School of Engineering & Design – Electronic & Computer Engineering

M.Sc. Course in
Distributed Computing Systems Engineering

Workshop

Embedded Systems

Internet of Things Applications Prototyping

Lecturer: Dionysios Satikidis, M.Sc. (dionysios.satikidis@gmail.com)

Date: 2018/03/24 and 2018/03/25

1. Aims and Objectives

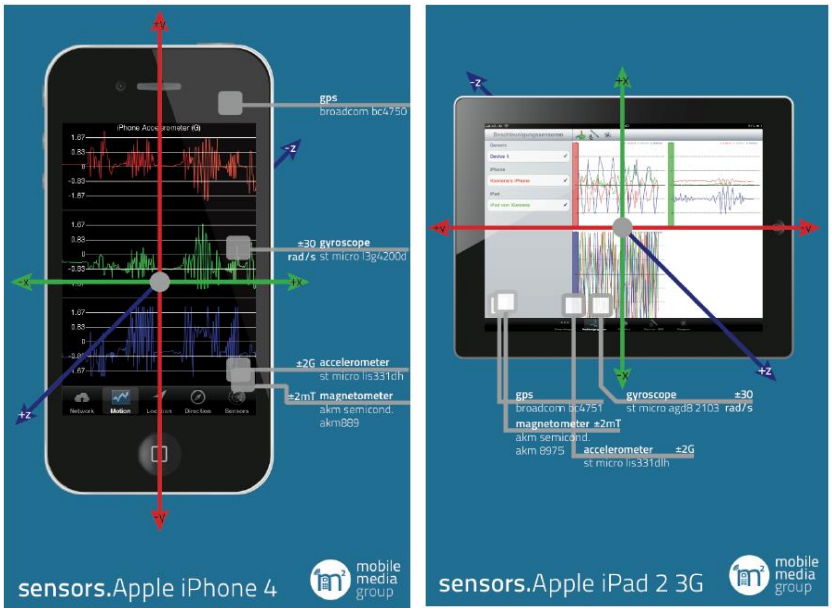
This is the workshop part of the corresponding lecture “Embedded Systems Engineering”. It covers the design, implementation and emulating aspects of embedded sensor systems. Beside the aim of realising an application on an embedded sensor target and solving your own specified problem definition, this workshop also targets to improve skills of scientific analysis and reporting. In context of this workshop, the word “Things” - from the term *Internet of Things* (short IoT) will be used as a collection of smartphones and other Things, which are connected to the prototyping development board particle photon.

The focus is set on the analysis of captured sensor data, and the iterative improvement of your application on the embedded sensor target. Therefore you should work out a selection of aspects that in your point of view are considered to be worth investigating. You should always give reasons for your decisions justifying your choices by discussion, and describing your expectations of corresponding results. Consider if your results are correlating to your conception.

2. Aspects of System Architecture

2.1 Background

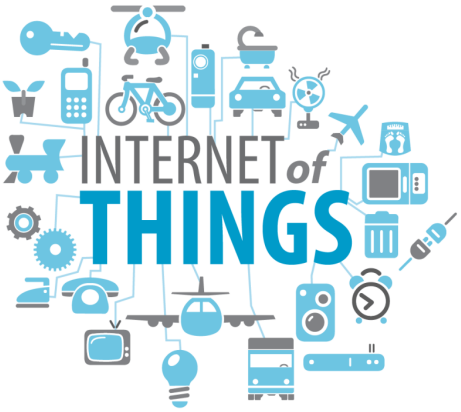
Nowadays smartphones have secured a strong place in our society. We take them everywhere with us, even to bed as an alarm clock to wake up. Smartphones are already an extension of us, an important limb to our body. Furthermore they are open and programmable and come with a growing number of powerful embedded sensors, such as an accelerometer, digital compass, gyroscope, GPS, microphone, and camera, which are enabling new sensing applications across a wide variety of domains like social networks, mobile health, entertainment, education and traffic management.

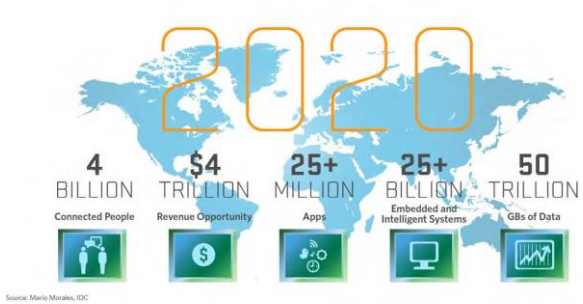


[1]

[1] M. Schirmer, H. Höpfner, Smartphone Hardware Sensors, Mobile Media Group

The Internet of Things defines the interconnection of physical devices, vehicles, buildings, and further Things, which are connected with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data and information. IoT empowers objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities resulting in improved efficiency, accuracy and economic benefit.





[2]

The sample applications Multimodal Mobility Analyser (short MMA) and Noise and Movement Detector (NMD) should serve as examples for the following aspects of architecture and process model. Based on these examples it should be monitored, how the background of different fields like digital signal processing, user interface design, object-oriented programming, real-time systems, IoT Ecosystems etc. can have an impact to the success and the invention of new applications.

---

[2] Source Mario Morales, IDC and David Matthewa, http://visualizetheweb.com

2.2 A sample application (based on SSF) called Multimodal Mobility Analyser (MMA)

The MMA is based on the idea, to detect the transportation mode (e.g. walking, biking, train, car, bus, subway etc.) of the smartphone owner by using the acceleration and GPS-Sensors (see conceptual presentation shown in Figure 1). Future and innovative smartphone-applications should use this functionality to assist users in making smarter individual transportation choices to collectively reduce carbon emissions in cities. Scientific realisations of this idea also focus the usability (e.g. application can work without a specific orientation of the smartphone) and aspects of social media (e.g. users can share low carbon routes in a community) as success factors and enablers for such an invention[3].

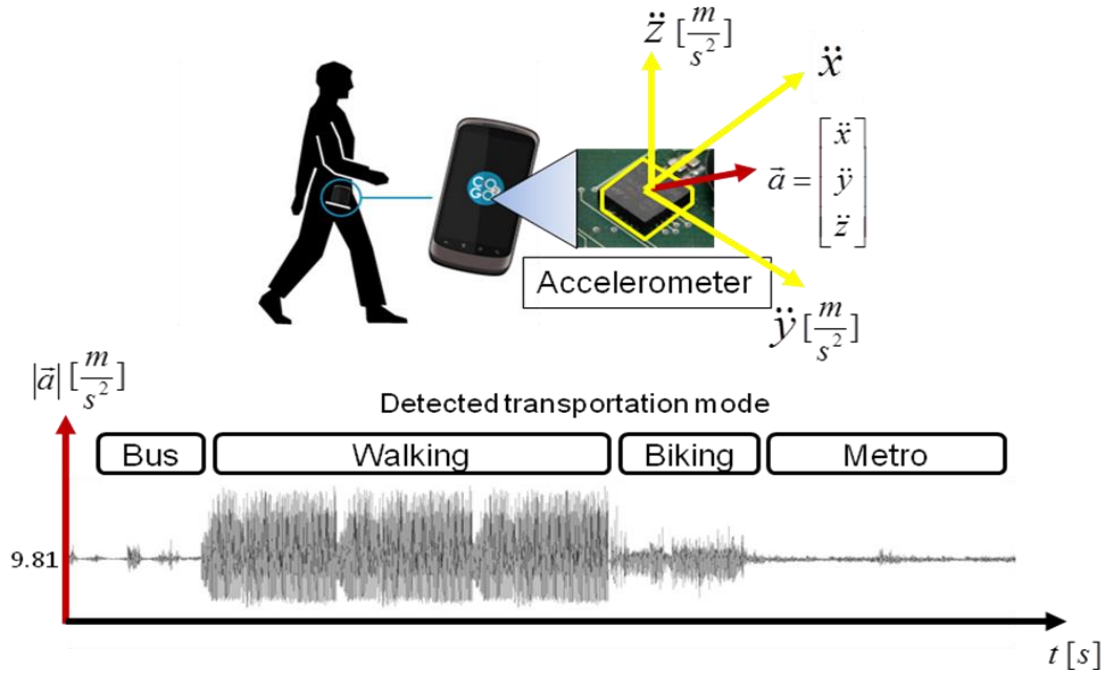


Figure 1: The Idea of MMAs' basic functionality

As visualised in Figure 1 the smartphone orientation can be neglected, if the absolute value of the acceleration vector $|\vec{a}|$ is regarded as input signal for the transportation mode detection engine. This engine could first filter the signal to eliminate captured outliner values, and after that interpret the cleaned signal using threshold algorithms, decision trees etc. However, a very important aspect of acceptance for such an application is the evaluation of the engines' correct operation (e.g. in percent). This can be fulfilled with an architecture which allows capturing and repeating scenarios, reproducing important effects and evaluating the system decisions. Such architecture is given with the Smartphone Sensing Framework (SSF), which is described in the following chapters.

[3] See C.Ratti, K. Kloeckl, CO2GO, MIT SENSEable City Lab, http://senseable.mit.edu/co2go/

2.3 A Smartphone Sensing Framework (SSF)

A simple and usable architecture allows creating and extending smartphone applications in a structured way and in parallel tasks. In context of this workshop, a sample architecture called “Smartphone Sensing Framework” (short SSF), should give examples how the user-interface, the application-logic and the hardware-drivers can be encapsulated and in which way robust interfaces can be used for a parallel development. Furthermore this architecture should demonstrate how the sensor-data can be captured and repeated in a simulation-mode.

Please open following checkout following Git-repo with your Lecture-Account (handed over in the lecture) and try out the sample App.

https://github.com/MrDio/Smartphone-Sensing-Framework/tree/MMA%40SSF2.0_AndroidStudio_2018

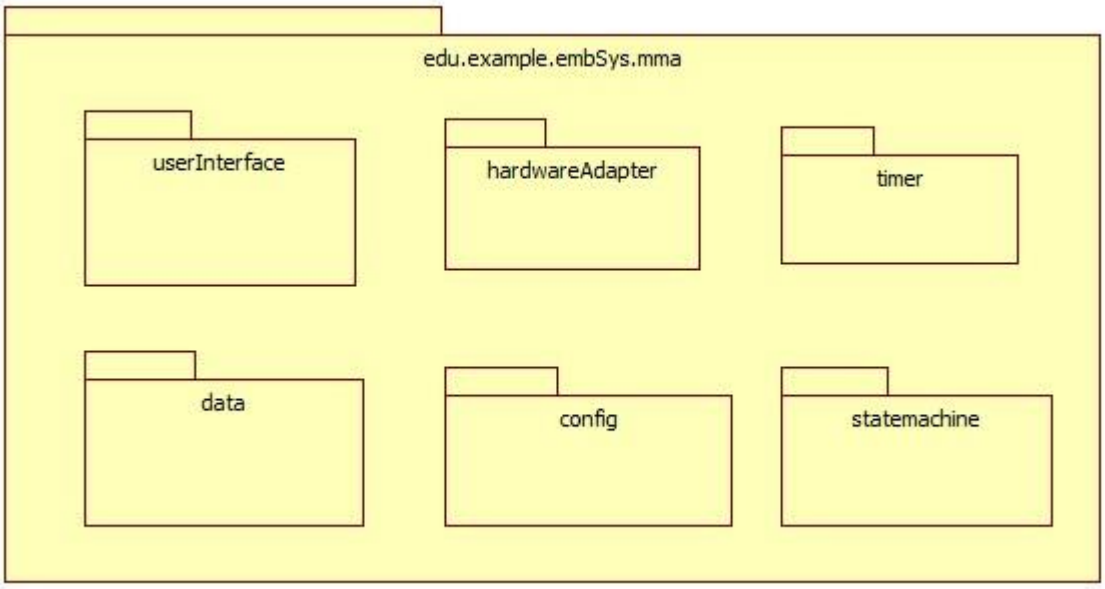


Figure 2: MMA based on SSF package-structure

As visualised in Figure 2 the SSF package-structure encapsulates the MMA-sample application, which is described as followed:

Package *userInterface contains*:

- The Android-related classes, like Activity-derived-classes, which control the overall application process like initialisation and control of the components. These Activity-derived-classes handle the applications context, defined in the res-folder of the application (especially the folders res/layout, res/menu, res/values).

- The handlers, which process the users input like the events, generated e.g. by clicking a button, and the corresponding logic to handle the input.

- An interface which allows updating the changed sensor-data in runtime (see interface IUserInterface).

Package *config* contains:

- Just one class with the application configuration like the sensor-sampling-time, the output-folder for the captured data (same folder at the host-PC and at target-Smartphone), the “Simulation-Mode”-flag etc.

Package *data* contains:

- A data-writer-class, which allows serialising the captured data into a csv-file. Thereby one row of the csv-file represents one sampling step of the implemented application, with the related timestamp, tick and the sampled sensor-data.

- A data-reader-class, which allows reading-out previous captured, or edited data of a csv-file. In this case the data-values in the csv-file are used as stimulation for the application logic. In this case the application should run in the “Simulation-Mode” (see *config.ConfigApp.isSimulation*).

- Data buffer-classes (e.g. the CurTickData-class which contains the values of the current tick) and a further class, which initialises the folder structure at application start.

Package *hardwareAdapter contains*:

- The drivers for the smartphone sensors. These drivers are encapsulated in sub-spackages (see GPS, accelerometer, microphone), which contain the realisation of the hardware interface (see IGPS, IAccelerometer, IMicrophone). This realisation exists in two ways for each sub-package. One class contains the concrete realisation of the interface, with the corresponding functional access to the smartphone hardware (see Accelerometer, GPS, Microphone). The other class simulates the hardware (see AccelerometerSim, GPSSim, MicrophoneSim), with access to previous captured or edited data in a csv-file, by using the data-reader-class (see package *data*).

- A hardware-factory-class (see HardwareFactory), which detects at application-start, if the “Simulation-Mode” is active, and creates the driver or the simulation classes (e.g. GPS or GPSSim, both classes implement the same interface IGPS).

Package *stateMachine* contains:

- The implementation of the desired state machine, which should encapsulate the application-logic.

- The abstract class AbstractState, which defines the methods and attributes of a generic state.

- The derived concrete states, which correspond to the application logic. In case of MMA the application logic contains three states (these are encapsulated in the classes StateUnknown, StateDriving, StateWalking).

- The callback-interface IParentStateMachine. This allows the concrete states to inform the state machine, if the state has changed.

- The state machine, which is represented by the class StateMachine.

-

- The interface IStateMachine, which encapsulates the complete package statemachine from other components and packages of the SSF.

2.4 An IoT Prototyping Framework (IoTPF)

With the IoT Prototyping Framework (IoTPF) the capability should be given to create IoT prototypes with the embedded prototyping board particle photon. This full-stack Internet of Things (IoT) device platform allows connecting directly sensors or non-connected devices to the web. The IoTPF implements the same logic as the SSF, without the ability to emulate the particle device on a local computer.

Please open following links with your Lecture-Account (handed over in the lecture) and try out the samples.

IoT-EcoSys_Sensor_Samples:
https://go.particle.io/shared_apps/5a9dafb5da1aee99c000076f

IoT-EcoSys_NMD_Sample:
https://go.particle.io/shared_apps/5a9dafdeda1aeecbb100072c

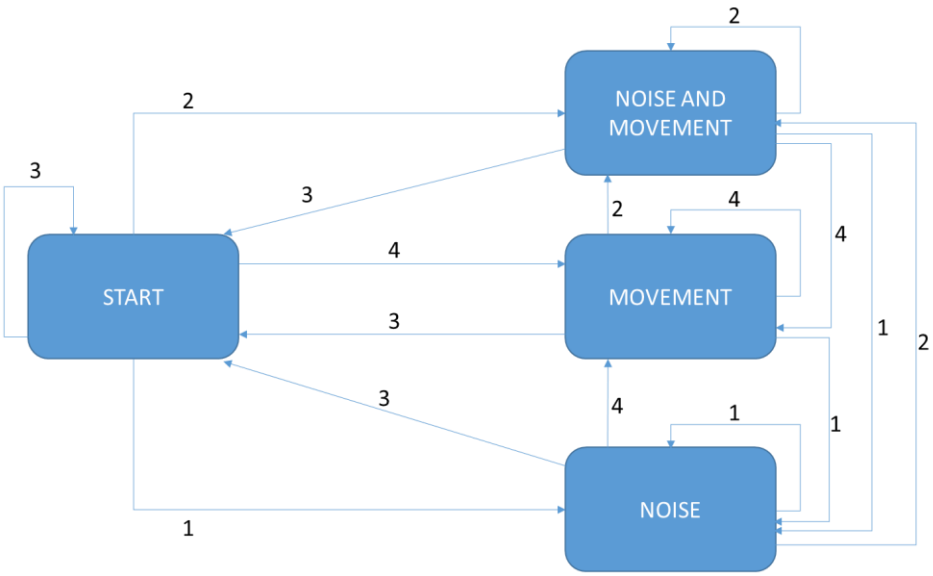
IoT-EcoSys_Losant_Sample:
https://go.particle.io/shared_apps/5a9daffdda1aeea3350005f7

IoT-Eco_IFTTT_Sample:
https://go.particle.io/shared_apps/5a9db011da1aee4d4400057e

IoT-Eco_IFTTT_Function_Sample:
https://go.particle.io/shared_apps/5a9db033da1aeea335000607

2.5 *A sample application (based on IoTPF) called Noise and Movement Detector (NMD)*

The application called Noise and Movement Detector (short NDM) serves as an example for an IoTPF implementation. NMD is based on the idea, to detect noise and movement (e.g. in for a use case as theft protection or baby phone etc.) of the environment the particle photon board is placed. The application can switch between four states (shown in Figure 3). Each states submits an event, which can be consumed by a cloud-application, triggering a notification or a user defined process.

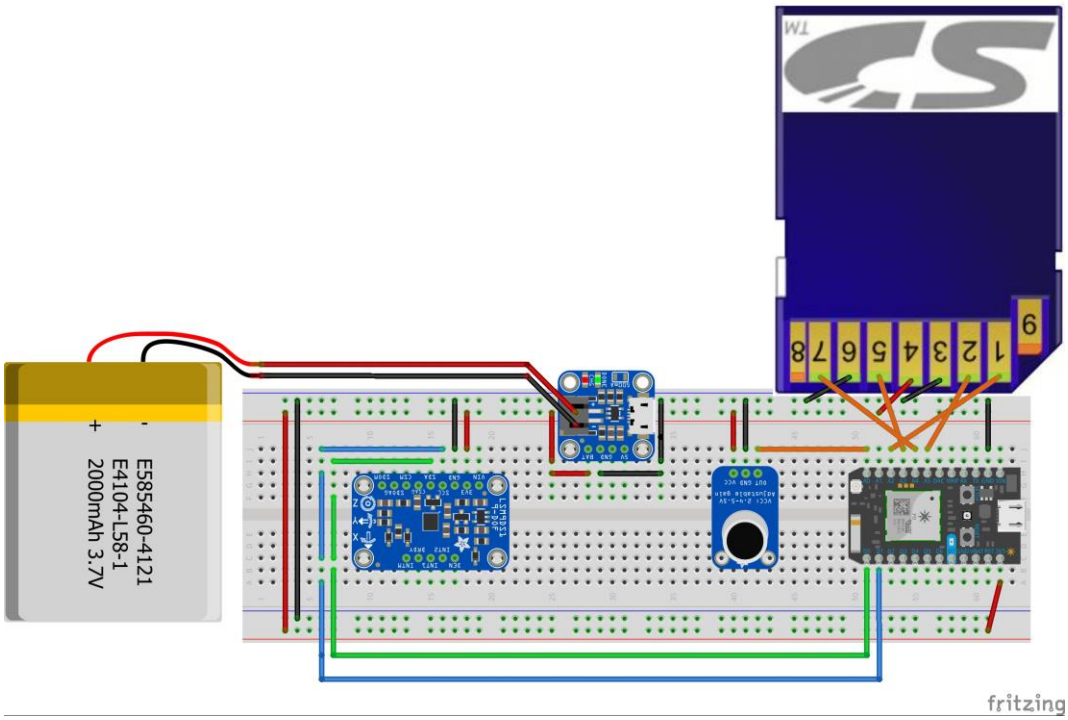


1. NoiseDetected() = true && MovementDetected() = false / Publish(Event(Noise Detected))
2. NoiseDetected() = true && MovementDetected() = true / Publish(Event(Noise Detected)), Publish(Event(Movement Detected))
3. NoiseDetected() = false && MovementDetected() = false /
4. NoiseDetected() = false && MovementDetected() = true / Publish(Event(Movement Detected))

Trigger is Loop TICK_TIMER

Figure 3: NMD State Automata

The particle photon board is wired with several sensors, which support sensor data during the system-loop. This sensor-data is written in a column separated values file (.csv) on the SD-Card. Getting started with the particle photon and the available sensors by connecting the prototyping board with the peripheral sensors and SD-Card. For more information and samples follow instructions on particle.io[4] website and the instructions in the WS6 introduction. The following schematic shows the wiring of the IoTPF Hardware. For your own application, you can use the tool fritzing[5], which will be given with the student support package.



[4]See http://particle.io
[5]See http://fritzing.org

2.6 *Aspects of IoT Ecosystems*

Internet of Things Ecosystems are IT-applications which run as usual on a cloud, and allow to connect Things, Services and Applications together. Thereby there is a variety of possibilities of a connection e.g. data-transfer and visualisation, event-Processing etc. In context of this workshop it is recommended to use two IoT Ecosystems to demonstrate your IoT application.

A simple easy to use IoT Ecosystem is given by IFTTT[6] (If this then that). This application allows you to connect Things (particle photon and/or the SSF-App on Android-devices) with a variety of services and to capture or to get notifications in case of events on the smartphone without implementing an Android-application.

A more complex IoT Ecosystem is given with Losant[7]. Losant, similar to IFTTT, gives also the capability to connect Things (particle photon and/or the SSF-App on Android-devices) but furthermore it allows to handle events and data in more complex way with a dedicated workflow engine for complex event processing.

For more information and samples follow instructions and examples on the ifttt.com or losant.com website and the instructions in the WS6 introduction. Sample applications will be given during the Workshop.

2.7 Aspects of advanced signal processing with pattern recognition

Different applications, implemented in past workshops, showed the necessity of recognising pattern in the processed data. For fulfilling that need in case you want to focus on advanced signal processing with pattern recognition in your scientific assignment, the toolkit which is recommended in context of this workshop is Googles TensorFlow. TensorFlow is an open-source software library for Machine Intelligence which allows you to develop neuronal networks with python and export these in different programming languages like C++ or Java. Pattern recognition can be used in this workshop e.g. direct on the particle photon board, on the Android-Device or in the IoT Ecosystems on the cloud as service. If you are interested using this toolkits, you can follow instructions on the TensorFlow [8]website.

[6]See http://ifttt.com
[7]See http://losant.com
[8]See https://www.tensorflow.org/

2.8 Aspects of a test driven development

Other important aspects, for developing and evaluating applications on a smartphone or particle sensor target, are the activities bundled in a process model. As visualised in Figure 4, the first step is to have an idea, how an application on a smartphone sensor target could improve or simplify a real-world problem. After the concept is clarified, the design activities of the state machine, user interface and the needed hardware interfaces (depends on the needed sensor-signals) can be executed nearly parallel. As demonstrated in SSF, clearly specified interfaces between hardware (see package *hardwareAdapter*), state machine (see package *statemachine*) and the user interface (see package *unserInterface*) can be the enabler for also executing the implementation activities in parallel. The final development activity is the integration into the other components of the SSF (see packages *timer, config* and *data*). The IoTPF does not provide an emulation of the particle photon board, for that activities developing and testing applications on the particle photon board cannot be parallelised and may be more time-consuming.

Finally the **most important activity in context of this workshop is the evolutionary improvement of your state machine, and the robust detection of signals and events using real-world captured data**. This real-world data should be captured in different scenarios, which are important to demonstrate the quality of your application.

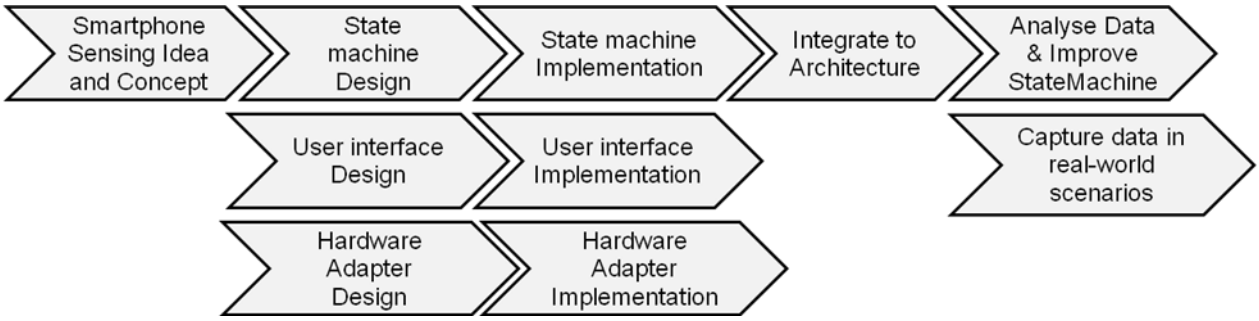


Figure 4: Proposed process model

2.9 Technology stack for this workshop

The recommended technology stack for this workshop is shown in Figure 5. In context of this workshop it's up to you, to select the technology as needed for your application.

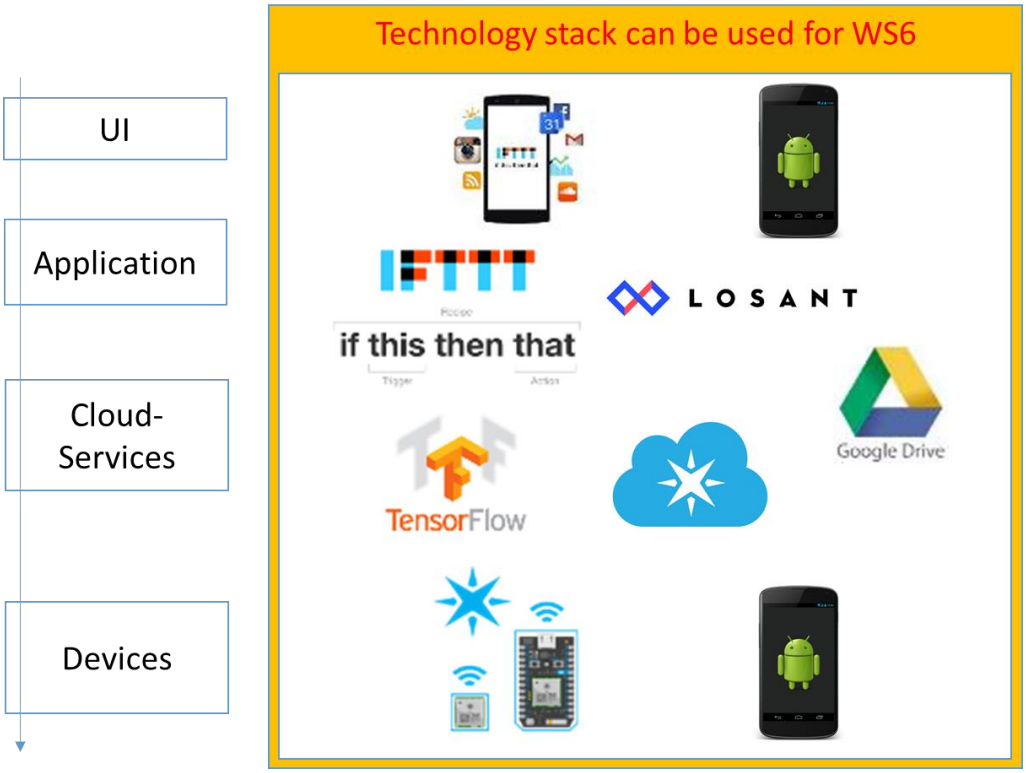


Figure 5: Recommended technology stack

In context of this workshop, the following aspects should be investigated related to their activities:

Sensing Idea and Concept of Things (smartphone and/or particle photon):

- How could your idea improve or simplify a real-world problem, perhaps in the society where you live?
- In which way could your application become popular?
- Is it possible to combine your application idea, with megatrends like social media, social networks, globalisation, urbanisation, individualisation, silver-society, new work, female shift etc.?
- Which is/are the unique selling point(s) of your idea?
- Which smartphone sensors are you going to use and in which way?
- The project leader of the project can create a **blog-entry on hackster.io**[9]to promote that idea.

State machine design and implementation:

- Describe the essence of your idea, which should be covered in the application-logic?
- How could this essence be represented in states and transitions?
- Do you have covered each case in each state (are all transitions identified)?
- Could you minimise or simplify your state machine?
- How could you test your state machine?
- Make a concept of the data, which should be captured and/or used as stimuli for testing your application.
- Realise the state machine by considering the implementation architectures presented in SSF and IoTPF.

User interface design and implementation:

- Tell an overall story about your final application. How it will cover your idea and your concept?
- Use mock-ups to visualise and design your application interface[10].
- Think about usability aspects, regardless if you will implement these.
- Decide which interface you are going to implement, and which one will just described in this design phase.
- If you are going to use IFTTT or Losant, you can create a short prototype.
- Design the interface to the other application components, as demonstrated in SSF.
- Which data should be visualised, and which data should be captured?
- Realise the user interface by considering the implementation architectures presented in SSF and IoTPF.

[9] See http://hackster.io

[10] See Mockups, Balsamiq Studios, http://www.balsamiq.com/

Hardware adapter design and implementation:

- Which data should be captured?
- How could the smartphone or Particle Photon target sensors capture the desired data?
- How correlate the real-world processes to your sensors?
- Does the quality and format of the sensor data reflect your requirements for your application?
- Which drivers should be implemented?
- Which should be an optimal sampling rate for the sensors?

- Design a robust interface to access the smartphone or Particle Photon hardware and a simulation class, both behind one interface and integrated to SSF (see *HardwareFactory*) and IoTPF.
- Realise additional hardware drivers, by considering the implementation architectures presented in SSF and IoTPF.
- There are a lot of frameworks and APIs. These could help you with advanced signal processing (e.g. TensorFlow). Just dare to use these, if you have an innovative idea[11].

Capture Data in real-world scenarios, analyse and improve your state machine:

- Design scenarios which cover the essence of your application.
- Try to capture and analyse sufficient data, for improving the quality of your application.
- Analyse the captured data to find out patterns and characteristics, which could help you to realise your application.
- Improve your algorithms, filters and predictors.
- Use concepts of digital signal processing. Perform a literature review if necessary.
- Improve the SSF method with test script automation if necessary.
- Compare your improvements of each iteration step, with the last version of your application-logic. How the detection-quality has changed?
- Visualise your data to demonstrate the quality of your application and to make your application more tangible[12].

[11] See https://www.tensorflow.org/

3. The Project

3.1 Experimental and Investigative Methods

Your analysis should be reflected by the measured results. All expectations made in advance and the results derived from the analysis should be discussed. Sophisticated discussion is the key point to a successful scientific report. You should visualise your results in tables, charts and figures and discuss these. Evaluate your results and keep in mind your conception and how both correlate. This is crucial to identify possible erroneous measurements. Keep this in mind while designing your application and test environment.

Reminder:
Finally the **most important activity in context of this workshop is the evolutionary improvement of your state machine, and the robust detection of signals and events using real-world captured data**. This real-world data should be captured in different scenarios, which are important to demonstrate the quality of your application.

3.2 Additional Motivation

The workshop results of all teams should be posted at one of the most-viewed blogs in US and Germany, as candidate-topic for further contact. With this option the workshop of the course Embedded Systems could be regarded as opportunity to present your assignment-work through a worldwide known platform. One team is going to be the winner team, which will be in particular called within an IoT article at Bosch, as well as within an abstract and hackster.io-post of the realised and researched application-idea. This article could be the base for further interests and publicity for the winner-team.

3.3 Project organisation

You can work in teams of 2 - 5 students, when planning and implementing your application.

The following team rolls should exist in each team:

Project leader (one team-member)

- Is responsible for the intern and extern team-communication (Meetings, Presentations, Discussions)
- **Is responsible for the Project-Promotion at hackster.io**
- Coordinates the activities of the project
- Creates and traces the work-breakdown structure
- Is responsible for the documentation structure
- Decides which idea should be realised
- **Has the responsibility to deliver the assignment, the attachments, the smartphone, the particle board with all components to the lecturer before closing date.**

Developer

- Sets up the development environment
- Designs the architecture and implements the application
- Decides the technology stack to be used
- Executes and evaluates tests
- Discusses points of improvements
- Decides which algorithms, filters, pattern recognizers should be used
- Delegates parts of the technical review to the team-members

Data-Analyst

- Sets up the data- and analysis-environment
- Analyses the output data and creates the test-data (also called stimuli-data)
- Specifies and evaluates tests together with the other team-members
- Trains Neuronal Networks in case of using TensorFlow
- Discusses points of improvements in the team (quality of the results, algorithms etc.)
- Visualises the data (charts, diagrams, virtual environments e.g. Google-Earth)

Documentation-Manager

- Is responsible for the project documentation and the quality of this
- Specifies unique documentation and visualisation templates
- Discusses the data to visualise with the data-analyst
- Integrates the different parts of the documentation to a clearly structured assignment
- Is responsible to select the interesting contents for the assignment, and the less interesting for the appendix

3.4 Time Schedule for the two Days

Day 1:

1. Come together and introduction to the workshop	Dionysios Satikidis	9.00 am
2. Find Teams (Assignment of team roles)	All students	09:30 am
3. Brainstorming: Smartphone Sensing Idea	Each Team	09:45 am
4. Presentation of your idea (5 min. per Team)	Each Team	11:00 am
5. Project work	Each Team	11:40 am – 6:00 pm

Day 2:

1. Project work	Each Team	8.00 am
2. Presentation of your concept (15 min. per Team) Team photo shoot for blog-article (optional)	Each Team	11:00 am
3. Project work	Each Team	12:00 am – 5:00 pm

3.5 Deliverables and Outcomes

It is very important to discuss and analyse your experimental work. If you fail to obtain satisfactory results, sophisticated discussion on these results and the reasons for failure may still be worth credit. Think over extensive implementation work, such as GUI frontends and architectural oversize, as this will not improve heavily the rating of your assignment. However, designing a modular, clear and reusable architecture will help you and your team-members to extend and improve your software until a final satisfactory state.

Each Team will receive:

-One particle photon board
-One Electronic Kit Bundle with Breadboard, Cable, Resistor, Capacitor, LED, Potentiometer
-One Microphone Adafruit MAX4466 Brakout Board
-One 9DOF Acceleration / Gyro / Mag Sensor Adafruit LSM9DS1
-One SD-Card Breakout and one Micro-SD card
-One LiPo Battery 3,7 v 2000 mAh
-One USB-Cable for the particle photon board
-One USB LiPo Adafruit Battery Charger Jack
-The NDM based on IoTPF as code in the Particle Photon IDE

-One smartphone – Huawei P8 with on-board sensors and android
-One USB-Cable for the smartphone
-The MMA based on SSF as code in the student support package (also stored on SD-Card).
-One Plug for power-socket and Boxes
-One account for Losant.com
-One account for Particle.io IDE
-One account for Gmail and Google Drive
-One account for IFTTT.com
-One account for Hackster.io

The deliverables for each Team are:

One bound report on paper:

- The assignment report with approximately 15 to 20 pages (This will be delivered to the TAE until the deadline).
- The report should contain a one-page abstract
- The report should contain the login credentials (Login, Password) of IFTTT, Losant, Gmail, Google Drive, Hackster, Particle Photon IDE, Google Drive etc. if these cloud-services are used and changed.
- **The deadline will be presented in the workshop**

Contents on CD:

- A copy of the assignment report as PDF and WORD to aid the plagiarism check
- The source code of the android software (project folder (eclipse or android studio) with all dependencies)
- The executable of the software (APK-package)
- The application source code of the Particle Photon as .ino file(s) with included libraries description
- The python and C++ / Java source code of the created Neuronal Network in the project folder
- The files containing the measurement and/or training data collected.

3. Sensing apps that have been explored in the past workshops

In context of this workshop, **it is not allowed** to **implement and research equal or nearly equal topics as presented in the past workshops**.

LaundReminder - 🏆 Winner team of the year 2017 🏆
The LaundReminder solution is to remind the operator of the washing machine to remove the laundry before it needs to be washed again. The solution comprises an embedded device that can be attached to the washing machine to monitor its use and operation. The device collects and processes sensor information to find out if the machine still contains a washing load that has not been removed yet. A Cloud-based application acts in lieu of the device to inform the user where reachable. The application routes the reminder about the forgotten laundry and is not mutable because its (intelligence) intent is to get the laundry saved from developing stains and smells. If the operator doesn't forget the laundry or as soon as the operator removes the laundry, the device detects that event and stops nagging the user.

ALARM (2017)
While people should leave the building in case of a fire, the people should lock there doors and lay on the floor, if there is a terror attack inside the building. In case of a chemical reaction or something similar, there are often protection areas the people have to go in case of an evacuation. However, people have to distinguish the warning signal just by the type of sound, so they can react in the right way. But what happens if they can't tell the difference or a new employee started working in the company and has no information about escape routes and warning signals. A different use case would be deaf or distracted people (e.g. Headphones) which can't recognize the sound at all. These are all points where the application "A.L.A.R.M" comes into play. The app records sound of the environment while running in the back of the smartphone. If it recognizes a defined warning signal, the smartphone starts to vibrate and provides feedback about the type of signal and what's the best behaviour in this case. Almost everybody is using a smartphone nowadays, therefore everybody can install the sound patterns of their company to use the application. With the help of A.L.A.R.M life's can be saved!

SmartphoneReminder (2017)
This assignment focuses on the design and implementation aspects needed through developing a small app using only sensors of a smartphone. The steps taken lead to a proper working app which can remind of the smartphone placed in its holder.
Based on the state machine evolutions as well as the final data analysis, results of the most important development steps are reported. As a short summary, the initial assumption was that there might be identifiable patterns within the sensor data of the car-leaving process. To prove this assumption prototypes have been developed to record data with the MMA framework and to visualize them with the language R. Based on the results it got clear that there are some noticeable patterns like walking or the engine in gear shift neutral. During further development more training and testing data are captured to prove the first observations. Therefore, the training datasets are analyzed visually and statistically which reveals some problems with too slow as well as too fast sampling rates.

BabyWatch (2017)

In our application we tried to develop a baby phone. In aspect of internet of things we used three components to develop the application an android smartphone, a particle photon development board and the Losant cloud data storage. The particle photon has a microphone which we use to listen to the surrounding. The collected data than gets transmitted to the Losant cloud via

MQTT, an internet of things message protocol. After that we get the new data from the cloud on the smartphone, via MQTT. On the smartphone we analyze the sound sample and notify the user that the baby is crying with a push notification. If it is not crying nothing happens.

SmartCart (2017)
The aim of the application is a smarter way of shopping to its users. The application owners the possibility to easily mark items as added to the cart and to navigate through a list via gestures in order to support and relieve the user during shopping. The recognition of gestures is done by the built-in acceleration sensor in combination with the magnetic field sensor. Instead of handling a shopping list with the help of pen and paper or by pushing buttons on a virtual list, this app easies the whole process via gesture control. The main focus is placed on two use cases ordered to the app's user: switching between the items in the shop-ping list as well as to check out the already found items with only one hand and simple but unambiguous gestures.

CarTooth - 🏆 Winner team of the year 2016 🏆
CarTooth is about recognizing the ‘entering’ and leaving’ of a car using an Android smartphone application based on SSF. If a hands-free car kit based on Bluetooth is used to communicate with people, it is necessary that Bluetooth is active on the smartphone. The activation is often forgotten and leads to holding the smartphone while driving to answer an incoming call, which is illegal. The smartphone should activate Bluetooth on entering the car to prevent this. Since it is hard to identify the ‘entering’ of a car, this is a difficult problem without any robust solutions for it until now. The results will help to decrease the illegal usage of smartphones while driving and ensure more security on the roads.

Knock to Unlock (2016)

The idea behind Morse Key is quite simple. It’s about unlocking the phone by using a decent knocking pattern. This idea is a result of the increasing demand for comfortability and while holding on to a basic amount of security.

On the current smartphone market the variety of methods to unlock your phone is typically extended when new smartphone models are introduced. In the early days of the smartphone or rather the mobile phone at all the sole method to unlock your phone was the insertion of a before defined Code. The further improvements regarding the unlocking methods were drawing a pattern on the screen or later unlocking your phone based on your fingerprint. Nowadays several different methods are implemented in smartphones.

In order to improve the comfortability for the user unlocking the phone by using a knocking pattern is a secure and easy method which is not developed yet. This knocking recognition is called Morse Key because it consists of “long and short knocks”. Since an actual knock is always short the time after the knock until the next knock defines if it's short or long.

GeoFi (2016)

This report describes the work done for the workshop which target is to write an app which uses sensors within the smartphone to support the user in reaching a goal. The app 'GeoFi' that is subject of this report shall support the user in being more secure when using wifi during travel situations. To reach this goal geofences are used which have to be configured for each stored wifi. Since the app shall help the user to have more mobile security an important point is that the app is easy to use to avoid errors which lead to a greater security risk than before starting to use this app. The main driver behind the idea that led to this app is that user security is breached

quite often according to newspaper and journal articles. Obviously not all possible attacking vectors can be eliminated by adding one security measure but each little step in securing the user while acting is a step in right direction.

Universal Gestrure Control (2016)

With the help of our app universal gesture control (short UGC) more comfort can be provided for controlling any device remotely.
Because of the abundance of apps for smartphones, people get confused which app is the best. Consequently, the final aim of this project is that people have the ability to record smartphone acceleration gestures and connect them with any app function. This app can be compared with the IF This Than That (short IFTTT) app. The app UGC will run as background service and run the connected app command to the recorded gesture.
The focus of this project is to validate how robust the gesture recognition can be developed in the given time.

Fitness Buddy - 🏆 Winner team of the year 2015 🏆

Fitness Buddy is an app which can measure the fitness of the user with calculating a Kettler-Based fitness value. This measurement happens in different states in usage of the external heartbeat sensor. First, the user has to reach a level of max. heart rate according to the entered body-data. After reaching this threshold, a relaxing phase measures, calculates and visualises the fitness value of the user. In context of this assignment it has been proved, that daily fitness activities improves the fitness related to the user characteristics.

Tales of Ludum (2015)

Using these capabilities, smartphones can become miniature companions that are able to help in emergencies or provide sophisticated entertainment. In the course of this project, the smartphones sensors were examined in the context of designing a multimodal, multidimensional game experience that encapsulates a variety of aspects and sensor inputs, including geo-location triggered events and tasks requiring different combinations of sensor data.
Incorporating the apparently controversial aspects of gaming and fitness, real-world role Role-Playing Game (RPG) concepts are meshed with fitness challenges and tasks in an attempt to find a common dominator and combine fun and advantageous sides of both.
As a proof of concept, this report serves as documentation for the first prototypical implementation of a battle system that requires the user to strike with the smartphone like a weapon, wherein the accelerometer is used to determine the nature of the strike and classify it according to predefined templates.

Shockpiano (2015)

The application presented in this work is called Shockpiano. The idea is to sense the strength of a vibration that is induced into a given surface when knocking on it and to translate it into an audio visual response. Different strengths of knocking shall lead to different sounds. That means that a strong knock shall play a different note than a weak knock. A knock of the same strength, but that is farther away from the device should also induce a weaker vibration. In this work it will be experimented with the distance and the strength of the knock. The data will be presented. The sound that the smart-phone plays could be any sound a smart-phone can play. From Drums to Voice Samples. This application shall play artificially generated Notes.

DSP - Dynamic Sound Profiler (2015)

This assignment report is about solving the problem of inadequate sound profiles in tedious situations with the application "DSP - Dynamic Sound Profiler". The overall idea of the application is the development of a dynamic sound volume adaption application for smartphones, by measuring and analysing real-time environment sound data. The benefits for smartphone users is the automated background adaption of individual sound profiles. The manual configuration of sound profiles will be obsolete. The state machine for the application is provided by the Smartphone Sensing Framework and has three noise and moving states, each, to be able to adapt sound profiles to the sound environment. For the detection of states and real-time data the acceleration, display state and microphone are used as sensors. The state definition and transition is most important for the application.

Track Optimizer (2014)

The name of the application developed by our team is Track Optimizer. The application mainly uses the GPS-Sensor of a smartphone to record and collect GPS-coordinates of the current location. The idea behind the application is to record and analyze tracks covered by the user. The user should have the ability to compare his tracks relating to distance or time he used for a track. Another main feature of the application is the so called Ghost Mode. This is the possibility to visualise the progress of the current track in comparison to previous tracks. The progress is displayed live on the smartphone display and user becomes an illusion to compete against himself. Through this feature the user can improve the time he needs for a certain track.

Black Box – 🏆 Winner team of the year 2014 🏆

Because of the high accident rate in public traffic a project was started that uses modern smartphones running an Android operating system, to detect accidents and give the driver the possibility to do a fast emergency call, so that in case of injury rescuers are informed as fast as possible. For detecting accidents smartphone integrated sensors like an accelerometer are used. After having analysed the sensor’s data characteristics a first prototype version was designed and implemented. Several accident simulation tests showed that the actual implementation works really good. There also have been done a couple of tests using the application in real traffic situations. Of course there was no real accident situation, but the application was robust against several traffic situations like hard stops or potholes, because there was no accident detected. The testing of the application showed that using smartphones and their integrated sensors is really powerful and can be used for detecting accidents. Future enhancements of the app could be the classification of an accident to determine the kind of accident, like a rollover or a side crash and depending on the heaviness of the accident the rescuers could be informed automatically.

APPnormal Driver – 🏆 Winner team of the year 2014 🏆

We decided to use the data, which is captured by the sensors, to score the users driving behaviour. We will use the accelerometer to detect dangerous or highly petrol consuming driving actions. The GPS interface will be used for debug purposes and later on to detect a movement

of the user. The resulting Android application is named APPnormal Driver. You can see the logo of the app ingure 1. The main function of the app will be to give you an overview of how good or bad your driving behaviour is. As a further extension you can compare your results with other users in a competitive way. The one with the safest and most moderate will be the leader of the high-score list.

Gesture Speaker (2014)
The project GestureSpeaker was focused on using a fast and natural way to express the user's words: with hand gestures. Using hand gestures in a conversation makes the speech-impaired user feel more comfortable than when typing, because most of the people use hand gestures when they are speaking.
The most important part of the project was the development of the gesture recognition, which is related to pattern recognition. For this first analysis, the project concentrates only on the use of the accelerometer sensor. That means the application GestureSpeaker processes arrays of acceleration values in the three axes as gestures. To recognise the similarities among gestures two different methods were studied. The most important problem to solve was the poor accuracy of the device's sensors, which could be detected through data analysis.

Running Beats (2013)
This assignment presents an idea of a control mechanism for joggers and runners. Through a simple user interface, users can setup their desired running speed, control it while jogging and in the end receive a detailed data analysis for the running duration. The control mechanism is solved with audible signals to give the users the possibility to concentrate on their training. During this workshop with a given time of four weeks the idea was created and an implementation was realised on an Android mobile device (Google Nexus). Additional a standalone application (Java) was implemented for a further data analysis of the captured data.

Unusual Situation Alarming (2013)
The aim of this project is to develop an android application named Unusual Situation Alarming which uses different sensors from an android device to identify such unusual situations of a child. A child plays in a park and through whatever event happens the child behaves unusual like leaving the park, driving in a car or stop moving for a longer period of time. To identify these unusual situations, the research focus on the gps sensor to identify if the child leaves the park and if the child moves faster than it is possible without a motorized vehicle.

Coach Fit – 🏆 Winner team of the year 2013 🏆
This project examines to what extent mobile devices can be used to help athletes improve their performance using the built-in sensors of Android smartphones. The goal is to develop an Android App which helps athletes as a personal fitness trainer doing exercises like pushups, pull-ups, squats and sit-ups. This has been done by examining the usefulness of different Android smartphone sensors, such as accelerometer, gyroscope or the rotation sensor to fulfill the above mentioned objectives. Upon examination of these Sensors we have developed an app which helps us to collect and analyze data to determine the athlete's fitness condition. Based on information collected the app gives clear audio recommendation to the athlete to improve his exercise and track his performance over time.
To make the app more comfortable an efficient voice recognition service was implemented, to allow starting and stopping different exercises. By doing so, we have given the athlete the possibility to free-handed control the application during his workout session.

Simple Self Care (2013)
With the project Simple-Self-Care a real world advanced mobile phone application is developed. The Simple-Self-Care application is running on the mobile of a needy person and can protect

the person in the case of a collapse at home, by calling a second assisting person. The application helps handicapped people to stay longer at home in life and prevents a residential care of the needy person. Therefore the life quality is improved and extended. The mobile device with the application is a real chance for a society with an increased expectancy of life time.