E155 Final Project
The µPainter

by
Austin Chun and Eyassu Shimelis
Dec. 9th, 2016

Abstract

The goal of this project was to accurately produce two-dimensional light paintings through long-exposure photography. The images had to be properly and consistently scaled, and needed to be easily uploaded and drawn. Thus an LED strip, driven by a Raspberry Pi, was mounted to a cart with positional measurements via rotary encoder. The rotary encoder data is processed by an FPGA and sent to the Raspberry Pi such that the LED strip updates properly, allowing for variable speed and bi-directional movement. A webserver hosted by the Raspberry Pi fetches an uploaded image and prepares the image for painting. The project was successful and produced detailed and properly scaled images in long-exposure photographs.

I. Introduction/Background

Light painting is a common activity in long-exposure photography. A long-exposure is a photo produced from opening the aperture for an extended period of time. During a long-exposure, it is possible to make paintings by moving a light source in the image. Most light-painting photographs use a point source that is used for "drawing" simple patterns (see Figure 1.1). However by using a strip of LEDs it is possible to draw images with a much wider stroke (see Figure 1.2). By quickly altering the pixel colors of the LEDs on a strip, it is possible to make more detailed, two-dimensional images. Current implementations of LED painting strips[1] have a timer-based system, which result in images that are improperly scaled and distorted based on the speed of the stroke.

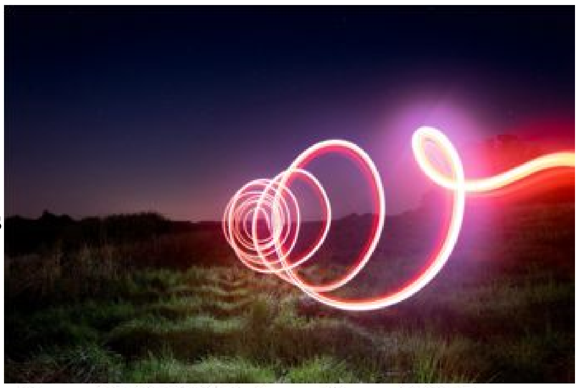
Figure 1.1: Light Painting with small light source

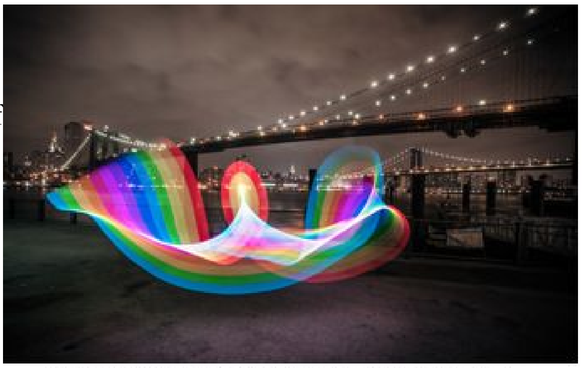
Figure 1.2: Light Painting with LED stick

The goal of the this project, the microPainter, is to create a device that more accurately produces two-dimensional images. Specifically, the microPainter must use positional measurements to appropriately update the LED strip so that the image remains accurately sized no matter the speed or direction (forwards or backwards) of the LED strip. Additionally, the device must support a simple user interface where a user can quickly upload an image to a website, then start painting.

To achieve these goals, the microPainter includes the following subsystems: a rotary encoder for positional measurements, an FPGA to process the rotary encoder data, an LED strip to paint the image, and a Raspberry Pi to host the web server, process images, read the FPGA data, and drive the LED strip accordingly. The overall block diagram of the system is shown below in Figure 1.3, and the schematic is shown in Figure 1.4.

[1] http://www.thepixelstick.com/

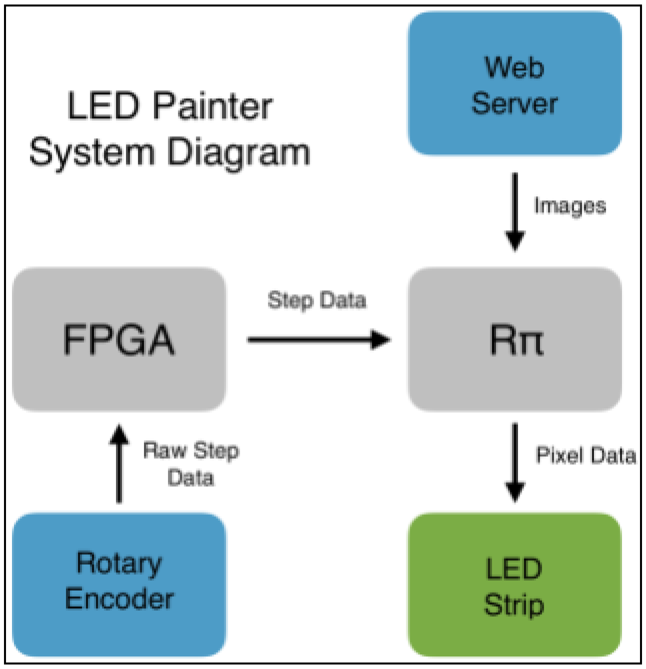

Figure 1.3: Overall Block Diagram of System

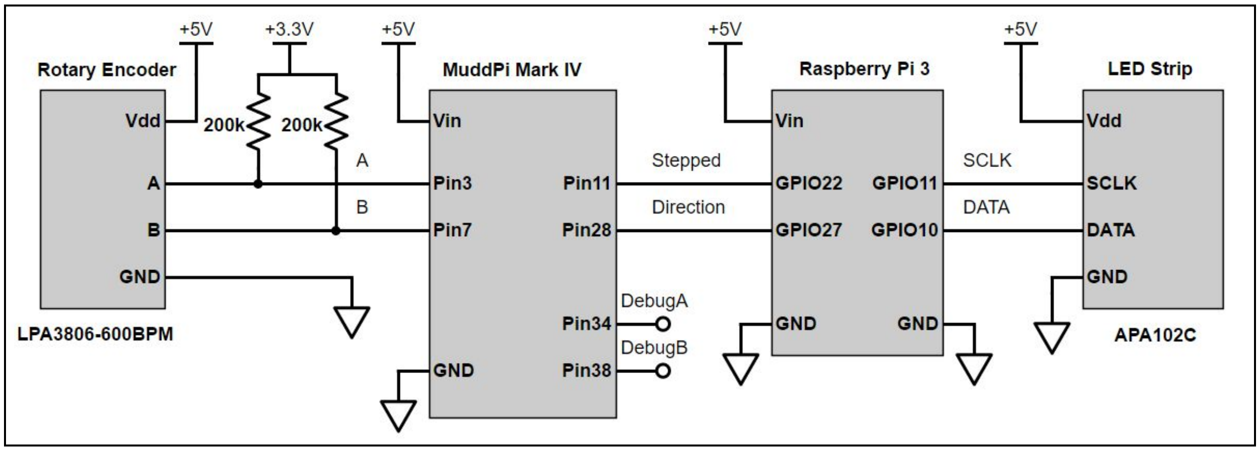

Figure 1.4: Schematic of System

II. Raspberry Pi

The Raspberry Pi handles three main functions: 1) hosting the web server 2) processing the image 3) and interfacing with the LED strip.

Web Server

The python web server[2] uses Flask (Python-based server framework) to accept, process, and paint user images. Users can simply drag-and-drop an image into the image queue, as seen in Figure 2.1. Once

[2] https://github.com/eshimelis/microPainter/blob/master/site/mainServer.py

an image is done uploading, the server, will process the image and start the main C program. Upon completion, the files are deleted and the site accepts new images.

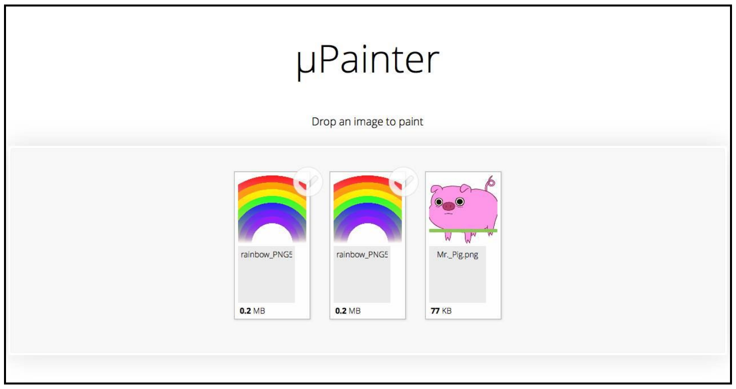

Figure 2.1: Front-end microPainter interface, used for image upload

Image Processing

A simple python file using the Pillow Image Processing Library loads a png file, scales the image to 144 pixels tall, and saves the file in bitmap format.

LED Interface - Hardware

A 144 LED/m LED strip carrying APA102C[3] IC chips was used because unlike the WS2812 (used in NeoPixels), the APA102 is driven with true SPI protocol, allowing for adjustable clock speeds. As is standard SPI protocol, the LED strip connects to the Pi with two wires, DATA and SCLK (along with GND). Using the Pi's SPI0, an image's column RGB data is sent to the strip according to the data specifications shown in Figure 2.2.

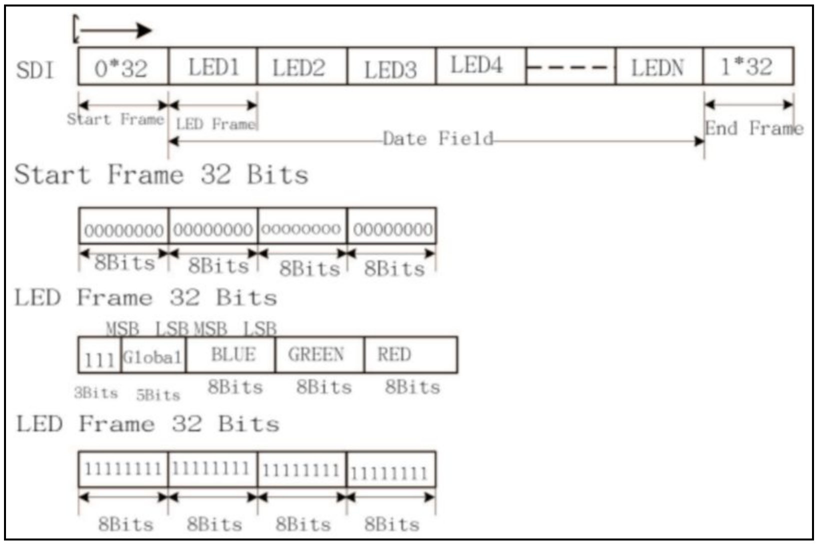

Figure 2.2: SPI Timing Diagram for APA102

3 https://cdn.sparkfun.com/datasheets/Components/LED/APA102C.pdf

A column data packet must be preceded by a 4-byte start frame of 0's, and followed by an end frame of 1's. In reality, the end frame actually requires N/2 bits of 1's, where N is the number of LEDs.[4] Although the global brightness bits are meant to adjust brightness, there have been reported problems with using these bits, so it is instead recommended to deal with brightness levels by scaling the RGB bytes manually. With an 8 MHz SPI clock, a column takes approximately 74 µs to update the LED strip.

The lLED strip is driven with a 5V 7A switching power supply. System logic levels are addressed in the recommendations section.

LED Interface - Software

The Pi's SPI interface was written in C, based on code from EasyPIO.h[5] written by Sarah Lichtman and Joshua Vasquez. The C program[6] handles two functions, parsing the bitmap data for RGB values of the image, and updating the LED strip. Pseudocode for the main function is shown in Figure 2.3. For sampling the FPGA step pin (indication when a step is made), a simple FSM is implemented in code to only catch the positive edge of the FPGA step (see Figure 2.4).

```
void main()
{
   Parse bitmap for RGB data
   Initialize SPI

   while still painting
   {
      Wait for FPGA stepped posedge
      Update column (forwards or backwards)

      Loop for 144 pixels
               {
         Send Start frame
         Scale RGB data for brightness
         Send RGB bytes
         Send End frame
      }
   }
   End with black LED strip
}
```

Figure 2.3: Pseudo-code for `main` function

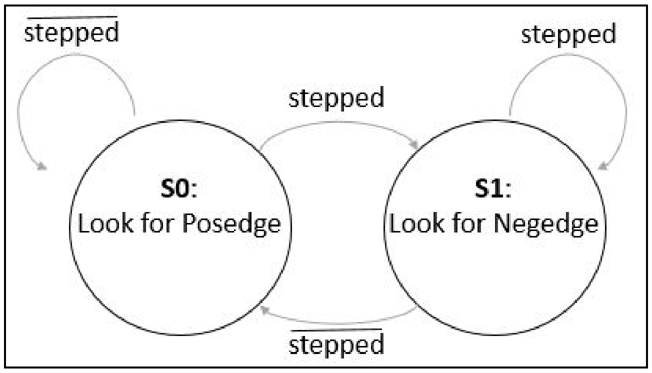

Figure 2.4: FSM diagram for Step detection

III. FPGA

The FPGA handled one main function: reading the raw data from the rotary encoder, counting steps, determining the direction, and sending the information to the Raspberry Pi.

Rotary Encoder

A rotary encoder is a device that converts angular motion into digital Gray code. Gray code is a form of binary code where successive values differ only by one bit. Since rotary encoders have two

[4] https://cpldcpu.com/2014/11/30/understanding-the-apa102-superled/
[5] http://pages.hmc.edu/harris/class/e155/EasyPIO.h
[6] https://github.com/eshimelis/microPainter/blob/master/RPi/microPainter.c

outputs that are 90 degrees out of phase, it can be used to measure both the number of steps and the direction of steps. This project utilizes a Signwise 600 pulse-per-revolution incremental rotary encoder.

This encoder is a four wire device: +5V, GND, Phase A, and Phase B. Each of the output phases are open collector transistors, requiring the use of a pull-up resistor (200KΩ). As the encoder steps, the transistors will turn on and off, thus periodically drawing current through the pull-up resistor, and pulling the node to ground. The circuit diagram is illustrated in Figure 3.1. These outputs were connected to two input pins on the FPGA.

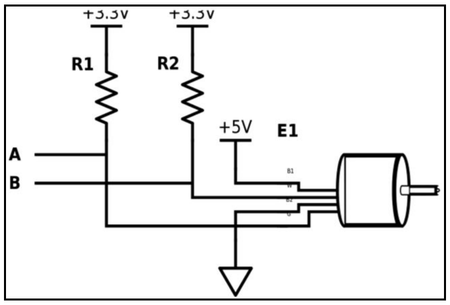

Figure 3.1: Rotary Encoder Circuitry

Rotary Encoder - Digital Hardware

Along with a reset button, the only other physical inputs to the MuddPi Mark IV board are the two output phases of the rotary encoder, called aUnsync and bUnsync. Since these signals are unsynchronized, they are first passed through a synchronizer to produce signals a and b in the FPGA (microPainter.sv[7]). Signals a and b are then delayed by a clock cycle to create two more signals: aDelayed and bDelayed. These delayed signals are used during XOR operations to create constant-width pulses.

The hardware described in the Verilog is shown below in Figure 3.2. Steps are detected by XORing the original and delayed step signals. This results in a step pulse on the rising and falling edge of each signal, effectively quadrupling the resolution of the step encoder from 600 to 2400 pulses per rotation.

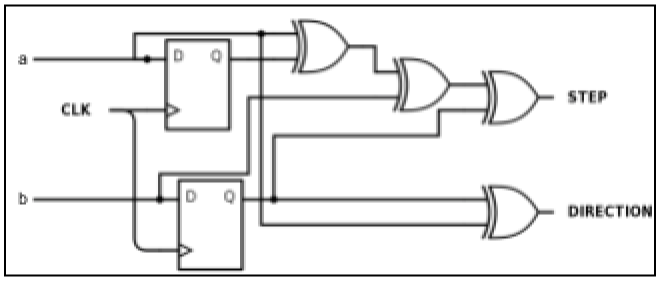

Figure 3.2: Digital circuit to detect steps and directions

Figure 3.3 is a simulation of the digital circuit above. The simulation has three stages. In the first stage, a and b are simulated to be turning clockwise, then a period of pause, then simulated counter clockwise turning. As expected, there is a pulse at the edge of each of the phases, corresponding to a measured step. Also, the count will increment and decrement on each step, depending on the direction of rotation.

[7] https://github.com/eshimelis/microPainter/blob/master/FPGA/microPainter.sv

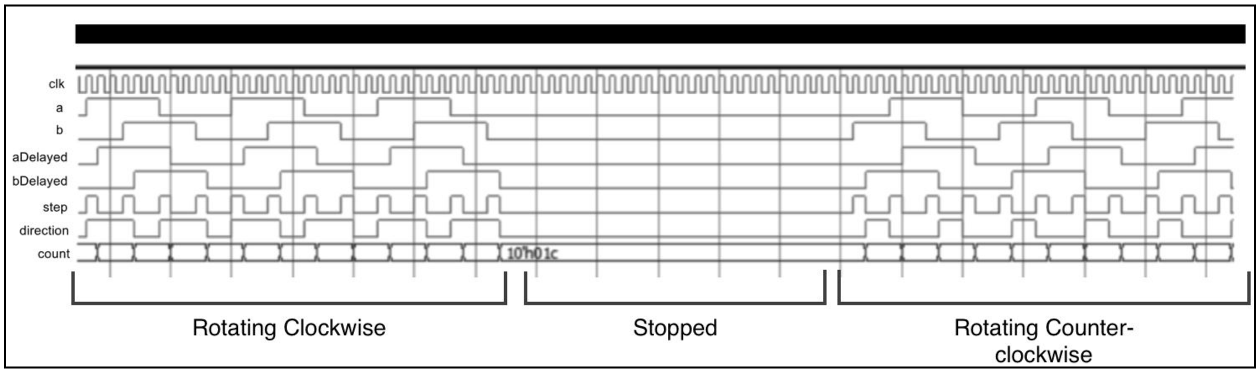

Figure 3.3: Digital circuit simulation demonstrating directional step counting

IV. Additional Hardware

A simple cart was constructed from 80/20 aluminum T-slotted beams to hold the LEDs in place and upright[8]. The rotary encoder was connected to a wheel with a belt to track position. A simple holding plate was laser cut to fasten the rotary encoder. The FPGA, Pi, and power supply were secured to the cart using velcro.

The LED strip was powered with a 5V 8A switching power supply. At max rating (full brightness white) the LEDs draw ~60 mA per LED, resulting in a total of 8.6A for the strip. Since we do not plan on running at full brightness, 8A was sufficient.

V. Results

The microPainter successfully produced two-dimensional light painted images for long-exposure photographs. A sample of these images is shown below (Figures 5.1-4). As desired, the painted images are seamlessly embedded in the true image, leaving little to no indication of the cart rig or the person moving the cart. Since the images are limited to 144 pixels tall, we found that cartoon images work best, however any arbitrary image can be reproduced (Figure 5.4, for example), just at a lower resolution.

One necessary improvement in the project would be making the system logic levels compatible. Specifically the APA102 registers a minimum logic high at $0.7*V_{DD} = 3.5$ V and the Pi outputs 3.3 V on the GPIO pins, thus a logic level shifter should be used to link the two systems.

There is also room for improvement in making the system simpler to use. One additional feature could be adding a motor to move the cart at a relatively constant rate. This would allow more automation of the process, eliminating the need for the person to physically move the cart. This would especially help for solo microPainting, such that a single person could control the camera and simply push a button to start the cart motion.

Another improvement for image quality would be brightness control. Due to the nature of long-exposure photography, when the cart is moving slowly, the sensor will be exposed with more light, resulting in a brighter image. Thus allowing for variable brightness based on cart motion could improve

[8] Thanks to Professor Dodds for providing the material for the cart!

image brightness uniformity. This could also take into account turning off the LEDs when moving backwards for the same reasons.

Although 144 LEDs per meter is the highest market standard for LED strips, adding another LED strip to make a two-meter tall system would produce higher quality images with a tradeoff on system size and maneuverability.

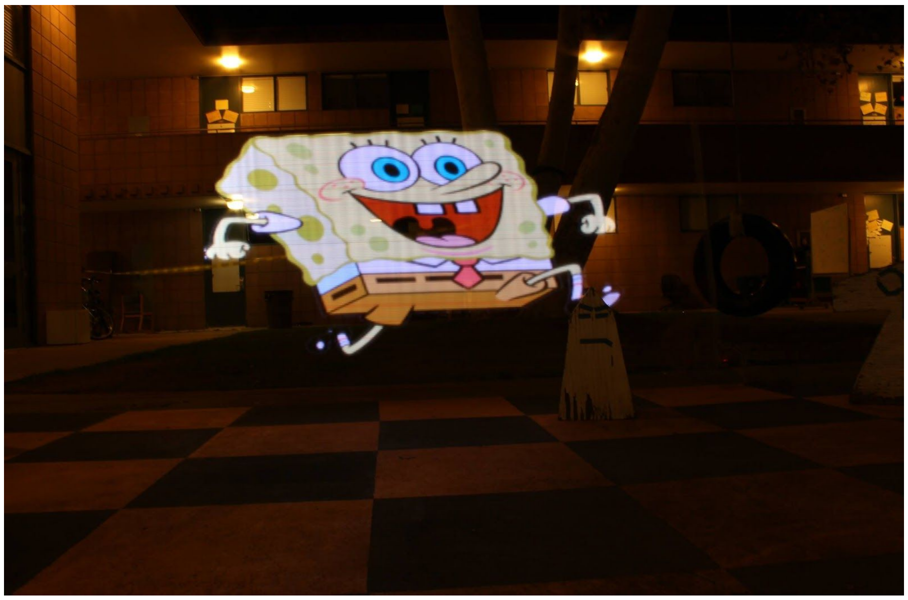

Figure 5.1: Light-painted Spongebob in South Dorm courtyard
(5 second exposure at night)

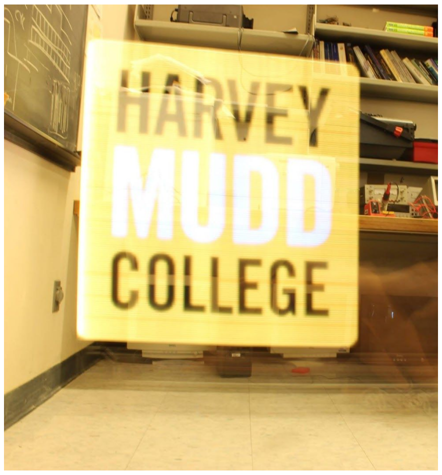

Figure 5.2: Harvey Mudd Logo
(6 second exposure)

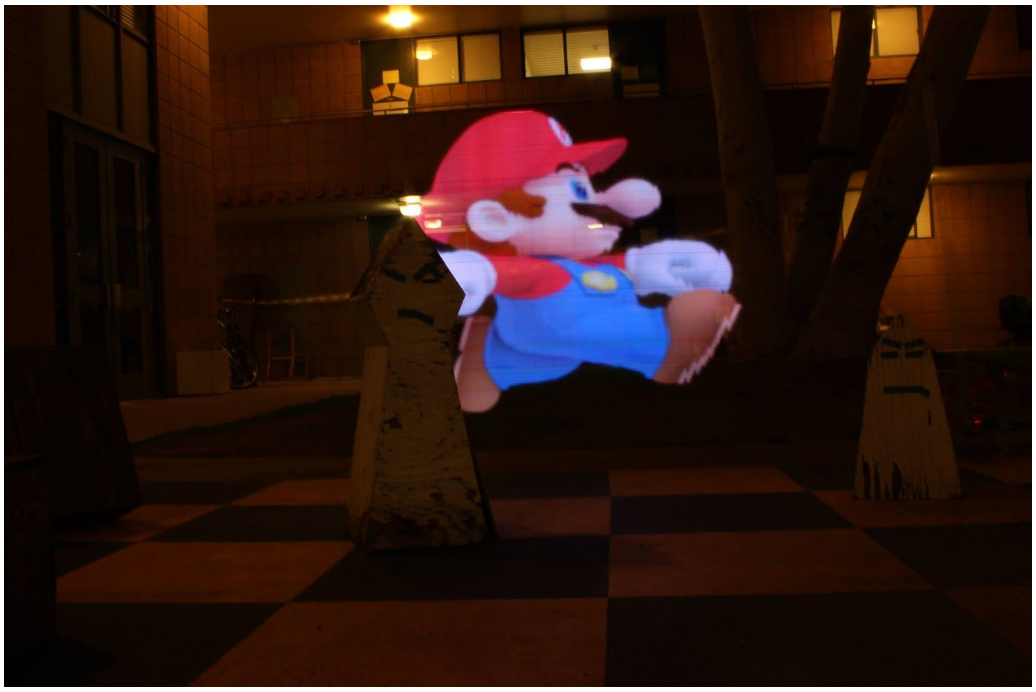

Figure 5.3: Mario in South Dorm courtyard
(5 second exposure at night)

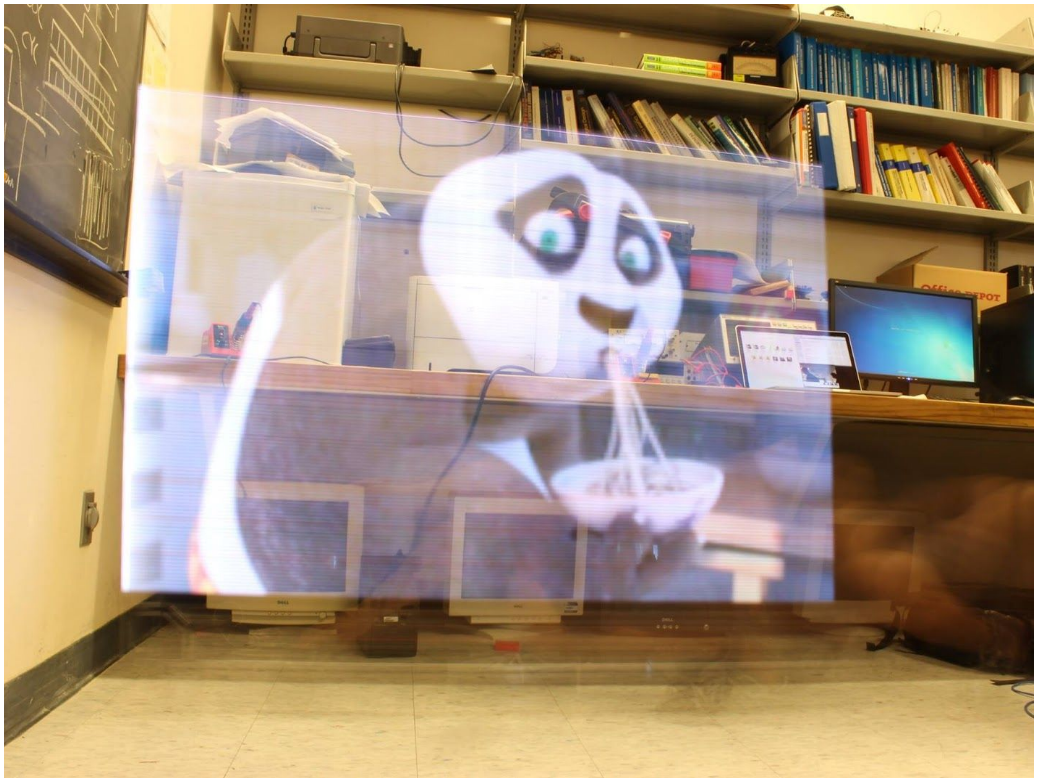

Figure 5.4: Po Eating in the MicroPs Lab
(5 second exposure)

VII. References

[1] PixelStick Home Page, http://www.thepixelstick.com/
[2] APA102C Datasheet, https://cdn.sparkfun.com/datasheets/Components/LED/APA102C.pdf
[3] “Tim’s Blog: Understanding the APA102 ‘Superled’”, https://cpldcpu.com/2014/11/30/understanding-the-apa102-superled/
[4] Sarah Lichtman and Joshua Vasquez “EasyPIO.h”, http://pages.hmc.edu/harris/class/e155/EasyPIO.h
[5] Quadrature Decoders, http://fpga4fun.com/QuadratureDecoder.html

VIII. Parts List

Name	**Part #**	**Source**	**Cost (Shipping)**
APA102, 144 leds/m, 5050 RGB	--	Ebay	27.88 (6.88)
600 Pulse Rotary Encoder	Signwise	Amazon	16.99
DC 5V 7A Switching Power Supply	uxcell	Amazon	11.42
LED Strip Diffuser	U02	Amazon	13.99
Raspberry Pi 3 Model B	--	Personal	--
MuddPi Mark IV	--	Personal	--
Various hardware parts for cart assembly	--	Personal	--
Total			**$70.28 (6.88)**

Appendix

See www.github.com/eshimelis/microPainter for syntax-highlighted code!