

**MEMS BASED SYSTEM FOR CONTINUOUS
HEALTHCARE MONITORING**

Dissertation submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

JATIN (121035)

DEEPANSH GOYAL (121036)

SASHANK KUMAR (121053)

UNDER THE GUIDANCE OF

MR. MUNISH SOOD

(ASSISTANT PROFESSOR)

to



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May, 2016

TABLE OF CONTENTS

DECLARATION BY THE SCHOLAR	iv
SUPERVISOR’S CERTIFICATE	v
ACKNOWLEDGEMENT	vi
LIST OF ACRONYMS AND ABBREVIATIONS	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
ABSTRACT	xiii
CHAPTER-1	
INTRODUCTION	1
1.1 HEALTHCARE MONITORING	1
1.2 MOTIVATION	2
1.3 OBJECTIVES	3
CHAPTER-2	
HARDWARE DESCRIPTION	7
2.1 MEMS	7
2.1.1 WHAT IS MEMS TECHNOLOGY?	7
2.1.2 COMPONENTS OF MEMS	9
2.2 ACCELEROMETER	11
2.2.1 PRINCIPLES OF OPERATION	11
2.2.2 TYPES OF ACCELEROMETER	12
2.2.3 ACCELEROMETER SPECIFICATIONS	12
2.3 ADXL345	13
2.3.1 THEORY OF OPERATION	14
2.3.2 APPLICATIONS	14
2.4 ARDUINO	15
2.4.1 FEATURES	16
2.4.2 WHY ARDUINO?	16
2.5 BLUETOOTH	17
2.5.1 BLUETOOTH MODULE HC-05	17
2.5.2 APPLICATIONS	19
2.6 PULSE SENSOR	19
CHAPTER-3	
PROTOCOLS	21
3.1 SPI	21

3.2	I2C	22
	3.2.1 I2C INTERFACE	23
	3.2.2 I2C ADDRESSES	23
	3.2.3 I2C PROTOCOL	23
3.3	SERIAL v/s PARALLEL	25
CHAPTER-4		
ALGORITHMS		26
4.1	STEP DETECTION	26
	4.1.1 ALGORITHM	26
4.2	FREE FALL ALGORITHM	28
4.3	TAP DETECTION ALGORITHM	29
4.4	PULSE SENSOR ALGORITHM	32
CHAPTER-5		
WEB APPLICATION		34
5.1	NODE.JS	34
	5.1.1 V8	35
	5.1.2 PACKET MANAGEMENT	35
5.2	SOCKET.IO	36
5.3	WEB SERVER	37
	5.3.1 WEB APPLICATION ARCHITECTURE	38
	5.3.2 CREATING WEB SERVER USING NODE	38
5.4	FILE SYSTEM MODULE	39
5.5	WEB PORTAL	40
CHAPTER-6		
READINGS		42
6.1	STEP DETECTION	42
6.2	TAP	43
	6.1.1 SINGLE TAP	45
	6.1.2 DOUBLE TAP	45
6.3	FREE FALL	46
6.4	PULSE SENSOR	47
FUTURE SCOPE AND LIMITATIONS		48
CONCLUSION		49
REFERENCES		50

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the B.Tech thesis entitled “**MEMS Based System For Continuous Healthcare Monitoring**” submitted at **Jaypee University of Information Technology, Wagnaghat India**, is an authentic record of my work carried out under the supervision of **Mr. Munish Sood**. I have not submitted this work elsewhere for any other degree or diploma.

JATIN : _____

DEEPANSH GOYAL : _____

SASHANK KUMAR : _____

Department of Electronics and Communication Engineering

Jaypee University of Information Technology, Wagnaghat

May, 2016

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech. thesis entitled “**MEMS Based System For Continuous Healthcare Monitoring System**”, submitted by **Jatin(121035), Deepansh Goyal(121036), Sashank Kumar(121053)** at **Jaypee University of Information Technology, Wagnaghat, India**, is a bonafide record of their original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

Mr. Munish Sood

(Assistant Professor)

Department of Electronics and Communication Engineering

Jaypee University of Information Technology, Wagnaghat

May, 2016

ACKNOWLEDGEMENT

We are very grateful and highly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our guide Mr. Munish Sood, Department of electronics and Communication Engineering, Jaypee University of Information Technology Waknaghat, in bringing this report to a successful completion.

We are grateful to Prof. Sunil Bhooshan, Head of the Department of Electronics and Communication Engineering, for permitting us to make the use of the facilities available in the department to carry out the project successfully.

We are also grateful to Mr. Mohan Sharma and Mr. Manoj Kumar Pandey, in providing us support in lab and help in solving our both hardware and software issues.

Last but not the least we express our sincere thanks to all of our teachers and friends who have patiently extended all sorts of help for accomplishing this undertaking.

Date: _____

Jatin : 121035

Deepansh Goyal : 121036

Sashank Kumar : 121053

LIST OF ACRONYMS AND ABBREVIATIONS

3G	Third Generation
AC	Alternating Current
AHF	Adaptive Frequency Hopping
API	Application Program Interface
AVR	Advanced Virtual RISC
BICMOS	Bipolar Complementary Metal Oxide Semiconductor
BPM	Beats Per Minute
BS	Base Station
CMOS	Complementary Metal Oxide Semiconductor
CPU	Control And Processing Unit
CS	Chip Select
DC	Direct Current
EDR	Enhanced Data Rate
EEPROM	Electrically Erasable Programmable Read Only Memory
FIFO	First In First Out
GPS	Global Positioning System
GSM	Global System For Mobile
GUI	Graphical User Interface
HDD	Hard Disk Drive
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I2C	Inter Integrated Chip
IC	Integrated Chip

ICSP	In-Circuit System Programming
IDE	Integrated Development Environment
ISM	Industrial, Scientific And Medicinal
ITC	Information Technology & Communication
LDR	Light Dependant Resistor
LED	Light Emitting Diode
LSB	Least Significant Bit
MCU	Micro Controller
MEMS	Micro Elecro Mechanical System
MIME	Multi-Purpose Internet Mail Extensions
NXP	Next Experience
OPAMP	Operational Amplifier
OS	Operating System
PAN	Personal Area Network
PC	Personal Computer
PIO	Programmable Input Output
PPG	Photoplethysmograph
PWM	Pulse Width Modulation
SCL	Serial Clock
SDA	Serial DATA
SPI	Serial Peripheral Interface
SPO2	Peripheral Capillary Oxygen Saturation
SPP	Serial Port Protocol
SRAM	Static Random Access Memory
TCP	Transmission Control Protocol

UART	Universal Asynchronous Receiver Transmitter
UHF	Ultra High Frequency
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF	Unicode Transformation Format
WBSN	Wireless Base Station Network
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Network

LIST OF TABLES

TABLE NO.	DESCRIPTION	PAGE NO.
1.1	Biosensors and Bio-signals	3
6.1	Single Tap Readings	45

LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE NO.
1.1	Architecture for proposed healthcare system in Hospital	5
1.2	Flowchart of proposed project	6
2.1	Elements of MEMS	8
2.2	A surface micromachined resonator fabricated by the MNX	8
2.3	Piezoelectric Accelerometer	11
2.4	ADXL345 pin diagram	13
2.5	ADXL345 Architecture	14
2.6	Arduino	15
2.7	Interfacing of Bluetooth module (HC-05) with Arduino	18
2.8	Pulse Sensor	20
3.1	SPI connection	21
3.2	SPI header	22
3.3	I2C Connection	22
3.4	I2C Wire Architecture	24
4.1	Step Variation Graph	26
4.2	Smoothing Filter	27
4.3	Step Detection	27
4.4	Number of steps	28
4.5	Free Fall Algorithm	29
4.6	Tap Detection	30
4.7	Invalid Double Tap if acceleration is found greater than THRESH TAP register in Latency window	31
4.8	Invalid Double Tap	31
4.9	PPG Plot	33
5.1	Web Application Architecture	38
5.2	Patient Detail Form	40

5.3	Monitoring Window	41
5.4	Monitoring window showing patient details	41
6.1	A step detection	42
6.2	5 steps taken as there are 5 peaks.	42
6.3	Readings of x axis	43
6.4	Readings of y axis	44
6.5	Readings of z axis	44
6.6	Single tap readings	45
6.7	Double tap	46
6.8	Freefall readings	46
6.9	Pulse Measure	47

ABSTRACT

In a hospital, health care monitoring system is necessary to constantly monitor the patient's physiological parameters. This project presents a monitoring system that has the capability to monitor physiological parameters from a patient body. In the proposed system, a coordinator node is attached on patient's body to collect all the signals from the wireless sensors and send them to the base station. The attached sensors on patient's body form a wireless body sensor network (WBSN) and they are able to sense the pulse rate, activity of the patient and so on. This system can detect the abnormal conditions, issues an alarm and sends a report to the physician. This system can be used in multi-patient architecture for hospital healthcare.

CHAPTER 1

INTRODUCTION

1.1 HEALTHCARE MONITORING

Health is like money, we never have its true idea of its value until we lose it. People live in different climatic conditions in different parts of the world. Medicines of various kinds such as antibiotics, ayurvedic medicines are used to cure different diseases which have different impact on human beings.

Wireless sensor networks in healthcare are used in different kinds of applications. Life expectancy increases with advancement in technologies, but chronic diseases are increasing dramatically. Digital healthcare leads to direct contact between clinician and patient and records their data in electronic healthcare records.

Remote wireless healthcare monitoring system is seen as an effective method of providing immediate care as it allows for continuous as well as emergency transmission of patient's information to the doctor or healthcare providers. Remote patient monitoring will not only redefine hospital care but also work, home, and recreational activities. These new technologies enable us to monitor patients on a regular basis, replacing the need to frequently visit the local doctor for a recurring illness. The treatment and diagnosis of the large crowd effectively along with summarization and aggregation of the data is very difficult and burden. So it may be possible that output contains some errors or redundancy. Finally then introduce the ITC in order to reduce the distance and time gap between the patient and health service provider or physician. IT technologies such as Wi-Fi, 3G and WSN provide a way for remote monitoring of the patients. Healthcare services can improve the living of patients, physically handicapped, elders, and chronically ill patients. So without frequent visit to the doctor, anyone can get proper treatment and improve his/her health conditions. In recent years, many e-Health systems have been developed and they are providing the remote monitoring of the patient. In the MEMS Based System For Continuous Healthcare Monitoring, we designed an integrated patient monitoring device with low cost which is used to continuously monitor the patient's health condition, for effective and accurate measurement of the patient's physiological parameters such as pulse rate of the patient, movement. This system sends the health information of a patient to the microcontroller, then send to the PC via a communicating

device. PC stores this data onto the local database. If there is any emergency, it displays an alert message on the screen and send a notification to the doctor. This new system provides the full utilization of the medical staff and other medical resources. The doctor, nurses and administrator can view the patient's profile that contains patient history etc. Based on the different privileges, the doctor, nurses or other medical staff can add, delete, and modify the patient profile. The administrator can update doctors and nurses profile in central server.

1.2 MOTIVATION

In past years, the people die due to lack of treatments and availability of health monitoring devices and doctors. Most of the countries in the world are facing this type of problems. There are number of systems which can provide health care services but they have some limitation such as high cost, lack of patient data and high communicational and computational overhead. According to the World Health Organization, the probability of dying between 15 and 60 years of age in male/female (per 1000 population) in India is nearly 250/169. In present years, the chronic diseases are introduced in the world, due to the changes in the environment.

In order to avoid existing problem, wearable health-monitoring systems have drawn a lot of attention from the research community and the industry during the last decade as it is pointed out by the numerous and yearly increasing corresponding research and development efforts. As healthcare costs are increasing and the world population is ageing, there has been a need to monitor a patient's health status while he is in the hospital also as it is very difficult to have a nurse all the time with the patient. To address this demand, a variety of system prototypes and commercial products have been introduced in the course of recent years, which aim at providing real-time feedback information about one's health condition, either to the user himself or to a medical centre or straight to a supervising professional physician, while being able to alert the physician in case of possible imminent health threatening conditions. In addition to that, wearable health-monitoring system constitute a new means to address the issue of monitoring chronic diseases for elderly people, postoperative rehabilitation patients, and persons with special abilities.

Wearable systems for health monitoring may comprise various types of miniature sensors, wearable or even implantable. These biosensors are capable of measuring significant

physiological parameters like heart rate, blood pressure, body and skin temperature, oxygen saturation, respiration rate, electrocardiogram, etc. The obtained measurements are communicated either via a wireless or a wired link to a central node, for example, a microcontroller board, which may then in turn display the according information on a user interface or transmit the aggregated vital signs to a medical centre. The wearable medical system may encompass a wide variety of components: sensors, wearable materials, actuators, power supplies, wireless communication modules and links, control and processing units, interface for the user, software, and advanced algorithms for data extracting and decision making.

Type of Bio-signals	Type of Sensor	Description of Measured data
Body Movements	Accelerometer	Measurement of acceleration forces in the 3D space.
Pulse Rate	Pulse Sensor	Frequency of the cardiac cycle.
Oxygen Saturation	Pulse Oximeter	Indicates the oxygenation or the amount of oxygen that is being “carried” in a patient’s blood.

TABLE 1.1: Biosensors and Bio-signals

As it is made obvious from the previous discussion, wearable systems for health monitoring need to satisfy certain strict medical criteria while operating under several constraints and significant hardware resource limitations. More specifically, a wearable health-monitoring system design needs to take into account several wearability criteria, for instance, the weight and the size factor of the system need to be kept small and the system should not hinder any of the user’s movements or actions. In addition to that, the security and the privacy of the collected personal medical data must be guaranteed by the system, while power consumption needs to be minimized to increase the system’s operational lifetime. Finally, such systems need to be affordable to ensure wide public access to low-cost ubiquitous health-monitoring services.

1.3 OBJECTIVES

In this project we are focusing on continuous monitoring of routine activities of a patient and measure the pulse rate of patient at every activity, and immediately inform the doctors if something odd is found. A continuous health monitoring has become quite mandatory these days.

Normally it is difficult to keep track on abnormalities in heartbeat count for patient itself manually. The average heartbeat per minute for 25-year old ranges between 80-90 beats per minute while for a 60-year old it is typically between 60-70 beats per minute. Patient's are not well versed with manual treatment which doctors normally use for tracking the count of heartbeat. So there must be some device which would help patient to keep track on their health by themselves. There are various instruments available in market to keep track on internal body changes. But there are many limitations regarding their maintenance due their heavy cost, size of instruments, and mobility of patients.

To overcome these limitations a device is used to keep track on heartbeat of patient and it should be easy to use, portable, light weighted, small size etc so that, it gives freedom of mobility to patient. The devices which can be carried everywhere to keep track on patient's health. If any varied change takes place it is notified. This notification would help to take an appropriate action at an instance of a time. This would help patients in providing a immediate help. This would also help patient's concern to doctor to take an appropriate action at proper time.

Moreover, after getting all the data, in our proposed system, the patient's physiological signals are acquired by the sensors attached on the patient body, and are then transmitted to the remote base-station and also a PC for storing and analyzing. In this project, a ubiquitous healthcare prototype system for hospitals is designed. The concept of ubiquitous healthcare system is to place unobtrusive wireless sensors on a person's body to form a wireless network which can communicate the patient's health status with base station connected to the monitoring PC.

The architecture and application of the proposed system are in Figure 1.1. The system consists of three parts:

1. The WBSN includes sensors which are responsible for collecting the physiological signals from patient.
2. A BS (Base Station) which receives the relayed data and sends it to the PC through a cable.
3. The graphical user interface (GUI) which is responsible for storing, analyzing and presenting the received data in graphical and text format, and generate alert.

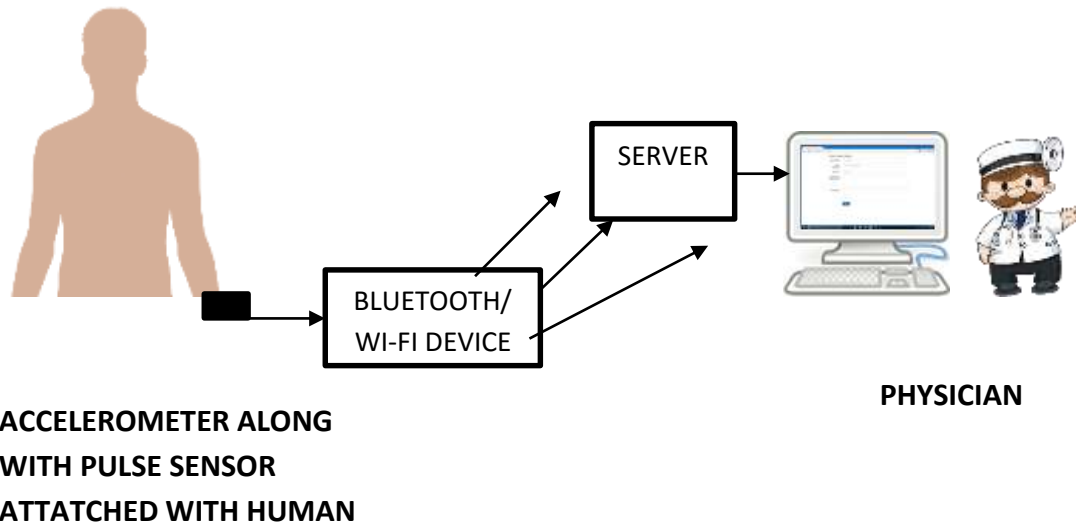


Figure 1.1: Architecture for proposed healthcare system in hospital

The key features of this project are:

1. Step Detection: To detect a step.
2. Gesture Control: To generate alert on double tap.
3. Free Fall Detection: To define the severity of fall based on impact and generate alert depending upon severity.
4. Pulse Measurement: To measure pulses of a patient at every activity defined above and match it with the standard and generate alert during mismatch.
5. Sending and retrieving data through Web Applications: Real time monitoring of a patient through web portal by using node.js, socket.io.

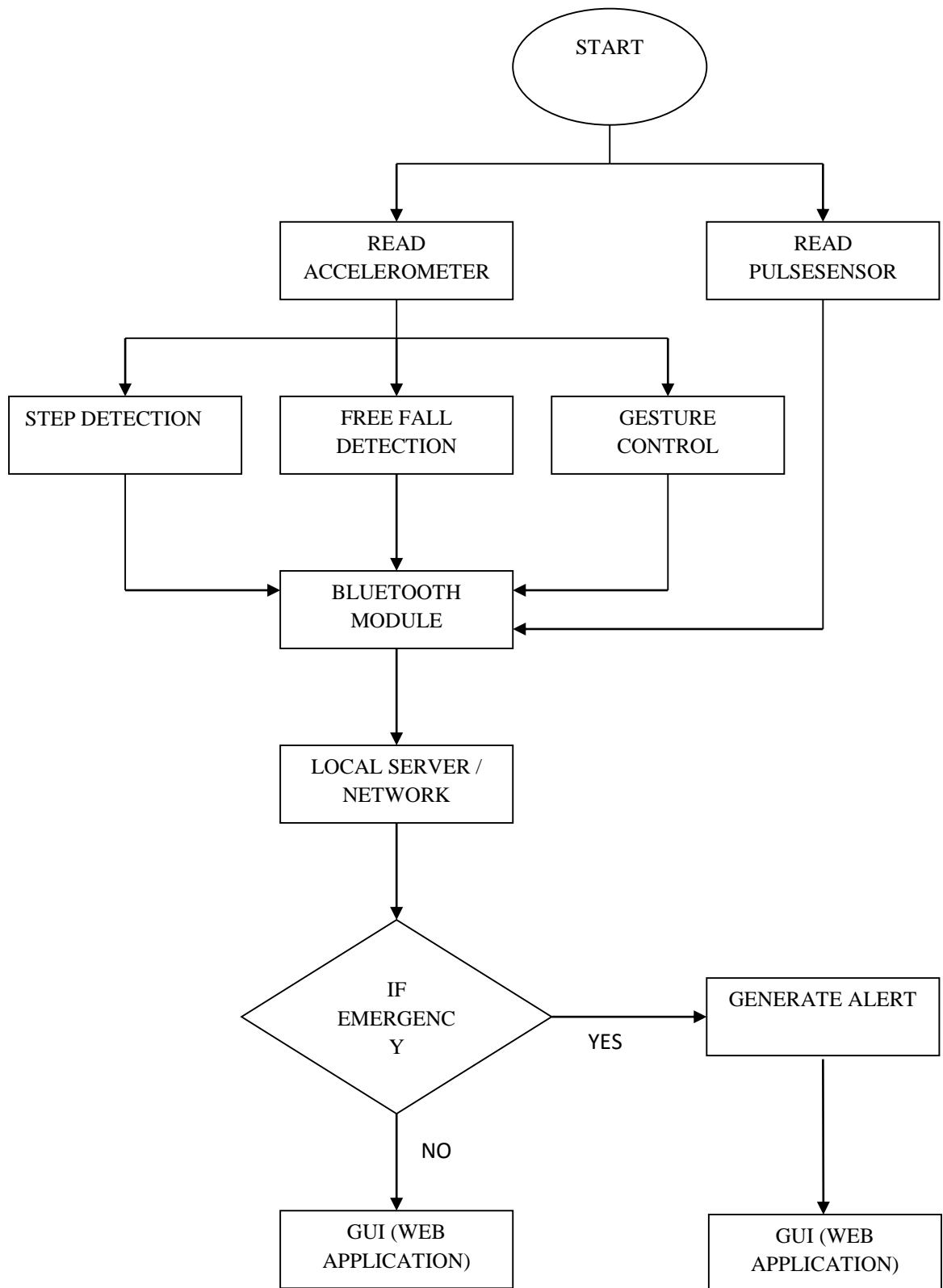


Figure 1.2: Flowchart of proposed project

CHAPTER 2

HARDWARE DESCRIPTION

2.1 MEMS

2.1.1 What is MEMS Technology?

Micro-Electro-Mechanical Systems, or MEMS, is a technology that in its most general form can be defined as miniaturized mechanical and electro-mechanical elements that are made using the techniques of micro fabrication. The critical physical dimensions of MEMS devices can vary from well below one micron on the lower end of the dimensional spectrum, all the way to several millimetres. Likewise, the types of MEMS devices can vary from relatively simple structures having no moving elements, to extremely complex electromechanical systems with multiple moving elements under the control of integrated microelectronics. The one main criterion of MEMS is that there are at least some elements having some sort of mechanical functionality whether or not these elements can move. The term used to define MEMS varies in different parts of the world, while in some other parts of the world they are called “Microsystems Technology” or “Micro machined devices”.

While the functional elements of MEMS are miniaturized structures, sensors, actuators, and microelectronics, the most notable elements are the microsensors and microactuators. Microsensors and microactuators are appropriately categorized as “transducers”, which are defined as devices that convert energy from one form to another. In the case of microsensors, the device typically converts a measured mechanical signal into an electrical signal.

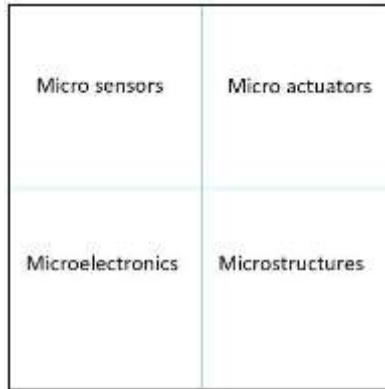


Figure 2.1: Elements of MEMS

The real potential of MEMS starts to become fulfilled when these miniaturized sensors, actuators, and structures can all be merged onto a common silicon substrate along with integrated circuits (i.e., microelectronics). While the electronics are fabricated using integrated circuit (IC) process sequences (e.g., CMOS, Bipolar, or BICMOS processes), the micromechanical components are fabricated using compatible "micromachining" processes that selectively etch away parts of the silicon wafer or add new structural layers to form the mechanical and electromechanical devices.

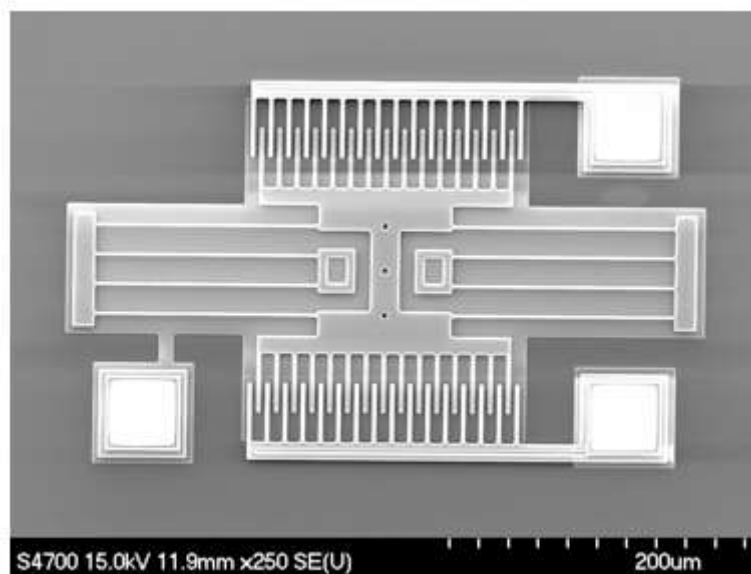


Figure 2.2 :A surface micromachined resonator fabricated by the MNX

This vision of MEMS whereby microsensors, microactuators and microelectronics and other technologies, can be integrated onto a single microchip is expected to be one of the most important technological breakthroughs of the future. This will enable the development of smart products by augmenting the computational ability of microelectronics with the perception and control capabilities of microsensors and

microactuators. Microelectronic integrated circuits can be thought of as the "brains" of a system and MEMS augments this decision-making capability with "eyes" and "arms", to allow microsystems to sense and control the environment. Sensors gather information from the environment through measuring mechanical, thermal, biological, chemical, optical, and magnetic phenomena. The electronics then process the information derived from the sensors and through some decision making capability direct the actuators to respond by moving, positioning, regulating, pumping, and filtering, thereby controlling the environment for some desired outcome or purpose. MEMS devices are manufactured using batch fabrication techniques, similar to ICs, unprecedented levels of functionality, reliability, and sophistication can be placed on a small silicon chip at a relatively low cost. MEMS technology is extremely diverse and fertile, both in its expected application areas, as well as in how the devices are designed and manufactured.

2.1.2 Components of MEMS

Micro sensors:

In the case of micro sensors, the device typically converts a measured mechanical signal into an electrical signal. Over the past several decades MEMS researchers and developers have demonstrated an extremely large number of micro sensors for almost every possible sensing modality including temperature, pressure, inertial forces, chemical species, magnetic fields, radiation, etc. Consequently, it is possible to not only achieve stellar device performance, but to do so at a relatively low cost level.

Micro Actuators:

Micro actuators are appropriately categorized as “transducers”, which are defined as devices that convert electrical energy to mechanical energy. including: micro valves for control of gas and liquid flows; optical switches and mirrors to redirect or modulate light beams; independently controlled micro mirror arrays for displays, micro resonators for a number of different applications, micro pumps to develop positive fluid pressures, micro flaps to modulate airstreams on air foils, as well as many others. These tiny actuators can perform mechanical feats far larger than their size would imply.

Microelectronics:

This vision of MEMS whereby micro sensors, micro actuators and microelectronics and other technologies, can be integrated onto a single microchip is expected to be one of the most important technological breakthroughs of the future. This will enable the development of smart products by augmenting the computational ability of microelectronics with the perception and control capabilities of micro sensors and micro actuators. Microelectronic integrated circuits can be thought of as the "brains" of a system and MEMS augments this decision-making capability with "eyes" and "arms", to allow microsystems to sense and control the environment. Sensors gather information from the environment through measuring mechanical, thermal, biological, chemical, optical, and magnetic phenomena. The electronics then process the information derived from the sensors and through some decision making capability direct the actuators to respond by moving, positioning, regulating, pumping, and filtering, thereby controlling the environment for some desired outcome or purpose. Furthermore, because MEMS devices are manufactured using batch fabrication techniques, similar to ICs, unprecedented levels of functionality, reliability, and sophistication can be placed on a small silicon chip at a relatively low cost. MEMS technology is extremely diverse and fertile, both in its expected application areas, as well as in how the devices are designed and manufactured. Already, MEMS is revolutionizing many product categories by enabling complete systems-on-a-chip to be realized.

Microstructures:

MEMS microstructures are manufactured in batch methodologies similar to computer microchips. The photolithographic techniques that mass-produce millions of complex microchips can also be used simultaneously to develop and produce mechanical sensors and actuators integrated with electronic circuitry. Most MEMS devices are built on wafers of silicon, adopting micromachining technologies from IC manufacturing and batch fabrication techniques.

2.2 Accelerometer

One of the most common inertial sensors is the **accelerometer**, a dynamic sensor capable of a vast range of sensing. Accelerometers are available that can measure acceleration in one, two, or three orthogonal axes. They are typically used in one of three modes:

- As an inertial measurement of velocity and position;
- As a sensor of inclination, tilt, or orientation in 2 or 3 dimensions, as referenced from the acceleration of gravity ($1\text{ g} = 9.8\text{m/s}^2$);
- As a vibration or impact (shock) sensor.

There are considerable advantages to using an analog accelerometer as opposed to an inclinometer such as a liquid tilt sensor – inclinometers tend to output binary information (indicating a state of on or off), thus it is only possible to detect when the tilt has exceeded some thresholding angle.

2.2.1 Principles of Operation

Most accelerometers are Micro-Electro-Mechanical Sensors (MEMS). The basic principle of operation behind the MEMS accelerometer is the displacement of a small proof mass etched into the silicon surface of the integrated circuit and suspended by small beams. Consistent with Newton's second law of motion ($\mathbf{F} = \mathbf{ma}$), as an acceleration is applied to the device, a force develops which displaces the mass. The support beams act as a spring, and the fluid (usually air) trapped inside the IC acts as a damper, resulting in a second order lumped physical system.

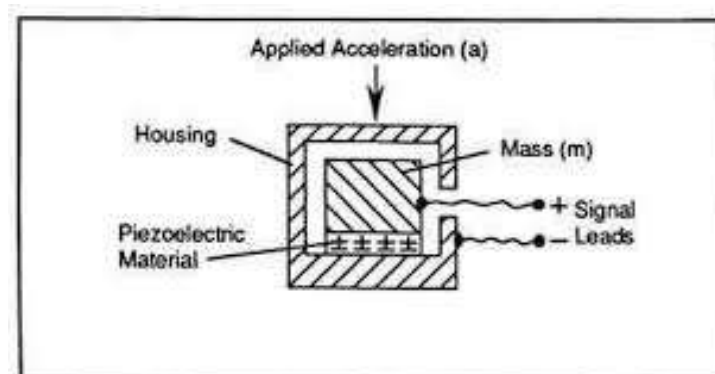


Figure 2.3: Piezoelectric Accelerometer

2.2.2 Types of Accelerometer

There are several different principles upon which an analog accelerometer can be built. Two very common types utilize capacitive sensing and the piezoelectric effect to sense the displacement of the proof mass proportional to the applied acceleration.

1. Capacitive

Accelerometers that implement capacitive sensing output a voltage dependent on the distance between two planar surfaces. One or both of these “plates” are charged with an electrical current. Changing the gap between the plates changes the electrical capacity of the system, which can be measured as a voltage output. This method of sensing is known for its high accuracy and stability. Capacitive accelerometers are also less prone to noise and variation with temperature.

2. Piezoelectric

Piezoelectric sensing of acceleration is natural, as acceleration is directly proportional to force. When certain types of crystal are compressed, charges of opposite polarity accumulate on opposite sides of the crystal. This is known as the piezoelectric effect. In a piezoelectric accelerometer, charge accumulates on the crystal and is translated and amplified into either an output current or voltage.

2.2.3 Accelerometer Specifications

1. Frequency Response – This parameter can be found out by analyzing the properties of the quartz crystal used and also the resonance frequency of the device.
2. Accelerometer Grounding – Grounding can be in two modes. One is called the Case Grounded Accelerometer which has the low side of the signal connected to their core. This device is susceptible to ground noise.
3. Resonant Frequency – It should be noted that the resonant frequency should be always higher the frequency response.
4. Temperature of Operation – An accelerometer has a temperature range between -50 degree Celsius to 120 degree Celsius. This range can be obtained only by accurate installment of the device.

5. Sensitivity – The device must be designed in such a way that it has higher sensitivity. That is, even for a small accelerative force, the electrical output signal should be very high. Thus a high signal can be measured easily and is sure to be accurate.
6. Axis – Most of the industrial applications requires only a 2-axis accelerometer. But if you want to go for 3D positioning, a 3-axis accelerometer will be needed.
7. Analog/Digital Output – You must take special care in choosing the type of output for the device. Analog output will be in the form of small changing voltages and digital output will be in PWM mode.

2.3 ADXL 345

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to $\pm 16g$. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I²C digital interface. The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0° . Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention.

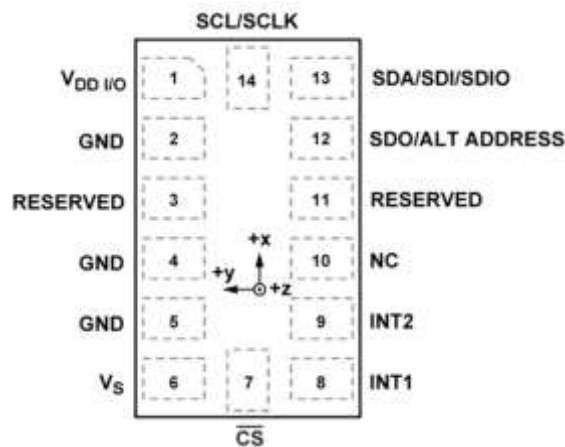
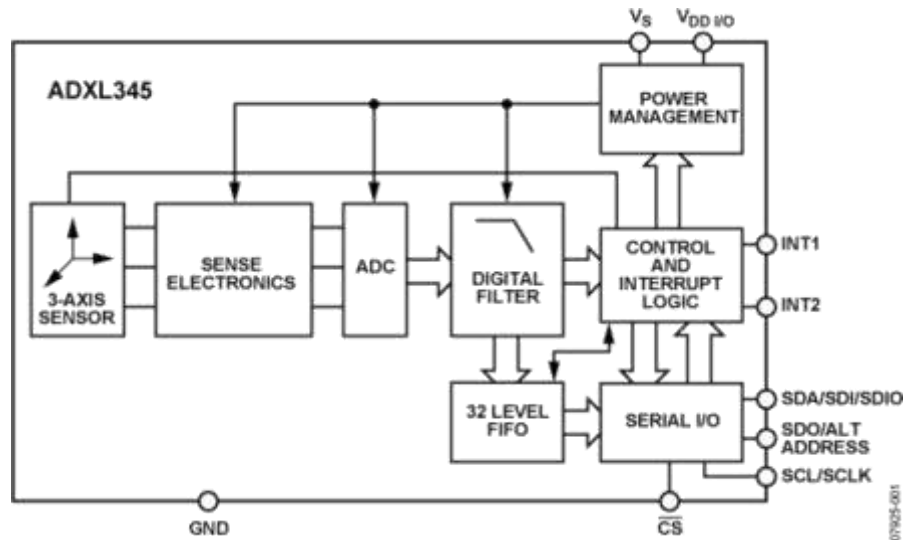


Figure 2.4: ADXL345 pin diagram

2.3.1 THEORY OF OPERATION



The ADXL345 is a complete 3-axis acceleration measurement system with a selectable measurement range of ± 2 g, ± 4 g, ± 8 g, or ± 16 g. It measures both dynamic acceleration resulting from motion or shock and static acceleration, such as gravity, that allows the device to be used as a tilt sensor. The sensor is a polysilicon surface-micro machined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against forces due to applied acceleration. Deflection of the structure is measured using differential capacitors that consist of independent fixed plates and plates attached to the moving mass. Acceleration deflects the proof mass and unbalances the differential capacitor, resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation is used to determine the magnitude and polarity of the acceleration.

2.3.2 APPLICATIONS

1. Handsets
2. Medical instrumentation
3. Gaming and pointing devices
4. Industrial instrumentation
5. Personal navigation devices
6. Hard disk drive (HDD) protection

2.4 ARDUINO

Arduino is a software company, project, and user community that designs and manufactures computer hardware, open-source, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog I/O pins that can interface to various expansion boards (termed shields) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment(IDE) based on a programming language named Processing, which also supports the languages C and C++.

The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong.



Figure 2.6: Arduino

2.4.1 Features:

- ATmega328 microcontroller
- Input voltage - 7-12V
- 14 Digital I/O Pins (6 PWM outputs)
- 6 Analog Inputs
- 32k Flash Memory
- 16Mhz Clock Speed
- EEPROM 1KB
- SRAM 2 KB

An Arduino board historically consists of an Atmel 8-, 16- or 32-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which lets users connect the CPU board to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus.

2.4.2 Why Arduino?

The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. There are many other microcontrollers and microcontroller platforms available for physical computing. All of the other tools take the lots details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand.

- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

2.5 Bluetooth

Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices, and building personal area networks (PANs). Invented by telecom vendor Ericsson in 1994, it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization.

2.5.1 Bluetooth Module HC-05:

HC-05 module is an easy to use Bluetooth SPP module, designed for transparent wireless serial connection setup. It is a class-2 Bluetooth module with Serial Port Profile, which can configure as either Master or slave. a Drop-in replacement for wired serial connections, transparent usage. You can use it simply for a serial port replacement to establish connection between MCU, PC to your embedded project.

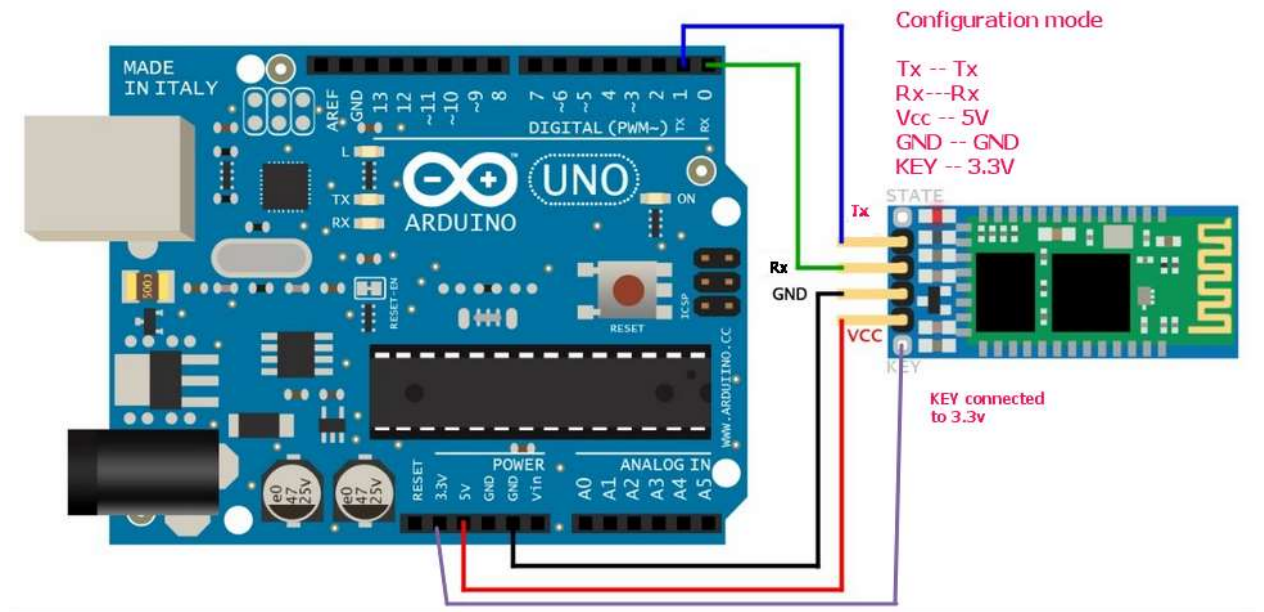


Figure 2.7: Interfacing of Bluetooth module (HC-05) with Arduino

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm.

Hardware Features:

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software Features:

- Default Baud rate: 38400, Data bits: 8, Stop bit: 1, Parity: No parity, Data control: has baud rate: 9600,19200,38400,57600,115200,230400,460800.

- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected.
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:”0000” as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

2.5.2 Applications

- Computer and peripheral devices
- GPS receiver
- Industrial control
- MCU projects

2.6 Pulse Sensor

The Pulse Sensor that we make is essentially a PPG, which is a well known medical device used for non-invasive heart rate monitoring. Sometimes, PPG measure blood-oxygen levels (SpO2), sometimes they don't. The heart pulse signal that comes out of a PPG is an analog fluctuation in voltage, and it has a predictable wave shape. The depiction of the pulse wave is called a PPG.

Heart rate data can be really useful whether you're designing an exercise routine, studying your activity or anxiety levels or just want your shirt to blink with your heart beat. The problem is that heart rate can be difficult to measure.

The Pulse Sensor Amped is a plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects. It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and

easy to get reliable pulse readings. Also, it sips power with just 4mA current draw at 5V so it's great for mobile applications.

Simply clip the Pulse Sensor to your earlobe or fingertip and plug it into your 3 or 5 Volt Arduino and you're ready to read heart rate! The 24" cable on the Pulse Sensor is terminated with standard male headers so there's no soldering required. Of course Arduino example code is available as well as a Processing sketch for visualizing heart rate data.

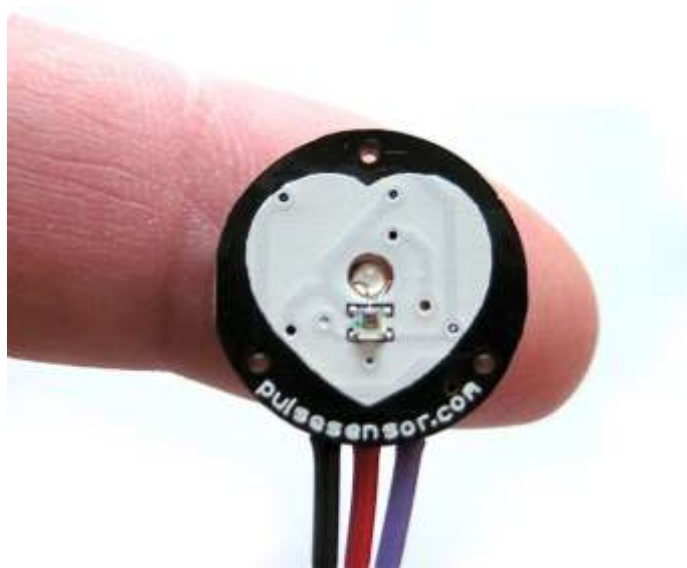


Figure 2.8: Pulse Sensor

Features:

- Working voltage: 5V
- Working current: 4mA

CHAPTER 3

PROTOCOLS

3.1 SPI

Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels.

Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical. Serial computer buses are becoming more common even at shorter distances, as improved signal integrity and transmission speeds in newer serial technologies have begun to outweigh the parallel bus's advantage of simplicity (no need for serialize and deserializer) and to outstrip its disadvantages (clock skew, interconnect density).

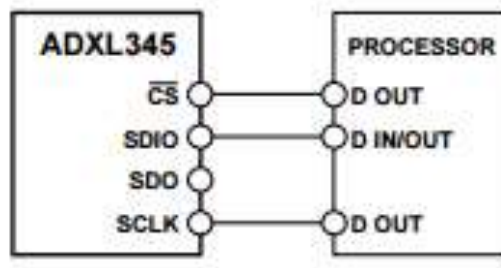


Figure 3.1: SPI Connection

The asynchronous serial protocol has a number of built-in rules - mechanisms that help ensure robust and error-free data transfers. These mechanisms, which we get for eschewing the external clock signal, are:

- Data bits,
- Synchronization bits,
- Parity bits,
- Baud rate.

Through the variety of these signaling mechanisms, you'll find that there's no one way to send data serially. The protocol is highly configurable. The critical part is making sure that both devices on a serial bus are configured to use the exact same protocols.



Figure 3.2: SPI Header

3.2 I2C

I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. It was invented by Philips and now it is used by almost all major IC manufacturers. Each I2C slave device needs an address – they must still be obtained from NXP (formerly Philips semiconductors).

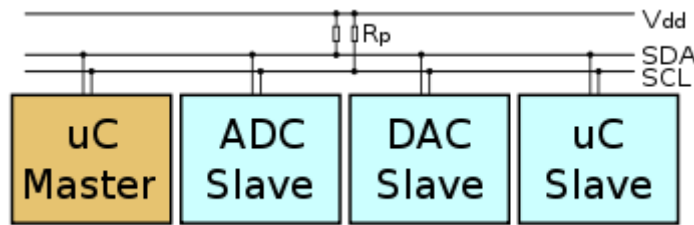


Figure 3.3: I2C Connection

I2C bus is popular because it is simple to use, there can be more than one master, only upper bus speed is defined and only two wires with pull-up resistors are needed to connect almost unlimited number of I2C devices. I2C can use even slower microcontrollers with general-purpose I/O pins since they only need to generate correct Start and Stop conditions in addition to functions for reading and writing a byte.

Each slave device has a unique address. Transfer from and to master device is serial and it is split into 8-bit packets. All these simple requirements make it very simple to implement I2C interface even with cheap microcontrollers that have no special I2C hardware controller. You only need 2 free I/O pins and few simple i2C routines to send and receive commands.

The initial I2C specifications defined maximum clock frequency of 100 kHz. This was later increased to 400 kHz as Fast mode. There is also a High speed mode which can go up to 3.4 MHz and there is also a 5 MHz ultra-fast mode.

3.2.1 I2C Interface

I2C uses only two wires: SCL (serial clock) and SDA (serial data). Both need to be pulled up with a resistor to +Vdd. There are also I2C level shifters which can be used to connect to two I2C buses with different voltages.

3.2.2 I2C Addresses

Basic I2C communication is using transfers of 8 bits or bytes. Each I2C slave device has a 7-bit address that needs to be unique on the bus. Some devices have fixed I2C address while others have few address lines which determine lower bits of the I2C address. This makes it very easy to have all I2C devices on the bus with unique I2C address. There are also devices which have 10-bit address as allowed by the specification.

7-bit address represents bits 7 to 1 while bit 0 is used to signal reading from or writing to the device. If bit 0 (in the address byte) is set to 1 then the master device will read from the slave I2C device.

Master device needs no address since it generates the clock (via SCL) and addresses individual I2C slave devices.

3.2.3 I2C Protocol

In normal state both lines (SCL and SDA) are high. The communication is initiated by the master device. It generates the Start condition (S) followed by the address of the slave device (B1). If the bit 0 of the address byte was set to 0 the master device will write to the slave device (B2). Otherwise, the next byte will be read from the slave device. Once all bytes are read or written (Bn) the master device generates Stop condition (P). This signals to other devices on the bus that the communication has ended and another device may use the bus.

Most I2C devices support repeated start condition. This means that before the communication ends with a stop condition, master device can repeat start condition with address byte and change the mode from writing to reading. I2C bus is used by many integrated circuits and is simple to implement. Any microcontroller can communicate with I2C devices even if it has no special I2C interface. I2C specifications are flexible - I2C bus can communicate with slow devices and can also use high speed modes to

transfer large amounts of data. Because of many advantages, I2C bus will remain as one of the most popular serial interfaces to connect integrated circuits on the board.

I2C brings the following advantages:-

- I2C only requires two signal lines
- Flexible data transmission rates
- Each device on the bus is independently addressable
- Devices have a simple Master/Slave relationship
- I2C is capable of handling multiple master communications by providing arbitration and communication collision detection
- Longer distance communication than SPI

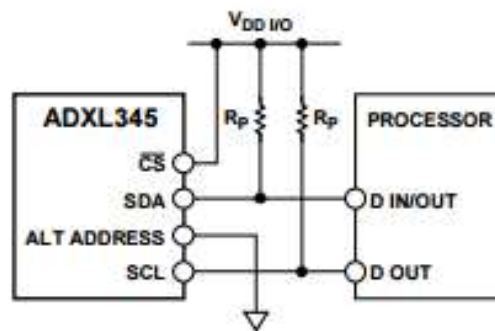


Figure 3.4: I2C Wire Architecture

I2C and SPI digital communications are available. In both cases, the ADXL345 operates as a slave. I2C mode is enabled if the CS pin is tied high to VDD I/O. The CS pin should always be tied high to VDD I/O or be driven by an external controller because there is no default mode if the CS pin is left unconnected. Therefore, not taking these precautions may result in an inability to communicate with the part. In SPI mode, the CS pin is controlled by the bus master. In both SPI and I2C modes of operation, data transmitted from the ADXL345 to the master device should be ignored during writes to the ADXL345.

3.3 Serial V/s Parallel

The communication links across which computers—or parts of computers—talk to one another may be either serial or parallel. A parallel link transmits several streams of data simultaneously along multiple channels (e.g., wires, printed circuit tracks, or optical fibres); whereas, a serial link transmits only a single stream of data.

Although a serial link may seem inferior to a parallel one, since it can transmit less data per clock cycle, it is often the case that serial links can be clocked considerably faster than parallel links in order to achieve a higher data rate. Several factors allow serial to be clocked at a higher rate:

- Clock skew between different channels is not an issue (for unclocked asynchronous serial communication links).
- A serial connection requires fewer interconnecting cables (e.g., wires/fibres) and hence occupies less space. The extra space allows for better isolation of the channel from its surroundings.
- Crosstalk is less of an issue, because there are fewer conductors in proximity.

In many cases, serial is a better option because it is cheaper to implement. Many ICs have serial interfaces, as opposed to parallel ones, so that they have fewer pins and are therefore less expensive.

CHAPTER 4

ALGORITHMS

4.1 Step Detection:

Let's think about the nature of step. Figure 4.1 depicts a single step, defined as a unit cycle of walking behaviour, showing the relationship between each stage of the walking cycle and the change in acceleration.

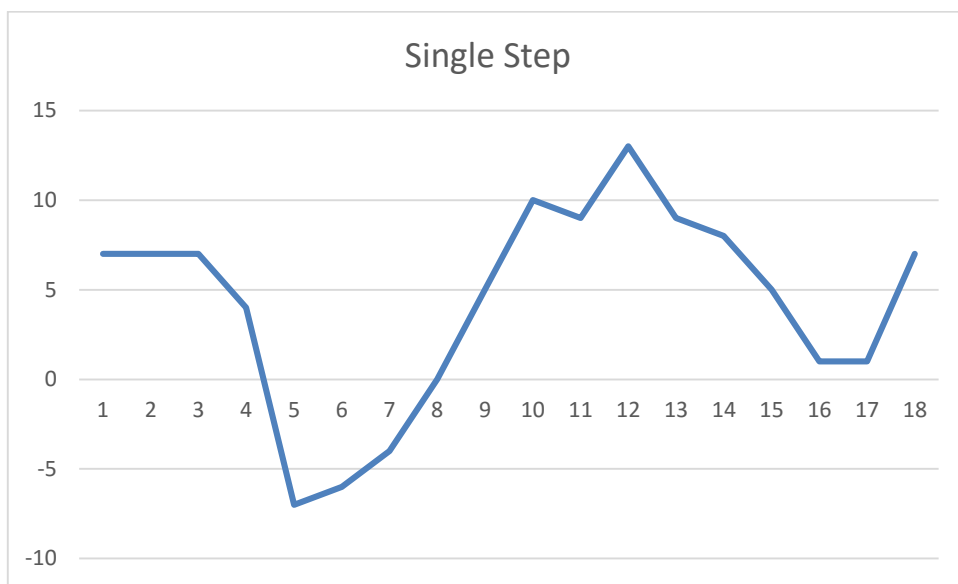


Figure4.1: Step Variation Graph

4.1.1 Algorithm

Steps Parameter

Smoothing Filter: First, a smoothing filter is needed to smooth the signals shown in Figure 4.2. Fifteen registers and a summing unit can be used, as shown in Figure 4.2. Of course, more registers could be used to make the acceleration data smoother, but the response time would be slower.

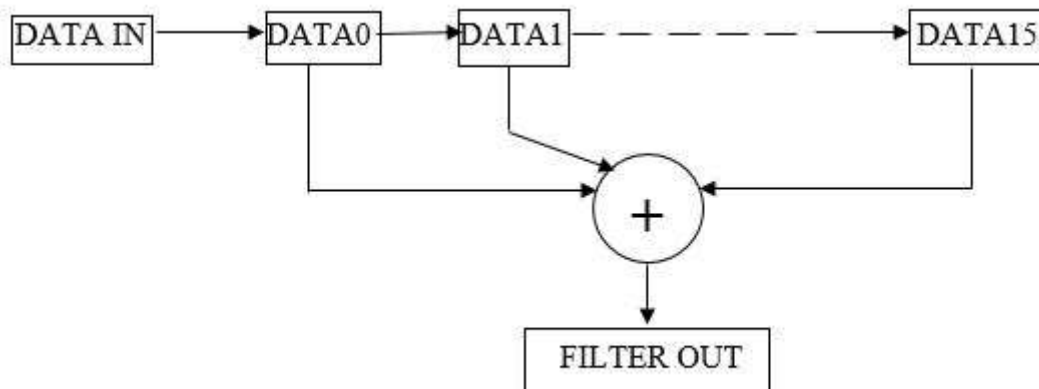


Figure 4.2: Smoothing Filter

The filtered data from the most active axis of a pedometer worn by a walking person. The peak-to-peak value would be higher for a runner. Filtered data of the most active axis.

Dynamic Threshold and Dynamic Precision: The system continuously updates the maximum and minimum values of the 3-axis acceleration every 30 samples. The average value, $(\text{Max} + \text{Min})/2$, is called the dynamic threshold level. For the following 30 samples, this threshold level is used to decide whether steps have been taken. As it is updated every 30 samples, the threshold is dynamic. This choice is adaptive and fast enough. In addition to dynamic threshold, dynamic precision is also used for further filtering. A linear-shift-register and the dynamic threshold are used to decide whether an effective step has been taken. The linear shift-register contains two registers, a sample_new register and a sample_old register. The data in these are called sample_new and sample_old, respectively.

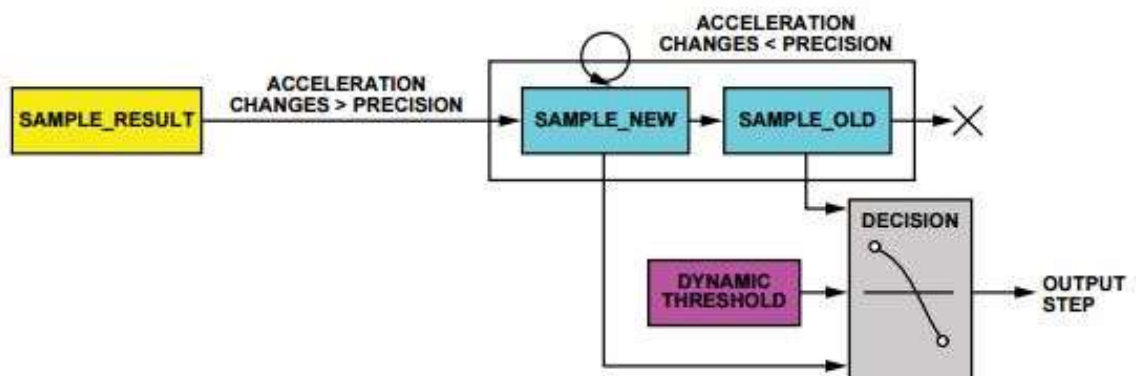


Figure 4.3: Step Detection

When a new data sample comes, `sample_new` is shifted to the `sample_old` register unconditionally. However, whether the `sample_result` will be shifted into the `sample_new` register depends on a condition: If the changes in acceleration are greater than a predefined precision, the newest sample result, `sample_result`, is shifted to the `sample_new` register; otherwise the `sample_new` register will remain unchanged. The shift register group can thus remove the high-frequency noise and make the decision more precise.

A step is defined as happening if there is a negative slope of the acceleration plot ($\text{sample_new} < \text{sample_old}$) when the acceleration curve crosses below the dynamic threshold.

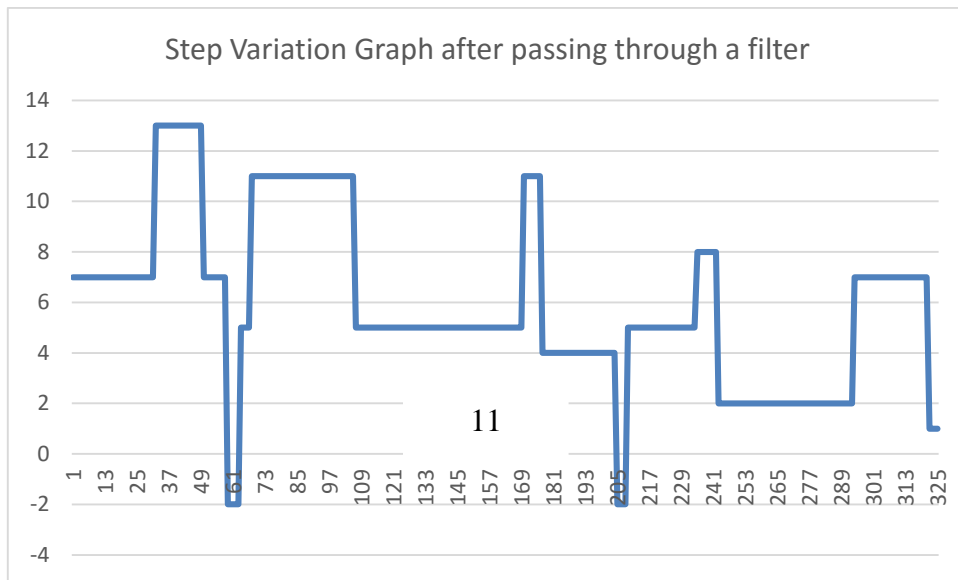


Figure 4.4: Number of steps (5)

4.2 Free Fall Algorithm:

In ADXL345, we have a free fall register which triggers an interrupt if the root mean square acceleration of all the three axes is less than the threshold set in the free fall register for the time duration set in `TIME_FF` register. The algorithm that is taking part in Free fall register is described below with the help of diagram.

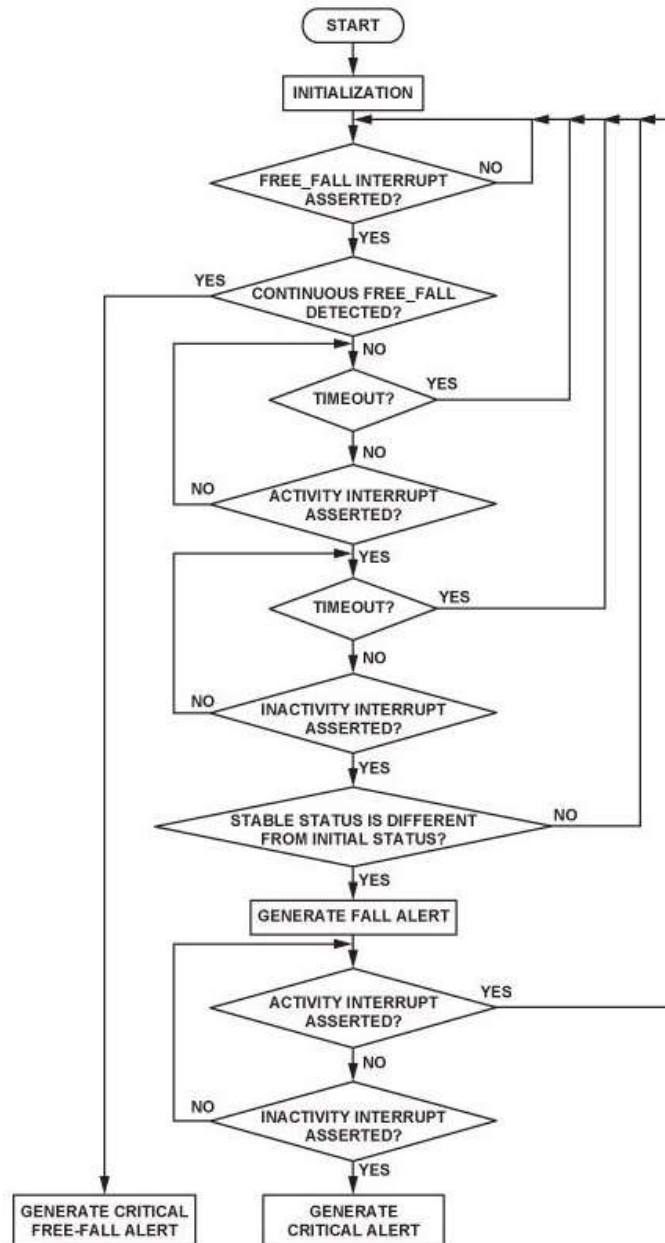


Figure 4.5: Free Fall Algorithm

4.3 Tap Detection Algorithm

The tap interrupt function is capable of detecting either single or double taps. The following parameters are shown in Figure for a valid single and valid double tap event:

- The tap detection threshold is defined by the THRESH_TAP register (Address 0x1D).
- The maximum tap duration time is defined by the DUR register (Address 0x21).

- The tap latency time is defined by the latent register (Address 0x22) and is the waiting period from the end of the first tap until the start of the time window, when a second tap can be detected, which is determined by the value in the window register (Address 0x23).
- The interval after the latency time (set by the latent register) is defined by the window register. Although a second tap must begin after the latency time has expired, it need not finish before the end of the time defined by the window register.

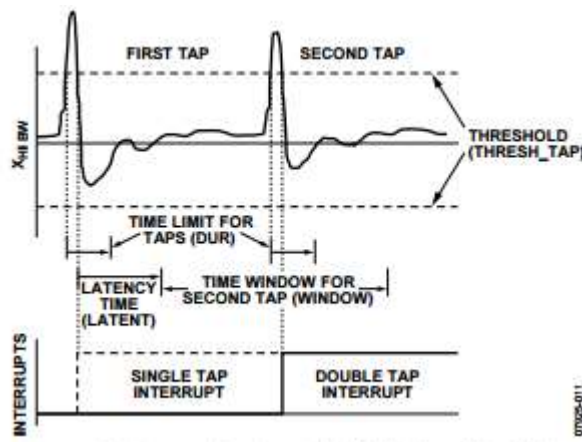


Figure 4.6: Tap Detection

If only the single tap function is in use, the single tap interrupt is triggered when the acceleration goes below the threshold, as long as DUR has not been exceeded. If both single and double tap functions are in use, the single tap interrupt is triggered when the double tap event has been either validated or invalidated. Several events can occur to invalidate the second tap of a double tap event. First, if the suppress bit in the TAP_AXES register (Address 0x2A) is set, any acceleration spike above the threshold during the latency time (set by the latent register) invalidates the double tap detection, as shown in Figure.

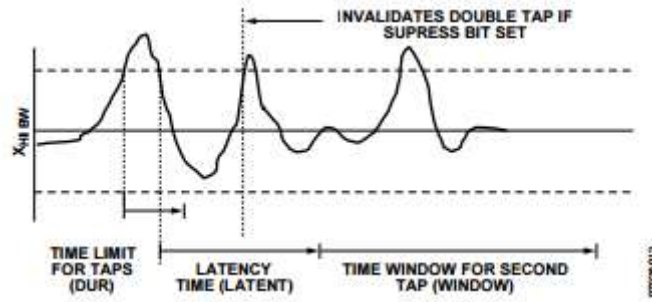


Figure 4.7: Invalid Double Tap if acceleration is found greater than THRESH TAP register in Latency window

A double tap event can also be invalidated if acceleration above the threshold is detected at the start of the time window for the second tap (set by the window register). This results in an invalid double tap at the start of this window, as shown in Figure 4.7. Additionally, a double tap event can be invalidated if an acceleration exceeds the time limit for taps (set by the DUR register), resulting in an invalid double tap at the end of the DUR time limit for the second tap event, also shown in Figure 4.8.

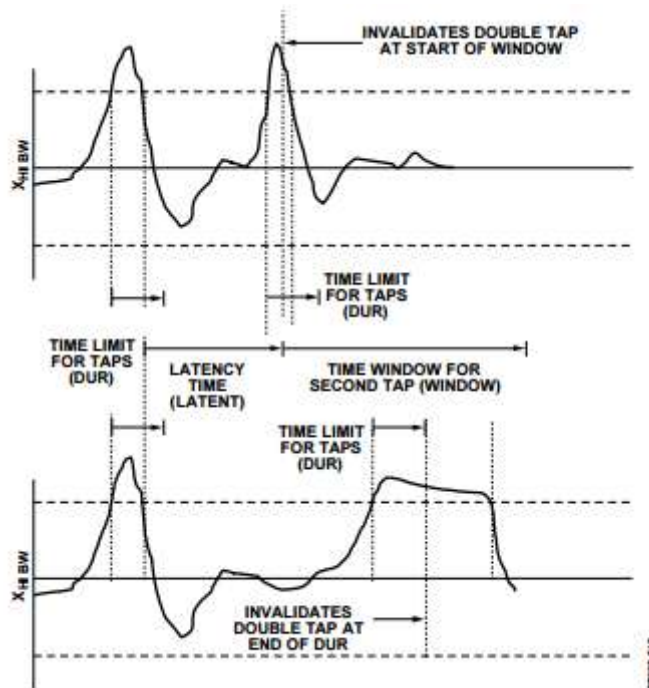


Figure 4.8: Invalid Double Tap

Single taps, double taps, or both can be detected by setting the respective bits in the INT_ENABLE register (Address 0x2E). Control over participation of each of the three axes in single tap/ double tap detection is exerted by setting the appropriate bits in the TAP_AXES register (Address 0x2A). For the double tap function to operate, both the latent and window registers must be set to a nonzero value. Every mechanical system has

somewhat different single tap/double tap responses based on the mechanical characteristics of the system. Therefore, some experimentation with values for the latent, window, and THRESH_TAP registers is required. In general, a good starting point is to set the latent register to a value greater than 0x10, to set the window register to a value greater than 0x10, and to set the THRESH_TAP register to be greater than 3 g. Setting a very low value in the latent, window, or THRESH_TAP register may result in an unpredictable response due to the accelerometer picking up echoes of the tap inputs. After a tap interrupt has been received, the first axis to exceed the THRESH_TAP level is reported in the ACT_TAP_STATUS register (Address 0x2B). This register is never cleared, but is overwritten with new data.

4.4 PULSE SENSOR

First we will detect the heart beat/pulse and count the pulses for one minute to get the beats per minute. So in order to detect the pulse we will pass light (using an LED) from one side of the finger and measure the intensity of light received on the other side (using an LDR). Whenever the heart pumps blood more light is absorbed by increased blood cells and we will observe a decrease in the intensity of light received on the LDR. As a result the resistance value of the LDR increases. This variation in resistance is converted into voltage variation using a signal conditioning circuit usually an OP-AMP. The signal is amplified enough to be detectable by the microcontroller inputs. The signal given to the microcontroller input will look somewhat like shown in the image above in a oscilloscope. The microcontroller can be programmed to receive an interrupt for every pulse detected and count the number of interrupts or pulses in a minute. The count value of pulses per minute will give you the Heart rate in bpm (Beats Per Minute).

When the heart pumps blood through the body, with every beat there is a pulse wave (kind of like a shock wave) that travels along all arteries to the very extremities of capillary tissue where the Pulse Sensor is attached. Actual blood circulates in the body much slower than the pulse wave travels. Let's follow events as they progress from point 'T' on the PPG below. A rapid upward rise in signal value occurs as the pulse wave passes under the sensor, then the signal falls back down toward the normal point. Sometimes, the dicroic notch (downward spike) is more pronounced than others, but generally the signal settles down to background noise before the next pulse wave washes through. Since the wave is

repeating and predictable, we could choose almost any recognizable feature as a reference point, say the peak, and measure the heart rate by doing math on the time between each peak. This, however, can run into false readings from the dicrotic notch, if present, and may be susceptible to inaccuracy from baseline noise as well. There are other good reasons not to base the beat-finding algorithm on arbitrary wave phenomena. Ideally, we want to find the instantaneous moment of the heart beat. This is important for accurate BPM calculation.

Some heart researchers say it's when the signal gets to 25% of the amplitude, some say when it's 50% of the amplitude, and some say it's the point when the slope is steepest during the upward rise event. Pulse Sensor algorithm is designed to measure the IBI by timing between moments when the signal crosses 50% of the wave amplitude during that fast upward rise. The BPM is derived every beat from an average of the previous 10 IBI times.

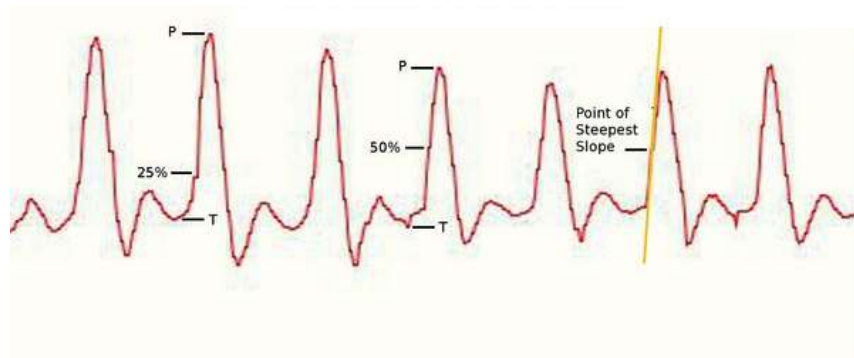


Figure 4.9: PPG Plot

CHAPTER 5

WEB APPLICATION

In the old days, the Internet used to be about consumption. Users would passively read, watch and download content. Chat was done on message boards or blogs, not in real time.

But as social media has increased in popularity, today's users want to be able to receive instantaneous feedback over the Web. Chatting, gaming and healthcare monitoring now take place in real time. Even more significantly, this is happening on an imposing scale with up to millions of users interacting on one site at one time.

Software developers know the Web wasn't built for this. Real-time response means that each users' client would be constantly querying the server, making for sluggish, inefficient response times.

5.1 NODE JS

Node.js is an open-source, cross-platform runtime environment for developing server-side Web applications. Although Node.js is not a JavaScript framework, many of its basic modules are written in JavaScript, and developers can write new modules in JavaScript. The runtime environment interprets JavaScript using Google's V8 JavaScript engine.

Node.js has an event-driven architecture capable of asynchronous I/O. It uses a function called the Event Loop that ensures that data isn't constantly being queried for, but simply transmitted when it exists. It's kind of like a holding cell for communication. So while the Event Loop is handling asynchronous tasks, Node.js can continue running the program normally and leave out the heavy lifting. It makes Node.js very good at idling when there's nothing to do, instead of constantly querying for more information.

This means that developers can use Node.js to build applications that scale to millions of users. Real time communication is managed by the event loop without taking up much memory, so developers can spend more time working on the functionality of the app than they might spend worrying about the app getting clogged with too many queries.

These design choices aim to optimize throughput and scalability in Web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

The Node.js distributed development project, governed by the Node.js Foundation, is facilitated by the Linux Foundation's Collaborative Projects program.

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionality and other core functions. Node.js modules use an API designed to reduce the complexity of writing server applications. Common frameworks include Connect, Express.js, Socket.IO, Koa.js, Hapi.js, Sails, Meteor, Derby, and many others.

5.1.1 V8

V8 is the JavaScript execution engine built for Google Chrome and open-sourced by Google in 2008. Written in C++, V8 compiles JavaScript source code to native machine code instead of interpreting it in real time.

Node.js uses libuv to handle asynchronous events. Libuv is an abstraction layer for network and file system functionality on both Windows and POSIX-based systems like Linux, Mac OS X, OSS on NonStop and UNIX.

The core functionality of Node.js resides in a JavaScript library. The Node.js bindings, written in C++, connect these technologies to each other and to the operating system.

5.1.2 Package management

npm is the pre-installed package manager for the Node.js server platform. It is used to install Node.js programs from the npm registry, organizing the installation and management of third-party Node.js programs. It is not used to load code; instead, it is used to install code and manage code dependencies from the command line. The packages found in the npm registry can range from simple helper libraries.

5.2 SOCKET.IO

Socket.IO is a JavaScript library for realtime web applications. It enables realtime, bi-directional communication between web clients and servers. It has two parts:

- a client-side library that runs in the browser, and
- a server-side library for node.js.

Both components have a nearly identical API. Like node.js, it is event-driven. There are implementations for the server-side library in other languages. Whichever server-side language implementation we so choose, the following 6 transports are supported:

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Socket.IO selects the most capable transport at runtime without affecting its APIs so that we can have realtime connectivity on every browser. Socket.IO primarily uses the WebSocket protocol.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol makes more interaction between a browser and a website possible, facilitating the real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. The communications are done over TCP port number 80. The WebSocket protocol is currently supported in most major browsers including Google Chrome, Internet Explorer, Firefox, Safari and Opera. WebSocket also requires web applications on the server to support it.

Although Socket.IO can be used as simply a wrapper for WebSocket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O. It can be installed with the npm tool.

Socket.IO provides the ability to implement real-time analytics, binary streaming, instant messaging, and document collaboration. Socket.IO handles the connection transparently. It will automatically upgrade to WebSocket if possible. This requires the programmer to only have Socket.IO knowledge.

Socket.IO is not a WebSocket library with fallback options to other realtime protocols. It is a custom realtime transport protocol implementation on top of other realtime protocols. Its protocol negotiation parts cause a client supporting standard WebSocket to not be able to contact a Socket.IO server. And a Socket.IO implementing client cannot talk to a non-Socket.IO based WebSocket or Long Polling Comet server. Therefore Socket.IO requires using the Socket.IO libraries on both client and server side.

5.3 WEB SERVER

A web server is an information technology that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. The term can refer either to the entire computer system, an appliance, or specifically to the software that accepts and supervises the HTTP requests. The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.

Web servers are not only used for serving the World Wide Web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring and/or administering the device in question. This usually means that no additional software has to be installed on the client computer, since only a web browser is required.

5.3.1 Web Application Architecture

A Web application is usually divided into four layers:

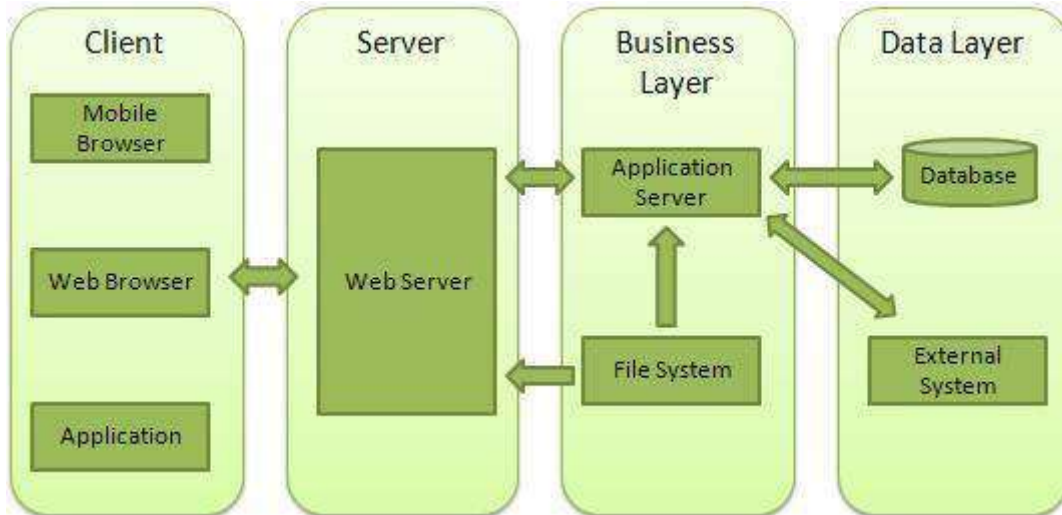


Figure 5.1: Web Application Architecture

- **Client** - This layer consists of web browsers, mobile browsers or applications which can make HTTP request to the web server.
- **Server** - This layer consists of Web server which can intercepts the request made by clients and pass them the response.
- **Business** - This layer consists of application server which is utilized by web server to do required processing. This layer interacts with data layer via data base or some external programs.
- **Data** - This layer consists of databases or any source of data.

5.3.2 Creating Web Server using Node

Node.js provides **http** module which can be used to create either HTTP client of server. The HTTP server is created using the `http` module's `createServer()` method. Like most Node.js functions, `createServer()` takes a callback function as an argument. This callback function is executed each time the server receives a new request.

The callback function takes two arguments, `request` and `response`. The `request` object contains information regarding the client's request, such as the URL, HTTP headers, and much more. Similarly, the `response` object is used to return data back to the client.

The callback function begins by calling the `response.writeHead()` method. This method sends an HTTP status code and a collection of response headers back to the client. The status code is used to indicate the result of the request. For example, everyone has encountered a 404 error before, indicating that a page could not be found. The example server returns the code 200, which indicates success.

Along with the status code, the server returns a number of HTTP headers which define the parameters of the response. If you do not specify headers, Node.js will implicitly send them for you. The example server specifies only the `Content-Type` header. This particular header defines the MIME type of the response. In the case of an HTML response, the MIME type is “text/html”.

Next, the server executes several calls to `response.write()`. These calls are used to write the HTML page. By default, UTF-8 character encoding is used. Technically, all of these calls could be combined into a single call to improve performance. However, for such a trivial example, performance has been sacrificed for the sake of code readability.

After the HTML page has been written, the `response.end()` method is called. By calling this method, we are telling the server that the response headers and body have been sent, and that the request has been fulfilled. The example server calls `end()` with no parameters. However, `end()` can also be called like `write()`, assuming only one call is needed.

5.4 File System Module

By default, Node.js installations come with the file system module, `fs`. For the most part, `fs` simply provides a wrapper for the standard file operations. We open the file using `open()`. The `r` argument denotes that the file is being opened for reading. The `open()` callback function provides a file descriptor, `fd` for accessing the newly opened file. Inside the callback function, we define a buffer to hold the file’s contents.

The file system module allows programs to watch for modifications to specific files. This is very useful in programs such as nodemon, which automatically restarts a program when its source code is modified. The `watch()` function takes three arguments. The first argument is the name of the file to watch. The second argument is optional, and provides

configuration settings. If present, the second argument should be an object containing a Boolean value named `persistent`. If `true`, `persistent` prevents the program from terminating. If the second argument is omitted, it defaults to `true`. The final argument is a callback which is triggered when the target file is modified.

5.5 WEB PORTAL

The web portal includes filling up the patient's details and monitoring the beats per minute along with the alert generated if the patient needs help or has fallen.

Firstly for getting the details about the patient like name, patient id, diseases and prescription etc. we need a form and get the data stored on the local network. The data is stored in a file using the node file system module. This file data would be retrieved on the monitoring window.

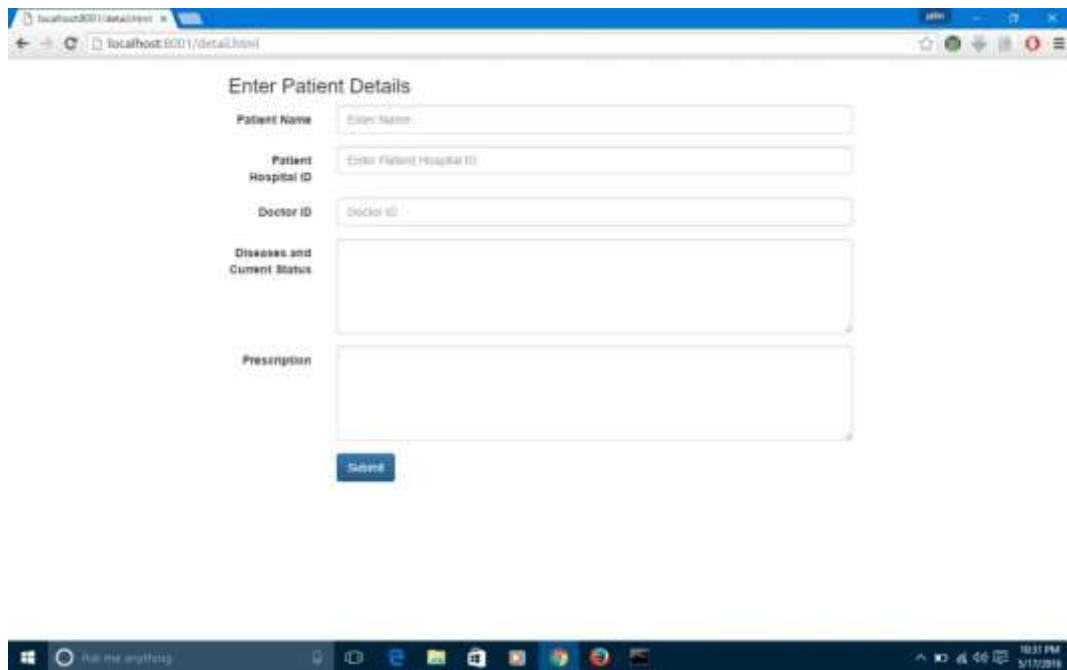
The image shows a web browser window with the address bar displaying 'localhost:6001/detail.html'. The main content area contains a form titled 'Enter Patient Details'. The form has five input fields: 'Patient Name' (with placeholder text 'Enter Name'), 'Patient Hospital ID' (with placeholder text 'Enter Patient Hospital ID'), 'Doctor ID' (with placeholder text 'Doctor ID'), 'Diseases and Current Status', and 'Prescription'. Below these fields is a blue 'Submit' button. The browser's taskbar at the bottom shows the Windows logo, search bar, and several application icons, with the system tray displaying the time as 10:31 PM on 5/17/2018.

Figure 5.2: Patient Detail Form

The monitoring window shows the details of the patients entered before and could be edited also. This part of the web portal produces a live plot of the beats per minute using the data from the pulse sensor and produces a speech alert if the patient need any

assistance or has fallen accidentally. The portal can also be configured to generate alert if the BPM of the patient falls below or rise above a certain level.

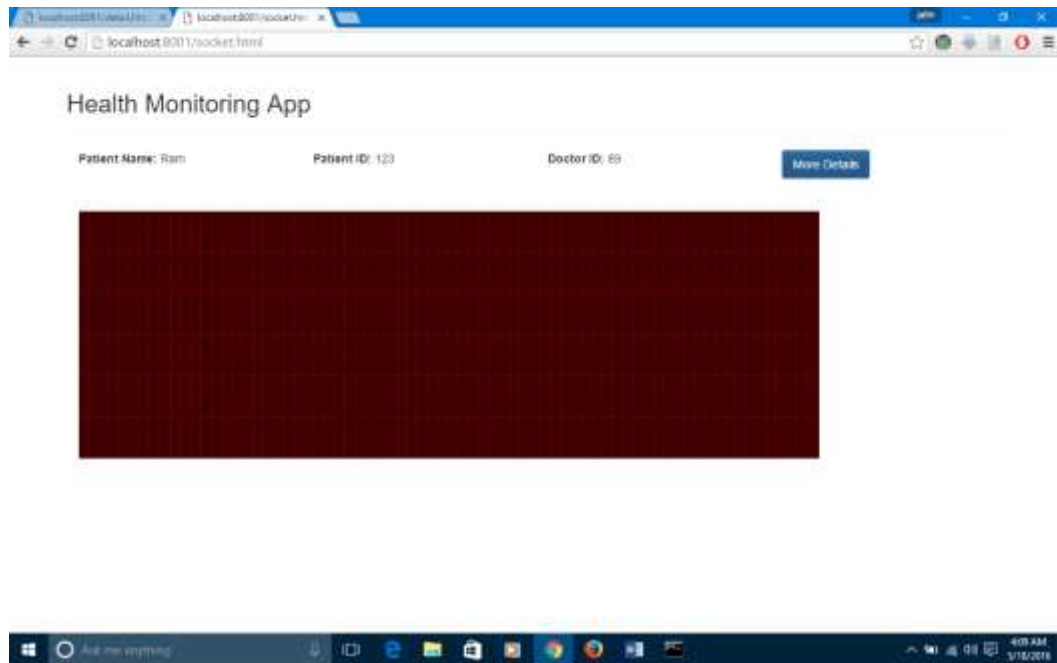


Figure 5.3: Monitoring Window

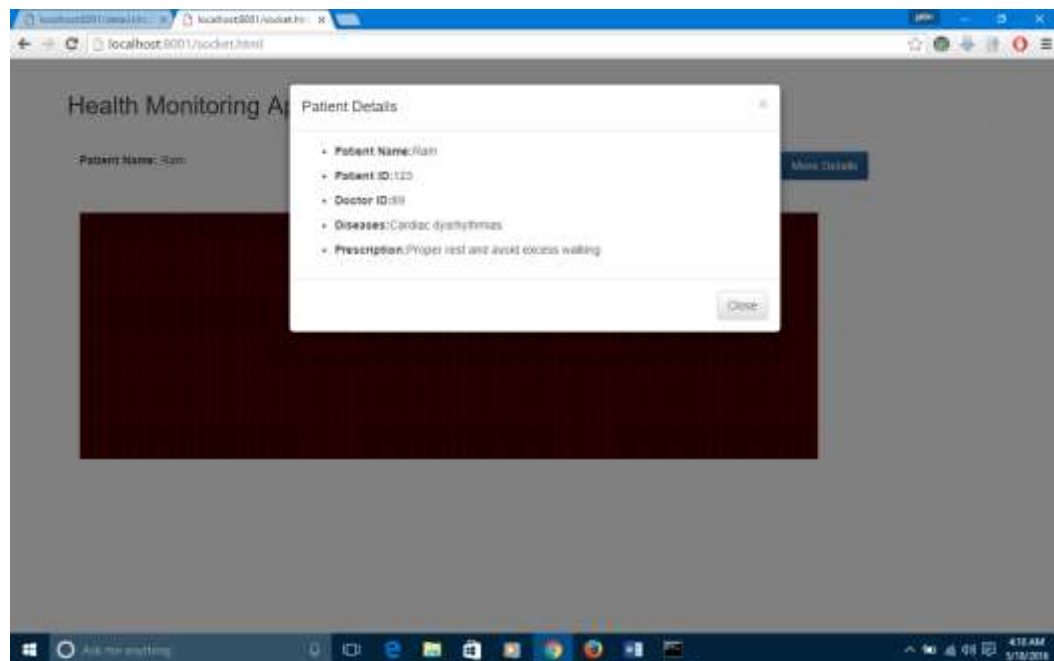


Figure 5.4: Monitoring window showing patient detail

CHAPTER 6

Readings

6.1 Step Detection

Figure 2 depicts a single step, defined as a unit cycle of walking behaviour, showing the relationship between each stage of the walking cycle and the change in acceleration.

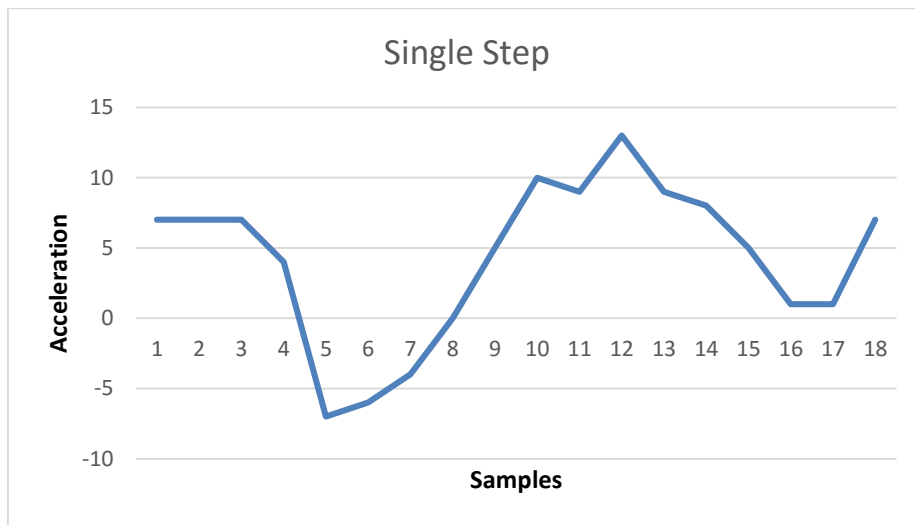


Figure 6.1: A step detection

So by the algorithm discussed above we can say that in the graph shown below, a person has taken 5 steps.

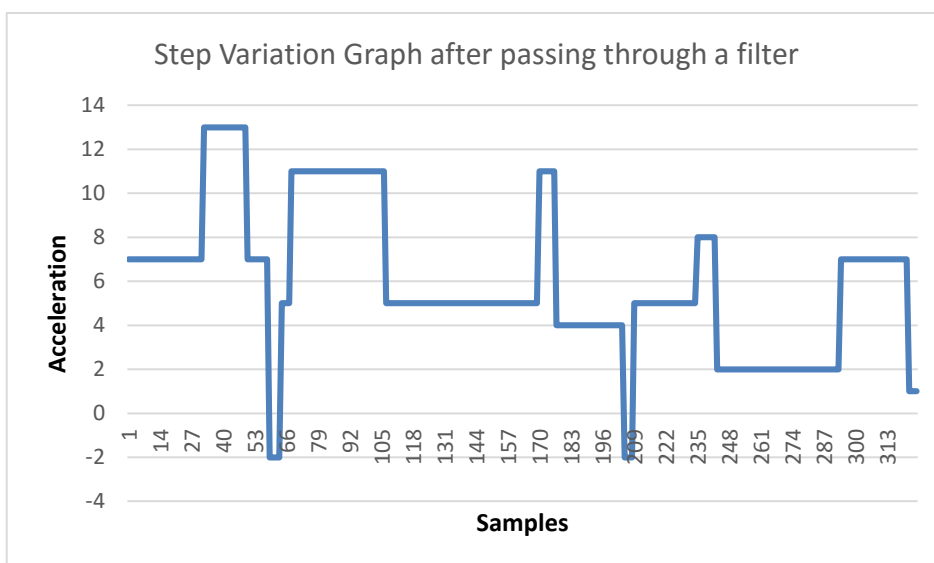


Figure 6.2: 5 steps taken as there are 5 peaks.

6.2 Tap

For Tap detection, to get the best output readings we first took the readings of single tap in all the different axis i.e. x axis, y axis, z axis with the help of the accelerometer (adx1345). Three different graphs of axis which is in respect to acceleration vs sample graph with the help of this graph we considered the most suitable axis which help us to determine that an event has occurred.

The Graphs are as follows:

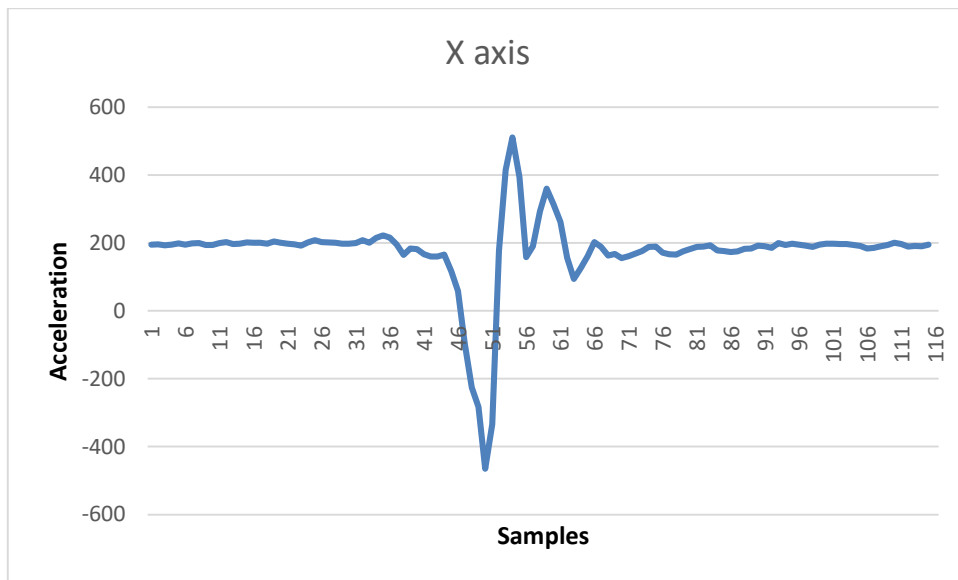


Figure 6.3: Readings of x axis

From the above graph we determined that in the x axis after the completion of an event i.e Single tap here the graph doesn't show a flat slope at the end .The two continuous turfs making it difficult to determine the actual timing or the occurrence of the event. Hence, making this axis not efficient enough for Single tap.

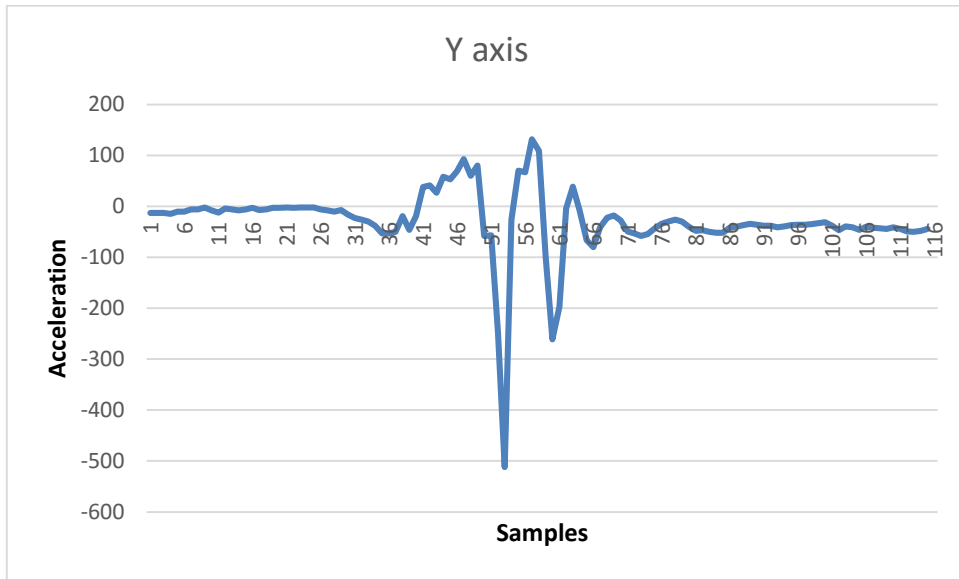


Figure 6.4: Readings of Y axis

From the above graph we determined that in the y axis the readings we received from the accelerometer forms a very fluctuating graph. The reading are very varying in this axis leading to a distorted graph making this axis not efficient for the Single tap.

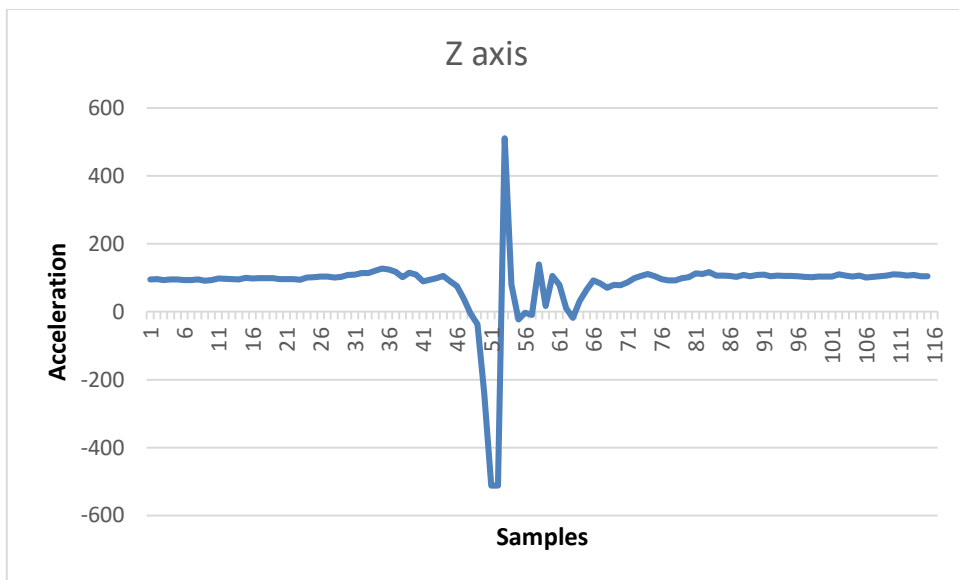


Figure 6.5: Readings of Z axis

From the above graph we determined that in the z axis the readings we received from the accelerometer forms a good graph with least fluctuation showing a single ripple for Single tap. Hence, making this axis suitable for the readings.

6.2.1 Single Tap

Below given are the readings of single tap response in the three different axis looking the same you can determine the suitable one.

The readings are as follows:

X axis	Y axis	Z axis
13	25	9
11	21	9
12	21	9
16	16	8
13	23	7
13	19	8
13	20	7
16	20	8
14	22	-3
-6	-5	7
-5	-4	-9

Table 6.1: Single tap readings

6.2.2 Double Tap

As we know, the DOUBLE_TAP bit is set when two acceleration events that are greater than the value in the THRESH_TAP register (Address 0x1D) occur for less time than is specified in the DUR register (Address 0x21), with the second tap starting after the time specified by the latent register (Address 0x22) but within the time specified in the window register (Address 0x23).

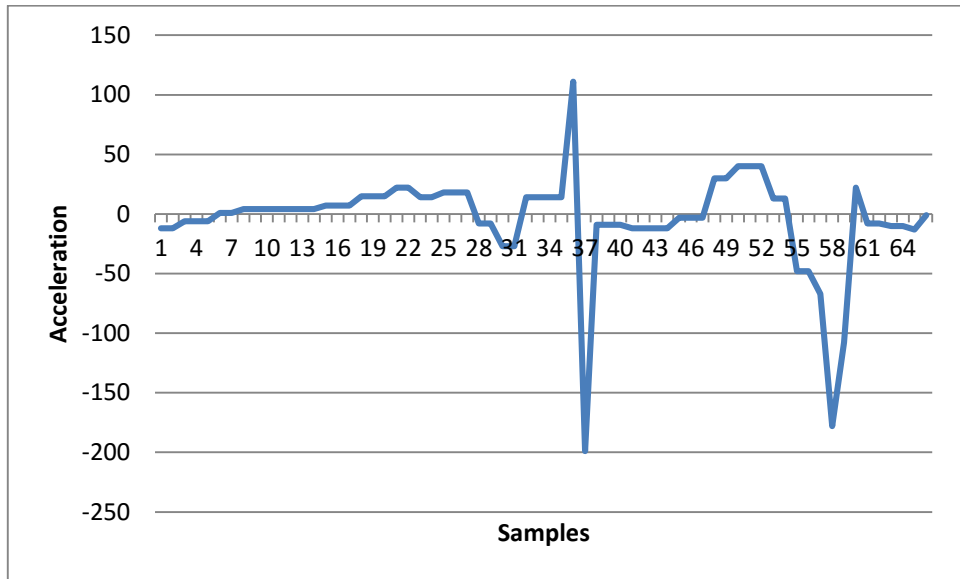


Figure 6.7: Double Tap Detection

In double tap, we have considered z axis as variation in this axis is more smooth and can be easily interpreted.

6.3 Freefall

In this we have taken the reading considering the acceleration of a falling body we have set a threshold value along with that a minimum of time taken in occurrence of the freefall. In, the case of freefall there is drastic change in acceleration in the body which could be seen from the graph.

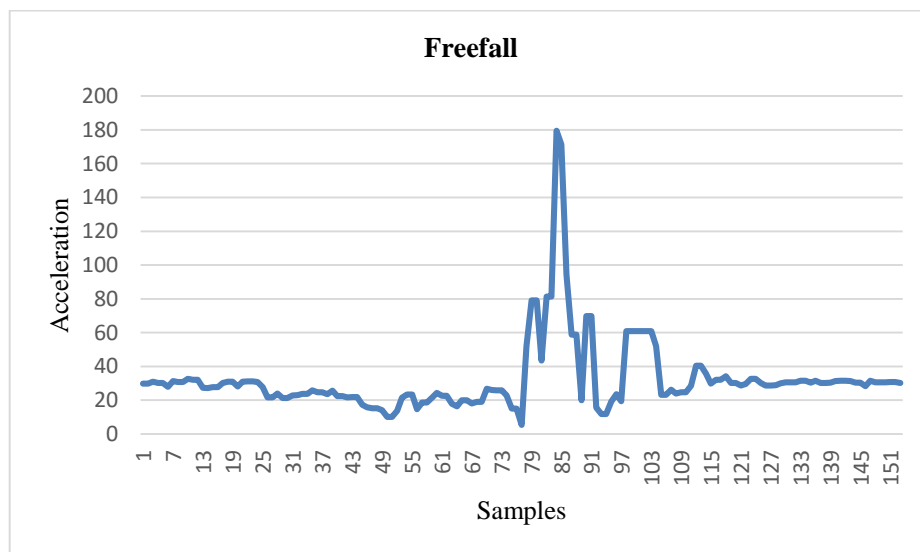


Figure 6.8: Freefall readings

From the above graph we determined that a freefall event has occurred. Now, depending upon the intensity a quick response could be given to the patient in need which can be done either by alarm generation or a simple type of alert on the local pc.

6.4 PULSE SENSOR:

In this we have taken the readings of pulses of a person during walking. The change in acceleration while walking follows a pattern as shown in figure. We implemented a code to measure pulses of a patient during different activities of a patient and compare them with the normal rate. .



Figure 6.9: Pulse Measure

From the above graph we determined that average pulses recorded of a patient while taking normal steps is nearly equal to 70.

FUTURE SCOPE AND LIMITATIONS

In addition to the system for a single patient, we can provide facility so that more than one patient can be monitored at the same time.

According to availability of sensors or development in biomedical trend more than one parameter can be sensed and monitored at the same time which will drastically improve the efficiency of the wireless monitoring system in biomedical field.

CONCLUSION

Wireless continuous healthcare monitoring technology is emerging as a significant element of next generation healthcare services. In this project, we implemented a wireless physiological monitoring system, which is able to continuously monitor the patient's pulse rate and other critical parameters in the hospital. The entire system consists of a node to acquire the patient's physiological data and a server to collect the data. The system is able to carry out a long-term monitoring on patient's health condition and is equipped with an emergency rescue mechanism by generating an alert.

REFERENCES

- [1] Aminian, M., and H. R. Naji. "A hospital healthcare monitoring system using wireless sensor networks." *J. Health Med. Inform* 4 (2013): 121.
- [2] Analog Devices Technical Staff, *ADXL345 Digital Accelerometer data sheet*, Analog Devices, 2009.
- [3] K.C. Kavitha and A.Bazila Banu, "Wireless Health Care Monitoring" in *International Conference on Innovations in Engineering and Technology*, Madurai, Tamil Nadu, India, 2014
- [4] Shelar, Manisha, Jaykaran Singh, and Mukesh Tiwari. "Wireless Patient Health Monitoring System." *International Journal of Computer Applications* 62.6 (2013).
- [5] Pantelopoulos, Alexandros, and Nikolaos G. Bourbakis. "A survey on wearable sensor-based systems for health monitoring and prognosis." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 40.1 (2010): 1-12.
- [6] Ning Jia, "Detecting Human Falls with a 3-Axis Digital Accelerometer" July 2009. [Online]. Available: http://www.analog.com/library/analogDialogue/archives/43-07/fall_detector.html. [Accessed Nov. 14, 2015].
- [7] MEMSnet[®], "What is MEMS Technology?", [Online]. Available: <http://www.memsnet.org/>. [Accessed Aug. 26, 2015].
- [8] Hughes-Croucher, Tom; Wilson, Mike (April 2012), "Up and Running with Node.js (First ed.)", O'Reilly Media, p. 204, ISBN 978-1-4493-9858-3
- [9] Paul Krill (2 June 2014). "Socket.IO JavaScript framework ready for real-time apps". InfoWorld.
- [10] Schlueter, Isaac Z. (25 March 2013). "Forget CommonJS. It's dead. **We are server side JavaScript.**". GitHub.
- [11] Brian w. Evans, "Arduino Programming Notebook", [Online]. Available: http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf.
- [12] Itead Studio, "Serial Bluetooth HC-05 data sheet", [Online]. Available: ftp://imall.iteadstudio.com/Modules/IM120723009/DS_IM120723009.pdf. [Accessed Nov. 14, 2015].