Dr. József Sütő
Department of IT Systems and Networks

Embedded Systems Laboratory

Lab 4: A brief overview of the Sense HAT

Project overview

The Sense HAT is an add-on board for Raspberry Pi. It gives you the opportunity to interact with the world that surrounds us, and to create interesting applications. On-board sensors enable you to gather data from accelerometer, gyroscope, magnetometer, pressure, humidity and temperature sensors. Then, with the help of the RGB LEDs, you can create visual feedback.

The following topics will be covered in this lab:

- Sense HAT emulators
- Sense HAT - LED matrix
- Sense HAT - Joystick

Technical requirements

The following components are required to complete this lab:

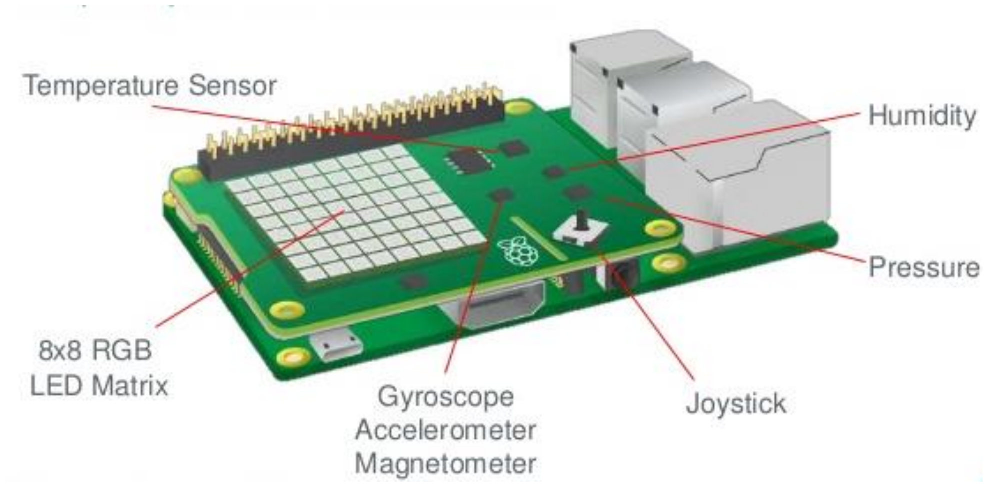
- Raspberry Pi 3 Model B+ (MicroSD card, power supply, keyboard, mouse)
- Sense HAT
- Or you can use the emulator!

1. Sense HAT

The HAT includes the following components:

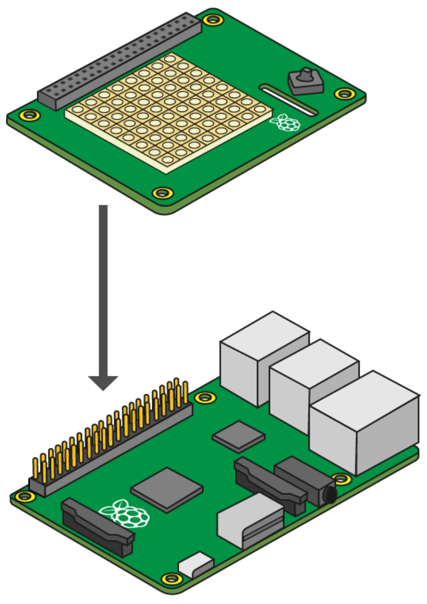
- 8x8 LED matrix display
- Accelerator
- Gyroscope
- Magnetometer
- Air pressure sensor
- Temperature sensor

- Humidity sensor
- Small joystick



Attaching the Sense HAT (probably the HAT already has been attached to your Pi):

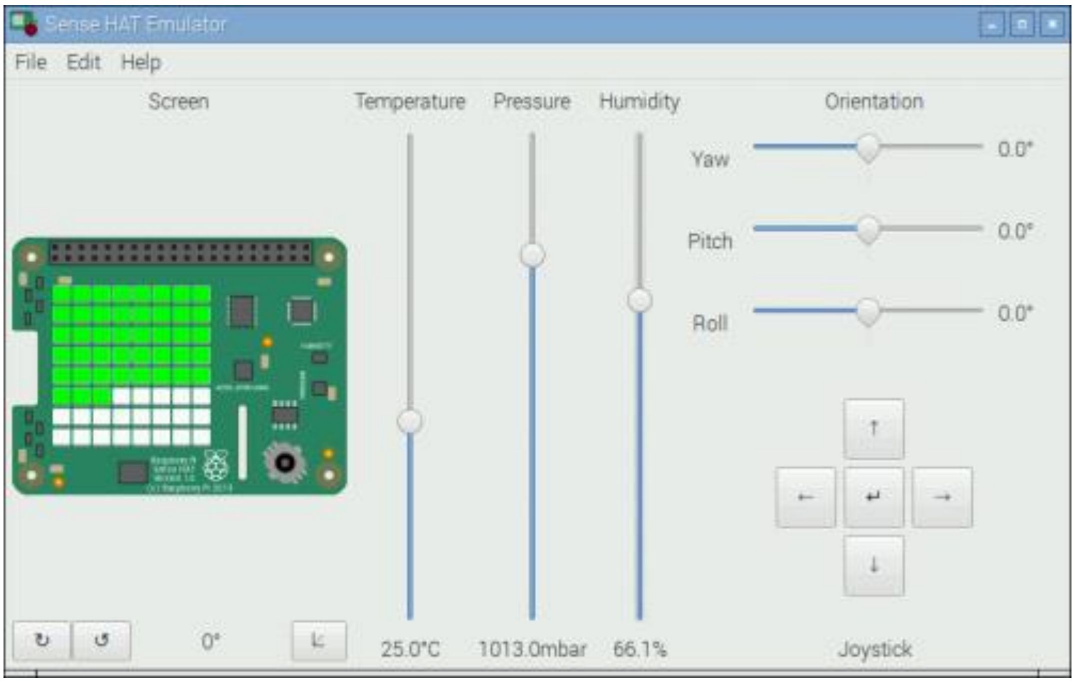
- Before attaching any HAT to your Raspberry Pi, **ensure that the Pi is shut down!!**
- Use two of the provided screws to attach the spacers to your Raspberry Pi (optional)
- Push the Sense HAT carefully onto the pins of your Raspberry Pi
- Be careful when taking off the Sense HAT, as the 40-pin black header can be twisted



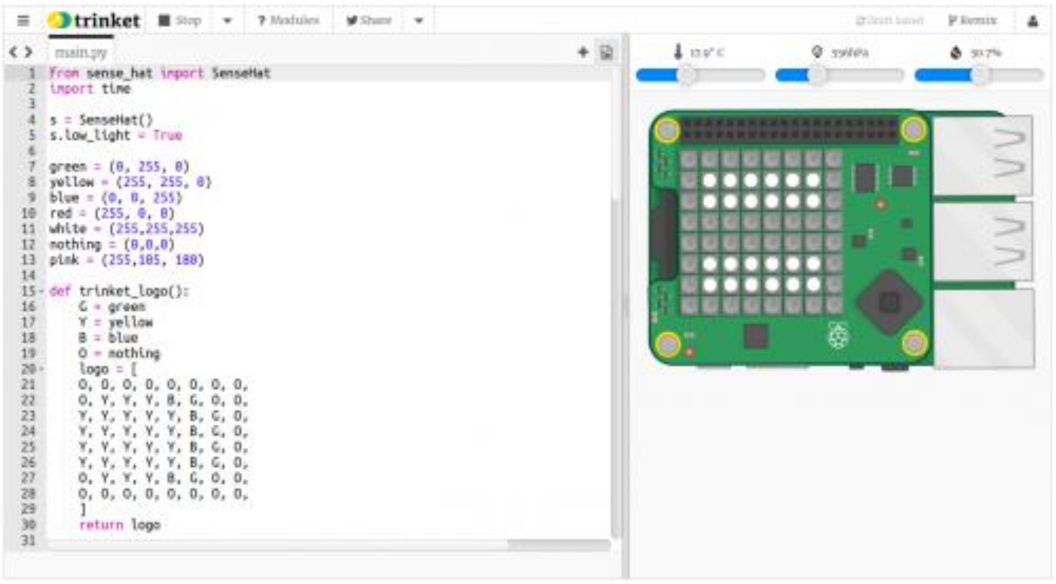
2. Sense HAT emulator

If you don't have a Sense HAT, you can use the emulators:

1. **Emulator on the Raspberry Pi:**
   - Main menu -> Programming -> Sense HAT emulator
   - A visual representation of the Sense HAT hardware appears and using the sliders and buttons you can emulate the HAT's features
   - If you are using this emulator, your Python code must import ***sense_emu***:
     - *from sense_emu import SenseHat*



2. **Online emulator**
   - Open an internet browser, go to https://trinket.io/sense-hat
   - You can create free account



```
from sense_hat import SenseHat
import time

s = SenseHat()
s.low_light = True

green = (0, 255, 0)
yellow = (255, 255, 0)
blue = (0, 0, 255)
red = (255, 0, 0)
white = (255,255,255)
nothing = (0,0,0)
pink = (255,105, 180)

def trinket_logo():
    G = green
    Y = yellow
    B = blue
    O = nothing
    logo = [
    O, O, O, O, O, O, O, O,
    O, Y, Y, Y, B, G, O, O,
    Y, Y, Y, Y, Y, B, G, O,
    Y, Y, Y, Y, Y, B, G, O,
    Y, Y, Y, Y, Y, B, G, O,
    Y, Y, Y, Y, Y, B, G, O,
    O, Y, Y, Y, B, G, O, O,
    O, O, O, O, O, O, O, O,
    ]
    return logo
```

3. Hello LED matrix

The color of an object depends on the color of the light that it reflects or emits. Humans see color because of special cells in our eyes. These cells are called *cones*. We have three types of cone cells, and each type detects either red, blue, or green light. Therefore, all the colors that we see are just mixtures of red, blue, and green. When we want to represent a color in a computer program, we can do this by defining the amounts of red, blue, and green that make up that color. These amounts are usually stored as a single byte. The following code displays the *"Hello world"* message on the LED matrix where the text colour is red while the background is blue.

3.1. Import the **SenseHat** class from **sense_hat** library:

from sense_hat import SenseHat

3.2. Create an object from the SenseHat class:

sense = SenseHat()

3.3. Call the object's *show_message()* method:

sense.show_message("Hello world", scroll_speed=0.05, text_colour = (255,0,0), back_colour=(0,0,255))

3.4. Change the values of the parameters. What do you see?

4. Letters

Display a letter "A" on your Sense HAT.

4.1. Import the **SenseHat** class.

```
from sense_hat import SenseHat
```

4.2. Create an object from the SenseHat class:

```
sense = SenseHat()
```

4.3. Call the object's *show_letter()* method:

```
sense.show_letter("A", text_colour=(255,0,0), back_colour=(0,0,255))
```

5. Display your name

Display the letters of your name one at a time, each in a different random colour with a 1 second pause between them.

5.1. Import necessary modules

5.2. Create an object from the SenseHat class:

```
sense = SenseHat()
```

5.3. Create a random color generator function:

```
# Generate a random color

def random_colour():

    # randint - random integer between an interval

    random_red = randint(0, 255)

    random_green = randint(0, 255)

    random_blue = randint(0, 255)

    return (random_red, random_green, random_blue)
```

5.4. Print out **your name (task to be solved alone)**:

sense.show_letter(“X”, random_colour())

# sleep - temporarily pause your program

sleep(1)

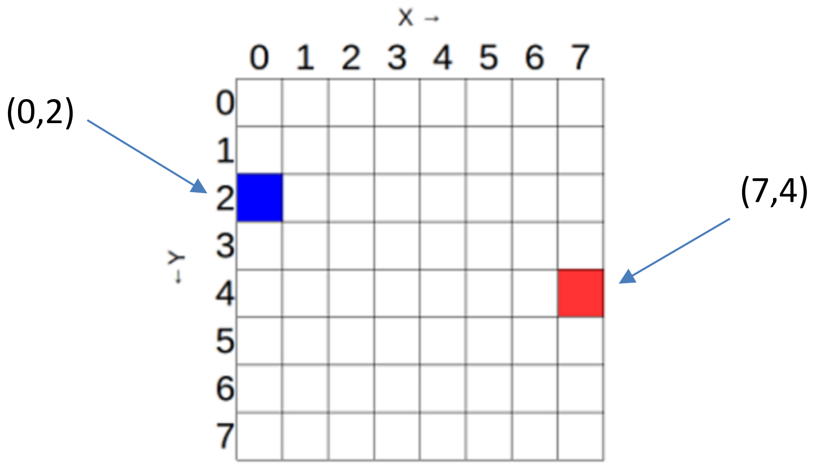
sense.show_letter(“Y", random_colour())

5.5. Call the *clear()* method to clear the LED matrix:

sense.clear()

6. Displaying images

- We can control each LED individually to create an image
- The Sense HAT’s LED matrix uses a coordinate system with an x- and a y-axis.
- The numbering of both axes begins at 0 in the top left-hand corner.
- Each LED can be used as one pixel of an image, and it can be addressed using the (x, y) notation
- Function to set an individual pixel (LED): **set_pixel()**
- Function to set multiple pixels: **set_pixels()**
- You can change the orientation of the LED matrix display on the Sense HAT
  - Rotate the screen: **set_rotation()**
  - Flip the screen horizontally or vertically: **flip_h(), flip_w()**



7. Smile face

Draw a smiley face to the LED matrix.

7.1. Create a SenseHat object

7.2. Define colours to the image drawing:

```
w = (255, 255, 255)       # white

b = (0, 0, 255)           # blue
```

7.3. Image design

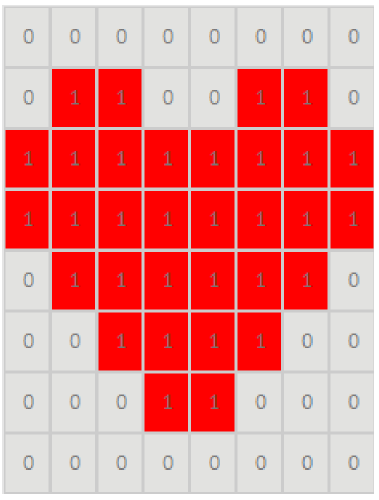
```
# Set up how the image look like

smiley_pixels = [

   w, w, w, w, w, w, w, w,

   w, b, b, w, w, b, b, w,

   w, b, b, w, w, b, b, w,

   w, w, w, w, w, w, w, w,

   w, w, w, w, w, w, w, w,

   w, b, w, w, w, w, b, w,

   w, w, b, b, b, b, w, w,

   w, w, w, w, w, w, w, w]
```

7.4. Display the above image on the LED matrix:

```
#  change all 64 LEDs

sense.set_pixels(smiley_pixels)
```

Tasks to be solved alone:

7.A. Modify the above code, so that a **red heart pattern** be visible on the LED matrix! **Hint:** use the provided pattern.



0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

8. Joystick

- You can detect when the Sense HAT's joystick is pressed, held, and released in five different directions: up, down, left, right, and middle.
- The Sense HAT joystick is mapped to the four keyboard cursor keys, and the joystick's middle-click is mapped to the *Return* key. This means that using the joystick has exactly the same effect as pressing those keys on the keyboard.
- The Sense HAT joystick can be used to trigger function calls in response to being moved.
- The function triggered by the event can either have no parameters, or it can take the event as a parameter.

9. Using the joystick

Depending on which way the joystick was pressed, display one of the letters **U, D, L, R or M** on the LED matrix.

9.1. Create a SenseHat object.

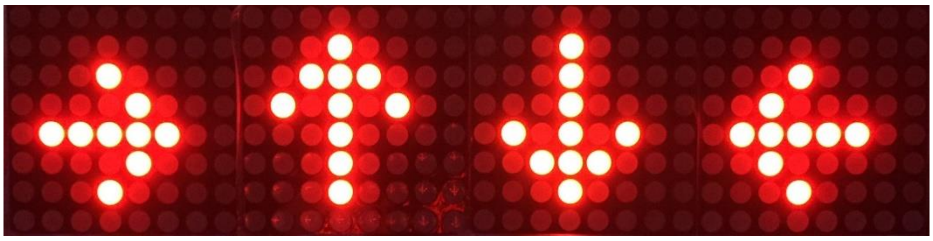
9.2. Check joystick's events continuously in an infinite loop. If a "press" event happened, print out the appropriate letter to the LED matrix:

```
while True:

        #go throw all joystick's events

        for event in sense.stick.get_events():

                # Check if the joystick was pressed
```

```
if event.action == "pressed":
    # Check which direction
    if event.direction == "up":
        sense.show_letter("U")     # Up arrow
    elif event.direction == "down":
        sense.show_letter("D")     # Down arrow
    elif event.direction == "left":
        sense.show_letter("L")     # Left arrow
    elif event.direction == "right":
        sense.show_letter("R")     # Right arrow
    elif event.direction == "middle":
        sense.show_letter("M")     # Enter key

    # Wait a while and then clear the screen
    sleep(0.5)
    sense.clear()
```

Tasks to be solved alone:

9.A. Modify the above code, so that the right **arrow pattern** be visible on the LED matrix!



10. Change the image white balance

The Sense HAT joystick can be used to trigger function calls in response to being moved. You can tell your program to continually "listen" for a specific event and then trigger a function in response. The function triggered by the event can either have no parameters, or it can take the event as a parameter.

Create functions to fill the LED matrix with four different colors. Add triggers to call one function for each possible direction in which the joystick can be pressed.

10.1. Create a SenseHat object.

10.2. Define functions:

```
# Define the functions
def red():
    #set a given color on all LEDs
    sense.clear(255, 0, 0)


def blue():
    sense.clear(0, 0, 255)


def green():
    sense.clear(0, 255, 0)


def yellow():
    sense.clear(255, 255, 0)
```

10.3. Define trigger events:

```
sense.stick.direction_up = red
sense.stick.direction_down = blue
sense.stick.direction_left = green
```

```
    sense.stick.direction_right = yellow

    sense.stick.direction_middle = sense.clear
```

10.4. Keep the program running:

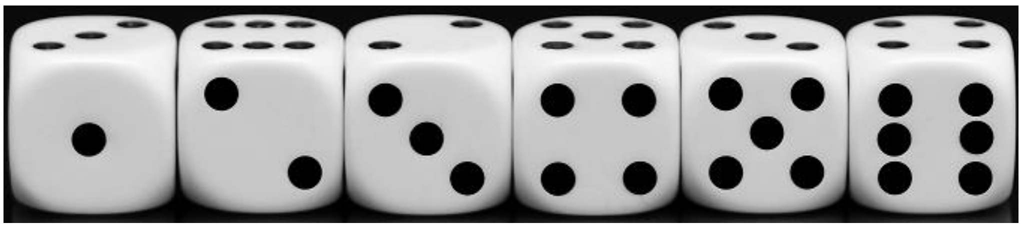
```
    while True:
        # do nothing
        pass
```

Tasks to be solved alone:

10.A. Modify the trigger event of the up direction, so that a **blinking red heart** pattern be visible on the LED matrix!

11. Electronic dice

Create an electronic dice where the press of the joystick triggers the roll of the dice.



11.1. Import necessary libraries.

11.2. Create a SenseHat object.

11.3. Define colors to the image drawing.

```
    o = (0,0,0)          #no color

    b = (0,0,255)
```

11.4. Define patterns (images) to all possible values (see the figure above).

```
    one_img = [o,o,o,o,o,o,o,o,

               o,o,o,o,o,o,o,o,
                o,o,o,o,o,o,o,o,
                o,o,o,b,b,o,o,o,
                o,o,o,b,b,o,o,o,
                o,o,o,o,o,o,o,o,
                o,o,o,o,o,o,o,o,
                o,o,o,o,o,o,o,o]
        # fill the content of the below lists!
        two_img = […]
        three_img = […]
        four_img = […]
        five_img = […]
        six_img = […]
```

11.5. Create an event handler function:

```
def number_gen(event):
        if event.action == "pressed":
                val = random.randint(1,6)
                print(val)
                if val == 1:
                        sense.set_pixels(one_img)
                elif val == 2:
                # complete the remaining elif branches!
                …
                sleep(2)
                sense.clear()
```

11.6. Define trigger event
11.7. Keep the program running

12. Improved electronic dice (task to be solved alone)

Improve the electronic dice with the impression than the dice is being rolled. It means that you have to generate random numbers (e.g. for 3 seconds) and visualize their patterns on the sensor HAT quickly (e.g. 10 times per second). A sample video can be found on course website.

References

[1] McManus S, Cook, M. Raspberry Pi for Dummies, 3rd edition, Wiley, 2017.
[2] Raspberry Pi projects, https://projects.raspberrypi.org/en. Accessed on 10/01/2020.

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.