5/1/2023

Mbed Project

ET095G: Introduction to Embedded System Programming

Saly Al Masri
Mid Sweden University

Content

Introduction

In this project I designed a simple problem-solving mathematical game, the purpose of the game is to enhance kids' talent in fast calculation. The game generates addition/subtraction/multiplication questions and displays them in the *LCD*. The user is required to use the *joysticks* to digitally input the answer. When the game starts a *timer* help to count down and before it reaches zero a red *LED* starts to periodically blink for three seconds. If the answer is correct, score goes up by one, else if the answer if incorrect, score goes down by one and in both cases the score is saved to a text document. The benefit of using a *file* to save data of the last score is to start from that level instead of counting from zero every time the user wants to restart the game.

Implementation

LCD

This is the first component that was added to the project, the component was connected to p5, p6, p7, p8, p11. The main purpose of the LCD is to show the game and interact with the player, as the problem, score and timer are printed in the screen. Using the "locate" function to control how the overall design of the game will look like. The library that was added to be able to use the LCD is the C12832.

Joystick

In order to control the values in the game, joysticks were used. The principle is to change the value of the answer by moving it in all directions. The pins are declared at the beginning of the project as DigitalIn.

Timer

Every problem has a certain amount of time to be completed else the player loses and the score decreases. Manage the time used, a timer is starts before the problem being generated, and as time increases, the time left decreases until it reaches zero. The formula used is as follows

$$time\_left = seconds - t.read()$$

Therefore, the function "read" was needed. When the problem is solved, the timer restarts counting. That occurs after the "reset" function is called at the end of the game.

PWM

Before the game ends and especially at the last three seconds, a red LED blinks three times to warn the player that the game will end. The RGB pins were assigned to the PWM in order to control them. The light was kept at an inactive state until time goes out. To change the colour of the LED, every pin was given a value starting from 0 until 1 and the colors in between are for different shades.

Timeout

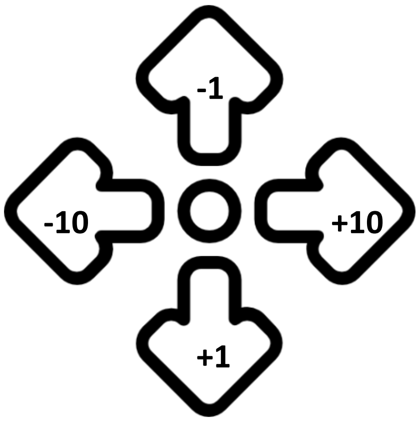
Timeout is used to distinguish when the RGB LED will be called. The blinking function was attached to the timeout with the time required to call the function. In this game, the total time is 25 seconds, so the time goes out at 22.

File

Adding a file to the game helps save the game's score for the player to start from where they ended in the previous game. In addition, the player can keep track of the score even after the game end. To implement this idea, a text document was created, and the score was and will be saved in it. To open the document two states were needed, read, and append. In the beginning, the read state is used to take the last value of the document, if it is the first time, then the score value will be zero. In the end, append state is needed to save value at the end without overwriting?? (is it needed). The functions used are: fseek, fopen, fgetc, fputc, fclose. As the values of the score are integers then "fgetc/fputc" are suitable.

How is everything connected?

When the game starts, a message shows up on the LCD to guide the player on how to start the game, which means it won't begin until an event occurs. If the centre button is pressed, the timer starts which gives the player a certain time to answer the question. To answer the questions, joysticks can be moved as shown below.



When players add the answer, they should wait until the time reaches zero for them to start the new problem. The answer will be shown on the LCD beside the time and score, so the player is provided every information needed. Before the end of the game, a red LED blinks for three seconds to warn the player. If the answer is wrong then the score decreases by one, and if it's correct, the answer increases by one. In both cases, a message shows up to the player to inform them that the score has changed and at the same time the score is saved to the data file. At this point, the answer is set equal to zero again, the timer restarts, and the LCD screen is cleared. If the player decides to end the game and start again some other time, they will start from the same score that was last saved in the data file.

Discussion

There are many constraints to the game I made, implementing it in the real world. One of them are price range, the microcontroller is expensive and lack of advanced creativity in the game would lead to a lower demand in the market which will not make the company eager to invest in it.

Another limitation is that the joystick, it has only a push button. When I tried it with my friend, a more gaming intellectual, he said that the joystick wasn't in par with the real-world joysticks like PlayStation's, more buttons to push and less waiting time for response (real-time constraint).

Finally, the limitation of memory, as it can't hold a lot of data will affect the performance of the microcontroller. To encounter the problem, I made it simpler by only saving the last data which in this game is the score and the program itself, leaving no pressure on the memory and the performance.