1. Simplified Wordle game

1.1 Introduction

The program runs a simplified wordle game, comparing the user's guesses with a randomly selected word from a set list of 20 words: Mushy, Crazy, Acorn, Beans, Trunk, Clove, Cream, Spoon, Cider, Vodka, Table, Horse, Eight, Short, Water, Built, Lunch, Crust, Salad, Olive. The symbol chosen to display a wrong letter is '*', as underscores are hard to count when they are written in succession (e.g. ___), so asterisks seemed a more user-friendly choice. The user has six attempts to guess the word before being shown a message saying they have run out of attempts; if they guess the correct word then the program will display a win message and the number of tries guessed it in.

1.2 Design description

The program starts by displaying a title message, and randomly selecting a word from a list of 20 using a randomly generated seed based on the current time; the aim is to get as close to a truly random number as possible. The program then enters a while loop which will continue until the user runs out of guesses or guesses the correct word. The user is prompted to give a guess which is then saved. If the user input is not five characters long or does not consist entirely of letters, then the program will keep asking for a new input until those conditions have been satisfied. Once the program has saved a valid input from the user, it will convert it to lowercase for comparison with a randomly selected word from the word bank. If a letter guess is in the correct position, it will change an asterisk in the 'guess' variable to that letter in uppercase and save it in the correct position of the 'guess' variable. If a letter guess is present in the randomly selected word but not in the correct position, it will do the same thing but as lowercase instead of upper. If a letter in the user input is not present at all in the answer, then an asterisk will be saved in the 'guess' variable, as shown in this part of the code below:

```
     for (i = 0; i < 5; i++){ //for loop used for comparing user input
with randomly selected word

        if(guess[i] != ("%c", blank[i])){
            guess[i] = ("%c", blank[i]);
        }
           if(input[i] == word[i]){ //checks if the user input letter is
in the same position as the answer
              guess[i] = ("%c", input[i]); //if so the display variable
'guess' is changed to show that letter
              guess[i] = toupper(guess[i]);//as the letter is in the
correct position it is then coverted to uppercase
              } else { //if the letter is not in the correct position it
will then check to see if that letter is  present anywhere else in the
answer
                  for(j=0; j < 5; j++){
         if (input[i] == word[j]){ //if that letter does exist elsewhere
in the answer then it is written to the display and converted to
lowercase
              guess[i] = ("%c", input[i]);
```

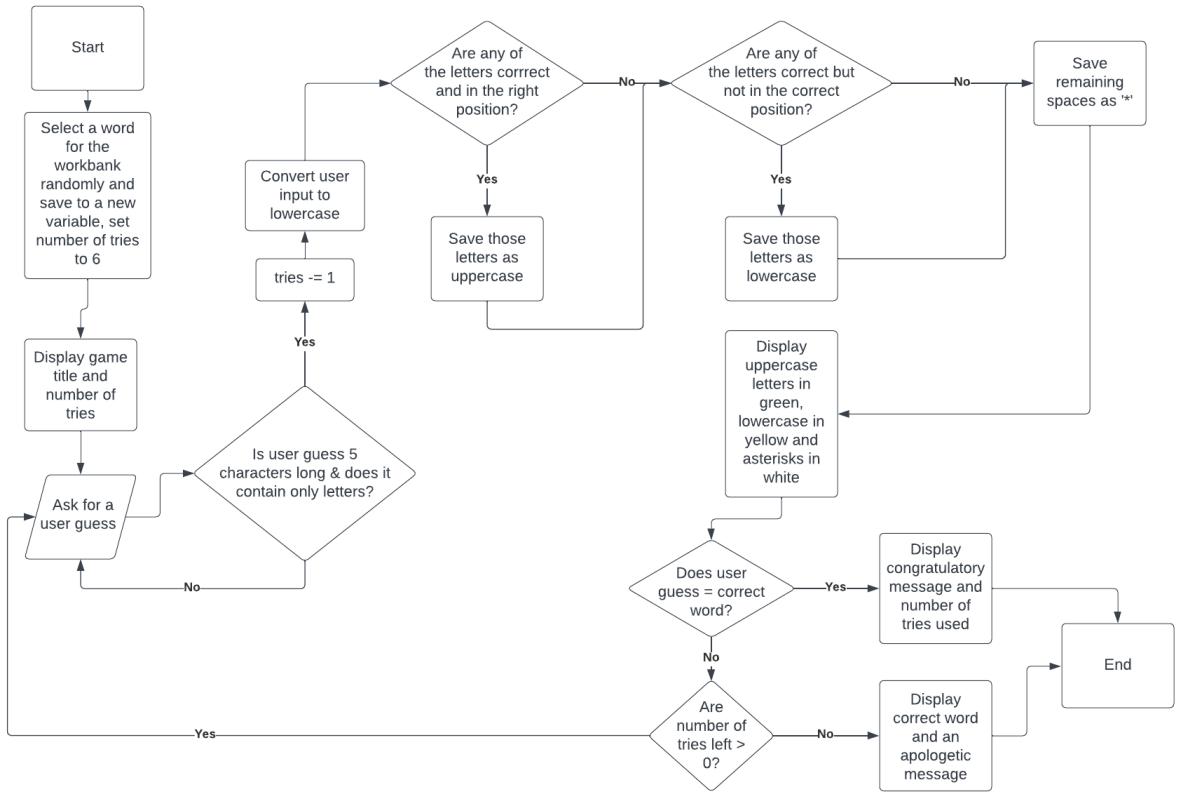
```
                guess[i] = tolower(guess[i]);
                }
        }
```

The guess variable will then be printed, which is where I have added the additional feature of changing the colour of the output to make it more like the real wordle game, green for correct position, yellow for correct letter but wrong position, and white for asterisks.

```
for(i = 0; i < 5; i++){
          if (isupper(guess[i]) != 0){ //if the letter is in the correct
position it will be displayed as green
              printf("\033[0;32m"); //changes font colour to green
              printf("%c",guess[i]); //prints letter
              printf("\033[0m"); //changes back to default font colour
          }
          else if(islower(guess[i]) != 0){ //if letter is present in the
answer but not in the right position then it will be displayed as yellow
              printf("\033[0;33m");
              printf("%c",guess[i]);
              printf("\033[0m");
          }
          else{ //otherwise an asterisk will be displayed
              printf("%c",guess[i]);
          }
      }
      printf("\n"); //new line
```

The program will then check if the user has guessed the correct word. If that is the case, then a win message will be displayed with the number of attempts and the while loop will be exited, ending the game. If the user runs out of attempts the while loop will also be exited displaying a message saying that they have lost and what the correct word was.

Flowchart:



1.3 Discussion

The program makes use of for and while loops to improve the efficiency and speed of the program. This could be improved further using functions for some parts of the program that are used often like the for loop for printing the guesses, the part of the program that compares the user input with the correct word and the while loop which checks for a valid user input. Despite this, the program is still correct as it achieves everything set out in the assignment brief. No extra variables are unnecessarily used, as variables are reused where possible (like 'i' in the various for loops) to avoid having memory used for storing unnecessary variables. I have also tried to make the program efficient by choosing the correct datatypes where possible to reduce the amount of memory put aside for each variable (i.e. using int instead of char). If else statements were also used as opposed to multiple if statements to improve the efficiency of the program.

1.4. Program code

```
#include <stdio.h> //built in header file
#include <stdlib.h> //library for random function
#include <time.h> //library for 'time'
#include <string.h> //library needed for checking string length
#include <ctype.h> //library for isalpha and tolower

int main()
{   //variable list
    char wordbank[][21] = {"mushy", "crazy", "acorn", "beans", "trunk",
"clove", "cream", "spoon", "cider", "vodka", "table", "horse", "eight",
"short", "water", "built", "lunch", "crust", "salad", "olive", '\0'};
//creates an array of 20 words
    int random; //creates a variable integer with the name 'random'
    char input[5]; //creates a variable used for storing the user input
    char blank[5] = "*****"; //creates a string with 5 asterisks
    int lettercheck; // used for validating the user input
    int tries = 6; //used for counting the number of tries
    char guess[5];//used for outputting the correct letters in the
user's guess
    int i; //i & j are used in for loops
    int j;
    int win = 0; //win condition will change to '1' when user has
guessed the correct word

    printf("**__Wordle__**\n"); //displays game title

     for(i=0; i < 5; i++){ //for loop writes '*****' to the output
variable 'guess' as at the start of game there has been no guesses yet
        guess[i] = ("%c", blank[i]);
    }

    srand(time(NULL)); //sets a random seed based on the current time
    random = (rand() % 20); //randomly assigns a value between 0 & 19 to
'random'
    char* word = wordbank[random]; //writes the randomly selected
variable to 'word' so it can be used further on in the code

while(win != 1 && tries != 0){ //this while loop will keep looping until
they either win the game (win == 1) or the number of tries left reaches
0

    printf("Enter a guess (5 letters) (tries left: %i)\n", tries);
//prompts user for an input and displays number of tries remaining
    scanf("%s",input); //stores user repsonse in the 'input' variable

    lettercheck = 0; //sets lettercheck to 0

     if (strlen(input) != 5){  //checks the length of the user input is
equal to 5
        lettercheck += 1;   // if length is not equal to 5 then
lettercheck will get altered
        }


        for (i = 0; i < 5; i++){ //runs a for loop to check that each
character inputted is a letter
            if (isalpha(input[i]) == 0){
```

```
                lettercheck += 1; //if not a letter lettercheck is
altered
            }
        }

   //if the user has entered a guess that is 5 characters in length and
all letters then it will skip this while loop below

    while(lettercheck  != 0){ //if the user has not made a valid guess
then this loop will run until they have made a valid guess
         printf("That was not five letters\n"); // informs user that
their guess was not valid
         printf("Enter a guess (5 letters) (tries left: %i)\n", tries);
//prompts user for an input and displays number of tries remaining
        scanf("%s",input); //stores user repsonse in the 'input'
variable

        lettercheck = 0; //checks the length of 'input' is equal to 5
        if (strlen(input) != 5){
        lettercheck += 1;
        }

        for (i = 0; i < 5; i++){ //checks all the characters in 'input'
are letters
            if (isalpha(input[i]) == 0){
                lettercheck += 1;
            }
        }
    }

    tries -= 1; //user has made a valid guess so can now subtract from
their number of attempts left

    for (i = 0; i < 5; i++){ //for loop converts user input to all
lowercase to be used to compare to the randomly selected word
              input[i]= tolower(input[i]);
            }

     for (i = 0; i < 5; i++){ //for loop used for comparing user input
with randomly selected word

        if(guess[i] != ("%c", blank[i])){
            guess[i] = ("%c", blank[i]);
        }
           if(input[i] == word[i]){ //checks if the user input letter is
in the same position as the answer
              guess[i] = ("%c", input[i]); //if so the display variable
'guess' is changed to show that letter
              guess[i] = toupper(guess[i]);//as the letter is in the
correct position it is then coverted to uppercase
              } else { //if the letter is not in the correct position it
will then check to see if that letter is  present anywhere else in the
answer
                 for(j=0; j < 5; j++){
         if (input[i] == word[j]){ //if that letter does exist elsewhere
in the answer then it is written to the display and converted to
lowercase
              guess[i] = ("%c", input[i]);
              guess[i] = tolower(guess[i]);
```

```
                }
            }
            }
        }
         //for loop for displaying the 'guess' variable, showing the user
what letters were guessed correctly
         for(i = 0; i < 5; i++){
             if (isupper(guess[i]) != 0){ //if the letter is in the correct
position it will be displayed as green
                 printf("\033[0;32m"); //changes font colour to green
                 printf("%c",guess[i]); //prints letter
                 printf("\033[0m"); //changes back to default font colour
             }
             else if(islower(guess[i]) != 0){ //if letter is present in the
answer but not in the right position then it will be displayed as yellow
                 printf("\033[0;33m");
                 printf("%c",guess[i]);
                 printf("\033[0m");
             }
             else{ //otherwise an asterisk will be displayed
                 printf("%c",guess[i]);
             }
         }
         printf("\n"); //new line

         if(strcmp(input, word) == 0){ //checks if the user has guessed the
correct word
           win=1; //changes win condition to 1 to break out of the while
loop
           int t = 6 - tries;
           printf("Correct! The word was %s - Guessed in %i attempt(s).\n",
word, t); //diplays a win message and number of tries remaining
           }
    }


if (tries == 0 && win == 0){ //checks if the user has run out of tries
and not won the game
    printf("You ran out of tries, the word was %s.\n", word); //displays
error message along with the correct answer
}
    return 0;
}
```

2. Simple lights game

2.1 Introduction

The aim of this part of the assignment was to implement a simple game involving LED lights and a servo motor on the ES2C4 development board. The last digit of my student ID is '9' which is 1001 in binary so therefore at the beginning of the game the red and orange lights should be lit. The second last digit in my student ID is '0' so s=A_{16} which makes T1 between 0 and 26 seconds. The third last digit is also '0' which makes T2 between 0 and 10 seconds. The fourth last digit is 8 so T3 is just set to 8 seconds.
Here is the peripheral Configuration below (the pin for Green Button had to be changed to PB_0 as PB_6 was affected by PA_6 being connected to the motor):

Peripheral	Configuration
Red LED	PB_1
Yellow LED	PB_5
Green LED	PA_11
Orange LED	PA_8
PWM (servo motor)	PA_6
Red Button	PA_9
Orange Button	PA_5
Green Button	PB_0

The program makes use of a for loop function for any delays, Systick for setting the frequency of the flashing lights, and TIM16 for the PWM signal for the servo motor.

2.2 Design description

The program starts by defining the three time delay functions for T1, T2 & T3,
e.g T3:

```
void T3 (void) //time delay function for T3
{
int i; //sets i as an integer variable
for (i=0; i<(8*250000); i++){} //for loop will run 8 seconds
}
```

Then the code generates two random variables for the T1 & T2 functions:

```
n1 = rand() % 27; //generates a random number between 0-26 for T1 time delay
      n2 = rand() % 11; //generates a random number between 0 & 10 for T2 time
delay
```

The program then enables the clock for GPIO ports A & B as well as the peripherals so the interrupts can function. After setting the interrupts for red and orange buttons specifically, I put in the majority of the code that set up the pins for the LEDs and buttons, clearing the appropriate register and either setting them to input or output before setting to pull down mode so their default state is 'off'. The LEDs could then be turned on or off by changing them to low or high frequency, e.g. `GPIOB -> ODR |= (0x1UL << 1);` would turn on the Red LED and resetting the same pin to 0 would turn it off again. I have made use of left shifts to ensure

the correct bits are written to the appropriate pins without overwriting other pins that have already been set.

After this, the program enters the main while loop for the game which starts out by setting the win count to 0 which will increase every time the user gets a correct orange button press within the correct time frame. The first if statement checks for a green button press, at which point the game will start by turning on the red and orange LEDs, running the T1 function and then turning off the lights again. If the 'win' variable is below three, it will then enter another while loop where SysTick will be used to set the frequency (a LOAD value of 50000 is equivalent to 5Hz), at which point the yellow LED will blink for T2 seconds using the following code:

```
for (i=0; i<(n2*250000); i++){ //for loop blinks light for T2 seconds
                    if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) //check
if systick is 'on'
                        {
                         GPIOB->ODR ^= (0x01UL << 5); //toggles Yellow LED to
make it blink
                                                                          }
            }
```

If an orange button press has been detected during the time T2 using the following interrupt, then the win count will be increased and the motor barrier will be raised.

```
void EXTI9_5_IRQHandler(void){ //interrupt function - gets triggered
whenever a button is pressed
      if(EXTI -> PR1 & (0x1UL <<9)){ //checks for a red button press
            EXTI-> PR1 |= (0X1UL << 9); //clears EXTI status flag
            NVIC_SystemReset(); //resets the program and lights
      }
      if(EXTI -> PR1 & (0x1UL <<5)){ //checks for an orange button press
                  EXTI-> PR1 |= (0X1UL << 5);//clears EXTI status flag
                  orng = 1; //sets orange button input variable to one


            }
}
```

When the red button is pressed, the interrupt detects this and will reset the program immediately using the `NVIC_SystemReset();` command.

To raise the motor, the following code is used to assign it to the appropriate registers and setup TIM16:

```
RCC -> APB2ENR |= RCC_APB2ENR_TIM16EN; //enables Timer 16 clock in the
RCC_APB2ENR register
                              GPIOA -> MODER |= (0x1UL <<13); //sets PA_6 to
alternate function mode
                               GPIOA -> AFR[0] &= ~GPIO_AFRL_AFSEL6_Msk;
//clears AFR register
                               GPIOA -> AFR[0] |= (0xEUL <<24); //sets AFR
register to alternate function mode
                               TIM16 -> PSC = 4000 - 1; //sets timer
(TIM2_PSC) 16 prescaler to 4000
                               TIM16 -> ARR = 19; //sets auto-reload
(TIM16_ARR) value to 20
                               TIM16 -> CNT = 0; //count starts at 0
```

```
                                TIM16 -> CCMR1 &= ~0x7FUL; //configures
channel 1 as the output in the TIM16_CCMR
                                TIM16 -> CCMR1 |= (0x6UL << 4); //sets the
output mode to PWM mode 1
                                TIM16 -> CCER |= 0x1UL; //enables capture
compare for channel 1 in the TIM16_CCER register
                               // In the TIMx_BDTR register (this is not
required for Timer 2 and the basic
                               //timers), set the MOE bit.
                                TIM16 -> BDTR |= (0x1UL <<15); //sets the MOE
bit in the TIMx_BDTR register
                               // In the TIM2_CR1 register enable the timer.
                                TIM16 -> CCR1 =1; //raises the barrier
                                TIM16 -> CR1 |= 0x1UL; // enables the timer in
the TIM2_CR1 register
```

The code then runs the T3 function before returning the barrier to its original position and reducing T2 by 25% of its current value.

When the user reaches three wins, the program sets up the registers for the green LED (MODER & PUPDR), and changes the frequency for SysTick which will then blink at a rate of 2Hz. Then, all the lights will blink for three seconds to indicate the game has been won, using this code:

```
for (i=0; i<(3*400000); i++){ //for loop for lights to blink for 3 seconds
                                 if (SysTick->CTRL &
SysTick_CTRL_COUNTFLAG_Msk) //check if systick is 'on'
                                    {
                                     GPIOB->ODR ^= (0x01UL << 5); //Toggles
Yellow LED
                                     GPIOB -> ODR ^= (0x1UL << 1); //Toggles
Red LED
                                     GPIOA -> ODR ^= (0x1UL << 8);  //Toggles
Orange LED
                                     GPIOA -> ODR ^= (0x1UL << 11); //Toggles
Green LED
                                    }
                            }
                             GPIOB->ODR &= (0x0UL << 5); //turns Yellow LED off
                             GPIOB->ODR &= (0x0UL << 1); //turns Red LED off
                             GPIOA -> ODR &= (0x0UL << 8);  //turns Orange LED
off
                             GPIOA -> ODR &= (0x0UL << 11); //turns Green LED
off

        }
                            }
```

2.3 Discussion

The program almost fulfils the brief except for having to change the green button pin from PB_11 to PB_0, as when the wire from PA_6 to the motor was connected on the board, the button functions as if always being held down. To work around this, I changed the pin for the green button and after that it worked as intended. The program would be more efficient if I made use of a timer like TIM2 for the delays instead of for loops and would have been something I would like to have added if I had more time.

I used while and for loops where necessary to reduce the number of lines of code needed and improve the efficiency. The design is relatively efficient, only having enough code required to define how the game works. From there, it keeps repeating parts of the code until the game is won or the red button is pressed.

If I had more time, the main thing I would focus on would be finding a way to avoid having to change PB_11 to PB_0 for the green button.

2.4. Program code

```
#include "stm32l4xx.h" //library for microcontroller commands
#include "main.h"
#include <stdio.h> //built-in header file
#include <stdlib.h> //library used for random function 'rand()'
#include <stdint.h> //used for PMW motor signal
int orng; //creates integer variable for saving orange button press
int win;//creates integer variable for counting how many times the orange
button has been correctly pressed
int n1; //creates integer variable for storing a random number used in T1
function
int n2;//creates integer variable for storing a random number used in T2
function

void T1 (void) //time delay function for T1
{
int i; //sets i as an integer variable
for (i=0; i<(n1*250000); i++){} //for loop will run for n1 number of seconds
therefore creating the desired delay
}

void T2 (void)//time delay function for T2
{
 int i; //sets i as an integer variable
 for (i=0; i<(n2*250000); i++){} //for loop will run for n2 number of
seconds therefore creating the desired delay
}

void T3 (void) //time delay function for T3
{
int i; //sets i as an integer variable
for (i=0; i<(8*250000); i++){} //for loop will run 8 seconds
}

int main(void)
{
      n1 = rand() % 27; //generates a random number between 0-26 for T1 time
delay
```

```
      n2 = rand() % 11; //generates a random number between 0 & 10 for T2
time delay

      RCC -> AHB2ENR |= 0x3UL; //enables clock for GPIO ports A & B

      //enables peripherials for use of interrupts
       RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN; // enables the SYSCFG
peripheral
       RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN; // enables the GPIOA peripheral
       RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN; // enables the GPIOB peripheral

      //sets up the interupt for the red button
      SYSCFG -> EXTICR[3] &= ~(0xFUL << 3); //sets SYSCFG to connect the
button EXTI line to GPIOA
      EXTI->IMR1 |= (0x1UL << 9); // Set EXTI interrupts for falling input
on the button pin
      EXTI->FTSR1 &= ~(0x1UL << 9); // Disable the falling edge' trigger
(button release)
      EXTI -> RTSR1 |= (0x1UL << 9); // Enable the 'rising edge' trigger
(button press)

      //ORANGE INTERRUPT
      SYSCFG -> EXTICR[2] &= ~(0xFUL << 3); //sets SYSCFG to connect the
button EXTI line to GPIOA
      EXTI->IMR1 |= (0x1UL << 5); // Set EXTI interrupts for falling input
on the button pin
      EXTI->FTSR1 &= ~(0x1UL << 5); // Disable the falling edge' trigger
(button release)
      EXTI -> RTSR1 |= (0x1UL << 5); // Enable the 'rising edge' trigger
(button press)

      NVIC_SetPriority(EXTI9_5_IRQn, 0x03); //enables the NVIC interrupt at
minimum priority value
      NVIC_EnableIRQ(EXTI9_5_IRQn);

      //PA outputs (Green & Orange LEDs)
       GPIOA -> MODER &= ~(0xFUL << 20); // clear pins to 0000
       GPIOA -> MODER |= (0x4UL << 20); //configure PA_11 to output mode
[GREEN LED]
       GPIOA -> MODER &= ~(0xFUL << 16); // clear pins to 0000
       GPIOA -> MODER |= (0x1UL << 16); //configure PA_8 to output mode
[ORANGE LED]

       GPIOA -> PUPDR &= ~(0xFUL << 20 ); //clear pins to 0000
       GPIOA -> PUPDR |= (0x8UL << 20); // set PA_11 to pull down mode
[GREEN LED]
       GPIOA -> PUPDR &= ~(0xFUL << 16); //clear pins to 0000
       GPIOA -> PUPDR |= (0x2UL << 16); // set PA_8 to pull down mode
[ORANGE LED]

   //PB outputs (Red & Yellow LEDs)
       GPIOB -> MODER &= ~(0xFUL << 8); // clear pins to 0000
       GPIOB -> MODER |= (0x4UL << 8); //configure PB_5 to output mode
[YELLOW LED]
       GPIOB -> MODER &= ~(0xFUL); // clear pins to 0000
     GPIOB -> MODER |= (0x4UL); //configure PB_1 to output mode [RED LED]

       GPIOB -> PUPDR &= ~(0xFUL << 8); //clear pins to 0000
       GPIOB -> PUPDR |= (0x4UL << 8); // set PB_5 to pull down mode [YELLOW
LED]
       GPIOB -> PUPDR &= ~(0xFUL); //clear pins to 0000
     GPIOB -> PUPDR |= (0x8UL); // set PB_1 to pull down mode [RED LED]
```

```
    //PA inputs (Red & Orange Buttons)
    GPIOA -> MODER &= ~(0xFUL << 11); //clear pins in the MODER register to
0000 (input mode)
    GPIOA -> PUPDR &= ~(0xFUL << 11); //clear pins in the PUPDR register to
0000
    GPIOA -> PUPDR |= (0X4UL << 11 ); //set PA_5 to pull-down mode [ORANGE
BUTTON]

    GPIOA -> MODER &= ~(0xFUL << 19); //clear pins in the MODER register to
0000 (input mode)
    GPIOA -> PUPDR &= ~(0xFUL << 19); //clear pins in the PUPDR register to
0000
    GPIOA -> PUPDR |= (0X1UL << 19); //set PA_9 to pull-down mode [RED
BUTTON]

   //PB inputs (Green Button)
     GPIOB -> MODER &= ~(0x3UL); //clear pins in the MODER register to 0000
(input mode)
     GPIOB -> PUPDR &= ~(0xFUL); //clear pins in the PUPDR register to 0000
     GPIOB -> PUPDR |= (0x1UL << 1);//set PB_0 to pull-down mode  [GREEN
BUTTON]


     while(1) //while loop will run for the duration of the program
     {
        win = 0; //sets the win condition to zero
                          //every time the user gets a correct orange button
press the value of win will increase by 1

       if (GPIOB -> IDR & (0x1UL)) //Check for green button press
        {                          //start sequence
             GPIOB -> ODR |= (0x1UL << 1); //set PB_1 [Red LED] to high
frequency (on)
             GPIOA -> ODR |= (0x1UL << 8);  //set PA_8 [Orange LED] to high
frequency (on)
             T1(); // wait for T1 seconds
             GPIOB -> ODR &= ~(0xFUL); //sets PB_1 to [Red LED] low
frequency (turns it off)
             GPIOA -> ODR &= ~(0xFUL << 8); //sets PB_8 to [Orange LED] low
frequency (turns it off)

             while(win != 3) // while loop will keep looping until the user
has got three correct button guesses
             {
             //Systick Setup
                   SysTick->LOAD = 50000 -1; //sets the length of the
systick (blink rate of 5Hz)
                   SysTick->VAL = 0; //the value of the systick counter will
start from 0
                   SysTick->CTRL |=0X3UL; //Set the clock source and enable
Systick;

                   orng = 0; //resets orange button press to zero

                   //code to make the yellow led flash
                   int i; //sets i as an integer variable

            for (i=0; i<(n2*250000); i++){ //for loop blinks light for T2
seconds
                   if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) //check
if systick is 'on'
```

```
                        {
                         GPIOB->ODR ^= (0x01UL << 5); //toggles Yellow LED to
make it blink
                                                                          }
             }
              GPIOB->ODR &= (0x0UL << 5); //turns yellow LED off

              if (orng == 1) { //checks if the orange button has been pressed
during this time
                    win += 1; //if the orange button has been pressed then
the value of 'win' is increased by 1

                    // Code for PWM signal to set up the Motor

                                RCC -> APB2ENR |= RCC_APB2ENR_TIM16EN;
//enables Timer 16 clock in the RCC_APB2ENR register
                                GPIOA -> MODER |= (0x1UL <<13); //sets PA_6 to
alternate function mode
                                 GPIOA -> AFR[0] &= ~GPIO_AFRL_AFSEL6_Msk;
//clears AFR register
                                 GPIOA -> AFR[0] |= (0xEUL <<24); //sets AFR
register to alternate function mode
                                 TIM16 -> PSC = 4000 - 1; //sets timer
(TIM2_PSC) 16 prescaler to 4000
                                 TIM16 -> ARR = 19; //sets auto-reload
(TIM16_ARR) value to 20
                                 TIM16 -> CNT = 0; //count starts at 0
                                 TIM16 -> CCMR1 &= ~0x7FUL; //configures
channel 1 as the output in the TIM16_CCMR
                                 TIM16 -> CCMR1 |= (0x6UL << 4); //sets the
output mode to PWM mode 1
                                 TIM16 -> CCER |= 0x1UL; //enables capture
compare for channel 1 in the TIM16_CCER register
                                // In the TIMx_BDTR register (this is not
required for Timer 2 and the basic
                                //timers), set the MOE bit.
                                 TIM16 -> BDTR |= (0x1UL <<15); //sets the MOE
bit in the TIMx_BDTR register
                                // In the TIM2_CR1 register enable the timer.
                                 TIM16 -> CCR1 =1; //raises the barrier
                                 TIM16 -> CR1 |= 0x1UL; // enables the timer in
the TIM2_CR1 register

                    T3(); //runs the T3 function (8 second delay)

                    TIM16 -> CCR1 = 2; //resets the position of the motor

                    n2 = n2 - (0.25*n2); //reduces the amount of time for T2
by 25% after each correct guess
              }
              orng = 0; //resets orange input variable
              }

              GPIOA -> MODER &= ~(0xFUL << 20); // clear pins to 0000
              GPIOA -> MODER |= (0x4UL << 20); //configure PA_11 to output
mode [GREEN LED]
              GPIOA -> PUPDR &= ~(0xFUL << 20 ); //clear pins to 0000
              GPIOA -> PUPDR |= (0x8UL << 20); // set PA_11 [GREEN LED] to
pull down mode

                                        SysTick->LOAD = 125000 -1; //sets the
length of the systick (blink rate of 2Hz)
```

```
                                SysTick->VAL = 0; //the value of the counter
should start from 0
                                int i; //sets i as an integer variable

                            for (i=0; i<(3*400000); i++){ //for loop for lights
to blink for 3 seconds
                                if (SysTick->CTRL &
SysTick_CTRL_COUNTFLAG_Msk) //check if systick is 'on'
                                    {
                                     GPIOB->ODR ^= (0x01UL << 5); //Toggles
Yellow LED
                                     GPIOB -> ODR ^= (0x1UL << 1); //Toggles
Red LED
                                     GPIOA -> ODR ^= (0x1UL << 8);  //Toggles
Orange LED
                                     GPIOA -> ODR ^= (0x1UL << 11); //Toggles
Green LED
                                    }
                            }
                             GPIOB->ODR &= (0x0UL << 5); //turns Yellow LED off
                             GPIOB->ODR &= (0x0UL << 1); //turns Red LED off
                             GPIOA -> ODR &= (0x0UL << 8);  //turns Orange LED
off
                             GPIOA -> ODR &= (0x0UL << 11); //turns Green LED
off

        }
                            }

             }


void EXTI9_5_IRQHandler(void){ //interrupt function - gets triggered
whenever a button is pressed
      if(EXTI -> PR1 & (0x1UL <<9)){ //checks for a red button press
            EXTI-> PR1 |= (0X1UL << 9); //clears EXTI status flag
            NVIC_SystemReset(); //resets the program and lights
      }
      if(EXTI -> PR1 & (0x1UL <<5)){ //checks for an orange button press
                  EXTI-> PR1 |= (0X1UL << 5);//clears EXTI status flag
                  orng = 1; //sets orange button input variable to one


            }
}
```