Data Availability Report

Introduction:

This data availability statement provide here is an explanation of everything used or needed for this research work, it goes from the pseudo code to the few sniped code of our implementation, we have organized it as follow: section 1, is all about our pseudo code; in section 2, we have enumerate the tools used in our work; section 3 show the installations process of those different tools used in our research ; section 4 shows the different sensors used for Structural Health Monitoring we used; and finally section 5, display few sniped codes of our implementation.

Pseudo code of Implementation

a) Algorithm for compression at sensor node level.

```
Procedure Compressdata
    ThresholdValue ← maximum tolerated value during a natural disaster assigned to sensors →
    "if a senses value exceeds this value then we must take that into consideration"
    waitValue ← amount of time to wait before sending the compressed value

    While TRUE do
        SensedData ← sensing the environment
        If sensedData is greater than ThresholdValue then

            collectedData ← sensedData → "add the sensed data to the collection of
            data that are greater than the threshold"

        end If
        If startSensingTime is greater or equal to waitValue
            compressedData ← ArithmeticEncoding(collectedData)
            CompressdataSink (compressedData)
        end If
    end while

end Procedure
```

A threshold value is to be set which is to be considered while sensing the environment; so, all the values bellow that value should be ignore by the sensors. Then, we set a ***waitValue*** too, which is the amount of time a sensor has to wait before transmitting data to the sink. At this time, we start sensing data; if the value sensed (***SensedData***) is greater than the threshold value; (***ThresholdValue***), we collect that value and store it in the variable containing the collection of values above the threshold value, and, we do check if the time the sensors is intended to wait before transmitting the data has elapsed, if so, we pass that collection of values as argument to the ***ArithmeticEncoding*** method, and it return the compressed data (***compressedData***) which in turn is sent to the sink node by the use of ***CompressdataSink*** method that take the ***compressedData*** as argument.

b) Algorithm description of compression at sink node level

```
Procedure CompressdataSink(compressedDataFromSensors)

    sinkWaitingTimeValue ← amount of time to wait before sending the compressed value to the terminal

    collectSensorsCompressedData ← compressedDataFromSensors collect the data compressed by sensors nodes
    If startCollectingTime is greater or equal to sinkWaitingTimeValue
        compressedSinkData ← ArithmeticEncoding(collectSensorsCompressedData)
        sendToTerminal(compressedSinkData)
    end If

end Procedure
```

Here we start by setting a waiting time reference value (*sinkWaitingTimeValue*), which is the amount of time the sink need to wait before compressing the received value from the sink; then we store the data from sensors in a variable called *collectSensorsCompressedData*; then, we check if the time the sink is intended to wait before sending the data to the terminal has elapsed; if so, we pass that collection of value receive from the sensors to the Arithmetic Encoding in order to compress the data again, and then, we send that double compressed data to the terminal, by using *sendToTerminal* method with *compressedSinkData* as argument.

A. Tools:

- Eclipse IDE
- TinyOs plugins
- Nesc(programming language)

B. Installation:

Processes of installation of different tools used:

- **Eclipse IDE:**

  Source: “https://websiteforstudens.com/how-to-install-eclipse-oxygen-ide-on-ubuntu-167-04-17-10-18-04/”

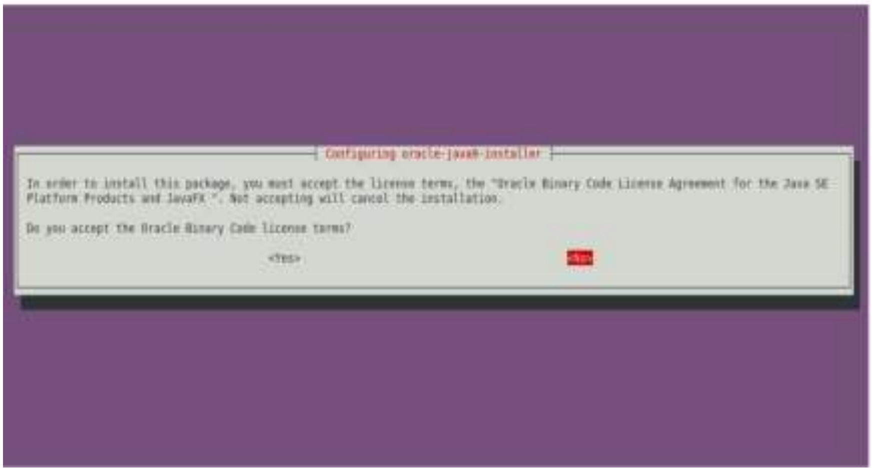
  1. Install java: to do so,
     a. Run the command bellow:

sudo add-apt-repository ppa:webupd8team/java

After running the above command, you should see a prompt to accept the PPA key onto Ubuntu. So accept and continue.
Now rung the command bellow to download java 9 installer. This installer should install the latest Java JDK 9 on your Ubuntu machines.

sudo apt update
sudo apt install oracle-java8-installer

After running the above commands, you will be ask to access the license terms of the software. So accept and continue.



Configuring oracle-java8-installer

In order to install this package, you must accept the license terms, the "Oracle Binary Code License Agreement for the Java SE Platform Products and JavaFX ". Not accepting will cancel the installation.

Do you accept the Oracle Binary Code license terms?

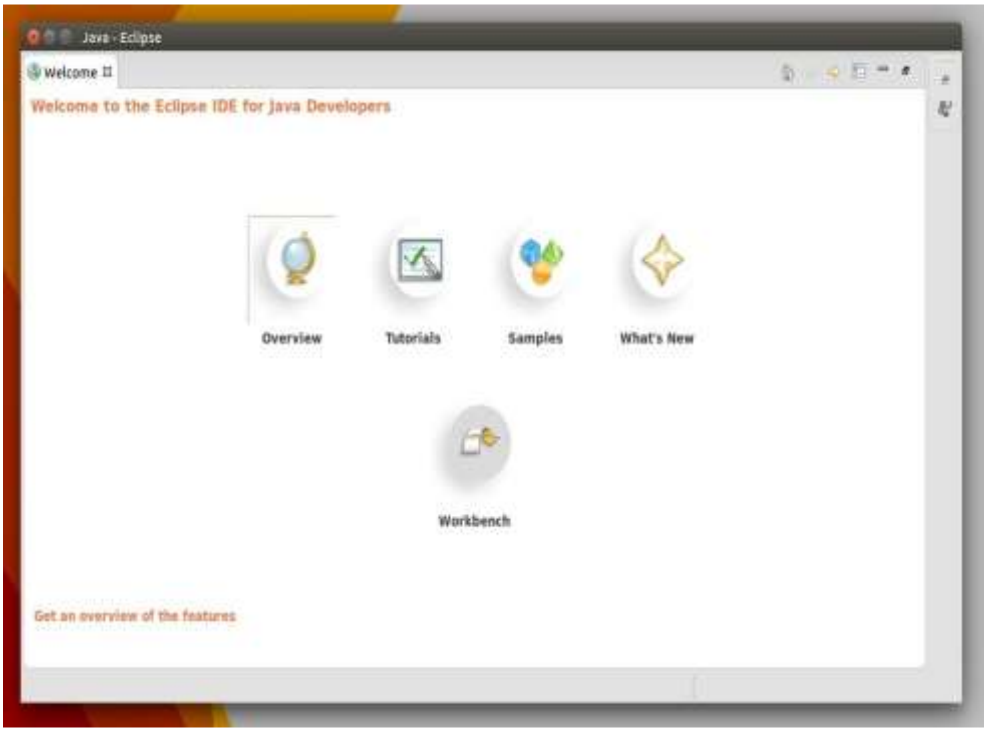
Set Oracle JDK8 as default, to do that, install the oracle-java8-set-default package. This will automatically set the JAVA environment Variable.

sudo apt- install oracle-java8-set-default

The above commands will automatically set java 9 as the default, and that should complete your installation, you can check your java version by running the following commands.

javac -version

b. You can also go on the software center of Ubuntu to search and install OpenJDK java 7 or 8.



Java - Eclipse

Welcome

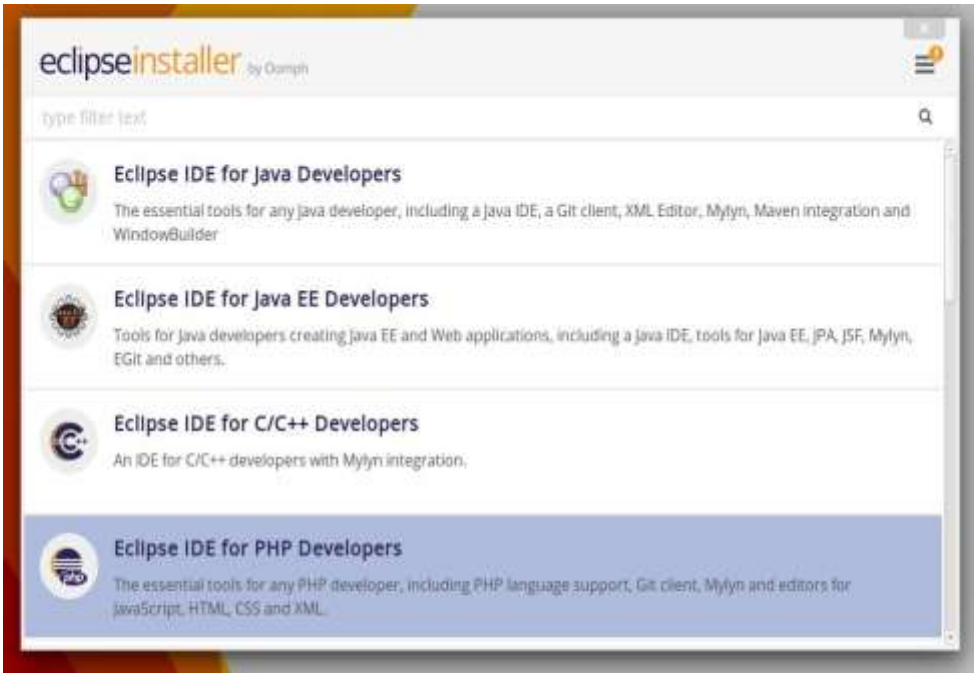
Welcome to the Eclipse IDE for Java Developers

Get an overview of the features

2. Download Eclipse from the official website according to your OS type whether it is a 32 or 64 bit version. (Go to Setting->Details->Overview).

3. Run the installer wizard: decompress the downloaded archive in your file browser and navigate to the result “eclipse-installer” folder, right-click on file eclipse-inst and select Run or go to file browser’s Menu Edit->Preferences->Behavior-> check “Run executable text files when they are opened”, and finally log out and back in (or run nautilus –q command on the terminal).

   When the wizard launches, select install item, then installation folder, and finally click INSTALL button.



Eclipse IDE for Java Developers

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder

Eclipse IDE for Java EE Developers

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

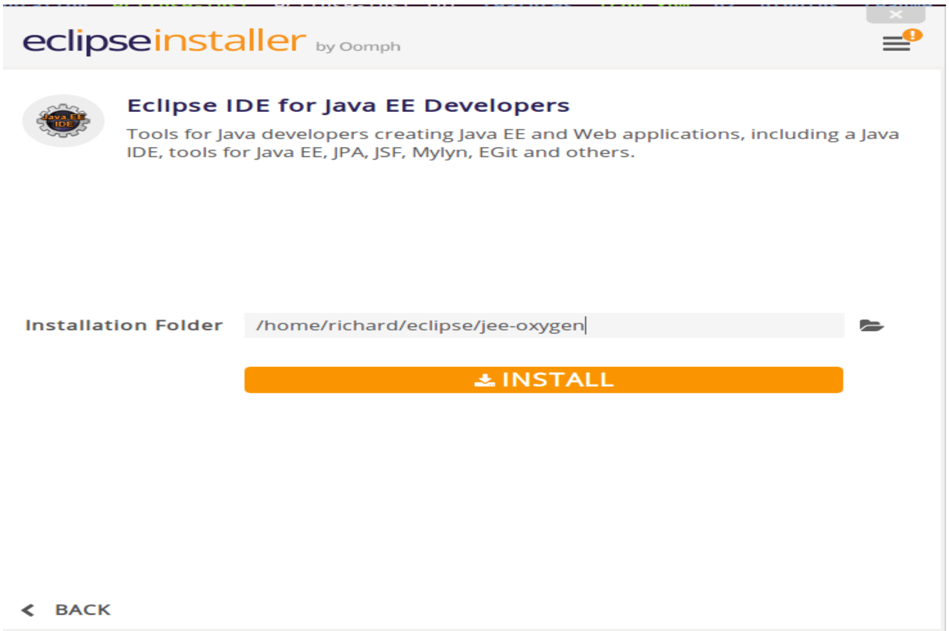
Eclipse IDE for C/C++ Developers

An IDE for C/C++ developers with Mylyn integration.

Eclipse IDE for PHP Developers

The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Use the onscreen instructions to complete the installer. (Accept the default installation directory and continue)

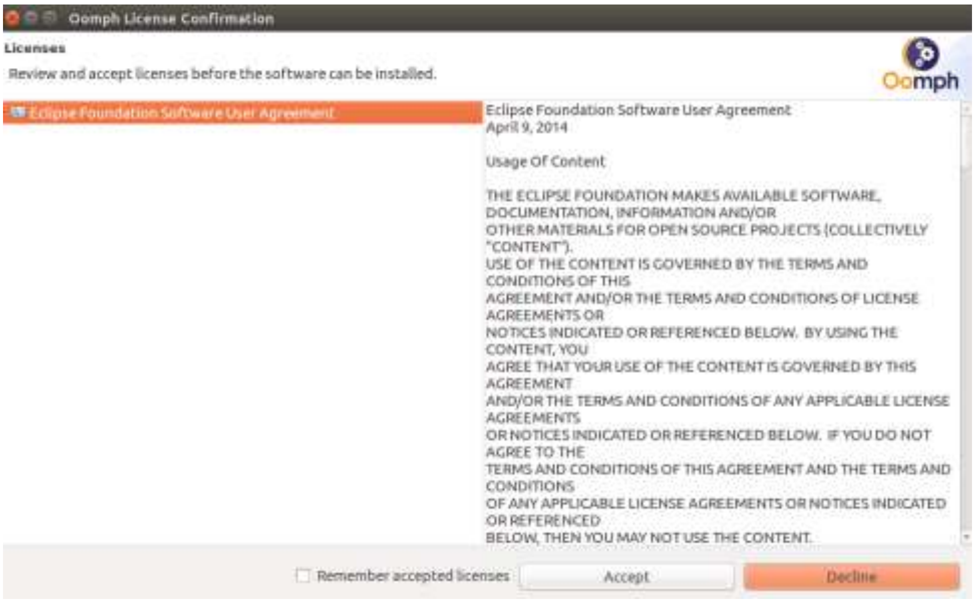


Eclipse IDE for Java EE Developers

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

Installation Folder /home/richard/eclipse/jee-oxygen

Next, accept the license terms and continue. (Wait for Eclipse installer to download and install all the packages.)


Oomph License Confirmation
Licenses
Oomph
Eclipse Foundation Software User Agreement
Accept
Decline

Review and accept licenses before the software can be installed.

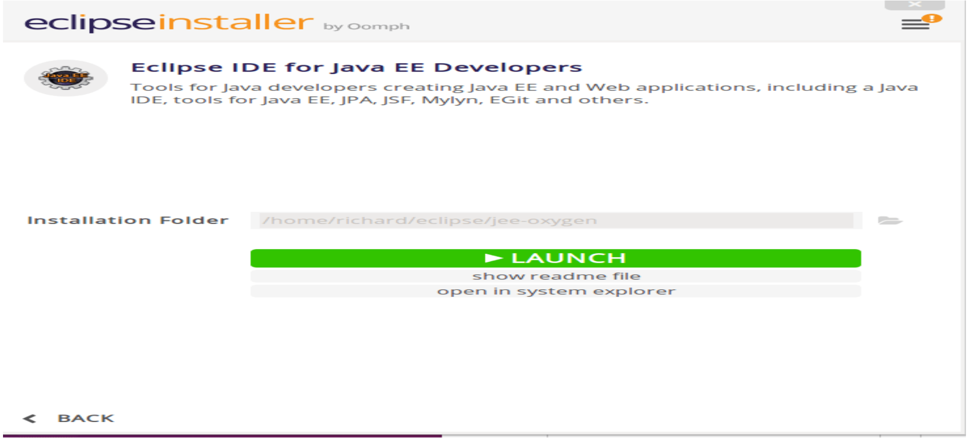
Eclipse Foundation Software User Agreement
April 9, 2014

Usage Of Content

THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW, THEN YOU MAY NOT USE THE CONTENT.

☐ Remember accepted licenses

After downloading the installer should complete. (All you have to do is launch the program.)



Eclipse IDE for Java EE Developers

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

Installation Folder /home/richard/eclipse/jee-oxygen

4. Create a launcher shortcut: open terminal via Ctrl+Alt+T shortcut key, then type the following:

```
gedit .local/share/application/eclipse.desktop
```

The command creates and opens a launcher file, for current user, with gedit text editor. When it opens, type the following

```
Name=Eclipse
Type=Application
Exec=/home/USERNAME/java-mars/eclipse/eclipse
Terminal=false
Icon=/home/USERNAME/java-mars/eclipse/icon.xpm
Comment=Integrated Development Environment
```

```
NoDisplay=false
Categories=Development; IDE;
Name[en]=Eclipse
```

Depending on your installation folder, check out in file browser, change the value in bold, and USERNAME is your personal directory.

- **TinyOS:**

Source: www.tinyprod.net/repos/debian/

For the TinyOS 2.1.2 release, you should install tnyos-tools-14 (tinyos-tools). TinyOS releases prior to 2.1.2 are not supported.

Tinyos-tools and tinyos-tools-14 are incompatible with tinyos-tools-devel. You can only have one of these packages installed. You must remove tinyos-tools (tinyos-tools-14) prior to installing tinyos-tools-devel and vice-versa.

```
$ sudo

$ apt-get purge tinyos-tools

$ apt-get install tinyos-tools-devel
```

You may need to remove old *-tinyos packages. To do so, follow these instructions:

- Remove previously install *-tinyos packages:

  ```
  $ sudo dpkg –P `dpkg –l nesc ‘*tinyos*’ | grep ^ii | awk ‘{ print $2}’ | xargs`
  $ sudo apt-get clean
  ```

  To use these packages, follow these instructions:

  a) Tell apt about the TinyProd signing key.

  ```
  $ weg –O – http:/tinyprod.net/repos/debian/tinyprod.key | sudo apt-key add
  ```

  Or..

  ```
  $gpg –keyserver keyserver.ubuntu.com –recv-keys A9B913B9
  $gpg –a –export A9B913B9 | sudo apt-key add –
  ```

  b) Add the following lines to /etc/apt/sources.list.d/tinyprod-debian.list:

  ```
  deb http://tinyprod.net/repos/debian wheezy main

  deb http://tinyprod.net/repos/debian msp430-46 main

  $ sudo -s

  $ cd /etc/apt/souces.list.d

  $ echo “deb http://tinyprod.net/repos/debian wheezy main” >> tinyprod-debian.list

  $ echo “deb http://tinyprod.net/repos/debian msp430-46 main” >> tinyprod-debian.list
  ```

c) Install the new packages:

```
$ sudo apt-get update
$sudo apt-get install nesc tinyos-tools msp430-46 avr-tinyos.
```

NOTE: if you ever wanted to uninstall the package, run the following commands:

```
$ sudo apt-get autoremove –purge nesc tinyos-tools msp430-46 avr-tinyos
```

C. Type of sensors used:

Based on what to measure different sensors are available:

a. Strain Gauges (sometime referred to as Stain gage): is a sensor whose resistance varies with applied force; it converts force, pressure tension, weight, etc. into change in electrical resistance which can then be measured. When external forces are applied to stationary object, stress and stain are the result.

b. Seismometers: is an instrument that measures motion of the ground, caused by, for example, an earthquake, a volcanic eruption, or the use of explosives. Record of seismic waves allow seismologist to map the interior of the earth and to locate and measure the size of events like these.

c. Piezoelectric Accelerometers: is an accelerometer that employs the piezoelectric effect of certain materials to measure dynamic changes in mechanical variables (e.g, acceleration, vibration, and mechanical shock).

d. Velocity Transducers/Sensor: consist of a moving coil suspended in the magnetic field of a permanent magnet. The velocity is given as the input, which causes the movement of the coil in the magnetic field. This causes an emf to be generated in the coil. This induced emf will be proportional to the input velocity and thus, is a measure of the velocity.

e. Laser Displacement Sensors (LDS): the principle of "laser displacement sensor" ranging is a method where triangulation is applied by combining the emitting element and the position sensitive device (PSD) to perform ranging (detecting the amount of displacement).

f. Fiber Optic Sensors: is a sensor that uses optical fiber either as the sensing element ("intrinsic sensors"), or as a means of relaying signals from a remote sensor to the electronics that process the signals ("extrinsic sensors").

g. Piezoelectric Sensors: is a device that uses the piezoelectric effect, to measure changes in pressure, acceleration, temperature, strain, or force by converting them to an electrical charge.

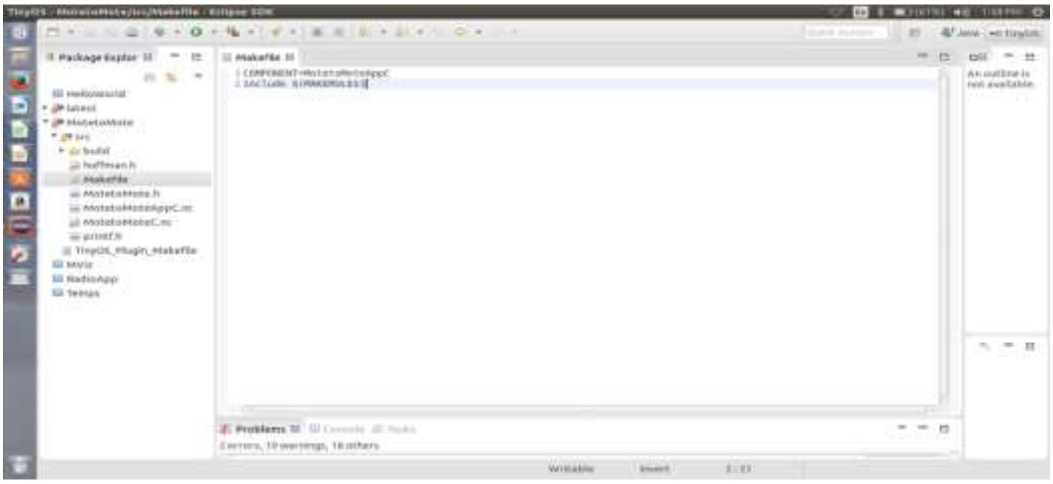
h. Accelerometer sensors: is an electromechanical device used to measure acceleration forces. Such forces may be static, like the continuous force of gravity or, as is the case with many mobile devices, dynamic to sense movement or vibrations. Acceleration is the measurement of the change in velocity, or speed divided by time.

D. Implementation

The implementation was done as follow.

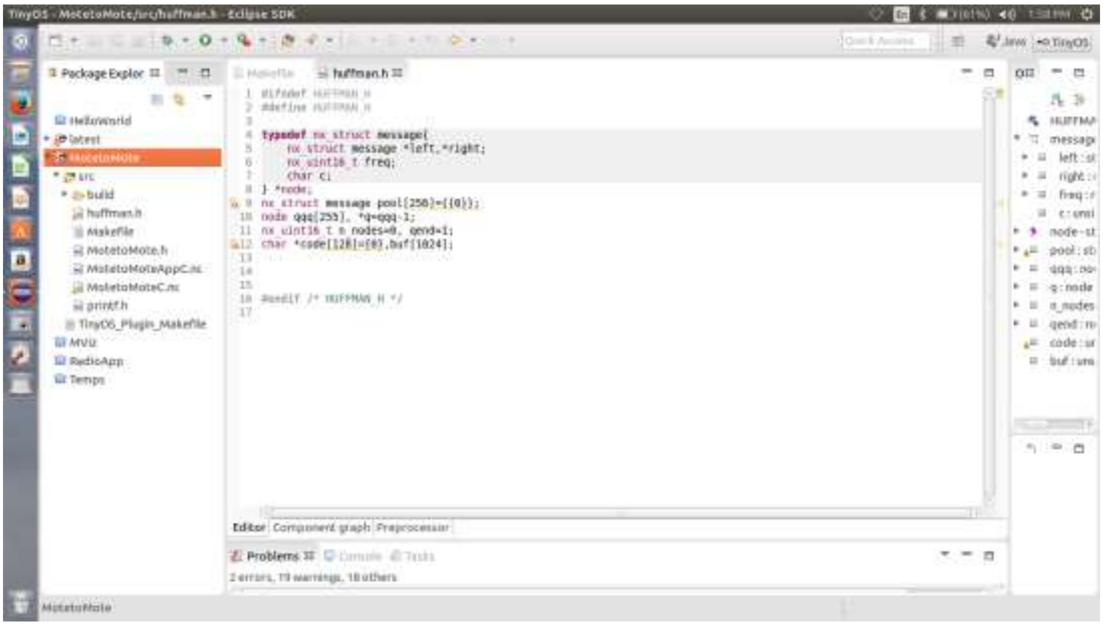
- The make file: which consisted only of the rule that should be applied to the other file.
- A Header file: where the main structure of our code was written.
- The Module file: was the file that contains all the interfaces as well as most of the source code
- The Configuration file: this file contains the code necessary to wired all the components of our program together.

C. Source Code snipped



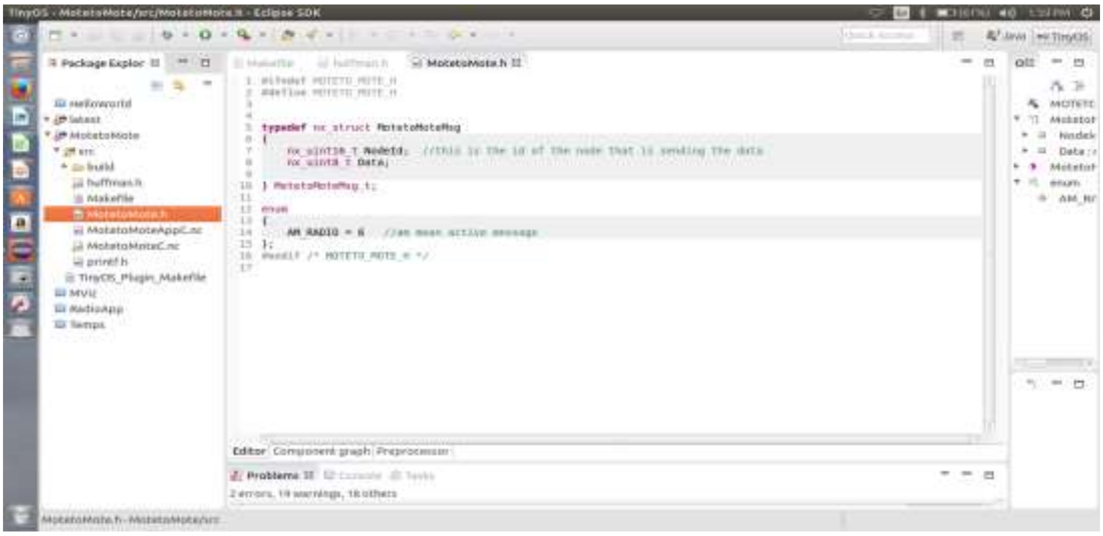
```
COMPONENT=MotetoMoteAppC
include $(MAKERULES)
```

Fig1: figure of the make file source code



```
#ifndef HUFFMAN_H
#define HUFFMAN_H

typedef nx_struct message{
     nx_struct message *left,*right;
     nx_uint16_t freq;
     char c;
} *node;
nx_struct message pool[256]={{0}};
node qqq[255], *q=qqq-1;
nx_uint16_t n_nodes=0, qend=1;
char *code[128]={0},buf[1024];



#endif /* HUFFMAN_H */
```

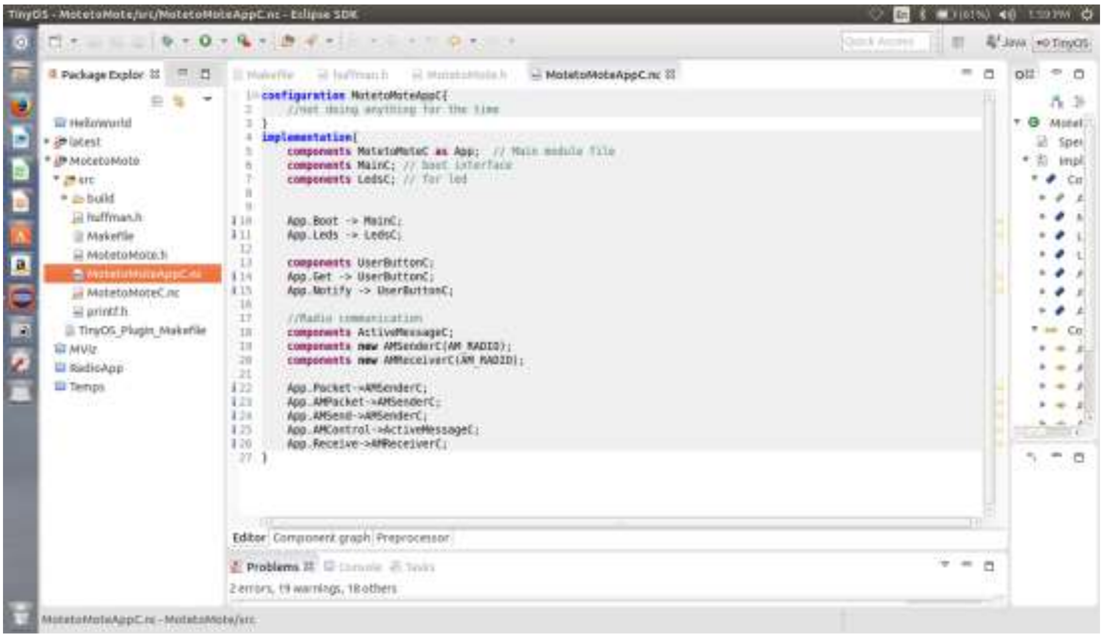
Fig2: snipped source code of the Huffman main structure



```
#ifndef MOTETO_MOTE_H
#define MOTETO_MOTE_H


typedef nx_struct MotetoMoteMsg
{
    nx_uint16_t NodeId;  //this is the id of the node that is sending the data
    nx_uint8_t Data;

} MotetoMoteMsg_t;

enum
{
    AM_RADIO = 6   //am mean active message
};
#endif /* MOTETO_MOTE_H */
```

2 errors, 19 warnings, 18 others

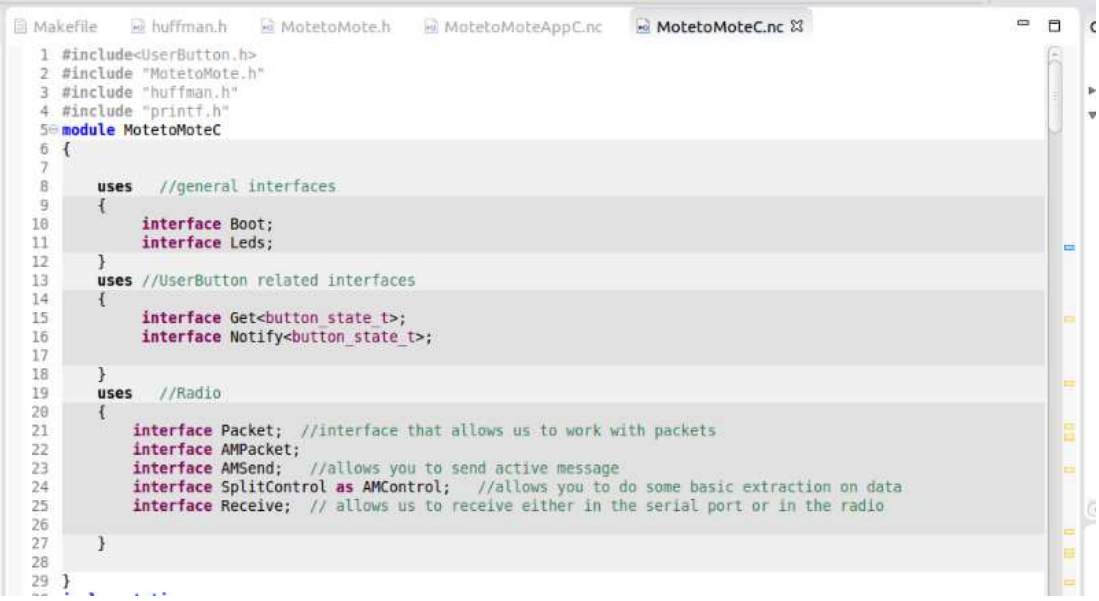
Fig3: snipped source code of the packet structure



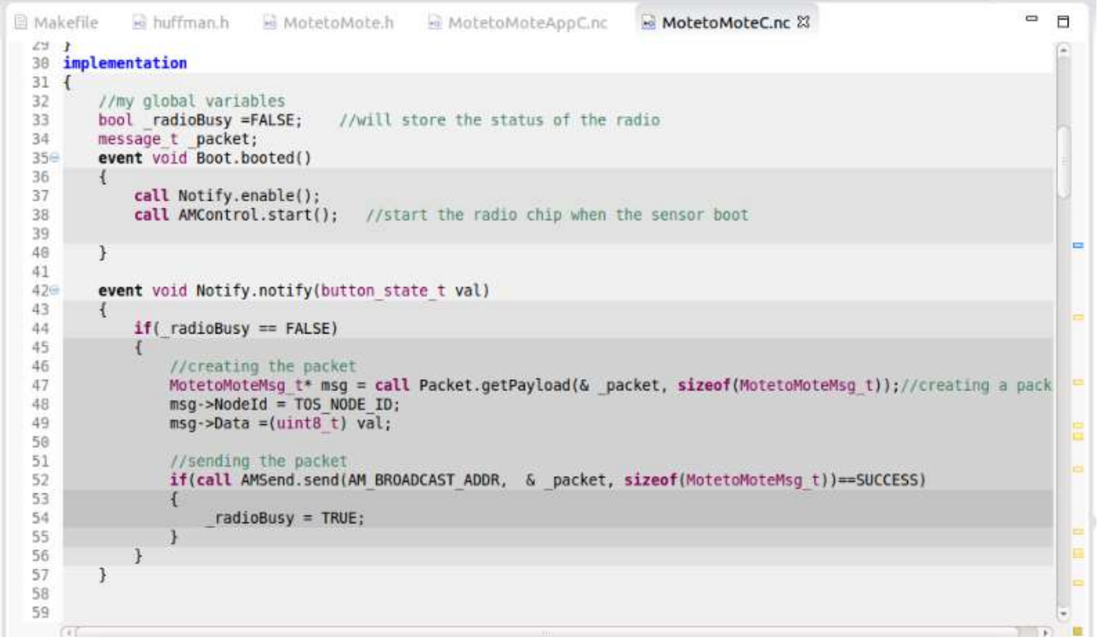
```
configuration MotetoMoteAppC{
    //not doing anything for the time
}
implementation{
    components MotetoMoteC as App;  // Main module file
    components MainC; // boot interface
    components LedsC; // for led


    App.Boot -> MainC;
    App.Leds -> LedsC;

    components UserButtonC;
    App.Get -> UserButtonC;
    App.Notify -> UserButtonC;

    //Radio communication
    components ActiveMessageC;
    components new AMSenderC(AM_RADIO);
    components new AMReceiverC(AM_RADIO);

    App.Packet->AMSenderC;
    App.AMPacket->AMSenderC;
    App.AMSend->AMSenderC;
    App.AMControl->ActiveMessageC;
    App.Receive->AMReceiverC;
}
```

2 errors, 19 warnings, 18 others

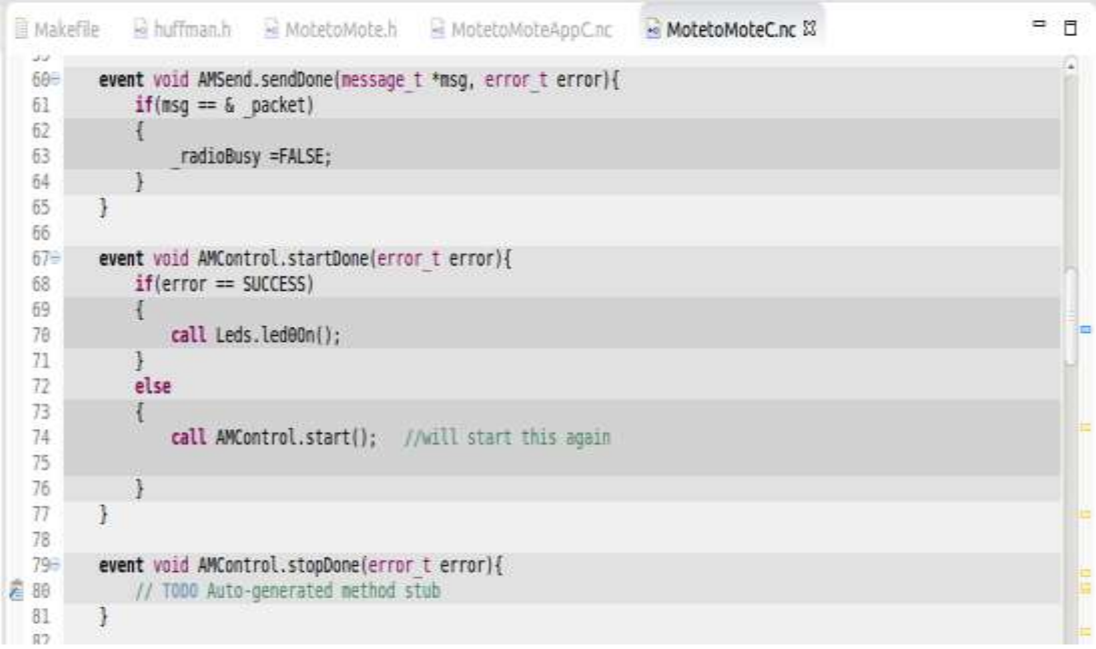
Fig3: snipped source code of the configuration file



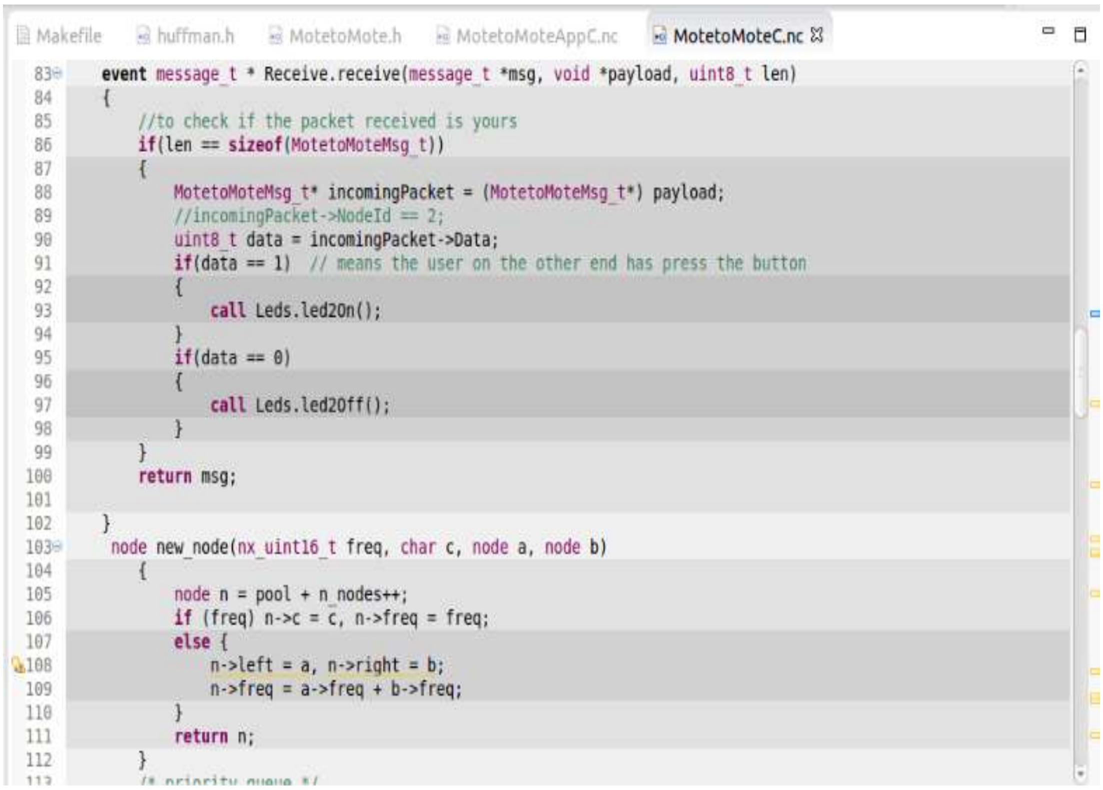
```
#include<UserButton.h>
#include "MotetoMote.h"
#include "huffman.h"
#include "printf.h"
module MotetoMoteC
{

    uses   //general interfaces
    {
         interface Boot;
         interface Leds;
    }
    uses //UserButton related interfaces
    {
         interface Get<button_state_t>;
         interface Notify<button_state_t>;

    }
    uses   //Radio
    {
        interface Packet;  //interface that allows us to work with packets
        interface AMPacket;
        interface AMSend;   //allows you to send active message
        interface SplitControl as AMControl;   //allows you to do some basic extraction on data
        interface Receive;  // allows us to receive either in the serial port or in the radio

    }

}
```



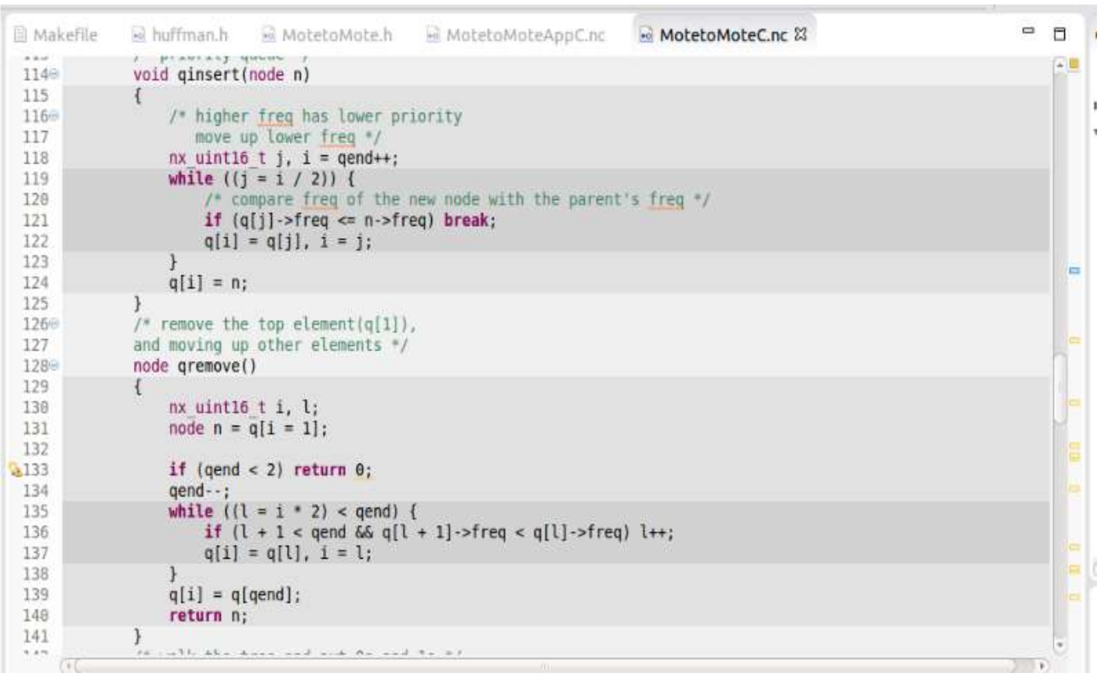
```
}
implementation
{
    //my global variables
    bool _radioBusy =FALSE;    //will store the status of the radio
    message_t _packet;
    event void Boot.booted()
    {
        call Notify.enable();
        call AMControl.start();   //start the radio chip when the sensor boot

    }

    event void Notify.notify(button_state_t val)
    {
        if(_radioBusy == FALSE)
        {
            //creating the packet
            MotetoMoteMsg_t* msg = call Packet.getPayload(& _packet, sizeof(MotetoMoteMsg_t));//creating a pack
            msg->NodeId = TOS_NODE_ID;
            msg->Data =(uint8_t) val;

            //sending the packet
            if(call AMSend.send(AM_BROADCAST_ADDR,  & _packet, sizeof(MotetoMoteMsg_t))==SUCCESS)
            {
                _radioBusy = TRUE;
            }
        }
    }


```



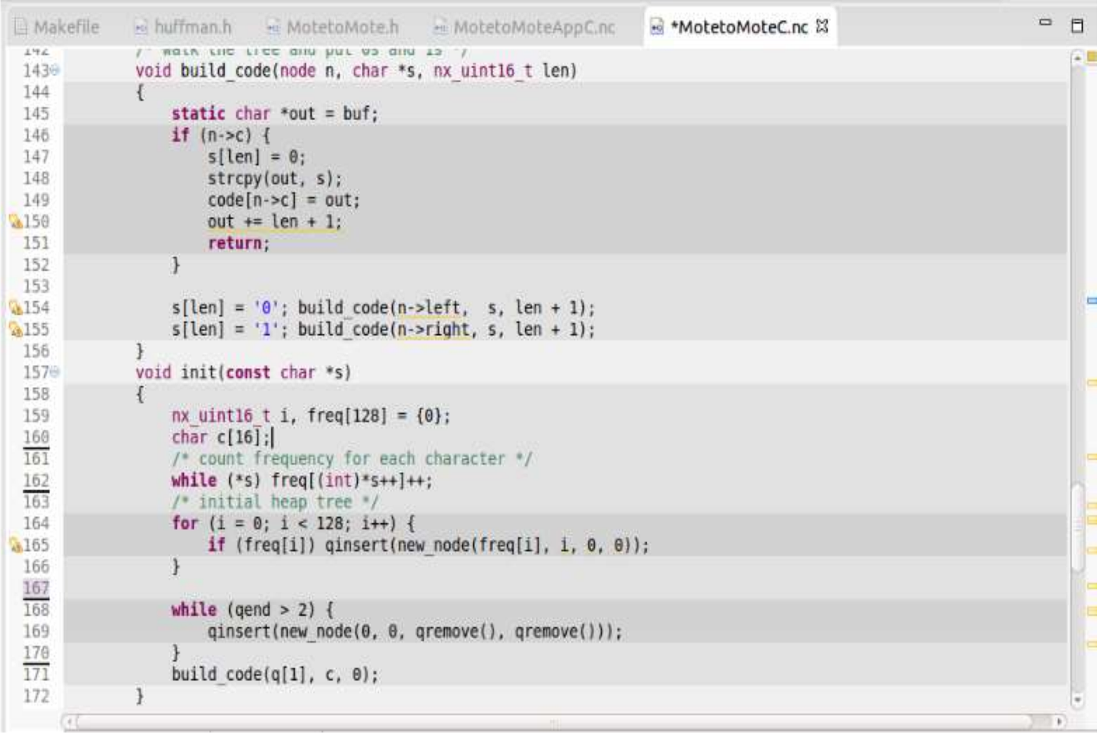
```
    event void AMSend.sendDone(message_t *msg, error_t error){
        if(msg == &_packet)
        {
            _radioBusy =FALSE;
        }
    }

    event void AMControl.startDone(error_t error){
        if(error == SUCCESS)
        {
            call Leds.led0On();
        }
        else
        {
            call AMControl.start();   //will start this again

        }
    }

    event void AMControl.stopDone(error_t error){
        // TODO Auto-generated method stub
    }
```



```
    event message_t * Receive.receive(message_t *msg, void *payload, uint8_t len)
    {
        //to check if the packet received is yours
        if(len == sizeof(MotetoMoteMsg_t))
        {
            MotetoMoteMsg_t* incomingPacket = (MotetoMoteMsg_t*) payload;
            //incomingPacket->NodeId == 2;
            uint8_t data = incomingPacket->Data;
            if(data == 1)  // means the user on the other end has press the button
            {
                call Leds.led2On();
            }
            if(data == 0)
            {
                call Leds.led2Off();
            }
        }
        return msg;

    }
     node new_node(nx_uint16_t freq, char c, node a, node b)
        {
            node n = pool + n_nodes++;
            if (freq) n->c = c, n->freq = freq;
            else {
                n->left = a, n->right = b;
                n->freq = a->freq + b->freq;
            }
            return n;
        }
        /* priority queue */
```



```
/* priority queue */
void qinsert(node n)
{
    /* higher freq has lower priority
       move up lower freq */
    nx_uint16_t j, i = qend++;
    while ((j = i / 2)) {
        /* compare freq of the new node with the parent's freq */
        if (q[j]->freq <= n->freq) break;
        q[i] = q[j], i = j;
    }
    q[i] = n;
}
/* remove the top element(q[1]),
and moving up other elements */
node qremove()
{
    nx_uint16_t i, l;
    node n = q[i = 1];

    if (qend < 2) return 0;
    qend--;
    while ((l = i * 2) < qend) {
        if (l + 1 < qend && q[l + 1]->freq < q[l]->freq) l++;
        q[i] = q[l], i = l;
    }
    q[i] = q[qend];
    return n;
}
/* walk the tree and put 0s and 1s */
```



```
/* walk the tree and put 0s and 1s */
void build_code(node n, char *s, nx_uint16_t len)
{
    static char *out = buf;
    if (n->c) {
        s[len] = 0;
        strcpy(out, s);
        code[n->c] = out;
        out += len + 1;
        return;
    }

    s[len] = '0'; build_code(n->left,  s, len + 1);
    s[len] = '1'; build_code(n->right, s, len + 1);
}
void init(const char *s)
{
    nx_uint16_t i, freq[128] = {0};
    char c[16];
    /* count frequency for each character */
    while (*s) freq[(int)*s++]++;
    /* initial heap tree */
    for (i = 0; i < 128; i++) {
        if (freq[i]) qinsert(new_node(freq[i], i, 0, 0));
    }

    while (qend > 2) {
        qinsert(new_node(0, 0, qremove(), qremove()));
    }
    build_code(q[1], c, 0);
}
```

Fig4: snipped code of the module file

Conclusion:

Within this document about our resources, we have given: the pseudo codes of our implementation follow by a clear explanations; then we have enumerate the tools used and then clearly explain how to install eclipse IDE (Integrated Development Environment), TinyOS on Ubuntu; after that we gave various type of sensors needed with their definitions and finally we gave some sniped code of our implementation. After extrapolation we can assure the readers that using these resources it is clearly possible for them to replicate our work with a laborious efforts.