A Very Social Raspberry Pi Network

Our project consists of a network of Raspberry Pi's connected via a wire that sends messages to each other. Our project resembles the basics of Ethernet in that each Pi sends packets of data to its connected Pi's. Each of these packets holds 4 bits representing the sender's ID, 4 bits representing the destination's ID (if ID's value is 8, the message sends to all of the Pi's in the network - this is the group message option), an 8 bit representation of the payload size, and a char array that can hold 128 characters. Messages can either be sent to one other designated Pi or the whole group of Pis. Additionally, we have built a user interface to resemble an Apple-2-esque version of iMessage with defined locations on the screen for messages being sent and received. To send a message, the user must type the destination ID (an integer between 0 and 8 inclusive), a colon, space, and then the message. An example of what the user may type is: "1: Hello, world!".

We reconfigured Julie's ring buffer code to accept our message packets instead of integers. One tricky part of changing the ring buffer was that we needed to pass the packets in by reference instead of by value. When passing in the packets by value, some of the bits were lost when placing them into the ring buffer, as the size of the packet was too large and prompted a stack overflow. We fixed this issue by using a pointer to the packet as a parameter for enqueueing. Another tricky (but ultimately pretty cool discovery) came out of testing our adapted ring buffer. During these tests, we printed the contents of the message as it was sent from one Pi to another. As we did this, we received multiple EOT messages but did not understand why. After searching through the assert.c, bootloader.c, and pi.c files, we realized our message, as it was being built, occasionally contained a '0x04' which, when printed, tells the program to exit. While this was a frustrating bug, it was a super cool realization, and we learned a lot about the other files we were reading at the same time!

There was a lot of collaboration throughout the project, and we all worked on the initial brainstorming for the requirements and execution plan for the system. Allison was responsible for building the user interface, constructing the connection between the protocol file and the user interface. Kylee also helped forge these connections, built the protocol, and adjusted the ring buffer file. Justin assisted in building the protocol file and did a lot of the debugging of the ring

buffer. As a group, we built the hardware that allows multiple Pi's to communicate and optimized the code.