# ENHAO SUN

ID Number: 4276716

Supervisor: Holger Schnadelbach

Module Code: G53IDS

# 8102/18

**[IoT sensing and building]**

Submitted [April 2018], in partial fulfilment of
the conditions for the award of the degree **[G4G7]**.

**[Enhao Sun]**

**[4276716]**

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

**Signature** _____ 孙恩浩 _____

**Date _23_ / _04_ / _2018_**

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.*

Public access to this dissertation is restricted until: 24/04/2018

*Only include this sentence if you do have all necessary rights and consents. For example, if you have including photographs or images from the web or from other papers or documents then you need to obtain explicit consent from the original copyright owner. If in doubt, delete this sentence. See Copyright Information for more details.

**Only include this sentence if there is some reason why your dissertation should not be accessible for some period of time, for example if it contains information which is commercially sensitive or might compromise an Intellectual Property claim. If included, fill in the date from which access should be allowed.

# Abstract

Human activity recognition (HAR) is one of the active research areas in human computer interaction, healthcare and smart buildings. Different approaches including applying machine learning techniques, deploying different sensors had been researched. This project introduces potential concept of recognizing human activities by an interactive system. The system consists of three components: a local server, sensors and a client with user interfaces. Machine learning program runs on the local server informing the client about the recognized activities. Meanwhile, the client is able to respond to the server with feedback. An innovative design is that every feedback from client will improve the recognizing performance of the system. Nevertheless, challenges such as disturbance from chaos in real-world, difficulties in user testing needed to be concerned for the future design.

# Contents

# 1. Introduction

The aim in this project is to design a system in a room which can recognize people's activities from series of observations on the human's actions and environmental conditions, enabling a user to interact with the system. The system presents those observations and activities to the user and receives feedback from the user. [2] The key objectives of the research are:

1. To deploy low-level modules connected to Arduino that can gather data about activity information using light sensors, motion sensors, force sensitive resistor, accelerometers and so on.

2. To select useful sensors (deployed on the different locations in a room) which can effectively recognize the activities of the inhabitant. [3]

3. To deploy a local server in Raspberry Pi combined with the wireless transceiver for collecting data, storing data and analysing data as well as sending messages to the user.

4. To recognize people's different activities, using sensors such as sonar sensors to count people getting in or out room, force sensitive resistor in cushion detecting whether there is an object on seat and motion sensor for detecting movement in the room. Further, synergy between different sensors can infer other activities which cannot be detected by a single sensor.

5. In training phase, after the sensor data is collected, preprocessing the raw data to remove noise and represent data. Then labelling the data. After this, choosing model and parameters to train a model for classification. [4]

6. In classification phase, following preprocessing, feature extraction and classification to classify the activity. [4]

## 2. Motivation

The concept of home automation has been discussed for several decades which aims to simplify people's lives by merging latest technology into their lifestyle. It now allows the user to control their home remotely. People can take good care of their home with the variety of systems such as security systems, environmental systems, which can monitor many aspects of daily life. [5]

However, traditional home automation systems need people to make decisions to control the system. On the one hand, it is convenient to control home devices remotely which is an essential part of home automation, but on the other hand, systems require users' awareness of when to control systems and how to control systems [5]. People may not always be careful to find that the environmental condition in their room is getting worse. They can be likely to forget to turn off the heater and light when they are leaving. Perhaps, they often forget to close the windows. If home systems can't recognize those conditions, people would not know about that. Such home systems cannot be able to solve the issues of energy waste and security problems. Therefore, home automation system which can perceive environmental conditions would be future trends to solve problems above. System, which can understand user's activity patterns in real-time, could be able to perform reaction and make decisions according to the changing conditions and events [6]. The capability of monitoring the environmental conditions and recognizing human's activities are the prerequisites for home automation systems to achieve this.

To design such a system, the main problem is to solve the human activity recognition which aims to recognize people's action from series of observations on the human's actions and environmental conditions. To solve this, there are three questions need to be concerned: How can human's activities be detected? What activities can be detected in homes? And how can this information be utilized? [2]

Sensor-based activity recognition uses simple inexpensive sensors installed in the home to detect human's action under different environmental conditions. Sensors should have low maintenance and be easy to maintain [1]. Systems can obtain data from sensors to know the current condition of the environment and infer human's behavior by analyzing changes in conditions. Therefore, it is important to select appropriate sensors and the method for selecting sensors should be discussed. Studies have shown that installing useful sensors in right places would help systems to recognize human's activities effectively [3]. Additionally, in order to fetch and use data from sensors, there are three steps to follow: First, collecting data from sensors; Second, processing and storing collected data; Third, developing programs to train data and use output model to infer activities. To achieve this, a local server will be implemented for collecting and storing data. A few of machine learning program will be run in a local server to process data and classify the activities.

# 3. Related Work

According to existing research [2], the cost of sensors and sensor acceptance should be considered as pivotal issues, especially in home and office. Many people are unwilling to live with cameras. Instead of using cameras to process image of the environment, this project would use radar motion sensors to detect human's movement. When selecting sensors, there are few choices including PIR motion sensor, ultrasonic sensor and radar motion sensor. The accuracy sensitivity of ultrasonic is strong. But it's detection range is relatively narrow in comparison with PIR and radar motion sensor. As for PIR motion sensor, it can be unreliable and impacted by harsh weather condition or the change of the difference in temperature. And PIR motion sensor lacks object identification which means false triggers. In contrast to above, radar motion sensor has increased detection range and precision object identification. Moreover, radar motion sensor works independently of hot weather. And this sensor is small and cheap and easy to replace. People in a room may not be aware of sensors.

Compared with existing WiFi and RFID radio-based activity recognition research [7], this project uses ZigBee wireless transceiver module to transfer data from sensors. ZigBee is a low-cost, low-power, wireless mesh network standard which is also widely used in the wireless sensor network. Besides, ZigBee has two different mode including one to one mode and broadcasting mode. This means strong application flexibility in different scenarios. And it cost less energy than WiFi but has much wider transmission distance than RFID. Because the wireless mesh network created by ZigBee is local, it will be relatively more secure than WiFi.

Furthermore, this project deploys local server rather than uploading data to the cloud server. It will, to a great extent, protect users' privacy.

The last but not least, this project aims to design an interactive system between home systems and humans. A crucial part to achieve the interaction is machine learning program. Machine learning program is used for training and classifying data from the user. This would provide system capabilities to keep learning about users' activities. The feedback from users would also help system reanalysis the data and update models. Researchers in researching interactive architecture [8] defined such systems as multiple-loop systems in which one enters into a conversation: a continual and constructive information exchange. If the design of such home systems is reliable, the interactive loop between the home system and humans will create a robust home system. The system will have more precise perception in detecting environment and be much likely to make right decisions. However, the definition of the reliability of the home system and how to create a perfect interactive loop are still under exploration.

# 4. Description of Work

As mentioned before, the system consists of deployed sensors, local server, and machine learning programs. The functionality is to recognize human's activities in a room. Further to this, the system should aim to learn and recognize new activities through the feedback from the user. In below sections, the functional specification and non-functional requirement have been made explicit.

## 4.1 Functional specification

Functions of each part of the system have been specified here:

### 4.1.1 Hardware specification

1. **Sensors:** environmental conditions will be detected by sensors.
- 1.1. Sensors should be able to detect environmental conditions continually.

- 1.2. Different types of sensors can detect different environmental conditions.

- 1.3. Combination of different sensors will have the synergistic effect such as improving the precision of prediction and finding out other activities which cannot be recognized by a single sensor.

2. **Controllers:** controllers will be a bridge between sensors and server for data transmission.

- 2.1. Controllers should be able to power sensors.

- 2.2. Controllers should be able to control multiple sensors.

- 2.3. Controllers should be able to send data to the server wirelessly.

- 2.4. [OPTIONAL] Controllers should be able to go to sleep and wake up while the condition of the environment is changing.

3. **Local Server:** Server will be able to receive data, store data, and process data. The server should be able to communicate with a mobile phone by Bluetooth.

- 3.1. The local server should be able to receive data from controllers wirelessly.

- 3.2. The local server should have enough memory to store data.

- 3.3. The local server should have the capability to process and analyse data.

- 3.4. The local server should be able to pair with a mobile phone by Bluetooth and transmit data with the mobile phone.

## 4.1.2 Software specification

1. **Programs on Controllers:** data should be collected correctly and sent to server.

- 1.1 Programs should be able to deal with different types of data from different sensors.

- 1.2 Programs should pack the data before sending it to the server.

- 1.3 Programs should listen to the instructions from the server before sending data to the server.

- 1.4 [OPTIONAL] Program should have watchdog timer setup to interrupt controller from sleep mode.

2. **Programs on Server:** Data receiving programs will be able to receive data and store data appropriately. Machine learning programs will be able to train and classify data. Other programs will label data and inform the user by sending messages.

 2.1 **Data receiving program:** program will be able to receive data and store data into the database.

- 2.1.1 Program should broadcast different data-request instructions to all of the controllers.

- 2.1.2 Program should listen to port continually to receive data.

- 2.1.2 Program should be able to receive different packs of data and deal with them.

- 2.1.3 Program should create a database appropriately if the database does not exist.

- 2.1.4 Program should be able to store data into the database.

 2.2 **Labelling program:** programs will be able to update the database with selected time by entering date time and specific activity.

- 2.2.1 Program will ask users to enter two time-slots to select data entries in between.

- 2.2.2 Program will ask users to enter activity that they want to update.

 2.3 **Machine learning program:** programs will be able to train data and classify data precisely.

- 2.3.1 Programs should be able to process data correctly before it can be used for training and classification

- 2.3.2 Program should train data and output a model

- 2.3.3 Program should be able to classify data into corresponding class

- 2.3.4 Program should be able to update model from user's feedback

2.4 **Notification program**: program will be able to send messages to the user and receive feedback from them.

- 2.4.1 Program should be able to send messages to the user at a set time.

- 2.4.2 Program should be able to receive the feedback from users and relabel some labeled data.

3. **Application on the mobile phone:** Application will send the data of accelerometer to the server at the set time and update environmental data from the server. It would be able to give feedback to the server.

- 3.1 Application should be able to scan Bluetooth devices around and pair with them

- 3.2 Application should alert the user to turn on the Bluetooth if the Bluetooth service is not running.

- 3.3 Application should collect accelerometer data of the mobile phone.

- 3.4 Application should send the data to the server.

- 3.5 Application should be able to receive the environmental information from the server and then let user give feedback back to the server.

## 4.2 Non-Functional Requirements
**1. Reliability**

It is likely that the training data and machine learning algorithm will determine the result of prediction for human's action. According to NON-FUNCTIONAL CHALLENGES in [25], data integrity and fault detection are great challenges for designing reliable algorithms. Therefore, it is important to ensure the quality of data is good and avoid any exception as far as possible.

**2. Security**

While the sensors and servers are used for collecting and storing user's data respectively, such data including user's living environment condition and activity of the user is private. Also, the server will need to receive the data from user's mobile phone and send a message to the user. Exposure of data and easy access to the server may lead to user's living environment and mobile phone monitored. Therefore, the project is responsible to ensure that the access to server and user's data are well protected.

# 5. Methodology

## 5.1 Primary Research

After the machine learning programs on server and mobile phone application have been developed, some user studies will be conducted in order to test the usability and reliability of the system. As it is supervised machine learning, it is important to set appropriate labels for each training. It would require many user tests to adjust the implementation of training models as well as getting feedback from the user. More research will be conducted in order to obtain a full understanding of existing works. Therefore, a portion of the project will be dedicated to research and user testing. This portion of the project will take times to experiment different algorithms, models with different parameters of machine learning program to come up with a solution that solves this problem as far as possible. The research conducted will include:

1. **Techniques development:** This will be used to learn how to use platforms and tools required in this system and choose the best solution to solve the problem.

2. **Algorithms design:** Different machine learning algorithm will be implemented in order to find a most fitting one according to the system specification and special environment.

## 5.2 Machine learning algorithms

As mentioned in the previous sections, a portion of this project would be the development of machine learning program which is the main approach for the system to be able to recognize human's activities by learning and adjusting. It is therefore important to choose machine learning algorithms for activity recognition.

### 5.2.1 Algorithms for sensor selection

Sensor selection algorithm is used to determine the selection and placement of sensors in a smart environment to minimize the number of sensors that are needed for both maintaining and improving activity recognition performance. According to the experiments done by other researchers [3,9], they conclude that the reduction of sensors for activity recognition would not only reduce the cost of sensors maintenance but also reduce the computational cost, at the same time, it promoted the accuracy of learning inhabitant's activities. The filter-based approaches search for best representative features without involving any learning algorithm which will reduce the cost of computation. In the work of Sook-Ling and Lee Kien [3], experiment results show that the information gain method that was used to measure the importance of the sensors to recognize the activities of the inhabitant can effectively identify important sensors on different datasets. Moreover, another researcher [27] proposed that a proper sensor selection and refinement algorithm could improve the reliability of the system. However, in this project, there are only a few of single sensors deployed in the room. Sensor selection algorithm above should be designed for complicated environment with much more sensors. It is relatively not practical to designing and applying such algorithm for few single sensors. Thus, this project simply tried several different places for different sensors and chose positions, where data collected was obvious and valid. For instance, in placing the accelerometer sensor onto the door, the experimental result showed that the changing of acceleration data would be much more distinct when the accelerometer sensor was placed near to the doorknob rather than the centre of the door. This case shows that it is still important and worthwhile to

consider the sensor selection. When experimental space increases, the complexity of environment will also increase. At this circumstance, sensor selection algorithm would take effect.

## 5.2.2 different models for different types of data

Algorithms used for common pattern recognition include hidden Markov models (HMM), naïve Bayes classifier, Gaussian mixture models and support vector machines. However, these are classification techniques used for classifying data which is already processed. To achieve activity recognition results with a high performance, additional steps which should be followed before classification include: preprocessing of sensor data, feature extraction and optionally dimension reduction. Appropriate algorithms used for these steps are equally important compared to the classification algorithms. It will also influence the determination of classification results [4]. Thus, further research about choosing such algorithms will be conducted.

# 6. Design

## 6.1 System Design

### 6.1.1 System overview

The activity recognition system consists of three main components - Data acquisition programs, a Client application and Local server. Each of them runs on different machines and environments since individual components are used to carry out specific subsets of system features. The Data Acquisition programs are executed on sensor-embedded actuators and Client application, as actuators and client application are both data acquisition units. The Client Application is implemented on client machine because it is also a user interface. The server runs on a hosted local machine.

The system diagram below (Figure 1) shows the overview of the approaches to activity recognition. The workflow consists of four main steps. Initially, human's action and environment changes were detected by sensors. In other words, sensors keep collecting environmental data such as acceleration data, illuminance data. Controllers read and process the data collected by sensors. After reading and processing, controllers begin to send data to a server. As mentioned before, ZigBee based wireless modules are connected to both controllers and the server, which improve data transmission to be wireless. Before training and classification, the data is preprocessed and stored. To be exact, only portions of data are going to be used for training, and the rest of them are used for testing. This process is flexible and depends on the size of data. Finally, if results are well classified, the server will send current environmental information, as characters, to mobile phones of users. The users can view the present environmental conditions and have options to decide whether send feedback to the server or not.



**HUMAN ACTIVITY RECOGNITION SYSTEM DIAGRAM**
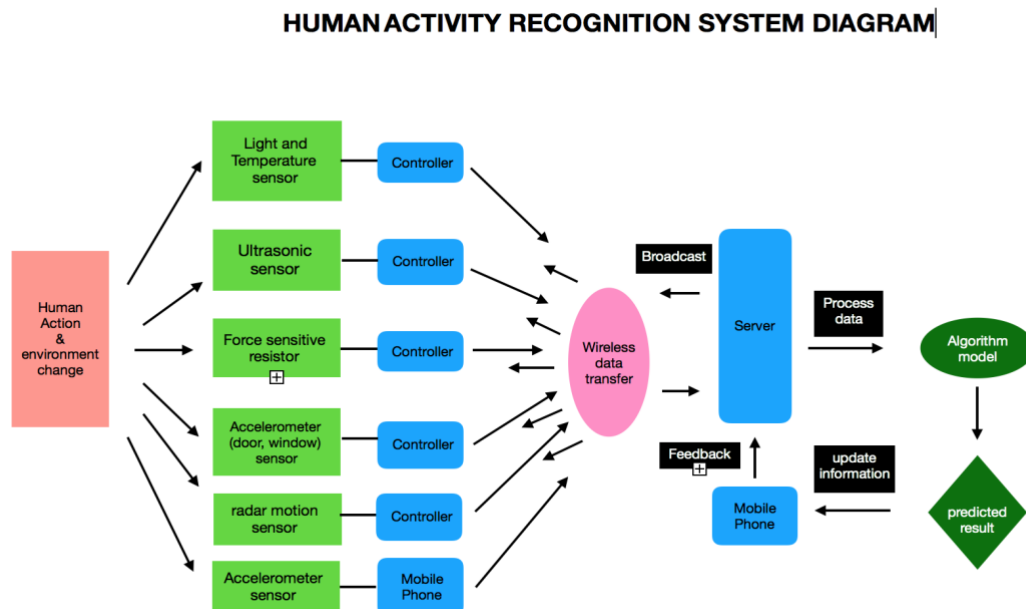
**Figure 1-1:** This picture shows the overview approaches of system to solve activity recognition and interactive between the system and users

## 6.1.2 Data acquisition unit

**Layout diagram and sensor units**

The Layout diagram (Figure 1-2) below shows the deployment of sensors in the room. In this diagram, different shapes with different colours represent different sensors.

Ultrasonic sensors placed on the bottom right corner can measure the distance from modules' positions to the door surface. It emits an ultrasound at 40000 Hz which travels through the air and if there is an object on its path, it will bounce back to the module. If someone passes by the door, the ultrasound may come back earlier. Hence, the passing action is caught.

Green square at the centre of the room is RCWL-0516 Microwave Radar Motion Detector. This sensor uses a microwave Doppler radar to detect moving objects. It has a sensitivity range of 7 meters with an angle of 120 degrees.

Blue circle, a force sensitive sensor (RFP-611), is used for detecting the pressure on that seat. RFP's basically changes its resistive value depending on how much it is pressed. It can measure up to 50-kg pressure.

Green circles on windows and doors are MPU-6050 accelerometers. With such sensor attached, push and pull of doors and windows would be detected.

Pink luminosity sensor(TSL-2561) on the left side of the room is an advanced digital light sensor which can accurately measure the illumination of this room. Yellow temperature sensors(DHT22) are placed by the window and on a table in order to measure the temperature difference between outside and inside when the window is opening. To make full use of actuators, sensors closed to each other share one actuator, thus reducing energy consumption.
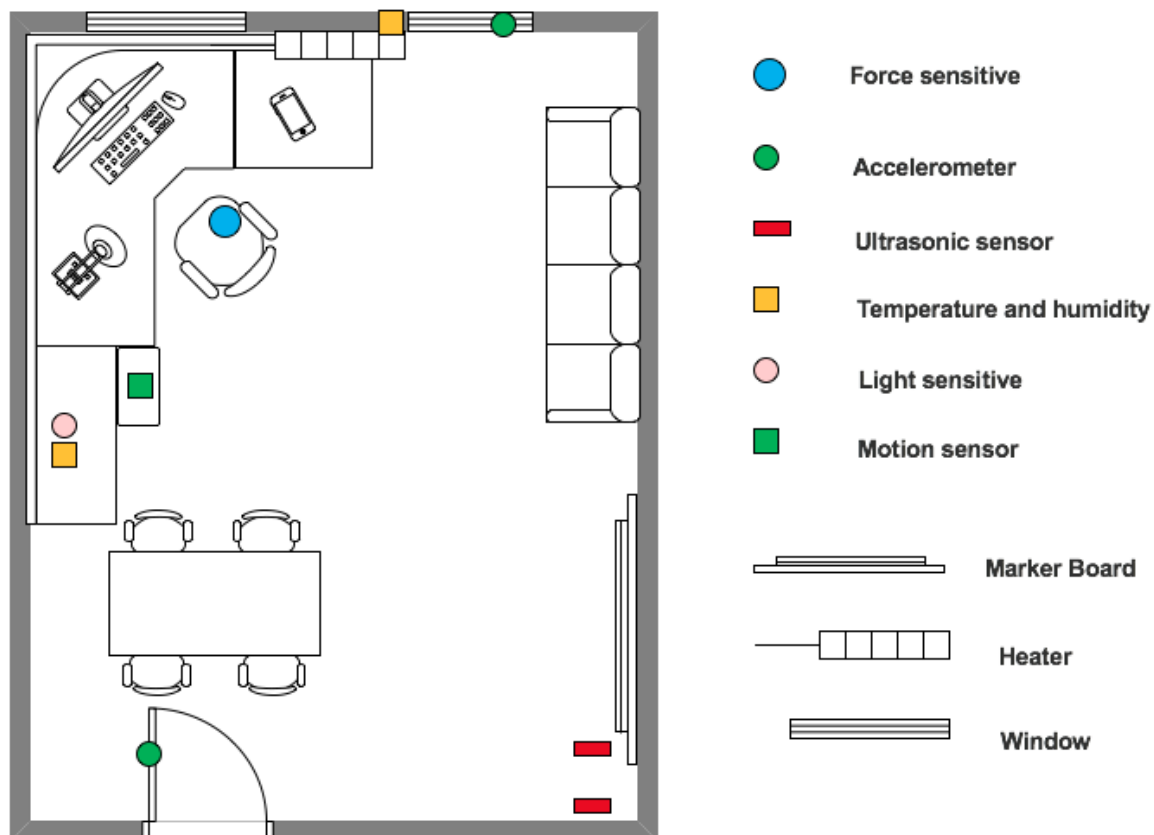


**Figure 1-2:** This picture shows the deployment of sensors in an office room

**Data acquisition program**

As mentioned before, the data acquisition programs run on sensor-connected actuators (Arduino Nano). Each program contains a setup function and a loop function. In the setup function, all of the programs set the same baud at the beginning. As the setup function is going to be executed only once, some variables are initialized and set. After creating a setup function, the loop function does precisely what its name suggests, and loops consecutively, allowing the program to change and respond. Therefore, most of the functions including data reading, data transformation, data packing and data transmission are executed in this loop.

To overcome the loss of data in transmission, data are packaged in specific formats. All the formats have two paired brackets between which specific characters are inserted. This allows the server to check the integrity of data and improve the efficiency of retrieving data. If the format of received data is not regular, data would be discarded, and the last message would be broadcasted again until data received have the standard format. Such design of packing data not only prevents incomplete data to be stored into the database but also reduces the error handling procedures in the server.

Programs in actuators which are either connected with multiple sensors or single element must be able to distinguish messages spread by the server. For this purpose, each sensor has its unique character predefined in the programs. When messages are received, the program compares it with its distinct character, thus deciding whether or not respond to the server with data read.

## 6.1.3 The Local Server
**The overview**
The objective of the local server is to store and compute data (validating, modifying and persisting). The server is responsible to make sure that data are valid before computing since the data imported into training programs are strictly formatted and specified. Therefore, it is crucially important to design programs which may access data in such a way that they all deal with data regularly and properly. Since it is very probable that data received or imported are invalid, rigorous procedures of data validation check should be used.

**Broadcast**
The entire system structure is designed in points. Every component in the system runs independently before connecting to the servers. Data jamming can easily occur when every actuator transmits the data package to the server at the same time. Under such circumstance, data received by the server would be chaotic and hard to be retrieved. Hence, the broadcast emitted by the server was designed to solve this problem by sequencing the transmission of each component. As mention before, the actuator would only respond to the server when the broadcasting message can pair with its specific character value. The server broadcasts single character to components in each iteration. The broadcast iterates every character in a list for every single sensor. The environmental information collected would be complete.

**Multiple data**
To ensure the integrity of the environmental information, the broadcasting mechanism was proposed. Every data entry was recorded by dictionary during the running time.

When each element of the dictionary was assigned, entries of the dictionary would be extracted and stored into the database. The dictionary would be cleared, and the server would start broadcasting again from the front of the character list until the end.

**Error handling**

The server is designed to be reliable by making assumptions that things will go wrong, and the error may occur in any part of the program. For this purpose, the server is required to be capable to handle wide exceptions. Such a design potentially gives the capability to distinguish errors to details, thus handling errors sequentially. (Data receiving) The exception on the high level is Serial timeout exception. The exception would be thrown if the serial port waited over the standard time. Before data is transported to be either parsed or reformatted, some data validation functions would be called to check the integration and validation of the data. Another exception is ValueError exception which is used to handle the errors of parsing.

**Data labelling and unavoidable bias**

Machine learning strategy, in this project, decides to use supervised learning principle. In supervised learning, training data are required to contain both features and labels to help machine learn to distinguish different data. As data collected are unlabelled, for this reason, data labelling strategy should be considered. Since the number the received data is enormous, the labelling program is designed to update data bundle efficiently and accurately. In the aspect of efficiency, the program provides the time slot selection for updating bunches of data which are distinguished to be labelling a same activity. In terms of labelling accuracy, the design of the labelling program is not responsible to guarantee this. For some experimental reasons, the activities of the series of data are decided and assigned by the experimenter, thus miss understanding could happen. In operating the devices, data between some time slots are very likely assigned with wrong activities, since reaction time does not allow the experimenters to distinguish what activities are happened. Those time slots including the last two seconds before the end of data-collecting operations and the first second after the start of the data-collecting operations. Worse more, data noisy which increases the difficulty of labelling data always exists. However, such bias could not be avoided in the real world. Therefore, miss distinguishing is allowed and the error has been considered in evaluation progress.

## 6.2 User Interface Design

**Overview**

The user interface was a Bluetooth based IOS mobile application.

As the phone was carried with the user most time, the mobile application could relatively monitor the movement of the user. In addition, signal range of Bluetooth could cover the activity area in the room. In IOS development, Swift was chosen instead of Objective-C because Swift is safer and easier to maintain. More than this, Swift requires less code but runs faster.

**Application usability and layout**

For the application required Bluetooth service, users need to turn on the Bluetooth service of the mobile phone before using the application. The application would alert

a warning message and remain out of service if the Bluetooth service has not been started.

The application had two pages. The first page called ViewController is designed simply for the user to search Bluetooth service. In considering that many Bluetooth devices could possibly appear in around, the table view was used to list services found. Each cell in the table list has the name of the service and signal strength of the Bluetooth service. The upper left corner shows a clear button which is used to clear content of the table list. The upper right has a scan button allowing the user to either scanning the Bluetooth service or stop scanning process. By selecting a target service, the current scene would turn to a Conversation page where main services provided by the server would be displayed. Figure 1-3 below shows the discovered peripheral devices listed in a table view:



Figure 1-3 Discovered peripheral devices

The main services were defined as the conversation between client and server. After the connection was built, a name of the service would be displayed on top of the page. Below the name, two switch buttons designed for data uploading and notification receiving were set in parallel. Since the experiment progress was not always continuous, the data-collecting operation would be interrupted in some points. Such a design of buttons allows switching either loading operation or description of notification service at any time, thus improving the efficiency of the experiment. As proposed in the system design, users would be able to interact with the system by replying messages from the server. The messages would consist room status including door, window, light, seat which were displayed in the black squared labels along with responding button.

In order to switch fluently between pages, a navigation bar was added. By pressing the button on the bar, the user could go back to the initial page to either rescan or connect to other servers. Further details about using the application can refer to Appendix B.

# 7. Implementation

## 7.1 The overview

In this activity recognition system, sensors need to be controlled by actuators and data collected would be stored in somewhere. After doing series of research, Arduino platform and Raspberry Pi were chosen as controllers and data storage respectively. The open-source Arduino Software (IDE) provides numerous libraries which are compatible with most of the sensors [20]. By combining with ZigBee wireless module, Arduino would be able to transit hex data to Raspberry Pi which has ZigBee module as well. Raspberry Pi as a credit card-sized computer can provide all the expected abilities that imply, at a low power consumption level. The huge number of community supports can be obtained quite easily for hardware and GNU/Linux software [21]. In terms of machine learning tools, Python is one of the best languages can be used to implement machine learning techniques for a few reasons. Python is one of the most popular languages among data scientists and web programmers. It is simple and powerful because of its simple syntax and huge community. There are tons of machine learning libraries already written for Python. The last point here is the most important. The machine learning algorithm is pretty complex and includes a lot of math. So writing then by ourselves would be the most difficult task. Luckily, there are plenty of smart and dedicated people out there that have done this hard work for us, so we can focus on the application at hand. Finally, the *scikit-learn* library was selected for our machine learning for which has a huge number of features for data mining and data analysis. The implementation of either software or hardware platforms would be discussed in detail later.

## 7.2 Hardware implementation

**ZigBee wireless module**

The ZigBee module used in this project is DL-20. Two patterns are provided by this module including point to point and broadcasting. To set up the module, three steps need to be followed including setting baud rate, setting channel and setting host/client. The detail of setting would be described in Appendix A.

**Arduino setup**

The Arduino board used is Arduino Nano which is a small, complete, and breadboard-friendly board based on the ATmega 328P. The Nano offers high connectivity and large spaces in a small form factor. The Nano accepts the 7 - 12 Volt input power not from the USB port, but from the Vin pin (pin30). For this reason, an appropriate power supply would be chosen. In choosing external power source, several batteries have experimented, and some problems were risen in this progress (which would be discussed later in problem encounter section). In the end, PP3, usually called 9-volt battery, was considered as the most suitable power supply in this project.

Sensors and ZigBee modules would be connected to Arduino Nano with cables. Details will be introduced in Appendix A.

**Raspberry Pi setup**

The Raspberry Pi used in this project is Raspberry Pi 3 Model B. Before going to implement the server, the LINUX operating system needs to be set on the Raspberry Pi. The operating system - RASPBIAN downloaded from the official website of

16

Raspberry Pi was installed into a small SD card which has 16 GB storage. The SD card would be inserted into SD Card slot where the operating system software and files were stored.
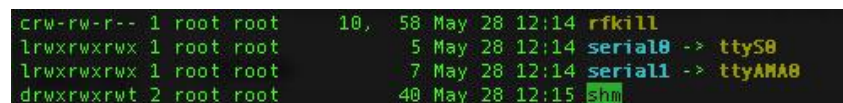
## 7.3 Software implementation

### *Server environment setup*

**Configuring the GPIO Serial Port**

The Raspberry Pi 3 has added Bluetooth service which can be turned on in system preferences. Before using the Bluetooth, the serial port mapping should be figured. On the Raspberry Pi 3, according to an online article [23], serial0 points to GPIO pins 14 and 15 and uses the "mini-uart" also known as /dev/ttyS0. Another port called serial1 points to the hardware UART which has high performance was nabbed for the Bluetooth. The serial1 is mapped to /dev/ttyAMA0 in default. To summarize the ports on the Raspberry Pi 3 and be clear, the default mapping of ports should look like below:

/dev/ttyAMA0 -> Bluetooth
/dev/ttyS0 -> GPIO serial port



Default Raspberry Pi 3 serial port aliases

**Language platform**

In order to setup server efficiently and implement data receiving program easily, we decided to use Python and NodeJS to build the server for receiving the data through serial(ZigBee) and Bluetooth respectively.

Since Python has the built-in library for operating serials, the progress of implementing the receiving program would be straightforward.

NodeJS is not a framework, but a runtime environment that allows executing JavaScript on the server side. A NodeJS module called bleno was used to implement BLE (Bluetooth Low Energy) peripherals [18]. The bleno module can easily customize the Bluetooth service for different requirements. In addition, NodeJS provides the non-blocking IO system that enables the developer to process numerous requests concurrently. Incoming requests are executed in a fast manner which is much better than in other languages like Ruby or Python.

**Databases**

In considering data would be received both in Python and NodeJS, database management system (DBMS) should be compatible with both of Python and NodeJS. MySQL as an open-source rational database management system provides reliable, stable and powerful tools for the user to manage the database. It is highly adaptable for Python and NodeJS. There were three steps to set up the database environment.
1.   Download MySQL Community server from dev.mysql.com
2.   Install PyMySQL library for Python by instructions
                    pip install PyMySQL
3.   Install MySQL for NodeJS by npm tools
                      npm install mysql

**Python libraries**

In addition to pre-installed libraries, scikit-learn machine learning library and other combined libraries were installed. The combined libraries consist of panda library, Numpy library and CSV library. All of the libraries can be installed via pip.

## Receiving data from Arduino

As mentioned before, data collected by Arduino would be transmitted through a serial port. Correspondingly, as a receiver, programs on the server would be listening to the serial port. Such Data receiving programs contained five parts including serial-write, serial-read, data validation check, data type distinguish and data storage. Using Python's built-in library serial, a serial object would be initialized at the start with the timeout of 1 second and baud rate of 9600 Hz. After that, the serial object would broadcast a message with one character from a list. The list contained seven single characters which represented each sensor. Each time after broadcasting, serial.read() function, which was at the start of While loop, would be called waiting for incoming datasets. If data was received successfully, validation-check functions would be called to test the integrity of data, otherwise, the previous character would be emitted again. The last two steps were to parse the data to appropriate type for storage. If ValueError exceptions were thrown during the parse procedure, that means the data was invalid. The system would call serial.write() again to broadcast the previous character and continue to the beginning of the While loop.

## Bluetooth service and application

### Bluetooth communication overview

There are two major players involved in BLE communication: the central and the peripheral. According to traditional client-server architectures, a peripheral typically has data that is needed by other devices. A central usually uses the information spread by peripherals to accomplish some task [19].

Peripherals broadcast some of the data they have in the form of advertising packets. An advertising packet is a relatively small bunch of data that contain useful information about what a peripheral has to offer, such as the peripheral's name and primary functionality [19]. In BLE, advertising is the primary way that peripherals make their presence known. A central device, on the other hand, can scan and listen for any peripheral device which is advertising information that it's interested in. For a central device can ask to connect to any peripheral that it has discovered advertising, the raspberry pi was allocated as a peripheral to provide services for uploading accelerometer data from mobile phone. In this case, the mobile phone would be central.

Peripherals may contain one or more services or provide useful information about their signal strength. Services themselves are made up of characteristics or included service (which is, references to other services). A characteristic provides further details about a peripheral's service. For instance, the blue service provided by raspberry pi may contain one characteristic that notifies the central about the room's status and another characteristic that allows the central to transmits accelerometer data. Figure displayed below illustrates one structure of a blue service and characteristics.

| **Service** Blue service | **Characteristic** Room status notify | **Characteristic** Accelerometer data load |
| --- | --- | --- |
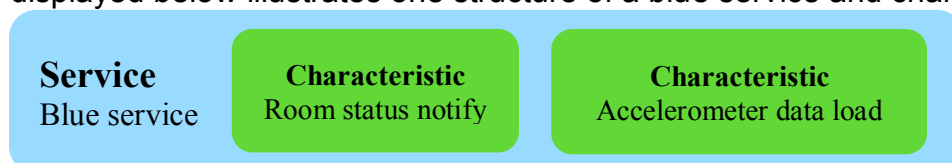
## IOS Bluetooth application

On the central side, a local central device (which is iPhone 6s in the experiment) is represented by a CBCentralManager object. The object is used to manage discovered or connected peripheral devices (represented by CBPeripheral objects), including scanning for, discovering, and connecting to advertising peripherals. As mentioned in the design section, discovered peripheral devices were displayed in a table view. The table view object has many trigger functions. One of them is *didSelectRowAt* which would tell the delegate that the specified row was selected. When the didSelectRowAt was called, selected peripherals would be assigned to BLEPeripheral which was specified globally and the CBCentralManager would try to connect to the BLEPeripheral. If the connection was built successfully, a didConnect method of centralManager object would awaken. As a result, the StoryBoard which manages the view of application would switch the current UIViewController to another one called ConversationViewController. The peripheral object and centralManager object would also be passed into the new UIViewController. Such method can be seen in the code snippet below:

```
154    func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral) {
155        print("Connected!")
156        // once connected, move to new view controller
157        let storyboard = UIStoryboard(name: "Main", bundle: nil)
158        let conversationViewController = storyboard.instantiateViewController(withIdentifier: "ConversationViewController") as! ConversationViewController
159        conversationViewController.peripheral = peripheral
160        conversationViewController.centralManager = central
```

After connecting to the peripheral, discoverable services and characteristics provided by the peripheral were listed and stored at first. Then, the central device would start to collect accelerometer data continually. For the peripheral created by raspberry pi provided write and notify services, two switch buttons can be turned to load data and subscribe notification-services at any time. If the button of loading data was switched on, the central device would write the just collected accelerometer data by utf8-encoded string to the peripheral. If the button of subscribing notification was switched on, the peripheral object would set notify-value as true so that the central device could receive message from the peripheral. Once the peripheral had broadcasted a message, the central device which had been subscribing the peripheral would receive a notification from the peripheral by calling didUpdateValueFor method of the peripheral function. Such method can be seen in the code snippet below:

```
378    func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic: CBCharacteristic, error: Error?) {
379        print("DidUpdateValue for characteristics")
```

The application also provided a method for replying to messages from the server. A reply button on the right bottom corner would be enabled when the central device had received messages. If no message had been received, the reply button remained disabled. The replyToServer method can be seen in the code snippet below:

```
267    @objc func replyToServer(){
268        let firstindex = lastMessage.index(of: "(") ?? lastMessage.endIndex
269        let lastindex = lastMessage.index(of: "\n") ?? lastMessage.endIndex
270        var message = String(lastMessage[firstindex..<lastindex])
271        if(decision){
272            message += "1"
273            self.writeToPeripheral(message: message)
274        }else{
275            message += "0"
276            self.writeToPeripheral(message: message)
277        }
```

## Bluetooth service on the server

Raspberry Pi as a peripheral advertised a service called Blue. The Blue service would call a Primary-Service provided by the bleno module. The Primary-Service has its unique UUID and two characteristics including WriteOnlyCharacteristic and NotifyOnlyCharacteristic.

The WriteOnlyCharacteristic aimed to receive accelerometer data from the central – mobile phone. This characteristic provided a writing service which allows the central, which had been connected to the peripheral, to transmit data to the peripheral. For data received were going to be inserted into the database, some If statements and functions would be used to parse and verify the data. Finally, an insert-function would be called by a handle-Data object to insert data into the database. For the convenience of the experiment, a process event was created to catch signals of keyboard interrupt. Once a signal of 'SIGINT' was caught which mean that (Ctrl + c) key combination was pressed, a global variable – insert would be switched between 0 and 1. Such a design could either stop or restart inserting data into the database at any time.

The NotifyOnlyCharacteristic was used to inform the users of a part of the current status of the room. The characteristic allowed a connected central to subscribe the peripheral so that the central device could receive information from the peripheral periodically. A period was typically set as five seconds. Such service of the characteristic can be seen in the code snippet below:

```
18  NotifyOnlyCharacteristic.prototype.onSubscribe = function(maxValueSize, updateValueCallback){
19      console.log('NotifyOnlyCharacteristic subscribe');
20
21      this.changeInterval = setInterval(function(){
22          const demo_str = "date(2018-03-20 12:15:23)\n"+ "Closing the window";
23          const data = Buffer.from(demo_str, 'ascii');
24          updateValueCallback(data);
25      }.bind(this), 5000);
26  };
```

As indicated as names of the characters, both the WriteOnlyCharacteristic and NotifyOnlyCharacteristic provided only one service. Such a structure of the service would be easier to maintain and read.

## *Machine Leaning progress*

**Labelling**

In supervised learning, training data includes a set of samples with paired input subjects (typically a vector) and desired output (which is also referred to as the supervisory signal). The supervisory signal should be set manually by experimenters. Since there was vast number of training data need to be labelled, reusable labelling programs were written to improve the labelling progress.

The labelling progress was divided into several steps including connecting to the database, defining table name, set time slots for selecting data samples and input activities to update data clauses. The time slots were accurate in seconds. Thus,

bunches of data in different time can be set accurately. An example of such a program can be seen in the code snippet below:

```
today = datetime.date.today()

time_start = input("What time do you want to start update?\n")
time_start = today.strftime('%Y') + " " + time_start
print(time_start)

time_end = input("What time do you want to end update?\n")
time_end = today.strftime('%Y') + " " + time_end
print(time_end)

activity = input("What activity do you want update?\n")
print(activity)

try:
    c.execute("UPDATE " + table_name + " set activity = '" + activity + "'" + "where date BETWEEN '%s' and '%s'" %
(time_start, time_end))
    db.commit()
    print ("Total number of rows updated : %d" % db.total_changes)
except:
    db.rollback()
```

## Data export and noise remove

Data used for training and classifying were fetched from the database and exported to CSV files. For data were collected from the real world, noise could not be avoided. Noisy data could miss lead machine learning program to select parameters and predicting data to wrong classes. Therefore, noisy data must be removed as far as possible. To distinguish noise, patterns of different types of data should be discovered. In this project, data patterns were quite straightforward. Thus, noises can be removed manually for each type of data.

## Model selection
### Model training strategy

In machine learning procedure, the strategy was to training models for each sensor at first. When such models were able to classify data collected from each sensor, an integrated model would be selected to classify activities from combined data. Input features of the model would be results predicted by models trained for single sensors. Labels were pre-signed. The strategy could possibly improve the accuracy of classifying activities happened in the whole space since features were reduced after applying each model trained for a single sensor.

## Data import

After preprocessing progress, data were ready to use. Panda library provides the interface to read contents of csv files. Using pd.read_csv function, data would be imported. The imported data were going to be divided into two matrices - features and activity. The total rows of two matrices were same and each row of the matrix represents a sample.

## Model training and selection
### Models for each sensor

Some researchers [11, 12] working in the area of human activity recognition adopt Support Vector Machines (SVMs) for solving problems of activity recognition using input from accelerometer data. Related experiments [13, 14] showed that SVM performed well in training accelerometer data. In terms of other data including

pressure and sonar data, the relationship between features(data) and labels(activity) is linear and straightforward. SVM with linear kernel could also make predictions precisely. Thus, this project decided to choose SVM for all of sensors.

Before training the model, feature selection for motion data was considered. According to researchers [24], there are several useful features for human activity recognition like: mean value, standard deviation of accelerometer axis. In this project, standard deviation of accelerometer axis and three accelerometer axes were used as features for training models separately.

In training SVM models, the main task was to choose appropriate parameters including c and kernels. The parameter c trades off misclassification of training examples against the simplicity of the decision surface. A low c makes the decision surface smooth, while a high c aims at classifying all training examples correctly. Proper choice of c and kernel is critical to the SVM's performance. Because c has wide range values corresponding to different data sets, adjusting c manually would take too much time. Therefore, it is advised to use *sklearn.module_selection.GridsearchCV* with c spaced exponentially far apart to choose a good value. Similarly, kernels were chosen in the same way. In the end, the best combination of parameters would be chosen to train a module for final uses. As *sklearn.external* includes a module called *joblib* which can persist the model for future use without having to retrain, model which was dumped previously could be loaded back later by *joblib.load*.

An example of such a program can be seen in the code snippet below:

```
39  Cs = np.logspace(-6, 3, 10)
40  parameters = [{'kernel': ['rbf'], 'C': Cs},
41                {'kernel': ['linear'], 'C': Cs}]
42
43  svc = SVC(random_state = 12)
44
45  clf = GridSearchCV(estimator = svc, param_grid = parameters, cv = 5, n_jobs = -1)
46  clf.fit(x_train, y_train)
47
48  print (clf.best_params_)
49  print (clf.best_score_)
50  test_score = clf.score(x_test, y_test)
51  print(test_score)
52  joblib.dump(clf, 'mobileACC.pkl')
```

**Model for integrated data**

When it came to predicting activities by assorted data from each sensor, data would be processed by the feature-reduction procedure at first. A sample of features looks like below:

| lux | tem_in | tem_win | sonar_in | sonar_out | move | p | \ | \ |
|-----|--------|---------|----------|-----------|------|-----|-----|-----|
| 40 | 19.5 | 19.7 | 292 | 298 | 0 | 103 | \ | \ |
| adx | ady | adz | awx | awy | awz | mox | moy | moz |
| -0.95 | 0.01 | -0.19 | -0.94 | -0.03 | -0.02 | 0.0034 | -0.0051 | -0.0051 |

The first-row lists titles for features. As shown in the table, there are sixteen features corresponding to different values. However, in the specification of strategy, features are reduced from sixteen to nine. For instance, 'sonar_in' and 'sonar_out' are integrated to single feature representing sonar signal. Each three of features behind 'p' indicate accelerometer. More specifically, (adx, ady, adz), (awx, awy, awz) and (mox, moy, moz) are data from an accelerometer on the door, an accelerometer on the window and accelerometers inside a mobile phone respectively.

The procedure of feature reduction was quite simple. As pre-trained models were ready to classify each type of data, features(data) discussed above would be passed into models. As a result, activities predicted by models were going to be recombined into feature vectors. Before regrouping such results to features, the results, which were encoded in string also known as text data, would be transformed to numerical data. Additional functions were written for such a purpose. Functions are similar which use if…else statements to distinguish different strings. An example of the functions can be seen in the code snippet below:

```python
22  def transform_pressure(var):
23      #SITTING:0, STAND:1
24      if var == "SITTING":
25          return 0
26      elif var == "STAND":
27          return 1
28      else:
29          return 2
```

All procedures above were to prepare data for training. It was crucial to process data following such steps which aimed to guarantee that training procedure could conduct successfully.

The last but not least, instead of using SVM, Decision Tree was considered as a better choice to train model. Though SVM and Decision Tree were both found to be effective and better than other classifiers [15], the results trained by Decision Tree can be more intuitional because of its visualized tree structure. When model had been trained, the model (tree) can be exported into a graph by itself. Such a program can be seen in the code snippet below:

```python
127  clf = tree.DecisionTreeClassifier()
128  clf = clf.fit(train_x, y_train)
129  dot_data = tree.export_graphviz(clf, out_file=None, feature_names = feature, filled=True, rounded = True,
         special_characters = True)
130  graph = graphviz.Source(dot_data)
131  graph.render("sensor.gv", view=True)
132  print(graph)
```

While the tree should be pruned after fitting the data, the Decision tree module in *sklearn* has done the pruning step for us. In addition, Tree algorithms such as ID3, C4.5 and CART which should be chosen beforehand had been adopted with an optimized version of CART algorithm by *scikit-learn*.

## 7.4 Problems encounter and solutions

*Hardware problems*

**Battery**

During the experiment, batteries used for powering the Arduino Nano experienced some failures. At the very beginning, two batteries with 1.5-volt voltage were installed. An Arduino Nano connected to a ZigBee module and several sensors could run successfully. However, the data received by the server (which is raspberry pi) were all invalid that the data were printed as NULL. The problem was finally found that the combined voltage of batteries was not enough to support the Arduino Nano, the ZigBee module and several sensors working normally at the same time. Therefore, to gain a high enough voltage, a voltage-booster module was connected to batteries. At this time, the data was received successfully by the server. However, the transmission was not stable that the data could be easily lost during the transfer. By measuring the voltage and current of the batteries, the output current was found out to be too small to power the ZigBee Module and sensors. Since power (P = VI) must be preserved. When the output voltage was boosted, the output current was decreased relatively. The relation of V and I can be seen as below:

$$P = VI \underset{(source)}{} = \underset{(output)}{V_\uparrow I_\downarrow}$$

Finally, a 9-volt battery was found to power Arduino Nano, the ZigBee module and sensors without problems. The data could still be lost during the transmission, but the reason was not the same now. This problem would be discussed later.

*Software problems*

**Data jamming**

In this project, Data-jamming was a phenomenon that happened when various streams of data were received by the serial port of the server. A reason for this problem was that data streams sent from Arduinos were out of order. While each Arduino had its own time clock, without communicating with other Arduinos, every Arduino would send data according to its own time. Therefore, Data-jamming had happened sometimes. Figure 2-1 below shows such a problem:
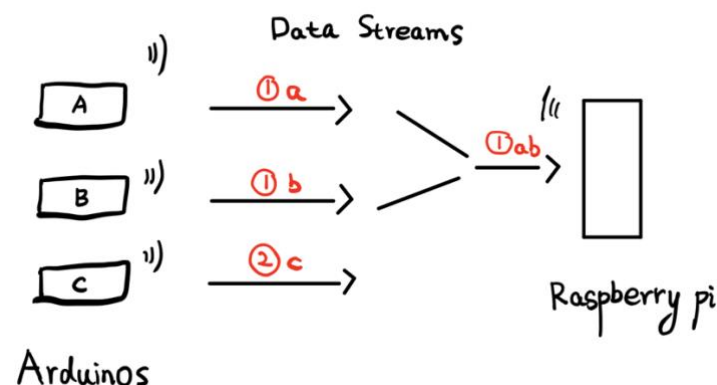
Figure 2-1 Data jamming

In Figure 2-1, Arduino A and B sent their data streams at the same time while C was waiting behind them. When the server (which is Raspberry pi) was about to receive the data sent from both A and B, the server was unable to distinguish which one was steam from A and which one was steam from B because the data just arrived at the same time. Thus, the serial port of raspberry-pi would pick the data which was at the front. Most of the time, the data retrieved was a mixture of *a* and *b*. Sometimes, the data would be a part of *a* and *b*. Hardly ever, the data was valid which was a complete *a* or *b*. In this data-transmission relationship, the server was passively waiting for the data streams.

To solve the problem, the data arrived at the serial-port should in the sequence that the server could retrieve a complete data stream. For such a purpose, a new protocol between the Arduino and Raspberry-pi was designed. By implementing the protocol, the server became initiative to request the data from Arduino. As mentioned in the section of receiving data, the server would broadcast a special character which was unique for each sensor. If the character matched a unique id of an Arduino, that Arduino would respond to the server with its data stream. By iterating a list of all sensors, the data of every sensor could be collected less than half seconds. Figure 2-2 below shows an example of the protocol:
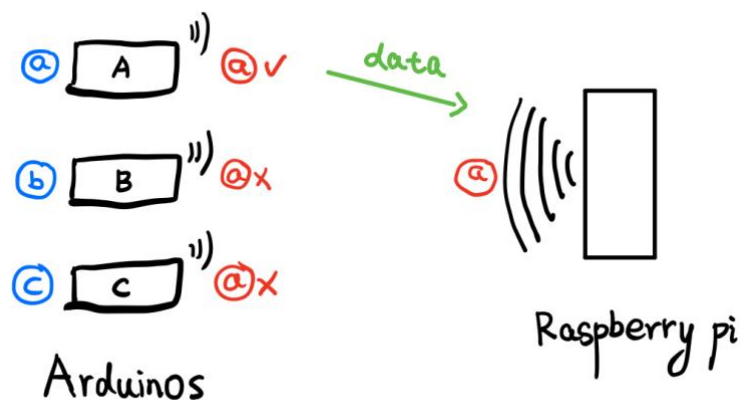


Figure 2-2 A new protocol for data transmission

Such a solution was useful when the number of sensors was relatively small. However, if the number of sensors was enormous, iterating the list of sensors would take too much time.

# 8. Evaluation

## Overview

As part of the original objectives set out in the introduction section, the classification results of trained models need to be evaluated. Initially, we will consider the quality of the data collected by sensors, along with the effectiveness of the data transmission implemented. Continuing from there, we will analyse the performance of the models by evaluating the accuracy of classification and looking to future changes if so required.

## Evaluation of collecting data

Due to the limitation of time, so far only data to be trained for each model had been collected while the rest of data for testing in integrated-model were not available. In this section, the procedure of collecting data and data quality would be discussed.

The evaluation of the procedure includes assessing the effectiveness and reliability of the programs. According to section 7.4, through the problem of data jamming was solved in the end, the data still could be lost in high probability. The unstable transmission happened more frequently in finally integrated data-collection procedure while other transmissions in collecting data from single sensor had rarely experienced losing data. The reason for such a difference is the unstable performance of ZigBee. According to researchers' evaluation [16], broadcast should be reliable in ZigBee Networks. However, none of ZigBee forward node selection algorithms had been implemented in ZigBee module in this project. Lacking knowledge of how the hardware performed caused much trouble in finding an appropriate way to collect data. In the end, my solution was to re-request the data if the data from the previous transmission was invalid which would be inefficient if enormous data was required.

In terms of the data quality, because the size of data collected was not much great, the quality of the data could be easily analysed in noise-remove process. As noise-remove was accomplished manually, many unobvious data was found. For example, the accelerometer data collected from mobile phone had part of undistinguished data which was labelled with WALKING. Such data had no difference with data labelled with STAND. Some examples can be seen as below:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1396 | 2018/3/20 11:22 | -0.0139 | 0.0014 | 0.0014 | STAND | 1197 | 2018/3/20 11:21 | 0.0349 | -0.0036 | -0.0036 | WALKING |
| 1397 | 2018/3/20 11:22 | -0.0073 | -0.0033 | -0.0033 | STAND | 1198 | 2018/3/20 11:21 | 0.0348 | -0.0056 | -0.0056 | WALKING |
| 1398 | 2018/3/20 11:22 | -0.0039 | 0.0014 | 0.0014 | STAND | 1199 | 2018/3/20 11:21 | 0.0354 | -0.0044 | -0.0044 | WALKING |
| 1399 | 2018/3/20 11:22 | 0.0012 | 0.0032 | 0.0032 | STAND | 1200 | 2018/3/20 11:21 | 0.0349 | -0.0048 | -0.0048 | WALKING |
| 1400 | 2018/3/20 11:22 | -0.0177 | -0.0052 | -0.0052 | STAND | 1201 | 2018/3/20 11:21 | 0.0341 | -0.0064 | -0.0064 | WALKING |
| 1401 | 2018/3/20 11:22 | 0.0005 | -0.0046 | -0.0046 | STAND | 1202 | 2018/3/20 11:21 | 0.0357 | -0.0042 | -0.0042 | WALKING |
| 1402 | 2018/3/20 11:22 | -0.011 | 0.0028 | 0.0028 | STAND | 1203 | 2018/3/20 11:21 | 0.0349 | -0.0044 | -0.0044 | WALKING |
| | | | | | | 1204 | 2018/3/20 11:21 | 0.0346 | -0.0047 | -0.0047 | WALKING |
| 1403 | 2018/3/20 11:22 | -0.0101 | 0.0038 | 0.0038 | STAND | 1205 | 2018/3/20 11:21 | 0.0358 | -0.0054 | -0.0054 | WALKING |
| 1404 | 2018/3/20 11:22 | -0.0082 | 0.003 | 0.003 | STAND | 1206 | 2018/3/20 11:21 | 0.0344 | -0.0042 | -0.0042 | WALKING |
| | | | | | | 1207 | 2018/3/20 11:21 | 0.0344 | -0.0051 | -0.0051 | WALKING |

The data all above could be considered to be 0 according to accelerometer's specification, which also means that the mobile phone remained static without having been moved. Some possible reasons proposed below could give the explanation:

- Reason 1: the accelerometer sensor had not been calibrated before starting experiments.
- Reason 2: the gravity was not considered when calculating the acceleration of three axes.
- Reason 3: the motion of range was not obvious in WALKING.

In my opinion, when the mobile phone was in the pocket, if the user was walking slowly, the applied force on the mobile phone could not generate acceleration that is

obvious enough to be measured. As a result, the trained model for the accelerometer data from mobile phone had poor performance in distinguishing whether the user was walking or standing.

Other data was distinct except the data of sonar sensor. When measuring the distance from the sonar sensor to the door, the accuracy differed when the status of door changed. The margin of error was small when the door was opened in half. When the door was semi-opened, the distance measured was variant. Some examples can be seen as below:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2018/3/20 11:54 | 262 | 178 | SEMI | 1064 | 2018/3/20 11:57 | 288 | 285 OPEN |
| 2018/3/20 11:54 | 286 | 269 | SEMI | 1065 | 2018/3/20 11:57 | 290 | 289 OPEN |
| 2018/3/20 11:54 | 269 | 150 | SEMI | 1066 | 2018/3/20 11:57 | 298 | 298 OPEN |
| 2018/3/20 11:54 | 267 | 171 | SEMI | 1067 | 2018/3/20 11:57 | 293 | 298 OPEN |
| 2018/3/20 11:54 | 113 | 218 | SEMI | 1068 | 2018/3/20 11:57 | 292 | 298 OPEN |
| 2018/3/20 11:54 | 226 | 137 | SEMI | 1069 | 2018/3/20 11:57 | 288 | 299 OPEN |
| 2018/3/20 11:54 | 233 | 125 | SEMI | 1070 | 2018/3/20 11:57 | 300 | 287 OPEN |
| 2018/3/20 11:54 | 313 | 292 | SEMI | 1071 | 2018/3/20 11:57 | 299 | 293 OPEN |
| 2018/3/20 11:54 | 210 | 180 | SEMI | 1072 | 2018/3/20 11:57 | 297 | 300 OPEN |
| 2018/3/20 11:54 | 246 | 294 | SEMI | | | | |
| 2018/3/20 11:54 | 122 | 127 | SEMI | | | | |

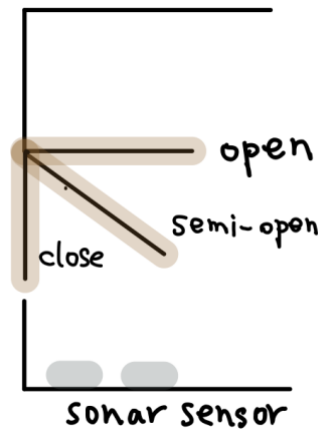Different status of the door can be referenced to Figure 2-3;



Figure 2-3: Different status of the door

Some Technical guides for ultrasonic sensors pointed out that if the target is tilted there is a great effect on the measurement. Therefore, the difference of distance to the semi-opened door could be explained.


# Evaluation of machine learning progress

## *The strategy of evaluation*

The evaluation of machine learning progress would focus on model evaluation. According to related theories (Novaković, J.D, 2017), using accuracy as a measure for evaluation has some disadvantages. Such measurement would ignore the differences between the types of errors. And not only that, the results depend on the distribution of class in datasets. In this case, F-measure score and precision score were proposed to evaluate models in a more comprehensive way.


## *Evaluation of types of models*

When the model was trained, the testing procedure would start. The test data would pass to the model. As a result, a score would be outputted which represents the mean accuracy of prediction. According to the documentation of model evaluation [23],

higher return values are better than lower return values. In addition, a simple cross-tabulation of predicted results would be printed as well. Such a table could clearly reflect miss-classified data and help calculate confusion matrix.

**Model for mobile phone's accelerometer data**
The output of this model can be seen below:

```
0.401534526854
Predicted  SIT_MOVE  STAND  WALKING  All
True
SIT               0    387        1  388
SIT_MOVE          0     13        4   17
STAND             0    275        0  275
WALKING           3     60       39  102
All               3    735       44  782
```

The decimal number 0.40 is the mean accuracy which is quite low. By observing the cross-tabulation, only STAND was classified in 100% precision. SIT_MOVE and SIT were both totally miss-classified to STAND and WALKING. In predicting WALKING, 39% of data was classified correctly which is poor. Another point is that nearly all of the SIT was considered as STAND. The reason for this is that the SIT and STAND are both belong to static. In other words, the accelerometer data would be relatively same while the user was either sitting or standing. After re-labelling the SIT and STAND data to STATIC, the mean accuracy increased. A new output can be seen as below:

```
0.877237851662
Predicted  SIT_MOVE  STATIC  WALKING  All
True
SIT_MOVE          0     14        3   17
STATIC            0    663        0  663
WALKING           3     76       23  102
All               3    753       26  782
```

However, as mentioned before, the accuracy score has disadvantages. According to the cross-tabulation, the sample of STATIC dominates while SIT_MOVE and WALKING are 1/7 proportions of the total. If the sample of SIT_MOVE and WALKING, the accuracy score would be poor because the model still could not distinguish between static and movement. To calculate the F-measure score and precision score, the cross-tabulation a confusion matrix.

The precision, recall and F-measure score can be calculated following the formula below:

$$P = \frac{Tp}{Tp+Fp} \qquad R = \frac{Tp}{Tp+Fn} \qquad F1 = 2\frac{P \times R}{P+R}$$

Therefore, precision and recall were calculated as below:

|  | SIT_MOVE | STATIC | WALKING |
|---|---|---|---|
| Recall | 0 | 1 | 23/102 |
| Precision | 0 | 221/251 | 23/26 |
| F1 | 0 | 0.99 | 0.36 |

To conclude briefly, the overall score of STATIC is pretty good which means that the model could recognize the activities well when the user remains sitting or standing. As for WALKING, although precision is not bad, the F1 score and recall are quite low with 0.36 and 0.26 respectively. Therefore, WALKING could sometimes be distinguished among activities.

As mentioned before, stand deviation of accelerometer axis would be a feature for training another model. Comparing to three accelerometer axis, this result was better:

```
0.901534526854
Predicted  STATIC  WALKING  All
True
SIT_MOVE       11        6   17
STATIC        662        1  663
WALKING        59       43  102
All           732       50  782
```

Corresponding precision and recall were calculated as below:

|  | SIT_MOVE | STATIC | WALKING |
|---|---|---|---|
| Recall | 0 | 662/663 | 43/102 |
| Precision | 0 | 331/366 | 43/50 |
| F1 | 0 | 0.99 | 0.56 |

Applying strategies of result comparison in [26], the precision and recall of predicting WALKING both increased. The F1 score was better as well. Such an improvement indicates that the model had stronger capability to distinguish WALKING among other activities. The evidence proves that the feature selection is very important. Therefore, more different features could experiment in the future.

Other models such as sonar, pressure and accelerometer on door and window are evaluated in the same ways. Their performance is all good excepting the sonar which can hardly recognize the semi-open status of the door.

**Model for integrated data**

Because of time limitation, test data had not been collected yet for evaluating the integrated model. Only a tree structure outputted can be used to do some evaluation. Figure 2-4 shows the structure of Decision Tree:
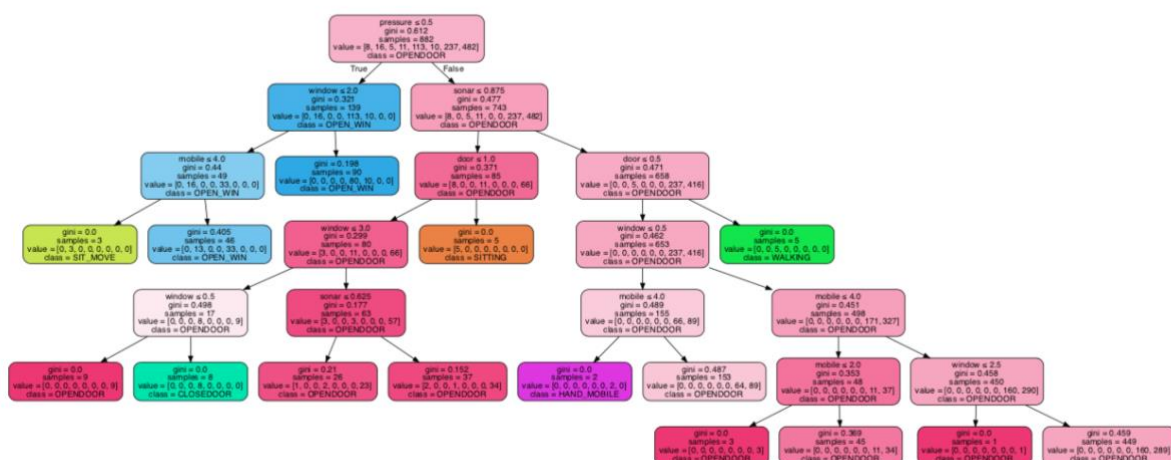


Figure 2-4 Decision-tree structure

The tree is a binary tree. The overall structure is pretty good. Each leaf of the tree reflects the classification results. Most of the leaves are pure except a leaf in depth of three with value = [0,13,0,0,33,0,0,0]. Such a leaf could not distinguish between 13 of samples and 33 of samples. Rather, the leaf predicted all of 46 samples as OPEN_WIN while these two parts should be classified to two different classes. Similar leave also can be found in depth of six. By discovering the detail of the tree-structure, the model still has the drawback of distinguishing some activities including OPENDOOR and OPEN_WIN which happened frequently. The performance of the model could not be evaluated with lacking test-data.

## *Constraints and Challenges*

First of all, the data was not representative enough because some of the activities happened just seconds that the collections of corresponding data of such activities were small. Moreover, the quality of collected data was not reliable which also mislead programs to train poor models. A way to improve the performance of the model is to guarantee the quality data which will be feed to models. To obtain data with good quality, reliable sensors and precise experimental operation are necessary.

## 9. Time Management

Reflecting on the Gantt chart (Appendix C), the schedule is slightly different due to the uncontrollable factors such as unexcepted batteries issues during the experiment. While some works including user testing and sensor deployment were logged, all the other plans were achieved on time. However, the expected user testing was not accomplished while user testing procedure was quite important. The negligence in arranging the time to collect data caused missing of user-testing. Hence, I think the time was not managed well.

## 10. Achievement

Over the course of the project, a number of small but important goals have been accomplished. A local server was set successfully on the Raspberry Pi, where data was collected and stored. Meanwhile, the Arduino Nano collected environmental data smoothly. In addition, ZigBee modules enable data to be transmitted wirelessly.
Another signification achievement was IOS Bluetooth application. The application has simple user interface which is intuitive for people to use. Such an application could also be an alternative to wearable motion sensors in some extents. Moreover, the application provides interfaces to receive information about the room from the server. The feedback functionality would help the programs adjusting the labelling strategy (Future design). The last one is programs of machine learning. Models were all trained and outputted successfully. Though the performance of models was not as exact as expected, at least, the idea of designing the programs to train models for recognizing human activities is practical and feasible.

## 11. Self-Reflection

This project is conducive to me in many aspects. First and foremost, learning to do related research before actually starting the implementation of system help me be aware of the importance of preparation. Even though unexpected issues of batteries still happened during the experiment, I solved it successfully in the end. Secondly, during the development of IOS application, I learned a new programming language – Swift. In addition, I understood how Bluetooth devices build connections and transmit data. Thirdly, being failed to train well-performed models let me know that the quality of data was crucial in machine learning. Meanwhile, I realized that real-world environment was much more complex than I thought before. I should consider carefully in every details. Finally, a wide range of practical skills was learned and improved during the development of the project including communication, researching unknown field including various sensors and wireless transmission protocols, interactive system designing and evaluation, report writing and project management skills.

## 12. Conclusion and Future work

Looking broadly at the project as a whole, functionalities of interactive system were accomplished to certain extents. The design proposed alongside with the implementation and evaluation on the machine learning models indicate that it is highly practical to apply machine learning technologies in recognizing human activities. By analysing the performance of models, we can find that the quality of data is a crucial factor to train a good model. In addition, good feature selection strategies also make effect. In general, designing an interactive system requires cross-domain knowledge

in both hardware and software. A wide range of possible issues should be considered in the design process.

This project will be continued over this year. The work will mainly focus on designing sensor selection algorithms, protocols of data transmission and labelling strategies. Meanwhile, the development of machine learning programs will keep going on.

# 13. Bibliography

1. Wilson D H, Atkeson C. Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors[C]//International Conference on Pervasive Computing. Springer, Berlin, Heidelberg, 2005: 62-79.
2. Tapia E M, Intille S S, Larson K. Activity recognition in the home using simple and ubiquitous sensors[C]//Pervasive. 2004, 4: 158-175.
3. Chua S L, Foo L K. Sensor selection in smart homes[J]. Procedia Computer Science, 2015, 69: 116-124.
4. Incel O D, Kose M, Ersoy C. A review and taxonomy of activity recognition on mobile phones[J]. BioNanoScience, 2013, 3(2): 145-171.
5. Tang B. The Emergence of Artificial Intelligence in the Home: Products, Services, and Broader Developments of Consumer Oriented AI[J]. 2017.
6. Bregman D. Smart Home Intelligence–The eHome that Learns[J]. International journal of smart home, 2010, 4(4): 35-46.
7. Wang S, Zhou G. A review on radio based activity recognition[J]. Digital Communications and Networks, 2015, 1(1): 20-29.
8. Michael F, Miles K. Interactive Architecture[J]. 2009.
9. Cook D J, Holder L B. Sensor selection to support practical use of health-monitoring smart environments[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2011, 1(4): 339-351.
10. Palumbo, F., Gallicchio, C., Pucci, R. and Micheli, A., 2016. Human activity recognition using multisensor data fusion based on reservoir computing. *Journal of Ambient Intelligence and Smart Environments*, *8*(2), pp.87-107.
11. Z. He and L. Jin, Activity recognition from acceleration data based on discrete consine transform and SVM, in: IEEE Inter- national Conference on Systems, Man and Cybernetics, 2009, SMC 2009, IEEE, 2009, pp. 5041–5044.
12. Ravi N, Dandekar N, Mysore P, et al. Activity recognition from accelerometer data[C]//Aaai. 2005, 5(2005): 1541-1546.
13. Jongboom, J. (2015). Live analyzing movement through machine learning. [online] Available at: http://blog.telenor.io/2015/10/26/machine-learning.html [Accessed 10 Apr. 2018].
14. Becker, N. (2016) Predicting Human Activity from Smartphone Accelerometer and Gyroscope Data. [online] Available at: https://beckernick.github.io/activity-prediction/ [Accessed 10 Apr. 2018].
15. Ahmad, M. and Khan, A.M., 2017. Seeking Optimum System Settings for Physical Activity Recognition on Smartwatches. *arXiv preprint arXiv:1706.01720*.
16. Boukerche, A. ed., 2008. Algorithms and protocols for wireless, mobile Ad Hoc networks (Vol. 77). John Wiley & Sons.
17. Novaković, J.D., Veljović, A., Ilić, S.S., Papić, Ž. and Milica, T., 2017. Evaluation of Classification Models in Machine Learning. *Theory and Applications of Mathematics & Computer Science*, *7*(1), pp.39-46.
18. GitHub, "A Node.js module for implementing BLE peripherals", 2015. [Online]. Available: https://github.com/noble/bleno
19. Apple, "Core Bluetooth Programming Guide", 2013. [Online]. Available: https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothOverview/CoreBluetoothOverview.html#//apple_ref/doc/uid/TP40013257-CH2-SW1
20. Arduino, "GETTING STARTED | FOUNDATION > introduction", 2018. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

21. Wikipedia, "Raspberry Pi", 2018. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi

22. Watkins, J. (2016). Configuring The GPIO Serial Port On Raspbian Jessie and Stretch Including Pi 3. [online] Available at: https://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3/#Enabling

23. Scikit-learn "Model evaluation: quantifying the quality of predictions". 2017. [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html

24. Casale, P., Pujol, O. and Radeva, P., 2011, June. Human activity recognition from accelerometer data using a wearable device. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 289-296). Springer, Berlin, Heidelberg.

25. Alsheikh, M.A., Lin, S., Niyato, D. and Tan, H.P., 2014. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, *16*(4), pp.1996-2018.

26. Liu, L., Peng, Y., Liu, M. and Huang, Z., 2015. Sensor-based human activity recognition system with a multilayered model using time series shapelets. *Knowledge-Based Systems*, *90*, pp.138-152.

27. Borges Neto, J.B., Silva, T.H., Assunção, R.M., Mini, R.A. and Loureiro, A.A., 2015. Sensing in the collaborative internet of things. *Sensors*, *15*(3), pp.6607-6632.