**Microcontroller Using C – Final Project**

Sakdipat Vijitvoravong and Dalton Crawford



In this project, we made a goal to recreate a game from a video and set of instructions. The game is a number guessing and memory game. The game materials include 1 button, 6 different LEDs, 7 seven segment display, and one decimal keypad.

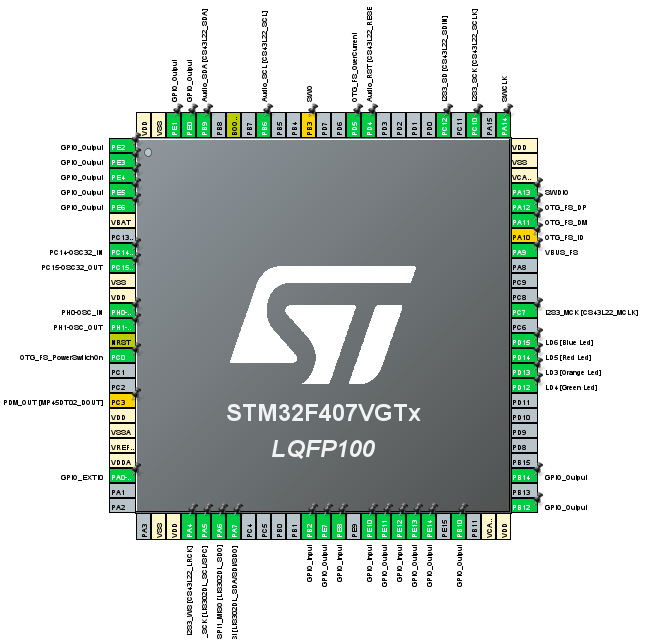
The game has a short light display to signal that it is on and working. Then the all the lights are off. This begins the guess phase. When you are ready, you can press the button and 4 LEDs in a loop on the board will begin blinking. They will continue to circle until you release the button. When you release the button, the light that was on will be selected and begin blinking. You can then use the keypad to guess numbers until you press the correct number. When you guess, the number will be displayed on the seven segment. If your guess is less than the random number, an external green led will light up. If your guess is greater than the random number, a red light will turn on. If you have guessed the correct number, the blinking light will turn on along with any of the other lights that you have guessed correctly. Then they will turn off and wait for you to press the button agin. Once you do, the circle of lights will again begin and you can guess again for any one you like. Once they are all guessed they will stay on. If you accidentally select a light that you have already guessed, it will be steady instead of blinking, and you can select a new light with the button.

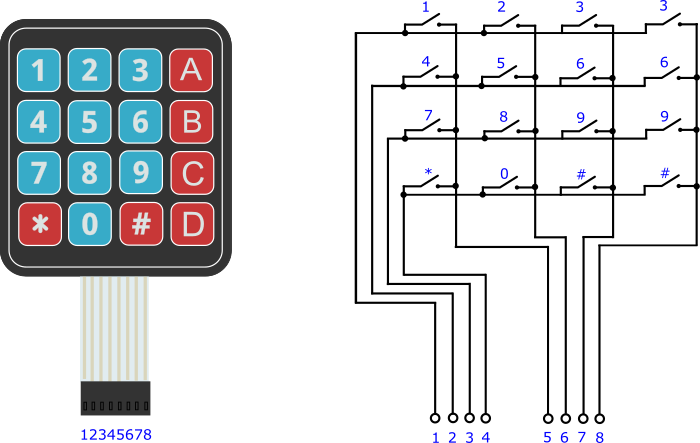
You can begin the memory phase by pressing the button after all of the numbers have been guessed the first time. You will only have one guess at each light. Once you press the button again, the green light will blink 3 times and then hold steady. You then input the number you guessed corresponding with the green onboard LED. Then the orange light will do the same, and you can input the number you remember through the keypad, then again with the red onboard LED and finally with the blue LED. Once you have guessed all of the lights correctly, the external LED’s will alternate blinking faster and faster until they are just on. If you did not, then no light party will happen. Either way, the lights will turn off and the intro light sequence will begin again and you can start a new game with brand new numbers.

The following images and acompaning discriptions of the code that we wrote and adjustments we made for the orpject to work.

Below is an image of the IOC. The IOC is a user interface that allows the user to select the pins directly and change their settings. With the help of IOC, assigning PINS to operate tasks is now easier than ever.

The pins were turn on for the external LED’s are RED LED at E13, and GREEN LED at E11. T he pins were turn on for the keypad are COLUMN3 at B2 for the letter column, COLUMN2 at E8 for the column 369#, COLUMN1 at E10 for the 2580 column, COLUMN0 at E12 for the 147\* column, ROW3 at E14 aligned with the letter D, ROW2 at B10 aligned with the letter C, ROW1 at B12 aligned with the letter B, and ROW0 at B14 aligned with the letter A. The pins were turn on for the seven segment are SEG\_A at E4, SEG\_B at E6, SEG\_C at E5, SEG\_D at E3, SEG\_E at E1, SEG\_F at E2, SEG\_G at E0, and SEG\_DP at E7.





Keypad:

Assigning the 4 pins in the row to be output to send power across the keypad. Each row will be turned on one at a time in a brief period because if all rows are on at once, the input would not be able to determine which button in the column is pressed.

The input pins () in column need to be set to pulldown because when the pins are not receiving any power, it should be pulled down to 0 instead of undermined.

Num Seg: Combination of previous functions Keypad and Numpad (not pictured) allows it to communicate together. Depending on the button pressed on the keypad, the seven segment will display accordingly and it will return the number pushed, or a number corresponding to a symbol or letter.

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Seven Segments:

There are 7 LEDs which could be controlled individually using 7 output pins and a voltage source. In order to turn on the LED, the output needs to be set to 0 to let the voltage flow through. There are 10 numbers that could be displayed from 0-9 depending on which pins are set to on or off.

Graphical user interface

Description automatically generatedGraphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, text

Description automatically generated

There are a total of 17 cases ranging from 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \*, #, A, B, C, D, and off

To turn display each desired character, the pins need to be set to set (0) or reset (1), which is reverse from the typical order. In other words, when the pin is reset, the LED will be on. For most of our code, SEVSEG (-1) is used to trigger the default case and turn all the lights off.

def\_delay: HAL\_Delay does not work in the interrupt due to it using the same clock. A new delay function needs to be implemented using for loop by multiplying input delay by 10,000 to replicate HAL\_Delay

Text

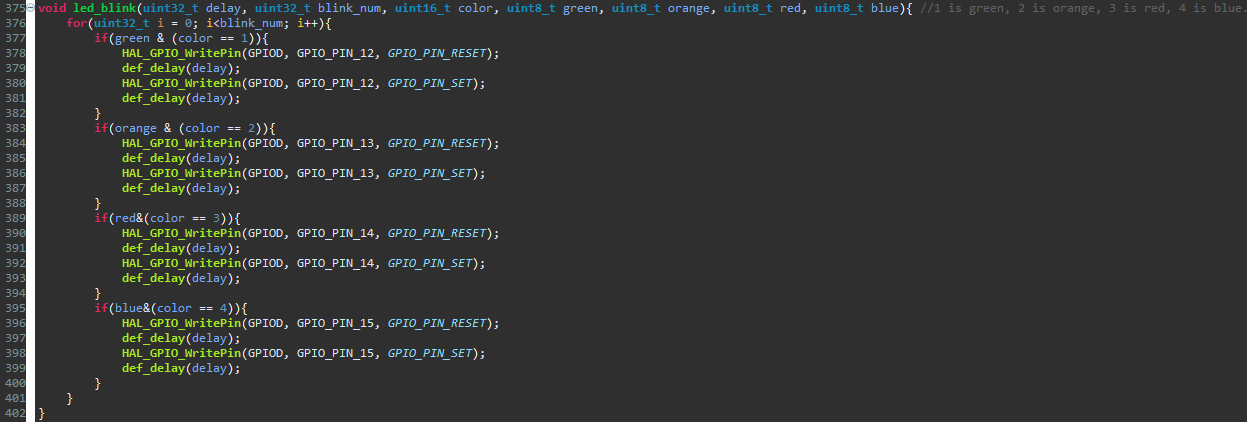
Description automatically generated

Light Checker: Inputting the 4 LEDS correct status, if the LEDs has already received the correct number, the light will be on to indicate that it is done.

Text

Description automatically generated

Led Blink: This blinks the LEDs. You have delay which uses the def\_delay function all throughout the function, the number of blinks you want, the color of LED you want to blink and then 4 enable variables. If the enable is true, and the color is selected to be 1, green light will blink N times where N is the blink\_num. All the other cases will be skipped. The enable variables are related to the score and prevent any accidental light signals from entering and disrupting the behavior of the lights.



Header File: To enable the functions to be used in the main, each function needs to be declared once in header for it to work. Below is the header file. The header file helps define ports in the code and declare functions. The pins for the keypad are in a block. The pins for the seven segments are all in the GPIOE, and the pins for the external LED’s are for simplicity as well.



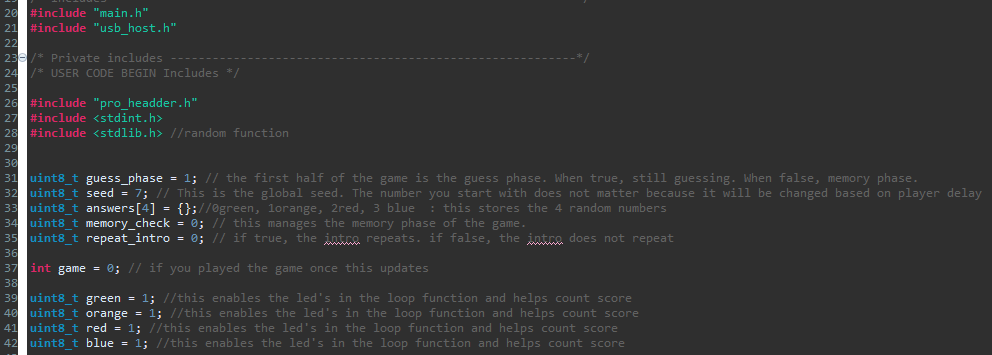
Intro: Alternating between Green with Red and Orange with Blue two times before turning off.

A screenshot of a computer

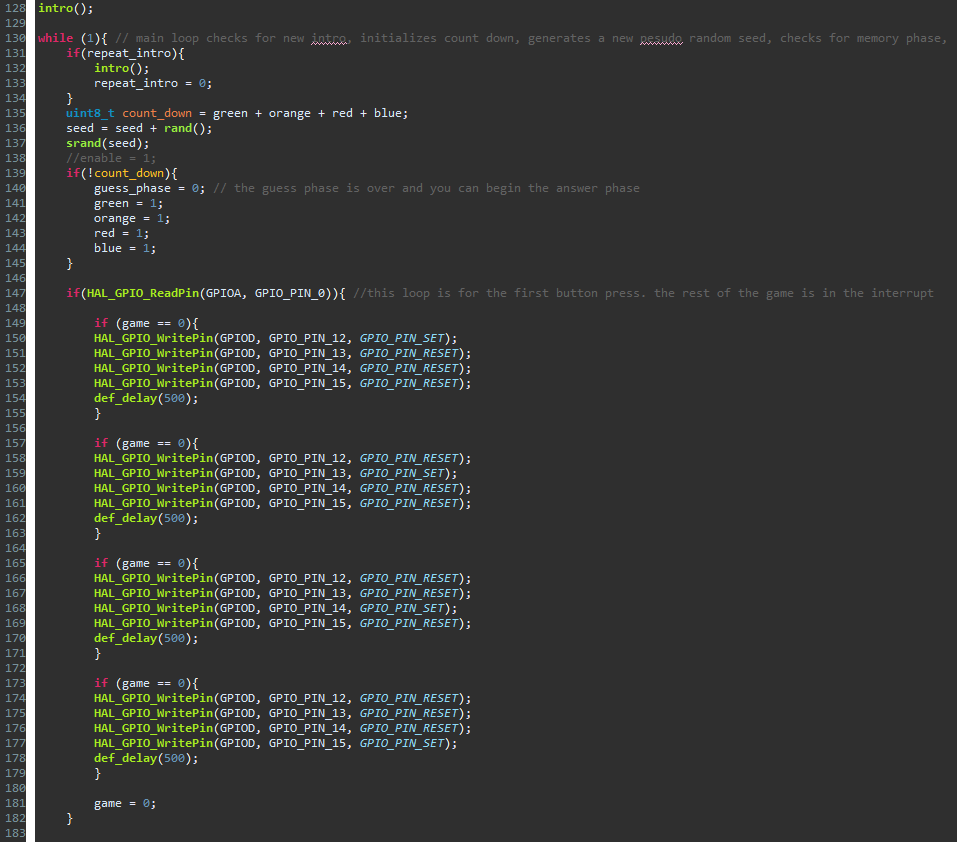
Description automatically generated with medium confidence

The Main Body:

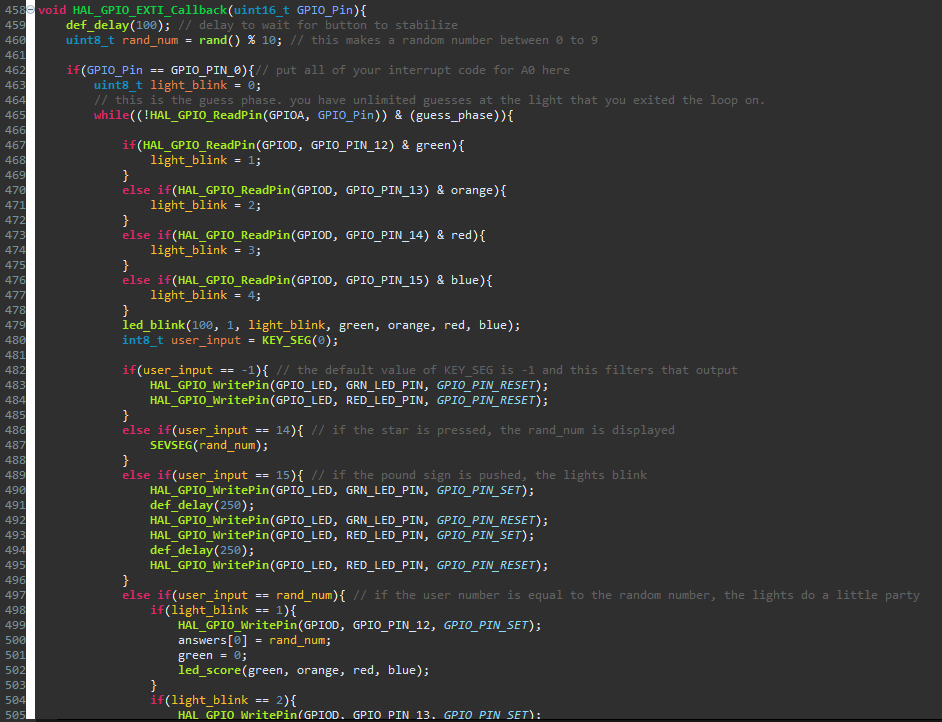
The main body of the code begins with inclusions and global variables. Our header is included as well as stdint for universal variable titles, and stdlib for the random functionality. The large list of global variables allows for wide communication between the main and the interrupt.



This is the main. It begins with the intro function, which blinks lights. The main while loop begins and checks with the repeat\_intro. repeat\_intro is set to zero in the declaration step and will skip on the first run. In subsequent runs, the switch will be turned on, and then turned off after the behavior is met. count\_down is declared. It is in the while loop because it needs to be updated after every guess. It is the sum of the light indicators, so no matter the order, once they are guessed, the count\_down will hit zero. The seed variable is added to itself and a random number. This random number will be the same at the beginning of every reset. Then srand creates a new literal seed. Next, if the count\_down is not true (or equals zero) then the guess\_phase is set to zero and is now over. The green, orange, red, and blue are reset. Finally, if the blue A0 button is pressed, the lights will loop. If the game has not been played, the lights will be on. If the game has been played once, the lights remaining in the loop will be set to off, so that the loop will always begin at green, like in the video. At the end of the loop, the game is again set to zero, meaning it has not been played again, allowing the lights to circle.

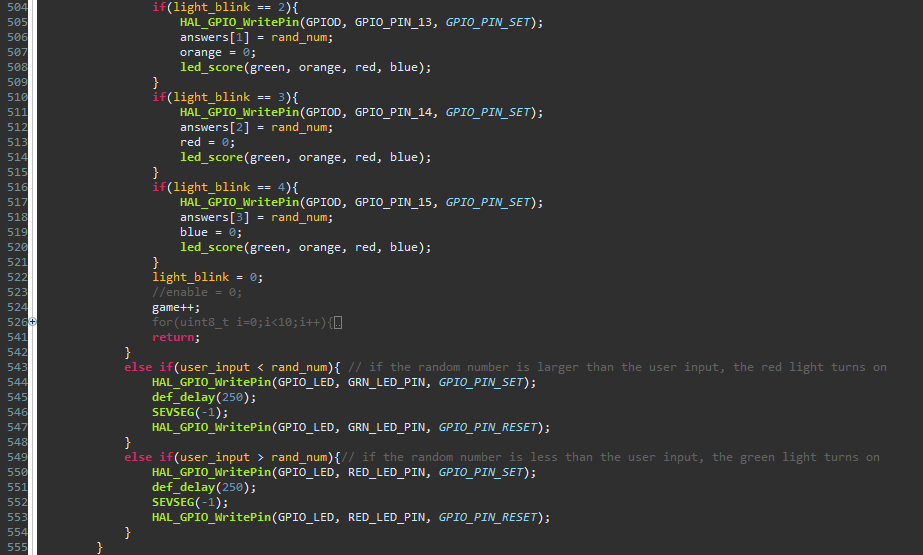


Once the button is released, the falling edge trigger alerts the interrupt function. The interrupt begins with a delay to allow the button to be fully off. Then the random number for this round is generated. If the interrupt is from pin zero, the game phases will begin. First the light blink is initialized and set to 0. While the button is not being pushed and the guess phase is still 1 or true, the guess phase begins. In the guess phase, you have unlimited guesses to find the right number behind the light selected. If the light selected is green, and the green is still 1 or true, light blink will be set to 1. Similar conditions are checked and if orange than 2, red than 3, and blue than 4. Line 479 operates the light blink function. 100 is for the approximate milliseconds of delay, 1 is for the number of blinks in a row, light\_blink indicates the color of light, and all the enable variables record which lights have already been blinked. Then line 480 is asking the user for the keypad to seven segment information. If there is no button pushed, the function outputs –1 and the seven segment is all off, otherwise, the function outputs the value being pressed at that time. If nothing is pressed, user\_input is –1 and the red and green lights are set to off. Else if user\_input is 14, the number paired with the asterisk symbol, the random number is displayed. If the input is equal to 15, the number paired with the pound symbol, the red and green light each blink once. This is just to test the hardware. Then if the user\_input is equal to the random number, lots of stuff happens. First if the green light was blinking, the green light is set on, the first index of the answers array is updated to the random number, the green global marker is set to 0 or false, and the led score function is displayed which turns all of the lights that have their global marker set to 0 or false.



First if the orange light was blinking, the orange light is set on, the first index of the answers array is updated to the random number, the orange global marker is set to 0 or false, and the led score function is displayed which turns all of the lights that have their global marker set to 0 or false. This is then done to red and blue. Light blink is again set to zero, and the game marker is set to 1 representing a single number guessed.

If the random number is greater than the user input, the red light is set on, and after some delay, the seven segment is turned off. Then the red led is set to off. If the random number is less than the user input, the green light is turned on, and after some delay, the seven segment is turned off. Then the green led is set to off.



If the button is not pressed and the guess\_phase is 0 or false, the memory\_phase begins. The memory phase is where you must remember all the guesses you made in the order green, orange, red, blue. Repeat intro is set to one, because whether you remember correctly, the game restarts. The for loop then checks each light in ascending order. The lights are all set on, then the led\_blink blinks with a delay of 200 for 3 separate times, blinking the light associated with the numbers 1-4. The seven segment is turned off, and the user\_input is set to KEY\_SEG (1). The 1 indicates that the loop is active, and the function will loop until a button is pushed. If the answer at index i-1 is equal to the user\_input, memory\_check increases by 1. If you get all four opportunities correct, memory\_check will equal 4. If memory\_check is equal to 4, then you get to see the light party. The external LEDs are alternatively blinked with decreasing delay until they appear to be on. They are then set on for about 250 milliseconds, then off. Regardless of success, the 4 on-board LEDs are turned off, the guess\_phase is re enabled the seed is again set to itself plus a random number. The answers array is set to all 11, because 11 is out of the scope of available answers. The remaining variables are set to the settings at the beginning of the game. There is no loop, so there is no return statement. The game begins again at the start of the while loop in the main, accept now that repeat intro is 1 or true, the intro repeats.

