ECE 350 Final Project Report: Ergs do Float

Matt Harris and Ryan Bergamini

December 8, 2018

Duke Community Standard

1 Overview of Project Design and Specifications

The overall purpose of this project was to apply concepts of digital systems to create quantitative metrics that reflect the technical form of a rower in order to provide them with feedback on how to improve. In order to achieve this, three metrics were developed, measured, and displayed, as well as a dynamically updating graph of the handle height on the erg. This project used two main arrays of sensors to develop measurements of form and inputs to our FPGA. Once the data was input into the FPGA, metrics including: ratio, rush, and consistency were calculated, either in hardware modules or in the processor using MIPS instructions, in order to produce the data displayed and used to quantify the form of a rower's stroke. Finally, a clean VGA metric screen was created on the FPGA, allowing a rower to see the quality of their stroke based on the metric values, as well as the handle height of the erg in real time. Although there were issues with the final form of this project, we were proud of our ability to develop and implement a creative project idea that is both practical and novel.

2 Input and Output

2.1 Input

One of the main aspects of our project was the collection of characteristic data from two separate arrays of sensors. These sensors measure the handle height and seat position on the erg. These two sets of data are taken as input into our top level module from the GPIO connector on the FPGA. Both arrays of sensors use a basic circuit diagram shown in Figure 4.

2.1.1 Handle Height

The first array of sensors, shown in Figure 2, allowed us to measure the handle height at any given point in time. This sensor was created by first designing a 3D printed shell, capable of housing 16 pairs of transmitter and receiver infrared LEDs. When the chain attached to the handle moves through a transmitter and receiver pair, a high digital signal is written to the corresponding wire that is taken as input into the FPGA. In this way, the handle height sensor array produces a 16-bit input, where one position will read high at any given point in time. This signal is used in order to generate the graph of handle height shown in our display in real time.

2.1.2 Seat Position

The second array of sensors, shown in Figure 3, was used to measure the seat position on the erg. Five pairs of transmitter and receiver infrared LEDs were mounted to the shaft of the erg, slightly below where the seat rolls when a rower takes a stroke. Each of these pairs produces either a digital high signal, when the pair is uncovered, or a digital low signal, when the pair is covered. These five separate signals were taken as inputs into our FPGA through the GPIO connector and used in our processor in order to calculate the Rush and Ratio metrics. Our original idea was to use the movement of one of these signals from low to high in order to determine the position of the seat and the place in the current stroke. For example, when the first sensor became uncovered (low to high), we would be able to determine a new drive had started and use this in our calculations. However, we overlooked the fact that as a rower moves through the drive, the LED's never become uncovered. Even though the seat moves back as the stroke progresses, in theory uncovering the LED, the rowers legs will move within range of the transmitter-receiver pair, continuing to cause the sensor to read as covered. This led to the failure of our project to accurately read the position of the rower in the stroke. However, if more time was spent processing the signals, combinations of the five signals moving from low to high and high to low could be used to generate the same input signals (startDrive, startRecovery, etc) used in our code for the rush and ratio calculations. (For example, all of the signals being covered would correspond to the start of the recovery, or the rowers return to the top of the stroke).

2.2 Output

All of the sensory inputs to our FPGA were used to calculate metrics of stroke form and in turn, directly affected the information we wanted to output. All of our output was displayed on a VGA screen and included graphical output and metric output.

2.2.1 Graphical Output

The graphical output of our project, displayed on the VGA, was entirely created with handle height data taken from the first array of sensors. The incoming handle height data was converted from a 16 bit input, where one bit was set to high, to a 32 bit value representing the relative handle height, ranging from 1 to 16. In the VGA controller, the x-coordinate of the current address being assigned an index was first compared against the x-coordinate range of our graphical data. Next, an acceptable y-coordinate range was calculated and determined from the current input handle height value. If the current address being assigned had x and y coordinates that fell into these bounds, then the pixel's index was overwritten from the normal index corresponding to the background color at that point, to the index corresponding to the color red in our index mif file. In this way, when the first array of sensors determined the handle height at a given moment, the corresponding line of pixels in the graph was displayed in red, producing a dynamically updating graph showing the current handle height at any given moment.

2.2.2 Metric Outputs

There were three metrics displayed on our VGA: ratio, rush, and consistency. The inputs from the second sensor array were used to calculate these metrics. Once a value had been calculated for ratio, rush, and consistency, a corresponding dataReady signal was set to high. This allowed us to update our display synchronously with the termination of stroke. When a new value was ready to be output, the x and y coordinates of the current address being assigned an index in the VGA controller were compared against the x and y coordinates of the position where the metrics were to be displayed. If the address fell into one of these coordinate bounds, the index corresponding to pixel color was overwritten from the corresponding background color index to a new color index, in a separate mif file, based on the value of the metric. For example, if the current value of ratio was calculated to be 2.0 and the ratioDataReady signal was high, each pixel's coordinates would be first checked against the coordinates of the ratio ones place box, then the index of the pixel would be remapped to a corresponding index in the mif file associated with displaying a 2. In this way, at the end of each stroke, the ratio ones place and tenths place, rush ones place and tenths place, and consistency ones, tenths, and hundredths place values could be updated synchronously.

3 Changes Made to Processor

Multiple changes were made to the processor to account for the the many forms of data input, as well as the calculation of consistency. All of the connections we made to inputs were made through the register file. In order to interface the inputs with our MIPS code, we hardwired inputs such as our startDrive input, raw handle height input, final reset button, and handleDataReady input to registers 26, 29, 30, and 28 respectively. These inputs served as the main control loops that were crucial in controlling the data flow of our program.

In addition to the hard-wiring of our inputs to register in the register file, we also attached multiplication and division modules to specified registers in order to add mult div functionality. For example, because of the complications of implementing multdiv with stalls, we set aside registers 12 and 13 as the inputs to a multiplication (where the product would be store in register 14). Additionally we set aside registers 15 and 16 as the inputs to a division (where the quotient would be store in register 17).

4 Challenges

4.1 Sensing and Input

One of the main challenges and shortcomings of our final project design was collecting reliable data from our sensors in order to accurately calculate and display our metrics. For the handle height sensing array, the transmitter and receiver LED pairs were held in a enclosed, 3D printed housing. This meant that the LED's were relatively unaffected by ambient light and the only thing capable of triggering the handle height input was the presence of the chain between the LED pairs. However, in the case of the seat position sensing array, the diode pairs were simply installed on bread boards, facing upwards towards the seat. This meant that the LED pairs in this array were much less reliable and directly affected by the ambient light in the room. In order to try and correct for this effect, we worked to calibrate the incoming seat position data in Arduino code, however this calibration seemed to change constantly depending on the time of day (changes in light coming into the lab through the windows) and the position of the erg in the room (changes in ambient light due to ceiling lights). Additionally, when the full system was built and the sensing array was mounted to the erg, we realized we had simply forgot to account for the legs of the rower. At the end of a drive, when the seat was in the furthest back position, we had written our code to expect only the furthest back sensor would be triggered, however due to the extension of the rowers legs over the other sensors, all sensors were triggered in this case. We did not realize our mistake until it was very near the deadline for the project. We attempted to correct our code and change the logic of control signals corresponding to positions in the stroke used in the calculation of ratio, rush, and consistency, however we were unable to fully implement this solution before our demo.

4.2 Display

The main challenge we faced in producing our display was accounting for the several different clocking signals that our system was dependent on. These included: the sampling rate of our sensors, the clocking speed of the processor, and the clocking speed of the VGA display. We were able to address this problem in order to display our metrics effectively at the end of each stroke, however creating a dynamically updating graph was more difficult. Originally, our goal was to display a dynamic graph of handle height that would show the changing value of throughout the progress of the rower's stroke. This meant the red line indicating the current value of handle height would have to be plotted in real time, from left to right, as samples were collected from the handle height sensing array. In addition, the previous values of handle height in the current stroke would have to be stored in order to have the trailing end of the red line remain in position until the end of the current stroke. We were unable to solve this problem and so for our final project we decided to create a slightly simpler graph that simply showed the handle height value at the current moment, across the full width of the graph. This graphical display was able to update in real time based upon the incoming data from the sensor array, however it does not allow a rower to see clearly on one graph how their handle height position changes over the course of their stroke.

5 Testing

Our project involved three main components: sensory data collection, metric calculation, and VGA display. This informed our decision on how to test our components as they were completed in order to determine how the pieces would fit together in the full system.

5.1 Testing of Sensor Arrays

The first task we began completing at the beginning of this project was the handle height sensing array. The sensing array for the handle height was based upon an earlier design Ryan had built for a project for the CoLab. Improvements were made to this design, including the 3D printed housing. This task was completed early and so we were able to do significant testing with the array attached to the erg in order to ensure the collected data was correct. The second sensing array was completed and constructed with less time remaining before our project demo. This means we had less of an opportunity to test our design thus leading to its shortcomings. We tested this design on the lab bench and tweaked it until we were reading correct values based on the presence of an object over the LED-pairs, however it was too late by the time we tested the array mounted to the erg to fix and debug the flaws in our design that became apparent once the system was fully constructed.

5.2 Testing of Metric Calculation

The calculations for ratio and rush were completed with hardware modules and so these were easy to test with waveforms. A sample set of input data was inputted into the modules through waveform simulation and the outputs were verified as the correctly calculated values.

5.3 Testing of Display

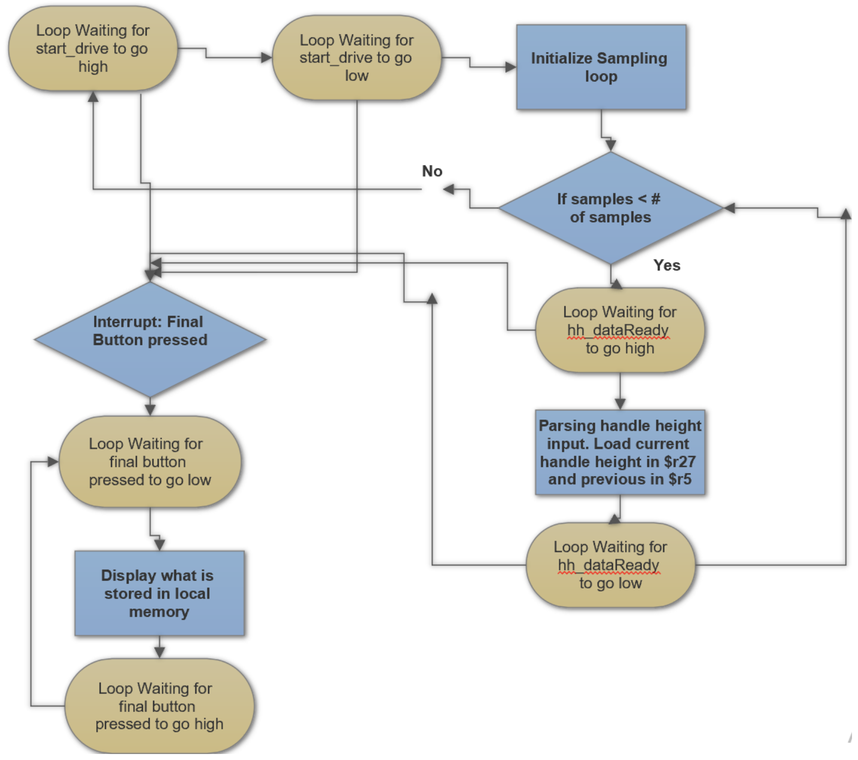
In order to test the display, sample values were first input as the metrics we wished to display. This allowed us to ensure that when a data value was input into the VGA controller, that the screen would properly represent the static image of the correct number based on the various index mif files. Next, arbitrarily changing values of data and dataReady signals were input into the VGA controller in order to test how the display would react to changing input data values.

6 Flow of Assembly Code

In the figure below you can see the rough outline of our MIPS Code. Our MIPS code was structure in such a way that it could map the sampling output from the handle height sensor into memory. By using a handle height data ready signal from the handle height sensor (*hh_dataReady*), we were able to design a baseline code that would be able to capture all of the data points of a stroke regardless of what the sampling rate was set to with the handle height input.

At its code, the MIPS Code was designed to take a specified amount samples from a handle height sensor whenever the *star_drive* input was triggered (which denoted the beginning of a new stroke). The code is also interwoven with checks for if *final_button* was pressed. If pressed, the program flow would escape its basic sampling loop and transfer to a loop where the user can scroll through stored values in memory. It was in this part of the program, the summary screen – where overall metrics such as average ratio, consistency, and rush – would have been calculated.

Figure 1: Flow Diagram of the MIPS Code


Loop Waiting for start_drive to go high
Loop Waiting for start_drive to go low
Initialize Sampling loop
No
If samples < # of samples
Yes
Interrupt: Final Button pressed
Loop Waiting for hh_dataReady to go high
Parsing handle height input. Load current handle height in $r27 and previous in $r5
Loop Waiting for final button pressed to go low
Loop Waiting for hh_dataReady to go low
Display what is stored in local memory
Loop Waiting for final button pressed to go high

7 Appendix

Figure 2: Sensor Array 1: Handle Height

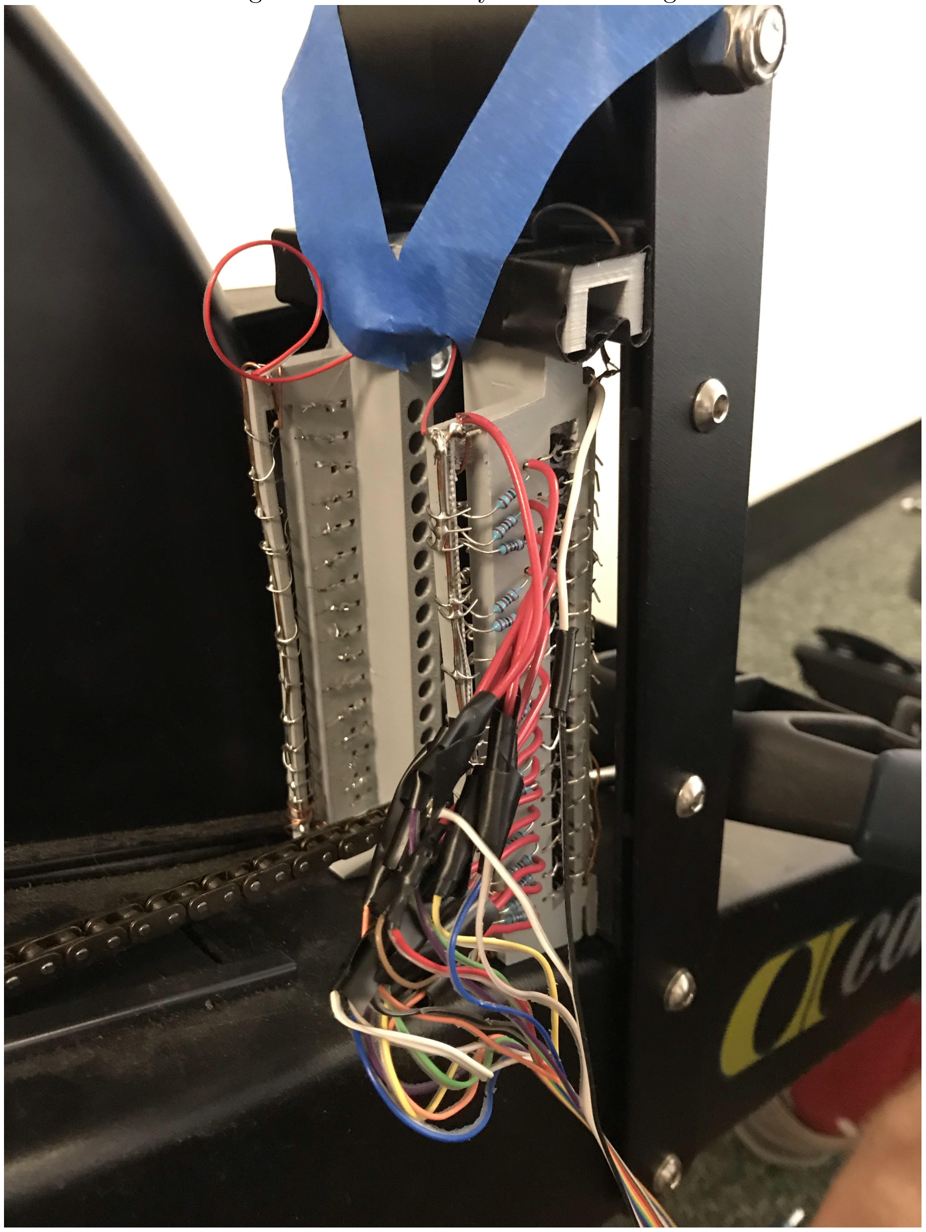


Figure 3: Sensor Array 2: Seat Position

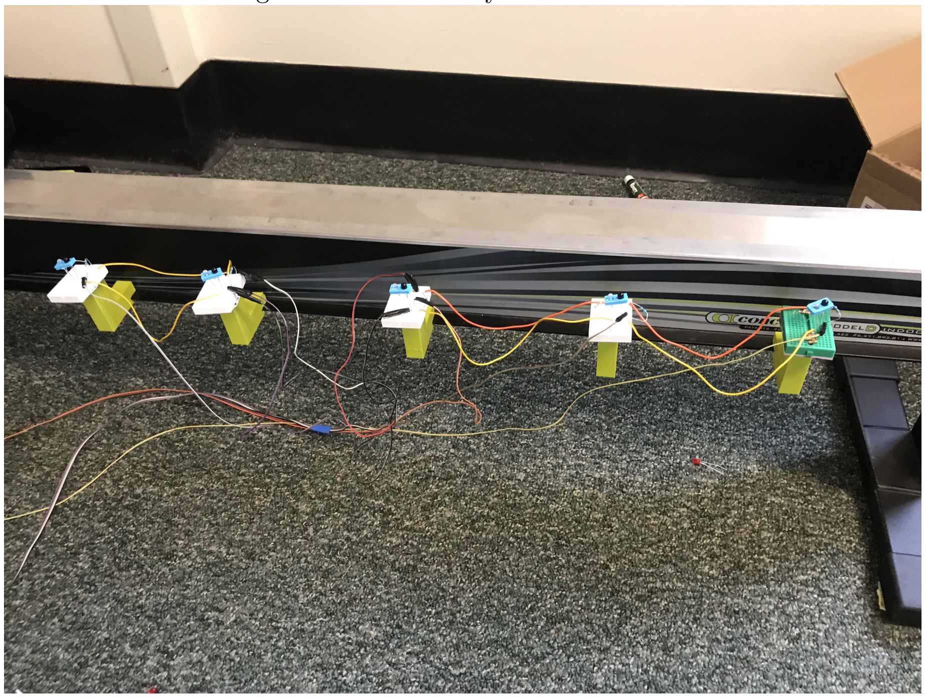
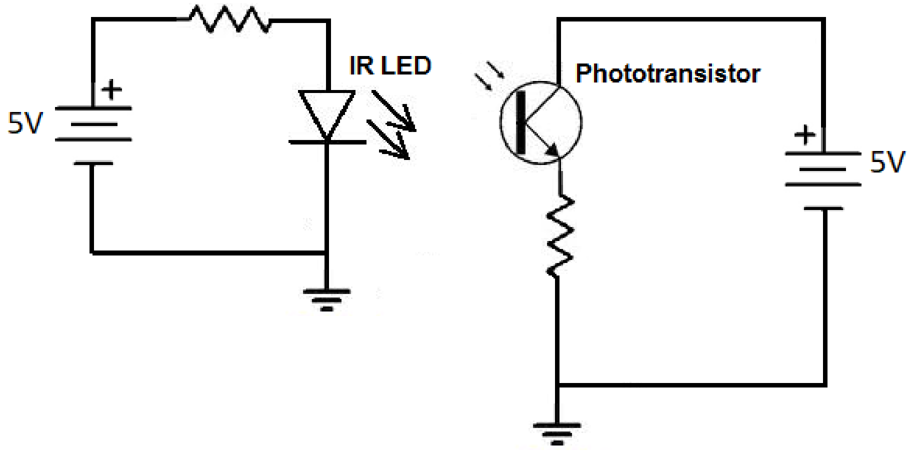


Figure 4: IR-LED Sensing Circuit (adapted: learningaboutelectronics.com)


IR LED
5V
Phototransistor
5V