ECE 1175
Embedded Systems Design

Lab 2 – Sense HAT & Interrupt

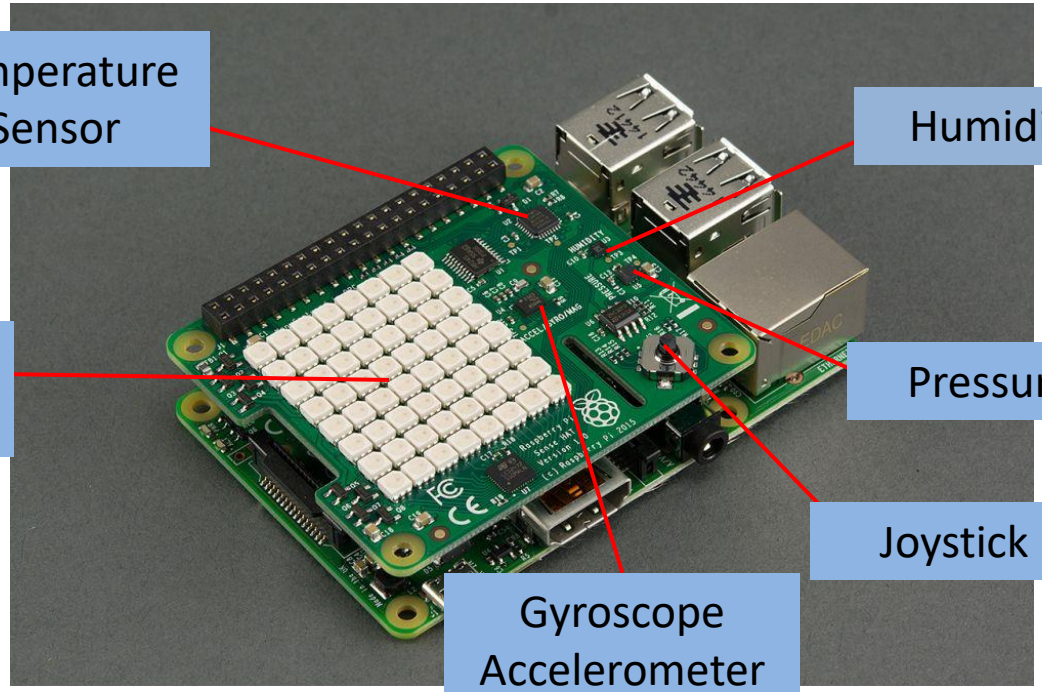
ECE 1175 – Lab 2

- **Sense HAT**
  - Sense HAT introduction
    - On-board peripherals
    - How to operate
  - Lab task 1 to 4
    - Play with Sense HAT using Python library
- **Interrupt**
  - How interrupt works on Linux
    - What is signal?
    - How to trigger interrupt
  - Lab task 5
    - Write your own program to trigger an interrupt on Linux system

Sense HAT

- **Raspberry Pi connected with Sense HAT**

LED matrix, sensors, and joystick on Sense HAT



Sense HAT

- **Sense HAT Python API**
  - Install by entering the following commands in the terminal
    - sudo apt-get update
    - sudo apt-get install sense-hat
    - sudo reboot

- **Python**
  - Beginner-friendly
  - Code is easy to write and read
  - Popular and widely used
  - https://docs.python.org/3/tutorial/

Sense HAT

- **Sense HAT library**
  - Display function
    - show_message()
    - show_letter()
    - set_pixel()
    - set_pixels()
    - set_rotation()
    - flip_v()
    - flip_h()

Sense HAT

- **show_message()**

```
from sense_hat import SenseHat
sense = SenseHat()   # the above two lines should always be in your every
                     # new .py file
sense.show_message("Hello my name is …")
```

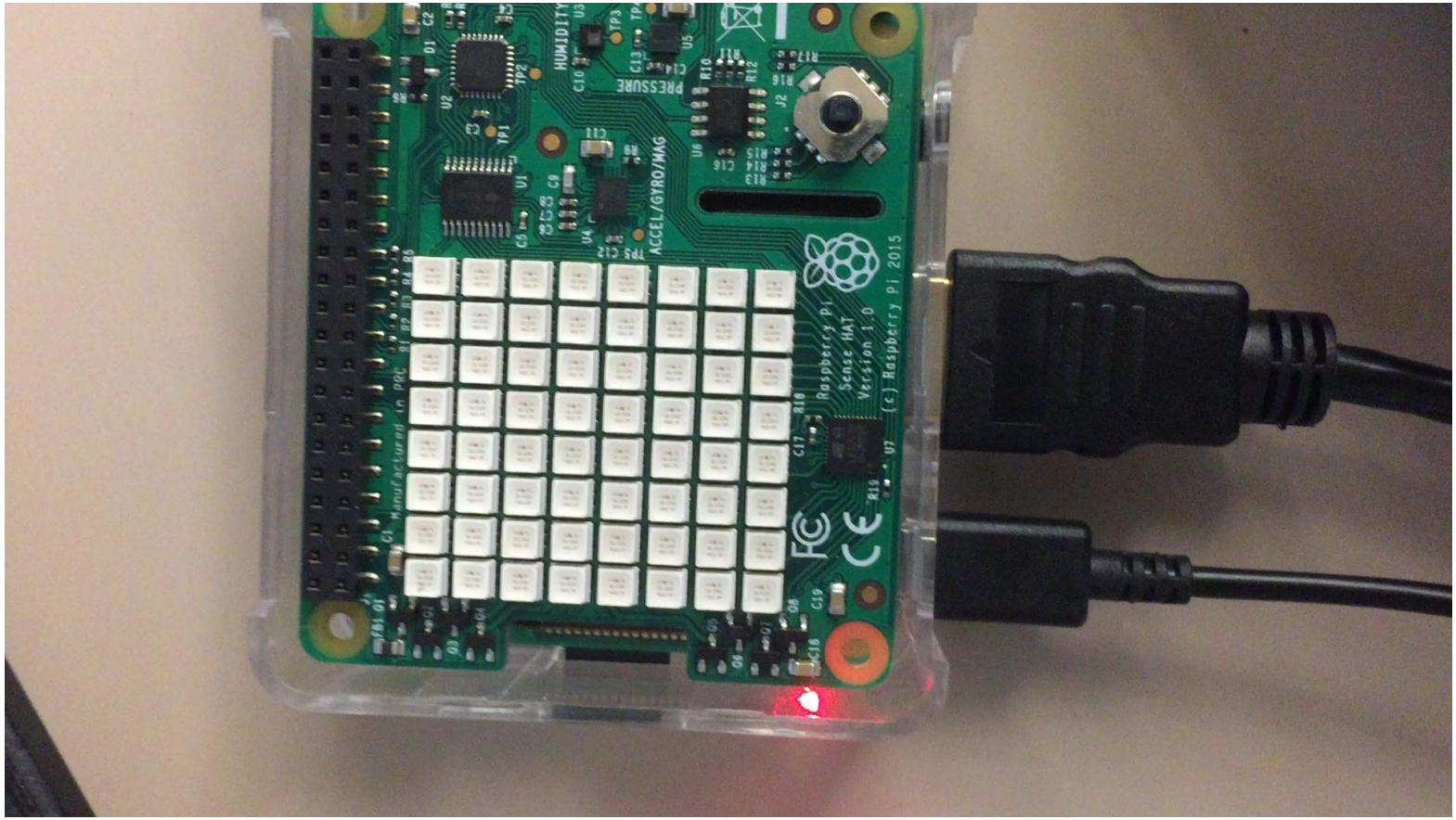
Sense HAT

- **show_message()**

```
from sense_hat import SenseHat
sense = SenseHat()

yellow = (255, 255, 0)
blue = (0, 0, 255)
speed = 0.05

message = “Raspberry Pi is awesome!!”

sense.show_message(message, speed, text_colour=yellow, back_colour=blue)
# or
sense.show_message(message, speed, text_colour = (255, 255, 0),
back_colour = (0, 0, 255))
```

Sense HAT

- **Demo – display “Hello my name is …”**



Sense HAT

- **show_letter()**

```
from sense_hat import SenseHat
sense = SenseHat()

r = (255, 0, 0)

sense.show_letter('J',r)   # display letter J in red
```

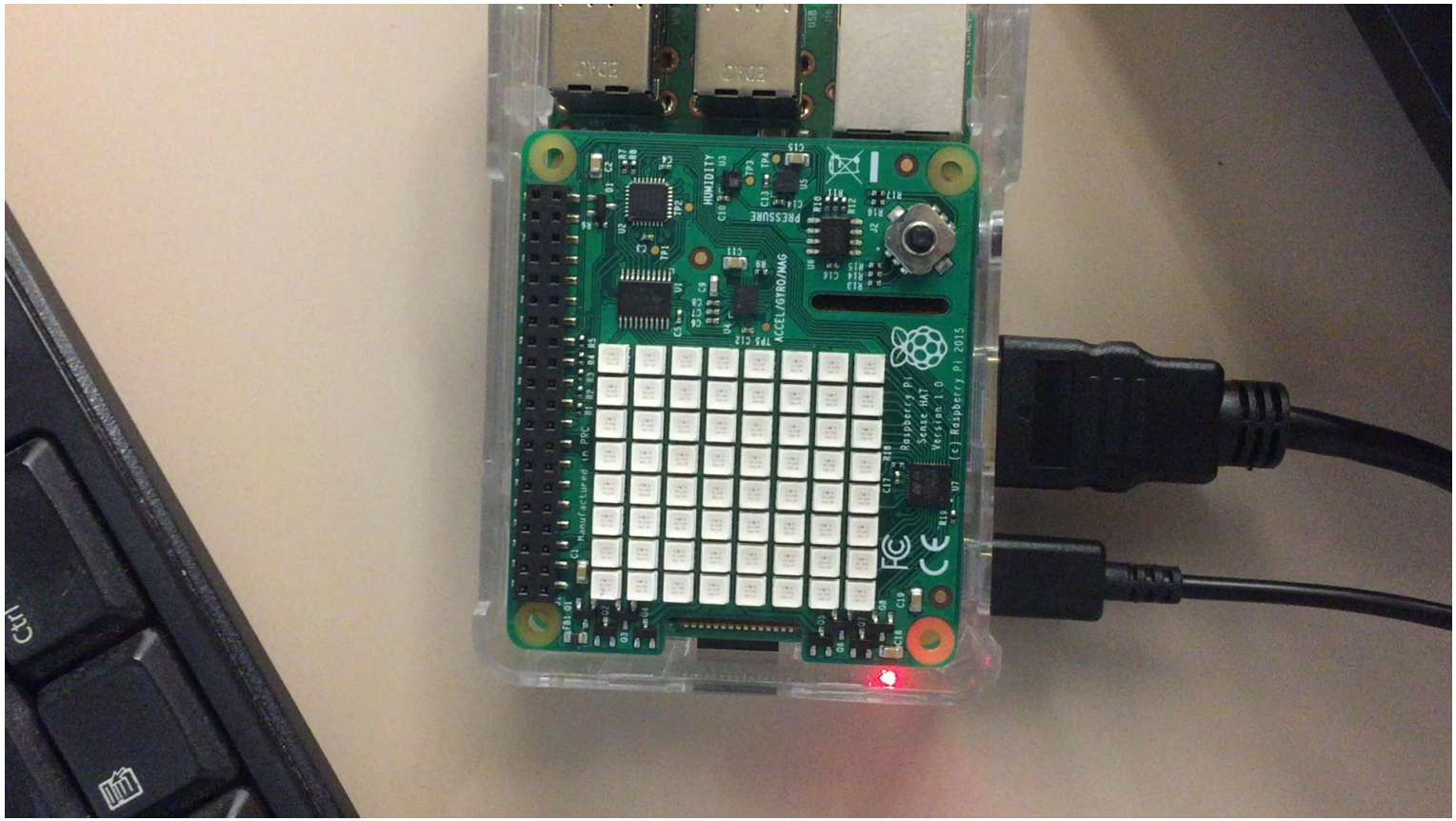
Sense HAT

- **sleep() & clear()**

```
from sense_hat import SenseHat
from time import sleep # import sleep() from time module

sense = SenseHat()
red = (255, 0, 0)
blue = (0, 0, 255)
green = (0, 255, 0)
black = (0, 0, 0)
white = (255, 255, 255)

sense.show_letter("O", red)
sleep(1)
sense.show_letter("M", blue)
sleep(1)
sense.clear()  # clear the LED matrix
```

Sense HAT

- **Demo – display “OMG!”**



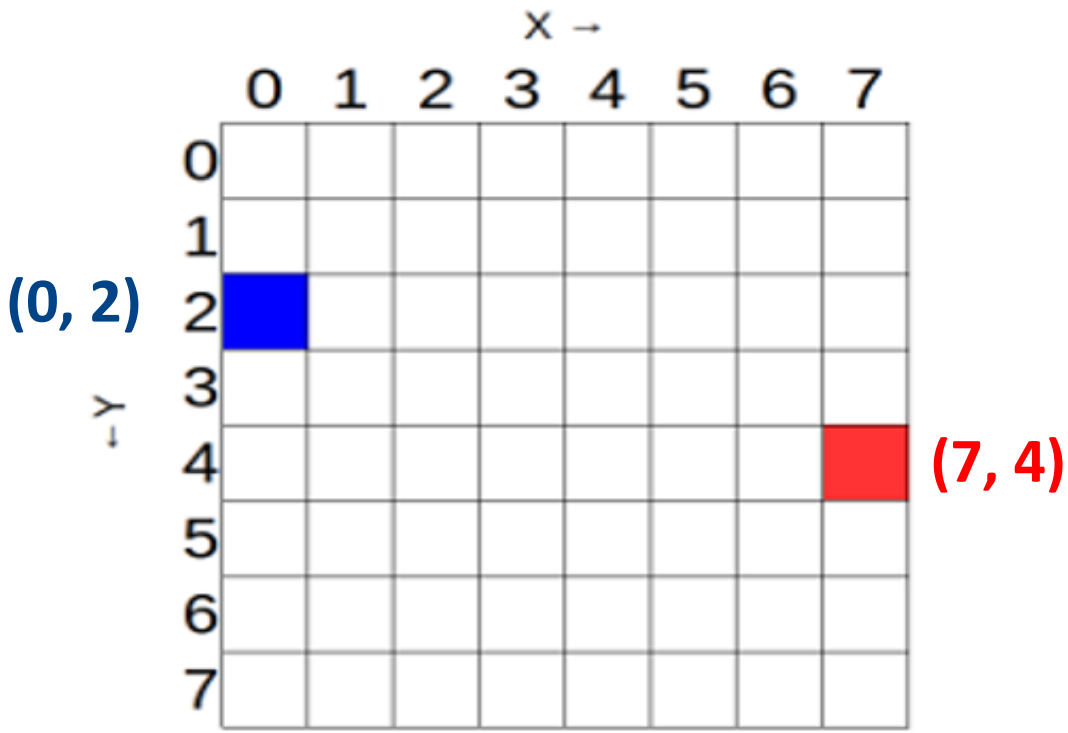
Sense HAT

- **set_pixel()**

```
from sense_hat import SenseHat
import time

sense = SenseHat()

while True:
    sense.set_pixel(0, 2, (0, 0, 255))
    time.sleep(1)
    sense.set_pixel(7, 4, (255, 0, 0))
    time.sleep(1)
```

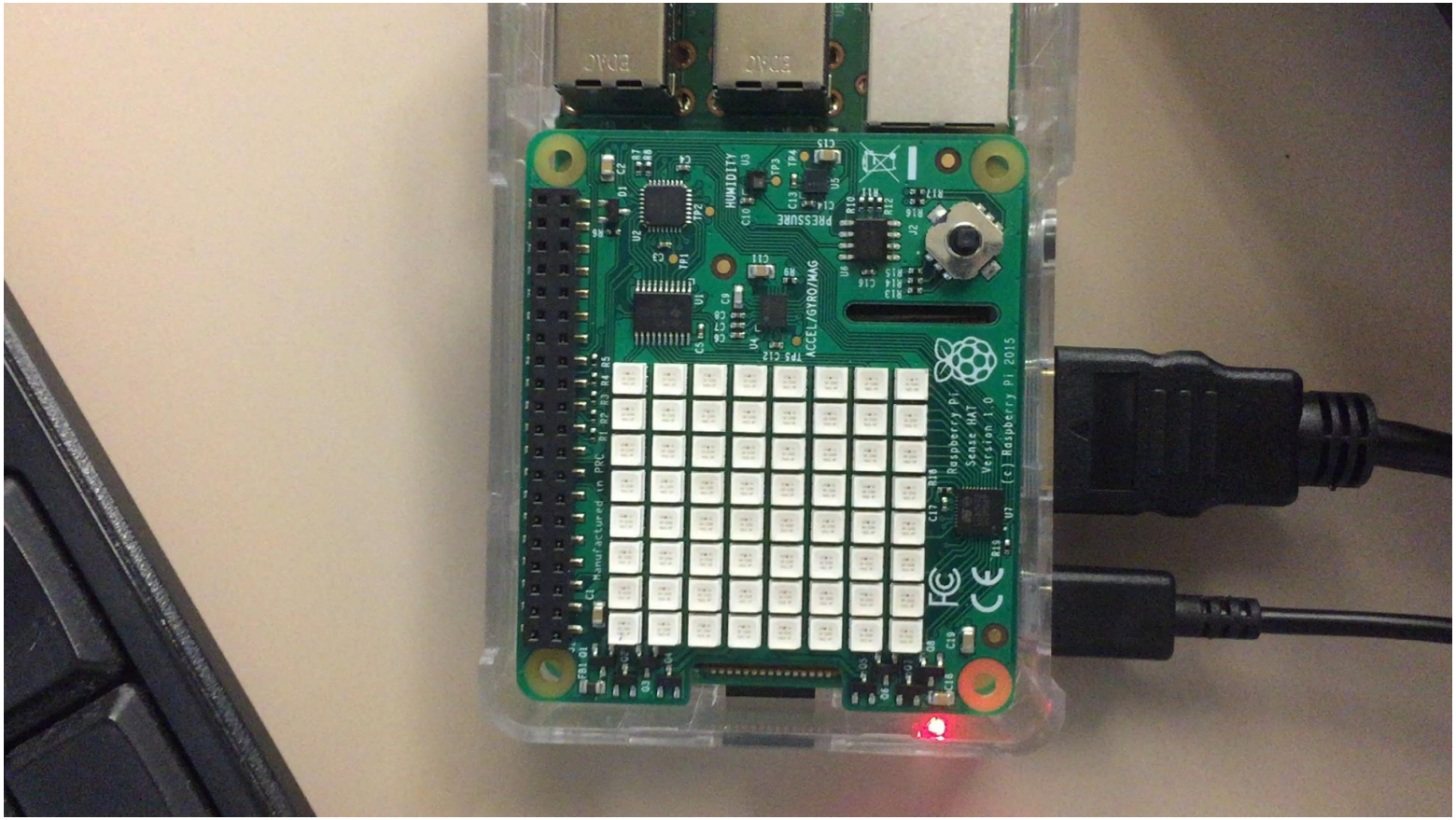
Sense HAT

- **LED matrix**



Sense HAT

- **Demo – set pixel (0, 2) blue and (7, 4) red**



Sense HAT

- **set_pixels()**

```
from sense_hat import SenseHat

sense = SenseHat()
r = (255, 0, 0)
o = (255, 127, 0)
y = (255, 255, 0)
g = (0, 255, 0)
b = (0, 0, 255)
i = (75, 0, 130)
v = (159, 0, 255)
e = (0, 0, 0)
image =
[e,e,e,e,e,e,e,e,e,e,e,e,r,r,e,e,e,e,r,r,o,o,r,r,e,r,o,o,y,y,o,o,r,o,y,y,g,g,y,y,o,y,g,g,b,b,g
,g,y,b,b,b,i,i,b,b,b,b,i,i,v,v,i,i,b]

sense.set_pixels(image)
```

Sense HAT

- **Sense HAT library**
  - Sensor library
    - get_humidity()
    - get_pressure()
    - get_temperature()
    - get_accelerometer_raw()
    - get_orientation()

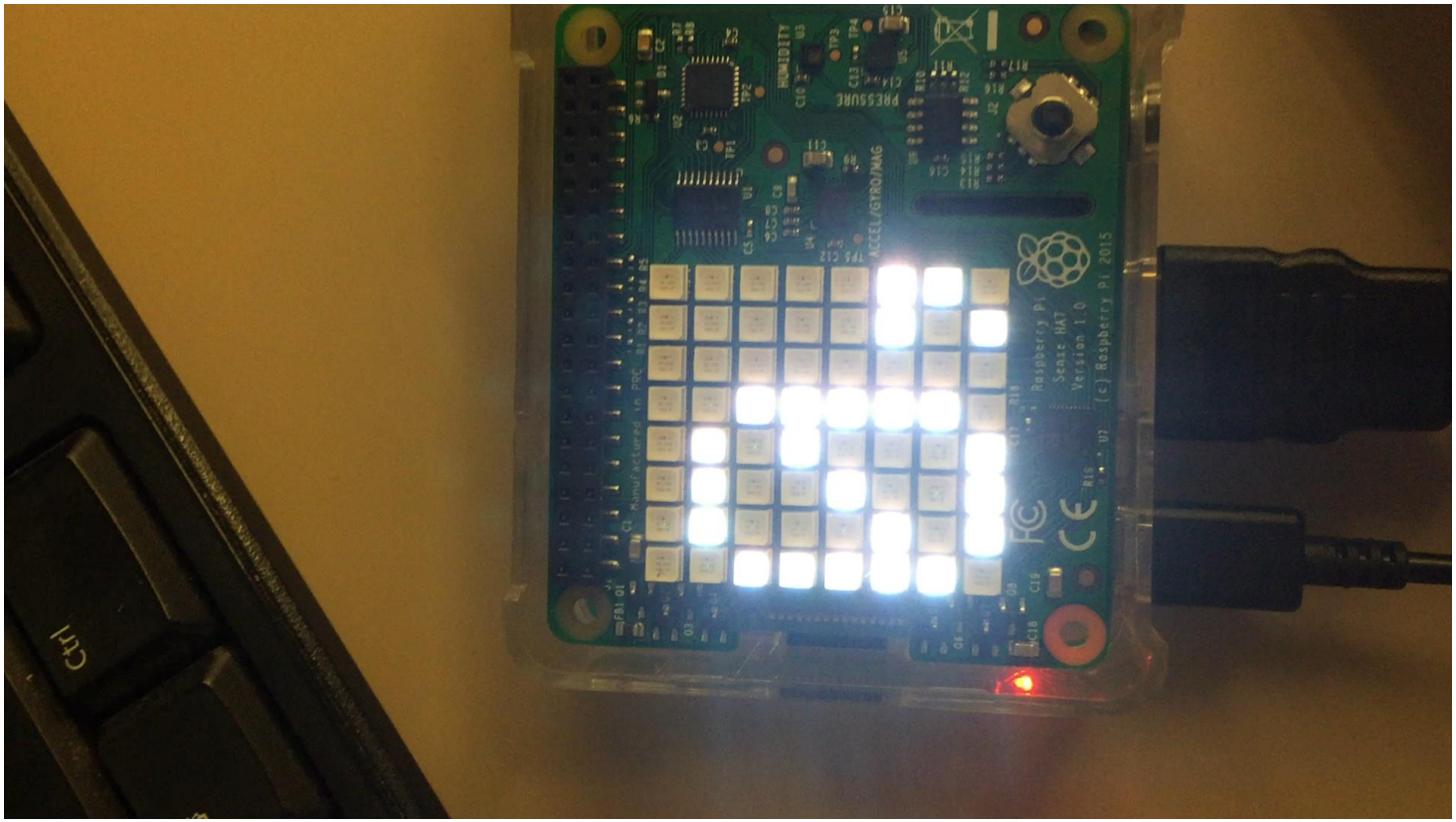
Sense HAT

- **get_humidity(), get_pressure(), get_temperature()**

```
from sense_hat import SenseHatsense = SenseHat()
while True:
    t = sense.get_temperature()
    p = sense.get_pressure()
    h = sense.get_humidity()

    t = round(t, 1)
    p = round(p, 1)
    h = round(h, 1)

    msg = "Temperature = {0}, Pressure = {1}, Humidity = {2}".format(t,p,h)
    sense.show_message(msg, scroll_speed=0.05)
```

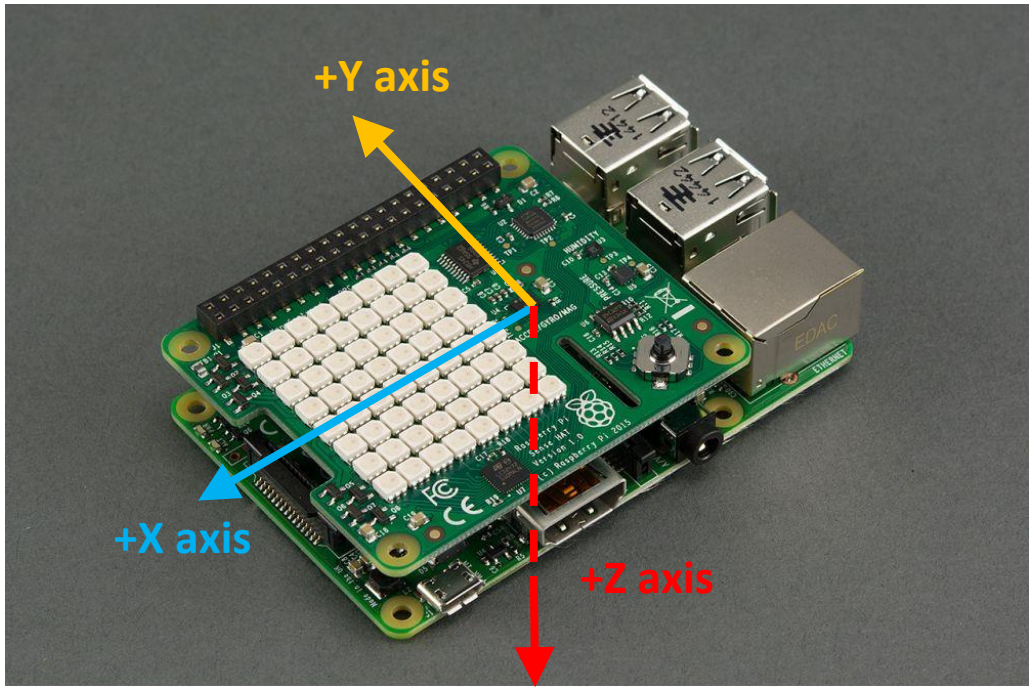
Sense HAT

- **Demo – show sensor readings on LED matrix**



Sense HAT

- **Accelerometer (x, y, z)**



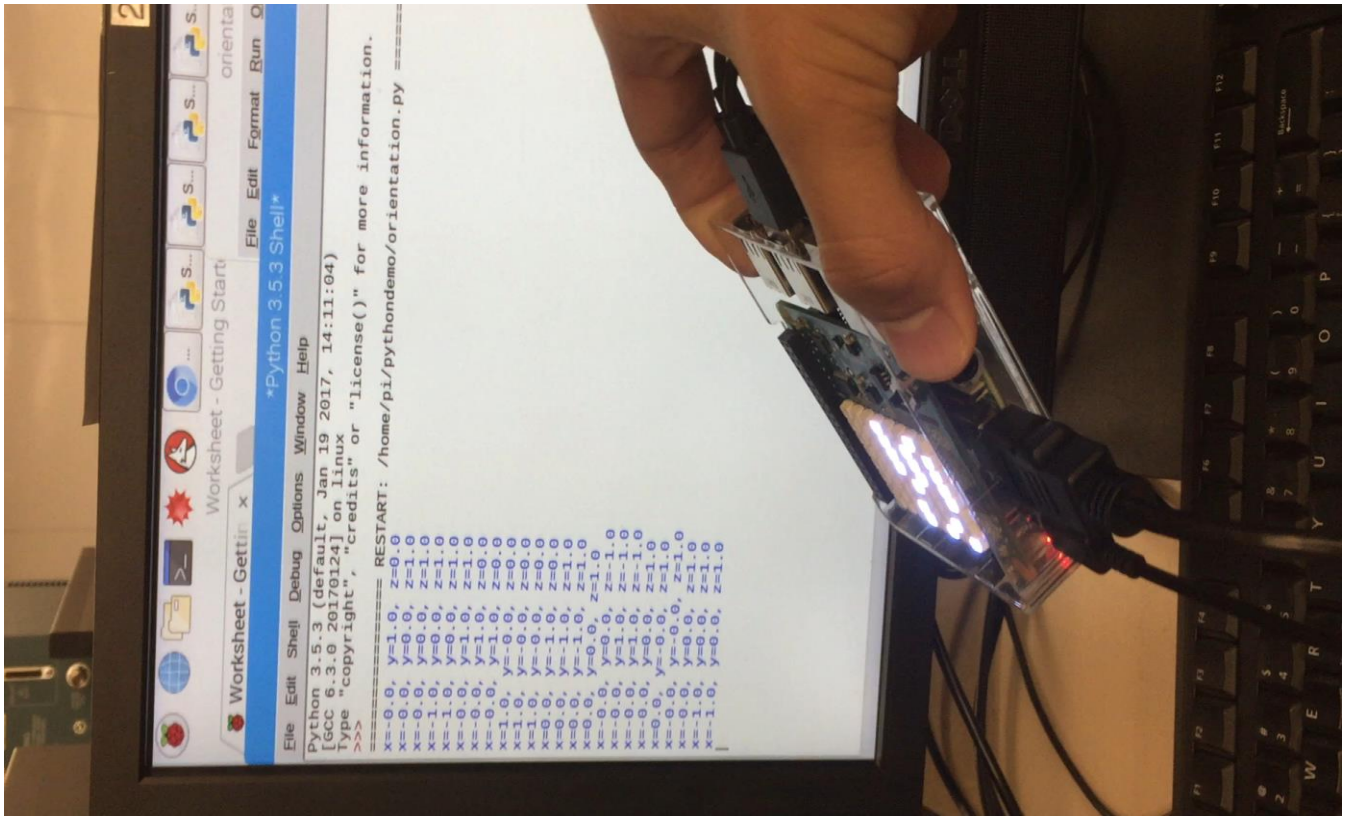
Sense HAT

- **get_accelerometer_raw()**

```
from sense_hat import SenseHat

sense = SenseHat()
while True:
    acceleration = sense.get_accelerometer_raw()

    x = acceleration['x']
    y = acceleration['y']
    z = acceleration['z']

    x=round(x, 1)
    y=round(y, 1)
    z=round(z, 1)

    print("x={0}, y={1}, z={2}".format(x, y, z))
```

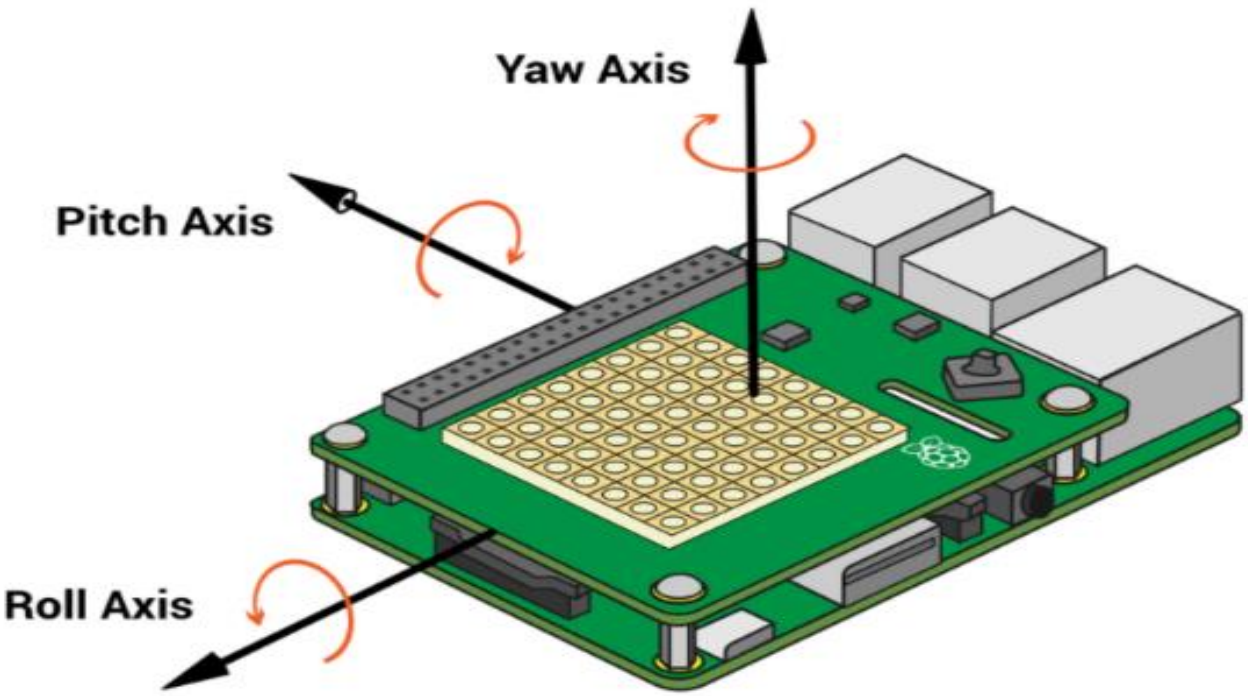
Sense HAT

- **Demo – retrieve accelerometer readings**



Sense HAT

- **Gyroscope (pitch, yaw, roll)**



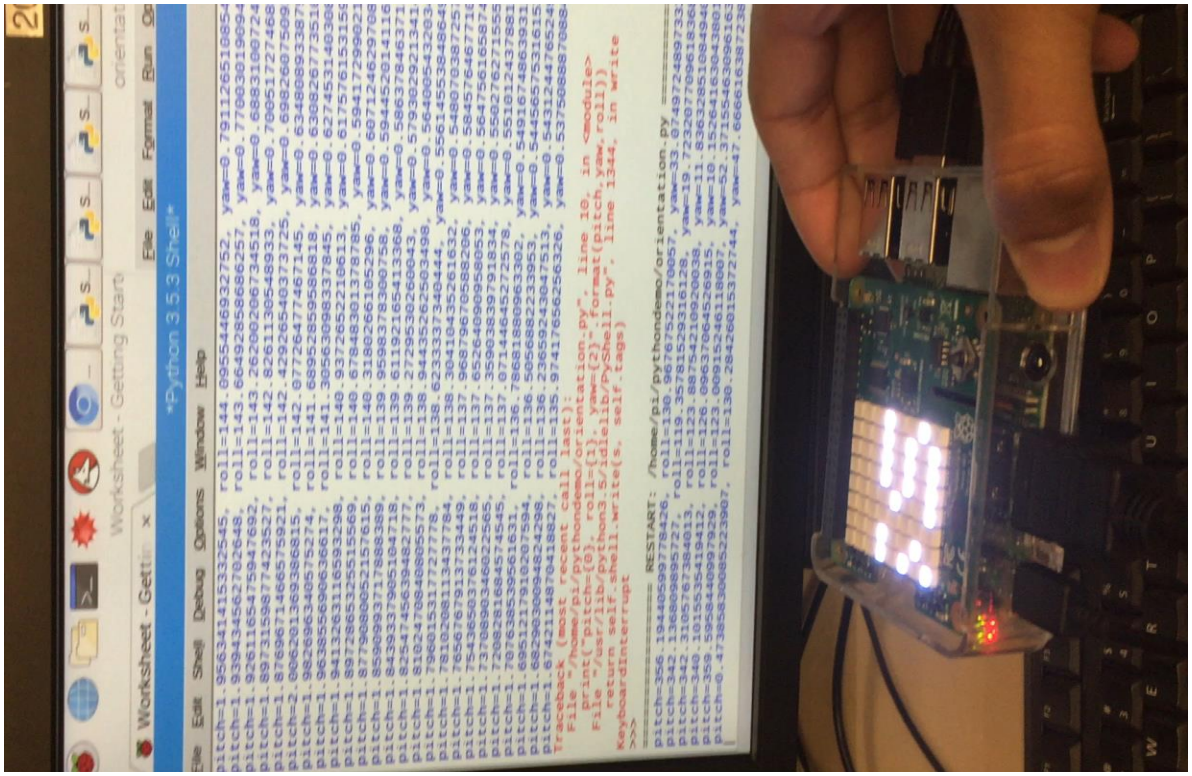
Sense HAT

- **get_orientation()**

```
from sense_hat import SenseHat

sense = SenseHat()

while True:
    orientation = sense.get_orientation()

    pitch = orientation['pitch']
    roll = orientation['roll']
    yaw = orientation['yaw']

    print("pitch={0}, roll={1}, yaw={2}".format(pitch,yaw,roll))
```

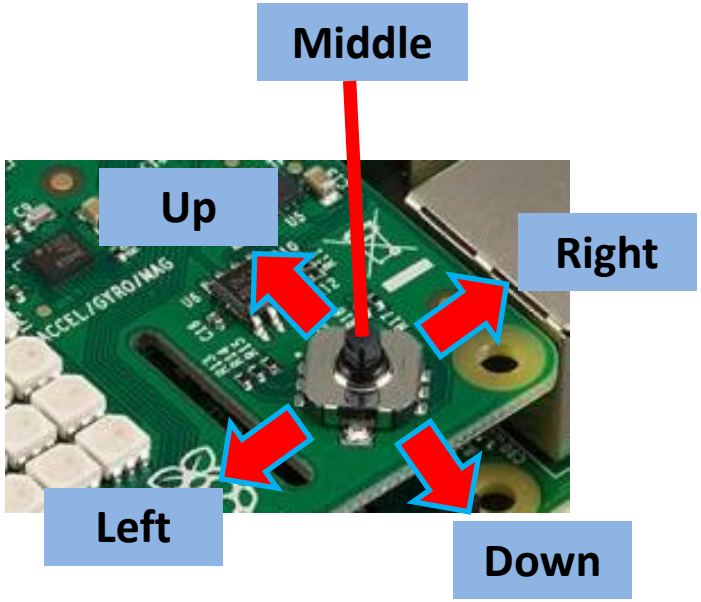
Sense HAT

- **Demo – retrieve gyroscope readings**



Joystick

- **Release/hold/press in five directions (up, down, left, right, middle)**



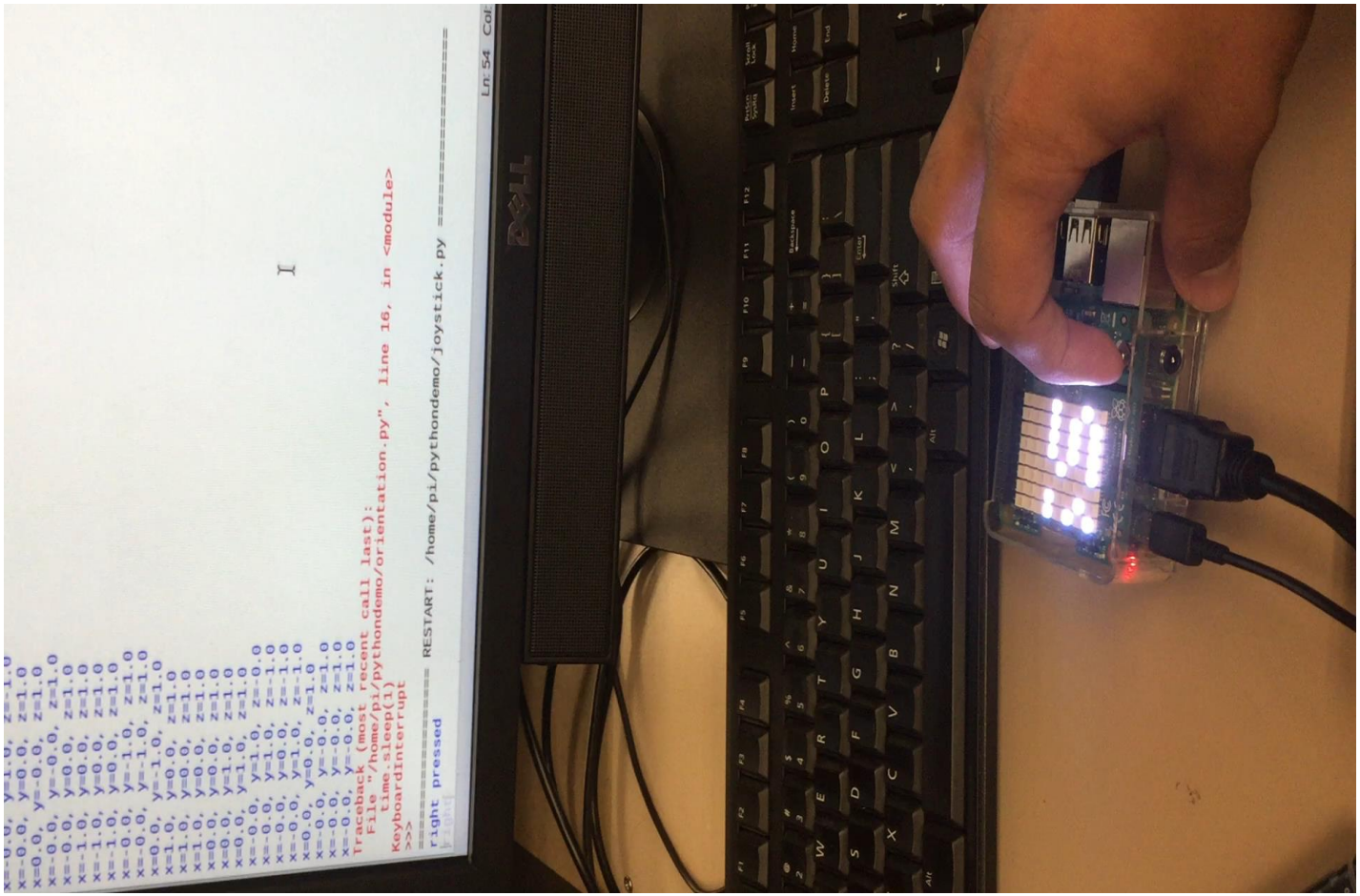
Sense HAT

- **Joystick**

```
from sense_hat import SenseHat

sense = SenseHat()

while True:
    for event in sense.stick.get_events():
        print(event.direction, event.action)
```

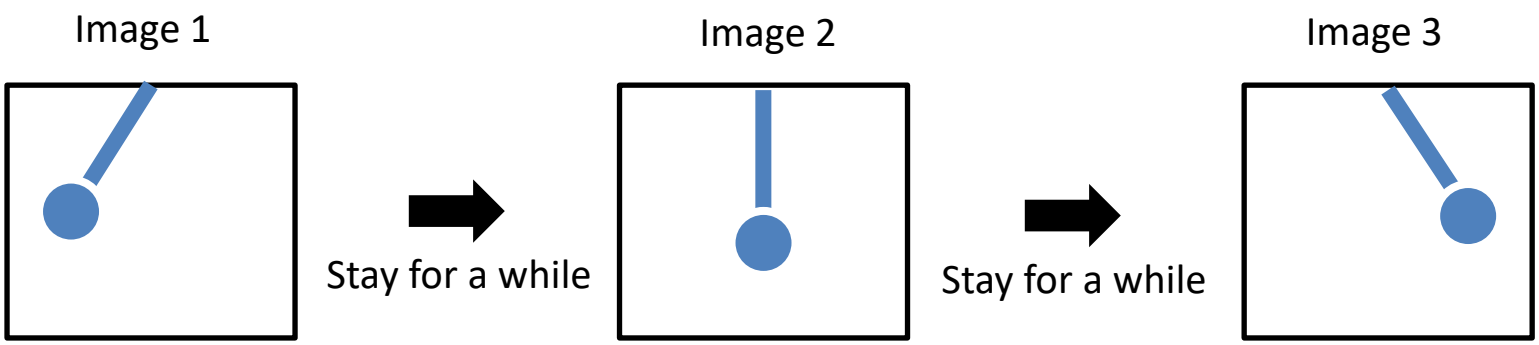
Sense HAT

- **Demo – detect joystick operations**



Sense HAT

- **Lab Task 1 – show simple animation on LED matrix**
  - Hints
    - Loop a series of static images to achieve dynamic
    - set_pixel(), set_pixels(), sleep(), clear()



Sense HAT

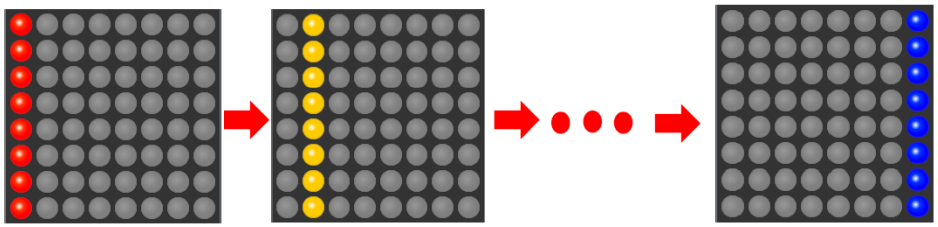
- **Lab Task 1 – demo**

Sense HAT

- **Lab Task 2 – different reactions to five Joystick operations**
  - Example
    - Display different types of sensor readings according to your joystick operations (up, down, right, left, middle)
  - Hints
    - Detect joystick events in an endless loop
    - Use if-elif-else to conduct different reactions
    - stick.get_events()

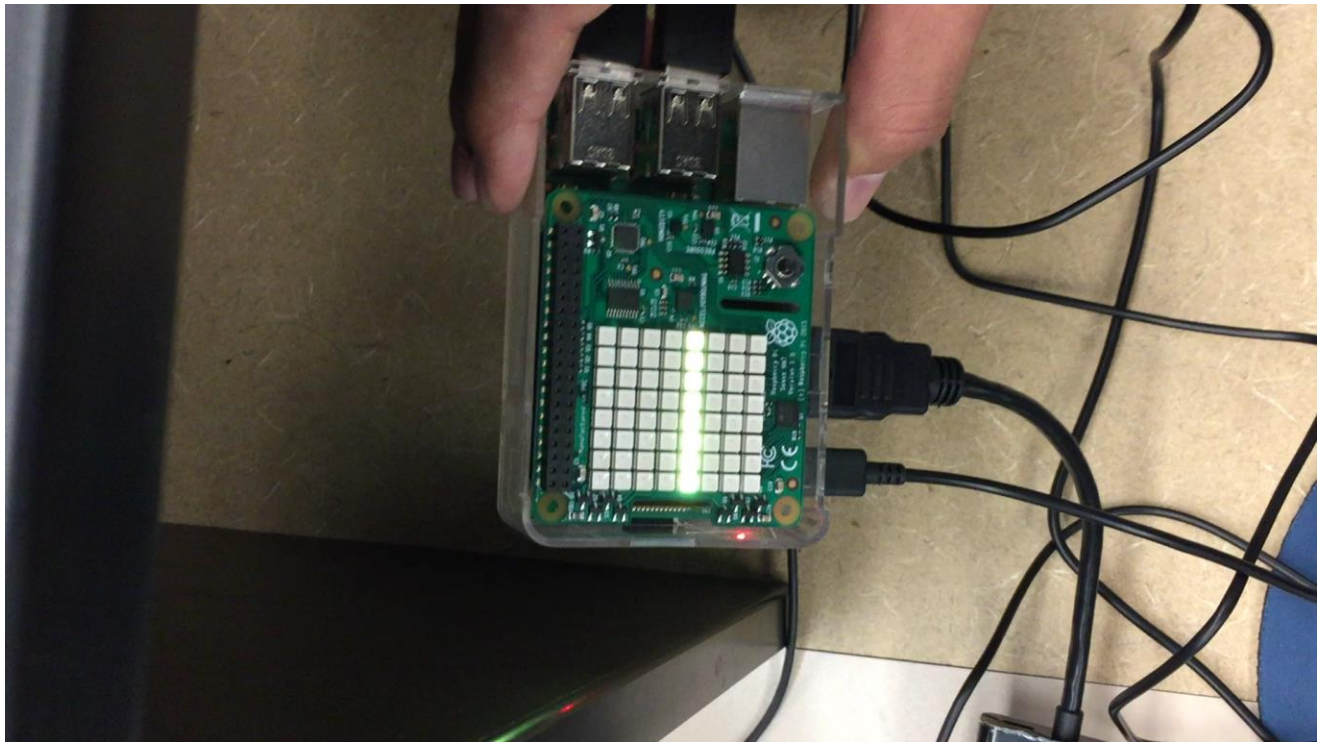
Sense HAT

- **Lab Task 3 – display a moving line on LED matrix**
  - Requirement
    - The moving direction of the line should be the same as tilting direction of your board.
    - Use different colors for each movement
  - Hints
    - Detect tilting directions using accelerometer readings in an endless loop
    - Create moving line animations according to the tilting directions
    - get_accelerometer_raw(), set_pixel(), sleep(), clear(), etc.



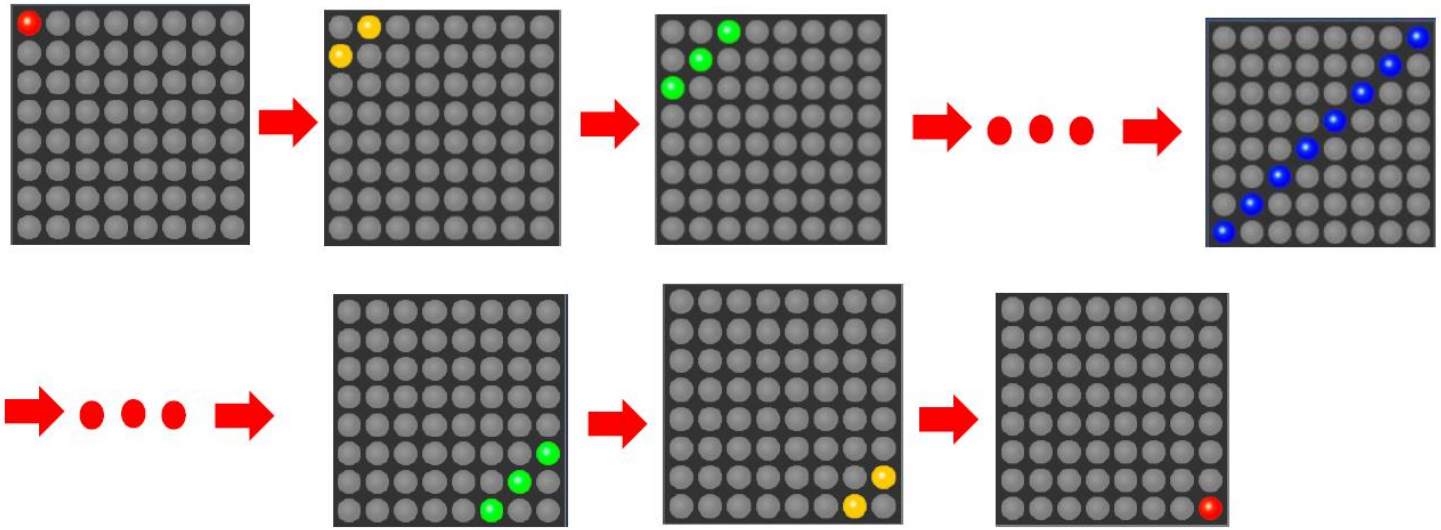
Sense HAT

- **Lab Task 3 – demo**



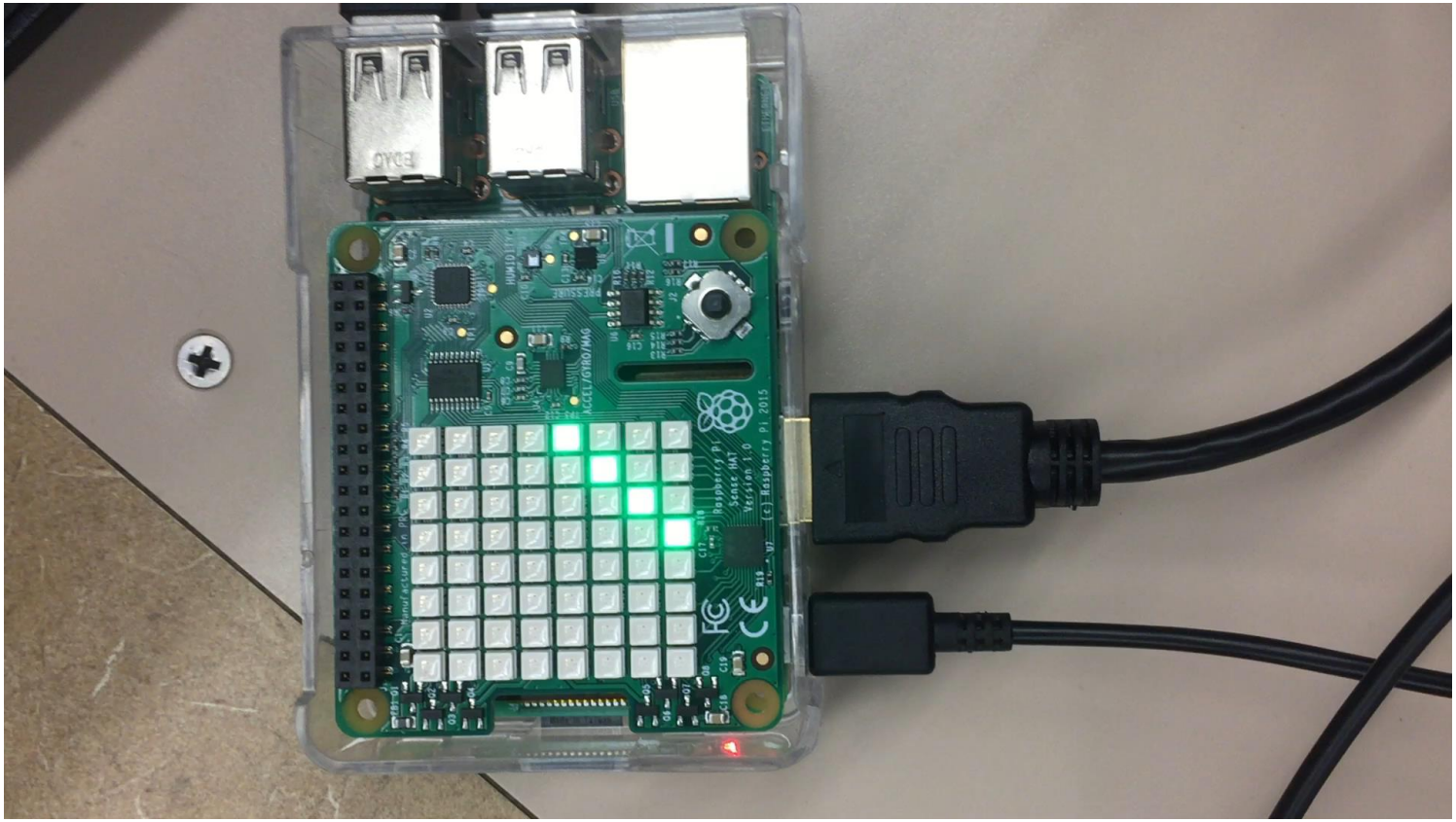
Sense HAT

- **Lab Task 4 – display a moving line on LED matrix**
  - Requirement
    - The line should move along the matrix diagonal.
    - Change your color for each movement



Sense HAT

- **Lab Task 4 – demo**



Useful Website

- You can find extra commands on this website:

https://pythonhosted.org/sense-hat/api/

- Or google sense hat API.

Interrupt

- **Signal**
  - Software interrupts sent to a process
  - Indicate that an important event has occurred
- **What happen when signal received?**
  - Call signal handler (default or specified by users)
- **How to send a signal?**
  - User keystrokes
    - Certain key combinations (e.g., Ctrl + C in terminal can kill a process)
  - Other process
    - Timer will generate an alarm signal when expires (e.g., setitimer())
  - Exception
    - Program failure (division by zero)

Interrupt

- **Signal**
  - Each signal has a numerical code

Signal Name	Signal Number	Description
SIGHUP	1	Hang up detected on controlling terminal or death of controlling process
SIGINT	2	Issued if the user sends an interrupt signal (Ctrl + C)
SIGQUIT	3	Issued if the user sends a quit signal (Ctrl + D)
SIGFPE	8	Issued if an illegal mathematical operation is attempted
SIGKILL	9	If a process gets this signal it must quit immediately and will not perform any clean-up operations
SIGALRM	14	Alarm clock signal (used for timers)
SIGTERM	15	Software termination signal (sent by kill by default)

Interrupt

- **Lab Task 5 – Use C to write a system-call-based interrupt on Raspberry Pi OS.**
  - Hints
    - Set a timer using ***setitimer()***. A SIGALRM signal will automatically be generated in each expiration of the timer.
    - Use ***sigaction()*** to detect the SIGALRM signal and execute your signal handler.
    - Include head files ***sys/time.h*** and ***signal.h***
    - You could print something in the signal handler for checking.
  - For more details
    - https://man7.org/linux/man-pages/man2/setitimer.2.html
    - https://man7.org/linux/man-pages/man2/sigaction.2.html

Interrupt

- **Set the timer**

```
int main ()
{
 struct sigaction sa;
 struct itimerval timer;

 /* Install timer_handler as the signal handler for SIGVTALRM. */
 memset (&sa, 0, sizeof (sa));
 sa.sa_handler = &timer_handler;
 sigaction (SIGVTALRM, &sa, NULL);

 /* Configure the timer to expire after 500 msec... */
 timer.it_value.tv_sec = 0;
 timer.it_value.tv_usec = 500000;
 /* ... and every 500 msec after that. */
 timer.it_interval.tv_sec = 0;
 timer.it_interval.tv_usec = 500000;
 /* Start a virtual timer. It counts down whenever this process is
   executing. */
 setitimer (ITIMER_VIRTUAL, &timer, NULL);

 /* Do busy work. */
 while (1);
}
```

Interrupt

- **Configure sigaction()**

```
int main ()
{
 struct sigaction sa;
 struct itimerval timer;

 /* Install timer_handler as the signal handler for SIGVTALRM. */
 memset (&sa, 0, sizeof (sa));
 sa.sa_handler = &timer_handler;
 sigaction (SIGVTALRM, &sa, NULL);

 /* Configure the timer to expire after 500 msec... */
 timer.it_value.tv_sec = 0;
 timer.it_value.tv_usec = 500000;
 /* ... and every 500 msec after that. */
 timer.it_interval.tv_sec = 0;
 timer.it_interval.tv_usec = 500000;
 /* Start a virtual timer. It counts down whenever this process is
   executing. */
 setitimer (ITIMER_VIRTUAL, &timer, NULL);

 /* Do busy work. */
 while (1);
}
```

Interrupt

- **Signal handler function**

```
void timer_handler (int signum)
{
 static int count = 0;
 struct timeval ts;

 count += 1;
 gettimeofday(&ts, NULL);
 printf ("%d.%06d: timer expired %d times\n", ts.tv_sec, ts.tv_usec, count);
}
```

Write your own handler function

Basics of Programming

- **Edit your code**
  - Vim, notepad++, Geany, etc.
- **Programming habits**
  - Write useful comments
  - Assign meaningful variable/function names
- **How to run (if use command line)**
  - Python (assume use python 3)
    - python3 ***my_code.py***
  - C (compile & run)
    - gcc ***my_code.c*** -o ***my_code***
    - ./***my_code***

Thanks!