Awia Warrior 210
User Manual

AwiaTech Corporation

www.awiatech.com

Table of Contents

FCC STATEMENT

1. This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) This device may not cause harmful interference, and
(2) This device must accept any interference received, including interference that may cause undesired operation.
2. Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation.
This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient or relocate the receiving antenna.
Increase the separation between the equipment and receiver.
Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
Consult the dealer or an experienced radio/TV technician for help.

FCC Radiation Exposure Statement

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment.
This equipment should be installed and operated with minimum distance 20cm between the radiator & your body

1. INTRODUCTION

A WirelessHART network consists of three key components: network manager/gateway, access points, and field devices, as show in Figure 2. The central network manager/gateway is the brain of the network, which schedules traffic inside the network. A field device powered by Awia Warrior stack is first admitted to the network by the manager. Then it can be configured to fulfill certain tasks in the process. This document describes the Awia Warrior development tool kit.

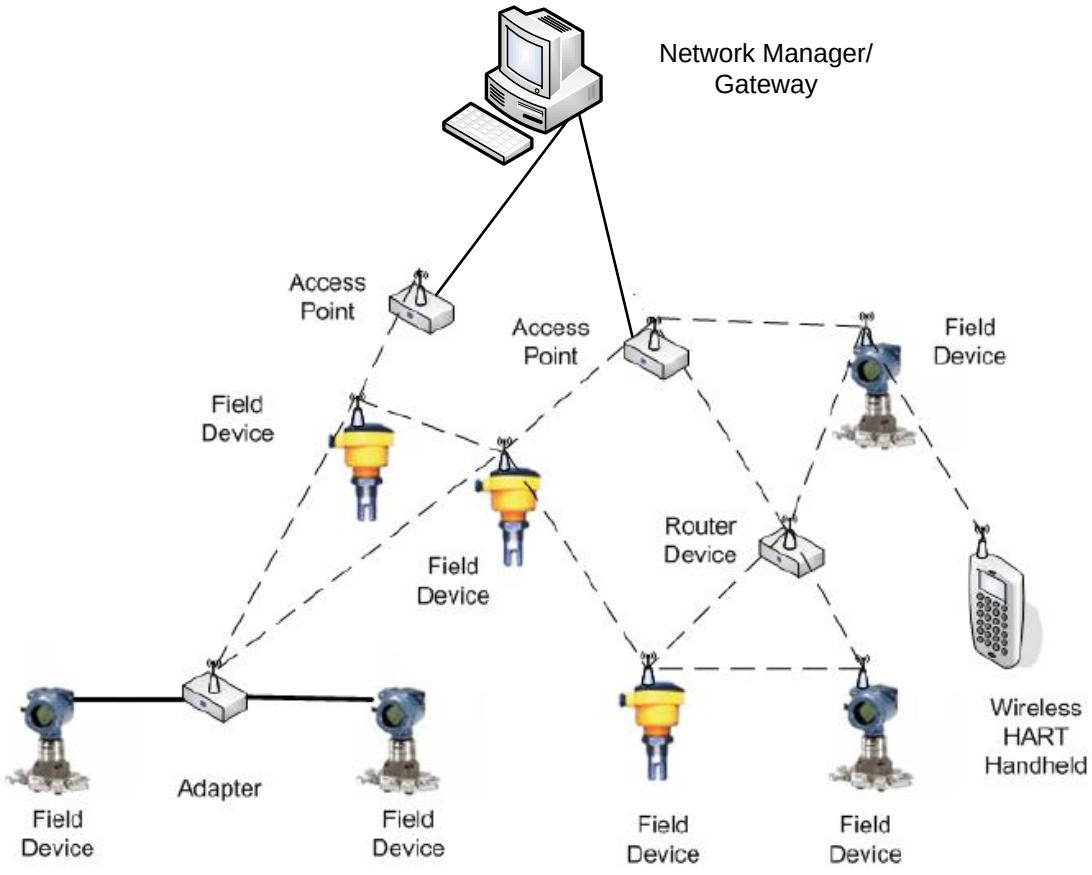

Figure 1. A typical WirelessHART network

Awia Warrior stack fully complies with the WirelessHART standard. Any of the functionality and interfaces described in this document, if falls under the scope of the WirelssHART standard, describes also the same of the standard.

Figure 2 illustrates the architecture of HART Communication Protocol according to the OSI 7-layer communication model, whose wireless part also describes the Awia Warrior 210

WirelessHART protocol stack. As shown in this figure, Awia Warrior 210 protocol stack includes five layers: physical layer, data link layer, network layer, transport layer and application layer.

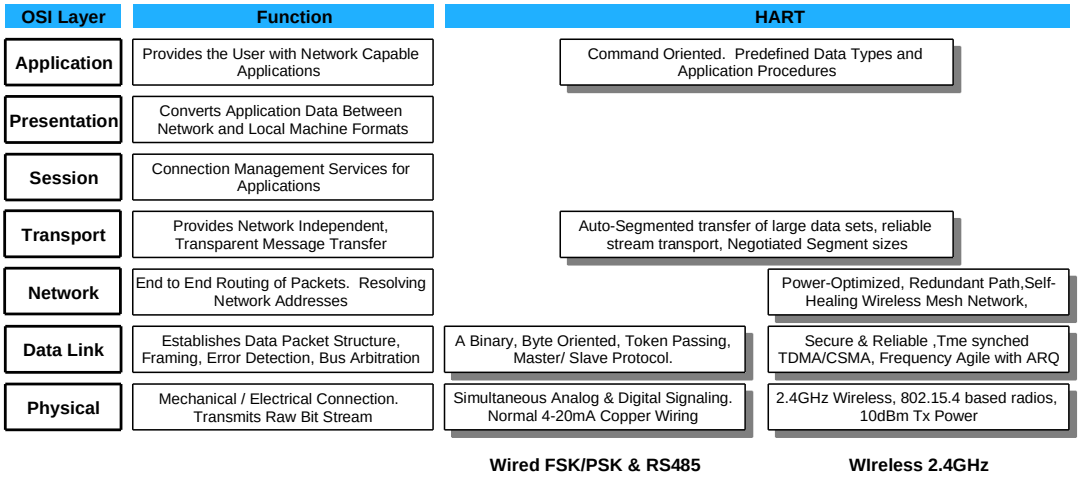

Figure 2 Architecture of HART Communication Protocol

2. AWIATECH WARRIOR STACK

2.1 Physical layer

The Awia Warrior 210 physical layer is based mostly on the IEEE STD 802.15.4-2006 2.4GHz DSSS physical layer. This layer defines radio characteristics, such as the signaling method, signal strength, and device sensitivity. Just as IEEE 802.15.4, Awia Warrior 210 operates in the 2400-2483.5MHz license-free ISM band with a data rate of 250 kbits/s. The physical communication channels are numbered from 11 to 26, with a 5MHz gap between two adjacent channels.

2.2 Data Link Layer

One distinct feature of Awia Warrior 210 is the time synchronized data link layer. Awia Warrior 210 defines a strict 10ms time slot and utilizes TDMA technology to provide collision free and deterministic communications. The concept of superframe is introduced to group a series of consecutive time slots. Note a superframe is periodical, with the total length of the member slots as the period. All superframes in an Awia Warrior 210 network start from the ASN (absolution slot number) 0, the time when the network is first created. Each superframe then repeats itself along the time based on its period. In Awia Warrior 210, a transaction in a time slot is described by a vector: {*frame id*, *index*, *type*, *src addr*, *dst addr*, *channel offset*} where *frame id* identifies the specific superframe; *index* is the index of the slot in the superframe; *type* indicates the type of the slot (transmit/receive/idle); *src addr* and *dst addr* are the addresses of the source device and destination device, respectively; *channel offset* provides the logical channel to be used in the transaction. To fine-tune the channel usage, Awia Warrior 210 uses the idea of channel blacklisting. Channels affected by consistent interferences could be put in the black list. In this way, the network administrator can disable the use of those channels in the black list totally. To support channel hopping, each device maintains an active channel table. Due to channel blacklisting, the table may have less than 16 entries. For a given slot and channel offset, the actual channel is determined from the formula:

$$\text{ActualChannel} = (\text{ChannelOffset} + \text{ASN}) \% \text{NumChannels}$$

The actual channel number is used as an index into the active channel table to get the physical channel number. Since the ASN is increasing constantly, the same channel offset may be mapped to different physical channels in different slots. Thus we provide channel diversity and enhance the communication reliability. Figure 3 describes the overall design of the data link layer which consists of six major modules as described in the follow subsections.

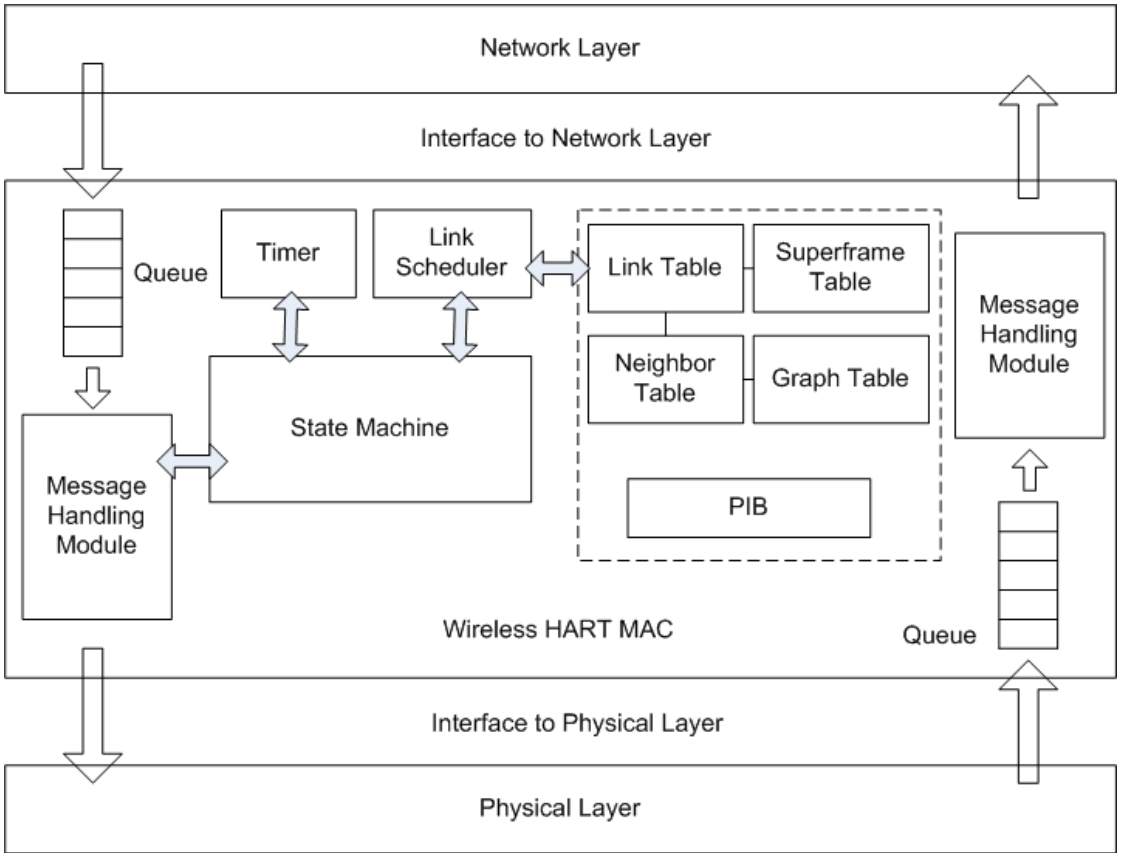

Figure 3 Awia Warrior 210 Data Link Layer Architecture

Interfaces

The interface between the MAC and PHY layer describes the service primitives provided by the physical layer, and the interface between the MAC and NETWORK layer defines the service primitives provided to the network layer.

Timer

Timer is a fundamental module in Awia Warrior 210. It provides accurate timing to ensure the correct operating of the system. One significant challenge we met during the implementation is how to design the timer module and keep those 10ms time slots in synchronization.

Communication Tables

Each network device maintains a collection of tables in the data link layer. The superframe table and link table store communication configurations created by the network manager; the neighbor table is a list of neighbor nodes that the device can reach directly and the graph table is used to collaborate with the network layer and record routing information.

Link Scheduler

The functionality of the link scheduler is to determine the next slot to be serviced based on the communication schedule in the superframe table and link table. The scheduler is complicated by such factors as transaction priorities, the link changes, and the enabling and disabling of

superframes. Every event that can affect link scheduling will cause the link schedule to be re-assessed.

Message Handling Module

The message handling module buffers the packets from the network layer and physical layer separately.

State Machine

The state machine in the data link layer consists of three primary components: the TDMA state machine, the XMIT and RECV engines. The TDMA state machine is responsible for executing the transaction in a slot and adjusting the timer clock. The XMIT and RECV engine deal with the hardware directly, which send and receive a packet over the transceiver, respectively.

2.3 Network Layer and Transport Layer

The network layer and transport layer cooperate to provide secure and reliable end-to-end communication for network devices 2. Figure 4describes the overall design of the network and transport layer.

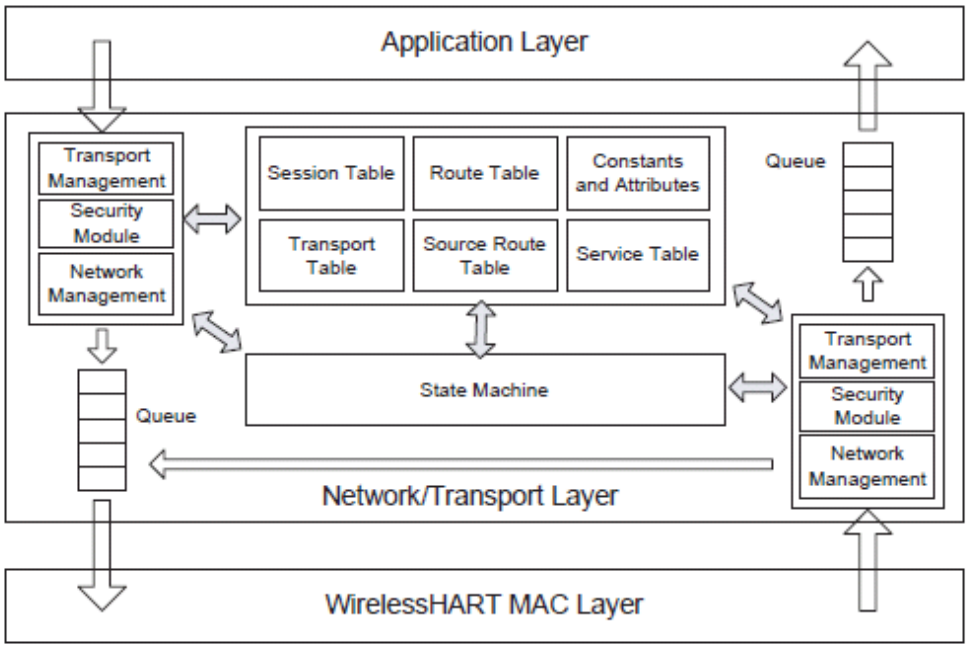

Figure 4 Awia Warrior 210 Network Layer Architecture

The basic elements of a typical Awia Warrior 210 network include: (1) Field Devices that are attached to the plant process, (2) Handheld which is a portable WirelessHART-enabled computer used to configure devices, run diagnostics, and perform calibrations, (3) A gateway that connects host applications with field devices, and (4) A network manager that is responsible for configuring the network, scheduling and managing communication between Awia Warrior 210 devices. To support the mesh communication technology, each Awia Warrior 210 device is

required to be able to forward packets on behalf of other devices. There are two routing protocols defined in Awia Warrior 210:

Graph Routing

A graph is a collection of paths that connect network nodes. The paths in each graph is explicitly created by the network manager and downloaded to each individual network device. To send a packet, the source device writes a specific graph ID (determined by the destination) in the network header. All network devices on the way to the destination must be pre-configured with graph information that specifies the neighbors to which the packets may be forwarded. Awia Warrior also supports WirelessHART superframe routing, a variation of graph routing.

Source Routing

Source Routing is a supplement of the graph routing aiming at network diagnostics. To send a packet to its destination, the source device includes in the header an ordered list of devices through which the packet must travel. As the packet is routed, each routing device utilizes the next network device address in the list to determine the next hop until the destination device is reached.

2.4 Application Layer

Figure 5describes the overall design of the application layer. The application layer is the topmost layer in Awia Warrior 210. It defines various device commands, responses, data types and status reporting. In Awia Warrior 210, the communication between the devices and gateway or network manager is based on commands and responses. The application layer is responsible for parsing the message content, extracting the command number, executing the specified command, and generating responses.

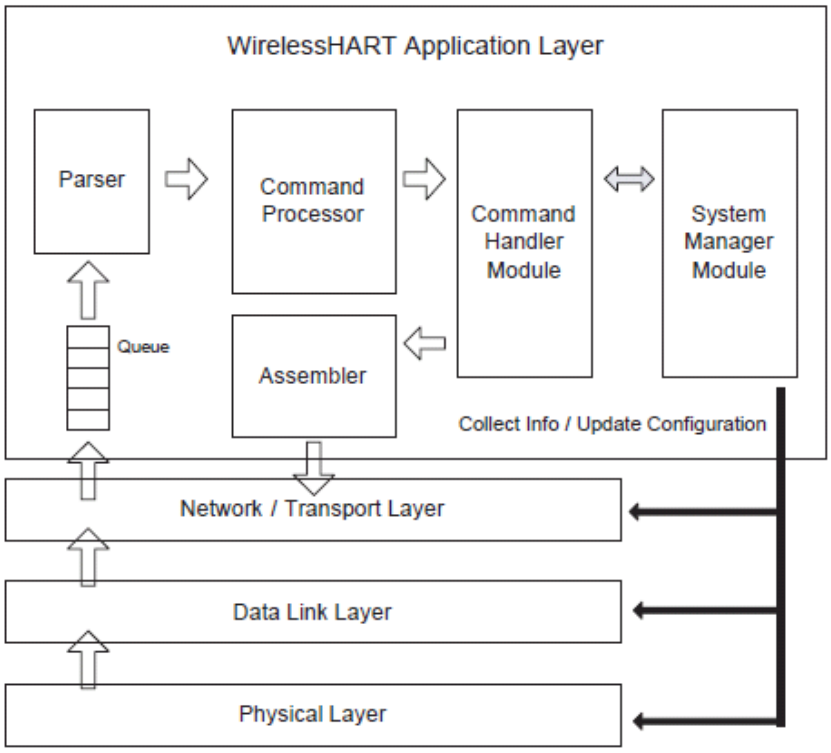

Figure 5 Awia Warrior 210 Application Layer Architecture

2.5 Security Architecture

Awia Warrior 210 is a secure network system. Both the MAC layer and network layer provide security services. The MAC layer provides hop-to-hop data integrity by using MIC. Both the sender and receiver use the CCM* mode together with AES-128 as the underlying block cipher to generate and compare the MIC. The network layer employs various keys to provide confidentiality and data integrity for end-to-end connections. Four types of keys are defined in the security architecture:

- One public Keys which is used to generate MICs on the MAC layer when network key is not applicable.
- Network Keys which are shared by all network devices and used by existing devices in the network to generate MAC MIC's.
- Join Keys that are unique to each network device and is used during the joining process to authenticate the joining device with the network manager.
- Session Keys that are generated by the network manager and is unique for each end-to-end connection between two network devices. It provides end-to-end confidentiality and data integrity.

Figure 6describes the usage of these keys under two different scenarios: 1) a new network device wants to join the network and 2) an existing network device is communicating with the network manager. In the first scenario, the joining device will use the public key to generate the MIC on MAC layer and use the join key to generate the network layer MIC and encrypt the join request. After the joining device is authenticated, the network manager will create a session key for the device and thus establish a secure session between them. In the second scenario, on the

MAC layer, the DLPDU is authenticated with the network key; on the network layer, the packet is authenticated and encrypted by the session key.

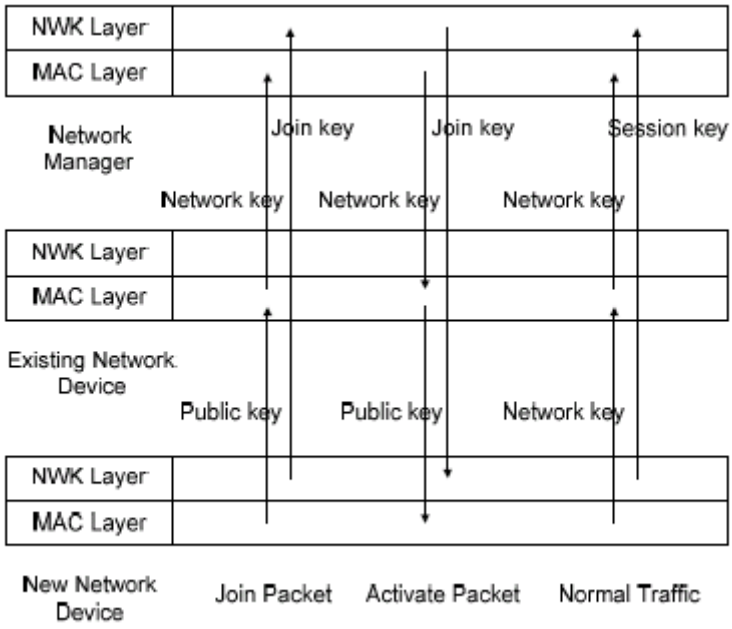

Figure 6 Awia Warrior 210 Keying Model

3. AWIATECH WARRIOR SYSTEM

3.1 System Architecture

AwiaTech WirelessHART network provides a wireless medium to transport sensor data to the host application and other data from the host to the sensor. There are two interfaces, one to the sensors, and one to the host application, as is shown in the Figure 7. The access point and warriors are AwiaTech modules. The Awia Gateway and network manager (GW & NM) are software programs running in a personal computer.

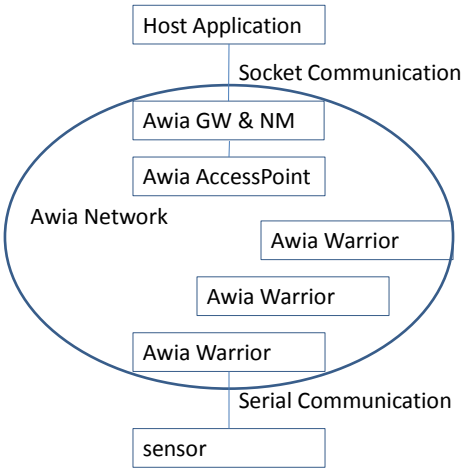

Figure 7 Awia Warrior System

Awia WirelessHART Development Tool Kit is a demo system with all the components within the circle of Figure 7 plus a demo host application. There is a simulated sensor built in the Awia Warrior which could be activated in the demo system.

3.2 Demo System Usage

- Launch the Awia network manager and gateway.
- Connect the access point to the PC. The access point is powered through the USB cable.
- Power on Awia Warriors.
- Start host application. Interact with the network via the host application.

Figure 8 shows the network manager window. Figure 9 shows the gateway window. Figure 10 shows the demo host window. Figure 11 shows the picture of the actual system.

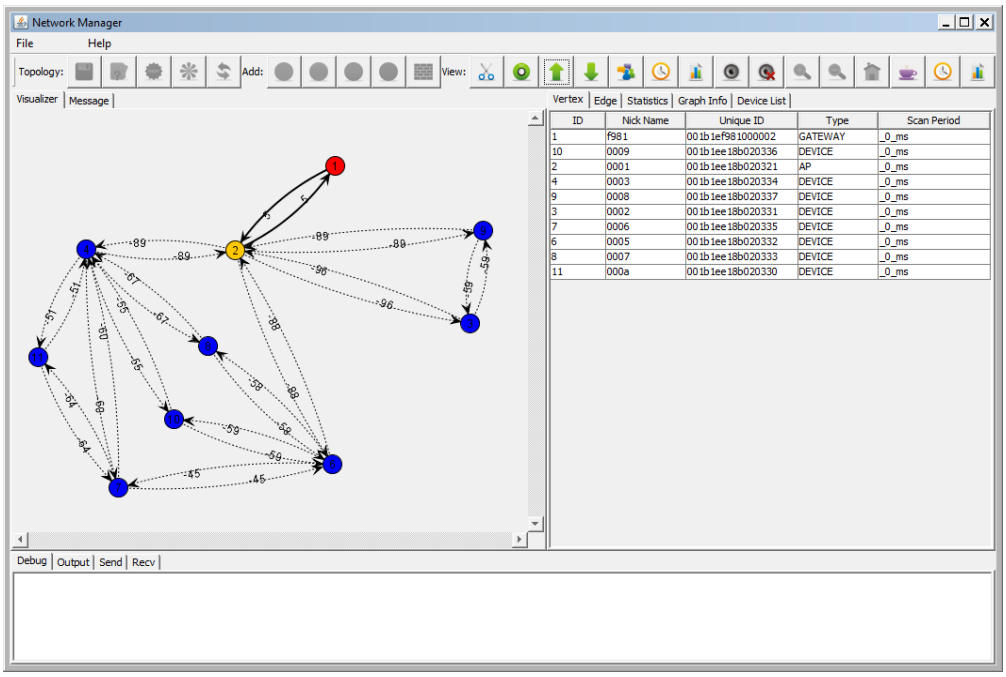

Figure 8 Awia network manager

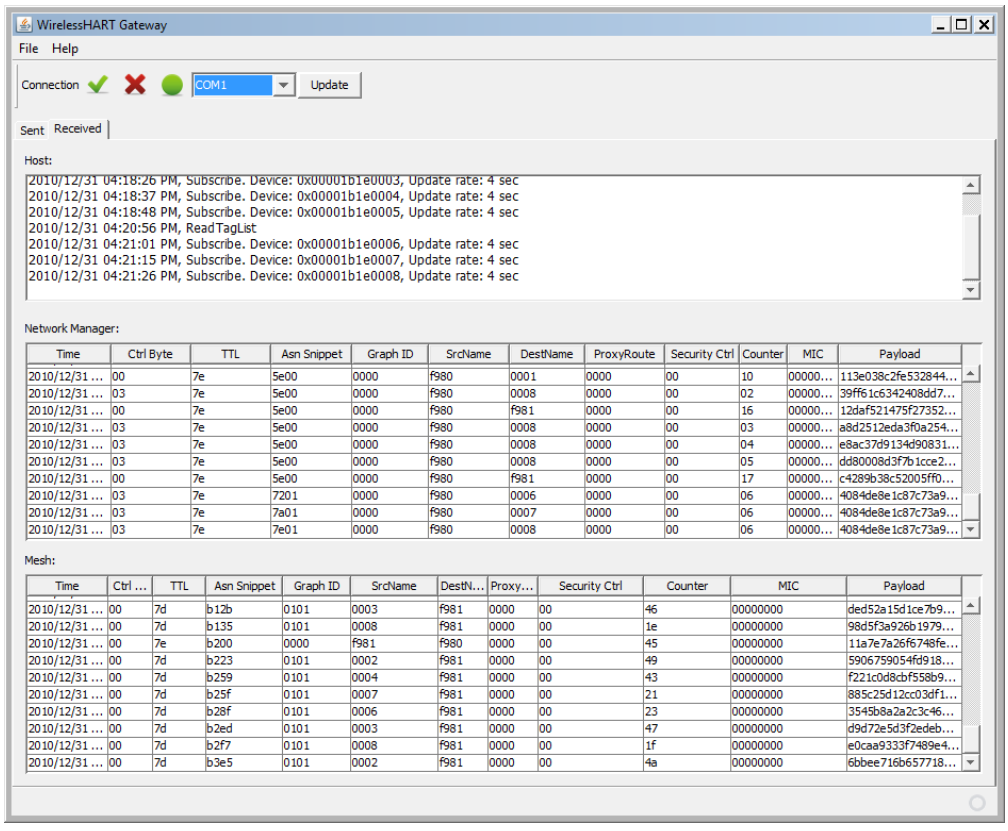

Figure 9 Awia gateway

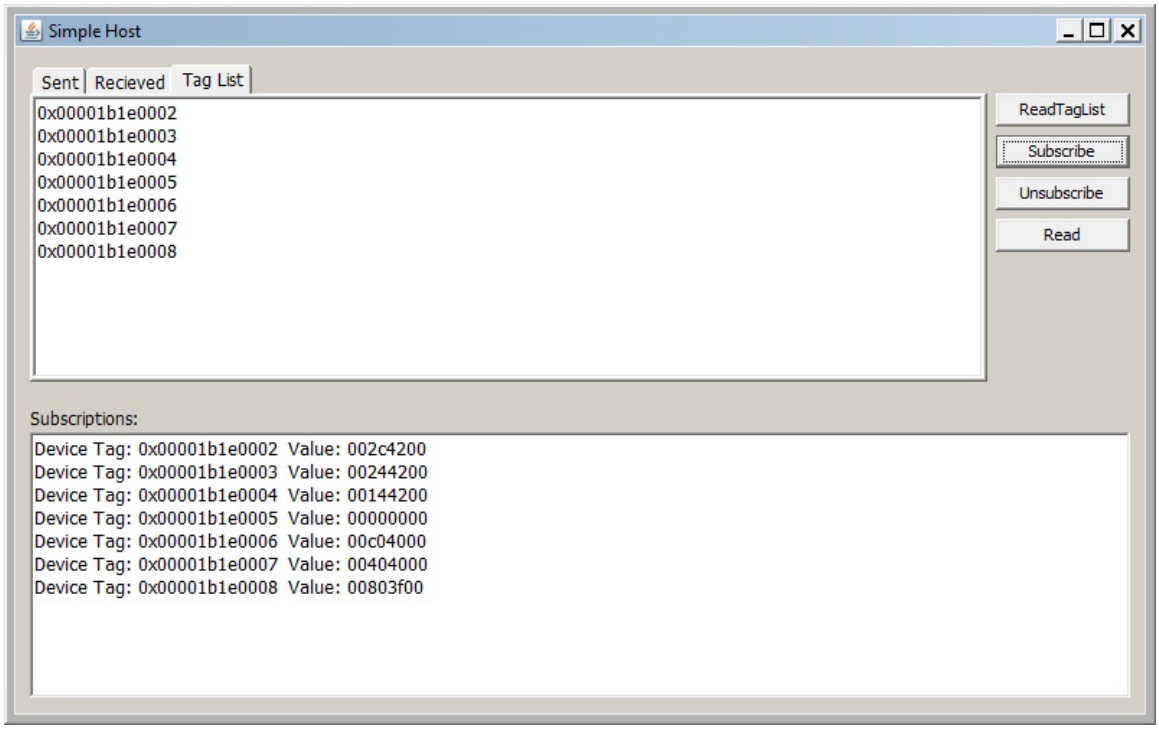

Figure 10 Awia demo host

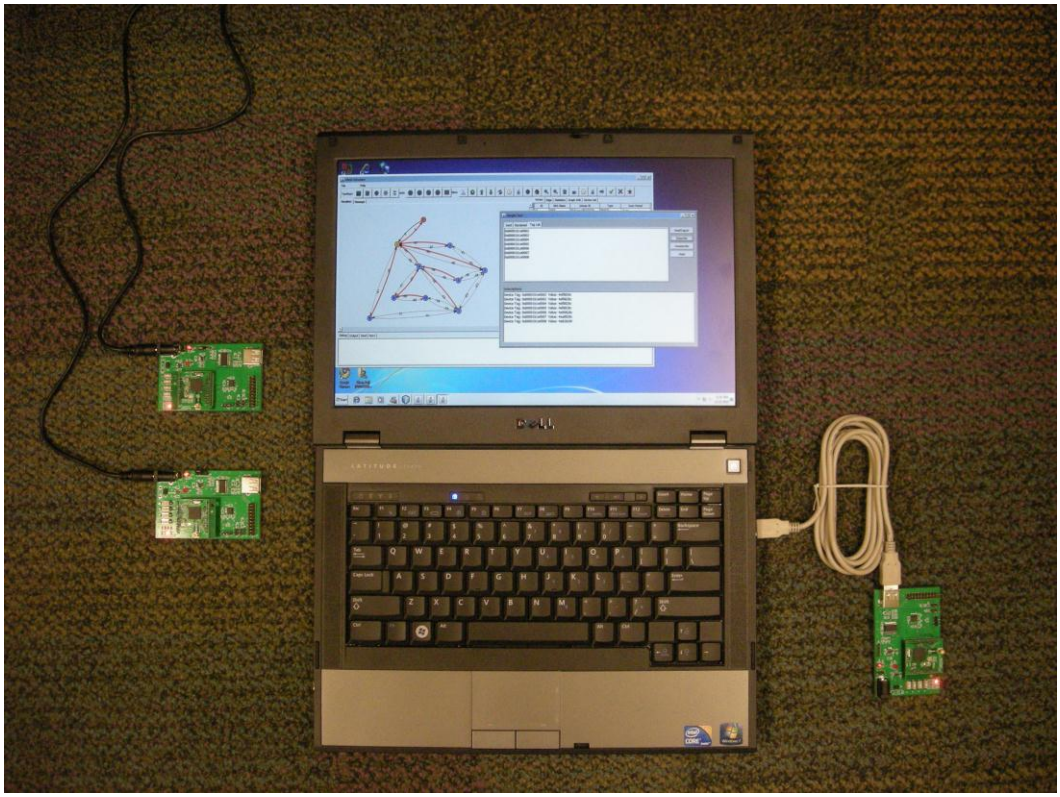

Figure 11 Awia system

4. ADVANCED DEVELOPMENT

Awia Warrior development tool kit provides the interfaces for sensor developers and host application developers.

4.1 Host API

Users could develop their own host applications using the API defined in the following. The demo host is written in JAVA. Users could use any programming language as the communication with the gateway is via standard socket.

4.1.1 Definition

The host application talks to the Awia Gateway via socket. The port number of the gateway is 8890. The port number of the host application is 8891.

The gateway and the host application exchange commands.

The command format is defined as follows:

<byte1(1)><byte2(1)><payload(n)>

- Byte1 (=1): 1 byte, indicating the data exchange protocol. Here we set it as 1 for simple data access.
- Byte2: command number

Payload: size differs for different commands

The command numbers are (payload in the parenthesis):

1. ReadTagList(): Read the list of devices in the network. From host to gateway
2. Subscribe(device tag, update rate, burst command number): Instruct the device to publish data to the gateway cache and instruct the gateway to forward to the host. From host to gateway
3. Unsubscribe(device tag): Stop publishing. From host to gateway
4. Read(device tag): Read data from the gateway cache. From host to gateway
5. TagList(tag list): Report the device list. From gateway to host
6. Data(device tag, data value): Report the data in the gateway cache. From gateway to host
7. Write(device tag, data value): Write data to the device. From host to gateway

The payload parameters are:

- device tag: 6 bytes.
- update rate: 1 byte. The frequency of which the Warrior will send data to the gateway cache. Its value ***v*** is from 0 to 12. The update period = (2^v ×250ms).
- burst command number: 2 bytes. 0x00,0x01: burst from the simulated sensor in the Warrior; 0x00, 0xAA (170): Burst the data from the real sensor.
- data value: 4 bytes for simulated sensor, vary for real sensor data. Maximum size is 72 bytes.

4.1.2 Usage

First the host application establishes the socket connection with the gateway.

The host application should call ReadTagList() constantly to find new sensors in the network.

Once having received TagList(), the host application calls Subscribe() to activate sensor publication. The sensor will then periodically send data to the cache in the gateway, which will forward the data to the host.

Whenever the host wants to view the latest cached data, it sends Read() to the gateway, who will respond by sending Data() back.

4.1.3 Sample code

The demo host application serves as the sample JAVA code. The following is C++ sample code.

Receive

```
	// load and create socket
	WORD myVersionRequest;
	WSADATA wsaData;
	myVersionRequest=MAKEWORD(1,1);
	int err;
	err=WSAStartup(myVersionRequest,&wsaData);

	serSocket=socket(AF_INET,SOCK_STREAM,0);

	SOCKADDR_IN addr;
	addr.sin_family=AF_INET;
	addr.sin_addr.S_un.S_addr=inet_addr(OPC_SERVER_ADDR);
	addr.sin_port=htons(OPC_SERVER_PORT);

	bind(serSocket,(SOCKADDR*)&addr,sizeof(SOCKADDR));
	listen(serSocket,5);

	// listening loop
while(true)
  {
	SOCKET serConn=accept(serSocket,NULL,NULL);
	char receiveBuf[200];
	int len = recv(serConn,receiveBuf,strlen(receiveBuf)+1,0);
```

```
        if(len!=-1)
        {
                char* npdubytes = new char[len];
                memcpy(npdubytes,receiveBuf,len);
                this->m_gwProcessor->addNpduBytes(npdubytes,len);
        }
  }

        // handle commands
        if(*(m_queryBytes+0) == 1)
        {
                m_apiIndex = *(m_queryBytes+1);
                switch((int)m_apiIndex)
                {
                case 1://ReadTagList()
                        break;
                case 2://Subscribe(deviceTag, updateRate)
                        break;
                case 3://Unsubscribe(deviceTag) and Read(deviceTag)
                        break;
                case 4:
                        break;
                case 5://TagList(tag list) - copies a variable amount of deviceTags
into byte array.
                        m_tagListLength = m_queryLength - 2;
                        m_tagList = new char[m_tagListLength];
                        memcpy(m_tagList,m_queryBytes+2,m_tagListLength);
                        break;
                case 6://Data(deviceTag, dataValue)
                        if(len<12) m_isenough = 0;
                        else if(len>12)
                        {m_istoomuch = 1; m_isenough = 1;
                        m_remaindata = new char[len-12];
                        memcpy(m_remaindata, m_queryBytes+12,len-12);
                        m_remaindataLength = len-12;
                        }
                        else {m_isenough=1; m_istoomuch=0;};
                        if(m_isenough ==1)
                        {
                        m_queryLength = 12;
                        m_deviceTag = new char[devicetagLength];
                        memcpy(m_deviceTag,m_queryBytes+2,devicetagLength);
                        m_dataValueLength = m_queryLength - 8;
                        m_dataValue = new char[m_dataValueLength];
                        memcpy(m_dataValue,m_queryBytes+8,m_dataValueLength);
                        }
                        break;
                }
        }
```

Send

```
        // Form command message: ReadTagList() - 2 bytes
        m_queryBytes = new char[2];
        m_queryLength = 2;
        *(m_queryBytes+0) = 1;
        *(m_queryBytes+1) = index; // =1
```

```
m_apiIndex = index;

// Send command
int err;
WORD versionRequired;
WSADATA wsaData;
versionRequired=MAKEWORD(1,1);
err=WSAStartup(versionRequired,&wsaData);


SOCKET clientSocket=socket(AF_INET,SOCK_STREAM,0);
SOCKADDR_IN clientsock_in;
clientsock_in.sin_addr.S_un.S_addr=inet_addr(GW_ADDR);
clientsock_in.sin_family=AF_INET;
clientsock_in.sin_port=htons(GW_PORT);

connect(clientSocket,(SOCKADDR*)&clientsock_in,sizeof(SOCKADDR));

send(clientSocket, message, len, 0);
closesocket(clientSocket);
WSACleanup();
```

4.2 Device API

As shown in Figure 7, an AwiaTech WirelessHART® stack enabled device can use serial port (COM or virtual COM over USB) to communicate with the Awia Warrior stack. The serial port is configured as follows:

- Parity: None
- DataBit: 8
- StopBit: 1
- Baudrate: 1200
- FlowControl: None

Note, please contact AwiaTech to customize above settings, such as higher baud rates.

The message data format is defined in Table 1.

Preambles	0x86	LEN	SEQ	Message Payload	CRC

Table 1 Command Format

The preambles are a fixed sequence of bytes (0xFF 0xFF 0xFF 0xFF 0xFF) to indicate the start of the command. Byte “0x86” is used as a delimiter. Then it is the *LEN* field, which is 1 byte long and includes the number of bytes from *SEQ* to *CRC* inclusively. The *SEQ* field indicates the sequence number of this packet, which usually starts from 0 and is incremented by one every time a packet is sent, wrapping around 255 to 0. *Message Payload* contains the HART command.

To detect possible data corruption, a 2-byte *CRC* field is added at the end of the packet. We use CCITT-16 checksum algorithm to calculate this field (http://en.wikipedia.org/wiki/Cyclic_redundancy_check). The CRC calculation is applied from *LEN* to *Message Payload* inclusively.

The format for *Message Payload* is defined in Table 2. It follows the transport layer and application layer of the WirelessHART standard. Please refer to Network Management Specification (HCF_SPEC_085) for more details.

Transport Byte	Device Status	Extended Status	Command No	Byte Count	Data

Table 2 Message Payload Format

Transport Byte is a one-byte field. Set it to 0 for command request; set it to 0x40 for command response. *Device Status* and *Extended Status* follow the HART standard. Both are set to 0 in normal cases. *Command No* is a two-byte field (in big endian), which is followed by the one-byte *Byte Count* of the command *Data* and then the actual command *Data*.

Via the serial connection, a receiver sees a sequence of bytes from the sender. It will search for the preamble, the 5 0xFF bytes, to determine the start of a message. All the bytes, if any, from the end of a previous message to the start of the next preamble are discarded.

Some of the commands are defined in the following.

4.2.1 Write Data (Command 170)

This is an AwiaTech device specific command. This command writes a series of bytes whose format is defined by the device manufacturer. It only has the command request version, used by both the stack and the sensor.

In this command, *Cmd No* is 170 in big endian. *Cmd Data* is the data bytes. Its allowed size is from 0 to 72.

Command Data Bytes (Request)

Byte	Format	Description
0-n	Unsigned-8	Sensor data string

4.2.2 Set Network ID (Command 773)

This is a standard WirelessHART command. The stack is the receiver of this command. This command configures the device to recognize the proper Network ID. Upon power on, the device starts to capture broadcasts from a network with such ID.

In this command, *Cmd No* is 773 in big endian. *Cmd Data* is the two byte network ID in big endian.

Command Data Bytes (Request and Response)

Byte	Format	Description
0-1	Unsigned-16	Network ID

4.2.3 Write Join Key (Command 961)

This is a standard WirelessHART command. The stack is the receiver of this command. This command is used to set the join key of the device. During the join process, the join key is used as an authentication token for this device to the network manager. The device will use the join key to encipher its data; and the network manager uses the same join key to decipher.

In this command, *Cmd No* is 961 in big endian. *Cmd Data* is the 16 byte join key.

Command Data Bytes (Request and Response)

Byte	Format	Description
0-15	Unsigned-128	Key value

4.3 Putting it together

4.3.1 Awia Warrior system as a data transmission medium

Just like other network infrastructure, Awia Warrior system serves as a medium for two peers to exchange data, in this case the host application and the devices in the field. For this purpose, the Awia system receives data from the host and delivers to the sensor, and vice versa.

From host to sensor: The host application calls Write() with a string of bytes to be sent to the sensor. The Awia system will route the data to the Warrior associated with that sensor, then sends Command 170 to the sensor with the same string of bytes.

From sensor to host: The sensor sends Command 170 to the Warrior with a string of bytes. The Warrior will route the data to the gateway. After having received an update from the Warrior, the gateway will put it in the cache and forward to the host by calling Data(). Occasionally, the host could call Read(), and in response the gateway calls Data() to forward the string of data in the cache to the host application.

The host application should first call Subscribe() with burst command number set to 170. Calling Subscribe() with burst command number set to 1 will activate the simulated sensor in the Warrior and the actual sensor data will not be transmitted to the gateway. It's up to the user to define the protocol between the host and the sensor. In other words, the host and sensor should manage the acknowledgement and retry. It is possible that the wireless network lose a

message. In addition, since there is a limit of 72 bytes on the data, the user should handle large size data, to break them up in the sender and reassemble in the receiver.

Table 3 shows the message field values for Command 170. In this example the data from host to sensor is “0x01 0x23 0x45 0x67 0x89” and the data from sensor to host is “0xAB 0xCD 0xEF”.

	Warrior -> Sensor	Sensor->Warrior
Preambles	0xFF 0xFF0 xFF 0xFF 0xFF	0xFF 0xFF0 xFF 0xFF 0xFF
0x86	0x86	0x86
LEN	0x0E	0x0C
SEQ	0x00	0x00
Transport Byte	0x00	0x40
Device Status	0x00	0x00
Extended Status	0x00	0x00
Command No	0x00 0xAA	0x00 0xAA
Byte Count	0x05	0x03
Data	0x01 0x23 0x45 0x67 0x89	0xAB 0xCD 0xEF
CRC	0xBB 0xFF	0xAE 0xAD

Table 3 Command 170 message field values

The following is the code snippet for calculating CRC.

```
/* update the crc for each byte x */
void crc_ccitt_update(unsigned short *crc, unsigned char x)
{
	unsigned short crc_new = (unsigned char)(*crc >> 8) | (*crc << 8);
	crc_new ^= x;
	crc_new ^= (unsigned char)(crc_new & 0xff) >> 4;
	crc_new ^= crc_new << 12;
	crc_new ^= (crc_new & 0xff) << 5;
	*crc = crc_new;
}

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned short crc = 0xFFFF;  // the initial crc
```

```
	unsigned char frame[200] = { 0x0C, 0x00,
	                             0x40, 0x00, 0x00,
	                             0x00, 0xAA,
	                             0x03,
	                             0xAB, 0xCD, 0xEF};
	unsigned char i;
	unsigned char length = 9;

	for(i = 0; i < length; i++) {
		crc_ccitt_update(&crc, frame[i]);
	}

	frame[length++] = (unsigned char)((crc >> 8) & 0xff);
	frame[length++] = (unsigned char)(crc & 0xff);

	return 0;
}
```

4.3.2 Awia Warrior as a WirelessHART device

AwiaTech Warrior could be experimented as a standalone WirelessHART device. Generally, a WirelessHART® device has to be configured before being deployed. This step is required for normal operation and security concerns. AwiaTech WirelessHART® stack is designed to require minimal configuration. There are only two required parameters for normal operations: first, the network that the device intends to join; second, the join key. The user could use the serial communication to write the join key and network ID into the Warrior. With these pieces of information, a device equipped with AwiaTech WirelessHART® stack can join the designated WirelessHART® network automatically upon power-on.

4.3.3 Awia Warrior system as a WirelessHART evaluation tool

A major purpose of the AwiaTech Warrior system development kit is for users to play with it and learn about WirelessHART. A WirelessHART network could be set up right out-of-the-box with the parts in the tool kit, as is described in Section 3.2. The user is encouraged to run Wi-Analys from HART Foundation to view the WirelessHART network traffic.

5. APPENDICES

5.1 Glossary of terms

4-20mA A point-to-point or multi-drop circuit mainly used in the process automation field to transmit signals from instruments and sensors in the field to a controller. It sends an analog signal from 4 to 20 mA that represents 0 to 100% of some process variable. As a current loop signal, 4-20 mA also powers the sensor transmitter on the same wire pair, and 4-20mA provides more resistance to interference than a voltage-based line.

Analog channel continuously varying electrical signal connecting a field device to the remainder of the data acquisition or control system NOTE 1 Some field devices support multiple analog channels (input or output) NOTE 2 Each analog channel transmits a single dynamic variable to or from the field device.

Application function or data structure for which data is consumed or produced

Broadcast the process of sending a PDU to all devices that are connected to the

network and are able to receive the transmission

Burst-mode device This is a device which provides a digital response carrying measurement or other data, at regular intervals, without the data being specifically requested, i.e., this device normally functions as an independently broadcast device. A Burst-Mode Device is defined to be a Slave Device with burst capability (hence the use of the word “mode” in describing the device type). When such a mode is enabled, the Slave Device is said to “be in burst mode”.

Channel RF frequency band used to transmit a modulated signal carrying packets

Channel blacklisting method of eliminating an RF channel from usage

Channel hopping regular change of transmit or receive frequency to combat interference and fading

Client a) an object which uses the services of another (server) object to perform a task b) an initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

Connection A data structure associated with graph routing that contains an ordered pair of Network Devices.

Cyclic term used to describe events which repeat in a regular and repetitive manner

Data-Link Layer Layer 2 in the OSI Basic Reference Model. This layer is responsible for the error-free communication of data. The Data-Link Layer defines the message structure, error detection strategy and bus arbitration rules.

Device any entity containing an implementation of Type 20 fieldbus

Field device network device that is connected to the process or to a plant equipment NOTE It directly connects to the sensor or actuator or performs process control function and it is directly connected to the physical layer of Type 20.

Frame A Data-Link Layer "packet" which contains the header and trailer information required by the physical medium. That is, Network Layer packets are encapsulated to become frames.

Frequency channel allocation of the frequency spectrum in a given frequency range

Gateway network device containing at least one host interface such as serial or Ethernet, acting as ingress or an egress point enabling communication between host applications and field devices

Graph routing structure that forms a directed end-to-end connection between network devices

Graph ID identifier used to indicate a specific graph entry

Handheld host application residing on a portable device

Host One of (possibly) several applications that can be executed sequentially or simultaneously on a master.

Join process by which a network device is authenticated and allowed to participate in the network NOTE A device is considered Joined when it has the network key, a network manager session and a normal (not join) superframe and links.

Join key security key that is used to start the join process

Latency time it takes for a packet to cross a network connection, from sender to receiver

Lease A lease is an agreement between the host and the WirelessHART Gateway to share a resource for a future period of time; after which the resources can be reallocated for other purpose.

Link full communication specification between adjacent devices in a network and it includes the communication parameters necessary to move a DLPDU one hop. NOTE: A Link is a function of source and destination address pairing, slot and channel offset assignment, direction of communication, dedicated or shared communication, and type. Links are assigned to superframes as part of the scheduling process.

Logical link control higher of the two data-link layer levels NOTE This level handles error detection, flow control, framing, and addressing.

Loop current value measured by a mA in series with the field device. NOTE: The loop current is a near DC analog 4-20 mA signal used to communicate a single value between the control system and the field device. Voltage mode devices use "Volts DC" as their engineering units where "loop current" values are used.

Maintenance port interface in the network device that is used for provisioning. NOTE: That means to write the join key, network ID and to monitor the join process

Master a device that initiates communication activity by sending request APDU to a device and expecting a response PDU

Medium access control lower of the two data-link layer levels NOTE This level controls the access to the communication channel.

Neighbor adjacent node in the network such that the receive signal level (RSL) from it suggests that the communication is possible in at least one direction

Neighbor table list of neighbors that a DLE has knowledge of NOTE It also stores the properties of the neighbor.

Network series of nodes connected by some type of communication medium NOTE The connection paths between any pair of nodes can include repeaters, routers and gateways.

Network device device with a direct physical layer connection to the network.

Network ID identifier used to indicate a network to which all intercommunicating devices are connected. NOTE: A device connected to one network can not send a PDU to another device connected to a different network.

Network manager entity that is responsible for configuration of the network, scheduling communication between network devices, management of the routing tables and monitoring and reporting the health of the network. NOTE: There is one and only one network manager per instance of Type 20 network. Although the network manager need not have a direct physical layer connection, it still has a Unique address.

Node addressable logical or physical device attached to the network

Physical Layer Layer 1 in the OSI Basic Reference Model. The Physical Layer is responsible for transmission of the raw bit stream and defines the physical (e.g., mechanical, electrical) connections and signaling parameters for devices.

Security Manager An application that manages the Network Device's security resources and monitors the status of the network security

Slot fixed time interval that may be used for communication between neighbors

Superframe collection of slots repeating at a constant rate; each slot has a link associated with it

Time division multiple access medium access control technique that uses time slots where communications between devices can occur. NOTE: It provides collision free, deterministic communications.

Transaction exchange of related, consecutive frames between two peer medium access control entities, required for a successful transmission. NOTE: A transaction consists of either (a) a single PhPDU transmission from a source device, or (b) one PhPDU from the source device followed by a second, link-level acknowledgement PhPDU from the destination device.

Transport Layer The Transport Layer provides reliable data transfer between two deices. Communication is transparent in that detailed low level knowledge of the communication is not required.

5.2 Related Documents

WirelessHART User Guide. HCF_LIT-84

Coexistence Test Plan. HCF_LIT-85

Approved IEEE 802.15.4 Transceivers. HCF_LIT-088

IEEE STD 802.15.4-2006. Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). 2006

AwiaTech Corporation © 2011. All rights reserved.

AwiaTech and Awia Warrior are trademarks of AwiaTech Corporation, registered in the U.S. and other countries. WirelessHART is a trademark of HART Communication Foundation. Other company and product names mentioned herein may be trademarks of their respective companies.

Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. AwiaTech assumes no responsibility with regard to the performance or use of these products. All understandings, agreements, or warranties, if any, take place directly between the vendors and the prospective users. Every effort has been made to ensure that the information in this manual is accurate. AwiaTech is not responsible for printing or clerical errors.

The product described in this manual incorporates copyright protection technology that is protected by method claims of certain U.S. patents and other intellectual property rights owned by AwiaTech Corporation and other rights owners. Use of this copyright protection technology must be authorized by AwiaTech Corporation. Reverse engineering or disassembly is prohibited.