



AwiaTech WirelessHART™ Evaluation Kit Manual

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.

Table of Contents

1. User Guide.....	5
1.1 What is WirelessHART™?.....	5
1.2 AwiaTech WirelessHART Products.....	7
1.3 What is in the Kit?.....	7
1.4 Install Software on PC.....	8
1.5 Activation.....	15
1.6 Start the Network.....	16
1.7 Explore and Evaluate.....	17
1.8 Device Description.....	18
1.9 Menu Description.....	19
1.10 Trouble Shooting.....	20
2. Development Guide.....	24
2.1 System Architecture.....	24
2.2 Host API.....	25
2.2.1 Definition.....	25
2.2.2 Usage.....	26
2.2.3 Sample code.....	27
2.3 Device API.....	29
2.3.1 From Warrior to Sensor: Host Data – Burst (Command 170).....	30
2.3.2 From Warrior to Sensor: Host Data – Non-Burst (Command 171).....	30
2.3.3 From Sensor to Warrior: Sensor Data – Burst (Command 170).....	30
2.3.4 From Sensor to Warrior: Sensor Data – Non-Burst (Command 171).....	30
2.3.5 From Sensor to Warrior: Set Network ID (Command 773).....	30
2.3.6 From Sensor to Warrior: Write Join Key (Command 961).....	31
2.4 Putting It All Together.....	31
2.4.1 Use Awia Net as a data transmission medium - Burst.....	31
2.4.2 Use Awia Net as a data transmission medium – Non-Burst.....	33

2.4.3	Use Awia Warrior in a WirelessHART device	34
2.4.4	Use Awia Net as a WirelessHART evaluation tool.....	34
3.	Technical Specification.....	35
4.	Glossary of Terms.....	36
5.	Index	38

1. USER GUIDE

1.1 What is WirelessHART™?

WirelessHART™ (Highway Addressable Remote Transducer) is an open-standard wireless networking technology developed by the HART Communication Foundation for industrial automation. It is also the first international wireless standard for industrial automation, IEC 62591. WirelessHART uses a time synchronized, self-organizing, and self-healing mesh network architecture. Its current version (HART 7) supports operation in the 2.4 GHz ISM Band on IEEE 802.15.4 standard radios.

A typical WirelessHART network consists of the following components:

- One or more WirelessHART field device(s)
- One WirelessHART Gateway (including Network Manager)
- One WirelessHART Access Point

It may also contain

- WirelessHART Adaptor
- WirelessHART Router Device

Figure shows a WirelessHART network example.

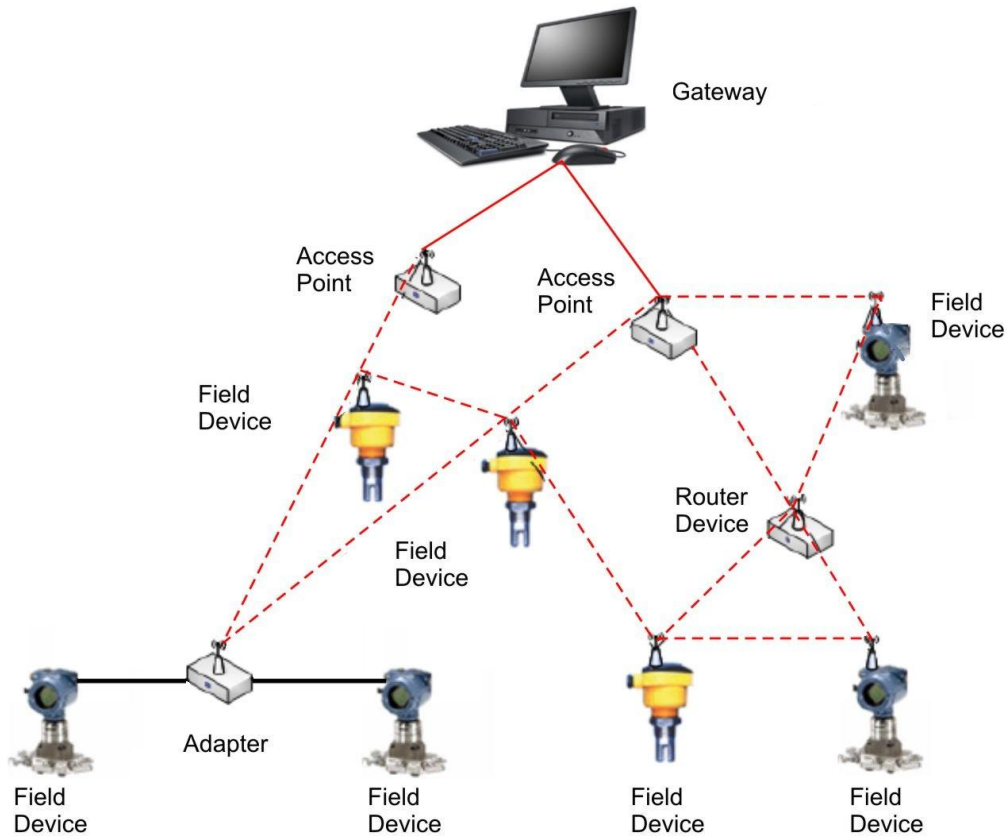


Figure 1 A WirelessHART network example

Figure 2 illustrates the architecture of WirelessHART protocol stack according to the OSI 7-layer communication model. It includes five layers: physical layer, data link layer, network layer, transport layer and application layer.

OSI Layer	Function	HART	
Application	Provides the User with Network Capable Applications	Command Oriented. Predefined Data Types and Application Procedures	
Presentation	Converts Application Data Between Network and Local Machine Formats		
Session	Connection Management Services for Applications		
Transport	Provides Network Independent, Transparent Message Transfer	Auto-Segmented transfer of large data sets, reliable stream transport, Negotiated Segment sizes	
Network	End to End Routing of Packets. Resolving Network Addresses	Power-Optimized, Redundant Path, Self-Healing Wireless Mesh Network,	
Data Link	Establishes Data Packet Structure, Framing, Error Detection, Bus Arbitration	A Binary, Byte Oriented, Token Passing, Master/ Slave Protocol.	Secure & Reliable, Time synched TDMA/CSMA, Frequency Agile with ARQ
Physical	Mechanical / Electrical Connection. Transmits Raw Bit Stream	Simultaneous Analog & Digital Signaling. Normal 4-20mA Copper Wiring	2.4GHz Wireless, 802.15.4 based radios, 10dBm Tx Power
		Wired FSK/PSK & RS485	Wireless 2.4GHz

Figure 2 Architecture of HART Communication Protocol

1.2 AwiaTech WirelessHART Products

The Awia Warrior stack fully conforms to the WirelessHART standard. Any functionality or interface described in this document that falls under the scope of the WirelessHART standard will contain the same description as the WirelessHART standard.

Throughout this document, Awia Warrior refers to a hardware module with wireless radio communication capabilities and running AwiaTech's WirelessHART network stack, thus it acts as a node to the network; Awia Vanguard refers to AwiaTech's WirelessHART Gateway, which includes Awia Commander, AwiaTech's WirelessHART Network Manager. Both Awia Vanguard and Awia Commander are software components. Awia Captain refers to AwiaTech's Access Point. Finally, AwiaNet refers to a functioning WirelessHART network consisting of at least one of each component mentioned above. A complete list of AwiaTech product definitions is listed in Table 1.

WirelessHART Component	Corresponding AwiaTech Product Name	AwiaTech Product Form
Network Stack	Awia Warrior Stack	Software
Device Module	Awia Warrior Module	Hardware + Software
Gateway	Awia Vanguard	Software
Network Manager	Awia Commander (as part of Awia Vanguard)	Software
Access Point	Awia Captain	Hardware + Software
Network	AwiaNet	Hardware + Software

Table 1 AwiaTech Product Definition

1.3 What is in the Kit?

First of all, please make sure your evaluation kit is complete and familiarize yourself with the content.

This evaluation kit contains:

- One Awia Captain (Access Point)
- Five Awia Warrior Modules, plus
 - AA Battery cases (Batteries not included)
- One Software CD containing

- Awia Vanguard (Gateway)
- User Manual
- One USB cable (for connecting the Awia Captain to a PC)
- One printed copy of user manual

1.4 Install Software on PC

Before you start an AwiaNet, please make sure the Awia Vanguard software is installed on a PC that meets the following requirements.

Minimum System Requirements:

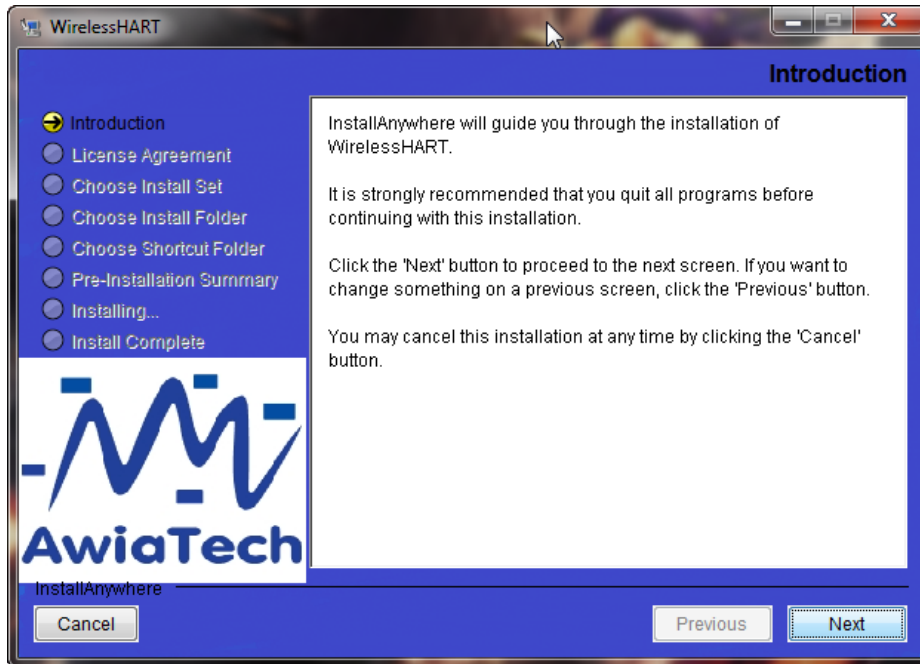
- Windows XP, Windows Vista, or Windows 7 (for both 32-bit and 64-bit)
- 200 MB available in the hard disk
- 1GB RAM
- An available USB port
- CD-ROM, 2x or higher

Recommended System Requirements:

- Windows 7 (for both 32-bit and 64-bit)
- 200 MB available in the hard disk
- 2GB RAM
- An available USB port
- CD-ROM, 2x or higher

Then follow the steps below to ensure a successful installation:

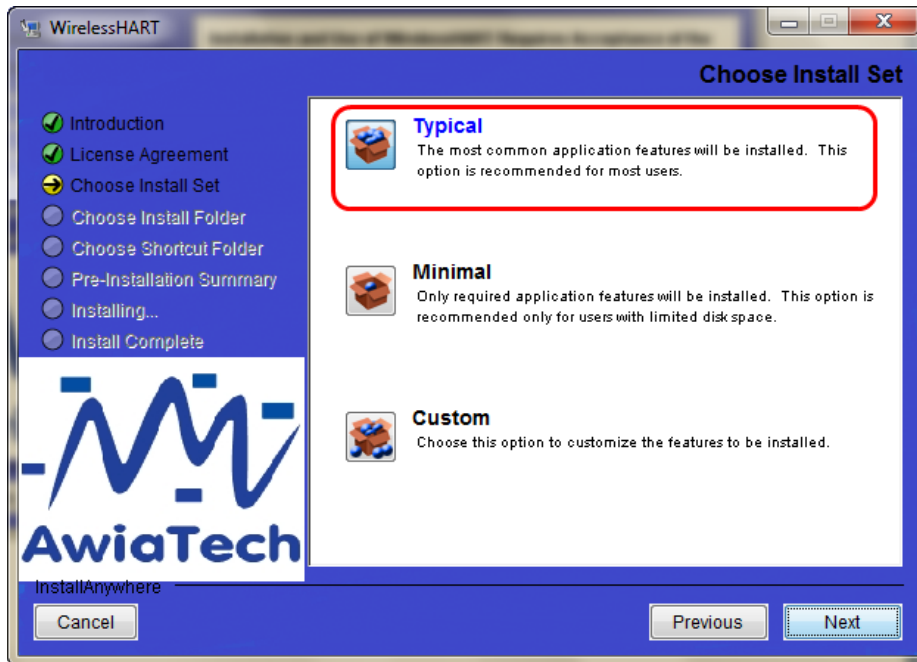
1. Double-click the WirelessHART installation file (setup.exe) on the CD
2. After the following window appears, click “Next”



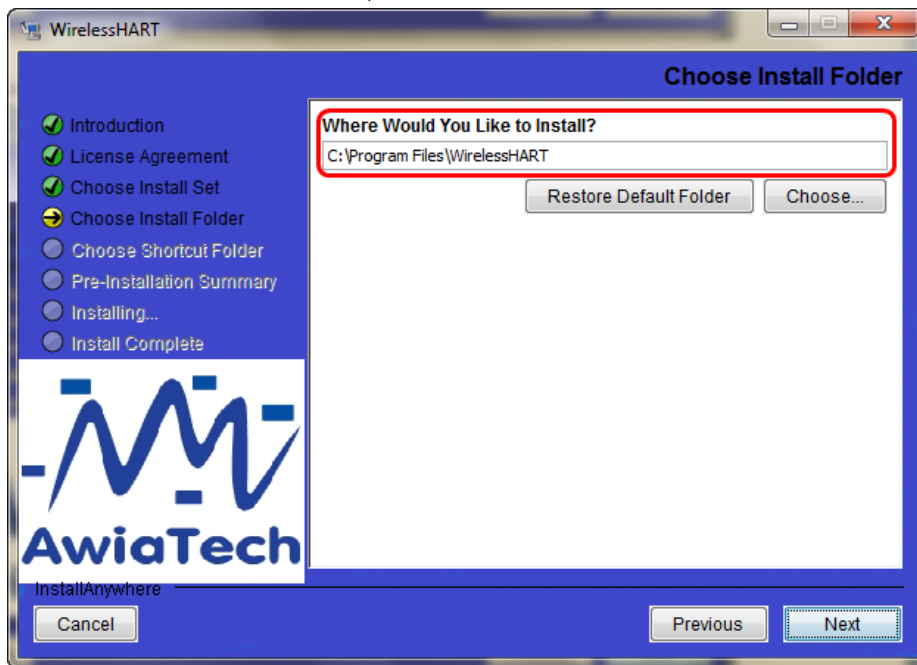
3. Read the End User License Agreement. If you agree, select “I accept the terms of the License Agreement” and click “Next”.



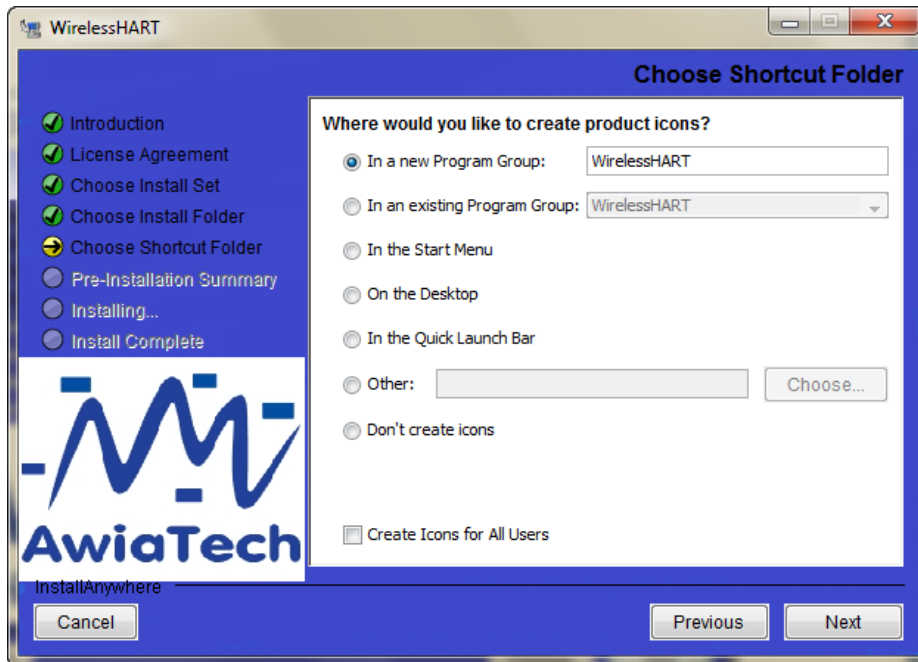
4. After the following window appears, select “Typical”, and click “Next”



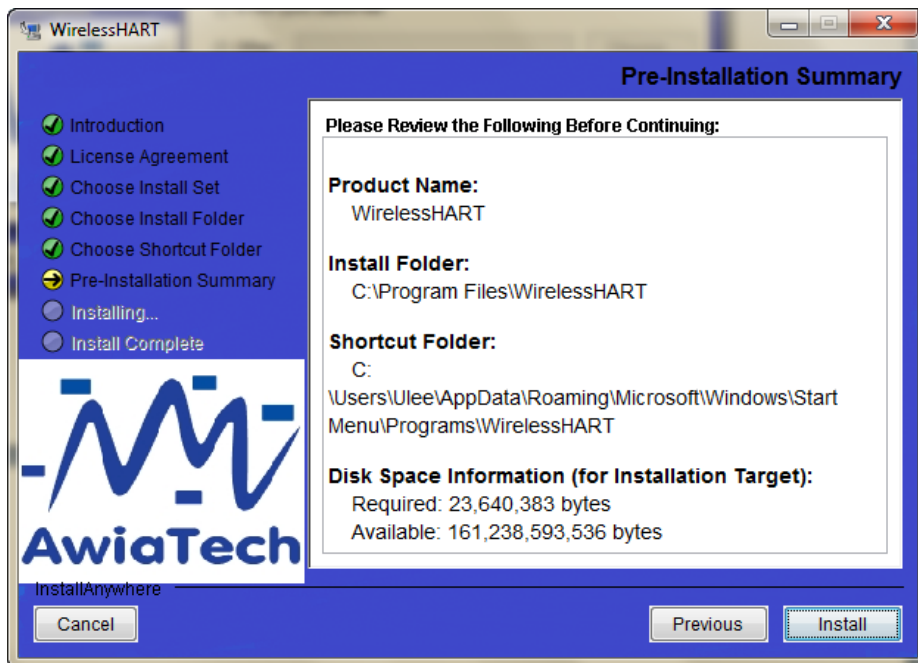
5. Select the installation folder, and click “Next”



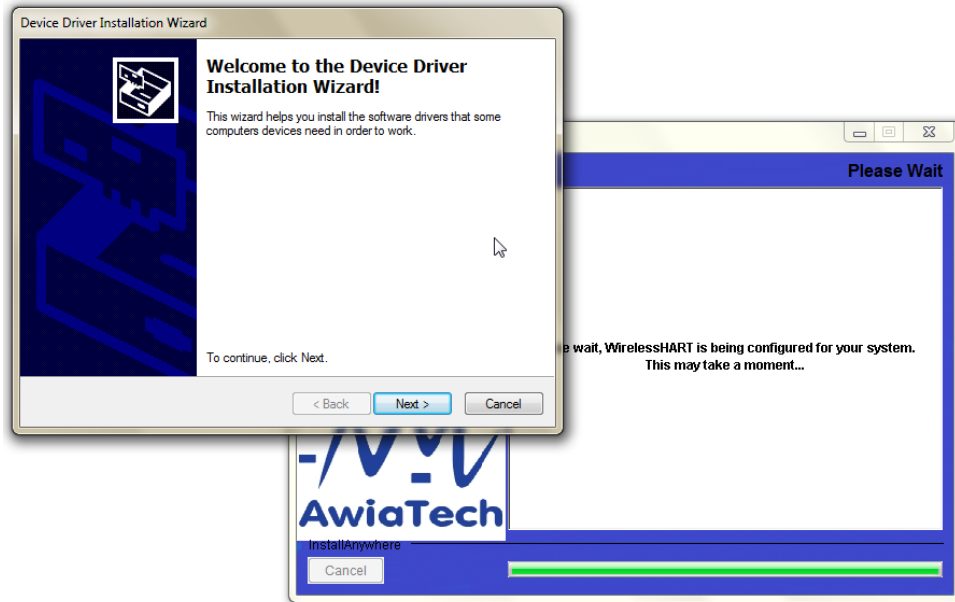
6. After the following window appears, click “Next”



7. After the following window appears, all information needed for the installation has been collected. Please click “Install”



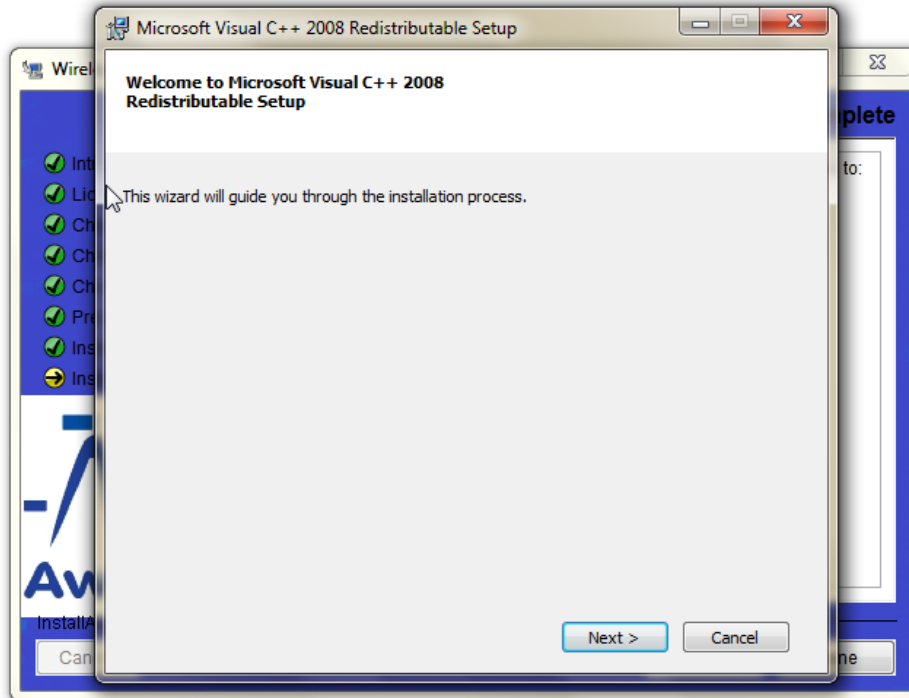
8. During the installation, a serial-port-over-USB device driver is installed. After the following “Device Driver Installation Wizard” window appears, click “Next”



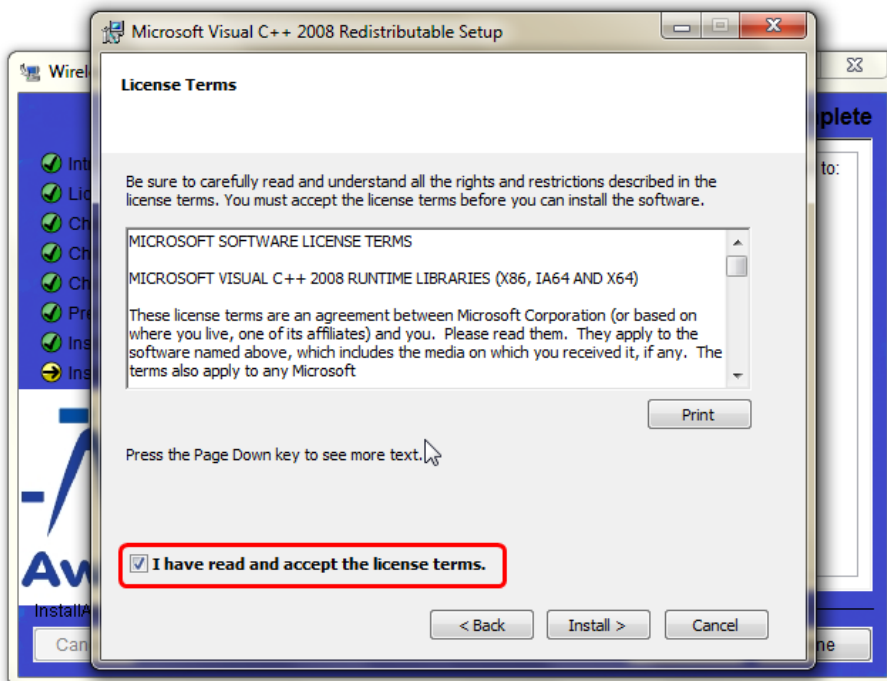
9. Click “Finish” after completing the device driver installation



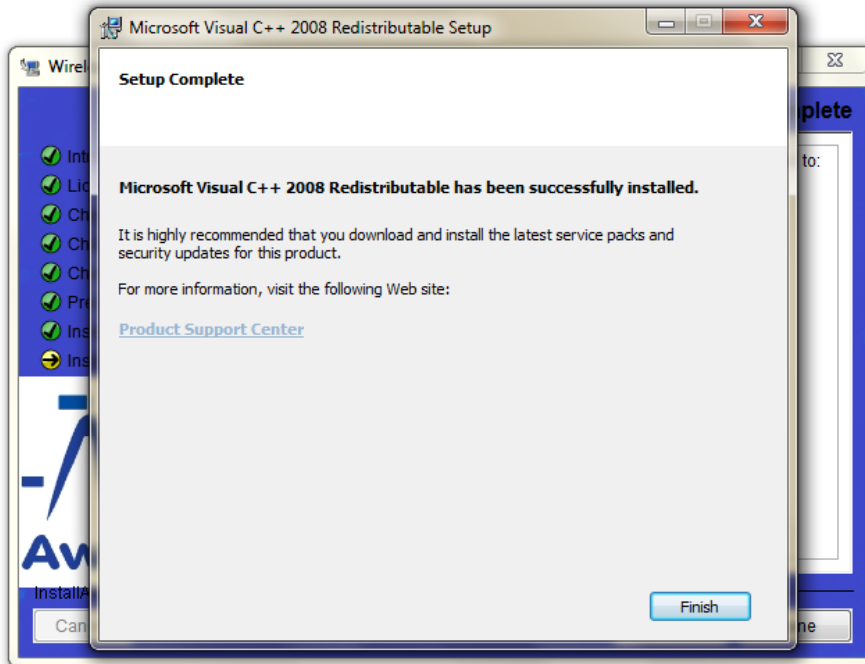
10. After the “Microsoft Visual C++ 2008 Redistributable” window appear, click “Next”



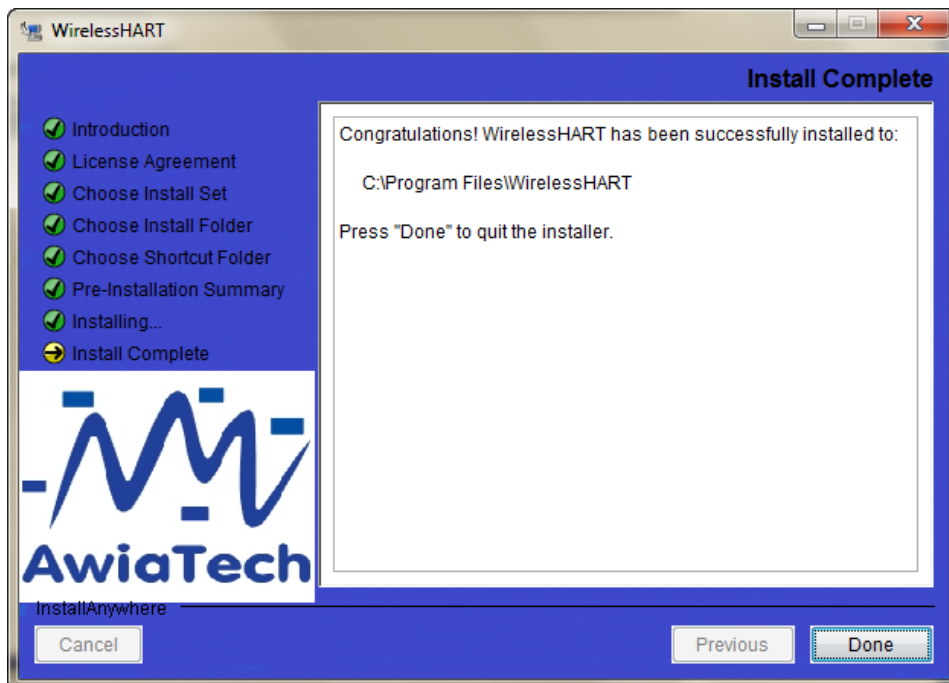
11. Read the Microsoft Software License Terms. If you agree, select “I have read and accept the license terms”, and click “Install”



12. Click “Finish” after “Microsoft Visual C++ 2008 Redistributable” setup completes



13. Click "Done" to finish the WirelessHART installation



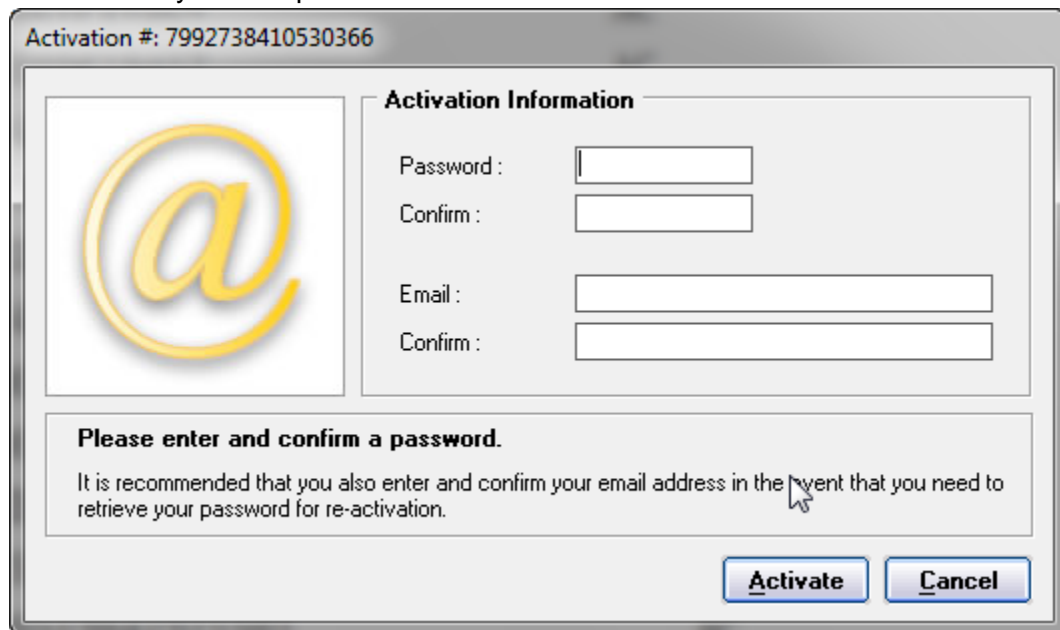
1.5 Activation

1. The AwiaVanguard software needs activation at its first running. The following window will pop up, and please enter the activation code. Make sure the PC has internet access, then click 'Next'.



The 'Product Activation' dialog box features a title bar with the text 'Product Activation'. On the left side, there is a square icon containing two keys, one silver and one gold. To the right of the icon, the text 'Please enter your Activation Code' is displayed above a single-line text input field labeled 'Activation Code :'. At the bottom of the dialog, there are four buttons: 'Request Eval Code' (disabled), 'Proxy Settings', 'Next >', and 'Cancel'.


2. If you enter an unused activation code, simply enter the password and email for this code, and then click 'Activation'. The wirelessHART software will be activated on your computer.



The 'Activation Information' dialog box has a title bar with the text 'Activation #: 7992738410530366'. On the left, there is a large yellow '@' symbol icon. To the right, under the heading 'Activation Information', there are four input fields: 'Password :', 'Confirm :', 'Email :', and 'Confirm :'. Below these fields, a text box contains the instruction: 'Please enter and confirm a password. It is recommended that you also enter and confirm your email address in the event that you need to retrieve your password for re-activation.' At the bottom right, there are two buttons: 'Activate' and 'Cancel'.

3. If you enter a used activation code, the following window will pop up. That means you can no longer use this activation code on your computer.

Activation #: 9TDWARXBJ5VGMZ7Y



Re-Activation Information

Current Password :

New Password :

Confirm :

This code has already been activated !

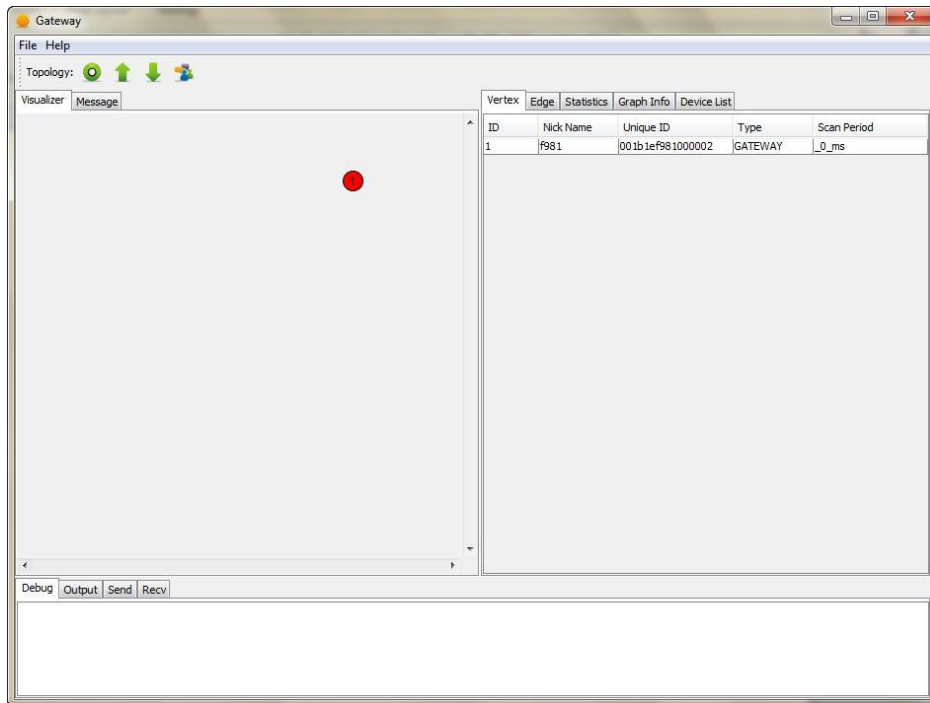
Please confirm that you wish to re-activate it by entering your current password and then a new password.

[Forgot your Password ?](#)

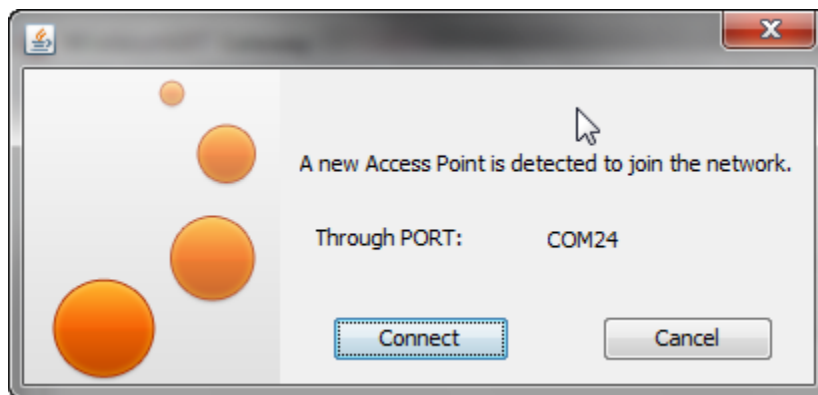
1.6 Start the Network

This section describes the steps to start an AwiaNet.

- On the PC, launch the Awia Vanguard



- Connect Awia Captain to the PC via a USB cable (included in the kit). Awia Captain is powered through the USB cable. You shall see a window similar to the one below. Click on the Connect button.



- Power on the Awia Warriors.
- Start the host application.

1.7 Explore and Evaluate

As its name implies, this evaluation kit allows you to explore and evaluate various kinds of scenarios in the field. You will notice the topology change on Awia Vanguard. Below are a few suggestions to start with.

1. Turn a single Awia Warrior module on and off
2. Move an Awia Warrior module to a remote location

3. Turn on Awia Warrior modules all at the same time vs. one by one (after the prior module has joined the network)

Notice that sometimes it takes some time for a new device to join. According the WirelessHART standard, it takes the following steps for a new device to join:

- The new device listens on one physical channel at a time for a certain period of time.
- The network devices randomly broadcast advertisements.

Therefore the change of the new device hears an advertisement is arbitrary. To increase the chance, the network device should be given more chances to advertise during network forming phase.

Notice that if you turn a device off, it will take a few minutes for it to disappear from the network display. There is a configurable timeout value. If a device is not communicating after this timeout, a path down failure will be generated and the device will be taken off the network. The device could rejoin before the timeout, in which it is reconfigured.

1.8 Device Description



An Device Picture:

LED Light 1: Power indicator. It is on when power is present

LED Light 2: Communication indicator. It is normally on and blinks when transmitting or receiving

1.9 Menu Description

AwiaVanguard (shown in Figure 1):

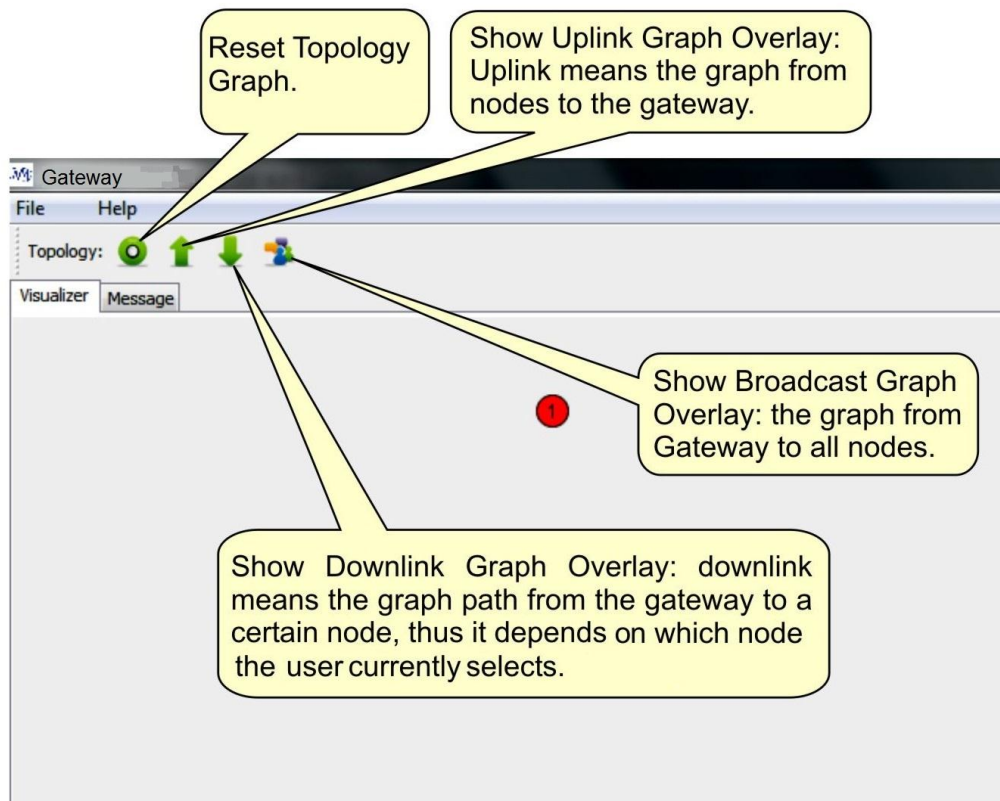


Figure 1. Awia Vanguard Menu Description

Demo Host (shown in Figure 4):

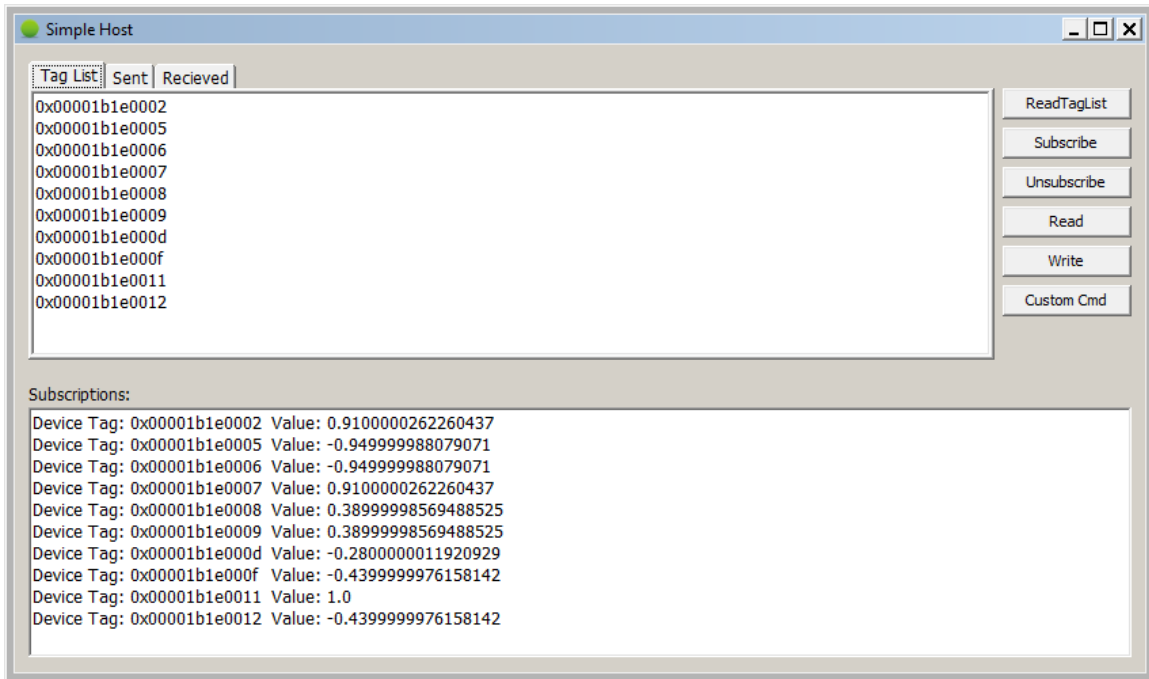


Figure 4. Awia demo host

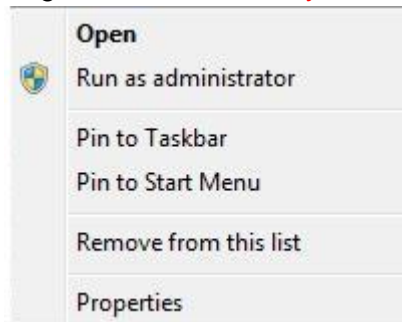
1.10 Trouble Shooting

Due to a vast number of configuration possibilities of the Windows OS on the PC, you may encounter the following issues. For an updated list of trouble shooting, please visit www.awiatech.com for the FAQ section.

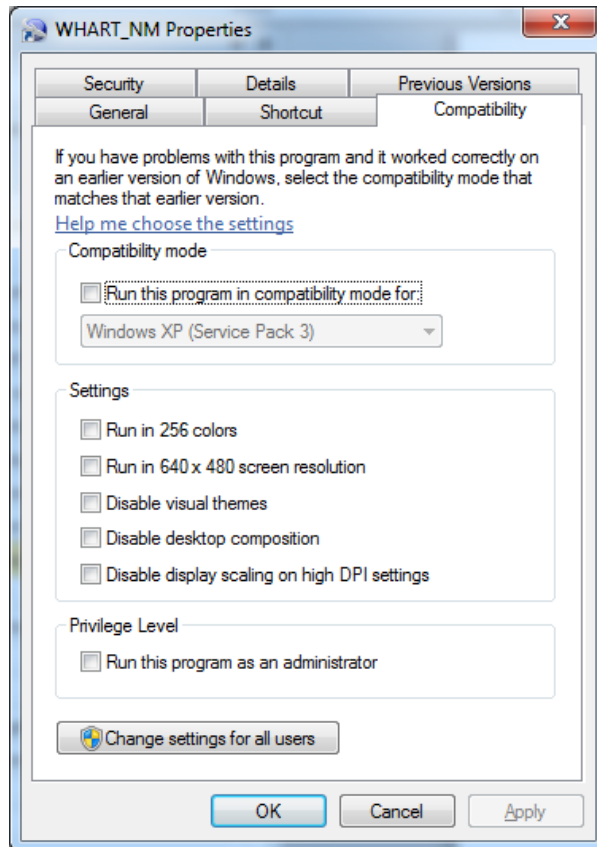
1. Awia Vanguard doesn't start.

- If the PC runs Windows Vista or Windows 7, try running Awia Vanguard as administrator. This can be set by either:

- Right-click the **Gateway Icon** and click **"Run as Administrator"**

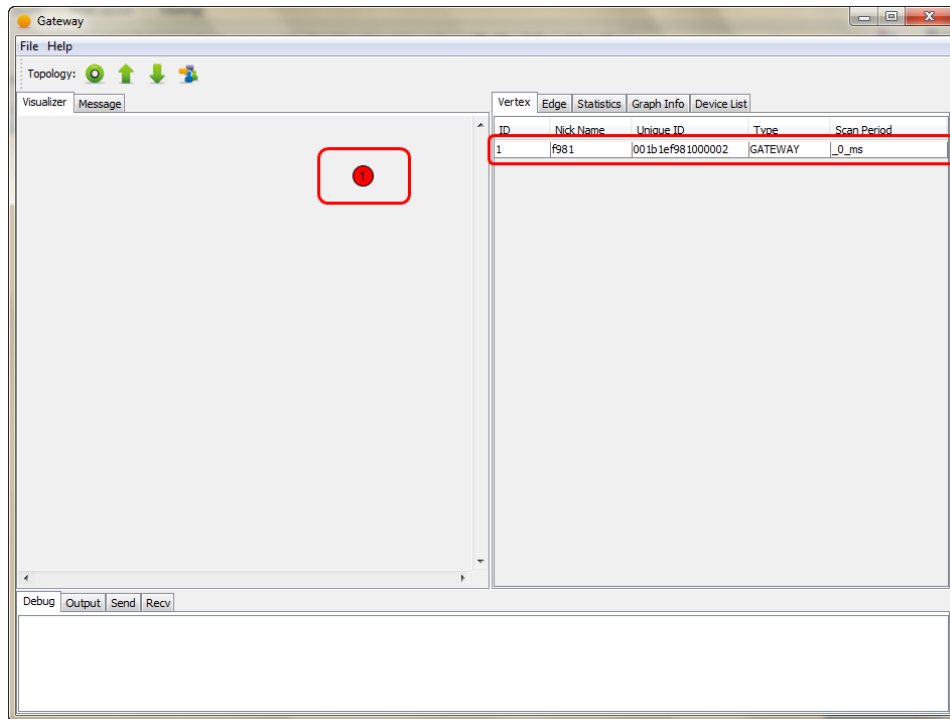


- Or right-click the **Gateway Icon**, click **"Properties"**, click **"Compatibility"** **Tab** and
- **Make sure the software is installed correctly.**



2. How do I know the WirelessHART Gateway is launched correctly?

First, make sure you see GATEWAY in the “Visualizer” graph and “Vertex” table in the GUI of WirelessHART as shown below



Second, make sure you have both “WirelessHART-Gateway.exe” and “WirelessHART-Daemon.exe” processes in the Windows Task Manager. If you don’t see them, exit AwiaVanguard and restart it.

3. What if I cannot find the GATEWAY in the GUI of WirelessHART mentioned in Question 2?

Make sure “Microsoft Visual C++ 2008 Redistributable” is installed correctly. Check in the Control Panel-> Programs.

4. How to make sure the serial port over USB driver is installed correctly?

Test your AP device. It cannot be “unknown device” in the Device Manager if the driver is installed correctly.

5. What if Access Point cannot be detected?

Check Question 2 and 4 first. If it still does not work, contact us through the email given in www.awiatech.com.

6. What if one device doesn’t join?

- Make sure its power Led is on
- Check if it is communicating: the other LED flashes
- Is the device within range?
- Does it show up in the network manager (message communicated)?

- Restart the device.

Other helpful hints:

- Look up online at www.awiatech.com
- Check the error messages from application and Windows Event Viewer.
- Email to support@awiatech.com

2. DEVELOPMENT GUIDE

2.1 System Architecture

An AwiaNet provides a wireless medium to transport sensor data to the host application and other data from the host to the sensors. There are two interfaces, one to the sensors, and one to the host application, as is shown in Figure 4.

In a WirelessHART network the conventional gateway is divided into three components, the access point that talks wirelessly to the network, the gateway (GW) that interfaces to the host applications, and the network manager (NM) that manages the wireless network. In the Awia network the access point is a hardware module that is connected to a personal computer. The gateway and network manager are software in the computer. The software is called Awia Vanguard and is installed from the CD. The Awia Warriors represent the rest of the Awia network. They are the network nodes.

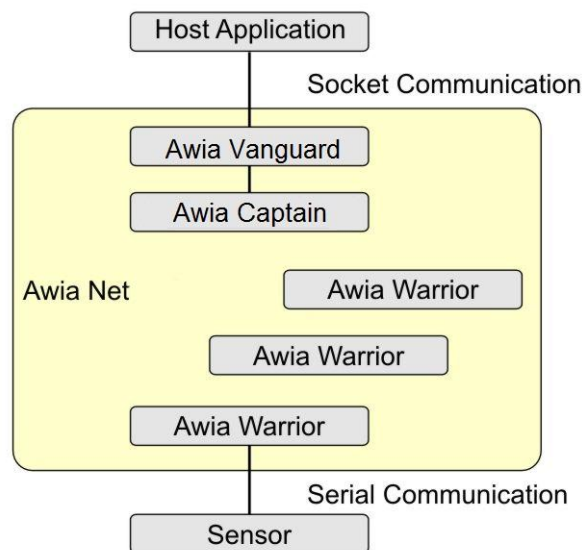


Figure 2. AwiaNet Network Diagram

AwiaTech WirelessHART Evaluation kit is a demo system with all the components within the circle of Figure 4 plus a demo host application. There is a simulated sensor built into the Awia Warrior which can be activated in the demo system.

AwiaTech WirelessHART evaluation kit provides the interfaces for host application developers and sensor developers.

2.2 Host API

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

2.2.1 Definition

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

The gateway and the host application exchange messages. The message data format is defined in Table .

Protocol Number	Payload
-----------------	---------

Table 1. Host API Message Format

The Protocol Number is a one byte field indicating the different communication protocols used. This evaluation kit release only supports Protocol 1, the “simple data access”. In other words, this field must be set to 1. The Protocol 1 message payloads are formatted as commands. The payload is one byte command number followed by the command content, whose size differs for different commands.

The commands are:

Cmd #	Name (content in the parenthesis)	Definition	Host->GW	GW->Host
1	<i>ReadTagList()</i>	Read the list of devices in the network.	Yes	No
2	<i>Subscribe(device tag, update rate, burst command number)</i>	Instruct the device to publish data to the host. The data is also saved in the gateway cache.	Yes	No
3	<i>Unsubscribe(device tag)</i>	Stop publishing.	Yes	No
4	<i>Read(device tag)</i>	Read data from the gateway cache.	Yes	No
5	<i>TagList(tag list)</i>	Report the device list.	No	Yes
6	<i>Data(device tag, data value)</i>	Forward the published data from the device or, as response to the Read() command, report data in the gateway cache.	No	Yes
7	<i>Write(device tag, data value)</i>	Write data to the device.	Yes	No
8	<i>HARTCmd(device tag, command number, command content)</i>	Send a HART command	Yes	Yes

Table 2. Command numbers

The content parameters are:

1.	device tag	6 bytes. Uniquely identifies a Warrior node.
2.	update rate	1 byte. The frequency of which the Warrior will publish data. Its value v is from 0 to 12. The update period = $(2^v \times 250\text{ms})$.
3.	burst command number	2 bytes. <ul style="list-style-type: none"> - 0x00, 0x01 (Command 1): Burst from the simulated sensor in the Warrior. - 0x00, 0xAA (Command 170): Burst the data in the Warrior cache updated by the real sensor.
4.	data value	<ul style="list-style-type: none"> - 0x00, 0x01 (Command 1): 4 bytes IEEE-754 (IEC 559) compatible single precision floating-point number. The value is sampled from a sinusoid curve in the range of [-1, 1]. Reference the source code in the sample host for its interpretation. - 0x00, 0xAA (Command 170): Byte string the sensor put in the Warrior cache. Maximum size is 72 bytes.
5.	command number	2 bytes. A valid HART command. Most significant byte first.
6.	command content	The payload of the HART command. The first byte is the status byte if it is a command response from the Warrior.

Table 3. Payload parameters

As an example, the host wants to read the network ID from device 0x00001b1e0002. It will send HART command 774 (0x0306) to that device. The host will then send message byte string 0x010800001b1e00020306 to the gateway. The message the host gets back will be 0x010800001b1e0002030600AAAA, which says the network ID is 0xAAAA.

2.2.2 Usage

In this section we describe how to set up the host to receive burst data from the Awia Warrior.

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

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

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

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

2.2.3 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();

```

2.3 Device API

As is shown in Figure 4, a sensor can use the 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

The message data format is defined in Table .

Preambles	0x86	LEN	SEQ	Message	CRC
-----------	------	-----	-----	---------	-----

Table 4. Command Format

The preambles are a fixed sequence of 5 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 1 byte *SEQ* field indicates the sequence number of this packet, which usually starts from 0 and is incremented by one every time when a packet is sent, wrapping around 255 to 0. *Message* 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* inclusively.

The format for *Message* is defined in Table 5. It follows the transport layer and application layer of the WirelessHART standard. Please refer to the Network Management Specification (HCF_SPEC_085) for more details.

Transport Byte	Device Status	Extended	Command	Byte Count	Data
----------------	---------------	----------	---------	------------	------

Table 5. 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* 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.

2.3.1 From Warrior to Sensor: Host Data – Burst (Command 170)

This is an AwiaTech device specific command. This command writes a series of bytes received by the gateway from the host. The host sends the bytes by calling Write() in the Host API. The host could also call HARTCmd() with “Command Number” set to 170. In either case the gateway will send Command 170 to Awia Warrior.

Awia Warrior sends Command 170 to the sensor when and only when it receives Command 170 from the gateway. At this point the Warrior copies the host bytes to its local data cache for Command 170, sends them to the sensor, and then sends Command 170 response back to the gateway. The Command 170 response includes the same host bytes.

Maximum allowed host data size is 72 bytes, but can also be 0 bytes.

2.3.2 From Warrior to Sensor: Host Data – Non-Burst (Command 171)

This is an AwiaTech device specific command. This command writes a series of bytes received by the gateway from the host. The host sends the bytes by calling HARTCmd() with “Command Number” set to 171, and the gateway will send Command 171 to the Awia Warrior.

The Warrior sends Command 171 to the sensor when and only when it receives Command 171 from the gateway. At this point the Awia Warrior sends the host data to the sensor without copying the host bytes to its local data cache for Command 171; it then sends Command 171 response back to the gateway. The Command 171 response includes the data from its local data cache for Command 171.

Maximum allowed host data size is 72 bytes, but can also be 0 bytes.

2.3.3 From Sensor to Warrior: Sensor Data – Burst (Command 170)

This is an AwiaTech device specific command. In this command the sensor sends a series of bytes for bursting. The Warrior copies the bytes to its local data cache for Command 170. If the host has subscribed for publishing Command 170, the Awia Warrior will periodically burst Command 170 to the gateway with the data in this local cache.

2.3.4 From Sensor to Warrior: Sensor Data – Non-Burst (Command 171)

This is an AwiaTech device specific command. In this command the sensor sends a series of bytes. The Awia Warrior copies the bytes to its local data cache for Command 171. The host retrieves the data in this cache by sending Command 171 to the Awia Warrior.

2.3.5 From Sensor to Warrior: Set Network ID (Command 773)

This is a standard WirelessHART command. This command configures the Awia Warrior 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 Data* is the two byte network ID in big endian.

Command Data Bytes

Byte	Format	Description
0-1	Unsigned-16	Network ID

2.3.6 From Sensor to Warrior: Write Join Key (Command 961)

This is a standard WirelessHART command. This command is used to set the join key of the device. During the join process, the join key is used for authentication with the network manager.

In this command, *Cmd Data* is the 16 byte join key.

Command Data Bytes

Byte	Format	Description
0-15	Unsigned-128	Key value

2.4 Putting It All Together

2.4.1 Use Awia Net as a data transmission medium - Burst

Just like other network infrastructure, the Awia Net can serve as a medium for two peers to exchange data. In this case the peers are the host application and the sensors in the field. For this purpose, the Awia Net 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 Net will route the data to the Warrior associated with that sensor, copy the data to the Warrior's cache for Command 170, then send Command 170 to the sensor with the same string of bytes.

From sensor to host: The sensor sends Command 170 to the Awia Warrior with a string of bytes. The Awia Warrior will cache the bytes. If bursting is established by the host, The Awia Warrior will periodically publish the data in its cache to the gateway. Upon the reception of an update from the Awia Warrior, the gateway puts the data in its own cache and forwards 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 its 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 Awia 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, i.e., the semantics of the byte string exchanged between the two. The host and sensor should manage the acknowledgement and retry; it is possible that the wireless network may lose a message. In addition, since there is a limit of 72 bytes per message, the user should handle data larger than 72 bytes by breaking them up in the sender and reassembling in the receiver.

Table 6 shows example 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 6. Command 170 messages field value

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;
}

```

2.4.2 Use Awia Net as a data transmission medium – Non-Burst

We use Command 170 for subscription based method to retrieve sensor data, i.e., “push” or unsolicited data publication. It works best for frequent sensor updates; the host does not need to ask for update each time.

In many other applications, though, the host only needs to read from the sensor once in a long while, or occasionally skips a read. This is better achieved with “pull” or solicited data retrieval. The host does not need to subscribe for data update. The Awia Net supports this with Command 171.

From host to sensor: The host application calls HARTCmd() with Command 171 and a string of bytes to be sent to the sensor. The Awia network will route the data to the Awia Warrior associated with that sensor, who sends Command 171 to the sensor with the same string of bytes.

From sensor to host: The sensor sends Command 171 to the Awia Warrior with a string of bytes. The Awia Warrior will copy the data to its local cache for Command 171. Independently, when the Awia Warrior receives Command 171 from the gateway, it will prepare Command 171 response with its cached data and send back to the gateway. Note in the response the Awia Warrior will prefix one byte status value to the cached data. When the gateway receives the response, it will send HARTCmd() with Command 171 to the host with the received string of bytes. There is no data cache for Command 171 in the gateway.

Similar to the subscription based method, it's up to the user to define the protocol between the host and the sensor.

An example: In this application, the first byte of the byte string will be a handle. The host increments the handle each time it sends to the sensor asking for a new data. The sensor sends back the data with the handle copied to the first byte.

The host could call a read() function whenever it needs to read a new data from the sensor. The pseudo code of read() is as follows:

```
char handle = 0;
```

```

char data[100];
char * read()
{
    handle++;
    do
    {
        HARTCmd(deviceTag, 171, handle);
        receive_HARTCmd(retDeviceTag, &cmdNo, data);
        sleep(1s);
    }
    while ((retDeviceTag != deviceTag) || (cmdNo != 171) || (data[1] != handle))

    return &(data[2]); // data[0] is the HART status byte
}

```

The sensor could implement the following endless loop:

```

char handle = 0;
char data[100];
while (1)
{
    receive_Command171(&handle);
    if (handle != data[0])
    {
        Handle = data[0];
        read_sensor(&(data[1]));
    }
    Command171(data);
}

```

2.4.3 Use Awia Warrior in a WirelessHART device

Awia 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. Awia Warrior 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. Or the user could configure the targeted network with the network ID (0x1236) and join key (0x000012360000000000000000) used in the evaluation kit. With these pieces of information, a device equipped with Awia Warrior can join the designated WirelessHART network automatically upon power-on.

2.4.4 Use Awia Net as a WirelessHART evaluation tool

A major purpose of the AwiaTech WirelessHART evaluation kit is for users to explore and learn about WirelessHART. A WirelessHART network could be set up right out-of-the-box with the parts in the kit, as is described in Section 1. The user can also run the Wi-Analys tool from the HART Communication Foundation side-by-side with this kit to view the WirelessHART network traffic.

3. TECHNICAL SPECIFICATION

WirelessHART Software

- 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

Awia Warrior 220 Hardware

Peripherals/Software	Detail
Input	UART, RS485, FSK HART over 4-20mA
HART FSK interface	FSK HART Maintenance Port
Radio	1x Awia Warrior 210 IEEE802.15.4 radio module
Hardware Chip	FreeScale MC1322X (RF+MCU)
Electrical & Mechanical	Detail
Input Voltage	4.5 .. 12 V DC
Main Board Dimensions	2.2in(W) x 3.4in(L); 5.5cm(W) x 8.7cm(L)
Certifications	Detail
EMI	FCC U.S, CE
Standard	WirelessHART-conforming, registration ready

Subject to change without prior notice.

4. GLOSSARY OF TERMS

Awia Captain AwiaTech's Access Point product

Awia Commander AwiaTech's WirelessHART Network Manager, which also includes AwiaTech WirelessHART Security Manager

AwiaNet a WirelessHART network consisting of AwiaTech's WirelessHART products

Awia Vanguard AwiaTech's WirelessHART Gateway

Awia Warrior a hardware module running AwiaTech's WirelessHART network stack, thus acting as a node to the network. It is usually attached to a sensor device

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

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

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

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

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

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

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.

5. INDEX

application layer, 6, 29

Awia Warrior, 7, 17, 18, 24, 26, 29, 30,
31, 33, 34, 35, 36, 39

data link layer, 6

Gateway, 5, 7, 8, 20, 21, 22, 36

IEC 62591, 5

IEEE 802.15.4, 5, 35

network layer, 6

OSI, 6

physical layer, 6, 37

transport layer, 6, 29

WirelessHART, 5, 6, 7, 8, 14, 18, 21, 22,
24, 29, 30, 31, 34, 35, 36, 39

AwiaTech Corporation © 2011. All rights reserved.

AwiaTech, Awia Warrior, Awia Vanguard, Awia Commander, Awia Captain, and AwiaNet are trademarks of AwiaTech Corporation. HART and WirelessHART is a trademark of the 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.