ECE 4240 – Lab 5 Report

SPI, DMA, IRQs and Fun?

Course: ECE 4240 – Microprocessor Interfacing

Instructor: Dr. Dean McNeill

Student: Dayaveer Gill (7969885)

Lab Partner: Severyn Shved

Date Performed: November 24, 2025

Date Submitted: December 2, 2025

University of Manitoba

Department of Electrical and Computer Engineering

Laboratory Experiments

Introduction:

This lab consists of three parts, part one will allow us to connect to the LCD and gain familiarity with it. Part two will use the LCD from part to compare the excution time difference with the non-DMA and DMA speeds. Part three requires to complete the ping pong game to be able to play the ping pong game on the LCD with a potentiometer as its controls.

Part 1:

For the first part of this lab we begun by following the procedure given in the lab 5 document. We configured the IOC as directed, downloaded and moved the given ST7735 files to the driver folder and then added the given code in its place. After all that was done we connected the board to the LCD using the pin-out given in the lab manual and ran the program. The LCD printed text in 3 different sizes and colors on the first screen then the screen changed to each color and ended with a picture of a parrot before restarting all over again. The output was shown to the TA as well the pictures are shown in figures 1, 2 and 3 in the appendix.

Part 2:

Part 2 built upon part 1 and we followed the lab manual closely, changing the code first and running the program which irritated through hundreds of drawing operations and printed the average execution time for each at the end which is shown in figure 4 in the appendix. We then modified the IOC as directed in the lab manual and changed the DMA SPI value from 0 to 1 in the st7735.h file. After doing al that we reran our code which irritated through hundreds of drawing operations and printed the average execution time for each at the end again but this time the the excution time was much faster and the results are shown in figure 5 in the appendix.

Part 3:

To complete and play the pong like game we began by downloading the required files and moving them to their appropriate locations. Then we enabled ADC1 with input on channel 1 Pin PC1, this input was taken from the potentiometer voltage divider. Then we wrote the code to complete the ping pong game for which the code is show in figure 6 in the appendix. After all this was done the game was functional and we showed the TA and pictures are shown in figures 7, 8 and 9 in the appendix.

Conclusion:

Overall this lab taught us how to use LCD, the time difference between the non-DMA and DMA, as well as how to set up a ping pong game that is controlled by a voltage divider potentiometer. Part 1 required us to follow the lab manual and see output on the LCD which we were able to observe. In part two we compared the speeds of non-DMA and DMA execution times and were able to see that DMA execution times were significantly faster, with a overall speed up of 73.08 times. Part 3 required us to finish the code for the ping pong game and play it using the potentiometer which we were able to do.

Laboratory Questions:

Question 1:

In part 1 the SPI2 mode is set to Transmit only master, and SD card needs the full-duplex SPI2 for it to function. Since right now there is no MISO the board will not be able to communicate with the SD card and it will not work.

Question 2:

The overall speed up ratio is 73.08.

Drawing Operation	non-DMA(µs)	DMA(µs)	Speed Improvement
Fill Screen	87046.49	809.41	107.54
Draw Pixel	28.46	0.47	60.55
Draw Line	2478.63	12.72	194.86
Draw Circle	2030.10	2.67	760.34
Fill Rectangle	5500.87	48.30	113.89
Fill Circle	1090.11	97.93	11.13
Fill Triangle	2094.11	153.95	13.60
Draw Triangle	4041.35	23.16	174.50
Horizontal Line	66.46	7.18	9.26
Vertical Line	88.4	1.95	45.33
Draw Image	94396.66	1563.47	60.38
Overall Time	198861.64	2721.21	73.08

Table 1: Speed Improvement between non-DMA and DMA.

Question 3:

The code in the st7735.c file uses double buffering in the ”CopyBufferDMA” function which helps speed up the drawing operations.

Question 5:

Question 5.A:

We can’t use DMA1 since it is being used by the SPI2_TX on DMA1 stream 4. DMA1 also is not capable of memory to memory mode.

Question 5.B:

In the st7735.c file on line 83 we see function ”CopyBufferDMA” is called to copy from frame-buffer to secondary frame-buffer.

Question 5.C:

It can stop or slow other peripherals depending on how many requests DMA2 is getting, such as from the ADC’s, TIM’s, etc.

Question 6:

Question 6.A:

No, using the raw ADC opposed to the calibrated ADC value does not matter much in this scenario since we don’t need require absolute voltage accuracy.

Question 6.B:

The effect of reducing the EWMA alpha constant is that the bar moves smoother but much slower.

Question 6.C:

Increasing it to 0.80 the bar moves faster, but is less smooth.

Question 6.D:

At EWMA alpha at 1, nothing is filtered.

Question 7:

After reading the lab manual, section 3.1.1 shows that 1 screen transfer takes 15.2 milliseconds. Therefore we get that the FPS is 1/0.0152 which is equal to 65.80 FPS.

Question 10:

One of the step you would need for a two player game is configuring the IOC to enable a 2nd ADC input for the 2nd player. Another step that you would need is setting up a potentiometer voltage divider for the 2nd player. controls.

Appendix:

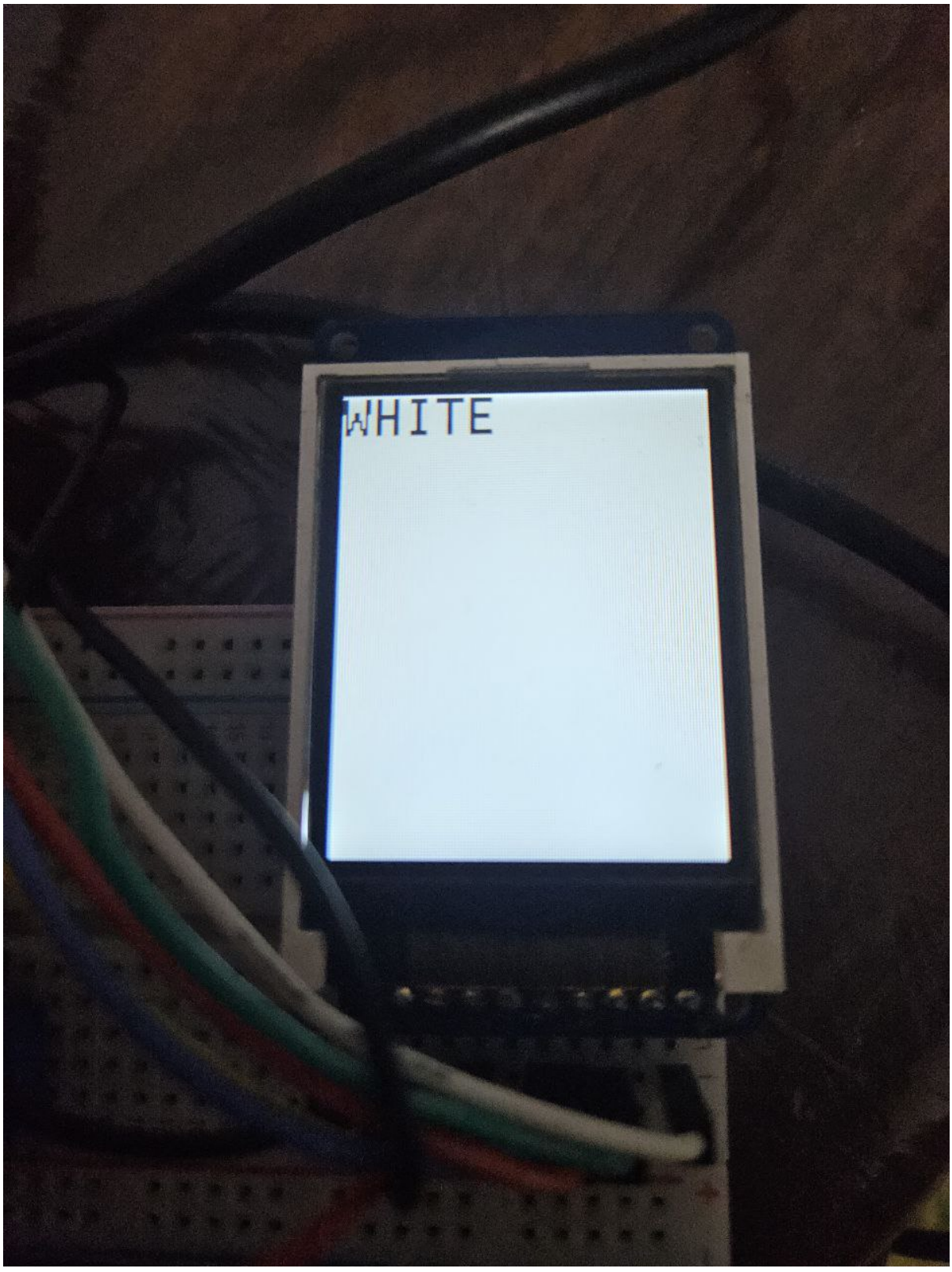

Figure 1: Part 1 Results.

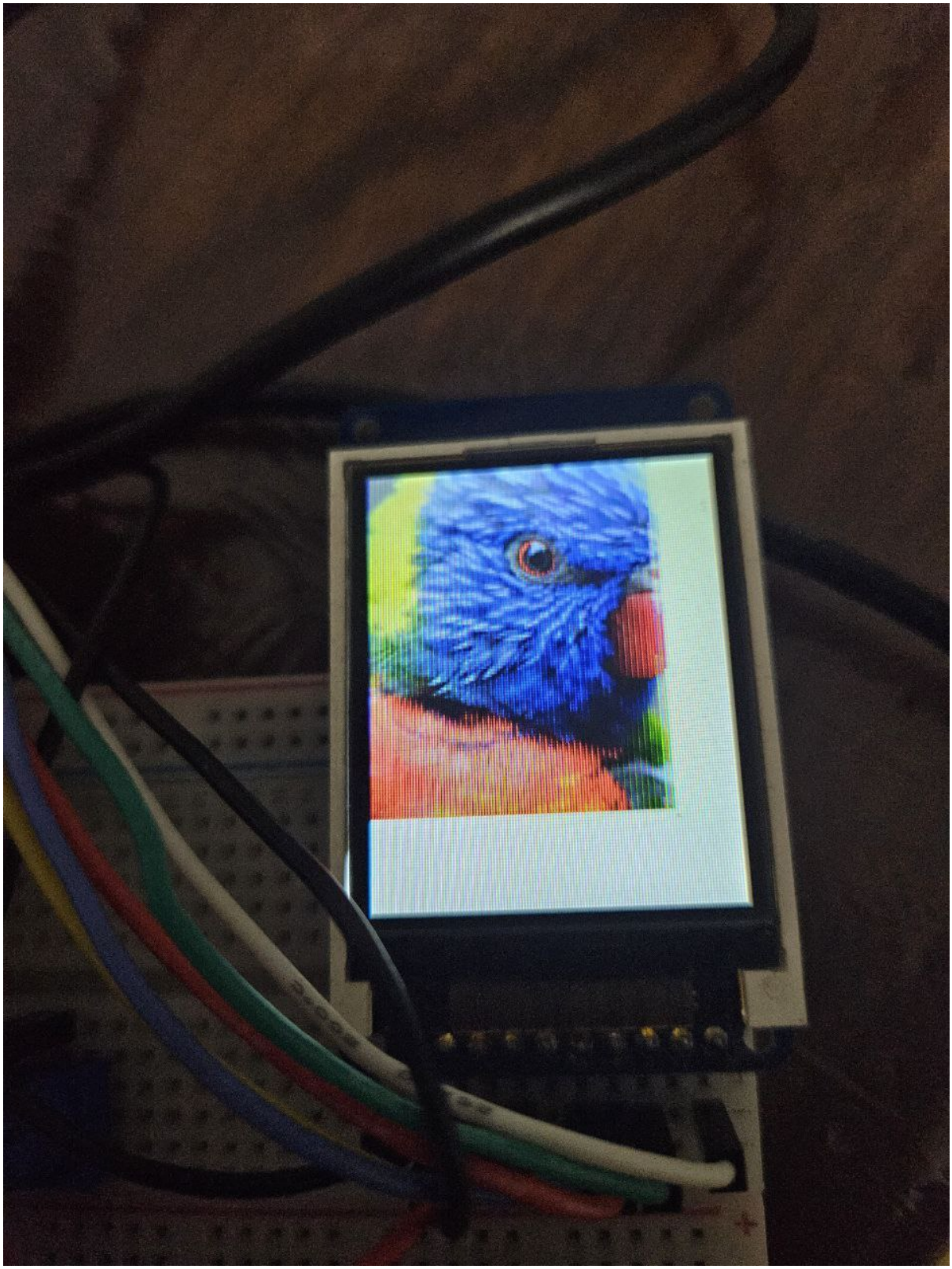

Figure 2: Part 1 Results.

Figure 3: Part 1 Results.

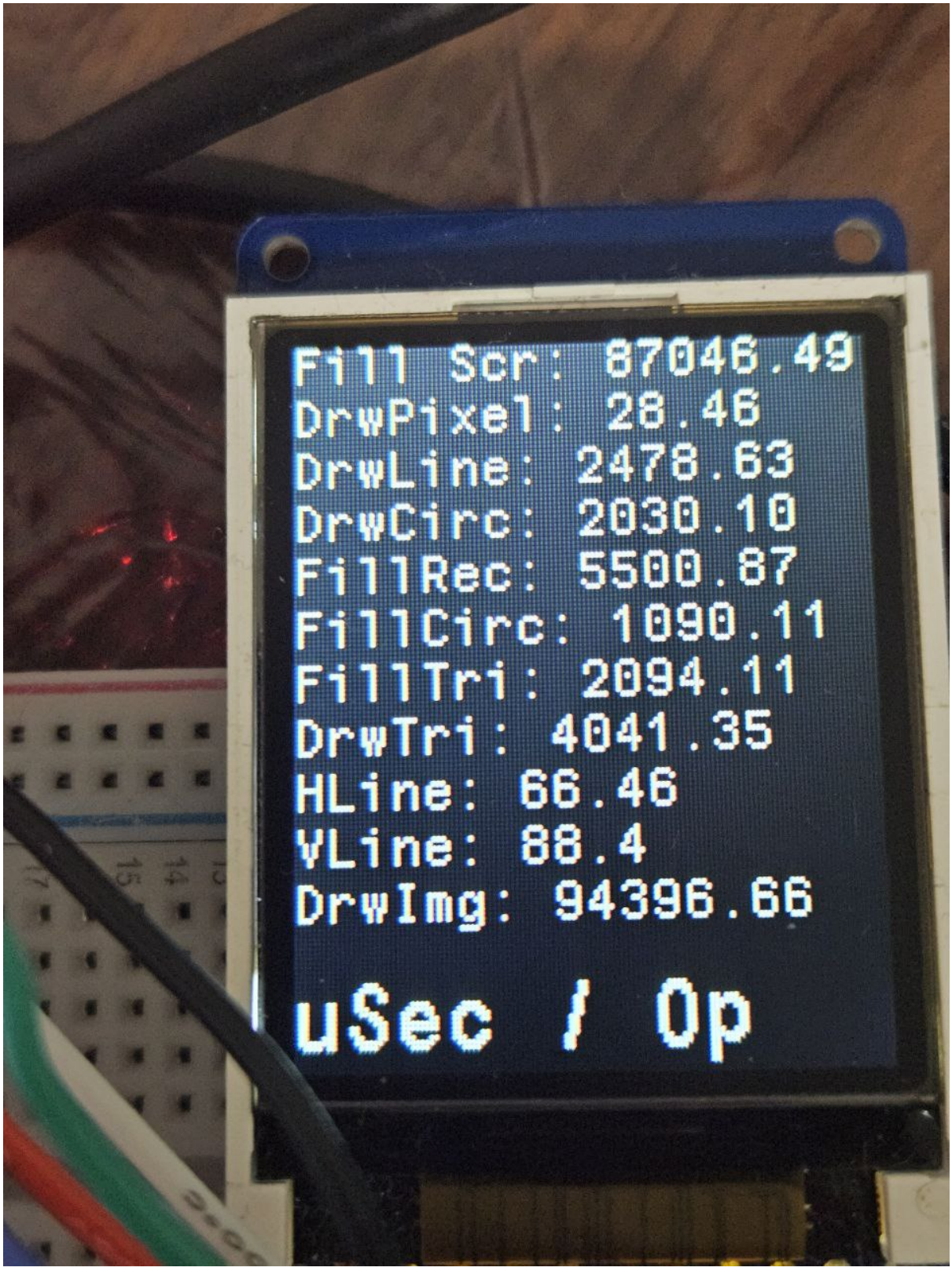

Figure 4: Part 2 Results.

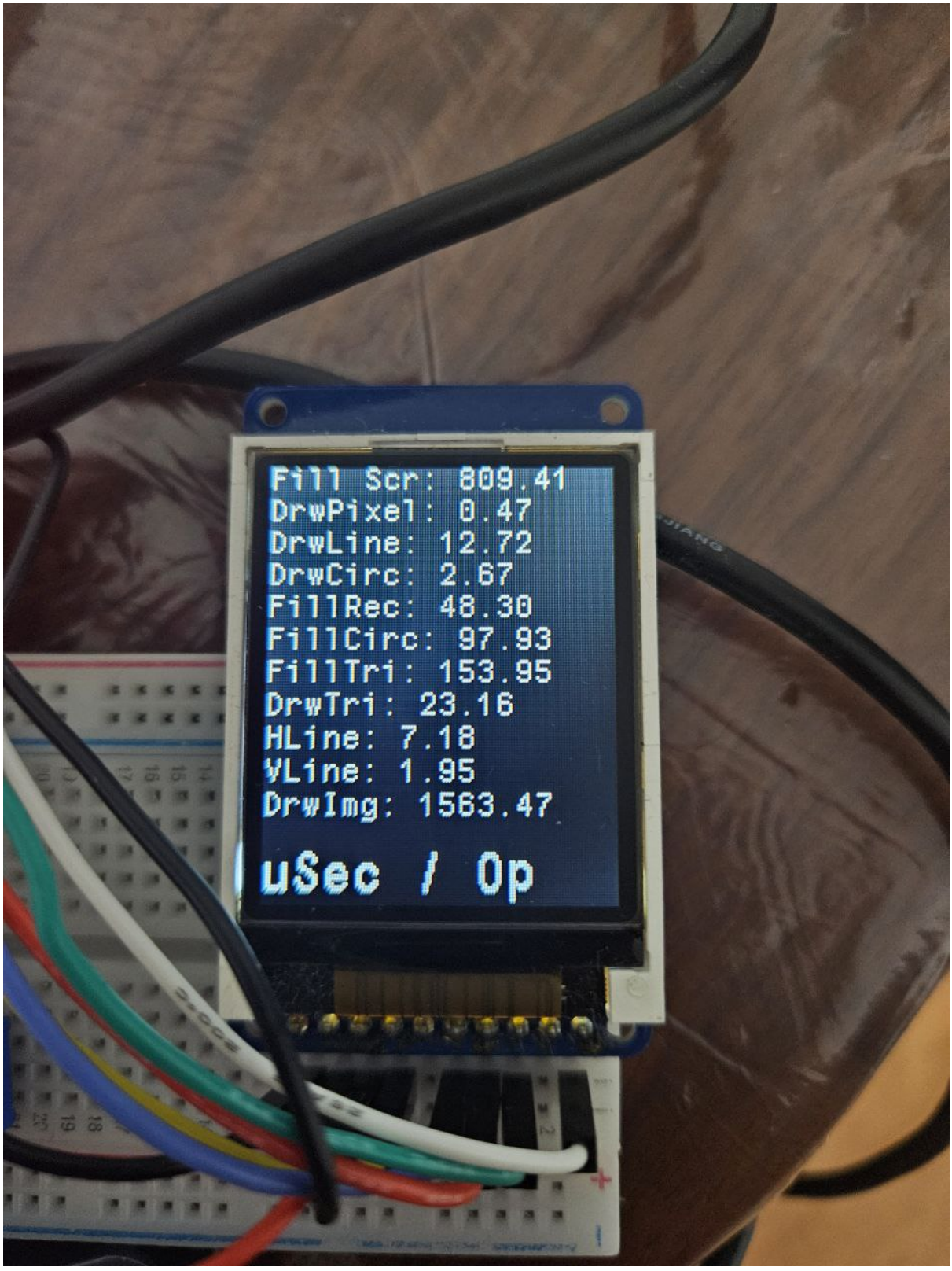

Figure 5: Part 2 Results.

```
while (1){
  start = HAL_GetTick();
  int i = 0;
```

```
uint32_t sum = 0;
while(i < 16){
    uint16_t ADCValue = sampleRawADCValue(&hadc1, ADC_CHANNEL_11,
    ADC_SAMPLETIME_480CYCLES);
    sum+=ADCValue;
    i++;
    }
oversampledADCAvg = sum/16;
float filteredADCValue = (EWMA_ALPHA * oversampledADCAvg)+ ((1
EWMA_ALPHA)*filteredADCValue);
runPongGame(filteredADCValue);
}
```

Figure 3: Part 3 Code.

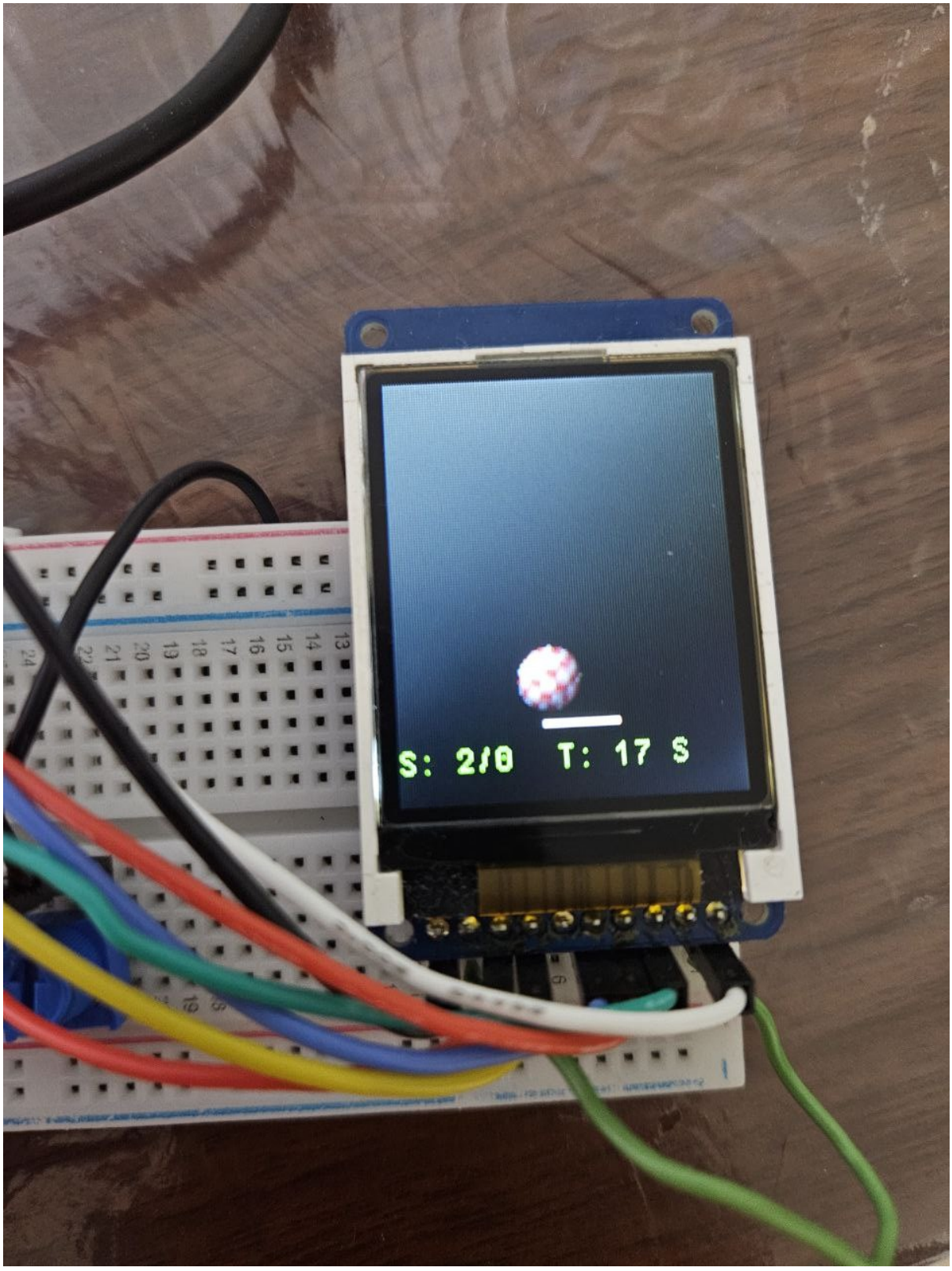

Figure 7: Part 3 Results.

Figure 8: Part 3 Results.

Figure 9: Part 3 Results.