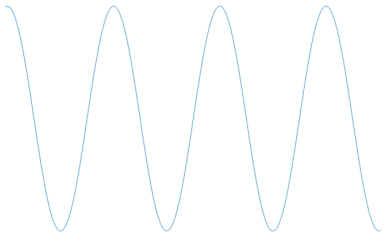
Design Specification

Robin Mannberg, Martin Andersson, Emma Beskow, Ella Grundin,
Joel Nilsson, Gabriel Suihko and Jianxin Qu

October 14, 2021

Version 1.0



Status

Reviewed	The project group	2021-10-14
Approved	Danyo Danev	2021-10-14

Project Identity

Group E-mail: tsks23group1@gmail.com

Orderer: Danyo Danev
Phone: +46 (0)13 28 13 35
E-mail: danyo.danev@liu.se

Customer: Danyo Danev
Phone: +46 (0)13-28 13 35
E-mail: danyo.danev@liu.se

Supervisor: Ema Becirovic
Phone: +46 (0)13-28 19 11
E-mail: ema.becirovic@liu.se

Supervisor: Jianan Bai
Phone: +46 (0)13-28 26 13
E-mail: jianan.bai@liu.se

Course Responsible: Danyo Danev
Phone: +46 (0)13-28 13 35
E-mail: danyo.danev@liu.se

Participants of the group

Name	**Role**	**E-mail**
Robin Mannberg	Project Manager	robma370@student.liu.se
Martin Andersson	Test Manager	maran594@student.liu.se
Emma Beskow	Document Manager	emmbe571@student.liu.se
Ella Grundin	Hardware Manager	ellgr825@student.liu.se
Joel Nilsson	Chief of Design	joeni078@student.liu.se
Gabriel Suihko	Graphics Manager	gabsu290@student.liu.se
Jianxin Qu	Software Manager	jiaqu952@student.liu.se

CONTENTS

DOCUMENT HISTORY

Version	Date	Changes made	Sign	Reviewer
1.0	2021-10-14	Sent to customer.	The project group	The project group
0.4	2021-10-12	Explicit mention of maximum likelihood estimates in hardware section. Minor error corrections.	The project group	Supervisors
0.3	2021-10-08	Correction of hardware section. Minor error corrections.	The project group	Supervisors
0.2	2021-10-05	Introduction and implementation details added.	The project group	Supervisors
0.1	2021-09-23	First draft.	The project group	Supervisors

1 INTRODUCTION

The research area of human activity recognition and detection in indoor environments based on analysis of channel state information (CSI) of radio frequency (RF) communication links has recently seen an upswing owing to the development of deep learning (DL) methods [1]. The development has grown following the failure of conventional machine learning (ML) approaches to solve more complex human activity recognition tasks. However, it is still of interest to investigate the performance of conventional ML models. A large number of detection and recognition methods have been developed, and even deployed commercially, by the use of DL methods, for instance, sleep monitoring, vital sign monitoring, fall detection, localisation and tracking, activity monitoring and crowd counting. RF sensing provides a less intruding, less energy-consuming way to detect and recognise human activity that is independent of both light conditions and line-of-sight [2]. It is possible to integrate these solutions into existing WiFi-equipment. All of this contributes to the attractiveness of human activity activity recognition or detection by RF sensing as an area of research.

This project aims to develop a system for detection of static and dynamic indoor environments. The system will use ADALM Pluto Software-Defined Radio (Pluto SDR) devices to estimate a channel in an indoor environment. After collecting the CSI, ML algorithms will be used to classify different types of environments as static (no moving object or person present) and dynamic, where the latter category may be broken down into several specific types of motion of an object or a person. The performance of the different models will be evaluated and compared. The system performance in other indoor environments will be tested.

1.1 Related Work

It is known that human activity in an environment causes interference in radio signals. There are several ways to use the CSI for activity recognition [1]. Early on, received signal strength indicator (RSSI) was popular as a feature for activity recognition, but it is not as refined as the CSI that is commonly used now [3]. The DL approach to human activity detection and recognition shows great promise, and the survey by Nirmal et al. [1] displays that there are several techniques that have been successfully applied to the task. Some DL methods (e.g., recurrent neural networks, RNN) focus on using the temporal patterns in the data [2], while other methods (e.g., convolutional neural networks) are suited for spatio-temporal patterns [4]. A downside of the DL approach is that it may require time and resources to be spent on data collection and labelling, and on training the algorithms [5], [6]. On the other hand, DL requires less attention on manual feature design than conventional ML methods.

Simpler neural network architectures (e.g. multi-layer perceptron, MLP) show potential at detection, localisation and recognition tasks [1]. CSI amplitude and phase can be used as features in MLPs. However, preprocessing the phase information may take some effort, especially if using commercial WiFi equipment, since the carrier frequency offset may render the phase estimates too noisy [7]. Sen et al. [8] used a linear transformation to process phase data used in an indoor localisation task, but the method was not applicable in the case of Wang et al. [7]. Instead, Wang et al. estimated the phase shift of the signal paths by inferring the path length change from the channel frequency response (CFR) power.

Ding and Wang [2] proposed a two step detector that used a decision tree that takes subcarrier variance and correlation coefficient from raw CSI data to detect human activities. A recurrent neural network was used to classify the activity type. The detector needed to be retrained in order to be successfully deployed in a new environment.

Generalising detection methods to different indoor environments and deploying the system outdoors are challenging tasks. Many studies focus on using WiFi-signals, as the techniques may be deployed in commercially available (and ubiquitously present) hardware platforms [1]. WiFi is limited in outdoor environments, making it unfeasible for long-range detection. Experiments with Long Range (LoRa) signals successfully located a person in a larger area (localisation error was less than 4.6 meters in 90% of cases) using a single transceiver pair [9]. This study will focus on activity recognition in indoor environments using signals in different frequency bands.

Minor changes in the (static) environment may have a significant impact on signal propagation [3], [5]. To overcome the need to retrain the system every time it is deployed in a new indoor environment (also called domain adaption), autoencoders (AE) can be deployed [1]. Chen et al. [3] used AEs to generalise CSI features used for indoor localisation. They established a CSI profile in a reference environment and implemented an algorithm for detecting a new static environment. Functions for transforming the new environment features into the reference environment features were estimated and applied to data if a new environment was detected. These features were given as input to an AE. The AE was trained to reproduce and emphasise critical features (CSI features relating to the new environment are suppressed), which boosted localisation performance. Wang et al. [7] leveraged environment independent motion speed measurements to achieve 80% recognition accuracy in a domain that the model had not been trained in. The key was to realise that the speed of the path length change is similar for the same activity regardless of the static environment. This study will investigate the the domain adaption capability of different methods.

Conventional (shallow) ML methods, including support vector machines (SVM), random forests, and k-nearest neighbours (kNN), have been successfully deployed at small scale activity recognition tasks [1]. The feature quality is important for shallow learning methods, and performance can be boosted significantly by using manually designed features compared to using raw or preprocessed CSI data [5]. Features can be designed through domain knowledge or applying dimension reduction techniques, such as principal component analysis (PCA), independent component analysis (ICA), or singular value decomposition (SVD). The unsupervised learning method k-means clustering can be applied for feature verification [10]. Wang et al. [7] created one hidden Markov model (HMM) for each activity, and used the likelihood of the observations (feature vectors), picking the activity that has the highest likelihood of producing the data.

2 SYSTEM DESCRIPTION

This document provides the reader with an overview of the components of the system, and how they interact to achieve its purpose. The complete set of system requirements can be found in [11]. The purpose of the system is to detect dynamic events in indoor environments by using ML algorithms to analyse signals from software defined radio (SDR) equipment. While SDR has no unique definition, it implies that the sent waveform can be modified by the software without changing the underlying hardware or environment [12]. This flexibility allows the system to optimise detection in different environments. The detector will be built from two, or more, Pluto SDR transceivers connected to a host computer. Following channel estimation, channel state information will be collected and labeled according to the event which it represents. The collected data will be used to train supervised ML algorithms. The detector will be designed

with the aim to make (accurate[1]) classification decisions in binary and multiple hypothesis scenarios. The system is divided into three subsystems as shown in Figure 1. The hardware subsystem is presented in Section 3, the software in Section 4 and the user interface in Section 5.

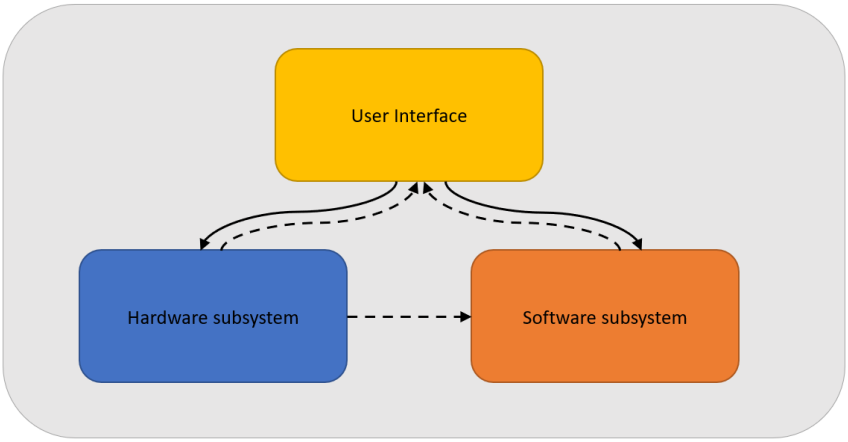


Figure 1: System overview. Solid lines indicate control signals, dashed lines indicate data flow.

3 HARDWARE

The hardware subsystem will consist of Pluto SDR devices with specifications according to [13]. These devices will be connected to host computers that handle their transmit and receive functionality. The software used to calibrate the Pluto SDR devices and to estimate the channel(s) between transmitter(s) and receiver(s) will be considered a part of the hardware subsystem in this project. The output of the hardware subsystem is the estimated CSI, that will be further processed by the software subsystem, as well as a label of which environment the data was collected in. If nothing else is stated, this section will consider single-carrier transmission.

The Pluto SDR devices allow us to abstract away the analog signal processing and it is thus sufficient to consider the data transmission as a discrete memory-less channel that takes IQ symbols (complex numbers, where the real and imaginary parts are called in-phase (I) and quadrature (Q) components, respectively) as input and gives distorted IQ symbols as output. However, the analog signal propagation properties can still be used in the analysis. The Pluto SDR devices will be fixed in place in the indoor environment during each data collection session, which means that the channel can be assumed to be time invariant in the static environment and that it will be time variant when the environment is dynamic. During the data collection phase, well chosen symbols known at both transmitter and receiver will be sent and based on the corresponding received symbols, the channel can be estimated at the receiver side. This is repeated until a sufficient amount of CSI data has been gathered. Each of these channel estimates will be assigned a label specifying the type of event that occurred during the corresponding data collection.

1 see the requirement specification [11].

3.1 Channel State Information

In wireless communication, the radio waves (signals) emitted from the transmitter antenna(s) reach the receiver antenna(s) along multiple paths. This section start with a derivation of a linear system model that handles multipath effects. Then the channel is estimated by considering sampled (discrete time) versions of the continuous-time signals. Finally, the channel estimates are evaluated.

3.1.1 *Signal Propagation*

The simplest case is to study the channel response of a sinusoidal input $x(t) = \cos(2\pi f_0 t)$, where f_0 is a chosen frequency. Assuming that there is no noise present, the received signal $y(t)$ can be written as in [14]:

$$y(t) = \sum_i a_i(t)x(t - \tau_i(t)), \tag{1}$$

where $a_i(t)$ and $\tau_i(t)$ are the overall attenuation and propagation delay at time t from the transmitter to the receiver on path i. The overall attenuation is affected by the reflection and diffraction in the environment, the transmitter and the receiver antenna patterns, as well as the transmission distance of the signals. In (1), we see that the channel is linear in each time instance and it can be interpreted as a varying function of t. The received signal $y(t)$ can be described by the convolution between the channel impulse response $h(\tau,t)$ at time t and the transmitted signal $x(t)$. The input and output relationship is given by [14]:

$$y(t) = h(\tau,t) * x(t) = \int_{-\infty}^{\infty} h(\tau,t)x(t-\tau)d\tau. \tag{2}$$

By comparing (1) and (2), we see that the impulse response for the fading multipath channel is

$$h(\tau,t) = \sum_i a_i(t)\delta(\tau - \tau_i(t)). \tag{3}$$

In the static case when the transmitter, receiver and the environment are all stationary, the attenuations $a_i(t)$ and propagation delays $\tau_i(t)$ do not depend on time t, and we have the usual linear time-invariant channel with an impulse response

$$h(\tau,t) = \sum_i a_i\delta(\tau - \tau_i). \tag{4}$$

For the time-varying impulse response $h(\tau,t)$, the time-varying frequency response is

$$H(f;t) = \int_{-\infty}^{\infty} h(\tau,t)e^{-j2\pi f\tau}d\tau = \sum_i a_i(t)e^{-j2\pi f\tau_i(t)}. \tag{5}$$

In the static case, or during short time intervals in a dynamic environment, the channel is time-invariant and it reduces to the usual frequency response by removing the dependence on t. In real systems there will of course also be noise present, which would change the equations in this section slightly.

3.1.2 *Channel Estimation*

As stated earlier, the analog signal propagation can be abstracted away when using the Pluto SDR devices. It will be assumed that the transmission is divided into time-frequency parts that fit into a coherence interval, which is the time-frequency interval during which the channel is time invariant and its frequency response is constant. The single-input single-output (SISO) transmission of a packet of N IQ symbols, where N channel uses are assumed to fit into a coherence interval, can be modelled as

$$\mathbf{y} = \sqrt{\rho}h\mathbf{x} + \mathbf{w}, \tag{6}$$

where $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{w}$ are the $1 \times N$ vectors of received symbols, transmitted symbols (assumed to be known) and independent and identically distributed (i.i.d.) $CN(0,1)$ noise entries, $h \in \mathbb{C}$ is the channel (assumed to be a random variable with a constant value during each packet of N channel uses) and ρ is the signal-to-noise ratio (SNR). Assuming that $\mathbf{x}$ fulfills $\|\mathbf{x}\|^2 = 1$, the channel can be estimated with the maximum likelihood estimator

$$\hat{h} = \underset{h}{\operatorname{argmax}}\{p(\mathbf{y}|h)\} = \underset{h}{\operatorname{argmax}}\{p(\mathbf{y}\mathbf{x}^H|h)\} = \underset{h}{\operatorname{argmin}}\{\|\mathbf{y}\mathbf{x}^H - \sqrt{\rho}h\|^2\} = \frac{\mathbf{y}\mathbf{x}^H}{\sqrt{\rho}}, \tag{7}$$

where $p(\mathbf{y}|h) \sim CN(\sqrt{\rho}h\mathbf{x}, \mathbf{I}_N)$ and $p(\mathbf{y}\mathbf{x}^H|h) \sim CN(\sqrt{\rho}h, 1)$ are the probability density functions of $\mathbf{y}$ and $\mathbf{y}\mathbf{x}^H$, respectively. When using multiple antennas, the channel between each pair of transmitter and receiver can be seen as a SISO channel. The transmission from the M_t transmitters to the M_r receivers over N channel uses can be written in matrix form as

$$\mathbf{Y} = \sqrt{\rho}\mathbf{H}\mathbf{X} + \mathbf{W}, \tag{8}$$

where $\mathbf{Y}$ and $\mathbf{X}$ are the $M_r \times N$ and $M_t \times N$ matrices of received and transmitted symbols, where each column consists of the transmitted/received symbols in one channel use, $\mathbf{H}$ is an $M_r \times M_t$ channel matrix where each entry is the channel between one pair of transmitter/receiver and $\mathbf{W}$ is an $M_r \times N$ matrix with i.i.d. $CN(0,1)$ entries. To estimate the channel, orthogonal symbol sequences are sent from each transmitter. In other words, the sent symbols over the N channel uses are chosen such that $\mathbf{X}\mathbf{X}^H = \mathbf{I}_{M_t}$ holds. The SNR is assumed to be approximately the same between each antenna pair. In this case, the channel can be estimated by analogous reasoning as in (7), which yields the maximum likelihood estimator

$$\hat{\mathbf{H}} = \frac{\mathbf{Y}\mathbf{X}^H}{\sqrt{\rho}}. \tag{9}$$

Furthermore, the Pluto SDR devices can also handle the transmission of Orthogonal Frequency Division Multiplexing (OFDM) signals, which uses multiple sub-carriers and can be used to obtain CSI simultaneously for the sub-carriers. In addition to the actual CSI, this allows the analysis of sub-carrier correlations. In the OFDM case, the channel estimations are performed just as in (7) and (9) , but for each sub-carrier independently of the others.

3.1.3 *Channel Evaluation*

The channel estimates can be evaluated by sending Binary Phase Shift Keying (BPSK) symbols when using the estimated channel in the decoding and calculating the bit error rate (BER). Sending BPSK symbols means that $\mathbf{x}$ in (6) and $\mathbf{X}$ in (8) only consist of the IQ symbols 1 and -1, i.e, one symbol corresponds to one bit. Hence, calculating the uncoded BER is equivalent to calculating the proportion of symbols that were not decoded correctly. Since the SNR is not known in advance, it can not be predetermined what a good BER is. Thus, the BER can only be used as a comparison between different channel estimates.

3.2 Data Collection

The data collection includes getting both training and test data for the classification algorithms. During one data collection session, the static environment and the chosen dynamic activities to be used in the classification will be performed a sufficient amount of times, until enough training and test data has been collected. Each time an activity is performed, it will be recorded for a fixed amount of time, during which M channel estimates are collected. Each channel estimate will be calculated based on a packet of N samples and it is assumed that the channel is time invariant during these N channel uses. The activity could e.g., be waving an aluminum foiled balloon or running around in the room, which has been exemplified in Figure 2 and the data collection process of one activity is shown in Figure 3. Finally, the M channel estimates are manually given a label corresponding to the activity that took place. This information is sent to the software subsystem (see Figure 1) for further processing. A flow chart describing the data collection process can be seen in Figure 4.

3.3 Environment

The experiment is going to be performed in the lab room. An approximate room sketch is given in Figure 5 (the placement of the Pluto SDR devices is just an example and might not be exactly as shown in the figure, and the number of transmitters and receivers might also be increased). The signals are transmitted between two or more Pluto SDR devices. The room environment is complex since there are many objects that can make signals reflect and refract. The lab room might be changed to a less complex room if discovered that the CSI is too complicated to analyse.

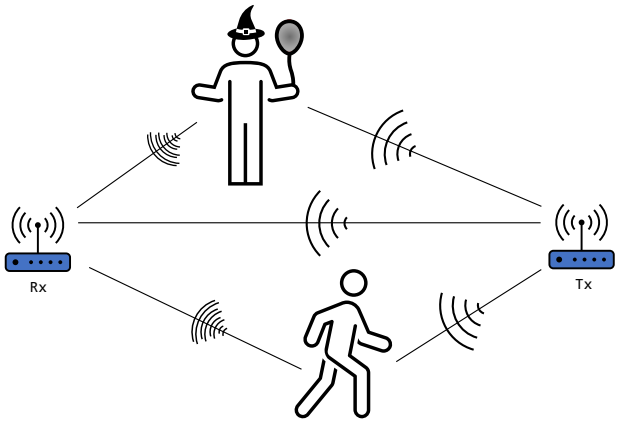


Figure 2: A data collection scenario for a dynamic event. One (or several) persons or objects perform some activity in the room. A signal is transmitted from the one Pluto SDR device (TX) to the other (RX).

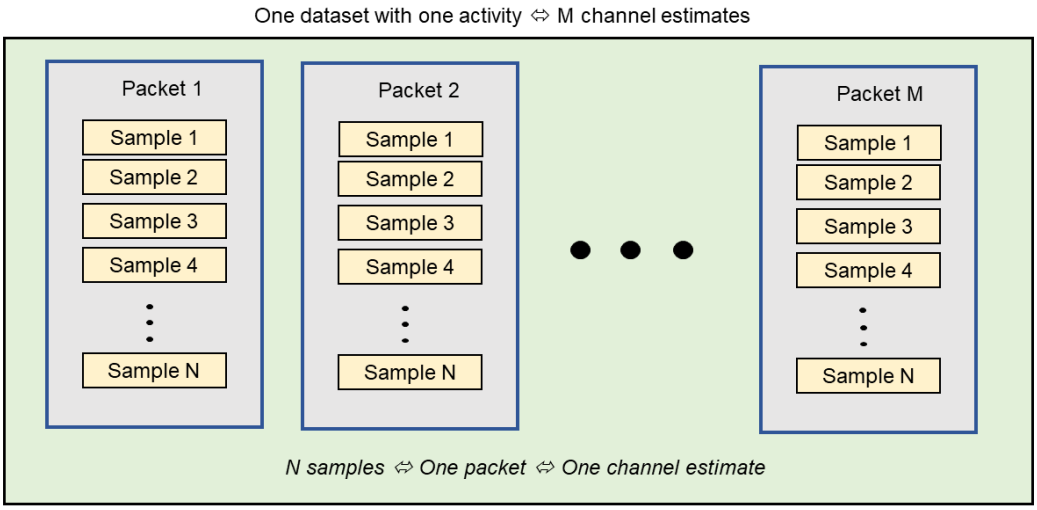


Figure 3: Description of one data collection for one activity. After collecting N samples (one packet), a channel estimate is performed. This is repeated until the activity stops after a predetermined time.

4 SOFTWARE

The software subsystem is defined as the functionality to extract relevant features, in addition to training, validation, and testing of the ML algorithms used for classification. These algorithms will include, but not limited to, k-means

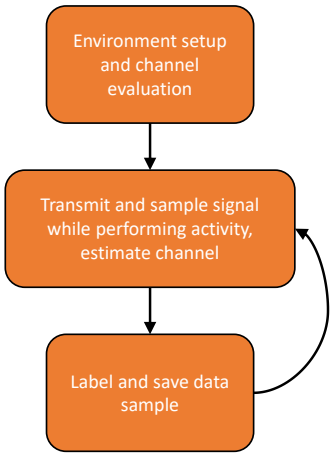


Figure 4: Data collection workflow.

clustering, HMM and SVM [11]. The software subsystem will be written in Python 3 [15]. A flowchart describing the software subsystem can be seen in Figure 6.

The CSI data consists of longer time series of channel estimates calculated and labelled by the hardware subsystem. The hardware subsystem has assured that the sampling interval is good. In the preprocessing step, the data will be split into smaller samples of desired length and if necessary, the data will undergo some additional processing. The next step is to extract features to use in the ML models and then train, validate, and test the models. The output from the software subsystem is the created models and the results.

4.1 Data Preprocessing and Feature Extraction

The CSI data needs to be split into smaller samples in order to get many samples of data that can be used to train the models. It is desirable that the model can classify new CSI data of different lengths and not just the same lengths that it was trained with. This could be achieved by designing the features in a way that makes them independent of the sample length.

Once the data is split into smaller pieces and labelled with the same label as the original sample, the training can begin. If the results from training the models with the raw data are not satisfactory, or if it is expected that better results can be achieved if the data is processed further, the preprocessing will be modified. Additional prepossessing could include filtering out high-frequency noise.

To increase the performance of the shallow ML models, features will be designed from the preprocessed CSI data. The complex valued CSI time series of length N can be transformed and possibly averaged over k evenly spaced intervals. Transformation can include extraction of real and imaginary parts, amplitude and phase, transformation to the frequency domain, computation of variance etc. The method of [16] suggests that the signal energy in the frequency domain can be mapped to different events. If m subcarriers are used, this will result in mk feature vectors

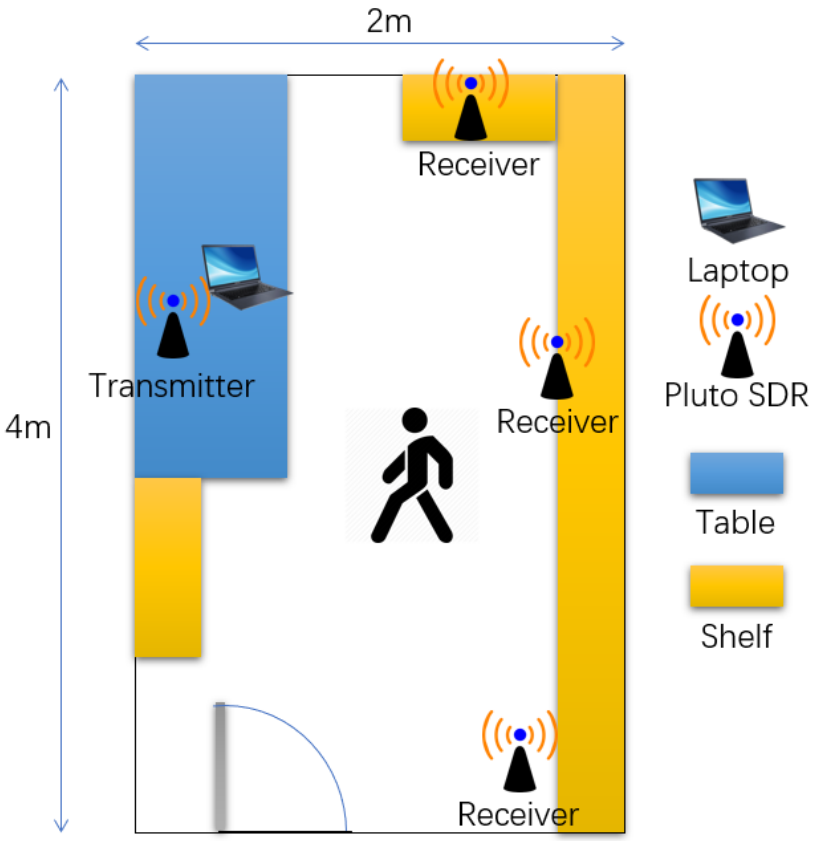


Figure 5: Lab room environment sketch. The two Pluto SDR devices operate as the transmitter and the receiver. The placement of the Pluto SDR devices is just an example.

for each labelled sample. In the binary hypothesis scenario, i.e., a detection task, a simpler set of features may be sufficient to achieve satisfactory performance. For the recognition task, more complex features might be required.

4.2 Machine Learning Models

In this section, we present the conventional ML models used for detection and classification. They will be tested with different settings of parameters and trained with a range of sets with features to evaluate under which circumstances each model preforms best and which of the models that is the overall most suitable to use for our application. The DL models that will be implemented in case of extra time are briefly presented.

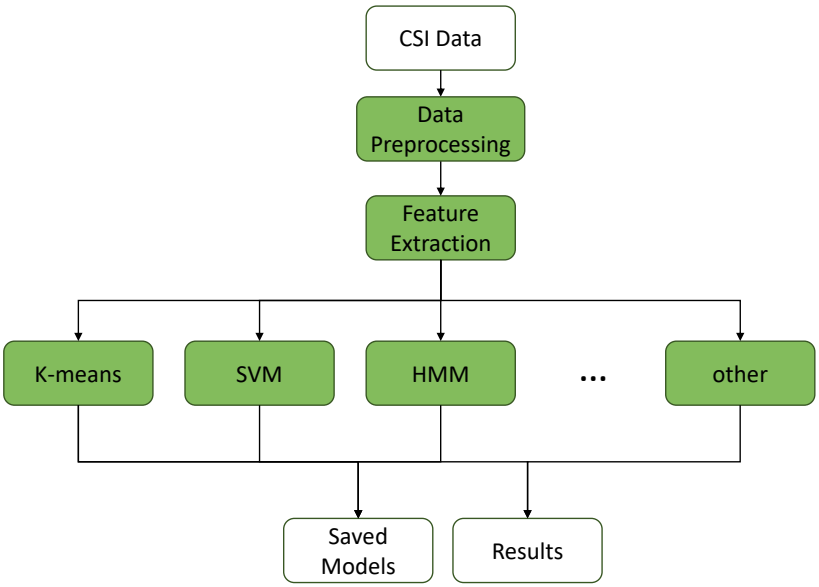


Figure 6: The flowchart describes the software subsystem.

4.2.1 *K-means Clustering*

The k-means clustering algorithm is originally an unsupervised learning method. The idea behind k-means clustering is to find a classification for the data points into k classes, when k is known [17, Chapter 14]. All data points should be assigned to the cluster they are closest to, and the class mean for each class should be calculated. This is done by randomly initialising the class means and then repeat two steps until the cluster center stabilise: First, assign each data point to the closest cluster center with respect to distance, usually the Euclidean distance. Then move each cluster center to the mean of all data points in the cluster. Figure 7 shows a conceptual clustering for two classes cluster centers are marked with hollow symbols.

Since information on the true classes is available, the project groups suggest not only fitting a k-means model to the data, but also compute the class means directly and use those for prediction. Creating a k-means model could though be of interest in the process of creating relevant features. If a set of features yields distinct clusters in the k-means clustering algorithm, it is probably because the features in a clear way are different between the classes in the data set too.

4.2.2 *Support Vector Machine*

The SVM model needs to be provided with extracted features for each sample to be trained with. A kernel function then needs to be selected to specify how the division of the feature space into decision regions should be done. The implemented SVM can be used in the case of binary as well as multiple hypotheses. The latter case is slightly more complicated since multiple models have to be trained and it is weaker to data imbalance [18, Chapter 7]. For a setup

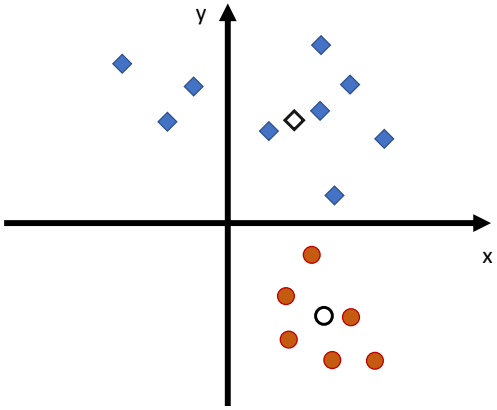


Figure 7: Example of a k-mean clustering with $k = 2$.

similar to ours, where humans were to be detected through a wall, the SVM-model was used successfully with 30 subcarriers [19]. The covariance between the subcarriers was used to do the classification. As we are not to detect objects behind a wall, the SVM model can probably still be useful even with just one subcarrier. Figure 8 shows an example of a SVM model, the decision boundary lies with maximum distance to the support vectors on the dashed lines. The result from a classification by the model is the label of the class that it is most likely to belong to.

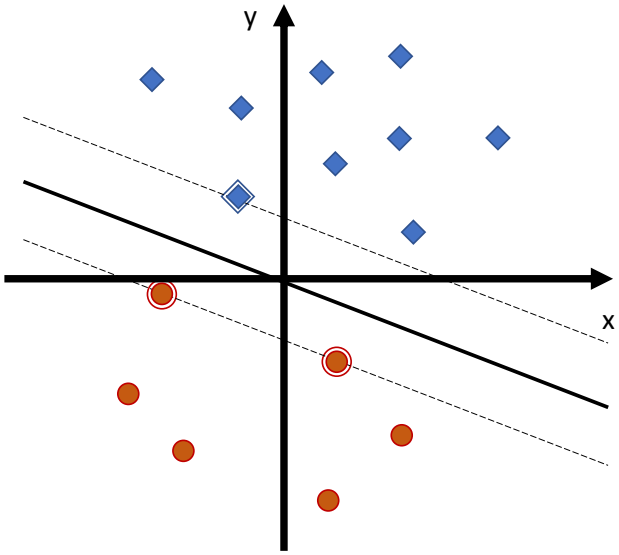


Figure 8: Example of a SVM model with two classes.

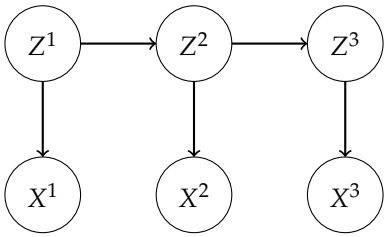


Figure 9: A simple HMM for 3 time steps. The X^i:s can be observed, and the Z^i:s are hidden.

4.2.3 *Hidden Markov Model*

The presence of a moving human or object in the environment affects the CFR, and the data points in such a time series are therefore correlated. A HMM is a state space model that captures the correlation patterns in time series data by introducing a set of observable variables and a set of *discrete* latent (hidden) variables [18, Chapter 13]. The observable variables are conditionally dependent on the hidden variables $p(X^t|Z^t)$, and the transitions between states of the hidden variables follow a stationary Markov process, i.e., the transition probabilities only depend on the current state and are independent of the time $p(Z^{t+1}|Z^t)$, $\forall t$. The conditional probabilities of the observed variables, also called emission probabilities, are also assumed to be constant in time. A simple HMM is shown in Figure 9. The arrows imply conditional probability densities. We may generalise to $X^t = (X_1^t, .., X_{d_X}^t) \in \mathbb{R}_X^d$, $Z^t = (Z_1^t, .., Z_{d_Z}^t) \in \mathbb{R}_Z^d$ (as any complex numbers may be represented by two real numbers), but there will be no edges between any variables in the set X^t:s, nor between variables in the set Z^t, for any t.

It has been shown that different events yield different frequency patterns in the collected data [16]. If one imagines the hidden states as the environment state (affected by movement speed of objects in the environment), and the observable states as combinations of signal energy over a set frequency intervals, it is possible to estimate a HMM from the collected (labelled) data. It may also be the case that there is a link between some other, less complex, quantity (e.g. the CSI amplitude) and the movement type. By training one model per activity, one can compute the likelihood of new observations given the model. The event corresponding to the model that has the highest likelihood of generating the sequence of observed features will be taken as the prediction of the system.

4.2.4 *Decision Tree*

To know what features to train the model with, it is identified which of the features that have most entropy [17, Chapter 12]. If a feature is very similar among all the samples, it will probably not help us as much as a feature with a larger spread of values. Once the feature with the most entropy have been used, the one with second most entropy is considered. A feature can also be used more than once in the decision tree. Figure 10 shows a simple example of a decision tree.

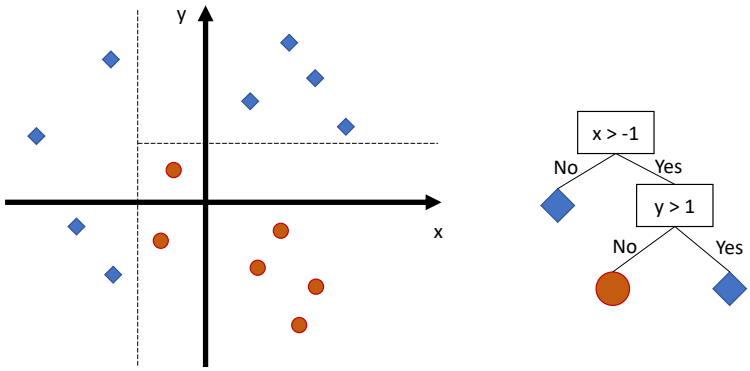


Figure 10: Simple example of a decision tree with two classes.

4.2.5 *Deep Learning Models*

If time allows, the system will also feature a deep neural network (DNN) for classification. The DL models has the potential to leverage complex patterns in the data, and may thus use the preprocessed CSI data without feature extraction. Models that are considered include MLPs and RNNs.

4.3 Software Packages and Modules

To implement the software in the projects a number of packages for Python will be used. For more basic data processing NumPy, SciPy and Pandas will be considered. For plotting, matplotlib will be used, and for the implementation of ML models, scikit-learn (sklearn) will be used for the shallow models, while a suitable package for a DNN is to be found if it is to be implemented later in the project. Possible alternatives are TensorFlow (with Keras), or PyTorch.

4.4 Training and Testing the System

The processed data containing samples with features and labels will be divided into training, validation and test data sets which are used for implementation and testing. The training data is used to train the models. The validation data is used to tune parameters in the models by training them with training data and then try different parameters in the model and evaluate the result with the validation data. Test data will be used to evaluate the result for the final models. For comparison, the same data will be used on all models. Of all the available data, 50% is used for training, 20% is used for validation and 30% is used for test. In Figure 11, a flow chart of the process is presented and also a graphic representation on how the data is divided.

The Python package scikit-learn contains functions that also can be used to tune parameters for models. These functions can, for example, evaluate different specified options for parameters using scores of users choice by cross validation. For this, the training and validation data is used. The function returns the final model.

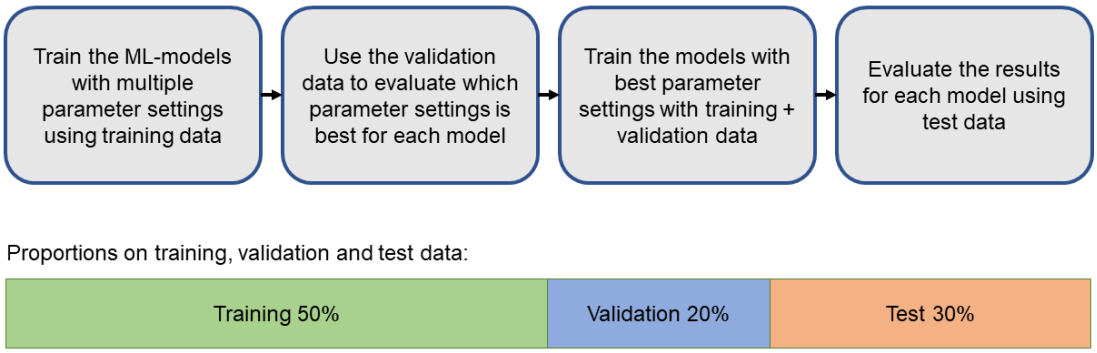


Figure 11: Flow of how to train the ML models and evaluate them.

4.5 Evaluation

Different kinds of evaluation methods and measurements is used to evaluate the models. True positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are defined for binary classification problems. The definition is shown in Table 1. Positive and negative are the classifications made, and true and false stands for if the classification was done correctly.

Table 1: Definitions of TP, TN, FP and FN samples respectively.

Predicted \ Actual	Positive	Negative
Positive	TP	FP
Negative	FN	TN

This kind of table, where the results from a classification is presented, is called confusion matrix and gives an overview on result from the classification. By using these definitions it is possible to define measures that gives a overview on how well a classification performed on a full data set. Test data is always used to evaluate a classification model. Accuracy is the rate of correctly classified samples and can be expressed as

$$\text{Accuracy} = \frac{\text{Correctly classified predictions}}{\text{Total number of predictions}} = \frac{\text{TP} + \text{FN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}. \tag{10}$$

Precision can be interpreted as the rate of how many classifications that was correct for one class in relationship to the number of samples for this class. Precision can be expressed as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{11}$$

Recall is the rate of how many classifications that was correct for one class in relationship to how many samples that got classified as this class. Recall can be expressed as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{12}$$

The harmonic mean between precision and recall is called F1-score, which can be expressed as

$$\text{F1-score} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{13}$$

Another evaluation measurement used is the number of seconds of data the model needs to make a classification. By this we mean for how long (in seconds) data is collected to make one sample for classification.

5 USER INTERFACE

The user interface is the interface from which the system is controlled. The user may use the interface to calibrate the hardware parameters, collect and store data, train and run the classification algorithms, and compare the classification results [11]. The user will also be able to see the channel logs, the BER, and a system log. The user interface will be written in Python 3 using the Tkinter package. A mock-up of the user interface is shown in Figure 12.

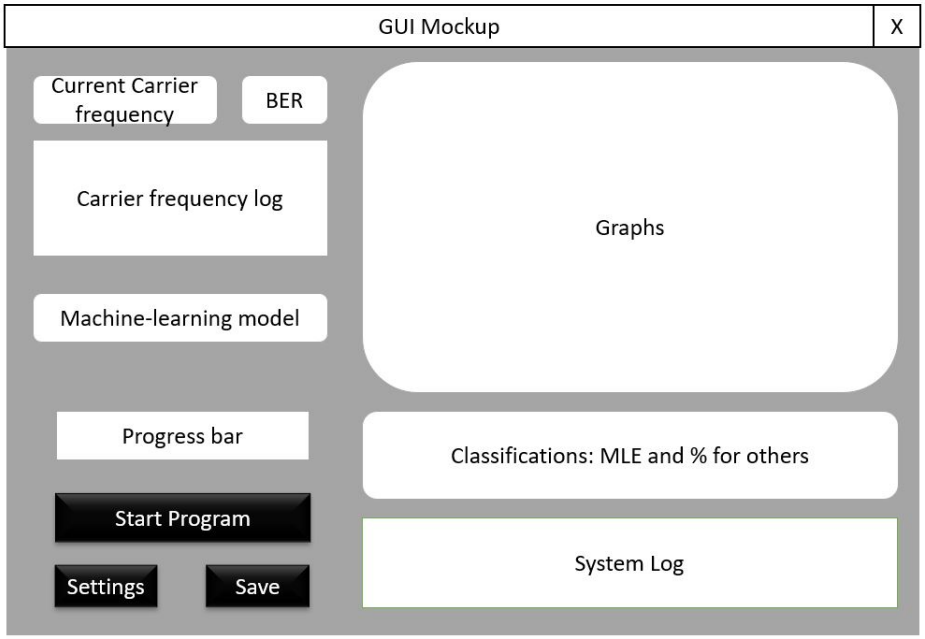


Figure 12: A mock-up of the user interface, displaying buttons (black) and different windows.

REFERENCES

[1] I. Nirmal, A. Khamis, M. Hassan, W. Hu, and X. Zhu, "Deep Learning for Radio-Based Human Sensing: Recent Advances and Future Directions," *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 995–1019, 2021.

[2] Ding, Jianyang and Wang, Yong, "WiFi CSI-Based Human Activity Recognition Using Deep Recurrent Neural Network," *IEEE Access*, vol. 7, pp. 174 257–174 269, 2019.

[3] X. Chen, C. Ma, M. Allegue, and X. Liu, "Taming the inconsistency of Wi-Fi fingerprints for device-free passive indoor localization," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[4] Y. Tian, G.-H. Lee, H. He, C.-Y. Hsu, and D. Katabi, "RF-Based Fall Monitoring Using Convolutional Neural Networks," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 3, Sep. 2018.

[5] Y. Ma, G. Zhou, and S. Wang, "WiFi Sensing with Channel State Information: A Survey," *ACM Comput. Surv.*, vol. 52, no. 3, Jun. 2019.

[6] Z. Wang, K. Jiang, Y. Hou, W. Dou, C. Zhang, Z. Huang, and Y. Guo, "A Survey on Human Behavior Recognition Using Channel State Information," *IEEE Access*, vol. 7, pp. 155 986–156 024, 2019.

[7] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and Modeling of WiFi Signal Based Human Activity Recognition," ser. MobiCom '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 65–76.

[8] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You Are Facing the Mona Lisa: Spot Localization Using PHY Layer Information," ser. MobiSys '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 183–196.

[9] L. Chen, J. Xiong, X. Chen, S. I. Lee, K. Chen, D. Han, D. Fang, Z. Tang, and Z. Wang, "WideSee: Towards Wide-Area Contactless Wireless Sensing," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, ser. SenSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 258–270.

[10] W. Li, B. Tan, Y. Xu, and R. J. Piechocki, "Log-Likelihood Clustering-Enabled Passive RF Sensing for Residential Activity Recognition," *IEEE Sensors Journal*, vol. 18, no. 13, pp. 5413–5421, 2018.

[11] M. Andersson, E. Beskow, E. Grundin, R. Mannberg, J. Nilsson, G. Suihko, and J. Qu, "Detection of static and dynamic indoor environments: Requirement specification, 2021."

[12] T. Ulversoy, "Software defined radio: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 12, no. 4, pp. 531–550, 2010.

[13] *ADALM-PLUTO for End Users.* Wilmington, MA, USA: Analog Devices, 2021, Accessed: Sep. 17, 2021. [Online]. Available: https://wiki.analog.com/university/tools/pluto/users

[14] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[15] Python Software Foundation. About Python. Accessed: Sep. 21, 2021. [Online]. Available: https://www.python.

org/about/

[16] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Device-Free Human Activity Recognition Using Commercial WiFi Devices," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1118–1131, 2017.

[17] S. Marsland, *Machine Learning - An Algorithmic Perspective.*, ser. Chapman and Hall / CRC machine learning and pattern recognition series. CRC Press, 2009.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] H. Zhu, F. Xiao, L. Sun, R. Wang, and P. Yang, "R-TTWD: Robust Device-Free Through-The-Wall Detection of Moving Human With WiFi," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1090–1103, 2017.