UNIVERSITY OF Cincinnati

Lab 1 - Reaction Timer Game

Presented by:
Kody Kloth, Colton Zeinner

Class section: 001
Due Date: 2/9/19

EECE 4038C Embedded Systems Design

1. Objective

The objective of lab 1 is to create a reaction time game for 2 players. The game is played by releasing a press button when an LED turns from red to green. The BASIC Stamp Homework Board will be used to implement the code written using Basic stamp editor.

2. Procedure

Items Used:
(1) LED – bicolor
(1) Resistor – 470 Ω (yellow-violet-brown)
(2) Pushbutton – normally open
(2) Resistor – 10 kΩ (brown-black-orange)
(2) Resistor – 220 Ω (red-red-brown)
(4) Jumper wires

The procedure for this lab starts with setting the Stamp board using the Stamp Manual. Once the board is set up, the circuit is built as shown is **Figure 1** for one player. A second player is added by using another push button circuit (220Ω, 10KΩ, and push button) and connecting the output to a different output pin.

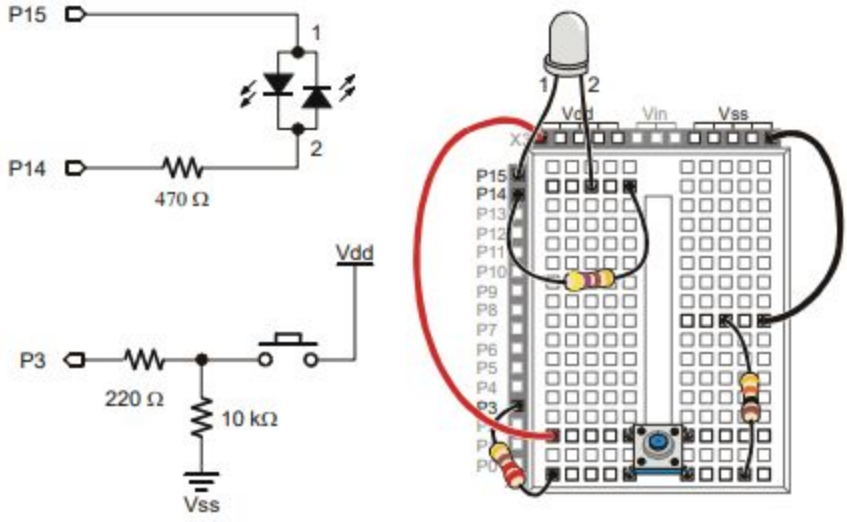
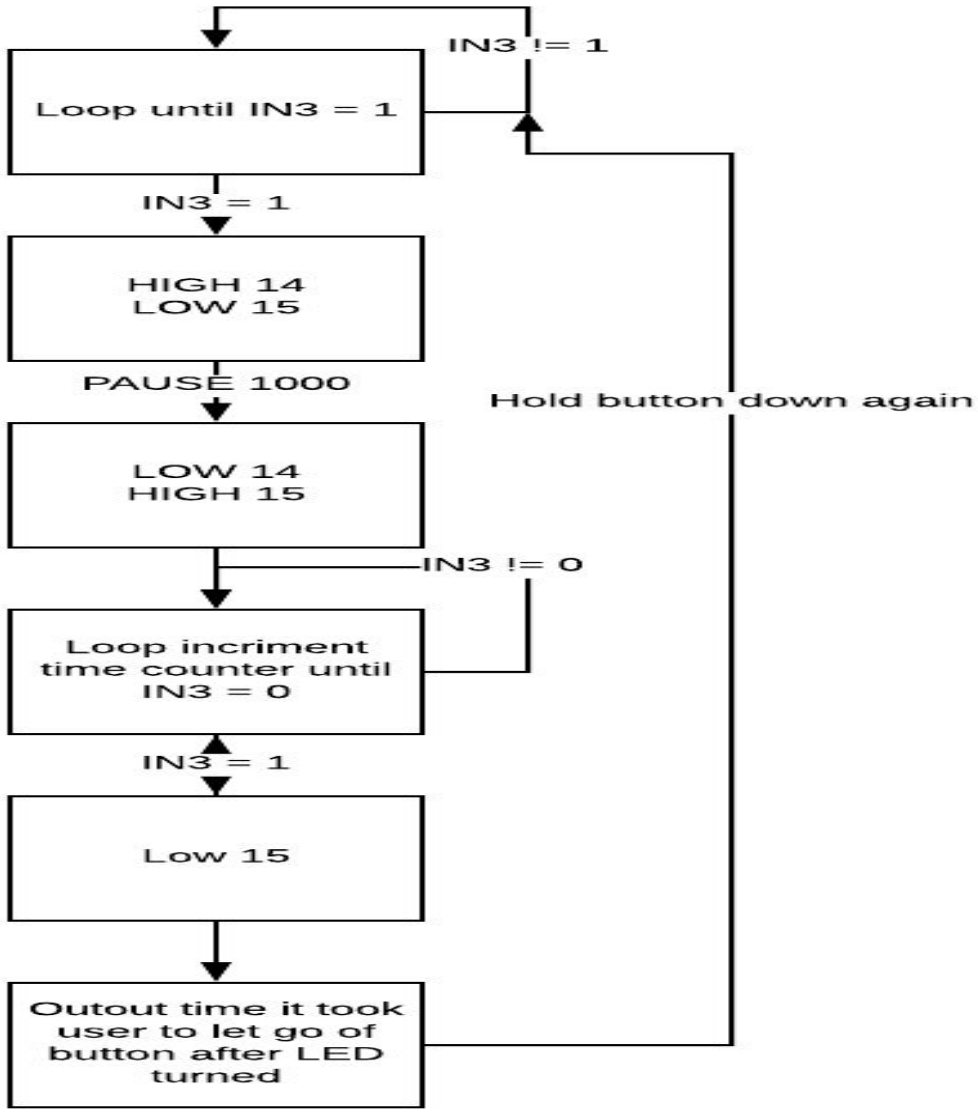


Figure 1: Circuit Diagram

Next code is written to create the game as described above in the objective. For the assignments, more code is needed to make the score more accurate to the actual time that the button was help, randomize the light switching timer, and make it so that releasing the button before the light changes doesn't give a good score. This can be seen below in the flowchart below.



The circuit in **Figure 1** is a very simple circuit with most of the components only being incorporated for protection. It consists of a 470Ω resistor that gets a input based on the code and passes it through the LED. Once the voltage to the LED switches, it changes color and records when the button was released. The push button circuit is a voltage to a pin on the push button that will eventually flow through to the output. The output has a 220Ω resistor on the input to limit the current to the controller. It also has a 10KΩ resistor on the ground for component safety.

3. Expected Results

Once the program is installed onto the board, the LED should remain off until the button is pressed by the player. Once it is pressed the light should stay red for a period of time then turn green. When the light turns green, the player releases the button as quickly as possible. The program records the time it takes to release the button once it turns green and records it as a score variable. It outputs the time score into the console and also directions to play it again.

4. Experiment and Design Revisions

As the team worked through the lab assignment each item was implemented separately on the original code then brought together upon final completion. Debugging statements were used periodically through the code to instruct the user on how to operate the device as well as return the winning "value" for the user(s). Due to availability of resistors in lab, in an instance we were required to use a series of resistors rather than the ideal single resistor.

5. Observations

When there is no button press the LED remains off as seen in **Figure 2**. To start the game the player presses and holds the button and the LED will show red as in **Figure 3**. After a randomly determined period of time, the LED turns green as shown in **Figure 4** and the player is supposed to release the button. When the button is released, the program records the time it took to release and outputs it in the console.

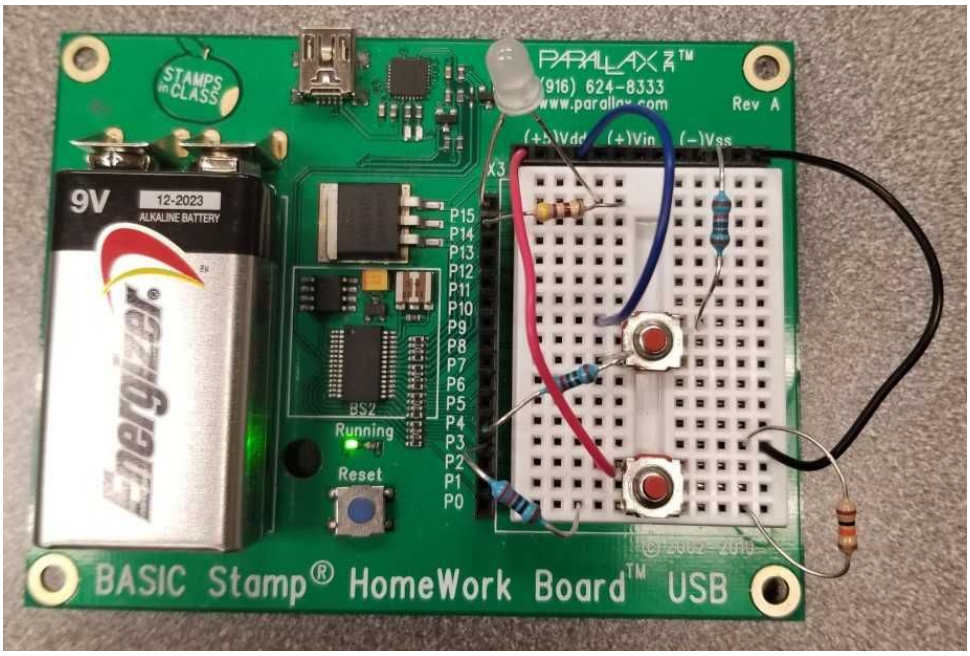


Figure 2: Physical Board Layout

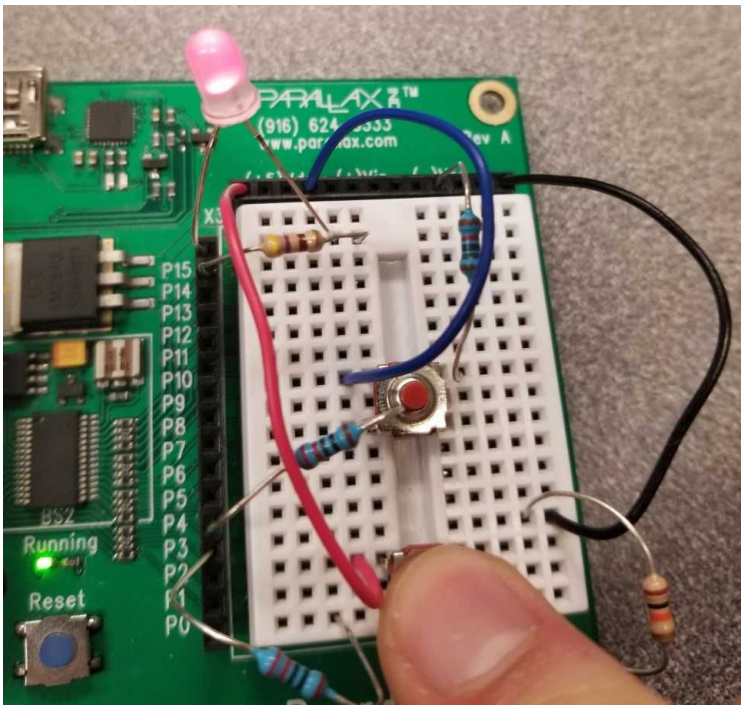


Figure 3: LED turns red when the button is pressed to start game

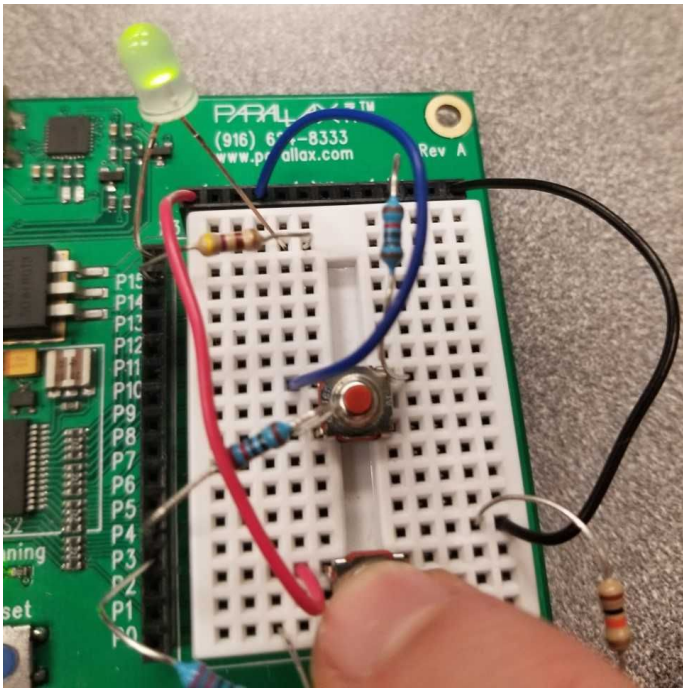


Figure 4: LED turns green when the player is supposed to release the button

6. Discussions

Things to understand: How polling and counting works. Polling is the process of checking something over and over again very quickly to see if it has changed. Counting is the process of adding a number to a variable each time something does (or does not) happen.

Things Learned: When a variable is declared in PBASIC, its value is automatically zero until a command sets it to a new value. The operator "*/" is used to multiply a value with a fraction in PBASIC.

Understanding the difference between pseudo random and actual random. Pseudo Random means that it seems random, but it is not truly random. Each time you start the program over again, the user will get the same sequence of values.

Understanding of what a seed is and how it's utilized. A seed is a value that is used to start the pseudo random sequence. If you use a different value for the seed, it will return a different pseudo random sequence.

7. Exercises

Item 1: When a player holds the button for 30 seconds, his score is actually around 14,000 ms, a measurement of 14 seconds. This has to be fixed!

Solution: Add a multiplier to timeCounter */ 548 to help assist with the code overhead, which is the amount of time it takes for the BASIC Stamp to execute it's given commands.

Item 2: Players soon figure out that the delay from red to green is 1 second. After playing it several times, they get better at predicting when to let go, and their score no longer reflects their true reaction time.

Solution: Add a random seed to the program that is added to the pause time for the LED light to make it less easy for a user to predict when the light will go off thus giving a more accurate representation of their true reaction time.

Item 3: A player that lets go of the button before the light turns green gets an unreasonably good score (1ms). Your microcontroller needs to figure out if the player is cheating.

Solution: Implement an If-Else check for the unreasonably good score. If the user is achieving a score that is essentially impossibly, tell them they're cheating and instruct them to play again. Otherwise, return the users score as normal.

Item 4: Modify RactionTimer.bs2 so that it is a two-player game. Add a second button wired to P4 for the second player.

Solution: Create a second circuit for the push button and a second set of variables with the same parameters as the first button. This will read the score and output Player 1 and Player 2 scores.

gistfile1.txt Raw

```
' What's a Microcontroller - ReactionTimer.bs2
' Test reaction time with a pushbutton and a bicolor LED.

' {$STAMP BS2}
' {$PBASIC 2.5}

PAUSE 1000 ' Wait 1 sec before 1st message.        _

timeCounter VAR Word ' Declare variable to store time.
value VAR Byte       ' ITEM 2
value = 23           ' ITEM 2

timeCounterA VAR Word  ' Time score of player A
timeCounterB VAR Word  ' Time score of player B

DEBUG "Press and hold pushbuttons", CR,   ' Display reaction instructions.
      "to make light turn red.", CR, CR,
      "When light turns green, let", CR,
      "go as fast as you can.", CR, CR

DO                                    ' Begin main loop.
  DO                                  ' Nested loop repeats...
  LOOP UNTIL (IN3 = 1) AND (IN4 = 1)  ' until pushbutton press.

 HIGH 14                              ' Bicolor LED red.
 LOW 15

 RANDOM value                                  ' ITEM 2
 DEBUG "Delay time ", ? 1000 + (value*4), CR   ' ITEM 2

 PAUSE 1000 + (value*4)                        ' Delay 1 second. & ITEM 2

 LOW 14                               ' Bicolor LED green.
 HIGH 15

 timeCounterA = 0                      ' Set timeCounter to zero.
 timeCounterB = 0
```

```
39  DO                                  ' Nested loop, count time...
40
41      PAUSE 1
42
43      IF (IN3 = 1) THEN
44        timeCounterA = timeCounterA + 1
45      ENDIF
46      IF (IN4 = 1) THEN
47        timeCounterB = timeCounterB + 1
48      endif
49
50      LOOP UNTIL (IN3 = 0) AND (IN3 = 0)   ' until pushbutton is released.
51
52      LOW 15                              ' Bicolor LED off.
53
54      timeCounter = timeCounter */ 548     ' ITEM 1
55
56      timeCounterA = timeCounterA */ 548
57      timeCounterB = timeCounterB */ 548
58
59      DEBUG "Player A Time: ", DEC timeCounterA, " ms. ", CR
60      DEBUG "Player B Time: ", DEC timeCounterB, " ms. ", CR, CR
61
62      IF (timeCounterA < timeCounterB) THEN
63        DEBUG "Player A is the winner!", CR
64      ELSEIF (timeCounterB < timeCounterA) THEN
65        DEBUG "Player B is the winner!", CR
66      ELSE ' A & B times are equal
67        DEBUG "It's a tie!", CR
68      ENDIF
69    DEBUG CR
70    DEBUG "To play again, hold the ", CR ' Play again instructions.
71    DEBUG "buttons down again.", CR, CR
72  LOOP
73
74
75
```