Final Project

Title

Ladder Game

Course

CSC 11

Section

48830

Due Date

December 10, 2019

Programmer

Christopher Alexman

Table of Contents

0 Overview

Here is the prompt for the final project:

Ladder Game - This game involves a setup of LEDs in a row and a button. The goal is to get from the bottom led all the way to the top without them resetting. The LEDs will flash, and you can only move up one led at a time, when the LED is lit up, or else you get reset to the bottom.

I would say there are 4 main components to this project in 2 categories. This is my checklist to show that I have fulfilled the requirements of the project.

Hardware:

1. Set up 8 LEDs in a row. Connect each to power (output from pi) and ground.
2. Add a button that is used for input to the pi.

Software:

1. Have the button push input signal the LED to turn on and move to the next LED.
2. If the button is pushed too late or when the LED is off, get reset to the bottom.

Overview:

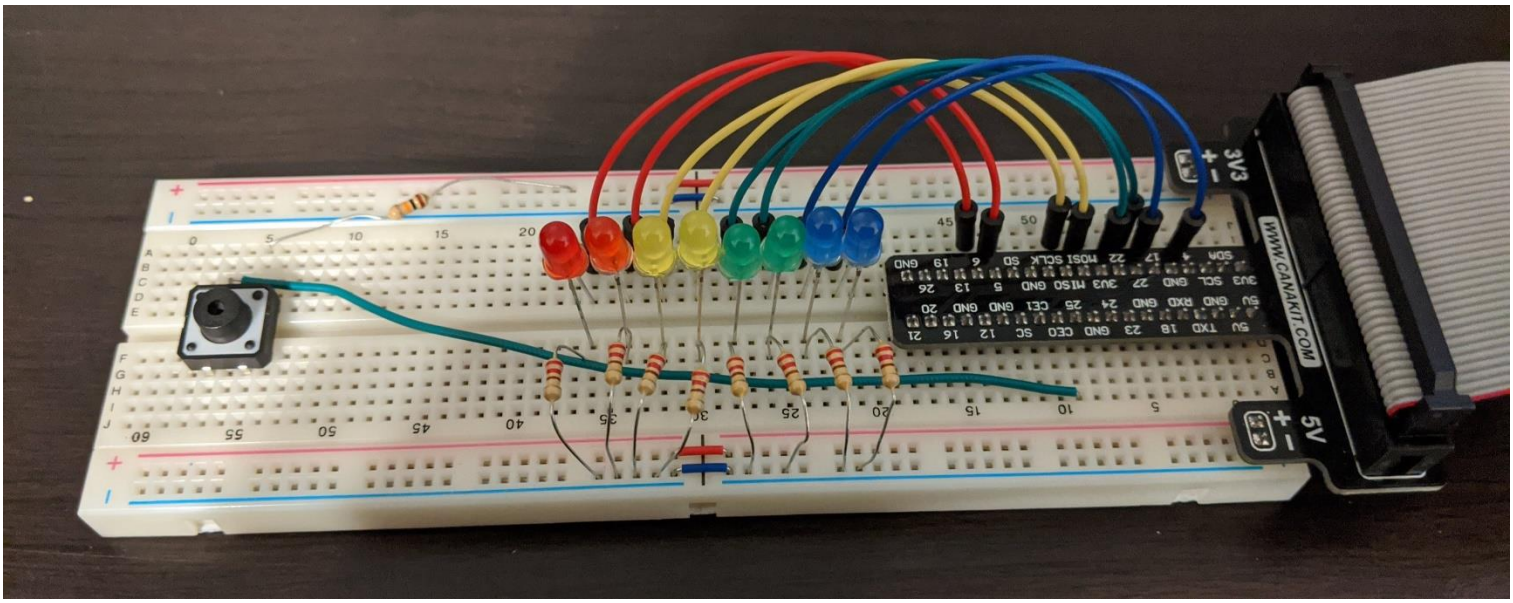
Total lines of code: 724

Sources used: wiringPi

1 Introduction

I have completed all of these. In this documentation I will go through the process of doing this project and go into detail about how I organized the information.

But first here is what my project looks like:



Here is some general information about the hardware:
I used wiringPi for the pin numbers:

.equ RED_ONE, 22

.equ RED_TWO, 21

.equ YLW_ONE, 13

.equ YLW_TWO, 12

.equ GRN_ONE, 3

.equ GRN_TWO, 2

.equ BLU_ONE, 0

.equ BLU_TWO, 7

The button pin:

.equ BTN_PIN, 6

2 Pseudocode

```
mainLoop:   while true blink LED on
            check if button pressed
                if pressed go to next level
            wait 1 second
            check if button pressed
                if pressed go to next level
            blink LED off
            wait 1 second
levelX:     turn on LED of previous level
            delay 0.25 seconds
            check if button pressed
            repeat last 2 steps 2 more times
            if button pressed go to next level
            turn off LED
            if timed out blink led 3 times
            turn off LEDs on a delay from top to bottom
            branch back to main loop
winLevel:   check if button pressed
            turn last LED on
            blink on all LEDs, delay 0.1 of a second
            turn off all LEDs, delay 0.1 of a second
            repeat this 5 times
            end program
```

3 The Process

I started this project out by going to your GitHub repository to get a refresher on the code you wrote during the hardware part of lab that we did. I read the projects you posted about turning on LEDs and using buttons to go back and forth between colors on the RGB LED.

I saw your comments about using wiringPi so I went to the website to read more about it.

http://wiringpi.com/

I found the chart on the main page that compares the BCM pin numbering to the wiringPi pin numbering. I used this to set up my basic circuit with my 8 LED pins. I decided to use pairs and go with color to signify difficulty. The LEDs go red, yellow, green, and blue in terms of difficulty. I decided to wire them up to the pins on the 3.3V side of the cobbler as it was the best placement for my keyboard and the power cable on my desk. I ran jumper wires of the same color from the led to the pins on the cobbler. The LEDs jump the gap in the middle of the breadboard and continue to the other 5 column row. They are then connected to ground by resistors to complete the circuit. This way only when the pins they are connected to are set to output a voltage do they turn on and light up. I used the chart to double check the number for the wiringPi pin. It turned out that they mapped to 22, 21, 13, 12, 3, 2, 0, 7 for the LEDs (red to blue) and pin 6 for the button. For the button I looked in the Raspberry Pi instruction book that came with the kit and used their diagram as a model. The resistor I am using connects from power to a button. The other side of the button has the wire that connects it to the pin on the cobbler. This way only when the button is being pressed does the circuit complete and the pin read an input voltage. I ran this long wire under the resistors as there was no more room to route it another way and I think it would have looked too crowded.

I checked the wiring of the circuit by using the gpio readall to double check the pin mapping. Then I turned on and off each LED in the terminal with gpio write 1 or 0 to check if they all turned on, which they did.

Then I began to write the code to make the game. I started with the names for the LEDs and the pins on the board they were connected to. Then I set up the button to its pin. I added in variables for high, low, input, and output. The next thing I did was do a setup function where I would set the pins to either input or output depending. The LEDs are all output and the button is the only input. I also saw the bl delay function being called and realized that was the key to the whole project in getting the timing right. I made variables for different time increments between a tenth of a second to a whole second. I knew that I wanted to make it more difficult as you got more LEDs lit up so I had smaller time delays. After this I tested the code to make sure the LEDs were connected to the right pins by adding in an allOn function which would turn all of the LEDs on. This is where I found something interesting.

I noticed that a lot of my code was repeating itself. This was not only from the fact that I had 8 LEDs so the code was bound to repeat but I realized I could do functions to reduce the length of code further. Functions, after all, are perfect for taking something that needs to be done over and over and making it in less lines. I saw on your GitHub that you had functions to set the pin as an input or an output depending by moving into r0 the pin and then branching with link to the function, which then moved into r1 either input or output and called the bl pinMode line. Now with those two functions I only had to do bl set On or Off instead of the rest of those lines which are now in the function. I noticed a similar thing with turning on or off an LED so I made 2 functions for those as well. I also added in a readButton function that saved space there too. With all of these working I continued working on the code.

One of the main problems I ran into with this project was getting the timing right. Initially I had no idea how I was going to make it to where the button needed to be pressed only when the LED was on. I got the LEDs to move up a level on their own but not in conjunction with the button. This is when I figured it out.

I got the idea that I could turn on the LED, delay for a small time, read the button, delay for a small time, read the button, and then repeat this until the timing delays added up to the intended difficulty for the specific LED. The good thing was I could take this logic and apply it to almost every level with only changing a few variables. Only the first and final levels would be different but 2 through 7 would be the same. For this I used the logic above and had the code branch to the next level if the button was pressed during the time it was on. I also gave a delay at the beginning of each level to give the player time to get ready to push the button again. If the player missed the button I had the LED they missed blink on and off 3 times and then branch to a timedOff function I made. The 3 flashes indicates that the player missed the button and the timedOff function is a cool way of resetting the game back to the first LED. The timedOff function essentially starts with the highest LED, turns it off, waits a tenth of a second, then turns the next LED off. I just thought it was a good way to signify losing after the 3 blinks other than immediately resetting back to the first level.

Another problem I ran into was that if the player held the button down it would go through all the levels one after another. I had forgot to account for pushing the button if the next LED was off. For this I included a read button function call as the next level began that would be after the initial delay but before that levels LED was turned on. It worked because the button pushing after the LED turned off wouldn't matter since the 3 blinking and timedOff function would already be called. I only had to worry about pushing the button before the LED came on. I tested this out and it worked.

The final thing I did was implement a winning function that would indicate the player had won and the function would terminate. To do this the player had to go through every level and hit the button only when the LED in question was on. Then at the 8^{th} level if the button was pressed at the right time all 8 of the LEDs would flash on and off simultaneously 5 times.

4 The Code

```
.equ INPUT,  0          // to set pins as input
.equ OUTPUT, 1          // to set pins as output

.equ LOW,  0            // low  state for LEDs
.equ HIGH, 1            // high state for LEDs

.equ RED_ONE, 22        // there are 4 colors of LED:
.equ RED_TWO, 21        // RED, YELLOW, GREEN, BLUE
.equ YLW_ONE, 13        // there are 2 of each, 8 total
.equ YLW_TWO, 12
.equ GRN_ONE, 3         // I used wiringPi and its pin numbers
.equ GRN_TWO, 2
.equ BLU_ONE, 0
.equ BLU_TWO, 7

.equ BTN_PIN, 6         // the pin for the button

.equ PAUSE_ONE,    1000      // pause for 1.0  second
.equ PAUSE_HALF,   500       // pause for 0.5  seconds
.equ PAUSE_QTR,    250       // pause for 0.25 seconds
.equ PAUSE_TWEN,   200       // pause for 0.20 seconds
.equ PAUSE_FIFT,   150       // pause for 0.15 seconds
.equ PAUSE_TENTH,  100       // pause for 0.10 seconds

.global main
.text
main:
    push {lr}

    bl wiringPiSetup    // using wiringPiSetup for the pin numbers

    bl setup        // call setup function

    bl allOff       // turn off all the LEDs

    bl startLoop        // start the game
```

```
37
38  // this function moves each led pin into r0
39  // then calls the function that will set it
40  // as either an output for the LEDs
41  // or an input for the button
42  setup:
43      push {lr}
44
45      mov r0, #RED_ONE
46      bl setPinOutput
47
48      mov r0, #RED_TWO
49      bl setPinOutput
50
51      mov r0, #YLW_ONE
52      bl setPinOutput
53
54      mov r0, #YLW_TWO
55      bl setPinOutput
56
57      mov r0, #GRN_ONE
58      bl setPinOutput
59
60      mov r0, #GRN_TWO
61      bl setPinOutput
62
63      mov r0, #BLU_ONE
64      bl setPinOutput
65
66      mov r0, #BLU_TWO
67      bl setPinOutput
68
69      mov r0, #BTN_PIN
70      bl setPinInput
71
72      pop {pc}
```

```
74  // this function moves each LED pin into r0
75  // then calls a function to turn that individual LED on
76  allOn:
77      push {lr}
78
79      mov r0, #RED_ONE
80      bl pinOn
81
82      mov r0, #RED_TWO
83      bl pinOn
84
85      mov r0, #YLW_ONE
86      bl pinOn
87
88      mov r0, #YLW_TWO
89      bl pinOn
90
91      mov r0, #GRN_ONE
92      bl pinOn
93
94      mov r0, #GRN_TWO
95      bl pinOn
96
97      mov r0, #BLU_ONE
98      bl pinOn
99
100     mov r0, #BLU_TWO
101     bl pinOn
102
103     pop {pc}
```

```
104
105 // this function moves each LED pin into r0
106 // then calls a function to turn that individual LED off
107 allOff:
108     push {lr}
109
110     mov r0, #RED_ONE
111     bl pinOff
112
113     mov r0, #RED_TWO
114     bl pinOff
115
116     mov r0, #YLW_ONE
117     bl pinOff
118
119     mov r0, #YLW_TWO
120     bl pinOff
121
122     mov r0, #GRN_ONE
123     bl pinOff
124
125     mov r0, #GRN_TWO
126     bl pinOff
127
128     mov r0, #BLU_ONE
129     bl pinOff
130
131     mov r0, #BLU_TWO
132     bl pinOff
133
134     mov r0, #BTN_PIN
135     bl pinOff
136
137     pop {pc}
```

```
139 // this function sets a pin as an input
140 setPinInput:
141     push {lr}
142     mov r1, #INPUT
143     bl pinMode
144     pop {pc}
145
146 // this function sets a pin as an output
147 setPinOutput:
148     push {lr}
149     mov r1, #OUTPUT
150     bl pinMode
151     pop {pc}
152
153 // this function turns a pin on
154 pinOn:
155     push {lr}
156     mov r1, #HIGH
157     bl digitalWrite
158     pop {pc}
159
160 // this function turns a pin off
161 pinOff:
162     push {lr}
163     mov r1, #LOW
164     bl digitalWrite
165     pop {pc}
166
167 // this function will read the button pin
168 // for either on or off, 1 or 0
169 readButton:
170     push {lr}
171     mov r0, #BTN_PIN
172     bl digitalRead
173     pop {pc}
```

```
// this is the start of the game, or level1
// it will loop through and blink the first LED on and off
startLoop:
    mov r0, #RED_ONE        // turn LED on
    bl pinOn

    ldr r0, =#PAUSE_QTR     // pause quarter of a second
    bl delay

    bl readButton           // read to see if button is pressed
    cmp r0, #HIGH           // if button is pressed
    bleq level2         // go to level2

    ldr r0, =#PAUSE_QTR     // pause quarter of a second
    bl delay

    bl readButton           // read to see if button is pressed
    cmp r0, #HIGH           // if button is pressed
    bleq level2         // go to level2

    ldr r0, =#PAUSE_QTR     // pause quarter of a second
    bl delay

    bl readButton           // read to see if button is pressed
    cmp r0, #HIGH           // if button is pressed
    bleq level2         // go to level2

    ldr r0, =#PAUSE_QTR     // pause quarter of a second
    bl delay

    mov r0, #RED_ONE        // turn LED off
    bl pinOff

    ldr r0, =#PAUSE_ONE     // pause a second
    bl delay

    bl startLoop            // recursively call itself
```

```
// levels 2 through 7 are very similar
//they all turn on the next LED
// check if the button has been pressed in the time limit
// then it either moves to the next level, or it ends
level2:
    ldr r0, =#PAUSE_HALF        // pause 0.5 second, give player time to get ready
    bl delay

    bl readButton           // if just holding the button down
    cmp r0, #HIGH
    bleq timedOff           // end the program

    mov r0, #RED_ONE        // turn on the first red LED
    bl pinOn

    ldr r0, =#PAUSE_QTR     // pause 0.25 second, give player time to get ready
    bl delay

    mov r0, #RED_TWO        // turn on second LED
    bl pinOn

    ldr r0, =#PAUSE_QTR     // pause a quarter of a second
    bl delay

    bl readButton           // read the button
    cmp r0, #HIGH           // if it is pressed
    bleq level3         // move on to level 3

    ldr r0, =#PAUSE_QTR     // pause a quarter of a second
    bl delay

    bl readButton           // read the button
    cmp r0, #HIGH           // if it is pressed
    bleq level3         // move on to level 3

    ldr r0, =#PAUSE_QTR     // pause a quarter of asecond
    bl delay

    bl readButton           // read the button
    cmp r0, #HIGH           // if it is pressed
    bleq level3         // move on to level 3

    ldr r0, =#PAUSE_QTR     // pause a quarter of a second
    bl delay

    mov r0, #RED_TWO        // flash the missed LED on/off 3 times
    bl flashOff
    mov r0, #RED_TWO
    bl flashOn
    mov r0, #RED_TWO
    bl flashOff
    mov r0, #RED_TWO
    bl flashOn
    mov r0, #RED_TWO
    bl flashOff
    mov r0, #RED_TWO
    bl flashOn

    bl timedOff         // call the end function
```

```
level3:
    ldr r0, =#PAUSE_HALF
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq timedOff

    mov r0, #RED_TWO
    bl pinOn

    ldr r0, =#PAUSE_QTR
    bl delay

    mov r0, #YLW_ONE
    bl pinOn

    ldr r0, =#PAUSE_TWEN
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level4

    ldr r0, =#PAUSE_TWEN
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level4

    ldr r0, =#PAUSE_TWEN
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level4

    ldr r0, =#PAUSE_TWEN
    bl delay

    mov r0, #YLW_ONE
    bl flashOff
    mov r0, #YLW_ONE
    bl flashOn
    mov r0, #YLW_ONE
    bl flashOff
    mov r0, #YLW_ONE
    bl flashOn
    mov r0, #YLW_ONE
    bl flashOff
    mov r0, #YLW_ONE
    bl flashOn

    bl timedOff
```

```
level4:
    ldr r0, =#PAUSE_HALF
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq timedOff

    mov r0, #YLW_ONE
    bl pinOn

    ldr r0, =#PAUSE_QTR
    bl delay

    mov r0, #YLW_TWO
    bl pinOn

    ldr r0, =#PAUSE_TWEN
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level5

    ldr r0, =#PAUSE_TWEN
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level5

    ldr r0, =#PAUSE_TWEN
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level5

    ldr r0, =#PAUSE_TWEN
    bl delay

    mov r0, #YLW_TWO
    bl flashOff
    mov r0, #YLW_TWO
    bl flashOn
    mov r0, #YLW_TWO
    bl flashOff
    mov r0, #YLW_TWO
    bl flashOn
    mov r0, #YLW_TWO
    bl flashOff
    mov r0, #YLW_TWO
    bl flashOn

    bl timedOff
```

```
level5:
    ldr r0, =#PAUSE_HALF
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq timedOff

    mov r0, #YLW_TWO
    bl pinOn

    ldr r0, =#PAUSE_QTR
    bl delay

    mov r0, #GRN_ONE
    bl pinOn

    ldr r0, =#PAUSE_FIFT
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level6

    ldr r0, =#PAUSE_FIFT
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level6

    ldr r0, =#PAUSE_FIFT
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level6

    ldr r0, =#PAUSE_FIFT
    bl delay

    mov r0, #GRN_ONE
    bl flashOff
    mov r0, #GRN_ONE
    bl flashOn
    mov r0, #GRN_ONE
    bl flashOff
    mov r0, #GRN_ONE
    bl flashOn
    mov r0, #GRN_ONE
    bl flashOff
    mov r0, #GRN_ONE
    bl flashOn

    bl timedOff
```

```
level6:
    ldr r0, =#PAUSE_HALF
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq timedOff

    mov r0, #GRN_ONE
    bl pinOn

    ldr r0, =#PAUSE_QTR
    bl delay

    mov r0, #GRN_TWO
    bl pinOn

    ldr r0, =#PAUSE_FIFT
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level7

    ldr r0, =#PAUSE_FIFT
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level7

    ldr r0, =#PAUSE_FIFT
    bl delay

    bl readButton
    cmp r0, #HIGH
    bleq level7

    ldr r0, =#PAUSE_FIFT
    bl delay

    mov r0, #GRN_TWO
    bl flashOff
    mov r0, #GRN_TWO
    bl flashOn
    mov r0, #GRN_TWO
    bl flashOff
    mov r0, #GRN_TWO
    bl flashOn
    mov r0, #GRN_TWO
    bl flashOff
    mov r0, #GRN_TWO
    bl flashOn

    bl timedOff
```

```
497 level7:
498     ldr r0, =#PAUSE_HALF
499     bl delay
500
501     bl readButton
502     cmp r0, #HIGH
503     bleq timedOff
504
505     mov r0, #GRN_TWO
506     bl pinOn
507
508     ldr r0, =#PAUSE_QTR
509     bl delay
510
511     mov r0, #BLU_ONE
512     bl pinOn
513
514     ldr r0, =#PAUSE_TENTH
515     bl delay
516
517     bl readButton
518     cmp r0, #HIGH
519     bleq level8
520
521     ldr r0, =#PAUSE_TENTH
522     bl delay
523
524     bl readButton
525     cmp r0, #HIGH
526     bleq level8
527
528     ldr r0, =#PAUSE_TENTH
529     bl delay
530
531     bl readButton
532     cmp r0, #HIGH
533     bleq level8
534
535     ldr r0, =#PAUSE_TENTH
536     bl delay
537
538     mov r0, #BLU_ONE
539     bl flashOff
540     mov r0, #BLU_ONE
541     bl flashOn
542     mov r0, #BLU_ONE
543     bl flashOff
544     mov r0, #BLU_ONE
545     bl flashOn
546     mov r0, #BLU_ONE
547     bl flashOff
548     mov r0, #BLU_ONE
549     bl flashOn
550
551     bl timedOff
```

```
553 level8:
554     ldr r0, =#PAUSE_HALF
555     bl delay
556
557     bl readButton
558     cmp r0, #HIGH
559     bleq timedOff
560
561     mov r0, #BLU_ONE
562     bl pinOn
563
564     ldr r0, =#PAUSE_QTR
565     bl delay
566
567     mov r0, #BLU_TWO
568     bl pinOn
569
570     ldr r0, =#PAUSE_TENTH
571     bl delay
572
573     bl readButton
574     cmp r0, #HIGH
575     bleq youWin
576
577     ldr r0, =#PAUSE_TENTH
578     bl delay
579
580     bl readButton
581     cmp r0, #HIGH
582     bleq youWin
583
584     ldr r0, =#PAUSE_TENTH
585     bl delay
586
587     bl readButton
588     cmp r0, #HIGH
589     bleq youWin
590
591     ldr r0, =#PAUSE_TENTH
592     bl delay
593
594     mov r0, #BLU_TWO
595     bl flashOff
596     mov r0, #BLU_TWO
597     bl flashOn
598     mov r0, #BLU_TWO
599     bl flashOff
600     mov r0, #BLU_TWO
601     bl flashOn
602     mov r0, #BLU_TWO
603     bl flashOff
604     mov r0, #BLU_TWO
605     bl flashOn
606
607     bl timedOff
```

```
609 // this function turns off the LED and waits for 0.1 seconds
610 flashOff:
611     push {lr}
612
613     bl pinOff           // turn off LED
614     ldr r0, =#PAUSE_TENTH
615     bl delay
616
617     pop {pc}
618
619 // this function turns on the LED and waits for 0.1 seconds
620 flashOn:
621     push {lr}
622
623     bl pinOn            // turn on LED
624     ldr r0, =#PAUSE_TENTH
625     bl delay
626
627     pop {pc}
628
629 // this function occurs when you have made it through all of the LEDs
630 // it flashes all the LEDs on and off before ending
631 youWin:
632     ldr r0, =#PAUSE_TENTH
633     bl delay
634     bl allOff
635
636     ldr r0, =#PAUSE_TENTH
637     bl delay
638     bl allOn
639
640     ldr r0, =#PAUSE_TENTH
641     bl delay
642     bl allOff
643
644     ldr r0, =#PAUSE_TENTH
645     bl delay
646     bl allOn
647
648     ldr r0, =#PAUSE_TENTH
649     bl delay
650     bl allOff
651
652     ldr r0, =#PAUSE_TENTH
653     bl delay
654     bl allOn
655
656     ldr r0, =#PAUSE_TENTH
657     bl delay
658     bl allOff
659
660     ldr r0, =#PAUSE_TENTH
661     bl delay
662     bl allOn
663
664     bl end
```

```
// this function turns off all the LEDs with a delay
// starting with the last to the first
timedOff:
    mov r0, #BLU_TWO
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #BLU_ONE
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #GRN_TWO
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #GRN_ONE
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #YLW_TWO
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #YLW_ONE
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #RED_TWO
    bl pinOff

    ldr r0, =#PAUSE_TENTH
    bl delay

    mov r0, #RED_ONE
    bl pinOff

    bl allOff

    bl startLoop

// this function ends the program by turning all of the LEDs off
// it also calls the original function to set up the game again
end:
    bl allOff       // turn all the LEDs off

    pop {pc}

```