DE MONTFORT UNIVERSITY LEICESTER

FACULTY OF TECHONOLOGY

M.SC MECHATRONICS

MICROPROCESSOR APPLICATIONS AND SIGNAL PROCESSING

ENGT5203

MICROPROCESSOR APPLICATIONS COURSEWORK

LECTURE: DR. IAN SEXTON

STUDENT: GIDEON ANTHONY D’SOUZA

P-NUMBER: P13175646

DATED: 29/05/2014

Contents

Introduction:

This report deal with the construction and programing of a temperature, humidity and light sensor. This system deals with monitoring the temperature humidity and light intensity in different rooms or places in a building. The data collect by this system is required to be collected and projected on a screen and also recorded for future references.

This system consists of two integral parts, one being the transmitter section. This part of the device collects the required data. Each transmitter is called a node. They are placed in different places in the building to record the temperature at different places in the building. The transmitter node transmits the data to the second section of the system which is the Hub. The hub acts as the receiver. It has the ability to select a node to receive data from it and also delete the node to stop receiving data. It also records the data received and stores it in a SD card as a csv file or in a html file.

The transmitter circuit consists of a temperature humidity sensor which is a DTH 11, an LDR and a RF transmitter soldered to a microcontroller. The receiver circuit consist of a RF receiver connected to microcontroller.

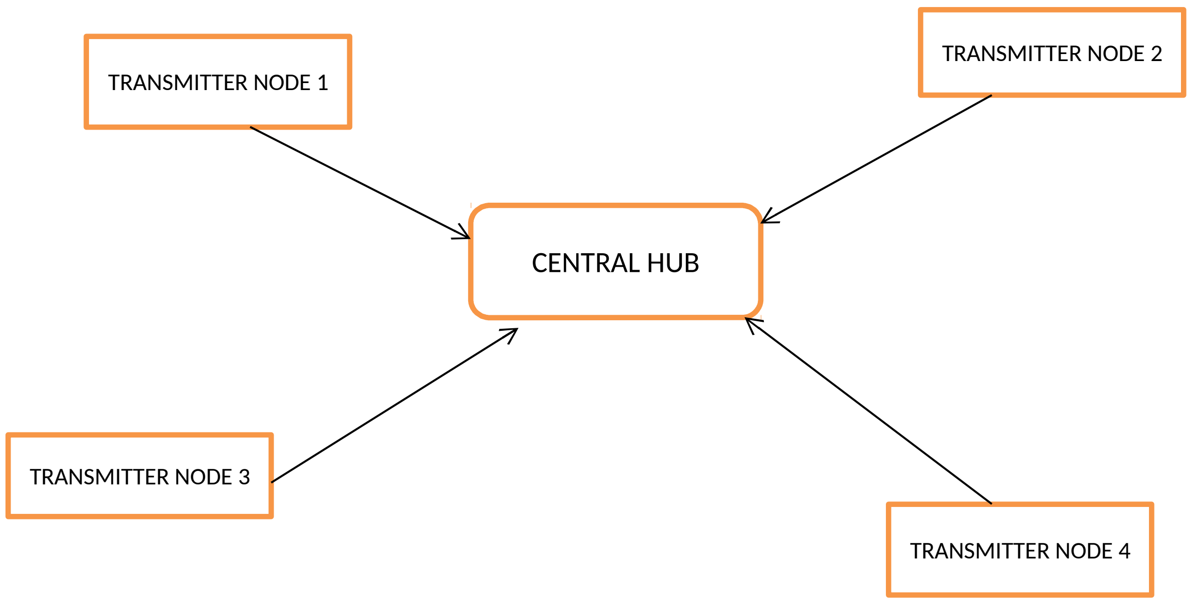


Figure 1 Schematic representation of the telemetry system

Aim:

To device and code a telemetry system which monitors temperature, humidity and light intensity.

Objective:

The objectives of this system are as follows.

- The transmitter should be able to read temperature humidity and light intensity levels in real time and transmit it to a central hub.
- The transmitter should sleep if it has been inactive for a certain amount of time.
- The central hub should be able to receive this data correctly.
- The central hub should have a menu system in which we can set time, set the system ID, add node, node name, delete node from system, dump data to a CSV file and dump data to a html file.

Methodology:

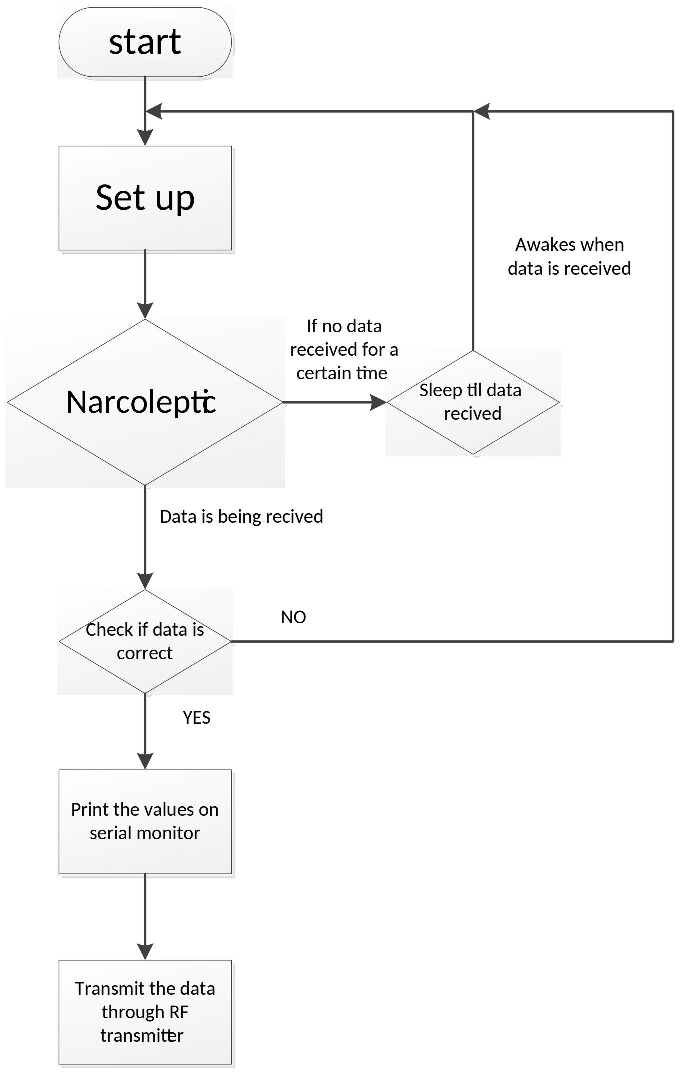
The methodology which was undertaken to achieve the above objectives is explained in this part. This system is split into two parts the transmission part and the receiver part. We shall analyse the working of this system by taking an in depth look in the working of these two parts.

Transmitter section:

This section consists of the sensors and the RF transmitter soldered to a microcontroller. The sensors used are DTH11 for temperate and humidity detection and an LDR for light intensity levels.

The command flow of the transmitter is shown below. A overall explanation is first give here and then we can take a deeper look.

From the flow chart we can see that when the program starts its first step is to set up all the required pins and variables required in the scope. It then moves into the loop in which it first checks if it receiving data if the transmitter is inactive for a given amount of time then the device goes into sleep mode. As the transmitter receives data it checks if the data is numerical if so it is displayed on the serial monitor if not then there is something wrong and a warning message will appear on the serial monitor. The next step is to transmit these values to the hub using a RF transmitter.



start
Set up
Narcoleptic
If no data received for a certain time
Sleep till data recived
Awakes when data is received
Data is being recived
Check if data is correct
NO
YES
Print the values on serial monitor
Transmit the data through RF transmitter

[Note: Snippets of the required code are used to explain each function. The entire code is in the appendixes of this report.]

Start:

When the program starts it firsts internalizes all required library's and variables

`#include "DHT.h"`
`#include <VirtualWire.h>`
`#include <Narcoleptic.h>`
`#define DHTPIN 2`
`#define DHTTYPE DHT11`
 	
`//DHT 11`
`#define VCC 13`
`#define Data 12`
`#define GND 10`
 	
`//TRANSMITTER`
`const int DATA_pin = 9;`
`const int VCC_pin = 8;`
`const int GND_pin = 7;`
 	
`//LDR`
`int LDR = A4;  //Pin for Photo resistor`
	• The library's used are defined and the DHT type is defined
	• It is defined as to where on the board the pins of the DHT sensor are connected.
	• It is defined as to where on the board the pins of the RF transmitter are connected.
	• It is defined as to where on the board the pins of the LDR sensors are connected.

int LDRValue = 0;
int Light_sensitivity = 255;
 	
DHT dht(Data, DHTTYPE);	

Set up:

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
 	
  pinMode(VCC,OUTPUT);
  pinMode(Data,OUTPUT);
  pinMode(GND,OUTPUT);
 	
  digitalWrite(VCC,HIGH);
  digitalWrite(GND,LOW);
 	
  pinMode(VCC_pin,OUTPUT);
  pinMode(DATA_pin,OUTPUT);
  pinMode(GND_pin,OUTPUT);
 	
  digitalWrite(VCC_pin,HIGH);
  digitalWrite(GND_pin,LOW);
 	
  pinMode(LDR,INPUT);
  // Initialise the IO and ISR
  vw_set_tx_pin(DATA_pin);
  vw_setup(2000);
 	
  dht.begin();
}
	• Here the first value set is thee baud rate which is set to 9600.
	• In the second step we set the pin mode for the DTH 11 sensor. Here VCC, data and GND are set to output.
	• Then we write a high value to the VCC which is a value ‘1’ normally +5v and a low value ‘0’ to the ground.
	• The pin modes for the RF transmitter are now set.
	• A digital high is set to the VCC_pin and a digital low is given to GND_pin.
	• Next the pin mode for the LDR is set.
	• DATA_pin in the RF transmitter section is set as the transmitter pin.
	• Then dht.begin() function is called from the DTH library.

The next step we enter the loop function. First step in that is

Narcoleptic:

Narcoleptic.delay(600000);	This command is used to put the device to sleep if it is inactive for 600000 milliseconds.

Check if data is correct and if so print:

In this step the system checks if the values obtained are numerical and not alphabets or characters. If the values are numerical from the sensor it prints it on the screen if not it gives an error message.

int h = dht.readHumidity();
int t = dht.readTemperature();
int light = analogRead(LDR);
 	
// check if returns are valid, if they are NaN (not a number) then something went wrong!
if (isnan(t) || isnan(h))
{
  Serial.println("Failed to read from DHT");
}
else {
	• At first the variables used to read the humidity, temperature and LDR readings are set.
	• We then check if the DHT is returning numerical values. If is not returning numerical values an error message is printed.
	• If it is numerical then the values are printed on the serial monitor.

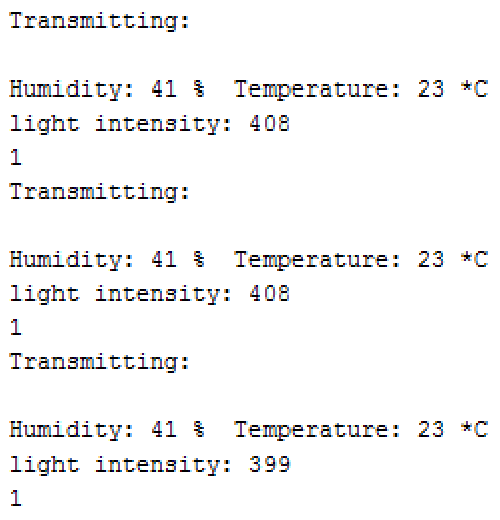
Serial.println("Transmitting: ");
delay(1000);
Serial.println("");
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.println(" *C");
Serial.print("light intensity: ");
Serial.println(light);
if(light > 200)
{
light = 1;Serial.println(light);
}
else
{
light = 0;Serial.println(light);
}
}
	• For the LDR it prints the value of the LDR and also prints one or zero if it is above or below a set value which is 200 over here. One indicates that the light is on and zero indicates that it is off.

Transmit to the hub:

After it's is all checked ad is printed on the screen the values are simultaneously transmitted using a RF transmitter in real time.

const int bytes = 4 * sizeof(int);
int name[bytes];
 	
name[0]=243;
name[1]=h;
name[2]=t;
name[3]=light;
 	
send((byte*)name,bytes);
delay(2000);
}
 	
void send(byte *name,int length)
{
vw_send(name,length);
vw_wait_tx();
 	
}
	• At first the variables used are initialised.
	• Then the buffers used to transmit the values are created.
	• Each buffer pertains to a particular value, first one gives the node name and the second for humidity 'h' third for temperature 't' and the last for light intensity 'light'.
	• A void send loop is used to transmit the values in buffer with the buffer name and its length.

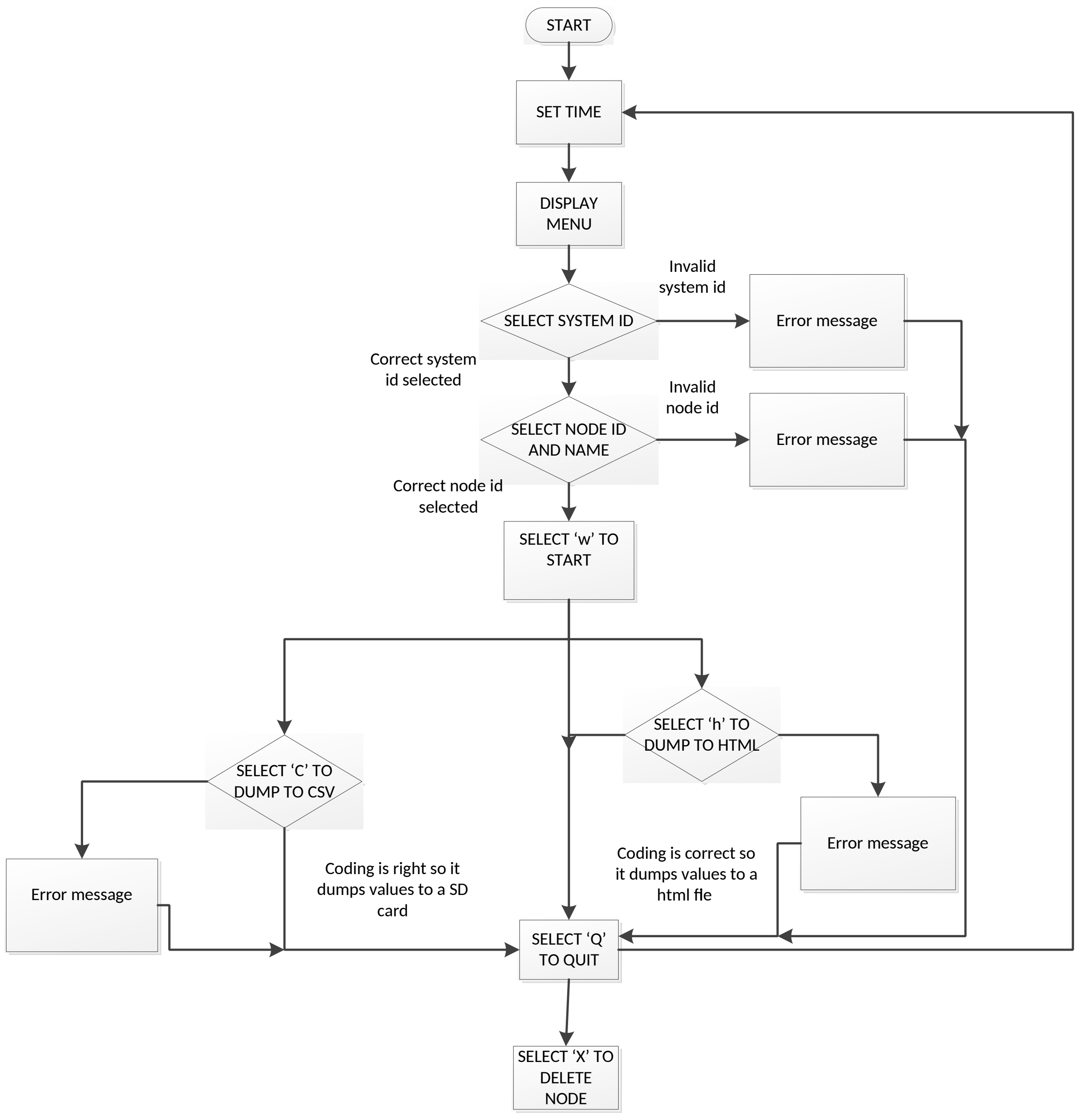
Outcome:



```
Transmitting:

Humidity: 41 %  Temperature: 23 *C
light intensity: 408
1
Transmitting:

Humidity: 41 %  Temperature: 23 *C
light intensity: 408
1
Transmitting:

Humidity: 41 %  Temperature: 23 *C
light intensity: 399
1
```

Receiver:

In this section we shall take a look at the receiver section of the system. The command flow is given as follows.



The process recommend to be followed is:

1. Start the program
2. Initialise the time by setting it to unix time.
3. Add the system id which is the hub id
4. Add the node to be selected with the node name by selecting <a><node id><node name>. example: a2bedroom

5. Select ‘w’ to start reading and displaying on the serial monitor.
6. To dump these values to a csv file select ‘c’.
7. To dump these values to an html file select ‘h’.
8. Select ‘q’ to quit from reading and displaying the values. When ‘q’ is selected it stops displaying the values and the menu is displayed again.
9. Select ‘x’ to delete the current node from the data base.
10. To add a new node now all we have to do is to repeat from step 4.

Now we shall take an in-depth look into the functioning:

Start:

When we start the receiver program we are required to add all required variables and libraries.

#include <Time.h>
#include <VirtualWire.h>
#include <SD.h>
 	
#define VCC 7
#define Data 6
#define GND 4
 	
#define TIME_HEADER "T"   // Header tag for serial time sync message
#define TIME_REQUEST 7   // ASCII bell character requests a time sync message
 	
int menu_item = 0;
int newNode[15];
int system_id;
 	
String room_names[15];
 	
const int chipSelect = 4;
	• First the required libraries are initialised
	• The pins for the RF receiver are defined next
	• The header tag and ASCII bell character are defined in order to set unix time for time.h library.
	• The required variables and string to be defined in the global scope is defined.

Set time:

This is used to initialise the device to unix time.

int settime()
{
  if (Serial.available())
  {
    processSyncMessage();
  }
  if (timeStatus()!= timeNotSet)
  {
    digitalClockDisplay();
  }
 	
  delay(1000);
	• The function which is called to set time is int settime() which is declared in the scope. To set time we should enter in the serial monitor <T><UNIX TIME> example: T1401415800
	• If there in an input then it calls the given functions which are also declared in the

```
  return 1;
}


void digitalClockDisplay(){
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits){
  // utility function for digital clock display: prints preceding
colon and leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}


void processSyncMessage() {
  unsigned long pctime;
  const unsigned long DEFAULT_TIME = 1357041600; // Jan
1 2013
  int daylight = 3600;

  if(Serial.find(TIME_HEADER)) {
    pctime = Serial.parseInt();
    pctime += daylight;
    if( pctime >= DEFAULT_TIME) { // check the integer is a
valid time (greater than Jan 1 2013)
      setTime(pctime); // Sync Arduino clock to the time
received on the serial port
    }
  }
}

time_t requestSync()
{
  Serial.write(TIME_REQUEST);
  return 0; // the time will be sent later in response to serial
mesg
}
```

scope.

- The first function declares the structure for a digital clock display for time.
- The second function is used to print the time in correct format.
- The next function is used to check if the input value is a valid time and it is also used to sync the Arduino clock to the time received.

Display menu:

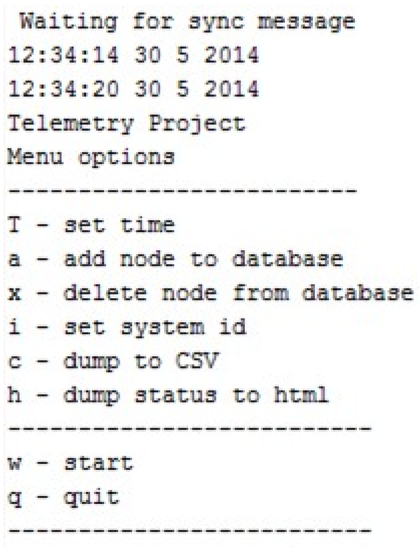
```
void displaymenu()
{
  digitalClockDisplay();
  Serial.println(F("Telemetry Project"));
  Serial.println(F("Menu options"));
  Serial.println(F("------------------------"));
  Serial.println(F("T - set time"));
  Serial.println(F("a - add node to database"));
```

- This menu is used to print the menu on the screen.
- Whenever the menu is to be displayed in the loop the displaymenu() function is called.

```
  Serial.println(F("x - delete node from database"));
  Serial.println(F("i - set system id"));
  Serial.println(F("c - dump to CSV"));
  Serial.println(F("h - dump status to html"));
  Serial.println(F("--------------------------"));
  Serial.println(F("w - start"));
  Serial.println(F("q - quit"));
  Serial.println(F("--------------------------"));
  return;
}
```

```
  if (Serial.available() > 0)     //ie a character has been received
  {
    char incomingByte = Serial.read();     // read the latest byte:
    switch (incomingByte) {
    case 'a':           // if user enters 'a' then add a tag
      addNode();
      break;
    case 'x':
      deleteNode();    // if user enters 'x' then erase database
      break;
    case 'i':          // if user enters 'i' then delete the last tag - superfluous?
      setSystemID();
      break;
    case'c':           // if user enters 'c' then print the EEPROM database to console
      dumpCSV();
      break;
    case'h':           // if user enters 'h' then set unix time (ten decimal digits)
      dumpHTML();
      break;
    case'T':           // if user enters 'T' then print the unix time
      settime();
      break;
    case 'w':
      Serial.println(F("string scan"));
      msgscan();
      Serial.println(F("stopped *****string scan"));
      displaymenu();
      break;
    }
```
	• This shows the case statement used for selection in the menu.
	• The case statement in present in the loop whereas each option functions is present outside the loop in the scope.
	• At first an if statement checks if any values have been entered.
	• It then cross-references the entered value with the values of each case by using serial read and obtaining the ASCII value and checking them with the option for each case.
	• When the required case is identified it enters that case and calls the function in it and processes only that case and returns when it is done.

Outcome:



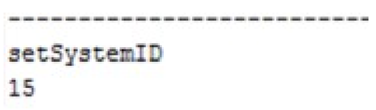
```
 Waiting for sync message
12:34:14 30 5 2014
12:34:20 30 5 2014
Telemetry Project
Menu options
-------------------------
T - set time
a - add node to database
x - delete node from database
i - set system id
c - dump to CSV
h - dump status to html
--------------------------
w - start
q - quit
--------------------------
```

Select system ID:

This is used to define the hub id.

int setSystemID()
{
  system_id= 15;
  Serial.println(F("setSystemID"));
  Serial.println(system_id);
  return 1;
}
	• This function is found in the scope it is called from the case statement in the loop by selecting‘s’.
	• The system Id is given as 15 and when selected it is displayed on the screen.

Outcome:



```
--------------------------
setSystemID
15
```

Add node id and node name:

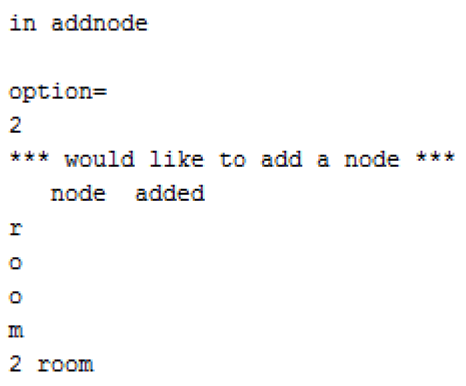
Here both the node id and the node name have been declared and done in a single function by selecting <a><node id><node name> example: a2bedroom

```
int addNode()
{
 Serial.println(F("in addnode "));
 delay(5000);
 boolean gotit = false;
 int option;
 while (!gotit)
 {
  if (Serial.available() > 0)
  {
   // read the incoming byte:
   option = Serial.parseInt();
   Serial.println(F(" "));
   Serial.println(F("option= "));
   Serial.println(option);
   delay(5000);
   if (option > 0 && option < 16)
   {
    gotit = true;
    newNode[option] = 1;
    Serial.println(F("*** would like to add a node ***"));
    //delay(500);

   }
   Serial.print(F("   node "));
   Serial.println(F(" added"));
   delay(500);

  }
  else
  {
   Serial.print(option);
   Serial.print(F("   ** out of range - enter a number
between 1 and 15 **"));
  }
 }
 // accept name of node
 String name;
 int finished = 0;
 int node_name = 0;
 char bytein;
 while(!finished)
 {

  if (Serial.available() > 0)
  {
   bytein = Serial.read();
   name += bytein;
   Serial.println(bytein);
  }
  else
```

- It first checks if it has got a value and then it reads the incoming byte.
- It checks if the input value is from 1 to 15. If so it adds the node and if not it gives an error message that it is out of range.
- To add node name it is entered along with the node id.
- The node name is first stored as an array which is of type string.
- Each character in the node name takes a place in the array.
- It is then added to the node id and corresponds to it and is displayed on the screen.

{
finished = 1;
}
}
 	
room_names[option] = name;
Serial.print(option);
Serial.print(" ");
Serial.println(room_names[option]);
 	
delay(2000);
}	

Outcome:



```
in addnode

option=
2
*** would like to add a node ***
   node  added
r
o
o
m
2 room
```

Select ‘w’ to start and q to quit:

To start printing the values we enter ‘w’ which is one of the cases in the loop. This intern calls a function which is declared in the scope outside the loop.

case 'w':
Serial.println(F("string scan"));
msgscan();
Serial.println(F("stopped *****string scan"));
displaymenu();
break;
	• This is the case for starting to print the values on the screen.
	• When it is selected it calls the function msgscan().

void msgscan()
{
uint8_t buflen = VW_MAX_MESSAGE_LEN;
uint8_t buf[buflen];
int exitnow = 0;
 	
while(!exitnow)
{
if (Serial.available() > 0)
{
if(Serial.read() == 'q')
exitnow = 1;
 	
}
 	
if (vw_get_message(buf, &buflen)) // check to see if anything has been received
	• This function first reads the incoming buffer messages in the receiver section.
	• It checks to see if anything is received.
	• It checks the buffer length and the values received from the buffer.
	• It then stores the received data in another buffer.
	• It splits the higher buffer values and lower buffer values as hid and nid

```
   {
     int i;
     // Message with a good checksum received.
     for (i = 0; i < buflen; i++)
     {
       Serial.print(i);
       Serial.print(buf[i]);
       // the received data is stored in buffer
     }
     Serial.println("");
   }

   int temp = buf[0];

   int nid = temp - 240;
   int hid = temp - nid;


   int err = 0 ;
   if(hid == 240)
   {
     if( newNode[nid] == 1)
     {
       Serial.print(F("System ID: "));
       Serial.println(hid,DEC);

       Serial.print(F("NODE ID: "));
       Serial.println(nid,DEC);
       Serial.print(F("Humidity: "));
       Serial.println(buf[2],DEC);

       Serial.print(F("Temperature: "));
       Serial.println(buf[4],DEC);
       Serial.println(F(""));

       Serial.println(F("light intensity: "));
       Serial.println(buf[6],DEC);
       Serial.println(F(""));
       delay(1000);
     }
     else
     {
       err = 1;
     }

   }

   else
   {
     err = 1;
   }
   if(err)
   {
     Serial.println("not recognized System ID:  ");

   }
  }
}
```

respectively to display the system id and node id.

- It then prints out all the values received from the buffer on to the screen.
- If the system does not recognise the node id or system id of if there is an error it would print not recognized system id.
- To stop displaying the values on the screen and to exit this function we are required to select ’q’.
- This would exit from the function and display the menu.

Outcome:

```
02431024130

System ID: 240

NODE ID: 3

Humidity: 41

Temperature: 201


light intensity:

108

02431024130

System ID: 240

NODE ID: 3

Humidity: 41

Temperature: 201


light intensity:

108
```

Dump to CSV:

This is used to log all the data received to a csv file. The option is selected by selecting ‘c’ in the case statement of the menu which in turn executes the dumpCSV() function.

```
int dumpCSV()
{
  // start sd copy
  Serial.print(F("Initializing SD card..."));
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:


  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println(F("Card failed, or not present"));
    // don't do anything more:

  }
  Serial.println(F("card initialized."));
  // make a string for assembling the data to log:
  String dataString = "";

  // read three sensors and append to the string:

  dataString += String("node");
```

- To dump to CSV we first store the data in a SD card.
- If there is no card present an error message will appear.
- The data is saved into a text file using string format.
- Each column is stored as an array in the format of a string.
- The file name is denoted in the program if the file is available it opens it and writes the data to it.

```
  dataString += ",";
  dataString += String("place");
  dataString += ",";
  dataString += String("humidity");
  dataString += ",";
  dataString += String("temp");
  dataString += ",";
  dataString += String("ligtht");
  dataString += ",";
  dataString += String("time");



  // open the file. note that only one file can be open at a
time,
  // so you have to close this one before opening another.
  File dataFile = SD.open("datalog.txt", FILE_WRITE);

  // if the file is available, write to it:
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
    // print to the serial port too:
    Serial.println(dataString);
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println(F("error opening datalog.txt"));
  }

  // end sd copy

  return 1;
}
```

- If the file isn’t open an error pops up.
- Once it is done copying it exits the function and displays the menu.

Delete node:

When a node is to be deleted we select x from the case statement. This in turn executes the deletNode() function.

```
int deleteNode()
{

  boolean gotit = false;
  int option;
  while (!gotit)
  {
    if (Serial.available() > 0)
    {
      // read the incoming byte:
      option = Serial.parseInt();
      Serial.println(F(" "));
      if (option > 0 && option < 16)
      {
        gotit = true;
        newNode[option] = 0;
        Serial.print(F("   node "));
        Serial.println(F(" deleted"));
        //delay(1000);
```

- It first checks if it has got a value and then it reads the incoming byte.
- It checks if the input value is from 1 to 15.
- If so it deletes the node and prints node deleted.
- If not it gives an error message that it is out of range.

```
     }
     else
     {
       Serial.print(option);
       Serial.print(F("   ** out of range - enter a number
between 1 and 15 **"));
     }
   }
 }
 return option;
}
```

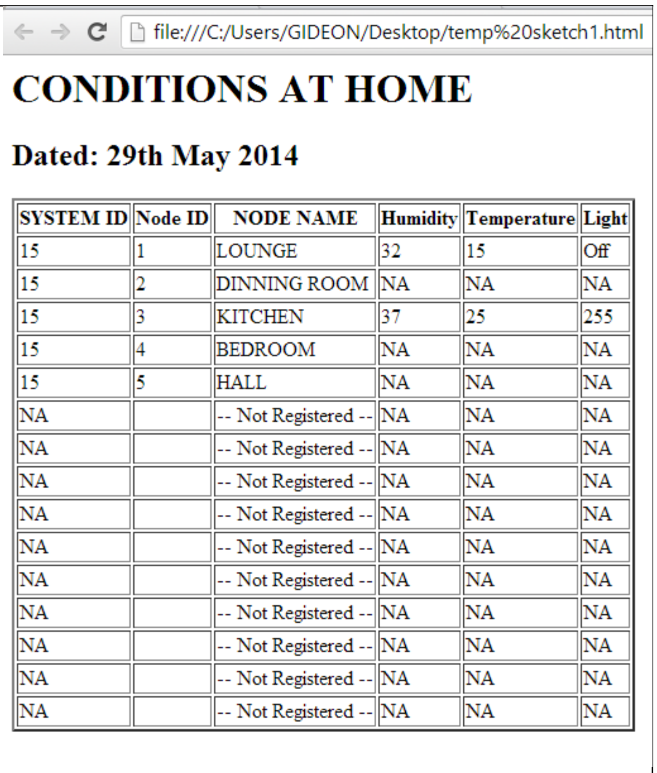
Outcome:

```
--------------------------

    node  deleted
```

Dump to html:

The structure of the html file is shown below this code is typed into a text file and saved with the extension .html.

```
<html>
<head>
<title>TEAPERATURE HUMIDITY AND LIGHT SENSORS</title>
                <body>
                <h1>CONDITIONS AT HOME</h1>
                <h2>Dated: 29th May 2014</h2>
                <table> border='2'>
                                <tr><th>SYSTEM ID<th>Node
ID<th>NODE
NAME<th>Humidity<th>Temperature<th>Light</tr>
                                <tr><td>15<td>1<td> LOUNGE <td>
32<td>15<td> Off</tr>
                                <tr><td>15 <td>2<td> DINNING ROOM
<td> NA<td> NA <td> NA</tr>



<tr><td>15<td>3<td>KITCHEN<td>37<td>25<td>255</tr>
                                <tr><td>15 <td>4<td> BEDROOM <td>
NA<td> NA <td> NA</tr>
                                <tr><td>15 <td>5<td> HALL <td>
NA<td> NA <td> NA</tr>
                                <tr><td> NA <td><td> -- Not Registered
-- <td> NA<td> NA <td> NA</tr>
                                <tr><td> NA<td><td> -- Not Registered
-- <td> NA<td> NA <td> NA</tr>
                                <tr><td> NA<td> <td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                                <tr><td> NA<td><td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                                <tr><td>NA <td><td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                                <tr><td>NA <td> <td> -- Not Registered
```



file:///C:/Users/GIDEON/Desktop/temp%20sketch1.html

CONDITIONS AT HOME

Dated: 29th May 2014

SYSTEM ID	Node ID	NODE NAME	Humidity	Temperature	Light
15	1	LOUNGE	32	15	Off
15	2	DINNING ROOM	NA	NA	NA
15	3	KITCHEN	37	25	255
15	4	BEDROOM	NA	NA	NA
15	5	HALL	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA
NA		-- Not Registered --	NA	NA	NA

```
-- <td> NA<td> NA<td> NA</tr>
                        <tr><td>NA <td><td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                        <tr><td> NA<td><td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                        <tr><td>NA <td> <td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                        <tr><td> NA<td> <td> -- Not Registered
-- <td> NA<td> NA<td> NA</tr>
                        </table>
            </body>
</html>
```

Conclusion:

Therefor at the end of this project we were able to fulfil the scope of at project and we were able to attain its objectives.

Appendices:

Transmitter code:

```
#include "DHT.h"
#include <VirtualWire.h>
#include <Narcoleptic.h>
#define DHTPIN 2     // what pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11   // DHT 11


//DHT 11
#define VCC 13
#define Data 12
#define GND 10


//TRANSMITTER
const int DATA_pin = 9;
const int VCC_pin = 8;
const int GND_pin = 7;

//LDR

int LDR = A4;  //Pin for Photo resistor
int LDRValue = 0;
int Light_sensitivity = 255;


DHT dht(Data, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");


  pinMode(VCC,OUTPUT);
  pinMode(Data,OUTPUT);
  pinMode(GND,OUTPUT);


  digitalWrite(VCC,HIGH);
  digitalWrite(GND,LOW);

  pinMode(VCC_pin,OUTPUT);
  pinMode(DATA_pin,OUTPUT);
  pinMode(GND_pin,OUTPUT);


  digitalWrite(VCC_pin,HIGH);
  digitalWrite(GND_pin,LOW);
  pinMode(LDR,INPUT);

  // Initialise the IO and ISR
  vw_set_tx_pin(DATA_pin);

  vw_setup(2000);


  dht.begin();
}

void loop()
{
   Narcoleptic.delay(600000); // During this time power consumption is minimised
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  int h = dht.readHumidity();
  int t = dht.readTemperature();
  int light = analogRead(LDR);



  // check if returns are valid, if they are NaN (not a number) then something went wrong!
  if (isnan(t) || isnan(h))
  {
    Serial.println("Failed to read from DHT");
  }
  else {
    Serial.println("Transmitting: ");
    delay(1000);
    Serial.println("");
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" *C");
    Serial.print("light intensity: ");
    Serial.println(light);
    if(light > 200)
    {
    light = 1;Serial.println(light);
    }
    else
    {
     light = 0;Serial.println(light);
    }
  }

 const int bytes = 4 * sizeof(int);
 int name[bytes];

  name[0]=243;
  name[1]=h;
  name[2]=t;
  name[3]=light;

 send((byte*)name,bytes);
 delay(2000);
}

void send(byte *name,int length)
{
  vw_send(name,length);
  vw_wait_tx();

}
```

Receiver code:

```
#include <Time.h>
#include <VirtualWire.h>
#include <SD.h>

#define VCC 7
#define Data 6
#define GND 4

#define TIME_HEADER  "T"   // Header tag for serial time sync message
#define TIME_REQUEST  7    // ASCII bell character requests a time sync message

int menu_item = 0;
int newNode[15];
int system_id;

String room_names[15];

const int chipSelect = 4;

void displaymenu()
{
  digitalClockDisplay();
  Serial.println(F("Telemetry Project"));
  Serial.println(F("Menu options"));
  Serial.println(F("-------------------------"));
  Serial.println(F("T - set time"));
  Serial.println(F("a - add node to database"));
  Serial.println(F("x - delete node from database"));
  Serial.println(F("i - set system id"));
  Serial.println(F("c - dump to CSV"));
  Serial.println(F("h - dump status to html"));
  Serial.println(F("--------------------------"));
  Serial.println(F("w - start"));
  Serial.println(F("q - quit"));
  Serial.println(F("--------------------------"));
  return;

}

int addNode()
{
  Serial.println(F("in addnode "));
  delay(5000);
  boolean gotit = false;
  int option;
  while (!gotit)
  {
    if (Serial.available() > 0)
    {
      // read the incoming byte:
      option = Serial.parseInt();
      Serial.println(F(" "));
      Serial.println(F("option= "));
      Serial.println(option);
      delay(5000);
      if (option > 0 && option < 16)
      {
        gotit = true;
        newNode[option] = 1;
        Serial.println(F("*** would like to add a node ***"));
        //delay(500);

      }
      Serial.print(F("   node "));
      Serial.println(F(" added"));
      delay(500);

    }
    else
    {
      Serial.print(option);
      Serial.print(F("   ** out of range - enter a number between 1 and 15 **"));
    }
  }
  // accept name of node
  String name;
  int finished = 0;
  int node_name = 0;
  char bytein;
  while(!finished)
  {

    if (Serial.available() > 0)
    {
      bytein = Serial.read();
      name += bytein;
      Serial.println(bytein);
    }
    else
    {
      finished = 1;
    }
  }

  room_names[option] = name;
  Serial.print(option);
  Serial.print(" ");
  Serial.println(room_names[option]);

  delay(2000);
}


int deleteNode()
{

  boolean gotit = false;
  int option;
  while (!gotit)
  {
    if (Serial.available() > 0)
    {
      // read the incoming byte:
      option = Serial.parseInt();
      Serial.println(F(" "));
      if (option > 0 && option < 16)
      {
        gotit = true;
        newNode[option] = 0;
        Serial.print(F("   node "));
        Serial.println(F(" deleted"));
        //delay(1000);

      }
      else
      {
        Serial.print(option);
        Serial.print(F("   ** out of range - enter a number between 1 and 15 **"));
      }
    }
  }
  return option;
}

int setSystemID()
{
  system_id= 15;
  Serial.println(F("setSystemID"));
  Serial.println(system_id);
  return 1;
}


int dumpCSV()
{
  // start sd copy
  Serial.print(F("Initializing SD card..."));
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:


  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println(F("Card failed, or not present"));
    // don't do anything more:

  }
  Serial.println(F("card initialized."));
  // make a string for assembling the data to log:
  String dataString = "";

  // read three sensors and append to the string:

  dataString += String("node");
  dataString += ",";
  dataString += String("place");
  dataString += ",";
  dataString += String("humidity");
  dataString += ",";
  dataString += String("temp");
  dataString += ",";
  dataString += String("ligtht");
  dataString += ",";
  dataString += String("time");
  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  File dataFile = SD.open("datalog.txt", FILE_WRITE);

  // if the file is available, write to it:
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
    // print to the serial port too:
    Serial.println(dataString);
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println(F("error opening datalog.txt"));
  }

  // end sd copy

  return 1;
}
int dumpHTML()
{
  File dataFile;
  dataFile.println(F("<html>"));
  dataFile.println(F("<head>"));
  dataFile.println(F("<title>TEAPERATURE HUMIDITY AND LIGHT SENSORS</title>"));
  dataFile.println(F("<body>"));
  dataFile.println(F("<h1>CONDITIONS AT HOME</h1>"));
  dataFile.println(F("<h2>Dated: 29th May 2014</h2>"));
  dataFile.println(F("<table border='2'>"));
  dataFile.println(F("<tr><th>SYSTEM ID<th>Node ID<th>NODE NAME<th>Humidity<th>Temperature<th>Light</tr>"));
  dataFile.println(F("<tr><td>15<td>1<td> LOUNGE <td> 32<td>15<td> Off</tr>"));
  dataFile.println(F("<tr><td>15 <td>2<td> DINNING ROOM <td> NA<td> NA <td> NA</tr>"));
  dataFile.println(F("<tr><td>15<td>3<td>KITCHEN<td>37<td>25<td>255</tr>"));
  dataFile.println(F("<tr><td>15 <td>4<td> BEDROOM <td> NA<td> NA <td> NA</tr>"));
  dataFile.println(F("<tr><td>15 <td>5<td> HALL <td> NA<td> NA <td> NA</tr>"));
  dataFile.println(F("<tr><td> NA <td><td> -- Not Registered -- <td> NA<td> NA <td> NA</tr>"));
  dataFile.println(F("<tr><td> NA<td><td> -- Not Registered -- <td> NA<td> NA <td> NA</tr>"));
  dataFile.println(F("<tr><td> NA<td> <td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td> NA<td><td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td>NA <td><td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td>NA <td> <td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td>NA <td><td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td> NA<td><td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td>NA <td> <td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("<tr><td> NA<td> <td> -- Not Registered -- <td> NA<td> NA<td> NA</tr>"));
  dataFile.println(F("</table>"));
  dataFile.println(F("</body>"));
  dataFile.println(F("</html>"));

  Serial.println(F("dumpHTML"));
  return 1;
}

int settime()
{
  if (Serial.available())
  {
    processSyncMessage();
  }
  if (timeStatus()!= timeNotSet)
  {
    digitalClockDisplay();
  }

  delay(1000);
  return 1;
}

void digitalClockDisplay(){
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits){
  // utility function for digital clock display: prints preceding colon and leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}


void processSyncMessage() {
  unsigned long pctime;
  const unsigned long DEFAULT_TIME = 1357041600; // Jan 1 2013
  int daylight = 3600;

  if(Serial.find(TIME_HEADER)) {
    pctime = Serial.parseInt();
    pctime += daylight;
    if( pctime >= DEFAULT_TIME) { // check the integer is a valid time (greater than Jan 1 2013)
      setTime(pctime); // Sync Arduino clock to the time received on the serial port
    }
  }
}

time_t requestSync()
{
  Serial.write(TIME_REQUEST);
  return 0; // the time will be sent later in response to serial mesg
}


void setup()
{
  Serial.begin(9600);                // The baudrate of the serial monitor is 9600

    setSyncProvider( requestSync);  //set function to call when sync required
  Serial.println("Waiting for sync message");
  delay(5000);
  settime();
  delay(5000);

  pinMode(VCC,OUTPUT);
  pinMode(GND,OUTPUT);
  digitalWrite(VCC,HIGH);
  digitalWrite(GND,LOW);

  vw_setup(2000);    // Bits per sec
  vw_set_rx_pin(Data);
  vw_rx_start();
  pinMode(10, OUTPUT);

  do
  {

    displaymenu();
    menu_item = 0;
  }
  while(menu_item>1);


}

void loop()
{

  if (Serial.available() > 0)     //ie a character has been received
  {
    char incomingByte = Serial.read();     // read the latest byte:
    switch (incomingByte) {
    case 'a':           // if user enters 'a' then add a tag
      addNode();
      break;
    case 'x':
      deleteNode();    // if user enters 'x' then erase database
      break;
    case 'i':          // if user enters 'i' then delete the last tag - superfluous?
      setSystemID();
      break;
    case'c':           // if user enters 'c' then print the EEPROM database to console
      dumpCSV();
      break;
    case'h':           // if user enters 'h' then set unix time (ten decimal digits)
      dumpHTML();
      break;
    case'T':           // if user enters 'T' then print the unix time
      settime();
      break;
    case 'w':
      Serial.println(F("string scan"));
      msgscan();
      Serial.println(F("stopped *****string scan"));
      displaymenu();
      break;
    }
  }
}

void msgscan()
{
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  uint8_t buf[buflen];
  int exitnow = 0;

  while(!exitnow)
  {
```

```
if (Serial.available() > 0)
{
  if(Serial.read() == 'q')
    exitnow = 1;

}

if (vw_get_message(buf, &buflen)) // check to see if anything has been received
{
  int i;
  // Message with a good checksum received.
  for (i = 0; i < buflen; i++)
  {
    Serial.print(i);
    Serial.print(buf[i]);
    // the received data is stored in buffer
  }
  Serial.println("");
}

int temp = buf[0];

int nid = temp - 240;
int hid = temp - nid;


int err = 0 ;
if(hid == 240)
{
  if( newNode[nid] == 1)
  {
    Serial.print(F("System ID: "));
    Serial.println(hid,DEC);

    Serial.print(F("NODE ID: "));
    Serial.println(nid,DEC);
    Serial.print(F("Humidity: "));
    Serial.println(buf[2],DEC);

    Serial.print(F("Temperature: "));
    Serial.println(buf[4],DEC);
    Serial.println(F(""));

    Serial.println(F("light intensity: "));
    Serial.println(buf[6],DEC);
    Serial.println(F(""));
    delay(1000);
  }
  else
  {
    err = 1;
  }

}

else
{
  err = 1;
}
if(err)
{
  Serial.println("not recognized System ID:  ");
```

```
    }
  }
}
```