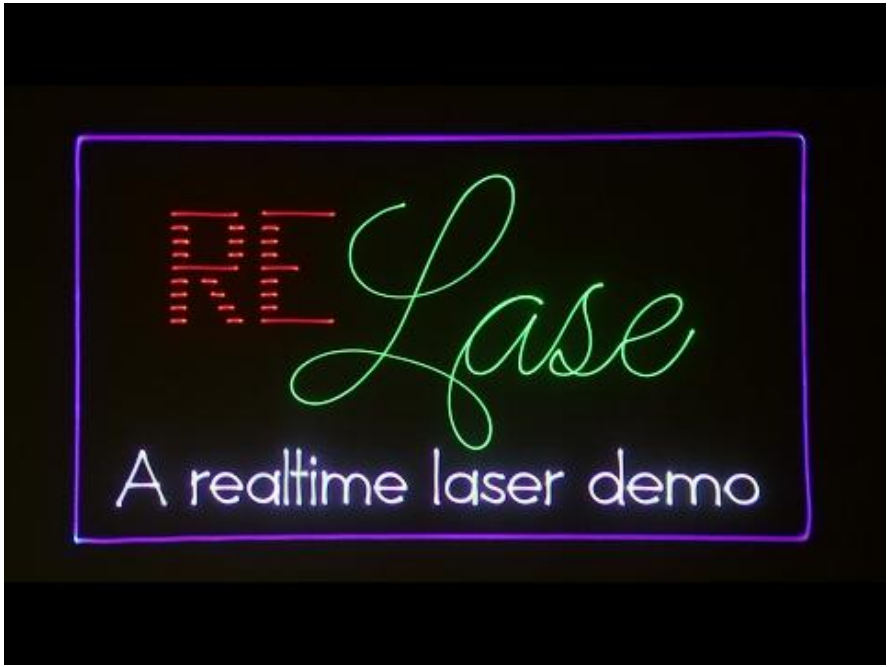
“Hey, wouldn’t it be cool if we hooked a laser up to the internet and made it draw stuff on the wall?”

~ us, struggling to come up with a project about two weeks ago

6.111 Fall 2020 Project Proposal Presentation

jaytlang, *fischerm*

Desired Result



Courtesy C4r0, Greek University of Technology

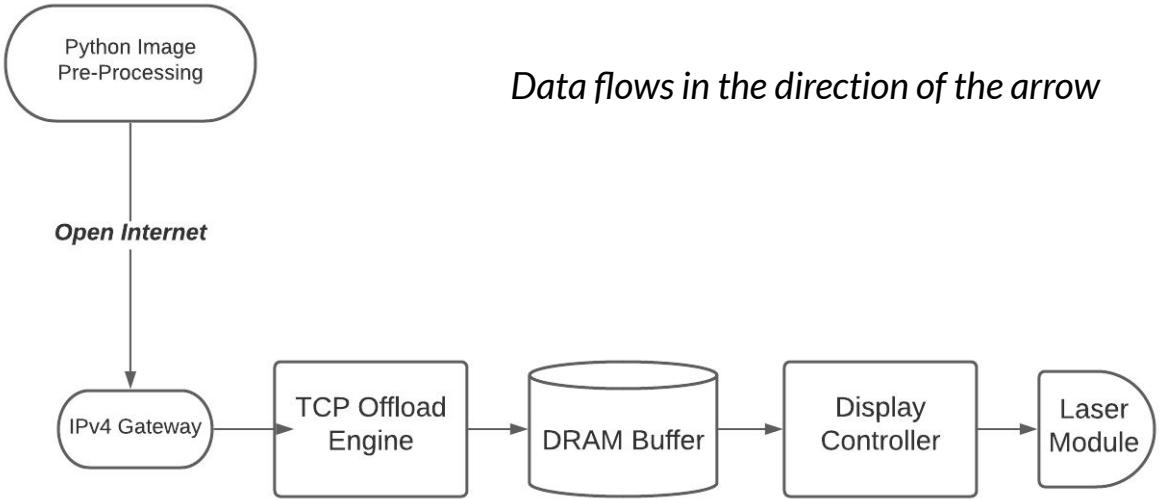
Courtesy *OpenLase Project*, marcan

Project Overview

- Design and fabricate a projector capable of moving a laser on a wall fast enough to create a resolvable, decent-looking image

- Design and implement a parallel-stack TCP Offload Engine capable of 100 Mbps full duplex communication of laser trajectory information over the open internet

The Broad, Full-System Block Diagram



Data flows in the direction of the arrow

Image Processing

Source Image

Post-Canny Filtering

Recolorized Image

*trajectory is also being exported as a .csv file

The Internet Part

The Internet in a Nutshell: OSDI Model

Layer	Function	Example
Application (7)	Services that are used with end user applications	SMTP,
Presentation (6)	Formats the data so that it can be viewed by the user Encrypt and decrypt	JPG, GIF, HTTPS, SSL, TLS
Session (5)	Establishes/ends connections between two hosts	NetBIOS, PPTP
Transport (4)	Responsible for the transport protocol and error handling	TCP, UDP
Network (3)	Reads the IP address form the packet.	Routers, Layer 3 Switches
Data Link (2)	Reads the MAC address from the data packet	Switches
Physical (1)	Send data on to the physical wire.	Hubs, NICS, Cable

How do you send trajectory information over the air?

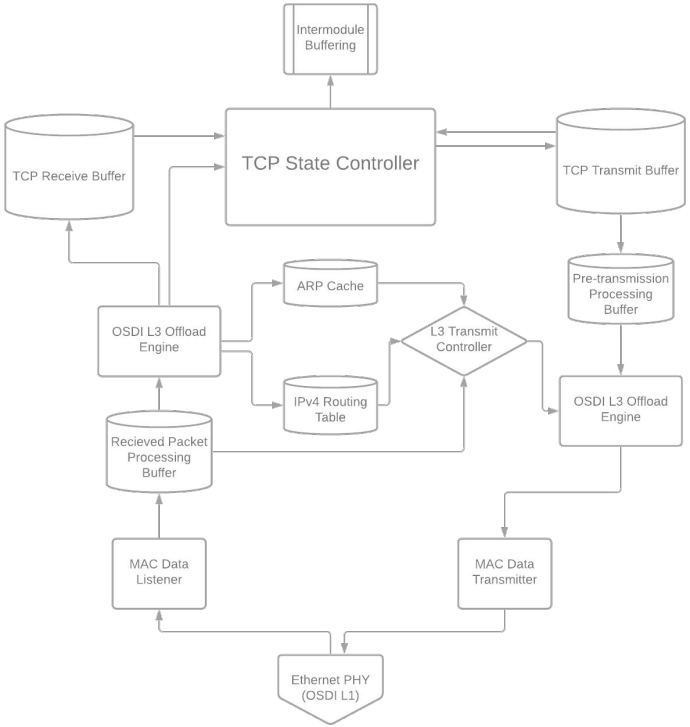
Networking Objective

Implement *de-encapsulation* and *encapsulation* for each of these different layers, so that we're able to *offload* normally software-defined transit layers into hardware.

OSDI Networking Layers

- Level 1: Ethernet Physical Layer (PHY)
- Level 2: Media Access Controller (MAC Module)
  - Communication with the PHY + addressing
- Level 3: Internet Protocol Version 4 (IPv4)
    - IPv4 Address Resolution Protocol (ARP)
    - Internet Control Message Protocol (ICMPv4)
  - Software defined networks
- Level 4: Transmission Control Protocol (TCP)
  - Reliable transmission of data streams

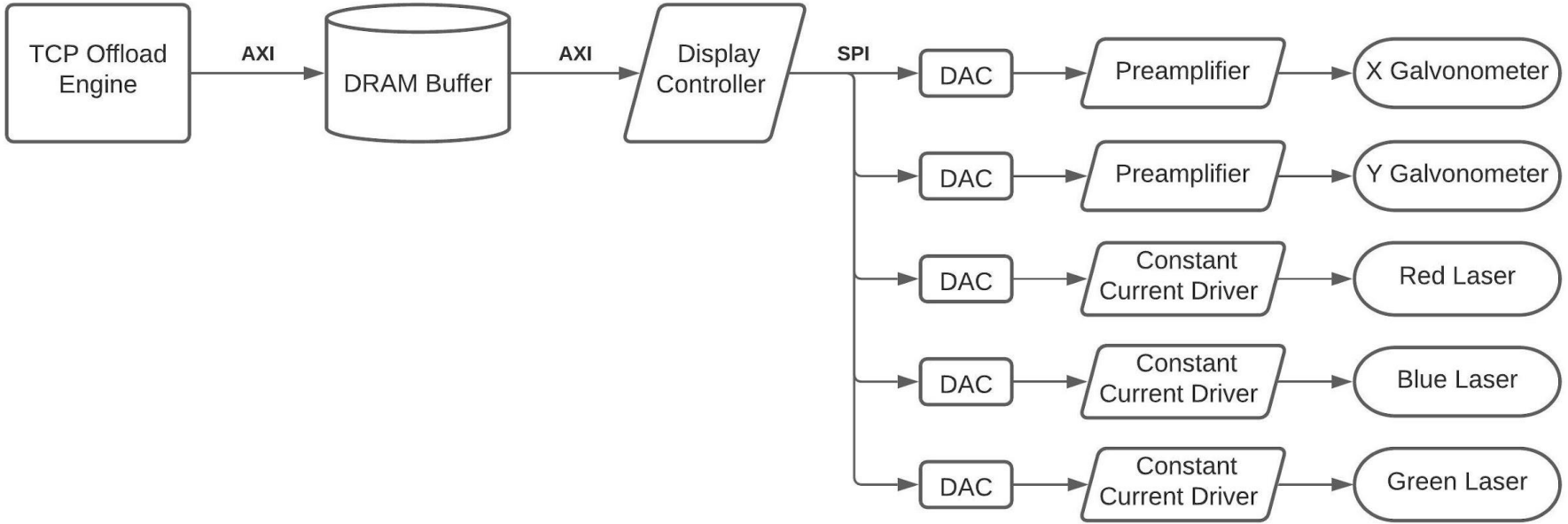
Getting it into hardware: TCP Offload Engine



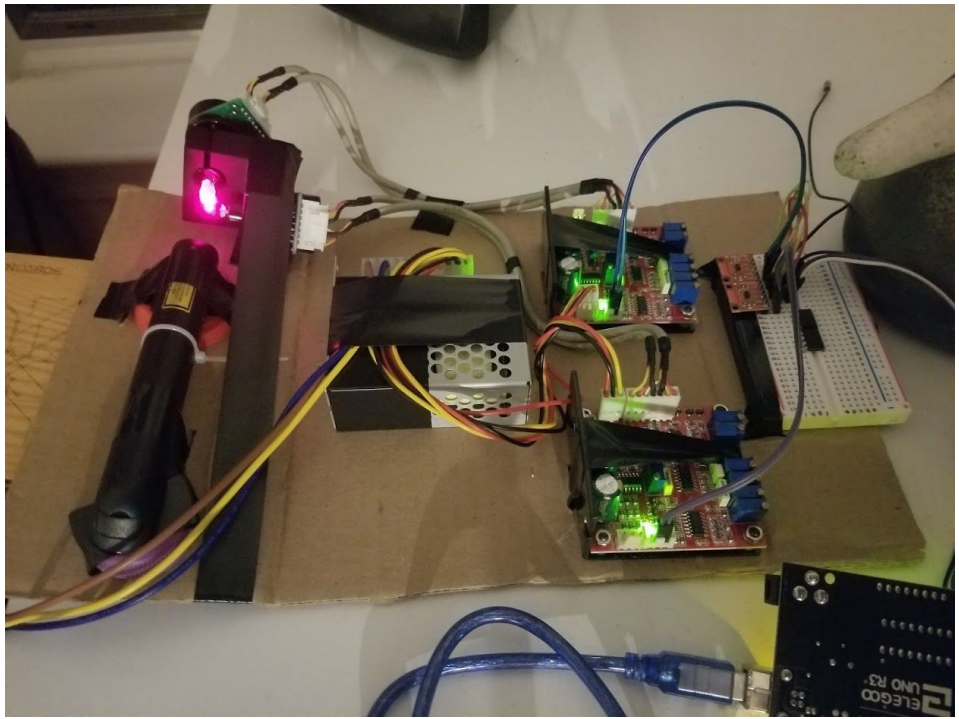
The fine print / key points:

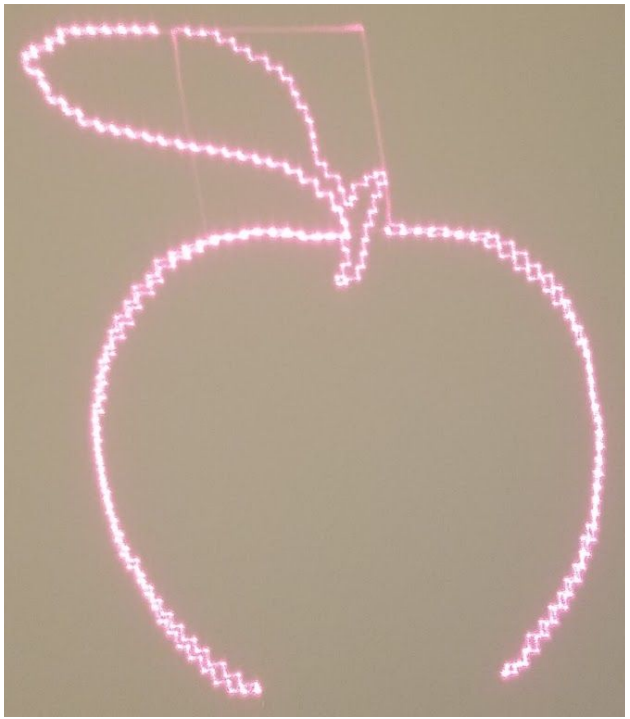
- BRAM buffering is utilized to perform operations on packets *in place*, such as determining packet length and transitioning to/from network byte order as necessary.
  - Both L3 offload engines (RX and TX) utilize a single-packet buffer to help do this
  - L3 offload engine mashes together L3 networking protocols to fast track processing/checksumming this buffer
- An ARP *cache* (MAC <-> IP mapping store) is stored in BRAM along with a primitive IPv4 routing table
- Should ARP or ICMP need to send a packet, they may overwrite the current outgoing TCP packet via the L3 transmit controller
- Note: these lines between modules usually represent AXI lines, where lines in and out of BRAM buffers represent reads and writes.

Display Hardware



Display Hardware





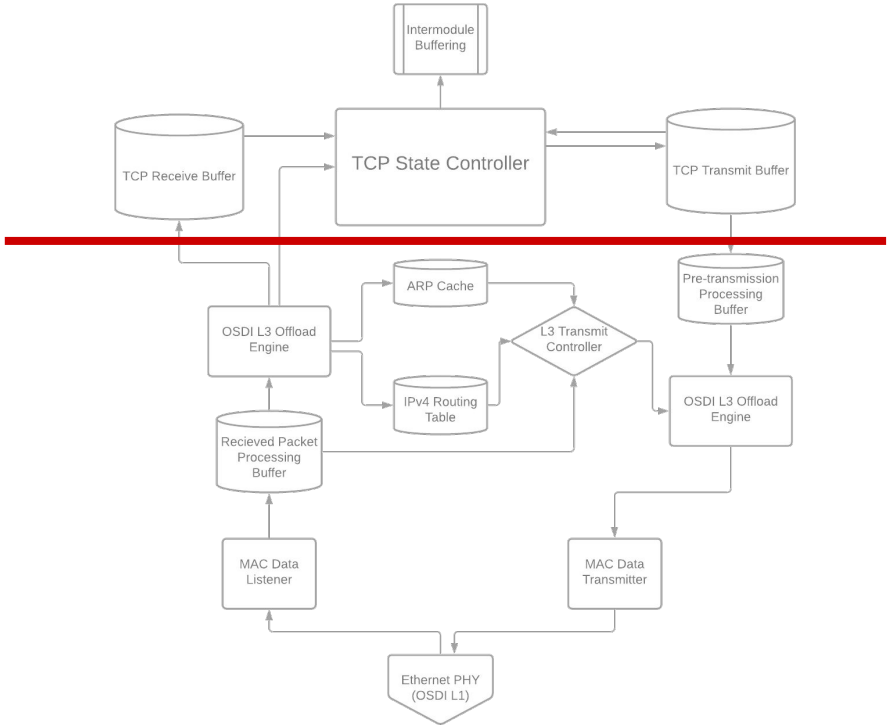
Timeline

Date	*Network (Jay)*	*Hardware (Fischer)*
26 October	Prototype PHY Layer Operational	Galvonometers Operational
2 November	Network Stack Design Finished	Dual-Channel SPI Output Functional
9 November	Packet Processing Buffer Done	DRAM Framebuffer Logic Complete
16 November	OSDI L3 + Open-Internet ICMP	EHS Approval Complete
23 November	TCP State Machine Implemented	RGB Laser Output Functional
30 November	Optional TCP Features In Place	Final Integration
7 December	Report Writing	Report Writing

Q&A!

We’ve got two minutes.

TCP Offload Engine: More Fine Print



Display Hardware: More Fine Print