TITLE: Raspberry Pi LED Light Schroder Piano

AUTHOR: 2.kfisher@gmail.com

DESCRIPTION: LED (Light Emitting Diode) and LDR (Light Dependant Resistor, or photoresitor) arrays are used to play musical notes using the Raspberry Pi Pygame MIDI sequencer. There are 15 paris of LED & LDR (12 for a full octave of notes, 1 to go up and octave, 1 to go down and octave, and 1 for menu). When the light between LED and LDR is broken, music is played through the Pygame MIDI sequencer. When the light between the Octave Up or Down LED/LDR is broken, all the other notes are shifted up or down an octave. The Pygame MIDI supports over 75 musical instruments and 128 notes per instrument (10 octaves). The menu key can be used to switch instruments. The Raspberry Pi, breadboard, and speaker are within a 20 inch x 30 inch wooden piano that looks like a miniature grand piano.

MATERIAL NEEDED:

1. Raspberry Pi. I used an older Model B, rev 2. Newer models have faster processors and more GPIO available that could expand the project
2. 15 pcs LED (Focus/Narrow light beam), bright
3. 2 pcs LED (generic) for status (optional)
4. 19 pcs 100 Ohm resistors
5. 15 pcs 47k Ohm resistors
6. 15 pcs LDR (Light Dependant Resistor)
7. 1 Breadboard
8. Wires to interconnect Raspberry Pi to breadboard, breadboad to LDR/LED/resistors
9. Piano Frame
    a. 20 inch x 30 inch 1/2” plywood
    b. 4 foot x 8 foot ⅛” hardboard
    c. 1 inch x 2 inch x 20 inch wood to mount LDR and LED (hardwood preferred)
    d. 2 inch x 2 inch x 40 inch internal posts
    e. ¾ inch x 15 inch dowel for legs

STEPS:

1. Build Piano Frame (see Rasberry Pi LED Light Schroeder Piano - Frame)
2. Build LDR/LED mounts with drill press holes for LDR and LED
3. Build Circuit (see Rasberry Pi LED Light Schroeder Piano - Schematic)
4. Copy code (see below)
5. Verify Python installation
6. Run Code

NOTES:

1. Using GPIO interrupts can be more responsive, but started resources for Pygame MIDI to play smoothly, so I chose to do the whole thing in polling. Newer Raspberry Pi may be able to better handle interrupts
2. A focus/narrow beam LED will allow the LED/LDR to be further apart. A 6 inch separation between LDR & LED (3 inch between mounts) works with this project.
3. Do not enable the internal pullup or pulldown of the GPIO as that can change the voltage divider of the LDR
4. LED brightness can be increased by lowering the resistor on the ground pin. Be careful to not draw too much current/voltage from the Raspberry Pi that is supported by your power supply. Check that the voltage is still over 5VDC.

TABLE OF GPIO CONNECTIONS:

Label	Position	GPIO Pin	GPIO #	Note/Key #
Menu	1	7	4	15
No Connect	2			
C	3	11	17	0
C#	4	13	27	1
D	5	15	22	2
D#	6	19	10	3
E	7	21	9	4
Octave Down	8	23	11	16
Octave Up	9	8	14	17
F	10	10	15	5
F#	11	12	18	6
G	12	16	23	7
G#	13	18	24	8
A	14	22	25	9
A#	15	24	8	10
B	16	26	7	11

CODE:

```
#!/usr/bin/python
#Harp.py
#DESCRIPTION: Play an LED Harp
#
#AUTHOR: Ken Fisher Family Christmas Project
#
#Raspberry Pi Model B Rev 2 (000e)
#IRQ for both Note ON and Note OFF???

#import pygame.mixer
import pygame.midi
import time
import random
import RPi.GPIO as GPIO

#PYGAME mixer and midi SETUP
#pygame.mixer.init()
pygame.init()
pygame.midi.init()
port = 2
latency = 1
instrument = 42 # Acoustic Grand Piano is 0
velocity = 127 # Max Volume is 127
baseNote = 60 #Middle C is 60
midiOutput = pygame.midi.Output(port, latency)
midiOutput.set_instrument(instrument)

#LDR status for note length--add 0 based item so 1 based works
ldr_status = {}

#GPIO SETUP
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
gpio_ldr = {
#GPIO #: Note/Offset/Key PIN NAME LABEL
# Note/Key 15-16 Octave Down/Up; 20-2: Menu
      #1      3.3V    LDR Bank1
      #2      5V      LED Bank1
#2: 20, #3      GPIO2   Menu30 (NOTE: 1.8kOhm external pull up resistor) - LED Activ HIGH,
100??
      #4      5V      LED Bank2
#3: 21, #5      GPIO3   Menu31 (NOTE: 1.8kOhm external pull up resistor) - LED??
```

```
      #6      GND     LED Bank1
4: 15,  #7      GPIO4   Menu32
14: 17, #8      GPIO14  OctiveUp21
      #9      GND     LED Bank1
15: 5,  #10     GPIO15  Note06
17: 0,  #11     GPIO17  Note01
18: 6,  #12     GPIO18  Note07
27: 1,  #13     GPIO27  Note02
      #14     GND     LED Bank2
22: 2,  #15     GPIO22  Note03
23: 7,  #16     GPIO23  Note08
      #17     3.3V    LDR Bank2
24: 8,  #18     GPIO24  Note09
10: 3,  #19     GPIO10  Note04
      #20     GND
9: 4,   #21     GPIO9   Note05
25: 9,  #22     GPIO25  Note10
11: 16, #23     GPIO11  OctiveDown20
8: 10,  #24     GPIO8   Note11
      #20     GND
7: 11,  #26     GPIO7   Note12
}

#Define Interupt that will start midi note
def gpio_ldr_irq(pin):
  global ldr_ptr
  global gpio_ldr
  global ldr_status
  print ("NOTE ON: %d, PIN: %d, STATUS: %d" % (gpio_ldr[pin],pin,ldr_status[pin]))
  if (pin != 14) and (pin != 11):
    if (ldr_status[pin] == 0):
      ldr_status[pin] = 1
      midiOutput.note_on(baseNote + gpio_ldr[pin],velocity)
    toExit = 0

ldr_ptr = 0
starttime = 0
toExit = 0
loopDelay = .25
#SETUP GPIO interface
for pin in gpio_ldr:
  #GPIO.setup(pin, GPIO.IN, GPIO.PUD_UP)
  GPIO.setup(pin, GPIO.IN)
```

```
  GPIO.add_event_detect(pin, GPIO.RISING, gpio_ldr_irq, 250) #Debuonce?
  ldr_status[pin] = 0

#SETUP RED LED indicators
GPIO.setup(2, GPIO.OUT)
GPIO.output(2, 1)

print ("Setup complete-Hold Note for 10 seconds to quit")

timeStamp = time.time()
print ("Local Current Time: ", time.asctime(time.localtime()))

#Main Loop
inputLoop = True
while inputLoop:
  time.sleep(loopDelay) #Delay to allow Pygame MIDI to play note properly
  for pin in gpio_ldr:
    if ldr_status[pin] == 1:
      if GPIO.input(pin) == 1:
        toExit += 1
        #print ("toExit %d: %d" % (gpio_ldr[pin],toExit))
        if toExit >= (3/loopDelay):
          inputLoop = False
          print ("toExit %d: %d" % (gpio_ldr[pin],toExit))
      if GPIO.input(pin) == 0:
        toExit = 0
        print ("NOTE OFF %d" % pin)
        midiOutput.note_off(baseNote+gpio_ldr[pin],velocity)
        midiOutput.note_off(baseNote+gpio_ldr[pin],velocity) # In case debounce failed
        ldr_status[pin] = 0
      if GPIO.input(14) == 1:
        baseNote = baseNote + 12
        if baseNote >= 108:
          baseNote = 108
      if GPIO.input(11) == 1:
        baseNote = baseNote - 12
        if baseNote <= 0:
          baseNote = 0

#CLEANUP and SHUTDOWN
del midiOutput
pygame.midi.quit()
GPIO.output(2, 0) #LED Off
```

```
GPIO.cleanup()
exit()
```