CSUN | COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

ECE 425

Microprocessor Systems

Final Project
Arithmetic Calculator

Abisai Vilches
Brandon Vargas
Instructor: Aaron Nanas
Fall 2024

Introduction

This project consisted of implementing an arithmetic calculator using EduBase Keypad buttons as inputs. The user interacts with the keypad buttons and visualization of the input and output is provided through the teraterm pro terminal which was used in previous UART experiment in the lab. The calculator functions by printing a layout of the keypad buttons, prompting the user for input (values and operation) and lastly prints out the result of the calculation. This report covers the components used, microcontroller peripherals involved, detailed functionality, challenges encountered and solutions implemented.

Background and Methodology

In this project, we used the following embedded systems applications: GPIO, SysTick, and UART.

- GPIO: used for the keypad (which in turn was used as the input for numbers and operations) and the EduBase board LEDs, which lit up depending on which keypad button was pressed
- SysTick Timer: used for debouncing, since the keypad does not have any debouncing hardware. This allowed for the desired inputs to be made (one input for each button press)
- UART: used to display the calculator layout, the user's inputs, and the result

Block Diagram

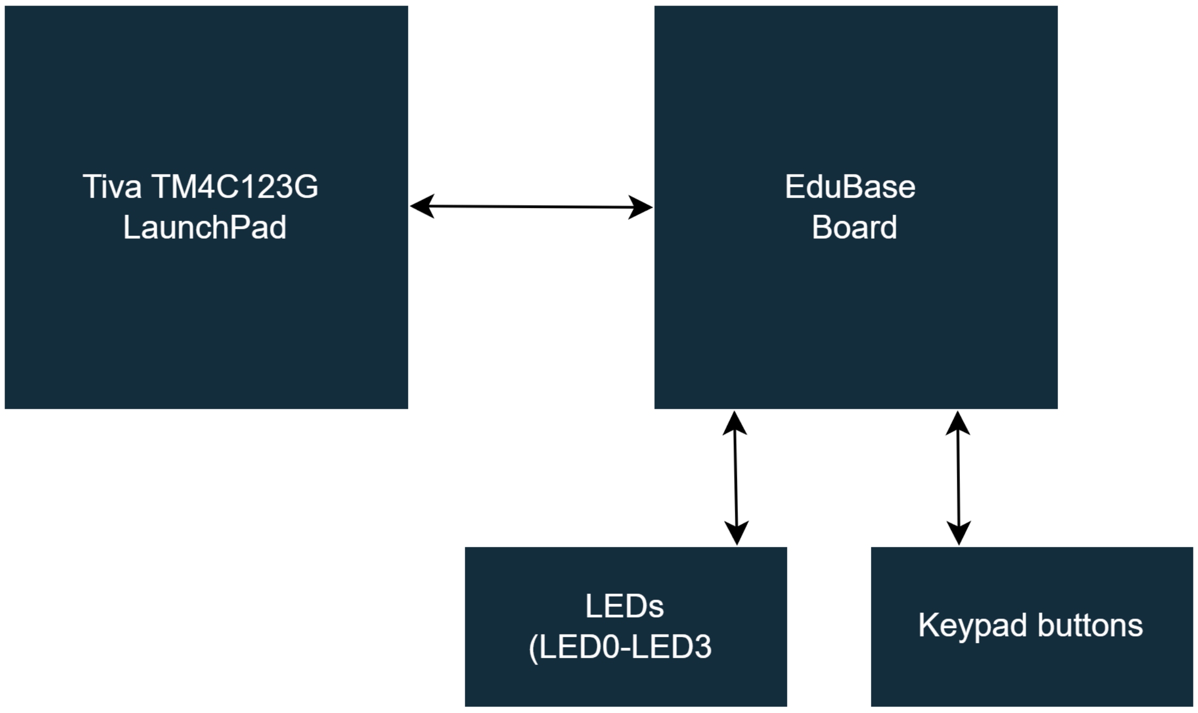


Figure 1: Block Diagram for peripherals used

List of Components

Description	Quantity	Manufacturer
Tiva TM4C123G LaunchPad	1	Texas Instruments
EduBase Board	1	Trainer4Edu
EduBase Board Keypad	1	Trainer4Edu
Teratermpro (terminal used to display calculator)	1	N/A
USB-A to Micro-USB Cable	1	N/A

Pinout

Pins	Function
PA2	Keypad
PA3	Keypad
PA4	Keypad
PA5	Keypad
PD0	Keypad
PD1	Keypad
PD2	Keypad
PD3	Keypad
PA1	UART

PA0	UART

Challenges/Solutions

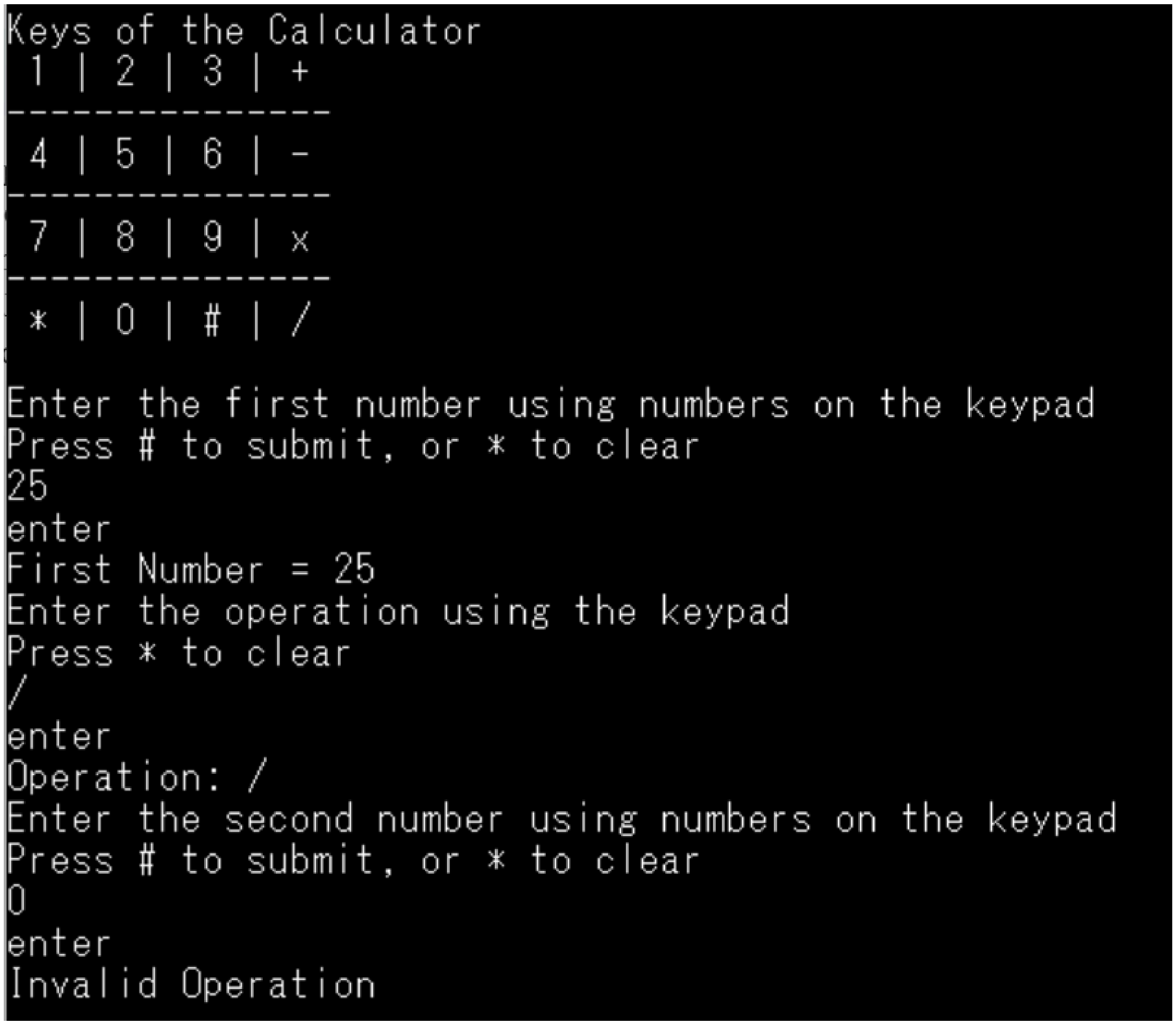
The initial project concept evolved as various challenges arose during the implementation process. For example, the calculator output was to be displayed on the EduBase Board LCD screen, but later determined that using the tera term pro terminal would streamline and facilitate the project. Another challenging area was creating a function in which the user could press any keypad as many times as needed to write their value and also display it on the terminal. To solve this problem, a while loop was used so that the user could press the '#' keypad button once their desired value was entered. This method was used for all the user input including the 1st value, operation, and 2nd value.

Another problem that arose was how to implement the clear functionality. We wanted to create a break in the code to reset, but breaking would cause us to exit the while loop used to run the calculator. To solve this, we made an outer while loop that always runs, and sets the variable *reset* to 0 at its beginning. In our code, we implemented the "input" variable to change to '#' when the clear button was pressed, exiting the Num_Decode_EduBase_Keypad function or the Op_Decode_EduBase_Keypad function (depending on what stage of the calculation process we are at). Additionally, pressing the clear button would set the *reset* to 1, which would then be checked for upon returning to the main function. If *reset* = 1, then we'd break out of the inner while loop and into the outer while loop, which sets *reset* back to 0 again and restarts the calculator process.

Analysis and Results

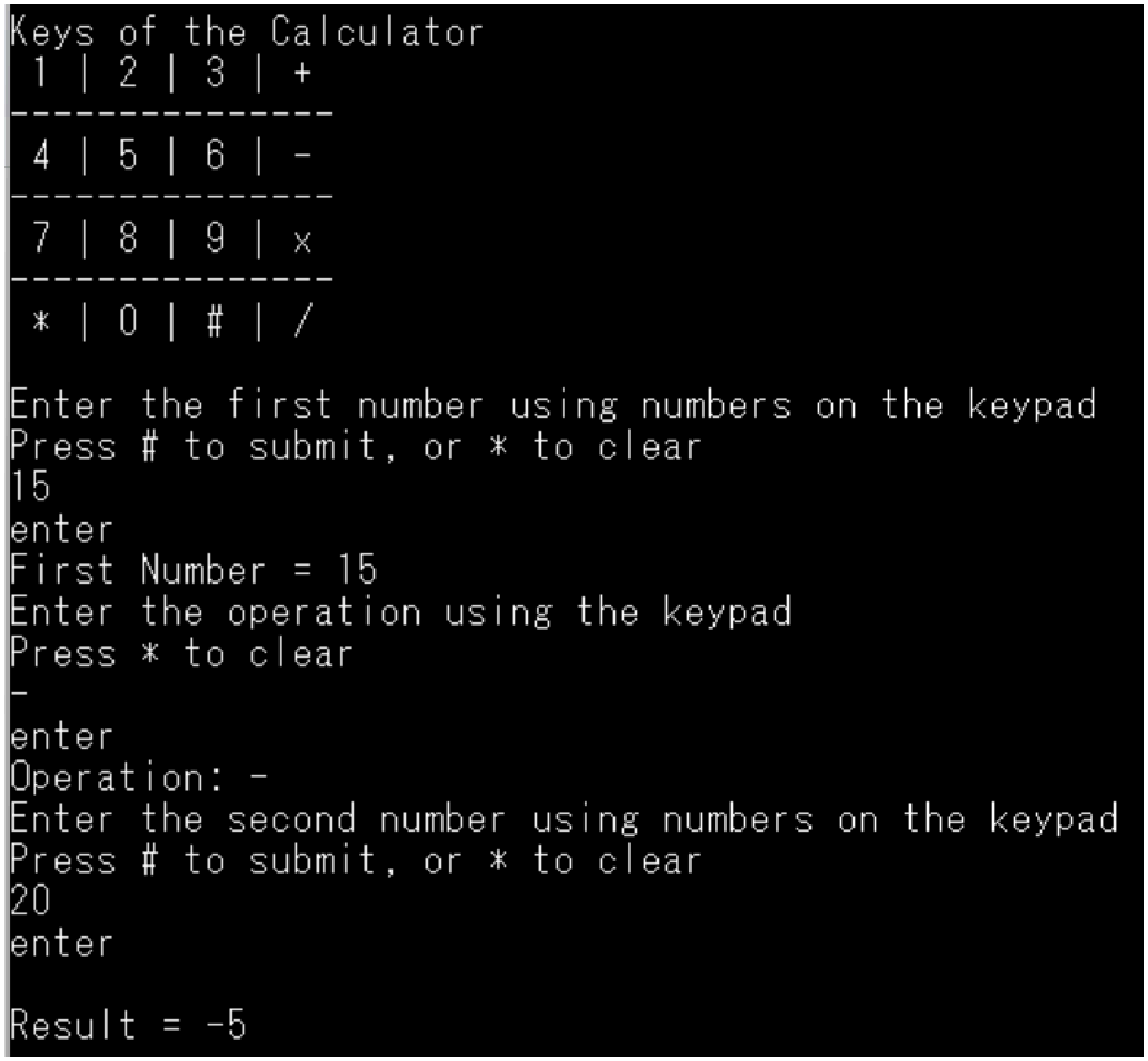
The following images show the results of our arithmetic calculator. Figure 2 shows the divide by zero case. This case results in an error message being displayed. Figure 3 depicts subtracting by a larger number than the first input. The UART code doesn't have any functions to display signed values, so we had to work around this. Instead of performing input1 - input2 (in this case 15 - 20) we performed input2 - input1 (20 - 15) and used the UART code's string display function to append the negative sign in front of the result. Figure 4 shows what happens when the reset button is pressed. A reset message is displayed on the screen, and then the first input is requested again. All previously input values have been reset. The image shows the reset happening during the operation request, but resetting can also be executed during either of the value request stages (i.e. when putting in the first or second number). Figure 5 shows the calculator performing a normal task, in this case multiplication.

There is an integer limit regarding the numbers that can be used or displayed: 4,294,967,295. If any further improvements were to be made to this project, one would be to prevent inputting or outputting values larger than the one just listed.



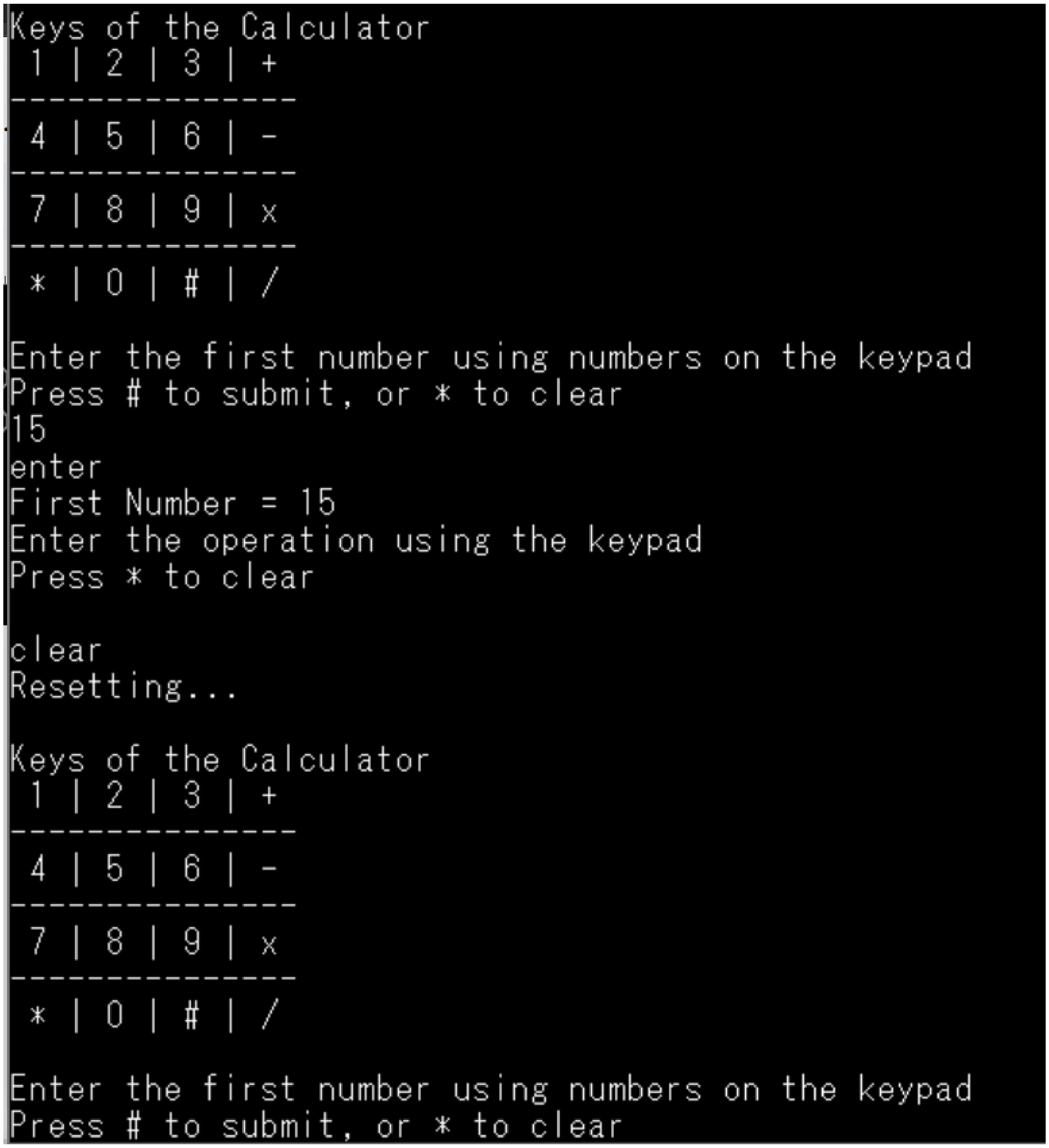
```
Keys of the Calculator
 1 | 2 | 3 | +
---------------
 4 | 5 | 6 | -
---------------
 7 | 8 | 9 | x
---------------
 * | 0 | # | /

Enter the first number using numbers on the keypad
Press # to submit, or * to clear
25
enter
First Number = 25
Enter the operation using the keypad
Press * to clear
/
enter
Operation: /
Enter the second number using numbers on the keypad
Press # to submit, or * to clear
0
enter
Invalid Operation
```

Figure 2: Arithmetic calculator output when dividing by zero



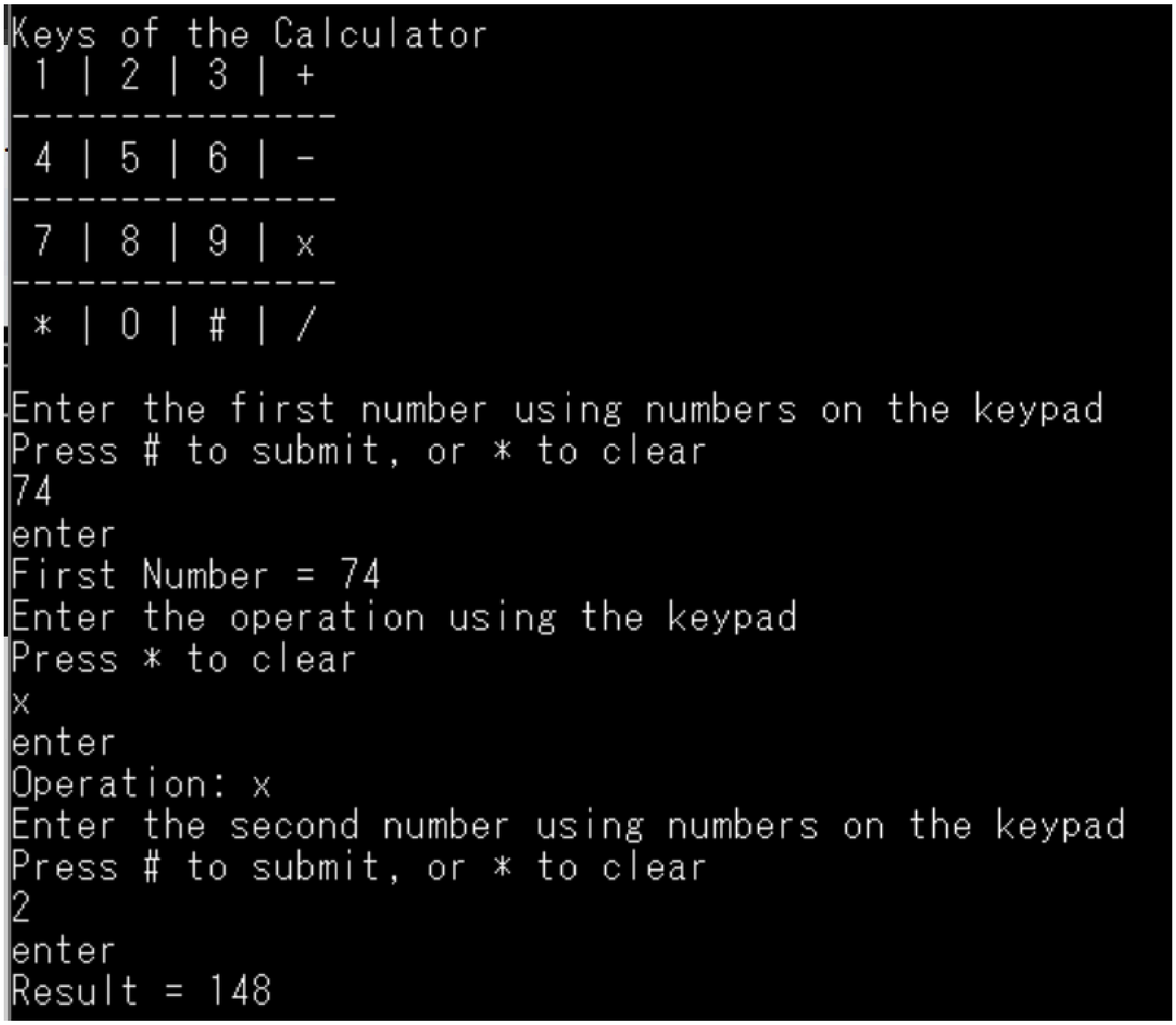
```
Keys of the Calculator
 1 | 2 | 3 | +
---------------
 4 | 5 | 6 | -
---------------
 7 | 8 | 9 | x
---------------
 * | 0 | # | /

Enter the first number using numbers on the keypad
Press # to submit, or * to clear
15
enter
First Number = 15
Enter the operation using the keypad
Press * to clear
-
enter
Operation: -
Enter the second number using numbers on the keypad
Press # to submit, or * to clear
20
enter

Result = -5
```

Figure 3: Arithmetic calculator output of negative numbers



```
Keys of the Calculator
 1 | 2 | 3 | +
---------------
 4 | 5 | 6 | -
---------------
 7 | 8 | 9 | x
---------------
 * | 0 | # | /

Enter the first number using numbers on the keypad
Press # to submit, or * to clear
15
enter
First Number = 15
Enter the operation using the keypad
Press * to clear

clear
Resetting...

Keys of the Calculator
 1 | 2 | 3 | +
---------------
 4 | 5 | 6 | -
---------------
 7 | 8 | 9 | x
---------------
 * | 0 | # | /

Enter the first number using numbers on the keypad
Press # to submit, or * to clear
```

Figure 4: Arithmetic calculator output when using the reset button



```
Keys of the Calculator
 1 | 2 | 3 | +
--------------
 4 | 5 | 6 | -
--------------
 7 | 8 | 9 | x
--------------
 * | 0 | # | /

Enter the first number using numbers on the keypad
Press # to submit, or * to clear
74
enter
First Number = 74
Enter the operation using the keypad
Press * to clear
x
enter
Operation: x
Enter the second number using numbers on the keypad
Press # to submit, or * to clear
2
enter
Result = 148
```

Figure 5: Arithmetic calculator output of regular calculation

Conclusion

In conclusion, this project helped us improve our ability to use GPIO, UART, and SysTick. We were able to successfully make an arithmetic calculator using the keypad as the input and the UART to be able to receive data from it and transmit it onto the screen. Additionally, the use of SysTick helped us understand a possible software solution to switch bouncing, as adding delays between button presses allowed for proper debouncing.