Reykjavík University

Mechatronics I

T-411-MECH

Individual Assignment 9

13/10/2020

Students:
Giacomo Menchi

Teacher:
Joseph T. Foley

Contents

1 Tasks

As usual, the code for this assignment is available at the following Github repo:

Link: GitHub Repo

I've also made a video which should contain everything requested in the assignment, but I will still make comments and explain how I proceeded task by task. In the video, the Raspberry Pi shown is not mine, but Matteo's, since halfway through the assignment I reinstalled the Bluetooth package by mistake, thus deleting the custom configuration which is automatically created during RPi first boot and preventing Bluetooth from working ever again, no matter what I did (I will fix this by formatting the RPi and starting from scratch ASAP).

Link: Video Link

1.1 Task 1

Task: Develop an Android app that uses Bluetooth and is able to configure it:

1. Have a list option where you can pick your Bluetooth device
2. Have a connect button to make a Bluetooth connection
3. Have a disconnect button to break the Bluetooth connection
4. Have an indicator show if it is connected or not.

Screenshot: This below is a screenshot showing what the "fully equipped" app looks like.

1. By clicking on "Scan Bluetooth Devices" I can access the Bluetooth devices list (Task 1.1).
2. By clicking on "Connect" I can open the Bluetooth connection (Task 1.2).

3. By clicking on "Disconnect" I can close said connection (Task 1.3).

4. The text just below is the connection indicator, which could be "Not connected" (as seen in the screenshot), "Connected" or "Disconnected" (Task 1.4).

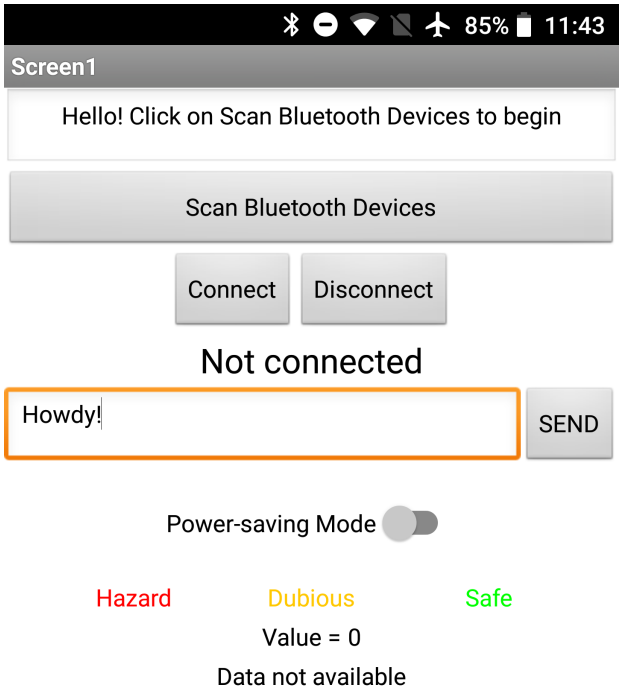

Figure 1: Main app screenshot.

1.2 Task 2

Task: Send and receive messages to the Pi

1. Have a text box that has status messages in them
2. Have a text box where you can write out messages
3. Have a button that can be pressed to send the message.

Explanation: This task's features are still visible in the previous task screenshot.

- The first text box on the top of the screen (says "Hello! Click on Scan Bluetooth Devices to begin" in the screenshot) contains status messages about the app, and also works as a tutorial for the first app steps (Task 2.1).
- The central text box (says "Howdy!" in the screenshot) is used to write out messages (Task 2.2).
- The button which says "SEND" is used to confirm and send the message (Task 2.3).

1.3 Task 3

Task: Have a slider marked power-saving mode and 3 battery indicators (lights/switches). **Important note: the battery indicator does not have to match the phone's actual battery, it is just a state variable you will create as mentioned below.**

1. Have the power-saving mode switch send a message over Bluetooth (to the Pi) to indicate that it was activated
2. Have the indicators be: red marked HAZARD, one Yellow marked Dubious, and one Green marked Safe.
3. Have the Indicators turn on when the battery value changes: 21+ (safe), 11-20 (dubious), 10-0 (hazard).

4. Write a program on the Pi to have the battery value change over time so you can demonstrate the functionality. It will need to keep track of a battery state value and go through charging and discharging cycles. The discharging cycle should be slower if the battery saver is turned on.

Explanation: The explanation below is still referring to the screenshot in Task 1.

1. The Power-saving Mode switch sends two different messages via Bluetooth when activated and deactivated (Task 3.1).

2. The three labels ("Hazard", "Dubious" and "Safe") are the battery indicators (Task 3.2).

3. At each time, there is always one visible ("on") label, which represents the current battery level, while the other two are invisible ("off"); the precise battery value is also written in another label (says "Value = 0" in the screenshot) just below the indicators (Task 3.3).

4. The Raspberry Pi sends temporized data to the Android app which raises or lowers the battery value every second (in the video I just made the Raspberry "charge" from 0 to 30 and "discharge" from 30 to 0 for convenience, it should go from 0 to 100 and from 100 to 0 normally), thus modifying the indicators when the level is changed; when the battery saver is turned on, the value decreases by 0.5 instead of 1, thus simulating the "power-saving mode" (Task 3.4).

2 Extra Credit

2.1 Extra Task 1

Task: Have the android app check if it is connected and if data is available every 10ms.

Explanation: The data availability label is the last one on the bottom of the page (says "Data not available" in the screenshot) and checks if the connection is activated and if there is data to read. If there is no connection

or no data to read, it stays like it is in the screenshot, while if there is data available, it switches to "Data available".

2.2 Extra Task 2

Task: Have the android app not produce any errors on runtime but print if an operation was not allowed in the status bar and why.

1. FX. when not connected the disconnect button should give in the status bar ERROR: Can't disconnect if never connected

Explanation: AppInventor gives users the ability to suppress errors and manage them in other ways, for example by writing them to a text box or a label. For this reason I created some custom errors and printed them inside the status box created in the task 2, for example:

- "ERROR: Can't disconnect if never connected", which is given when clicking on the "Disconnect" button while not connected.
- "ERROR: Connection failed! Is the device available?", which is given when clicking on the "Connect" button while the selected device can't be reached or fails in connecting.
- "ERROR: Can't send message if disconnected", which is given when clicking on the "SEND" button while not connected.
- "ERROR: Can't communicate if disconnected", which is given when clicking on the "Power-saving Mode" switch while not connected.