6.111 Project Proposal
Luis Torres and Noah Moroze

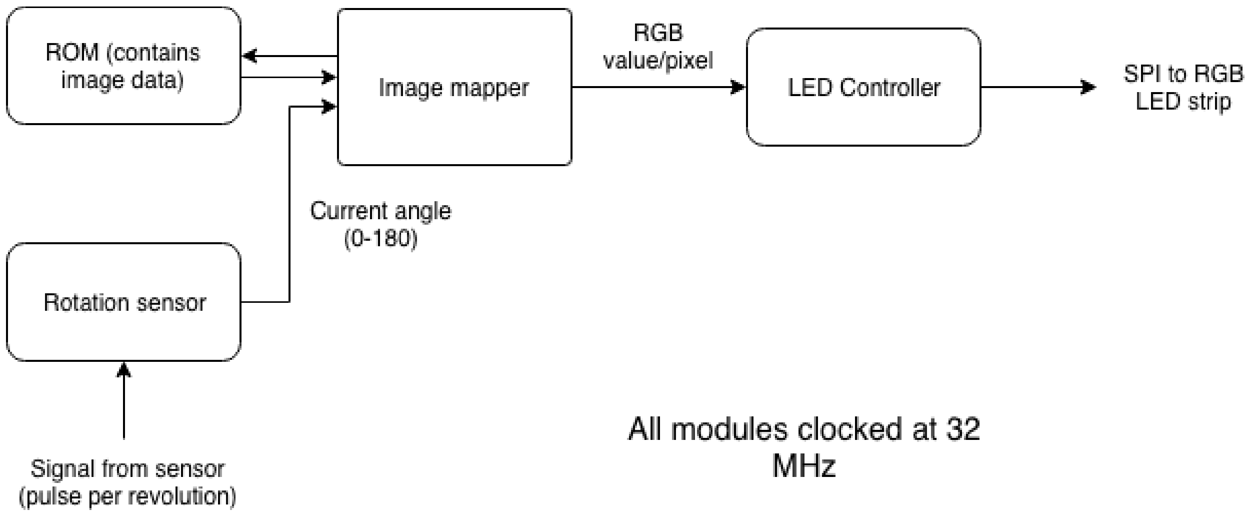
Spherical Persistence of Vision Display

Description

For our final project we are planning to develop an FPGA-controlled persistence of vision display, similar to something like this: https://youtu.be/2hASOre63Nk. The concept behind the display is that by spinning a single strip of lights at high speed and precisely controlling which LEDs are lit at any given position, we can take advantage of the eye's slow response time to create the illusion of a continuous display. One unique feature of our proposal is that we plan to design the geometry of our spinner so that the LED strip is bent in a semicircular arc spinning about one of its endpoints, creating the illusion that our images are being projected around a sphere. As a base goal, we're hoping to have this device be able to display images stored in memory. We have a lot of ideas for how to expand this to various stretch goals, which will be fleshed out in our proposal. Some of our ideas as of now include expanding from one strip to multiple for higher resolution, support for storing/displaying animations, and some sort of live control of what's being displayed.

We plan to use a half meter LED strip with 72 individually addressable RGB LEDs. This will be arranged in an arc, creating a sphere of roughly six inch radius. An FPGA and battery pack will be placed on a platform that sits in the middle of this arc. The entire contraption will be spun by a DC motor with a belt drive. We're planning to use a small form-factor Cmod A7 FPGA due to its light-weight size. We will be able to program its flash memory so it will be configured on boot.

Block Diagram



Modules

All modules have clock and reset.

LED Controller
Inputs: `rgb[71:0][23:0], en`
Outputs: `sck, mosi`

The LED controller takes in a 72-value array of 24-bit RGB values. While `en` is asserted, it continuously latches these values and then outputs them via SPI to the LED strip.

Image Mapper
Inputs: `theta[7:0]`, `rom_data[23:0]`
Outputs: `rgb[71:0][23:0]`, `controller_len`, `rom_addr[13:0]`

The image mapper takes in the current angle as calculated by the rotation sensor module, converts that angle into read addresses that it sends to the ROM, and then gets back color data from the ROM. This will likely require some pipelining to get timing right.

Rotation Sensor
Inputs: `rotation_sensor_input`
Outputs: `theta[7:0]`

The rotation sensor module takes in a pulse from an IR sensor once per rotation. It counts time between pulses to calculate an average speed over time, and then integrates this average speed to calculate angular position (resetting every time the sensor passes 0). This position is outputted and fed into the image mapper.

Animation (Stretch)

Once we get to implementing it, the animation module will sit between the image mapper and our ROM and effectively replace the image we're outputting every certain fixed timestep. If we implement this, we'll have to multiply the size of the ROM by the number of animation frames we need.

ROM
180x72x(24 bits)

We will use the FPGA's ROM to store an image for display.

BOM

Item	**Link/Source**	**Cost**
Frame	Laser cut wood in EDS	n/a
LED strip	https://www.adafruit.com/product/2241	$50
IR break beam sensor	https://www.adafruit.com/product/2168	$6.50
Motor	Already have	n/a
Breadboard FPGA	Get from Joe (https://store.digilentinc.com/cmod-a7-breadboardable-artix-7-fpga-module/)	n/a
Battery pack	https://www.amazon.com/gp/product/B01CU1EC6Y	$20

Potential performance or hardware limitations

Power
One potential limitation is the power capacity of the battery pack. The LED strip draws about 60mA per LED, for a worst-case current draw of $60mA * 72 = 4.32A$. The FPGA's current draw is about 130mA, giving a total current draw of 4.45A. Given that the mobile phone battery pack we're planning to use can supply 2A max, we'll need to drive the LEDs at 13/31 brightness (based on their datasheet, it appears that the LEDs perform current-based brightness control).

Given the battery we've chosen, the device will last around 2.5 hours.