**EGRE 364 – Microcomputer System**

**Laboratory Number 8**

**Sensor Interfacing**

**Lab Section: Tuesday 1PM**

***Lab conducted on: November 26th , 2019***

***Report Submitted on: December 9th , 2018***

**Papa Beye & Omar Amr**

**Major: CPE & CPE**

**Introduction**

The purpose of this lab was to utilize the use the ADC modules, LCD driver, GPIO pins to interface with the distance and reflectance sensor and gathered data and display the data on the LCD display using C language in the Keil µVision software development environment. To achieve this objective, we used GPIO registers, provided LCD driver & ADC initialization, internal ADC conversion, LCD pin configuration, and GPIO interfacing. The main objective was to utilize the given ADC and LCD library to gather the analog signal from distance sensor and convert it into a distance in cm and use GPIO I/O line manipulation to detect low or high reflectance from the reflectance sensor. We used the provided LCD driver to display a string to the LCD display and use the provided LCD bars to indicate the a scaled ratio between the distance sensor’s operating range from 10 cm – 50 cm. We reported the sensor values of “B” or “W” for each sensor to the LCD display, along with the current distance and the appropriate distance bars. The distance work by outputting an analog voltage signal proportional to the distance it detects. We can read the analog signal to the ADC to convert it to a value, since we were using a 12 bit length and , we could get the read input voltage from the distance, as seen below in the equation. Once we determine the , we could use the line of best fit function determined experimentally to find the distance, as seen below. For the reflectance sensor, since these are digital I/O sensors, we created a routine that would set the I/O line to output and set high and allow for the capacitor to charge using a delay; then we set GPIO pin to an input and measured the length of time that capacitor needed to discharge thus allowing us to determine if it detected “black” or “white” by compare the length of time to the given threshold. The LCD driver and functions “LCDDisplay\_String” and “LCDDisplay\_char” allowed us to display strings, which included the distance in cm and the “W” or “B”. The LCD display that we used a LCD 4x24 segments, 4 commons, multiplexed 1/4 duty, 1/3 bias. The configuration, setup, and C function used in this lab can be seen below. We implemented a delay function to be able to smoothly gather and process data and to be able to detect a change in the reflectance sensor. In this lab, we used the practiced methods and material that we learned in class about LCD, ADC, and sensor detection.

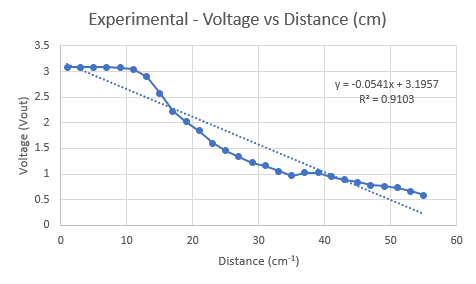
**Functionality and Correctness**

The STM32L4 discovery board has many GPIO ports but we used two for the distance sensor and the two reflectance sensors. In order to interface and gather data from the distance sensor and reflectance sensors, we had to utilize the given library of ADC, LCD Driver, and Sensor configuration jointly to be able design functions for the sensors. We learned that to correctly use the distance sensor, we had to initialize the GPIO port A pin to connect to the ADC, start the conversion through software, wait for the result and read the translated result from the ADC\_DR (Data Register). Using the formulas below, we calculate the distance from the two equations and displayed the distance(int) onto the LCD display. Next, for the reflectance sensor, we had to follow the procedure detail below; the reflectance sensor was designed to interface with a digital I/O line to charge the capacitor and observe the length of time it takes to discharge thus allowing us to determine the reflectance value. To create the read reflectance sensor function, we manipulated GPIO configurations and initialize the GPIO port E to use for both reflectance sensors. In order to set the LCD display to display “W” or “B” for each sensor we used the provided function “LCDDisplay\_String” function. The provided functions un

LEDs connected to PE8 & PB2 pins of the board processor. Similar to lab 1, we enabled the clock of both ports, set the mode of the pins as outputs, and set the value of the output of each pin using the ODR to toggle the LED on and off. We used the delay function shown below to create a delay right after we toggle the value of the ODR value of the red LED (PB2). Our delay used the SysTick interrupts ability to interrupt every 1 ms, allowing us to delay for 1000 ms, or 1 sec, for the periodic flashing of the red LED. Using a global variable, we used two other external interrupts from the left button (EXTI1) and the right button (EXTI2) to alter the delay variable to simulate a quicker or slower flashing rate. In order to the 3 external interrupts, we must initialize them beforehand. First, like in the SysTick initialization, we set enable the interrupt number of EXTI0, EXTI1, EXTI2 by using the given function NVIC\_EnableIRQ(), which is detailed below. Next, we set the SYSCFG external interrupt configuration for EXTI0, 1, 2 by clearing their respective bit in the control register and then set the bit. In order to cause the external interrupt with the press of the button, transition from ‘0’ to ‘1’, we need to set the external interrupts as rising edge by setting the EXTI rising edge trigger selection register for each respective bit. Finally, we needed set each respective bit of EXTI 0, 1, 2 of the EXTI interrupt mask register. The initialization documented here is detailed below. After initialization, we were able to write the interrupt handler, which are called by the NVIC when the external interrupt is triggered. In each interrupt handler, we had to clear the corresponding pending bit to allow future interrupts by writing the pending to a “1”. Doing so, we were to complete tasks inside each interrupt handler, such as increasing or decreasing the delay or toggling the green LED.

**Equation for ADC Result and**

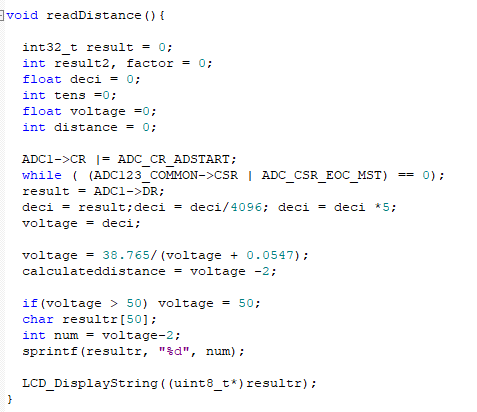
**Experimentally found Voltage at numerous distance to produce a linear line of best fit**

****

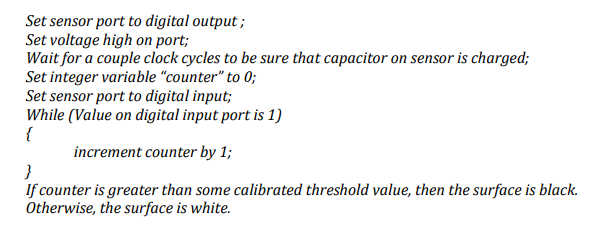
**Using experimental data, found inverse distance and found line of best fit, used in code to determine distance using the input analog voltage converted to digital through ADC**

****

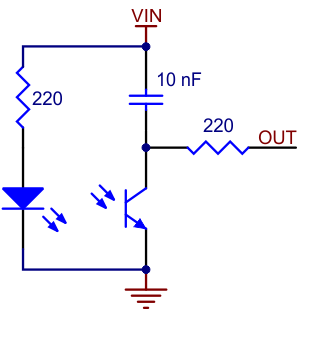
**Detailed code of Read Distance function**

****

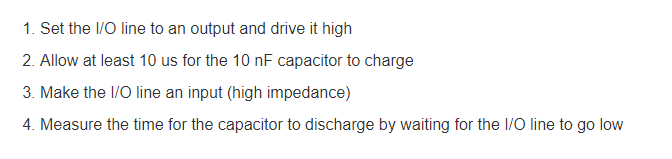
**Pseudo Code/routine to determine if reflectance detects White or Black**

****

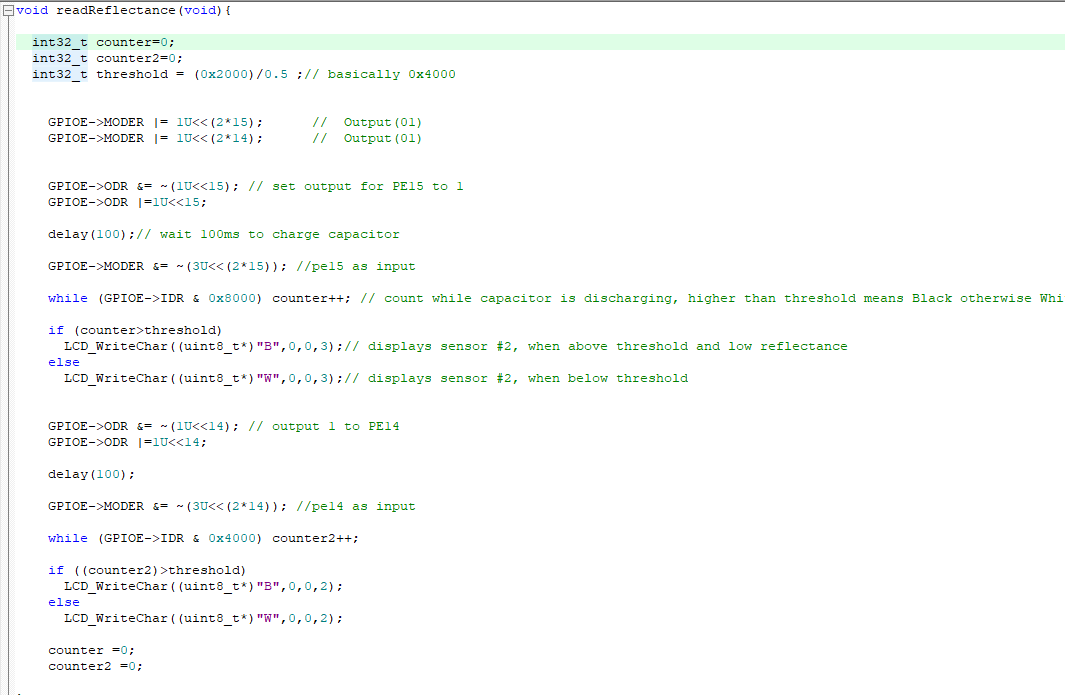
**Diagram of Polulu Reflectance Sensor**

****

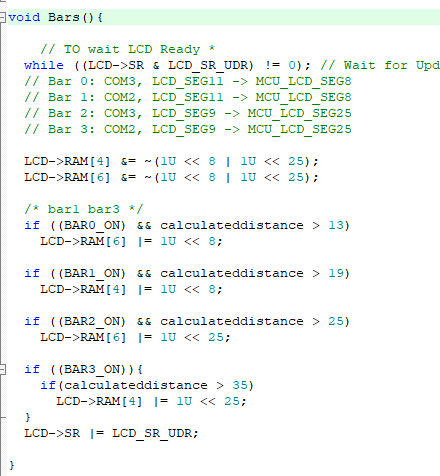
**Sequence used to determine using I/O lines**

****

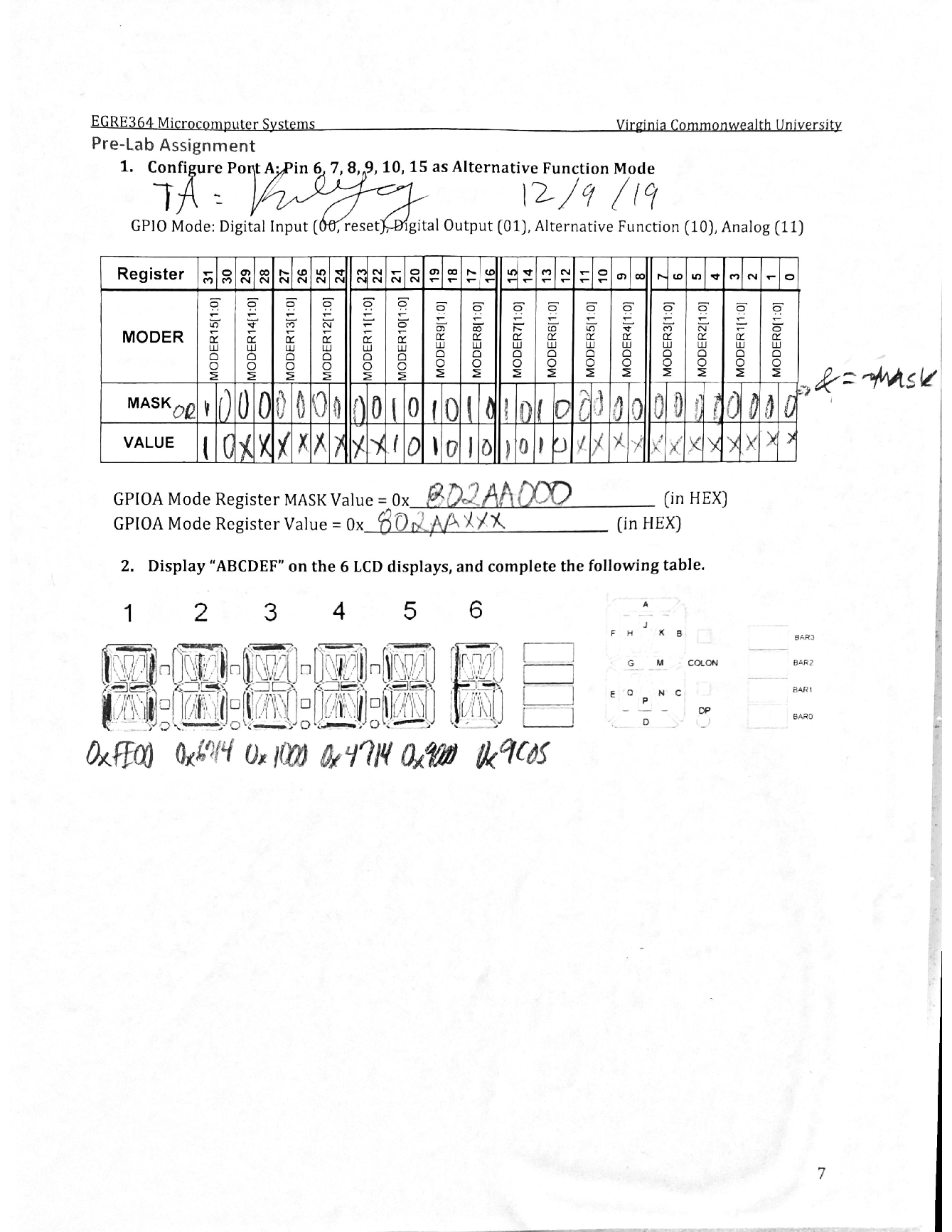
**Detailed code of Read Reflectance function**

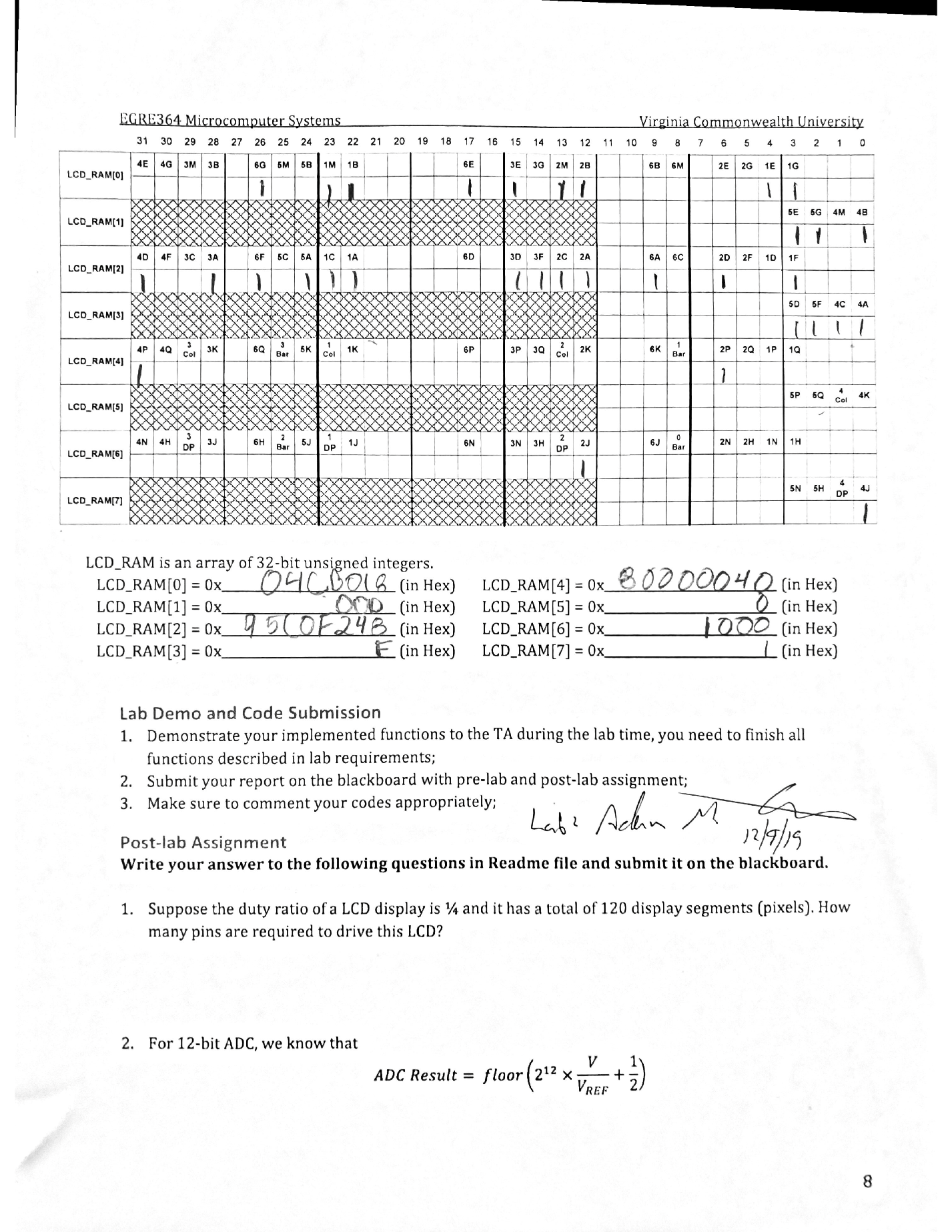
****

**Bars Function used to turn on LCD Bars based on global distance determined from distance sensor**

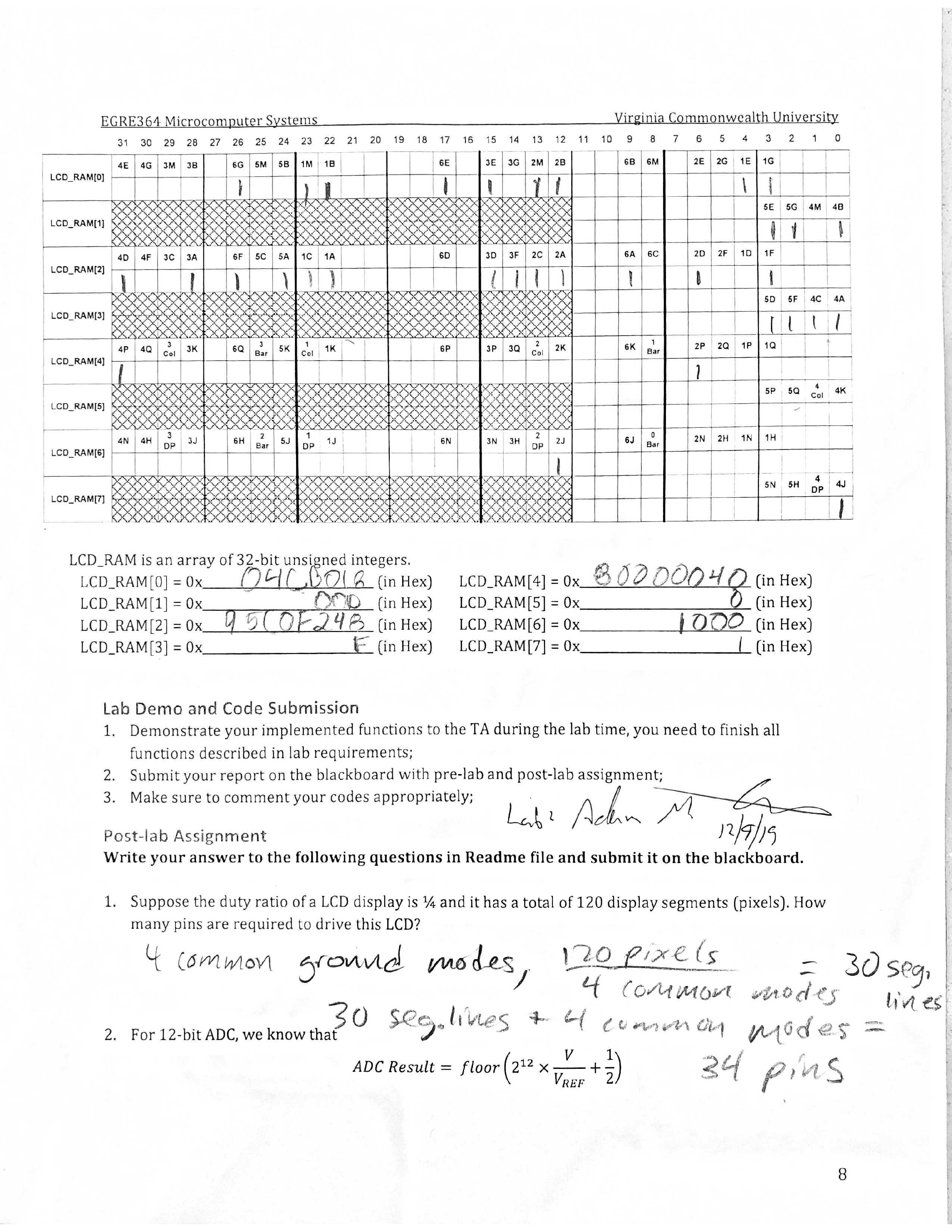
****

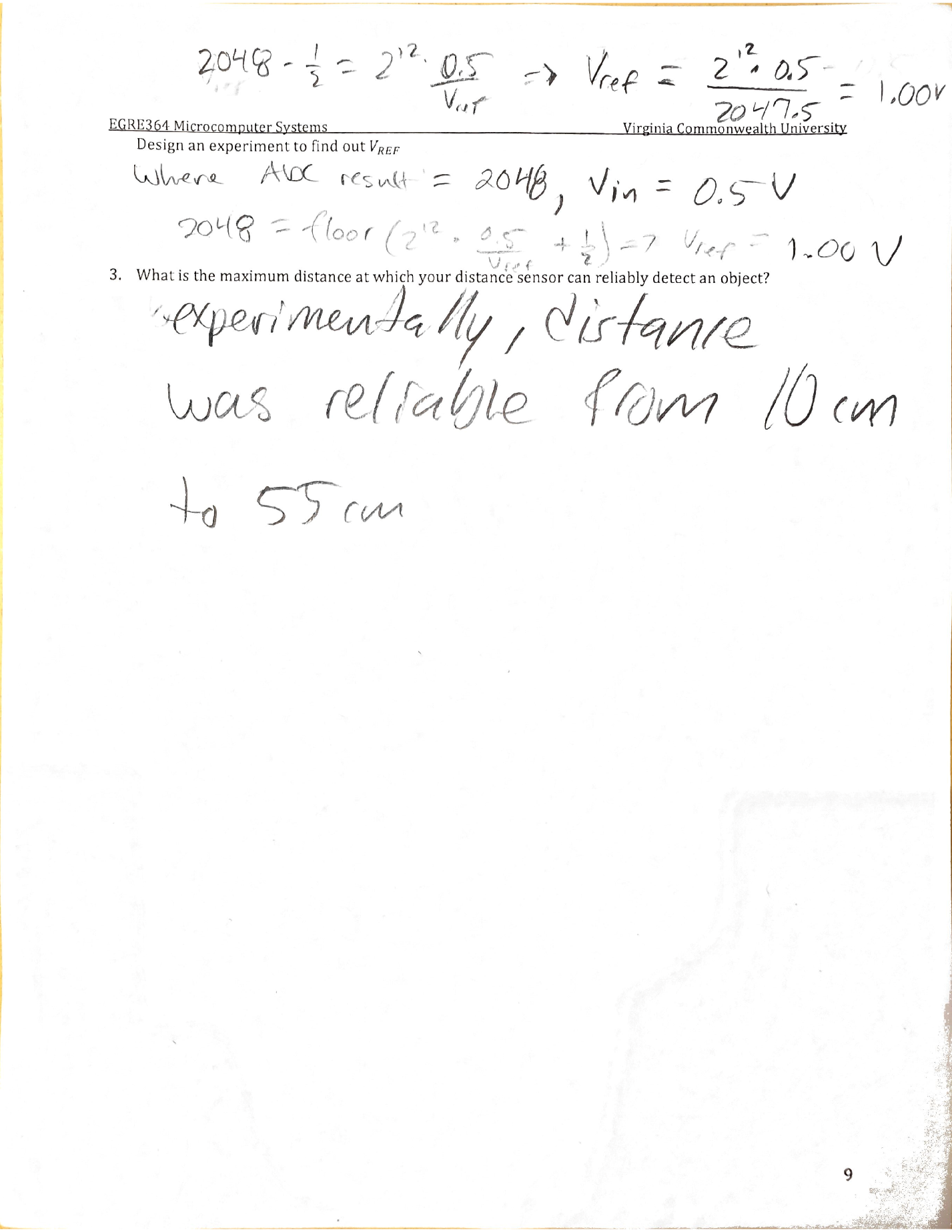
**Pre-Lab and Demo**

****



**Post-Lab**

****

****

**Conclusion**

In this lab, we became more comfortable with interfacing with practical GPIO initialization and utilization alongside external interrupts to occur while flashing the red LED. In this lab, we learned about initializing and utilizing external interrupts to complete tasks and how to apply it with our STM32L4 board. We were able to achieve the objective of flashing the red LED and toggling the green LED with an external interrupt; we also implemented two other external interrupts using the two side buttons on the joystick to increase or decrease the rate/delay of the flashing LED. We used previous knowledge of implementing the flashing green and red LED from lab 1 to complete this lab. We learned a lot and gained more experience about how to use external interrupts and how they work on the STM32L4 board. We didn’t face any issues but there was a learning curve that allowed us to learn and understanding how initialized and use external interrupts in our lab to properly complete our objective.