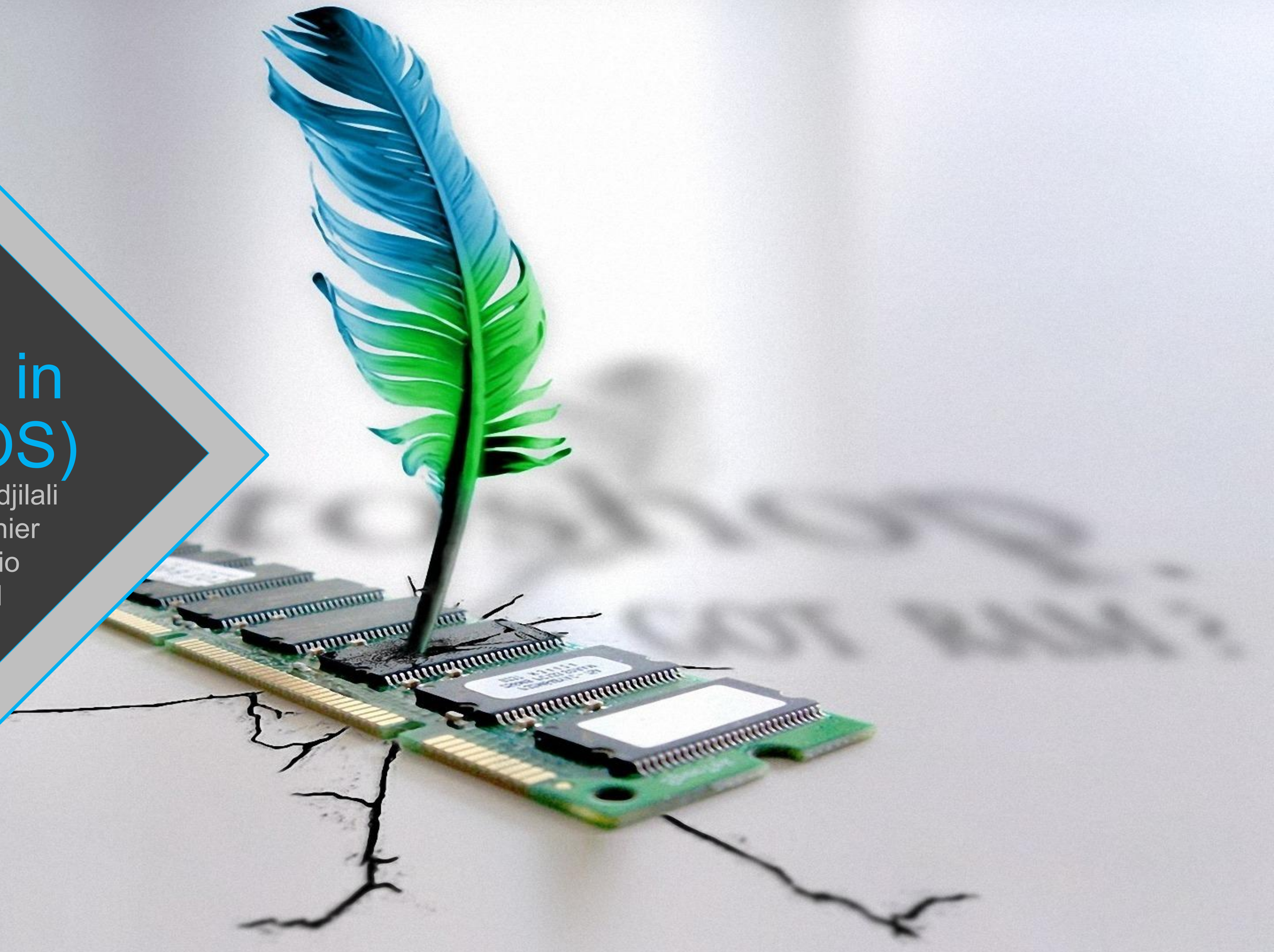
Drawing in the air (OS)

Mohamed-Ali Beldjilali
Yohann Marmonier
Thomas Lieghio
Julien Giraud



Welcome to our presentation

We choose the theme Drawing in the air to unlock an OS.

The main idea is about designing a software able to record movements, to translate it on a 2D space and to compare the pattern obtained with a model stored in memory.

Summary

Part 1 :

Steps of the project

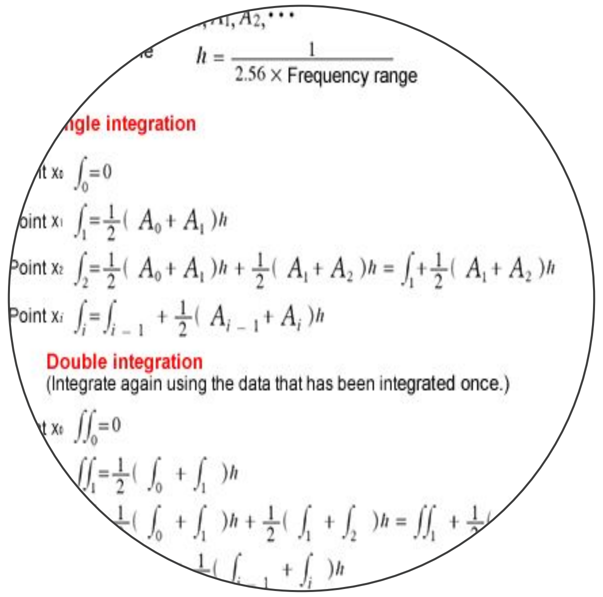
Project’s Main Tasks

All were quite pleasant…

Movement Sensing

Was easy

We need to implement a recording system which would be triggered with a user interaction.



Position Calculation

Quick Maths

We need to get position evolution from the acceleration values acquired from the previous part.

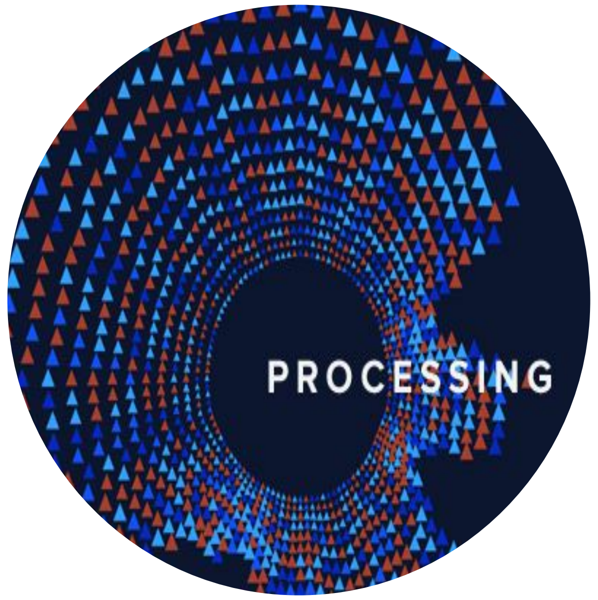


Image Comparison

Blurred and laborious

We now have two images we need to compare in order to determine if the OS should be unlocked or not.

Our group organization

Ain’t no work in off

Thomas

In charge of the work flow : ensure that time is not lost by giving objectives. Managing ressources, synchronizating, testing optimizations.

Julien

Mainly working on the Raspberry system to capture movement with user interaction. Deploy and tests programs on the Raspberry.

Mohammed

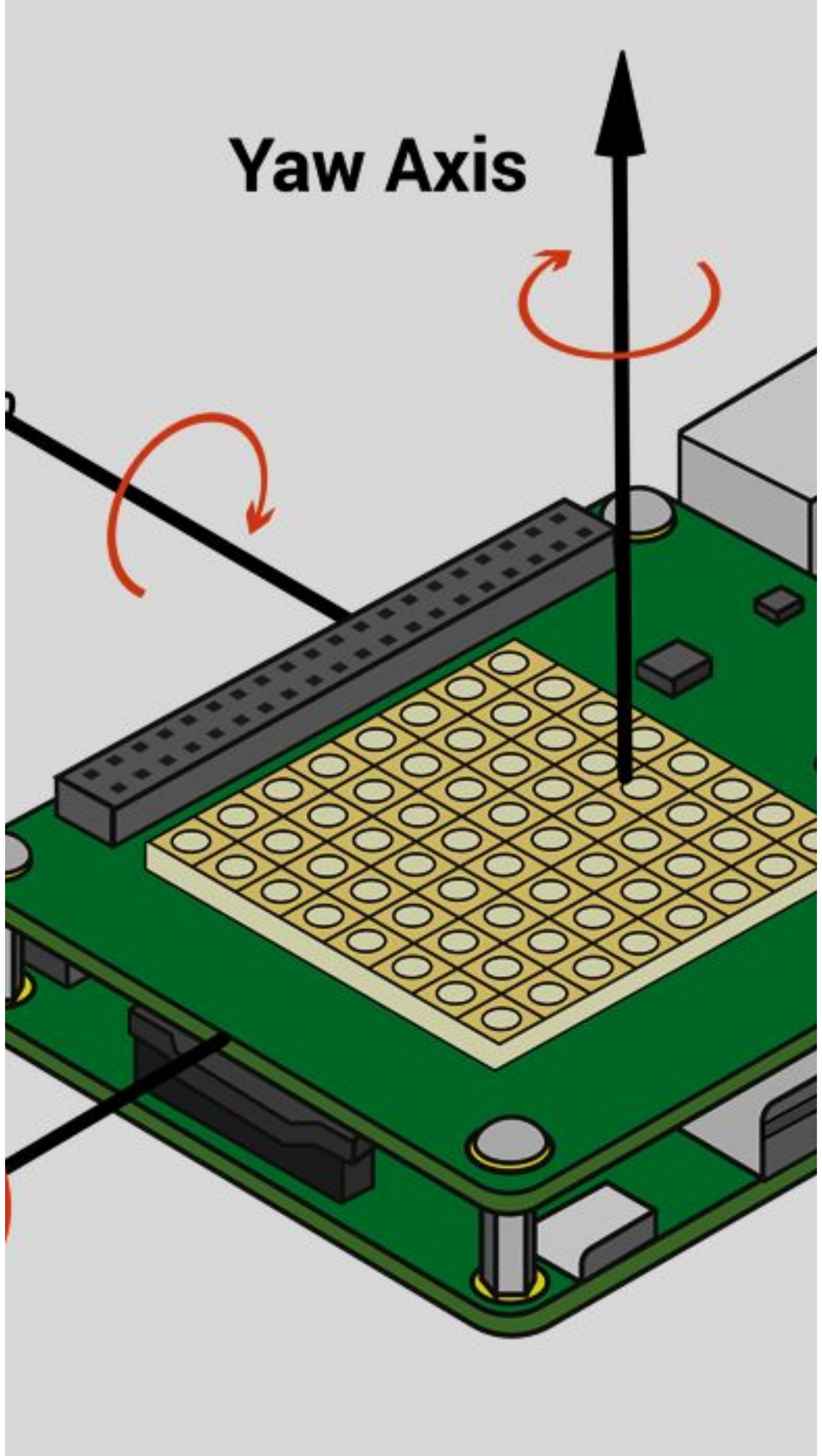
Helping when problems occurs. Doing supporting tasks to help principal tasks to advance. Work on image comparison. In charge of code review.

Yohann

In charge of movement integration and image processing algorithms. Focusing on the main project aim.

Part 2 :

What did we used ?



Movement Sensing : The Sense Hat

- Capting acceleration values on each dimension : x, y and z
- Pressure button to interact with user
- Python libraries to use the hardware

Image generation and processing

Matplotlib
The open source library we used to display drawings

ImageIo
The open source library we used for image i/o

Skimage
An open source library providing processing algorithms implementation

MSE
An algorithm designed to perform a pixel oriented image comparison

SSIM
Structural similarity algorithm

Part 3 :

Explanation of the code

A. Movement Recording

```python
if __name__ == "__main__":
    sense = SenseHat()
    print("Start")
    recording = 0
    event = sense.stick.wait_for_event()
    while True:
```

Beginning of the script...

Print “Start” on the screen

Initializes recording mode

Wait for user input

Enters the loop...

```python
        if event != None and event.action == 'pressed':
            while event.action != 'released':
                event = sense.stick.wait_for_event()
            if recording == 0:
                recording = 1
            elif recording == 1:
                recording = 2
            elif recording == 2:
                recording = 1
```

If we are in Start-up mode :

We wait for user input

If we have press the button, then we change the recording mode :

Start-up → Recording

Recording → Reading

Reading → Start-up again

```python
if recording == 1:
    acceleration = sense.get_accelerometer_raw()
    x = acceleration['x']
    y = acceleration['y']
    z = acceleration['z']
    acceleration_values_x.append(x)
    acceleration_values_y.append(y)
    acceleration_values_z.append(z)
    print("Recording")
```

If we are in recording mode :

We get acceleration values

We add them to a data table

We print “Recording”

```python
if recording == 2: #Yohann's part
```

```python
events = sense.stick.get_events()
try:
    event = events[0]
except:
    event = None
```

We get the actual value of the button, if there is one...

And let’s go back to our ‘While True”

B. Acceleration Integration

$$y = \int v\,dt$$

$$= \int (v_0 + at)\,dt$$

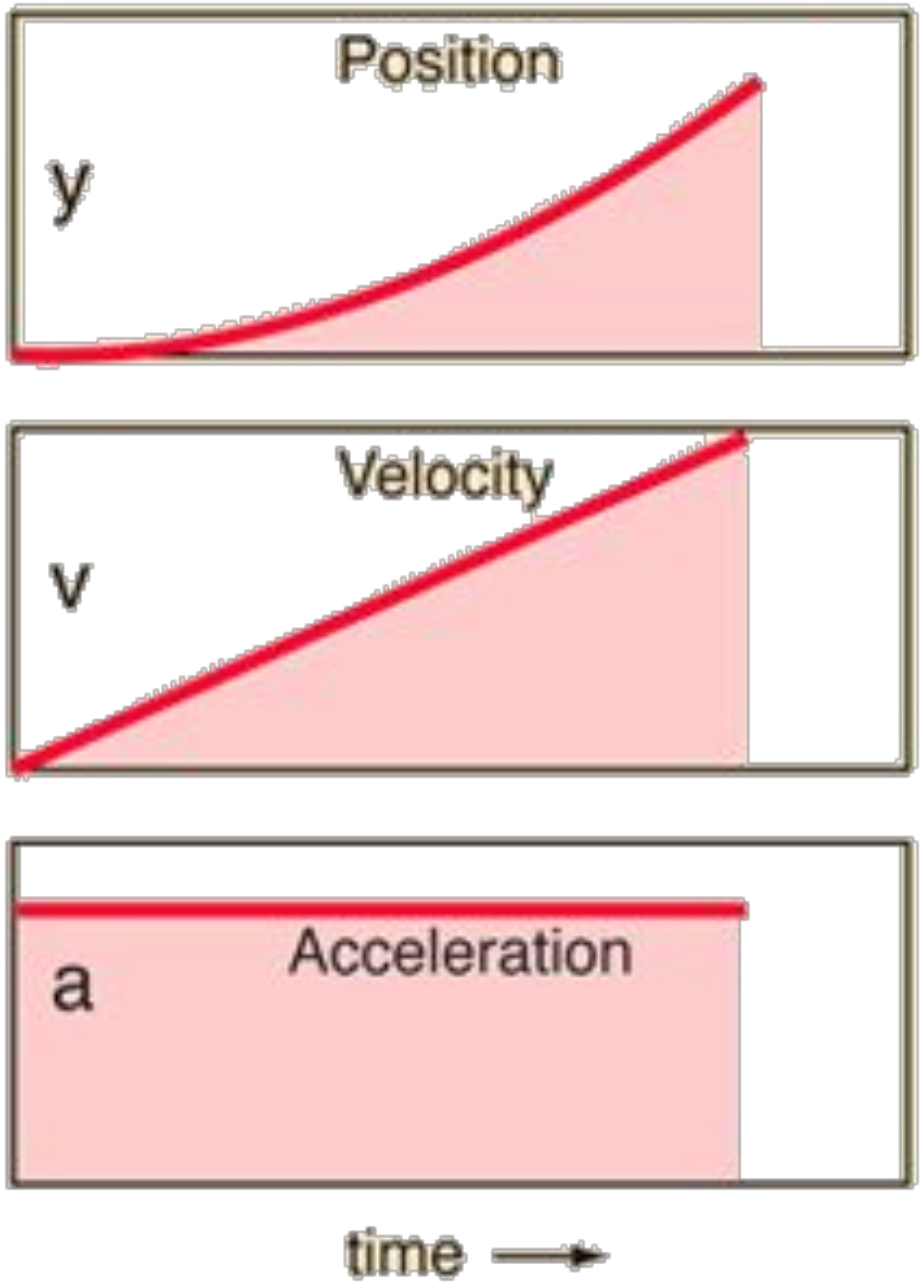
$$y = y_0 + v_0 t + \frac{1}{2} a t^2$$

Integrate velocity to get position

$$v = \int a\,dt = v_0 + at$$

Integrate acceleration to get velocity

$$a = \text{constant}$$



Motion relationships in one dimension.

$$y = y_0 + v_0 t + \frac{1}{2} a t^2$$

Derivative of position is velocity

$$v = \frac{dy}{dt}$$

$$v = v_0 + at$$

Derivitive of velocity is acceleration

$$a = \frac{dv}{dt} = a$$

Instantaneous velocity, is the limit of the average acceleration over an infinitesimal interval of time. In the terms of calculus, instantaneous velocity is the integral of the velocity vector with respect to time:

$$\boldsymbol{v} = \int \boldsymbol{a}\, dt.$$

In our case, time is already infinitesimal. So we can simply approximate the velocity of the movement by calculating the mean of the acceleration values given by the accelerometer. We can reproduce the same process to obtain the position : we have our draw.

We start by collecting acceleration values in a list :

```
acceleration = sense.get_accelerometer_raw()

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

acceleration_values_x.append(x)
acceleration_values_y.append(y)
acceleration_values_z.append(z)
```

And then we calculate the velocity and the position :

```
for i in range(len(acceleration_values_x - 2)):
    velocity_values_x.append((acceleration_values_x[i] + acceleration_values_x[i+1])/2)
    velocity_values_y.append((acceleration_values_y[i] + acceleration_values_y[i+1])/2)
    velocity_values_z.append((acceleration_values_z[i] + acceleration_values_z[i+1])/2)

for i in range(len(velocity_values_x - 2)):
    position_values_x.append((velocity_values_x[i] + velocity_values_x[i+1])/2)
    position_values_y.append((velocity_values_y[i] + velocity_values_y[i+1])/2)
    position_values_z.append((velocity_values_z[i] + velocity_values_z[i+1])/2)
```

We can now use the matplotlib library to draw the resulted positions.

```
displayDrawing(position_values_x, position_values_y, position_values_z)
```

```

def displayDrawing(xData, yData, zData):

    ax = plt.axes(projection='3d')
    ax.plot3D(xData, yData, zData, 'gray')
    plt.show()

```

C. Image Processing

List of the algorithms we used

MSE (Mean Squared Error)

Euclidean Distance

SSIM (Structural SIMilarity)

MSE (Mean Squared Error)

$$MSE = \frac{1}{NxM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}[X(i,j) - Y(i,j)]^2$$

```
def mse(imageA, imageB):

    err = np.sum((imageA.astype("float") - imageB.astype("float")) ** 2)
    err /= float(imageA.shape[0] * imageA.shape[1])

    return err
```

Part 4 :
Difficulties

Difficulties

Python’s libraries problems : compability between python2 and python3

Recording problems : the libraries we had made us able to detect events from button

Problem of the acceleromator

Too big accurency values

To compare the model and the drawing

Converting the 3 acceleration values of 3 axis

Tests : videos

Part 5 :

Tests & conclusion

Drawing a circle (not a cross)

Drawing a cross (not a circle)