2024

IoT Task Sensor Data Acquisition and Visualization

JOLIANA EMAD KAMAL NAGUIB

SAMSUNG INNOVATION CAMPUS

TASK CH4

Task Assignment

Task: Sensor Data Acquisition and Visualization

Objective:

* Design and implement a circuit utilizing an ultrasonic and temperature sensor.

* Record sensor readings to a log file.

* Generate a plot displaying both sensor readings over time.

Deliverables:

* A well-documented code file.

* A log file containing sensor data.

* A generated plot visualizing both sensor readings.

Python file:

VM2 - VMware Workstation 17 Player (Non-commercial use only)

Player

Activities Terminal

ubuntu

GNU nano 6.2

```python
import RPi.GPIO as GPIO
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from datetime import datetime, date
import serial
import csv
import time

GPIO.setmode(GPIO.BCM)
TRIG = 23
ECHO = 24
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

esp_board = serial.Serial(port='COM4', baudrate=115200, timeout=.1)

today = date.today()
current_date = today.strftime("%b-%d-%Y")
print("Date: ", current_date)

dates = []
distances = []
temperatures = []

temp_thresh_high = 30
temp_thresh_low = 20
dist_thresh_high = 60
dist_thresh_low = 30

Sensor_log_file = f"sensor_log_{current_date}.csv"
with open(Sensor_log_file, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["Timestamp", "Distance (cm)", "Temperature (C)"])

fig, (dist_plot, temp_plot) = plt.subplots(1, 2, figsize=(12, 6))
```

VM2 - VMware Workstation 17 Player (Non-commercial use only)

Player

Activities Terminal

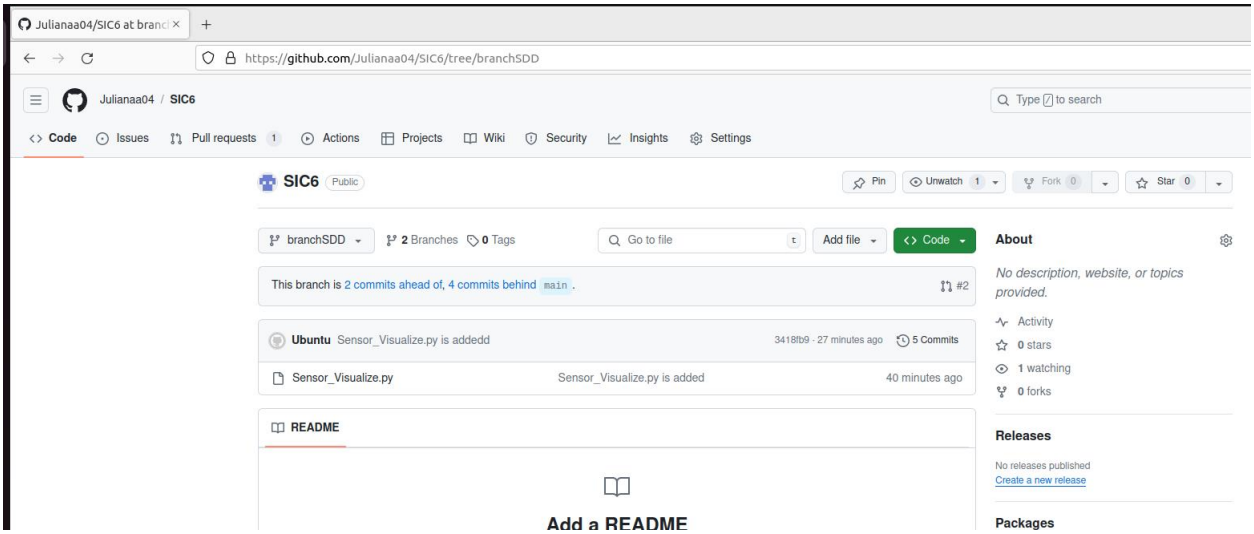
ubur

GNU nano 6.2

```
fig, (dist_plot, temp_plot) = plt.subplots(1, 2, figsize=(12, 6))

def measure_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.5)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = (pulse_duration * 34300) / 2
    return distance

def read_temperature():
    data = esp_board.readline().decode('utf-8').rstrip()
    if data:
        try:
            temperature = float(data)
            return temperature
        except ValueError:
            print("Invalid data received:", data)
            return None
    return None

def add_new_data(frame):
    distance = measure_distance()
    temperature = read_temperature()

    if distance is not None and temperature is not None:
        now = datetime.now()
        dates.append(now)
        distances.append(distance)
        temperatures.append(temperature)

        with open(Sensor_log_file, 'a') as file:
            writer = csv.writer(file)
            writer.writerow([now.strftime('%Y-%m-%d %H:%M:%S'), distance, temperature])

    dist_plot.cla()
    temp_plot.cla()
```

```
    dist_plot.cla()
    temp_plot.cla()

    dist_plot.set_title('Distance (cm)')
    dist_plot.set_xlabel('Time')
    dist_plot.set_ylabel('Distance (cm)')
    dist_plot.plot(dates, distances, 'b-', label='Distance')

    temp_plot.set_title('Temperature (°C)')
    temp_plot.set_xlabel('Time')
    temp_plot.set_ylabel('Temperature (°C)')
    temp_plot.plot(dates, temperatures, 'r-', label='Temperature')

    temp_plot.axhline(y=temp_thresh_high, color='r', linestyle='--', label='High Temp Threshold')
    temp_plot.axhline(y=temp_thresh_low, color='r', linestyle='--', label='Low Temp Threshold')
    dist_plot.axhline(y=dist_thresh_high, color='r', linestyle='--', label='High Distance Threshold')
    dist_plot.axhline(y=dist_thresh_low, color='r', linestyle='--', label='Low Distance Threshold')

    fig.autofmt_xdate()
    fig.suptitle(f'Sensor Data on {current_date}', fontsize=16)
    dist_plot.legend(loc="best")
    temp_plot.legend(loc="best")

ani = FuncAnimation(fig, add_new_data, interval=1700)
plt.show()

GPIO.cleanup()
```

A New BranchSDD is created , Add Sensor_Visualize.py.



Open a pull request