Simon Says Game

Design Brief

Design an electronic memory game.

Circuit Explanation

The circuit is fairly straight forward, four switches are arranged next to four LEDs of different colours. A piezo is also used to produce noises.

The principle of the game follows the principles of the electronic 'Simon' game of the early Eighties. The system provides a sequence of lights and sounds. The player then has to repeat this sequence by pressing the switches in the correct order. If the player is correct the game continues with a longer sequence.

Program Explanation

The program is fairly complex and makes extensive use of variables to remember how many lights are in the sequence, and how many switches the player has pressed. The light sequence is stored in the EEPROM memory of the PICAXE-28 microcontroller, and so in theory up to 64 steps are available!

The first job is to wait for the start switch to be pressed. When this switch is pushed the EEPROM is loaded with a random sequence of 64 values, which can be the values 1, 2, 3 or 4. As the random command returns a number between 0 and 255, a simple comparison test is made between 0-60, 60-120, 120 –180, 180-255 to load just the values 1 to 4 into memory. Note how the values are made up as a series of 1+1+1+1 steps, this is a useful programming technique which can reduce the total number of lines in a program.

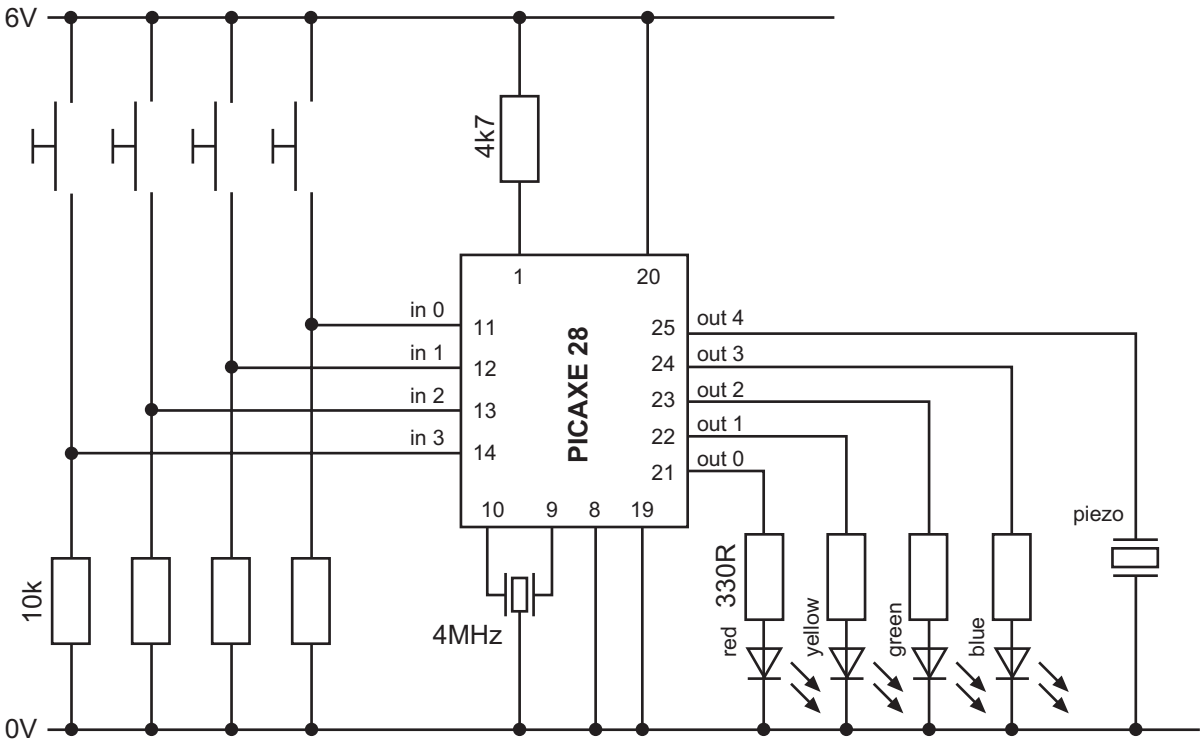
A new game is then started with just two steps in the sequence. This is achieved by loading the value 1 into a variable called counter, so that there will at first be two steps in the playback loop (step 0 and step 1).

The playback loop then reads the memory and makes the beep noise/lights the LED as appropriate via the 'beep' sub-procedure.

It is then turn for the player to respond. The number of presses a player has made is stored in the variable called 'position, and so that viable is first reset to 0. Then the correct response is read back from memory into variable 'key'.

The PICAXE then waits for a key to be pressed. If the key is correct the beep/LED are shown, the position is incremented, and the game continues. When the sequence is completed program flows jumps down to 'success', which makes a noise and then adds one to counter so that the game can be continued with one extra step in the sequence.

If the incorrect key is pressed program flow jumps to 'fail' and a failure noise is made before the program jumps back to the start.



  Revolution Education Ltd. Email: info@rev-ed.co.uk Web: www.rev-ed.co.uk

Program Listing

```
‘ Simon Says Game
‘ For PICAXE-28

‘ Start switch on input 0
‘ Push swithes on inputs 1-4
‘ LEDs on outputs 1-4
‘ Piezo on output 5

symbol rand = b1     ‘ random for loading memory
symbol key = b2      ‘ switch 1-2-3-4
symbol position = b3 ‘ position of player in game
symbol freq = b4     ‘ sound variable
symbol counter = b5  ‘ number of steps in sequence

‘ *** This section waits for start ***

‘ wait for start switch to be pushed
‘ with all four LEDs lit
‘ preload rand with any number by repeatedly
‘ using the random command in the loop

init:
     let pins = %00001111
     random rand
     if pin0 = 1 then preload
     goto init

‘ *** This section loads memory for game ***

‘ load EEPROM memory with 64 numbers
‘ first get the random number (0 to 255)
‘ and then change to either 1,2,3 or 4
‘ and then save into data memory

preload:
     for position = 0 to 63

     let key = 0
     random rand
     if rand > 180 then set1
     if rand > 120 then set2
     if rand > 60 then set3

set4: let key = key + 1
set3: let key = key + 1
set2: let key = key + 1
set1: let key = key + 1

     write position,key

     next position


‘ *** This section plays back a sequence ***

‘ switch off the LEDs and then start
‘ a game with the end counter as 1
     let pins = %00000000
     let counter = 1
```

  Revolution Education Ltd. Email: info@rev-ed.co.uk Web: www.rev-ed.co.uk

```
‘ playback the game sequence
playback:
     for position = 0 to counter
        read position,key
        gosub beep
        pause 500
     next position

‘ *** This section detects the players reply sequence ***

‘ now the user responds
‘ reset the players position to 0
     position = 0

‘ get the correct key player is supposed to hit
‘ from the EEPROM memory
gameloop:
     read position,key

‘ if position is equal to counter then all done
     if position = counter then success

‘ else wait for switch to be pressed
loop:if pin1 = 1 then played1
     if pin2 = 1 then played2
     if pin3 = 1 then played3
     if pin4 = 1 then played4
     if pin0 = 1 then preload
     goto loop

‘ switch pressed so check it is the correct one
‘ if it is make a beep sound and then continue
‘ else fail the game
played1:
     if key <> 1 then fail
     let position = position + 1
     gosub beep
     goto gameloop

played2:
     if key <> 2 then fail
     let position = position + 1
     gosub beep
     goto gameloop

played3:
     if key <> 3 then fail
     let position = position + 1
     gosub beep
     goto gameloop

played4:
     if key <> 4 then fail
     let position = position + 1
     gosub beep
     goto gameloop

‘ *** Failed so make noise and jump back to start ***
```



```
‘ failed so make failed noise, light all LEDs
‘ and go back to start
fail:
     pause 250
     let pins = %00011110
     sound 5,(100,300)
     goto init

‘ *** Succeeded so add another step to sequence and loop ***

‘ success so make a success sound
‘ and then increment counter and do another sequence
success:
     pause 50
     sound 5,(50,100)
     pause 50
     sound 5,(50,100)
     pause 500
     let counter = counter + 1
     goto playback

‘ *** sub light LED and beep ***

‘sub-procedure to light correct LED
‘and make a different beep sound for each LED
‘key always contains number 1,2,3 or 4
‘multiply sound freq by 10 to give larger difference

beep:
     high key
     freq = key * 10
     sound 5,(freq,100)
     low key
     return
```

