Project – 2

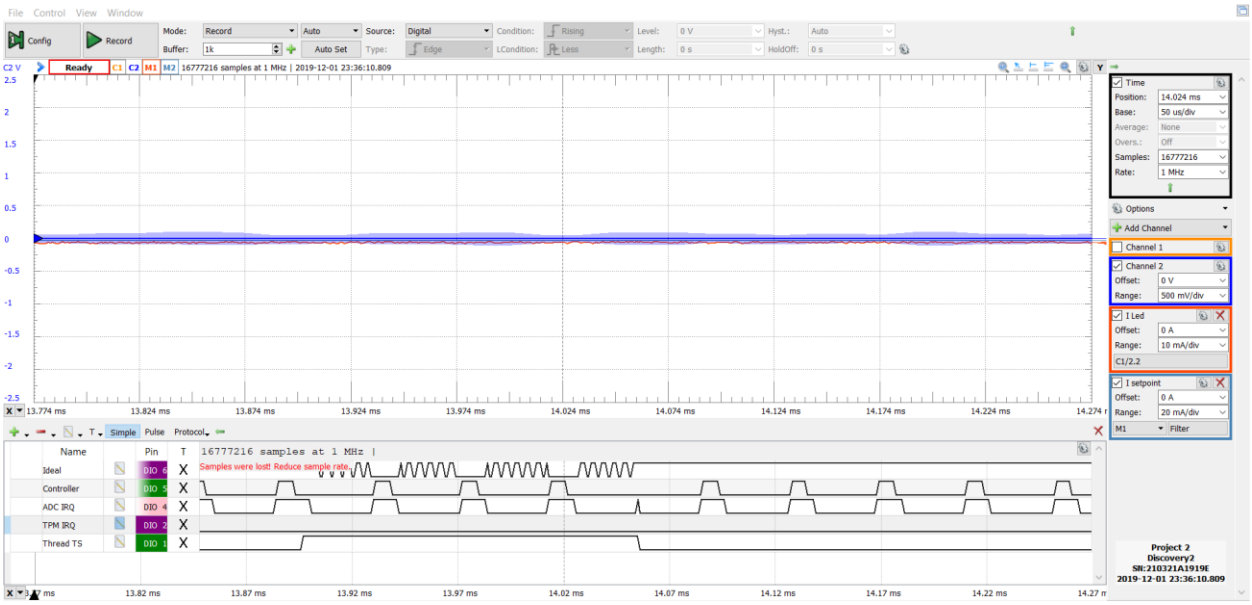
Displaying Currents and Sharing the ADC

Kashyap Ravichandran – kravich2 – ECE 560

Mixed Signal Screenshot showing correct system operation:

The voltage across 2.2 Ohm resistors was measured and the current obtained by Ohm's law is plotted. This current is the feedback current that is supplied back to the system. This is denoted by the orange plot. The blue plot is the DAC output of the set current. This explains the steep increase (almost instantaneous) and steep decrease in value. This screenshot is like the one observed in Lab 3. The important point to note here is that the width of the signal is close to 25 mS and the maximum current set is 40mA.

The Debug signals obtained for the system is shown below:



We can clearly see that there is an ADC pulse between two ADC pulses. This is also a place where the control HB LED function is not called. This means that one of the touch screen values is converted in here. These screenshots show the correct operation of the system.

Current Sharing between the control thread and the plotting thread:

We have two global arrays. These arrays are sort of treated as queues. The control HB LED function puts the g_set_current and g_measured_current into their respective queue. The plotting thread (Update Screen) uses these threads to plot points present in the array. Each array has the capacity to hold 20mS of data, as they are 480 elements big. The update screen thread starts plotting only after the all 480 elements are saved. This reduces the overhead that might arise between two threads trying to communicate with each other. AS mentioned earlier there are a total of 480 elements in each array, however, the horizontally, the LCD can accommodate only 240 elements. (Each pixel represents a time difference and we can't plot 480 elements linearly. The code is therefore written in such a way that, the graph first goes vertically up by 'x' pixels, here x represents the difference in current measurements between two successive measurements, and then horizontally by 1 pixel. This gives us a smooth curve. A diagrammatically representation of the plotting procedure is attached as a scanned picture.

To synchronize the graph to the rising edge of measured current, we employ few static variables. The measured current, is something that oscillates randomly and therefore can not be used for synchronization. We can see that g_set_current is a globally variable that is periodically altered by the software. The value moves between 0mA and a user defined maximum current. Let us consider that this user defined current in our situation is 40mA. The transition from 0mA to 40mA is instantaneous. A simple difference based if condition can spot this transition. The code therefore employs a static variable and it keeps checking it with the current value of g_set_current. If the difference is greater than zero and the number of elements in the array is zero a flag is set. This flag is made zero again only when the global array is full.

In the update screen function, we use these arrays to plot the data onto the screen as mentioned earlier. After using all 480 elements, the number of elements is made zero so that the array could bring in the new set of values.

Refreshing the display:

The display needs to be refreshed every time there needs to be new data to be displayed. Refreshing the display constantly using LCD erase is not the most effective solution as the screen keeps flickering and sometimes, we might not get the default screen at all. Therefore, it is important that we refresh it display selectively. One method to do so is to plot black points in the place where the graph would be. Though the display is refreshed every so often, the process is not as effective as the LCD erase option.

A much more elegant approach to this problem is to employ another array to hold the previous values and then black out the segment of screen that was previously made yellow or blue. This is a much more efficient and an effective way to solve this problem. However, a major drawback with this approach at this moment is that, when we change the wave quickly, the queue doesn't capture all the previous values properly because the values were changing extremely fast. This therefore leaves few residual plot points on the display that is not desired.

Both the solution is present in the code and currently the refreshing code is employed for cleaning out the screen and changing the value of the macro "use_refresh" in "config.h" would enable the selective clearing code.

How many queued ADC request can be serviced at a moment?

The image previously presented for the first question can be used here to explain this answer too. Since we stall the detect Touch screen thread till, we get a value from the ADC, the ADC can service only one queued request between two priority requests. The ADC IRQ looks at the toADC queue and then decides between a priority conversion and a queued conversion. So, the only the ADC IRQ can trigger a software triggered conversion (which these queued conversions are). Therefore, the ADC IRQ can't service more than 1 IRQ between two priority requests. The situation is also explained diagrammatically later in this report.

Extra Credit Work:

1. The color scheme used is altered to match one of my favorite film Star Wars. Apart from the color scheme, an initial screen called a loading screen is made which resembles the starry sky that is shown initially in the movie as the scroll appears.

Reference: https://www.shutterstock.com/image-vector/may-4th-be-you-holiday-greetings-1055961362

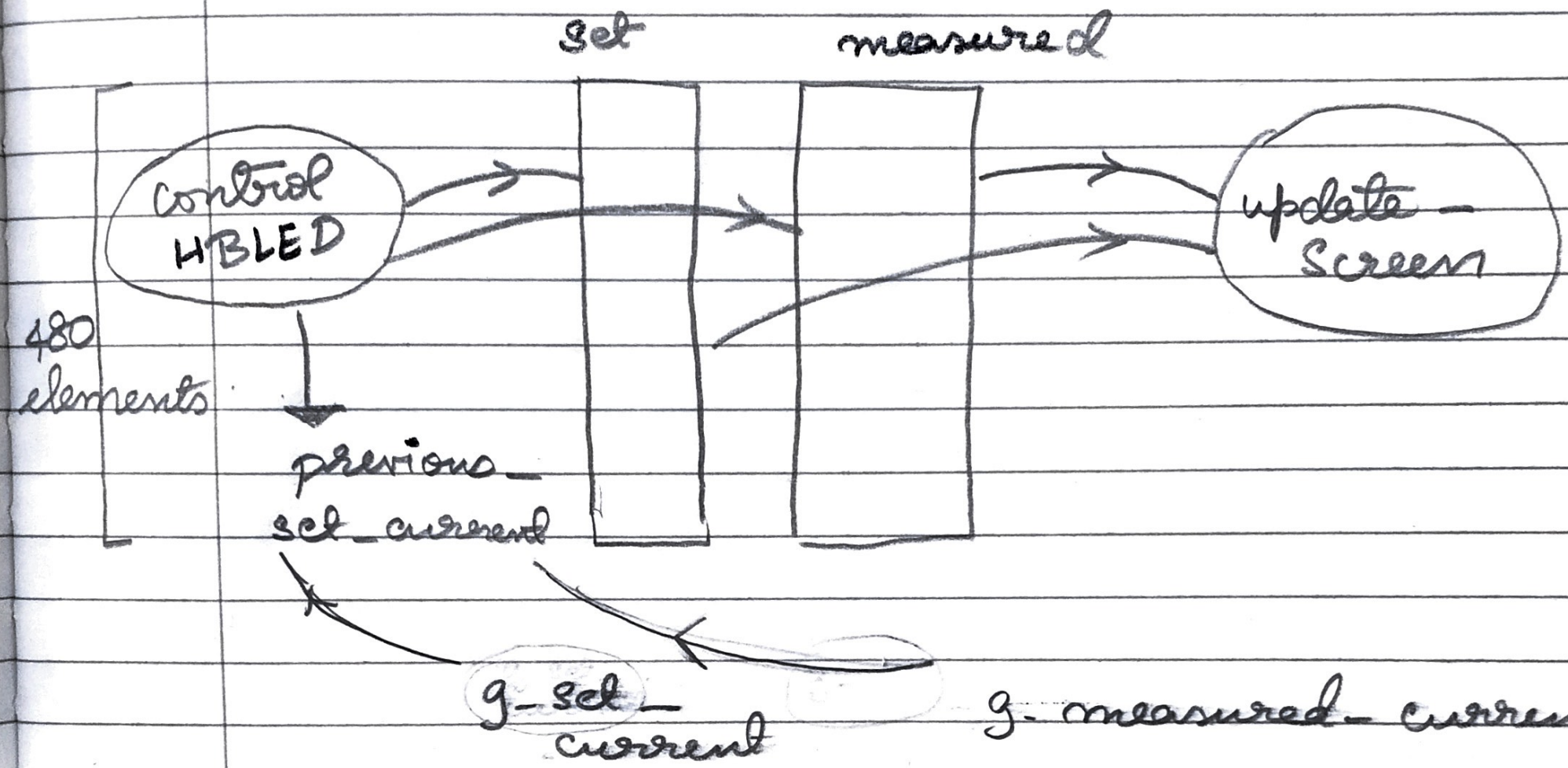
2. A video is made which explains how changing the parameters affect the LED characteristics (pulse width, maximum brightness, etc.)

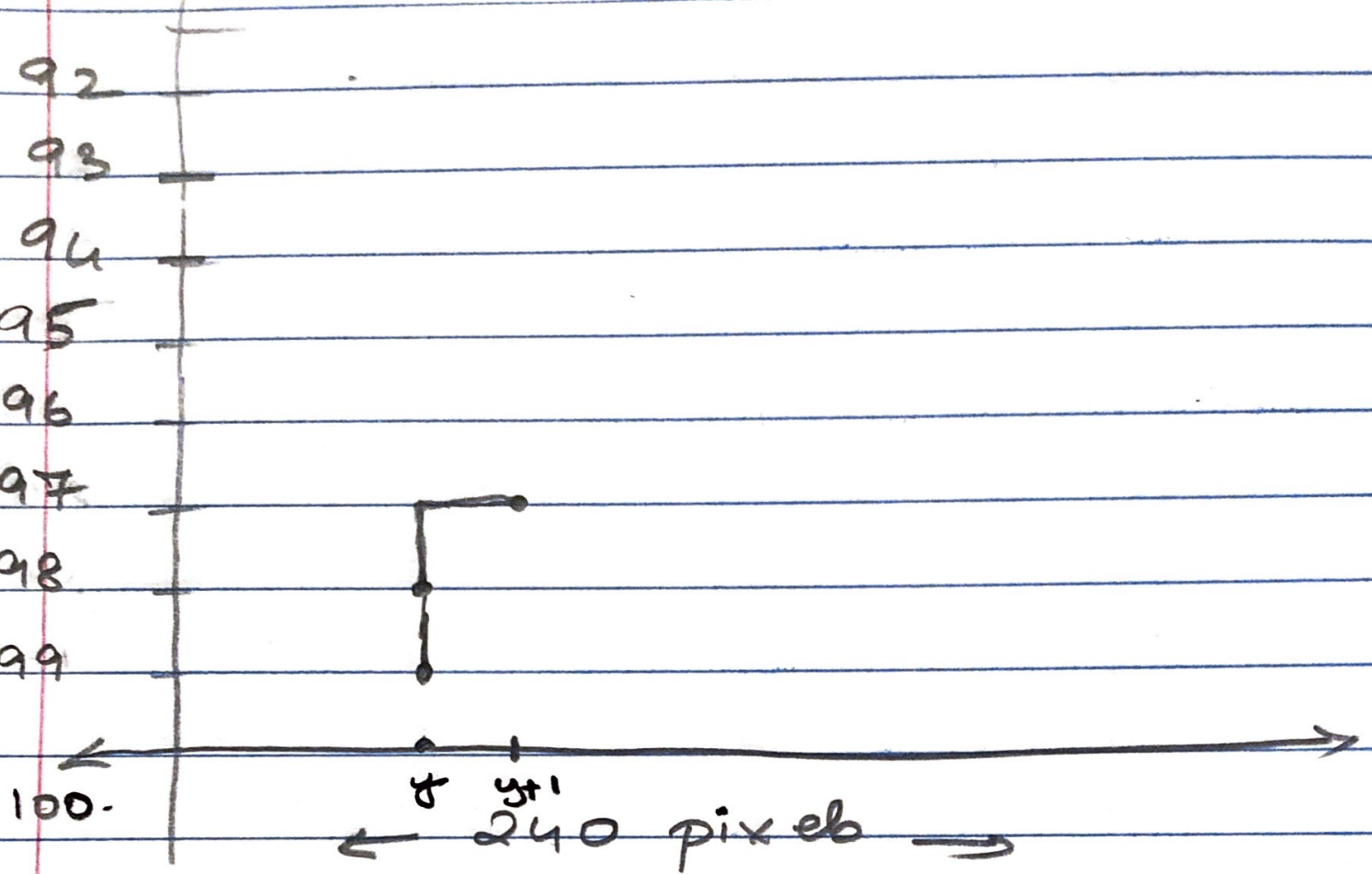
   Link: https://youtu.be/NhM87d34uh4

3. I would also want the refreshing mechanism proposed earlier in this report be considered as work done for extra credit. Also would like the way I have used the plot points and the way in which I have plotted my datapoints to be considered, as I feel that the that particular section of code is unique.

Note: I was given an extension for 36 hours by Dr. Dean, as my shield got shorted somehow and the LEDs, weren't glowing. I have also attached a screenshot of the email thread for your convenience.

Sharing data between control HBLED and thread Update Screen





i = y (arbitrary value)

Say: measured [0] = 1mA

[1] = 2mA.

[2] = 3mA.