Activity Recognition and Analysis of IoT Smart-Wearable Sensor Data and Personalised Recommender by Applying Machine Learning Algorithm

THIRD REVIEW REPORT

for

Master Thesis
Course Code: CSE6099

Submitted by

Abhirup Dey
Reg No. 17MCS0044

Under the Guidance of

Prof. Rajkumar R

School of Computer Science and Engineering
VIT University, Tamil Nadu, India

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Declaration

I hereby declare that the project named **"IoT Security with the Application of Wearable Device Technology"** submitted by me to the *School of Computer Science and Engineering, Vellore Institute of Technology, Vellore-14* towards the partial fulfilment of requirements for the award of degree of **Master of Technology** in **Computer Science and Engineering** is a record of bonafide work carried out by me under the supervision of **Prof. Rajkumar R**.

I further declare that the work reported in this has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

_____

Abhirup Dey
(17MCS0044)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled **"Activity Recognition and Analysis of IoT Smart-Wearable Sensor Data and Personalised Recommender by Applying Machine Learning Algorithm"** is prepared and submitted by Abhirup Dey (Register Number: 17MCS0044), has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering in Vellore Institute of Technology, Vellore-14, India.

Prof. Rajkumar R	**Head of the Department**	**External Examiner**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

ACKNOWLEDEMENT

The project **"Activity Recognition and Analysis of IoT Smart-Wearable Sensor Data and Personalised Recommender by Applying Machine Learning Algorithm"** was made possible because of inestimable inputs from individuals involved, directly or indirectly. I would first like to thank my guide, **Prof Rajkumar R**, who was highly supportive and provided innovative base for this project but also crucial and constructive inputs that helped make my final product. My guide has helped me perform research in specified area and improving my understanding. I am very thankful for his guidance all throughout the project.

_____

ABHIRUP DEY

Table of Content

ABSTRACT

Wearable Technology also called wearable gadget, is a category of technology devices with low processing capabilities that can be worn by a user with the aim to provide information and ease of access to the master devices its pairing with. The impact of wearable technology becomes significant when people start their invention in wearable computing, where their mobile devices become one of the computation sources. However, wearable technology is not mature yet in term of device security and privacy acceptance of the public. The low processing due to less computing power of wearable device cause the developer's inability to equip some complicated security mechanisms and algorithm on the device.

In this smart emerging world, modern day equipment, like wearable devices, not only provides functionality or advancements in lifestyles but also becoming a trending fashion choice. Most of the devices which are wearable provides basic functionalities like display time or date. But implementation of more smart features like displaying message, phone call or even medical activity recognition can lead the productivity in dense and holds a potential to create a product demanded by huge number of customers. Smart wearable devices connected to internet approaches the methodology and required application and implementation of secure IoT environment and cloud infrastructure. Compared to other internet connected devices wearable devices like smart watches are designed to be capable of monitoring activity for 24 hours a day. Mostly they are designed as durable and water resistance with addition of appropriate sensors for required functionalities and detection. In this paper we are proposing a model for identifying requirements of activity and inactivity recognition by implementing on a secure and smartly designed cloud infrastructure. Here we are also defining a new measurement of heart-rate data applying various machine learning methods.

KEYWORDS

IoT, Smart-Wearable-Devices, Cloud, Machine Learning, Web Application, Activity Recognition, Personalized Recommendation

1.Introduction

The ongoing trend of not wearing a watch because of an available phone with clock functionality seems to be stopped. The origin functionality of watches, just to display the time, is tremendously enhanced by the new generation of smart watches. purposes, smart watches provide additional sensor functionality that might be used for third party application providers. Hereby mainly sport or wellness applications are in focus of the manufactures. Because of the smart watch´s sensor functionality, the device enables new and innovative applications that expand the original application field. The smart watches allow a permanent monitoring of the activities of the user. The acquired information on the physical activity of a person can be used for various purposes, such as wellness, safety, various psychological identifier (degree of attention), or detection of micro-activities in an overall state of inactivity.

For activity and inactivity recognition by smart watches it is necessary to be aware of capabilities of the sensors. This paper introduces the current work of sensor technology and describes a generic identifier of the performance of the acceleration sensing functionality of the smart watches. The paper closes with a summary and an outlook of the future work in the area of activity recognition on smart watches.

21st century wearable devices like smart watches not only offer visual display of time and date, but also gives us several other feature rich functionalities which helps to create advancement in day to day human life. Most recognised feature for smart watch is health monitoring. Wearable technology introduced the methodology of continuous monitoring of medical and personal data. This not only gives us productivity and efficiency, but also provide us a better way to live our life- A smart life.

Adding sensors to wearable device enhance the functionalities for collecting data about user activity. By collecting and storing the data into a database or secure storage we can provide a platform of innovation for third party vendors. There is various availability of application of accelerometer and gyroscope application to recognise activity of a particular user. Activity and inactivity recognition of individuals have become a current

development scenario for wearable devices. But we can do lot of enhancements and predictive approach by applying machine learning to those data set.

Activity recognition research originally utilized specially engineered devices distributed across a subject's body to identify the subject's physical activities, but in recent years much of this research began utilizing commercial smartphones which include the requisite sensors (i.e., accelerometers and gyroscopes). The use of these ubiquitous commercial devices greatly expanded the applications of activity recognition, but also introduced limitations due to their placement on a user's body and inconsistent orientation. For example, the smartphone could shift in a person's pocket and the pocket position (near the upper thigh) is not ideal for tracking hand-based activities.

Smartphone-based activity recognition is especially limited for women, since they typically do not keep the phone in their pocket. Most of these limitations are addressed by smartwatches, which are worn in a consistent position and which are ideally situated for tracking hand-based activities. Furthermore, since virtually all smartwatches work in tandem with smartphones, the sensor information from both devices can be utilized for activity recognition. This paper examines the use of smartphones and smartwatches for activity recognition. The performance of smartphone-based activity recognition is compared with the performance of smartwatch-based activity recognition— although we recognize that ultimately a combination of both devices may work best. The efficacy of the smartwatch accelerometer is also compared with the efficacy of the smartwatch gyroscope sensor for performing activity recognition. Our prior research demonstrated that smartphone-based personal activity recognition models— built with training data from the intended user—vastly outperform impersonal models, and this study shows that this advantage extends to smartwatch-based activity recognition models. Finally, this study also extends our prior work by recognizing many more activities, including hand-based activities (e.g., typing and writing). Of special note, this study includes many eating-related activities (e.g., eating soup, eating a sandwich, drinking), which opens up the possibility for new health-related activity recognition applications. The results in this paper demonstrate that smartwatches have the potential to accurately identify a large variety of activities, including hand-based and eating-based activities that cannot be effectively recognized by smartphones. Consistent with our prior smartphone work the

classification models are induced from labeled training data using standard machine learning algorithms.

Smartwatch and smartphone-based activity recognition has many applications. Generally, these devices can operate more intelligently if they are aware of what their user is doing (e.g., forwarding a call to voicemail during a meal). However, the main application of our research has been to improve people's health and wellbeing. Physical inactivity and unhealthy eating are two of the most powerful, modifiable risk factors for disease. Performing a sufficient amount of physical activity is important because physical inactivity dramatically increases the health risks for cardiovascular disease, colon cancer, and many other diseases—while a healthy amount of physical activity reduces the risk of all-cause mortality and could prevent two million deaths per year Similarly, there are significant health risks associated with excessive caloric intake. While many types of interventions seek to reduce the tendency toward overeating, long-term dietary adherence remains a major challenge. Activity monitoring can help combat both inactivity and overeating by providing accurate, real-time information about sedentary behaviour, exercise, and eating behaviour. The smartwatch is perfectly poised to convey this information since it is always readily and unobtrusively accessible—which is one reason why smartwatch manufacturers tout its potential to improve health. While there are some basic activity recognition applications for smartwatches, our work involves much more specific activities (including a variety of eating activities). Smartwatch-based apps capable of tracking eating activities could ultimately replace (or augment) the manually intensive methods for maintaining a food diary.

In this paper we will discuss about possible futuristic implementation of personalised activity recommendation based on the data collected through smart watch. Besides we will present the scenario of collecting heart rate data from test cases and predicting upcoming health issues by applying available machine learning model into it. Which does not only add valuable medical functionality, but also gives a boost to the existing scenario and an upgraded technical application.

2.Literature Review

2.1 Existing Scenario

Recently there has been a rapid increase in the number internet capable devices. A paradigm shift in computing has caused rapid development of personal devices. The latest trend in computing aims to interconnect sensors, devices, embedded systems, software and hardware to collect and exchange data. The Internet of Things (IoT) allows devices to connect to the internet thus creating a ubiquitous environment where devices share data and information for increased connectivity. IoT devices comprise an embedded system with valuable data which is being communicated often via short range wireless signals like Bluetooth. The IoT is a dense population of interconnected smart devices like smartphones, televisions, vehicles, health monitoring devices, home automation systems and wearable devices to name a few. Interconnectivity, varying purpose and capability creates a compelling case for ensuring security and safety of devices in the IoT to protect the user's data and privacy.

Wearable computing is an area that is closely related to pervasive and ubiquitous computing. Consumer wearable devices are being marketed that can be worn on the body for both business and personal use. What sets the wearable technology apart from conventional mobile devices is the fact that these devices are designed to be worn on the body and get carried. Wearable devices are also designed to augment knowledge and learning by enhancing users experience. By design these are miniature electronic devices composed of multiple sensors controlled by a microcontroller. The sensors can for example sense movement, physiological signals, atmospheric changes like temperature and humidity. The most common sensors found in modern wearable devices are accelerometers and gyroscopes. Lifestyle, health monitoring, fitness monitoring, entertainment, gaming and industrial support wearable devices cannot provide the wide range of features without these sensors.

- **Health monitoring**—devices embedded with sensors which can detect changes in the physiological signals of the body.

- **Fitness monitoring**—devices intended to enhance sports and fitness related activities through monitoring and tracking. Unlike health monitoring these devices are not intended to be worn by the user at all times.
- **Lifestyle**—general purpose wearable articles like smart watches and smart clothing which can provide cellular, internet connectivity etc.
- **Entertainment**—wearable devices that can stream audio and video. The devices in this category can be wireless headphones, speakers, and wearable displays with the ability of connecting to wide range of entertainment systems like smart TV's, digital media players, home theatre systems etc.
- **Gaming**—devices designed to enhance user experience during gameplay. Advanced devices are intended to create an immersive augmented environment through the use of 3D head wearable and walkable surround displays.
- **Industrial support**—devices that fall in this category are designed to aid the user in performing their task safely while increasing productivity and efficiency. These devices are intended to be worn by professionals in an industrial setting.

Consumers can purchase an internet capable version of almost every home appliance; but many devices still do not possess the ability nor the resources required for the provision of security. Devices that possess poor physical security, authentication, encryption, interfaces can become a target of spoofing, key theft and identity theft. Conventional security algorithms rely on stored cryptographic keys. If the keys are exposed then the system and its data is compromised. The ICMetric technology aims to resolve the issue of key theft by using the features of a device to create a device identification which can then be used for cryptographic key generation.

2.2. Related Work

In the 90´s, some watches [1] were designed with an integrated light sensor that enabled a connection to a PC. Hereby data were transferred to optical information on the monitor of the PC and could be received by a light sensor of the watch (e.g. Timex datalink). Nowadays, the connectivity has been realized by wireless communication, e.g. Bluetooth or propriety protocols. The current models of smart watches are connected via Bluetooth to a smart phone. Because the watch is located at the wrist, a permanent look on the micro display of the watch is possible. The main application of smart watches is to display messages of services like Facebook, SMS, RSS or incoming calls. One of the most used sensors of a smart watch is an acceleration sensor to support fitness and wellness applications with pedometer functionality. The smart watch idea provides two general concepts. The first concept of smart watches is the autarkical watch. The smart watch provides computational power. Hereby a wireless connection to the internet or phone is only needed for data synchronization purposes. This concept allows sensor data processing directly on the watch. The second concept of smart watches is the idea of a dedicated terminal. Each event, e.g. pressing a button, screen touch event, or sensor readout has to be transmitted to a hub, the smart phone. On the phone, the data will be processed and events will be sent back to the watch wirelessly. Hereby the phone might transmit the vibration event or the screen content. The watch needs no extra computational power which helps saving battery energy. If acceleration data are to be read out permanently (e.g. for activity recognition) the transmission needs significant more energy than an onboard data processing would consume. 2.2 Activity Recognition The integrated acceleration sensor of a smart watch enables the feature of physical activity recognition. Because of the location of the watch, the integrated sensor measures the acceleration force of the wrist. The wrist is not the ideal location for physical activity recognition in comparison to the hip. If the user is shaking the hip, the acceleration data are a good indicator that the whole body is in motion. In [2], [3] is shown that the pattern of a hip movement correlates to the basic activities such as walking, cycling, car driving or jogging. The wrist movements are very complex and are significantly different to the hip movements. Nevertheless Polar could proof by AW200 that easy activity recognition for walking and different states of running is possible on a commercial product [4]. Hereby an acceleration and air pressure sensor were integrated into the watch. The research project eWatch [5] designed an activity monitoring system that uses additional sensors, e.g. light and audio. WristQue by MIT, USA [6] is a research prototype with integrated location sensor, temperature and humidity. The focus of InfoPulse [7] is to act as a handsfree

email reading device, perfect to catch a quick glance without distracting too much from a user's primary task. Texas Instrument introduced the Chronos [8], a very inexpensive smart watch, which is based on the embedded system MSP430 that provides an SDK for individual programs. Newer smart watches are Bluetooth enabled and some are water resistant. These smart watches are designed to be worn 24 hours a day and allow a long shower or a bath in the swimming pool. 2.3 Inactivity Recognition The major difference in activity recognition between a smart phone and a smart watch is the wearing behavior. If the user returns to his home, he will put off the phone. A smart watch can be worn constantly at the wearer's body, no matter if he is doing sports or is at rest or sleeps. Some systems are available that are detecting sleep or resting by wearing a sensor at the wrist. The Actiwatch is an actigraphy based data logger designed for rest activity patterns, quantify physical activity or sleep [9]. Another watch-based sleep recognition solution is the Sleeptracker [10] that detects sleep and allows a wake up at the estimated optimum time. Some other solution for sleep detection exists whereas some other sensor technology is used, e.g. EEG signals. 2.4 Acceleration Sensor The performance of acceleration sensors can be measured in accuracy, quantization, sampling rate, sampling stability, range, noise, and energy consumption [11]. Early types of acceleration sensors were metal balls within a coil environment that generated electric charge or an induction field while it was moved. Other system concepts were piezo crystal based; hereby a weight pressures on a piezo crystal that response to applied mechanical stress by electric charge. Accelerometers can also use strain gauges. Earthquake observatories were using optical levers or mechanical linkages to amplify the small motions. Nowadays, the acceleration sensors are using electronics. The most common sensor for home consumer products, e.g. phones, cameras, cars etc. is a MEMS, micro-electro-mechanical system, which is very cheap and low on power consumption. The usage of the MEMS in smart watches is a tradeoff between performance, energy consumption, size and costs. The MEMS-sensor in itself provides

usually a high frequency sampling rate, e.g. more than 1000 Hertz. Due to the operating system and power requirements, the sampling rate is self-adjusting (Android) or is limited to a lower frequency. [2] was using 75 Hz, [14] suggested 100 Hz, [15] was using 36 Hz, [16] only 20 Hz. It is obvious that a high sampling rate provides more data than a lower rate, a good compromise in sampling rate seems to be at 32 Hz [11]. One quality criteria of an acceleration sensor is the accuracy. We assume that the MEMS sensor show a hysteresis effect to the measured signal. Furthermore, the data shows an offset and a not linear gain. Sensor data sheets and trials show a tremendous offset of each axis. This offset can be around 0.5 m/s2, often the

amplification factor shows an error of 2-3 percent. The offset is usually too high for physical way-time calculations. The following paragraph will discuss what requirements are necessary for activity and inactivity recognition for acceleration sensors.

Fig: Smart Watch with integrated sensor, connectivity and micro matrix display

3. CHALLENGES

For adequate activity recognition, the sensor has to provide reliably sensor information. Because the most important sensor for activity monitoring is the acceleration sensor, it is necessary to know what the key features are. The sensor requirements are distinguished between the application fields of

- Activity monitoring for health and wellness applications
- Inactivity recognition
- Gesture recognition and interaction
- Industrial monitoring concerning health issues of harmful hand or arm vibrations

In [11], significant features for activity monitoring are described but some new features for inactivity are to be worked out in order to detect a fall of a person, and to differentiate states of unconsciousness, sleep or a nap. In case of inactivity, some new features and requirements to the sensor are necessary.

Some smart watches provide an integrated light sensor. The light condition which surrounds a person is an important factor for the estimation of his activity state. Some light sensors are measuring the RGB-light proportions which indicate if artificial light or sunlight is ambient. This sensor provides important information in hospital environment but in every day live it seems to be different. Here, people do a nap while it is light or sleep in front of the T.V. at night by a full light condition. If the user is wearing a shirt with long leaves, the detected light is different to the real light condition. Furthermore, often the geometrical constellation of the integrated light sensor of the smart watch cannot detect the ambient light but only the light which shines directly into the light sensor.

3.1 SECURITY AND PRIVACY ISSUES

There are a few key challenges faces in WT which are power consumption, communication capacity, design constraints, and security and privacy issues. Furthermore, the major challenge in WT which is the security issues is highlighted as it is the focus of this research study. It can be further categorized into three major parts which are security vulnerabilities, attacks and security solutions. The security vulnerabilities in WT can be exploited by an array of possible security and privacy attacks. The security attacks can be further divided according to two main types: passive attack and active attack. Passive attacks try to get the user's password and sensitive information without breaking and affect the system while active attacks contrast with passive attacks, in which try to break and alter the system. When the security vulnerability is exploited, there will be a loss. The loss can be loss in term of Confidentiality, Integrity, Availability or Authenticity. On the other hands, privacy attacks are categorized by user identity and data integrity attacks and time and location-based attacks.

The security solutions can mainly be discussed using two different terms which are authentication and encryption. Authentication can be further divided into two main types which are single-factor authentication and multi-factor authentication. There are several common challenges identified in the wearable technology that will need to be addressed by further researchers in order to improve it which include:

3.1.1 Power consumption [7]-[8]. One of the major challenges is the high power consumption of wearable devices. The battery power of wearable devices can only last for one to two days since most devices use wireless networks, GPS, and other technologies that consume a lot of power. Hence, short battery life and high power consumption of wearable devices will cause people reducing the usage and adoption.

3.1.2 Communication capacity. The communication range is limited, which mean that the covered area range of wireless transmissions is usually limited because of both technological and energy-savings considerations [7]-[8].

3.1.3 Design constraints. Some wearables are designed in bulky size and it does not really make users feel comfortable to it such as "Holter-type" system [6].

3.1.4 Security issues [5]-[8], [27]. Security, privacy is still an unresolved issue in WT. The wearable devices contain a lot of user's data which putting users' security and privacy on the risk. Moreover, it also may consist of sensitive information data such as address, credit card number, and health-related data. Therefore, security issues will be the key challenges for WT to be adopted widely in the market.

3.2 WEARABLETECHNOLOGY (WT) SECURITY VULNERABILITIES

Security and privacy issues could be the major reasons of it. It can lead to the serious breach and loss if the security vulnerability is not handled properly. The loss could be either static assets such as files, documents or dynamic asset like credit card number. At the end, it will cause data and financial loss or even safety issues. Furthermore, user's trustworthiness towards wearable will decrease and discourage people to get their own wearable. For instance, among the top consumer concerns about the IoT, which is defined as devices that connect with each other or to the Internet—28% of respondent's concern about "or someone hacking into the device and doing something malicious" and 26% concern about "not knowing how the information collected by the devices will be used" [28]. This implies that security issues are the major concerns that potentially reduce user acceptance and trustworthiness towards devices. Therefore, it is important to investigate on the security vulnerability on the devices for user protection. Basically, wearable devices can be exploited from different attack surfaces shown in the Figure 2.

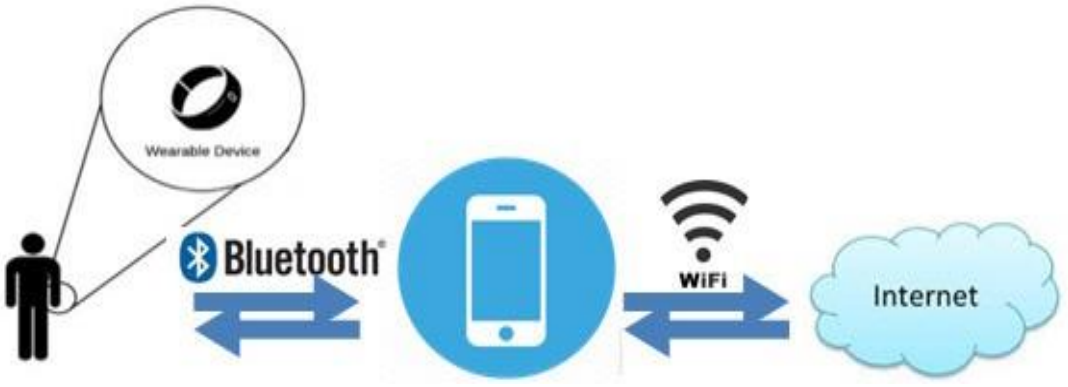


Figure 2: Generic Data Acquisition Architecture in Wearable Technology

The most common security vulnerabilities that can be found in wearable devices

from the attack surfaces shown in Figure 2. The factors such as i) Unsure transmission of data Bluetooth for local device storage; ii) Software communication to the Cloud via a cellular or Wi-Fi network; iii) Insecure data storage on Cloud; (iv) Lack of authentication and authorization and (v) Lack of physical security controls that contributes to the security attacks are discussed below.

(i) Unsecure transmission of data via Bluetooth for local device storage

Wearable devices rely on Bluetooth (see Figure 2) to do transmission of data collected from embedded wearable sensors to integration devices such as smartphone as currently it cannot communicate directly to the Internet. As a result, attacker can exploit the bug in the device to extract data stored locally, such as health-related records by using wearable as an access point. For example, an attacker can simply make use of sniffers [29] to steal unauthorized data by detecting the broadcast signals while a wearable device communicated over Bluetooth. As a consequence, there will be a loss on either in term of monetary, the safety or even life of the people.

(ii) Software communication to the Cloud via a cellular or Wi-Fi network

This is more vulnerable compared with the transmission of data via Bluetooth for local device storage discussed previously. This is because more personal sensitive data can be stolen as data transmitted from the local storage in the smartphone to the Cloud application (see Figure 2.3) is typically combined with personally identifiable data such as name, email, telephone number, and location to ensure that the data is being sent to a proper account. The attacks that can take place by exploiting this security vulnerability include man-in-the-middle and redirection attacks, which could cause data to be sent to the wrong server. Hence, the potential loss of private data is high and privacy and safety issues might arise as well if the wearer is identified.

(iii) Insecure data storage on Cloud

Cloud refers to a public or semi-public space on transmission lines that exists between the endpoints of a transmission [30]. Cloud storage provides better file accessibility, which the file stored in the Cloud can be accessed at any time from any place as long as you have the internet access. This could be the most vulnerable area in the wearable world due to the amount of Personally Identifiable Information (PII) that is available [29]. The data synchronized to Cloud could be posed by a number of risks, including distributed denial of service (DDoS) attacks, SQL injection, or back door attacks. The attacks on the Cloud are typically carried out by highly skilled cyber criminals. For instance, a cybercriminal gang was reported to steal up to $1 Billion by impersonating bank employees through the use of malware [31].

(iv) Lack of authentication and authorization

Most of the wearable devices are often do not come with a built-in security mechanism such as user authentication or PIN system protection features and they usually store data locally without encryption [32]. Besides that, wearable devices require higher communications [33] security regarding encryption, data integrity, confidentiality and other security services since it relies on the uncontrolled wireless network either Bluetooth or Wi-Fi connection to transfer data. However, it is difficult to apply with higher security measurements due to its' small size and limited bandwidth and finally, result in easier to be attacked. For instance, HP study [34] revealed that 30 percent of the tested smartwatches were vulnerable to account harvesting, which is an attack that gain access to the device and data by looking for weak password policy, lack out account lockout and user enumeration.

(v) Lack of physical security controls

The other security vulnerability for wearable devices is the potential for the loss of the device itself. The small and tiny size of wearable device such as fitness band is most likely to be misplaced or lost. The lost or stolen devices will pose a risk on the exposure of the personal data information complies with its confidentiality, integrity and availability if it has fallen into the wrong hands. Furthermore, most of the

wearable devices are often do not come with a built-in security mechanism such as user authentication or PIN system protection features and they usually store data locally without encryption [32]. For instance, Apple Watch and Google's Android Wear platform do not have any security measures in place protecting their pricey wearables from loss or theft [35].

Next an analysis on various privacy challenges and attacks on wearable devices aspect is given.

3.3 WT PRIVACY CHALLENGES& ATTACKS

Apadmi [36] conducted a survey that asked their respondents about "Do you feel wearable technology poses a threat to your privacy?" The results show that 42 percent said yes and another 40 percent were doing not knows and remaining 18 percent replied no. This implies that people worry about wearable technology that exposes the potential privacy risks of devices that can record and capture personal private data. The privacy issue is one of the major challenges yet to be solved in wearable computing. Not only because wearable can sense, capture and store sensitive information about the users, and his surrounding but it also able to do it continuously and discreetly [37]. The privacy attacks that poses in WT can be categorized into user-based privacy and data-based privacy time-based privacy and location-based privacy.

3.3.1 User Identity and Data Privacy

Embedded sensors such as cameras and microphones, capture data about the individual and also the surroundings, often without their consent. These data often personal, confidential and sensitive, which invade users' privacy and poses privacy challenges such as surveillance. For example, Glass can be easily hacked as it has no strong authentication implementation [38]. The attacker can take full control the Glass and monitor everything the owner doing with the camera and microphone. Besides that, security researcher from Stanford University and Israel's defense research group Rafael found that gyroscope, a sensor that used to measure angular

velocity can even eavesdrop on the conversation as well [39] despite of using microphones. They found that the MEMS (microelectromechanical systems) gyroscope is sensitive enough to recognize the sound and can pick up some sound waves and turn them into crude microphones. Therefore, it proved that the conversation could easily eavesdrop without user consent since iOS and Android do not require special permissions from users to access gyroscope.

3.3.2 Time and location-based privacy

GPS embedded inside wearable able to track a person's location at a specific time. It brings greater benefits for people to do navigation, but it also poses greater risks as well. It raises serious issues on the user's privacy, if the location of the people can be tracked. For instance, Symantec
[40] revealed that wearable can also do location tracking, although there is no GPS sensor built- in. This happened because the data exchange process between phone and fitness band could potentially broadcast the location information. They performed an experiment to illustrate that by using a portable scanner, a unique hardware address that each fitness tracker emits when syncing to the user's phone via Bluetooth could be picked up. In addition, some collected data that are shared consist some information that an individual did not intend to share. For example, IMU sensor data such as accelerometers and gyroscopes shared with caregivers may reveal some sensitive medical conditions, such as seizures, that one may wish to keep private [41].

3.4 AUTHENTICATION PROBLEMS IN WEARABLE DEVICES

Based on the research findings done in the previous section, all of the wearable devices that have been analyzed are lack of authentication. Why this happened? It is supposed that wearable devices should be protected with secure authentication mechanism as it contains a huge amount of sensitive information. The reason could be wearable devices are typically lacking a keyboard or even a touch screen. Therefore, it is a challenge to implement password or PIN-based authentication. On the other hand, due to its' small device size so their processing power and bandwidth

are further limited and this makes it difficult to apply with higher security measurements. In addition, Symantec threat researcher Candid Wueest [42] recently revealed that the danger of wearable devices at this point is that developers are not prioritizing security and privacy. Hence, wearable device resulted to be less security compared with other devices without a strong authentication scheme in place.

However, as wearables making more and more use of user personal data - from fitness stats to health records, security should be put into a high priority. Unlike mobile devices, wearable devices are potentially always-on and always gathering data. In this way, it not only bring the uniqueness on the WT with the added dimension, but at the same time it is also open to more threats to user sensitive information and activities at any time anywhere without user's consent. Security measures are not only important for protecting personal data, but are critical as smartwatch are introduced to the workplace and connected to corporate networks as well. Therefore, it is very important to be paid extra attention to maintaining confidentiality, integrity and availability (CIA triad) for wearable displays.

Could you imagine how serious security breaches could happen if the wearable device that has poor user authentication are being attacked? Take for an example, a smartwatch can be used to make online transactions. However, the smartwatch is lack of user authentication with unencrypted data. If in the case that the smartwatch was stolen or lost, someone with bad intention could likely access all of your personal data or even modify transaction content or insert additional transactions with the credit card information. In the consequences, it not only causes loss of security, but it violates the confidentiality and integrity policy and results in loss of money as well. As a result, the users' security and privacy are in risks without a strong authentication scheme in place. It is important to be paid more attention on wearable authentication in order to protect users' data from malicious, unauthorized access since a massive quantity of private data to be collected from the wearables. Security measures are not only important for protecting personal data, but are critical as smartwatch are introduced to the workplace and connected to corporate networks as well. Therefore, authentication issue must be tackled definitely to ensure the security of wearable is preserved thoroughly.

4. ACTIVITY UNIT

The acceleration sensor is the most important sensor for activity recognition. The sensor detects three-dimensional acceleration forces that are typically measured in a Cartesian coordinate system. The axes are x, y, and z and they are orthogonal to each other. If the sensor is motionless, only the gravity g is affecting the sensor data. The acceleration values are in a specific relation to reach other, no matter which orientation the sensor shows. If the variables x, y, z are the sensor values in g (9.81 m/s2), the motionless sensor shows:

$$\mathbf{gl} = \sum (\mathbf{x}^2 + \mathbf{y}^2 + \mathbf{z}^2)^{\mathbf{0,5}}$$

If the sensor will be accelerated, the right term of this equation can't be used to determine the average acceleration. One reason is that the gravity g is not constant over the world (pole and equator have different gravity influence). Furthermore, the sum of the squared acceleration values is not corresponding to the acceleration, if one average acceleration value of one axis is negative. In addition, the sensor offset of each axis is affecting the equation so a possible error occurs.

We developed the algorithm of a new feature. This feature defines the mean acceleration of the sensor in the three dimensional space per second. The feature is called "activity unit (AU)" and describes the acceleration force. This feature is defined in m/s3; hereby meter and seconds are elements of the international system of units [17].

The gravity constantly influences every material on earth, that's why the activity unit considers only the additional acceleration forces. Furthermore we introduce the simplification that the acceleration force of interest is not vectored. In contrast to the physical jerk that describes the rate of change of acceleration [18], the AU is a scalar magnitude, normalized to the time interval of one second. The jerk represents only the derivative of acceleration with respect to time. The jerk does not consider if the present acceleration is zero, low or high; only the change of acceleration is regarded.

In contrast to the jerk, the AU is respecting the present level of acceleration while the change of acceleration occurs. To give an example, the jerk would be the same (zero) in the case that the sensor is motionless or in the case that the sensor is accelerated with a constant value. The AU is zero in case of the motionless sensor too, but in the case of the accelerated sensor - it is different. Hereby the AU represents the difference of the current acceleration to the average acceleration condition.

This leads to the following definition of the activity unit:

$$AU = \frac{1}{N}\sum_{n=1}^{N}\left((x_n - x_{mean})^2 + (y_n - y_{mean})^2 + (z_n - z_{mean})^2\right)^{\frac{1}{2}}$$

Here the recorded variables are –

- x, y, z are the sensor values in m/s^2
- x_n, y_n, z_n are the values received by the sensor in each readout cycle n.
- x_{mean}, y_{mean}, z_{mean} are the mean of the sensor values for each axis.

For a motionless sensor, only the gravity forces are affecting the measured values and these forces are constant. Because of the moving average, the algorithm will detect no force after a while, zero AU. If a translation occurs, the acceleration will be detected and measured. If the sensor will be rotated, the sensor detects different gravity force for each axis.

If the sensor is motionless after the turn again, the algorithm should detect no acceleration force after a defined time interval. This is performed by the implementation of the moving average algorithm. We assume that a sensor will be read out with 32 Hertz. If the sensor is turned by 90 degree, we require that the acceleration force of one axis, that had no gravity influence before, will adapt to the gravity force of one tau (63.2 percent) within 2 seconds. The two second period will provide 64 datatriples which is a usable window frame length for activity recognition [11]. The time requirement leads to the average factor of a=0.95 by the following average algorithm, the given average is analogue to y_{mean} and z_{mean}:

$$x_{mean}(n) = X_{mean}(n-1) \cdot a + X(n) \cdot (1-a)$$

Here the recorded variables are –

- X(n)= current sensor value
- a= absolute term
- x_{mean}(n)= calculated average
- n=readout cycle
- x_{mean}(n-1) = calculated average for (n-1)

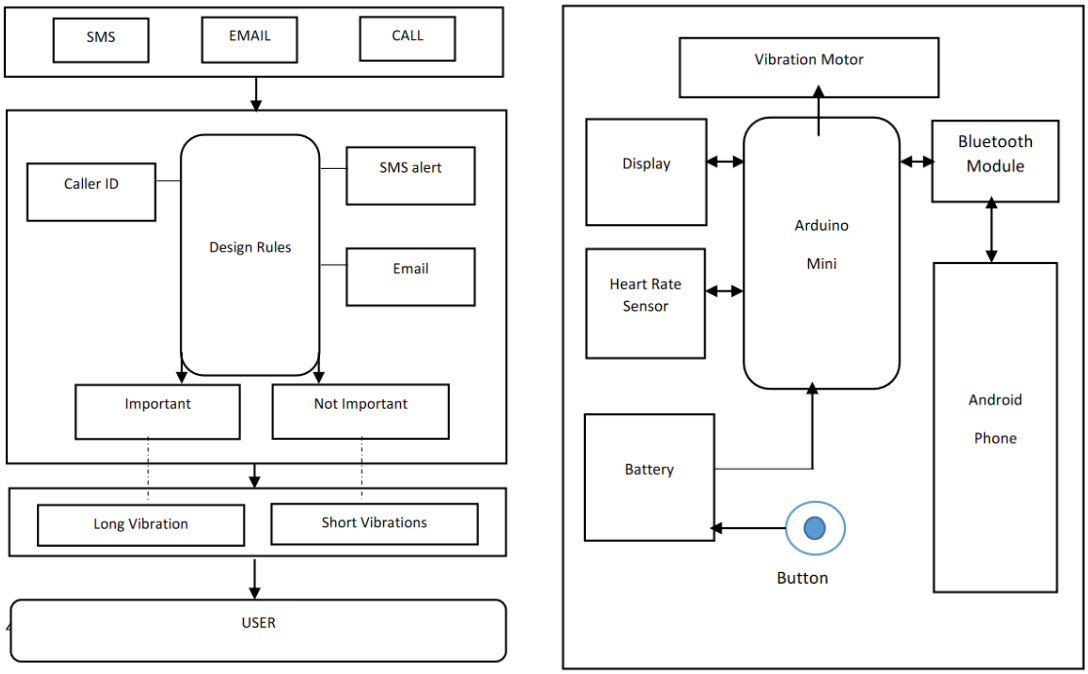
Hereby x(n) is the current sensor value, the factor a is an absolute term and x_{mean}(n) is the calculated average. n describes the readout cycle. x_{mean}(n-1) is the calculated average of the readout cycle n-1. If parameter "a" would be set on a = 0, the algorithm for AU is directly matching the jerk.

The sensor values of the x, y, and z-axis are represented by the SIunit m/s^2, so 1 AU is 1 m/s^3 (m/s^2 per second).

5. SYSTEM ARCHITECTURE

The generic principle is that the smart watch interaction should be natural, simple and easy to use, which translates to the following design principles:

- Minimum number of menus on screen.
- Total number of steps required to execute the app's function should be less than steps required on smartphone.
- Visually appealing. The user interface should be simple to use and visually appealing.



5.1 BLOCK DIAGRAM

ARDUINO MINI - The microcontroller used in this project is ATmega 328p mini chip. It has 2KB RAM, 32KB EPROM, 64KB FLASH.

DISPLAY – Normal LCD display of 240 x 240 pixels.

CAMERA MODULE – 0.3 MP vga camera module has used in the prototype.

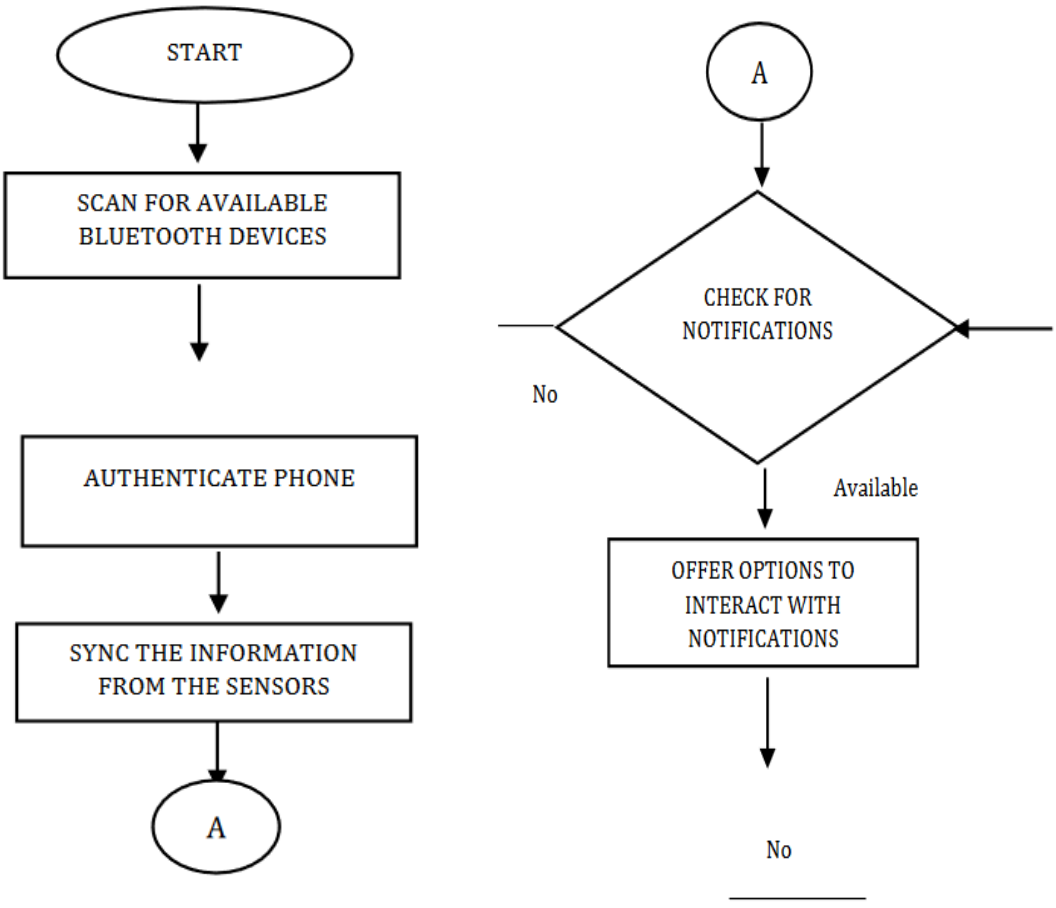
BLUETOOTH MODULE –A Bluetooth Serial Transceiver HC06 is used. The HC06 Bluetooth module is the most widely used module with an Arduino project. It has low

5.2 ALGORITHM AND SOFTWARE

The open source Arduino libraries are used for the development of the basic source code. The main android app is developed in Android Java Platform.

- Powering on the Smart watch switches on the components connected to it.
- The user needs to synchronize his Smart Phone with the Watch via Bluetooth for the notifications on the fly functionality.
- Upon successful pairing with the Bluetooth module and installation of the Android app notifications can be displayed on the watch and can be interpreted accordingly.
- The heart rate sensor monitors the users heart rate and shows it on the screen when that particular menu is selected.

5.3 FLOW CHART

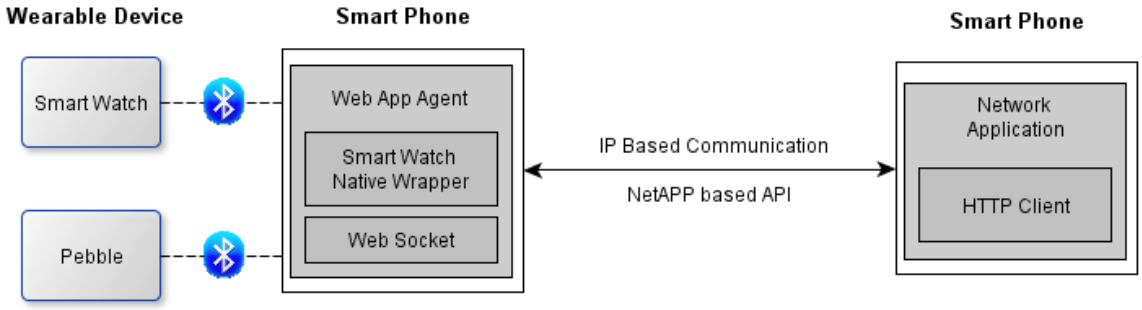


In the above section we can see the flow chart for the required process using the wearable device and measurement.

- First it will it will search for available Bluetooth device.
- Then authenticate with android device using QR code generation
- The cloud will provide the QR code. For any security issue new QR code will be generated.
- After authentication it will check for Notification.
- If any notification is available, it will show options for performing operations.

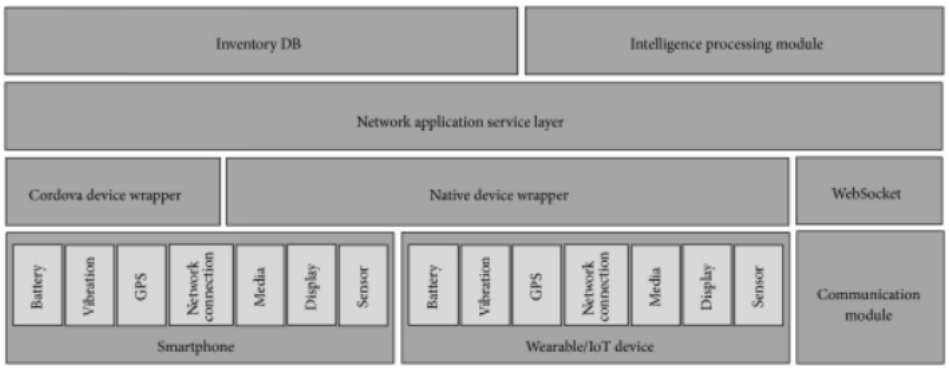
5.4 Connection between two Smart Phones using API and Bluetooth

The next process will be connecting two smart device using Bluetooth and API based application.



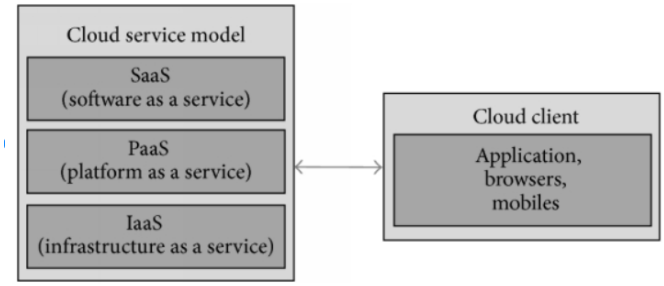
One smartphone will be running HTTP Client and the another, which is basically the smart watch will be running the native wrapper and the WebSocket for communication.

Node Architecture



This will be the node architecture for the API interface

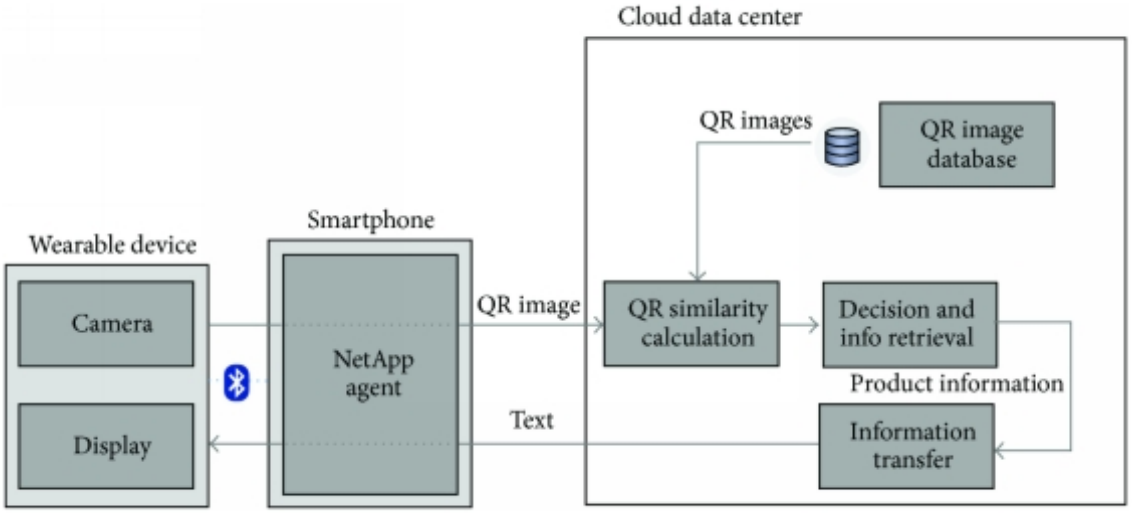
Cloud Server Client Model



The cloud server model which will provide the available services like SaaS, PaaS and IaaS

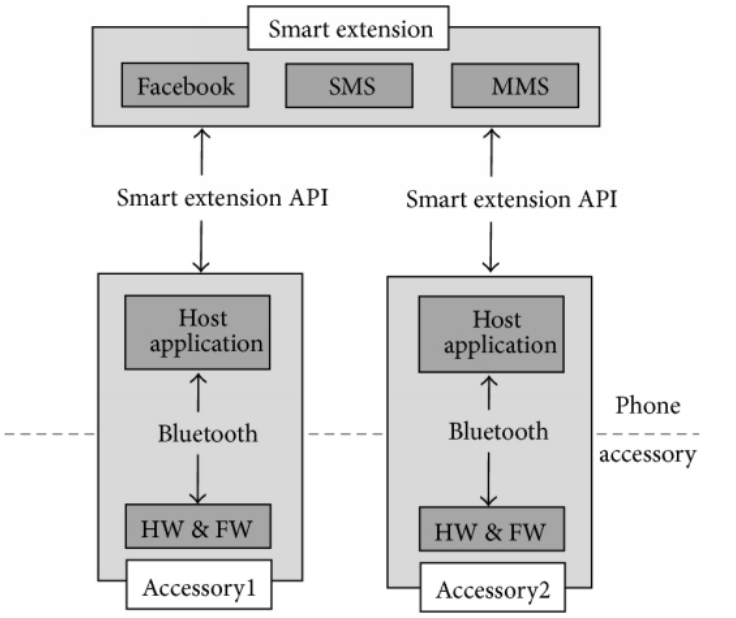
The model will be precepted as depicted in the above picture.

5.5 Interface with Cloud Architecture with Wearable Device



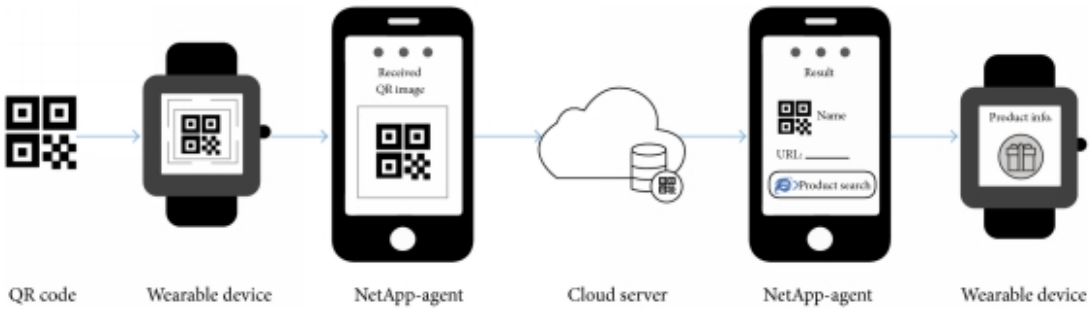
The interface with the cloud server model will be implemented as the picture has depicted. The device will be self-sustained with a camera, and Bluetooth functionality. Which will connect to the netapp agent. The netapp agent will scan the generated QR code send a request to the server for which will be authenticated using the implemented database. After authentication the system will send a message in text format which will display on the user mobile screen. After successful authentication user will be able to use the functionalities of the Wearable device.

5.6 Software Application and API architecture



The software or API architecture which will be installed in both of the devices will be like the above picture. In the outer layer the applications will be available. Using smart extension API the host application will connect to the application. In the inner level there will be hardware which will provide Bluetooth connection.

Other accessories attached to the device can be accessible through the wearable device after successful authentication.



Security Patch: QR code Generation for Authentication

The security authentication process is depicted visually in the above picture, how it will generate and Authenticated the system. The authentication process is usually between two devices, like a android phone and a smart watch.

6. SYSTEM ANALYSIS

6.1 introduction:

The proposed system and analysis are depicted into the following section. The Main goal of this project is to provide utility and security to the user with ease. For providing security and reliability we are supposed to make the back-end process more secure and effective. So here we will discuss the implementation of secure API (application program interface) into the system which will provide secure channel data transmission and monitoring for the system.

Though the system is built on basis of a small device which is wearable on human body. But the data which is collected by the system is crucial and very important that to be secured and stored into a correct place. Not only that, while passing through the channel it should be a secure channel and interruption free. The data must not be accessible from the database or any other network which is not authenticated from the main system.

Here we will discuss several methods what we have planned and designed to implement into the system. We will also discuss the functional and non-functional requirements for the system.

Due to the rapid growth of network infrastructure and sensor, the age of the IoT (internet of things) that can be implemented into the smart car, smart home, smart building, and smart city is coming. IoT is a very useful ecosystem that provides various services (e.g., amazon echo); however, at the same time, risk can be huge too. Collecting information to help people could lead serious information leakage, and if IoT is combined with critical control system (e.g., train control system), security attack would cause loss of lives. Furthermore, research on IoT security requirements is insufficient now. Therefore, this paper focuses on IoT security, and its requirements. First, we propose basic security requirements of IoT by analyzing three basic characteristics (i.e., heterogeneity, resource constraint, dynamic environment). Then, we suggest six key elements of IoT (i.e., IoT network, cloud, user, attacker, service, platform) and analyze their security issues for overall security requirements. In addition, we evaluate several IoT security requirement researches.

6.2 Requirement Analysis:

The era of IoT is opening with rapid growth of network infrastructure and sensor. There is no strict definition of IoT yet; however, usually IoT is described as collaborative ecosystem of context-aware, intelligent and automated device connected network for specific purpose.

Gartner, Cisco, and IDC (International Data Corporation) evaluate IoT as a promising technology of the future, and most of the corporation also think that IoT will become a key of their next-generation growth power. According to IDC, the IoT market scale is expected to grow from $655.8 billion in 2014 to $1.7 trillion in 2020 with a CAGR (compound annual growth rate) of 16.9%.

Accordingly, a lot of corporations in the world are developing IoT-related devices, services, and technologies to dominate the market in advance. However, they do not consider security as a functional requirement so that security concerns have come down in priority list. Therefore, corporations are reluctant to apply security sufficiently to the devices and services. For example, when TCP/IP was invented, the importance of security was not widely known, and most people did not know what attackers can do with security vulnerability. Because of these reasons, TCP/IP was not designed to secure enough. As a result of the insecure design, attackers are able to use a lot of vulnerability which causes security attack and huge monetary damage.

However, we currently know why security is critical, and what attackers can exactly do with security vulnerability. Thus, when IoT-related standards are being developed, we must discuss IoT security to avoid repeating the mistakes of the past. In order to overcome the problems, this paper proposes security requirements based on three characteristics of IoT and six key elements in IoT. In addition, evaluation is conducted with several researches.

6.2.1 Security Requirement Analysis:

This section analyzes security requirements based on 3 typical IoT characteristics that have been researched in other researches. These security requirements are commonly applied in IoT security. Therefore, it is important to understand and take advantage of it to design security mechanisms in IoT environment.

A. *Heterogeneity*

In IoT, heterogeneity means diversity of hardware performances (e.g. CPU computation, memory footprint), protocols, platforms, policies, etc. The biggest problem of heterogeneity is absence of common security service. heterogeneity weakens interoperability and causes extra fees about performance and money to interpret each other. Besides, making security-related policies and updates are more complex. In order to solve these problems, we can use some technologies (e.g., meta data registry (MDR), middleware); however, it is not a fundamental solution. For providing common security service, unified IoT security standard has to be established. Then, developer who are related to IoT development should follow standards strictly. Recently, standards organizations (e.g., ITU-T, ETSI, ISO/IEC JTC 1) develop some standards for the security in the IoT.

B. *Resource Constraint*

Most IoT devices are lacking performance and battery capacity. Therefore, legacy security services, such as TLS (transport layer security), AES (advanced encryption standard), cannot be applied to IoT devices directly. Therefore, these services or algorithms should be designed to be lightweight and straightforward to increase efficiency of CPU, memory and battery. In addition, scalability has to be considered.

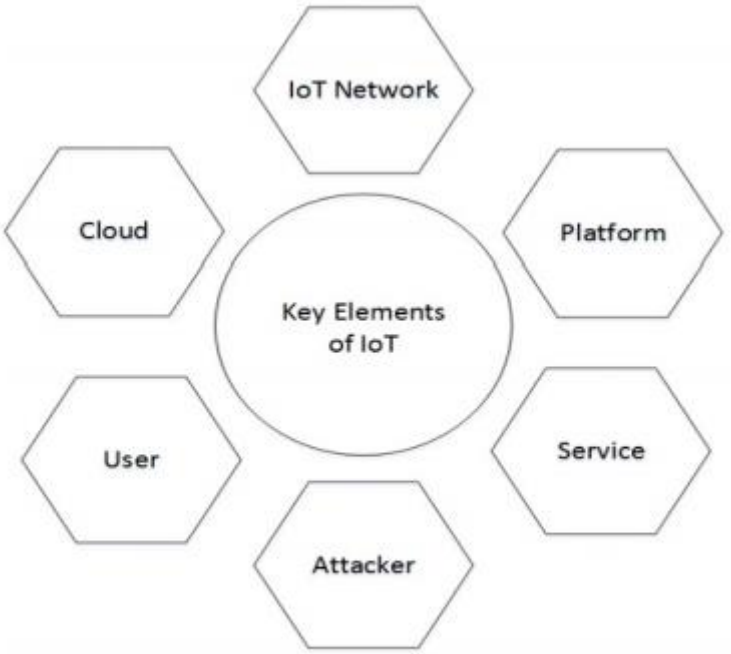
Apart from lack of performance of device and network bandwidth also low, so that multicast is more effective than unicast. Note that, CoAP (constrained application protocol) supports multicast in RFC 7252 officially, but MQTT (message queue telemetry transport) does not.

C. Dynamic Environment

Due to mobility and bad connections, IoT has a dynamic network topology. In very demanding cases (e.g., smart city), numerous devices may send a large number of requests. Hence, not only flexibility, but also scalability is required in IoT communication protocols. Cisco forecasts that 50 billion devices will be created by 2020, and after that, more and more devices will be made. Consequently, flexibility and scalability will be key security requirements of IoT.

6.3 Security Issues and requirements for IoT environments:

The following Fig. shows six key elements of IoT (i.e., IoT network, cloud, user, attacker, service, and platform). We consider reviewing security requirements from the elements to be the most effective way. A more detailed description is in the following subsections.



Six Key Elements of IoT

A. IoT Network

IoT network is a specialized form of conventional network. It has three features as described in Section 3. In IoT network, there are many Things (e.g., gateways, sensors), and they may communicate using lightweight communication protocols, such as MQTT and CoAP based on IEEE 802.15.4.

The most important fact is that IoT network is basically not different from conventional networks. Therefore, most existing problems (e.g., fragmentation, security attacks) could happen in IoT network. In this subsection, we focus on the following issues: privacy, security in multicasting and bootstrapping.

Privacy. IoT is becoming more and more closer to human life like ubiquitous.

It can be used anywhere, anytime with anything. People will be monitored by CCTV in everywhere, and sensors will send any sensed information to the network. Additionally, types of information will be diversified and amount of information will be gradually increased because of big data. In this situation, ensuring privacy is critical. Thus, from now on, we need to research security for privacy under understanding of characteristics and security requirements of IoT. Particularly, encryption and authentication is necessary to be made use of bitwise operation rather than mathematical algorithm like ECC (elliptic curve cryptography) in order to make lightweight security service. Finally, privacy does not always have to be protected. If user is in emergency situation (e.g., car accident), privacy information is required to provide to doctor or close people.

Security in multicasting. IoT environment which is resource-constrained can be used multicasting more effective than conventional network. When using multicasting, multicast group should be created with authenticated users, and secret key that is shared with group members is required to keep securely.

Security in bootstrapping. Bootstrapping is a process that sends data (e.g., configuration information, key material) to participate secure IoT network. After that, authentication and authorization are performed. D. Garcia-Carrillo et al. proposes a flexible, scalable and lightweight bootstrapping for demanding environment like smart city using AAA (authentication, authorization, accounting), EAP (extensible authentication protocol) and CoAP. Likewise, bootstrapping is required to designed in a flexible, scalable and lightweight manner. These are essential characteristics in the IoT environment.

Finally, it cannot be guaranteed to prevent security attack with only network security. So, devices should provide built-in security. Embedded security can support dynamic prevention, detection, diagnosis and isolation.

B. Cloud

Usually, IoT devices use cloud because they cannot save the data in their low memory capacity. In some cases, sensitive data (e.g., home CCTV video, personal location, health information) can be used for rescue people. However, if cloud out of order for some reasons, IoT devices cannot save the data. Then critical data that will be used for

rescue can be missing. As a result, rescue service that require the data may be stopped. Therefore, in this case, availability is highly necessary, so that device should have back-up cloud to be replaced with original cloud.

There are a lot of data sent from many devices in cloud. To protect the data from unauthorized user, cloud should use proper access control (i.e., authentication, authorization), encryption, data anonymity, etc. In addition, the data may not be fully needed to be encrypted based on the importance of data. In this case, security-level based encryption is required for efficiency. It is the similar concept with contextual integrity in.

C. User

User is the most vulnerable element in IoT security. Even if information system is implemented securely, if a user, especially system engineer, is careless to manage, any security system will be useless. For example, in ID-password authentication model, if a user makes the password with a simple and guessable passphrase, attackers could crack the password easily using brute force attack or dictionary attack which is well known security attack. That is, the user has to follow strictly the security rules, and the user needs to be educated about social engineering.

D. Attacker

Security service can be compromised by attacker Although a user follows security rule. Due to IoT devices are connected to network, it can be victim anytime. most of IoT devices cannot apply strong security service because of its constrained resources. Besides, current IoT security services have not been fully validated. For these reasons, IoT is easy target to attack so that security attack will be increased and diversified. Thus, in this subsection, we analyze security requirements against security threats.

In IoT environment, security threats can be categorized into non-physical threats and physical threats. Non-physical threat can be described as threats which uses network, and physical threats also can be described as all threat except for nonphysical threat. In contrast to conventional PC environment, IoT devices may not be placed on secure environment. Consequently, IoT devices could be easily destructed by nature and

people who has malicious intention. If an attacker is able to access the device in insecure environment, the attacker can do side channel attack, or can analyze vulnerability of firmware or platform. Therefore, security mechanisms for the IoT device in harsh environment is required to be considered when device manufacturer implement the device.

Most of non-physical threats (e.g., buffer overflow, sniffing, man in the middle attack, spoofing) are attacks on confidentiality, integrity and availability. Due to heterogeneity, IoT security is too complex to prevent the types of attacks. Additionally, IoT device has constrained resource to adopt the strong security services (e.g., intrusion detection system, antivirus) for protecting system from security threats. For these reasons, IoT device is more vulnerable to security attacks (e.g., DDoS) than conventional PC environment. Even simple DoS (Denial of Service) can stop operation of device if its performance is not enough. Gateway that performance is relatively sufficient compared to IoT device could adopt IDS or anti-virus on behalf of the device.

Attackers can attack on IoT system using the vulnerability of firmware, platform or communication protocol, etc. The security vulnerability is made by system designers and developers' inexperience or mistake. Each vulnerability has different impacts on system. Some of vulnerability is sufficient to break security system, or can steal root permission. Therefore, vulnerability must be minimized by using secure coding, static analysis, dynamic analysis, reverse engineering, etc.

In critical control field, control permission never be lost and changed by attacker, and operation should be supported continuously. Therefore, control system must need a fault tolerance and back-up devices. Back-up device can be replaced when main device is compromised or stopped.

E. Service

In this subsection, we analyze security issues (i.e., trust, access control, middleware, storage) as illustrated in Fig. Before we describe the security requirements based on security issues, each element of the scenario in Fig. will be explained. To take advantage of a service, the user needs to trust the server, and the server needs to provide

privacy to the user. If the user decides the server is trustworthy, the user will use service provided by the server or group of devices with smart phone, smart watch, or some kind of network devices. After that, the devices have to progress bootstrapping and access control (i.e., authentication and authorization). Thereby, devices obtain trust from server. Especially, automated, intelligent and context-aware devices in real IoT environment might be operated by itself without human intervention. Finally, the attacker can compromise the server for malicious intentions (e.g., collecting personal information).

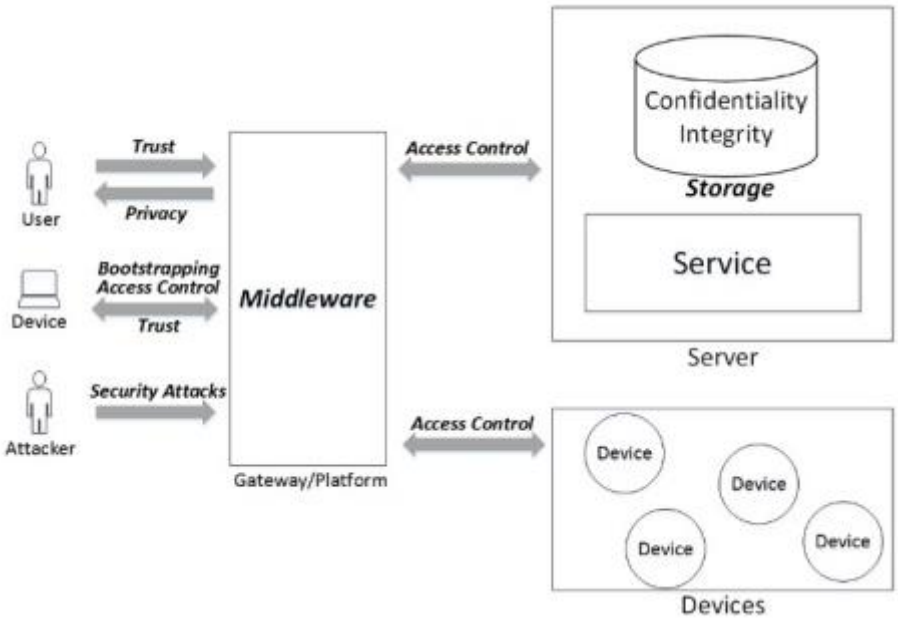


Fig: Security Requirements among User, Device, Attacker, and Service

Trust is complex concept. In spite of its importance, there is no clear definition. Due to unclear definition of trust, it is hard to estimate and evaluate . Therefore, we have to define trust clearly, and need to make the method for estimate and evaluate of trust in order to the IoT security. In fact, 'Trust' was used with two different meaning in Fig. In case of the user, trust means belief. However, in other case of the device, trust means not only belief but also interoperability that any kinds of devices can be communicated with each device.

In IoT environment, middleware can be used for platform that support interoperability, and it can provide security for devices and data. Therefore, when we design middleware architecture, security, privacy, and use of multi communication medium should be considered.

As mentioned previously, access control is security service including authentication and authorization for ensuring security and privacy. Due to dynamic environment and resource constraint of IoT, access control should be flexible and scalable. Furthermore, access control needs to handle various types of access control policies. In addition, identification method to authorize needs to be strict, and the process for managing authority should be easy and simple. As a result of the access control, certificate will be issued. There is a problem in the number of certificates. It could be created too many owing to scale of IoT (e.g., smart city). So, in this case, delegation of authentication and authorization will be necessary.

The last is storage issue. Confidentiality and integrity of data in storage should be supported as the cloud environment.

F. Platform

All Seen, oneM2M, OIC (open interconnect consortium) and other standards organizations have been established IoT platform standards. Open IoT platform (e.g., Mobius, OneM2M, AllJoyn, COMUS) provides multiple functions (e.g., distributed cooperation, execution control, interoperability between heterogeneous devices to share data). They are focusing on the functionality of platform mainly, however, security is considered only in common services (e.g., encryption, access control through authentication and authorization, signature). At this time, as mentioned earlier, because it is necessary to consider the performance of various IoT devices, all of security services should be lightweight. In addition, according to the security level of device, security service should be supported optionally in order to overcome heterogeneity of device performance. Furthermore, for authentication and authorization, method for understanding many different security policies used in diverse devices and domain is required.

Finally, as mentioned in Subsection, attackers can seize device, and analyze vulnerability of the platform. Therefore, platform needs to minimize its vulnerability, and should be verified itself using the trusted platform module (TPM).

6.4 Sensor Requirements for Activity Recognition on Smart Watches

The new generation of watches is smart. Smart watches are connected to the internet and provide sensor functionality that allows an enhanced human-computer-interaction. Smart watches provide a gesture interaction and a permanent monitoring of physical activities. In comparison to other electronic home consumer devices with integrated sensors, Smart watches provide monitoring data for 24h per day, many watches are water resistant and can be worn constantly. The integrated sensors are varying in performance and are not intended to distinguish between different states of activity and inactivity. This paper reports on identified requirements on sensors of smart watches for detection of activity, inactivity as well as sleep detection. Hereby a new measurement quantity is introduced and applications of heart beat detection or wearing situation are presented.

6.4.1 Functional requirements:

6.4.1.1 Product Perspective:

The ongoing trend of not wearing a watch because of an available phone with clock functionality seems to be stopped. The origin functionality of watches, just to display the time, is tremendously enhanced by the new generation of smart watches.

6.4.1.2 Product Features:

Smart watches are designed to display various messages such as SMS, RSS-feeds, Emails, or Facebook messages. For interaction purposes, smart watches provide additional sensor functionality that might be used for third party application providers. Hereby mainly sport or wellness applications are in focus of the manufactures. Because of the smart watch´s sensor functionality, the device enables new and innovative applications that expand the original application field. The smart watches allow a permanent monitoring of the activities of the user. The acquired information on the physical activity of a person can be used for various purposes, such as wellness, safety, various psychological identifier (degree of attention), or detection of micro-activities in an overall state of inactivity.

6.4.1.3 User Requirements:

In the era of 90's watches had been designed to be connected to modern computers by integrating light sensors in. After the invention of modern wireless technologies connectivity for smart devices has been redesigned [1].

Smart watches are wearable in hands and is placed in a particular place to get stable data. Consisting with screen a smart watch is designed to capable of displaying communication services like SMS, Facebook Feed, WhatsApp and even incoming calls. Some watches are enabled to control music and attached with camera module to capture details in stealth.

Smart watches are classified into two separate genres based on the implementation and design - **The first type** is known as autarkical watch. The type which comes out with a built-in computational power and dedicated wireless connection for synchronization over the internet. Thus, the processing unit directly process the data with the watch itself. Which undoubtedly demands more power for running [2].

The Second type consist of a dedicated terminal for event listening and transmitting the records to a hub like a smart-phone device. The heavy work of processing data and transmit back the events to the smart watch. In this way the smart-watch is able to save battery and no extra requirement for computational power.

4.3.1.4 Assumption and Dependencies:

The smart watch idea provides two general concepts. The first concept of smart watches is the autarkical watch. The smart watch provides computational power. Hereby a wireless connection to the internet or phone is only needed for data synchronization purposes. This concept allows sensor data processing directly on the watch.

The second concept of smart watches is the idea of a dedicated terminal. Each event, e.g. pressing a button, screen touch event, or sensor readout has to be transmitted to a hub, the smart phone. On the phone, the data will be processed and events will be sent back to the watch wirelessly. Hereby the phone might transmit the vibration event or the screen content. The watch needs no extra computational power which helps saving battery energy. If acceleration data are to be read out permanently (e.g. for activity recognition) the transmission needs significant more energy than an onboard data processing would consume.

6.4.2 Non-functional requirements:

6.4.2.1 Product requirements:

6.4.2.1.1 Efficiency:

In terms of efficiency smart watch technology is the most modern and effective methodology to enhance the lifestyle of day to day human life. It is capable to provide most advanced functionality to a user to apply the features in their life. The security and features of smart watches are designed to be efficient and user friendly. It opens up a whole new area of smart devices which will change the engineering and market to some extent.

6.4.2.1.2 Reliability:

Wearable devices are reliable in terms of health and social connections. But it's under question that the data which is being transferred and stored are secured or not. The topic of our project is to implement security into the existing scenario which will be discussed in detail in the following sections.

6.4.2.1.3 Portability:

Wearable devices are extremely portable as it is designed to carry by a human / user almost 24 x 7 every day. Mostly wearable devices are purchased on the basis of the key feature of monitoring health conditions and data collection. Therefore, the device must be portable without any ease. In some cases, the devices are made with high quality material to make it light weight and comfortable to the user. Especially, when it is a wearable device like smart contact lenses. Which may sound futuristic, but it actually exists.

6.4.2.1.4 Usability:

In terms of usability the smart wearable devices are gem packed with features which is not applicable to other devices. Mostly modern people are reliable on smart phones for various functions, taking notes and socializing. But regarding health monitoring smart phones are not

effective because it’s heavy enough to carry on our skin / body all the time. Sometimes the smart watches are given ornamental features to make it look more stylish and similarly useful for smart applications.

6.4.2.2 Organizational Requirements:

6.4.2.2.1 Implementation Requirements (in terms of deployment):

Activity Recognition

The wrist movements are very complex and are significantly different to the hip movements. Nevertheless Polar could proof by AW200 that easy activity recognition for walking and different states of running is possible on a commercial product. Hereby an acceleration and air pressure sensor were integrated into the watch. The research project eWatch designed an activity monitoring system that uses additional sensors, e.g. light and audio. WristQue by MIT, USA is a research prototype with integrated location sensor, temperature and humidity. The focus of InfoPulse is to act as a handsfree email reading device, perfect to catch a quick glance without distracting too much from a user's primary task. Texas Instrument introduced the Chronos, a very inexpensive smart watch, which is based on the embedded system MSP430 that provides an SDK for individual programs. Newer smart watches are Bluetooth enabled and some are water resistant. These smart watches are designed to be worn 24 hours a day and allow a long shower or a bath in the swimming pool.

Inactivity Recognition

The major difference in activity recognition between a smart phone and a smart watch is the wearing behaviour. If the user returns to his home, he will put off the phone. A smart watch can be worn constantly at the wearer’s body, no matter if he is doing sports or is at rest or sleeps.

Some systems are available that are detecting sleep or resting by wearing a sensor at the wrist. The Actiwatch is an actigraphy based data logger designed for rest activity patterns, quantify physical activity or sleep. Another watch-based sleep recognition solution is the Sleep tracker that detects sleep and allows a wake up at the estimated optimum time. Some other solution for sleep detection exists whereas some other sensor technology is used, e.g. EEG signals.

Acceleration Sensor

The performance of acceleration sensors can be measured in accuracy, quantization, sampling rate, sampling stability, range, noise, and energy consumption. Early types of acceleration sensors were metal balls within a coil environment that generated electric charge or an induction field while it was moved. Other system concepts were piezo crystal based; hereby a weight pressures on a piezo crystal that response to applied mechanical stress by electric charge. Accelerometers can also use strain gauges. Earthquake observatories were using optical levers or mechanical linkages to amplify the small motions. Nowadays, the acceleration sensors are using electronics. The most common sensor for home consumer products, e.g. phones, cameras, cars etc. is a MEMS, micro-electro-mechanical system, which is very cheap and low on power consumption. The usage of the MEMS in smart watches is a tradeoff between performance, energy consumption, size and costs. The MEMS-sensor in itself provides usually a high frequency sampling rate, e.g. more than 1000 Hertz. Due to the operating system and power requirements, the sampling rate is self-adjusting (Android) or is limited to a lower frequency. It is obvious that a high sampling rate provides more data than a lower rate, a good compromise in sampling rate seems to be at 32 Hz. One quality criteria of an acceleration sensor is the accuracy. We assume that the MEMS sensor show a hysteresis effect to the measured signal. Furthermore, the data shows an offset and a not linear gain. Sensor data sheets and trials show a tremendous offset of each axis. This offset can be around 0.5 m/s2, often the amplification factor shows an error of 2-3 percent. The offset is usually too high for physical way-time calculations. The following paragraph will discuss what requirements are necessary for activity and inactivity recognition for acceleration sensors.

6.4.2.2.2 Engineering Standard Requirements:

- Smart watches are mobile devices with a small display and a vibration feedback to indicate incoming messages. Because of the sensor functionality, smart watches can be used to support indoor navigation features in construction or maintenance work. In construction environments, the environment might be noisy and dirty. If a worker needs guidance to a place of interest, he can be guided by vibration signals of the smart watch.

- Within the research project eKon, we are developing new indoor navigation technologies in the context of ship construction and maintenance. The research project eKon is funded by the German Federal Ministry of Economics and Technology. Hereby, Smart watches are useful interaction devices for model based navigation, while the 3D plans of the environment are known. Smart watch supports inertial navigation as well because of the capability of a distance measuring functionality (pedometer).

- In some working environments, machines are transferring vibrations to the hand or arm of the worker. The Control of Vibration at Work Regulations 2005 (the Vibration Regulations), came into force on 6 July 2005 in Great Britain by the HSE. The Control of Vibration is based on a European Union Directive to protect workers from risks to their health and safety from vibration. Hereby the regulations on hand-arm vibration define a maximum of daily exposure value of 5 m/s2 A(8); and a daily exposure action value of 2.5 m/s2 A(8). The maximum vibration force on hand or arm might be very high, common devices provide a measuring range of approx. +/-500 g, some others +/- 1000 g. In comparison, the standard MEMS sensor has a range of max. +/- 16 g. The direct exposure to the arm or hand cannot be measured exactly by standard MEMS sensors. However, the smart watch is able to perform pattern recognition of the acceleration signals that allows the identification of the used tools and their using duration. We expect that the smart watch will be used for monitoring the hand and arm vibration exposition in combination with the recreation time in the near future.

7. PROPOSED ARCHITECHTURE AND CLOUD BUILD MODEL

Due to low computing power and limit power supply input from a battery around 200mAh the device itself is unable to handle the load of analysis and modelling the intended features of a smart wearable. By implementing the proper utilisation of modern IoT infrastructure we will be able to connect the device to live server and by uploading the data to the server we can analyse and classify user activity. Based on the classification, the system will generate recommendation based on the application of several machine learning model. We will discuss the various models and figure out which model will be the best according to the accuracy.

The cloud architecture is established based on a particular model architecture combining several components. Which is resulting creating a hybrid model for IoT and machine learning by connecting them through a web-based platform. By adding more functionalities, we can expand due to scalability.

7.1 Components -

The whole architecture is can be divided into two parts based on the model architecture. The first is the IoT model which is the user interaction model and the other is the machine learning part, which is the server-side model.

(a) Device (IoT): The device which will be attached to the user as a wearable device will be consisting of a hub of sensors collecting data from the user on a time interval set as the server requirements. Due to lack of space or computational power the user data will transfer the data temporarily to smart-phone using Bluetooth. The smart-phone will be able to send the data live to the cloud whenever it is in the range of proper internet or Wi-Fi connection.

(b)) Web Acquisition (API): The data which is being transferred to the smart-phone via smart wearable device, can be uploaded to the server via web APIs. Web API services are fast models for accessing and transferring data in a JSON format.

(c) Storage (DB): Uploaded data through API will be stored in a Database. The database can be of two types- SQL or NoSQL. For our system simple MySQL database is implemented to reduce the cost for storage. As NoSQL architectural infrastructure demands more cost for implementation.

7.2 ARCHITECTURE DIAGRAM

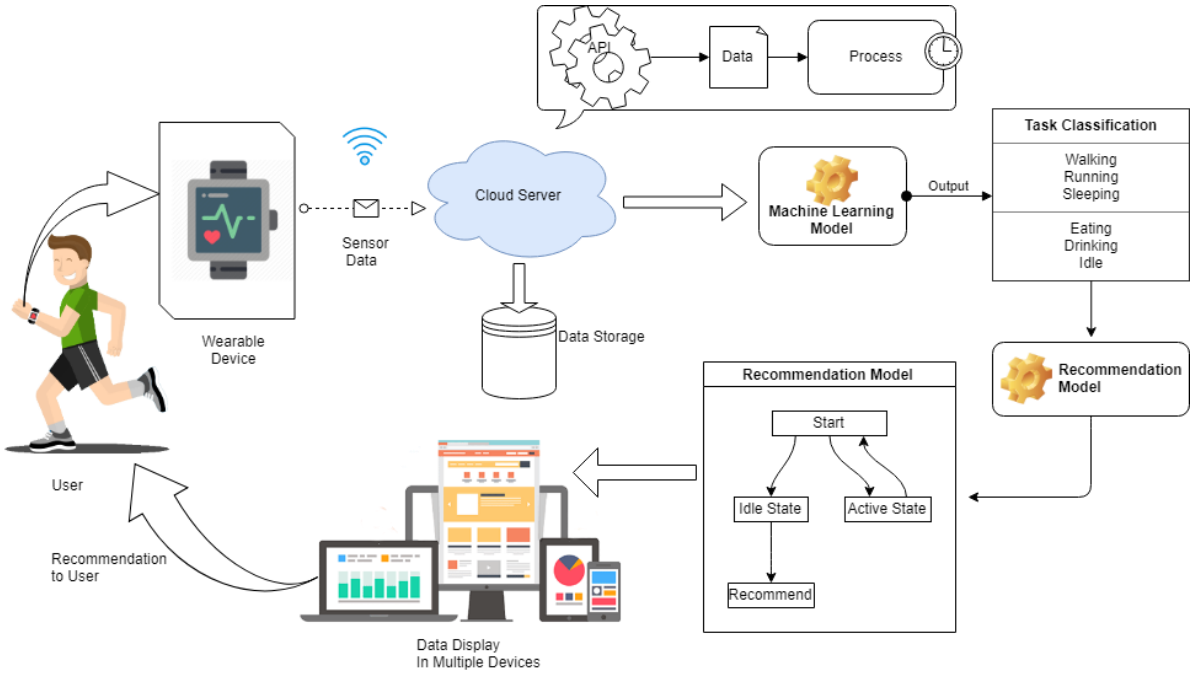


Fig: Proposed Model Architecture

8. Deep Learning

These techniques are an extension or a re-branding of neural networks [69]. They are based on the idea of modelling problems with high-level abstractions in data by using multiple processing layers that basically perform multiple non-linear transformations. These multiple transformations are organized hierarchically, with each layer processing the outputs of the previous layer, exploiting the idea of incremental learning, in which more abstract concepts are learned from the lower level ones. The processing layers can have various architectures, such as deep neural networks, convolutional deep neural networks, deep belief networks or recurrent neural networks. The idea of using deep learning layers is very interesting to researchers in many fields because these techniques can automatically build high-level representations of the raw input, offering a more generic solution because the feature construction process can be fully automated. Deep learning techniques have been successfully applied to many different fields, including computer vision, natural language processing, speech recognition, and so on, and have achieved state-ofthe-art results on various tasks [70,71,72]. Regarding activity recognition, deep learning techniques have been applied in a few works; a representative set can be found in [26,27,28,29,30,31].

- For a deep neural network (DNN), we chose the TensorFlow framework, recently introduced by Google (Mountain View, CA, USA) [73]. Additionally, to achieve a better integration with scikit-learn, we used the skflow package, which provides a simplified interface for accessing and mimicking TensorFlow scikit. As mentioned before, deep learning techniques have the advantage of being able to build high-level representations of the raw input data without the need for data processing and segmentation. While this is an interesting approach, we have identified a very high impact of the network architecture on the results, a behavior consistent with what we have found in the literature [32], including the fact that some architectures do not work at all. Therefore, in this work, to create an objective comparison, we used the same preprocessed data to test all of the classifiers, including the DNNs. The main motivation for testing these techniques with preprocessed data is to investigate whether their ability to extract and/or transform features among the 280 attributes translates into competitive classification performances. Given the importance of the architecture design, for this work, we have tested numerous topologies (combinations of tensors and dense and convolutional layers). However, after running a large set of experiments, we observed

through the evolution of the logistic regression loss function that simple architectures were sufficient to converge within a few iterations obtaining competitive results, as shown in Figure 4. Adding more complexity to the topology required considerably more computation time, all to achieve only very small improvements in accuracy (or even no improvement at all due to overfitting). The results of our tests with different architectures pointed out two things: (1) incorrect topology designs dramatically affect the results; and (2) the results obtained with simple architectures were probably so close to the global optimum that adding more complexity in the layers did not lead to any significant improvement.

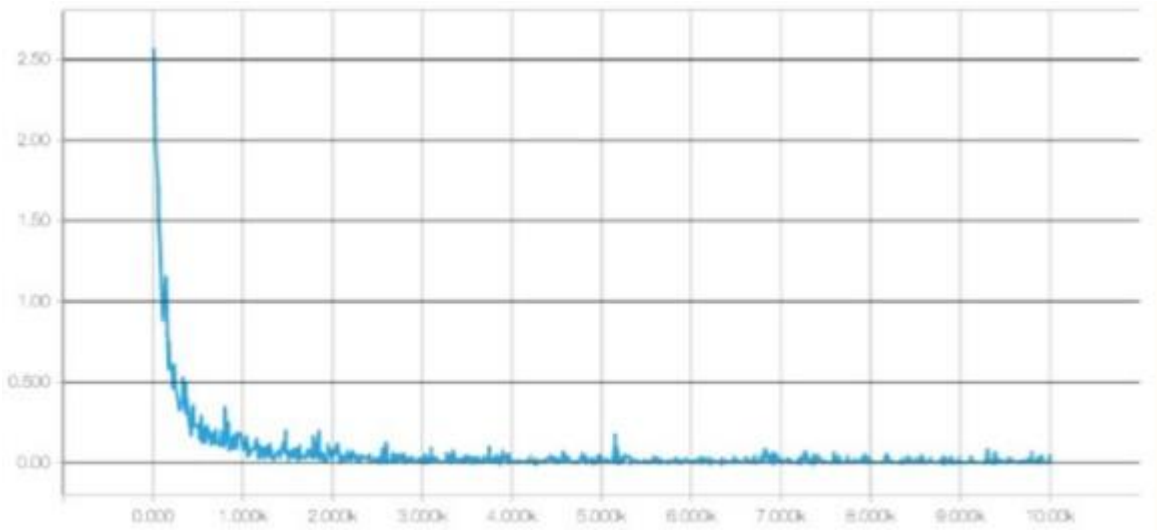


Fig: Evaluation of Logistic Regression loss functions among iterations of deep neural network training process, generated with TensorBoard

Consequently, for comparison purposes, we finally selected two configurations: DNN 1L, which has 1 hidden dense layer with 280 nodes, and DNN 2L, which has 2 hidden dense layers with 560 and 280 nodes, respectively. For both configurations, we used a logistic regression as the cost function, 10,000 iterations with the Adam optimizer, a dropout probability of 0.5 and a learning rate of 0.001. The computational graph for training such a deep neural network, as generated by TensorBoard, is shown in Figure 5.

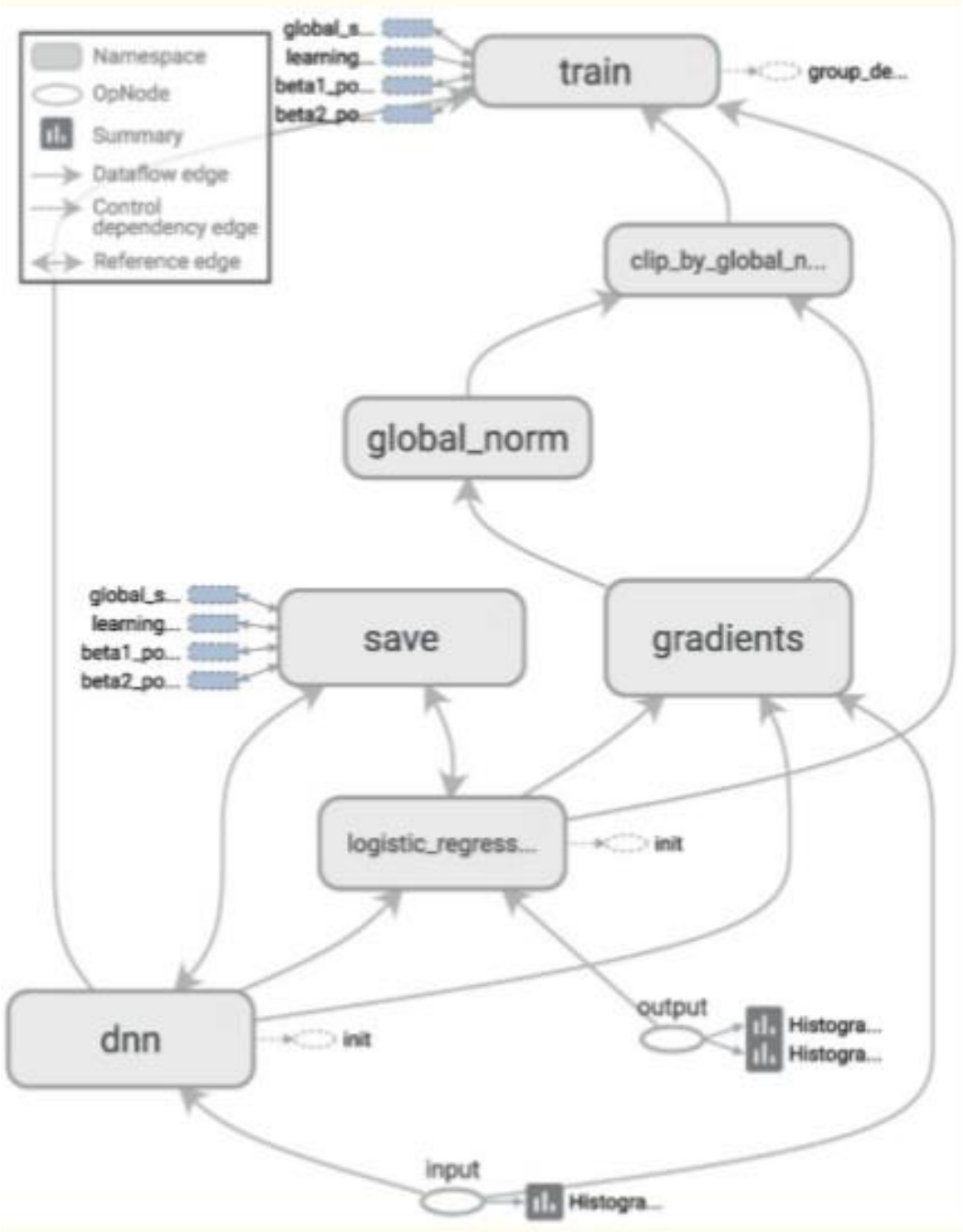


Fig 5- Architecture of a TensorFlow graph for training a deep neural network, generated with TensorBoard.

9. Machine Learning Model Acquisition

The activity recognition task involves mapping time series sensor data from a smartwatch and/or smartphone to a single activity. In our approach the time series data is aggregated into examples based on non-overlapping 10second intervals of data; an activity is recognized correctly if the single activity that occurred during the 10-second interval is correctly classified. Table I lists the eighteen activities included in this study. The activities are grouped into three general categories. The details associated with the data collection process are described in the next section.

- General activities (not hand-oriented)
  - Walking
  - Jogging
  - Climbing Stairs
  - Sitting
  - Standing
  - Kicking Soccer Ball
- General activities (hand-oriented)
  - Dribbling Basketball
  - Playing Catch with Tennis Ball (two people)
  - Typing
  - Handwriting
  - Clapping
  - Brushing Teeth
  - Folding Clothes
- Eating activities (hand-oriented)
  - Eating Pasta
  - Eating Soup
  - Eating Sandwich
  - Eating Chips
  - Drinking from a Cup

Table: Eighteen activities utilized in this study

The methodology associated with generating and evaluating the activity recognition models is discussed in this section. This includes the data collection procedure, the method for transforming the low-level time-series sensor data into examples, and the model induction

process. Some of this material has been presented in our prior smartphone based work, so in these cases some details may be omitted and replaced by appropriate citations.

A. Data Collection:

The data for this study was collected from 17 test subjects, each of whom were asked to perform the 18 activities listed in Table I. Each activity was performed for two minutes while the subject wore a smartwatch on their dominant hand and a paired smartphone in their front-right pocket with the phone oriented upright with the screen facing outward. All data was collected using the LG G Watch smartwatch and the Samsung Galaxy S4 smartphone, both of which were running the Android Wear mobile operating system. The data collection process took approximately one hour per person. All test subjects provided written informed consent prior to participating in this study, which was approved by our university's Institutional Review Board.

The data collection process utilizes our custom designed smartwatch/smartphone app. The app collects data from the accelerometer and gyroscope on both the phone and watch at a rate of 20Hz, with each sensor providing data for the three spatial dimensions. After two minutes of data is collected for an activity, the smartphone automatically sends the data, in an email message, to a server for storage and later retrieval. Unfortunately, unresolved and sporadic issues with the phone's gyroscope prevented sufficient phone-gyroscope data from being collected and hence this sensor is not incorporated into our analysis. The data received by the server was later trimmed to ensure that the collected data corresponded to an activity (sometimes test subjects delayed in starting the activity or ended early). The trimming resulted in having approximately 100 seconds of data for each activity rather than the full two minutes.

B. Data Transformation:

Conventional classification algorithms do not operate on time-series data, so the raw time series sensor data was transformed into examples by taking 10-second chunks of data and generating higher level features that describe the behaviour over this time period. The 43 higher level features involve computing averages, standard deviations, and other measures from the x, y, and z axis values for the sensor. The details of the data

transformation process, including a description of the higher level features, is provided in our prior work on smartphone-based activity recognition. Note that in this study we generate examples from only one sensor at a time; thus when we present the results in Section IV, the activity recognition performance of each sensor is provided separately (i.e., the models utilize a single sensor).

C. Model Induction:

The activity recognition models are induced from the labelled training examples using the WEKA [13] Random Forest (RF) algorithm, J48 decision tree algorithm, IB3 instance-based learning algorithm, Naïve Bayes (NB) algorithm, and the multi-layer perceptron (MLP) algorithm. The default parameter settings for each algorithm are used, except for NB where kernel estimation is enabled, and the instance-based learning algorithm where 3 nearest neighbours are used (there are dozens of default parameters for the algorithms employed in this paper so please refer to WEKA online documentation for more details).

Two types of models are induced. Personal models utilize data from only the intended user while impersonal models are built using training data from everyone but the intended user. Our prior research has shown that personal models vastly outperform impersonal models for activity recognition, but at the cost of requiring each user to provide labelled training data. In order to build and evaluate the personal models, data from each of the 17 subjects is partitioned into training and test sets using 10-fold cross validation. Therefore, the results for the personal models in the next section are based on results averaged over 17 ×10 = 170 models. The impersonal models are generated by taking training data from 16 users, building a model, and then evaluating that model on the remaining user; this is repeated 17 times so that all subjects are evaluated. In this case 10fold cross validation is not used since we have plenty of training data. The results for the impersonal models are averages over the 17 generated models.

The activity recognition results are presented and analysed in this section. The results for the personal and impersonal activity recognition models are presented in Table II and Table III, respectively. Results are measured using classification accuracy, which in this case corresponds to the percentage of the classifications that correctly identify the activity the user is performing. Each model utilizes a single sensor—models using the watch-accelerometer, phone-accelerometer, and watch-gyroscope are all evaluated. A model is also generated using each of the five learning algorithms described earlier. All results in Table II and III are based on 17 test subjects who performed the 18 activities that were listed in Table I.

TABLE II. OVERALL ACCURACY FOR PERSONAL MODELS

Algorithm	Phone accel (%)	Watch accel (%)	Watch gyroscope (%)
RF	75.5	93.3	79.0
J48	65.5	86.1	73.0
IB3	67.7	93.3	60.1
NB	77.1	92.7	80.2
MLP	77.0	94.2	70.0
Ave	72.6	91.9	72.4

TABLE III. OVERALL ACCURACY FOR IMPERSONAL MODELS

Algorithm	Phone accel (%)	Watch accel (%)	Watch gyroscope (%)
RF	35.1	70.3	57.5
J48	24.1	59.3	49.6
IB3	22.5	62.0	49.3
NB	26.2	63.8	53.5
MLP	18.9	64.6	57.7
Ave	25.3	64.0	53.5

The results show that, consistent with our previous smartphone-based activity recognition results, personal models outperform impersonal models, even though they are built using much less training data. The results also show that the watch accelerometer provides much better results than the phone accelerometer, with an average accuracy of 91.9% versus 72.6% for personal models and 64.0% versus 25.3% for impersonal models. These differences are largely due to the hand-based activities included in this study. The

watch gyroscope performs much more poorly than the watch accelerometer, with an average accuracy of 72.4% versus 91.9% for the personal models and 53.5% versus 64.0% for the impersonal models (as mentioned earlier there were technical difficulties in capturing the phone gyroscope data). While it makes sense to generate models using the fusion of multiple sensors, this is not done because of issues synchronizing the data from different sensors—possibly running on different devices. Since the watch accelerometer yields the best models for both personal and impersonal models, we focus most of our attention on this sensor. It is worth emphasizing that the results for this sensor are quite impressive, given the diversity of activities that are tracked, the granularity of the activities (i.e., eating activities are not all grouped together), and the fact that a strategy of guessing the most common activity would yield an accuracy of only about 5% (i.e., about 1 in 18).

TABLE IV. PER-ACTIVITY ACCURACY FOR RF MODELS

Activity	Impersonal (%)			Personal (%)		
	Watch accel	Phone accel	Watch gyro	Watch accel	Phone accel	Watch gyro
Walking	79.8	60.7	87.0	94.2	88.5	93.5
Jogging	97.7	93.8	48.6	99.2	68.8	98.1
Stairs	58.5	66.7	43.1	88.9	66.7	80.0
Sitting	84.9	26.9	70.5	97.5	87.0	82.2
Standing	96.3	65.9	57.9	98.1	73.1	68.6
Kicking	71.3	72.5	41.4	88.7	91.7	67.9
Dribbling	89.3	26.1	86.0	98.7	84.8	96.9
Catch	66.0	26.1	68.9	93.3	78.3	94.6
Typing	80.4	76.9	60.8	99.4	72.0	88.6
Handwriting	85.2	12.9	63.1	100.0	75.9	80.5
Clapping	76.3	40.9	67.9	96.9	77.3	95.6
Brush Teeth	84.5	19.2	66.2	97.3	96.2	89.6
Fold Clothes	80.8	8.3	37.8	95.0	79.2	73.1
Eat Pasta	47.1	0.0	57.9	88.6	40.0	72.9
Eat Soup	52.7	0.0	47.7	90.7	82.4	69.8
Eat Sandwich	29.0	7.1	31.1	68.9	63.0	44.2
Eat Chips	65.0	16.0	50.6	83.4	76.0	52.5
Drink	62.7	31.8	61.1	93.3	77.3	78.5
Overall	70.3	35.1	57.5	93.3	75.5	79.0

It is informative to examine the accuracies associated with individual activities, in order to determine which activities are easy to recognize and which ones are difficult to recognize. Due to space limitations, we focus our analysis on models induced using the Random Forest (RF) algorithm, since this algorithm performs well over the 6 configurations (3 sensors and 2 types of models). The accuracies of the RF models, over the 18 activities for both personal and impersonal models, are shown in Table IV. These results indicate that the accuracy for the activities varies widely. The last group of activities (i.e., eating activities) seems to have the lowest accuracy when compared to the results for the other two groupings. This suggests perhaps that the eating activities may be similar and may be getting confused with one another. We also see that the phone sensor performs much worse than the two watch sensors for the eating activity, for both the impersonal and personal models—but especially for the impersonal models. This is most likely due to that fact that many of the hand-based activities in the second grouping also involve significant body motion (e.g., dribbling and folding clothes). It is interesting that a phone in one's pocket can be effective at identifying many hand-based activities, but is ineffective at identifying eating activities. We had previously seen that personal models outperform impersonal models, but the results in Table IV show that personal models are particularly effective at boosting the performance of the eating activities—perhaps suggesting that there is wide variance in how people eat.

The results in Table IV do not tell us how there errors are distributed or—more to the point—which activities are confused with one another. A model's confusion matrix can answer this, but since each confusion matrix has dimensions 18×18, it is not practical to display them here. But we can analyze these matrices for the most problematic cases. For example, if we look at the results for the personal RF models built from the watch-accelerometer sensor, we see that the hardest activity to recognize is "eating a sandwich," which has a 68.9% accuracy. The confusion matrix indicates that this activity is misclassified as "eating soup" 10.9% of the time, "eating chips" 9.4% of the time, and "eating pasta" 5.8% of the time (other misclassifications occur less frequently). This shows that the common mistakes involve other eating activities, suggesting that much higher accuracy could be achieved by grouping all of the eating activities together. Similarly, the impersonal model using the phone accelerometer has a 0% accuracy for "eating soup"

and the underlying confusion matrix shows that the single largest source of errors is due to misclassifying this activity as “drinking.” It seems reasonable that these two activities would appear similar based on the motion measured at a person’s upper thigh (i.e., by the pants pocket).

10. RESULTS

A useful tool to visualize at real time data of the MPU9250 sent through a COM port, using PyQtgraph and PyQt5.

The MPU-9250 is a sensor from Invensense Inc. that combine in one package both accelerometer, gyroscope and magnetometer. It embeds also a thermometer and other useful things.

Repository Contents

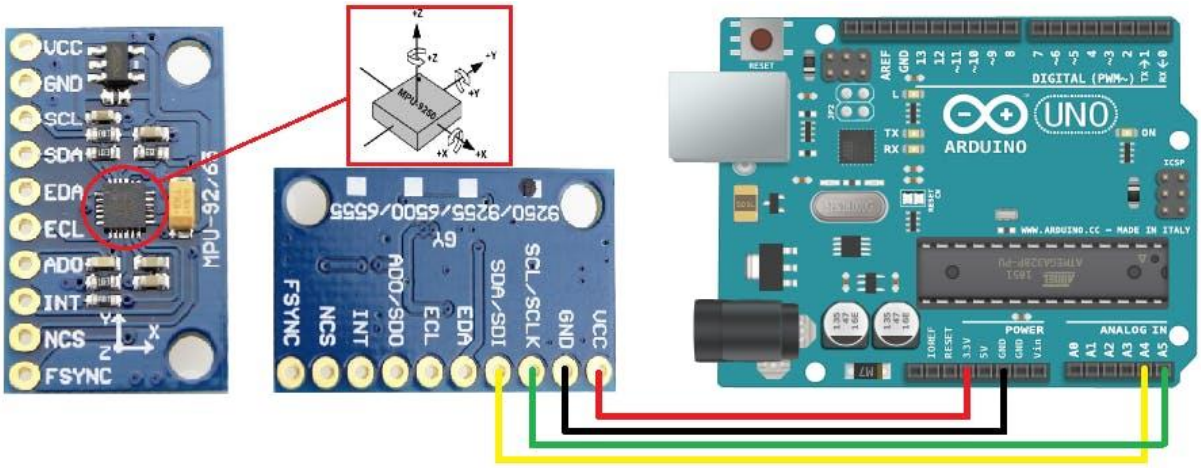
- /Libraries — An Arduino library from sparkfun
- /acq_mpu9250 — The Arduino code to send data to a COM port
- mpuScrollingPlot.py — A Python script for "scrolling plotting"
- mpuPlotSavedData.py — to visualize saved data

Required Python Package

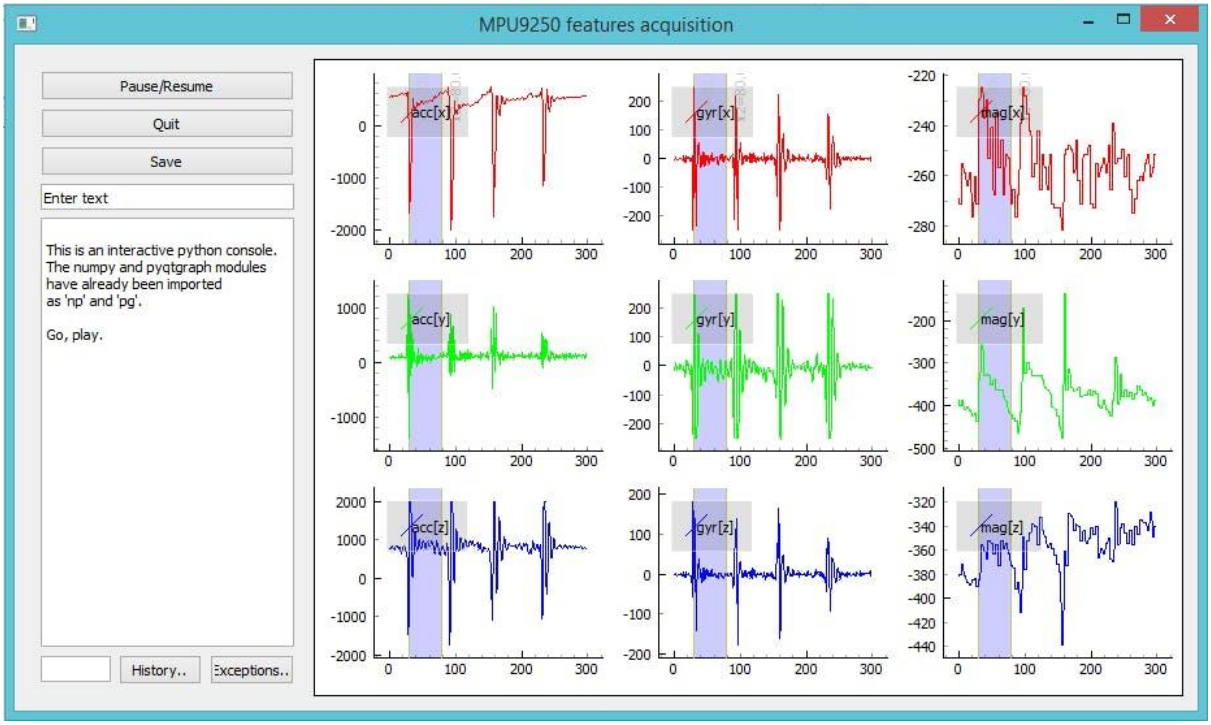
- Import serial
- Import time
- Import numpy as np
- From matplotlib import pyplot as plt
- Import pyqtgraph as pg
- From pyqtgraph.Qt import QtCore, QtGui
- Import os
- Import pyqtgraph.console
- Import PyQt5

Sending data to a COM port

I use an Arduino board. Here is how I do it:

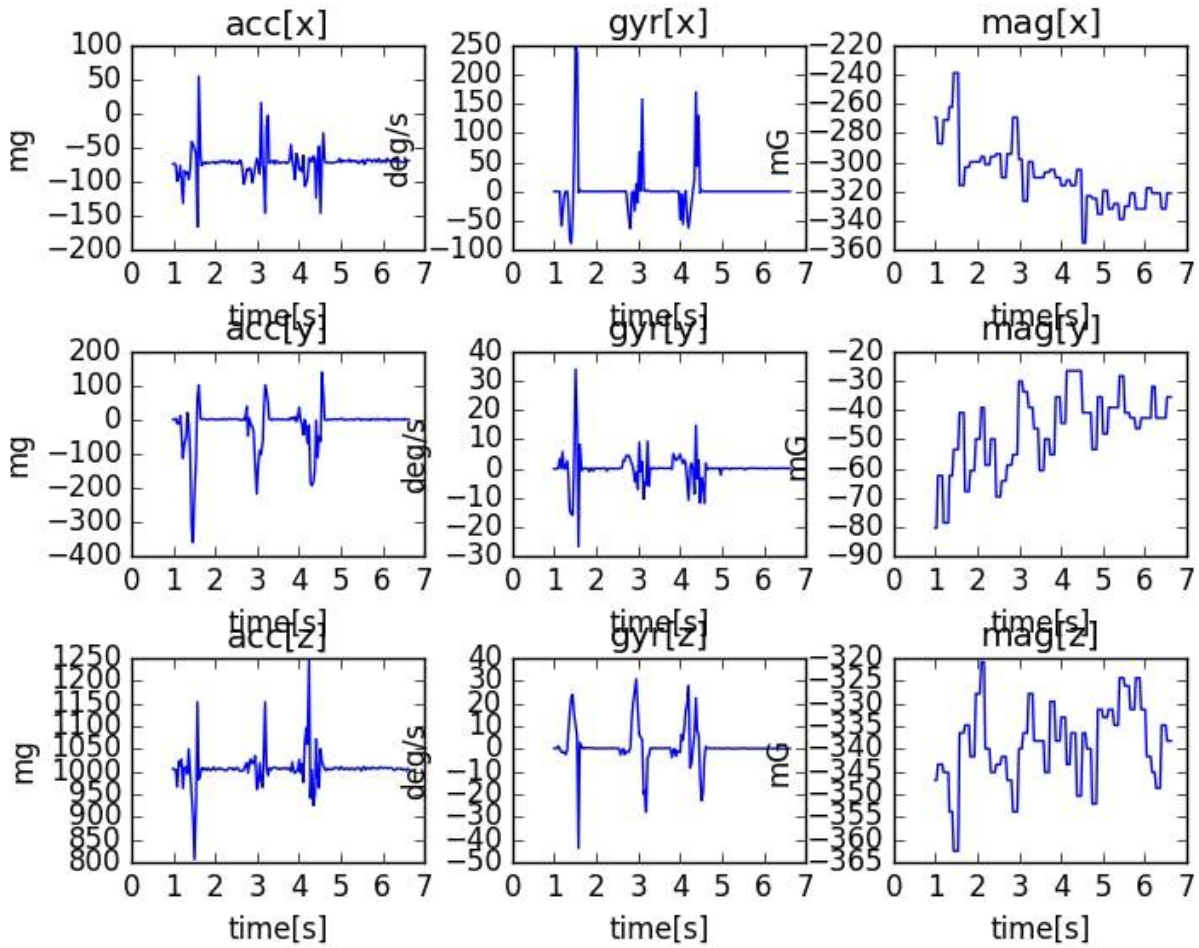


Here is the interface. It allows you to visualize data. You can also save the data between cursors and plot them later.



The data was collected from several users by attaching the module to user body. The accelerometer and gyroscope data are captured on the basis of the axis- x, y & z. The data was able to send through COM port and also by utilizing the functionalities of PyQtgraph and PyQt5, we were able to visualise the data as a live aspect. In Fig -3 we have displayed a sample of such data plotting using python matplot library module.

The script **mpuPlotSavedData.py** allows you to plot the saved data using matplotlib.



The six activities performed were as follows:

1. Walking
2. Walking Upstairs
3. Walking Downstairs
4. Sitting
5. Standing
6. Laying

The movement data recorded was the x, y, and z accelerometer data (linear acceleration) and gyroscopic data (angular velocity) from the smart phone, specifically a Samsung Galaxy S II.

Observations were recorded at 50 Hz (i.e. 50 data points per second). Each subject performed the sequence of activities twice, once with the device on their left-hand-side and once with the device on their right-hand side.

A group of 30 volunteers with ages ranging from 19 to 48 years were selected for this task. Each person was instructed to follow a protocol of activities while wearing a waist-mounted Samsung Galaxy S II smartphone. The six selected ADL were standing, sitting, laying down, walking, walking downstairs and upstairs. Each subject performed the protocol twice: on the first trial the smartphone was fixed on the left side of the belt and on the second it was placed by the user himself as preferred –

— A Public Domain Dataset for Human Activity Recognition Using Smartphones, 2013.

The raw data is not available. Instead, a pre-processed version of the dataset was made available.

The pre-processing steps included:

- Pre-processing accelerometer and gyroscope using noise filters.
- Splitting data into fixed windows of 2.56 seconds (128 data points) with 50% overlap.
- Splitting of accelerometer data into gravitational (total) and body motion components.

These signals were preprocessed for noise reduction with a median filter and a 3rd order low-pass Butterworth filter with a 20 Hz cutoff frequency. […] The acceleration signal, which has

gravitational and body motion components, was separated using another Butterworth low-pass filter into body acceleration and gravity.

Feature engineering was applied to the window data, and a copy of the data with these engineered features was made available.

A number of time and frequency features commonly used in the field of human activity recognition were extracted from each window. The result was a 561 element vector of features.

The dataset was split into train (70%) and test (30%) sets based on data for subjects, e.g. 21 subjects for train and nine for test.

This suggests a framing of the problem where a sequence of movement activity is used as input to predict the portion (2.56 seconds) of the current activity being performed, where a model trained on known subjects is used to predict the activity from movement on new subjects.

Early experiment results with a support vector machine intended for use on a smartphone (e.g. fixed-point arithmetic) resulted in a predictive accuracy of 89% on the test dataset, achieving similar results as an unmodified SVM implementation.

This method adapts the standard Support Vector Machine (SVM) and exploits fixed-point arithmetic for computational cost reduction. A comparison with the traditional SVM shows a significant improvement in terms of computational costs while maintaining similar accuracy […]

— Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine, 2012.

Now that we are familiar with the prediction problem, we will look at loading and exploring this dataset.

The contents of the "train" and "test" folders are similar (e.g. folders and file names), although with differences in the specific data they contain.

Inspecting the "train" folder shows a few important elements:

- An “Inertial Signals” folder that contains the preprocessed data.
- The “X_train.txt” file that contains the engineered features intended for fitting a model.
- The “y_train.txt” that contains the class labels for each observation (1-6).
- The “subject_train.txt” file that contains a mapping of each line in the data files with their subject identifier (1-30).

The number of lines in each file match, indicating that one row is one record in each data file.

The “*Inertial Signals*” directory contains 9 files.

- *Gravitational acceleration* data files for x, y and z axes: *total_acc_x_train.txt*, *total_acc_y_train.txt*, *total_acc_z_train.txt*.
- *Body acceleration* data files for x, y and z axes: *body_acc_x_train.txt*, *body_acc_y_train.txt*, body_acc_z_train.txt.
- *Body gyroscope* data files for x, y and z axes: *body_gyro_x_train.txt*, *body_gyro_y_train.txt*, *body_gyro_z_train.txt*.

The structure is mirrored in the “test” directory.

We will focus our attention on the data in the “Inertial Signals” as this is most interesting in developing machine learning models that can learn a suitable representation, instead of using the domain-specific feature engineering.

Inspecting a datafile shows that columns are separated by whitespace and values appear to be scaled to the range -1, 1. This scaling can be confirmed by a note in the README.txt file provided with the dataset.

Now that we know what data we have, we can figure out how to load it into memory.

In this section, we will develop some code to load the dataset into memory.

First, we need to load a single file.

LOAD DATA

In this section, we will develop some code to load the dataset into memory.

First, we need to load a single file.

We can use the read_csv() Pandas function to load a single data file and specify that the file has no header and to separate columns using white space.

```
In [1]:
###
# import packages
###
import os
import pickle

import pandas as pd
import numpy as np
```

```
In [3]:
# load dataset
from pandas import read_csv
```

```
In [4]:
# load a single file as a numpy array
def load_file(filepath):
        dataframe = read_csv(filepath, header=None, delim_whitespace=True)
        return dataframe.values

data = load_file('HARDataset/train/Inertial Signals/total_acc_y_train.txt')
print(data.shape)
```

```
(7352, 128)
```

We can wrap this in a function named load_file(). The complete example of this function is listed below.

```
In [8]:
# load dataset
from numpy import dstack
from pandas import read_csv

# load a single file as a numpy array
def load_file(filepath):
        dataframe = read_csv(filepath, header=None, delim_whitespace=True)
        return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
        loaded = list()
        for name in filenames:
                data = load_file(prefix + name)
                loaded.append(data)
        # stack group so that features are the 3rd dimension
        loaded = dstack(loaded)
        return loaded

# load the total acc data
filenames = ['total_acc_x_train.txt', 'total_acc_y_train.txt', 'total_acc_z_train.txt']
total_acc = load_group(filenames, prefix='HARDataset/train/Inertial Signals/')
print(total_acc.shape)
```

```
(7352, 128, 3)
```

Running the example loads the file '*total_acc_y_train.txt*', returns a NumPy array, and prints the shape of the array.

We can see that the training data is comprised of 7,352 rows or windows of data, where each window has 128 observations.

Next, it would be useful to load a group of files, such as all of the body acceleration data files as a single group.

Ideally, when working with multivariate time series data, it is useful to have the data structured in the format:

```
[samples, timesteps, features]
```

This is helpful for analysis and is the expectation of deep learning models such as convolutional neural networks and recurrent neural networks.

We can achieve this by calling the above *load_file()* function multiple times, once for each file in a group.
Once we have loaded each file as a NumPy array, we can combine or stack all three arrays together. We can use the dstack() NumPy function to ensure that each array is stacked in such a way that the features are separated in the third dimension, as we would prefer.
The function *load_group()* implements this behavior for a list of file names and is listed below.

```
# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
	loaded = list()
	for name in filenames:
		data = load_file(prefix + name)
		loaded.append(data)
	# stack group so that features are the 3rd dimension
	loaded = dstack(loaded)
	return loaded
```

We can demonstrate this function by loading all of the total acceleration files.

The complete example is listed below.

```
# load dataset
from numpy import dstack
from pandas import read_csv

# load a single file as a numpy array
def load_file(filepath):
	dataframe = read_csv(filepath, header=None, delim_whitespace=True)
	return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
	loaded = list()
	for name in filenames:
		data = load_file(prefix + name)
```

```
            loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load the total acc data
filenames = ['total_acc_x_train.txt', 'total_acc_y_train.txt', 'total_acc_z_train.txt']
total_acc = load_group(filenames, prefix='HARDataset/train/Inertial Signals/')
print(total_acc.shape)
```

Running the example prints the shape of the returned NumPy array, showing the expected number of samples and time steps with the three features, x, y, and z for the dataset.

```
(7352, 128, 3)
```

Finally, we can use the two functions developed so far to load all data for the train and the test dataset.

```
In [9]: # load a dataset group, such as train or test
        def load_dataset(group, prefix=''):
            filepath = prefix + group + '/Inertial Signals/'
            # load all 9 files as a single array
            filenames = list()
            # total acceleration
            filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
            # body acceleration
            filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
            # body gyroscope
            filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
            # load input data
            X = load_group(filenames, filepath)
            # load class output
            y = load_file(prefix + group + '/y_'+group+'.txt')
            return X, y
```

Running the example loads the train and test datasets.

We can see that the test dataset has 2,947 rows of window data. As expected, we can see that the size of windows in the train and test sets matches and the size of the output (y) in each the train and test case matches the number of samples.

```
(7352, 128, 9) (7352, 1)
(2947, 128, 9) (2947, 1)
```

Now that we know how to load the data, we can start to explore it.

Balance of Activity Classes

A good first check of the data is to investigate the balance of each activity.

We believe that each of the 30 subjects performed each of the six activities.

Confirming this expectation will both check that the data is indeed balanced, making it easier to model, and confirm that we are correctly loading and interpreting the dataset.

We can develop a function that summarizes the breakdown of the output variables, e.g. the y variable.

The function *class_breakdown()* below implements this behavior, first wrapping the provided NumPy array in a DataFrame, grouping the rows by the class value, and calculating the size of each group (number of rows). The results are then summarized, including the count and the percentage.

```
# summarize the balance of classes in an output variable column
def class_breakdown(data):
	# convert the numpy array into a dataframe
	df = DataFrame(data)
	# group data by the class value and calculate the number of rows
	counts = df.groupby(0).size()
	# retrieve raw rows
	counts = counts.values
	# summarize
	for i in range(len(counts)):
		percent = counts[i] / len(df) * 100
		print('Class=%d, total=%d, percentage=%.3f' % (i+1, counts[i], percent))
```

It may be useful to summarize the breakdown of the classes in the train and test datasets to ensure they have a similar breakdown, then compare the result to the breakdown on the combined dataset.

Running the example first summarizes the breakdown for the training set. We can see a pretty similar distribution of each class hovering between 13% and 19% of the dataset.

The result on the test set and on both datasets together look very similar.

It is likely safe to work with the dataset assuming the distribution of classes is balanced per train and test set and perhaps per subject.

```
Train Dataset
Class=1, total=1226, percentage=16.676
Class=2, total=1073, percentage=14.595
Class=3, total=986, percentage=13.411
Class=4, total=1286, percentage=17.492
Class=5, total=1374, percentage=18.689
Class=6, total=1407, percentage=19.138
```

```
 8
 9  Test Dataset
10  Class=1, total=496, percentage=16.831
11  Class=2, total=471, percentage=15.982
12  Class=3, total=420, percentage=14.252
13  Class=4, total=491, percentage=16.661
14  Class=5, total=532, percentage=18.052
15  Class=6, total=537, percentage=18.222
16
17  Both
18  Class=1, total=1722, percentage=16.720
19  Class=2, total=1544, percentage=14.992
20  Class=3, total=1406, percentage=13.652
21  Class=4, total=1777, percentage=17.254
22  Class=5, total=1906, percentage=18.507
23  Class=6, total=1944, percentage=18.876
```

Plot Time Series Data for One Subject

We are working with time series data, therefore an import check is to create a line plot of the raw data.

The raw data is comprised of windows of time series data per variable, and the windows do have a 50% overlap. This suggests we may see some repetition in the observations as a line plot unless the overlap is removed.

We can start off by loading the training dataset using the functions developed above.

```
1 # load data
2 trainX, trainy = load_dataset('train', 'HARDataset/')
```

Next, we can load the '*subject_train.txt*' in the '*train*' directory that provides a mapping of rows to the subject to which it belongs.
We can load this file using the *load_file()* function. Once loaded, we can also use the *unique()*NumPy function to retrieve a list of the unique subjects in the training dataset.

```
1  sub_map = load_file('HARDataset/train/subject_train.txt')
2  train_subjects = unique(sub_map)
3  print(train_subjects)
```

Next, we need a way to retrieve all of the rows for a single subject, e.g. subject number 1.

We can do this by finding all of the row numbers that belong to a given subject and use those row numbers to select the samples from the loaded X and y data from the training dataset.

The *data_for_subject()* function below implements this behavior. It will take the loaded training data, the loaded mapping of row number to subjects, and the subject identification number for the subject that we are interested in, and will return the *X* and *y* data for only that subject.

```
# get all data for one subject
def data_for_subject(X, y, sub_map, sub_id):
	# get row indexes for the subject id
	ix = [i for i in range(len(sub_map)) if sub_map[i]==sub_id]
	# return the selected samples
	return X[ix, :, :], y[ix]
```

Now that we have data for one subject, we can plot it.

The data is comprised of windows with overlap. We can write a function to remove this overlap and squash the windows down for a given variable into one long sequence that can be plotted directly as a line plot.

The *to_series()* function below implements this behavior for a given variable, e.g. array of windows.

```
# convert a series of windows to a 1D list
def to_series(windows):
	series = list()
	for window in windows:
		# remove the overlap from the window
		half = int(len(window) / 2) - 1
		for value in window[-half:]:
			series.append(value)
	return series
```

Finally, we have enough to plot the data. We can plot each of the nine variables for the subject in turn and a final plot for the activity level.

Each series will have the same number of time steps (length of x-axis), therefore, it may be useful to create a subplot for each variable and align all plots vertically so we can compare the movement on each variable.

The *plot_subject()* function below implements this behavior for the *X* and *y* data for a single subject. The function assumes the same order of the variables (3rd axis) as was loaded in the *load_dataset()* function. A crude title is also added to each plot so we don't get easily confused about what we are looking at.

```
# plot the data for one subject
def plot_subject(X, y):
	pyplot.figure()
	# determine the total number of plots
	n, off = X.shape[2] + 1, 0
	# plot total acc
	for i in range(3):
		pyplot.subplot(n, 1, off+1)
		pyplot.plot(to_series(X[:, :, off]))
		pyplot.title('total acc '+str(i), y=0, loc='left')
		off += 1
	# plot body acc
	for i in range(3):
		pyplot.subplot(n, 1, off+1)
		pyplot.plot(to_series(X[:, :, off]))
		pyplot.title('body acc '+str(i), y=0, loc='left')
		off += 1
	# plot body gyro
	for i in range(3):
		pyplot.subplot(n, 1, off+1)
		pyplot.plot(to_series(X[:, :, off]))
		pyplot.title('body gyro '+str(i), y=0, loc='left')
		off += 1
	# plot activities
	pyplot.subplot(n, 1, n)
	pyplot.plot(y)
	pyplot.title('activity', y=0, loc='left')
	pyplot.show()
```

Running the example prints the unique subjects in the training dataset, the sample of the data for the first subject, and creates one figure with 10 plots, one for each of the nine input variables and the output class.

```
[ 1 3 5 6 7 8 11 14 15 16 17 19 21 22 23 25 26 27 28 29 30]
(341, 128, 9) (341, 1)
```

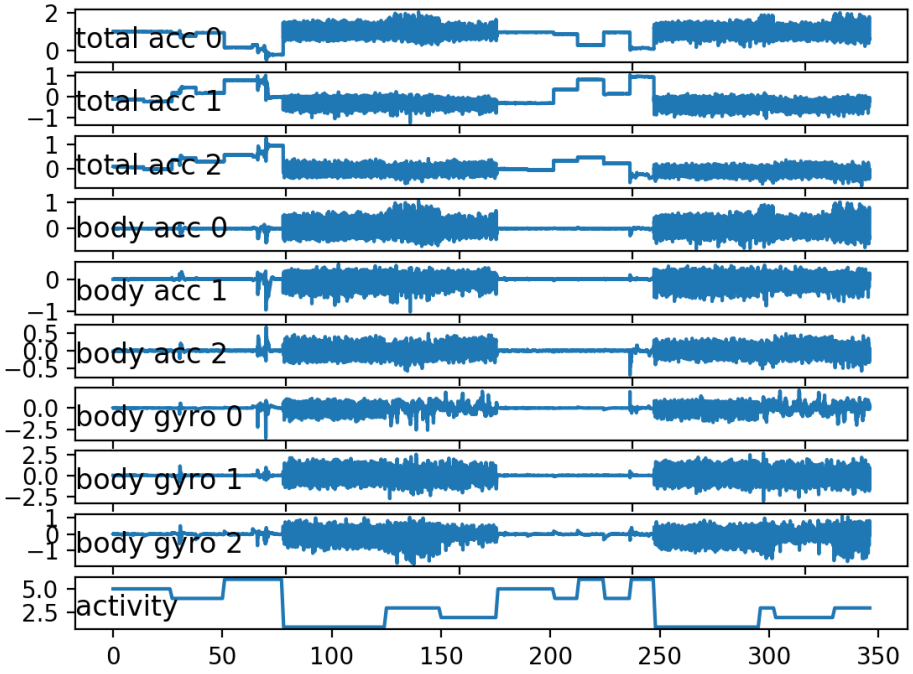
In the plot, we can see periods of large movement corresponding with activities 1, 2, and 3: the walking activities. We can also see much less activity (i.e. a relatively straight line) for higher numbered activities, 4, 5, and 6 (sitting, standing, and laying).

This is good confirmation that we have correctly loaded interpreted the raw dataset.

We can see that this subject has performed the same general sequence of activities twice, and some activities are performed more than two times. This suggests that for a given subject, we should not make assumptions about what activities may have been performed or their order.

We can also see some relatively large movement for some stationary activities, such as laying. It is possible that these are outliers or related to activity transitions. It may be possible to smooth or remove these observations as outliers.

Finally, we see a lot of commonality across the nine variables. It is very likely that only a subset of these traces are required to develop a predictive model.



Plotting Histograms Per Subject

As the problem is framed, we are interested in using the movement data from some subjects to predict activities from the movement of other subjects.

This suggests that there must be regularity in the movement data across subjects. We know that the data has been scaled between -1 and 1, presumably per subject, suggesting that the amplitude of the detected movements will be similar.

We would also expect that the distribution of movement data would be similar across subjects, given that they performed the same actions.

We can check for this by plotting and comparing the histograms of the movement data across subjects. A useful approach would be to create one plot per subject and plot all three axis of a given data (e.g. total acceleration), then repeat this for multiple subjects. The plots can be modified to use the same axis and aligned horizontally so that the distributions for each variable across subjects can be compared.

The *plot_subject_histograms()* function below implements this behavior. The function takes the loaded dataset and mapping of rows to subjects as well as a maximum number of subjects to plot, fixed at 10 by default.
A plot is created for each subject and the three variables for one data type are plotted as histograms with 100 bins, to help to make the distribution obvious. Each plot shares the same axis, which is fixed at the bounds of -1 and 1.

```
# plot histograms for multiple subjects
def plot_subject_histograms(X, y, sub_map, n=10):
	pyplot.figure()
	# get unique subjects
	subject_ids = unique(sub_map[:,0])
	# enumerate subjects
	xaxis = None
	for k in range(n):
		sub_id = subject_ids[k]
		# get data for one subject
		subX, _ = data_for_subject(X, y, sub_map, sub_id)
		# total acc
		for i in range(3):
```

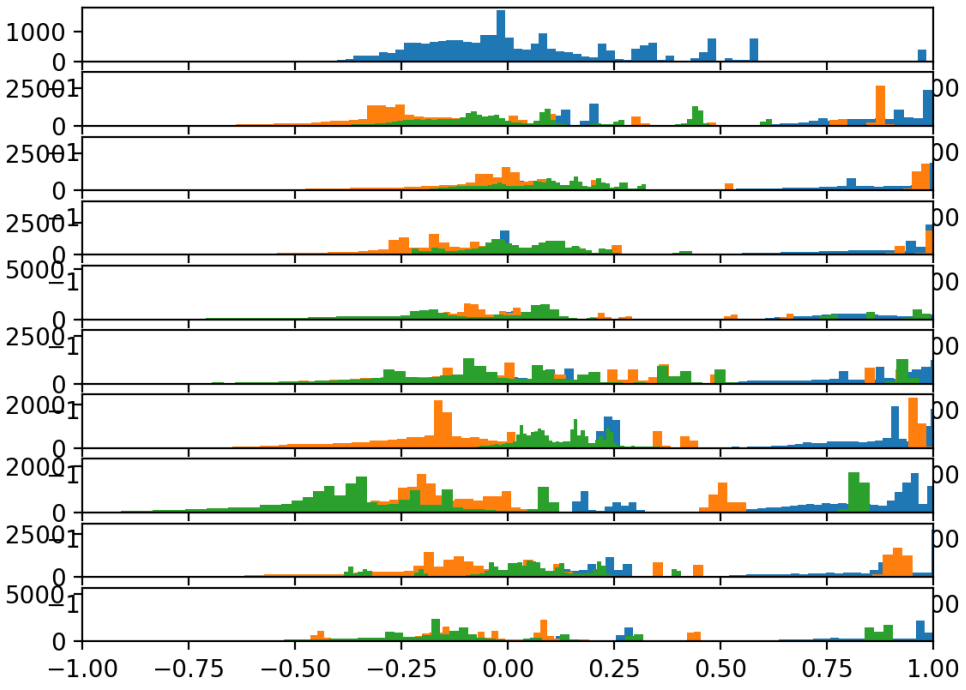
```
14                              ax = pyplot.subplot(n, 1, k+1, sharex=xaxis)
15                              ax.set_xlim(-1,1)
16                              if k == 0:
17                                      xaxis = ax
18                              pyplot.hist(to_series(subX[:,:,i]), bins=100)
19      pyplot.show()
```

Running the example creates a single figure with 10 plots with histograms for the three axis of the total acceleration data.

Each of the three axes on a given plot have a different color, specifically x, y, and z are blue, orange, and green respectively.

We can see that the distribution for a given axis does appear Gaussian with large separate groups of data.

We can see some of the distributions align (e.g. main groups in the middle around 0.0), suggesting there may be some continuity of the movement data across subjects, at least for this data.



Plot Histograms Per Activity

We are interested in discriminating between activities based on activity data.

The simplest case for this would be to discriminate between activities for a single subject. One way to investigate this would be to review the distribution of movement data for a subject by activity. We would expect to see some difference in the distribution between the movement data for different activities by a single subject.

We can review this by creating a histogram plot per activity, with the three axis of a given data type on each plot. Again, the plots can be arranged horizontally to compare the distribution of each data axis by activity. We would expect to see differences in the distributions across activities down the plots.

First, we must group the traces for a subject by activity. The *data_by_activity()* function below implements this behaviour.

```
# group data by activity
def data_by_activity(X, y, activities):
	# group windows by activity
	return {a:X[y[:,0]==a, :, :] for a in activities}
```

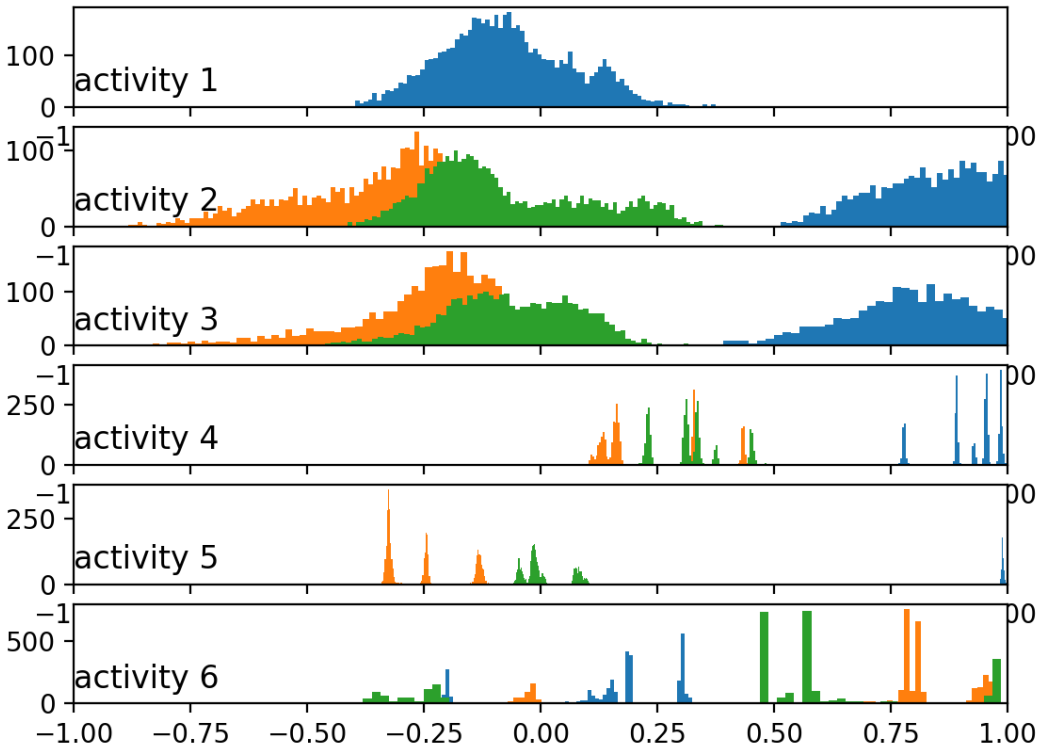
We can now create plots per activity for a given subject.

The *plot_activity_histograms()* function below implements this function for the traces data for a given subject.
First, the data is grouped by activity, then one subplot is created for each activity and each axis of the data type is added as a histogram. The function only enumerates the first three features of the data, which are the total acceleration variables.

```
# plot histograms for each activity for a subject
def plot_activity_histograms(X, y):
	# get a list of unique activities for the subject
	activity_ids = unique(y[:,0])
	# group windows by activity
	grouped = data_by_activity(X, y, activity_ids)
	# plot per activity, histograms for each axis
	pyplot.figure()
	xaxis = None
	for k in range(len(activity_ids)):
		act_id = activity_ids[k]
		# total acceleration
		for i in range(3):
			ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
			ax.set_xlim(-1,1)
			if k == 0:
				xaxis = ax
			pyplot.hist(to_series(grouped[act_id][:,:,i]), bins=100)
			pyplot.title('activity '+str(act_id), y=0, loc='left')
	pyplot.show()
```

Running the example creates the plot with six subplots, one for each activity for the first subject in the train dataset. Each of the x, y, and z axes for the total acceleration data have a blue, orange, and green histogram respectively.

We can see that each activity has a different data distribution, with a marked difference between the large movement (first three activities) with the stationary activities (last three activities). Data distributions for the first three activities look Gaussian with perhaps differing means and standard deviations. Distributions for the latter activities look multi-modal (i.e. multiple peaks).



We can re-run the same example with an updated version of the *plot_activity_histograms()*that plots the body acceleration data instead. The updated function is listed below.

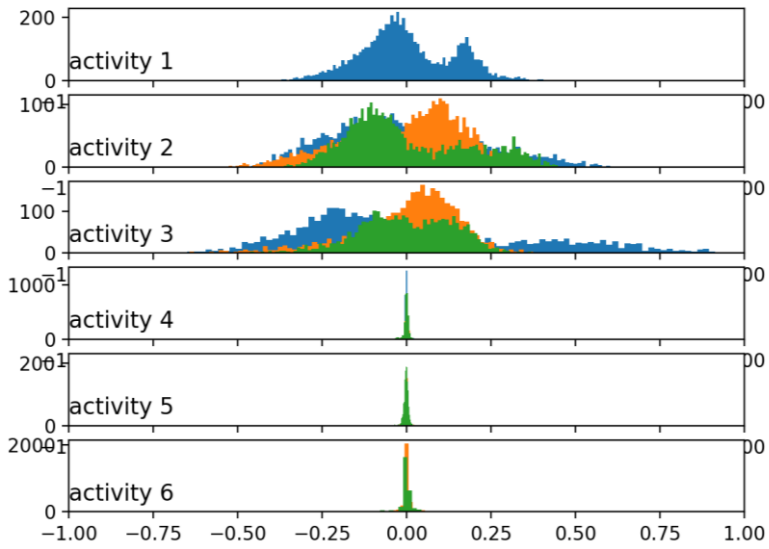
```
# plot histograms for each activity for a subject
def plot_activity_histograms(X, y):
```

```
# get a list of unique activities for the subject
activity_ids = unique(y[:,0])
# group windows by activity
grouped = data_by_activity(X, y, activity_ids)
# plot per activity, histograms for each axis
pyplot.figure()
xaxis = None
for k in range(len(activity_ids)):
        act_id = activity_ids[k]
        # total acceleration
        for i in range(3):
                ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
                ax.set_xlim(-1,1)
                if k == 0:
                        xaxis = ax
                pyplot.hist(to_series(grouped[act_id][:,:,3+i]), bins=100)
                pyplot.title('activity '+str(act_id), y=0, loc='left')
pyplot.show()
```

Running the updated example creates a new plot.

Here, we can see more similar distributions across the activities amongst the in-motion vs. stationary activities. The data looks bimodal in the case of the in-motion activities and perhaps Gaussian or exponential in the case of the stationary activities.

The pattern we see with the total vs. body acceleration distributions by activity mirrors what we see with the same data types across subjects in the previous section. Perhaps the total acceleration data is the key to discriminating the activities.



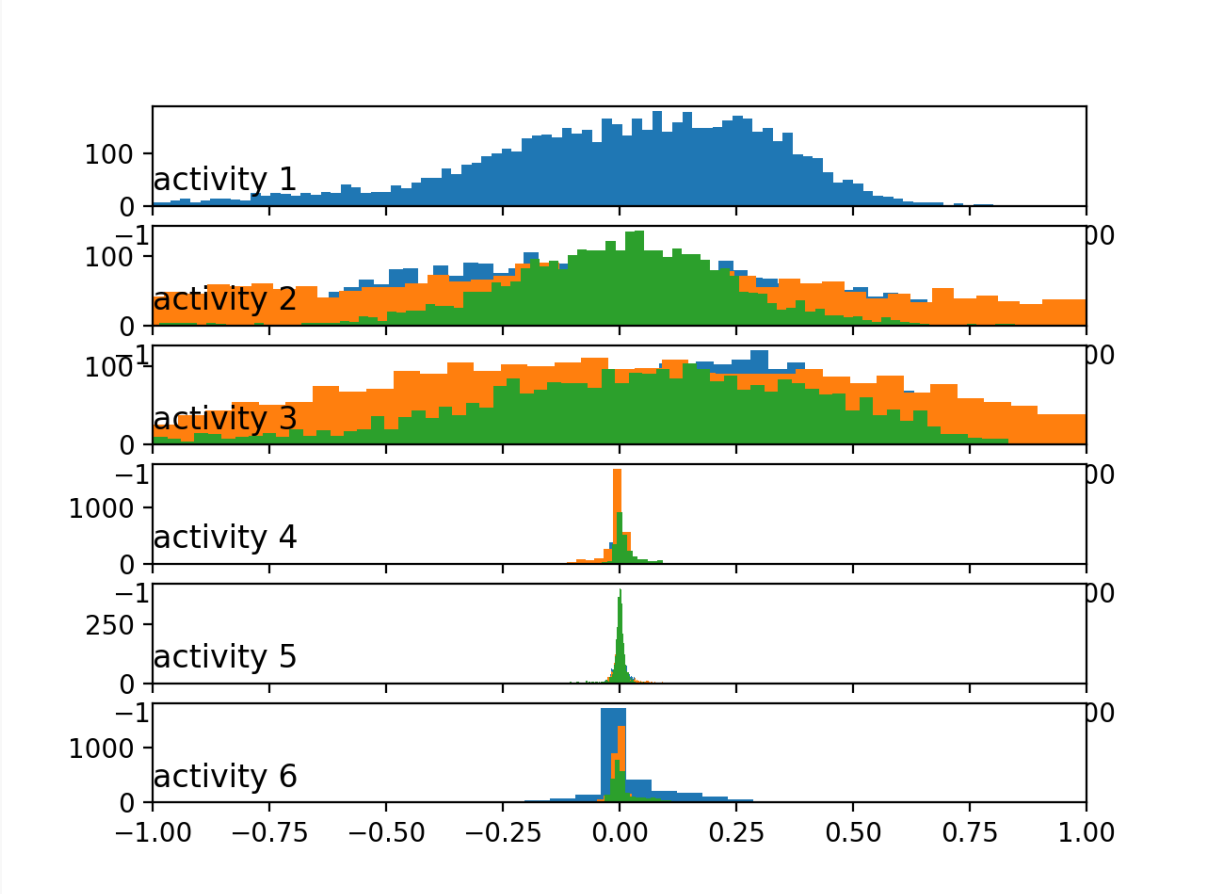
Histograms of the body acceleration data by activity

Finally, we can update the example one more time to plot the histograms per activity for the gyroscopic data.

The updated function is listed below.

```
def plot_activity_histograms(X, y):
	# get a list of unique activities for the subject
	activity_ids = unique(y[:,0])
	# group windows by activity
	grouped = data_by_activity(X, y, activity_ids)
	# plot per activity, histograms for each axis
	pyplot.figure()
	xaxis = None
	for k in range(len(activity_ids)):
		act_id = activity_ids[k]
		# total acceleration
		for i in range(3):
			ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
			ax.set_xlim(-1,1)
			if k == 0:
				xaxis = ax
			pyplot.hist(to_series(grouped[act_id][:,:,6+i]), bins=100)
			pyplot.title('activity '+str(act_id), y=0, loc='left')
	pyplot.show()
```

Running the example creates plots with the similar pattern as the body acceleration data, although showing perhaps fat-tailed Gaussian-like distributions instead of bimodal distributions for the in-motion activities.



Histograms of the body gyroscope data by activity

All of these plots were created for the first subject, and we would expect to see similar distributions and relationships for the movement data across activities for other subjects.

Classify Activity-

The classification of activity based on various algorithm differs based on accuracy constrains. We will discuss four models applied for this data set and compare the results to pursue the most suitable choice.

In this experiment we have used four different algorithms – Support Vector Classifier (SVC), Logistic Regression (LR), K Nearest Neighbour (KNN) and Random Forest (RF). We have converted the accuracy score to percentage by multiplying the output with 100 for conversion.

Algorithm	***Accuracy (%)***
SVC	94.02782490668477
LR	96.19952494061758
KNN	90.02375296912113
RF	89.68442483881914

So, we can see that the algorithm of Logistic regression reaches the most accuracy among the other models which is around 96%. The comparison data is visualized in the above table.

11. RECOMMENDER SYSTEM

Based on the activity monitored by the wearable device, we can track what a particular person is doing for a period of time and what is his interest. We can recommend something based on that activity to that person when the person is apparently idle. For example, you can recommend a person for watching movies, or listening to songs or to do exercise when the person is idle for a long time.

This kind of recommendation can be achieved by using data collected from various users and we can analyse the trending activities, which others are doing. Mostly people from same region would like to watch similar kind of movies available for release. So, for recommendation to a single person, both that person's data and the other user's activity data is important. Forming neural network to model this kind of recommendation system is a highly challenging scenario. We will discuss the methodology, classification and process for implementing such a recommender system.

Basically, there are to different kind of filtering techniques for recommender system-

Content-based Filtering- In this filtering process we can generate a list of properties for certain activities. If an individual's activity matches with the certain category, then we can recommend that person an activity. So, let's take an example-

	Waking	Sitting	Sleeping
Hand oriented	No	No	No
Non-hand Oriented	Yes	Yes	No
Both hand and non-hand oriented	Yes	Yes	No

The above table is classifying three basic activity of a person like walking, sitting and sleeping based on the engagement of hand orientation. So, if a person if more active in hand-oriented work, then we can suggest him/her to do more non-hand-oriented work to maintain a balance between all parts of body, and may suggest to do a little exercise to stay fit. We can rank activities for a person's interest based on the record of his past activity.

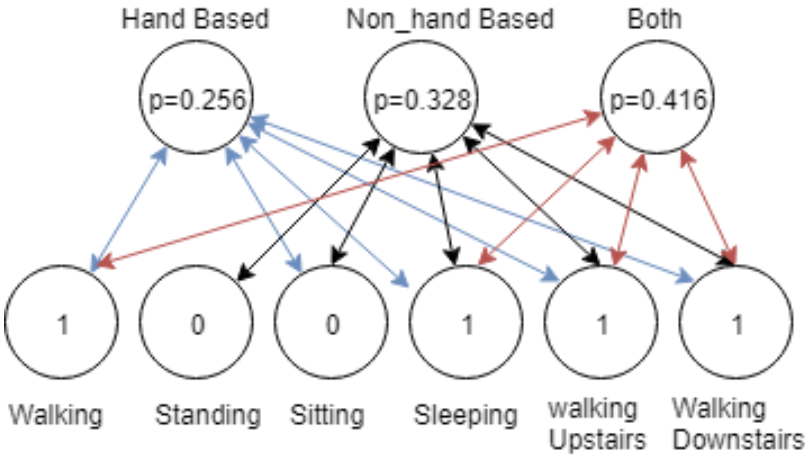
Collaborative Filtering- This kind of filtering is based on the other user activity data. Surveying certain activities liked by an individual, we can predict similar interest for another person. Different places will generate this data in different ways. We can also classify the data using the time and date perspective. Also, there is another possibility for filtering age-based activity. Because in certain ages people grows interest on certain things. Like a teenager is more likely to be watching Netflix than a news channel.

	Waking	Sitting	Sleeping
User 1	Yes	No	No
User 2	No	Yes	No
User 3	Yes	No	Yes
User 4	Yes	Yes	No
	3 Vote	2 Vote	1 Vote

Let's take another example, here we can see that the activities are classified based on user. User 1, User 3 and User 4 has recorded activity of walking. But User 2 is likely to be sitting most of the time. Based on that perspective our system might recommend 'user 2' to go for a walk.

Restrictive Boltzmann Machine-

The Restrictive Boltzmann Machine is a learning model notably applicable for such a scenario we are implementing here. The RBM is a simple neural network consisting of two layers- one is the visible or input layer and the other is the hidden layer.



Here is a simple example calculated probability based on 'hand' or 'non-hand-oriented' activity applied on six recorded activity data of an individual. As a result, we can see here that the probability for both usage of hand or non-hand orientation is the highest with a value of p=0.416 for this individual. So, it's obvious to suggest that person to do something which involves both Hand and Non-hand-oriented postures. Now, it is also required to keep on note that, if a particular person is working too hard for a long time, then the recommendation system should suggest to take rest or to relax to that user.

There are lots of algorithms tested and researched with certain equations and functions. We would like to find out better solution for implementing such functionality with lesser work load and computational power, so that it is easy to implement in smart phone like devices. So, the result will be faster and more local.

7. FUTURE ENHANCEMENTS

Smart wearable device undoubtedly one of the most trending technological advancement in the modern world. By the invention of foldable screen technology, which is more applicable devices which are being wear on an uneven surface like wrists. It is also being targeted to be more durable, cheap and feature rich.

The ideology for implementing such a recommender system based on the collected data from users, is a unique and futuristic concept. We can promote such system for product advertisement and business prospects. By applying this recommendation system to the world-wide market place and online business it can be made more flexible and easier for user interaction.

This proposed work is in its earlier stage. We will be keep working for adding more functionalities and better logical implementation so that the whole world can utilise this as a acceptable option, and not only for personal usage but also for business reasons.

By adding more advanced sensor and faster cloud technologies the system would be more capable of delivering most awaited features to customers.

8. CONCLUSION

In this paper we have introduced a smart wearable device implementation using IoT cloud platform and then applying machine learning algorithm we are recognising activity for certain individuals. Adding recommender system to that data output we are able to recommend activities to users.

Due low-cost implementation we were unable to collect raw data which is suitable for certain machine learning model. The more the data the better output for this activity recognition system. Despite we have found out certain flaws in the system which must be improved in later research and implementation.

(a) The acceleration and gyroscope data must be accurate and easy to record using API.
(b) Sometimes the data is not being loaded properly into the system database.
(c) The intelligent model in cloud is unable to distinguish activity performed in certain aspects, like sitting inside room vs sitting inside a car.
(d) The moving data is inaccurate to distinguish between walking and jogging.
(e) Location based classification is not possible because no GPRS system is implemented.

We would like to disqualify this kind of issues by upgrading requirements for the system and by using advanced devices for the system.

This proposed model has huge possibilities for future enhancements and including more features or functionalities can be possible by adding more sensors into it. We are aware about the fact that big business companies are investing a chunk amount of money for this kind of recommendation system to grow their business and reach out more user. Even 10% revenue in the business aspects can bring huge profit in overall.

REFRENCES

[1] Lloyd C.: Time Comes for the Superwatch, The Times, Newspaper, 3 Thomas More Square, London, E98 1XY, 13. Jan. 199

[2] Sensor requirements for activity recognition on smart watches, Conference: Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments, May 2013

[3] Smartwatch-based Activity Recognition: A Machine Learning Approach, Gary M. Weiss, Jessica L. Timko, Catherine M. Gallagher, Kenichi Yoneda, and Andrew J. Schreiber

[4] SI - International Bureau of Weights and Measures, The International System of Units (SI), Brochure (8th ed.), 2006, ISBN 92-822-2213-6

[5] Bieber G., Voskamp J., Urban B., Activity Recognition for Everyday Life on Mobile Phones, Lecture Notes in Computer Science, Volume 5615/2009, pp.289-296, Springer Berlin / Heidelberg, 2009, ISSN 0302-9743

[6] I. H. Witten, and E. Frank - Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann, 2005.

[7] I. H. Witten, and E. Frank - Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann, 2005.,

[8] Anguita, Davide, Ghio, Alessandro Oneto, Luca Parra Perez, Xavier Reyes Ortiz, Jorge Luis - A public domain dataset for human activity recognition using smartphones, ISBN978-2-87419-081-0, 2013

[9] Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine –by Davide AnguitaAlessandro GhioLuca OnetoXavier ParraJorge L. Reyes-Ortiz, IWAAL 2012: Ambient Assisted Living and Home Care pp 216-223

[10] Dataset- Human Activity Recognition with Smartphones Recordings of 30 study participants performing activities of daily living. https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

[11] Wearable Multisensor Heart Rate Monitor, Grajales, L., & Nicolaescu, I. V. (n.d.). Wearable Multisensor Heart Rate Monitor. International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06). doi:10.1109/bsn.2006.58

[12] J. Yang, "Toward physical activity diary: Motion recognition using simple acceleration features with mobile phones," In First Int. Workshop on Interactive Multimedia for Consumer Electronics at ACM Multimedia, pp. 1-10, 2009

[1]

[2] Daniel D. 2013. Privacy Implications of Google Glass. (cited 21 Oct, 2015).[Online] Available: http://resources.infosecinstitute.com/privacy-implications-of-google-glass/

[3] Marc R. (17 Jul, 2013). Hacking the Internet of Things for Good. (cited 19 Oct,2015).[Online] Available: https://blog.lookout.com/blog/2013/07/17/hacking-the-internet-of-things-for-good/

[4] Candid W. (18 Jul, 2013). Google Glass Still Vulnerable to WiFi Hijacking Despite QR Photobombing Patch. (cited 21 Oct, 2015).[Online] Available: http://www.symantec.com/connect/blogs/google-glass-still-vulnerable-wifi-hijacking-despite-qr- photobombing-patch

[5] [fitbit. (cited 21 Oct, 2015).[Online] Available: https://www.fitbit.com/my

[6] Michael S. (11 Jun, 2015). Internet of Things Security Evaluation of nine Fitness Trackers. (cited 21 Oct, 2015).[Online] Available: https://www.av-test.org/fileadmin/pdf/avtest_2015- 06_fitness_tracker_english.pdf

[7] Rahman, M., B. Carbunar, and M. Banik, Fit and vulnerable: Attacks and defenses for a health monitoring device. arXiv preprint arXiv:1304.5672, 2013.

[19]J acob B. (03 Aug, 2015). Surveillance Society: Wearable fitness devices often carry security risks. (cited 21 Oct, 2015).[Online] Available: http://www.post-gazette.com/news/surveillance- society/2015/08/03/Surveillance-Society-Wearable-fitness-devices-often-carry-security- risks/stories/201508030023

[20] Cyr, B., et al., Security Analysis of Wearable Fitness Devices (Fitbit). Massachusets Institute of Technology, 2014.

[21] Carly P. (24 May, 2015). iPhone users' privacy at risk due to leaky Bluetooth technology. (cited 24 Oct, 2015).[Online] Available: http://www.v3.co.uk/v3-uk/news/2409939/iphone-users-privacy-at- risk-due-to-leaky-bluetooth-technology

[22] Kristi R. (22 Jul, 2015). HP Study Reveals Smartwatches Vulnerable to Attack. (cited 4 Oct, 2015).[Online] Available: http://www8.hp.com/us/en/hp-news/press-release.html?id=2037386#.Vi18G7crLIU

[23] Liviu A. (12 Sep, 2014). Bitdefender Research Exposes Security Risks of Android Wearable Devices. (cited 24 Oct, 2015).[Online] Available: http://www.darkreading.com/partner-perspectives/bitdefender/bitdefender-research-exposes-security-risks-of-android-wearable-devices-
/a/d-id/1318005

[24] Ryan G. (01 Oct, 2013). Accelerometer vs. Gyroscope: What's the Difference? (cited 23 Oct, 2015).[Online] Available: http://www.livescience.com/40103-accelerometer-vs-gyroscope.html

[25] Indian Institute of Technology Kanpur Commonwealth of Learning Vancouver. 2013. SENSORS ON ANDROID PHONES. (cited 23 Oct, 2015).[Online] Available: http://m4d.colfinder.org/sites/default/files/Slides/M4D_Week2_sensors.pdf

[26] Engineer's Handbook. 2006. Mechanical Components - Sound Sensors. (cited 2 Oct,2015).[Online]Available:http://www.engineershandbook.com/Components/soundsensors.html

[27] Technavio. (21 Jul 2014). Exploring Five Challenges in the Wearable Technology Market. (cited 31 Oct, 2015). [Online] Available: http://www.technavio.com/blog/exploring-five-challenges-in-the- wearable-technology-market

[28] Julie F. (12 Nov, 2014). ISACA Survey: Most Consumers in Australia Aware of Major Data Breaches, But Fewer Than Half Have Changed Key Shopping Behaviors. (cited 4 Oct, 2015).[Online] Available: http://www.isaca.org/About-ISACA/Press-room/News-Releases/2014/Pages/ISACA- Survey-Most-Consumers-in-Australia-Aware-of-Major-Data-Breaches-But-Fewer-Than-Half-Have-
Changed-Shopping-Behaviors.aspx

[29] Nroseth. (27 Mar, 2015). Data Security in a Wearables World. (cited 4 Oct, 2015).[Online]
Available: http://www.swatsolutions.com/data-security-in-a-wearables-world/

[30] Vangie B. cloud. (cited 4 Oct, 2015).[Online] Available: http://www.webopedia.com/TERM/C/cloud.html

[31] David E. Sanger and Nicole P. (14 Feb 2015). Bank Hackers Steal Millions via Malware. (cited 17 Oct, 2015).[Online] Available: http://www.nytimes.com/2015/02/15/world/bank-hackers-steal- millions-via-malware.html?_r

[32] Michael C. Wearables security: Do enterprises need a separate WYOD policy? (cited 17 Oct, 2015).[Online] Available: http://searchsecurity.techtarget.com/answer/Wearables-security-Do-enterprises-need-a-separate-WYOD-policy

[33] Mellisa T. (May 30, 2013). 4 Security Challenges for Fitbit, Google Glass + Other Wearable Devices. (cited 4 Oct, 2015).[Online] Available: http://siliconangle.com/blog/2013/05/30/4-security- challenges-for-fitbit-google-glass-other-wearable-devices/

[34] Kristi R. (22 Jul, 2015). HP Study Reveals Smartwatches Vulnerable to Attack. (cited 4 Oct, 2015).[Online] Available: http://www8.hp.com/us/en/hp-news/press-release.html?id=2037386#.Vi18G7crLIU

[35] Eric Z. (14 May, 2015). Apple Watch, Android Wear Lack Theft Protection. (cited 17 Oct, 2015). [Online] Available: http://www.informationweek.com/it-life/apple-watch-android-wear-lack-theft- protection/a/d-id/1320430

[36] Apadmi. Apadmi's Wearable Tech Study:Do Potential Customers Think Wearable Tech Poses a Privacy Risk? (cited 20 Oct, 2015).[Online] Available: http://www.apadmi.com/wearable-technology- trends/wearable-tech privacy/#WTP-2

[37] Motti, V. and K. Caine, Users' Privacy Concerns About Wearables, in Financial Cryptography and Data Security, M. Brenner, et al., Editors. 2015, Springer Berlin Heidelberg. p. 231-244.

[38] Charles A. (01 May, 2013). Google Glass security failings may threaten owner's privacy. (cited 20 Oct, 2015).[Online] Available: http://www.theguardian.com/technology/2013/may/01/google-glass- security-privacy-risk

[39] Michalevsky, Y., D. Boneh, and G. Nakibly. Gyrophone: Recognizing speech from gyroscope signals. in Proc. 23rd USENIX Security Symposium (SEC'14), USENIX Association. 2014.

[40] Lisa E. (09 Oct, 2014). A New Wave Of Gadgets Can Collect Your Personal Information Like Never Before. (cited 22 Oct, 2015).[Online] Available: http://www.businessinsider.my/privacy-fitness- trackers-smartwatches-2014-10/#GDuZGvtShqZO79S5.97

[41] Raij, A., et al., Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2011, ACM: Vancouver, BC, Canada. p. 11-20

69. Rumelhart D.E., Hinton G.E., Williams R.J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1. MIT Press; Cambridge, MA, USA: 1986. Learning Internal Representations by Error Propagation; pp. 318–362.

70. Le Q.V., Zou W.Y., Yeung S.Y., Ng A.Y. Learning Hierarchical Invariant Spatio-temporal Features for Action Recognition with Independent Subspace Analysis; Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11); Colorado Springs, CO, USA. 20–25 June 2011; pp. 3361–3368.

71. Dean J., Corrado G., Monga R., Chen K., Devin M., Mao M., Ranzato M., Senior A., Tucker P., Yang K., et al. Advances in Neural Information Processing Systems. Volume 25. NIPS Foundation; La Jolla, CA, USA: 2012. Large Scale Distributed Deep Networks; pp. 1232–1240.

72. Deng L., Yu D. Deep Learning: Methods and Applications. Found. Trends Signal Process. 2014;7:197–387. doi: 10.1561/2000000039. [CrossRef]

73. Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G.S., Davis A., Dean J., Devin M., et al. TensorFlow: Large-Scale Machine Learning on Heterogeneus Distributed Systems. Google Research; Mountain View, CA, USA: 2015. Technical Report.

74. Allwein E.L., Schapire R.E., Singer Y. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. J. Mach. Learn. Res. 2001;1:113–141.

75. Landgrebe T., Duin R.P.W. Approximating the multiclass ROC by pairwise analysis. Pattern Recognit. Lett. 2007;28:1747–1758. doi: 10.1016/j.patrec.2007.05.001. [CrossRef]

76. Arif M., Kattan A. Physical Activities Monitoring Using Wearable Acceleration Sensors Attached to the Body. PLoS ONE. 2015;10:e0130851 doi: 10.1371/journal.pone.0130851. [PMC free article] [PubMed] [CrossRef]

77. Gupta P., Dallas T. Feature Selection and Activity Recognition System Using a Single Triaxial Accelerometer. IEEE Trans. Biomed. Eng. 2014;61:1780–1786. doi: 10.1109/TBME.2014.2307069. [PubMed] [CrossRef]

78. Shoaib M., Scholten H., Havinga P. Towards physical activity recognition using smartphone sensors; Proceedings of the 10th IEEE International Conference on Ubiquitous Intelligence and Computing and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC); Vietri sul Mare, Italy. 18–21 December 2013; pp. 80–87.

79. Anguita D., Ghio A., Oneto L., Parra X., Reyes-Ortiz J.L. A Public Domain Dataset for Human Activity Recognition Using Smartphones; Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013; Bruges, Belgium. 24–26 April 2013.

80. Gyllensten I., Bonomi A. Identifying Types of Physical Activity With a Single Accelerometer: Evaluating Laboratory-trained Algorithms in Daily Life. IEEE Trans. Biomed. Eng. 2011;58:2656–2663. doi: 10.1109/TBME.2011.2160723. [PubMed] [CrossRef]

81. Khan A.M., Lee Y.K., Lee S.Y., Kim T.S. A Triaxial Accelerometer-based Physical-activity Recognition via Augmented-signal Features and a Hierarchical Recognizer. IEEE Trans. Inf. Technol. Biomed. 2010;14:1166–1172. doi: 10.1109/TITB.2010.2051955. [PubMed] [CrossRef]

82. Bao L., Intille S.S. Pervasive Computing. Volume 3001. Springer; Berlin, Germany: 2004. Activity recognition from user-annotated acceleration data; pp. 1–17. Lecture Notes in Computer Science.

83. Kwapisz J.R., Weiss G.M., Moore S.A. Activity Recognition Using Cell Phone Accelerometers. ACM SIGKDD Explor. Newsl. 2011;12:74–82. doi: 10.1145/1964897.1964918. [CrossRef]