CÓDIGO UTILIZADO:

- Para el medidor de temperatura y humedad:

Registrador de datos desde la terminal:

```
import sys
import Adafruit_DHT

while True:
    humidity, temperature = Adafruit_DHT.read_retry(11, 23)
```

Gráfico en tiempo real:

```
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import Adafruit_DHT

sensor = Adafruit_DHT.DHT11
pin = 23

fig = plt.figure()
ax = plt.axes(xlim=(0, 30), ylim=(15, 45))
max_points = 30
line, = ax.plot(np.arange(max_points),
            np.ones(max_points, dtype=float) * np.nan, lw=1, c="blue", marker="d", ms=2)

def init():
    return line

h,t = Adafruit_DHT.read_retry(sensor, pin)

def animate(i):
    h, t = Adafruit_DHT.read_retry(sensor,pin)
    y = t
    old_y = line.get_ydata()
    new_y = np.r_[old_y[1:],y]
    line.set_ydata(new_y)
    return line,

anim = animation.FuncAnimation(fig, animate, init_func=init, frames = 200, interval=20,blit =
False)
plt.show()
```

- Para el medidor de NDVI:

Convertidor de imagen común a imagen con niveles de NDVI:

```
import cv2
import numpy as np
from fastiecm import fastiecm

original = cv2.imread('/home/Luu/3.jpg')

def display(image, image_name):
    image = np.array(image, dtype=float)/float(255)
    shape = image.shape
    height = int(shape[0] / 5) #definen el tamaño
    width = int(shape[1] / 5) #definen el tamaño
    image = cv2.resize(image, (width, height))
    cv2.namedWindow(image_name)
    cv2.imshow(image_name, image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def contrast_stretch(im):
    in_min = np.percentile(im, 5)
    in_max = np.percentile(im, 95)
    out_min = 0.0
    out_max = 255.0
    out = im - in_min
    out *= ((out_min - out_max) / (in_min - in_max))
    out += in_min
    return out

def calc_ndvi(image):
    b, g, r = cv2.split(image)
    bottom = (r.astype(float) + b.astype(float))
    bottom[bottom==0] = 0.01
    ndvi = (b.astype(float) - r) / bottom
    return ndvi

display(original, 'Original')
contrasted = contrast_stretch(original)
display(contrasted, 'Contrasted original')
cv2.imwrite('contrasted.png', contrasted)
ndvi = calc_ndvi(contrasted)
display(ndvi, 'NDVI')
ndvi_contrasted = contrast_stretch(ndvi)
display(ndvi_contrasted, 'NDVI Contrasted')
cv2.imwrite('ndvi_contrasted.png', ndvi_contrasted)
color_mapped_prep = ndvi_contrasted.astype(np.uint8)
color_mapped_image = cv2.applyColorMap(color_mapped_prep, fastiecm)
display(color_mapped_image, 'Color mapped')
cv2.imwrite('color_mapped_image.png', color_mapped_image)
```

Cámara que mide en tiempo real el nivel de vida de la planta:

```
import cv2
import numpy as np
from fastiecm import fastiecm

class FastieColorMap(object):
    def apply_color_map(self, ndvi_frame):
        normalized_ndvi = (ndvi_frame + 1) / 2
        color_mapped_prep = (normalized_ndvi * 255).astype(np.uint8)
        color_mapped_image = cv2.applyColorMap(color_mapped_prep, fastiecm)
        return color_mapped_image

class SimulatedNDVI(object):
    def convert(self, frame):
        blue = frame[:, :, 0].astype('float')
        red = frame[:, :, 2].astype('float')
        bottom = (blue + red)
        bottom[bottom == 0] = 1  # avoid division by zero
        sim_ndvi = (red - blue) / bottom
        return sim_ndvi

def main():
    simulated_ndvi = SimulatedNDVI()
    fastie_color_map = FastieColorMap()
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Error: Couldn't capture frame")
            break
        resized_frame = cv2.resize(frame, (720, 600))
        sim_ndvi_frame = simulated_ndvi.convert(resized_frame)

        color_mapped_image = fastie_color_map.apply_color_map(sim_ndvi_frame)
        cv2.imshow('Fastie Color Map', color_mapped_image)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```