Information Engineering and Technology Faculty
German University in Cairo

GUC Bus Fleet Speed Control

Bachelor Thesis

Author:	Ahmed Mamdouh
Supervisor:	Dr. Amr Talaat
Reviewer:	Name of the Reviewer
Submission Date:	15 June, 2011

This is to certify that:

(i) the thesis comprises only my original work towards the Bachelor Degree

(ii) due acknowledgement has been made in the text to all other material used

_____

Ahmed Mamdouh

15 June, 2011

Acknowledgments

I would like to thank all of my family members who provided me with all kinds of support and motivation. Also I would like to thank my highly appreciated instructor and supervisor Dr. Amr Talaat for his devotion with time and effort supporting us in drawing an outline , time plan and visualizing our projects early before we even started as well as the useful feedback on our weekly meetings . Moreover, delivering my thanks to Fady Gamal and Yousef Sedky for their support by providing material and useful links for software and tools downloads, thanks as well to Ashraf El Miligy for his continuous support trying different approaches at the zigbee stage. Also my Great thanks to Eng. Diaa Motawe for his support in the Data base stage . Last but no least my thanks to Eng. Mostafa Mamdouh for supplying me with handy resources and useful guidance in my thesis writing.

Abstract

Nowadays, monitoring and control functions became essential and crucial to ensure success of any project or organization and positive perception for the targeted customers of this service. Moreover, no matter what services or products an entity offers its projects must be efficiently and effectively developed and delivered. And hence to maintain and enhance the quality of service provided through the GUC transportation network a monitoring system was required for the GUC bus fleet. This monitoring system keeps trace of 3 parameters ; maximum speed, distance covered and fuel consumption in each round evaluating drivers efficiency and performance through recording data then transferring these data via transceiver devices to a central server on campus grounds which analyses these data then stores them on a central database to finally produce full evaluation report .

Table of Contents

Chapter 3 Project Specification and implementation

Chapter 4 Conclusion

Chapter 1

Introduction

1.1 Motivation

Complaints raised by students using the GUC bus fleet focusing on late arrivals , less ride safety . Issues raised by Budgeting and Financing Departments highlighting the excessive fuel cost over budget.

1.2 Aim of the project

This project was implemented to monitor and control the GUC bus fleet drivers performance according to a pre defined bench mark .This value added service is the channel for the GUC in order to reach the targeted level of quality. Generally speaking this project serves any organization or institute having a bus fleet.

1.3 Methodology

In order to obtain the required output. It was essential to divide the methodologies used in the project into stages

Stage1: A survey was done on monitoring systems and relative studies using technologies that shall be used in this project implementation.

Stage2: It's the planning stage , figuring out all entities in the project and how will they get linked and interact as a system.

Stage3: This stage included deciding which programming language to be used for implementation .

Stage 4: implementing the project according to the plan predefined using project tools.

Stage 5: testing stage

Stage 6: thesis writing

1.4 Thesis outline

This report is divided into 4 chapters, each chapter starts with a brief introduction and then sectioned according to its content and type.

Chapter 2: Discusses the relative background to the scope of study focusing on technologies used in implementing this project.

Chapter3: This chapter defines project specifications giving a brief description on the project and defines all entities . Moreover, it covers the tools in implementation as well as implementation steps and coding .

Chapter4: Conclusion covering 2 parts first ,the conclusion part regarding the report the second part stating briefly explained ideas that can be implemented or extended in parallel with this project.

Chapter 2
State-of-Art

This chapter will cover a brief idea on monitoring systems and the theoretical background of the wireless networks , standards and zigbee technology in general as well as the major role of databases in the archiving process and recording of vital information in any system discussing their different structures advantages and disadvantages of each.

2.1 Monitoring Systems : Brief idea

Since early ages managers valued the concept of monitoring and keeping an eye on their subordinates and employees working in their business or below them in the business hierarchy. The process of monitoring was performed in various ways and techniques fitting with the timing and the culture of the business ,institute or organization.

Monitoring is the systematic collection and analysis of information as a project progresses. It is aimed at improving the efficiency and effectiveness of a project or organization. It is based on target sets and activities planned during the planning phases of work. Also it keep the work on track, acknowledging management when anything occurs away from the previously planned. Moreover, it provides a useful base for evaluating the entity's performance as whole and employees or workers on an individual scale. In addition , it draws the management attention to the efficiency of processing and the degree of resources utilization ensuring that the flow of work is going as planned .

There are several types of monitoring :

1-Staff Monitoring

2-Resources Monitoring

3-Production Monitoring

Scope:

What the project focuses on is Monitoring the driving performance of the GUC bus fleet drivers first, the project targeted speed control but it extended to tackle distance covered as well as fuel consumption in each round through these 3 parameters drivers driving performance and quality are evaluated .

2.2 Wireless Networks

refer to any type of network that is not connected with cables.

There are several types of Wireless Networks :

- Metropolitan Area Network “MAN”

Connects several LANs

- Wide Area Network “WAN”

Network covering large areas also connecting several LANs ,with wireless connections between access points are point to point micro-wave links.

- Local Area Network “LAN”

A wireless network linking between 2 or more devices providing a connection through an access point.

- Personal Area Network “PAN”

Interconnect devices within a relatively small area of about 10 meters using Bluetooth, Infrared as well as Zigbee devices.

2.2.1 Wireless Personal Area Network WPAN

General Description :

Wireless personal area network is a low range area network which covers an area of only few dozen meters about 10 meters range . This type network is generally used linking peripheral devices to a computer or even 2 near by computers. Although the limited range it showed great value in various applications. There are several kinds of technology used for WPANs like Bluetooth, infra red and Zigbee transceiver devices.

The Zigbee technology complies with the well known standard IEEE 802.15.4 used to connect devices wirelessly at :

- Very low cost.
- Little energy consumption.
- Operates on frequency range of 2.4 GHz on 16 channels.
- Transfer speeds up to 250 Kbps.
- Max. range up to 100 meters.

2.2.1.1Transceivers

A transceiver or transmitter/receiver is a device which combines transmission and reception capability on shared circuitry. There are a number of different types of transceivers designed for an assortment of uses, and the transceiver is the cornerstone of wireless communication. One common example of a transceiver

is a cellular phone, which is capable of sending and receiving data, unlike a basic radio, which can only receive signals.

Transceivers can be divided into two rough categories: full and half duplex. In a full duplex transceiver, the device can transmit and receive at the same time. Cell phones are, again, an excellent example of a full duplex transceiver, as both parties can talk at once. By contrast, a half duplex transceiver silences one party while the other transmits. Many radio systems operate on a half duplex method, which is why people signal when they are going "out," alerting the other user to the fact that the frequency is open for transmission.

The main purpose of a data transceiver is to provide a seamless transfer of information between two or more points. In order to accomplish this task, the transceiver will process data as it is released, qualify the data for reception, and then finally deliver the data to the recipient. Depending on the degree of processing that is necessary to manage the successful delivery, there may be a slight delay that usually amounts to no more that a second or so. For example, if there is a need to convert analog data into a digital format in order to complete the exchange, this may take slightly longer than if all the points involved in the data exchange are utilizing the same type of signaling.

As with all types of transceivers, the data transceiver offers the benefit of doing away with the need to employ some type of transmission device along with a receiver. Since the same device manages both functions, this means a more streamlined movement of the data from a point of origin to a point of termination. Assuming that the speed of the data stream and the capacity of the transceiver are in sync, the communication of the data may almost appear to take place in real time, since the slight lag will be negligible to the parties involved in the exchange. The end result is the exchange of data without any loss of integrity, making the tasks of sharing and receiving information quick, easy, and secure.

2.2.1.2 IEEE 802.15.4 Standards

The IEEE 802.15.4 standard is a simple packet data protocol for lightweight wireless networks, control of lights, security alarms. IEEE 802.15.4 specifies the Physical (PHY) and Medium Access Control (MAC) layers to ensure low power consumption and to supply reliable data transmission up

to 100 meters.. IEEE 802.15.4 is typically less than 32 kb in size, featuring a 64-bit address space, source and destination addressing, error detection, and advanced power management.

2.2.1.3 Zigbee Technology

Zigbee is the only standards-based wireless technology designed to address the unique needs of low-cost, low-power wireless sensor and control networks in just about any market. Since Zigbee can be used almost anywhere, is easy to implement and needs little power to operate, the opportunity for growth into new markets, as well as innovation in existing markets, is limitless. Here are some facts about Zigbee:

- With hundreds of members around the globe, Zigbee uses the 2.4 GHz radio frequency to deliver a variety of reliable and easy-to-use standards anywhere in the world.
- Consumer, business, government and industrial users rely on a variety of smart and easy-to-use Zigbee standards to gain greater control of everyday activities.
- With reliable wireless performance and battery operation, Zigbee provides the freedom and flexibility for various applications.

2.2.1.4 Zigbee Characteristics

The focus of network application under the IEEE 802.15.4 / Zigbee standard include the features of low power consumption . just needed for only two modes (Tx / Rx or sleep) , high density of nodes per network , low costs and simple implementation.

These feature are enabled by the following characteristics :

- 2.4 GHz and 868/915 MHz dual PHY modes.

- Allocated 16 bit short or 64 bit extended addresses.

- Maximum data rates allowed for each of these frequency bands are fixed as 25o Kbps @2.4 GHz, 40 Kbps @915 MHz and20Kbps @868 MHz.

- Allocation of guaranteed time slots (GTS)

- Link quality indication(LQI)

- Energy detection (ED)

- Fully “Hand-shake” acknowledged protocol for transfer reliability .

- Carrier sense multiple access with collision avoidance (CSMA-CD) channel access Yields high through put and low latency for low duty cyclesdevices like sensors and controls.

- Multiple topologies: Star ,Peer to Peer , mesh topologies.

2.2.1.5 Devices Types

The Zigbee PAN Coordinator node :

The most capable device , coordinator forms the root of the network tree and might bridge to other networks its able to store information about the network . there is one and only one zigbee coordinator node in each network to act as a router to other networks. Also acts as a repository for security keys.

The Full Function Device (FFD):

The FFD is an intermediary router transmitting data from other devices. It requires less memory capacity from zigbee coordinator mode, as it requires less manufacturing costs acting as a coordinator in all topologies.

The Reduced Function Device (RFD):

This device is capable of talking within the network without relaying data from other devices , requiring even less memory (no flash and very little ROM & RAM), and hence its cheaper than FFD. Finally this device talks only to the network coordinator and only implemented in star topology.

A FFD can talk to RFDs or other FFDs, while a RFD can tak only to another FFD.

A RFD is intended for extremely simple applications like light switch or passive infra red sensor , they don’t need to send large amount of data and may only associate with a single FFD at a time .

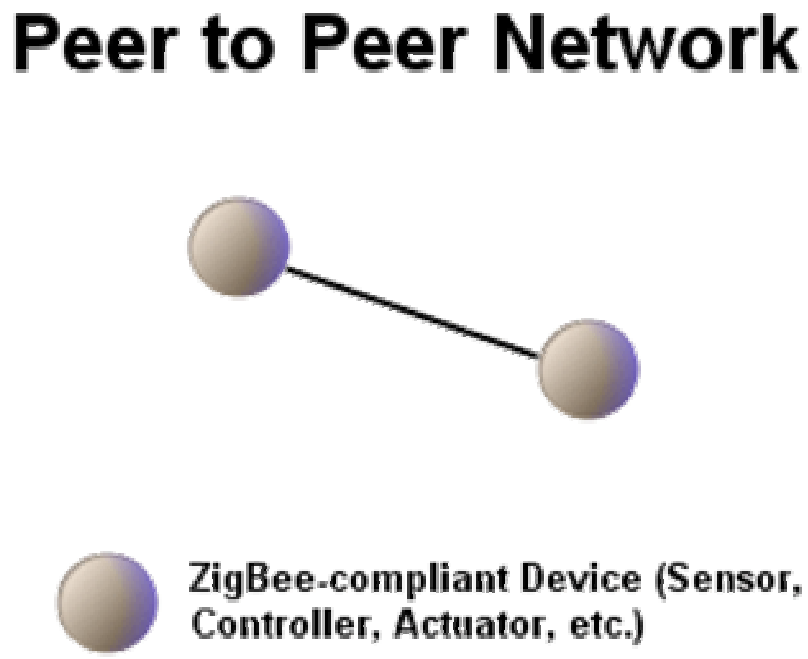
2.2.1.6 Network Topologies

Types of Zigbee Networks :

Zigbee Networks can be configured to operate in a variety of different ways to suit the application and environment .

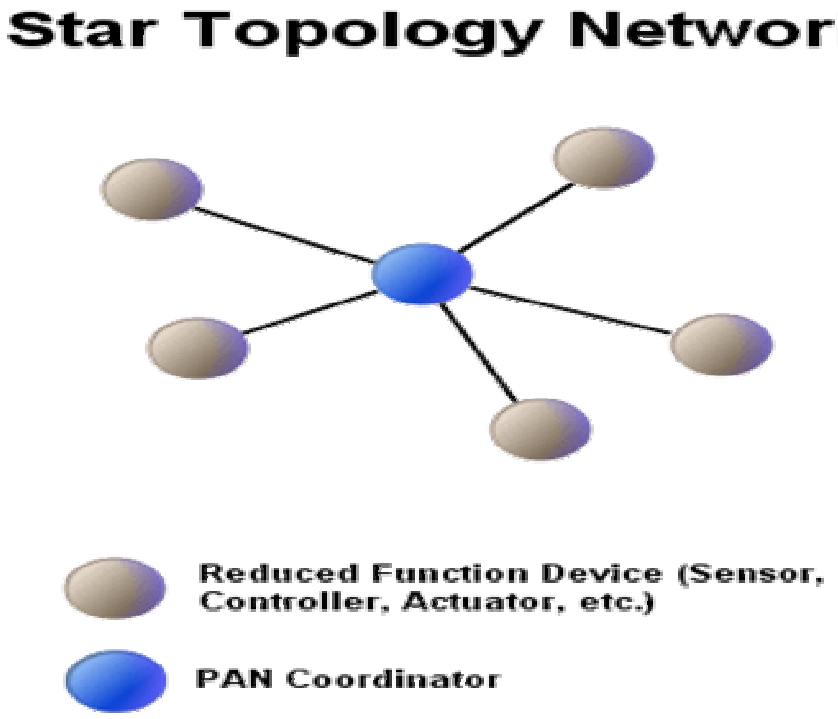
- Peer to Peer (Ad-hoc) Topology

In peer to peer topology zigbees are networked directly to each other . One node of zigbee device has the option of connecting multiple devices and hence forming a network in a grid like structure.



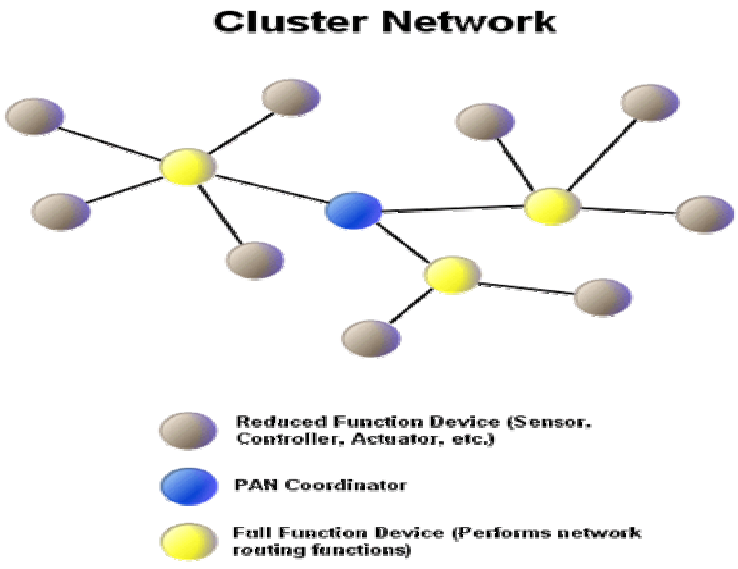
- **Star Topology**

All devices have one central control unit and device are attached through coordination with the help of multiple routers to increase zigbee network range .



- **Cluster Tree Topology**

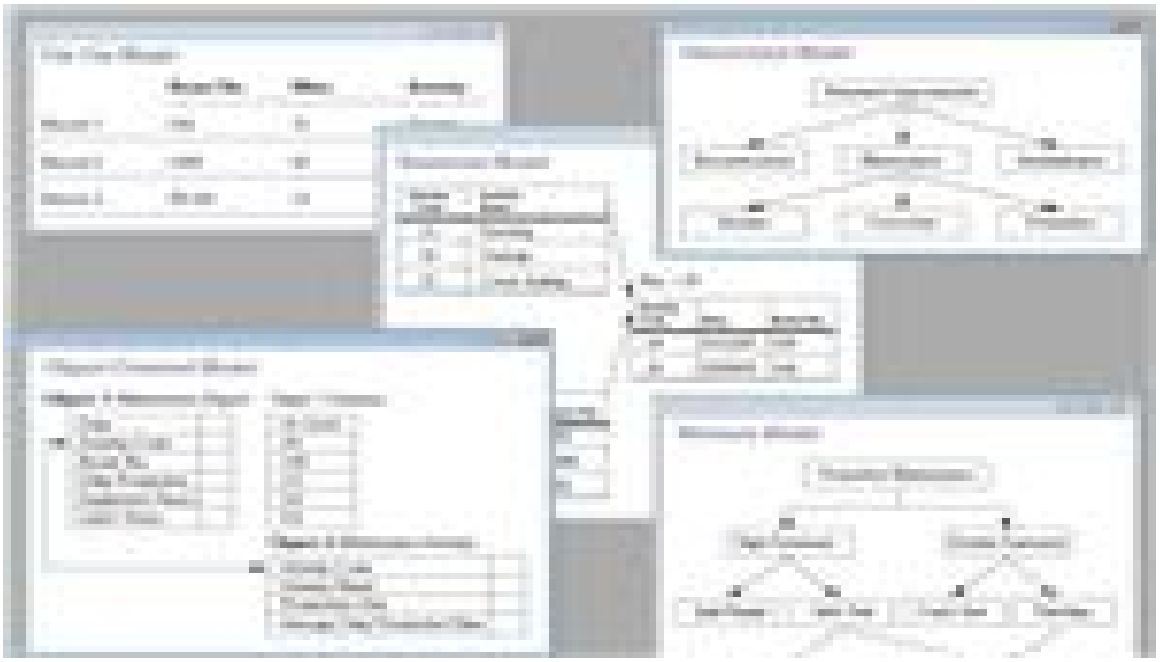
These clusters are many stars with routers connected with the PAN coordinator to form a Zigbee Network. In this topology the network never stops working even if one device or even a star is not in a working condition. It continues functioning with the available functional nodes.



2.3 Database Background

1.1 2.3.1 What is a Database Management System?

A Database Management System or DBMS is a single or set of computer programs that are responsible for creating, editing, deleting and generally maintaining a database or collection of data records. They type of database management system is determined by the database model. A database model is the manner in which the data collection is stored, managed and administered. The various database management systems based on these data models are:



These are the five major classifications for types of database management systems:

2.3.2.1 Relational Database Management Systems

Relational database management systems are the most widely used database management systems today. They are relatively easy to use. Relational database management systems are named so because of the characteristic of normalizing the data which is usually stored in tables. The relational model relies on normalizing data within rows and columns in tables. The data can be related to other data in the same table or other tables which has to be correctly managed by joining one or more tables. Relational models may be somewhat less efficient than other models; however this may not be a problem with the processing power and memory found in modern computers. Data in this type of model is stored is fixed predefined structures and are usually manipulated using Structured Query Language (SQL). Relational database management systems include Oracle, Ms SQLServer, IBM DB2, mySQL, SQLite and PostgreSQL among others.

2.3.2.2 Flat File Based Database Management Systems

Flat File based database management systems are probably the simplest of them all. These are sometimes called Flat models. These come in human readable text formats as well as in binary formats. These are ideal for stand alone applications, holding software configuration and native format storage models. Flat files in a formatted row and column model rely on assumptions that every item in a particular model consists of the same data. One common example of this type of database is the CSV (Comma Separated Values) and another is a spreadsheet such as Ms Excel.

2.3.2.3 Hierarchical Database Management Systems

Hierarchical database management systems operates on the parent child tree-like model. These normally have a 1:N relationship and are good for storing data with items describing attributes, features and so on. These could store a book with information on chapters and verses. They can also be used to store a database of songs, recipes, models of phones and anything that can be stored in a nested format.

Hierarchical database management systems are not quite efficient for various real world operations. One such example of a Hierarchical database management system is a XML document

2.3.2.4 Network Database Management Systems

A Network database management system uses a data model similar to Hierarchical database management systems The major difference here is that the tree structure in the Network models can have a many parent to many child relational model. The Network model structure is based on records and sets and most of these databases use SQL for manipulation of their data. Network database management systems tend to be very flexible but are rarely used ad were very quite common in the1960s and 1970s. Searching for an item in this model requires the program to traverse the entire data set which is quit cumbersome. These have mainly been replaced by Relational database management systems in today's modern computing.

2.3.2.5 Object-oriented Database Management Systems

Object-oriented database management systems borrow from the model of the Object-oriented programming paradigm. In this database model, the Object and its data or attributes are seen as one ad accessed through pointers rather than stored in relational table models. Object-oriented database models consist of diverse structures and is quite extensible. This data model was designed to work closely with programs built with Object-oriented programming languages thereby almost making the data and the program operate as one. With this model applications are able to treat the data as native code. There is little commercial implementation of this database model as it is still developing. Examples of Object-oriented database management systems include IBM DB4o and DTS/S1 from Obsidian Dynamics.

Chapter 3

3.1 Project specification

This chapter defines the project specifications and the system overview. It also clarifies the project structure, the different nodes in the project included and how these nodes or stages communicate together.

3.1.1 Brief Project Description

The project as previously mentioned in early chapters is a monitoring system keeping trace ,recording and evaluating the overall driving performance of the GUC bus fleet drivers through tackling 3 parameters which are: maximum speed reached , distance covered and fuel consumption in each round.

This system is implemented in 4 stages. Firstly, data is stored from a specialized black box that records all the three variables / parameters during the bus trip. Secondly, data is being fetched once the bus arrives to the gate, where a Zigbee transceiver device at the bus side communicates with another zigbee device at the gate's side. The gate won't open unless the driver presses on a button that transmits the stored data, then opens the gate, once the transmitted data is received and validated successfully. Thirdly, data is transferred from the Zigbee raw format to a database, which is MySQL server that sorts out these data , stores them on a central database in the university. Forthly, a set of sql scripts rungs against the database and generates ll reports which evaluates the performance of each driver on each ride on demand. The is

done through a User friendly application , Graphical User Interface “GUI”. Different options could be added to the systems, such as DSS, Decision Support System, which automatically analyzes the data, provide user with different warnings and performance indiciators, to further enhance and reduce both cost and time loss in future maintenance and routes for the buses. The system could also be enhanced with different GPS routing system, that could provide feedback on routes used, shortest path, and cost related to different routes, where new routes could be generated. New routes generated could be longer in milage, but more cost effective due to a more streamlined traffic flow.

3.2 Project Implementation & Experimental Results

This section covers the setup of tools used and steps taken for implementation. This assumes the first stage has been completed where the data bas been collected inside the bus blackbox.

3.2.1 Zigbee stage

This stage covers the setup of the devices, and the transfer of the data from the bus to the computer storing all bus related data, from the bus black box for the current round. This is stage 2.

Setting up XBee ZNet 2.5 (Series 2) modules

relating two XBee ZNet modules with the appropriate firmwares and setting them up to communicate with each other.

Materials needed:

2 - XBee ZNet 2.5 Modules (Digikey Part#: XB24-BWIT-004-ND)

1 - XBee Breakout Board (Sparkfun SKU#: BOB-08276 or from NKC Electronics)

2 - 2mm, 10 pin XBee socket (Sparkfun SKU#: PRT-08272 or included with NKC kit)

1 - Male Header row for XBee breakout (Sparkfun SKU#: PRT-00117 or included with NKC kit)

1 - FT232RL Breakout Board (Sparkfun SKU#: BOB-00718)

- male or female headers for the FT232RL Breakout board

2 – LEDs

1 – Reset Switch

Breadboard

Jumper wires

USB miniB cable

Arduino Diecimila

XBee Shield from NKC Electronics (http://www.nkcelectronics.com/freeduino-arduino-xbee-shield-kit.html)

A computer running Windows

3.3V Power Supply

A XBee ZNet 2.5 Module with Sparkfun's XBee

breakout board

Step 1: Constructing the circuit

Build the XBee Breakout board and plug one of the XBee's into it. Set up the circuit on the bread board using the diagram. Instead of using a reset switch, I just plug one end of a jumper wire into the XBee reset pin and leave the other end of that jumper hanging off the edge of the board; when I need to reset the XBee, I just momentarily touch the bare end of the jumper to

ground.

Step 2: Power-up the XBee and prep the software

After double checking your connections, use the USB-miniB cable to plug the FT232 Breakout

into a USB port on your computer. Both LEDs connected to the XBee should light up and stay on Download the X-CTU software from Digi and install it.

Step 3: Run X-CTU and connect to the XBee

Run the X-CTU software. You should see a screen like the one to the right. Single click on the USB COM port that the XBee is connected to. If you're not sure, you can click the "Test/Query" button to read each COM port to discover which one has the XBee.

Step 4: Update the firmware Click on the "Modem Configuration" tab and then click on the "Download new versions…" button to download all of the updated firm wares. After the firmware downloads have completed, click on the "Read" button. The window will display all of the current settings of the attached XBee. Select "ZNet 2.5 Router/End Device AT" as the firmware in the pull-down menu. Change the "Node Identifier" to something like "router1". to change the "PAN ID" to a unique id.

Check the box that says "Always update firmware" and click the "Write" button. The X-CTU software may ask you to push the reset button during the firmware upgrade process.

Step 5: Test the Xbee Read through this step before you do it since

once you enter command mode, to enter a new command within 5 seconds or the Xbee will exit command mode and you'll need to start this step again. Click on the terminal tab and type "+++" in

the window to enter command mode. The XBee should respond in a second or two with "OK". Type "ATVR" to check the firmware version on the XBee. This should match the firmware version upgraded to (1241 for

Router/End Device or 1041 for Coordinator). Type "ATID" to check the PAN Network ID that the XBee is using. It should respond with "123" or whatever unique PAN ID you set in step 4.

Type "ATNI" to check the Node Identifier. The XBee should respond with "router1" or whatever unique Node Identifier you set in step 4.

Type "ATCN" to exit command mode. The XBee will respond with "OK". If those commands

Quit the X-CTU program and disconnect the USB-miniB cable from the computer.

If any of those commands did not return the correct information, click back to the "Modem

Configuration" tab, change the desired settings, and click the "Write" button again.

Step 6: Creating the network

At this point, XBee updated with the "ZNet 2.5 Router" firmware plugged into the

XBee Shield on top of an Arduino and the other XBee updated to the "ZNet 2.5 Coordinator" firmware and connected to the FT232 breakout which is plugged into

your computer. The "Associate" LED on the Coordinator Xbee should be blinking once-per second.

Using the X-CTU terminal connected to the coordinator, type "+++", wait for the "OK" response, then type "ATND". The XBee will respond with the serial number, node identifier and other information about the router XBee connected to the Arduino.

Step 7: Sending Serial Data

When using the ZNet 2.5 firmware, routers or end devices will communicate with the coordinator by default. send serial data to the XBee router connected to the Arduino and it will be received by the coordinator where you can read it into your computer.

Connect the Arduino with the XBee Shield to the computer using a USB cable. The jumpers on the XBee shield should still be disconnected. Load the following sketch on to the Arduino.

```
/* Serial Test code
Sends "testing..." over the serial connection once per second
while blinking the LED on pin 13.
Adapted from the Soft Serial Demonstration code
*/
#define ledPin 13
byte pinState = 0;
void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}
void loop() {
Serial.println("testing...");
// toggle an LED just so you see the thing's alive.
toggle(13);
```

```
delay(1000);
}
void toggle(int pinNum) {
// set the LED pin using the pinState variable:
digitalWrite(pinNum, pinState);
// if pinState = 0, set it to 1, and vice versa:
pinState = !pinState;
}
```

Below are 2 codes used for transmission of data and reception from the zigbee device to the mySQL server :

TRANSMISSION:

```
from xbee import ZigBee
import serial
serial_p=serial.Serial(40,9600)
serial_p.open
tran=ZigBee(serial_p)

print "start_transmitter/coordinator/COM41/"
print "-----------------------------------------------------"
```

tran.tx(id='\x10',frame_id='B',dest_addr_long='\x00\x13\xa2\x00\x40\x64\xf2\x11',dest_addr='\xff\xff',broadcast_radius='\x00',options='\x00',data='Ahmed')

print tran.wait_read_frame()

RECEPTION:

for i in range(0,30): # method to receive frame but in hexadecimal

Y=hex(ord(serial_port2.read())) # method receive one byte only so we do for loop to receive

 # all bytes for example frame of 30 bytes could be more or less play on the for loop number of iterations

print Y

3.2.2 Transfer from the Zigbee raw format to the database.

Once stage 2 is completed, all received data is transferred to the database, mySQL database, through a script written in Python. This data is inserted into different data with all required parameters. The data is also validated during this stage, where any exceptions are raised to the technical team to address.

The following code illustrates the steps taken from the Zigbee raw file, which was received through stage 2, from the bus, and will be read to be reformated, ready to be inserted into the database.

The script assumes the following:

1. Zigbee file is called input2.txt.
2. The file has three parameters:
    a. Max Speed.

b. Distance traveled.
c. Consumption.

3. The file has a date and time, and device id.
4. The file has a fixed length field format, which is defined as a segment. Here, the segment is defined as a 23 character field. This is flexible enough to change the size of the segment when the device is upgraded or a different format is being chosed with a new version.

The flexibility of this approach allows us in the future to change the formatting of the file, and just update the field parameters with the right length and number of fields, field separation parameters. This happens whenever the hardware / device on the bus is being updated with firmware or upgraded, where no code changes will be required. Instead, just field parameters would be changed to accomodate for those changes.

The function returns a list with all the data read to the calling function to be further processed and prepared for database insertion.

The next part will take all the variables in the list and prepare it to be inserted into different database tables and fields.

```
def getDevData():
    f = open("c:\python27\input2.txt")
    s = []
    s = f.readline()
    print " Read : "+ s + " : Count:"+ str(len(s))
    segment = 23
    scounter=0
    ecounter=0
```

```
counter=0
data=[]
dataCounter=0
while counter<len(s):
        counter = counter +  segment
        print "L: "+ s[scounter:counter]+ " - " + str(scounter) + " e:"+str(counter)
        data.append(s[scounter:counter])
        scounter=counter
        ecounter=+segment
        dataCounter = dataCounter + 1
for f in data:
        print " Looping in Result : " + f
return data
```

3.2.3 Database stage

Once data arrives at the mySQL part its sorted using the following code.

This part will reformat the data, validate it, then insert it into the databse. As the database is designed in normalized form, the data has to be inserted in different tables at different stages. This approach has to be followed to maintain the integrity of the database, and to comply with the constraint rules as defined in the database. The database stores which Zigbee device is in which bus, the current bus round, and the driver in the current round in which bus.

A brief description of the database tables :

1. Buses:

   This tables stores all the data related to the buses, make, model and any related info or parameters. It also stores the Zigbee device id installed into that bus.

2. Drivers:

   This tables stores all the drivers in the GUC University, and any related details pertaining to each driver.

3. Rounds:

   This tables stores all related rounds during any day.

4. Bus Rounds:

   This tables stores all bus rounds for every day, with the current bus, for which round, and who is the driver driving that particular bus. This tables is used to resolve different relationships between the bus, driver and the round. When data is received from the bus Device ID, the only information is the device ID. By using this table, the system can resolve the correct bus used, in which round, and the current driver using the bus. This approach reduces the overhead of having to enter different information in the Zigbee device in the bus, and eleminates the need for any direct daily updates on the bus device.

5. Round Bus Drivers

   This tables stores all the daily round information. This information is our main source of data for all bus related data. The main three variables used: Max Speed, Distance, and fuel consumption are stored in this table.

   By creating a join between this table and the other relevant tables, the system is able to produce a detailed report about every aspect of the bus and the driver, with accurate information stamped with date and time. This inforamtion is used by the analytical reports to produce different reports and alerts to the management for the performance of each driver.

The above tables, with the exception of the Round bus drivers, are all populated by the user during the system setup. However, the Bus round tables is populated and updated on a daily basis, when a driver roster is available.

The script only inserts into the Round Bus Drivers table. In order to achieve this part, the script has to split the different records, and resolve any relations, resolve the correct driver and the correct bus to populate the data against. This process is performed through the current piece of code, which performs different reads from different tables with the key information related to the current Zigbee device id as received from the bus. The device id is the key field in this whole function and process.

The result of this part of the code, is a successful insertion of all the device data into the table, ready for any reports to run against the database for further reporting. This part has been testing for successful insertion, and the data was successfully committed to the database.

import MySQLdb as mdb

```
def getDevData():
    f = open("c:\python27\input2.txt")
    s = []
    s = f.readline()
    print " Read : "+ s + " : Count:"+ str(len(s))
    segment = 23
    scounter=0
    ecounter=0
    counter=0
    data=[]
    dataCounter=0
    while counter<len(s):
        counter = counter +  segment
        print "L: "+ s[scounter:counter]+ " - " + str(scounter) + " e:"+str(counter)
        data.append(s[scounter:counter])
        scounter=counter
        ecounter=+segment
        dataCounter = dataCounter + 1
    for f in data:
        print " Looping in Result : " + f
    return data
```

```
def getBusId(devId,cursor):
        try:
                print " rec dev id : " + devId
                cursor.execute('select bus_id from buses where trans_id=%s',(devId))
                busid = cursor.fetchone()
                print "Bus Id: " + str(busid[0])
        except mdb.Error, e:
                print "Error %d: %s" % (e.args[0], e.args[1])
        return int(busid[0])


def splitAndIns(cursor,devData,bus_id):
        cdevId = 4
        cdevTime  = 4
        cdevDate  = 8
        cdevVar   = 2
        cdevData  = 5
        print " SAIS"
        for f in devData:
                vdevId          = f[0:cdevId]
                vdevTime  = f[cdevId:cdevId+cdevTime]
                vdevDate  = f[cdevId+cdevTime:cdevId+cdevTime+cdevDate]
                vdevVar   = f[cdevId+cdevTime+cdevDate:cdevId+cdevTime+cdevDate+cdevVar]
                vdevData                                                                        =
f[cdevId+cdevTime+cdevDate+cdevVar:cdevId+cdevTime+cdevDate+cdevVar+cdevData]
```

```
            print " Data: id:" + vdevId + " time: " + vdevTime + " date:" + vdevDate+ " Var:"+
vdevVar+" vdevDate:" + vdevData
            if vdevVar == '01':
                  speed = vdevData
            elif vdevVar =='02':
                  distance = vdevData
            elif vdevVar =='03':
                  consumption = vdevData
            else:
                  print " Error in Data..."



      cursor.execute('INSERT INTO BUS_ROUNDS(bus_bus_id,speed,distance,consumption,
time) VALUES ("%d","%s","%s","%s","%s")' %(bus_id,speed,distance,consumption,
vdevTime+" "+vdevDate))


def getDevId(devData):
      devId = devData[0][0:4]
      print "New Device: " + devId
      return devId


if __name__ == "__main__":
      devData = getDevData()
      DevId = getDevId(devData)
      for f in devData:
```

```
            print " main: data: " + f[0:4]
    try:
            conn = mdb.connect('localhost','zigbee','zigbee','zigbee');
            cursor = conn.cursor()
            busId = getBusId(DevId,cursor);
            print " Ret Bus : " + str(busId)
            splitAndIns(cursor, devData,busId)
            conn.commit()
            cursor.close
            conn.close()
    except mdb.Error, e:
            print "Error %d: %s" % (e.args[0], e.args[1])
```

The Output is demonstrated in the database and a report is produced upon inquiry.

After comparing several database management systems the relational model was preferable due to :

- Data entry, updates and deletions will be efficient.
- Data retrieval, summarization and reporting will also be efficient.
- Since the database follows a well-formulated model, it behaves predictably.
- Since much of the information is stored in the database rather than in the application, the database is somewhat self-documenting.
- Changes to the database schema are easy to make.
- No redundancy in the database.
- No data flaw problems.
- Strict constraint rules eliminate any invlaid data storage.

- Database integrity is maintained, eliminating the possibility of any incorrect insertion of update of the data.

In the project design ,we have several related tables:

- Table zigbee_drivers

- **Table zigbee_buses**

```
CREATE DATABASE  IF NOT EXISTS `zigbee` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `zigbee`;
-- MySQL dump 10.13  Distrib 5.5.9, for Win32 (x86)
--
-- Host: localhost    Database: zigbee
-- ------------------------------------------------------
-- Server version  5.5.13
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
```

```
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

-- Table structure for table `buses`

DROP TABLE IF EXISTS `buses`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `buses` (

  `bus_id` int(11) NOT NULL AUTO_INCREMENT,

  `type` varchar(100) DEFAULT NULL,

  `year` varchar(20) DEFAULT NULL,

  `Model` varchar(100) DEFAULT NULL,

  `trans_id` varchar(50) NOT NULL,

  PRIMARY KEY (`bus_id`),

  UNIQUE KEY `UK` (`type`,`year`,`Model`)

) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

-- Dumping data for table `buses`

LOCK TABLES `buses` WRITE;

/*!40000 ALTER TABLE `buses` DISABLE KEYS */;

INSERT INTO `buses` VALUES
(17,'mini','2011','K10','1234'),(18,'shuttle','2011','K10','1235'),(19,'cruise','2009','S66','1236'),(20,'mini','2008','J8','1237'),(21,'sh
uttle','2011','J8','1238');

/*!40000 ALTER TABLE `buses` ENABLE KEYS */;

UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
```

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2011-06-12 1:39:24

- **Table Zigbee_rounds**

REATE DATABASE IF NOT EXISTS `zigbee` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `zigbee`;

-- MySQL dump 10.13 Distrib 5.5.9, for Win32 (x86)

-- Host: localhost   Database: zigbee

-- --

-- Server version 5.5.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8 */;

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;

/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

```
-- Table structure for table `rounds`

DROP TABLE IF EXISTS `rounds`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `rounds` (

  `round_id` int(11) NOT NULL AUTO_INCREMENT,

  `code` varchar(45) DEFAULT NULL,

  `description` varchar(450) DEFAULT NULL,

  `location` varchar(450) DEFAULT NULL,

  PRIMARY KEY (`round_id`)

) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

-- Dumping data for table `rounds`

LOCK TABLES `rounds` WRITE;

/*!40000 ALTER TABLE `rounds` DISABLE KEYS */;

INSERT INTO `rounds` VALUES
(1,'a1',NULL,'mohandeseen'),(2,'b2',NULL,'maadi'),(3,'c3',NULL,'tagamo'),(4,'d4',NULL,'zamalek'),(5,'e5',NULL,'say
edah');

/*!40000 ALTER TABLE `rounds` ENABLE KEYS */;

UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

- **Table Zigbee_bus_rounds**

```
CREATE DATABASE  IF NOT EXISTS `zigbee` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `zigbee`;

-- MySQL dump 10.13  Distrib 5.5.9, for Win32 (x86)

-- Host: localhost    Database: zigbee

-- ------------------------------------------------------

-- Server version  5.5.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8 */;

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;

/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

-- Table structure for table `bus_rounds`

DROP TABLE IF EXISTS `bus_rounds`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `bus_rounds` (

  `busround_id` int(11) NOT NULL AUTO_INCREMENT,
```

```
  `bus_bus_id` int(11) DEFAULT NULL,

  `round_round_id` int(11) DEFAULT NULL,

  `driver_driver_id` int(11) DEFAULT NULL,

  `speed` varchar(10) DEFAULT NULL,

  `distance` int(10) DEFAULT NULL,

  `consumption` int(20) DEFAULT NULL,

  `time` varchar(20) DEFAULT NULL,

  `comments` varchar(450) DEFAULT NULL,

  `cd` datetime DEFAULT NULL,

  `cu` varchar(45) DEFAULT NULL,

  PRIMARY KEY (`busround_id`),

  KEY `bus_bus_id` (`bus_bus_id`),

  KEY `round_round_id` (`round_round_id`),

  KEY `driver_driver_id` (`driver_driver_id`),

  CONSTRAINT `bus_bus_id` FOREIGN KEY (`bus_bus_id`) REFERENCES `buses` (`bus_id`) ON DELETE NO
ACTION ON UPDATE NO ACTION,

  CONSTRAINT `driver_driver_id` FOREIGN KEY (`driver_driver_id`) REFERENCES `drivers` (`driver_id`) ON
DELETE NO ACTION ON UPDATE NO ACTION,

  CONSTRAINT `round_round_id` FOREIGN KEY (`round_round_id`) REFERENCES `rounds` (`round_id`) ON
DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

-- Dumping data for table `bus_rounds`

LOCK TABLES `bus_rounds` WRITE;

/*!40000 ALTER TABLE `bus_rounds` DISABLE KEYS */;

INSERT INTO `bus_rounds` VALUES (20,17,NULL,NULL,'00120',11,132,'1335
11052011',NULL,NULL,NULL),(21,18,NULL,NULL,'00190',15,232,'1335 11052011',NULL,NULL,NULL);

/*!40000 ALTER TABLE `bus_rounds` ENABLE KEYS */;
```

```
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

- **Table Zigbee_round_bus_drivers**

```
CREATE DATABASE  IF NOT EXISTS `zigbee` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `zigbee`;
-- MySQL dump 10.13  Distrib 5.5.9, for Win32 (x86)
--
-- Host: localhost    Database: zigbee
-- ------------------------------------------------------
-- Server version       5.5.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
```

```
/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

-- Table structure for table `round_bus_drivers`

DROP TABLE IF EXISTS `round_bus_drivers`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `round_bus_drivers` (

  `roundbusdriver_id` int(11) NOT NULL AUTO_INCREMENT,

  `bus_bus_id` int(11) DEFAULT NULL,

  `driver_driver_id` int(11) DEFAULT NULL,

  `round_round_id` int(11) DEFAULT NULL,

  `run_time_start` datetime DEFAULT NULL,

  `run_time_end` datetime DEFAULT NULL,

  `comments` varchar(450) DEFAULT NULL,

  PRIMARY KEY (`roundbusdriver_id`)

) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

-- Dumping data for table `round_bus_drivers`

LOCK TABLES `round_bus_drivers` WRITE;

/*!40000 ALTER TABLE `round_bus_drivers` DISABLE KEYS */;

INSERT INTO `round_bus_drivers` VALUES (1,19,58,1,'2011-06-11 03:00:00','2011-06-11 04:00:00','on
time'),(2,20,59,2,'2011-06-11 05:00:00','2011-06-11 06:00:00','on time '),(3,17,60,3,'2011-06-11 07:00:00','2011-06-11
09:00:00','late');
```

```
/*!40000 ALTER TABLE `round_bus_drivers` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Chapter 4

Conclusion & Future work

As the GUC is a huge association owning a huge bus fleet that forms a remarkable transportation network serving in various locations same as any other department within the campus grounds the GUC fleet reflects the GUC image and quality of service.

And as the GUC has always proved of being first in class service deliverer in all aspects, when a flow was detected in the proficiency level of the transportation service a need for a monitoring system showed up.

After the data was stored during each round a wireless network was required to transfer data from each bus to a central server within campus grounds. The first obstacle was to transform data to its original format then sorting data into categories for further assessment on the server part ,finally storing all these records on the central data base and producing reports upon inquiry through a user friendly application to conduct a full evaluation report analyzing drivers performance .

For future monitoring system development :

- Notification system can be added to alert drivers whenever they exceed any of the monitored parameters formulated benchmark .

- Moreover, the wireless network could be extended to conduct a continuous automatic trace rather than a final data delivery at the end of each ride.

References :

[1] B.Goodyear, J.Jamil, C.Tulodzieski, "Home Monitoring System" ,pp. 14-19 ,July 2009.

[2] S.Arun ,"ZIGBEE" ,pp.5-15 ,August 2008.

[3] William stalling ,"wireless communication and networks", Fourth edition Pearson publication limited,20042.

[4] Andrew S. Tenenbaum, "Computer Networks", Fourth Edition Pearson Publication, Limited, 20033.

[5] Behrouz A. Frouzan, "Data Communication", Third Edition, Tata McGraw-Hill Publishing , company Limited, 20044.

[6] http://www.zigbee.org/en/documents/zigbeeoverview4.pdf