Amanda Ke
Allan Ko
6.111 Project Proposal
November 1, 2016

LASERNET

1. Introduction

Free-space optical (FSO) communication systems transmit wireless data using visible-frequency electromagnetic waves propagating in free space. FSO systems using lasers as the communication medium are of particular interest, as they theoretically enable point-to-point long-range communication links with data rates comparable to (if not better than) those achievable by radio broadcast or fiber optic cables, without the infrastructural cost of wired connections. Laser-based communication systems therefore have a wide range of theoretical applications, such as providing network connectivity in emergency situations or improving the data rates of deep space transmissions.

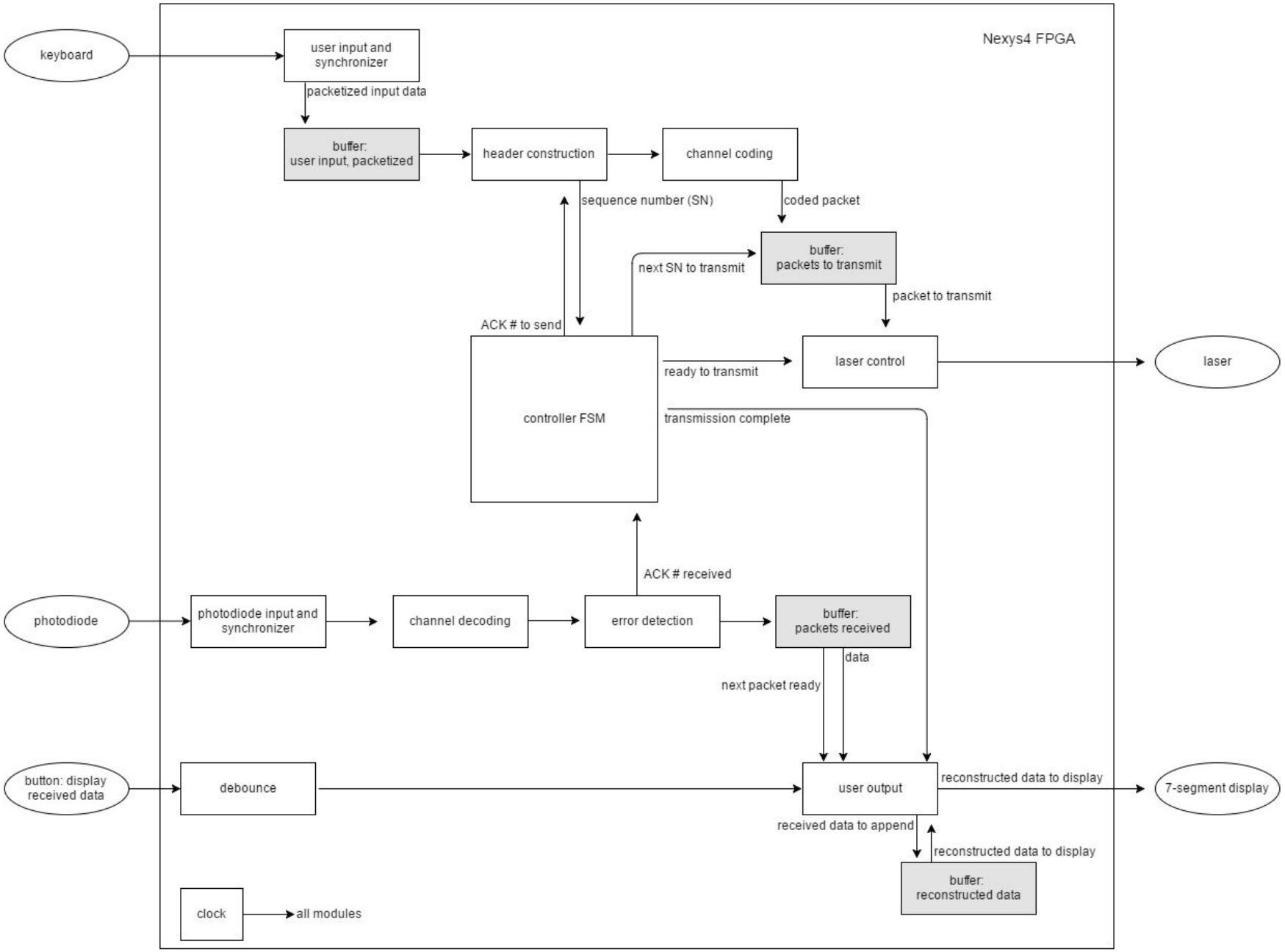
Inspired by laser communications research underway at NASA and Facebook's Connectivity Lab, LASERNET is an FSO communication system implemented with FPGAs and off-the-shelf lasers. In this proposal, we will describe the basic logic architecture for transmitting messages between two nodes, which communicate via laser using an Internet-like Transmission Control Protocol (TCP) connection. We also sketch an approach for extending this platform to multiple nodes and establishing some basic routing protocols, forming a simple "laser internet."

1.1. System overview

LASERNET will be implemented on the Digilent Nexys4 platform, which is powered by the Xilinx Artix-7 FPGA. Each node will consist of one Nexys4 board, a laser transmitter, and a photodiode receiver. The FPGA software architecture consists broadly of a controller module which operates a finite state machine (FSM) implementing a TCP-like connection protocol. The FSM controls eight primary submodules: four submodules for transmitting data and four submodules for receiving data. These are illustrated in the block diagram on the next page, and described in more detail in Section 2.

1.2. External components

The only external components necessary will be a laser for data transmission, a photodiode/amplifier for data reception, and a USB keyboard for user input.


Nexys4 FPGA
keyboard
user input and synchronizer
packetized input data
buffer: user input, packetized
header construction
channel coding
sequence number (SN)
coded packet
buffer: packets to transmit
next SN to transmit
packet to transmit
ACK # to send
controller FSM
ready to transmit
laser control
laser
transmission complete
ACK # received
photodiode
photodiode input and synchronizer
channel decoding
error detection
buffer: packets received
data
next packet ready
button: display received data
debounce
user output
reconstructed data to display
7-segment display
received data to append
reconstructed data to display
buffer: reconstructed data
clock
all modules

2. Software Architecture

Because all nodes in our simple network transmit and receive data, every node runs on the same software architecture. We describe this architecture below.

The overarching control module operates a finite state machine (FSM) that establishes TCP-like connections, tracks which packets have been acknowledged, retransmits packets when necessary, processes data for user input/output, and closes the TCP connection when the transmission is complete. The full TCP specification is described by RFC 793, which sketches the state diagram of a TCP connection on page 23.

The control module directs the submodules detailed in the following sections, which we have organized into “transmission” submodules (user input, header construction, channel coding, and laser control) and “reception” submodules (photodiode input, channel decoding, error checking, and user output).

2.1. Transmission submodules (Allan)

User input

The user input module reads in new data from user input, buffers the data as necessary, segments it into equally-sized packets, and forwards the next available packet to the FSM when the FSM signals ready. The user will enter data using a USB keyboard. This is supported by the Nexys4 board via the Auxiliary Function microcontroller, which takes USB keyboard input and emulates the PS/2 protocol for the FPGA.

Header construction

The header construction module reads the next available packet and constructs a packet header according to the TCP header format. The key segments of the TCP header are the source and destination port, the sequence number, the acknowledgement number, which are given by the controlling FSM. The module also computes and inserts the checksum, which by TCP standards is the bitwise complement of the one’s complement sum of all 16-bit “words” in the packet and header.

Channel coding

The channel coding module reads in the next packet to be transmitted (as indicated by the controller FSM) and encodes it with a forward error correcting code in order to make the connection more robust to transmission errors. The specific code to be implemented is yet to be determined, as we don’t know yet how noisy the laser transmission scheme will be. However, a convolutional code with an interleaver would be a standard coding scheme to use. Another option would be a linear block code, such as a Hamming code.

Laser control

The laser control module reads in the next message (as directed by the controller FSM) and transmits it, bit-by-bit, by flashing the laser diode at the photodiode of the receiving node.

2.2. Reception submodules (Amanda)

Photodiode input

The photodiode input module reads, synchronizes, and buffers incoming data. Incoming data is read as a stream of bits from the photodiode as it receives laser flashes from the transmitting node.

Channel decoding

The channel decoding module reads the incoming data from the photodiode input module and decodes whatever error-correcting code was implemented. For instance, if a convolutional code was used in transmission, this module might implement a Viterbi decoder; or, if a Hamming code was used, this module would implement a syndrome decoder.

Error detection

The error detection module performs the TCP checksum and verifies that the new packet is error-free. If so, it reports to the controlling FSM to acknowledge the received packet and forwards the packet to the user output module. If the packet is found to contain errors, the packet is discarded.

User output

The user output module buffers all the incoming packets verified by the error detection module. As sequential packets are received, the user output module reassembles the original data in the correct order. Once transmission is complete and the TCP connection is closed, the user output module can be prompted by a button press to display the received text message as a scrolling marquee on the Nexys4 7-segment display.

3. Development Plan

3.1. Schedule

Our proposed week-by-week development schedule is detailed in the following table.

	Allan	**Amanda**
Week of Oct 31	Refine software specification: memory requirements, inputs, outputs, timing Begin drafting state diagram for controlling FSM	Order analog components Complete analog circuits for (1) laser control and (2) data reception via photodiode
Week of Nov 7	Complete user input and laser control submodules	Complete photodiode input and user output submodules
	Integrate above submodules with basic FSM, demonstrate basic data transmission without headers, channel coding, packet acknowledgment, or error correction	
Week of Nov 14	Complete header construction and channel coding submodules	Complete channel decoding and error detection submodules
	Verify submodules with testbenches in simulation Project checkoff checklist meeting	
Week of Nov 21	Integrate channel coding/decoding and TCP transmission protocols (headers, checksum, and packet acknowledgements) into the controlling FSM Test and verify successful connection, message transmission, and connection termination. This fulfills baseline requirements.	
Week of Nov 28	Buffer week Work on stretch goals if baseline requirements complete	
Week of Dec 5	Buffer week Work on stretch goals if baseline requirements complete Begin drafting final project report	
Week of Dec 12	12/12 - Final project checkoff 12/13 - Project demos and videotaping 12/14 - Turn in final project report	

3.2. Stretch goals and potential upgrades

Because of the modular nature of communications architectures, there are many ways to expand on our baseline two-node system, if we have time. Once the TCP data link architecture is complete, we might transmit larger packets by reading and writing data using an SD card on each node. To handle larger packets, we might implement compression, perhaps using the Lempel-Ziv-Welch compression scheme. It should also be straightforward to implement an encryption layer between the user input and header construction steps.

Other interesting ways to make LASERNET more Internet-like would be to implement a User Datagram Protocol (UDP) mode, perhaps building up to a Real-time Transport Protocol (RTP) implementation. We could use this to demonstrate audio streaming or Voice over IP (VoIP) by tweaking the user input and output modules.

Of course, the most interesting upgrade of all would be to add more nodes to the network, constructing an ad-hoc local area network connected entirely via optical lasers and communicating with a simple implementation of Internet Protocol (IP). This would require some extra FPGAs (one for each node) as well as extra lasers and photodiodes (one of each for each edge).