Assignment 1

Team members: Benedetta Pacilli, Valentina Pieri, Parisa Sokuti

Industry 4.0 - Assignment 1

Task 1

a)

active BLE connection = 0.6 mA
accelerometer at 12.5 Hz = 0.35 mA
100% CPU utilization = 4.00 mA

during the 5 ms of ANN execution:
Total Current: 0.6 mA + 0.35 mA + 4.00 mA = 4.95 mA
Energy: 0.00495 A * 3 V = 0.01485 W

during 1.6 s of rest:
Total Current: 0.6 mA + 0.35 mA = 0.95 mA
Energy: 0.00095 A * 3 V = 0.00285 W

b)

Battery capacity = 250 mAh
Battery voltage = 3V
Battery energy capacity -> 0.25 Ah * 3V = 0.75 Wh

(1 Wh is 3600 Joules)
Energy consumption of a battery -> 0.75 Wh * 3600 J/Wh = 2700 Joules

Energy consumption of one gesture classification -> 0.0025 W * 1.6 s = 0.004 Joules

Estimated number of times the gesture classification can be ran on the battery -> 2700 Joules / 0.004 Joules = 675.000 times

c)

A discrete-time power trace records the power consumed by a device at specific time intervals. The digital signal tells us whether a task is running or not. From the power trace we can retrieve the power consumed during the execution time interval of the task; while the digital signal tells when the task starts and ends, enabling us to calculate the execution time. With the power consumed by the task and the task's execution time we can calculate the

energy consumption of the task:
Energy(Joules) = Power(Watt) * Time Interval(s)

Task 2

```
i = 0;

var interval = setInterval(function() {
  var accel = Puck.accel().acc;
  if (i < 20)
  {
    var x = Math.floor(((accel.x-(-32768))/(32767-(-32768)))*(127-(-127))+
(-127));
    var y = Math.floor(((accel.y-(-32768))/(32767-(-32768)))*(127-(-127))+
(-127));
    var z = Math.floor(((accel.z-(-32768))/(32767-(-32768)))*(127-(-127))+
(-127));
    var sample = { x: x, y: y, z: z };
    print("Sample number " + i + ": ", sample);
    Infxl.insert(i, x, y, z);
    i++;
  }
  else
  {
    print("Classification result: " + Infxl.model());
    i = 0;
  }
}, 500);
```

Task 3

```
NRF.findDevices(function(devices) {
  print(devices);
}, { filters: [{ services: ['6e400001-b5a3-f393-e0a9-e50e24dcca9e'] }],
timeout: 2000, active:true });

i = 0;

setInterval(function() {
    if (i == 0){
      sample = "";
    }
    var accel = Puck.accel().acc;
    if (i < 20)
    {
      var x = ((accel.x-(-32768))/(32767-(-32768)))*(127-(-127))+(-127);
      var y = ((accel.y-(-32768))/(32767-(-32768)))*(127-(-127))+(-127);
```

```
      var z = ((accel.z-(-32768))/(32767-(-32768)))*(127-(-127))+(-127);
      if (i == 19) {
        sample += x + "," + y + "," + z;
      } else {
        sample += x + "," + y + "," + z + ",";
      }
      i++;
    }
    else {
      print("Ready");
    }
  }, 500);

function getData() {
  i = 0;
  return sample;
}
```

```
<html>
    <head>
    </head>
    <script src="model.js"></script>
    <script src='http://www.espruino.com/js/uart.js'></script>
    <body>
        <script>

            function extractDataFromResponse(response){
                if(response ≠ null){
                    return response.split(",").map(str ⇒ parseInt(str));
                }
            }

            // function doClassification(data){
            //     console.log(data, data.length);
            //     classify(data)
            //     // var [p1, p2, p3] = classify(data)
            //     // document.getElementById("result").innerHTML =
"Classifer output: " + p1 + ", " + p2 + ", " + p3;
            // }

                function getData() {
                UART.eval('getData()', function(response) {
                    document.getElementById("sensordata").value = response;
                    UART.close();
                    var data = extractDataFromResponse(response);
                    //makeChart(data);
                    var [p1, p2, p3] = classify(data);
                    document.getElementById("result").innerHTML = "Classifer
output: " + p1 + ", " + p2 + ", " + p3;
```

```
                });
                }
            // function copy() {
            //     var copyText = document.getElementById("sensordata");
            //     copyText.select();
            //     copyText.setSelectionRange(0, 99999); /* For mobile
devices */
            //     navigator.clipboard.writeText(copyText.value);
            //     alert("Copied the text: " + copyText.value);
            // }

        </script>

        <div>
            <button style="margin: 20px;" onclick='getData()'>Retrieve and
process data</button>
            <input style="margin: 20px; width: 500px" type="text" value=".."
id="sensordata">
            <!— <button onclick="copy()">Copy</button> —>
        </div>
        <div>
            <label style="margin: 20px; width: 500px" id="result">Classifer
output: ... </label>
        </div>
    </body>
</html>
```

Task 4

```
average_power = np.mean(P_app) # P_app = 1 * V_app * I_app

for start_index, end_index in zip(app_start_index, app_end_index):
    if app_active_window[start_index]:
        task_window = slice(start_index, end_index)


energy_consumption = np.sum(P_app[task_window] * dt_app[task_window])
average_power, energy_consumption

print(f"Average Power Consumption: {average_power}")
print(f"Average Energy Consumption: {energy_consumption}")
```

On-board data

Average Power Consumption: 0.0026328308021782757
Average Energy Consumption: 6.195104320907516e-05

Off-board data

Average Power Consumption: 0.002731176085921592
Average Energy Consumption: 0.0007340946255344708

Task 5

Similarities:

- Both versions involve sampling data from the accelerometer.
- Both versions utilize an Artificial Neural Network (ANN) model for classification.
- Both versions incur some energy overhead for data transmission and processing.

Differences:

- In the on-board version, the ANN model is executed locally on the Puck.js device, while in the off-board version, the data is sent to a web application via Web Bluetooth, and the ANN model is executed on the pc, an off-board device.
- The off-board version incurs higher energy consumption, primarily due to the data transmission over Bluetooth and the computational overhead on the off-board device.
- The on-board version consumes less energy as it performs all computations **locally**, reducing the need for data transmission and off-board processing.

Overheads in Energy:

- In the off-board version, the biggest overhead in terms of energy is the data transmission over Bluetooth. Bluetooth communication itself consumes significant power.

Power Optimization:

To maximize battery lifetime, several power optimization techniques could be implemented:

1. **Data Compression Techniques**: Compressing the data transmitted over Bluetooth to decrease data size, thus reducing energy consumption during transmission.
2. **Low-Power Modes**: Switching to low-power modes of the Bluetooth module and microcontroller when not actively transmitting or processing data.

Trade-offs:

- The on-board version generally consumes **less energy** due to **localized** processing, while the off-board version incurs higher energy consumption due to **data transmission** and off-board processing.
- Unlike the on-board version, the off-board one may introduce **higher latency** due to the time taken for **data transmission** and processing on the off-board device.

- The off-board version requires additional hardware (e.g., a computer or server for the off-board device) and software infrastructure (the web application) compared to the on-board version

In conclusion, while the on-board version offers lower energy consumption and potentially lower latency, the off-board version provides flexibility and scalability by offloading processing to external devices. The choice between the two approaches depends on specific requirements such as energy constraints, latency sensitivity, and system complexity.