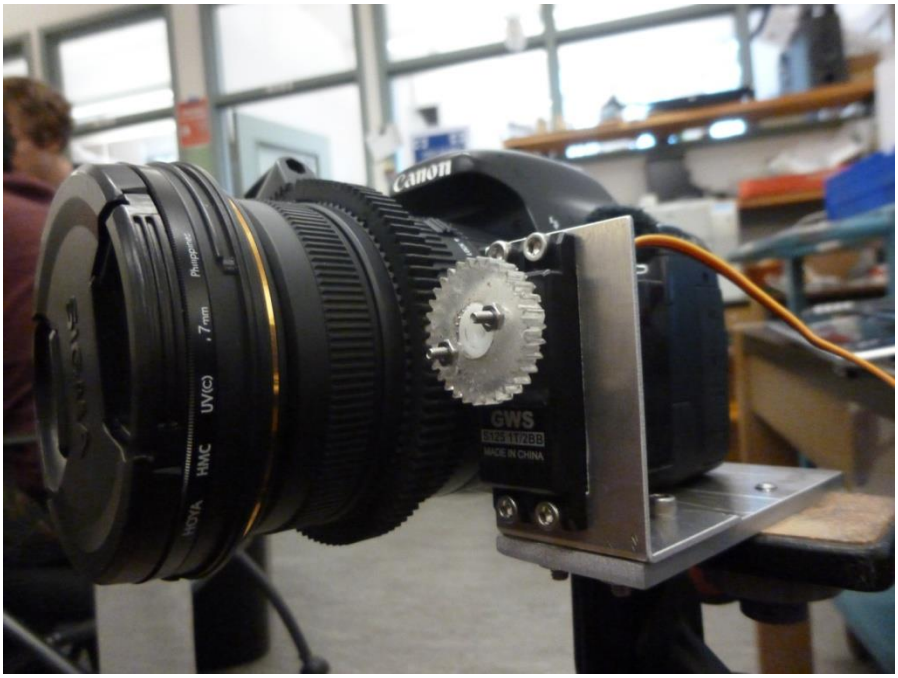
MOTORIZED ZOOM CONTROL



Daljit Bagri
Maurizio von flotow

Project 1314
Engineering Physics 459
Engineering Physics Project Laboratory
The University of British Columbia
April 2nd 2013

1.0 Executive Summary

This report is a complete and in-depth look into all of the work that went into building a motorized zoom controller for DSLR cameras. The key objectives of this project were to build a motorized zoom controller that would be much cheaper than any commercialized product of its type, while still retaining full functionality. As an additional objective, the design would allow for an autonomous control of zoom or focus during time lapse photography. This was done by iteratively designing, building,and debugging a mechanism from September 2012 to April 2013.

Each unsuccessful design gave light to a new more functional one. Upon finally assembling a sufficiently functional design, we began shooting test footage of both time lapse and zoom control and further improve the design. By the end of the allotted project work time, there remain a few bugs. But the design has accomplished its main goals.

We have created a prototype device that is capable of adapting to multiple lens geometries to somewhat smoothly control the zoom with adequate speed proportional to a rocker potentiometer. The very same device is capable of smoothly changing the focus of a camera lens while shooting time lapse photography, allowing for uncommon and eye-capturing time lapse footage. The smallest tested focus steps were about 0.21°, which was nearly the exact step size proposed in our project proposal (submitted Fall of 2012) as an adequately small step for smooth zoom or focus. This focus control is easily adaptable to different time lapse settings using a simple Arduino code.

The project is left somewhat incomplete, with the main necessary additions being the elimination of jerkiness in the zoom motion, a glitch in the servo control code, mounting the electronics and power supply more portably, and some further time lapse control tests. The future of this device is bright with relatively easy implementation of wireless control and the possibility of swapping motors to a more suitable one.

Table of Contents

List of Figures

2.0 Introduction

The motivation for this project comes from a recent surge in the popularity of DSLR cameras for shooting video. One large draw back from DSLRs versus conventional camcorders is the lack of ability to "smoothly" transition the field of view (or zoom) during video recording. In cinematography for large production films, changing the field of view during a shot is uncommon, and even frowned upon. Chris Gallagher, a film production professor at UBC, says that is not the case in documentary filming. *"[For filming a] documentary it [a zoom] is essential as things are more unexpected and one often needs to reframe during a shot. Shooting documentary with the DSLR is difficult as one cannot execute a smooth zoom. Often in documentary shooting one needs to zoom very slowly and smoothly hoping the viewer will not notice the zoom and without more sophisticated control that is impossible on a DSLR."*

The major market for this design would be in sports cinematography (especially action sports, where the object/person you are filming is radically changing positions), the news, timelapse photography, and as Gallagher added, documentary cinematography

There are few current solutions for this problem:

i. Zoom directly with your hand on the lens
ii. "Follow-focus" style mechanism
iii. Motorized fluid-zoom control

<u>i) Zoom directly with your hand</u>

This is a very crude method that produces footage with shaky, non-repeatable, and variable/jerking speed fluctuations in the field of view. All these aspects lead to unusable footage.

<u>ii) "Follow-Focus" mechanism</u>

Follow focuses are widely used to create precise focus transitions. They are usually mechanisms that gear down your hand's rotational velocity to produce a more smooth transition between focus points. Many also have marked positions to allow for repeatability of precise focus locations. These devices can be used to actuate the zoom ring, accompanied by the following flaws:

- The change in field of view is not directly proportional to the rotation speed of the zoom ring
- Repeatability of zoom position transitions is challenging
- "Feathered" stops and starts are challenging, or even impossible. A feathered start/stop is when the rotational velocity of the gear ring gradually increases from 0 to the desired speed, rather than instantaneously jumping to the full speed.

Figures 1 and 2 show examples of follow focus mechanisms adapted for zoom use.

Figure 1:Chrosziel's Fluid-Zoom drivers

Figure 2:Chrosziel's Fluid-Zoom drivers

iii) Motorized fluid-zoom control

This is the approach we designed for. There are very few products of this type available. The main flaws for a motorized zoom are:

- The noise of the motor could ruin audio of the footage
- Portability
- Battery life
- Exposure changes (as you decrease the field of view, you decrease the number of photons reaching the camera's sensor)

Figure 3 and 4 show the only two motorized zoom control mechanisms that we were able to find, both of which are still in development, and are currently not available for purchase.

There are a few lenses currently being produced with built in motors, but they are in a much higher price range (>$20,000).

Figure 3: Camera Turret's "Feather Touch" motorized zoom controller.

Figure 4: Viztools' "Handizoom" motorized zoom controller, which includes an adjustable zoom speed, programmable soft stops, iris controls, and more.

2.1 Scope and Limitations

The scope of this report is to fully examine the construction of a motorized zoom controller for a DSLR camera. This report will cover all of the iterations taken in order to reach our final model, and will also cover all of the issues encountered during construction of all of the models created.We will discuss the Arduino microprocessor that we have used, as well as the servo mechanism and the rocker potentiometer used in conjunction with it. We will also be discussing the code used to: drive our servo proportional to speed rather than position, slow down the rotation for smoother zoom, and for the implementation of time lapse photography.

This report does not cover the linearity of the focus or zoom of a DSLR lens as a function of its corresponding control ring's rotation. In other words, we do not explore whether 2 degrees of rotation at the beginning of the focus/zoom travel changing the focus/zoom more or less than 2 degrees at the end of travel. As well, this report does not cover any audio muffling of the motorized zoom mechanism. Both of these items are only discussed briefly in the recommendations section.

3.0 Discussion

In this section of the report we will discuss the evidence and arguments for: our methods and testing protocol for the mechanical, electrical and software portions of the project, and a discussion of our results.

3.1 Methods/Testing Protocol

The following sections outline in detail the mechanical, electrical, and software portions of the project. They also explain the troubles we came across, what we did to fix them, and what we went with for final designs of each of these areas.

3.1.1 Mechanical

At the beginning of the project very simple methods were used to test technical requirements such as torque required to rotate the camera lens. This was done by wrapping a zip tie around the lens of the camera attached to a fish scale to roughly measure the force needed to turn the lens. The Forces (~2lbs) was then multiplied by the radius of the camera lens (1.5in) to get the required torque to turn the lens.We concluded that the motor torque required for our design would be 3lbs-in. As well, we visited multiple camera shops around the area to examine different zoom lenses. We noticed most lenses had a zoom ring diameter between 2.5in and 3.5in, and all zoom lenses we evaluated required at most 90° of travel for a full zoom.

Along with final design decisions, these tests helped us choose a servo motor that would meet requirements for torque and rotation so that we would not be limited by the gear ratio. We decided on 5 different servos. From here our decision was made based on online reviews and price comparisons.

<u>i) Iteration 1</u>

After these initial tests we moved forward with fabrication of our first prototype; we figured building and debugging early is the best way to proceed. We put together a Solidworks model of our first prototype (as seen in Figure 5 below), and quickly ran into problems. The main idea behind this iteration was to mount a servo directly onto an adjustable rack that would mount to a stationary part of the camera lens itself. The idea to mount the device to the lens itself came from a suggestion made by Professor Gallagher of the UBC Film Production Department.

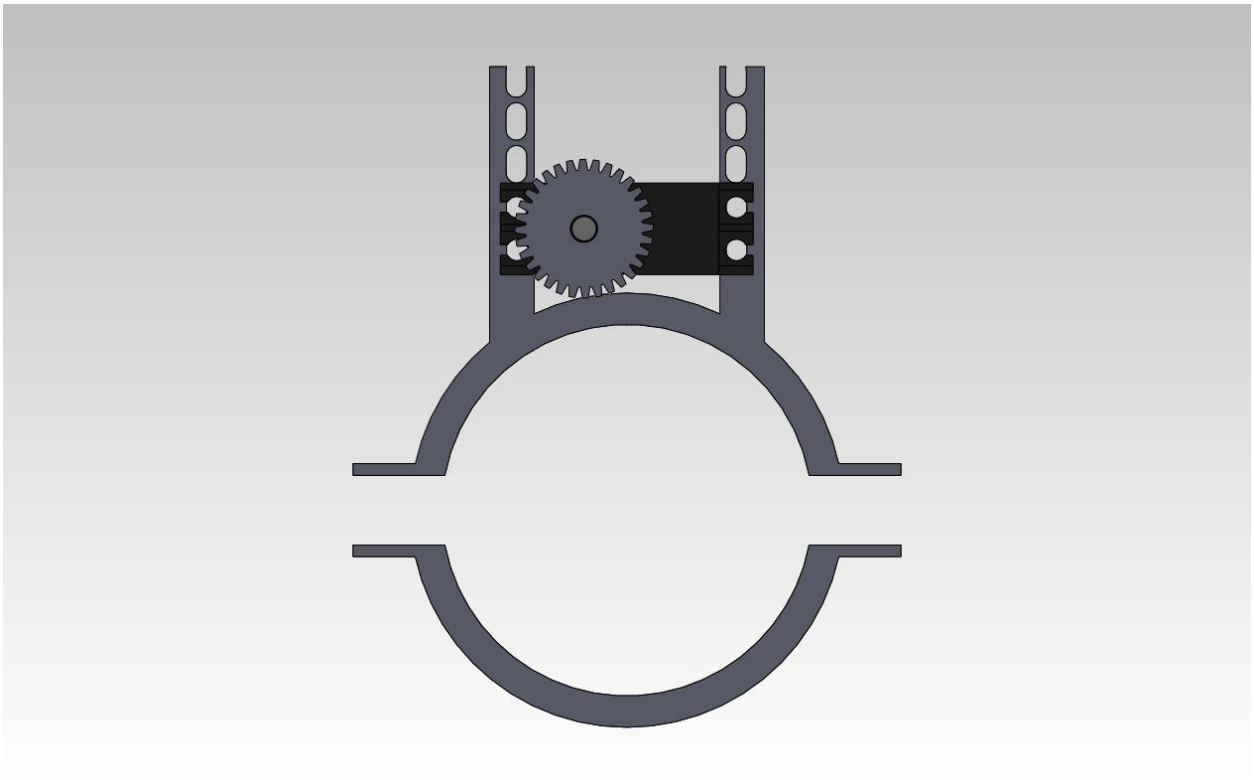


Figure 5: The First Prototype

The two main problems that we came across in this model were:

1. The material choice was too brittle and fractured during fabrication
2. Mounting the device to the non-rotating part of the lens was impossible without blocking crucial buttons (see figure 6 below for an example).

Figure 6: Blocking of Button Functionality (first prototype)

ii) Iteration 2

After encountering these problems a quick decision was made to redesign and use aluminum cut by a water jet rather than Durand cut by the laser cutter. The redesigned model called for a new mounting location and it was decided that the rotating part of the lens would be this location. The main feature of this design was a servo enclosed between two rails,both able to rotate with the lens, while the servo itself was constrained from movement (model shown in Figure 7 below).

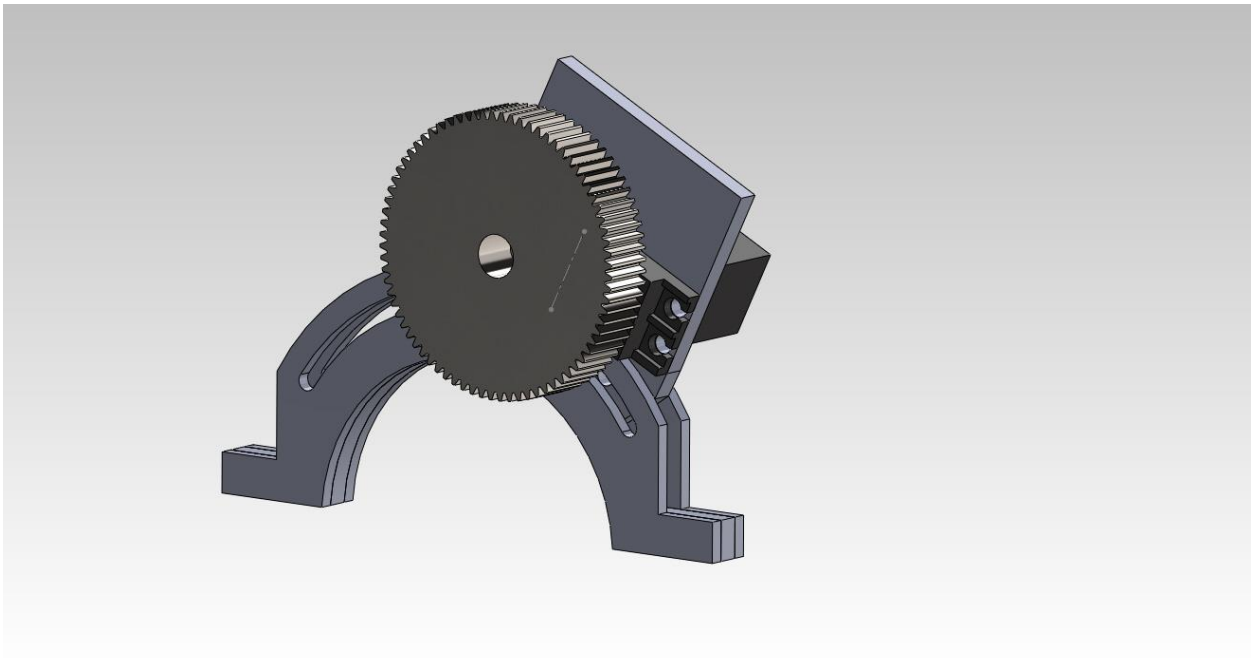


Figure 7: Iteration 2

Once again this prototype came along with its problems. The main problem was the instability and extra play in the servo mount. It caused extremely jerky zoom motions and jamming of the assembly. Another fairly detrimental problem was that even though the gear we were using had enough force to smoothly rotate the lens, the size of the gear was too large to be able to zoom at the desired slow speeds (large gear seen in Figure 8).The jerky and extremely fast zoom motions of Iteration 2 were documented in the following video: https://vimeo.com/63221297.

As you can see from the video above, the zoom motions that resulted from Iteration 2 were both jerky and extremely quick. These problems were worked on to no avail until the third and final iteration.

Figure 8: Size of gear to large

iii) Iteration 3

The final iteration of our design incorporated a complete redesign of our product. After the unusable gear ratios of the previous iterations, we decided our servo would have to me much closer to the lens. To solve the problem of instability, we used a new mounting plate that attached to ¼-20 screw hole in the bottom of the camera (this is where the tripod mount usually attaches, Figure 9 below shows this final design).

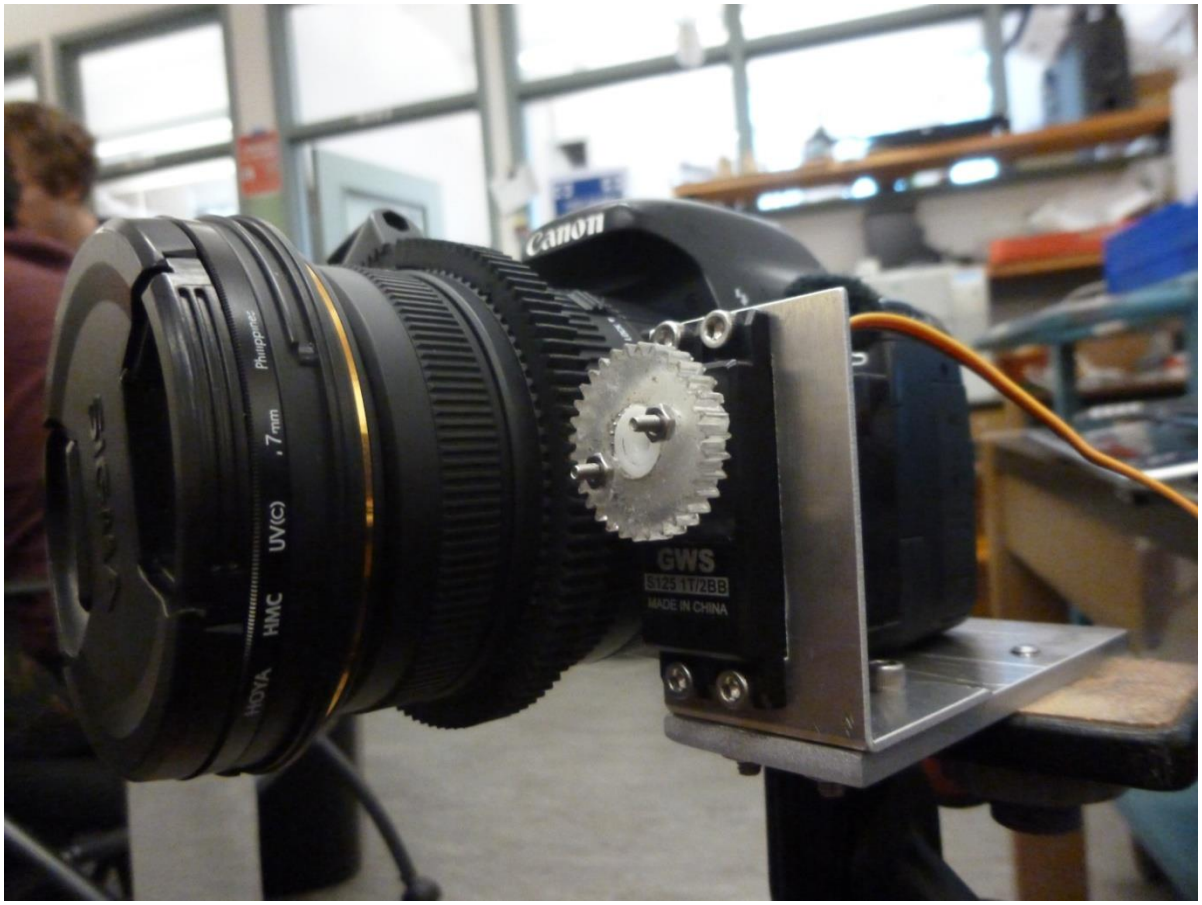


Figure 9: Final Design

This new design reduced our pinion gear from 53 teeth to 31 teeth, a more than 40% reduction. This reduction is clearly seen if Figures 8 and 9 are compared, and helped to increase the rotational resolution of the lens. The base plate is a multipurpose design: besides the obvious stability it provides for the servo bracket, the camera is mounted to it using a slot in the bottom such that the servo can be easily moved along the lens to actuate both the focus and zoom rings of many lenses (not all lenses have the same axial positioning of their zoom/focus rings). The servo is also mounted to the base plate using slots to account for different diameter lenses (Figures 10 and 11, below). The device was tested using multiple different lenses to ensure this functionality was correct (Figure 11).

Throughout the iteration process, we recorded multiple video tests to check our zoom actuation. Some of these tests are discussed in section 3.2 of this report.

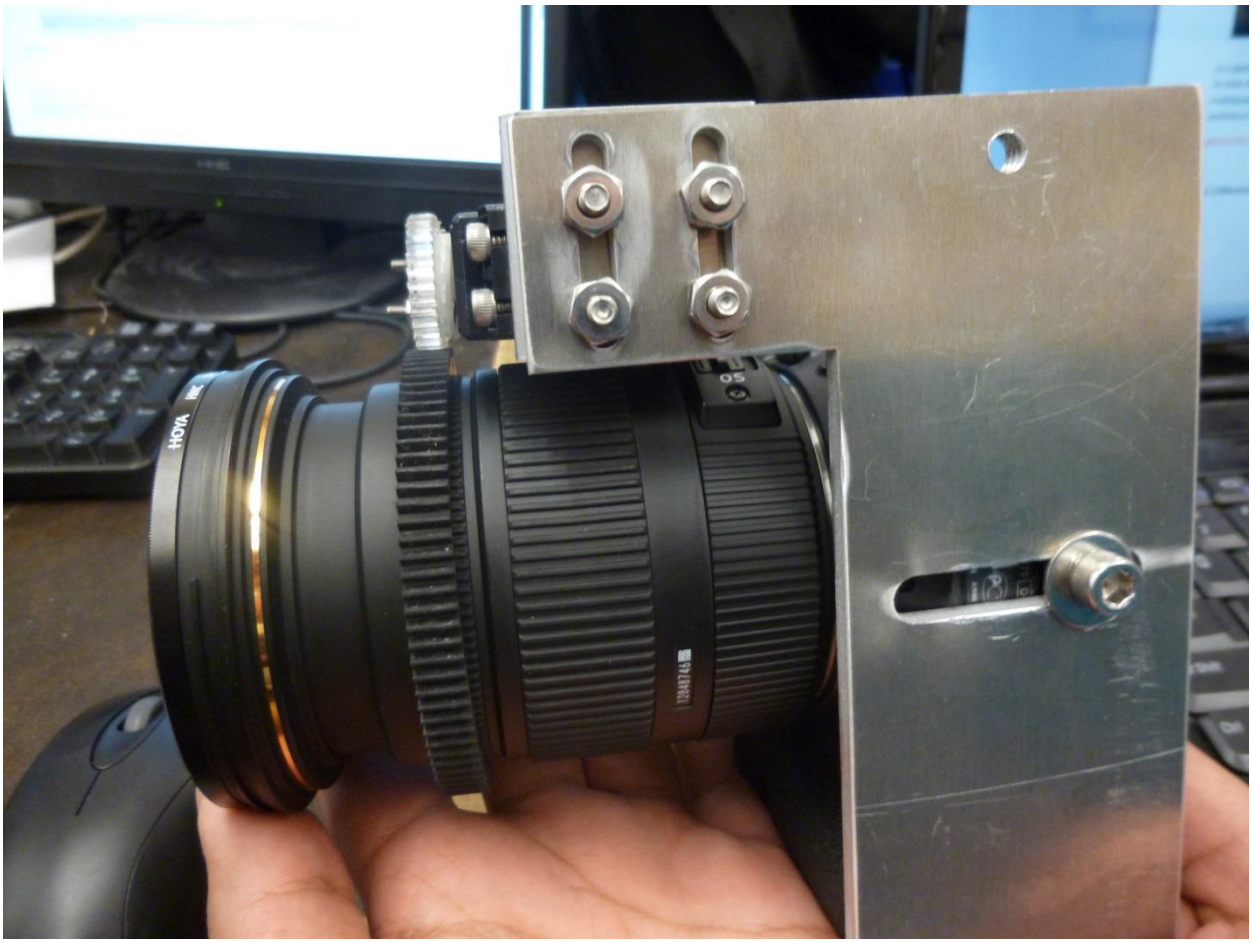


Figure 10: Slots for adjusting base plate and servo locations

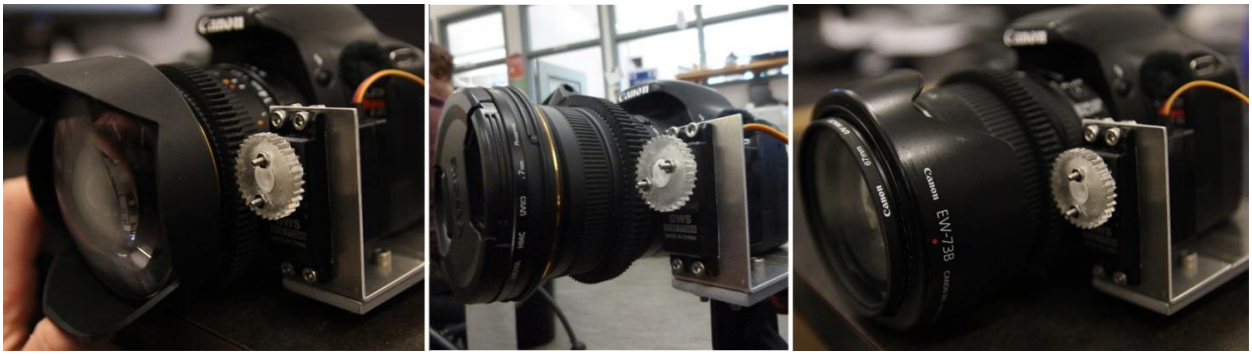


Figure 11: Mounting to different lenses

3.1.2 Electrical

The electrical portion of this project was much less time consuming compared to the mechanical portion, yet still very important. The components involve a sail winch servo, a rocker potentiometer, an Arduino microprocessor, an external power supply, and a small circuit used to connect all of these components based on which functionality was needed.

We began by building the circuit with an Arduino UNO and a joy stick rocker potentiometer that we had taken from a RC plane controller. This joy stick rocker POT was meant to be a temporary measure before the traditional rocker POT we had ordered came in, but it turned out we were unable solder the traditional POT into our circuit and thus had to continue using the joy stick rocker POT for the rest of our testing.

During testing we ran into two main problems that we believed were due to our electrical set up.

i) Shunt Resistor Clutch

The first was the fact that even after the zoom ring had turned a total of 90°, the servo could still run and potentially damage a component of our system or the camera lens. We tried to correct this by implementing a clutch in the form of a soft stop. The soft stop was created using a 1 Ohm current sensingShunt Resistor. The Arduino read the voltage drop across the Shunt Resistor from the servo's ground to common ground, as shown in Figure 12.From this voltage drop, and by knowing the size of the Shunt Resistor, we were able to detect current spikes sent to the servo. The idea was that when the servo reached the end of the lens' travel, it would stall and draw a spike of current. We would detect this current spike and disable the servo's power.

However the current sent to the servo was not consistent enough for us to smoothly implement this clutch. With a little more time, it is believed we could get these soft stops working with some more sophisticated coding, but they have been disabled for now.

ii) Servo Glitch Moves

The next issue we had was a glitch in the servo functionality;we noticed that anytime we ran the servo in a particular direction and let go of the joy stick allowing it to snap back to its neutral position, the servo would run at full speed in the opposite direction we had just been driving it. We tried to correct this by placing a capacitor in our circuit between the variable resistor and the Arduino to eliminate resistance spikes in the POT values (This circuit can be shown in figure 12 below, the capacitor is shown in green and the Shunt resistor is shown in red). This modification did little to no help in this glitch, and we thus concluded it must have been coding problem we were unable to detect.

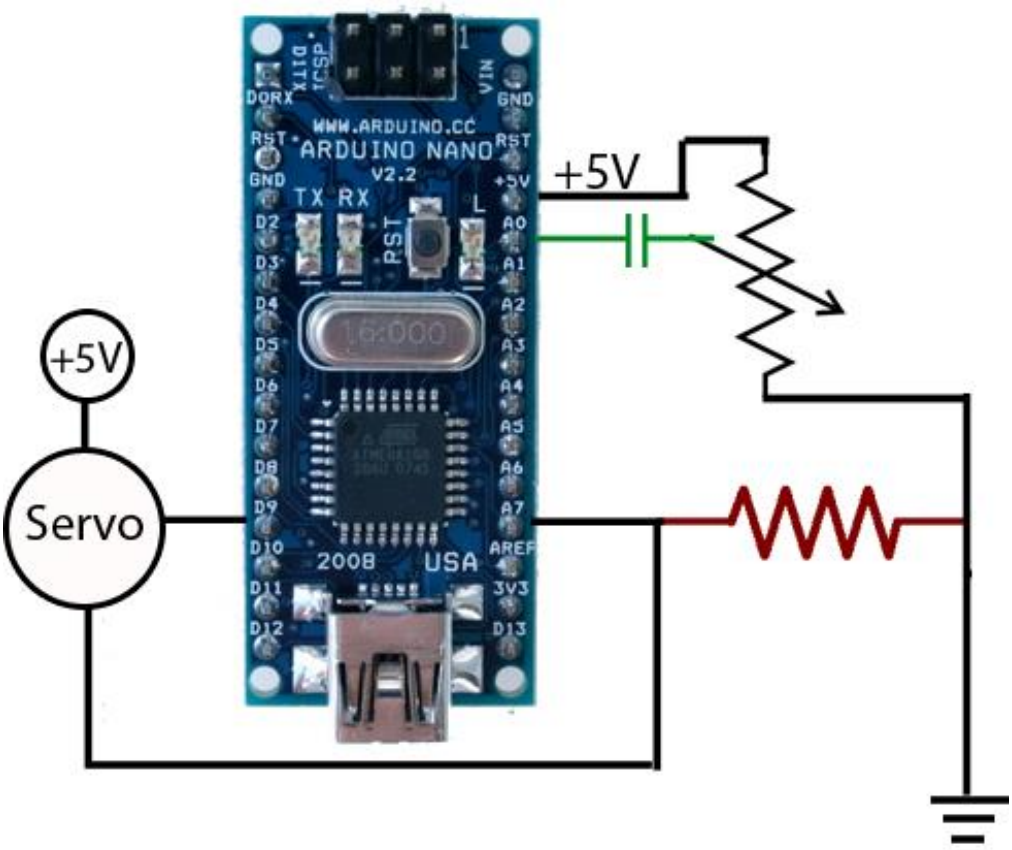


Figure 12: Video zoom control circuit diagram

After getting all of the electronics to work in conjunction with each other, we decided to switch from the Arduino UNO to the smaller NANO. This Arduino is much smaller and if we had time to create our own printed circuit board, the overall size would be greatly decreased (the sizes are compared below in Figure 13).

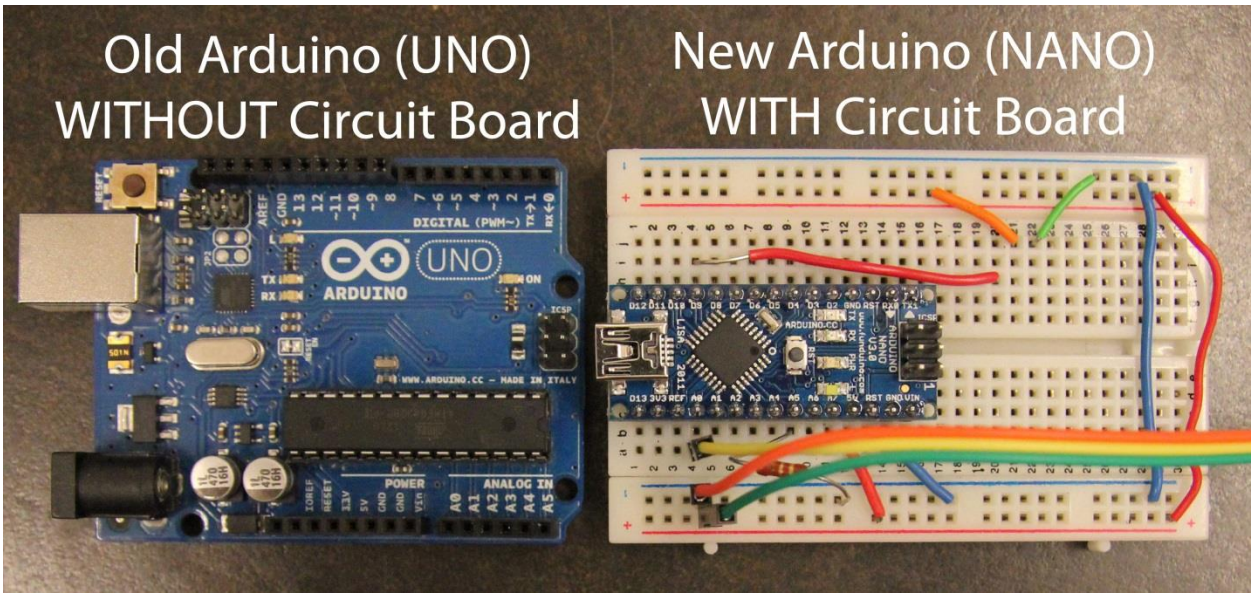


Figure 13: Size Comparison between the Arduino UNO and NANO

3.1.3 Software

Video Zoom Control – VarSpeedServo Library

Initially, we were worried about the feasibility of controlling the *speed* of a servo with a potentiometer. However, thanks to the Arduino forums, we found this was already a much solved problem. Some kind soul had provided an entire Arduino library, called "VarSpeedServo", that allows you to give a servo speed commands instead of position commands.

What this library does (i.e. the work this library saved us) is it maps your given speed command (from 0 to 28) into a pulse width. The domain that this pulse width is mapped into is varied based on your specific servo's maximum and minimum pulse widths (these pulse widths you tell the code in the form of Arduino sketch constants).

Our Implementation

From here, we simply used the Arduino to read a POT value within a loop and varied the speed command based on this constantly updating value. To account for any imperfections in the POT's neutral position value, we added an appropriately sized buffer resistance into the code close to the POT's neutral position resistance.

Time Lapse Focus Control – Servo Library

The time lapse focus control code was much more basic to write. It is essentially a servo sweep (moving from one end of travel to the other in incremental steps, with a delay between each step), taken from the stock Arduino example sketches, with a larger delay and smaller steps. This delay was turned into a

constant that corresponds to the picture interval time. As well, the step size of each step of the sweep had to be varied. More on this procedure is described in the following section 3.2

Both of these Arduino sketches can be found in Appendix B.

3.2 Results and Discussion

a) Video Zoom Control:

The primary function we hoped to accomplish with this project was a smooth zoom control using a rocker POT. After the multiple iterations described above, we produced a semi-functional prototype.

i) Adaptable to Multiple Lenses

The flexible ring gear used to actuate the lens rotation can be fit to virtually any diameter lens. As well, the slots described in section 3.1.1 allow for the servo's pinion gear to mate with variable axially located zoom *and* focus rings as well as lens diameters.

ii) Zoom Speed Proportional to POT Position

Using the code described in 3.1.3 and located in Appendix A, we were able to vary the speed of the servo proportionally with the value of the POT. The desired "feathered" starts and stops were perfectly executed. Multiple field tests were performed to test the functionality of the zoom control, and a few were collaborated here:https://vimeo.com/63221298

As you can see in Slow Zoom Test 2, the zoom control can be fully actuated at both low and high speeds with the joystick rocker POT. However it is also apparent that the device still has some issues in its smooth actuation.

iii) Glitches and Jerky Motion

Due most probably to flex, play, and/or misalignment in the design, the servo actuates the zoom ring in jerky motions. If this problem persists, it completely destroys the smooth zoom functionality of the device as it eliminates any smooth zooming. The jerkiness was first thought to be misalignment of the pinion gear on the servo shaft. However the jerks occurred more frequently than once per revolution and thus could not be due to this. The jerky motion was recorded and can be seen in the Slow Zoom Test 2 video above as well as here: https://vimeo.com/63115341.

As seen most clearly in the full screen image of the video linked above, the jerky motion causes unpleasant and mostly unusable footage. What the last two video *do* show, however, is that the speed of the zoom transitions can be performed at adequately slow and fast rates using this design.

The servo glitch issue was described in section 3.1.2, along with our attempts to fix it. With careful driving of the device, it is possible to avoid this glitch. Discussion on the future efforts to fix this glitch can be found in section 6.0 of this report.

b) Time Lapse Focus Control:

Initially we thought we could distinguish our design from other available solutions by offering zoom control during time lapse photography. While this is possible with our design, we decided to pursue a more desirable time lapse control feature. After email discussion with Gunther Wegner, a professional time lapse photographer, we were advised that focus control during time lapses is more sought after than zoom control. Our design was then driven to include this feature.

i) Larger Step Size

Initially, we were disappointed with the resolution of our servo motor for focus control. At its smallest step size, the servo would actuate the focus ring through its entire travel in 45 steps (about 0.5° steps). This means if we wanted to shoot a time lapse with a picture taken every second and perfectly smooth focus transitions throughout, we would only be able to take 45 pictures before the end of focus travel. This translates to 1.5 seconds of video footage after compiling the pictures into a 30 frame per second sequence.

To compensate, we hypothesized that we did not need to step the focus between *every* shot but rather we could take a step every, say, 8 pictures. So we tried that in a focus control test time lapse: https://vimeo.com/63115342

Figure 14: Screen Shot from Time Lapse Focus Control, Test 1

Time Lapse Focus Control, Test 1:

Time Lapse Control:
- -1 picture very 1 second
- -Time lapse duration ≈ 6min

Servo Control:
- -Step size ≈ 0.5°
- -Servo steps once per 8 pictures taken (otherwise focus ring reaches end of travel too quickly)
- -Results in jerky focus transitions

As you can see in the first test video of the time lapse focus control, actuating the focus ring 0.5° every 8 pictures resulted in unsatisfactory and jerky focus transitions throughout the video sequence. This shows that for a pleasantly smooth transition in focus throughout a time lapse, there must be focus ring rotations more frequently than every 8 pictures.

But exactly frequent is frequent enough to create a smooth transition? To avoid this question, we tried to get a step size small enough to be able to actuate the focus ring between *every* picture taken.

ii) Smaller Step Size

To obtain a smaller step size, we began by examining the library that Arduino provides to control servos (we are using this Servo Library to control our servo in the time lapse control). Initially, the for-loop in the Arduino sketch responsible for stepping the servo looked like this:

```
for(pos = 1; pos< 180; pos += StepSize)   // goes from 1 degrees to 180 degrees in steps of
                                          "StepSize" degrees

{
myservo.write(pos);                       // tell servo to go to position in variable 'pos'

delay(PicInterval*1000);                  // waits (PicInterval) seconds for the next picture

 }
```

In the Servo Library, the function *write(value)* first assumes *value* to be an integer between 0 and 180, then maps that *value* into a domain between *SERVO_MIN* and *SERVO_MAX (*the minimum and maximum pulse widths for the servo). It then uses a function called *writeMicroseconds(value)* to assign a pulse width to the servo based on *value*. The*write(value)* function is as follows:

```
void Servo::write(int value)
{
   …
value = map(value, 0, 180, SERVO_MIN(),  SERVO_MAX());        //Re-maps "value"
this->writeMicroseconds(value);                               //Calls the pulse width commanding
                                                              function
}
```

Because *value* had to be an integer, it was at first impossible to move the servo at any step size smaller than 0.5°. However, we only had to change the mapping command to map *value* into a larger domain than 0 to 180, such that each step we command is seen as a smaller step to the function *writeMicroseconds(value).* We tried different orders of magnitude, and found that a map from 0 to 18000 resulted in adequately small step sizes. The updated Arduino Servo library now looks as follows:

```
void Servo::write(int value)
{
   …
value = map(value, 0, 18000, SERVO_MIN(),  SERVO_MAX());   //Re-maps "value"
```

this->writeMicroseconds(value); *//Calls the pulse width commanding function*

}

This small change in code allowed for a much finer resolution of focus control. With this specific mapping of *value*, it takes about 420 steps to traverse the entire focus ring travel, or a step size of about 0.21°. A discussion of how small we could command these steps can be found in Section 6.0. We performed a time lapse focus control test with these finer resolution steps and were pleased with the outcome: https://vimeo.com/63141684.

Time Lapse Focus Control, Test 2:

Time Lapse Control:
- -1 picture very 1 second
- -Time lapse duration ≈ 7min of focus control

Servo Control:
- -Step size ≈ 0.21°
- -Servo steps once per picture taken
- -Results in smooth focus transitions
- -Servo unplugged when end of travel reached to allow time lapse to finish (sun to set), did not disrupt shooting

Figure 15: Screen shot from Time Lapse Focus Control, Test 2

Notice the smooth transition in focus throughout the initial time lapse duration. As well, once the focus ring had been actuated to the desired location (such that the background was in focus), we were able to easily deactivate the servo and continue shooting the time lapse until the sun had set without disrupting the shots; a very handy feature.

In our initial project proposal submitted in the Fall of 2012, we proposed the adequately small step necessary for smooth zoom or focus control was 0.2°. This value came from a crude test where we judged the changes in zoom and focus after small rotations of their control rings, then estimated that angle. After obtaining a step size of 0.21°, this portion of the project is considered a success.

Further testing will be applied in the following months to determine how fine of resolution can be obtained by increasing the domain mapping of our position *value*.

4.0 Conclusions

There were two main sections of this project: time lapse focus control and video zoom control. This section of the report will summarize the conclusions made from each.

4.1 Video Zoom

The video zoom control was semi-successful. We produced a design that was compact, usable on multiple lenses and camera bodies, but only semi-functional. At its seemingly slowest speed, the servo actuated the zoom ring of a Sigma 17-50mm (mid-range zoom lens) through its full range of motion in about 10 seconds. This translates into what seems to be an adequately slow zoom motion, while the fastest zoom motion was faster than necessary for quick zoom transitions.

The dysfunctional part of the zoom control is the jerky motion produced. Caused, we believe, by flexibility and play in the design; the servo periodically speeds up and slows down. This pattern repeats itself 5-10 times throughout an entire zoom motion. With this jerky motion, the zoom control becomes somewhat useless. As well, there is a repeatable glitch in our software/circuit somewhere that makes our servo drive to its start position at full speed. This glitch is triggered when the joystick POT "snaps back" to its home position. We will work to address these issues in the coming weeks. Video tests have been provided in section 3.2 Results showing this video zoom control.

4.2 Time Lapse Focus

The time lapse focus control was a complete success. We were able to step our focus ring about 0.21° at the smallest tested interval. This minimum step size could theoretically be made even smaller with a quick modification of the Arduino Servo library, but as is meets the proposed adequately small step size for smooth motion introduced in the Fall of 2012 project proposal. Instead of using a GUI, we created a simple Arduino sketch with easy to replace constants that control the servo's movements. This sketch controls the interval between movements, total angle of rotation, actuation duration, and the size of each step taken by the servo. Video tests have been provided in section 3.2 Results showing this time lapse focus control.

5.0 Project Deliverables

The deliverables for this project are organized below in primarily as a list of deliverables and then a short financial summary as well.

5.1 List of Deliverables

Our initial list of deliverables from the project proposal was fairly optimistic, but we were able to complete all deliverables in some form. The list of deliverables ended up as follows:

a) Prototype device:

   Our final prototype is functional in the time lapse setting, but still provides a jerky zoom motion. As well, the rocker potentiometer is functional but not ideal for portability (see videos in section 3.2 Results).

b) GUI that can determine the step size and frequency necessary for specific time lapses:

   We did create a GUI for this purpose, but decided it would be more convenient to have this in the form of an Arduino sketch with constants we can vary (See Appendix A)

c) Solidworks drawings of complete design:

   Complete, along with Solidworks drawings for each iteration.

d) Circuit diagrams of every circuit:

   Complete, See Appendix B.

e) Complete list of components and pricing:

   Complete, see section 5.2 Financial Summary.

f) Multiple field tests for various configurations, iterations, and stepper motor settings:

   Complete, and recorded in the form of video and time lapse tests (obviously using a servo, not stepper motor). Some of these videos have been uploaded to the web, and can be found in section 3.2 Results.

5.2 Financial Summary

#	Description	Quantity Purchased	Quantity Destroyed	Vendor	Cost per unit	Purchased by
1	Rocker POT	2	1	uscamera.com	$23.71	PL (Jon)
2	Arduino UNO	1	0	Lee's Electronics	~$30	Maurizio
3	ArduinoNANO	1	0	?	~$10	PL
4	Joystick Rocker POT	1 (taken from retired RC controller)	0	--	--	--
5	Sail Winch Servo	3	1	pololu.com	$19.95	PL (Jon)
6	Waterjetted parts	~20 minutes	--	--	~$1/min	PL
7	Flexible Ring Gear	1	0	F&V USA	$19	Maurizio

6.0 Ongoing Commitments and Recommendations

Current Design Fixes

As a self-sponsored project, we will continue to work on this project for our own benefit. Maurizio hopes to use this device in his videography work, in both time lapse focus control and video zoom control. Although the focus control is working well, the zoom functionality is performing poorly. Specifically, the concerns we wish to address are:

-Eliminate the jerky zoom motion by reducing the flex and play in the design. The first improvement we'll implement is putting a brace in the 90° bend of the servo mount.

-Replace the joystick rocker POT with a traditional rocker POT with variable mounting positions. We'll first need to acquire a functional traditional rocker POT.

-For portability, mount the Arduino MINI to the base plate of the device and power the Arduino with a battery instead of a computer's USB port.

-Use a function for the change in step size for focus control, such that we may remain focused on an object for some time before continuing with the focus-to-infinity.

-Test the limit of how small we can make the servo step size. The larger the domain of the mapping of our position value, the smaller the step each position increment will create. The lower limit for this should be related to the minimum pulse width our servo can accept.

These concerns will bring our design to the functionality we initially planned and will be the main focus of our future work. They will be addressed by the end of May so the device can be used for videography projects throughout the summer. In addition to these concerns, we see fit other future design extensions.

Future Additions

Wireless Control

Eventually (hopefully by the end of the summer), we would like to introduce wireless control to the zoom functionality. One fast growing practice where this could be helpful is in multi-copter videography. Many of these UAVs are flying DSLR cameras with interchangeable lenses, as they are much lighter than traditional camcorders. Currently it is impractical and unusual to vary focus or zoom during flight. This could, however, cause a complication in needing yet another operator. Any multi-copter that flies a camera on a gimbal requires one operator for the UAV and one for gimbal driving. If we wish to make a design useful for these flights, it will be necessary to integrate our controls with those of the gimbal to eliminate the need for a third operator.

Motor Replacement

Although the servo we have chosen is working well for the time lapse control we have tested, there is much room to improve. The time lapse durations we tested our design with were relatively short; many people shoot time lapses that are hours, days, or even months in duration. To produce smooth

focus transitions in these time lapses, we would need a much higher precision motor. Since our servo seemed to have more than enough torque, it could be feasible to replace our servo with a higher-precision lower-torque one. Alternatively, strictly for time lapse control, we could revert to our original idea of using a stepper motor with either very small steps and/or implement micro-stepping.

Focus/Zoom Linearity

One test that was discussed multiple times throughout this project but never tested was the linearity of the zoom and focus throughout their corresponding control ring's rotation. A worthwhile test for future work would be to test this linearity. If, in fact, it is found that the focus and/or zoom do *not* change linearly with the rotation of their control rings, it would be necessary to implement code to change the rotation of these rings such that the focus and/or zoom respond linearly to the POT commands.

Audio Muffling

Especially with the physical placement of the servo motor with respect to the on-camera microphone of most DSLR cameras, muffling the audio of this servo will be a very important addition to this design in future iterations. Without doing this, an external audio source will be necessary for many applications.

7.0 Appendices

The following is additional information on the coding and electrical aspects of the motorized zoom controller.

7.1 Appendix A – Coding

Time Lapse Focus Control

```
#include <Servo.h>

constintServoPin = 9;                    //Servo Pin
constintFocusRingRotation = 70;          //Total desired rotation of the focus ring, in degrees
constintPicInterval = 1;                 //Picture every __ seconds (ie take a ep every __ seconds)
constintTLDuration = 5;            //Desired time lapse shooting duration, in minutes

Servo myservo;                           // create servo object to control a servo

intStepSize = FocusRingRotation*PicInterval/(60*TLDuration);   //Equation for calculating the step size.
                                                               //This will eventually be replaced by a
                                                               //function for variable step sizes.
intpos = 0;                              //Variable to store the servo position

void setup()
{
myservo.attach(ServoPin);                // attaches the servo on pin 9 to the servo object
}


void loop()
{
for(pos = 10; pos< 1800; pos += StepSize)  // goes from 0 degrees to 180 degrees in steps of "StepSize"
degrees
 {
myservo.write(pos);                      // tell servo to go to position in variable 'pos'
delay(PicInterval*1000);                 // waits (PicInterval)ms for the servo to reach the position
 }
for(pos = 1800; pos>=1; pos-=StepSize)    // goes from 180 degrees to 0 degrees
 {
myservo.write(pos);                      // tell servo to go to position in variable 'pos'
delay(PicInterval*1000);                 // waits (PicInterval)ms for the servo to reach the position
 }
}
```

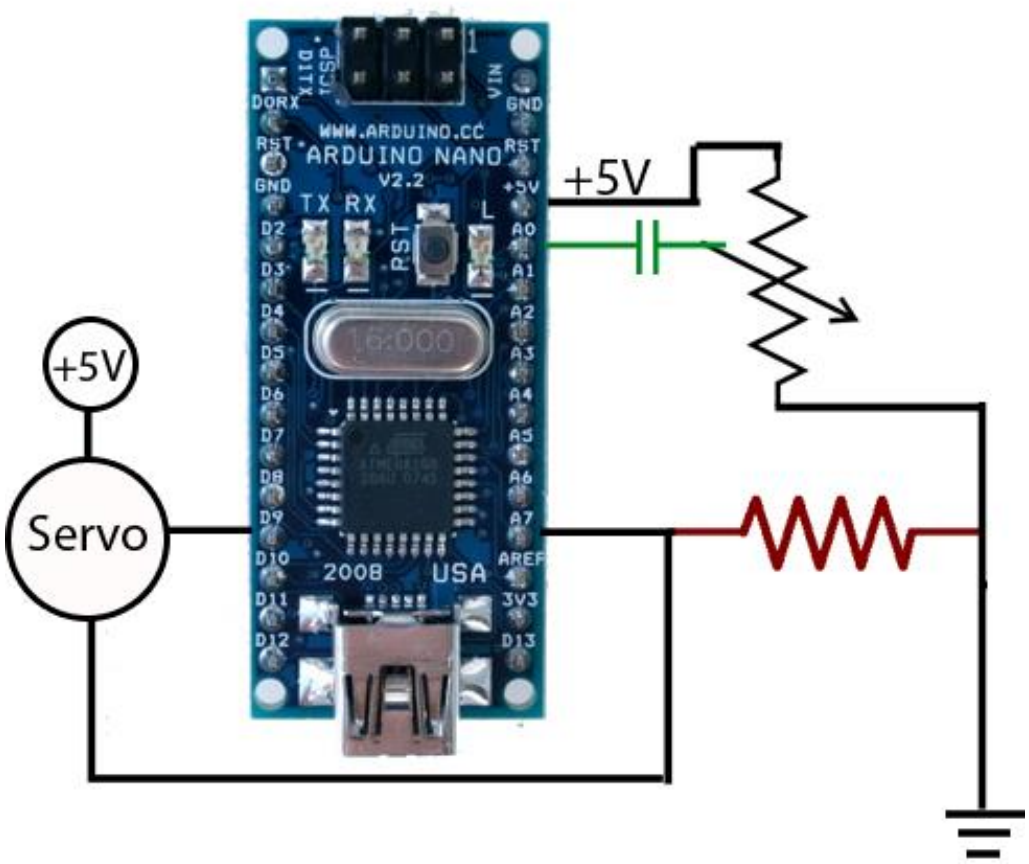
Video Zoom Control

```
#include <VarSpeedServo.h>

constintpotPin =  0;              // the number of the potentiometer pin (Analog 0)
constintmainPin = 9;              // Main servo control line
constintServoMin = 900;           // Minimum pulse width for servos in ms.
constintServoMax = 2100;          // Maximum pulse width for servos in ms.
constintPotiMin = 107;  // Minimum value read from potentiometer
constintPotiMax = 790;  // Maximum value read from potentiometer
constintPotiCenter = 445;         //Center POT value
constintPotiBuffer = 20;          //A buffer for the center value so the servo doesn't "creep"
constintSpeedScaler = 12;         //The upper level for the POT to speed map
constint RESISTANCE = 1;          //Resistance value of the current sensor resistor (in Ohms)
constintCurrentSensorPin = 5;   //Shunt resistor pin
intcurrenttargetvalue = 0 ;       //Current target value

VarSpeedServomyServo; // Servo

void setup() {
  // Initialize Servos
myServo.attach (mainPin, ServoMin, ServoMax);

  // Initialize Serial communication
Serial.begin (9600);
}


void loop() {

intpotRead = analogRead (potPin);                                   //Read POT
doublevoltageDrop =  analogRead( CurrentSensorPin );                //Read voltage drop across current
                                                                    //sensorresistor

  //If max/min POT values are above possible values, assign them to defined max/min vals
if ( potRead<PotiMin )
potRead = PotiMin;
if ( potRead>PotiMax )
potRead = PotiMax;

int speed = map (potRead, PotiMin, PotiMax, 1, SpeedScaler) & 0xff;

//Check stall current – this portion of the code was disabled as it was dysfunctional.
 /*
if(voltageDrop>= 5 )
  {
currenttargetvalue = myServo.read();          //I don't believe this value matters, as the speed will be
                                              //0 anyway.
```

```
speed = map (PotiCenter, PotiMin, PotiMax, 1, SpeedScaler) & 0xff;
myServo.slowmove( currenttargetvalue, speed);
 }
 */

 //Move backward
if(potRead<PotiCenter - PotiBuffer )
 {
  //Set target value to 0
currenttargetvalue = 0 ;
  //Map POT vals from PotiCenter-PotMin to speed ranges
int speed = map (analogRead( potPin ), PotiCenter, PotiMin, 1, SpeedScaler) & 0xff;

myServo.slowmove( currenttargetvalue, speed);
 }

//Move Forward
else if( potRead>PotiCenter + PotiBuffer )
 {
  //Set target value to 180
currenttargetvalue = 180;
  //Map POT vals from PotiCenter-PotMax to speed ranges
int speed = map (analogRead( potPin ), PotiCenter, PotiMax, 1, SpeedScaler) & 0xff;

myServo.slowmove( currenttargetvalue, speed);
 }

 //POT at 0 +/- buffer, don't move
else
 {
intcurrenttargetvalue = myServo.read();
speed = 14;                                          //oddly, the center value for speed is 14 (not 0)
myServo.slowmove( currenttargetvalue, speed);
 }

}
```

7.2 Appendix B – Circuits

Video Zoom Control Circuit



Time Lapse Focus Control Circuit