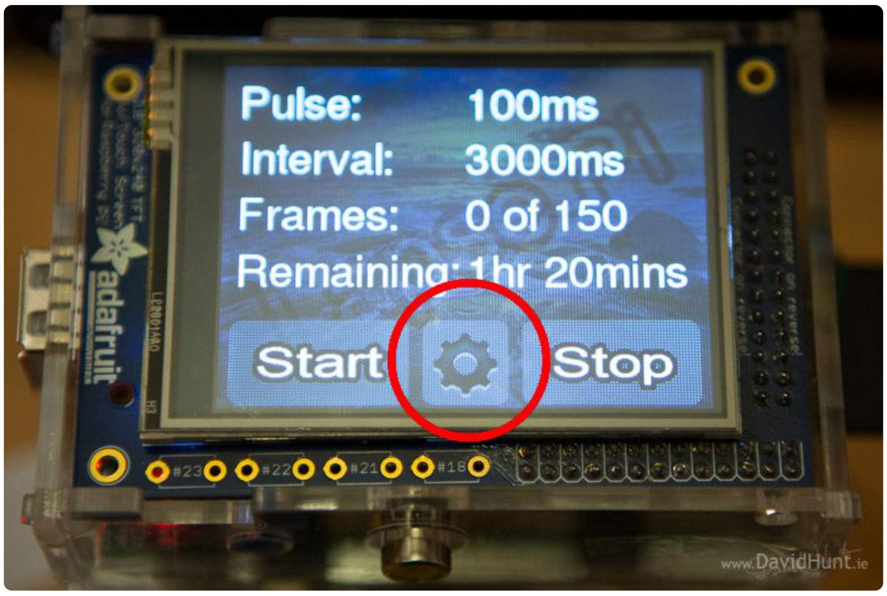


You should then see the main Lapse-Pi screen. This screen shows the current settings, and allows the user to start/stop the timelapse process, as well as call up the screen that allows the user to change the settings.

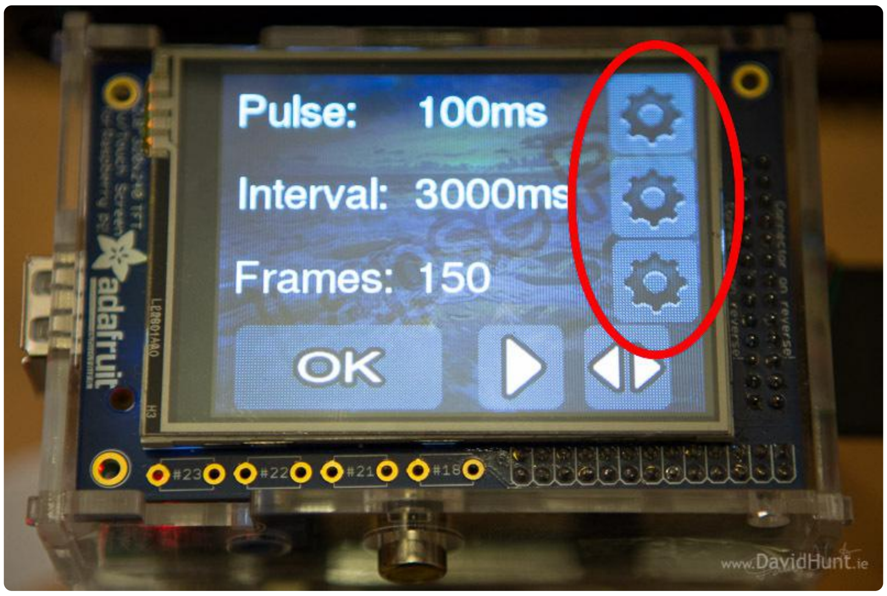


Pulse: 100ms
Interval: 3000ms
Frames: 0 of 150
Remaining: 1hr 20mins

Changing the settings is done my pressing the gear icon at the centre of the row of icons, which will then bring up the settings screen.



This screen allows the user to change the motor pulse, the interval between frames, or number of frames. To change any of the values, press one of the gear icons.

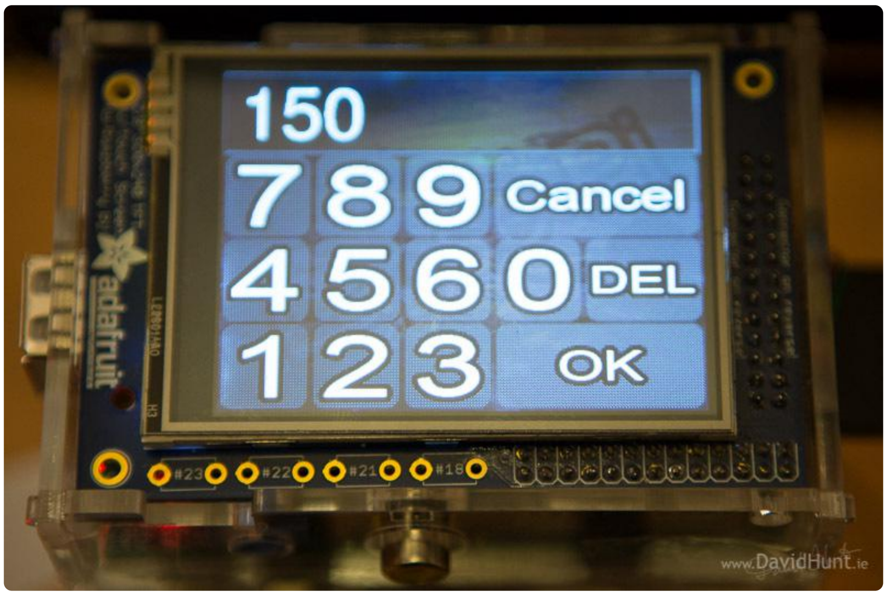


The pulse is the amount of time (in milliseconds) that the motor will be active between shots. Normally (depending on the motor) 150ms will be enough to move the dolly about 5mm. But again, that depends on the motor drive mechanism.

The interval is the delay between shots (again, in milliseconds). This is added to the pulse for the total time between shots.

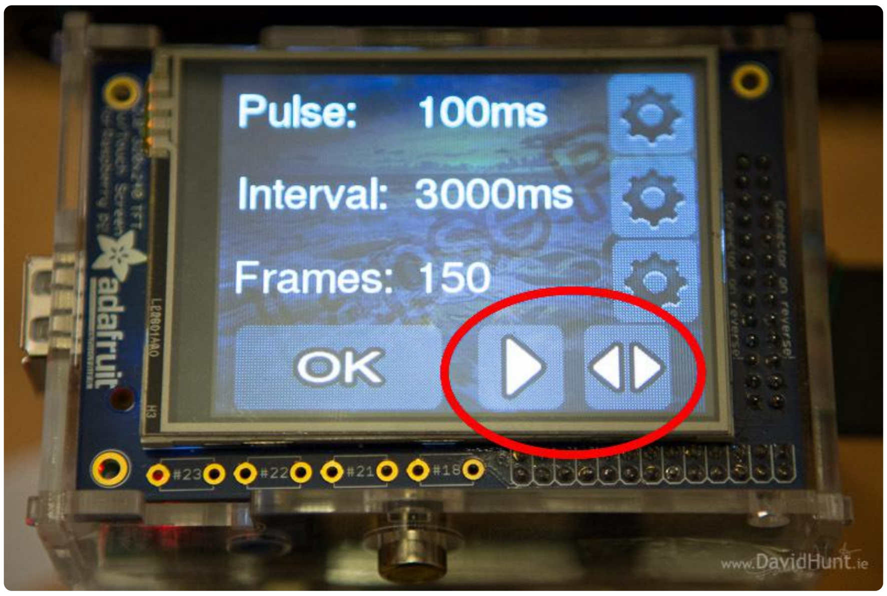
And the images value is the total number of shots to be taken for a given timelapse sequence.

When you press the gear icon for any of the values, a numeric keypad is shown.

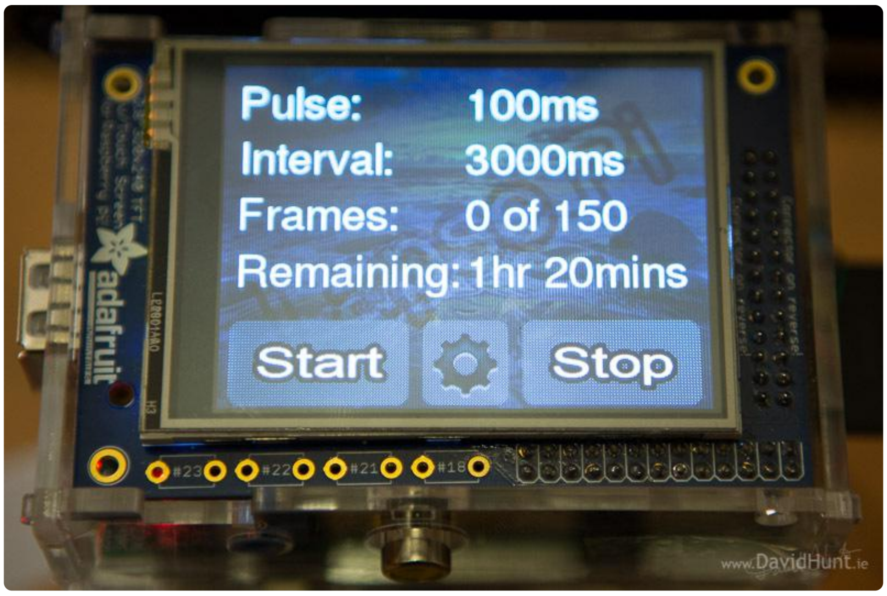


If you want to change the value, you press the ‘del’ icon to delete the digits, then type in the new value, and hit ‘Ok’

The right arrow and bi-directional arrow keys are for driving the motor manually. The right (play) icon will move the motor, whereas the bi-directional arrow will change it’s direction. THis allows you to move the dolly to the start of the rail before starting a timelapse sequence.



Once you click "OK", you are brought back to the main screen, where you can press Start to run the time-lapse thread. This will run the time-lapse in the background, so you can still go into the settings and change the values, which will also change for the running thread, so be careful!



Next Steps

There are so many things that can be done next. Here's a couple:

- Pull the thumbnails off the camera via USB and make a movie of current progress, like a "time-lapse live-view" :)
- Add panning and tilting of the camera so as well as moving sideways, the camera can turn at all sorts of angles during the sequence

If you're interested in more information on building the rail itself there are loads if pictures on Dave Hunt's blog here () and here (). There's a lot of metalwork involved, though!

Building the motorized rail is certainly worth the effort, though, as the panning effect gives a much more dynamic feeling to the resulting time-lapse sequences.