STEMSEL LCD Project 5 : Reaction Test

Problem
We want to design a reaction game using the STEMSEL kit.

Background
Games are a good way to practice programming skills, since they can be either simple or complex, and will always have a specific set of rules that the programmer must think about. You may even be able to improve the game by adding new rules in order to make it more challenging, or to stop people cheating.

One important skill to have when playing many games is a fast reaction time. When your senses detect something, they need to send a signal to the brain, which must then decide what to do, and send an appropriate signal out to do something. For example, if you are playing cricket or baseball and the batter hits the ball near you, your eyes will see the ball coming and send a signal to your brain. The brain must decide the best way to move in order to get to where the ball will go, and then send a signal out to your arms and legs to make you run, jump or bend over. The less time that whole process takes, the faster your reaction time is.

Actually, microchips work in the same way: they use sensors like the thermistor or LDRs to detect something, which sends a signal to the microchip that makes a decision and sends another signal to outputs like a LED or motor.

The faster your reaction time is, the quicker you can respond, which makes it more likely you will catch a ball or start a little earlier in a race. In this project we will make a reaction game that will test your reaction speed.

Ideas
What do we need for our reaction game? A button is needed for the player, but how does the player know when to press it? What can we use as the trigger for the player that needs the player to wait for? Should we have another output to tell the player to get ready? How long should the delay be before the light turns on? Should this be a fixed delay, or would a random delay be better?

Plan
A LCD will be used to tell the player to get ready and also record the reaction time. We can use a LED and a buzzer to tell the player to push the button. In this case, the buzzer and the LED should be activated at the same time and the push-button will be used to turn them off. The program will record how long the LED and the buzzer have been on and display the time through the LCD. In conclusion, we will use the push-button as an input and the LED, Buzzer and the LCD as outputs as shown below.

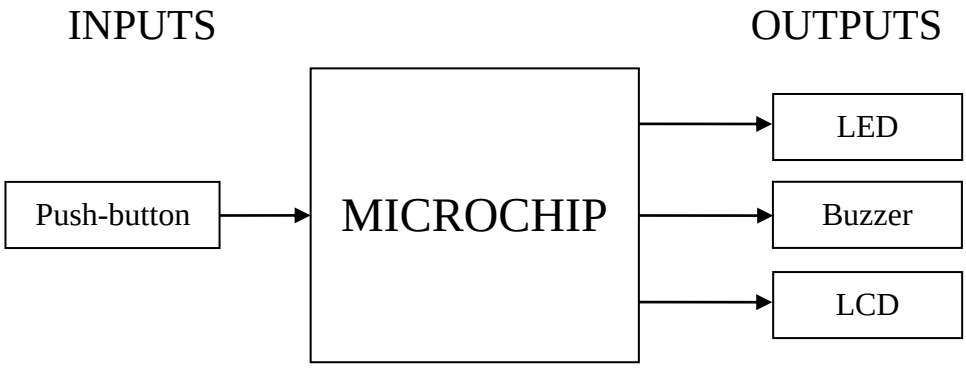

Figure 1: Inputs and Outputs

Design

Open the ezCircuit Designer and begin a new STEMSEL project and then add each of the components from the plan into the design. For the pushbutton, we can use the A3_pushbutton built onto the STEMSEL Controller board. All the components can be found in the circuit group.

The completed design should appear as shown below:

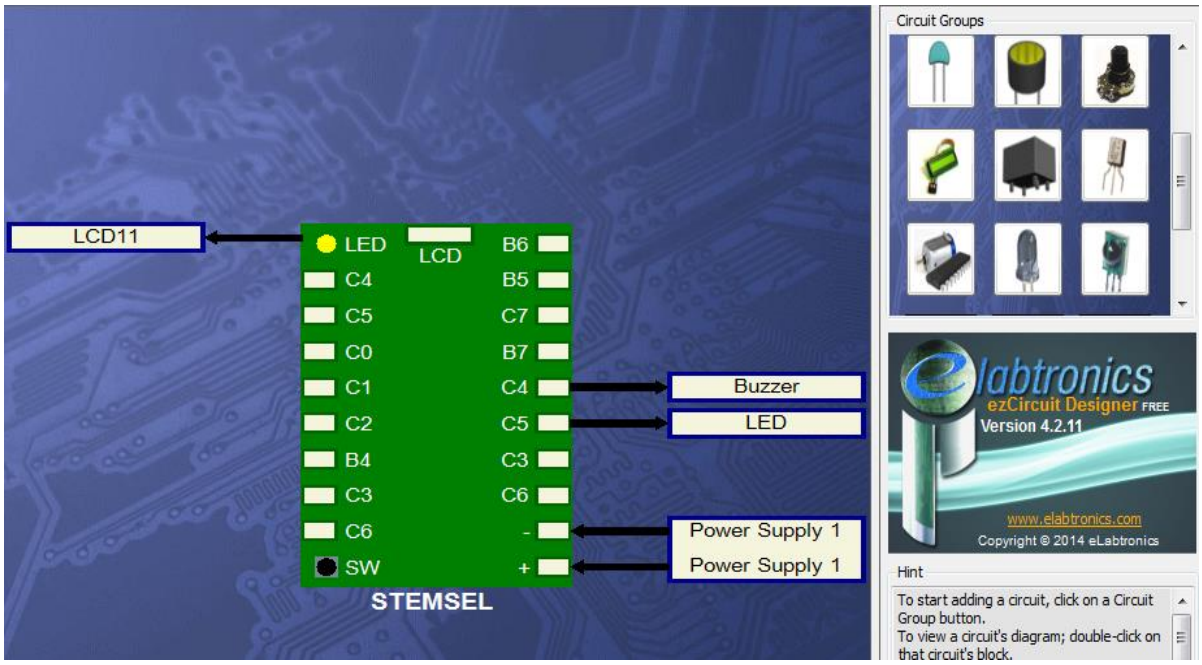

Figure 2: Circuit Design

Build

Use the ezCircuit Designer I/O diagram to connect the hardware. Make sure that black wires go into the negative terminal, red wires go into the positive terminal, and white wires go into the pin designated in the design.

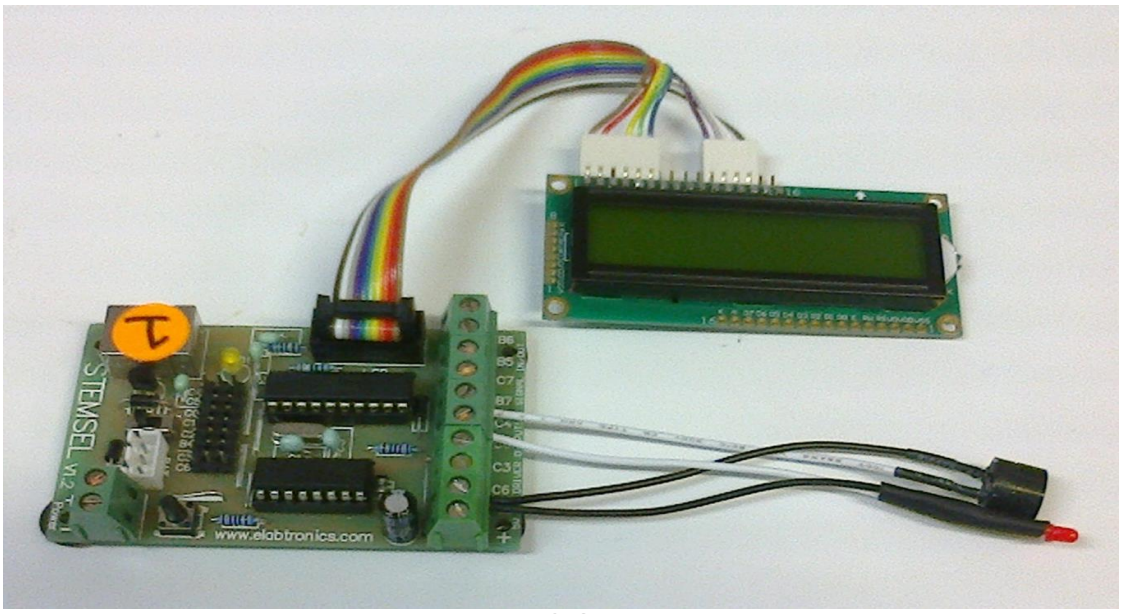

Figure 3: Circuit

Programming

With the design and assembly completed, send the design to CoreChart to program the microchip. Make sure the button is pushed and the controller is switched off when connecting the STEMSEL controller board to your computer. If the program is compiled successfully, plug the batteries in. Before programming, the STEMSEL controller board and the components should be tested using the default testing program.

1. Use the buzzer and the LCD to tell the player to get ready. Use an OnOffPin icon to turn the buzzer on, and an LCDMessage icon to display the message "Get Ready" on the LCD.
2. We will give the player 3 seconds to ready themselves, so add a TimeDelay icon and set it for 3 seconds.
3. At the end of the delay, the reaction test should trigger. Use two OnOffPin icons to turn the buzzer off and the LED on. We can also update the LCD by using an LDCFormat icon to clear the display with no delay, then using a new LCDMessage icon to display "GO!!!".
4. We will need to count how long it takes for the player to press their button. Use a SetNumber icon to set the number 0 and save it as "Count".
5. Next, add an address called "Button_Check", which will be the top of our button checking loop, and another address called "Button_Press" that will be the end of our checking loop.
6. Immediately after the Button_Check address, place a decision icon to check if the A3_Pushbutton is off (i.e. if the button is pressed). If so, add a GOTO icon to go to the Button_Press address.

7. If the button has not been pressed yet, we need to increment our timer. Use an Add icon to add 1 to our Count variable and save the result back as Count, then a time delay of one hundredth of a second.

8. Place another GOTO icon between the time delay and the Button_Press address, using it to go back to the Button_Check address. The icons placed in Steps 7 and 8 will allow us to count how many hundredths of a second it takes for the player to press their button.
9. Now we need to display this time on the LCD. Place a new LCDFormat icon after the Button_Press address to clear the LCD with no delay, then an LCDMessage icon to show the message "Your time".
10. Display the Count variable using a DispNumberLCD icon, then place another TimeDelay icon to display the time for 5 seconds.
11. Finally, clear the LCD once more with no delay using an LCDFormat icon, turn the LED off using an OnOffPin icon, then a GOTO icon to go back to START. Send your program to the chip and test it.

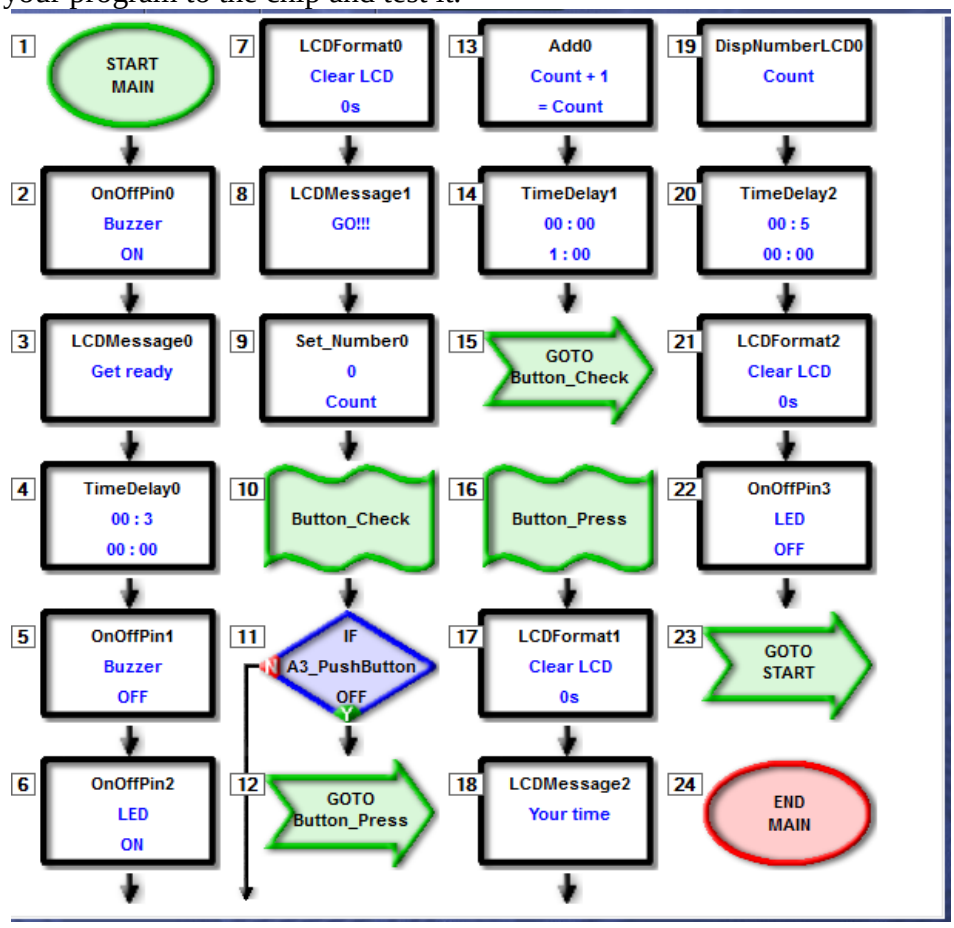

Figure 4: Code

Right now, our get ready delay is always 3 seconds, which means players can cheat by simply counting out the 3 seconds. We will now make this a random delay of between 2 and 4.5 seconds.

1. We will need to use an icon from the advanced mode. Click the Change Mode button up near the top of the CoreChart window. Just click OK on the warning that appears.

Figure 5: the Change Mode button

2. Click on the yellow icon labelled 'A+B=C'. In the Icon Properties section, click the Add New button to add a new variable.
3. In the Variables window that appears, type Random into the name window, and click Add Variable. This will create a new variable called Random.

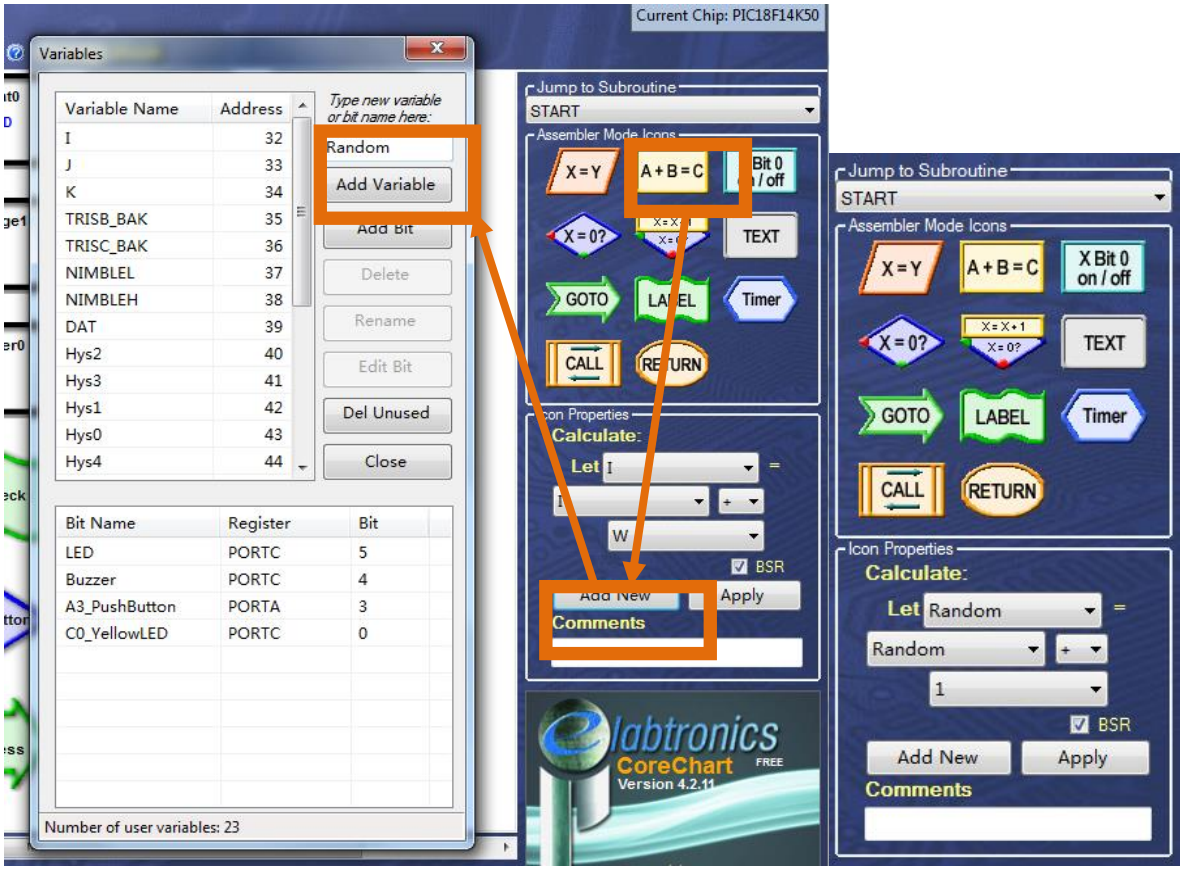

Figure 6: Creating the "Random" variable

4. Click close on the Variables window, and use the icon properties section to set our new icon to Let Random = Random + 1. Then, add the new icon at the very start of our program.
5. Click the Change Mode button to return to normal mode. After the Random = Random+1 icon we just added, place a new Decision icon to check if the A3_Pushbutton is on (i.e. button is not pressed) and if so go back to START using a GOTO icon.
6. These two icons will cause our Random variable to constantly count up until it reaches 255 then reset back to zero then start counting up all over again, a bit like a number wheel. This 'wheel' will spin around thousands of times per second, so the

player will have no idea what value the Random variable is. The player now also needs to press their button once to start the game.

7. Now we need to use this Random variable for our 'get ready' delay. Open up the 3 second time delay we added earlier, and change it to 2 seconds and select "Random" in the hundredths. This will give us our delay of between 2 and 4.55 seconds (Can you tell why?).
8. Send the new program to the chip and test it. Don't forget, you need to press the button once to start the game. Can you notice the random time delay?

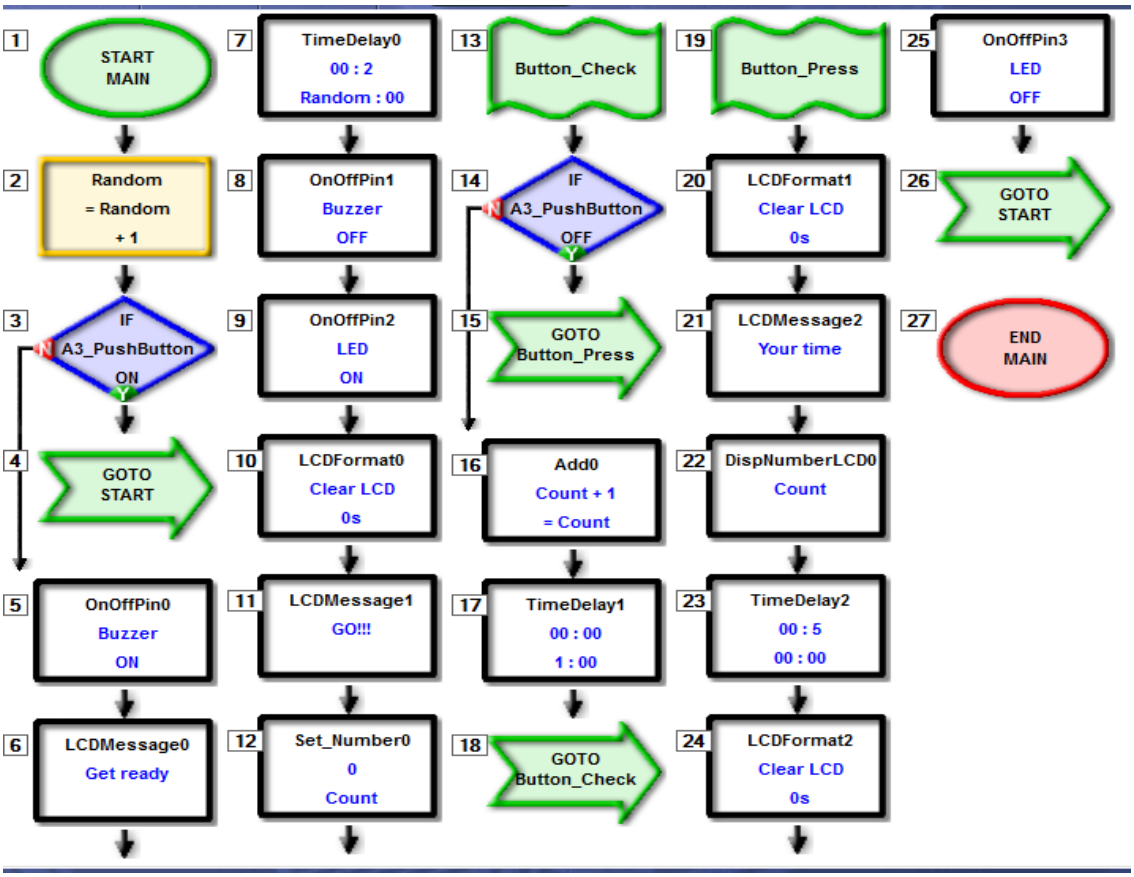

Figure 7: Program with the random time delay

Extension

At the moment, it is possible to cheat by holding the button down. How can we stop players doing this? (Hint: the button should not be pressed just before the buzzer turns off and the LED turns on).

Summary

A fast reaction speed is a good thing to have, not just for games but in everyday life as well, since it allows us to respond to things happening around us faster. Games are a good way to practice programming skills, since they have rules that you need to think about and incorporate in the program. By building this reaction test game, we have been able to combine these two aspects by programming a game we can use to test and improve our reactions.