Week 5:

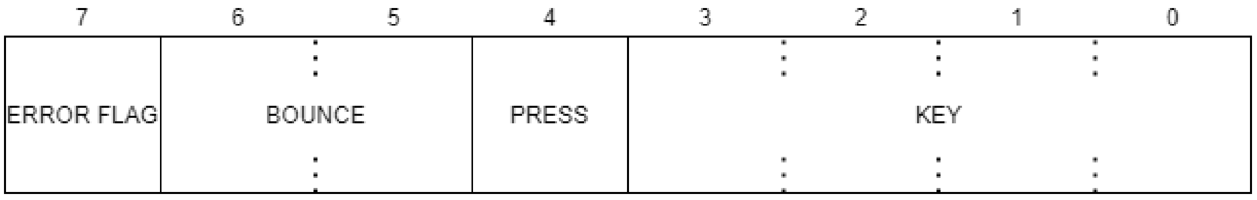
Date: 09/23/2022

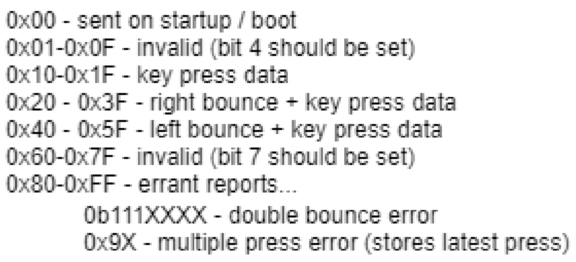
Hours: 14

Description of design efforts:

Systems Engineering:

I finalized the UART packet structure, by reworking where each bit is set. The design is laid out in Figure 1, below. The key data has been moved to bits 0-3 and is only valid if bit 4 is set. Bits 5 and 6 share which side of the table a bounce has been detected on, and errant behavior occurs if both are set. As such, bit 7 will be set, to report an error. Bit 7 also will be set if we press multiple buttons before the packet is polled by the laptop, clearing it. In both error cases, the laptop will interpret them to mean “increase the polling speed.”





0x00 - sent on startup / boot
0x01-0x0F - invalid (bit 4 should be set)
0x10-0x1F - key press data
0x20 - 0x3F - right bounce + key press data
0x40 - 0x5F - left bounce + key press data
0x60-0x7F - invalid (bit 7 should be set)
0x80-0xFF - errant reports...
    0b111XXXX - double bounce error
    0x9X - multiple press error (stores latest press)

Figure 1. UART Packet Structure

Hardware:

Hardware this week felt like a setback – when switching to test our ping-pong table instead of the sheet of plywood I have been working with, I found that our output from the piezoelectric sensors was greatly reduced. I believed it was due to a different resonant frequency of the new material, and likely due to it being surrounded by a stiff, metal frame that dampens this oscillation.

I set up 2 piezoelectric sensors on our table and connected them both through two separate piezo processing circuits I created last week, reproduced for your convenience in Figure 2. On the plywood sheet, this reported a typical “bounce” at around 1 V, but on the table, each was negligible.

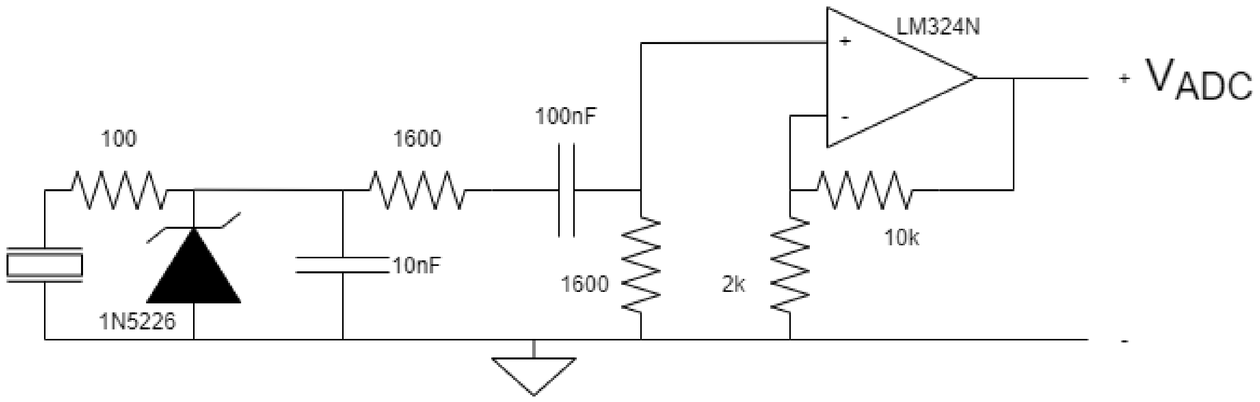


Figure 2. Piezoelectric Sensor Processing Circuit

I began to tweak the above circuit until I was able to produce a consistent output, like what would be seen on the plywood, for the above circuit. The current solution follows Figure 3, below. It appears to work well, however, there is a consideration seen in “Microcontroller Programming” below that may be indicative of an issue. Here, the diode circuit has been removed, as the operational amplifier is biased between 0 and 3.3V, so any output signal will be safe for the microcontroller to handle. Note that this circuit is not yet complete – more testing needs to be done to ensure a proper output is read.

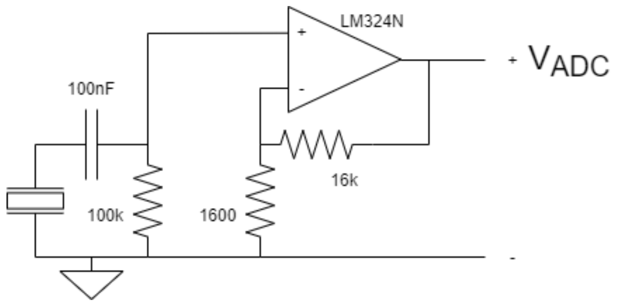


Figure 3. Revised Piezoelectric Circuit

Microcontroller Programming:

I set up packet handling on the microcontroller to act as we expect, according to the structure designed this week, again, in Figure 1. I also configured the packet to only be sent through the UART channel upon receiving the "Request Data" packet, from the same channel. The packet is sent and cleared, to be updated in the next cycle. Note that we handle "clearing" from the laptop, and plan to poll faster than every 30 ms. Thus, it is possible, but unlikely, that we can find 2 button presses or 2 bounces in this time frame. This would be an example of an error, and in this circumstance, bit 7 can be read by the laptop, in agreeance with 5 and 6, or 4, to determine which error occurred.

With the above changes, I added a "debug mode" to the microcontroller code, as a method to override the need to calibrate the ADC. Here, there is a hardcoded value set as the threshold (determined from my calibration sequence) that is enacted, as well as some debugging information shown – the development board's red LED will light up on a bounce on 1 side of the table, and the blue LED will light up on the other side. This was used to detect an error. Unfortunately, at the time of this report being written, our table is unusable, as it is undergoing a re-painting process. I have a video of the following operation, but it is not helpful to the reader, so I will explain the error below.

The blue LED flashes every time the ball bounces on the right side of the table, correctly inferring a bounce. However, on the left side of the table, both the red and blue LEDs flash, predicting bounces on both sides of the table. I swapped the sensors to their counterparts' filter systems (after ensuring they are identical, beyond electrical tolerances) and saw the same errant behavior, but this time, the LEDs have been switched. I believe one of two things is happening here:

1) One of the piezoelectric sensors is not functioning properly. Of course, it "works", but not as well as the other. This could mean that the "full functioning" piezo will detect a bounce on the wrong side, and that we prefer the broken one, or vice versa. There is not enough information to determine this yet. If our "full functioning" piezo works as we want, good! We can just replace the broken one. If the broken one works as we want, we cannot "break" the other one until its signals match the one, we like – this problem is a little more complex. We would need to rework the filtering system / calibration values.
2) The two sides of the table are built differently enough that their resonant frequencies are far enough from each other that 1 is being filtered out while the other is not. This simply would require our filters to be redesigned.

Next Week:

Next week, I intend to pursue completion of the following tasks:

1. Add error reporting logic in packet configuration logic. That is, if we're filling out information in a non-empty packet, we have produced an error.
2. Resolve the piezoelectric sensor problem. This is a broad task and may consume most of my time next week.
3. Design a robust calibration sequence to handle bounce detection with the following goals:
    - Detect 90% of bounces (losing only "quiet" ones)
    - Falsely detect less than 5% of bounces on the other side of the table
4. Test the limits of the ADC sample rate to determine:
    1. How slow can we sample and still detect every bounce?
    2. How many microphones can we handle reading?

With these tasks complete, I will begin to assist with other work, relating to either PCB design or software development of ball tracking.