Timelapse Camera Slide from a 3D Printer

I have always loved timelapse photography and have tried doing it myself with some success. When I started seeing timelapse sequences other people had created that introduced camera movement I was blown away and knew that I had to build a timelapse camera dolly or camera slide. This was a project that was going to be some time in the future once I had finished building my own 3D printer. One day I got annoyed at the 3D printer project and pulled it to pieces. Contrary to my urge to jump up and down on the bits to demonstrate to them how annoyed I was, I quietly put them aside. After a bit I suddenly realised that the pieces would be perfect as the basis for the camera slide, and so the project to build one began. This document describes the project and what is required to build your own one with relatively simple bits.

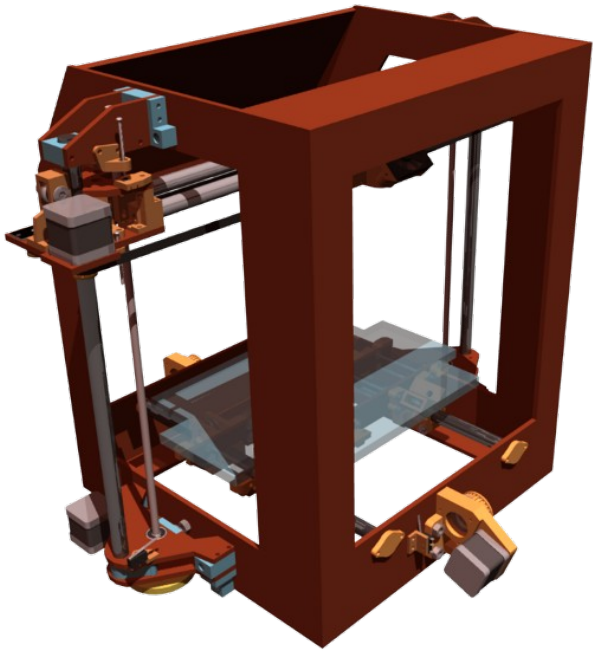
Now that I have a timelapse camera slide I just need to get out and about (preferably into the mountains) to make use of it. Here in windy Wellington (NZ), the clouds move so fast that timelapse photography is not needed.

Project History - 3D Printer to Camera Slide

Several years ago my Partner and I attended the 8th Fablab conference/gathering/happy-clappy-singalong whatever you might call it. One of the sessions was “Machines that make machines” which was a hands-on workshop on technologies for small run manufacture. Whoever was going to run the session failed to turn up and so it fell to Vik Olliver of Reprap fame to run the session.

Vik started us designing a new 3D Printer based on a Mendel Reprap but making use of the tools available in a Fablab with the view of it being suitable for a Fablab to manufacture. This meant that the design could use laser cut panels to produce a very sturdy frame, and then use printed bits for the more complex components such as bearing holders, and extruder parts. The small group of us dug into designing this thing using Vik’s design philosophy “*if it couldn’t be assembled with just a mallet, it was too complex*” .... or something like that. Which was fine, but my own philosophy is to use screws and bolts so that the devices I design can be disassembled easily for upgrade and repairs etc.

After the Fablab conference/gathering/happy-clappy-singalong whatever you might call it, I continued the collaboration for a while and Vik got a prototype working. This was called the Rroofl (There are some pages on the Reprap.org forums for the early part of the development - http://reprap.org/wiki/Rroofl). The name is derived from something like “Reprap out of Fab Lab”. Eventually I heard nothing more from Vik and so assumed the project had quietly vanished. I decided to continue the project based on my own philosophy of assembling with bolts and machine screws. The design shown below is what I came up with and what I ended up building (sorry I don't think I have any photos of it in its fully assembled glory). I made use of the standard reprap RAMPS board, Arduino Mega, and the Marlin coding. The frame and large parts of the structure were simply 6mm thick MDF panel cut out with a scrollsaw. This made a very sturdy frame. If you are really interested, you can download the model file here https://app.box.com/s/p28sij1qkuqapl55f07294ebiikmswa3



Incidentally the Rroofl resurfaced in about 2014 redesigned and now a commercial 3D Printer called the Diamondmind. This was from a collaboration between Vik Olliver (Diamond Age Solutions- http://diamondage.co.nz) and the good people of Mindkits (http://www.mindkits.co.nz/) .

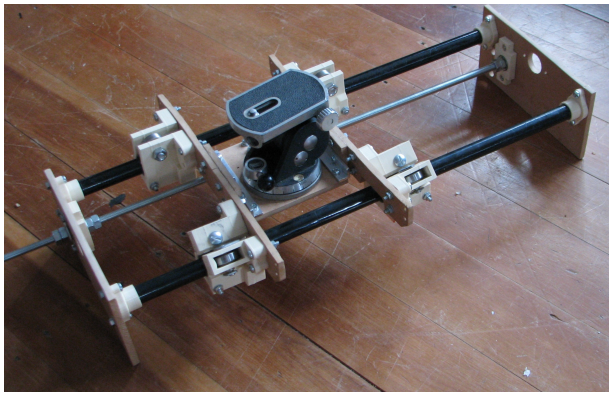
Anyway, I eventually had all the bits of my printer assembled, and had the code working properly. I was into the final stages of installing the endstops and drive belts and went to try the printer out. Much to my annoyance I found that I had accidentally crushed the thermistor in the extruder head. I sighed. I looked at the hulking great assembly of MDF board, threaded rod, bearings, metal plates, metal tubes, motors, electronics, and 3Dprinted parts and thought to myself "Why am I doing this? It will never be very fast, it will not be fabulously accurate, and the world does not need yet another 3DPrinter anyway." Rather than spending the next couple of hours digging in to replace the thermistor, I spent the afternoon disassembling it to its most basic components..... and I felt good about it. Sometimes a project really does need to be killed off.

Camera Slide Hardware

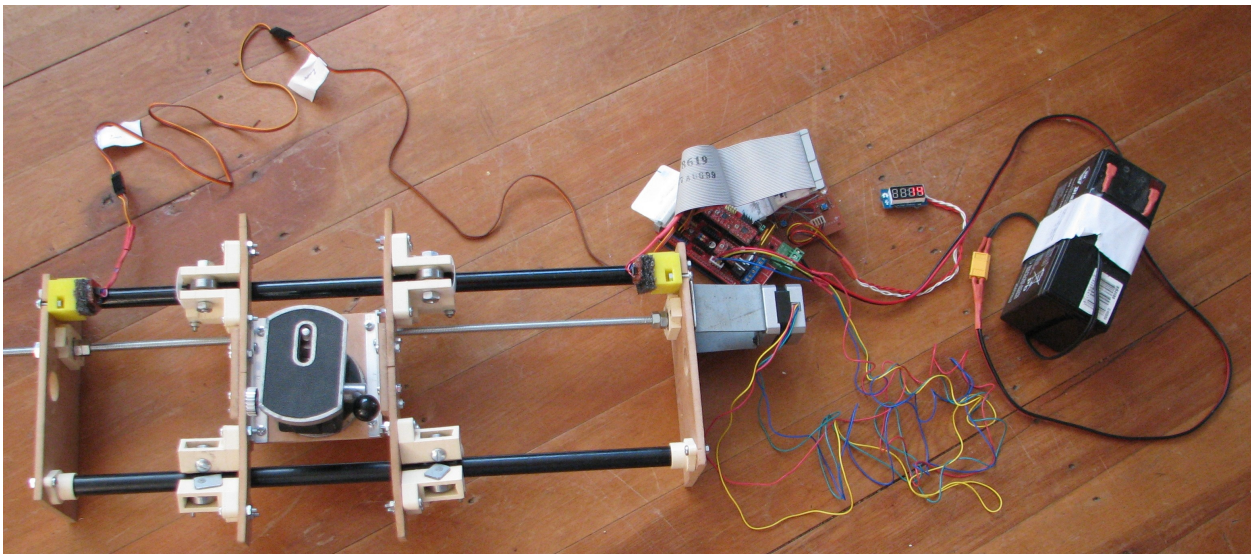
Now I had a large pile of really useful bits and pieces and a pile of interestingly shaped 3D printed bits. A month or two later I was looking at the pile and thinking, "I should come up with a project to make use of all this stuff". It suddenly occurred to me that I could fast-track one of my future projects by repurposing the 3D Printer bits. The project was a ***camera slide*** for timelapse photography. I have always loved timelapse photography and the introduction of camera movement really adds another dimension to the videos produced. If you think turning a 3D printer into a timelapse camera slide is weird see what happens to 3D printer heater beds that misbehave – http://elfnor.com/a-novel-use-for-a-3d-heated-printer-bed.html, yep a duckling warmer.

Ex 3D Printer Grab Bag.

Very quickly I was able to come up with an assembly of pieces that were formerly the 3Dprinter Y-axis and turn them into the camera slide rails. Bits of the z-axis and the x-axis became the drive for the camera traveller. This is shown in the photos below.



Camera Slide at the end of an afternoon's happy assembly.



Development Camera Slide and Electronics

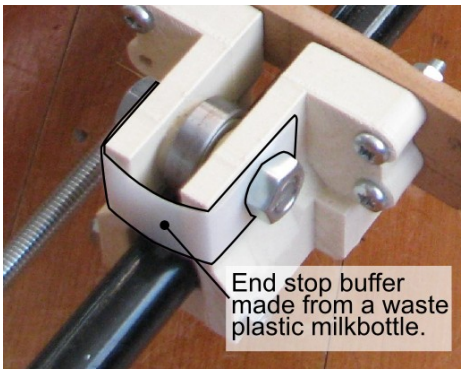
As you can see this camera slide is very short. This is the development one simply because I wanted something small while I developed the electronics rather than lugging around the full size one. To change it to the full size one all I need to do is replace the short 16mm diameter tubes with the 2m long ones, and change the M8 drive screw for a suitably long one. The wooden end panels will not need to be changed. In the code there is a parameter for the rail length that would need adjusting. This is discussed later.

The hardware used is;

- A camera mount of some sort - repurposed tripods and the like are ideal
- Two lengths of 16mm outside diameter aluminim tube.
- A length of M8 threaded rod.
- Six 22mm outside diameter and 8mm inside diameter skate bearings.
- M4 threaded rod with nuts and washers.
- M4 machine screws.
- Some M8 machine screws or whatever you can scrounge for the axles on the traveller.
- Some aluminium angle (12 x 12 x 1 is good). This is to fix the traveller tray to the traveller cross-panels.
- M8 nuts, including one nylon bushed lock nut.

- 6mm MDF panel
- Some soft foam plastic packing material for the end stop buffers.
- Some waste plastic (milkbottle) with reasonable strength to act as a buffer on the traveller where it connects with the end-stop buttons (see image below).
- 3D Printed bits.
- Something to couple the stepper motor to the M8 threaded rod drive shaft.

The last item on this list is something I have made with some machined metal with grub screws, but there is no reason it could not be a 3D printed coupler.

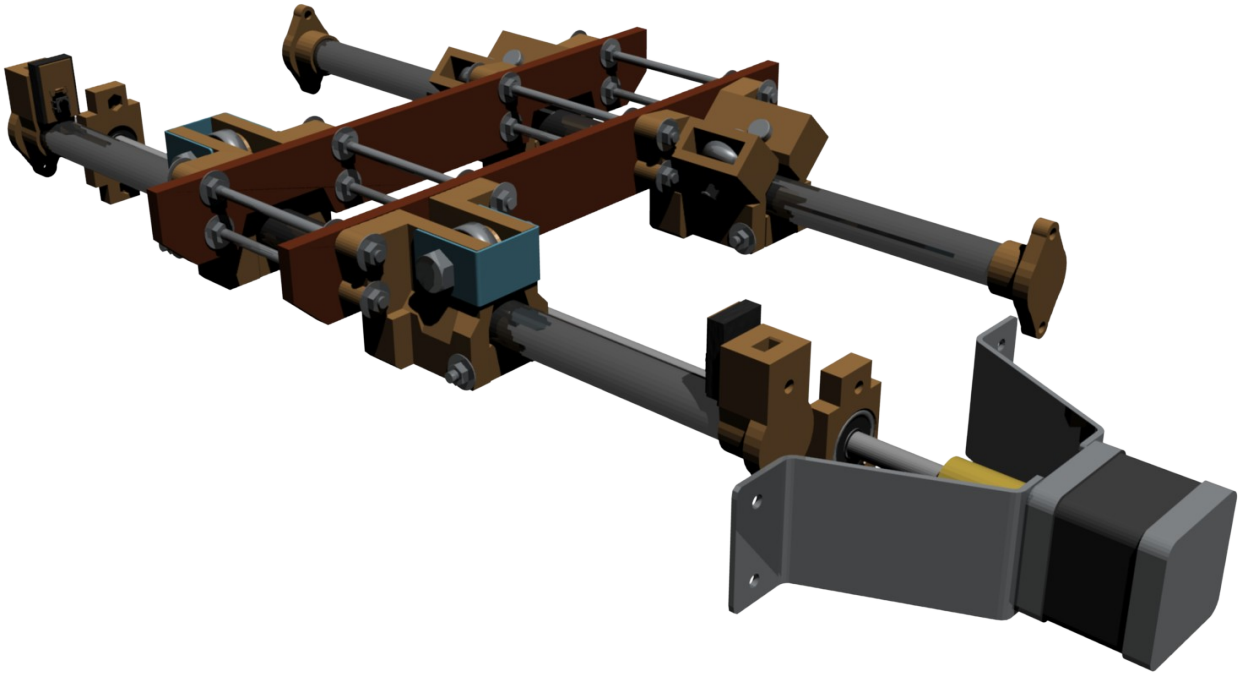


The 3D Printed parts can be found within the Blender 2.57 assembly model and collection of STL model files that can be downloaded from here: www.techmonkeybusiness.com/models/CameraSlide.zip

I have included some drawings of the wooden panels for the sake of completion. I just re-cut the ones I had made for the 3D printer, and so the camera slide appearing in the photo is a little rougher than if it was built from scratch using the files on this webpage.

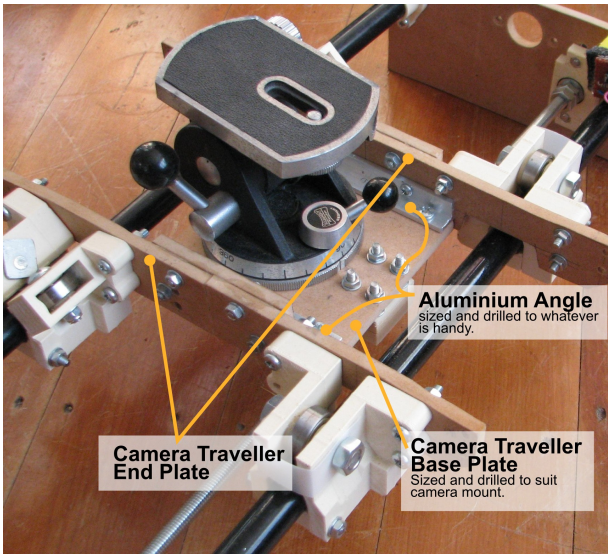
The panel drawings and TurboCAD, DWG, and DXF files can be found here:
www.techmonkeybusiness.com/models/CameraSlidePanelsCAD.zip

and as pdfs for A3 and A4 printing here:
www.techmonkeybusiness.com/pdfs/CameraSlide_PanelsA3.pdf
www.techmonkeybusiness.com/pdfs/CameraSlide_PanelsA4.pdf.



Blender Model of the Camera Slide

The MDF panel drawings do not show where every hole needs to be drilled because this will vary depending on what motors are used and how they are attached, what sort of camera mount you have available, and what size angle plate you have available to attach the traveller base plate to the traveller end plates. You may also need to add a length of wood or something in parallel with the rails and drive screw to firmly hold the track end panels, but again this will depend on what you have to hand.



Electronic Hardware

Because they were left over from the aborted 3D printer build I had a RAMPS board, Arduino Mega, stepper motors, and end stops, to build and run the camera slide. Because I was not intending to control the camera pitch, tilt, and pan, the RAMPS / Arduino Mega combination would not be working very hard at all. Apart from changing the pin assignments, an Arduino UNO or Arduino Nano driving an Easystepper or Pololu stepper driver could easily run this system.

To code the Arduino Mega, I pulled out the schematics of the RAMPS board from the Reprap.org website (http://www.reprap.org/wiki/Ramps) and figured out which pins I needed to use. In the end, the pin assignments were:

Endstop 1 - RAMPS Z-MAX D19 (Interrupt 2 – on my Arduino Mega this was INT4)
Endstop 2 - RAMPS Z-MIN D18 (Interrupt 3 – on my Arduino Mega this was INT5)
Stop button - RAMPS Y-MAX D15 (Supposedly this was Interrupt 9, but in reality my Arduino Mega did not recognise it as an interrupt)

RAMPS AUX-4 Connections
Start Fast Reposition button direction 0 - D33
Start Fast Reposition button direction 1 - D35
Start Timelapse Travel button direction 0 - D37
Start Timelapse Travel button direction 1 - D39
Transit speed setting button up - D17
Transit speed setting button down - D16

TM1637 Display
CLK - D25
DIO - D23

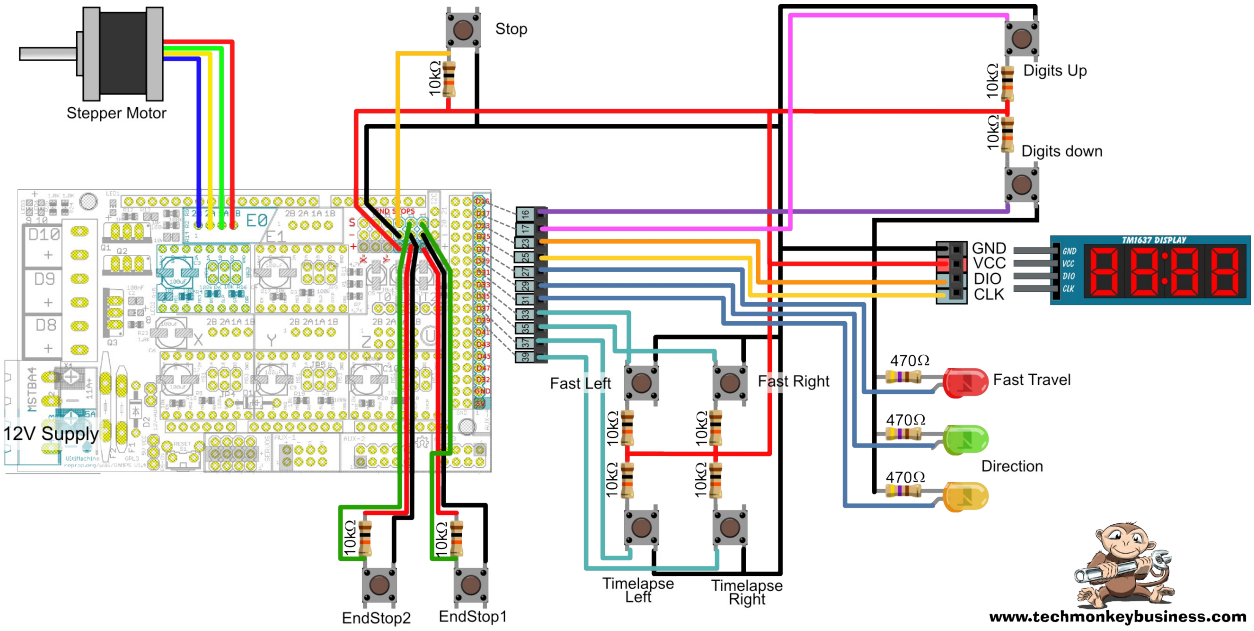
Indicator LEDs on pins D27, D29, and D31

5V supply to perfboard panel from the RAMPS Y-MAX connection
GND to perfboard panel from the RAMPS Y-MAX connection

Stepper Control Pins:
Step - D26
Dir – D28
Enable - D24

The control board is relatively simple and consists of a number of buttons with 10kΩ pull up resistors (The pin is high ordinarily and goes low when triggered), several LEDs and a TM1637 based display. The display is used to show the time in minutes for a full length traverse when running in timelapse mode. The buttons on pins D17 and D16 are used to set this traverse time.

The circuit is shown below and where it attaches to the RAMPS board.



Code

The code is not perfect, but functions quickly and efficiently. A smart operator is required to avoid doing silly things like driving the traveller in the direction of the end it is stopped against after the end stop switch has triggered. I figure that clear indication of travel direction on the electronics housing and with the LEDs should minimise this.

The code makes use of the Accelstepper library which can be found here:
http://www.airspayce.com/mikem/arduino/AccelStepper/

The buttons used are to set the stepper motor running in either direction in timelapse mode or fast reposition mode. There is a button to stop the motor, and two buttons to set the traverse time when the motor is not running.

The LEDs indicate direction of travel and mode.

Throughout the code flags are used to indicate the mode of travel and by setting these flags to zero the system can be stopped immediately. This makes it very easy and efficient for use within the endstop interrupt routines.

Depending on the track length used the code will need to be modified to reflect the actual travel. This is done by setting the `trklength` constant to the available length of travel in millimeters.

You can download the sketch from here;
www.techmonkeybusiness.com/Code/RAMPSCameraSlidev4.ino

Camera Slide Sketch

```
/* RAMPSCameraSlidev4.ino
The purpose of this sketch is to develop the logic
used with the RAMP board driving the camera slide.

The speed during timelapse shooting will be described
as time for a traverse of the track in minutes.  This
will be displayed on a 4-Digit display that uses the
TM1637 driver chip.

Obviously seeing as we are using a RAMPS board we are
using an Arduino Mega.  It's all a bit overkill but that
is what I had to hand.

Pin assignments are:
Endstop 1 - RAMPS Z-MAX D19 (Interrupt 2)
Endstop 2 - RAMPS Z-MIN D18 (Interrupt 3)
Stop button - RAMPS Y-MAX D15  (Interrupt 9)

Note:  On my MEGA, the interrupts do not match the
official assignments.  RAMPS Z-MAX D19 is INT 4
and RAMPS Z-MIN D18 is INT 5.
Note also that D15 does not appear to interrupt anyway
so this edition introduces an if statement within the scheme
to try to stop it.

RAMPS AUX-4 Connections
Start Fast Reposition button direction 0 - D33
Start Fast Reposition button direction 1 - D35
Start Timelapse Travel button direction 0 - D37
Start Timelapse Travel button direction 1 - D39
Transit speed setting button up - D17
Transit speed setting button down - D16

TM1637 Display
CLK - D25
DIO - D23

Indicator LEDs on pins D27, D29, and D31

5V supply to breadboard
GND to breadboard

Stepper Control Pins are on the RAMPS Extruder E0:
Step - D26
Dir - D28
Enable - D24


*/
```

```
#include <TM1637Display.h>

#include <AccelStepper.h>
// Define the stepper and the pins it will use
//The numbers are (don't know, Step, Dir)

AccelStepper stepper1(1, 26, 28);

const int CLK = 25;
const int DIO = 23;

const int Estop1 = 18;
const int Estop2 = 19;
const int Allstop = 15;

const int FastRepButton1 = 33;
const int FastRepButton2 = 35;
const int RunButton1 = 37;
const int RunButton2 = 39;
const int AdjTimeup = 17;
const int AdjTimedn = 16;
const int LEDcol1 = 31; //LED to indicate Fast direction 1
const int LEDcol2 = 29; //LED to indicate Fast direction 2
const int LEDFast = 27; //LED to indicate Fast travel.

volatile boolean Fast = 0;  //Flag for repositioning
volatile boolean RunMe = 0; //Flag for timelapse photography run.
volatile boolean dir = 0;  //Flag for direction to run.

const int trklength = 300;  //track length in mm
int TravTime = 60;  //default time (minutes)to traverse length of track
const float ScrewPitch = 1.25; //pitch of the drive screw
const int StepSize = 400; //Steps per rotation
const int MSteps = 8; //Microsteps per step - a fudge factor.
const long TotSteps = trklength/ScrewPitch * StepSize * MSteps; //Total steps
for track
int TravSpeed = int(float(TotSteps/(TravTime*60))); // Step per second

int FastSpeed = 8000;  //Speed for the fast traverse
int FastSpdDir = 1;    //Direction multiplier for fast traverse direction.


TM1637Display display(CLK, DIO);  //set up the 4-Digit Display.

void setup()
{
  display.setBrightness(0x0a);
  pinMode(Estop1, INPUT);
  pinMode(Estop2, INPUT);
  pinMode(Allstop, INPUT);
  pinMode(FastRepButton1, INPUT);
  pinMode(FastRepButton2, INPUT);
  pinMode(RunButton1, INPUT);
  pinMode(RunButton2, INPUT);
  pinMode(AdjTimeup, INPUT);
```

```
  pinMode(AdjTimedn, INPUT);
  pinMode(LEDcol1,OUTPUT);
  pinMode(LEDcol2,OUTPUT);
  pinMode(LEDFast,OUTPUT);

  attachInterrupt(5,StopSys,FALLING); //D18 Pin *
  attachInterrupt(4,StopSys,FALLING); //D19 Pin *

  // * Note interrupts different from official ones.
  // don't ask me why this is.

  // The only AccelStepper value we have to set here is the max speeed, which
is higher than we'll ever go
  stepper1.setMaxSpeed(10000.0);
  stepper1.setAcceleration(100);

  digitalWrite(LEDcol1,LOW);
  digitalWrite(LEDcol2,LOW);
  digitalWrite(LEDFast,LOW);


}


void loop()
{
  display.showNumberDec(TravTime); //Display the traverse time;
  TravSpeed = int(float(TotSteps/(TravTime*60)));

  //The control buttons are used to set the flags which determine
  //the mode of operation and the direction.
  if(digitalRead(FastRepButton1) == LOW)
  {
    RunMe = 0;  //Disable the Timelapse mode
    Fast = 1; //enable the fast travel mode
    dir = 0; //Set the direction to 0
    stepper1.setSpeed(FastSpeed);  //Sets the speed.
  }
  if(digitalRead(FastRepButton2) == LOW)
  {
    RunMe = 0;  //Disable the Timelapse mode
    Fast = 1; //enable the fast travel mode
    dir = 1; //Set the direction to 1
    stepper1.setSpeed(-FastSpeed);  //Sets the speed in negative direction.
  }
  if(digitalRead(RunButton1) == LOW)
  {
    RunMe = 1;  //enable the Timelapse mode
    Fast = 0; //disable the fast travel mode
    dir = 0; //Set the direction to 0
    stepper1.setSpeed(TravSpeed);  //Sets the speed for slow traverse.
  }
  if(digitalRead(RunButton2) == LOW)
  {
    RunMe = 1;  //enable the Timelapse mode
```

```
    Fast = 0; //disable the fast travel mode
    dir = 1; //Set the direction to 1
    stepper1.setSpeed(-TravSpeed);  //Sets the speed for slow traverse.
  }
  //LED will be used to indicate that the motor is running
  //and the direction it is going.

  digitalWrite(LEDcol1,dir);  //The LED pins hold their values
  digitalWrite(LEDcol2,!dir);  //while the other processes
  digitalWrite(LEDFast,Fast);  // are running.

  while(Fast == 1 | RunMe ==1)
  {
    stepper1.runSpeed();
    if(digitalRead(Allstop) == LOW)
    {
      Fast = 0;  //If panic button hit then stop everything.
      RunMe = 0;
    }

  }

  //If the platform is not in either fast relocate or
  //Timelapse shooting mode, then allow the Traverse time
  //to be set.
  if(RunMe == 0 && Fast == 0)
  {
    if(digitalRead(AdjTimeup) == LOW)
    {
      TravTime = TravTime +1;  //Increase Traverse Time by 1 minute
      delay(10);  //Just to slow things a little.
    }
    if(digitalRead(AdjTimedn) == LOW)
    {
      TravTime = TravTime -1;  //Increase Traverse Time by 1 minute
      if(TravTime <= 1)  // No negative time eh.
      {
        TravTime = 1;
      }
      delay(10);  //Just to slow things a little.
    }
  }

}

void StopSys()
{
  Fast = 0;
  RunMe = 0;
  //Stop the motor.
}
```

Hamish Trolove

www.techmonkeybusiness.com