Project 3: Target Bowling

By: Rohan J. Dani

Intro to Microcontroller Interfacing

Project Time Frame: April 8th – April 30nd

Report Summary

In this report, I'll cover what this game we created does and my adventure through creating this game. It will also cover the output of the analog to digital converter control output I received along with how the TimerA was used for the Buzzer to make sound along with what duty cycle was used. Along with the bonus features I have included to further enhance the game.

Project Description

In this project, a bowling game was implemented which allowed to store the score and see what the high score is. The bowling game is able to move the ball left/right, change the speed of the ball, and change the angle of the ball thrown. These features are controlled by the joystick and throwing of the ball is done by pressing down the joystick. The pins are placed in random positions so goal is to maneuver the ball's throw to hit the pin's location. In this project, we're not allowed to use the inbuilt random function, so another method of randomness has to be chosen. The high score and how to play screens were implemented as well, so the user can easily see how the game is played and what the highest scores are.

The bonus features implemented are that whenever the pin is hit – a splash blurb appears saying that the pin has been hit. However, if the pin is missed then the opposite is said! Also, the gutter has been implemented, so if the ball hits the line then the ball disappears and a 0 score is given. Another feature that was added is whenever the ball is hit, a song is played to congratulate the user for hitting the pin!

Sound Generation

The buzzer sensor onboard the booster pack was used to play certain notes then the pin is hit. The blocking version of the song was implemented so if the pin were hit then a specific note would be played. The notes were given a value based off the note and the length multiplied by the duration of the note that should be played. This was all driven by the PWM, which set the frequencies of the different notes. The compare register and the output mode were chosen based off the specifications of the booster pack. Then the clock source for TimerA was set which lets the system know what clock source is used and the system clock was set as default. The timer period as part of the PWM configuration is found by dividing the multiplication of the tone frequency of each note by the clock source divider. The duty cycle was chosen based on the duty cycle of the note and it was chosen by multiplying the timer period by 0.5 accordingly. This means that the duty cycle, which is in terms of the counter cycles, has 50% duty cycle.

Using Joystick

The joystick was used very intelligently as the ADC drove most of the logic that was used as part of this game. For example, the input received from the buttons or the movement of the joystick would be used to either throw the bowling ball, move the ball left and right, move the arrow for the speed of the ball up and down, etc. For when I had to move something back forth I had to develop a FSM to use along with the input from the ADC. What would be done is

that four Boolean values would be set to false then in the first set of if and else if statements I would set each to false which is setting up the reset for the FSM. Two of the bools would be regarding movement one way and the other two would be for movement the other way. Afterwards, in the else of the first set of if and else I would set the variables to true depending on specific movement of the joystick. As seen in the picture say if the joystick is moved all the way to the left then the x and y values of the joystick will be changed and based off that the Boolean of whether the joystick is going one way can be set. That is ultimately how the logic is driven based off the behavior of the buttons and the joystick.

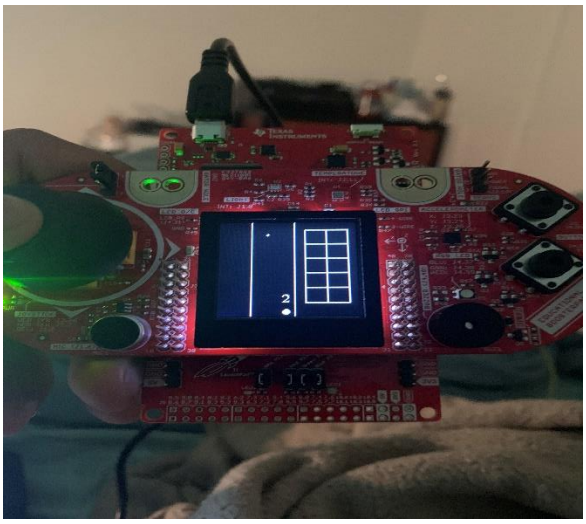

Figure 1: Game

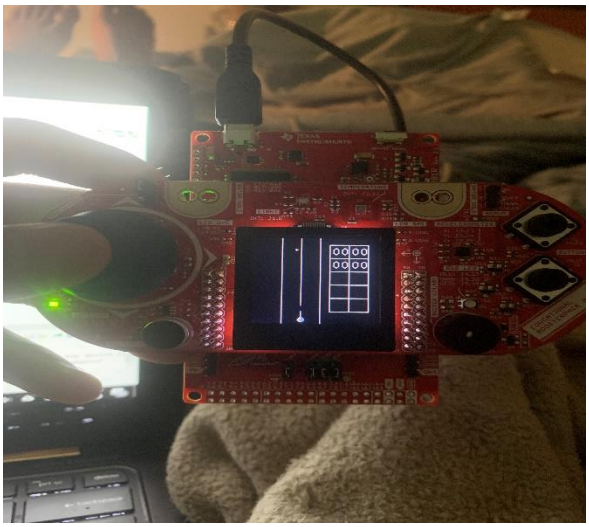

Figure 2: Game

State Machine

```
if (onLeft)
{
    joyStickPushedtoLeft = false;
    if (!joyStickLeft(vx))
    {
        onLeft = false;
    }
}
else if (onRight)
{
    joyStickPushedtoRight = false;
    if (!joyStickRight(vx))
    {
        onRight = false;
    }
}
else
{
    if (joyStickLeft(vx))
    {
        joyStickPushedtoLeft = true;
        onLeft = true;
    }
    else if (joyStickRight(vx))
    {
        joyStickPushedtoRight = true;
        onRight = true;
    }
```

Figure 3: FSM

This was the overall structure of the FSM that was used for each control movement that required to go back and forth. In the else case, that is where the movement is set to true based off the ADC of the joystick and the logic was written accordingly if they were given a true value.

Conclusion

I learned a lot from this project as it wrapped together all the techniques we learned from the homework from the debouncing, HAL functions, structs, advanced FSM techniques, ADC, sound generation, etc. This will definitely help me in the future as I am becoming more and more familiar with this microcontroller.