PROJECT AIOT: HUMAN GESTURE RECOGNITION

Participants:ΔΗΜΗΤΡΙΟΣ ΜΠΑΡΟΥΤΗΣ 1084549, ΓΕΩΡΓΙΟΣ ΜΗΤΣΑΙΝΑΣ 1084527

1. We executed the process through bluetooth connection with pc. We implemented a sensor\_data\_combined.py script which includes the metawear python packages so as to efficiently collect the data. Firstly, we took the mac address of the wearable and defined the format of the output file(name, type, path). Moreover, we defined the data\_log array which appends the rows of the acceleration and gyroscope metrics. In particular, we used functions that concentrate acc & gyro data acc\_callback() & gyro\_callback() and a function that combines them combine\_and\_store().The acc\_callback does that by casting the acc data to acc.x, acc.y, acc.z coordinates which are saved to latest array too. The same procedure takes place in gyro\_callback. They both call the combine\_and\_store() in the end. This function combines the datasets and inserts them in data\_log including a timestamp.The stream runs for 10 seconds. Therefore, we connect the device, tune acc & gyro settings (100 Hz) and write the data to csv files(classes).
2. We used MongoDB configuration from config.yml so as to insert the csv datasets with classes to the remote database aiot\_course which has the data collection. In other words, we took the path of the data directory and for each subdirectory nose rubbing, scratch and wave that was found we took the corresponding csvs. In addition, we formatted each document in mongo to have fields data with subfields the metrics, label with class name and datetime. In conclusion, we inserted the csv in mongo in document form (mongo.py).
3. We initially imported the required libraries and the functions from utills.py and utils\_visual.py to process the data and visualize the results. We also loaded the documents from mongo db and made them as a single dataframe with data and label fields. Then, we ordered the fields in our needs with df\_rebase(). Besides that, we applied a low pass filter with apply\_filter() so as to minimize the spikes from steep movements that may have occurred from the collection process. What’s more, we created window instances by using sliding\_window\_pd() with size ws=20 and overlap=20 because we observed that each movement had about 20 samples per period. Additionally, we filtered the numeric instances again with a low pass filter. After this, we flattened the numeric data with flatten\_instances\_df() and renamed the columns of the frame with the labels too, using rename\_df\_column values(). For instance, we flattened the data because we needed to split them in X\_train, X\_test, y\_train, y\_test datasets at random with tain\_test\_split(). Apart from that, we normalized the X datasets with MinMaxScaler() and we implemented 3D PCA fitting to them with PCA(n\_components=3). We also needed to plot the components so as to see if they are discriminated clearly in order to have good results. All in all, we trained and evaluated SVC Classifier, Random Forest Classifier and GridSearchCV for SVC. The training parameters used for its model are formulated in config.yml. We evaluated each model in AI\_model.py by using classification\_report (y\_test,y\_pred\_classifier) and confusion\_matrix(y\_test,y\_pred\_classifier, labels =classifier.classes). At this point, we refer to 0=nose rubbing , 1=scratch and 2=wave. From the results, we conclude that the best classifier is GSCV with accuracy 90%, second the RFC with 89% and worst the SVC with 86%.We also observe that all classifiers have better results at wave prediction than the others because scratch and nose rubbing are similar. Precision, recall, f1-score and the confusion matrix can be seen by executing the script.