# ELEC/COMP 317: EMBEDDED SYSTEMS

# Project Report

2021 SPRING

***Project Members:*** MERYEM KARAKAS, MEHMET ALI SIMSEK

## *DEFUSE THE BOMB*

## INTRODUCTION AND GAMEPLAY

In this project, we designed a game that represents the defusal of a bomb mechanism. Here is the working procedure of the game. When the leftmost button of the PORTD is pressed, the program assigns a random 4-letter password for the mechanism. The defusal process consists of two parts. In the first part, the player enters some letters on the EZTerminal, to find the words included in the bomb password. For each correct attempt, the buzzer of the board buzzes the morse code of the written word. Otherwise, a denial voice (3-sec beep noise) is buzzed by the buzzer. The player has 90 seconds to complete the first part of the game. If s/he finds all four letters, the board automatically assigns the player to part 2. Otherwise, the buzzer buzzes continuously to represent the bomb explosion.

In part 2, the countdown will stop, and the password letters will be written on the seven-segment display in the order of their discoveries. The task of the player is to put these letters in the correct order. To do that, s/he uses the buttons of the PORTB. By exchanging the order of the neighboring letters, the player tries to find the correct password. S/he will have five attempts for this task. If victorious, the morse code of the password will buzz to represent the bomb defusal; if not, the continuous buzz will be active to represent the bomb explosion.

## PROGRAM OVERVIEW AND CODE EXPLANATION

When the user pushes the button PD7, interrupts are enabled, and depending on the pushing time, 4-letters password are generated randomly by the **generate_pw** subroutine. tmp2 is incremented until the user presses the button PD7, and we are using tmp2 to generate password, so the password is created randomly by the pushing time. If the user finds the letters of the password, the 4-letter password will be displayed on the seven-segment after the **PHASE1**, so there are limited number of letters that can be displayed on the seven-segment display. There are 20 letters that can be displayed, and we encoded them with the numbers:

| | | | | | | |
|---|---|---|---|---|---|---|
| 1-A | 2-b | 3-C | 4-d | 5-E | 6-F | 7-g |
| 8-H | 9-I | 10-J | 11-L | 12-n | 13-o | 14-P |
| 15-q | 16-r | 17-S | 18-t | 19-U | 0(20)-Y | |

After the user pushes the button, the remaining time to find the letters is displayed on the seven-segment display. We used Timer 0 to simulate the time counter. The user has 90 seconds to find the letters of the password. To find the letter, the user should connect the board with a computer. Then, s/he should press the letters by the keyboard to learn whether the pushed letter is in the password or not. When the user pushes a letter, the ASCII code of the letter is converted to the corresponding number in our encoding system by the **CONVERT_ASCII** subroutine. After the conversion is completed, the **CONTROL_INPUT** subroutine is called to control whether the given letter is in the password. If the pushed letter is in the password, the morse code of the letter is buzzed from the buzzer, otherwise, the buzzer buzzes for a while to indicate the given letter is not in the password. If the entered letter is found once in the password, the morse code is buzzed only on the first keyboard press, if it is found twice, it will be buzzed again when the same letter is pressed a second time, so the user can understand how many times a letter is repeated in the password. The user should find the 4 letters of the password before time is up. Otherwise, the code calls the **BOM** subroutine which enables the buzzer and enters an infinite loop, simulating the explosion of the bomb. Also, we used Timer 1 to generate a sine wave, there is a **SIN TABLE** that has 128 samples at the end of the code, with the numbers in the table, a sine wave is generated.

If the user finds all letters of the password, **PHASE1** is completed and **PHASE2** of the deactivation is started. In the second phase, Timer 0 and USART are disabled. The user should find the correct combination of the letters that s/he found. The 4-letters are displayed on the seven-segment in the order that the user found. To change the location of the letters, there are 4 buttons on the PORT B. PB7 swaps the locations of the first two letters, PB6 swaps the locations of the second and third letters, PB5 swaps the locations of the third and fourth letters, and PB4 swaps the locations of the first and fourth letters. The user can swap letters by using these buttons. After deciding the positions of the letters, the user should push PB0 to see the password is correct or not, s/he has 5 chances to try the password. If the locations of the letters are correct when the user pushes the PB0, the morse code of the password buzzes, and the **FINISH** subroutine is called which displays "done" on the seven-segment display. Otherwise, the buzzer buzzes for a while to indicate the password is not correct. If the user cannot find the correct combination in 5 trials, the bomb explodes, and the code calls the **BOM** subroutine which simulates the explosion.

## CONCLUSION

In conclusion, in this project, we implemented a game on the AVRBoard using several interfaces that we studied throughout the course. Subroutines, interrupts, timer interrupts, random number generation, sine-wave generation (from the lookup table), morse code generation, and UART communication are the significant concepts that we implemented in our project. Even though we experience some difficulties when dealing with the implementation of the code (especially allocation of the limited number of registers was a challanging task) we overcame these issues and completed the project. Therefore, we further improved our understanding of the course subjects and produced a fun game as our final project.

## SOURCE CODE

Since our code is too long (it takes 45 pages in the Word), we are not adding it to here, but we are submitting the asm file together with this report.