Embedded Systems Workshop

Course No.	EC3.202
Term	Monsoon 2023
Group No.	8
Team Name:	*int_elligence*
Team Members:	Prakhar Singhal (2022111025)
	Chetan Mahipal (2022101096)
	Archit Pethani (2022101098)
	Priet Ukani (2022111039)

1 Hardware Progress

- Integrated the circuit to amplify and record the microphone signals on the PCB.
- Integrated the circuit for the flex sensor on the PCB.
- Shifted the health sensor to a different ESP32 for ease of use.

2 Software Progress

- Completed a Python script for plotting basic graphs on collected data (RAW as well as PROCESSED signals) and setting up a basic Neural network (data yet to be imported) to 'measure' knee health of the user based on previous heuristics.
- Started with the user interface (UI) part of the project.

```

# Model to preict Knee Health Based on EMG, pressure and sound data
import flask as fl
import requests as rq
import json as js
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import spectrogram
import pywt
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam


# get data through GET request from OM2M server

url_sound = "http://192.168.137.1:5089/~/in-cse/in-name/AE-TEST/Microphone/"
url_pressure = "http://192.168.137.1:5089/~/in-cse/in-name/AE-TEST/Peizo_Sensor/"
url_health = "http://192.168.137.1:5089/~/in-cse/in-name/AE-TEST/Health_Sensor/"
url_flex = "http://192.168.137.1:5089/~/in-cse/in-name/AE-TEST/Flex_Sensor/"
url_emg = "http://192.168.137.1:5089/~/in-cse/in-name/AE-TEST/EMG_Sensor/"

```

```python
payload = {}
headers = {
  'X-M2M-Origin': 'admin:admin',
  'Accept': 'application/json'
}

sound_data = rq.request("GET", url_sound, headers=headers,
    data=payload)
pressure_data = rq.request("GET", url_pressure, headers=
    headers, data=payload)
health_data = rq.request("GET", url_health, headers=headers,
    data=payload)
emg_data = rq.request("GET", url_emg, headers=headers, data=
    payload)
flex_data = rq.request("GET", url_flex, headers=headers, data
    =payload)

print(sound_data)
print(pressure_data)
print(health_data)
print(emg_data)
print(flex_data)

# extract data from json file


# wavelet transform for emg data
# wavelet = 'db4'
# level = 4

# coeffs = pywt.wavedec(emg_data, wavelet, level=level)

# make a spectrogram for the sound data

sample_rate = 20 # Hz
nfft = 256 # number of samples per window
noverlap = 128 # number of samples that overlap between
    windows
window = "hamming"

freq, time, Sxx = spectrogram(sound_data, fs=sample_rate,
    nfft=nfft, noverlap=noverlap, window=window)

# display graphs for all datas extracted

# Raw Sound Data Graph
plt.figure(figsize=(10, 6))
plt.plot(sound_data)
plt.xlabel('Time (s)')
```

```
plt.ylabel('Voltage (mV)')
plt.title('Raw Sound Data')
plt.colorbar(label='mV')
plt.show()


# Processed Sound Data Graph
plt.figure(figsize=(10, 6))
plt.pcolormesh(time, freq, 10 * np.log10(Sxx), shading='auto'
    )
plt.colorbar(label='dB')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.title('Spectrogram of Voltage Data')
plt.show()


# make a graph for the pressure data

threshold = 3000 # threshold for pressure sensor
digital_pressure = [max(pressure_data) if pressure >
    threshold else 0 for pressure in pressure_data] # digital
    pressure data

plt.figure(figsize=(10, 6))
plt.plot(digital_pressure)
plt.plot(pressure_data)
plt.legend(['Digital Threshold Reading', 'Analog Pressure'])
plt.xlabel('Time (s)')
plt.ylabel('Pressure (mV)')
plt.title('Pressure Data')
plt.colorbar(label='mV')
plt.show()


# make a graph for the Raw-EMG data

plt.figure(figsize=(10, 6))
plt.plot(emg_data)
plt.xlabel('Time (s)')
plt.ylabel('Voltage (mV)')
plt.title('EMG Data')
plt.colorbar(label='mV')
plt.show()

# make a graph for the Processed-EMG data
for i in range(1, level+2):
    plt.subplot(level+2, 1, i+1)
    plt.plot(coeffs[i-1])
```

```python
    plt.title(f'Detail {i}' if i < level+1 else '
    Approximation')

plt.tight_layout()
plt.show()

# make a graph for the flex sensor data

plt.figure(figsize=(10, 6))
plt.plot(flex_data)
plt.xlabel('Time (s)')
plt.ylabel('Voltage (mV)')
plt.title('Flex Sensor Data')
plt.colorbar(label='mV')
plt.show()


# make health data graph

heart_rate = health_data[2]
blood_oxygen = health_data[3]
sys_pressure = health_data[0]
dia_pressure = health_data[1]

plt.figure(figsize=(10, 6))
plt.plot(heart_rate)
plt.xlabel('Time (s)')
plt.ylabel('Heart Rate (bpm)')
plt.title('Heart Rate')
plt.colorbar(label='bpm')
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(blood_oxygen)
plt.xlabel('Time (s)')
plt.ylabel('Blood Oxygen (%)')
plt.title('Blood Oxygen')
plt.colorbar(label='%')
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(sys_pressure)
plt.plot(dia_pressure)
plt.legend(['Systolic', 'Diastolic'])
plt.xlabel('Time (s)')
plt.ylabel('Blood Pressure (mmHg)')
plt.title('Blood Pressure')
plt.colorbar(label='mmHg')
plt.show()
```

```python



# make a neural network that takes in (EMG, pressure, sound) and predicts the knee health in score of 100

# Generating random input data
# You can replace this with your actual input data
np.random.seed(42)
X = np.random.rand(100, 3)  # 100 samples, 3 features

# Generating random output data within the range of 0-100
y = np.random.uniform(0, 100, 100)

# Creating a neural network model
model = Sequential()
model.add(Dense(8, input_dim=3, activation='relu'))  # 8 neurons in the hidden layer
model.add(Dense(1, activation='linear'))  # Output layer with linear activation

# Compiling the model
model.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=0.001))

# Training the model
model.fit(X, y, epochs=50, batch_size=10, verbose=1)
```

Listing 1: DataModel.py

3 GitHub Repository

Link to Our GitHub Repository