

# **Wireless Sensor Network Integrated Development Environment**

## **Application Programmer's Interface (API) 0.1**

Prepared by  
Poonguzhali P and Mahesh U.Patil  
National Ubiquitous Computing Research Center  
Centre for Development of Advanced Computing  
Hyderabad

## Table of Contents

1. Introduction.....	3
2. AFWA API Overview.....	3
2.1 AFWA Architecture .....	3
3. AFWA WSN Interfaces.....	5
3.1 AFWA Network Interfaces.....	6
3.2 AFWA Security Interfaces .....	10
3.3 AFWA Time Synchronization Interfaces.....	12

## ***1. Introduction***

This document describes the programming interface to the Adaptive Framework for WSN Application. The development to deployment flow of WSN application is time consuming and challenging because of the intricacies of low level interfaces, which varies between application. The AFWA implementation abstracts the underlying primitives, providing an interface to access the low level interfaces thus enabling faster service evaluation iterations

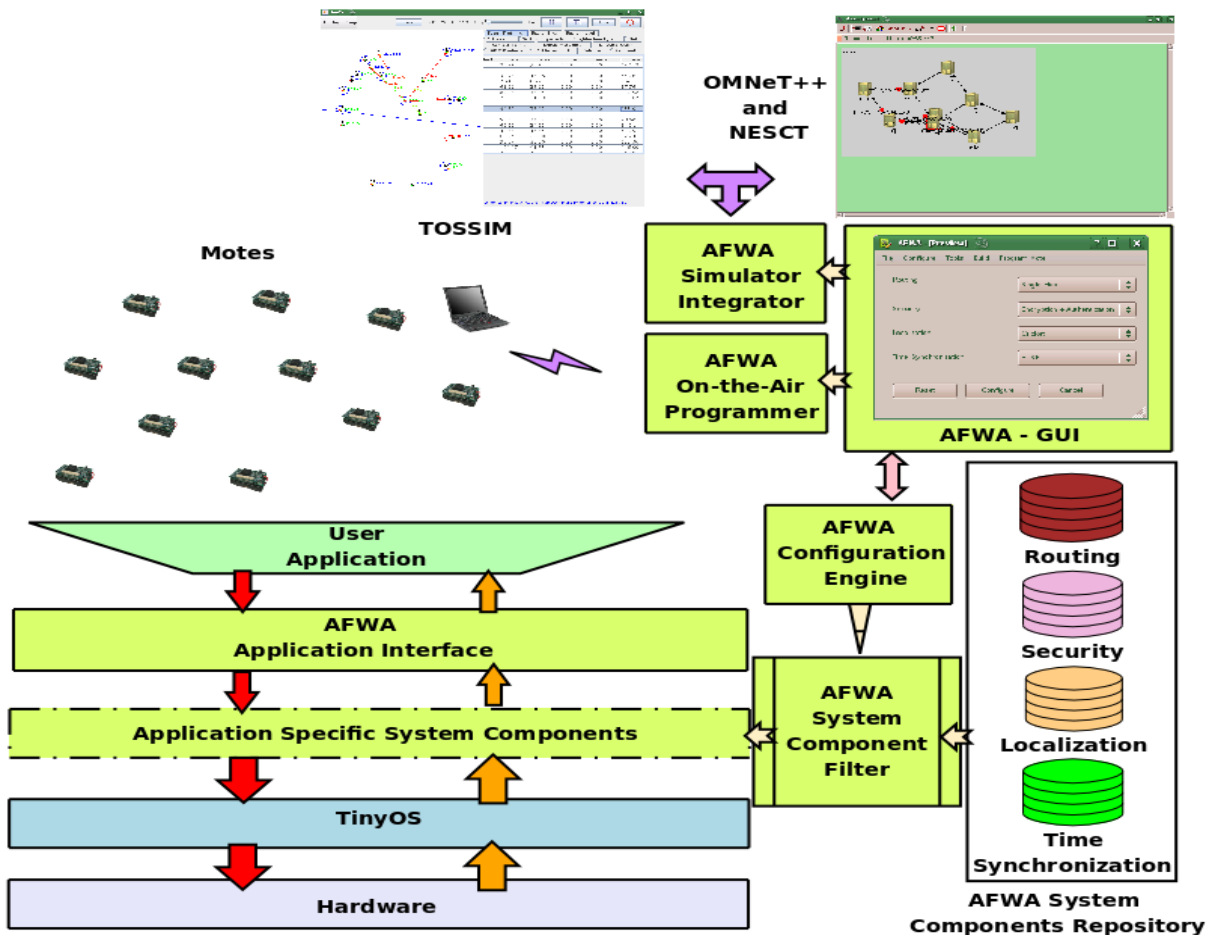
## ***2. AFWA API Overview***

### ***2.1 AFWA Architecture***

The figure below depicts the system architecture of AFWA. AFWA consists of two software components, one that resides and executes on the motes or target and the other utilized on the host or development platform. This system architecture is applicable for the first version of AFWA which focuses on TinyOS-1.x and TinyOS-2.x. AFWA provides three major functionalities namely:

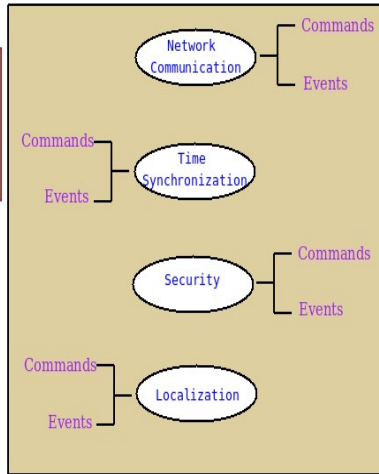
1. Facility to opt required system component for WSN Application Development
2. Facility to simulate the WSN Application
3. Facility to program the developed WSN Application On-the-Air and also In-System Programmable

AFWA GUI as shown in the figure provides the graphical interface for the WSN application developer to use the above mentioned facilities. AFWA System Component Repository consists of algorithms for Network Communication, Time Synchronization, Localization and Security. Based on the user application requirement and configuration the AFWA System Component Filter provides the interface for the WSN application. The AFWA Application Interface provides generic interface to similar system components which minimizes the modifications in the application code while opting different system component of similar type. The WSN application developer can then choose to either simulate or program the WSN application developed. These functionalities are provided by the AFWA Simulator Integrator and AFWA On-the-Air Programmer.



### 3. AFWA WSN Interfaces

- AFWA\_TSLAFWAgetLocalTime()
- AFWA\_TSLAFWAgetGlobalTime(uint32\_t \*time)
- AFWA\_TSLAFWAlocal2Global(uint32\_t \*time)
- AFWA\_TSLAFWAglobal2Local(uint32\_t \*time)
- AFWA\_TSLAFWAgetTime()
- AFWA\_TSLAFWAperiodicSync(uint16\_t period)



- AFWA\_NCLAFWAsend(AFWATOS\_MsgPtr msg, uint16\_t length)
- AFWA\_NCLAFWAgetBuffer(AFWATOS\_MsgPtr msg, uint16\_t \*length)
- AFWA\_NCLAFWAsendMsg(uint16\_t address, uint8\_t length, AFWATOS\_MsgPtr msg)
- AFWA\_NCLAFWAgetParent()
- AFWA\_NCLAFWAgetSender(TOS\_MsgPtr msg)

- event result\_t AFWA\_NCLAFWAsendDone(AFWATOS\_MsgPtr msg, result\_t success){ }
- event AFWATOS\_MsgPtr AFWA\_NCLAFWAreceive(AFWATOS\_MsgPtr msg, void\* payload, uint16\_t payloadLen){ }
- event AFWATOS\_MsgPtr AFWA\_NCLAFWAreceiveMsg(AFWATOS\_MsgPtr m){ }

- AFWA\_SECLAFWAsetTransmitMode(uint8\_t mode)
- AFWA\_SECLAFWAsetReceiveMode(uint8\_t mode)
- AFWA\_SECLAFWAgetTransmitMode()
- AFWA\_SECLAFWAgetReceiveMode()

### 3.1 AFWA Network Interfaces

Commands :

Syntax :

```
AFWA_NCI.AFWAsend(AFWATOS_MsgPtr msg, uint16_t length)
```

Description :

Send a message buffer with a data payload of specified length

Parameters :

msg - Buffer to Send  
length – Length of data buffer

Return Values :

Send Request Status (SUCCESS/FAIL)

Syntax :

```
AFWA_NCI.AFWAgetBuffer(AFWATOS_MsgPtr msg, uint16_t*  
length)
```

Description :

Pointer to the data buffer of the application and the length which allows the application to send a specific buffer while not requiring the knowledge of packet structure

Parameters :

msg – Message to get the data  
length – Pointer to a field to store the length of data

Return Values :

Pointer to Data field

Syntax :

```
AFWA_NCI.AFWAsendMsg(uint16_t address, uint8_t length,  
                      AFWATOS_MsgPtr msg)
```

Description :

Send a message with the data payload of specified length to the specified address

Parameters :

address – Destination Address

length – Length of the buffer

msg – Buffer to send

Return Values :

Send Request Status (SUCCESS/FAIL)

Syntax :

```
AFWA_NCI.AFWAgetParent()
```

Description :

Get the nodes present Parent Address

Parameters :

void

Return Values :

Address of Parent

Syntax :

```
AFWA_NCI.AFWAgetSender(TOS_MsgPtr msg)
```

Description :

Get the previous hop sender for the given message

Parameters :

msg – Pointer to the message of interest

Return Values :

Address of Sender

### *Events*

Syntax :

```
event result_t AFWA_NCI.AFWAsendDone(AFWATOS_MsgPtr msg,  
result_t success) { }
```

Description :

Signalled when the packet is sent successfully

Parameters :

msg – Message Sent

success – Status

Return Values :

Should always return SUCCESS

Syntax :

```
event AFWATOS_MsgPtr AFWA_NCI.AFWAreceive(AFWATOS_MsgPtr  
msg, void* payload, uint16_t payloadLen) {}
```

Description :

Received a message buffer addressed to us

Parameters :

msg – Received data buffer

payload – Payload of the packet received

payoadLen – Length of Payload buffer

Return Values :

Buffer to use for the next receive event

Syntax :

```
event AFWATOS_MsgPtr AFWA_NCI.AFWAreceiveMsg(AFWATOS_MsgPtr m) { }
```

Description :

Packet Received

Parameters :

m – Received Data

Return Values :

Buffer for the provider to use for the next packet

## 3.2 AFWA Security Interfaces

### Commands

#### Syntax :

```
AFWA_SECI.AFWAsetTransmitMode(uint8_t mode)
```

#### Description :

Set the transmit mode for TinySec

#### Parameters :

mode – Should be one of the following

TINYSEC\_AUTH\_ONLY

TINYSEC\_ENCRYPT\_AND\_AUTH

TINYSEC\_DISABLED

#### Return Values :

SUCCESS if mode parameter is set

#### Syntax :

```
AFWA_SECI.AFWAsetReceiveMode(uint8_t mode)
```

#### Description :

Sets the receive mode for TinySec

#### Parameters :

mode – Should be one of the following

TINYSEC\_RECEIVE\_AUTHENTICATED

TINYSEC\_RECEIVE\_CRC

TINYSEC\_RECEIVE\_ANY

#### Return Values :

SUCCESS if the mode parameter is valid

Syntax :

```
AFWA_SECI.AFWAgetTransmitMode( )
```

Description :

Gets the current transmit mode for TinySec

Parameters :

none

Return Values :

Current transmit mode

Syntax :

```
AFWA_SECI.AFWAgetReceiveMode( )
```

Description :

Gets the current receive mode for TinySec

Parameters :

none

Return Values :

Current receive mode

### 3.3 AFWA Time Synchronization Interfaces

#### Commands

Syntax :

```
AFWA_TSI.AFWAgetLocalTime()
```

Description :

Reads the current local time

Parameters :

none

Return Values :

Current local time of the mote

Syntax :

```
AFWA_TSI.AFWAgetGlobalTime(uint32_t *time)
```

Description :

Reads the current global time

Parameters :

time – time value

Return Values :

SUCCESS if the mote is synchronized

Syntax :

```
AFWA_TSI.AFWAlocal2Global(uint32_t *time)
```

Description :

Converts the local time to the corresponding global time

Parameters :

time – local time value

Return Values :

SUCCESS if the mote is synchronized

Syntax :

```
AFWA_TSI.AFWAglobal2Local(uint32_t *time)
```

Description :

Converts the global time to the corresponding local time

Parameters :

time – global time

Return Values :

SUCCESS if the mote is synchronized

Syntax :

```
AFWA_TSI.AFWAgetTime ( )
```

Description :

Gets the mticks of the current global time

Parameters :

none

Return Values :

Mticks corresponding to the current global time

Syntax :

```
AFWA_TSI.AFWAperiodicSync(uint16_t period)
```

Description :

Command to initiate the periodic synchronization mode

Parameters :

period – Specifies the number of ticks after which the nodes exchange synchronization packets

Return Values :

none