UNIVERSITY of DEBRECEN

Dr. József Sütő
Department of IT Systems and Networks

Embedded Systems Laboratory 6

Lab6: Simple applications on the Sense HAT

Project overview

We already know how to turned a LED on and off on the Sense HAT. In this laboratory, you will create some well-known applications such the Knight Rider running light and traffic light control.

The following topics will be covered in this lab:

- State machine
- LED matrix control

Technical requirements

The following components are required to complete this lab:

- Raspberry Pi 3 Model B+ (MicroSD card, power supply, keyboard, mouse)
- Sense HAT
- Or you can use the emulator!

1. Blinking LED control with button

In this exercise you will control the **blinking** (turn on and off for a short time) rate of the LED matrix by the joystick.

1.1. Import necessary libraries

```
from sense_hat import SenseHat

import time
```

1.2. Create an object from the SenseHat class:

```
sense = SenseHat()
```

1.3. Define a variable which controls the blink rated delay:

```
delay_val = 1.0
```

1.4. Define on/off patterns:

```
w = (255,255,255)

n = (0,0,0)


on = [
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w,
   w, w, w, w, w, w, w, w
   ]


off = [
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
    n, n, n, n, n, n, n, n,
    n, n, n, n, n, n, n, n,
    n, n, n, n, n, n, n, n
    ]
```

1.5. Create an event handler to the joystick button where you will control the delay value:

```
def delay(event):
    global delay_val
    # delay_val is a global variable because it
    # has been defined outside of this function
    if event.action == 'pressed':
        delay_val = 0.2
    elif event.action == 'released':
        delay_val = 1.0
```

1.6. Define joystick's trigger event:

```
sense.stick.direction_middle = delay
```

1.7. Finally, create a main loop (infinite loop) where the on and off patterns are changing continuously. The changing frequency is controlled by the delay value:

```
while True:
    sense.set_pixels(on)
    time.sleep(delay_val)
    sense.set_pixels(off)
    time.sleep(delay_val)
```

Tasks to be solved alone:

1.A. You need to associate different blinking rates to all joystick events!

2. Traffic light

Create a traffic light project. The crossing has a three-light traffic control — **red, yellow, and green**. By a button we can change between normal and "out of order" modes (flashing yellow light).

2.1. Import necessary libraries

2.2. Create an object from the SenseHat class:

```
sense = SenseHat()
```

2.2. Create a variable which keeps track the state of our state machine:

```
state = 0
```

2.3. Define red, yellow, and green patterns to the LED matrix:

```
w = (255,255,255)

r = (255,0,0)

g = (0,255,0)

y = (255,255,0)

n = (0,0,0)


red = [

   n, n, n, r, r, n, n, n,

   n, n, n, r, r, n, n, n,

   n, n, n, n, n, n, n, n,

   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n
   ]

red_yellow = [
   n, n, n, r, r, n, n, n,
   n, n, n, r, r, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, y, y, n, n, n,
   n, n, n, y, y, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n
   ]


yellow = [
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, y, y, n, n, n,
   n, n, n, y, y, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n
   ]

green = [
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, n, n, n, n, n,
   n, n, n, g, g, n, n, n,
   n, n, n, g, g, n, n, n
   ]
```

2.4. Write 5 functions to the 5 states. The last state is an "**out of order state**". Other state functions have a duration parameter which determine how much time the state machine should spend in the state (see the figure below).

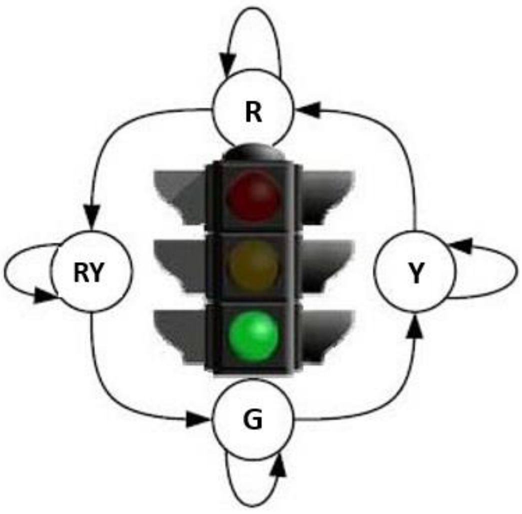
```
def red_state(duration):
    sense.set_pixels(red)
    time.sleep(duration)
    sense.clear()


def red_yellow_state(duration):
    sense.set_pixels(red_yellow)
    time.sleep(duration)
    sense.clear()
```

```
def yellow_state(duration):
        sense.set_pixels(yellow)
        time.sleep(duration)
        sense.clear()

def green_state(duration):
        sense.set_pixels(green)
        time.sleep(duration)
        sense.clear()

def out_of_order_state():
        sense.set_pixels(yellow)
        time.sleep(0.5)
        sense.clear()
        time.sleep(0.5)
```



2.5. Write a *state maintenance* function:

```
def set_state():
    global state
    # state variable has been defined outside
    if state < 3:
        state += 1
    elif state == 3:
        state = 0
    else:
        pass
```

2.6. Create an event handler to the joystick button. At the first press of the button, the state machine goes into the out of order mode (state = 4) while the next press turns back the state machine into normal mode:

```
def button_event(event):
    global state
    if event.action == 'released':
        if state != 4:
            state = 4
        else:
            state = 3
```

2.7. Define joystick's trigger event:

```
sense.stick.direction_middle = button_event
```

2.8. The main function is a state machine which is running in an infinite loop:

```
def main():
    global state
    while True:
        if state == 0:
```

```
        red_state(3)
elif state == 1:
        red_yellow_state(1)
elif state == 2:
        green_state(2)
elif state == 3:
        yellow_state(1)
else:
        out_of_order_state()
set_state()
```

Tasks to be solved alone:

2.A. The above detailed code is redundant! State controller functions (*red_state(), red_yellow_ state(), yellow_state(), green_state()*) can be defined in a single function. Modify the above program code according to it!

2.B. Suppose that, there are more traffic lights on a road. Sometimes, we should synchronize them. To the synchronization, you have to add an additional button event to your code which will halt and unleash the state machine.

3. Knight Rider light bar

The goal of this exercise is to create a sliding red light similarly as the light found on "KITT" from Knight Rider series. The light looks like as a comet in the night sky. Actually, in this exercise we will create a running light instead of a real Knight Rider light because Knight Rider light would require pulse width modulation (PWM).

3.1. Import necessary libraries

3.2. Create an object from the SenseHat class:

3.3. Create a variable (p) which keeps track the rightmost point of light beam. In addition, define some constants related to the light beam length, space size and moving speed of the light beam:

```
p = [2,3]
light_len = 3

space_size = 8

speed = 1/7
```

3.4. Define the initial position of the light beam on the LED matrix:

```
r = (255,0,0)

n = (0,0,0)


space = [

n, n, n, n, n, n, n, n,

n, n, n, n, n, n, n, n,

n, n, n, n, n, n, n, n,

r, r, r, n, n, n, n, n,

n, n, n, n, n, n, n, n,

n, n, n, n, n, n, n, n,
n, n, n, n, n, n, n, n,
 n, n, n, n, n, n, n, n
 ]
```

3.5. Create 2 functions which shift the light beam to the left side and to the right side (**task to be solved alone**):

```
def shift_right():
        …


def shift_left():

        …
```

3.6. Main function which is moving the light beam from right to leaf and left to right continuously:

```
def main():

        global p

         while True:

                while True:

                        shift_right()

                        time.sleep(speed)

                        if p[0] == space_size-1: break


                while True:

                        shift_left()

                        time.sleep(speed)

                        if p[0] == light_len-1: break
```

4. Rain effect (task to be solved alone)

Create a program which will generate rain effect. At the beginning all LEDs in the LED matrix are turned off. The program randomly turns on a LED (with blue color) in the first row of the LED matrix. Thereafter, the program waits 500mS and shifts all rows of the LED matrix one line down. The program performs this process continuously in an infinite loop.

This work was supported by the construction EFOP-3.4.3-16-2016-00021. The project was supported by the European Union, co-financed by the European Social Fund.