Hardware 2 Project Evaluation criteria

Metropolia University of Applied Sciences 2024

This document introduces the evaluation criteria for the technical properties of the Hardware 2 project. In addition to these technical requirements, there are also other factors that affect your grade. These include the quality of your project report and code. Furthermore, self and peer evaluation results may affect the personal project grade.

This document also includes examples of how the user interface of the device could be constructed on the device, including the OLED display. Please note that these example UI images are merely examples, and you are free to design the look and use of your user interface in a different way.

Minimum requirements (grade 1-2)

- The system must work as a standalone system.
- The demo must start with an empty system without any libraries installed.
- The user interface on the OLED screen must have a single-page menu.
- The device must have an option to start and stop the measurement when the user wants.
- The device must update and display the updated BPM value every 5 seconds.
- The calculated BPM values must be possible human heart rate values.
- The heart rate values must be within +-10% when compared to a reference measurement using a pulse oximeter (SpO2 device).

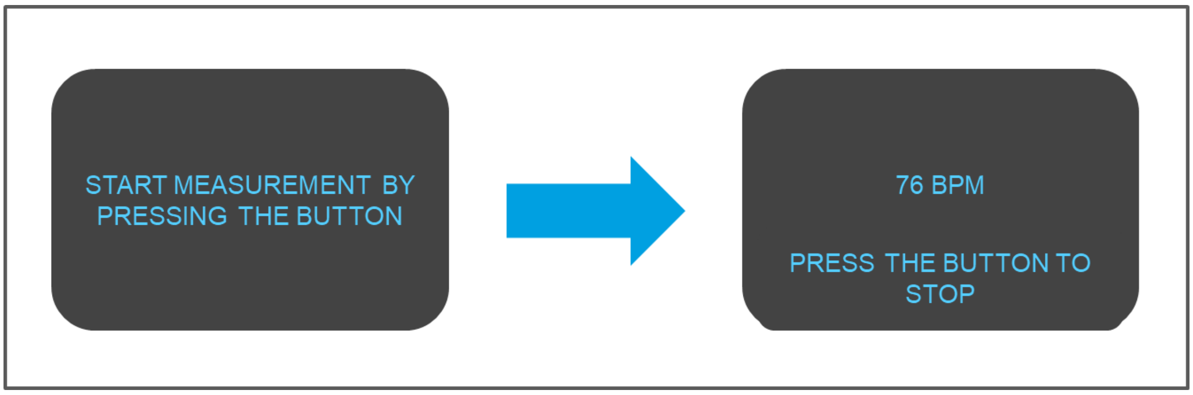

Figure 1: UI example for minimum requirements

Second level requirements (grade 3)

- All minimum level requirements must be filled.
- The device must have a graphical menu with options to choose from.
- Menu option 1 is for displaying the heart rate the same way as in the minimum requirements.
- Menu option 2 is for showing basic HRV analysis calculated locally on the Pico board:
    - mean PPI
    - mean HR
    - RMSSD
    - SDNN
- For the HRV analysis, the device must capture at least 30 seconds of data.
- The device must connect to the group's Wi-Fi network and send messages to a client laptop over MQTT.
- The MQTT broker runs on the Raspberry Pi computer.
- Basic HRV analysis information is shown on the client laptop.

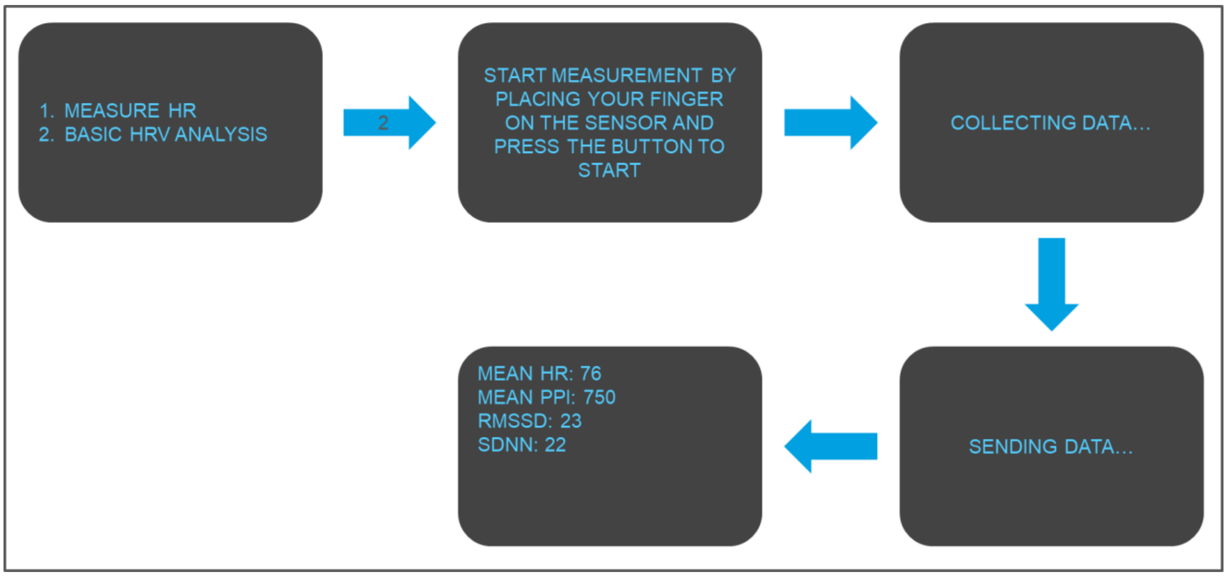

Figure 2: UI example for second level requirements

Example code for connecting to the Wi-Fi network and publishing MQTT messages on the Pico board can be found here: https://gitlab.metropolia.fi/saanapi/hardware-1-networks-final-assignment.

To send the measurement data over MQTT, the data must first be constructed into a dictionary:

```
measurement = {
    "mean_hr": mean_hr,
    "mean_ppi": mean_ppi,
    "rmssd": rmssd,
    "sdnn": sdnn
}
```

The dictionary must then be converted into a JSON string to be able to send the data in a single message. To use the JSON commands, you must import the uJSON library:

```
import ujson
```

And here is the conversion:

```
json_message = measurement.json()
```

The JSON string can then be published to MQTT like any other message by specifying a topic.

Third level requirements (grade 4-5)

- All minimum and second level requirements must be filled.
- The menu must have four options:
  - Measure heart rate
  - Basic HRV analysis
  - History
  - Kubios
- The history option must show a history of previous measurements. At least one previous measurement must be stored with a timestamp.
- The device must show a live PPG signal when measuring heart rate with option 1. (Not shown in the example below).
- When the Kubios option is selected, the device must collect data for at least 30 seconds, send the data to Kubios Cloud, wait for the results and display the results on the OLED screen.

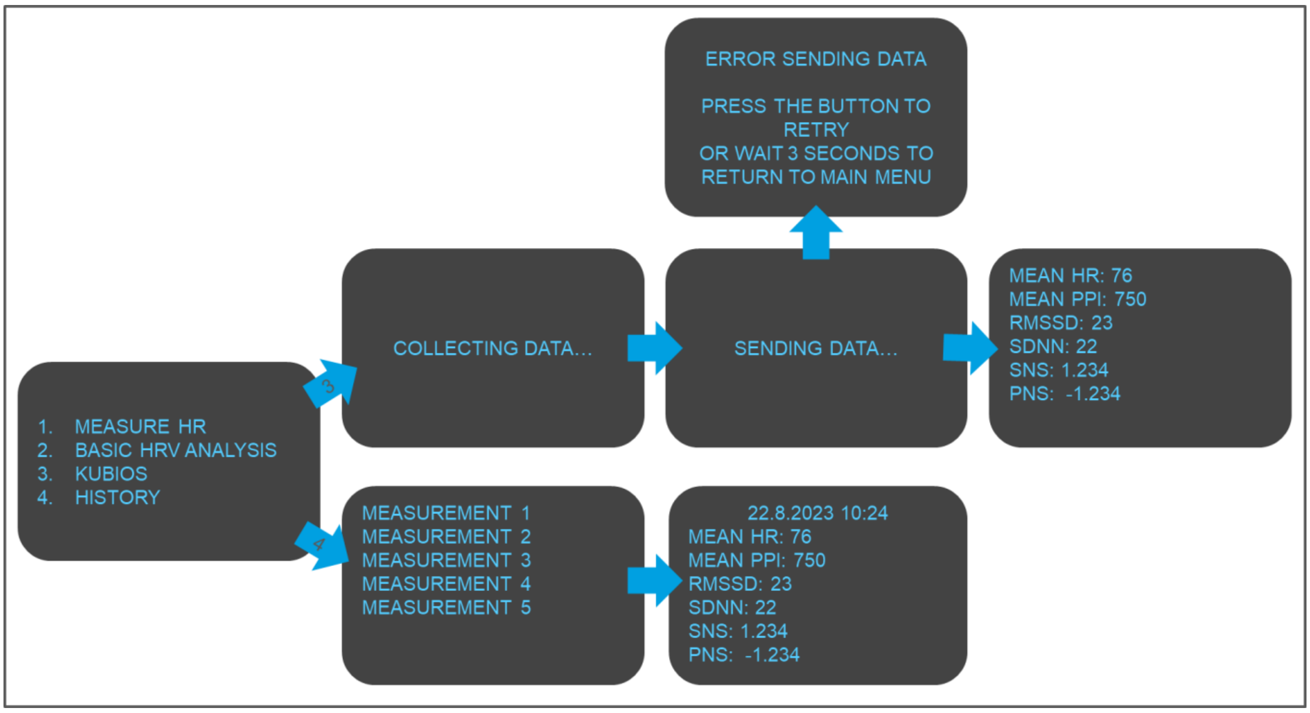

Figure 3: UI example for third level requirements

The data sent to Kubios is the peak-to-peak intervals in milliseconds. When reading and processing the data, the peak-to-peak intervals should be stored in a list.

Before you make a request to the Kubios Cloud, you must convert the interval data into a dictionary. The format of the dictionary is defined by Kubios:

```
dataset = {
    "type": "RRI",
    "data": intervals,
    "analysis": {"type": "readiness"}
}
```

Example code for sending a POST request to Kubios Cloud is on the next page. The response typically arrives within a few seconds. Note that the Pico board must first establish an Internet connection before it sends any requests to the Kubios Cloud. The parts you need to change in the code have been highlighted.

Example code for sending a request to Kubios:

```python
import urequests as requests
import ujson
import network
from time import sleep

APIKEY = "pbZRUi49X48I56oL1Lq8y8NDjq6rPfzX3AQeNo3a"
CLIENT_ID = "3pjgjdmamlj759te85icf0lucv"
CLIENT_SECRET = "111fqsli1eo7mejcrlffbklvftcnfl4keoadrdv1o45vt9pndlef"

LOGIN_URL = "https://kubioscloud.auth.eu-west-1.amazoncognito.com/login"
TOKEN_URL = "https://kubioscloud.auth.eu-west-1.amazoncognito.com/oauth2/token"
REDIRECT_URI = "https://analysis.kubioscloud.com/v1/portal/login"

response = requests.post(
    url = TOKEN_URL,
    data = 'grant_type=client_credentials&client_id={}'.format(CLIENT_ID),
    headers = {'Content-Type':'application/x-www-form-urlencoded'},
    auth = (CLIENT_ID, CLIENT_SECRET))

response = response.json() #Parse JSON response into a python dictionary
access_token = response["access_token"] #Parse access token

#Interval data to be sent to Kubios Cloud. Replace with your own data:
intervals = [828, 836, 852, 760, 800, 796, 856, 824, 808, 776, 724, 816, 800, 812, 812, 812,
756, 820, 812, 800]

#Create the dataset dictionary HERE


# Make the readiness analysis with the given data
response = requests.post(
    url = "https://analysis.kubioscloud.com/v2/analytics/analyze",
    headers = { "Authorization": "Bearer {}".format(access_token), #use access token to
access your Kubios Cloud analysis session
                "X-Api-Key": APIKEY},
    json = dataset) #dataset will be automatically converted to JSON by the urequests
library

response = response.json()

#Print out the SNS and PNS values on the OLED screen here
```