Lehrstuhl für Regelungstechnik und Systemtheorie
Embedded Systems
Wintersemester 2015/2016

Creating a new class derived from ShowSensorInfoActivity

All steps assume you are working with android studio and have an up-to-date checkout of de.rub.rus.sensorlister.

1 Derive a new class

Steps:

- Assume we want to create ShowAccelerometerSensorInfoActivity and recall it is supposed to be a subclass of ShowSensorInfoActivity.
- Right-click on app/java/de.rub.rus.sensorlister in the project view and choose → New Activity → Blank Activity. In the new window that opens, call the activity "ShowAccelerometerSensorInfoActivity". Leave all auto-generated entries as they are.
- The two files activity_show_accelerometer_sensor_info.xml and java/ShowAccelerometerSensorInfoActivity are opened automatically.
- For the file res/layout/activity_show_accelerometer_sensor_info.xml, it is actually easier to overwrite its auto-generated content by copying and adjusting the content of activity_show_rotation_vector_sensor_info.xml. Copy its content and change all fields to apply to the accelerometer instead of the rotation vector sensor (at least one context and two ids need to be changed, possibly more).
- For the file ShowAccelerometerSensorInfoActivity, it is also easier to change the file by copying-and-pasting from ShowRotationVectorSensorInfoActivity.
  - Make sure ShowAccelerometerSensorInfoActivity extends ShowSensorInfoActivity. Copy, for example, everything from ShowRotationVectorSensorInfoActivity starting from the keyword "extends".
  - Afterwards change the file to apply to the accelerometer instead of the rotation vector sensor. At least change
    - the sensor type in setSensorType,
    - the TextViews defaultSensorInfoTextView and allSensorsInfoTextView in setTextViews,
    - the activity in getContentViewId,

    using the auto-completion feature of studio.
- We need to switch on the "Info" button for the new activity.
  - Add a variable for the button in MainActivity.java by adding the line "private Button accelerometerInfoButton, accelerometerRunSensorButton;". Also add the line "private TextView accelerometerInfoText;" or add "accelerometerInfoText" in the existing TextView declarations. Note that we only actually only need the button accelerometerInfoButton, but we declare accelerometerRunSensorButton and accelerometerInfoText at the same time, because all three are needed in the call checkSensor() described below. Try to insert the variable declarations in such a way that the list of buttons remains neat. Try to keep definitions, calls etc. in the same order as the sensors are listed in the running SensorLister app.

  - Assign accelerometerInfoButton, accelerometerRunSensorButton and accelerometerInfoText their values in MainActivity.identifyButtonsAndTextViews() by mimicking the code that already exists for one of the other sensors. Try to keep the code in the same order as the sensors are listed in the running SensorLister app.

- Having created classes and xml files, we need to tell MainActivity when to invoke the new code.
  - In res/layout/activity_main.xml, find @+id/accelerometerInfoButton and add an xml entry android:onClick=”sendAccelerometerInfoButtonPushed”. This tells the main activity to invoke the method MainActivity. sendAccelerometerInfoButtonPushed when the info button for the accelerometer is pushed.
  - In java/MainActivity.java, add a public void method called sendAccelerometerInfoButtonPushed, copying and adjusting sendRotationVectorInfoButtonPushed, for example.

- Compile, run and test, and commit and push your changes. You can switch to the Terminal view in android studio and run “git status .” to get an overview on all new and changed files. Use ’git commit -m “Added info activity for accelerometer” .’ to commit your changes locally. Use “git push origin master” to push them to the repository.