Christine Mayer (clm293) and Carly Swetz (cms464)
ECE 3140 Final Project Report

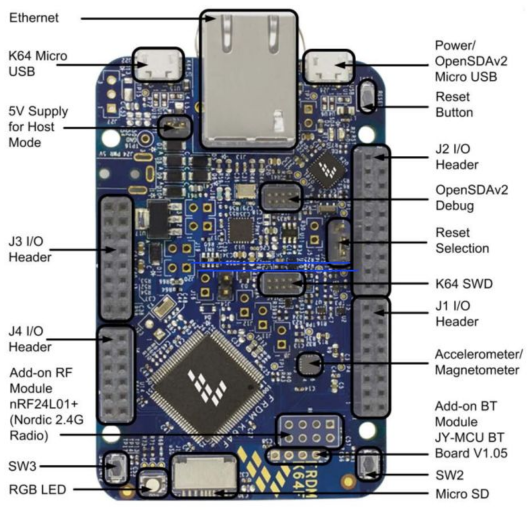
# Reaction Time Game

Demo Link: https://youtu.be/mRaO8si9Ox8

## Introduction

Our final project is a two-player reaction time game that uses two buttons and the LED. The game begins when the RESET button is pressed. Before each round, the green LED flashes three times to signal to the players that the white LED is coming. After the green LED blinks three times, the white LED will come on in anywhere from 10 to 50 tenths of a second (1 to 5 seconds). The first player to press their respective button wins that round and their color will flash. Another round then starts by first blinking green three times. The game will continue until a player has won three rounds. We also added a false start feature to our game. If a player is too eager and presses their button before the white LED comes on, they will lose that round and give the other player a point. We assume that a player will not false start during the three green flashes. As soon as a player has won the game, their color LED will stay on.

## System Diagram



The system uses the built-in switches on the FRDM-K64F board, indicated here by the labels SW2 and SW3. Additionally, it uses the RGB LED on the board.

We use three interrupts in our code. The first interrupt is PIT0 channel and is a timer for when the LED should come on. Our other two interrupts are for the player buttons, which are SW2 and SW3.

## Hardware Description

We did not use any external hardware other than the NXP FRDM-K64F board.

## Detailed Software Description

Nearly all of the code in our project was written by the group, with the only exceptions being utils.c and utils.h which were given to us for the other labs. We used the functions in the given filles to control the LEDs for our game.

The structure of our code involves an initial setup function and then relies on interrupts for the execution of the game. Throughout the code, we are careful to implement interrupts correctly, using interrupt handlers and remembering to clear the interrupt flags.

Setup initializes the LEDs, enables the two switches, selects GPIO for the switches, sets them as inputs, and connects them to interrupts. Then, it sets up the timer interrupt that turns on the reaction light. Finally, it calls start().

The start function first sets the boolean GameOn to 1, which prevents a new game from starting while there is already one being played. It flashes the green LED three times to indicate to the players to get ready to react and assigns a random time between 1.0 and 5.0 seconds to the timer.

Next, the timer interrupt occurs, and the interrupt handler turns all LEDs on for a white light.

An interrupt will come from whichever switch was pressed first; their interrupt handlers are symmetric. If GameOn is 1, meaning the game is active, we first turn all LEDs off. We then check the indicator variable for false start, described below. If no false start occurred, we increase the player's score and flash their color LED to indicate a point.

If one of the switches was pressed before the timer interrupt turned the white light on, a false start occurred. Crucially, we are careful to clear the timer when GameOn goes to 0 after a button is pressed, as there would be time leftover that would still trigger the timer interrupt. Then we simply give the opposite player a point and indicate with their LED.

At the end of the switch interrupt handlers, we check if the game is over (a player has reached 3 points), and if so turn on their LED to indicate a best of 5 victory. Otherwise, start() is called again from the main function after GameOn goes to 0.

## Testing

The only way to really test our reaction-time game was to actually play it. To make sure that almost every combination of ways the game could pan out, we played the game many times with different results. This included games that included no false starts, games with only false starts, and games with several different combinations of the two. In addition, to see if the game was understandable to someone other than us, we had our friends play several rounds, after only familiarizing them with the board and giving them the instructions. After countless games performing as expected, we had convinced ourselves that our implementation is correct.

## Results and Challenges

We did not achieve what we had proposed. Our original plan was to create a heart rate monitor that could be used for patients or athletes. We purchased the Grove Ear-clip Heart Rate Sensor to identify a pulse. Figuring out how the peripheral device communicated with our board was certainly the most complicated part of our design. The code to calculate the heart rate was relatively easy to understand once going through it a couple times. With the device, we first attached the power and ground wires to the board, respectively. The third wire coming from the device was our signal. For every heart pulse, the signal went high. We figured this out by using an oscilloscope and a voltmeter. However, for some reason, an interrupt would not occur on the high signal. In order for our code to calculate the heart rate, we needed these interrupts to happen. We tried various things to try to understand why this was happening, including making sure we correctly set up the GPIO, using external power, and trying to use pull up and pull down resistors.

After spending many (12+) hours at office hours without figuring out the problem, we had to make the tough decision to use the rest of our time wisely and start a whole new project. We opted to do a simpler project because of the endless issues we had with our previous one. In addition to the short time window remaining constrained us to a project with no other peripheral device. To make the game more interesting, we had the LED come on at a random time and added the false start feature.

Next time, the one thing I would do differently is try to find a peripheral device that communicated relatively simply with the FRDM-K64F board. Other than that, we really tried to get our original project to work, but we are happy with what our final project turned out to be.

## Work Distribution

To communicate about logistics throughout the lab, we communicated via text, and simply sent the one main file back and forth via email after working since we did not have too many different code updates.  We wrote all the code for the final project together using the pair programming technique, and additionally attended office hours together.  We each contributed to writing the sections of the preproposal, proposal, and report remotely on a Google Doc.

## References, Software Reuse

We wrote our main file *reaction.c*. The other two files in our project *utils.c* and *utils.h* were provided in previous labs.