

Getting Started

for

Wireless Sensor Networks Integrated Development Environment

Release 1.0

Version 0.1

Prepared by Poonguzhali .P & Mahesh U.Patil

Centre for Development of Advanced Computing, Hyderabad

27/04/2009

Revision History

| Name | Date | Changes | Version |
|--------------------------------|--------------------------------|--|----------------|
| Poonguzhali P & Mahesh U.Patil | 26 th November 2009 | Added Packet Structure Information | 0.1 |
| Poonguzhali P & Mahesh U.Patil | 29 th November 2009 | New Snapshots for adding target to Project build | 0.1 |

WSN Application Development using WSNIDE

A Getting Started Guide

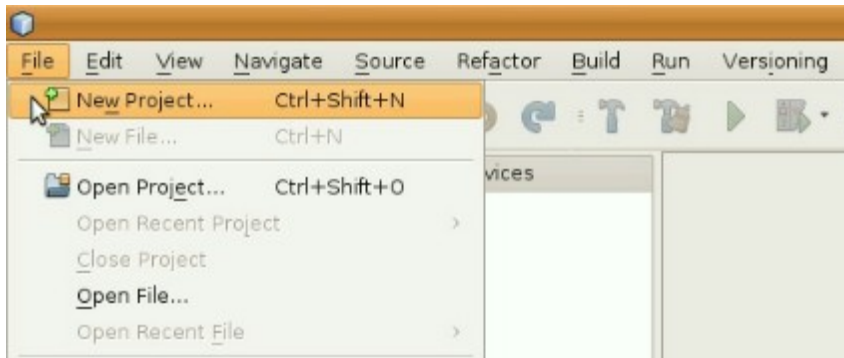
Pre-requisite:

Required Software

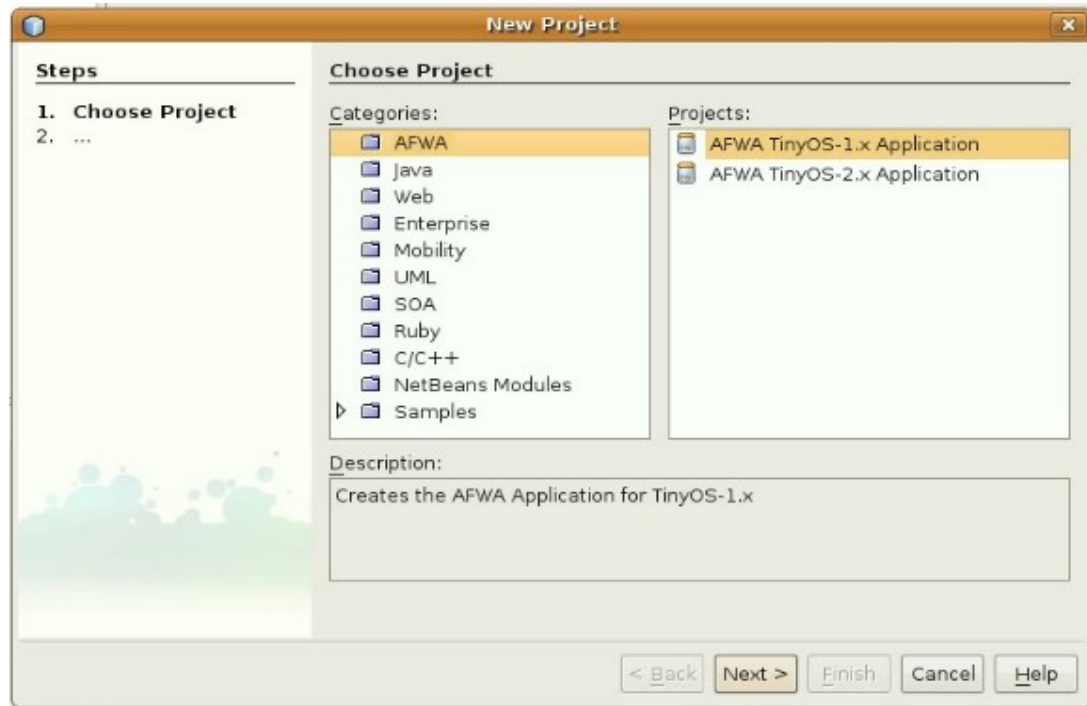
1. WSNIDE (www.cdachyd.in/ubicomp/AFWA)
2. JDK 6 (<http://java.sun.com/javase/downloads/index.jsp>)
3. TinyOS-1.x (<http://www.tinyos.net/download.html>)
4. TinyOS toolchain (<http://www.tinyos.net/download.html>)
5. TinyOS AVR and TinyOS MSP430 toolchain
6. Perl

Application Development:

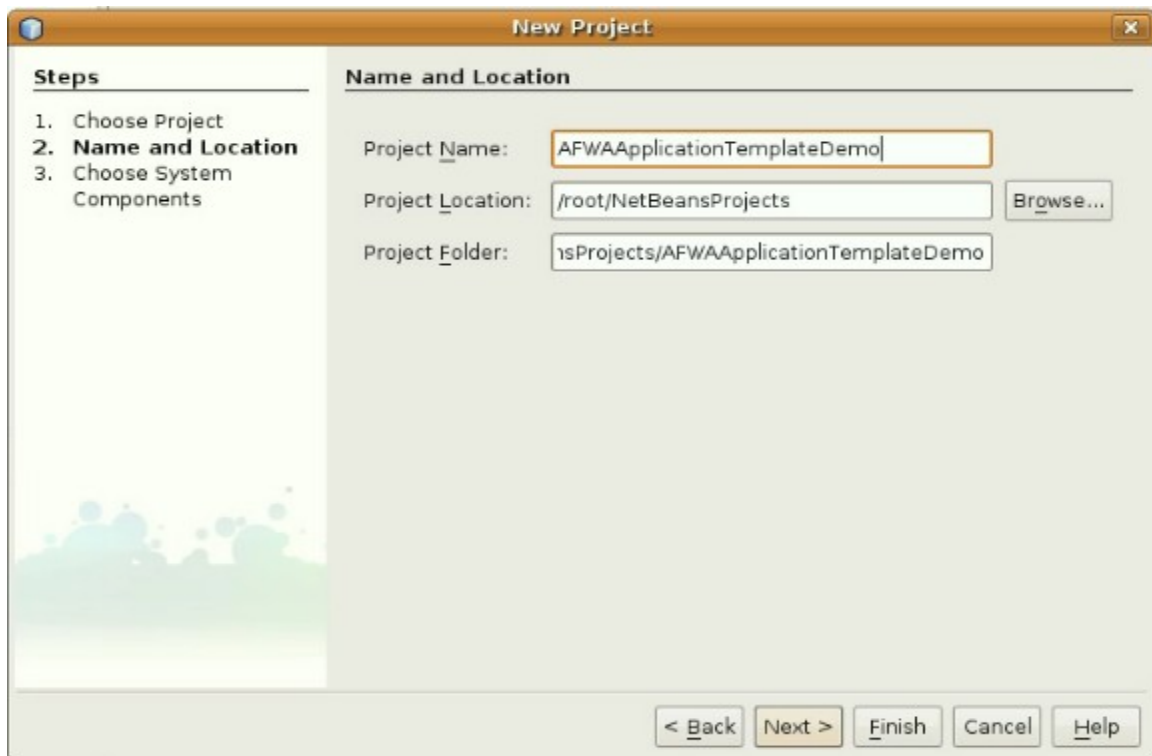
1. Invoke the Netbeans IDE
 1. Start the Netbeans IDE
2. Creating a New Project
 1. In the IDE choose File -> New Project



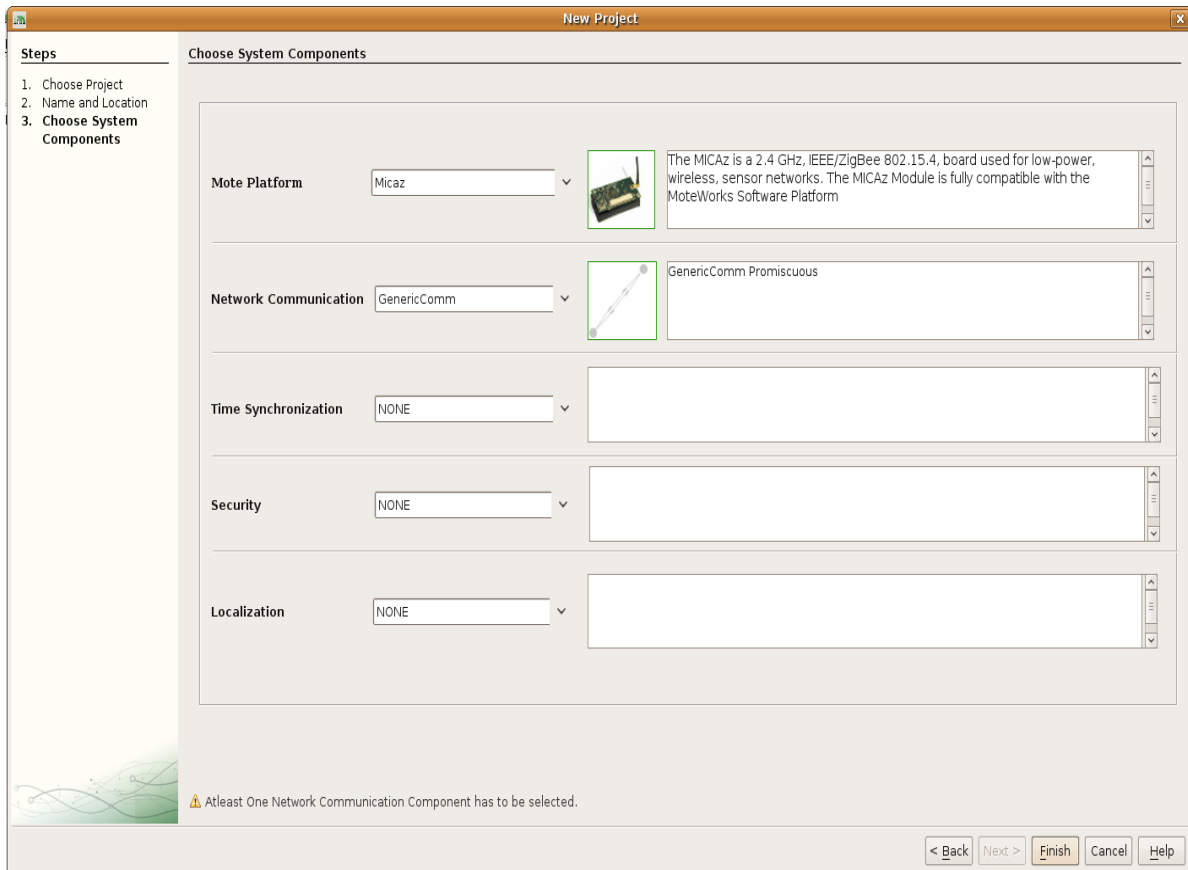
2. In the New Project Wizard, **Choose Project** expand the category AFWA and select AFWA TinyOS-1.x Application as show below and click Next



3. In the *Project Name and Location* page of the wizard
 1. In the Project Name field, enter the project name
 2. In the Location of the project, give the location of the project directory
 3. And provide the project folder name in the corresponding field
 4. Click Next

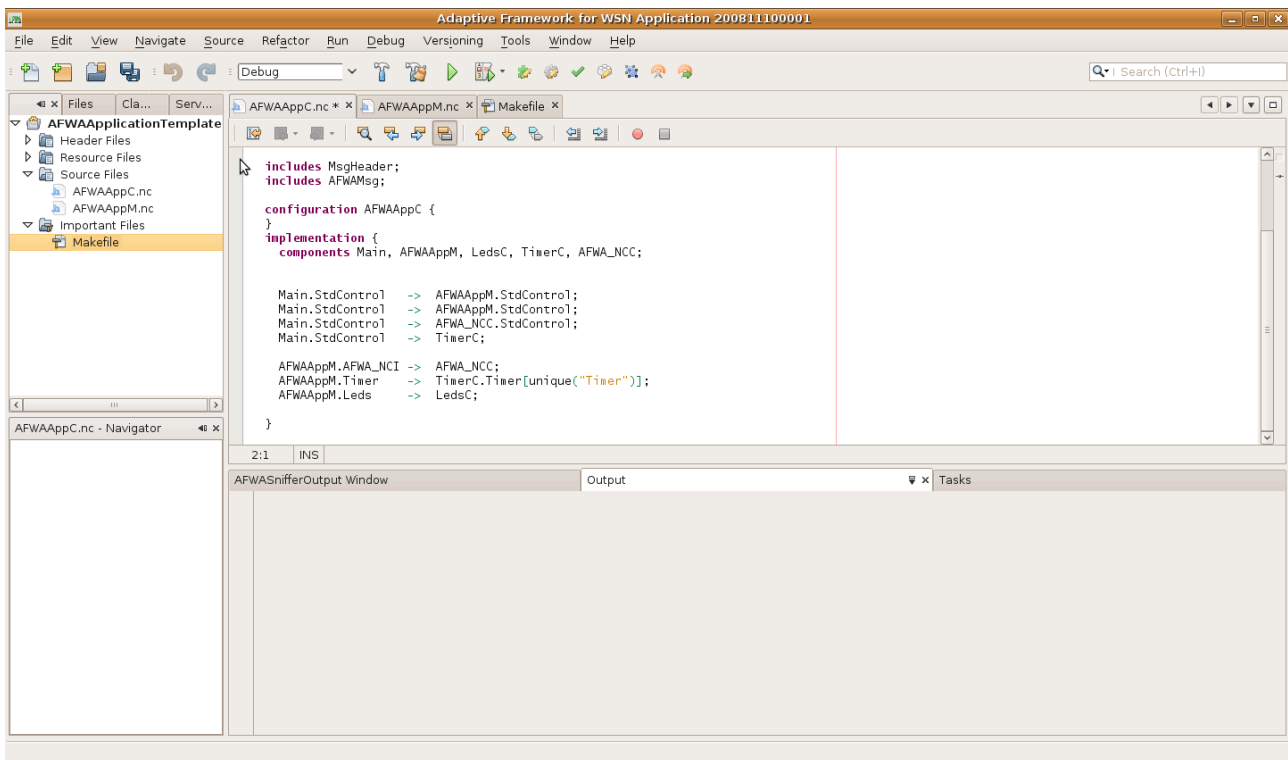


4. In the *Choose System Components* wizard, do select an
 1. MotePlatform
 2. Network Communication
 3. Time Synchronization
 4. Localization
 5. Security
 6. Click Finish

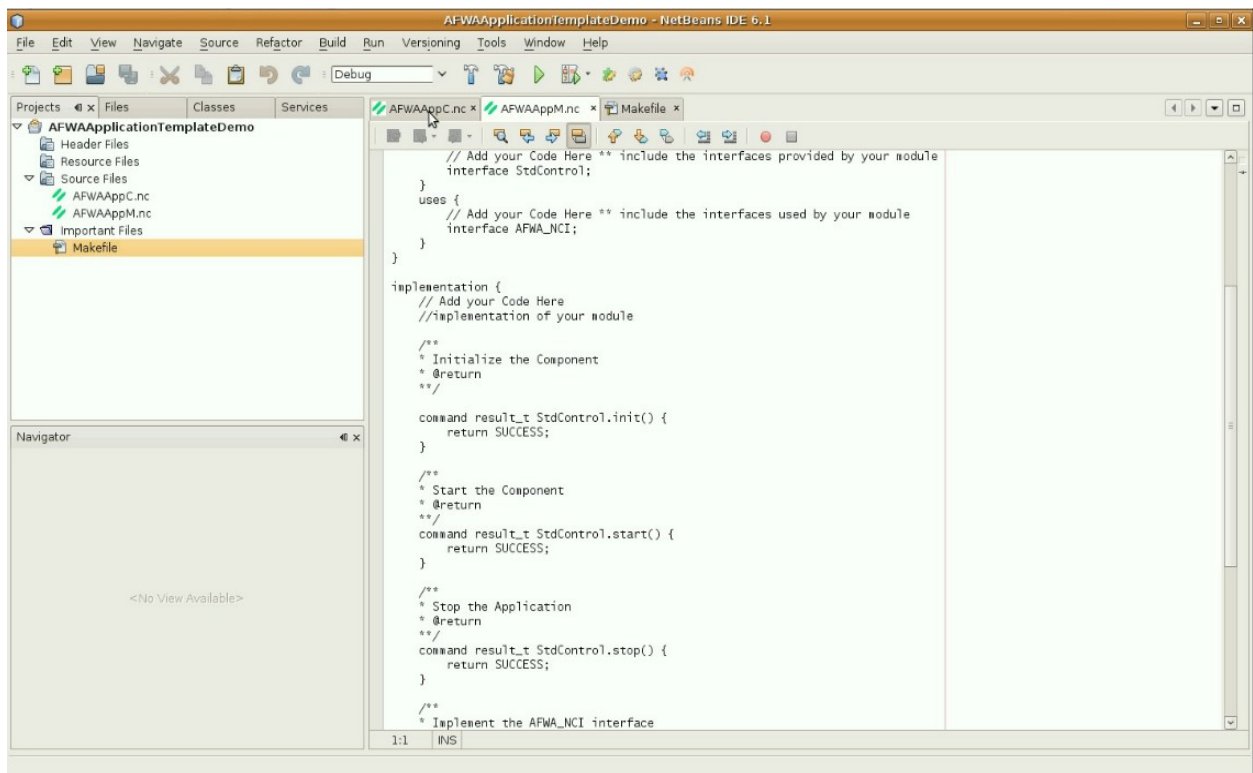


3. Project Creation

1. This Creates the project and you should be able to see the following on the IDE
 1. The Projects window, which contains a tree view of the components of the project, including source files, libraries required by your application
 2. Under Source Files you should be able to see bare template files AFWAAppC.nc and AFWAAppM.nc
 3. Under Important Files , the Makefile of your project exists
 4. The Application Configuration details of your project can be seen in Application.xml file under Important Files
 5. In the Source Editor window the files AFWAAppC.nc and AFWAAppM.nc are opened

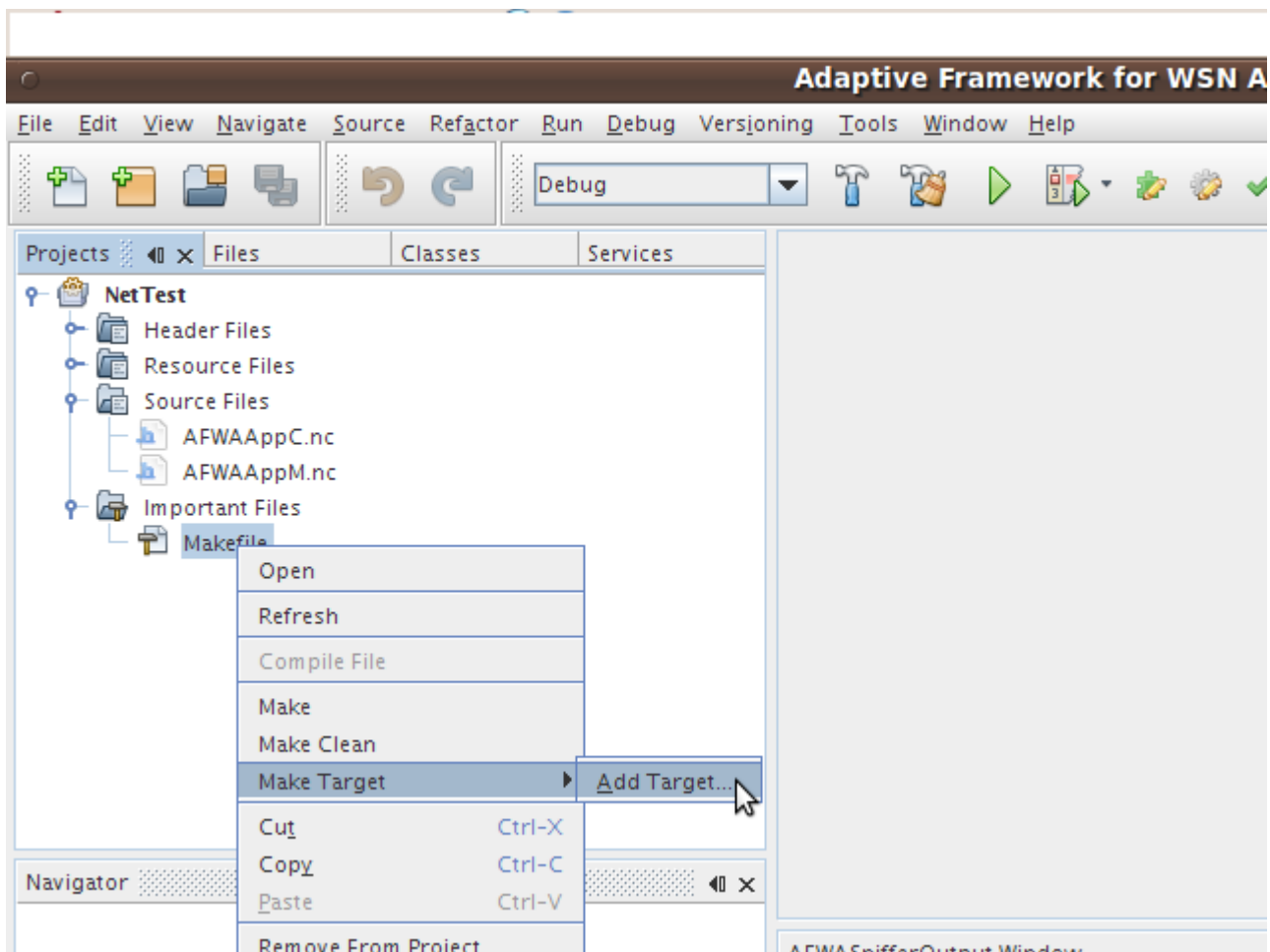


4. Add your application logic to the generated code

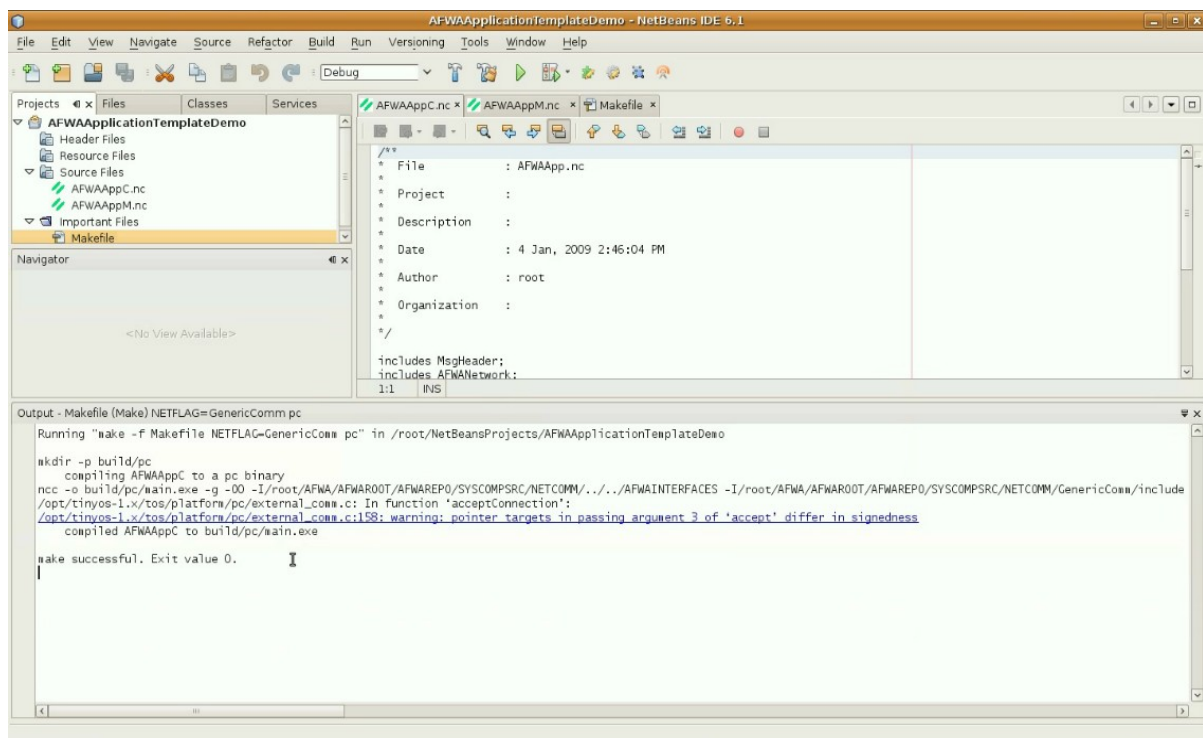
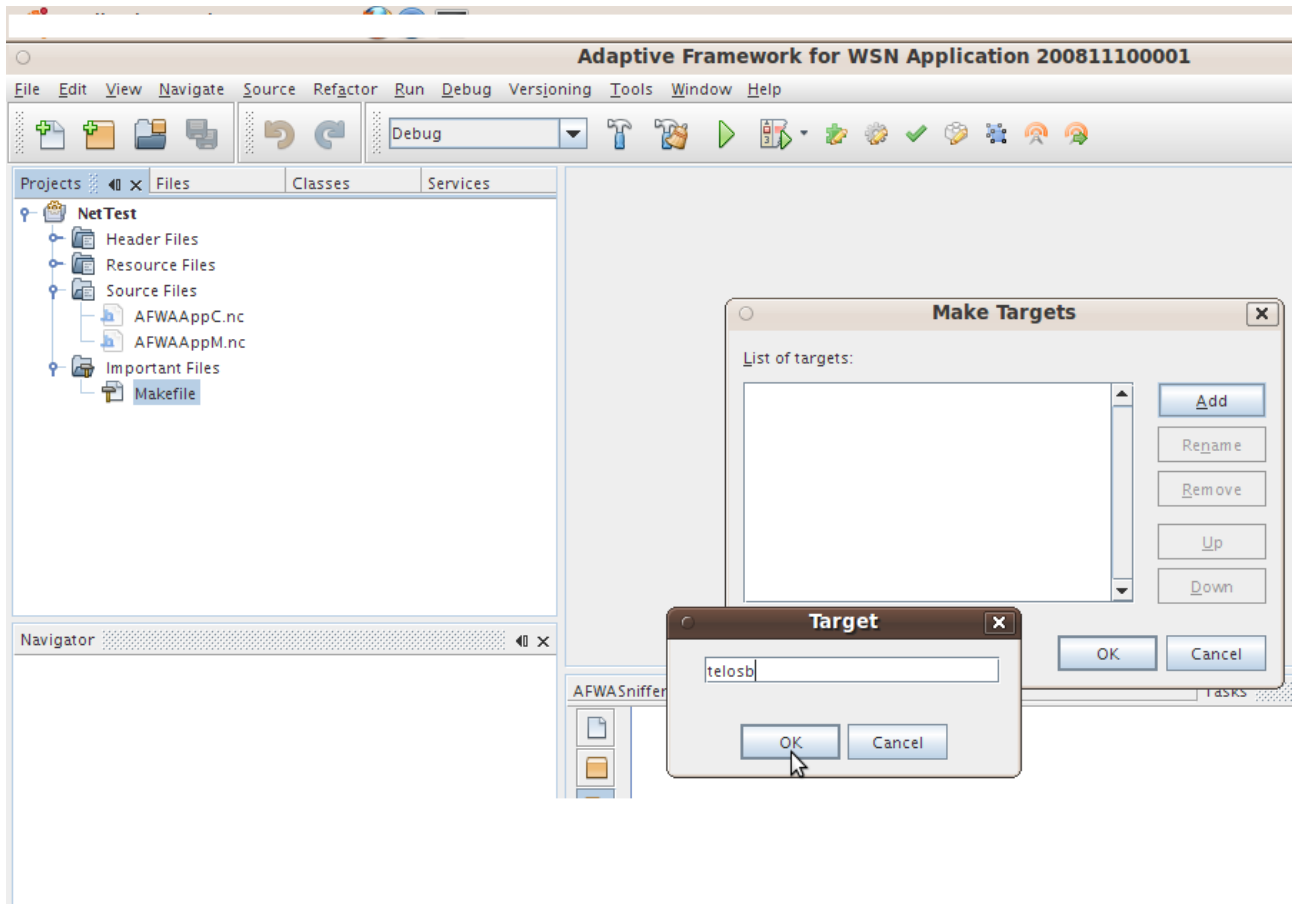


5. Compile and Build the Application

1. Right Click Makefile and Click Make Target and enter the following details in Add
 1. make <Platform>
 2. For example make telosb , make pc , make micaz
 3. Compile the application by clicking make target
 4. The output of compilation can be seen in the output window



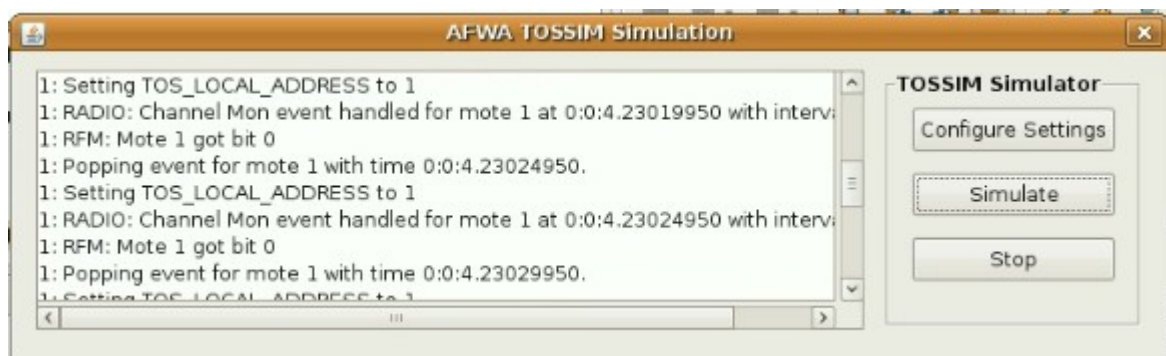
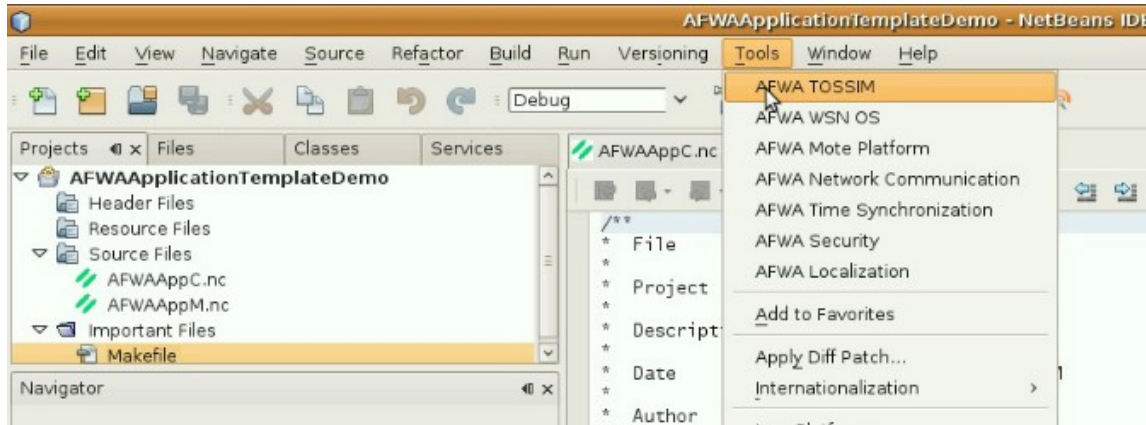
AFWA Getting Started Guide



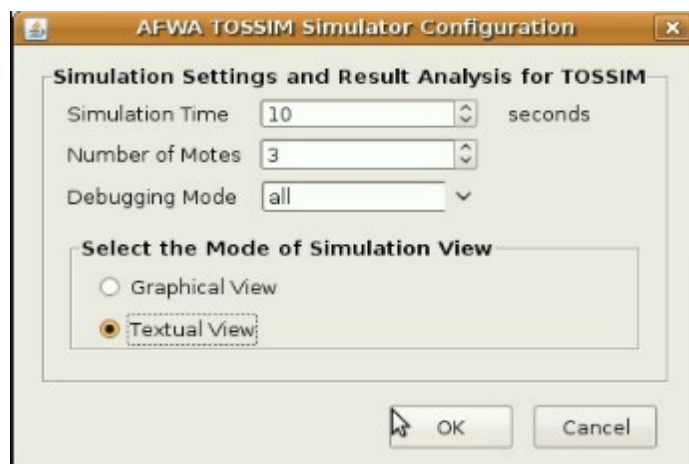
6. Simulate the Application

1. In the IDE, select AFWA TOSSIM and configure settings

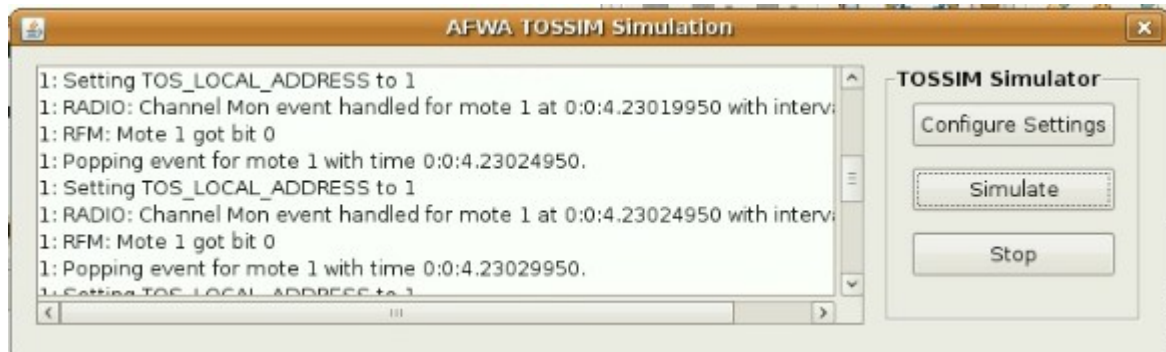
1. Give the Application requirements on the Maximum simulation time, Number of motes, Debugging mode and the view mode for the analysis



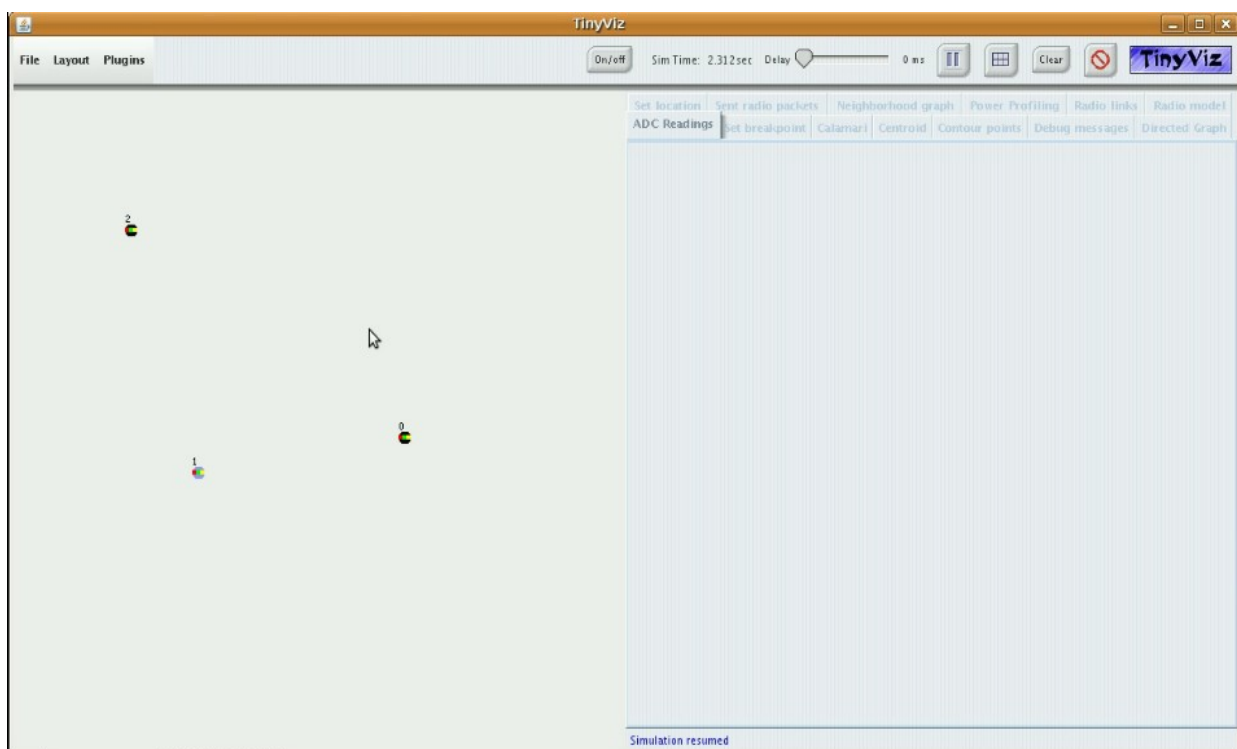
2. Select Graphical for graphical analysis and Textual for textual analysis



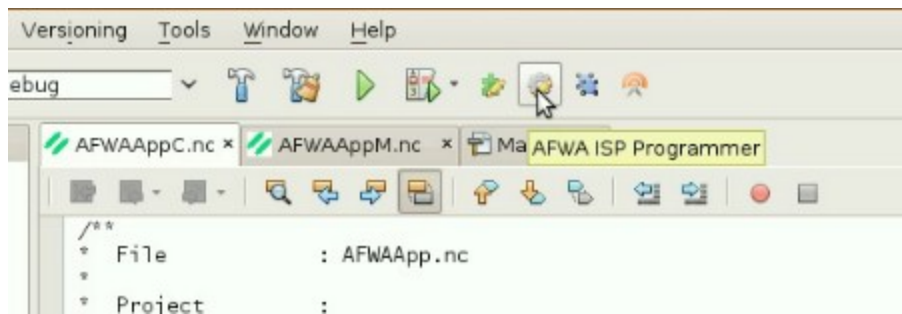
3. Click Simulate
4. The output when Textual mode selected



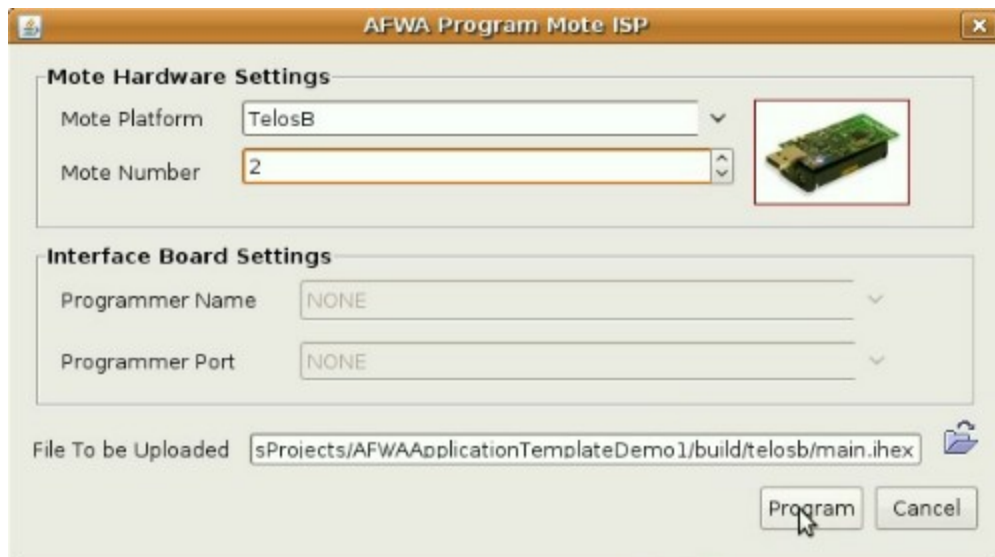
5. The output when Graphical mode selected



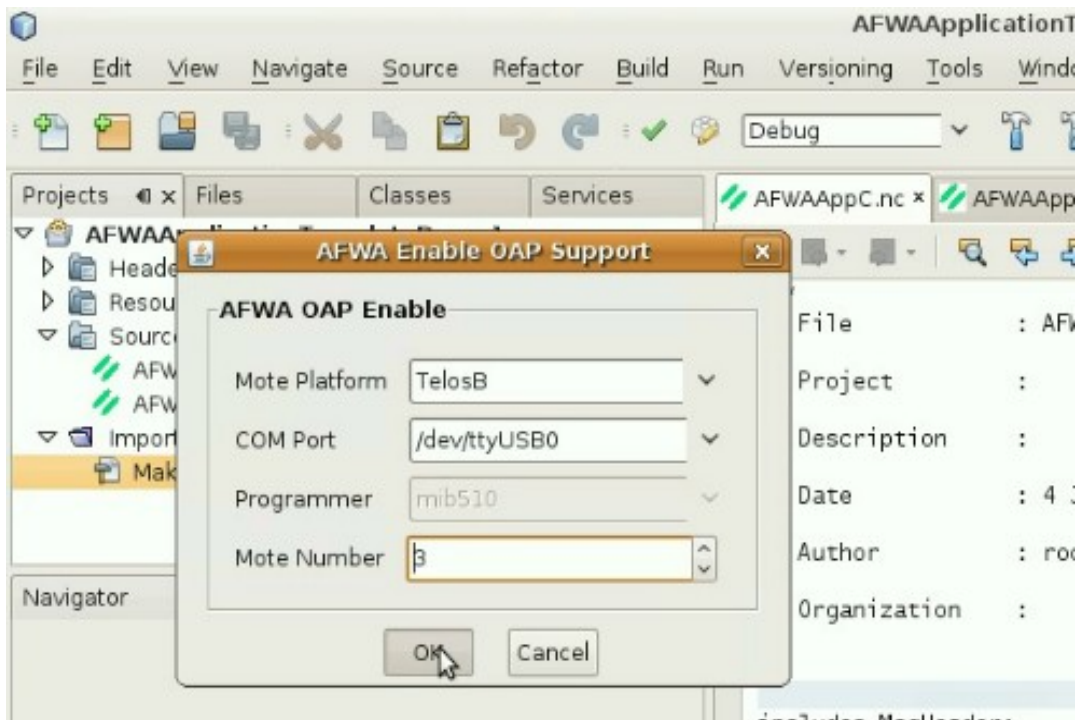
7. Program the developed application onto the motes.
 1. Programming can be done either by InSystem Programming Technique or Over the Air Programming Technique
 2. If InSystem Programming is selected



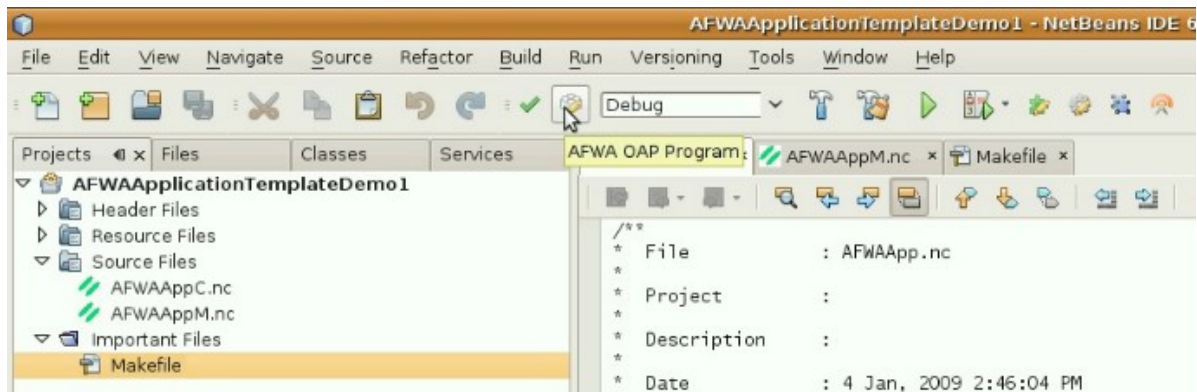
3. The ISP Configuration window opens up. Give your choice in the required fields and click Program. This Programs your mote with the built application.



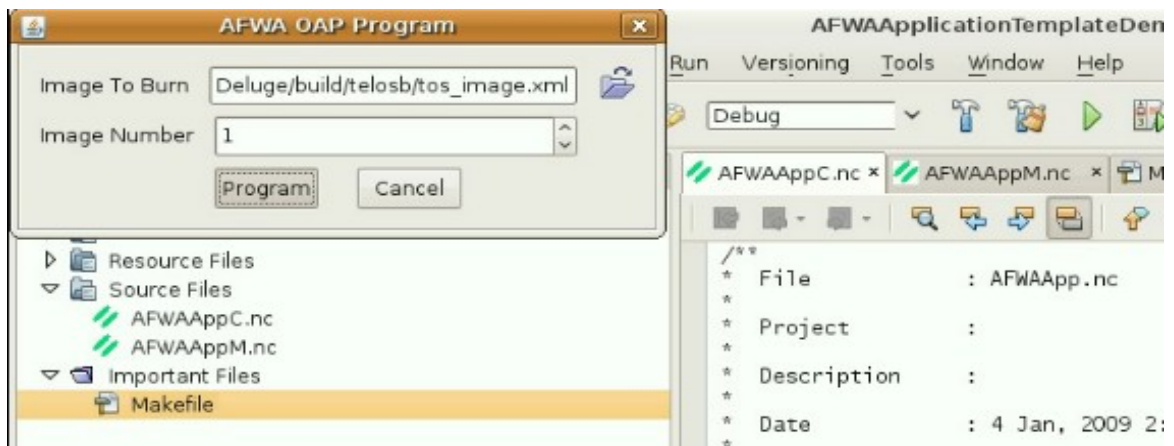
4. If OAP technique is selected, first the motes should be enabled with the OAP support.
5. Click on the icon showing OAP Enable and give the required inputs. This burns the OAP bootloader onto the motes.



6. Now, Click on the OAP Program button

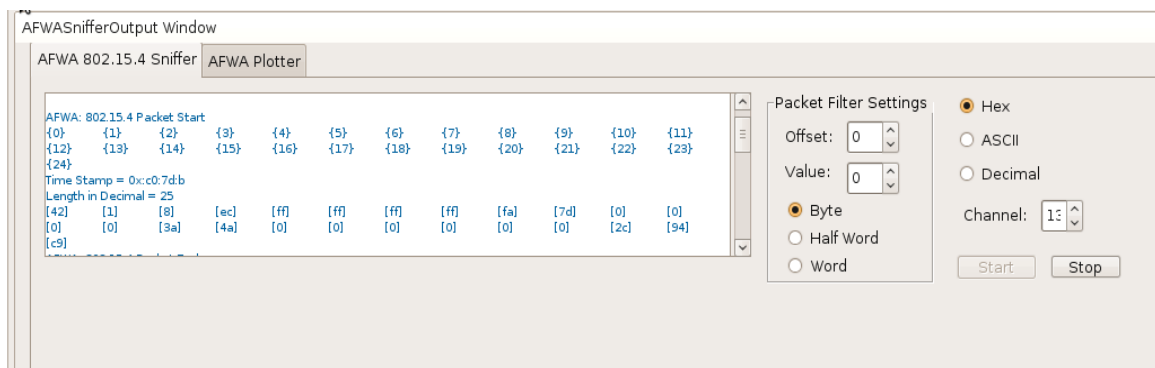


7. And give the required fields and Click Program.



8. Sniff the packets if required

1. Window - AFWA Sniffer



Annexure – Packet Format

TinyOS Message

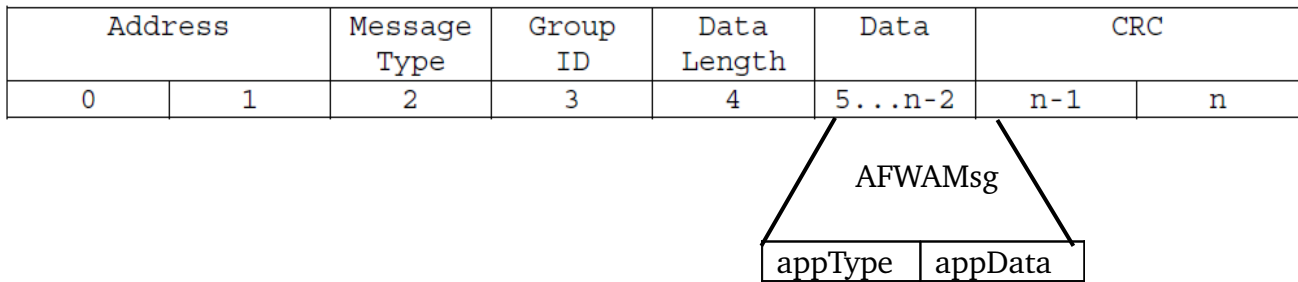
The payload data will typically be a type of TinyOS message, as defined by the struct TOS_Msg in the file /tos/types/AM.h. This data structure is defined as follows:

```
typedef struct TOS_Msg
{
/* The following fields are transmitted/received on the radio. */
    uint16_t addr;
    uint8_t type;
    uint8_t group;
    uint8_t length;
    int8_t data[TOSH_DATA_LENGTH];
    uint16_t crc;
/* The following fields are not actually transmitted or received
 * on the radio! They are used for internal accounting only.
 * The reason they are in this structure is that the AM interface
 * requires them to be part of the TOS_Msg that is passed to
 * send/receive operations.
 */
    uint16_t strength;
    uint8_t ack;
    uint16_t time;
    uint8_t sendSecurityMode;
    uint8_t receiveSecurityMode;
} TOS_Msg;
```

AFWAMsg.h

```
typedef struct AFWAMsg {
    uint8_t appType;
    uint16_t appData;
};
```

The TOS_Msg data packet is described in the following diagram and table:



| Byte # | Field | Description |
|---------|-----------------|--|
| 0 - 1 | Message Address | One of 3 possible value types: <ul style="list-style-type: none"> Broadcast Address (0xFFFF) – message to all nodes. UART Address (0x007e)– message from a node to the gateway serial port. All incoming messages will have this address. Node Address – the unique ID of a node to receive message. |
| 2 | Message Type | Active Message (AM) unique identifier for the type of message it is. Typically each application will have its own message type. Examples include: <ul style="list-style-type: none"> AMTYPE_XUART = 0x00 AMTYPE_MHOP_DEBUG = 0x03 AMTYPE_SURGE_MSG = 0x11 AMTYPE_XSENSOR = 0x32 AMTYPE_XMULTIHOP = 0x33 AMTYPE_MHOP_MSG = 0xFA |
| 3 | Group ID | Unique identified for the group of motes participating in the network. The default value is 125 (0x7d). Only motes with the same group id will talk to each other. |
| 4 | Data Length | The length (<i>l</i>) in bytes of the data payload. This does not include the CRC or frame synch bytes. |
| 5...n-2 | Payload data | The actual message content. The data resides at byte 5 through byte 5 plus the length of the data (<i>l</i> from above). The data will be specific to the message type. Specific message types are discussed in the next section. |
| n-1, n | CRC | Two byte code that ensures the integrity of the message. The CRC includes the Packet Type plus the entire unescaped TinyOS message. A discussion on how the CRC is computed is included in the Appendix. |