# Deliverable Session 04 - Information and Communication Technology

## Rafael Antonio Echevarria Silva

In this session, our goal is to communicate the Sense Hat sensors with the Raspberry Pi using Python.

[https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat/]

### A. Reading the sensor information

We have a total of 8 sensors on the Raspberry Pi, 3 of them environmental and 5 for internal measurements. For the first three, we can see their functions on the previous webpage in the "Sensing the environment" section, where we can see examples of how to measure pressure, temperature, and humidity.

```
sense.get_pressure()
sense.get_temperature()
sense.get_humidity()
```

For the 5 IMU sensors, we can use the "Detecting Movement" section, which explains how the sensors work and how to use them, as well as which variables are returned when calling the function.

```
sense.get_compass()
sense.get_compass_raw() #Units (x, y, z) from the magnetometer
sense.get_gyroscope_raw()
sense.get_accelerometer_raw()
sense.get_orientation()
```

### B. Read sensor information and display it in a nice and appropriate format

In this section, we only want to make the data more visually appealing when printing; we can do this using f-string (formatted string).

### C. Use a loop to read sensor information

In this function, to avoid a ***While True:** loop, we prefer to use a **for i in range n:** loop where "n" is a predefined, limited number of samples.

### D. To calculate the computation time using “print” to show sensor information on the screen

To calculate the processing times using the print functions we had already defined, we import the **time** library to initialize a counter at the beginning of the loop and end it before closing the loop to perform the calculation. We do this by including the following functions:

```
start_time = time.perf_counter() # Start timer
...
end_time = time.perf_counter() # End timer
elapsed_time = end_time - start_time
print(f"Computation Time: {elapsed_time:.4f} seconds")
```

### E. To calculate the computation time without using "print" to show sensor information on the screen

In this case, we only need to comment out, or in our case, delete, the prints we used to display the captured data. We see an improvement of approximately two-tenths of a second.

### F. To obtain the "sample frequency" for each sensor

To calculate the frequency of each sensor, we simply calculate how long it takes to execute and divide 1 by this interval. In other words, we should have a function like this:

```
start_press = time.time() # Start timer
press = round(sense.get_pressure(), 1)
end_press = time.time() # End timer
elapsed_press = end_press - start_press
freq_temp = 1 / elapsed_press if elapsed_press != 0 else float('inf')
print(f"Pressure: {press} hPa, Sample Frequency: {freq_temp:.2f} Hz")
```

### G. To include date information (date & time)

To include date and time information, in this case, we must include the **Datetime** library, which will give us access to The following function can be used to print the resulting time to the screen.

```
timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]
print(f"Timestamp: {timestamp}")
```

### H. To write sensor information in a common ".csv" file

For this section, we need to include the **csv** library, which will provide us with the necessary functions to save all the sensor samples in a common file. To do this we have to incorporate the following functions:

```
import csv
csv_filename = "sensor_data.csv"

with open(csv_filename, mode='w', newline='') as file:
writer = csv.writer(file)
writer.writerow([
"Time", "Temp (C)", "Pressure (hPa)", "Humidity (%)",
"North (°)",
```

```
"Magneto X", "Magneto Y", "Magneto Z",
"Gyro X", "Gyro Y", "Gyro Z",
"Accel X", "Accel Y", "Accel Z",
"Pitch", "Roll", "Yaw"
])

for i in range n:

...

writer.writerow([
timestamp, temp, pres, hum,
north,
round(magneto['x'], 2), round(magneto['y'], 2), round(magneto['z'], 2),
round(gyro['x'], 2), round(gyro['y'], 2), round(gyro['z'], 2),
round(accel['x'], 2), round(accel['y'], 2), round(accel['z'], 2),
round(orientation['pitch'], 2), round(orientation['roll'], 2),
round(orientation['yaw'], 2)
])
```

### I. To write sensor information in a ".csv" file, but using a file for each sensor

Similar to the previous section, before loading the sensor data, we must create the files for each one. This is done by initializing a list of file names and headers, which are created with a for loop, and then loading them with data within the second loop.

### J. To acquire slower samples using the "sleep" function (Sleep parameter: 0.1 sec)

Using the **Time** library that we incorporated in previous sections, we can use the **time.sleep(0.1)** function, which will pause the loop for 0.1 seconds after executing it.

### K. To control the time and measure sensor information every 1 second

To get the loop to take a measurement every 1 second, we need to calculate how long it takes to calculate all the measurements and then calculate the remaining time the loop would have to sleep to make a 1-second difference with the next sample.

```
interval = 1 # seconds

start_time = time.time()

for i in range(n):
current_time = time.time()

...

next_sample_time = start_time + (i + 1) * interval
sleep_time = max(0, next_sample_time - current_time)
time.sleep(sleep_time)
```