Activity recognition system with ESP8266

Jakob Maležič[a,1]

[a]University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, SI-1000 Ljubljana, Slovenia

The manuscript was compiled on June 24, 2021

In this report, we describe the implementation of an activity recognition system on ESP8266. We focus on how to recognize certain types of human physical activities using acceleration data generated by user's movement of the EPS8266 board. We implement a recognition system that uses a digital low-pass filter in order to isolate the component of gravity acceleration from that of body acceleration in the raw data. The system is trained and tested experimentally. For classification of the activity, a simple server is implemented with a random forest classifier that uses various statistical features.

Project structure The project is split into two parts: gathering accelerometer data on the ESP8266 and activity classification on the server. The two communicate via http. The program on ESP8266 is implemented in C++, while the server is implemented in Python using the Flask library.

Gathering data on ESP8266

Structure. The program for gathering accelerometer data on ESP8266 is located in the `src` folder and contains the following files and folders:

- `Accelerometer.cpp` - a simple wrapper for reading accelerometer data
- `WebServerUtils.cpp` - utility functions for the web server that serves as the user interface on the board
- `src.ino` - main program
- `data` - folder that contains the html files that present the user interface on the board. This files are transffered to the file system of the board and are then read using SPIFFS filesystem.

Main program. As every Arduino program, our program contains a setup and a loop function. In the setup function, the web server on the board, that serves as the user interface for using the system, is created. After the web server is started, we then calibrate the accelerometer.

In the loop function we handle the requests that come to the server and send data to the classification server when requested. For sending data, we use `ArduinoJson` library that allows us to transform data into json format. The data is then sent to the classification server via http using the post method.

Web server. The web server contains four pages: login, record, predict and root. All web pages are written in seperate files using html, css and javascript. The javascript is mostly used for making requests to the classification server as it works faster that way, because it runs on the users's browser.

Login page. Login page only contains a simple input form for logging into the system. The default username and password are set to *admin*. Once logged in, the server sets a cookie and after that, it always automatically redirects to the root page, whenever the user opens the login page. If the user logs off, the cookie is deleted.

Root page. Root page is a very simple menu, so the user can navigate through pages. By clicking the button disconnect, the user can log off.

Record page. Record page allows the user to record the activity and send the accelerometer data to the classification server. This data can then be used for training a classification model. The page contains input text field, where the user can set the recording name, a dropdown menu with different activities for labeling the data and three buttons:

- `record` - starts recording the accelerometer data. The button also changes from *Start recording* to *Stop recording* and can then be clicked to stop the recording.
- `save` - sends a request for saving the data on the classification server. The data is saved into a `csv` file.
- `delete` - deletes the accelerometer data that was sent to the server.

Predict page. Predict page contains only one button for starting the prediction of the users physical activity. After clicking the button, similarly to recording the accelerometer data, the accelerometer data is sent to the classification server which then makes a prediction. The prediction is then showed on the page and updated every second, by making a request to the server. The page also displays three graphs that show the accelerometer data through time by x, y and z axis. They are drawn with javascript library called `highchart`.

Classification

Classification files are located in the `backend` folder:

- `backend.py` - Python flask server for recording data and making predictions
- `model.joblib` - exported random forest classification model
- `prediction.py` - a simple python script for generating the aforementioned classification model
- `data` - folder that contains accelerometer data recordings that were used to train the classification model

[1]To whom correspondence should be addressed. E-mail: jm6421@student.uni-lj.si.

Classification server. The classification server is written in Python using the Flask library. It's a very simple server, that contains the following endpoints:

- `POST /data` - accepts accelerometer data in JSON format, extracts features and labels, and saves them to the array.
- `PUT /data` - accepts the name of the recording and saves the accelerometer data to a file
- `DELETE /data` - deletes all current data in the array
- `POST /prediction` - accepts accelerometer data in JSON format and makes prediction based on it
- `GET /prediction` - returns the latest prediction result. It is called every second when we start predicting on the board in order to update the prediction.

Feature evaluation and feature extraction. Feature evaluation and feature extraction was done similary to what is described in (1). First, a low pass filter is used on the ESP8266 board in order to separate the AC component from the DC component in each time series. Then on the server, the following features are extracted: mean along z-axis, average peak of frequency along x-axis, y-axis and z-axis, variance of the peak frequency, standard deviation along x-axis, y-axis and z-axis, root mean squared along x-axis, y-axis and z-axis, correlation between z-axis and y-axis, and MinMax along x-axis, y-axis and z-axis.

Classification model. Classification model is trained with Random forest classifier, from the `sklearn` library. After the model is trained, it's dumped to `joblib` file, which is then used by the server for making predictions.

Usage

For running the program you will need the ESP8266 board with the MPU9250 sensor. You will also need to ArduinoIDE with ArduinoJson library installed. In the `src/WebServerUtils.h` you have to change the Wifi credentials and optionally, you can change the username and password. Lastly, you have to change the `serverUrl` in `src/src.ino` to your server's IP. Before running the program, you also need to upload the `src/data` folder to the board. You can do this with `Arduino IDE -> Tools -> ESP8266 Sketch Data Upload`.

To run the server, you will need Python installed. After that, you have to install all the requirements specified in the `requirements.txt`. You can install them using `pip` with the following command: `pip install -r ./implementation-extraction/requirements.txt`. You can then run the server with `python ./backend.py`, which will start the server on port `8080`.

The code and more detailed setup guide can be found on https://github.com/Blarc/activity-recognition-esp8266. For best results with the pretrained model the user should attach the board to a powerbank and hold it as shown in the figure 1.

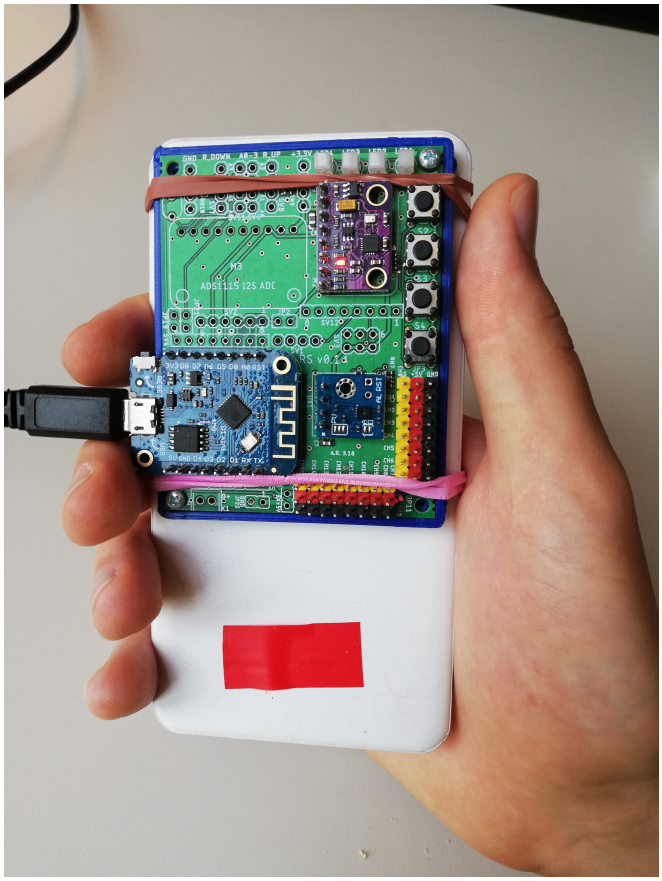

Fig. 1. Abstract illustration of graph structure

1. Akram Bayat, Marc Pomplun, and Duc A. Tran. A study on human activity recognition using accelerometer data from smartphones. *Procedia Computer Science*, 34:450–457, 2014. ISSN 1877-0509. . The 9th International Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.