

Master Thesis

Wireless Sensor network platform for testing of
distributed algorithms

Submitted by:

TOQEER RAZA

1st Supervisor: Prof.Dr-Ing .Walter Lang

2nd Supervisor: Dr.-Ing.Reiner Jedermann

Table of Contents

<i>Chapter 1</i>	8
1.1 Theoretical background	9
1.2 Motivation.....	9
1.3 Thesis Implementation Structure	9
1.3.1 Average calculation	10
1.3.2 PKF Algorithm Implementation	10
1.4 Organization of report.....	11
<i>Chapter 2</i>	12
2.1 Wireless Sensor Network.....	13
2.1.1 Gateway	13
2.1.2 Cluster head	13
2.2 Sun SPOT.....	14
2.2.1 Communication of Main Board	14
2.2.2 Operation mode of Sun SPOT	15
2.3 LCD	16
2.3.1 Communication protocol for LCD.....	16
2.3.2 SPI communication in Sun SPOT	18
2.3.3 Hardware connections between LCD and Sun SPOT.....	19
2.3.3.1 NOKIA 5110 LCD Pins.....	19
2.3.3.2 Sun SPOT GPIO and High current Pins	20
2.3.4 Software for Driving LCD.....	20
2.3.4.1 Important methods (functions) for driver library	20
2.3.4.2 Code Location on SVN.....	22
<i>Chapter 3</i>	23
3.1 Distributed Algorithms	24
3.2 Kalman Filter	24
3.2.1 Applications of Kalman Filter.....	24
3.2.2 Basic Equations of Kalman Filter	25
3.3 Predictor Kalman Filter (PKF).....	26
3.3.1 KF implementation in PKF.....	28

3.3.2 Implementation of PKF.....	30
3.3 Matrix in Java	34
3.3.1 JAMA.....	34
3.4 Implementation of PKF in Java	35
3.4.1 Parameters from Matlab to Java.....	35
3.4.1.1 Explanation of Java Class using UML.....	37
3.4.2 Converting PKF in Java	39
3.4.2.1 Different methods in PKF algorithm class.....	40
3.4.3 Testing of PKF using Cross Platform Matlab	41
3.4.3.1 Compare the result	42
<i>Chapter 4.....</i>	<i>44</i>
4.1 Structure of Software implementation	45
4.2 Communication protocols for Sun SPOT nodes	46
4.2.1 Radiostreams.....	46
4.4.2 Radiograms	47
4.3 Flash memory in Sun SPOT.....	48
4.3.1 FlashFile.....	48
4.3.2 Record Management Store (RMS).....	49
4.3.3 Flash memory Implementation	50
4.3.3.1 Important methods	50
4.4 Software design for Average Calculation	53
4.4.1 PC Section.....	53
4.4.1.1 FileMethod.....	53
4.4.1.2 SendDataDemoCluster.....	55
4.4.2 Cluster head	57
4.4.2.1 ClusterHead.....	58
4.4.2.2 JDCD8544.....	62
4.4.3 Node section.....	62
4.4.3.1 SensorNode	62
4.5 Software design for PKF Algorithm implementation	64
4.5.1 PC Section.....	64
4.5.1.1 FileMethod.....	65
4.5.1.2 SendDataDemoCluster.....	65

4.5.2 Leaf node	70
4.5.2.1 Parmeters_Pkf	71
4.5.2.2 PK_function	71
4.5.2.3 SensorNode	72
4.5.3 Cluster head	73
4.5.3.1 CLusterHead	73
4.5.3.2 JPCD8544	74
4.5.3.3 PK_function	74
4.5.3.4 Parameters_Pkf	74
4.6 Code Path for SVN	74
<i>Chapter 5</i>	75
5.1 Graphical user interface in Java	76
5.2 Front Panel of GUI design	77
5.2.1 GUI front panel in different Scenarios.....	79
5.2.2 Flow chart of GUI operation.....	80
5.2.3 Graph using XChart	80
5.3 Results.....	82
Summary and Conclusion:.....	85
References:	87

Statement

I hereby assure that I have mastered my master's work without the help of others, and that I have not used any sources other than the sources and aids I have indicated.

All passages, which are taken literally or meaningfully from publications, I have indicated as such with the sources.

The master thesis may not be altered after submission.

Bremen, October 2016

(Name)

Abstract

There are different factors which can affect any wireless sensor network; energy cost for communication is one of the major problems. It is possible to reduce the communication volume by implementing the in-network-processing (INP) algorithms directly onto the nodes. These algorithms are used to estimate the parametric model for the spatial field of the measured properties (e.g. temperature); there is also one possibility to estimate the locations of the maxima of field.

These algorithms are usually tested and implemented on the main server not on the real hardware systems. There are several reasons but the most important two reasons are given below.

- Repeatability of inputs data is required for the comparison and implementations of different algorithms which is not feasible with the actual measurement from sensor nodes.
- For the implementation of INP algorithms, we required higher number of nodes e.g. 100 or more. For testing different algorithms reprogramming of such high number of nodes is not feasible.

In this thesis the main goal is to develop a test platform which can overcome the above mentioned problems. Here SunSpot sensor nodes from Oracle are used as hardware. Sensor nodes are programmed in such a manner that they can playback the data which is stored in the flash memory of the sensor node. There is a host application which runs on the PC with basestation. This application write data from a text file into the flash memory and read data on to the basestation.

Real hardware will be implemented only cluster nodes of the network. There is one cluster head which is used to send commands and data received from basestation to a specific node. On node Predicted Kalman filter is implemented with predictor. While on cluster head there is only predictor implemented.

A GUI will be designed which is used to upload the data from basestation to directly onto the nodes. This will be done using the mac address of the each node.

At last LCD will be interfaced with the each node using a serial protocol. A driver will be written in Java to control the LCD. This will display the status of each node in case of receiving and processing data with the node.

Acknowledgement

Perfection is not attainable, but if we chase perfection we can catch excellence.

Vinc Lombardi

I take this opportunity to express my gratitude and deep regard to Prof.Dr-Ing.Walter Lang the head of department of IMSAS to give me an opportunity for doing my master thesis under his supervision. After that I would like to thanks my second supervisor Dr.-Ing.Reiner Jedermann who gives me an opportunity to carry on my thesis after finishing my project. During my whole time period of thesis his presence and involvement in the thesis give me energy to work and implement the idea into reality.

At the end I would like to thanks my parents my whole family and my friends for giving me courage, love and backing me during the ups and downs in the thesis.

TOQEER RAZA

Chapter 1

INTRODUCTION

The chapter includes the basic idea behind this thesis. How the idea was implemented and what was the motivation behind this idea.

1.1 Theoretical background

Wireless sensor networks (WSNs) are widely used in different areas and also lots of research work is going on in this area. WSNs are comprised of nodes which are battery powered. Each node consists of different sensors integrated with a node. These sensors are used to monitor different phenomenon such as temperature, humidity and air flow etc. Sensor nodes are equipped with small batteries so energy consumption is an important factor. So it is important to make WSN more energy efficient. To make WSN energy efficient we must consider the efficiency of whole network not on a single node [27].

There are different factors which can consume energy in the network but most significant effect is the wireless communication effect. If we make a comparison between a computation and transmission then we come up with a huge difference. A single bit transmission consumes much more energy as compared to the 32 bit computation. WSNs have used different approaches to reduce energy lost by communication. These approaches are used in such a way that data is compressed so minimum energy is lost due to communication effect.

For this an approach named as Predictor Kalman Filter is used. This approach is distributed in leaf node and cluster head. The main algorithm including Kalman filter and predictor is implemented on the leaf node. While on the other hand cluster head has only predictor. In this technique when there is no update from the leaf node to cluster head then cluster head will predict the future value. This value is based on the previous data available to cluster head. Cluster head predict the value within a tolerable range [16].

In this thesis it was required to develop a prototype to demonstrate the distributed algorithm on real network. The basic idea was to use Sun SPOT kit and also to use the simulated temperature data which was saved in a text file. For algorithm implementation we have chosen a PKF (Predictor Kalman filter) which was implemented in Matlab.

1.2 Motivation

Algorithms which are implemented on wireless sensor networks are usually implemented on the server side. These algorithms are not implemented on the real hardware. There are some reasons which make them difficult to implement these algorithms on real hardware. For testing of algorithms and comparison it is important that we have repeatable data. This cannot be achieved by using actual measurements from sensor nodes. The second reason which is also made these algorithms implementation difficult is the number of nodes. If we have lots of nodes then it is difficult to reprogram to implement the algorithm.

These difficulties were handled using a simulated data instead of using real time data. The testing of algorithm is done using a single node and a cluster head. The algorithm was not written in Java so this algorithm which was written and mentioned in [16] is converted into Java.

1.3 Thesis Implementation Structure

There are two block diagrams which are given below. These block diagram shows how overall implementation was done.

1.3.1 Average calculation

The first block diagram is the starting point. This implementation was done for testing purpose. It is the building block for final implementation of algorithm. From block diagram it can be seen that the first block shows the PC section. This section was the main section where the application runs to communicate and for data transmission in the network. The second block shows the base station which is connected with the PC with USB cable. Base station transmits the temperature data which was read from text file using desktop application.

The third block is named as cluster head which is used to provide a mean to communicate with other nodes. Cluster head has programmed in such a way that it distributes data according to the node address. There are total three nodes which are used to communicate with cluster head. Each node receives different data sorted by desktop application.

In the receive section each node sends single value to cluster head which add three values and calculate the average of three values. This Average value is then transmitted to base station which is displayed on the console of desktop application.

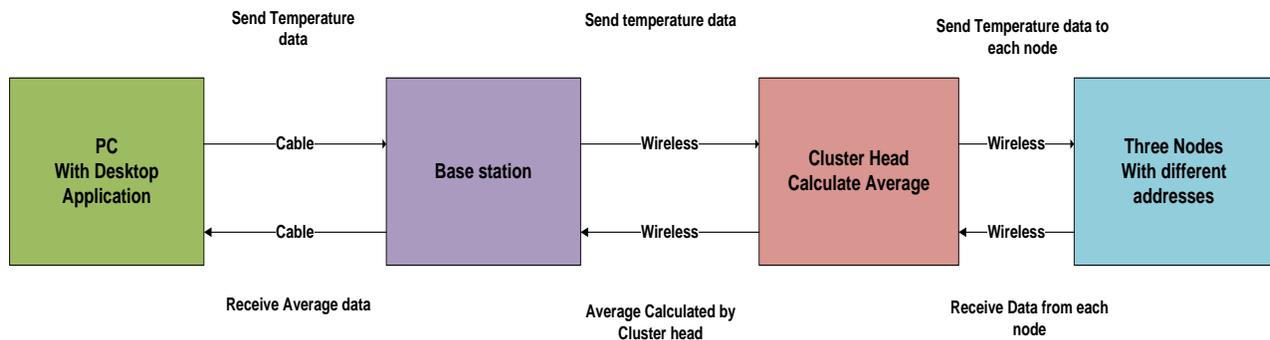


Figure 1-1 Block diagram for Average Calculation Implementation

1.3.2 PKF Algorithm Implementation

The second section shows the implementation of Predictor Kalman filter (PKF). This algorithm was implemented in Matlab. The detail about this algorithm is discussed in chapter 3. We have converted this algorithm in Java and implemented it. In this section there is one cluster head and one leaf node. The PKF algorithm is implemented on leaf node. While on the cluster head there is only predictor implemented. First the data is sent to the node through cluster head.

This data is stored in the flash memory of leaf node. When a command is sent to receive data then data is sent by leaf node as shown in the last block. The data is sent by leaf node only when there is new update available, otherwise the cluster head will predict the new value using previous value and sent it to the PC through base station. This is displayed on the GUI designed on the desktop application.

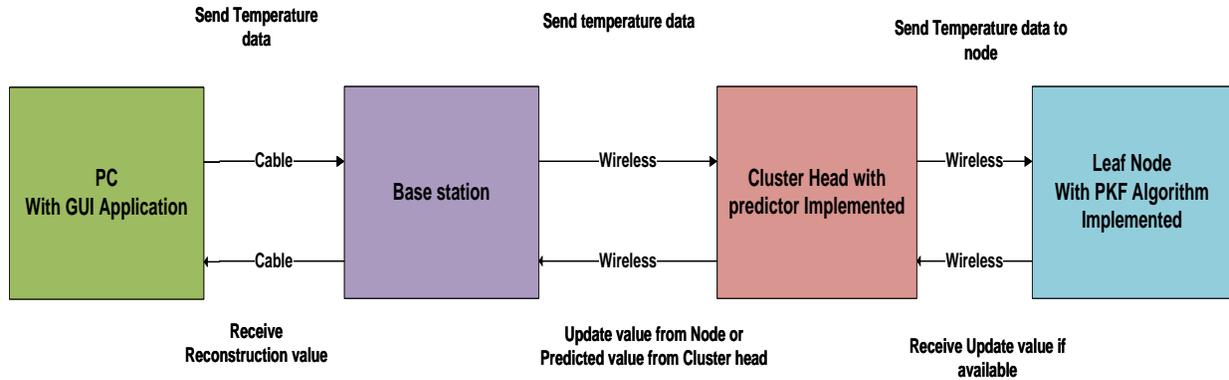


Figure 1-2 Block diagram for PKF Algorithm Implementation

1.4 Organization of report

This is an introductory chapter the report further organized in following way

- Chapter 2: This chapter describes about the hardware including Sun SPOT kit and Nokia 5110 LCD.
- Chapter 3: This chapter explains about Predictor Kalman Filter (PKF).
- Chapter 4: This chapter explains in depth detail about software section.
- Chapter 5: This chapter explains about the GUI and results.

Chapter 2

Hardware

In this chapter explanation of the hardware is given in detail. Explanation of wireless sensor node and LCD interfaced is given in this chapter

1-SunSpot 2-Nokia LCD

2.1 Wireless Sensor Network

Wireless Sensor networks are used to implemented sensors in vast area of technologies. It was impossible to think about such areas a decade before. These fields may include medical and health sector, security, scientific research areas etc. There are two basic approaches which are used nowadays in wireless sensor networks. These approaches can be differentiated on the basis of sensors implemented on the node. There are nodes which are based on single sensor on the other hand there also nodes having multiple sensor integrated on a node.

Wireless sensor nodes are made depending on the requirement of the application. There are many places where we have to monitor more than one parameter. In such places if wireless node with single sensor is used then we need more than one sensor node to monitor the required parameters which can increases the complexity of wireless sensor network [1].

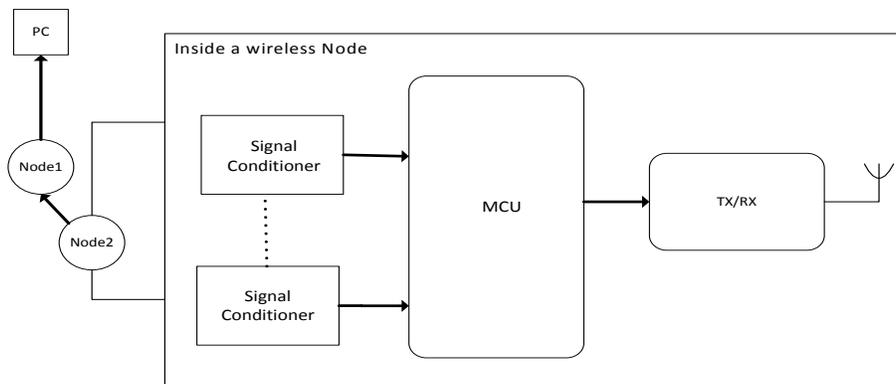


Figure 2-1: Inside of a Wireless Node [1]

When there are multiple sensor implemented in a wireless sensor node then it should be keep in mind that we have to implement separate signal conditioning for each sensor depending on the type of sensor used.

2.1.1 Gateway

Gateway provides a mean of communicating wireless nodes with user end application. With the help of gateways user can get data from nodes in the user defined application. On the other hand there is also a possibility to send commands to nodes using this gateway [2].

2.1.2 Cluster head

Nowadays implementing wireless sensor networks in clusters form is very popular. This technique is used to make network more energy efficient and extend network lifetime. Clusters are beneficial in large scale network environments. Cluster heads are usually communicate with sensor nodes and base station but in large scale clusters are also used to communicate with other cluster heads.

Cluster heads have different function as compared to each node. These cluster heads are used to collect data from all other nodes in the cluster and forward it to the basestation [3].

2.2 Sun SPOT

Sun SPOTs (Sun Small Programmable Technology) are wireless sensor devices. These devices are powered up by small batteries. One of the best things about these small devices is that, these devices are programmed using Java a high level language. There is no operating system used instead of that Java micro edition with the help of virtual machine is implemented directly onto the processor. The virtual machine which is used is named as Squawk. With the help of this virtual machine you can run multiple applications.

The development kit has two main parts one is called basestation other is called eSPOT board which is also called the main board. This board is consists of main board a battery and a daughter board named as eDEMO board. The main board consists of 400 MHZ ARM 926ej-S Processor AT91SAM9G20. It is also based on a radio receiver which works on IEEE 802.15.4(CC2420) [4].

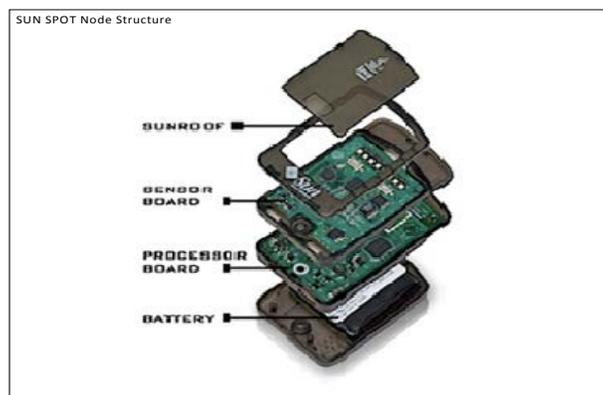


Figure 2-2 SUN SPOT node [5]

As we can see from the above figure that SUN SPOT node consists of different parts first of all there is a plastic sheet which covers the area of sensor board or eDEMO board, under the sensor board there is a main board which consists of main ARM processor and last but not the least there is a battery below the main processor board.

There are lot of peripherals which are interfaced with the main processor. These peripherals include USB Port, Serial peripheral interface(SPI) controller ,programmable I/O controller , TWI two-wire(I2C) etc.

2.2.1 Communication of Main Board

There are two main communication protocols which are used by the main board processor.

- USB
- SPI

2.2.1.1 USB

There is a mini USB connector which is used to program a Sun SPOT node. While the same USB connector is used to communicate host application with basestation SPOTs.

2.2.1.2 SPI

Sun SPOT node has different communication protocols but SPI is the basic communication protocol, this is used by main board for on board devices communication. This protocol is also used to communicate with other daughter boards such as eDEMO board. Radio receiver IC CC2420 and power controller which are integrated on the main board have a communication channel provided by SPI.

2.2.2 Operation mode of Sun SPOT

Power conservation is very important for battery operated devices so nowadays most of wireless node devices have power conservation phenomenon built in there firmware. Sun Spots also have this phenomenon which is used to conserve the battery of Sun SPOT and make it feasible for long period of time operational. Depending on the operation there are three basic operation mode which are used by Sun SPOTs

- Run
- Idle
- Deep-sleep

2.2.2.1 Run

Run operation is the basic operation of Sun SPOT. This operation is also most power consumable operation of Sun SPOT. In this operation all processes are working and radio is also running. Main processor consume round about 70mA to 120mA power. On the other hand when daughter board such as eDEMO board is running this power consumption will increase round about 400mA.

2.2.2.2 Idle

As compared to Run mode of operation idle mode is less power consumable mode. In this mode the clock of ARM9 processor clocks are off while the radio is also shut off. The minimum power consumption of this mode is round about 24mA.

2.2.2.3 Deep-sleep

Among all modes Deep-sleep mode has less power consumption mode. In this mode all regulators are shutoff. There are few which are not shutoff such power control Atmega and pSRAM. One condition for switching to Deep-sleep mode is that radio should be off. If the radio will not off it will not possible to switch from normal mode to Deep-sleep mode. The power consumption of this mode is about 32 μ A. It is very important to keep in mind the startup time which is used to bring back the Sun SPOT from Deep-sleep mode to its normal operation. Typical time for startup is ranges from 2msec to 10msec. When USB is connected or external power is supplied then Deep-sleep mode cannot be activated. Activation of Deep-sleep mode can be done by different methods. This can be done using through programming or by

pressing and holding the attention button for 3sec. The transition of different power consumption modes can be seen through the diagram given below

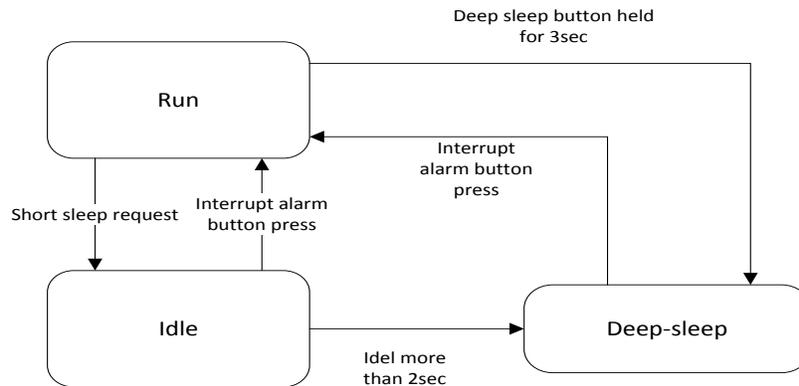


Figure 2-3 Sun SPOT operation modes [4]

2.3 LCD

The second hardware which is used is Nokia 5110 LCD. This is a monochrome LCD with only a single chip driver. This LCD has different properties which make it feasible for small applications. Its output can be displayed on 48 rows and 84 columns. The voltage level varies from 2.7 to 3.3V.



Figure 2-4 Nokia 5110 LCD [6]

This is based on PCD8544 low power LCD controller/driver. All necessary functions which are used to drive this LCD are built in on a single chip. This type of manufacturing helps in providing low power consumption [6].

2.3.1 Communication protocol for LCD

Nokia 5110 LCD works on serial communication. This LCD use SPI communication protocol to receive data for display and commands from microcontroller. There are two types of data set which is send to LCD. This data may be a command or data set to display and can be distinguished by using a GPIO pin [7].

2.3.1.1 SPI (Serial Peripheral Interface)

There are two types of serial communication that can be

- Asynchronous Serial Communication
- Synchronous Serial Communication

Asynchronous Serial Communication

This is a simple form of serial communication in this form of communication there are two pins which are normally named as TX and RX. One is used for transmission while other is used for receiving data. In this type of communication the data sending is not controlled. There is technique which is used to communicate two systems with slightly different clocks. In this method a start and a stop bit is send this will synchronize the data at receiver end [8].



Figure 2-5 Asynchronous Serial Communication [8]

Synchronous Serial Communication

Synchronous serial communication works differently as compared to asynchronous serial communication. SPI bus is one example of synchronous serial communication. SPI bus has two different lines for data and clock as shown in the figure given below.

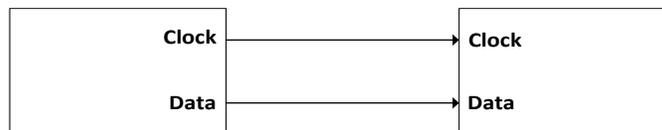


Figure 2-6 Synchronous Serial Communication

In SPI communication there is always a master and slave combination. The device which generates clock is called master while the other device is called slave. Basic implementation is shown below

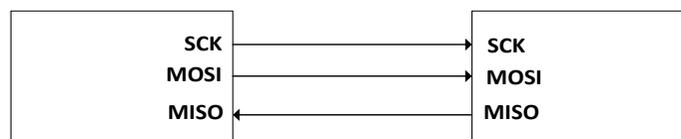


Figure 2-6 SPI communication

In SPI communication multiple slaves can be connected. This can be done using pin named as CS chip select or SS slave select. There are two methods which are used to select the slave for sending data. As we can see from the figure given below every slave has its own SS pin which is connected with the master.

This is the most general way of connecting multiple slaves with a master in SPI communication. The method of implementation is very simple when a data is sent to a specific slave then SS pin of that slave must be low, while on the other hand all remaining slaves SS pin must be high.

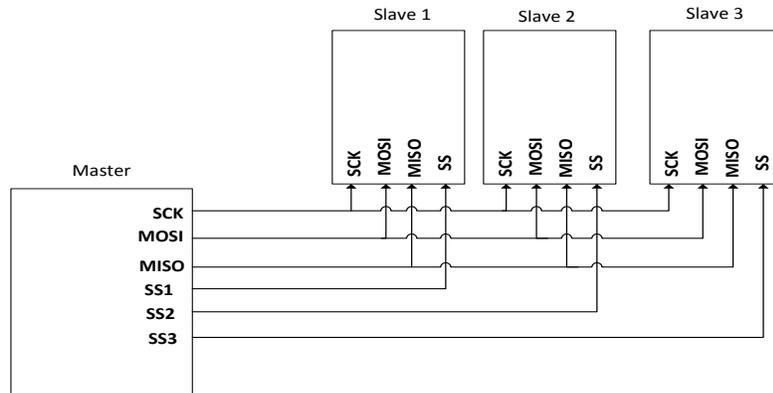


Figure 2-7 Multiple slave for SPI

2.3.2 SPI communication in Sun SPOT

In Sun SPOT SPI communication is done using SPI dedicated pins. But these dedicated pins are available only on main Sun SPOT board. But these pins are only used to communicate on board devices or external board such as eDEMO board. So there are no SPI pins available on the eDEMO board for communication with LCD. Instead of using dedicated SPI pins here we have used Sun SPOT digital pins available on eDEMO board to interface with Nokia 5110 LCD. This technique is also called SPI bit bang technique. The eDEMO board top view can be seen in the figure below

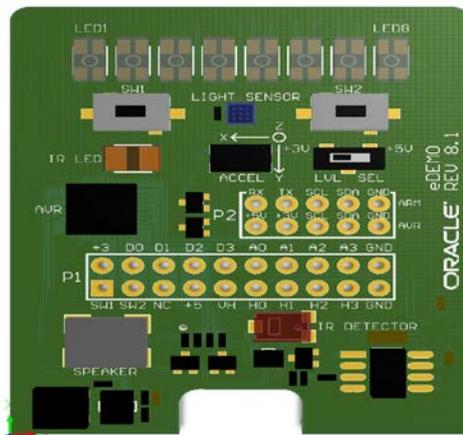


Figure 2-8 eDEMO board top view [9]

It can be clearly seen from the figure that there are different communication protocols pins available on this board SPI pins are not there. So digital pins available in P1 connector are used for LCD interface.

2.3.3 Hardware connections between LCD and Sun SPOT

Hardware connection between LCD and Sun SPOT can be subdivided in two parts

- **NOKIA 5110 LCD Pins**
- **Sun SPOT Pins**

2.3.3.1 NOKIA 5110 LCD Pins

There are total eight pins on Nokia 5110 LCD. These pins must be properly connected with Sun SPOT to get display. Every pin has a specific function each pin is described below

RST

This is an external reset pin available on LCD.

CE

This pin is called chip enable. This is used to control pin operation.

DC

This pin is used to select between data and command. If it is low this means that data is command while if it is high the data is for writing.

DIN

This pin is called serial data line to receive serial data.

CLK

This pin is called serial clock pin. This pin used to synchronize the data with clock.

VCC

This pin is connected with the supply voltage. The voltage ranges from 2.7 to 3.3V.

LED

This pin is used to switch on the background light of LCD.

GND

This pin is connected with ground pin of device to be connected [6].

2.3.3.2 Sun SPOT GPIO and High current Pins

There are total five GPIO (general purpose input output) pins are available on P1 connector of eDEMO board. Among these pins pin number D4 is not connected on the new Sun SPOT board. Although it was connected previously. These pins also have dual functionality for example D0 and D1 also act as UART pins TX and RX. Similarly other two pins D2 and D3 are used for I2C communication. Connecting Sun SPOT with Nokia 5110 LCD we need five digital pins. But unfortunately P1 connector has only four pins available, so for that reason we have used a high current output H0 to complete this connection. The connection between Sun SPOT and Nokia 5110 LCD can be seen in the figure given below

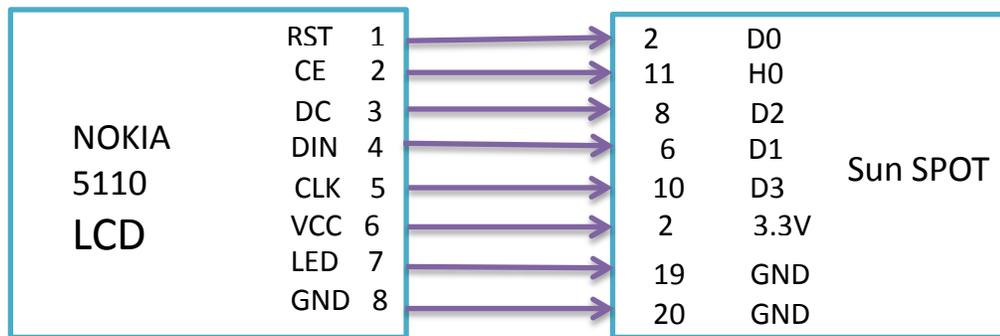


Figure 2-9 Connections between Sun SPOT and LCD

The reason to connect H0 pin with chip select pin that there is only one slave connected.

2.3.4 Software for Driving LCD

The software section describes about the detail how Sun SPOT is programmed according to LCD. Due to unavailability of SPI hardware pins on eDEMO board digital pins are used to communicate with LCD. An open source driver library was used to program. This library was actually designed to interface Arduino with LCD. This library is modified according to Sun SPOT requirements.

2.3.4.1 Important methods (functions) for driver library

There are lots of methods which are used in this library in this section we will describe some important methods of this library which are used to communicate with LCD.

2.3.4.1.1 LCDInit

There are two methods which are used to initialize the LCD. These two LCDInit methods are used for setting the pins according to requirements.

LCDInit (Default GPIO)

This method is used to initialize using the default GPIO pins like in this case Sun SPOT default GPIO pins are used. There is only one parameter for this method which is the contrast value for LCD.

LCDInit(User Defined)

This method is used to set the GPIO pins according to user requirements. There are following parameters in our case

- IOPin dinPin
- IOPin sclkPin
- IOPin dcPin
- IOPin rstPin
- IOutputPin csPin
- Contrast

In this method the contrast value is the value which affects the display. To see the characters proper value for contrast ranges from 40 to 50. This is an integer value. There are different things which are implemented through this method the most important things are given below

Text size

This method is also used to set the text size to display normally the text size value should be 1.

Text Color

For this LCD the color is black which is also set here.

LCD cursor Set

When the LCD is initialized then the cursor is adjusted at the zero position.

These are basic functions which are implemented through this method. In the similar way the digital pins of Sun SPOT are also set as output pins in this method. For making digital pin as output pin following built in function of Sun SPOT library is used

```
pinD0.setAsOutput(true);
```

Instead of this initialization there are few commands which are sent to the LCD, such as the LCD normal mode command is sent to LCD to adjust the normal mode. Similarly command is sent to set the display to normal display etc.

2.3.4.1.2 Power

This method was not available because on wireless node we cannot add a math library with built in power function. Parameters for this method are two integers and the return type of this method is also an integer. The first parameter is used as base value while the second parameter is used as a power. So we can calculate power of integer with respect to parameters given.

This method also uses the functionality that if we give power of zero of any integer it will return the value one. This method is used in character draw function where it will display the character.

2.3.4.1.3 *shiftout*

This is the main method which is used to send data using bitbang technique on the GPIO pins of Sun SPOT. This method use following parameters

- IOPin dataPin
- IOPin clockPin
- int bitOrder
- long vals

The data rate depends on the CPU clock. Which varies according to the target platform.

2.3.4.1.4 *showMe*

This method is used to display the data you want to display on the LCD. There are two parameters which can be modified according to the requirement. Here we have to display the wireless node name and the address of the node for that reason two parameters are given as string.

In this method there are different built in methods used. These are as follow

- LCDClear
- LCDDrawString
- LCDDisplay

The first method named as LCDClear is used to clear the LCD. This will remove anything which is previously display on the LCD.

The second method named as LCDDrawString is used to define the string position where we have to draw our required string. Here we have to give column and row number to display the required string.

The third and last method which is named as LCDDisplay is used to issue command to display the data received from the method LCDDrawString.

2.3.4.2 Code Location on SVN

The code is also available on SVN. The following path will lead to the different files.

S:\raza\Code2\Sensor_ClusterHead2\src\de\imsas\inpDemo

The file name for LCD code is “**JPCD8544.java**”.

Chapter 3

Distributed Algorithms Predictor Kalman Filter

In this chapter explanation of the distributed algorithms is given and conversion and implementation of PKF (Predictor Kalman filter)

3.1 Distributed Algorithms

Wireless Sensor Network is a latest field where lot of work is going on. There are different aspects which make this field very challenging among these challenges include network optimization and distributed network computing. Wireless sensor network nowadays different as compared to the classical networks [10].

There are lots of reasons due to which distributed algorithms are much more fascinating for designing wireless sensor network protocols. First important reason behind implementation of distributed algorithms is that nowadays it is important for a node to reduce amount of transmitted data. So for a node it is important to performed calculation locally [11].

3.2 Kalman Filter

Kalman filter is used when we are not certain about the data set or our data set is continuously changing. Estimate states by observation of system outputs. One of the basic advantages of Kalman filter is that it uses less memory because it only depends on the previous state so it is not necessary to save all the states. Due to its property of not depending on all states it is quite fast [13].

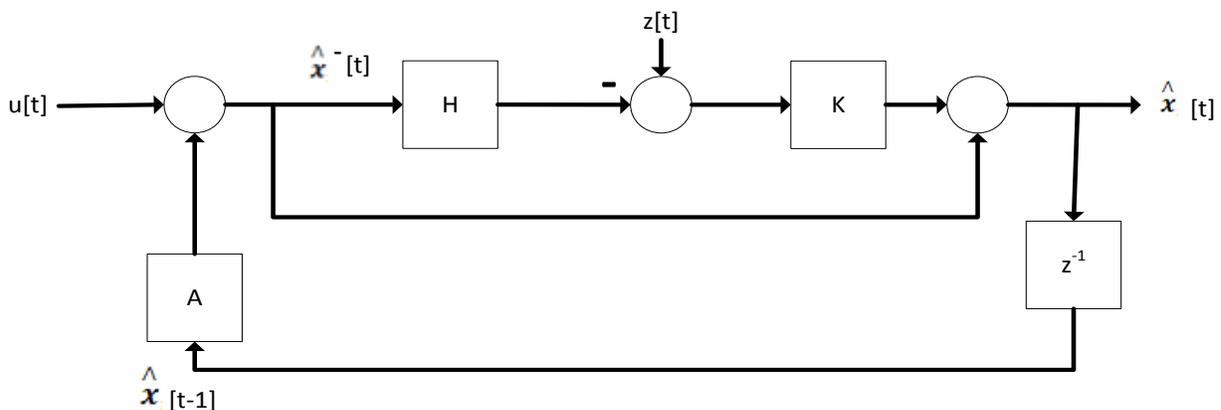


Figure 3-1 Basic Diagram of Kalman Filter

3.2.1 Applications of Kalman Filter

Kalman filter is very important algorithm which is used in different fields according to the requirements of the field such as

- Find location or tracking of an object. This is used to estimate the states of the system. So called as state estimator.
- Computer vision applications
 - Tracking of a particular feature
 - Application in cluster tracking
 - Finding depth from a stereo camera

- Application in Navigation

3.2.2 Basic Equations of Kalman Filter

There some basic equations which are used to implement the Kalman filter these equations are divided in two sections

Predict phase:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t \mathbf{u}_t \quad (3.1)$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_t \quad (3.2)$$

Update phase:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1}) \quad (3.3)$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^T + \mathbf{R}_t)^{-1} \quad (3.4)$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_{t|t-1} \quad (3.5)$$

The above equations are basic equations to implement a Kalman filter. Among these equations there are lots of things which are explained as follow

Variable Name	Explanation
$\hat{\mathbf{x}}$	This variable is used in the above equations is named as estimated value. System state vector.
\mathbf{u}	This variable is called control variable or control input.
\mathbf{F}	This is called state transition matrix.
\mathbf{B}	This is called control matrix.
\mathbf{P}	This matrix is called state variance matrix. It is also named as error of estimation.
\mathbf{Q}	This is also a matrix and this matrix is defined for process variance. This is used to estimate error in process.
\mathbf{y}	This is a variable which is also called measurement variable.
\mathbf{H}	This is a matrix which is called as measurement matrix. Observed output.

K	This is known as Kalman gain.
R	This matrix is used to estimate errors in measurements also known as measurement variance matrix.

In these equations there are some subscripts which are related to time events. These subscripts defined when an event is happened such as if the subscript is $t|t$ then it shows that this is at the current time event step. If the subscript is $t-1|t-1$ then it shows that this step is the previous step or past event similarly if the subscript will be $t|t-1$ then it will show that there is an event which is intermediate step.

There are some important points which should keep in mind before implementing Kalman filter. The Kalman filter plays an important role in removing noise from data. When this is implemented it uses a predefined model of the data set, this predefined model should be appropriate so that it can follow the real system. If the model defined according to data set is not suitable or varies much more than the real data set then Kalman filter will not give the proper results as required [14].

In implementation of Kalman filter it is very important that you break the problem in small parts according to the required situation. Because implementing it on whole problem without breaking into small parts sometimes make the problem more difficult. Maybe sometimes this method will not work at first time but later it will work fine. So initially make simple model.

In Kalman filter noise measurement is very important parameter. So when model a noise it should be according to the filter noise. In Kalman filter noise is assumed to be Gaussian noise. So in the model there error should be according to this noise. Testing of filter is very important before implementation on the real environment. So before implementation the filter it should be tested with some data that it is working according to requirement.

There are different aspects which can be used to refine the filter but basic idea to refine the filter is to change the noise parameters. Because this is the most simple way to refine filter [14].

3.3 Predictor Kalman Filter (PKF)

Wireless Sensor network (WSN) consists of different nodes which are usually battery powered. In wireless sensor network it is very necessary to make it power efficient. Because wireless nodes are used monitor some phenomenon such as temperature, humidity etc. for very long time depends on the application. There are lots of factors which affect energy consumption of wireless sensor nodes but most significant affect is the radio communication among the nodes.

This is also very interesting fact that when a single bit is transferred over a radio transmission it will consume not much more energy as compared to the 32 bit data. It is more suitable to send data collectively as compared to single bit data with less energy consumption. In WSNs there are lots of other

factors which are also responsible for energy consumption such as some energy is also consumed at the startup to switch on radio similarly when there is no data or idle listening, some energy is also consumed during data collision. In recent era of WSN research it is very important to make the network energy efficient to reduce the communication cost.

When there is matter of reducing transmission cost or communication cost then one thing which plays an important role is data compression. Data compression techniques are very popular because these techniques use physical phenomena with respect to inheritance existence and temporal correlation [15].

This should be noticed that there are two paths which are quite different from each other. One is called data compression other is called data aggregation. The main difference between data aggregation and data compression is that data aggregation is complex technique which is used where network is very dense and there is possibility of multihop; routing algorithm is also needed where data aggregation is implemented. The figure below shows a difference between data compression and data aggregation as we can see that for data aggregation we need a routing algorithm as compared to the data compression techniques [18].

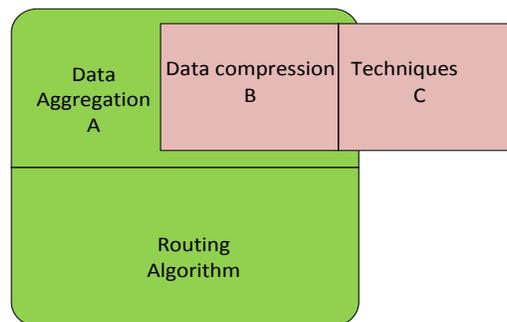


Figure 3-2 Data Compression and Data Aggregation

Data compression techniques are also classified in two categories. These are shown in figure below

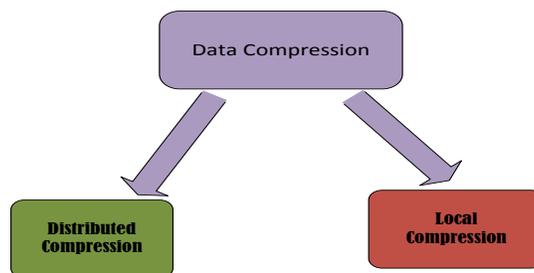


Figure 3-3 Data Compression categories

In these data compression techniques distributed data compression technique is much more network dependent this technique is applied on the dense network where each node cooperate with each other to play a role in data compression. On the other hand local data compression is not network dependent. Local compression can compress data at node level. Local compression either compress packet size or transmission rate.

Predictor Kalman filter is latest technique which is different from DKF (dual Kalman Filters). In DKF approach Kalman filter is implemented on both sides. First Kalman filter is used to remove noise from the node side. Secondly a Kalman filter on cluster head side is implemented which is used according to each leaf node. Cluster head is used this filter to predict the future value of each leaf node. On the other hand leaf node also runs Kalman filter to ensure that the predict value should be according to requirement.

Predictor Kalman filter is different from traditional compression techniques because in traditional compression techniques the focus is to reduce the transmission packet size. While on the other hand in PKF the communication energy is saved by reducing the transmission rate. There is a difference between other predictor schemes and PKF, PKF has a property to use the full system model this will provide more appropriate predictions.

Reduced transmission rate can effect on the quality of reconstruction signal. This means that reconstruction signal is not so accurate. This reduction in the quality of reconstruction can be removed by using a KF in the leaf node. This KF not only enhanced the reconstruction signal quality but also remove the noise from the signal. After this KF implementation reconstruction signal become more accurate as compared to the raw data set values [16].

3.3.1 KF implementation in PKF

In this section how the KF is implemented in the PKF will be described. In KF implementation there are several steps which we have to follow according to the data set. PKF can be equally implemented either the system is time variant or it is time invariant. For the sake of simplicity here time invariant system was selected. KF is used to estimate the improved current state with the help of a current measurement with noise and the previous step measurement. Due to its recursive property this algorithm when implemented on a system having Gaussian noise will generate minimum mean square error.

In PKF there are two phases of KF which are implemented alongside with estimation error. This implementation can be further elaborated using mathematical equations. As described earlier the system for mathematical description is time invariant. This will help in simplicity of mathematical derivations [16].

The true state or actual state of x_k when time is k can be manipulated with the help of the state at time step $(k-1)$.

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \quad (3.6)$$

The above equation can explain how a state at time k can be manipulated with the help of state at time $k-1$ there different elements which are used in the above equation these elements can be explained as follow

A: This is called transition matrix

B: It is called input matrix order is depending on the number of inputs.

\mathbf{u}_k : This is called control input vector used in the equation 3.6.

\mathbf{w}_k : This is responsible of the irregularities of the model which can be shown as

$$\mathbf{w}_k \sim N(\mathbf{0}, \mathbf{Q})$$

Q : This is called covariance of process noise.

When new observation is needed at time interval k then it cannot be calculated directly. To evaluate observation such as z_k we need a true state at that time interval such as x_k observation matrix which is denoted by H and this matrix not in a pure state while having a zero mean white Gaussian noise included in it which is denoted by v_k . The equation for this step is given below

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (3.7)$$

The Gaussian noise can be further elaborated with following equation given below

$$\mathbf{v}_k \sim N(\mathbf{0}, \mathbf{R})$$

R: This is called covariance of measurement noise.

KF is implemented in two steps first the new value have to predict and then it is updated later on. In the first step when we have to predict the new value then it is predicted with the help of previous step estimation. Let suppose we have following notations

Current state: $\hat{\mathbf{x}}_k^-$

Previous state: $\hat{\mathbf{x}}_{k-1}$

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \quad (3.8)$$

It is also necessary to find out the error between the estimate and true state which can be denoted as follow

$$\mathbf{e}_k^- = \hat{\mathbf{x}}_k^- - \mathbf{x}_k$$

This error can be further explained and the uncertainty of the prediction can be found with the help finding covariance of this error which is previously explained by equation written above

$$\mathbf{P}_k^- = E\{\mathbf{e}_k^- \mathbf{e}_k^{-T}\} = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \quad (3.9)$$

The above equations which are discussed in detail show the prediction phase of KF. These equations just give the few steps through which we can just predict a future value. Now the update phase will be discussed in the further section. When the update phase comes this phase use the current measurement which is previously denoted by z_k with the help of previous prediction generates a new improved state value which we can be denoted by $\hat{\mathbf{x}}_k$. This value is also called optimal value. In this update phase idea of weighted average is used to estimate the value. The weight which is denoted by K_k is given to the state which has higher priority. This is also called Kalman gain. The equation for this section can be given below [16].

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1} \quad (3.10)$$

Now the estimated value which is updated from the previous step can be evaluated from the predicted and measured state which is given by the following equation

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \quad (3.11)$$

Now this is the optimal state as described before. Now we can similarly find the error of this optimal state which was previously calculated for the other state

$$\hat{\mathbf{e}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k$$

This is called optimal state error. Now if we take the covariance of this error this will tell about that how much will be there uncertainty in this estimation. The equation is given below

$$\mathbf{P}_k = \left\{ \hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T \right\} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \quad (3.12)$$

KF can be reached to a steady state after number of steps. When KF reached to a steady state then it will simplify the work now Kalman gain and covariance will try to reach constant values. These values will converge as follow

$$\mathbf{K}_{k \rightarrow \infty} = \mathbf{K}, \quad \mathbf{P}_{k \rightarrow \infty}^- = \mathbf{P}^-, \quad \mathbf{P}_{k \rightarrow \infty} = \mathbf{P}$$

3.3.2 Implementation of PKF

In this section PKF implementation will be explained. As in previous section all the steps are explained thorough which a KF filter can be implemented. Here procedure of PKF will be explained in detail, this means how PKF implemented in a node and on a cluster head. In implementation of PKF this thing kept in mind that the main idea of this algorithm is to reduced communication between a leaf node and the cluster head. This means that cluster head should predict next value with tolerable error so that this communication between cluster head and the leaf node is reduced as much as possible.

Leaf node section is cannot be left without filter. As we know that when leaf node have observations these observations have some noise, to remove this noise first of all we have to implement KF on the node section. After this filtration process now the leaf node has optimal values. As previously discussed now the cluster head will run a predictor which is used to reduce communication energy cost. This will try to follow the optimal values of node. On the other hand leaf node will also run a predictor so that the quality of predicted values remains constant. On a leaf node it is compared that predict value will remain within a tolerable threshold. When this value exceeds from a set threshold then the leaf node will transmit current state to the cluster head side so that vector optimal value quality will remain constant [16].

Before implementing this it is obvious that model parameters which are calculated according to the given model must be known to both ends. This means that model parameters should be there on leaf node side and the cluster head.

We can elaborate with a simple example suppose that there is a leaf node as shown in the figure below with letter A. First this node will run a KF to remove the noise in the data. After this implementation of KF we will get an optimal value which is denoted by \hat{x}_k . While on the other hand cluster head which is denoted by B in figure runs a predictor [16]

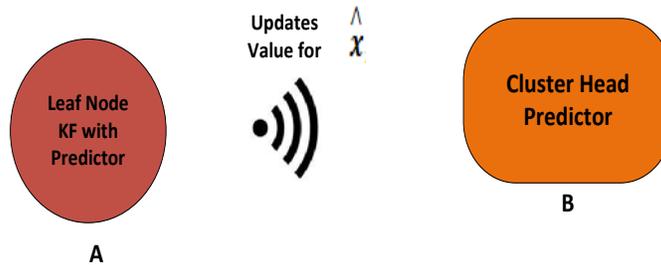


Figure 3-4 Example of a Leave node and Cluster head

This predictor can be implemented on the cluster head side using following equation

$$\tilde{x}_k = A\tilde{x}_{k-1} + Bu_k \quad (3.13)$$

Before initializing the procedure first we have to give the initial value to the cluster head sides such as it can be denoted by \tilde{x}_0 . This predictor is working on recursive basis. There will be always some difference in the observed and predicted value. This difference is known as predicted error which can be evaluated

$$\epsilon = H(\tilde{x}_k - \hat{x}_k)$$

Among these values one is the predicted value while other is the estimated value. The predicted observation can be written as

$$\tilde{z}_k = H\tilde{x}_k$$

Similarly on the other hand estimated observation can be written as

$$\hat{z}_k = H\hat{x}_k$$

It is also very important to keep track of the error generated by the cluster head values, to make sure about the reliability of the predicted value from cluster head it is also necessary to run a predictor also in the leaf node side. The error calculated from the node side is compared with a specific threshold value. If this value is greater than the threshold value then it will transmit to the cluster head. The current predicted value both from cluster head and node side which is represented by \tilde{x}_k is replaced by the value \hat{x}_k . Now the final reconstructed value from the cluster head can be written as [16]

$$\bar{z}_k = H\bar{x}_k$$

How the value of \bar{x}_k can be represented as below

$$\bar{x}_k = \begin{cases} \tilde{x}_k, & \epsilon_k \leq \tau \\ \hat{x}_k, & \epsilon_k > \tau \end{cases}$$

Threshold value is depended on the requirement of the data set. So it can be adjusted according to the requirements.

The PKF algorithm can be represented by the block diagram. The block diagram of the leaf node is shown in figure (3-5)

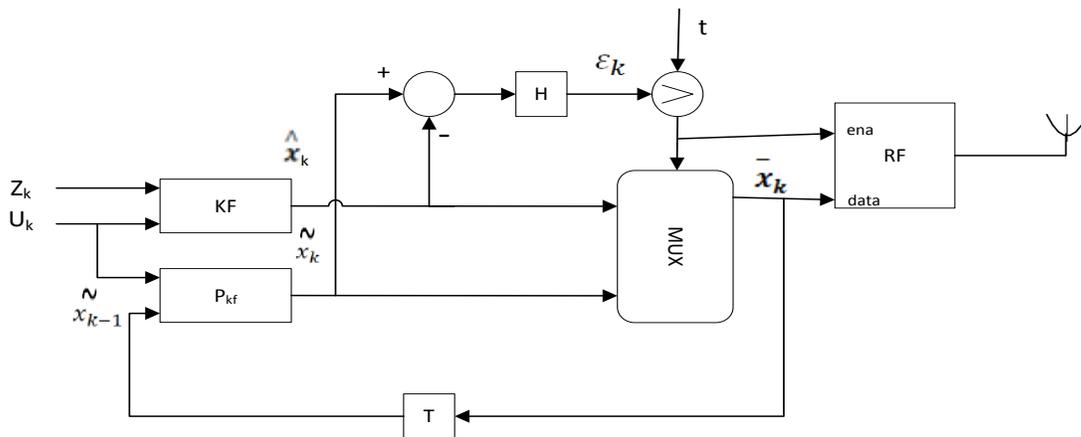


Figure 3-5 Block diagram showing leaf node section

This block diagram depicts the whole process through which a leaf node goes through. To get the optimal value the equations are used from 3.8 to 3.12. After this error is calculated as shown in the figure. This error is compared with the threshold value. This threshold value is set according to the requirements. If this error value is larger than the value of threshold then this value is transmitted to the cluster head.

3.3.2.1 PKF Algorithm Node Section

PKF algorithm on node side is given as below

Step#1 Initialize \hat{x}_0 , \tilde{x}_0 and \hat{P}_0

Step#2

for each z_k **do**

Calculate \hat{x}_k using equations from (3.8) to (3.12)

$$\tilde{x}_k \leftarrow A\tilde{x}_{k-1} + Bu_k$$

if $\epsilon_k > \tau$ **then**

```

Send
 $\tilde{x}_k \leftarrow \hat{x}_k$ 

end if

end for

```

3.3.2.2 PKF Algorithm Cluster head Section

The block diagram of cluster head section of PKF algorithm is given below in the figure (3-6)

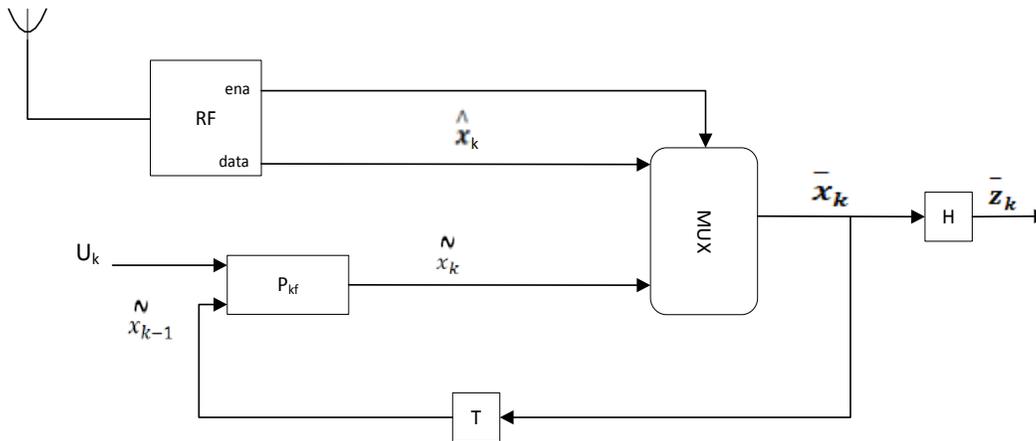


Figure 3-6 Block diagram showing cluster head section

This block diagram shows the cluster head side. In cluster head section there is only predictor implemented. Initial value we have to set same as the starting value of the optimal value. From next value there are two conditions to get new value. First if the update value is available from the leaf node then it will update with that value otherwise the new value is calculated from the equation given below. This value is then transmitted to the base station.

Step#1 Initialize \tilde{x}_0

Step#2s

for each prediction time **do**

if update \hat{x}_k is available **then**

$$\bar{x}_k \leftarrow \hat{x}_k$$

$$\tilde{x}_k \leftarrow \bar{x}_k$$

else

$$\tilde{x}_k \leftarrow A\tilde{x}_{k-1} + Bu_k$$

$$\bar{x}_k \leftarrow \tilde{x}_k$$

end if

$$\bar{z}_k = H \bar{x}_k$$

end for

3.3 Matrix in Java

Matrix calculation is very important in the implementation of PKF (Predictor Kalman Filter) algorithm. Algorithm was written in Matlab where matrix calculation can be done directly. While on the other hand in Java matrix calculation is not so straight forward as compared to Matlab.

Matrix calculation was the key feature in implementing this algorithm. So for implementing matrix we have to utilize some external package which can be easily integrated. For this reason a package named as Jama was utilized. The description about the package is given below

3.3.1 JAMA

JAMA is a mathematical package for Java this package is used to manipulate linear algebra. This package is also very useful to manipulate very dense matrices. Although there are two types of matrices which can be calculated one is called real matrices and other is called complex matrices. This package is very efficient and can any type of real matrix, but at the moment this package cannot be used to calculate complex matrices.

This package is not like a simple array class which is used to calculate simple basic operations. So for that reason the basic array operations which are implemented element wise such as log, sine etc. are not implemented. The below is given a table showing capabilities of JAMA package [12].

Object Manipulation	<p>There are basic object level manipulation can be done using this package such as</p> <ul style="list-style-type: none"> • Constructors • Set elements • Get elements • Copy • Clone
Elementary operations	<p>Some elementary operations which can be easily performed with help of this package are</p> <ul style="list-style-type: none"> • Addition • Subtraction • Multiplication • Transpose • Norm
Equation Solution	<p>With the help of this package there is possibility of different equation solutions such as</p> <ul style="list-style-type: none"> • Nonsingular systems • Least squares

Derived Quantities	Derived quantities are important and calculation of these quantities is not so simple so it is very helpful to use JAMA package to calculate these quantities that may include <ul style="list-style-type: none"> • Determinant • Rank • Condition number • Inverse
---------------------------	---

3.4 Implementation of PKF in Java

In the previous sections detail about PKF algorithm is explained. In this section how the algorithm was implemented in Java will be explained. There are different steps which were taken to convert this algorithm in Java. The block diagram given below will explain the steps involve.

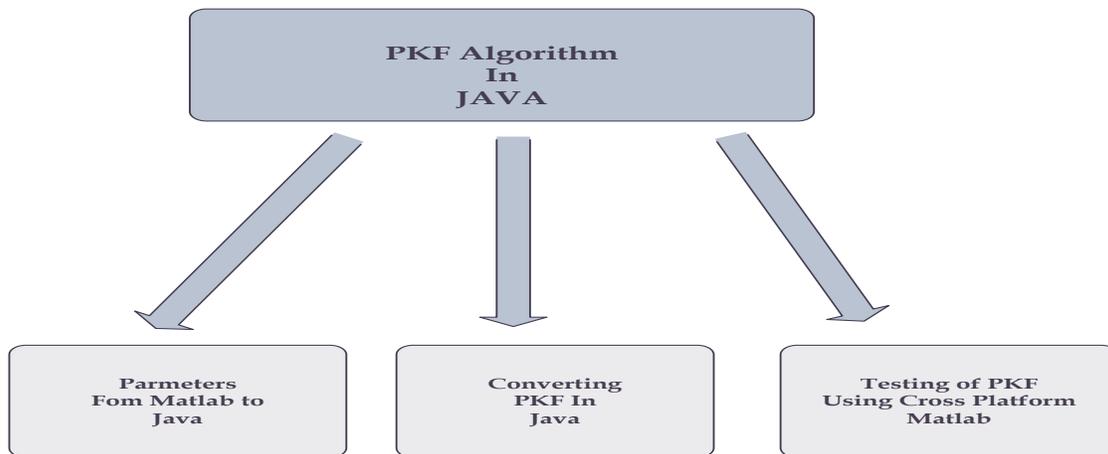


Figure 3-7 Steps of converting PKF Algorithm in Java

3.4.1 Parameters from Matlab to Java

First step was to calculate the parameters for PKF using Matlab. These parameters are calculated according to the data set. For parameters calculation system identification module of Matlab was used. We have temperature data set saved in a text file. This temperature data was used to calculate the parameters. There are different parameters which are calculated by this method these are

A: This parameter which is calculated by using Matlab is called transition matrix. The order of this matrix is 2×2 .

U: This matrix is called control input matrix. This matrix which is used must have order according to the size of data set provided. Here in our case the order was 107×1 .

H: This is called observation matrix. This matrix consists of one row and two columns.

K: This matrix is called Kalman gain matrix. The order of this matrix is 2×1 .

Q: This is called covariance matrix. The order of this parameter matrix is 2x2.

R: This is called measurement variance matrix. This matrix consists of one row and one column.

M: This matrix is used in Kalman gain calculation. The order of this matrix is 2x1.

x0: This is also a matrix. This matrix is used to save the initial value for the calculation of predicted values. The order of this matrix is 2x1.

These are the parameters which are calculated using Matlab. These all parameters are data set dependent this means that whenever the data set is changed these parameters will also change. The whole procedure can be explained by the flow chart given below

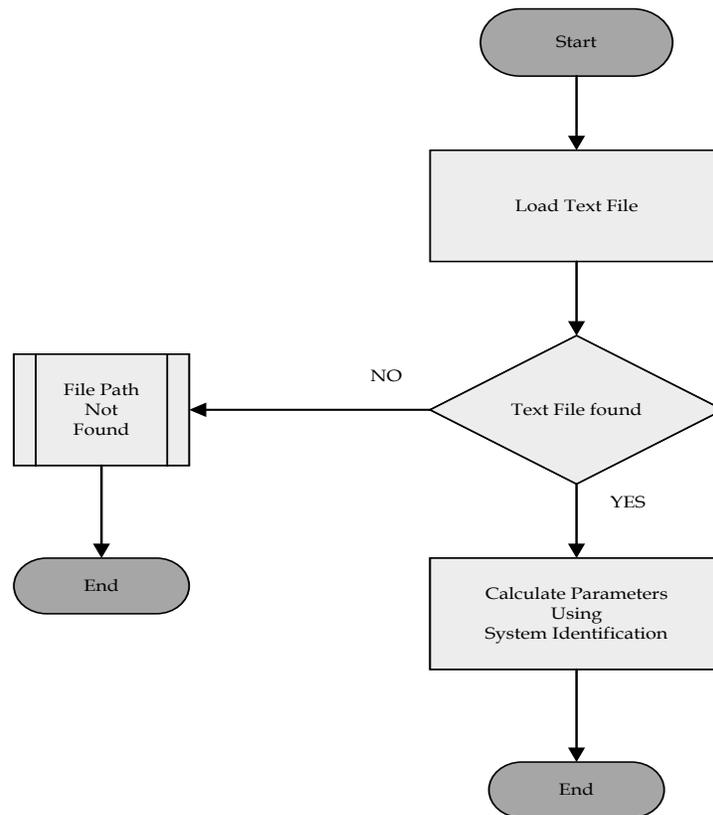


Figure 3-8 Flow chart describing Parameters Calculation in Matlab

This flow chart explain how this whole procedure works. First of all in the Matlab we have to give path of our text file. This text file contains the temperature data set. First Matlab loads the text file. Here in the next section of flow chart there is a conditional box this box checks shows if there is the exact path of file then it will move further and calculate the parameters using system identification.

3.4.1.1 Explanation of Java Class using UML

In this section we will explain the parameters calculated by Matlab implemented in Java using UML. First little bit explanation of UML.

What is UML?

UML means Unified Modeling Language. This language is quite different from basic programming languages such as C, Python and Java. This language is more pictorial language as compared to script language. This language is visual and it is used to document the software section. Although it is not a programming language but there are different tools available, which can be used to convert UML diagram into code.

The UML modeling plays an important role to see a system in different ways or perspectives. There are different perspectives, which are very important in designing software. How these perspectives are subdivided with respect to the UML model. There are following design perspectives for the UML given below [19]

- Design
- Implementation
- Process
- Deployment

Design:

When we define a system then design is used to represent the system using classes and interfaces. Design is supported by UML diagram with the help of Class diagram, object diagram.

Implementation:

Implementation perspective is used to combine different components to make an overall system. A system consists of different small parts which must be interconnected to make an overall system. UML component diagram plays an important role to use the implementation perspective.

Process:

In every system, there is specific flow of the process. This flow defines how the system will perform. To implement this perspective one can use the same patterns as used in design perspective.

Deployment:

When it is necessary to define the hardware of a system, then deployment perspective is introduced. This perspective is used to implement hardware part of the system. To implement this perspective UML deployment diagram is feasible to implement this perspective.

There are different types of UML modeling. The diagram given below gives the description about that

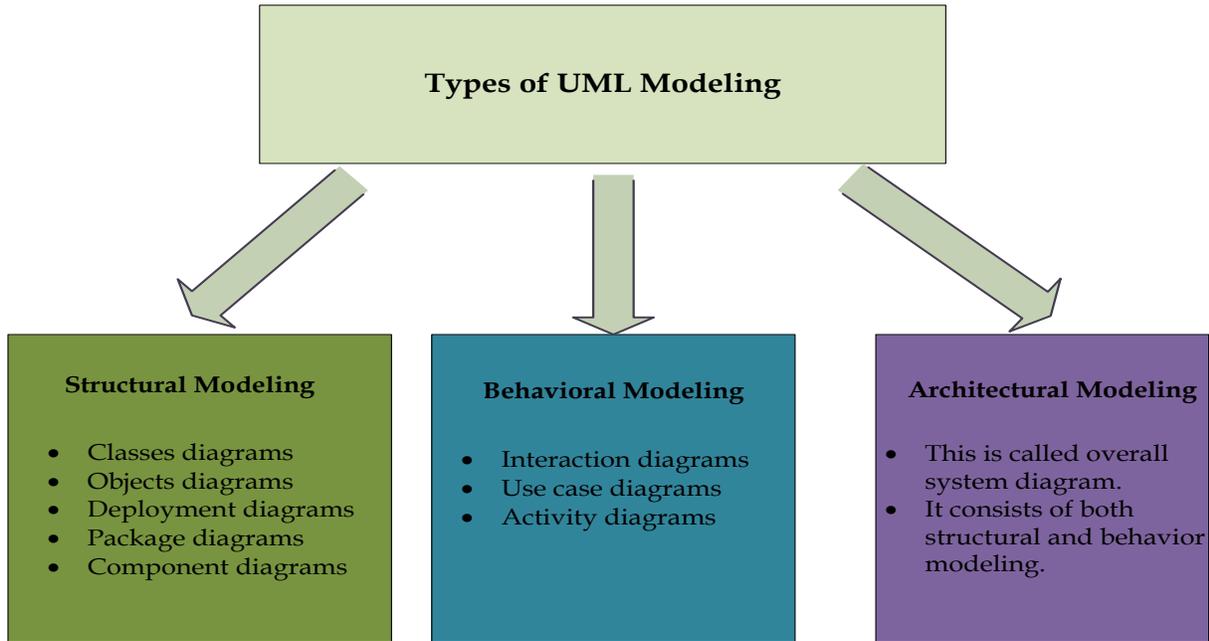


Figure 3-9 Types of UML Modeling

UML Class diagram of Parameters :

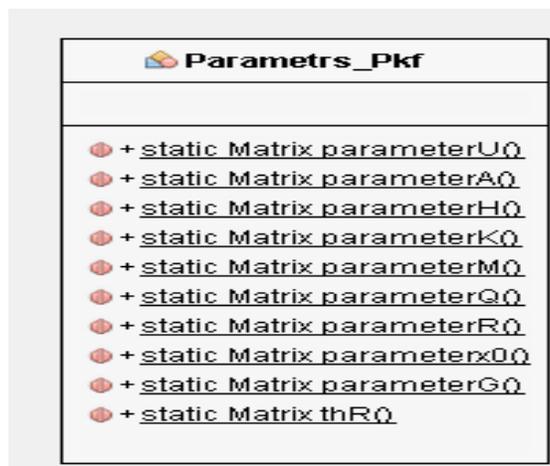


Figure 3-10 UML class diagram of Parameters class

This UML diagram is created by using easyUML plugin for Netbeans. The parameters that are calculated using Matlab are stored in a class file named as Parameter_Pkf. In Java, matrix calculation is not straightforward. In previous section description about matrix conversion in Java was given. We have used a package named as JAMA.

From UML diagram, it is visible that there are different methods, which are used to store parameters. It can also be seen that each method has a return type named as Matrix. Each method return matrix when we will use.

3.4.2 Converting PKF in Java

In previous section detail about parameters conversion from Matlab to Java was discussed. In this section, we will explain about the PKF algorithm conversion from Matlab to Java. In Matlab matrix calculation is not a problem as compared to Java. However, when we use matrices in Java we have to keep track of each thing, because in utilizing matrices we have to introduce two-dimensional arrays. In PKF algorithm matrices are not only used but also involved in equation calculation.

Although there are different methods, which are used to implement the PKF algorithm, but there are two important methods, which are the main methods of PKF algorithm class. These methods are the basic methods which are used to implement the whole algorithm. These two methods are classified according to the requirements.

Each method is used either on a node or on a cluster head. Classification of these two methods is given below

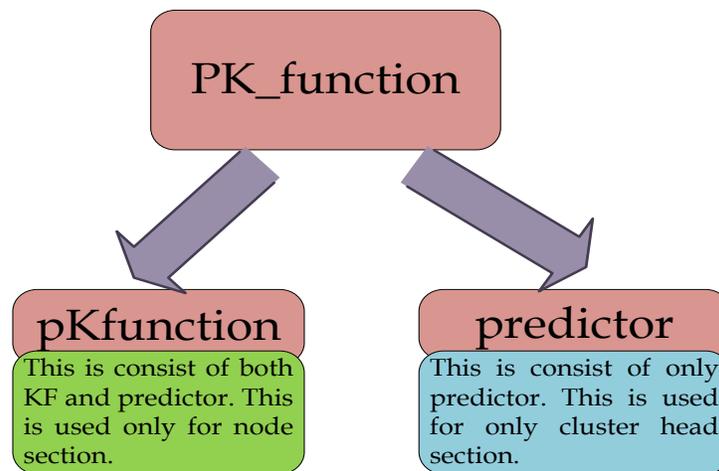


Figure 3-11 Main Methods of PKF algorithm

UML Class diagram of PKF algorithm class:

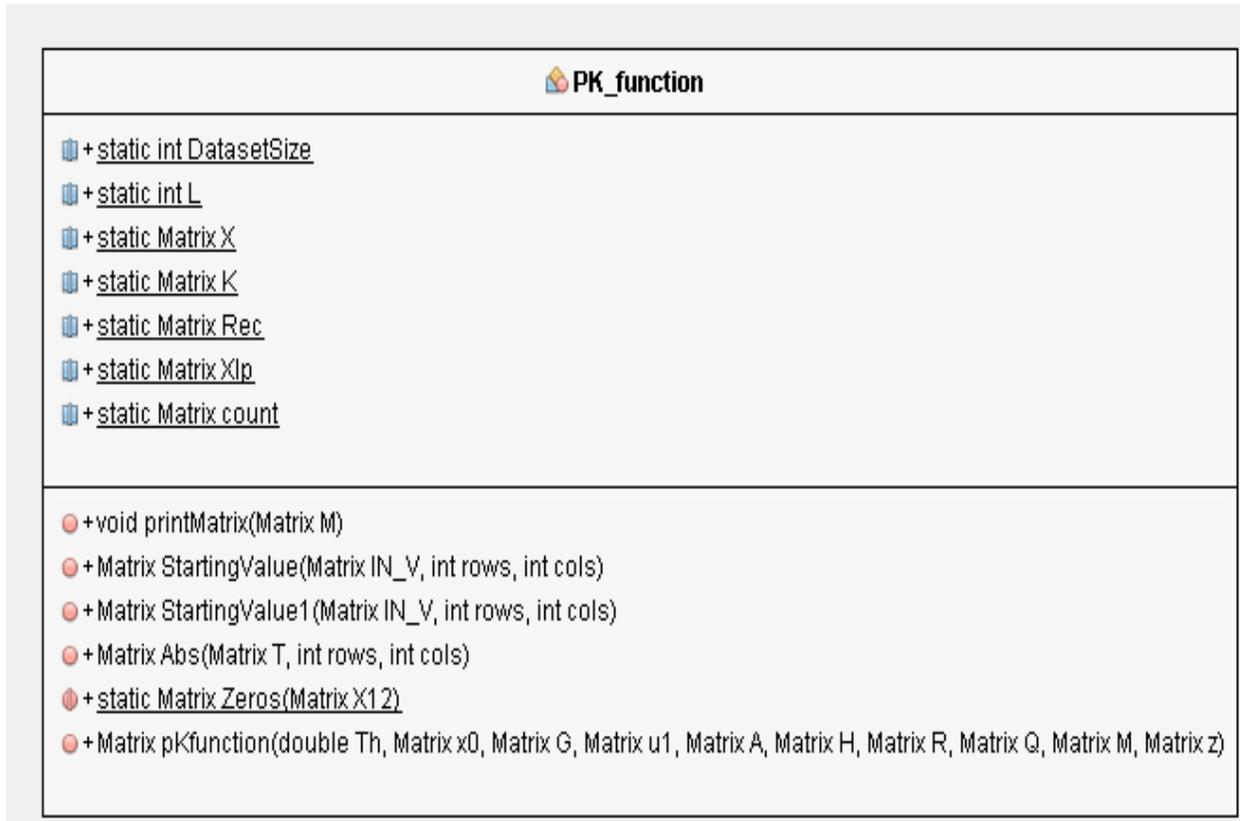


Figure 3-12 UML class diagram for PK_function class

3.4.2.1 Different methods in PKF algorithm class

There are different methods which are used in the PKF algorithm class. These methods can also be seen in UML diagram.

3.4.2.1.1 Starting Value

This method is used to make a matrix of specific rows and columns.

Parameters:

This method consists of three parameters

- Matrix
- rows
- cols

Matrix parameter is the specific matrix which we have to convert to a specific rows and columns matrix. Similarly rows is the parameter to make that matrix according to required number of rows matrix. Cols parameter is also used to make that matrix according to given number of columns. One thing of this

matrix is that it is specified for the rows. This means if a matrix have starting rows zero then this method will be used, or it can also be said that if the number of rows are changing then this method will be use.

Return Value:

This method has a return value as a matrix.

3.4.2.1.2 StartingValue1

This method has same number of parameters and same return value. One thing different in this method is that whenever a matrix which has changing number of columns then this method will be used.

3.4.2.1.3 Abs

This method is used to get absolute value of each element of matrix.

Parameters:

This method consists of three parameters

- Matrix
- rows
- cols

Matrix parameter is the matrix which we need to get absolute values. It gets number of rows and number of columns also as a parameter. This method loops through number of rows and columns and convert each value to absolute value by using math absolute function.

Return Value:

This method also has a matrix as a return value.

3.4.3 Testing of PKF using Cross Platform Matlab

After converting PKF algorithm from Matlab to Java, first task was to test this algorithm. Testing of this algorithm was done by using option from Matlab to call the Java code in the Matlab. There are two classes in Java one is for parameters which is discussed before, similarly the second class consists of PKF algorithm.

The procedure for this testing is straight forward first of all we have to give the path of the Java class in Matlab. Here we have two classes so we have to give path of both classes. After giving the path of each class we can create object from that class to use the method of class which we need according to requirements.

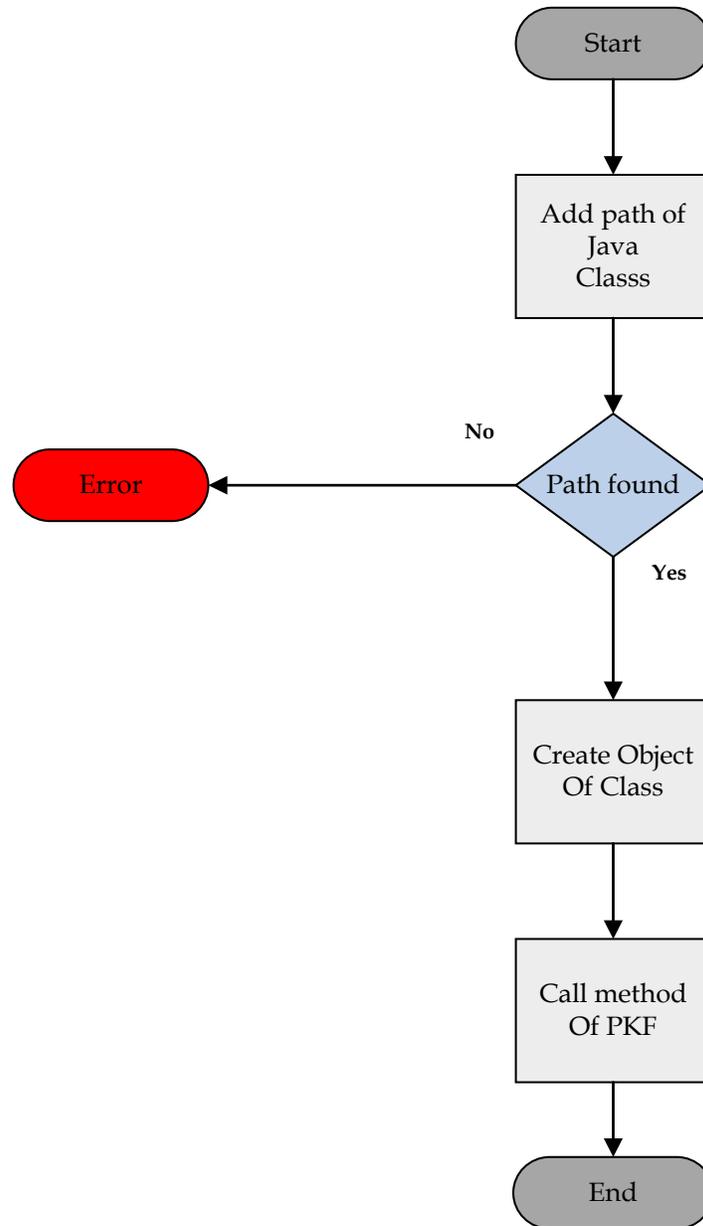


Figure 3-0-13 Method of testing Java code in Matlab

3.4.3.1 Compare the result

In order to compare the both Matlab and Java algorithm we have used the plot in Matlab. After implementing both Matlab and Java code we get the following result given in the figure below

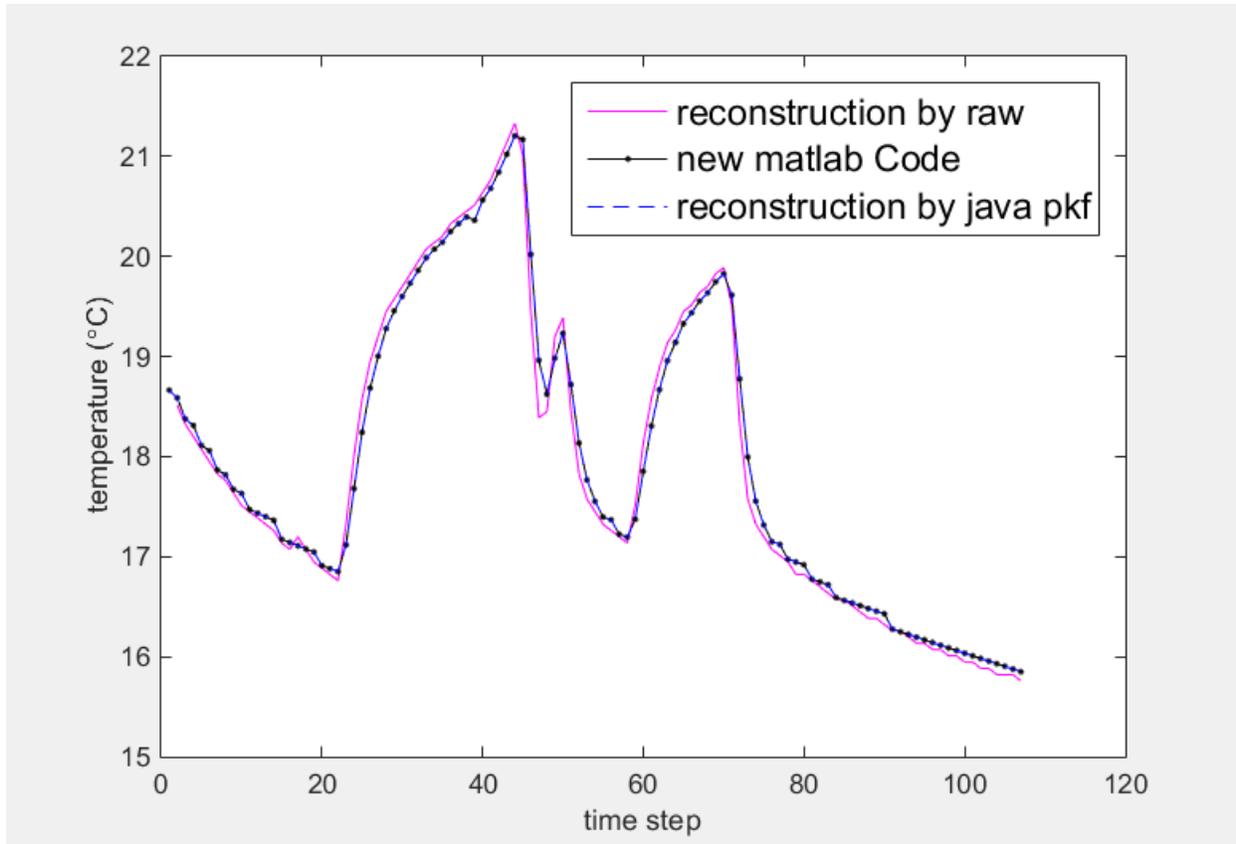


Figure 3-14 Matlab testing of PKF JAVA

From the figure, it can be seen that both Java and Matlab graph are overlapping. The third line with pink is the raw data value. This plot is the reconstruction value obtained after implementing the PKF algorithm both in Matlab and Java.

Chapter 4

Software Design

This chapter explains about software design for algorithm implementation and overall software structure

4.1 Structure of Software implementation

The software structure which is implemented can be comprised of two different paths. Initially the goal was to implement the software in such a manner that data can be transmitted from the base station, this data was a temperature data which was stored in a text file. For this section of software we had three nodes and one cluster head which were interconnected using wireless communication. The data was sent to cluster head which is used as a main point for data distribution. Cluster head then according to the node addresses send data to each node.

The data which was transmitted to each node was stored in the flash memory of each sensor node. In this section of code we have to organize the code in such a manner that the data which is stored in the flash memory of each node can be accessed or playback at any time from the node. The data is which is received from each node is received as a single value from each node with some delay this data is then gathered at cluster head and average is calculated which is then transferred to the base station.

The second section of software design is organized in such a way that in this section the data is transferred to a single node from cluster head. This section of code is designed in such a way that we have to implement the algorithm on the node and cluster head. The PKF algorithm which was converted from Matlab is implemented. The algorithm is divided in such a way that node section has implemented both KF (Kalaman filter) and predictor, while on the cluster head side there is only predictor implemented. The data which is received by predictor on cluster head was then transmitted to the base station. The overall structure of software design can be understood by the figure given below

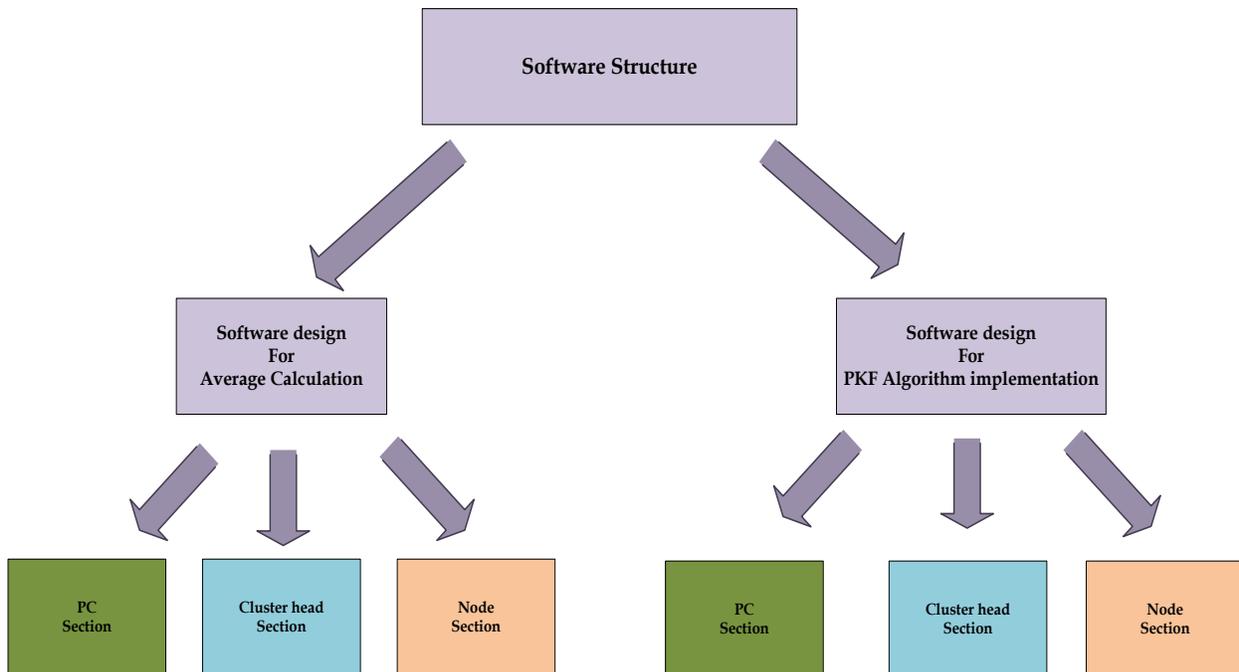


Figure 4-1 Overall structure of Software design

4.2 Communication protocols for Sun SPOT nodes

Sun SPOT kit is usually equipped with a radio transceiver based on CC2420. The frequency of this kit is 2.4GHz. There are two radio communication methods which are used to communicate either from base station to nodes or node to node. These methods are given below

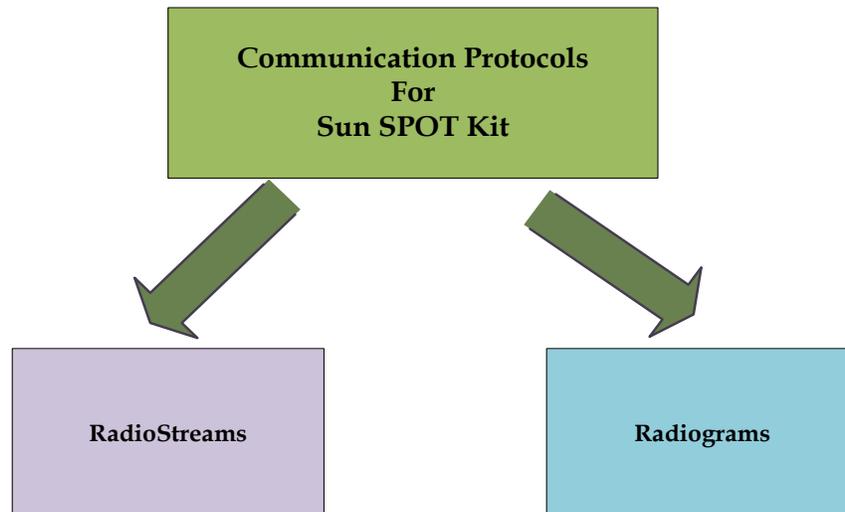


Figure 4-2 Communication Protocols

4.2.1 Radiostreams

The connection which is based on radio stream act as socket like protocol. This protocol is similar to peer to peer combination. This protocol has the property of reliability and also when there is a connection between two devices it performs like buffered stream based IO. Depending on the application or the network topology it can be single-hop or multi-hop.

It is straight forward to open a connection using radio stream first we have to give initial name to the connection

```
StreamConnection firstCon = (StreamConnection) Connector.open
("radiostream://0014.4F01.0000.75E7:xxx");
```

With the help of above method we can open a stream connection. In the above two lines of initialization there are few things which should be noticed. The first thing is the address 0014.4F01.0000.75E7 which is a 64 bit IEEE address of the radio. The second thing which is mentioned as xxx should be the port number. This port number can be selected from the range 0-255. The port number can be any number in between this range.

In order to open connection between two nodes the port number should be similar on both sides. After initialization of this connection the data can be sent or receive easily this can be done using following lines of code [24]

```
DataInputStream input = firstCon.openDataInputStream();
```

```
DataOutputStream output = firstCon.openDataOutputStream();
```

4.4.2 Radiograms

This type of connection named as Radiogram connection. In this connection data packets are interchanged between the two devices using datagram protocol.

Before establishing a connection between two devices it should be made sure that both devices have open connection for communication. The IEEE addresses must be accurate and the connection port must also be same for both the devices. Following steps are needed to open connection and send data

```
DatagramConnection data = (DatagramConnection) Connector.open("radiogram://" + IEEEaddress + ":110");
```

```
Datagram sd = data.newDatagram(data.getMaximumLength());
```

```
sd.writeUTF(" Send the data");
```

```
data.send(sd);
```

These are few steps through which data can be sent from one device to other device. On the other hand the receiver end has same initialization as before but there are some difference on the receive section

```
data.receive(sd);
```

```
String Receive = sd.readUTF();
```

In radiogram transmission there is one technique through which we can transmit data to all the listeners. This technique is called the broadcast technique.

```
DatagramConnection brd = (DatagramConnection) Connector.open("radiogram://broadcast:110");
```

```
Datagram sd = data.newDatagram(data.getMaximumLength());
```

```
sd.writeUTF("Broadcast a message");
```

```
brd.send(sd);
```

Although using broadcast mode we can send data to multiple listeners, but when the datagram is larger than 200bytes than usually it is not recommended to use broadcast mode. Broadcast mode has no acknowledgement method so has no guarantee of data delivery.

If we use unicast method both in radiograms and radiostreams then it has a mechanism. During the implementation of unicast with the help of radiograms there is built in mechanism named as ACK/retry that will ensure the data transmission on failure. In radiostream technique there is one additional property which ensures that fragmented data will be recollected in proper order.

To get reliable data in the broadcast technique fragmentation plays an important role. If break up the broadcast datagrams in two fragments than in this case the data transmission will be very reliable. While

802.15.4 radio packet has the limitation of the packet size. The data packet which is used normally has capacity of carrying 100bytes [24].

4.3 Flash memory in Sun SPOT

Flash memory is the type of memory which comes in the category of nonvolatile memory. The flash memory is fast as compared to the EEPROM. The thing which makes flash memory fast is its block level data writing. On the other hand EEPROM do it at byte level. Similarly if we want to replace the DRAM or SRAM with flash memory, it will be also not possible due to the speed mismatch [22].

Sun SPOT node has flash memory of 8Mbytes. Flash memory can be accessed within 70nsec while if we compare the page time this is less about 30nsec. Flash memory remains shut down if the Sun SPOT node is in the deep sleep mode and it is operated on 3V [24].

There are two methods to access the Sun SPOT flash memory. These two methods can be defined as in the figure given below.

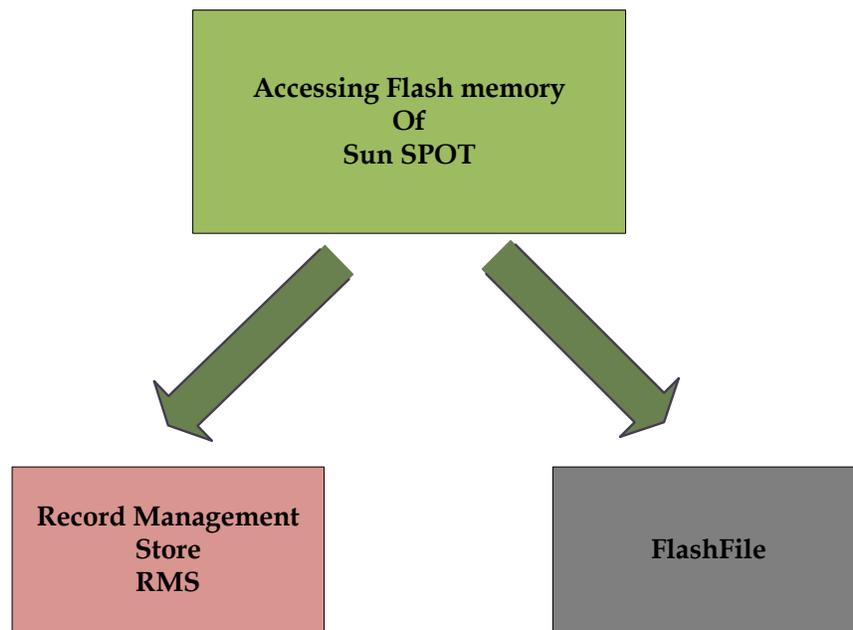


Figure 4-3 How to access flash memory

4.3.1 FlashFile

This method of flash memory is not usually recommended. FlashFile system mechanism works at the lower level. If we want to use the specific area of flash memory then we have to use the FlashFile method of flash memory access.

There is a main class which is used to access the flash memory using FlashFile system

com.sun.spot.espot.flashmangement.FlashFile

If we want to access specific areas or sectors of the flash memory then we can use two other classes which can allow us to read or write in a specific sector of the flash memory these classes are given as [25].

com.sun.squak.flashmangement.FlashFileOutputStream

com.sun.squak.flashmangement.FlashFileInputStream

4.3.2 Record Management Store (RMS)

Record Management Store is basically provided a way to store a persistent data. How data can be stored in a record store can be seen from the figure given below:

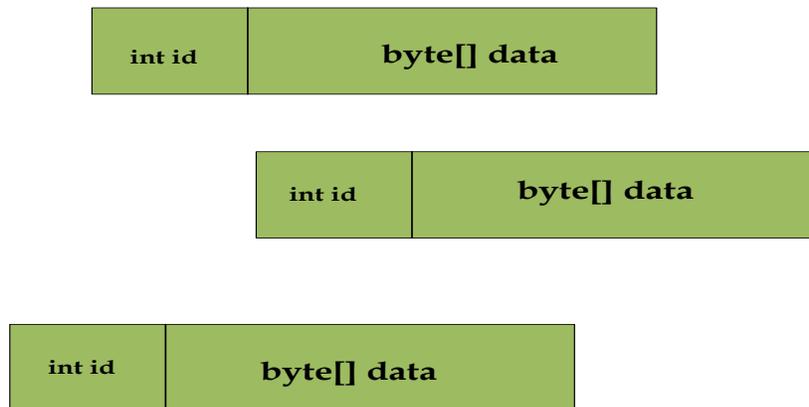


Figure 4-4 how data store in record store

The data record or simply called record consists of collection of bytes array. When a record is stored it stores at a particular location which is assigned by a specific number or called integer identification number (id). This is shown in the figure (4-4). There are few steps which are necessary to write data in a record store [26]

Step#1

First step is to open recordstore. It is necessary to give a specific name to that record store

```
RecordStore rms = RecordStore.openRecordStore("basicRecord",true);
```

Step#2

In this step we have to provide the data in the form of byte array.

```
byte[] basicdata= new byte[]{3,6,32,7,9};
```

Step#3

When we have a data in the form of byte array this data should be added to the record with a specific id

```
int recordID =rms.addRecord(basicdata,0,basicdata.length);
```

Step#4

Now if we want to get data stored in a record with a specific id then we can get as follow

```
byte[] receivedata = rms.getRecord(recordID);
```

Step#5

The final and the last step close the record store. If a record store is open multiple times than we have to close that record store same number of time it was opened [25].

```
rms.closeRecordStore();
```

There are few important methods which plays an important role to make the record store more accessible. When we want to remove a specific record which is placed at a specific record Id we can use following method

```
deleteRecord(ID);
```

It is used to delete a record the ID must be a specific id attached with a particular record in the recordstore.

It is also important to keep track of the record store. We can find the record id of the upcoming data with the help of following method

```
getNextRecordID();
```

Total records which are stored in a recordstore can be counted and the number can be displayed by using following built in method

```
getNumRecords();
```

4.3.3 Flash memory Implementation

In this section we will explain how the flash memory of Sun SPOT in our case was used. In our scenario flash memory was used only on leaf nodes. In cluster head section we don't have to save the data in flash memory. As previously described there are two methods to utilize the flash memory of Sun SPOT. In our case there was no need to access the lower level memory management so the first case of using FlashFile system was not suitable here. So we have decided to implement the Record management store method to access the flash memory. This method is more efficient and recommended for this scenario.

4.3.3.1 Important methods

We have defined some methods to store data on the flash memory of Sun SPOT. And method to get data from the record store, for utilizing this data in calculation and for further transmission to the cluster head.

4.3.3.1.1 openRecStore

This is the method which is called to open a record store with a specific name.

4.3.3.1.2 closeRecStore

In order to close a record store which is already open this method is used.

4.3.3.1.3 writeRecord

This method is used to write data in the RecordStore. In this method array is pass to write data.

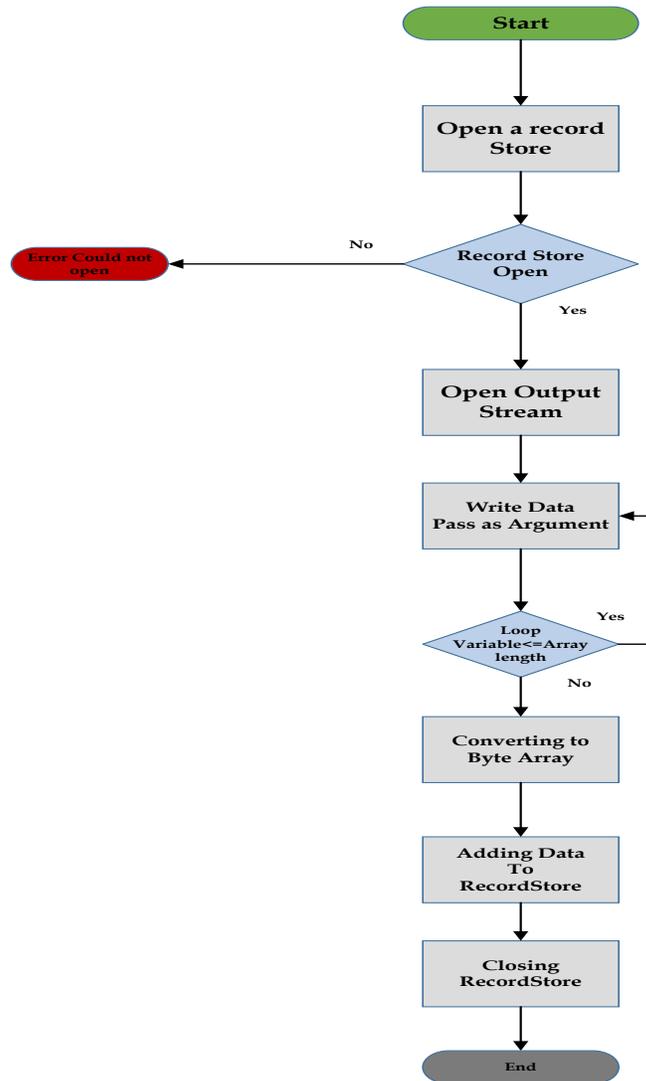


Figure 4-5 Flow chart for method to write data in Record Store

4.3.3.1.4 readRecords

This method is used to read data which is stored on the specific record Id. First it receives the data from the RecordStore and then stores this data into an array. This array is later on pass as a return value of this method.

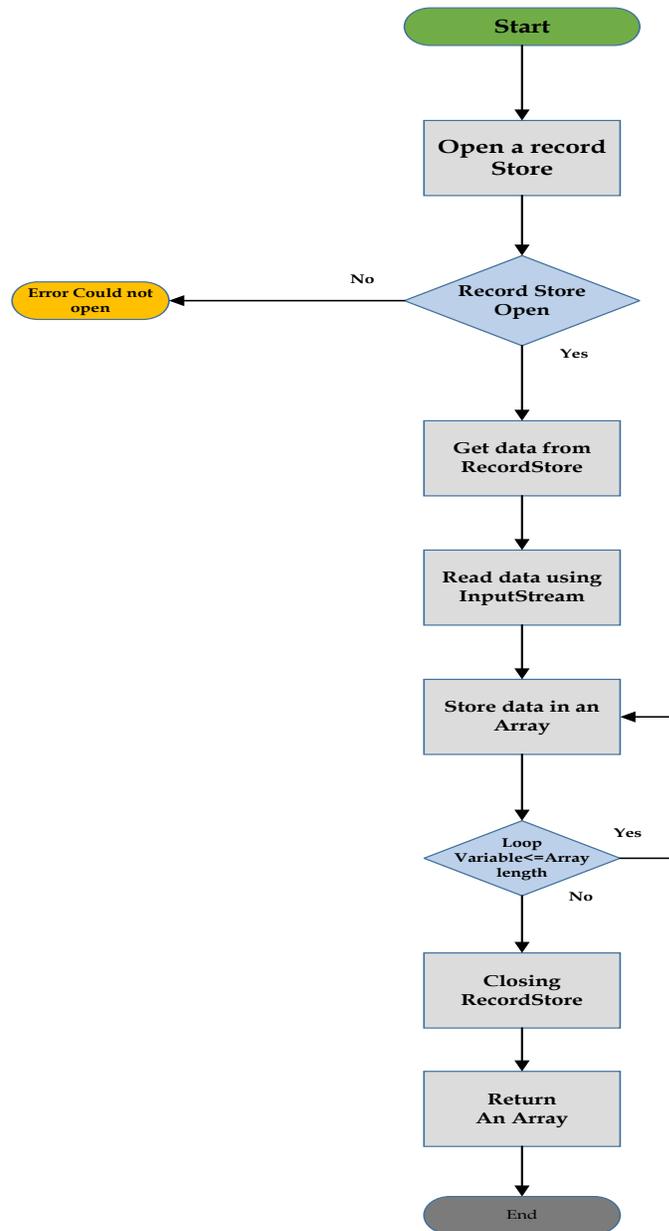


Figure 4-6 Flow chart shows a method working use to read data from RecordStore

4.3.3.1.5 deleteRecStore

This method is used to delete the record store. This method only deletes one record to remove all records there is another method.

4.3.3.1.6 deleteRecordStore

This method is used to remove all the records which are saved with different record ids. This method iterates through a loop and removes all the records.

4.4 Software design for Average Calculation

This software section was used for initial test and single value demonstration based on three sensor nodes and one cluster head. In this section we have write code in such a manner to receive an average value of data coming from three sensor nodes. The average calculation is done on the cluster head side. After this calculation the data is transmitted to the base station. In this code structure we have three sections of code which are as follow

- PC Section
- Cluster head
- Node Section

4.4.1 PC Section

This section of code runs on the PC. This application does not run on any node. In Sun SPOT the application which runs through the base station is different from the application which is actually runs on Sun SPOT node. This application has a main function which can run on computer. In Sun SPOT the base station is used to communicate with the other nodes.

The idea on this section of code was to read data from a text file. This data is temperature data which was stored in the format of different columns according to the node section. It depends on us to send which column of data to which node. In this section of code we have two classes which are shown in the figure given below.

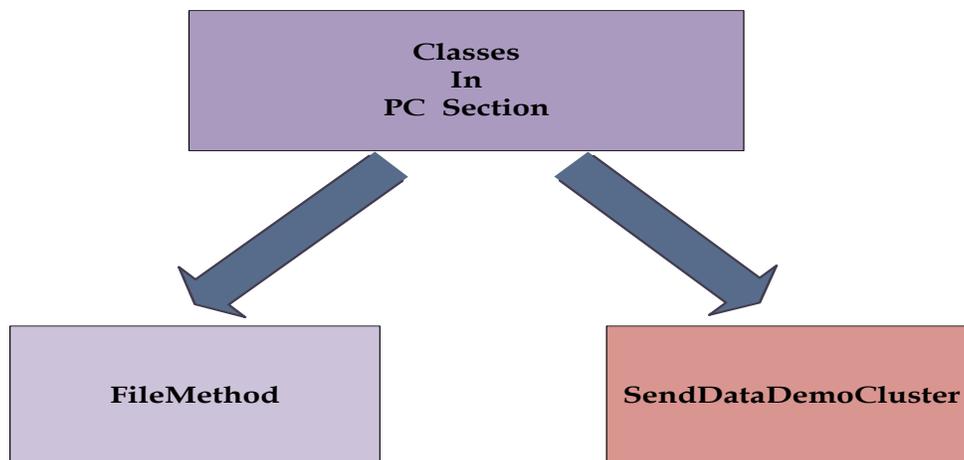


Figure 4-7 Classes in PC section of code

4.4.1.1 FileMethod

This class is one of the starting classes of base station section. This class is used to read data from text file

and classified that data according to requirements. There is only one method in this class named fileRead

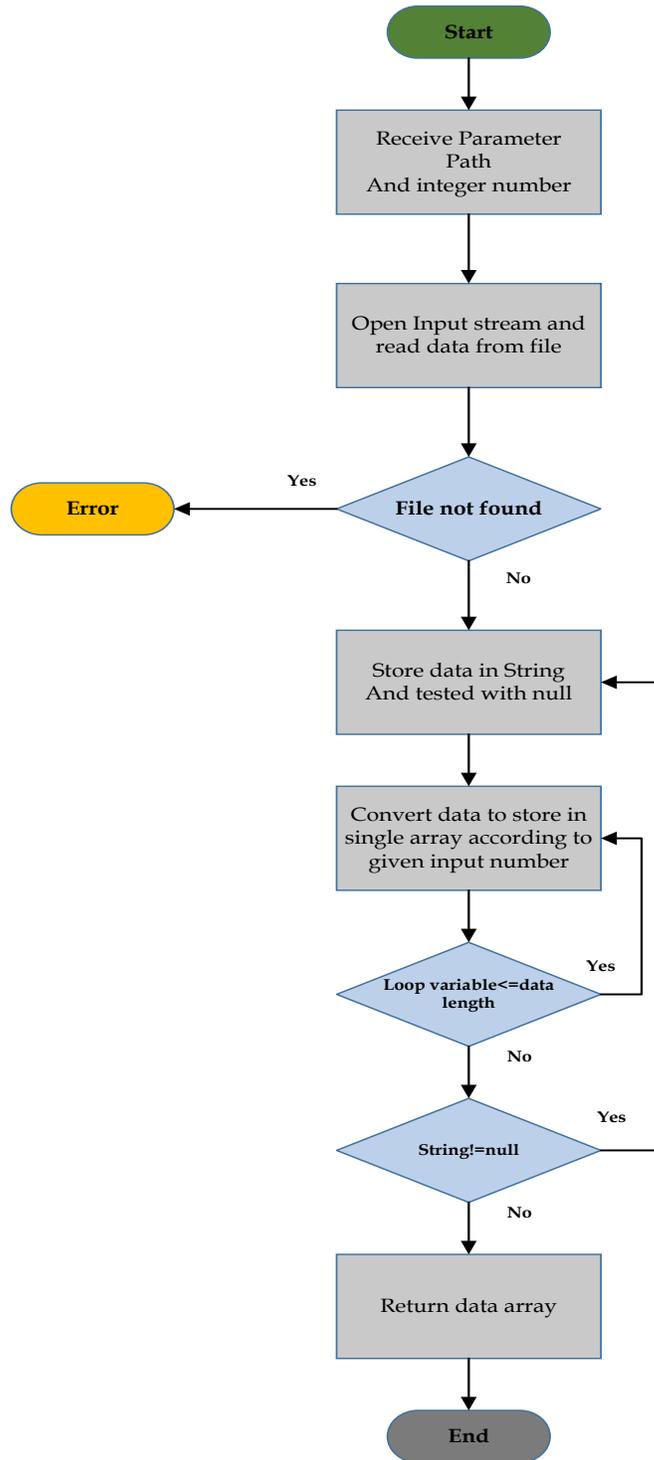


Figure 4-8 Flow chart of class FileMethod method named fileRead

4.4.1.1.1 ClassDiagram

The figure given below is the class diagram of the class FileMethod. We can see that there is a method having two parameters.

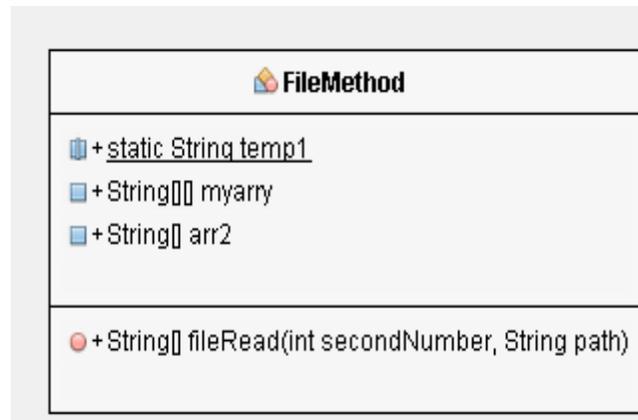


Figure 4-9 Class diagram of class FileMethod

4.4.1.2 SendDataDemoCluster

This is the main class for base station section of code. This class includes all the methods which are necessary to communicate and to transmit data from base station to cluster head. The methods which are used to receive average value after calculation from cluster head also included in this class. There are following methods

4.4.1.2.1 select_BasetoCluster

This method is used for selection of data. It means that it is used to select either we want to write data or get data. This selection is done by asking the user to type a character to select write we have used 'S' as a character. If we want to read data then we have to type 'R'. If we select a read then the average value which is calculated by the cluster head will be printed on the console.

In case of send data the data is firstly sent to the cluster head then it is transmitted to each node according to the address of the node. There is only one parameter passed to this method which is a character. The connection established at port 104 for this method.

4.4.1.2.2 connect_Recieve

This method is used to establish a connection between cluster head and the base station. This method is used when the data from the cluster head is received. With the help of this method the made connection is used to receive the average value which is calculated on the cluster head side. The connection is established at port number 90.

4.4.1.2.3 connect_BasetoCluster

This method is used to transfer data from the base station to cluster head. This data is got from the text file. When this method is used the data is send according to the nodes addresses. The data which is read by text file is sorted and then it is sent to each node. In this case it is also very important to know that data is sent collectively to a node. This means that once whole data of one node is transmitted then the data of second node is sent. This data is further stored in the flash memory of each node. The connection for this method is open at port 100.

The flow chart of this class is given below.

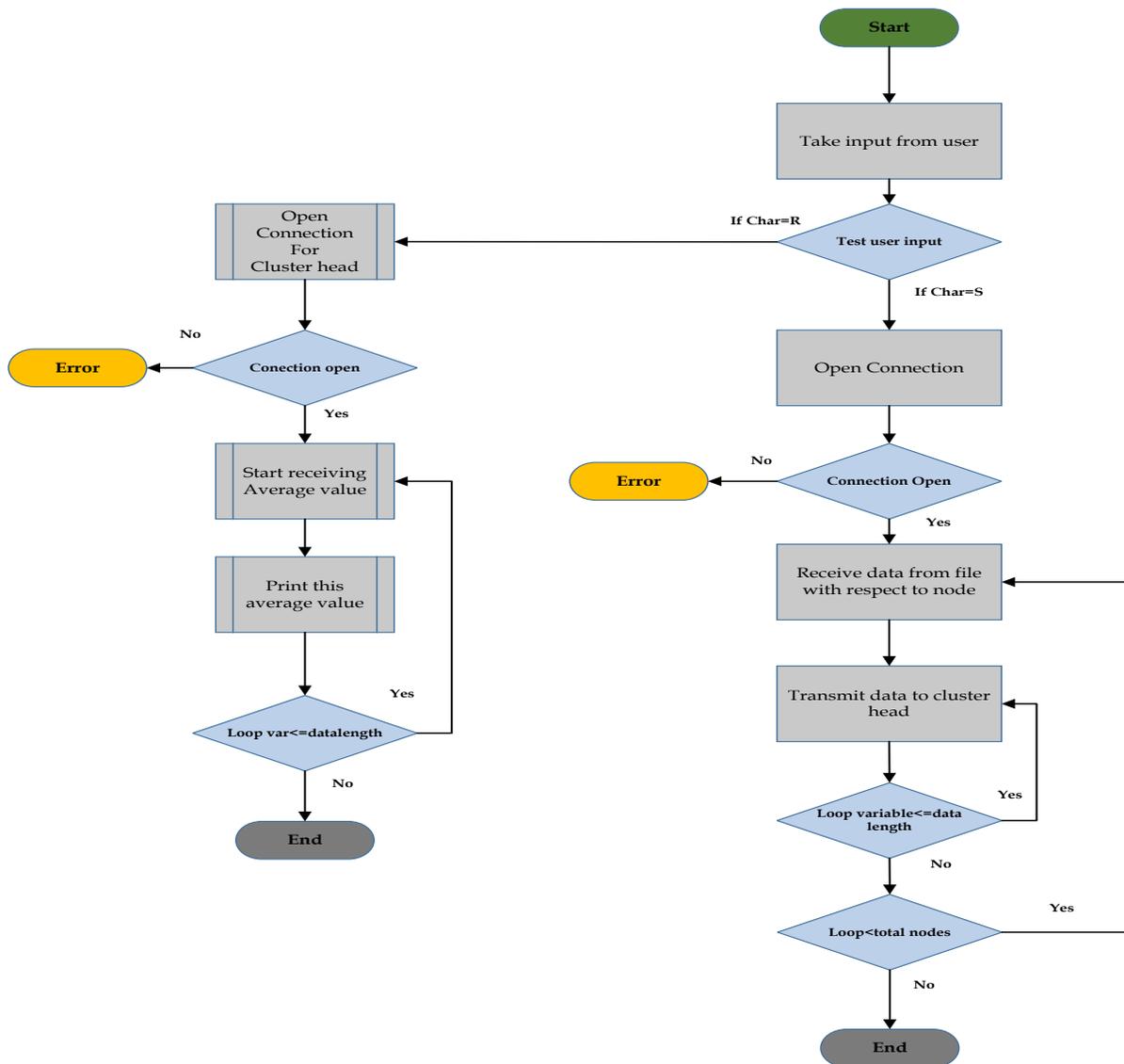


Figure 4-10 Flow chart of class SendDataDemoCluster

4.4.1.2.4 Class Diagram

This class diagram shows different method which are used in this class. It can also be seen that there is a main method. This class will run on the desktop and base station will be connected with the system.

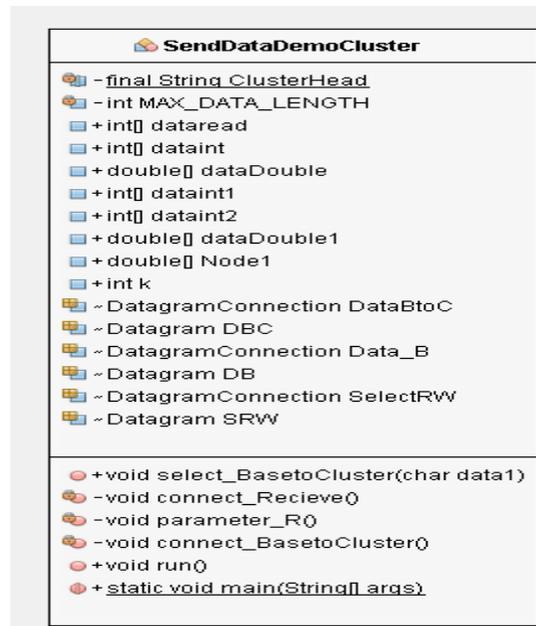


Figure 4-11 Class diagram

4.4.2 Cluster head

This section explains how the code on cluster head section was implemented. Cluster head provides a mean of base station to connect with the node and also all the calculation is done on the cluster head. There are two classes which are used in this section the name of these classes are given below

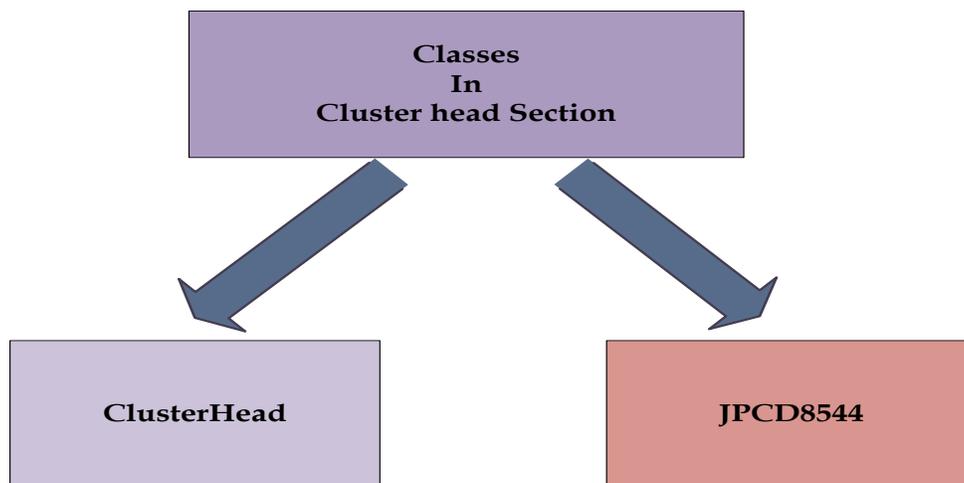


Figure 4-12 Classes in Cluster head

4.4.2.1 ClusterHead

This class is the main class of the cluster head section. This class declares and uses all the methods which are necessary to communicate both with base station and nodes. Now we will describe methods for this class

4.4.2.1.1 radioConnect_base2Cluster

This method initializes a connection between cluster head and base station. This connection is used when the data from base station to cluster head is ready to send. This data is collectively send for one node and after receiving data cluster head will forward this data to the respective node. The connection between cluster head and the base station is opened at port 100.

4.4.2.1.2 connect_Cluster2base

When data is received from all nodes and average is calculated then we have to send this average value to base station for displaying on the console. This data is transmitted one by one to the base station. This method is used to open a connection for sending this data to the base station from cluster head. The connection is opened for this communication at port number 90.

4.4.2.1.3 select_Base2Cluster_OpenConnection

Before using communication with base station cluster head must know which type of communication is going to start. It means either the communication is for writing a data on the nodes for storing in flash memory or it is for receiving data from each node to calculate the average. This method is used to open communication connection between base station and cluster head for this purpose. The connection is opened at port 104.

4.4.2.1.4 select_Base2Cluster_Receive

This method will receive a character from the base station. Depending on this character cluster head will start writing or reading data. This method will return a character which is further use in the program to classified sending data or receiving values from nodes.

4.4.2.1.5 data_BasetoCluster

When base station sends data to cluster head this method is used to collect data and after collecting and saving data this method returns an array. The size of array depends on the maximum data length which is set according to the base station data.

4.4.2.1.6 data_Cluster2Base

Cluster head calculate average of three values which are received from different nodes. After calculating this average it sends this average value to base station. This method is used to send that value to base station. This method has only one parameter which is double value when we pass average value as argument to this method it sends it to the base station.

4.4.2.1.7 cluster2NodeSendCommandC

There are total three nodes in this network with one cluster head. This method is used to open connection between three nodes and cluster head for writing or reading data. Normally connection is opened by giving one address for the target node. But in this method we have declare address as an array. So all the addresses of nodes are saved in an array. This method can be called in a loop so that data can be transmitted to each node according to the address which is saved in the array. This method has one parameter which is named as targetNode. This parameter selects the node where data should be transmitted. The connection is opened at port number 106.

4.4.2.1.8 radioConnect_Cluster2Node

This method opens connection for specific node to send data and also sends data to the specific node. There are two parameters for this method one parameter selects the node and second parameter is an array which is going to send on a specific node. The flow chart explains about the working of this method.

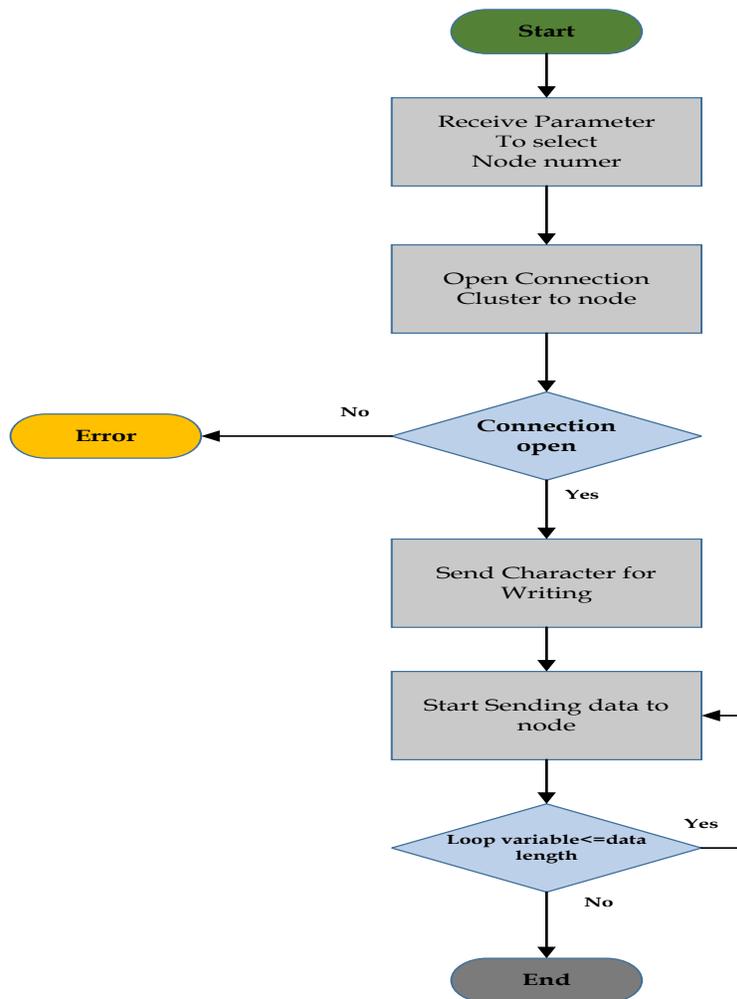


Figure 4-13 Flow chart describing method operation

The figure shows at first connection is opened for the communication with a specific node while in the second step it will send a character for writing purpose.

It is also important that the connection between the specific node and cluster head must be properly established otherwise it will give an error. After that it will start sending the data which it is received as a argument in an array format. This method will run until the data in the array will finish.

4.4.2.1.9 cluster2NodeSendCommand

This method sends a character to the nodes in the network from cluster head. This character gives command to nodes for start sending or receiving data. This method has two parameters one parameter is an integer which selects the node address, while second parameter is character which selects send or receive option.

4.4.2.1.10 radioConnect_Nodes2Cluster

When cluster head receive a command from the base station that now start sending average data to the base station then it is necessary to build connection between cluster head other nodes for receiving that data. This method is used to make this connection for receiving data from each node. In this method there is only one integer parameter. This parameter selects the node address from the array of node addresses.

4.4.2.1.11 nodes2ClusterQ

When data is sent to the nodes it is sent collectively. It means the data whole data array is sent to a node and then to other node, but when average value is calculated it is calculated by receiving single value from each node and then calculating average. To make sure this one value from each node a method for query is defined.

The purpose of this method is to send a query to each node that now sends value from flash memory to the cluster head. This method has only one parameter which selects the node number for sending query.

4.4.2.1.12 nodes2Cluster

This method receives data from each node. This method has two parameters these both parameters are integer values. One parameter is to select the node number from where the data comes. While the second number is to keep track of the how much data values have been received.

4.4.2.1.13 Main application

Above are all methods which can be utilized to achieve the required goal of cluster head section. In the main application we have to organize these methods in such a way that we can send or receive data according to requirements. In main application we have initially open connection between cluster head and base station. This connection will use to receive the character value which is used to select the reading or writing.

After receiving character the main application is divided into two sections which we have arranged using a switch case. The writing data section of the application is simple because here we do not need a query

section to receive single value from each node. This section receives whole data which is stored in an array. After receiving whole data it transmits this data to single node. This process remains continue until the total number of nodes is completed.

On the other hand the receive section is more complicated than the send section. In this section first connection is opened between cluster head and the nodes. Then data is received one by one from each node. To keep this continues a query is sent for each value to all nodes. When three values from each node are collected then average is calculated which is sent to the base station.

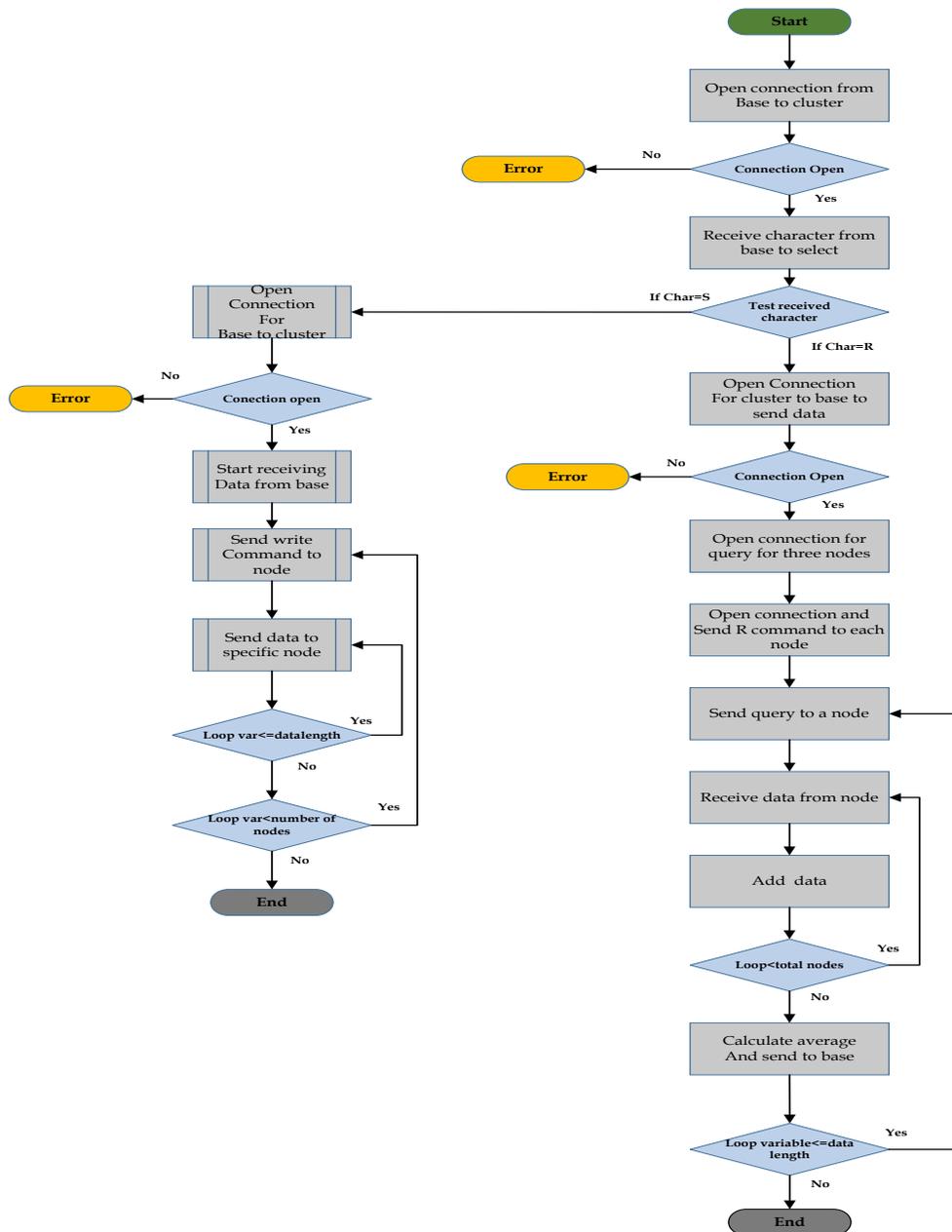


Figure 4-14 Flow chart for main application section

4.4.2.1.14 Class Diagram

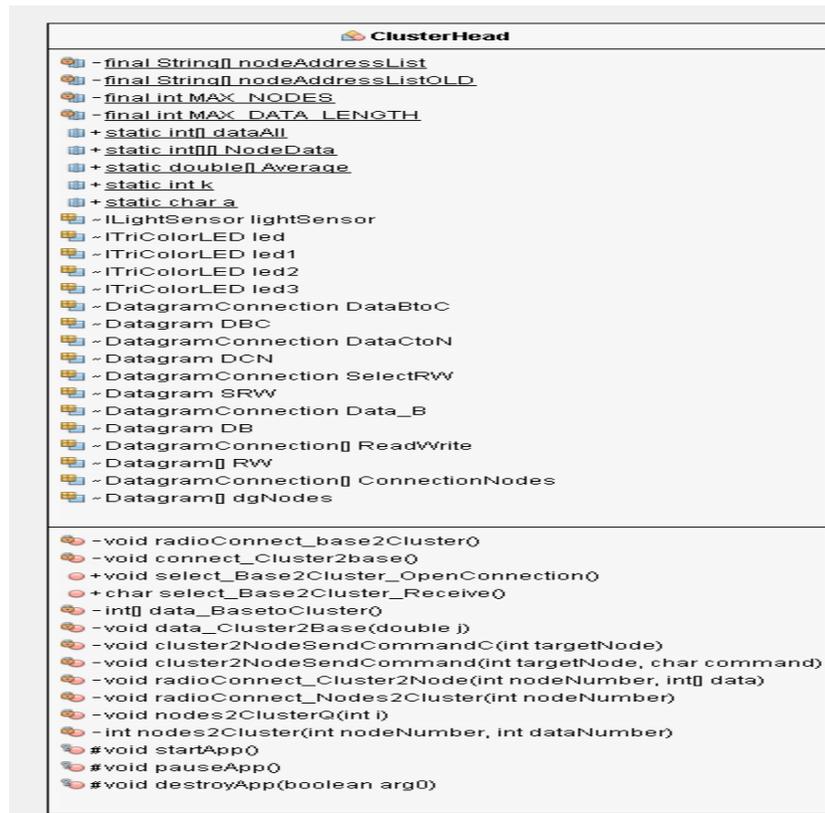


Figure 4-15 Class diagram of ClusterHead

4.4.2.2 JDCD8544

This class is actually the open source modified class for interfacing Nokia 5110 LCD. This class has been used to interface LCD with cluster head. This class is already described in hardware chapter.

4.4.3 Node section

This section of code consists of only one class. Sensor node section code is same for all three nodes. So this code is implemented on all nodes.

4.4.3.1 SensorNode

This section of code consists of method for saving data in the flash memory. All these methods are described in flash memory section in the start of this chapter. In this section we will describe few methods which are used to connect with cluster head to receive data from the cluster head and also to send data to cluster head.

4.4.3.1.1 newDataConnection_Nodes

This method is used to established connection between the cluster head and the nodes. The connection datagram are open in an array. This is used for all the three nodes to open connection to communicate with the cluster head. The connection is opened at the port number 105.

4.4.3.1.2 nodesSendData

This method is used to send data saved in the flash memory to cluster head. This method has two parameters one is data which is going to send, while the second parameter is the integer to select the node at which data has to be sent.

The flow chart of main application of this class is given below

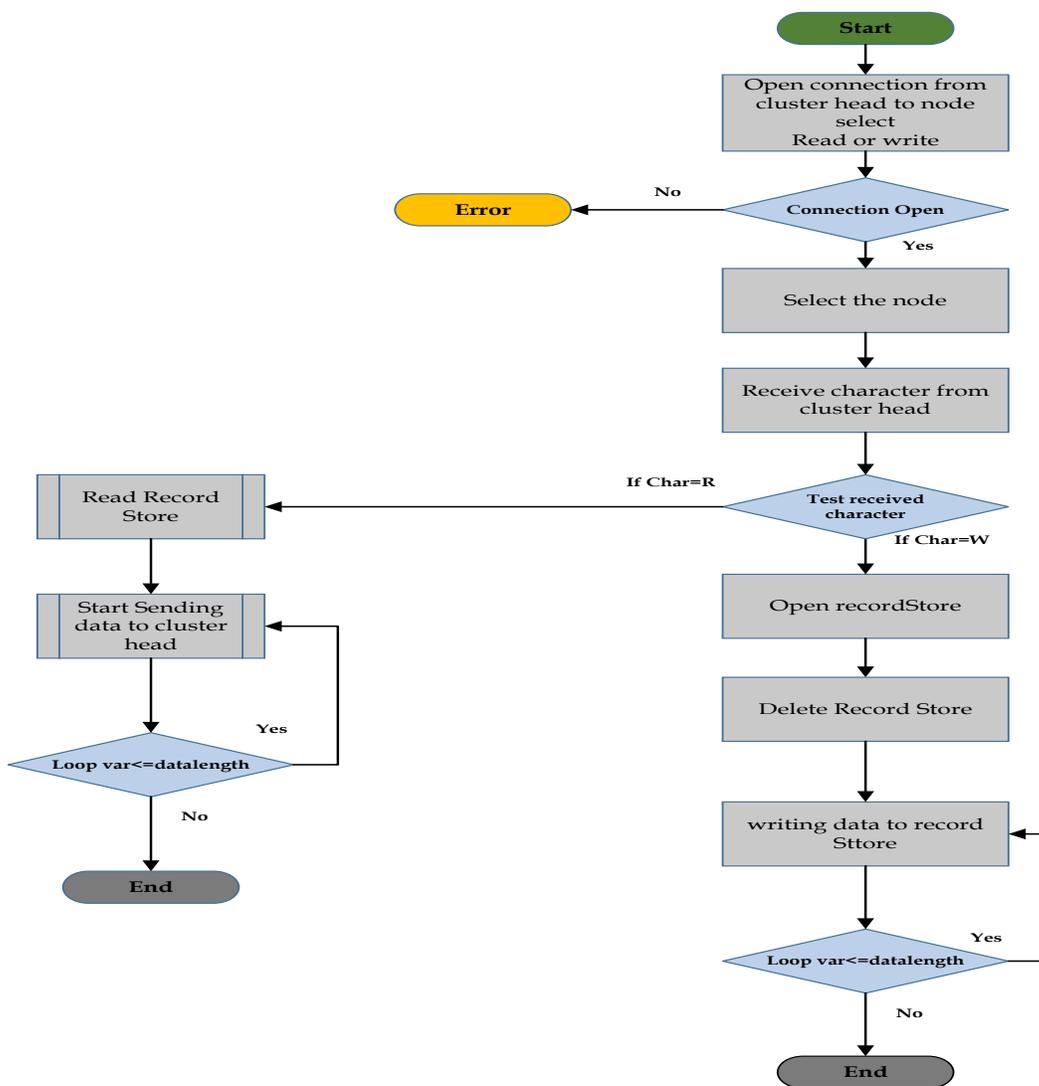


Figure 4-16 Flow chart of SensoNode class main application

4.4.3.1.2 Class diagram

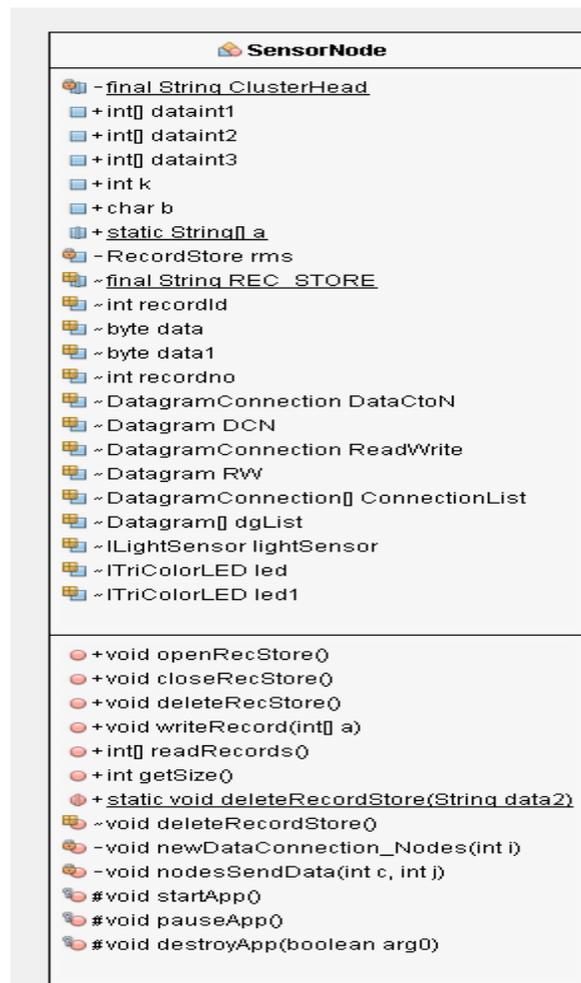


Figure 4-17 Class diagram of class SensorNode

4.5 Software design for PKF Algorithm implementation

This section of software design will explain how we have implement PKF algorithm which was converted from Matlab to Java. There are following parts which will be explained in this section

- PC Section
- Leaf node
- Cluster head

4.5.1 PC Section

Although the implementation of PKF algorithm was on a single node and cluster but PC section of code was also modified to create a GUI interface. The GUI was created using Swing components. The front panel of the GUI will be explained in the GUI section but here we will explain the methods which are created for this GUI and how these methods were implemented to create overall structure. In this section there are few methods which are used to communicate with the cluster head for sending the data from the

text file while the other methods are used to receive data after the calculation. There are following classes include in this section

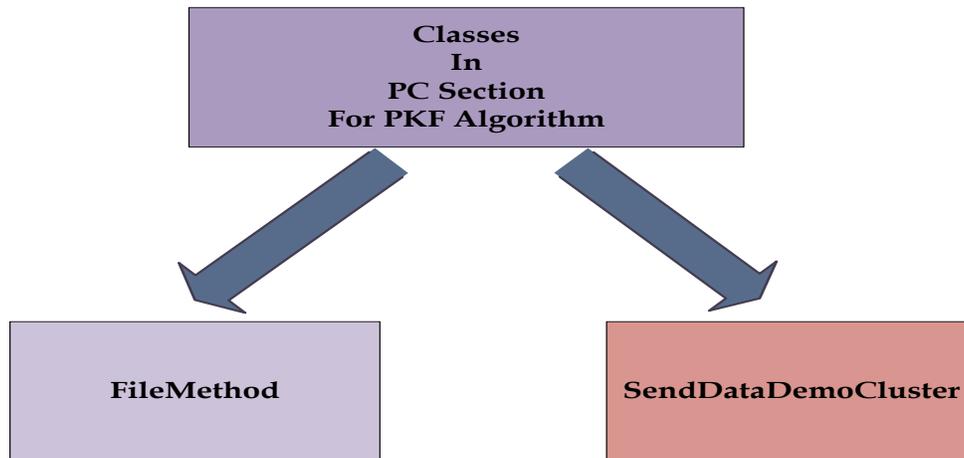


Figure 4-0-18 Classes in PC section

4.5.1.1 FileMethod

This class has the same functionality which was explained in the previous section. It reads data from cluster head from text file convert that data into one dimensional array.

4.5.1.2 SendDataDemoCluster

Although this class has also the same name as described in the previous section but this class functionality is totally different with respect to the previous class. In this section we will explain the methods which are used here to send data and receive data and also how the GUI works.

4.5.1.2.1 *connection_BasetoCluster*

This method is the starting point of this class. It enables communication between base station and cluster head. When connection is opened we can send character for writing or reading purpose selection. The connection through this method is opened at port number 104.

4.5.1.2.2 *select_BasetoCluster*

This method is used to send character to the cluster head for selection purpose. There is only parameter for this method which is of character type.

4.5.1.2.3 *connect_Recieve*

This method is used to open a connection for receiving data from the cluster head. The data which is received through this connection is received after the implementation of PKF algorithm. The connection opened at port number 90.

4.5.1.2.4 connect

This method is used to connect base station to cluster head but this method can be used within a loop to avoid the error for opening a connection multiple times. There is a Boolean condition inside this loop which avoids calling this multiple times.

4.5.1.2.5 rMS

This method is used to calculate root mean square of an array. There is one parameter which is an array of type double and a return value as double.

4.5.1.2.6 GUI Section Implementation

In this section we will explain how the GUI was implemented using different buttons and text areas. Actually this section will be more programming oriented. Below there are different front panel buttons and other things and how they are declared with different names.

- **Connect**
This button is used to enable a connection from base station to cluster head. This button is defined by following name Connection. This button has also attached a label which is used to display connection status. In this button it is also make sure that until the connection button is not click all other buttons remain disable.
- **Run**
After the connection is established now all the other buttons are enabled. Run button is used to run all the available options on this GUI. In order to run a selected option we have to select that option and then click the Run. There is also one function which is associated with this button that if we click this button without selecting any option then it will give an error. The variable for this button is denoted by jButton1.
- **Send Data**
This is a radio button which is displayed on the front panel. It is used to send data to cluster head which is transmitted to the leaf node. It is represented by the variable name rSendData.
- **Receive Data**
This radio button shows on the front panel is used to receive reconstruction value which is received from the cluster head after implementation of PKF algorithm. This is represented by the following variable name rReceiveData.
- **Graph**
This radio button is used to draw a graph. There is an open source library which is named as XChart is used to draw the graph. This button will generate a graph to compare the reconstruction data with our raw values data set. It is represented with a variable name Dgraph.
- **RMSE**
As the name of this show it is used to calculate the root mean square error. It is used to calculate the RMSE for the raw values of data set and the reconstruction values. It is represented by the variable name rdifference.

- Send Data progress Bar**
 This progress bar is used to show the progress of data which is sent from the base station to cluster head for sending it to leaf node. This progress bar also represented data progress in percentage. This progress bar is represented by the variable name jProgressBar2.
- Receive Data progress Bar**
 This progress bar shows the progress of data received from the cluster head. This data is based on the reconstruction values. It is represented by the variable name jProgressBar2.
- Exit**
 This is used to exit from the GUI. It is a button with variable name jButton2.

4.5.1.2.7 Flow chart for Send Data

This flow chart explains how the send data radio button works. Initially it sends character to cluster head for selection of sending data to the cluster head. Then it open connection for sending data to the cluster head. If connection will not open it will give error. Then it call method to read text file for data if the path of file not found it will also give error then it start sending data for cluster head and also to text area and progress bar. After all the data will be sent it will show all data on the text area.

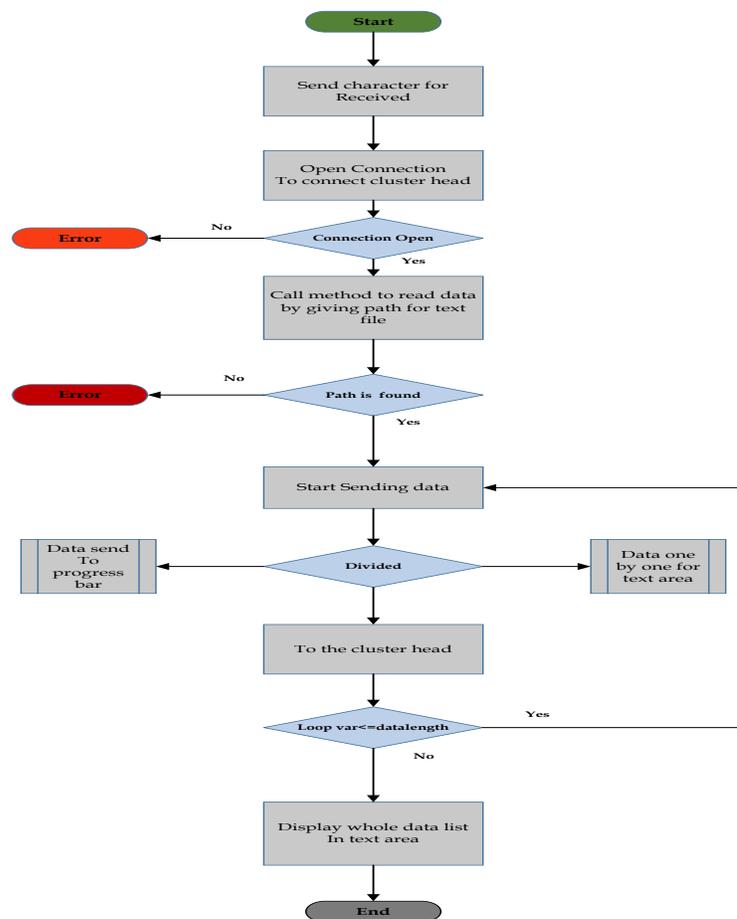


Figure 4-19 Flow chart explain functionality of Send data radio button

4.5.1.2.8 Flow chart for Receive data

This flow chart explains how reconstruction value data is received from the cluster head after implementation of PKF on leaf node. First command for receive data is sent then a connection is opened for receiving data from cluster head. If connection is not successfully opened it will give an error. If connection is opened then data is start coming from cluster head which is used both for display in text area and after for progress bar. After that there is a delay of 200ms in each value. When the whole data is received it is sent to display on text area.

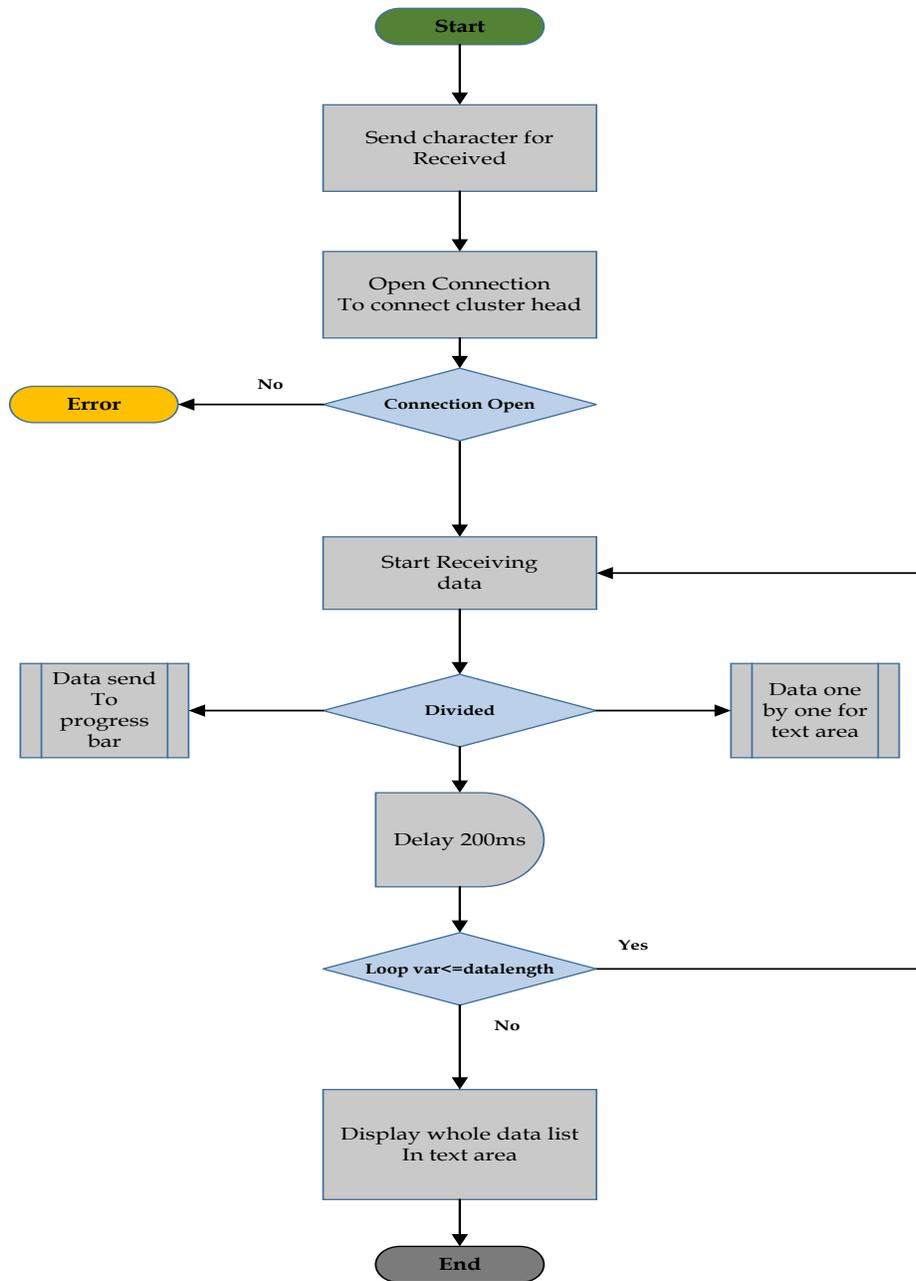


Figure 4-20 Flow chart showing how to receive data from cluster head

4.5.1.2.9 Flow chart for Graph

This flow chart explains the functionality behind the radio button. In plotting this graph we have utilize the free source library named as XChart. First step is to set a graph by using method and giving proper arguments. Then graph font and text is adjusted by built in methods. After that there are two sections because we have to create two data set graphs. First step is to create array list of whole data set. Then making a series using x axis and y axis data. After that displaying both data set on same graph.

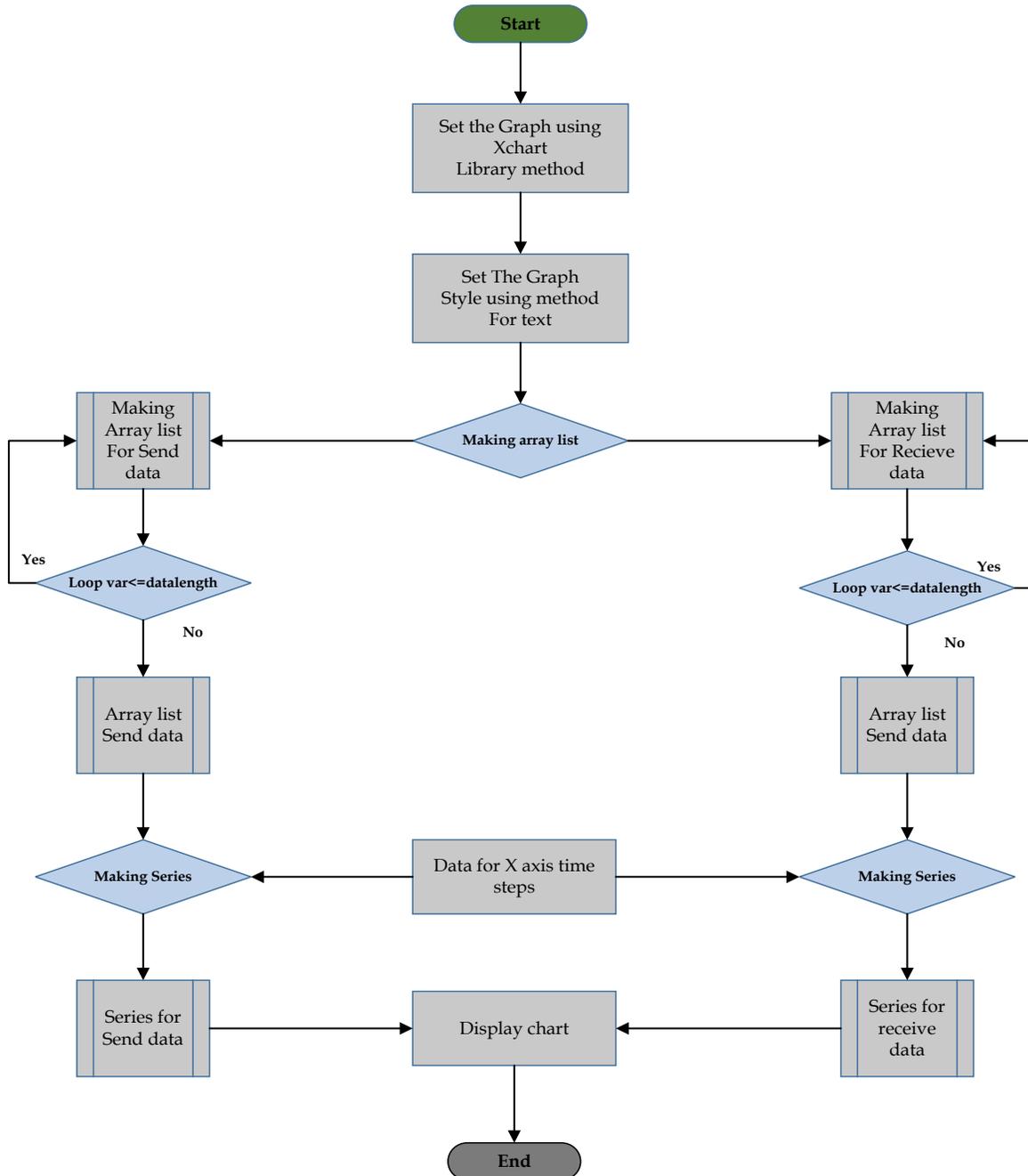


Figure 4-21 Flow chart for creating graph

4.5.1.2.10 Flow chart for Run button

This chart explains about the flow chart describing how run button works to implement any radio button. It is shown that if we select run button and not select any radio button then it will give error. Otherwise we have four options for selecting the radio button depending on the application we need.

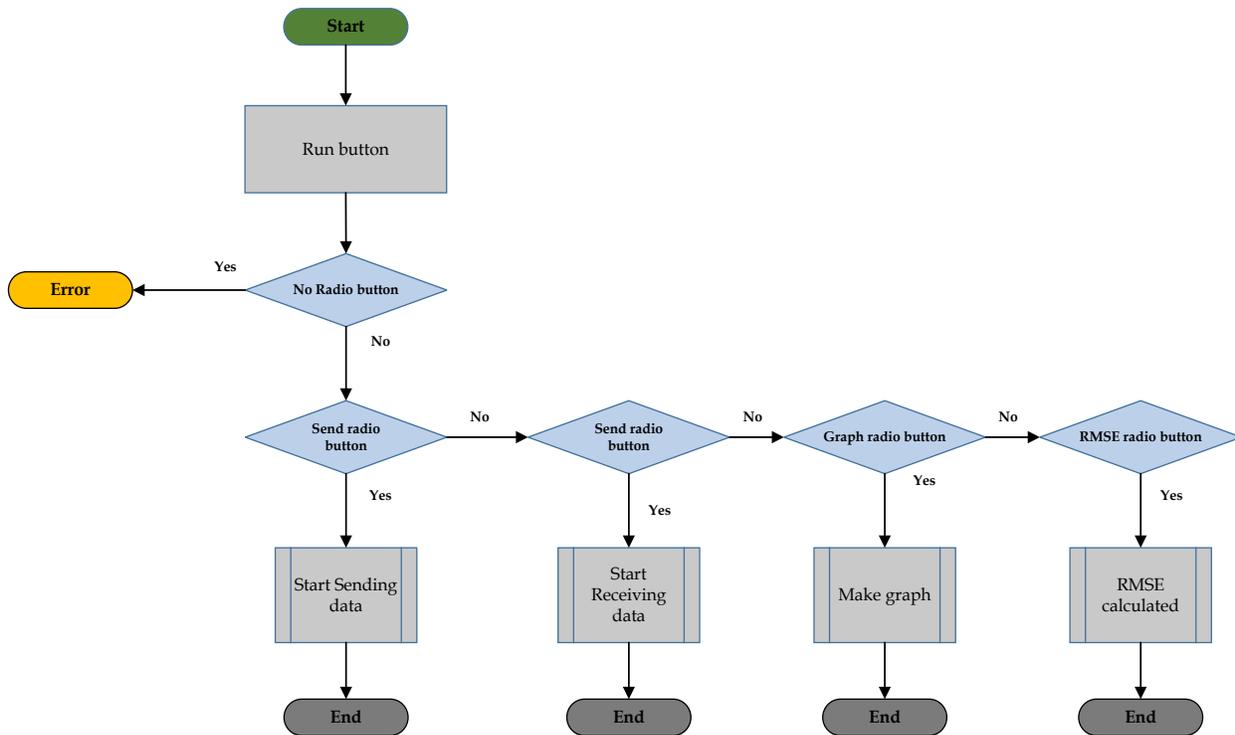


Figure 4-22 Flow chart how run button is attached with these radio buttons

4.5.2 Leaf node

This section consists of code which is run on a leaf node. There are following classes which are used to implement this section of code in the next section we will explain these classes one by one

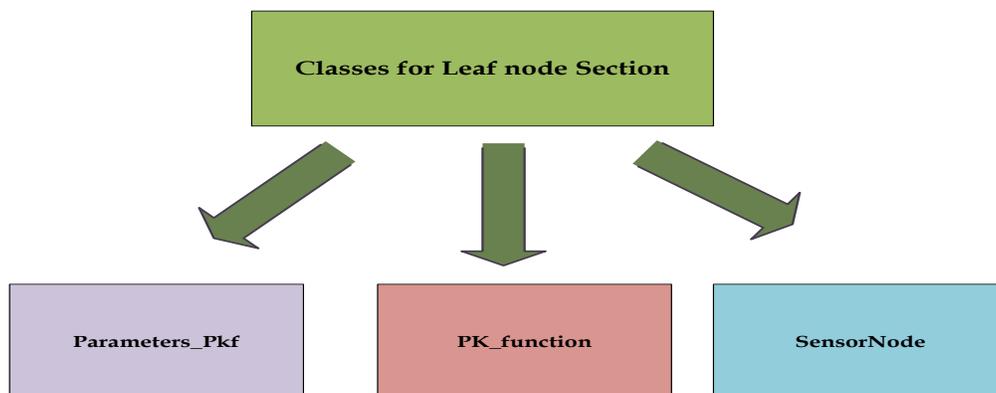


Figure 4-23 Classes in the leaf node section

4.5.2.1 Parameters_Pkf

This class is used to store parameters in the form of methods. These parameters are calculated using Matlab. These parameters are further used in the PKF algorithm implementation. This class is discussed in third chapter with class diagram.

4.5.2.2 PK_function

This class is the main class which implements PKF algorithm. There are different methods include in this class which assist to implement PKF. The implementation is not straight forward because in this implementation matrix calculation is involved. First starting value of matrices is assigned. After that in first step KF is calculated using equations. After that linear prediction is implemented and error calculated. After that error is compared if the value of error is less than threshold it will do nothing.

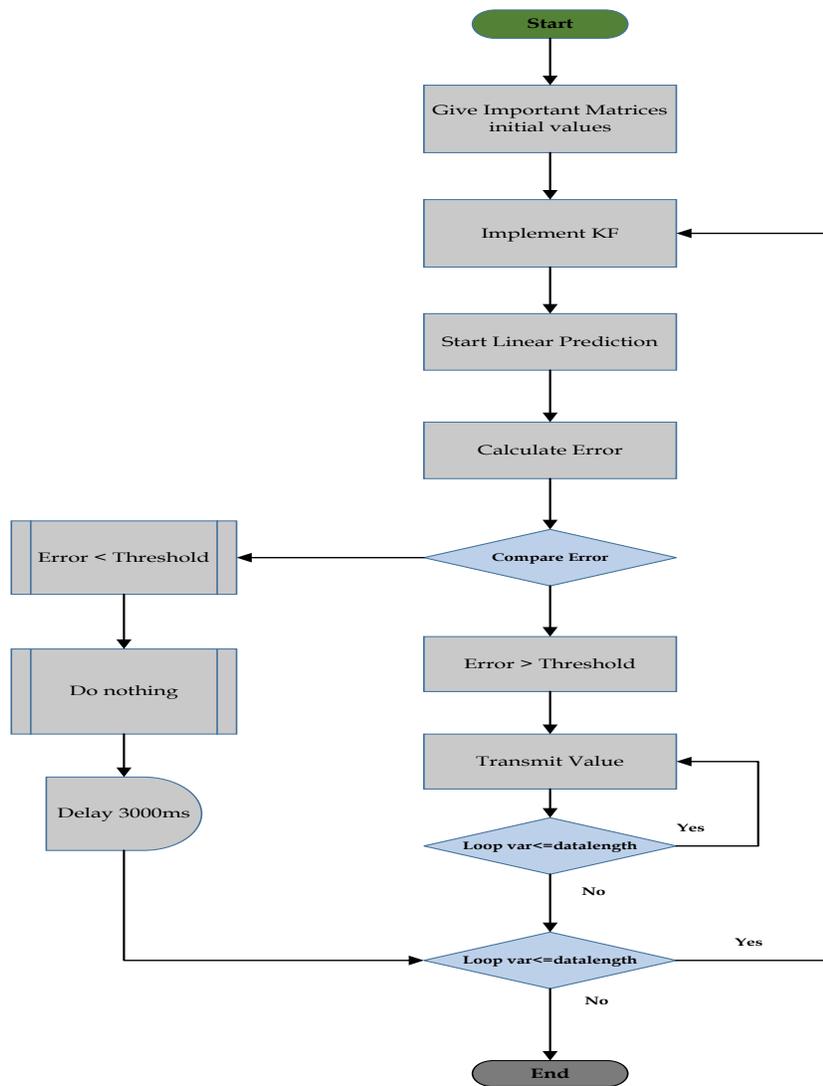


Figure 4-24 Implementation of PKF Algorithm

There is a delay in this section so that we can distinguish on the cluster head side that there is no transmission at the moment and now it can send predicted value to the base station. On the other hand if the error is greater than threshold then it should transmit the respective value to the cluster head. This will continue until the loop will finish.

4.5.2.3 SensorNode

This class is the main class in this section which is used to make communication with cluster head and also to call the PKF algorithm function. The flow chart for main function is given below

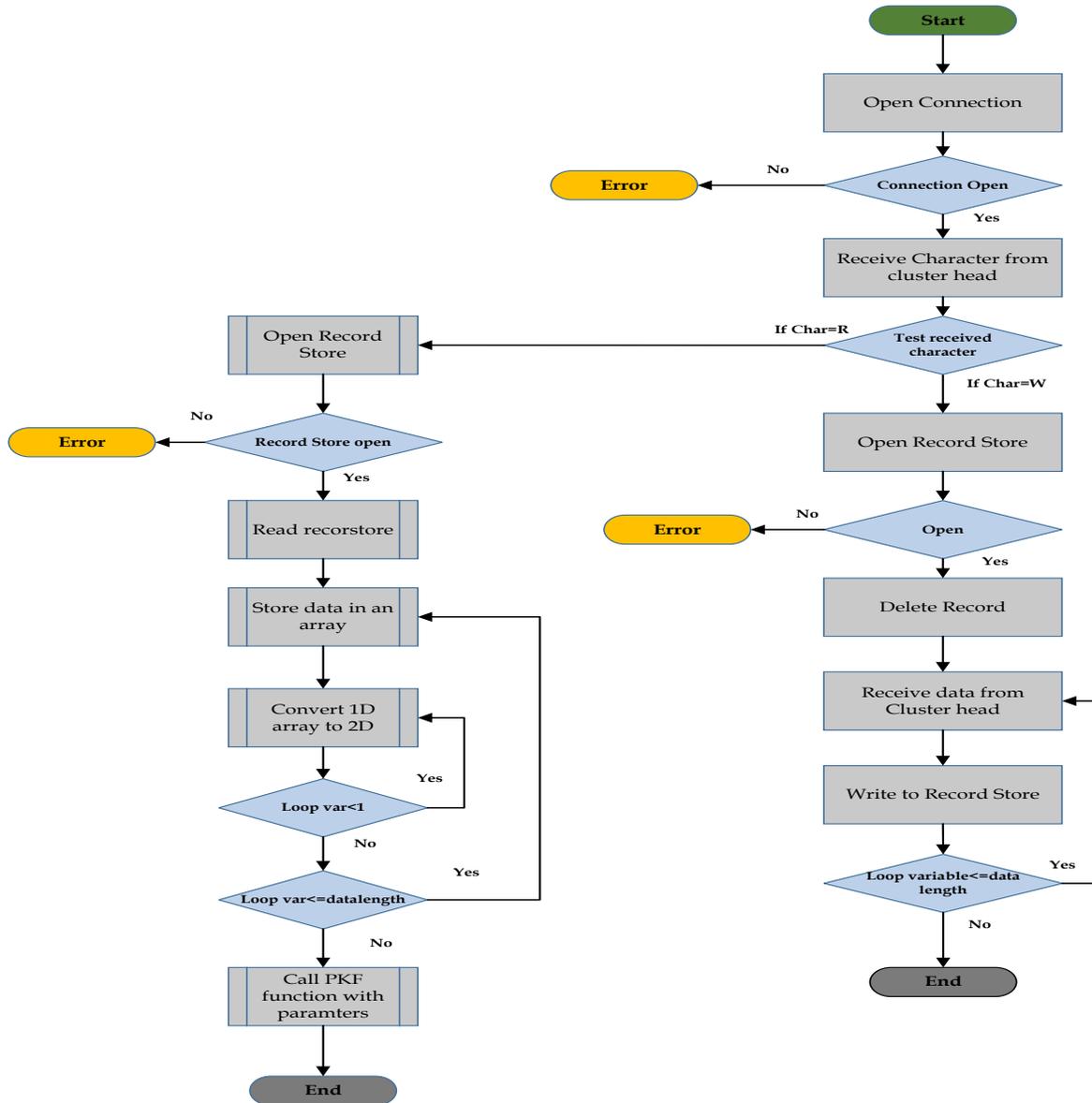


Figure 4-25 Flow chart for main function of SensorNode class

Figure (4-25) shows the main function flow chart for this class. First it open connection and receive character from cluster for selection of writing data into record store or receiving data from record store. If the character received is R then it will first open record store and start reading data from record. Then it will store this data into an array. This one dimensional array is then converted to two dimensional array. This process will continue until the whole data will finish. Now this two dimensional array is converted into matrix. This matrix and other parameters are passed as argument to PKF function.

The second section of flow chart shows about the data writing in the record store. First it open record store and then it delete previous record. Then it starts receiving data from cluster head. And then write data into record store. This will continue until total data length.

4.5.3 Cluster head

This section of code is for implementing on cluster head side. It consists of three classes.

4.5.3.1 ClusterHead

This class is the main class which is used to implement the predictor section of PKF algorithm.

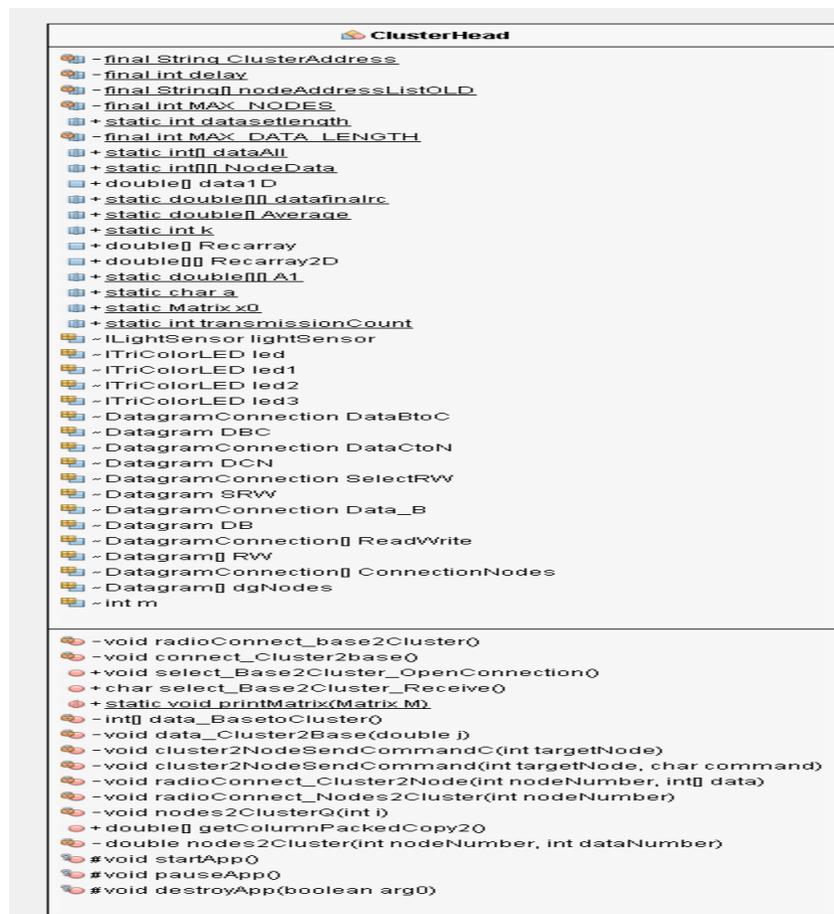


Figure 4-26 Class diagram for ClusterHead class

The class diagram of this class shows that all the methods are similar which were described before in the section of average calculation. The main difference is in the main function where predictor is implemented.

4.5.3.2 JPCD8544

This class is also the same class which was described in the previous section. This class is used to interface the LCD section.

4.5.3.3 PK_function

This is same class with modified PKF function here only we have implemented predictor. This predictor only predicts the future value there is no KF in this section of code.

4.5.3.4 Parameters_Pkf

This class is for parameters. It is also same as described before.

4.6 Code Path for SVN

This is path for the code section on SVN and classes in each section.

Section	Classes	Purpose	SVN Path
Desktop Section	<ul style="list-style-type: none"> FileMethod SendDataDemo Cluster 	This code section is used to run on desktop. It is used to send data and receive data.	S:\raza\Code2\Sensor_Desktop_Receive\src\org\sunspotworld\demo
Cluster head Section	<ul style="list-style-type: none"> ClusterHead JPCD8544 Paramers_Pkf PK_function 	This section of code is implemented on cluster head to communicate with base station and leaf node. It is also implemented predictor.	S:\raza\Code2\ClusterHead_May\src\de\imsas\inpDemo
Leaf Node Section	<ul style="list-style-type: none"> SensorNode Parameters_Pkf PK_function 	This section of code is implemented on leaf node. This section of code implement PKF algorithm and send value to cluster head.	S:\raza\Code2\Sensor_Node\src\de\imsas\inpDemo

Chapter 5

Graphical user interface (GUI)

This chapter consists of explanation about GUI for display of data and graph made for comparison.

5.1 Graphical user interface in Java

In Java there are two components which are used to design a GUI. These components have different pros and cons according to their applications. To design a GUI in Java it is very important to use a GUI component, which is most suitable to the application. There are two GUI components, which are used for GUI development in Java. These two components are named as

- AWT
- Swing

The first component which is named as AWT known as Abstract Window Toolkit. This component is used for GUI programming in Java. This is also called GUI library with respect to its applications it is a portable library. This library consists of large number of user interface. One of the basic properties of AWT is that it gives high level abstraction for a Java program, it hides lots of detail from the user at lower level.

The second component is named as Swing. It is based on AWT technology. One of basic properties of Swing is that it is totally implemented in Java programming. It has also property as a pluggable component. Swing components due to its property of Java implementation are not depended on peers. This property makes Swing components more lightweight as compared to the AWT components.

Both AWT and Swing components have some pros and cons depending on their applications and implementation in Java. These pros and cons are very important in the sense of their implementation in GUI design [20].

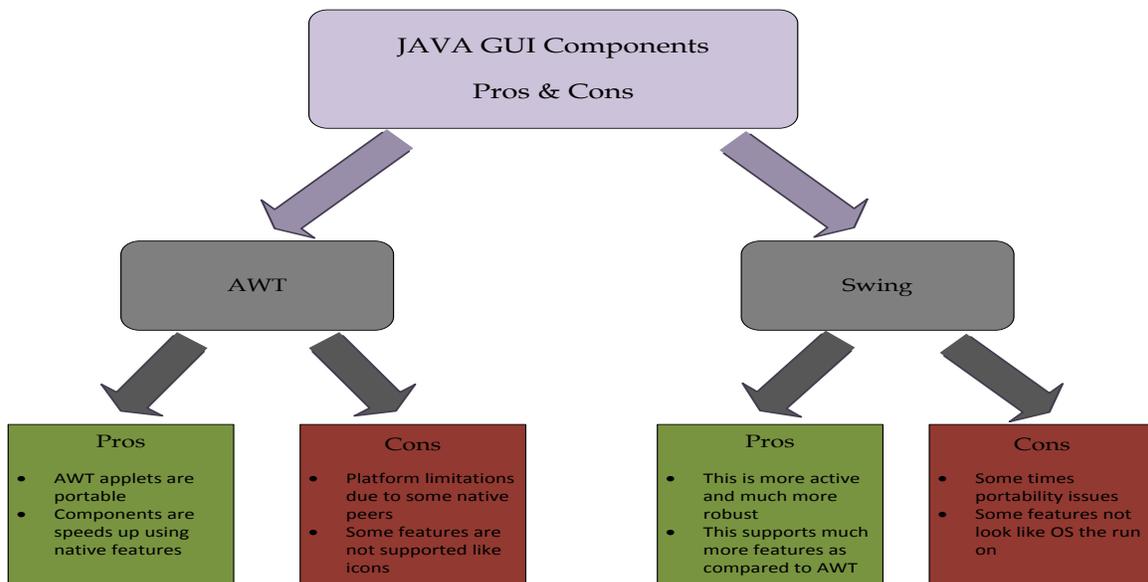


Figure 5-1 GUI components in Java

5.2 Front Panel of GUI design

In the front panel there are different buttons and text box. Total components on the front panel can be classified as follow

- Buttons
- Radio Buttons
- Text Area
- Progress Bar
- Labels

In the front panel there are three buttons which are used to select different options. First of all we can see there is a button in green color which is stated as Connect. When this button is selected it makes a connection of base station with cluster head. It is also being noted that when it is selected then it will release the other radio buttons for selection. When this is selected and connection is established successfully then it will display the status of connection.

The second button which is named as Run is very important button because after the connection established the entire GUI run with this button option. The third and the last button which is named as “Exit” on its button. This button is used to exit from the GUI. When all the functions are finished we can exit GUI by using this button.

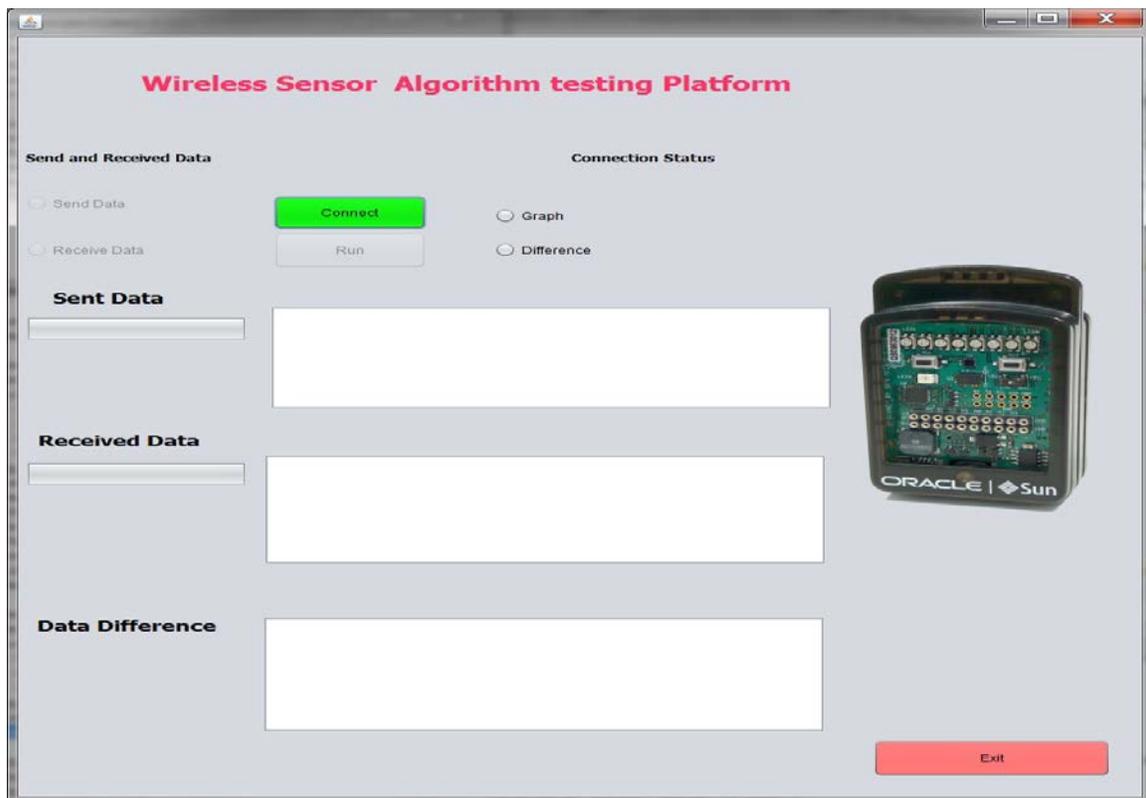


Figure 5-2 Front Panel of GUI design

There are total four radio buttons which are used for the different purposes. The radio button which is named as “Send Data” is used to send data on cluster head which forward data to leaf node. This data is taken from text file and then transmit over radio to a cluster head. Similarly there is radio button named as “Receive Data” this radio button is used when we want to get data from cluster head and node after implementation of algorithm.

The radio button which is named as difference is used when we want to calculate the difference between the send data and receive data. This radio button is used to confirm that the data send and receive is in tolerance range. The fourth and the last radio button is named as graph. This radio button is used to create a graph between send data and the receive button.

Text areas are used to display the text. Here in this GUI we have used the JTextArea these text areas are dynamic this means that when these text areas are used they adjust their size according to data set. It means when the data is more than the size of text area it will automatically generate the scroll bar to the text area. We have three text areas in the GUI which display three different data sets.

The first text area is related to the send data when we send data from the basestation to the cluster head the data which is sent to the cluster head will be displayed here. Similarly the text area which is associated with the receive data will display the data which is received from the cluster head after the implementation of algorithm. The third and the last text area is used to display the data which is received after the calculation of difference from the send and the receive data.

There are two progress bars which are associated with two text areas. The first progress bar which is related to the send data text area will give the progress according to the data displayed in the send data text area. This progress bar tells the progress of data in the percentage. Similarly the second progress bar will tell the progress of the data set receive in the receive data text area. This will show the progress of data set which is received after the implementation of algorithm.

There are also labels used in this GUI front panel. These labels are used to give the name of different areas of the GUI. There are total seven labels which are used here in the GUI. First label which is used is the title of the GUI. There are other labels which showed other text for the display. Among these labels one label is used to show the picture of the wireless sensor node named as Sun SPOT. There are labels used to give the name of text areas which are used to display the data. These three labels show the name such as Send Data, Receive Data and the Difference between the data. There are two other labels which are used to display the connection status section and also to display the section of data send and receive.

In the programming section it was such organized that every step will work in a proper way. Such that if we want to select a radio button before that connection enabled then it is not possible. Similarly it is also keep in mind the selection of radio buttons. Radio buttons are organized such that they cannot be utilized without the run button. Here it is also grouped with an error message. If someone clicked the run button without selecting any radio button then it will automatically generate an error displaying that first select a radio button and then run button. The data which is displayed in the text areas is displaying one by one. It is also displayed the current value with the overall progress. This will also show the total number of data set value and how many values are completed. And when all the data is received then it will show the whole data in the respective text areas. In the next section there will be GUI in different scenarios displayed.

5.2.1 GUI front panel in different Scenarios

In this section GUI will be shown in different operational conditions that will overview

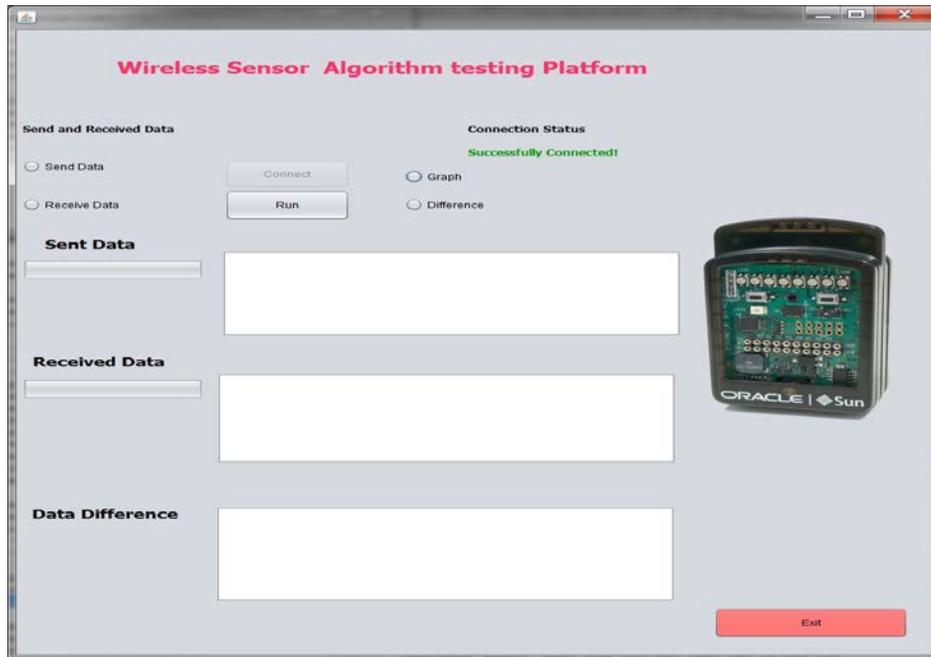


Figure 5-3 GUI displayed after the connection established

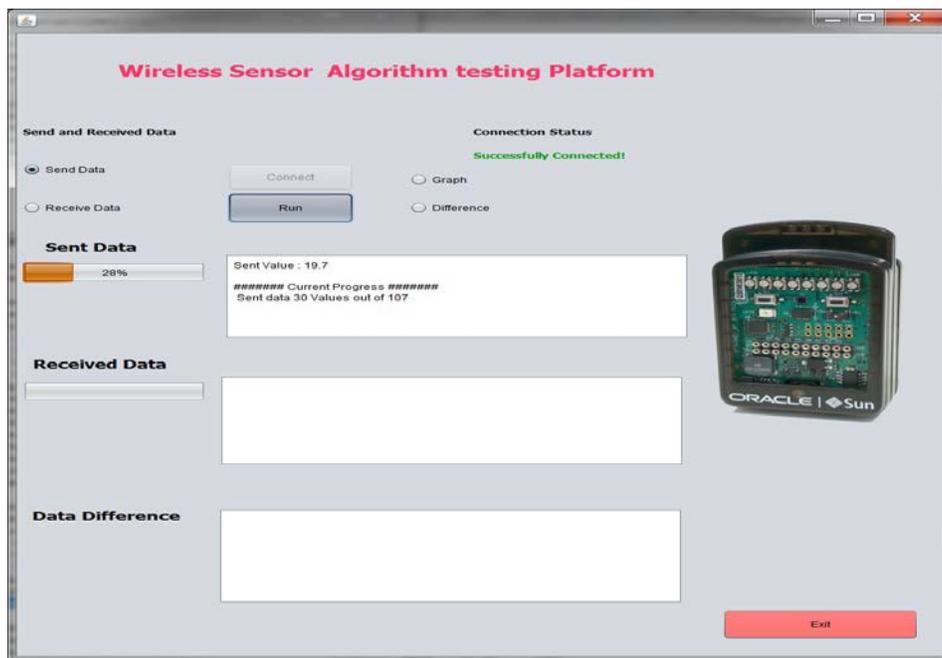


Figure 5-4 GUI displayed when data is being sent

5.2.2 Flow chart of GUI operation

GUI works in specific pattern. This pattern can be explained using a flow chart. The flow chart which is given in figure below explains how GUI works and how in some conditions GUI will stop working.

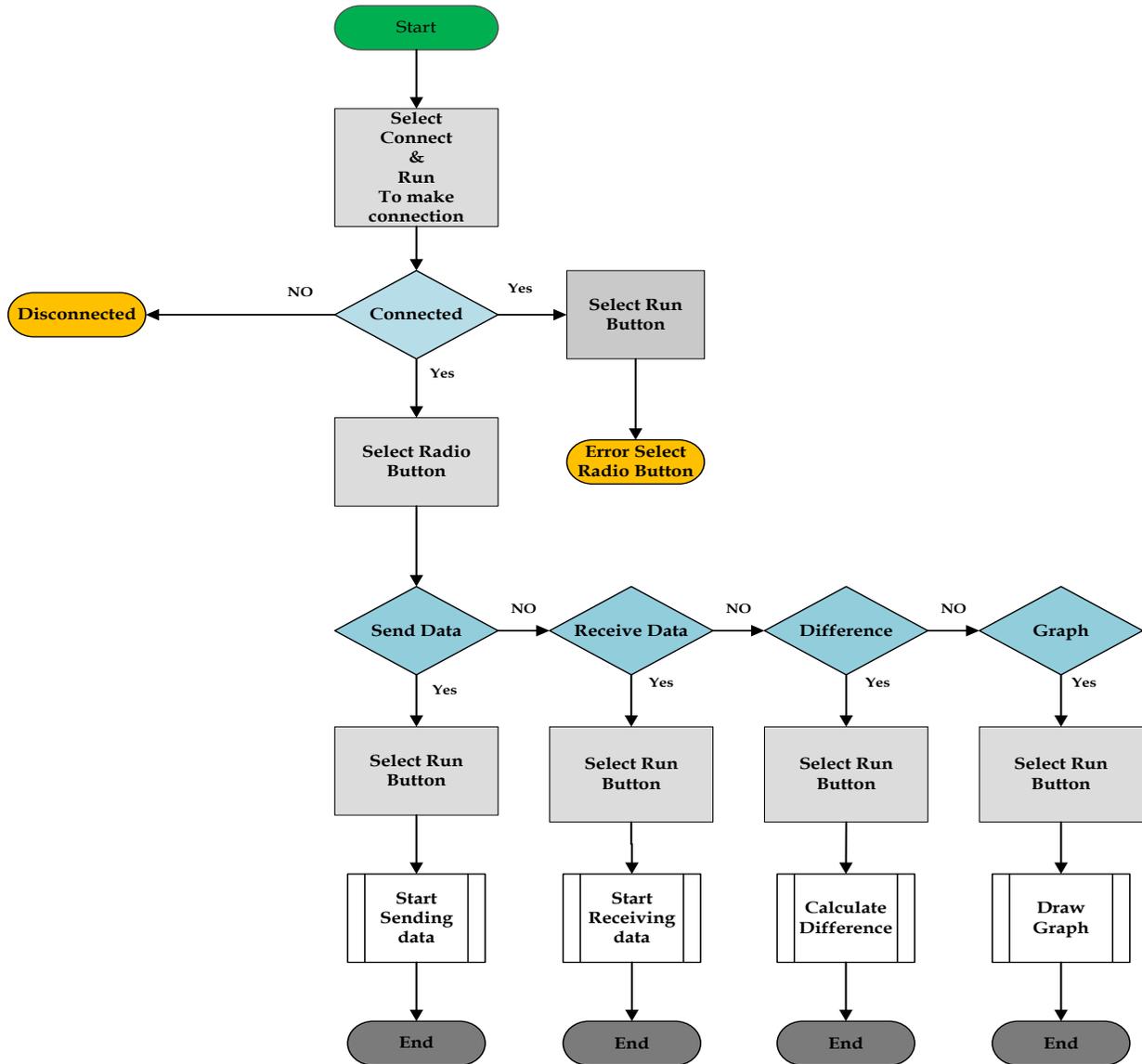


Figure 5-4 Flow chart of overall operation of GUI

5.2.3 Graph using XChart

The better visualization for comparison will be done by using charts or graphs. In this whole implementation of algorithm and data set on wireless sensor nodes, we have two types of data set. One data is set is our raw data or stored temperature data which was transmitted from base station to the wireless sensor. The second data set was the data set which we have received from node after the implementation of PKF algorithm. This is also called the reconstruction value.

Furthermore we have a text box in the GUI front panel. This text box which is named as difference is used to get the difference from both data sets. This difference is can give an idea between two data sets. But more elegant way to get a difference between two data sets is to use a graph. There is no direct way to plot a graph in GUI using Swing. So we have used an open source library named as XChart. This library is not much heavier than jfree chart library very easy to use and we can easily plot data using this library.

The plotting using this library is not so complex. First of all we have to create a chart using chart builder. Then we have to add data according to requirements. The each data set is stored in an array list. After storing the data in array list we can either save it or display it using display method. In our case we have two data sets which have to be plotted on the same graph.

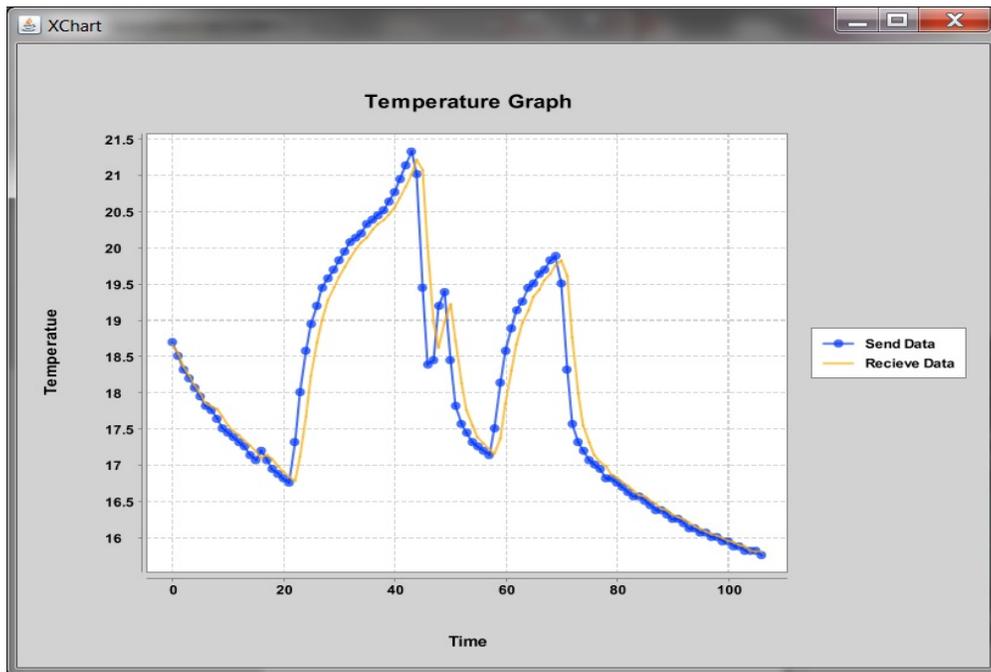


Figure 5-6 Graph using Xchart library for data Comparison

5.3 Results

Algorithm is tested by using different threshold values. Threshold value effects on the transmission ratio the table given below will tell how much the transmission at different threshold value

Threshold	Total data	Update Values from node	Transmission rate %	RMSE Root mean Square Error
0.008	214	202	96	0.147
0.03	214	166	80	0.148
0.13	214	104	50	0.352

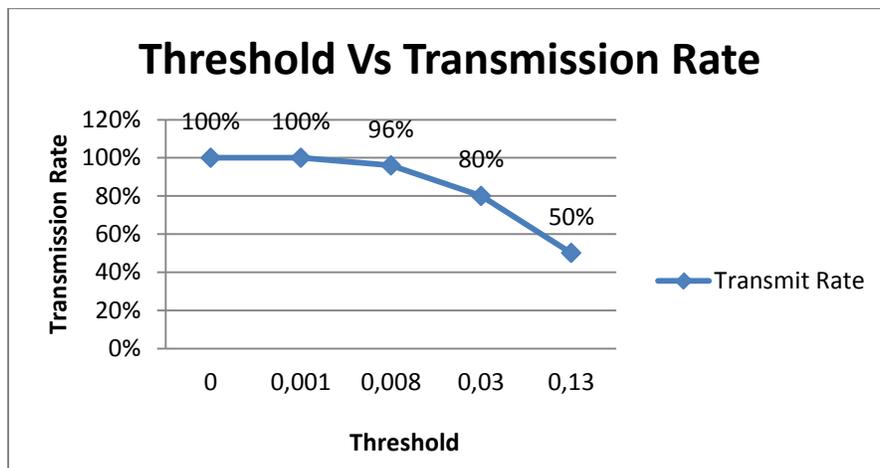


Figure 5-7 Relation between threshold and transmission rate

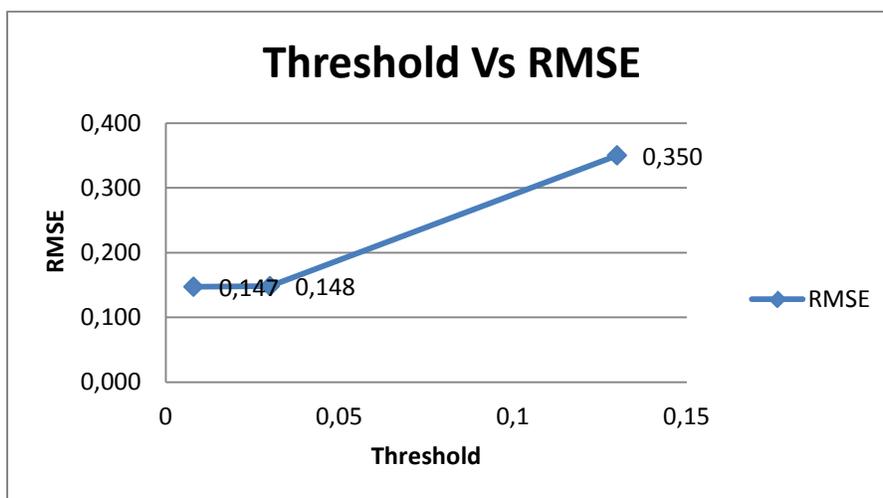


Figure 5-8 Relation between threshold and RMSE

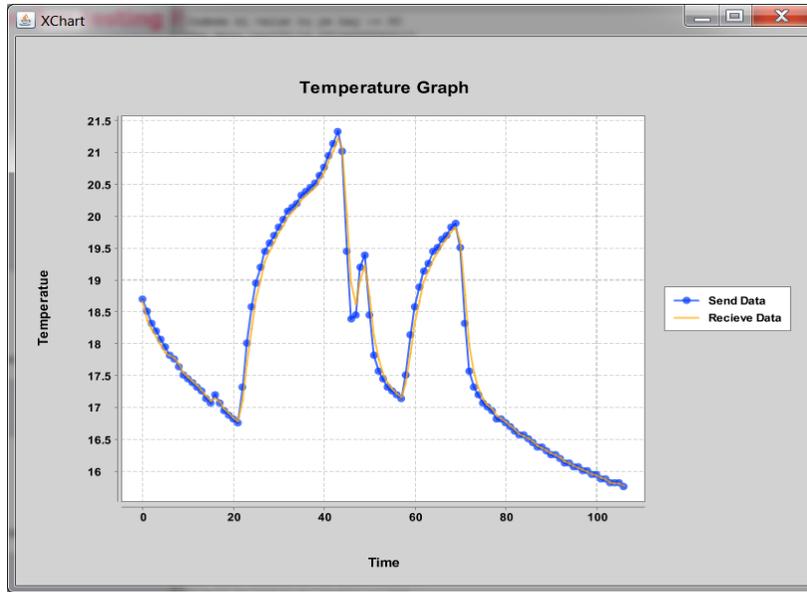


Figure 5-8 Graph displayed with 96% Transmission

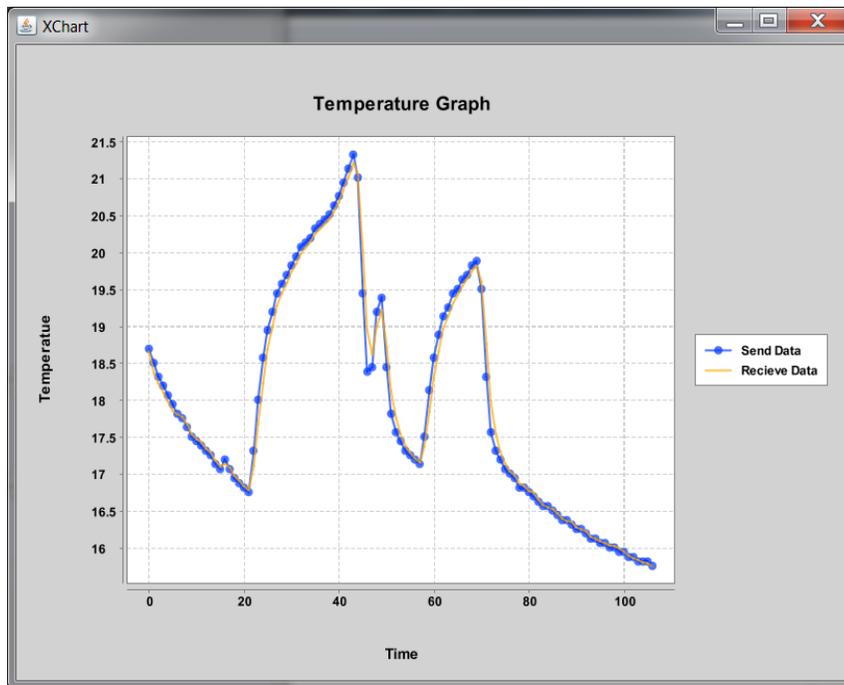


Figure 5-9 Graph displayed with 80% Transmission

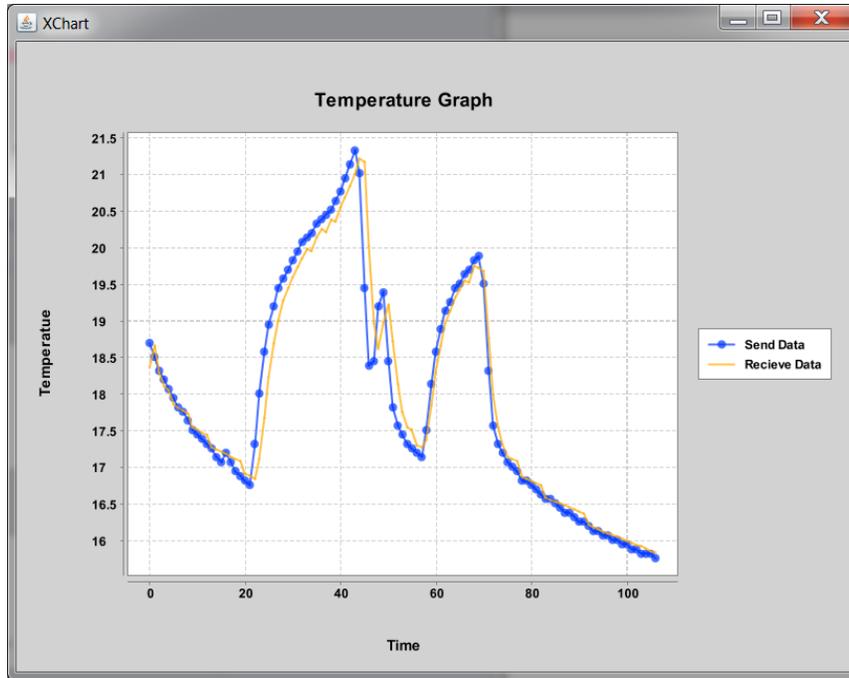


Figure 5-10 Graph displayed with 50% Transmission

Summary and Conclusion:

The task of thesis was to develop a platform based on Sun SPOT kit which can demonstrate successfully implementation of distributed algorithm in a wireless sensor network. The task was successfully achieved. It was divided in such a way that first we have to make sure that the data of temperature which was locally stored in a text file should be transmitted to the wireless sensor nodes using a cluster head. It can be named as the initial task of this thesis that was a building block to this master thesis.

In this task data was transmitted to each node in such way that it was stored in the flash memory of each node. There are total three nodes and one cluster head which were used to test this thing. The program is designed in such a way that it can be extended to any number of nodes depending on the requirement. After storing the data in the flash memory it can be easily playback from the flash memory of the nodes. This was initial requirement of this thesis. To test this thing at the beginning an average was calculated from all the three nodes data on cluster head which was transmitted to base station.

In the hardware section the task was to integrate a LCD with cluster head to display the data received by different nodes and addresses of nodes. The task was to search market and find some economical solution for LCD. It was successfully achieved. The most economical LCD Nokia 5110 was selected and integrates with the cluster head. Although the driver library was not available for Sun SPOT kit the available library was compatible with Arduino. We have convert this library and made it compatible with Sun SPOT kit. There are no SPI pins available on Sun SPOT demo board so digital pins were used using bit banging technique.

Although initially the task was to implement any available algorithm to test this platform. But later on algorithm which was proposed by ITEM department was implemented. The main achievement during this thesis was to convert PKF algorithm from Matlab to Java. The algorithm was originally written in Matlab so after conversion this Matlab algorithm into Java the whole algorithm was tested and verified using cross platform technique using in Matlab. This algorithm was not only converted but also implemented on the wireless sensor network consists of one node and one Cluster head.

Compression rate of 50% can be achieved by adjusting the threshold value. But to achieve this compression rate the root mean square error (RMSE) value will be larger. Computation time for PKF algorithm on node side is about 128ms for single value and for total data set which comprises of data length of 107 values this time is about 6427ms. While on the other side the predictor which is implemented on cluster head has computation time about 40ms.

The visualization plays an important role to demonstrate and understand a typical process. For this it was thought that there should be a GUI in the thesis that will demonstrate our whole implementation properly. This task was on choice that if we have a time left than it will be implemented. So it was successfully implemented by using Swing component in Java. One thing which was additionally added to this GUI was Graphical representation for comparison of data. This was successfully implemented using an open source library named as XChart.

Although the algorithm is implemented successfully using one cluster head but in future it is also possible to implement this algorithm using multiple cluster heads in the network. This means this algorithm can also be extended according to the requirement in the future. Hopefully in near future a research paper will also be written on this topic.

References:

1. A. C. Ranasinghe, L. K. Rasnayake and M. Kalyanapala, "Reconfigurable universal sensor interface for distributed wireless sensor nodes," *Advances in ICT for Emerging Regions (ICTer)*, 2013 International Conference on, Colombo, 2013, pp. 189-193.
doi: 10.1109/ICTer.2013.6761177
2. J. Cecilio, P. Furtado Wireless Sensor Networks: Concepts and Components, *Wireless Sensors in Heterogeneous Networked Systems*, Computer Communications and Networks, DOI 10.1007/978-3-319-09280-5_2
3. O. Younis, M. Krunz and S. Ramasubramanian, "Node clustering in wireless sensor networks: recent developments and deployment challenges," in *IEEE Network*, vol. 20, no. 3, pp. 20-25, May-June 2006.doi: 10.1109/MNET.2006.1637928
URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1637928&isnumber=34335>
4. Sun SPOT manual theory of operation <http://www.sunspotdev.org/docs/index.html>
5. Sun SPOT http://anrg.usc.edu/ee579_2012/Group07/files/sunspots.html
6. LCD datasheet <https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf>
7. Nokia LCD 5110 <https://www.engr.colostate.edu/ECE251/Labs/Lab9.pdf>
8. SPI Communication <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all.pdf>
9. Sun SPOT eDEMO technical datadsheet <http://www.sunspotdev.org/docs/index.html>
10. Distributed Algorithms in Wireless Sensor Networks, Tim Nieberg Universiteit Twente.
11. Christoph Lenzen¹ and Roger Wattenhofer², Distributed Algorithms for Sensor Networks ,¹School of Engineering and Computer Science, Hebrew University of Jerusalem Edmond Safra Campus, Givat Ram, 91904 Jerusalem, Israel ²Computer Engineering and Networks Laboratory, ETH Zurich Gloriastrasse 35, 8092 Zurich, Switzerland, 2012.
12. JAMA Package <http://math.nist.gov/javanumerics/jama/>
13. Kalman filter <http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/>
14. Kalman filter <https://www.cs.cornell.edu/courses/cs4758/2012sp/materials/mi63slides.pdf>
15. Yanqiu Huang, Wanli Yu, and Alberto Garcia-Ortiz, PKF-ST: A Communication Cost Reduction Scheme Using Spatial and Temporal Correlation for Wireless Sensor Networks, EWSN' 16 Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks, Pages 47-52
16. Yanqiu Huang, Wanli Yu, Christof Osewold, and Alberto Garcia-Ortiz, Analysis of PKF: A Communication Cost Reduction Scheme for Wireless Sensor Networks, IEEE Transaction on Wireless Communications, Vol. 15, Pages 843-856, NO. 2, February 2016,
17. Wireless Signal Symbol http://png.clipart.me/graphics/thumbs/209/vector-wireless-network-icon_209116744.jpg
18. Tossaporn Srisooksai^a, Kamol Keamarungsi^b, Poonlap Lamsrichan^c, Kiyomichi Araki^d , Practical data compression in wireless sensor networks: A survey
Journal of Network and Computer Applications , Volume 35, Issue 1, January 2012, Pages 37–59
19. UML Diagram https://www.tutorialspoint.com/uml/uml_overview.htm
20. GUI Packages <http://www.users.di.uniroma1.it/~parisi/Risorse/AWTvsSwing.pdf>
21. XChart library <http://knowm.org/open-source/xchart/>
22. Basic of Flash memory <http://www.eeherald.com/section/design-guide/esmod16.html>
23. Stephan Korsholm VIA University College Horsens, Denmark, Flash memory in embedded Java programs

24. eSPOT manual <http://www.sunspotdev.org/docs/index.html>
25. Sun SPOT programmer Manual <http://www.sunspotdev.org/docs/index.html>
26. Record Management Store <http://www.inf.unibz.it/~ricci/MS/slides-2010-2011/4-J2ME-DATABASE.pdf>
27. V. Raghunathan, C. Schurgers, Sung Park and M. B. Srivastava, "Energy-aware wireless microsensor networks," in IEEE Signal Processing Magazine, vol. 19, no. 2, pp. 40-50, Mar 2002. doi: 10.1109/79.985679
28. M. Kumrawat and M. Dhawan, "Optimizing energy consumption in wireless sensor network through distributed weighted clustering algorithm," Computer, Communication and Control (IC4), 2015 International Conference on, Indore, 2015, pp. 1-5.
29. F. Kiani and M. Fahim, "Energy Efficiency in Wireless Sensor and Actor Networks by Distributed Time Synchronization Algorithm," Computational Intelligence & Communication Technology (CICT), 2015 IEEE International Conference on, Ghaziabad, 2015, pp. 344-348.
30. A. A. Abba Ari, A. Gueroui, B. O. Yenke and N. Labraoui, "Energy efficient clustering algorithm for Wireless Sensor Networks using the ABC metaheuristic," 2016 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, 2016, pp. 1-6.