Week 3:

Date: 09/09/2022

Hours: 14

Description of design efforts:

System Planning:

This week I spent some time planning out the system diagram a bit more thoroughly. I created a block diagram, highlighting each of our major components – the peripheral controller, main controller, Linux laptop (substituted for Raspberry Pi due to driver issues), camera, projector, and current peripherals, as well as their connections. The main block diagram is shown below in Figure 1.

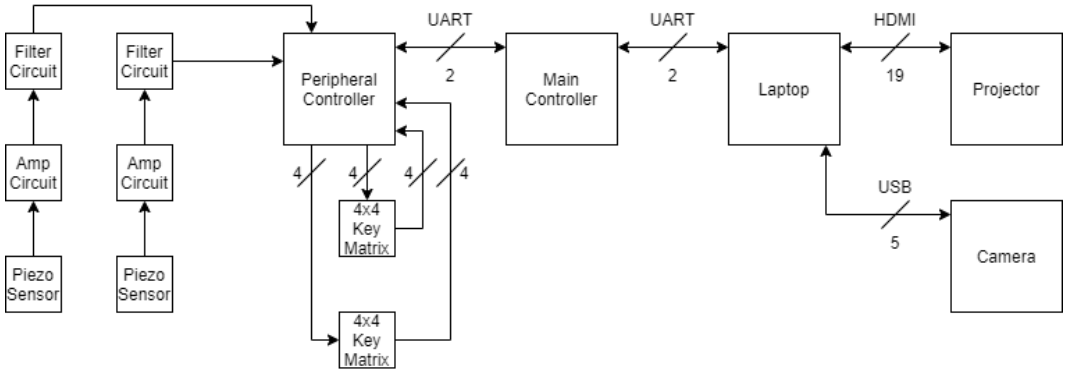

Figure 1. Full System Diagram

Here, we see that the peripheral controller communicates with the peripherals and shares that information to the main controller via UART. Any updates that require graphical changes will be then shared to the laptop, which will then cause graphical changes within its environment that will be seen by the projector.

Besides the main diagram, I delved into each sub-part a little more specifically and described (the current knowledge) of more specifically what the components will do and how they will communicate with each other. The majority of my work will be done on the peripheral controller, shown in Figure 2.

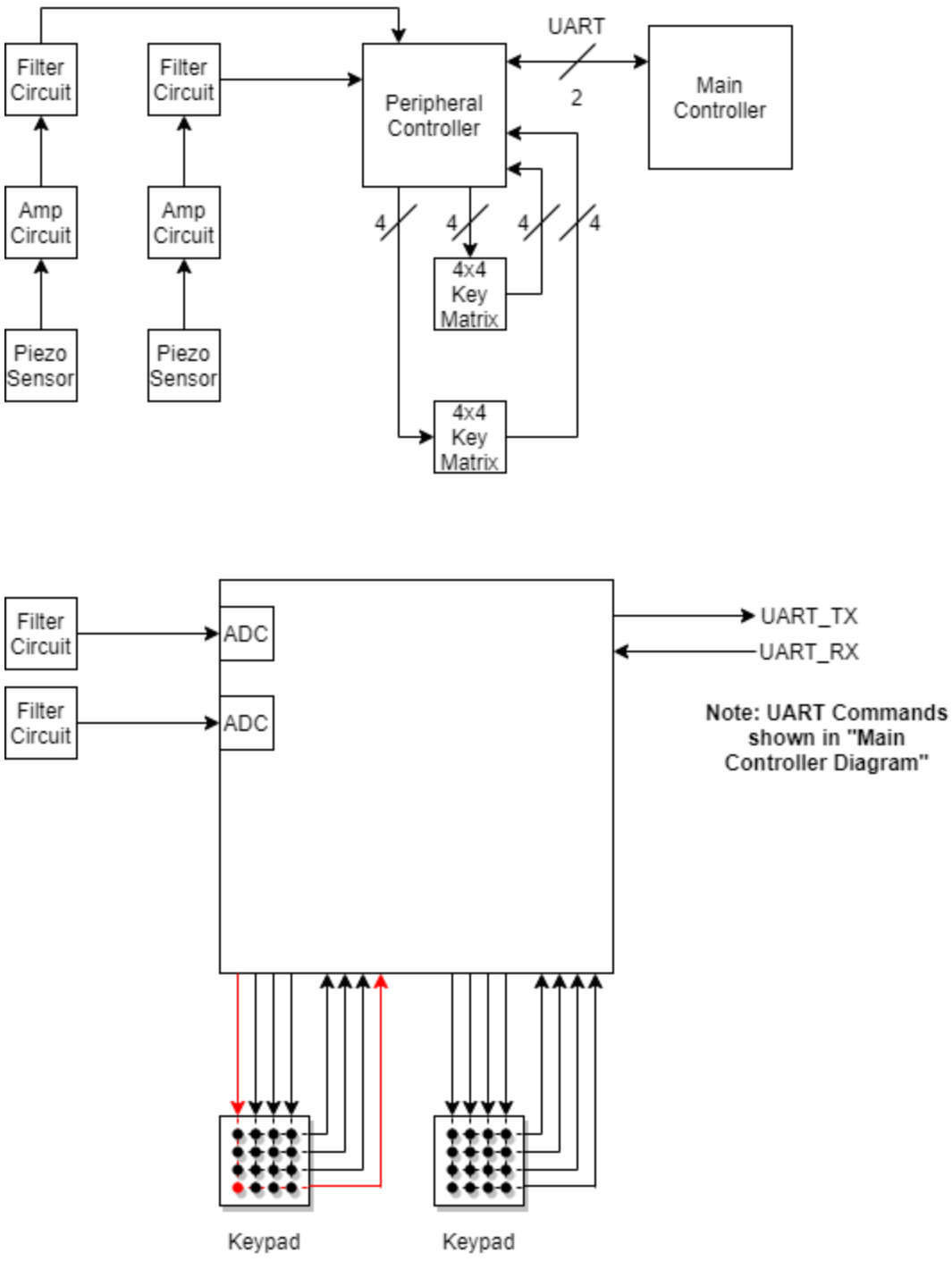

Figure 2. Peripheral Controller Diagram

Here, we see that this controller handles the peripherals – 2 piezoelectric sensors connected to 2 ADC channels and 2 keypad matrices connected to 16 GPIO pins (8 each – 4 input and 4 output). The ADC conversion logic is incomplete – the next steps are detailed later in the “Next Week” tab.

Below, in Figure 3, we see the main controller diagram. For now, this document is mostly relevant for describing the expected UART communication system between the subsystems.

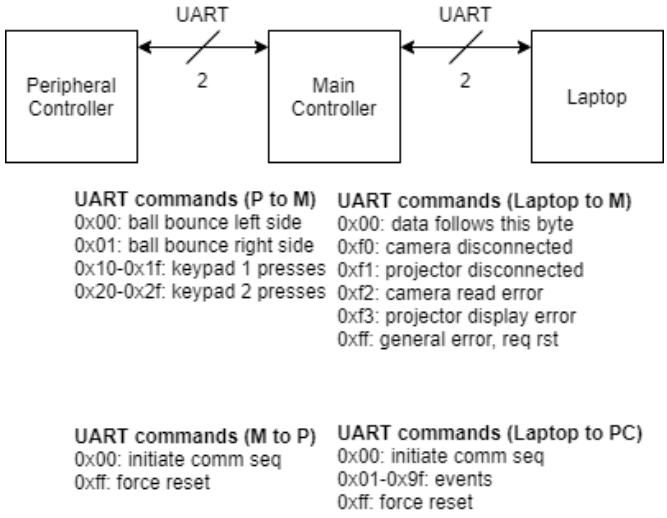

Figure 3. Main Controller Diagram

Finally, I created a diagram for the laptop's functionality, seen in Figure 4. Hardware-wise, this is quite simple – we simply interface between the camera, projector, and game logic controller. However, unseen in this document is extensive logic used to detect the ball and display graphics.

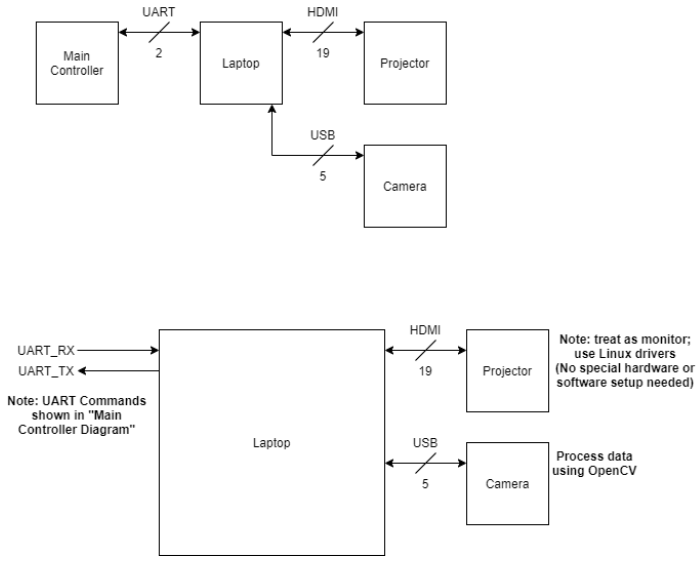

Figure 4. Laptop Diagram

Microcontroller Programming:

To handle reading data from the piezoelectric sensors, I added code to the peripheral controller to enable the on-chip ADC. With that working, I connected a revised amplifier circuit to the piezoelectric sensor and stored the constantly outputted data in an array to check the output. With this setup, we could detect bounces on a threshold basis, although, not as accurately as I would like. The output voltage from the amplifier circuit is not strong enough to show impulses from bounces substantially different from no significant input. Figure 5 displays an oscilloscope output of a single bounce. The impulse peaks at approximately 300 mV, which is not a significant change from 0.

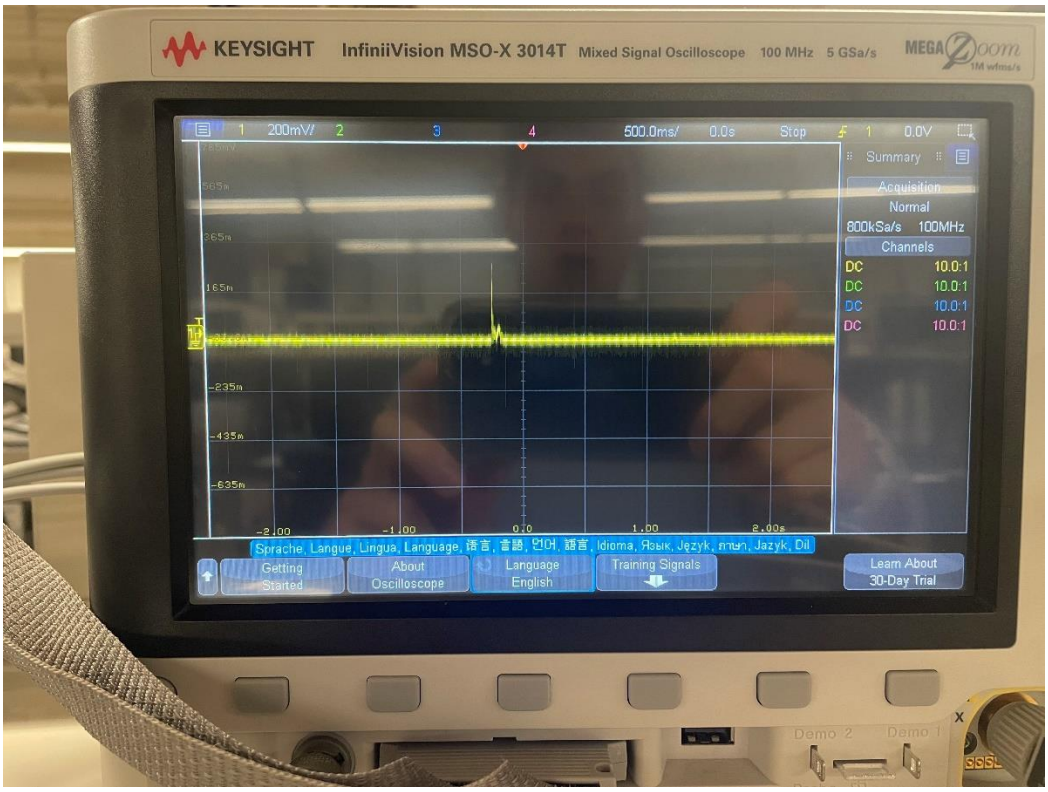

Figure 5. Oscilloscope Bounce Impulse

Debugging:

I ran into an issue with my UART protocol, where it appeared that one of my microcontrollers was consistently spewing garbage data, instead of waiting for keypresses to send anything. I reverted my code to a version I knew worked, and changed the code I borrowed from 362 that used puts() to send information, and directly handle the UART communication myself, by waiting for the TX register to be empty, then filling it, to send a character. I am still uncertain why this issue surfaced, as it initially had not caused problems.

Next Week:

We are currently in the process of deciding if we want to keep the second microcontroller. The first is necessary for peripheral controlling (but also for a course requirement); the second's logic can simply be handled by the laptop. With a 2-controller setup, the 2nd controller simply tells the laptop what to send to the projector, so this will remove the need for the extra communication system.

Either way, we will need a microcontroller to communicate via UART to the laptop. As we are no longer able to connect to GPIO pins to handle this through the Pi, we will need to interface UART with a new communication system that the laptop accepts (without modifications). As such, Jack and I will spend time next week verifying that we can use the FTDI UART to USB adapter to handle this 2-way conversion.

Currently, we can read 1 ADC conversion consistently. However, we will need to read at least 2 conversions. To do so, I have a few ideas:

1. Ping-pong (fitting!) buffer filtering, where we read 1 sample from ADC channel 0, then channel 1, and repeat. Note that in this case, we will miss every other sample.
2. Superimposing the 2 (or more) signals and reading them as a combined value. We can detect that a bounce occurred, then, if there is processing time, determine which side it occurred on. If not, that logic can be handled by the camera.
3. Use a separate ADC IC to handle conversions and read multiple ADC outputs simultaneously.

Of course, first, we need to determine how clean and strong of an impulse we can retrieve from a ball bounce on the piezoelectric sensor. My priority is to alter the filter and amplifier circuits enough that we can correctly determine from the ADC that a ball bounce has occurred, then worry about handling 2 signals. My intention was to have a "threshold" that we must cross that indicates a bounce, but instead, it may be easier to check for a rising or falling edge, indicating a large change in input.