Covid-A App: Hand Washing Recognition

EUGENIA PETRANGELI, DANIELA COMOLA, LEONARDO FONTANELLI

Washing Hands Module

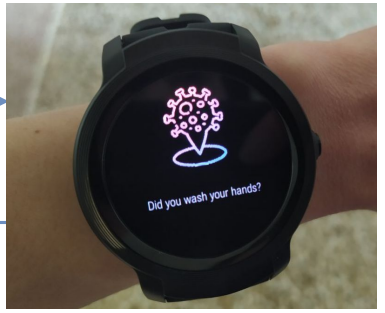
Two applications are needed:

- Collecting Data Application
  - Collect Data
  - Features Extraction
  - Feature Selection
  - Train the classifier
- CovidApp Application

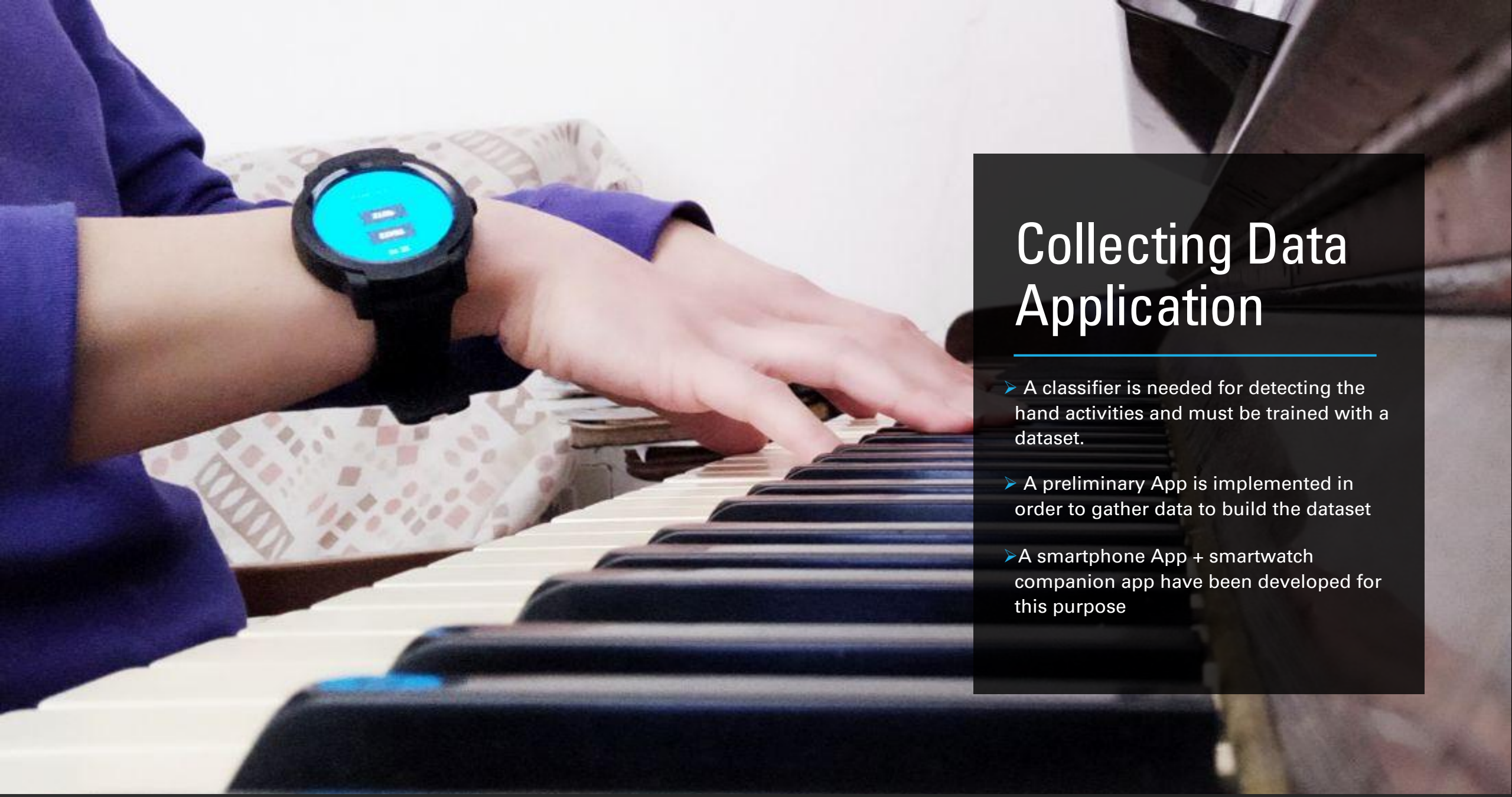
Extract Features
Classify

Start/Stop Samplling

Data



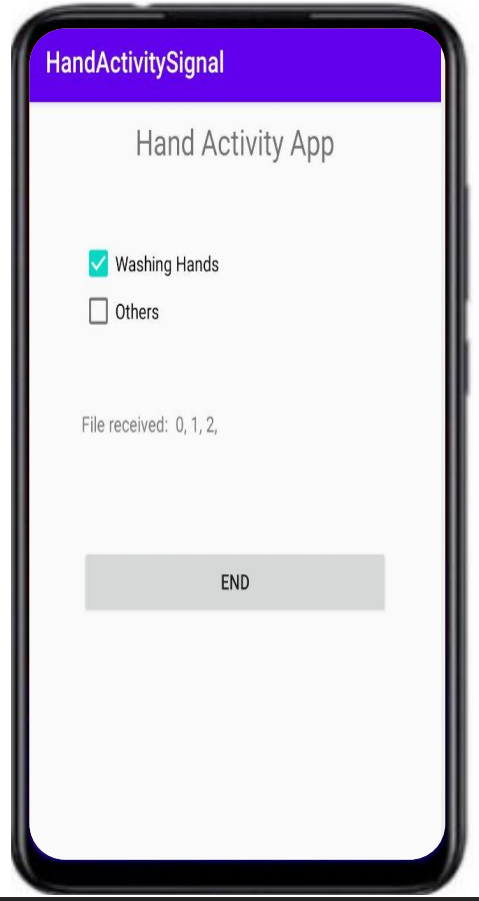
Sampling data



Collecting Data Application

- A classifier is needed for detecting the hand activities and must be trained with a dataset.
- A preliminary App is implemented in order to gather data to build the dataset
- A smartphone App + smartwatch companion app have been developed for this purpose

Structure of the System



Experiments: Energy Consumption vs Accuracy

Sensors used are «non-wakeup»:

- Accelerometer
- Gyroscope
- Linear Acceleration
- Rotation
- Gravity

Experiments performed:

1. Experiment 1 -> Energy saving but not so accurate
   - CPU can be in «Suspend» mode during the fast sampling
     - many data are lost because the hardware FIFO is too small
     - CPU in »Suspend» mode reduce the power consumption by 100 times or more
2. Experiment 2 -> More accurate at the cost of increased power consumption
   - The power state of the application is in «PARTIAL_WAKE_LOCK» level when the sensors work in fast sampling.
     - More Energy hungry but the classifier is more accurate

Collecting Data

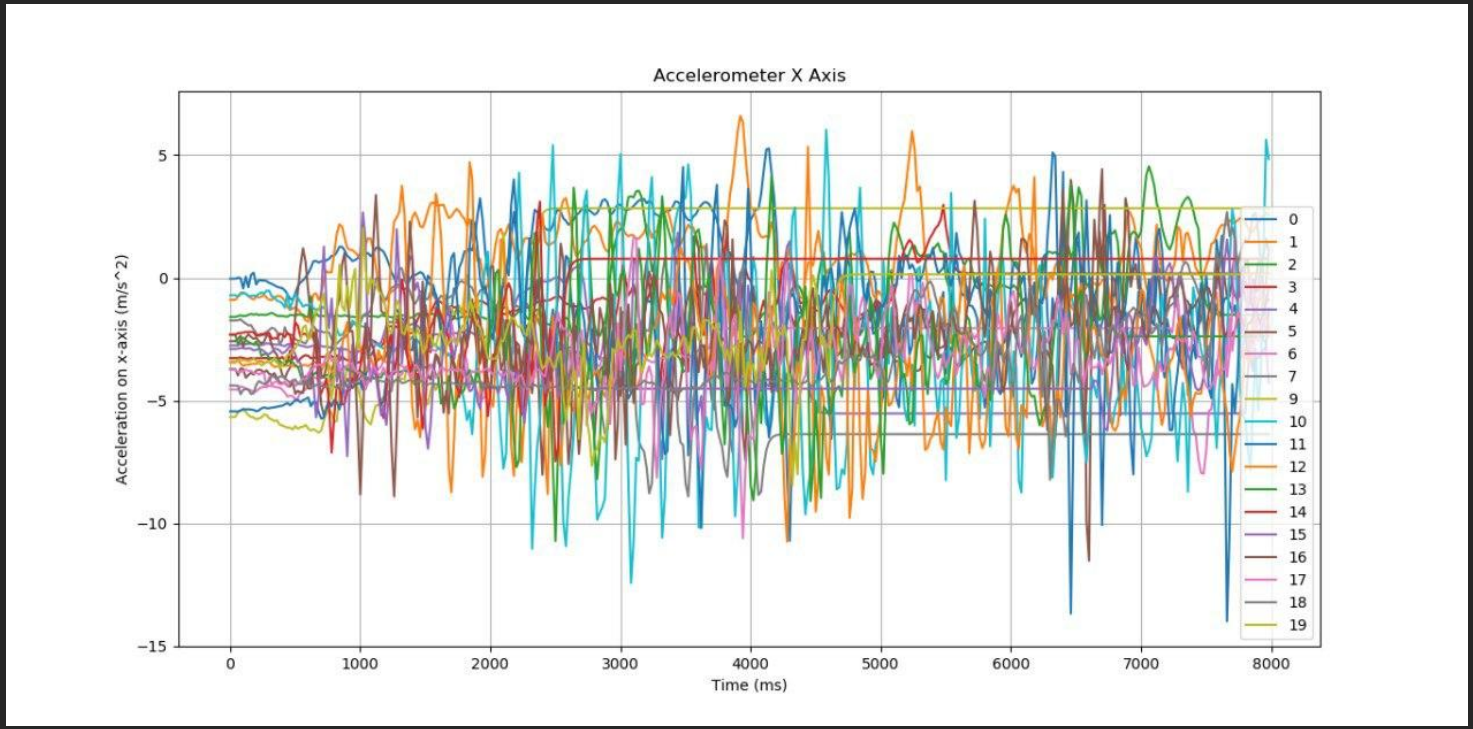
- Traces of 40 seconds at *50Hz* are sensed and then divided into fragments of 8 seconds and are labeled with “Washing Hands” or “Others”.
- Fragments of 8 seconds are taken in consideration as input for the classifier, so that it could be able to recognize a whole sequence of actions
- In order to better characterize the different actions that make up the single fragment, features are then computed on contiguous windows of 2 or 4 seconds for increasing the precision of the classifier and distinguish atomic actions.

Feature Extraction

- For every window, the feature considered are: **Mean, Standard Deviation, Skewness, Kurtosis.**
- The first 10 samples are discarded as “warm-up period” of the signals.
- Two different datasets are built, both in the wake-up and non wake-up sensors cases, in order to assess the accuracy of both and choose the best one.

Missing Value Replacement

- For the “non-wake-up sensors” experiment, there were some missing values in the signals.
- A mechanism for the replacement is needed in order to cover the values in order to have a coherent sampling of the signals, to have a temporal correspondence of the windows for every signal.
- The signal is inspected and if a missing value is detected, it is replaced with the mean of the following sensed sample and the previous.
- This technique affects the accuracy of the classifier when there are a lot of missing values because, as shown from the graphs, straight lines are created. Therefore, features are also less significant.



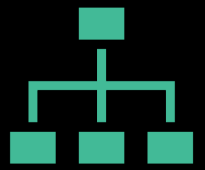
Classification – Feature Selection

- The Feature Selection phase has been performed with the Weka Software.
- 4 *".arff"* input files have been generated from the previous phase. For "wake-up sensors" and "non-wake-up sensors" and for 2 and 4-seconds windows.
- As Selector has been used *InfoGainAttributeEval* with 10 cross fold validation and for search method the *Ranker* are resulted as the best after many test.
- The features that were selected with *rank > 0* have been chosen for training.

Classification – Classificator Choice

EVERY TRAINING TEST HAVE BEEN REPEATED FOR 10 TIMES WITH 10 DIFFERENT SEEDS, AND THE AVERAGE OF EVERY TEST IS USED TO COMPARE THE PERFORMANCE OF THE CLASSIFIER.

AS PERFORMANCE METRIC IT HAS BEEN USED THE “TRUE POSITIVE RATE”.



STARTING FROM THE *MULTILAYER PERCEPTRON*, THE BEST CLASSIFICATOR WAS THE *RANDOM FOREST*.

Classification Training

- To assess the best model, the mean of the True Positive rates for 10 experiments were compared.
- Considering the ”wake-up sensors” experiments, the resulting classifiers deriving from the 2 and 4-seconds windows datasets were compared.
- 2 seconds windows:
  - TP rate for Washing Hands: 119,8 / 125
  - TP rate for Others: 142,2 / 150
- 4 seconds windows:
  - TP rate for Washing Hands: 121,3 / 125
  - TP rate for Others: 145,2 / 150
- The best model resulted to be the one for the 4 seconds windows with an accuracy of 97,454%.
- The “non-wake-up sensors” experiment was discarded because, even if the accuracy was good, the performance in the real-life tests was worse.

Classification – Implementation in Android

- For implementing the model in Android, the Weka API has been used.
- The ".model" file has been imported by placing it in the "asset" folder and it can be retrieved by calling *context.getAssets()* method.
- When the smartwatch sends the data that must be labeled, the smartphone prepares the *"unlabeled.arff"* file, computing the features selected in the previous steps, that will be given as input to the classifier using the *Instances* class.
- With the *"classifyInstance()"* of the RandomForest class applied on the Instance, we obtain a value that corresponds to the classified activity: if the value is equal to 0.0, the activity is classified as *"Others"*, otherwise if it's equal to 1.0, the activity is classified as *"Washing Hands"*.

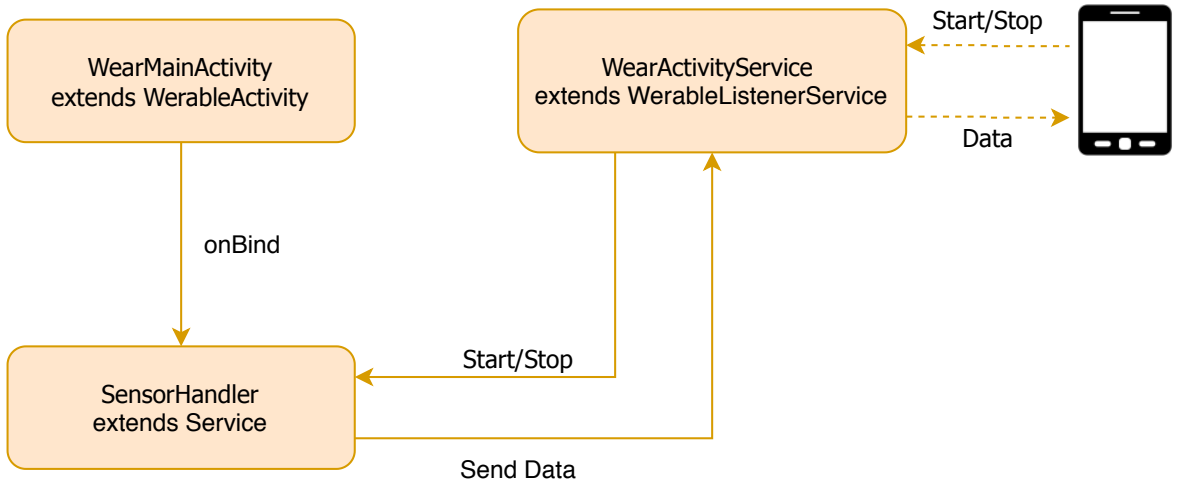
CovidApp Application – Wear Module

WearActivityService

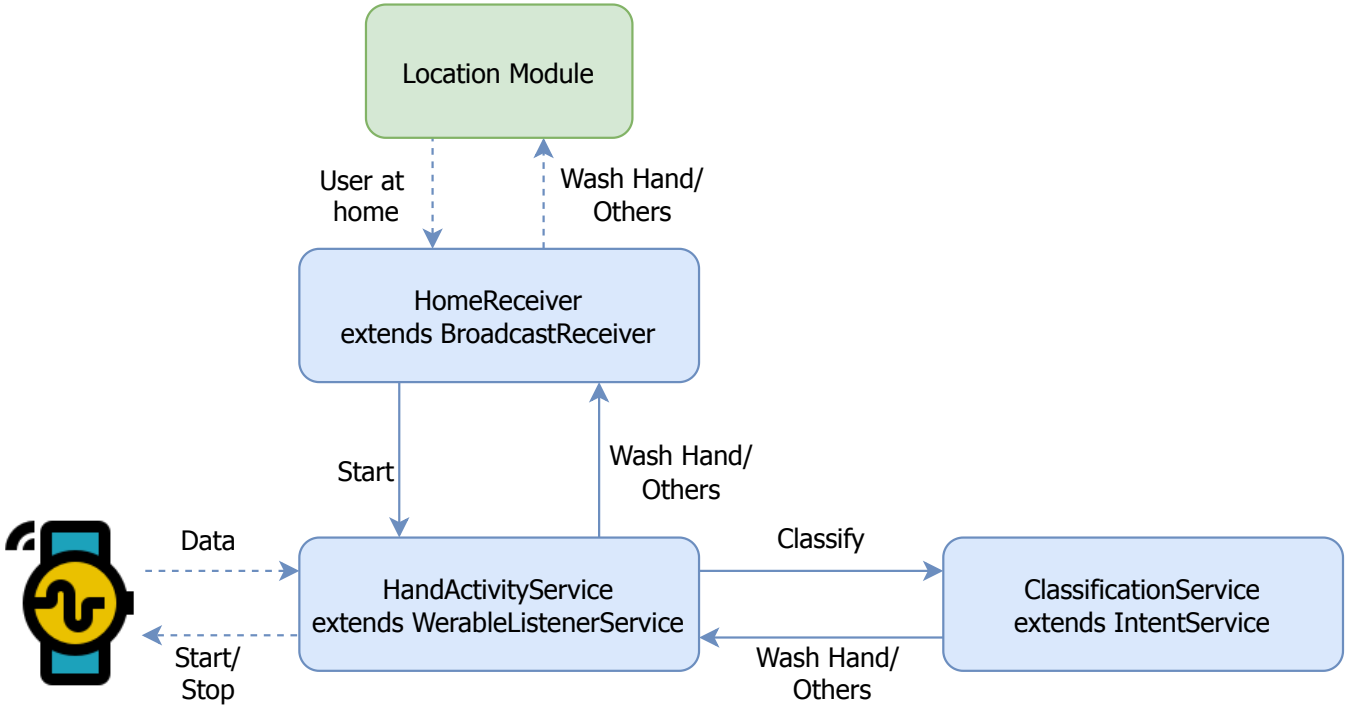
- Handles data sent from the mobile phone. It will start the sensing or stop it, according to the request from the phone
- Send collected data to the mobile phone

SensorHandler

- Starts or stop the sensing according to the command received by the WearActivityService
- There are two timers:
  - A longer one used to specify the maximum sensing period.
  - A smaller one related to the period in which samples are collected with a faster rate



CovidApp Application – Mobile Module



HomeReceiver

- Starts hands activity detection
- Filters intent containing results when an activity is recognized by the classifier.

HandActivityService

- Starts sampling on smartwatch when the user is at home
- Stops sampling on smartwatch when the timer expired/wash hand activity is recognized/ application is closed.
- Starts features extraction and classification operations.
- Sends the result of the activity classification in broadcast.

ClassificationService

- Performs the features extraction and the classification
- Sends the result of the classification

Conclusions

- Learn more about the management of the wake-up and non-wake-up sensors to ensure that data is not lost even without leaving the CPU always active during the fast sampling. We could keep the AP in suspend mode for a certain period of time and send the collected data over this period all at once. We could analyze the size of the hardware FIFO dedicated to each sensor to get an estimation of how much to keep the CPU asleep.
- Future work could be to extend our module for using it outside the house. We could measure how many times a person washes his hands during a day away from house, also based on the action he does.
- Increase the dataset taking in consideration more people in the training phase in order to have a more variegated dataset.

Thank you for the attention

QUESTIONS?

DEMO