User Manual

WICED

Creating a Cloud-Supported WICED™ IoT Solution

BROADCOM
MASS MARKET
PLATFORM

WICED-UM200-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

November 19, 2015

Revision History

Revision	***Date***	***Change Description***
WICED-UM200-R	11/19/2015	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

About This Document 7
Purpose and Audience 7
Acronyms and Abbreviations 7
Document Conventions 7
References 8
Technical Support 8
Section 1: Cloud-Supported IoT Solution Overview 9
Introduction 9
Basic IoT Solution Block Diagram and Description 10
AWS Summary 10
IoT Device Demonstration Platform Summary 11
Section 2: IoT Demonstration Platform Description 12
Overview 12
Interfaces 15
Ambient-Light Sensor 15
Arduino Shield Headers 15
JX1 Pinout and Signal Map 16
JX2 Pinout and Signal Map 17
JX3 Pinout and Signal Map 18
JX4 Pinout and Signal Map 19
JTAG 19
LEDs 20
D1—WLAN Activity LED 20
D2 and D3—User LEDs 20
D4—JTAG Activity LED 20
D5—UART Activity LED 20
D6—3.3V Power LED 20
Micro-USB Connector (J5) 20
PMOD Connector (J6) 21
Power 21
Pushbutton Switches (SW1 and SW2) 21
UART 22
Wi-Fi Debug UART Header (J7) 22
Wi-Fi JTAG Header (J8) 22
Wi-Fi Module 23
Mechanical Information 25

Section 3: IoT Platform Application Development 27
Introduction 27
System Requirements for WICED Application Development 28
Downloading and Installing a WICED IDE/SDK 28
Downloading a WICED IDE/SDK 29
Installing the WICED IDE/SDK on a Windows System 30
Installing the WICED IDE/SDK on a MAC OS X System 31
Updating to WICED SDK 3.4.0-AWS 31
WICED SDK Features 32
Connecting to the WICED 4343W IoT Starter Kit 33
Building and Downloading an App to the Demo Platform 34
Running an App on the Demo Platform 38
Debugging an App on the Demo Platform 40
Configuring a Terminal Application 42
Setting Up PuTTY for Windows 42
Setting Up CoolTerm for OS X 44
Section 4: Sample AWS IoT Applications 46
Section 5: AWS IoT Service and Device Setup 47
Introduction 47
Getting Started with the AWS IoT Service 48
Resolving a Failed-to-Create-Thing Error Message 51
Viewing and Editing IoT Device Information 52
Configuring an IoT Device 53
Reconfiguring an IoT Device 54

List of Figures

Figure 1: IoT Solution Example 10
Figure 2: Demonstration Platform—Top View of Key Component Placements 12
Figure 3: Demonstration Platform Block Diagram 13
Figure 4: WICED 4343W Starter Kit Demonstration Platform 14
Figure 5: Light Sensor Circuit 15
Figure 6: JX1 Pinout 16
Figure 7: JX2 Pinout 17
Figure 8: JX3 Pinout 18
Figure 9: JX4 Pinout 19
Figure 10: J6 PMOD Connector Detail 21
Figure 11: Wi-Fi Module Pinout 23
Figure 12: Carrier Board Silkscreen 25
Figure 13: Arduino Uno R3 Board Dimensions on Which the Carrier Board is Based 26
Figure 14: WICED IDE Windows Setup 30
Figure 15: WICED SDK OS X Setup 31
Figure 16: The WICED Integrated Development Environment 34
Figure 17: AWS High-Level Block Diagram 47

List of Tables

Table 1: Ambient-Light Sensor Connection to the Wi-Fi Module 15
Table 2: JX1 to Wi-Fi Module Signal Mapping 16
Table 3: JX2 to Wi-Fi Module Mapping (Including MCU Pin to WICED_GPIO_X Enumeration) 17
Table 4: JX3 to Wi-Fi Module Signal Mapping (Including MCU Pin to WICED_GPIO_X Enumeration) 18
Table 5: JX4 to Wi-Fi Module Signal Mapping (Including MCU Pin to WICED_GPIO_X Enumeration) 19
Table 6: WLAN Activity LED Connection to the Wi-Fi Module 20
Table 7: User LED Connections to the Wi-Fi Module 20
Table 8: J6 PMOD Connector Pinout 21
Table 9: Pushbutton Switches Pinouts 21
Table 10: Wi-Fi Debug UART Header—J7 22
Table 11: Wi-Fi JTAG Header Signal List 22
Table 12: Wi-Fi Module Pinout and Mapping to the Carrier Board 24
Table 13: Overview of the WICED SDK Top-Level Directory 32

About This Document

Purpose and Audience

This document provides information for IoT device and application developers who plan to:

- Use Broadcom® chips in their IoT devices.
- Use the Broadcom WICED™ SDK to develop IoT device applications.
- Integrate cloud services into their IoT system solutions.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	***Description***
Bold	User input and actions: for example, type **exit**, click **OK**, press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

References

The references in this section may be used in conjunction with this document.

Note: Broadcom provides customer access to technical documentation and software through its Broadcom Support Community website (community.broadcom.com).

For Broadcom documents, replace the “xx” in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document (or Item) Name	***Number***
Broadcom Items	
[1] Single-Chip IEEE 802.11 b/g/n MAC/Baseband/Radio with Bluetooth 4.1, an FM Receiver, and Wireless Charging	4343W-DS1xx-R

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal. For a CSP account, contact your Broadcom Sales or Engineering support representative.

General WICED support is available to registered users in the Broadcom Support Community forum:

http://community.broadcom.com/welcome

Section 1: Cloud-Supported IoT Solution Overview

Introduction

A typical cloud-supported IoT solution contains two core elements:

1. One or more IoT devices (as publishers and/or subscribers)
2. A cloud solution that supports:
   – A secure messaging service for reading data from publishers and writing data to subscribers
   – Management application development
   – An application management console and a management command-line interface
   – An IoT device state-shadowing service
   – A database storage service
   – A computing service
   – A notification service

To help those developing the first core element listed above, Broadcom provides connectivity devices (chips and/or boards), a software development kit (SDK), supporting documentation, and ongoing support via its community website (community.broadcom.com). For the purposes of this document, a Broadcom BCM4343W chip is used on an AVNET-developed IoT device demonstration platform and a special SDK with some simple applications (see Section 4: “Sample AWS IoT Applications,” on page 46) is made available. For more information on the sample SDK (which is named WICED SDK 3.4.0-AWS), see “Downloading and Installing a WICED IDE/SDK” on page 28.

The second core element, the cloud solution, is a collection of Amazon Web Services (AWS), particularly the new AWS IoT service, which is in beta as of the publication of this document.

Besides the block diagram that follows in this section, the remaining sections in this document provide the following information:

- Section 2: “IoT Demonstration Platform Description,” on page 12
  This section describes an IoT demonstration platform based on the Broadcom BCM4343W.
- Section 3: “IoT Platform Application Development,” on page 27
  This section describes how to get started developing applications that will get loaded and executed on the IoT demonstration platform described in Section 2: “IoT Demonstration Platform Description,” on page 12.
- Section 4: “Sample AWS IoT Applications,” on page 46
  This section summarizes a few simple IoT applications that are a combination of WICED applications running on the IoT demonstration platform and cloud-based applications that use AWS.
- Section 5: “AWS IoT Service and Device Setup,” on page 47
  This section provides information on setting up and using AWS.

Basic IoT Solution Block Diagram and Description

Figure 1 shows a simple IoT solution example.

Figure 1: IoT Solution Example

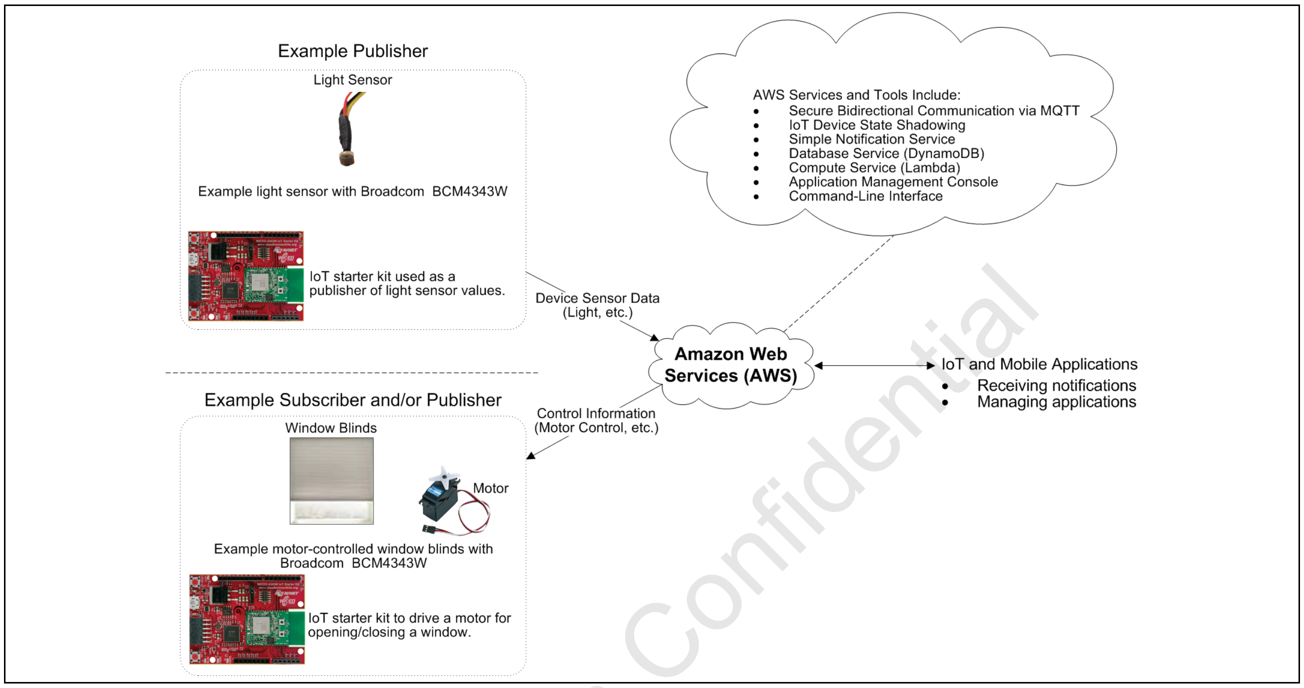

Example Publisher
Light Sensor
Example light sensor with Broadcom BCM4343W
IoT starter kit used as a publisher of light sensor values.
Device Sensor Data (Light, etc.)
AWS Services and Tools Include:
Secure Bidirectional Communication via MQTT
IoT Device State Shadowing
Simple Notification Service
Database Service (DynamoDB)
Compute Service (Lambda)
Application Management Console
Command-Line Interface
Amazon Web Services (AWS)
IoT and Mobile Applications
Receiving notifications
Managing applications
Example Subscriber and/or Publisher
Window Blinds
Motor
Control Information (Motor Control, etc.)
Example motor-controlled window blinds with Broadcom BCM4343W
IoT starter kit to drive a motor for opening/closing a window.

Figure 1 shows three entities: a light sensor IoT device, a motor controller IoT device (for controlling window blind opening and closing), and AWS. The light sensor regularly publishes light intensity readings taken at a certain location. Rule assessments and computations within AWS determine if the window blinds at the same location should be adjusted.

AWS Summary

In Figure 1 on page 10, AWS services and tools provide:

- A secure bidirectional communication service that provides a publication/subscription broker that uses the Message Queuing Telemetry Transport (MQTT) protocol.
  - Allows light intensity readings to be sent from the light sensor IoT device to AWS.
  - Allows motor control information to be sent from AWS to the motor-control IoT device (for adjusting window blinds).
- State shadowing of the connected IoT devices.
- A rules engine that uses SQL-like syntax for evaluating user-supplied rules.

  Rule evaluation can lead to other actions. For example, initiating a computation (using the Lambda service), sending an MQTT message to an IoT device (for example, to adjust the blinds), pushing notifications to other Internet-connected devices, etc.

For more information on AWS, see Section 5: "AWS IoT Service and Device Setup," on page 47.

IoT Device Demonstration Platform Summary

The demonstration platform, otherwise known as the WICED 4343W IoT Starter Kit, supports the following features and interfaces:

- A certified AVNET BCM4343W-based Wi-Fi module board (also referred to as the SoC module) with the following features:
  - IEEE 802.11b/g/n Wi-Fi.
  - Bluetooth 4.1 with an upgrade path to Bluetooth 4.2.
  - An STM32F411 ARM Cortex M4 MCU.
  - 512 KB of flash.
  - 128 KB of SRAM.
  - 8 Mb of SPI serial flash.
  - Dual fractal PCB antennas that support antenna diversity.
- An Arduino form-factor carrier board (also referred to as a base board) with the following features and interfaces:
  - Arduino-compatible shield expansion connectors that support:
    - Four GPIO.
    - Three analog inputs.
    - Two I^{2}C ports (one of which is shared).
    - One SPI port.
    - Two UARTs (one of which is shared).
  - One 2×6, PMOD-compatible, peripheral expansion connector (contains the shared I^{2}C port).
  - USB-based JTAG debugger/programmer and serial UART port.
  - A reset push-button switch.
  - A user push-button switch.
  - Two user LEDs.
  - Four status LEDs (UART, JTAG and WLAN activity, and 3.3V power).
  - One ambient light sensor.
  - A Micro-USB connector used to power (5V) a high-capacity 3.3V regulated supply and for data communications.
  - 3.3V I/O.

For complete information on the IoT device demonstration platform shown in Figure 1 on page 10, see Section 2: "IoT Demonstration Platform Description," on page 12.

For information on getting started developing IoT device applications using the combination of the demonstration platform and the Broadcom WICED SDK, see Section 3: "IoT Platform Application Development," on page 27.

Section 2: IoT Demonstration Platform Description

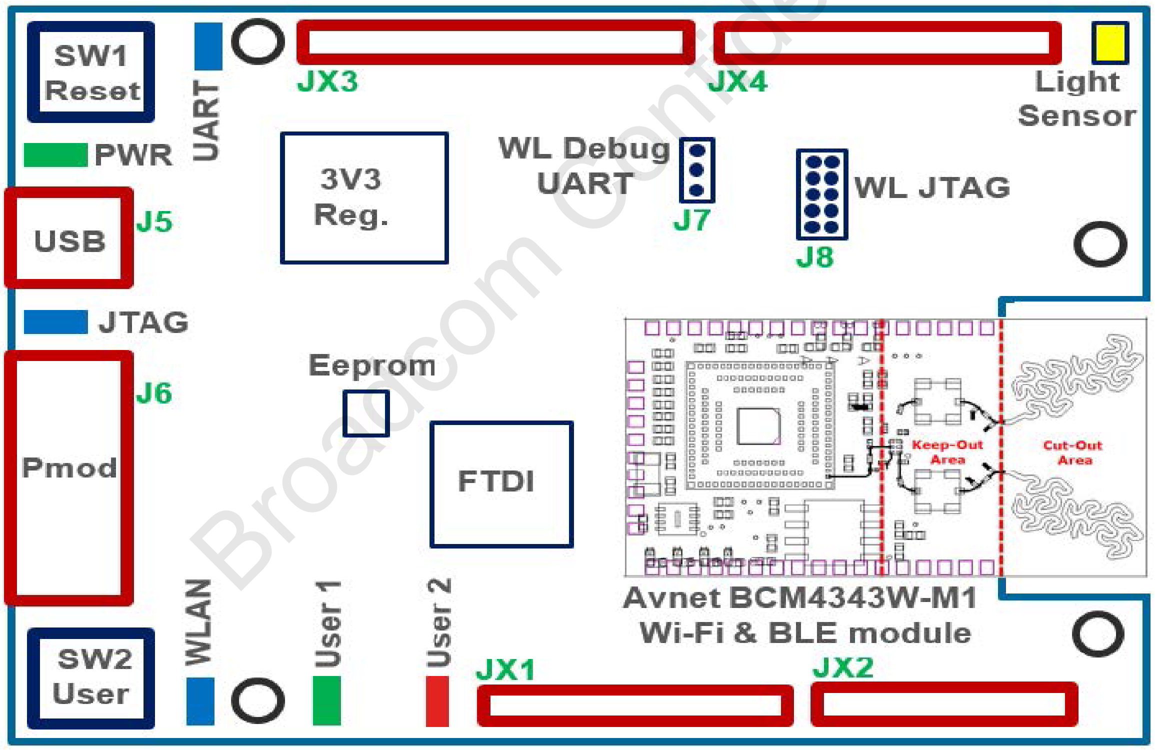
Overview

The demonstration platform contains two boards: a BCM4343W-based Wi-Fi module board and a carrier board that provides access to interfaces, power, switches, etc. The Wi-Fi module board is attached to the carrier board. The demonstration platform is also sometimes referred to as the WICED BCM4343W IoT Starter Kit.

Throughout the remainder of the document, the two attached boards that comprise the demonstration platform are referred to as the carrier board and the Wi-Fi module.

Figure 2 shows a top view of key component placements on the demonstration platform.

Figure 2: Demonstration Platform—Top View of Key Component Placements



See Figure 4 on page 14 for a picture of the actual board and Figure 12 on page 25 for a silkscreen of the board.

Figure 3 is a demonstration platform block diagram. See “Interfaces” on page 15 for more information on each of the interfaces shown in the figure.

Figure 3: Demonstration Platform Block Diagram

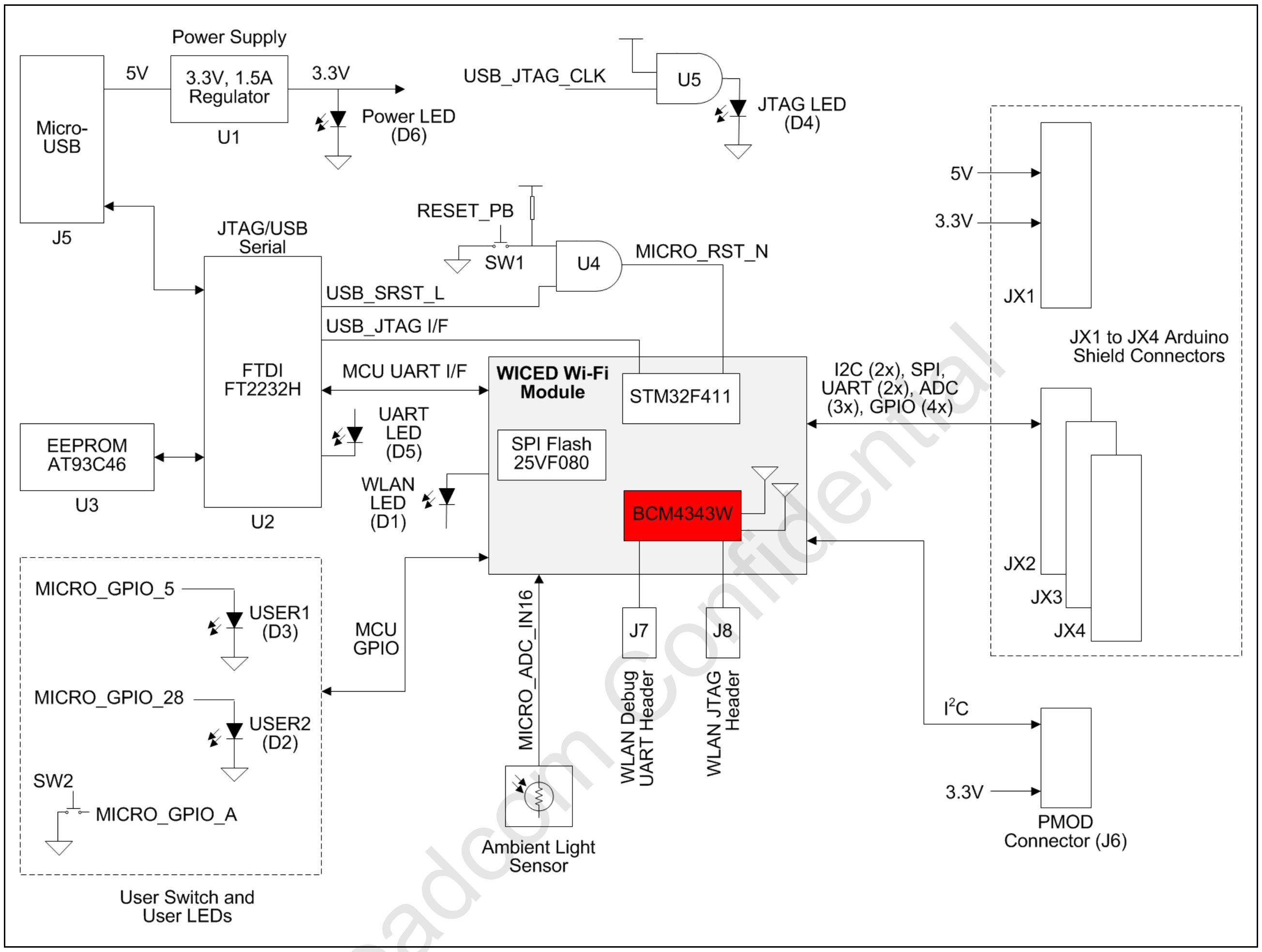
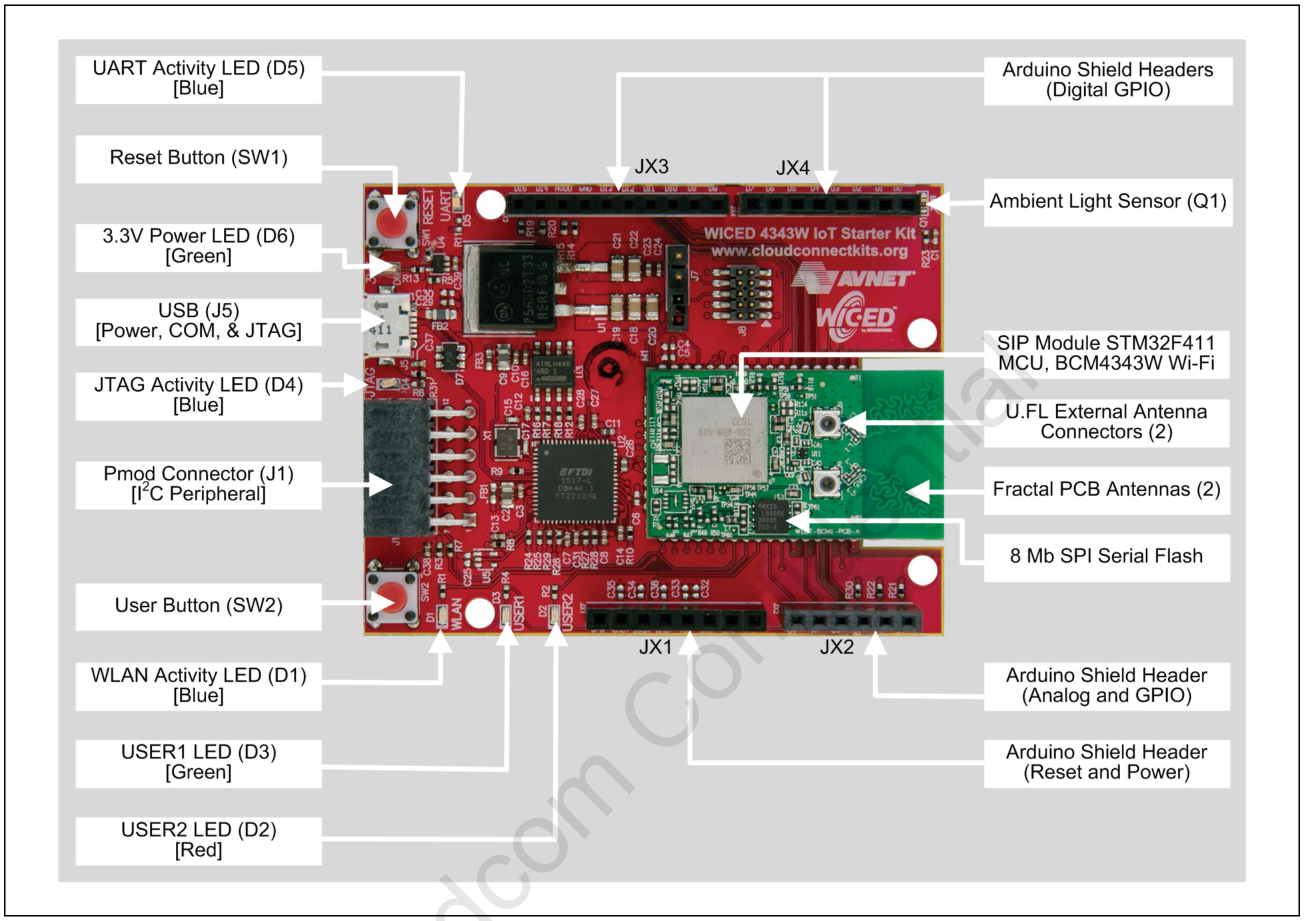


Figure 4 shows the WICED 4343W Starter Kit demonstration platform as well as its interfaces. For more information on the various interfaces, see “Interfaces” on page 15.

Figure 4: WICED 4343W Starter Kit Demonstration Platform


UART Activity LED (D5) [Blue]
Reset Button (SW1)
3.3V Power LED (D6) [Green]
USB (J5) [Power, COM, & JTAG]
JTAG Activity LED (D4) [Blue]
Pmod Connector (J1) [I2C Peripheral]
User Button (SW2)
WLAN Activity LED (D1) [Blue]
USER1 LED (D3) [Green]
USER2 LED (D2) [Red]
Arduino Shield Headers (Digital GPIO)
JX3
JX4
Ambient Light Sensor (Q1)
SIP Module STM32F411 MCU, BCM4343W Wi-Fi
U.FL External Antenna Connectors (2)
Fractal PCB Antennas (2)
8 Mb SPI Serial Flash
JX1
JX2
Arduino Shield Header (Analog and GPIO)
Arduino Shield Header (Reset and Power)
WICED 4343W IoT Starter Kit
www.cloudconnectkits.org
AVNET
WICED

Interfaces

Ambient-Light Sensor

The carrier board includes a low-cost, SMD phototransistor, ambient light sensor from EverLight. The sensor output connects directly to the ADC input of the Wi-Fi module microcontroller (STM32F711).

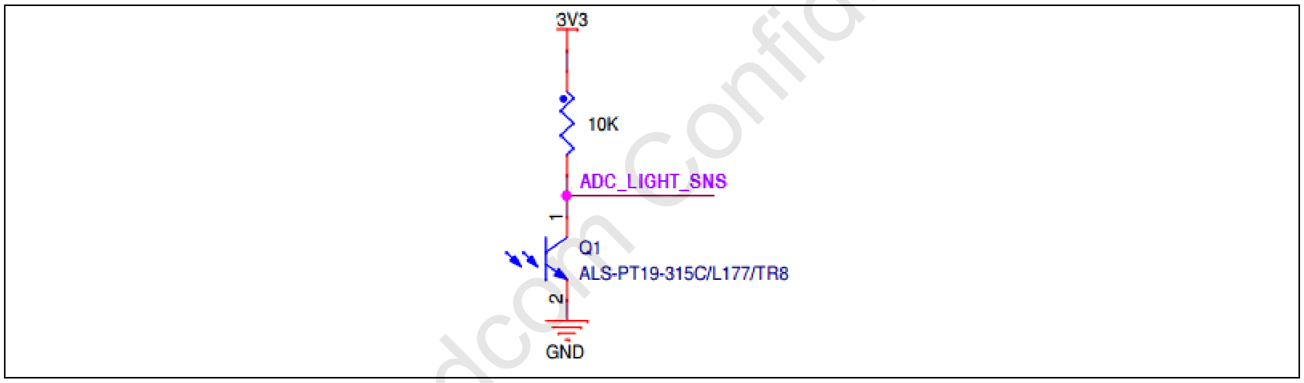
Table 1 provides the light sensor connection to the Wi-Fi module.

Table 1: Ambient-Light Sensor Connection to the Wi-Fi Module

Light Sensor		***Wi-Fi Module Pinout***	
Pin	***Net Name***	***Pin***	***Net Name***
1	ADC_LIGHT_SNS	43	MICRO_ADC_IN15

Figure 5 shows the light sensor circuit.

Figure 5: Light Sensor Circuit



Arduino Shield Headers

Arduino-compatible shield header sockets facilitate design expansion.

Assignment of peripheral I/O between carrier board connectors JX1–JX4 and the Wi-Fi module are provided in "JX1 Pinout and Signal Map" on page 16 through "JX4 Pinout and Signal Map" on page 19.

JX1 Pinout and Signal Map

Figure 6 shows the JX1 header pinout and Table 2 shows the JX1 signal mapping to the Wi-Fi module.

Figure 6: JX1 Pinout

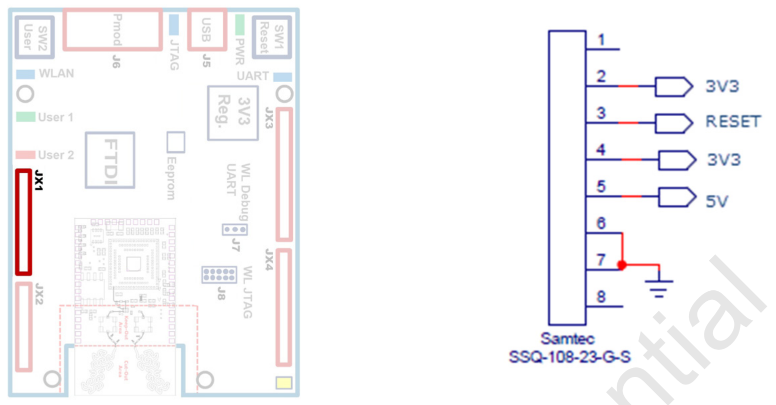


Table 2: JX1 to Wi-Fi Module Signal Mapping

JX1			*Wi-Fi Module*	
Pin	***Signal Name***	***Arduino R3 UNO Name***	***Pin***	***Name***
JX1-1	N/C	–	–	N/C
JX1-2	3V3	IOREF	16, 26	VDD_3V3_SIP, VBAT_SIP
JX1-3	RESET	RESET	36	MICRO_RST_N
JX1-4	3V3	3.3V	16, 26	VDD_3V3_SIP, VBAT_SIP
JX1-5	5V	5V	–	5V
JX1-6	GND	GND	1, 15, 17, 25, 27, 39, 44, 45	GND
JX1-7	GND	GND		GND
JX1-8	N/C	VIN	N/C	–

JX2 Pinout and Signal Map

Figure 7 shows the JX2 header pinout and Table 3 shows the JX2 signal mapping to the Wi-Fi module.

Figure 7: JX2 Pinout

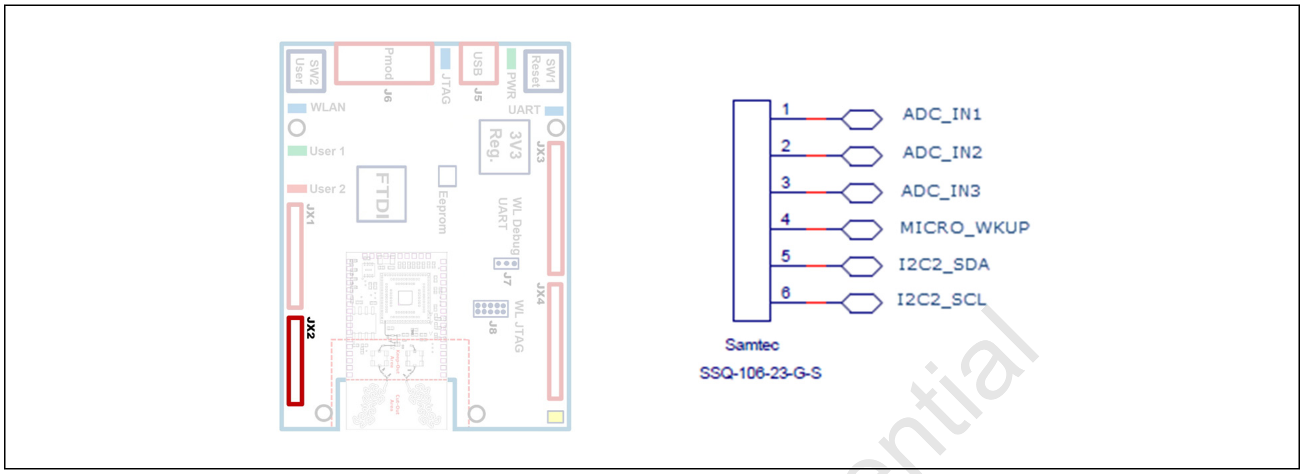


Table 3: JX2 to Wi-Fi Module Mapping (Including MCU Pin to WICED_GPIO_X Enumeration)

JX2			***Wi-Fi Module***		***WICED Software Detail***	
Pin	***Signal Name***	***Arduino R3 UNO Name***	***Pin***	***Name***	***MCU Pin***	***wiced_gpio_t enumeration***
JX2-1	ADC_IN1	A0	37	MICRO_ADC_IN1	A1	WICED_GPIO_2
JX2-2	ADC_IN2	A1	40	MICRO_ADC_IN2	A2	WICED_GPIO_3
JX2-3	ADC_IN3	A2	41	MICRO_ADC_IN3	A3	WICED_GPIO_4
JX2-4	MICRO_WKUP	A3	38	MICRO_WKUP	A0	WICED_GPIO_1
JX2-5	I2C2_SDA	A4	3	MICRO_I2C2_SDA	B9	WICED_GPIO_21
JX2-6	I2C2_SCL	A5	2	MICRO_I2C2_SCL	B10	WICED_GPIO_20

JX3 Pinout and Signal Map

Figure 8 shows the JX3 header pinout and Table 4 shows the JX3 signal mapping to the Wi-Fi module.

Figure 8: JX3 Pinout

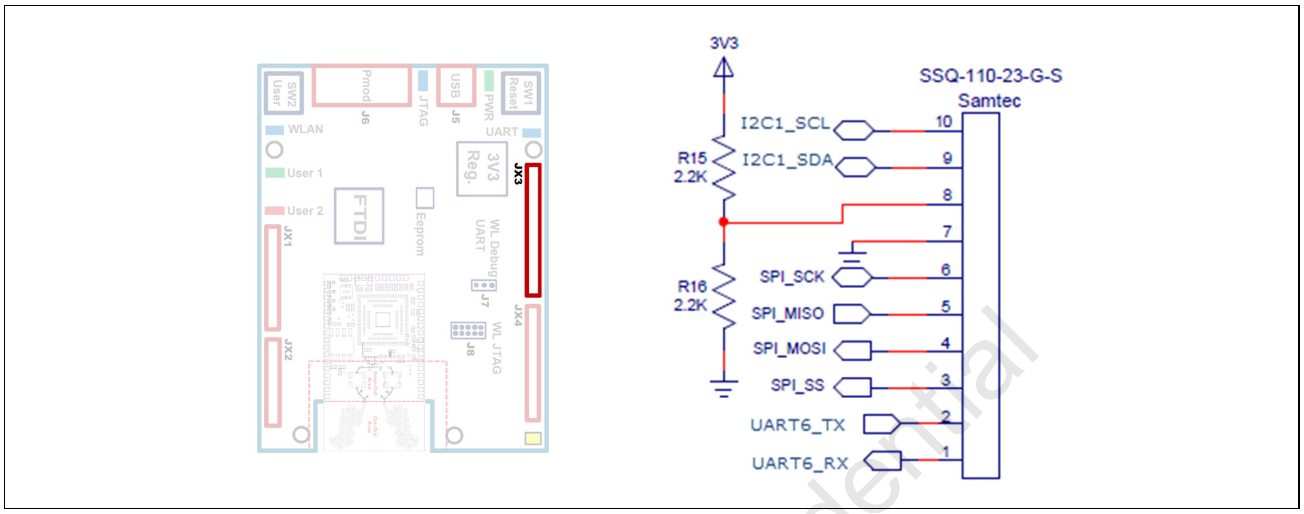


Table 4: JX3 to Wi-Fi Module Signal Mapping (Including MCU Pin to WICED_GPIO_X Enumeration)

JX3			***Wi-Fi Module***		***WICED Software Detail***	
Pin	***Signal Name***	***Arduino R3 UNO Name***	***Pin***	***Name***	***MCU Pin***	***wiced_gpio_t enumeration***
JX3-10	I2C1_SCL	D15	28	MICRO_I2C1_SCL	B6	WICED_GPIO_11
JX3-9	I2C1_SDA	D14	29	MICRO_I2C1_SDA	B7	WICED_GPIO_12
JX3-8	–	AREF	–	–	–	–
JX3-7	GND	GND	–	–	–	–
JX3-6	SPI_SCK	D13	4	MICRO_SPI2_SCK	B13	WICED_GPIO_23
JX3-5	SPI_MISO	D12	6	MICRO_SPI2_MISO	B14	WICED_GPIO_24
JX3-4	SPI_MOSI	D11	7	MICRO_SPI2_MOSI	B15	WICED_GPIO_25
JX3-3	SPI_SS	D10	5	MICRO_SPI2_SSN	B12	WICED_GPIO_22
JX3-2	UART6_TX	D9	8	USART6_TX_I2S2_MCK	C6	WICED_GPIO_13
JX3-1	UART6_RX	D8	9	USART6_RX_I2S2_CK	C7	WICED_GPIO_14

JX4 Pinout and Signal Map

Figure 9 shows the JX4 header pinout and Table 5 shows the JX4 signal mapping to the Wi-Fi module.

Figure 9: JX4 Pinout

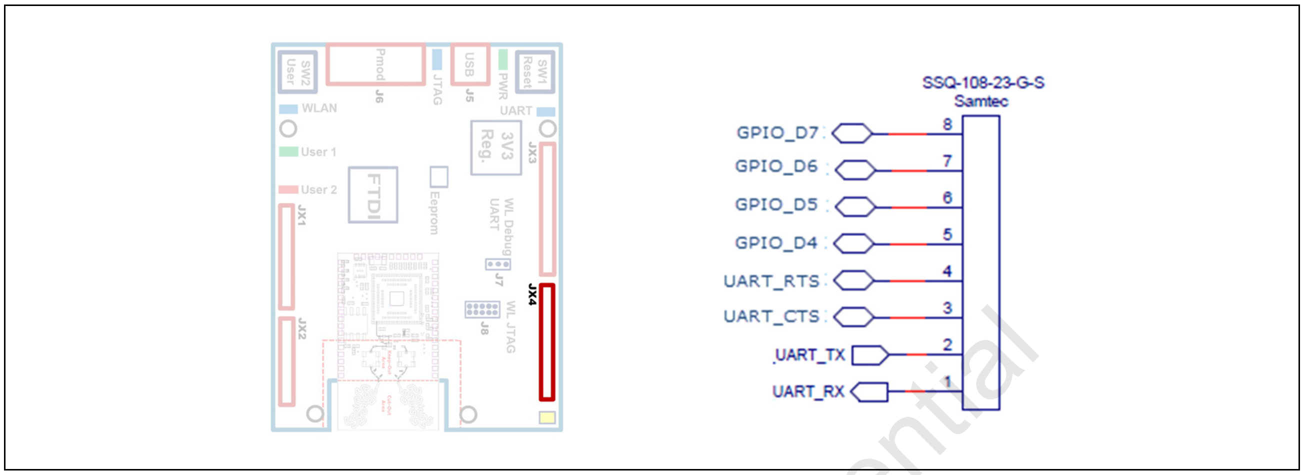


Table 5: JX4 to Wi-Fi Module Signal Mapping (Including MCU Pin to WICED_GPIO_X Enumeration)

JX4			Wi-Fi Module		WICED Software Detail	
Pin	**Signal Name**	**Arduino R3 UNO Name**	**Pin**	**Name**	**MCU Pin**	**wiced_gpio_t enumeration**
JX4-8	GPIO_D7	D7	33	MICRO_GPIO_B	C3	WICED_GPIO_17
JX4-7	GPIO_D6	D6	32	MICRO_GPIO_17	C2	WICED_GPIO_28
JX4-6	GPIO_D5	D5	30	MICRO_GPIO_2	C1	WICED_GPIO_27
JX4-5	GPIO_D4	D4	31	MICRO_GPIO_1	C0	WICED_GPIO_26
JX4-4	UART_RTS	D3	13	MICRO_UART_RTS	A12	WICED_GPIO_16
JX4-3	UART_CTS	D2	12	MICRO_UART_CTS	A11	WICED_GPIO_15
JX4-2	UART_TX	D1	10	MICRO_UART_TX	A9	WICED_GPIO_9
JX4-1	UART_RX	D0	11	MICRO_UART_RX	A10	WICED_GPIO_10

JTAG

The JTAG interface signals are routed to the Wi-Fi module STM32F411 microcontroller. They provide a programming and debugging interface to the BCM4343W.

LEDs

The board has six LEDs as status indicators.

D1—WLAN Activity LED

The WLAN activity LED is blue when the Wi-Fi module sends/receives Wi-Fi data. The LED is activated by source current from the WIFI_GPIO_1 signal.

Table 6: WLAN Activity LED Connection to the Wi-Fi Module

User LEDs on Carrier Board		***Wi-FiI Module Pinout***	
LED Name	***Color***	***Pin***	***Net Name***
WLAN (D1)	Blue	14	WIFI_GPIO_1

D2 and D3—User LEDs

There are two user LEDs, USER1 (green) and USER2 (red). The LED control signals are GPIO from the Wi-Fi module microcontroller (STM32F411). They are active-low signals controlled by user application firmware.

Table 7: User LED Connections to the Wi-Fi Module

User LEDs on Carrier Board		***Wi-FiI Module Pinout***	
LED Name	***Color***	***Pin***	***Net Name***
USER1 (D3)	Green	42	MICRO_GPIO_5
USER2 (D2)	Red	35	MICRO_GPIO_28

D4—JTAG Activity LED

The JTAG activity LED is blue when the Wi-Fi module microcontroller is being programmed via the JTAG interface.

D5—UART Activity LED

The UART activity LED is blue when there is Wi-Fi module serial console activity.

D6—3.3V Power LED

The 3.3V green power LED, which is connected to the output of the 3.3V regulator, is illuminated when the power supply output status is good.

Micro-USB Connector (J5)

A Micro-USB connector (J5) provides power as well as access to a dual-interface device. The dual-interface device provides UART communication (via Channel B) and a JTAG interface (Channel A) for programming and debugging.

The FT2232HQ USB to serial device (U2) on the carrier board implements the UART and JTAG interfaces.

PMOD Connector (J6)

This is a 2×6, right angle, female connector. It is used to provide a Digilent PMOD-compatible I^2C interface with the I^2C pins duplicated on the upper and lower connector rows.

Figure 10 shows the J6 PMOD connector detail.

Figure 10: J6 PMOD Connector Detail

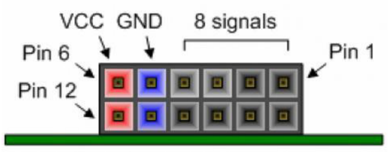


Table 8 shows the J6 PMOD connector pinout.

Table 8: J6 PMOD Connector Pinout

Pin	Wi-Fi Module Net Name	Pin	Wi-Fi Module Net Name
1	–	7	–
2	–	8	–
3	SCL[a]	9	SCL[a]
4	SDA[b]	10	SDA[b]
5	GND	11	GND
6	VCC	12	VCC

a. The Wi-Fi module net name for this signal is MICRO_I2C2_SCL.
b. The Wi-Fi module net name for this signal is MICRO_I2C2_SDA.

Power

+5V is sourced from the Micro-USB connector (J5). 3.3V regulation is done by an onboard 1.5A linear regulator.

Pushbutton Switches (SW1 and SW2)

The carrier board has two vertically activated pushbutton switches. SW1 is the board reset and SW2 is for users to define via application software.

Table 9: Pushbutton Switches Pinouts

Switch	Wi-Fi Module Pinout	
	Pin	Net Name
SW1 (RESET)	36	MICRO_RST_N
SW2 (USER)	34	MICRO_GPIO_A

UART

The UART provides a console interface. UART configuration information is stored in an EEPROM (U3 on the carrier board).

Wi-Fi Debug UART Header (J7)

An unpopulated through-hole three-pin header provides easy cable access even when an Arduino-compatible shield is stacked on the carrier board.

In the event a user wants to utilize this interface for WLAN debugging purposes, a right-angle 3-pin header will need to be added to the board (and suitable USB to UART serial cable and terminal software used).

Table 10: Wi-Fi Debug UART Header—J7

J7 Carrier Board Header		*Wi-Fi Module Pinout*	
Pin	***Net Name***	***Pin***	***Net Name***
1	WL_DBG_UART_TX	19	WL_JTAG_TDO
2	WL_DBG_UART_RX	18	WL_JTAG_TDI
3	GND	GND	GND

Wi-Fi JTAG Header (J8)

An unpopulated SMT 2×5 pin header is provided.

Table 11 provides the J8 Wi-Fi JTAG header signal list.

Table 11: Wi-Fi JTAG Header Signal List

J8 Carrier Board Header		*Wi-Fi Module Pinout*	
Pin	***Net Name***	***Pin***	***Net Name***
1	Vcc	3V3	VCC
2	WL_JTAG_TMS	6	MICRO_SPI2_MISO
3	GND	GND	GND
4	WL_JTAG_CLK	4	MICRO_SPI2_SCK
5	GND	GND	GND
6	WL_DBG_UART_TX	19	WL_JTAG_TDO
7	N/C	–	–
8	WL_DBG_UART_RX	18	WL_JTAG_TDI
9	N/C	–	–
10	WL_JTAG_TRSTN	5	MICRO_SPI2_SSN

Wi-Fi Module

The BCM4343W-based Wi-Fi module supports IEEE 802.11b/g/n and Bluetooth 4.1 (with a growth option to Bluetooth 4.2). For more information on the BCM4343W, see Reference [1] on page 8.

Figure 11 shows the pinout of the Wi-Fi module. The interface is comprised of 45 pins on three sides of the wireless module plus a GND pad positioned directly under the system-in-a-package (SIP) device.

Figure 11: Wi-Fi Module Pinout

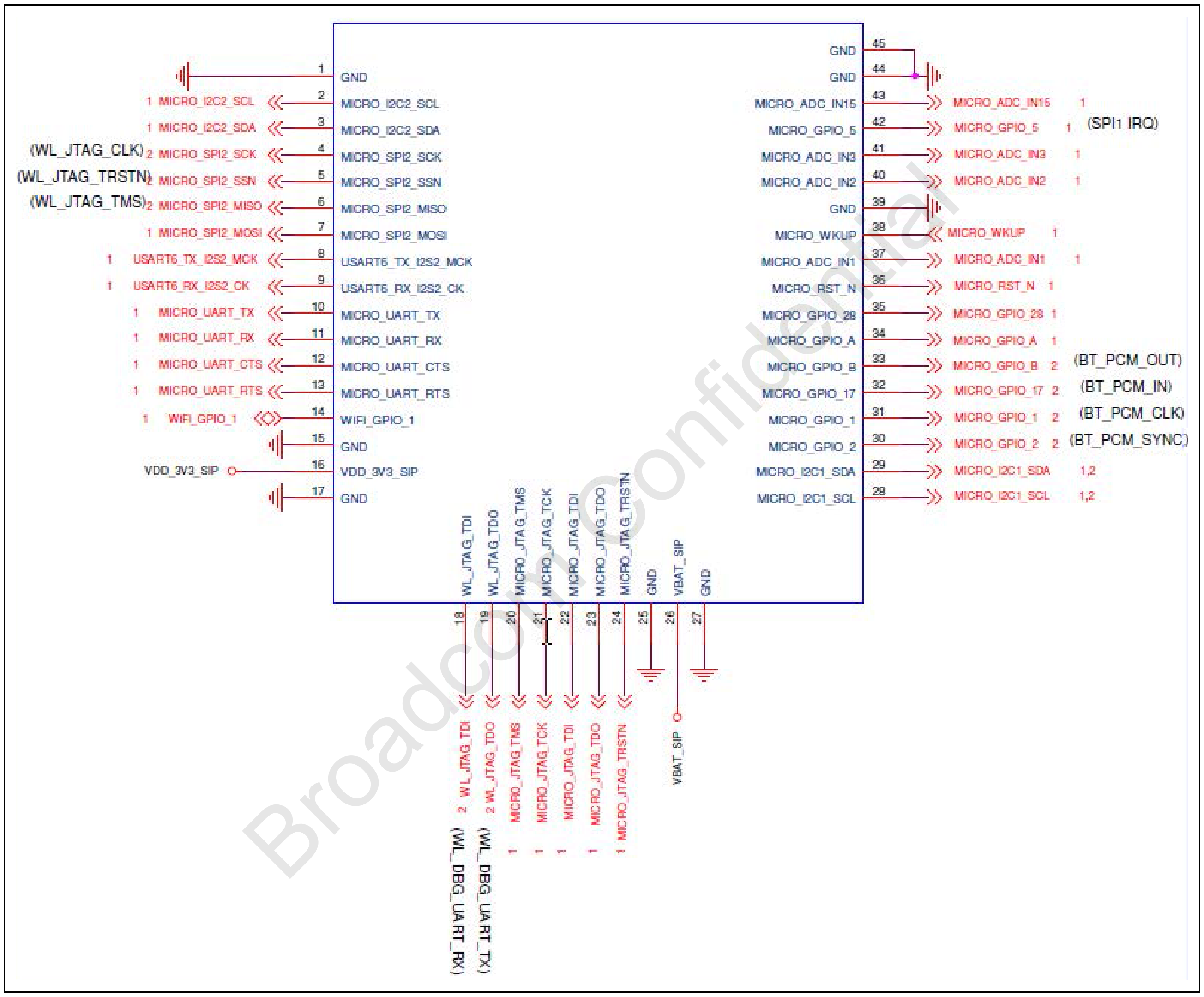


Table 12 provides a Wi-Fi module pinout list and the pin mapping between the Wi-Fi module and the carrier board.

Table 12: Wi-Fi Module Pinout and Mapping to the Carrier Board

Module Pin#	Wireless Module Signal Name	Connector / Device Pin#	Other Name / Comments
1	GND	–	GND
2	MICRO_I2C2_SCL	JX2.6, J6.3, J6.9	A5
3	MICRO_I2C2_SDA	JX2.5, J6.4, J6.10	A4
4	MICRO_SPI2_SCK	JX3.6, J8.4	D13
5	MICRO_SPI2_SSN	JX3.3, J8.10	D10
6	MICRO_SPI2_MISO	JX3.5, J8.2	D12
7	MICRO_SPI2_MOSI	JX3.4	D11
8	USART6_TX_I2S2_MCK	JX3.2	D9
9	USART6_RX_I2S2_CK	JX3.1	D8
10	MICRO_UART_TX	JX4.2	D1
11	MICRO_UART_RX	JX4.1	D0
12	MICRO_UART_CTS	JX4.3	D2
13	MICRO_UART_RTS	JX4.4	D3
14	WIFI_GPIO_1	–	WLAN Activity LED (Blue)
15	GND	JX3.7, JX1.6, JX1.7	GND
16	VDD_3V3_SIP	JX1.2, JX1.4	IOREF/3V3
17	GND	JX3.7, JX1.6, JX1.7	GND
18	WL_JTAG_TDI	J8.8, J7.2	WL_DBG_UART_RX
19	WL_JTAG_TDO	J8.6, J7.1	WL_DBG_UART_TX
20	MICRO_JTAG_TMS	–	–
21	MICRO_JTAG_TCK	–	–
22	MICRO_JTAG_TDI	–	–
23	MICRO_JTAG_TDO	–	–
24	MICRO_JTAG_TRSTN	–	–
25	GND	JX3.7, JX1.6, JX1.7	GND
26	VBAT_SIP	JX1.2, JX1.4	IOREF/3V3
27	GND	JX3.7, JX1.6, JX1.7	GND
28	MICRO_I2C1_SCL	JX3.10	D15
29	MICRO_I2C1_SDA	JX3.9	D14
30	MICRO_GPIO_2	JX4.6	D5
31	MICRO_GPIO_1	JX4.5	D4
32	MICRO_GPIO_17	JX4.7	D6
33	MICRO_GPIO_B	JX4.8	D7
34	MICRO_GPIO_A	–	USER switch (SW2)
35	MICRO_GPIO_28	–	USER2 LED (Red)

Table 12: Wi-Fi Module Pinout and Mapping to the Carrier Board (Cont.)

Module Pin#	Wireless Module Signal Name	Connector / Device Pin#	Other Name / Comments
36	MICRO_RST_N	JTAG or SW1 pushbutton reset, JX1-3	Reset
37	MICRO_ADC_IN1	JX2.1	A0
38	MICRO_WKUP	JX2.4	A3
39	GND	JX3.7, JX1.6, JX1.7	GND
40	MICRO_ADC_IN2	JX2.2	A1
41	MICRO_ADC_IN3	JX2.3	A2
42	MICRO_GPIO_5	–	USER1 LED (Green)
43	MICRO_ADC_IN15	–	Ambient light sensor
44	GND	JX3.7, JX1.6, JX1.7	GND
45	GND	JX3.7, JX1.6, JX1.7	GND

Mechanical Information

Figure 12 shows a silkscreen of the carrier board.

Figure 12: Carrier Board Silkscreen

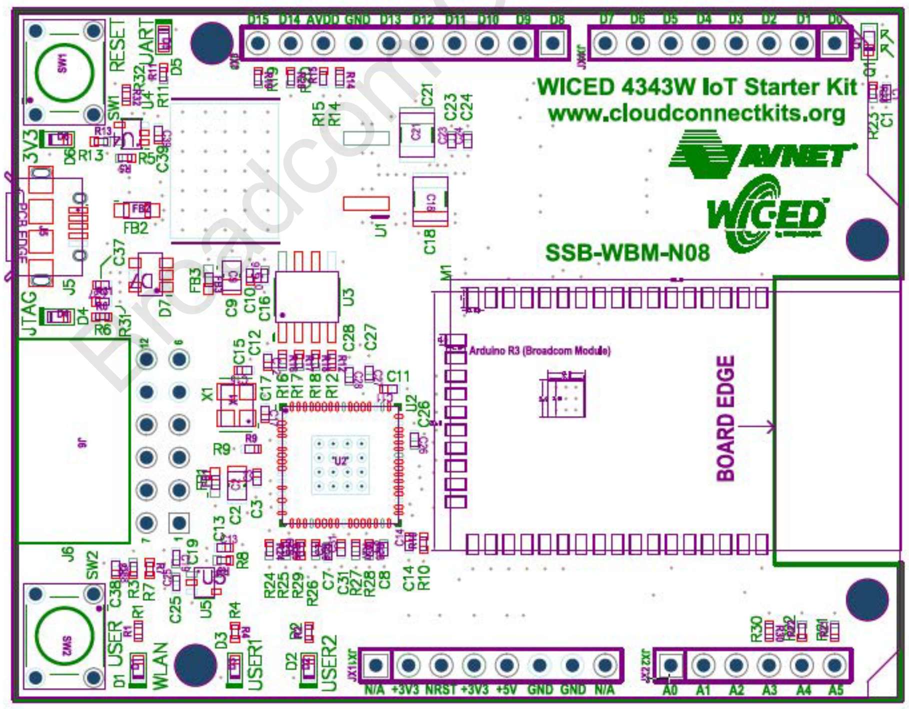
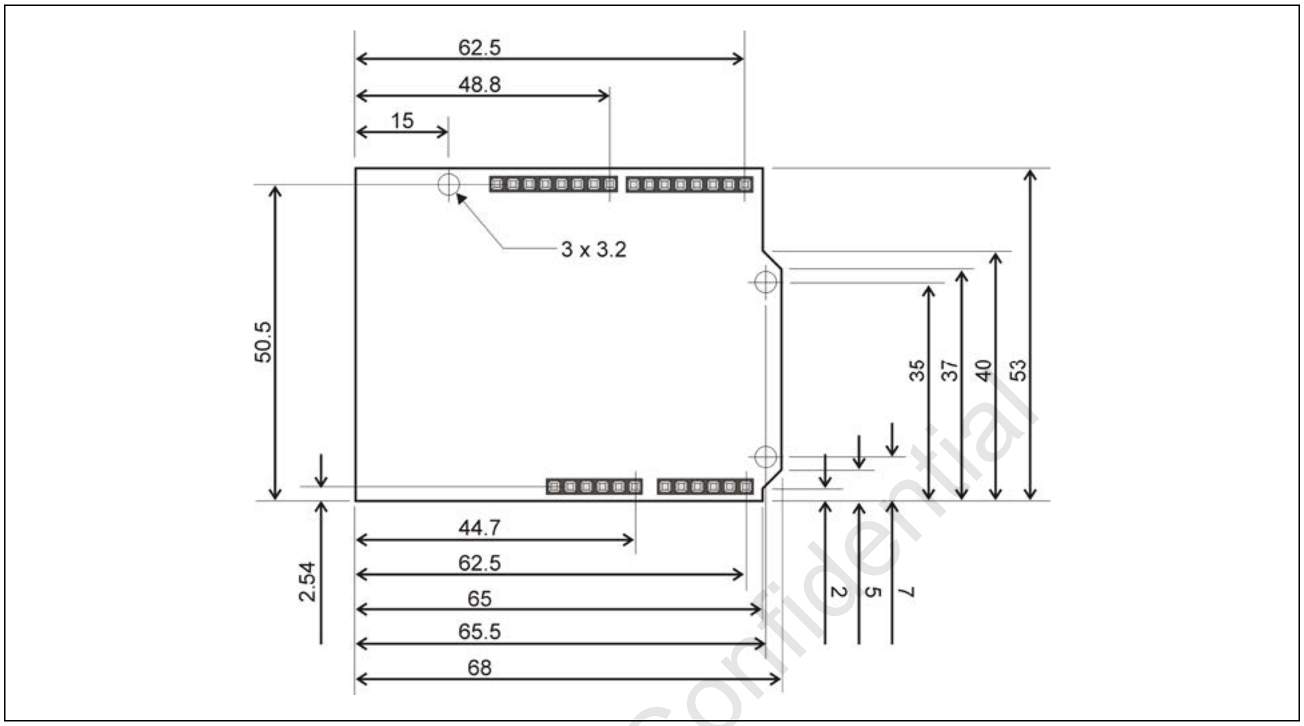


Figure 13 shows the dimensions on which the carrier board dimensions are based.

Figure 13: Arduino Uno R3 Board Dimensions on Which the Carrier Board is Based



The carrier board has four mounting holes and the dimensions of the wireless module are 35 mm × 20 mm.

Section 3: IoT Platform Application Development

Introduction

This section contains the following information:

- “System Requirements for WICED Application Development”
- “Downloading and Installing a WICED IDE/SDK” on page 28
- “Connecting to the WICED 4343W IoT Starter Kit” on page 33
- “Building and Downloading an App to the Demo Platform” on page 34
- “Running an App on the Demo Platform” on page 38
- “Debugging an App on the Demo Platform” on page 40
- “Configuring a Terminal Application” on page 42

A development computer in conjunction with the WICED 4343W IoT Starter Kit demonstration board is needed to perform IoT application development.

Caution! Do not plug the WICED 4343W IoT Starter Kit demonstration board into the development computer prior to installing WICED SDK 3.4.0-AWS. Doing so may cause the incorrect driver to load.

Note: To install WICED SDK 3.4.0-AWS, see “Downloading and Installing a WICED IDE/SDK” on page 28.

System Requirements for WICED Application Development

In order to develop WICED applications, consider the following hardware and software requirements:

- The WICED SDK supports 32- and 64-bit versions of Microsoft Windows® XP and Win7, Mac OS X 10.5 or later, and 32/64-bit Linux.
- The SDK is distributed as a standalone 7zip file suitable for all operating systems, and is often bundled together with the WICED Integrated Development Environment (IDE) as an executable installer. The installer is only provided for Windows® and OS X operating systems.
- The development computer requires a single USB port to connect to the WICED 4343W IoT Starter Kit demonstration platform.
- A terminal emulation program such as PuTTY (Windows®) or CoolTerm (OS X) is required.

Note: The 7zip extraction utility is available from 7-zip.org. 7zip is needed if you plan to extract the standalone WICED SDK 7zip archive. The standard Windows zip-file extraction utility may silently corrupt the SDK archive during the extraction process. Do not use it.

Downloading and Installing a WICED IDE/SDK

WICED SDK 3.4.0-AWS, which contains the functionality to support the AWS IoT service and includes the sample AWS IoT apps briefly described in Section 4: “Sample AWS IoT Applications,” on page 46, does not come with an installer.

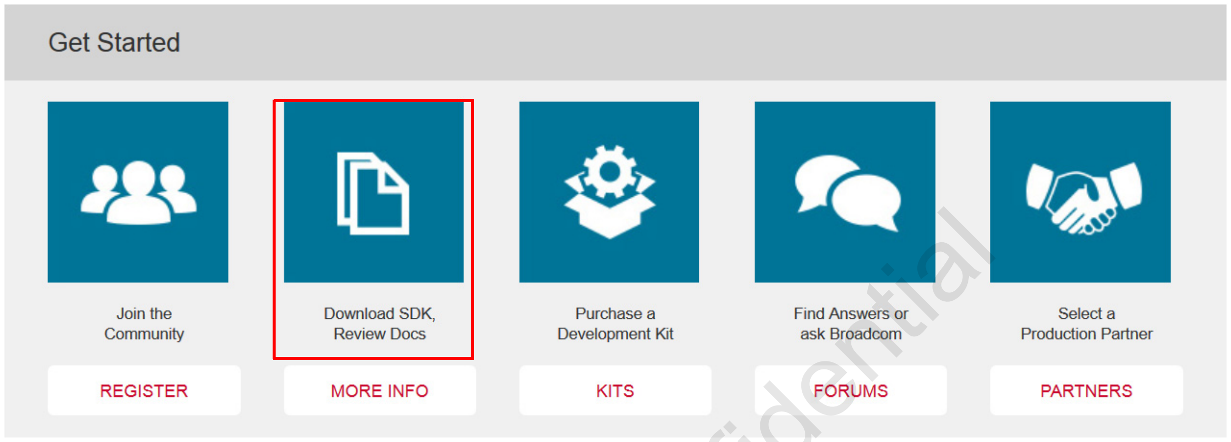
Those with a previous WICED IDE installed, such as the IDE installed using the installer provided with WICED Software Development Kit 3.3.1, can proceed to “Updating to WICED SDK 3.4.0-AWS” on page 31.

Those without a WICED IDE installed on their development system should first download a WICED IDE, such as the one provided with WICED Software Development Kit 3.3.1.

Downloading a WICED IDE/SDK

To download a WICED IDE/SDK:

1. In a browser, go to the Broadcom® Community website (https://community.broadcom.com) and click on the WICED™ Wi-Fi block or go directly to the WICED Wi-Fi page located at https://community.broadcom.com/community/wiced-wifi.
2. In the Get Started block, click on **Download SDK, Review Docs**.


Get Started
Join the Community
REGISTER
Download SDK, Review Docs
MORE INFO
Purchase a Development Kit
KITS
Find Answers or ask Broadcom
FORUMS
Select a Production Partner
PARTNERS

Note: Follow the onscreen instructions to easily set up a community website account if prompted to do so.

3. In the Download the SDK block, download the item that you want to install on your development system.

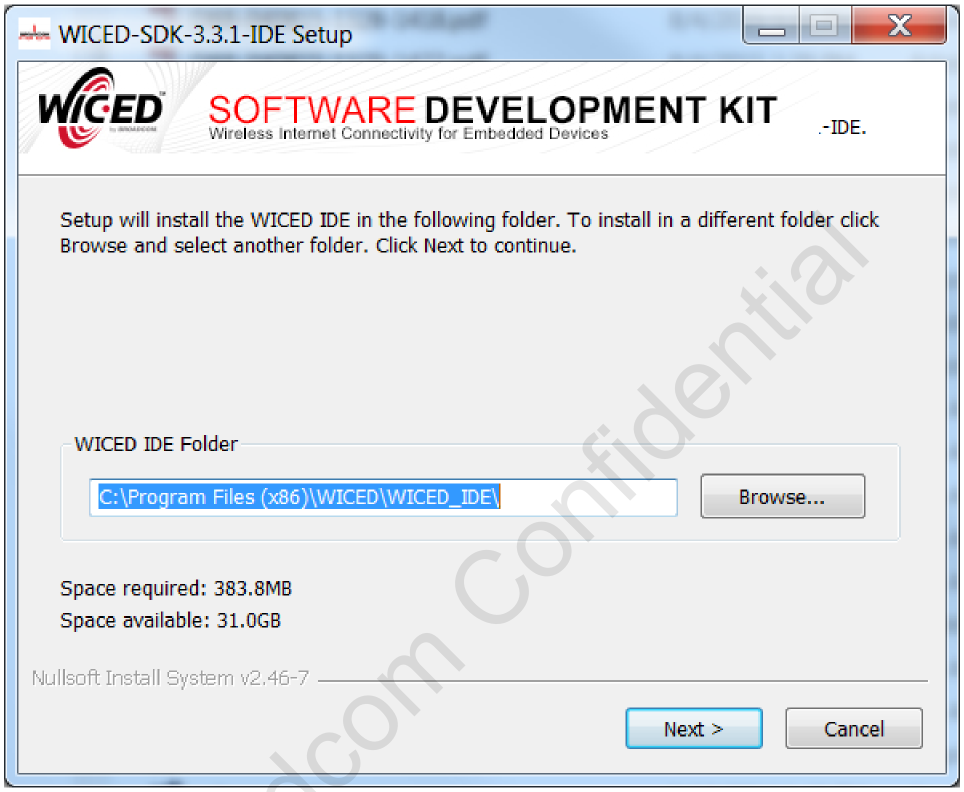
To install the WICED IDE/SDK after downloading an installer file see:

- "Installing the WICED IDE/SDK on a Windows System" on page 30 or
- "Installing the WICED IDE/SDK on a MAC OS X System" on page 31

Installing the WICED IDE/SDK on a Windows System

The WICED SDK is provided as a self-installing executable file. Double-click the installer file (for example, WICED-SDK-3.3.1-IDE-Installer.exe) to begin the installation on a Windows® XP or Windows® 7 system. A setup window similar to Figure 14 will be displayed.

Figure 14: WICED IDE Windows Setup


WICED-SDK-3.3.1-IDE Setup
WICED
SOFTWARE DEVELOPMENT KIT
Wireless Internet Connectivity for Embedded Devices
-IDE.

Setup will install the WICED IDE in the following folder. To install in a different folder click Browse and select another folder. Click Next to continue.

WICED IDE Folder

C:\Program Files (x86)\WICED\WICED_IDE\

Space required: 383.8MB

Space available: 31.0GB

Browse...
Nullsoft Install System v2.46-7
Next >
Cancel

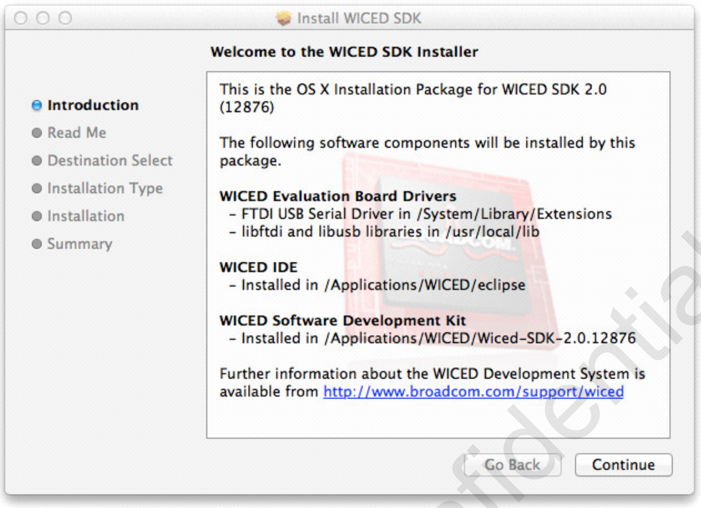
Choose the installation folder for the WICED IDE and click **Next**, then choose the installation workspace folder for the WICED SDK and click **Install**. Once the installation completes, click **Finish** to immediately start the WICED IDE, or deselect the **Start WICED IDE now** options and click **Finish** to exit.

If the WICED evaluation board still does not appear in the Device Manager, verify the 3V3 LED is turned ON and/or replace the USB cable.

Installing the WICED IDE/SDK on a MAC OS X System

The WICED SDK is provided as an OS X package installer. Double-click the Wiced-SDK-2.3.x.pkg file to begin the installation. A setup window similar to Figure 15 displays.

Figure 15: WICED SDK OS X Setup



Welcome to the WICED SDK Installer

This is the OS X Installation Package for WICED SDK 2.0 (12876)

The following software components will be installed by this package.

WICED Evaluation Board Drivers
- FTDI USB Serial Driver in /System/Library/Extensions
- libftdi and libusb libraries in /usr/local/lib

WICED IDE
- Installed in /Applications/WICED/eclipse

WICED Software Development Kit
- Installed in /Applications/WICED/Wiced-SDK-2.0.12876

Further information about the WICED Development System is available from http://www.broadcom.com/support/wiced

Click **Continue** once to read the welcome note, then click **Continue** again and select installation permissions. After installation permissions are selected, click **Continue** and then click **Install**. Enter your account password if asked, and follow the prompts to complete the installation.

Updating to WICED SDK 3.4.0-AWS

In order to use WICED SDK 3.4.0-AWS, first ensure that your development system has a previous WICED IDE installed, such as the IDE that gets installed using the installer provided with WICED SDK 3.3.1.

Follow the instructions for importing a new WICED SDK into an existing WICED Eclipse IDE. The instructions can be found on the Broadcom community website at:

https://community.broadcom.com/community/wiced-wifi/wiced-wifi-forums/blog/2015/08/12/how-to-import-a-wiced-sdk-into-an-existing-wiced-eclipse-ide-2

Note: An account is required to access software and documentation from the Broadcom community website. To create an account, follow the onscreen instructions if prompted to enter login credentials.

WICED SDK Features

The WICED SDK includes:

- WICED software development tools:
  - Utilities and OS drivers to support development in the Windows® environment
  - WICED software stack, development tools and demonstration applications
  - ThreadX and FreeRTOS Real-Time Operating Systems (RTOS), and NetX/NetXDuo IPv4/IPv6 and lwIP IPv4 TCP/IP network stack implementations
  - Embedded security libraries including TLS & HTTPS
  - A WICED Wi-Fi driver and API
  - WICED Application Framework (WAF)
  - Manufacturing test and Iperf applications to enable system performance testing
- WICED API Documentation, this guide, and related documents

Table 13 is an overview of the top-level directory of the WICED SDK.

Table 13: Overview of the WICED SDK Top-Level Directory

WICED SDK Directories	*Directory Contents*
Apps	Demo applications and snippets, test utilities, and WAF components
Doc	API documentation, reference documentation, schematics
Drivers	Windows USB drivers for the WICED evaluation board
Include	WICED API function prototypes, constants, and defaults
Include/platforms	Platform description and I/O definitions
Library	Daemons, servers, protocols, and peripheral libraries
Resources	Resources used by the WICED webserver, including HTML, images, styles, etc.
Tools	Toolchain including compiler, debugger, and other utilities/scripts
Wiced	WICED core components: RTOS, TCP stack, security & platform definitions
Wiced/WWD	The WICED Wi-Fi driver

Connecting to the WICED 4343W IoT Starter Kit

To connect the WICED 4343W IoT Starter Kit demonstration platform to the development PC:

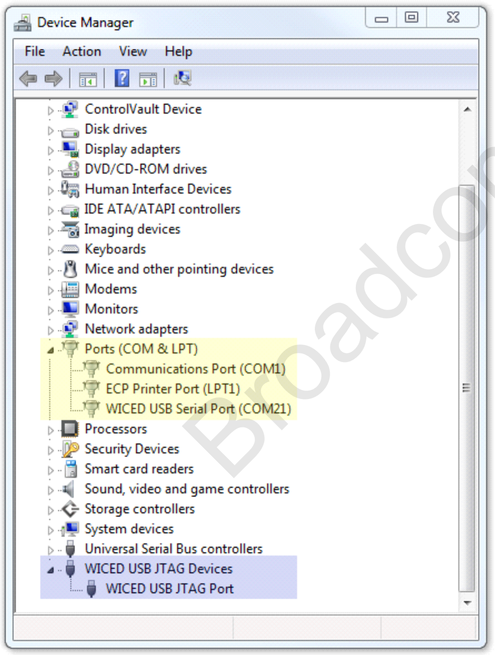
1. Use a USB cable to connect the demonstration platform to the PC. (The driver should load automatically.)
2. Verify the 3.3V LED (D6) on the demonstration platform is on, indicating that the board is powered.

The USB interface provides +5V power as well as individual programming/debug and UART interfaces to the microcontroller on the WICED module. A separate +5V power supply is not required.

The WICED demonstration platform has two logical USB devices: a USB-JTAG device and a USB-UART device.

To verify that the driver installed successfully on a Windows system, do the following:

1. Open the Device Manager (right-click **My Compute** and select **Properties**).
2. In the System Properties window, select **Hardware** and then **Device Manager**.
   a. The WICED USB Serial Port is listed under MY-LAPTOP\Ports (COM & LPT, shaded yellow).
   b. The WICED USB JTAG Port is listed under MY-LAPTOP\WICED USB JTAG Devices (shaded blue).
   The Device Manager window identifies the WICED USB Serial COM port as COM21. The assigned port number varies between systems.

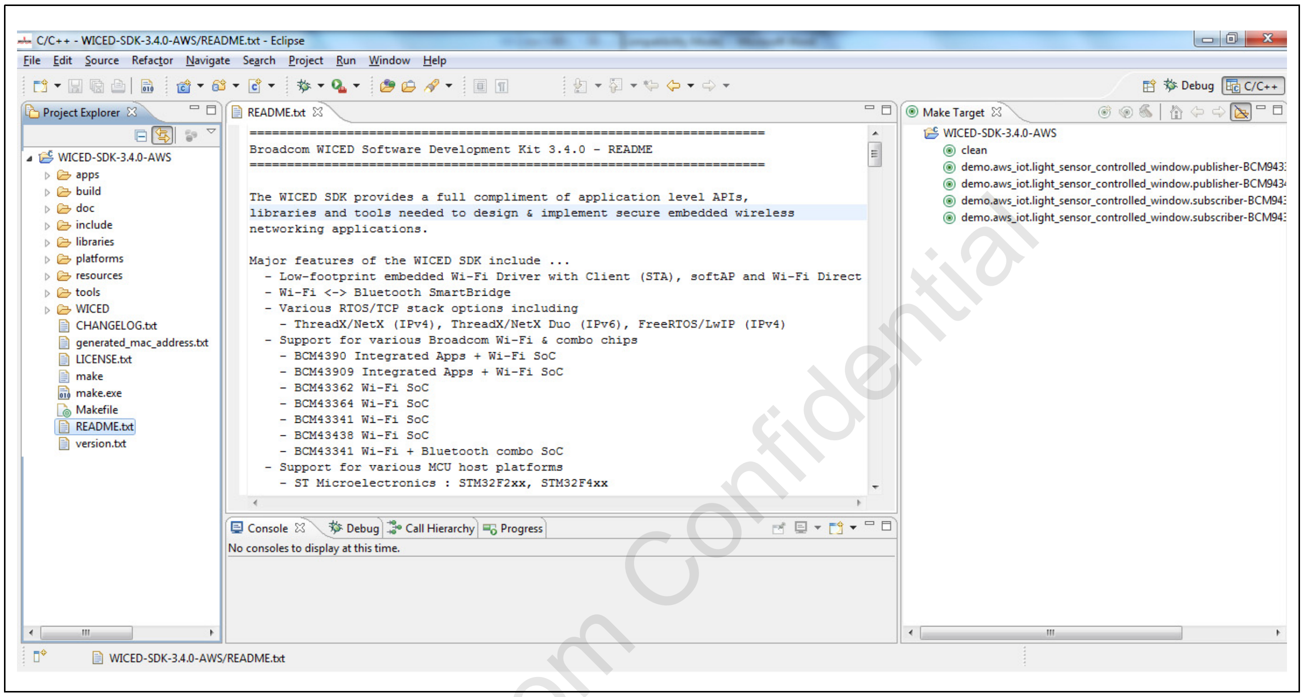


Note: If an error occurs during the automatic driver installation process, the driver may be manually installed from the <WICED-SDK>\Drivers directory.

Building and Downloading an App to the Demo Platform

On a Windows PC, start the WICED IDE by selecting **START**, **All Programs**, **Broadcom**, and then **WICED IDE**. On a Mac running OS X, use the Finder application to locate Applications, the WICED directory, and then double-click the **WICED IDE** shortcut. After startup, the WICED IDE looks similar to Figure 16.

Figure 16: The WICED Integrated Development Environment


C/C++ - WICED-SDK-3.4.0-AWS/README.txt - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
C/C++
Project Explorer
WICED-SDK-3.4.0-AWS
apps
build
doc
include
libraries
platforms
resources
tools
WICED
CHANGELOG.txt
generated_mac_address.txt
LICENSE.txt
make
make.exe
Makefile
README.txt
version.txt
README.txt
Make Target
WICED-SDK-3.4.0-AWS
clean
demo.aws_iot.light_sensor_controlled_window.publisher-BCM943
demo.aws_iot.light_sensor_controlled_window.publisher-BCM943
demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943
demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943
Console
Debug
Call Hierarchy
Progress
WICED-SDK-3.4.0-AWS/README.txt

```
======================================================================
Broadcom WICED Software Development Kit 3.4.0 - README
======================================================================

The WICED SDK provides a full compliment of application level APIs,
libraries and tools needed to design & implement secure embedded wireless
networking applications.

Major features of the WICED SDK include ...
  - Low-footprint embedded Wi-Fi Driver with Client (STA), softAP and Wi-Fi Direct
  - Wi-Fi <-> Bluetooth SmartBridge
  - Various RTOS/TCP stack options including
    - ThreadX/NetX (IPv4), ThreadX/NetX Duo (IPv6), FreeRTOS/LwIP (IPv4)
  - Support for various Broadcom Wi-Fi & combo chips
    - BCM4390 Integrated Apps + Wi-Fi SoC
    - BCM43909 Integrated Apps + Wi-Fi SoC
    - BCM43362 Wi-Fi SoC
    - BCM43364 Wi-Fi SoC
    - BCM43341 Wi-Fi SoC
    - BCM43438 Wi-Fi SoC
    - BCM43341 Wi-Fi + Bluetooth combo SoC
  - Support for various MCU host platforms
    - ST Microelectronics : STM32F2xx, STM32F4xx
```

No consoles to display at this time.

The Help tab to the right of the WICED IDE window describes how to build and download the sample applications shown in the Make Target tab (located above the Help tab). Multiple build targets are preconfigured for a number of sample applications and WICED hardware platforms.

For this example, the aws_iot.light_sensor_controlled_window.subscriber application is built and downloaded to the BCM943364WCD1 WICED module. If the application or module on your evaluation board is different, follow the instructions to modify or copy one of the build targets to match your hardware platform.

Double-click the demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1 target to build the application; the IDE console window displays the build progress. The build target is appended with the 'download' and 'run' options. These options tell the toolchain to download the firmware and run the application after the build completes.

The build output looks similar to the following:

```
**** Build of configuration Default for project WICED-SDK-3.4.0-AWS ****

make demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1 download run
Making config file for first time
+-----------------------------------------------------------------------------------
----+
| IMPORTANT NOTES                                                                                  |
+--------------------------------------------------------------------------------------------------
----+
| Wi-Fi MAC Address                                                                                |
|    The target Wi-Fi MAC address is defined in <WICED-SDK>/generated_mac_address.txt              |
|    Ensure each target device has a unique address.                                               |
+--------------------------------------------------------------------------------------------------
----+
| MCU & Wi-Fi Power Save                                                                           |
|    It is *critical* that applications using WICED Powersave API functions connect an accurate 32kHz
|
|    reference clock to the sleep clock input pin of the WLAN chip. Please read the WICED Powersave  |
|    Application Note located in the documentation directory if you plan to use powersave features.  |
+--------------------------------------------------------------------------------------------------
----+
Building Bootloader
Finished Building Bootloader

Processing resources
Creating security credentials
Making DCT image
Compiling App_aws_demo_subcriber
Compiling Platform_BCM943364WCD1
Compiling WICED
Compiling Lib_MQTT_Client
Compiling Lib_cJSON
Compiling Lib_HTTP_Server
Compiling Lib_DNS_Redirect_Daemon
Compiling Lib_DNS
Compiling Lib_SPI_Flash_Library_BCM943364WCD1
Compiling Lib_GPIO_button
Compiling WWD_ThreadX_Interface
Compiling WICED_ThreadX_Interface
Compiling WWD_for_SDIO_ThreadX
Compiling Supplicant_BESL
Compiling NetX_Duo
Compiling Lib_Wiced_RO_FS
Compiling STM32F4xx
Compiling Lib_Linked_List
Compiling Lib_TLV
Compiling Lib_base64
Compiling Lib_crypto_open
Compiling Lib_micro_ecc
Compiling WWD_NetX_Duo_Interface
Compiling WICED_NetX_Duo_Interface
Compiling common_GCC
Compiling STM32F4xx_Peripheral_Drivers
Compiling Lib_Ring_Buffer
Compiling Lib_DHCP_Server
Compiling STM32F4xx_Peripheral_Libraries
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
App_aws_demo_subcriber.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Platform_BCM943364WCD1.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
WICED.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
```

```
Lib_MQTT_Client.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_cJSON.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_HTTP_Server.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_DNS_Redirect_Daemon.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_DNS.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_SPI_Flash_Library_BCM943364WCD1.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_GPIO_button.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
WWD_ThreadX_Interface.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
WICED_ThreadX_Interface.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
WWD_for_SDIO_ThreadX.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Supplicant_BESL.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
NetX_Duo.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_Wiced_RO_FS.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
STM32F4xx.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_Linked_List.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_TLV.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_base64.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_crypto_open.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_micro_ecc.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
WWD_NetX_Duo_Interface.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
WICED_NetX_Duo_Interface.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
common_GCC.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
STM32F4xx_Peripheral_Drivers.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_Ring_Buffer.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
Lib_DHCP_Server.a
Making build/demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1/libraries/
STM32F4xx_Peripheral_Libraries.a
Making demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1.elf
Making demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1.bin
demo.aws_iot.light_sensor_controlled_window.subscriber-BCM943364WCD1
----------------------------------|---------|---------|
                                  |         | Static  |
              Module              |  Flash  |  RAM    |
----------------------------------+---------+---------|
App                               |   8830  |   8587  |
base64                            |    523  |      0  |
cJSON                             |   1901  |     12  |
crypto_open                       |  46752  |      0  |
DHCP_Server                       |   1578  |    132  |
DNS                               |   1500  |     44  |
DNS_Redirect_Daemon               |    645  |      0  |
Host MCU-family library           |  13952  |   2484  |
HTTP_Server                       |   3334  |      0  |
Interrupt Vectors                 |    388  |      0  |
libc                              |  41352  |   3141  |
Linked_List                       |    504  |      0  |
micro_ecc                         |   5848  |      0  |
MQTT_Client                       |   5493  |    736  |
Networking                        |   5648  |  13184  |
NetX-Duo - Interfaces & Stacks    |      0  |     16  |
Other                             |  44398  |    535  |
Packet Buffers                    |      0  |  23086  |
platform                          |   1196  |    168  |
RAM Initialisation                |   2740  |      0  |
resources                         |  55212  |      0  |
Ring_Buffer                       |     92  |      0  |
SPI_Flash_Library_BCM943364WCD1   |    516  |      0  |
Startup Stack & Link Script fill  |     73  |     19  |
Supplicant - BESL                 |  33407  |    536  |
ThreadX                           |   8600  |    396  |
TLV                               |    206  |      0  |
WICED                             |   4487  |    848  |
WWD                               |  14985  |   3036  |
----------------------------------+---------+---------|
TOTAL (bytes)                     | 301420  |  56960  |
----------------------------------|---------|---------|

Downloading Bootloader ...
Download complete

Downloading DCT ...
Download complete

Downloading Application ...
Download complete

Resetting target
Target running
Build complete
Making .gdbinit
```

During firmware download, the blue JTAG LED (D4) illuminates to indicate that a JTAG firmware download is in progress.

If the board is not recognized by the programming tools, it may be necessary to disconnect and then reconnect it to the computer before trying again. A message similar to the following message indicates that there was an error with the download process:

```
"**** OpenOCD failed - ensure you have installed the driver from the drivers directory, and that the
debugger is not running **** In Linux this may be due to USB access permissions. In a virtual machine
it may be due to USB passthrough settings ****"
Resetting target
make: *** [run] Error 1
```

Running an App on the Demo Platform

This section assumes that you have successfully completed "Building and Downloading an App to the Demo Platform" on page 34, and that an application is running on the WICED module.

Several methods are available to verify that the application is working. After power-on-reset, the application prints status messages to the USB UART of the WICED evaluation board.

To verify printing, follow the instructions in "Configuring a Terminal Application" on page 42 to setup a terminal application such as PuTTY (Windows) or CoolTerm (OS X). Start the terminal application, connect to the WICED evaluation board, and then press the reset button on the board.

For an IoT device running the light_sensor_controlled_window publisher app, the following text is indicative of what might be displayed to a terminal while the application runs:

```
Starting WICED v3.4.0_AWS
Platform BCM94343W_AVN initialised
Started ThreadX v5.6
Initialising NetX_Duo v5.7_sp2
Creating Packet pools
WWD SDIO interface initialised
WLAN MAC Address : B0:38:29:3A:42:BE

WLAN Firmware    : wl0: Jun 24 2015 17:47:50 version 7.45.41 (r554772) FWID 01-5eb3f26a

 Please wait, connecting to network...
(To return to SSID console screen, hold USER switch for 5 seconds during RESET to clear DCT
configuration)
Joining : vikram
Successfully joined : vikram
Obtaining IPv4 address via DHCP
DHCP CLIENT hostname WICED IP
IPv4 network ready IP: 192.168.2.3
Setting IPv6 link-local address
IPv6 network ready IP: FE80:0000:0000:0000:B238:29FF:FE3A:42BE
[MQTT] Connecting to broker 52.6.53.137 ...

Thing Name: lightbulb
Shadow State Topic: $aws/things/lightbulb/shadow/update
Shadow Delta Topic: $aws/things/lightbulb/shadow/update/delta
Reading the certificate and private key from DCT...
[MQTT] Connecting to MQTT Broker...
[MQTT] Successfully connected MQTT Broker
Publish SUCCEEDED for topic [$aws/things/lightbulb/shadow/update]
Subscribe SUCCEEDED for topic [$aws/things/lightbulb/shadow/update/delta]
Subscribe SUCCEEDED for topic [lightbulb]
[MQTT] Received OFF  for TOPIC : lightbulb
Requested WICED_BULB State[OFF] Current WICED_BULB State [OFF]
button_pressed 1
Publish SUCCEEDED for topic [$aws/things/lightbulb/shadow/update]
[MQTT] Received ON  for TOPIC : lightbulb
Requested WICED_BULB State[ON] Current WICED_BULB State [OFF]
[motor_thread_main] [225] motor_run [1]
Publish SUCCEEDED for topic [$aws/things/lightbulb/shadow/update]
```

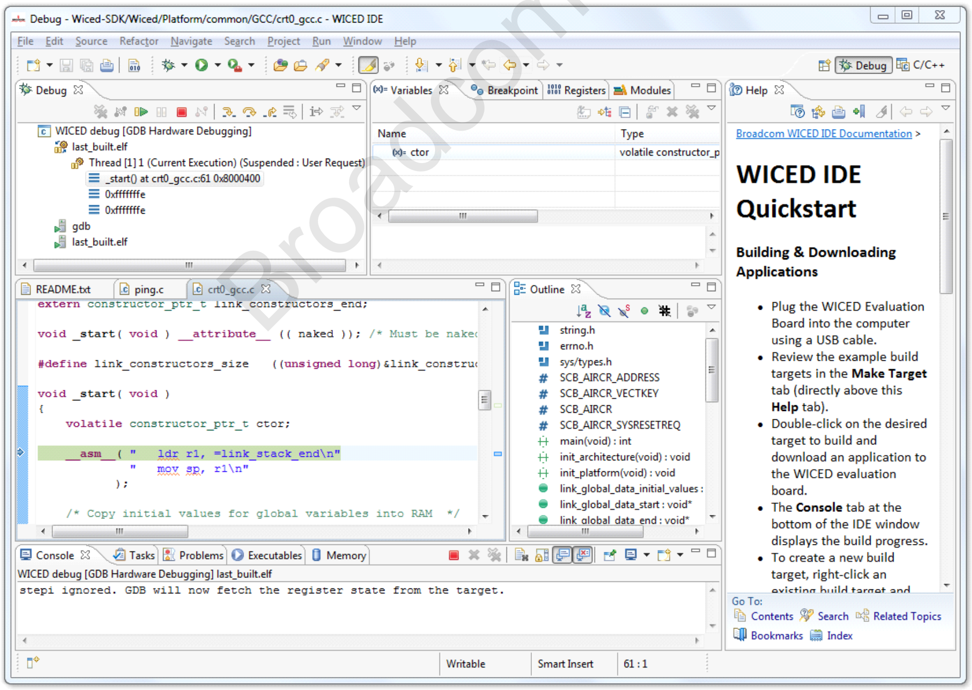
```
[MQTT] Received OFF  for TOPIC : lightbulb
Requested WICED_BULB State[OFF] Current WICED_BULB State [ON]
[motor_thread_main] [234] motor_run [0]
Publish SUCCEEDED for topic [$aws/things/lightbulb/shadow/update]
[MQTT] Received ON  for TOPIC : lightbulb
Requested WICED_BULB State[ON] Current WICED_BULB State [OFF]
[motor_thread_main] [225] motor_run [1]
Publish SUCCEEDED for topic [$aws/things/lightbulb/shadow/update]
[MQTT] Received OFF  for TOPIC : lightbulb
Requested WICED_BULB State[OFF] Current WICED_BULB State [ON]
[motor_thread_main] [234] motor_run [0]
Publish SUCCEEDED for topic [$aws/things/lightbulb/shadow/update]
```

Debugging an App on the Demo Platform

The WICED development system supports single-step thread-aware interactive application debugging with the WICED IDE. Brief instructions to start debugging an application are provided in the WICED IDE Help tab. The following example demonstrates how to start debugging an example application (the 'ping' application is used in the remaining steps below).

1. Setup a breakpoint:
   a. Using the Project Explorer tab on the left of the IDE, navigate to the 'WICED-SDK/Apps/snip/ping' directory. Double-click on 'ping.c' to open the source file in the WICED IDE editor window.
   b. Scroll to the 'application_start' function and locate the call to 'wiced_init' (at approximately line 180).
   c. Double-click in the column to the left of the 'wiced_init' function to place a break point at this function (alternately right-click in the column and select 'Toggle Breakpoint'). A green dot appears in the column.
2. Setup the 'ping' debug build target: In the WICED IDE Make Target tab, right click on the 'ping-BCM943362WCD4 download run' build target and click Edit (alternately, copy-paste and then edit a new build target). Change the build target to 'ping-BCM943362WCD4-debug download'.
3. Build and download the debug firmware image: Double-click the newly created 'ping' debug target, the debug firmware image builds and downloads to the 'BCM943362WCD4' module on the WICED evaluation board.
4. Launch the debugger: Click the green bug icon on the WICED IDE toolbar (or press F11 on Windows® systems). The debugger starts. If the WICED IDE displays a Confirm Perspective Switch dialogue box, click Yes to show the debug view.

   To confirm the debugger is running, verify the blue JTAG LED on the WICED evaluation board is flashing, and that the WICED IDE looks similar to the following screen capture.



WICED debug [GDB Hardware Debugging]
last_built.elf
Thread [1] 1 (Current Execution) (Suspended : User Request)
_start() at crt0_gcc.c:61 0x8000400
0xfffffffe
0xfffffffe
gdb
last_built.elf

Name	Type
ctor	volatile constructor_p

```
extern constructor_ptr_t link_constructors_end;

void _start( void ) __attribute__ (( naked )); /* Must be naked

#define link_constructors_size   ((unsigned long)&link_construc

void _start( void )
{
    volatile constructor_ptr_t ctor;

    __asm__( "   ldr r1, =link_stack_end\n"
             "   mov sp, r1\n"
           );

    /* Copy initial values for global variables into RAM  */
```

string.h
errno.h
sys/types.h
SCB_AIRCR_ADDRESS
SCB_AIRCR_VECTKEY
SCB_AIRCR
SCB_AIRCR_SYSRESETREQ
main(void) : int
init_architecture(void) : void
init_platform(void) : void
link_global_data_initial_values :
link_global_data_start : void*
link_global_data_end : void*

WICED debug [GDB Hardware Debugging] last_built.elf

```
stepi ignored. GDB will now fetch the register state from the target.
```

Broadcom WICED IDE Documentation >

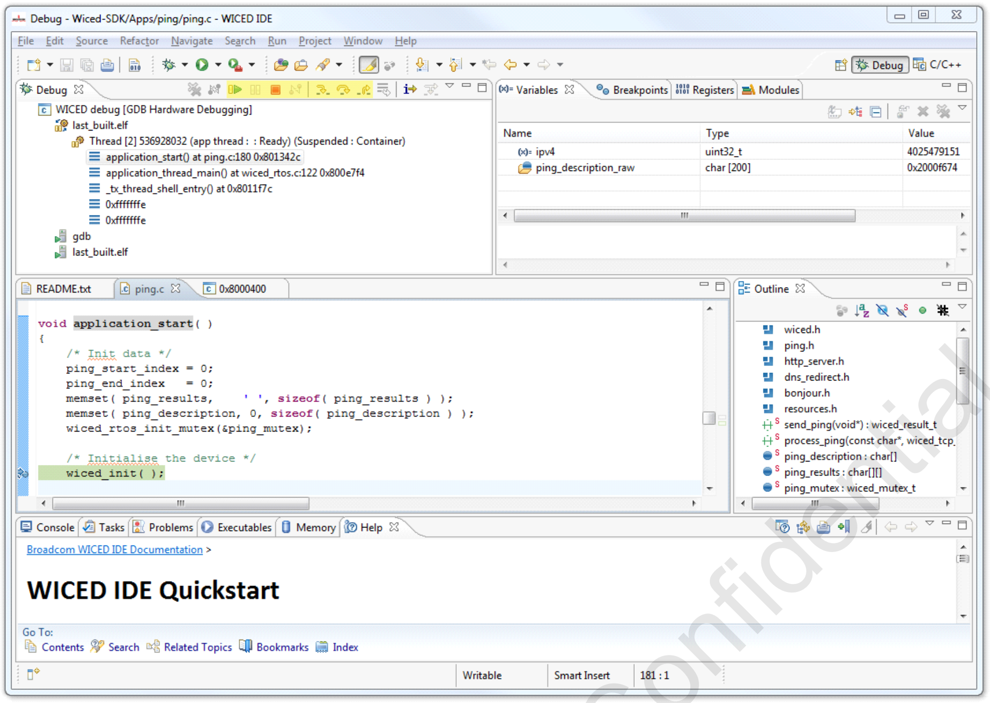
WICED IDE Quickstart

Building & Downloading Applications

- Plug the WICED Evaluation Board into the computer using a USB cable.
- Review the example build targets in the **Make Target** tab (directly above this **Help** tab).
- Double-click on the desired target to build and download an application to the WICED evaluation board.
- The **Console** tab at the bottom of the IDE window displays the build progress.
- To create a new build target, right-click an existing build target and

Go To: Contents Search Related Topics Bookmarks Index

5. Run to a breakpoint: To run the application to the 'wiced_init' breakpoint configured in Step 1, click the yellow/ green pause/play button in the Debug tab. The Debug tab control buttons are highlighted in yellow in the following screen capture.


Debug - Wiced-SDK/Apps/ping/ping.c - WICED IDE
File Edit Source Refactor Navigate Search Run Project Window Help
Debug C/C++
Debug
Variables Breakpoints Registers Modules
README.txt ping.c 0x8000400
Outline
Console Tasks Problems Executables Memory Help
Broadcom WICED IDE Documentation >
WICED IDE Quickstart
Go To: Contents Search Related Topics Bookmarks Index
Writable Smart Insert 181 : 1

```
WICED debug [GDB Hardware Debugging]
    last_built.elf
        Thread [2] 536928032 (app thread : : Ready) (Suspended : Container)
            application_start() at ping.c:180 0x801342c
            application_thread_main() at wiced_rtos.c:122 0x800e7f4
            _tx_thread_shell_entry() at 0x8011f7c
            0xfffffffe
            0xfffffffe
    gdb
    last_built.elf
```

Name	Type	Value
ipv4	uint32_t	4025479151
ping_description_raw	char [200]	0x2000f674

```
void application_start( )
{
    /* Init data */
    ping_start_index = 0;
    ping_end_index   = 0;
    memset( ping_results,    ' ', sizeof( ping_results ) );
    memset( ping_description, 0, sizeof( ping_description ) );
    wiced_rtos_init_mutex(&ping_mutex);

    /* Initialise the device */
    wiced_init( );
```

```
wiced.h
ping.h
http_server.h
dns_redirect.h
bonjour.h
resources.h
send_ping(void*) : wiced_result_t
process_ping(const char*, wiced_tcp_
ping_description : char[]
ping_results : char[][]
ping_mutex : wiced_mutex_t
```

6. View a breakpoint: When the debugger halts at a breakpoint, the WICED IDE does not automatically switch to the current thread as the context for the debugger. It is necessary to manually check the current task in the running thread is selected before analyzing debug information.

   To find the breakpoint, click on the 'application_start()' function under Thread[2] in the Debug tab. The 'ping.c' source file opens and the wiced_init function is highlighted in green to show where the program halted.

7. Step program execution: Step Into, Step Over and Step Return options are available in the WICED IDE Run menu. Alternately, Step shortcut icons are provided in the Debug tab, and on Windows® Systems, by pressing F5, F6 and F7, respectively.

8. Stop debugging: To stop debugging, click the square red stop button in the Debug tab.

Note: If the debugger fails to launch, it may be necessary to terminate an existing debug process. On Windows®, press Ctrl-Alt-Delete to open the Windows® Task Manager, then select the Processes tab. Search for, and terminate, all 'arm-none-eabi-gdb' processes.

Configuring a Terminal Application

The following instructions describe how to obtain and install a serial terminal application for use on computers running a Windows or OS X operations system. Broadcom recommends using PuTTY for Windows systems and CoolTerm for OS X systems, however other equivalent applications may work equally well.

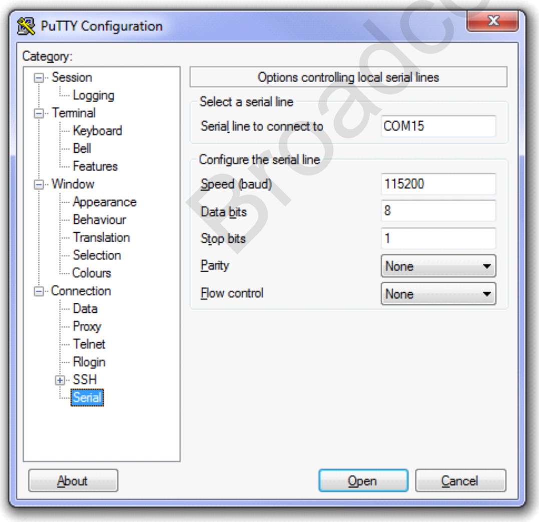
Ensure the WICED Development System is already installed on the computer, then plug the WICED evaluation board into the computer using a USB cable before continuing.

Setting Up PuTTY for Windows

PuTTY is available as a free download from http://putty.org. Download and install PuTTY.

The following procedure describes how to establish a UART serial interface between PuTTY and the WICED evaluation board.

1. Start the PuTTY application. The PuTTY Configuration window opens. Set the configuration options as follows:
   - Category: Serial
   - Serial line to connect to: type in the COM port that was assigned after the USB and serial port drivers were installed. Refer to Step 2(a) in Section 3.1.3.
   - Speed (baud): 115200
   - Data bits: 8
   - Stop bits: 1
   - Parity: None
   - Flow control: None



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM15

Configure the serial line

Speed (baud): 115200

Data bits: 8

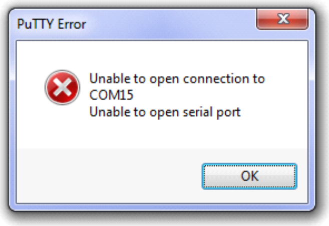
Stop bits: 1

Parity: None

Flow control: None

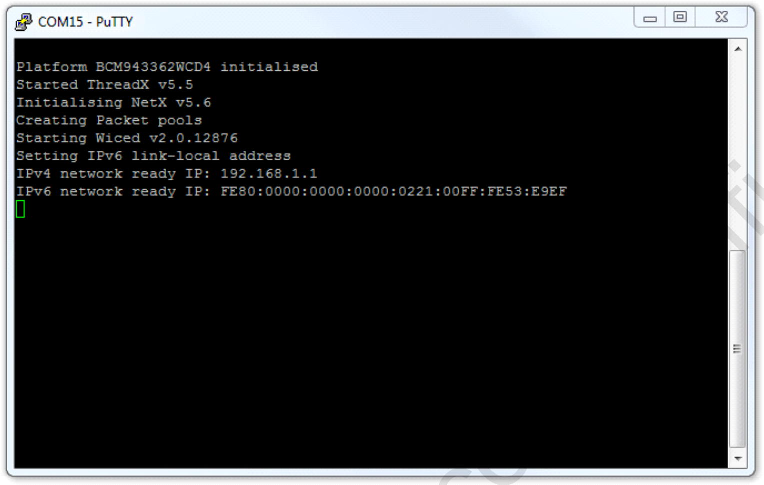
2. In the Category pane, select **Session**.
3. Under the Connection type, select the Serial option then click **Open**.

4. A blank terminal window opens with the selected COM port specified in the window title. If the specified COM port is incorrect or unavailable, PuTTY displays an error message as shown in the following screen capture. If this happens, verify the correct COM port has been selected and try again.



Unable to open connection to
COM15
Unable to open serial port

5. Assuming the ping application is running on the WICED evaluation board, press the reset button on the board to view application prints during the boot and run process.



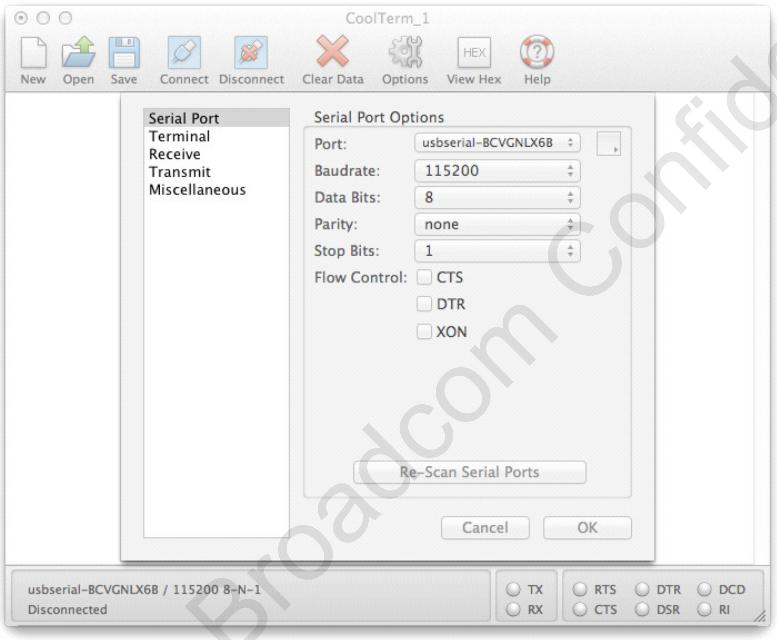
```
Platform BCM943362WCD4 initialised
Started ThreadX v5.5
Initialising NetX v5.6
Creating Packet pools
Starting Wiced v2.0.12876
Setting IPv6 link-local address
IPv4 network ready IP: 192.168.1.1
IPv6 network ready IP: FE80:0000:0000:0000:0221:00FF:FE53:E9EF
```

Setting Up CoolTerm for OS X

CoolTerm is available as a free download from http://freeware.the-meiers.org/CoolTermMac.zip. Download and install CoolTerm.

The following procedure describes how to establish a UART serial interface between CoolTerm and the WICED evaluation board.

1. Start the CoolTerm application and click the **Options** menu icon. The CoolTerm Configuration window opens. Set the Serial Port configuration options as follows:
   - Port: usbserial-XXXXXXXX
   - Baudrate: 115200
   - Data bits: 8
   - Parity: none
   - Stop bits: 1
   - Flow control: Deselect all options

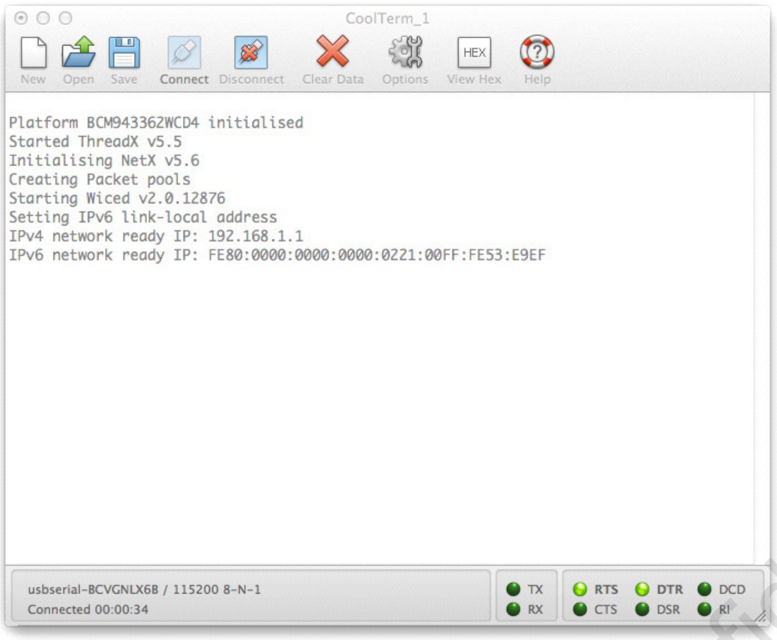


Serial Port Options
Port: usbserial-BCVGNLX6B
Baudrate: 115200
Data Bits: 8
Parity: none
Stop Bits: 1
Flow Control: CTS DTR XON

usbserial-BCVGNLX6B / 115200 8-N-1
Disconnected

2. Click **OK**.
3. Click the **Connect** menu icon. The CoolTerm application connects to the WICED evaluation board.

4. Assuming the ping application is running on the WICED evaluation board, press the reset button on the board to view application prints during the boot and run process.



```
Platform BCM943362WCD4 initialised
Started ThreadX v5.5
Initialising NetX v5.6
Creating Packet pools
Starting Wiced v2.0.12876
Setting IPv6 link-local address
IPv4 network ready IP: 192.168.1.1
IPv6 network ready IP: FE80:0000:0000:0000:0221:00FF:FE53:E9EF
```

usbserial-BCVGNLX6B / 115200 8-N-1
Connected 00:00:34

Section 4: Sample AWS IoT Applications

The sample applications provided in this section have been built to work with the 4343W IoT starter kit.

Note: The source code for the applications in this section comes packaged with WICED SDK 3.4.0-AWS. The relevant source code is also available at the following link: https://github.com/CloudConnectKits/BCM4343W.

shadow app

This application is a single application. It publishes the state of an LED and subscribes to the state of the overall device. It essentially tests the functionality of the AWS Things Shadow.

After the 4343W IoT starter kit board is configured as an AWS IoT device that will work with the AWS IoT service (see "Configuring an IoT Device" on page 53), this application, which has been preloaded, will run automatically.

To test the functionality of the application:

1. Press **SW2** on the board to toggle the on/off state of the USER1 LED (D3).
2. Monitor the shadowed state of the IoT device using the AWS IoT management console.

Before running this preloaded application or any of the other applications, perform the procedural steps in Section 5: "AWS IoT Service and Device Setup," on page 47.

pub_sub app

This application is actually two applications. One application runs on a device that publishes to a specific AWS topic each time a button is pressed. The second application runs on a device that subscribes to the same topic and toggles an LED based on received messages.

light_sensor_controlled_window

This application is also actually two applications. One application runs on a remote sensor device that monitors and publishes light intensity to a specific AWS topic (WICED_BULB). The second application runs on a controlled device that subscribes to the WICED_BULB topic that the remote sensor device publishes to. The state of the remote sensor device is shadowed using the AWS Things Shadow.

temperature_sensor_controlled_window

This application is also actually two applications. One application runs on a remote sensor device that computes the delta between room temperature and the outside temperature retrieved from an Internet source (such as weather.com). Based on the computation, the remote sensor device publishes LIGHT ON or LIGHT OFF to the AWS WICED_BULB topic. A second application runs on another device that subscribes to the AWS WICED_BULB topic and toggles the state of an LED based on received messages.

Section 5: AWS IoT Service and Device Setup

Introduction

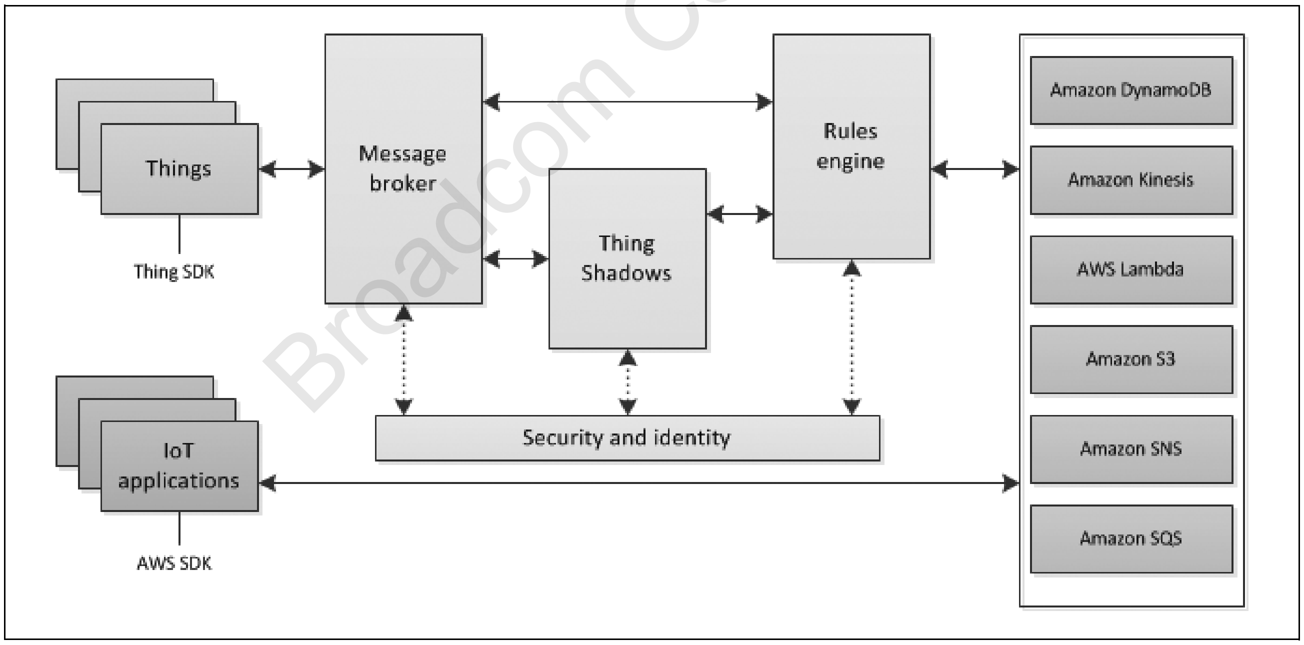
The AWS IoT service enables secure, bidirectional communication between IoT devices, sometimes referred to as Internet-connected things or simply *things* (sensors, actuators, devices, applications, etc.), and the cloud over MQTT and HTTP.

Things are authenticated using AWS IoT service-provided X.509 certificates. Once a certificate is provisioned and activated it can be installed on a thing. The thing will then use that certificate to send all requests to AWS MQTT. Authorization is controlled by JSON policy files that allow you to specify which resources a specific device (certificate) may access.

You can interact with AWS MQTT in a number of ways: The AWS MQTT Thing SDK allows you to write applications in C that run on Internet-connected things. The AWS Command-Line Interface (CLI) allows you to configure AWS services. The AWS SDKs allow you to write applications on top of AWS MQTT.

Figure 17 shows the high-level block diagram of AWS IoT service capabilities.

Figure 17: AWS High-Level Block Diagram



In Figure 17, things are any clients such as microcontrollers, sensors, actuators, mobile devices, or applications that connect to the AWS cloud.

Thing SDK makes it simpler to write code running on Internet-connected things to communicate with the AWS MQTT service.

The gateway allows things to securely connect using AWS MQTT or HTTP. Things are authenticated using X.509 certificates or Identity and Access Management (IAM) roles.

The message broker is an MQTT broker that enables things to publish and/or subscribe to MQTT messages.

The rules engine can be used to configure the continuous processing of MQTT messages received from things. This includes transforming, augmenting, or routing messages to other services. Rules are written in an SQL-like syntax. They enable filtering, aggregating, and forwarding messages, as well as taking appropriate actions on receiving relevant data, such as invoking Lambda functions or writing message data to DynamoDB tables.

Thing Shadows provide a virtual representation of a thing. Things can be any type of AWS MQTT clients such as microcontrollers, sensors, actuators, mobile devices, or applications. Thing Shadows enable you to request the state or update the state of a thing using a REST API or AWS CLI. The state of a thing is represented as a JSON object.

Getting Started with the AWS IoT Service

For complete information on getting started using the AWS IoT service, see: https://us-west-2.console.aws.amazon.com/iot/home.

The following steps summarize what a user should do to get started using the AWS IoT service with a device:

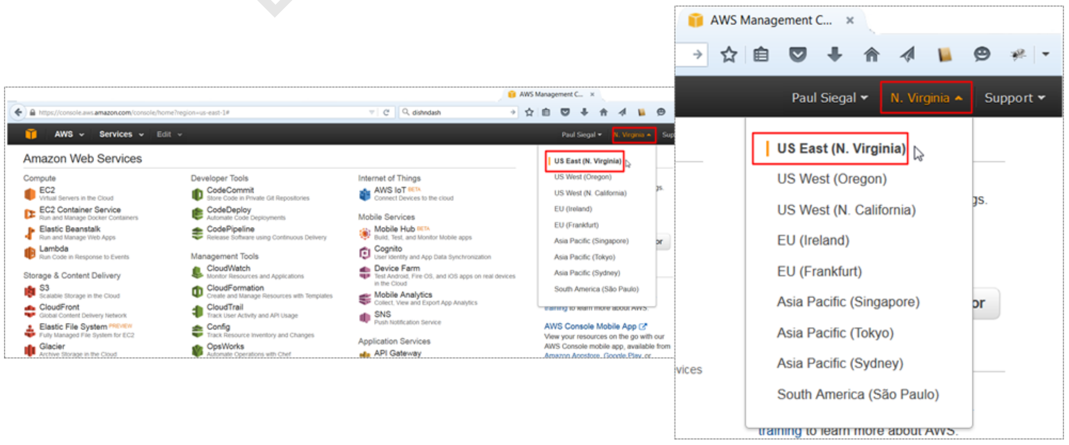
1. Go to the AWS IoT service by clicking on or entering http://aws.amazon.com/iot/ in a browser.
2. Create an AWS account by clicking on **Get started with AWS IoT**.

Note: A credit card number is required upon creating an account. The first year of service is free as long as the published and delivered message count does not exceed 250,000 messages per month.

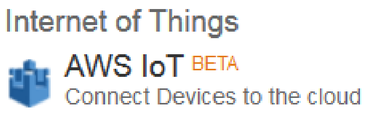
3. Sign in to the AWS Management Console by clicking on or navigating to https://aws.amazon.com/console/ and then clicking on **Sign in to the AWS Console**.:

Sign in to the AWS Console

4. In the webpage that opens, select **US East (N. Virginia)** as the server region for management console use during the AWS IoT beta:



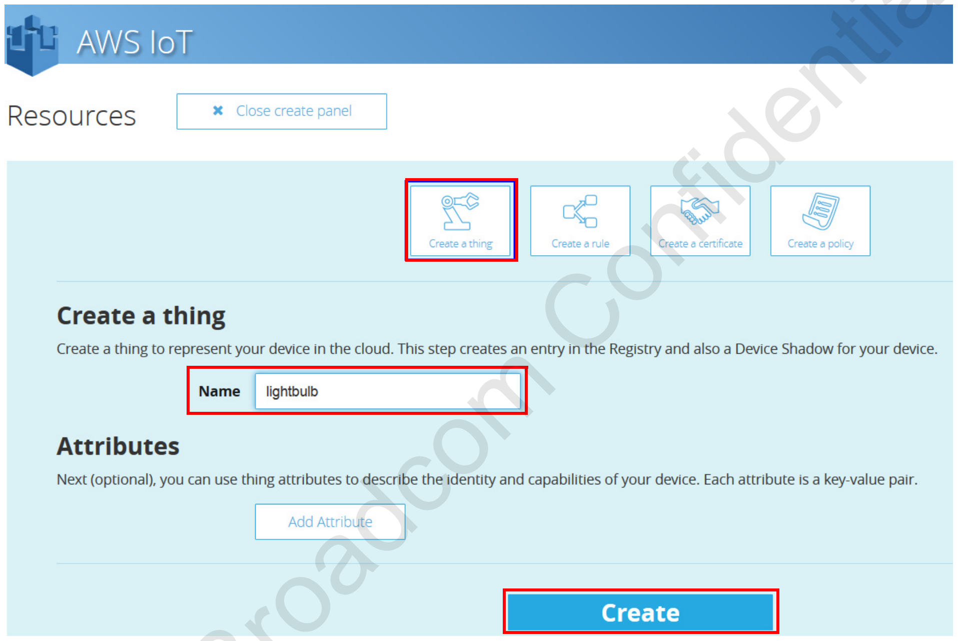
5. In the Internet of Things column, click **AWS IoT** $^{\textbf{BETA}}$ to start using the management console



6. In the webpage that opens, click **Get started**:

7. In the AWS IoT management console webpage:
   a. Click **Create a resource**.

   b. Click **Create a thing**, enter the name of the thing (for example, lightbulb) then click **Create**.



Create a thing

Create a thing to represent your device in the cloud. This step creates an entry in the Registry and also a Device Shadow for your device.

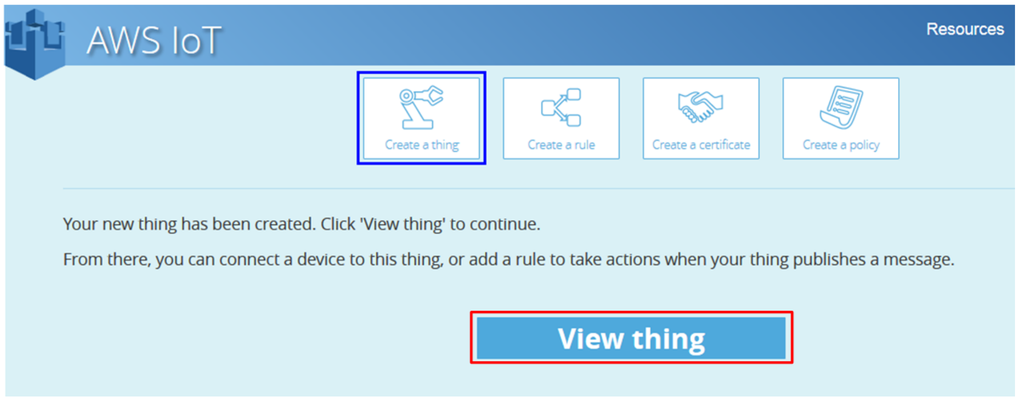
Name: lightbulb

Attributes

Next (optional), you can use thing attributes to describe the identity and capabilities of your device. Each attribute is a key-value pair.

Note: If a *Failed to Create Thing* error message appears upon clicking Create, see "Resolving a Failed-to-Create-Thing Error Message" on page 51

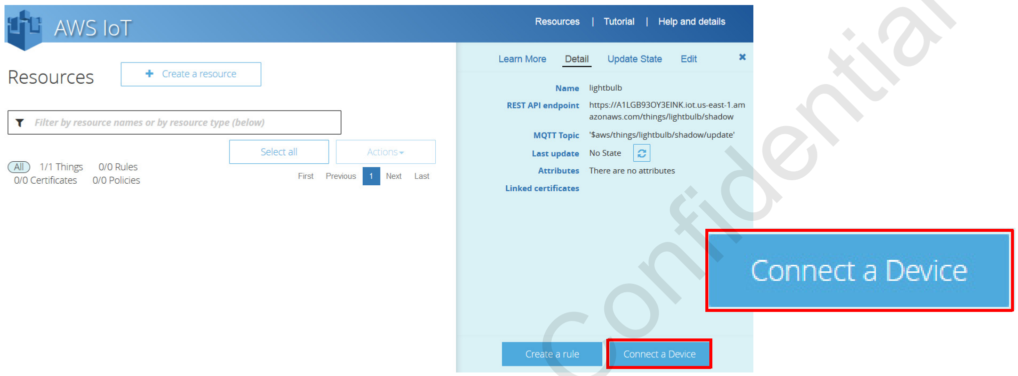
c. Click **View thing** in order to connect a device.



Your new thing has been created. Click 'View thing' to continue.

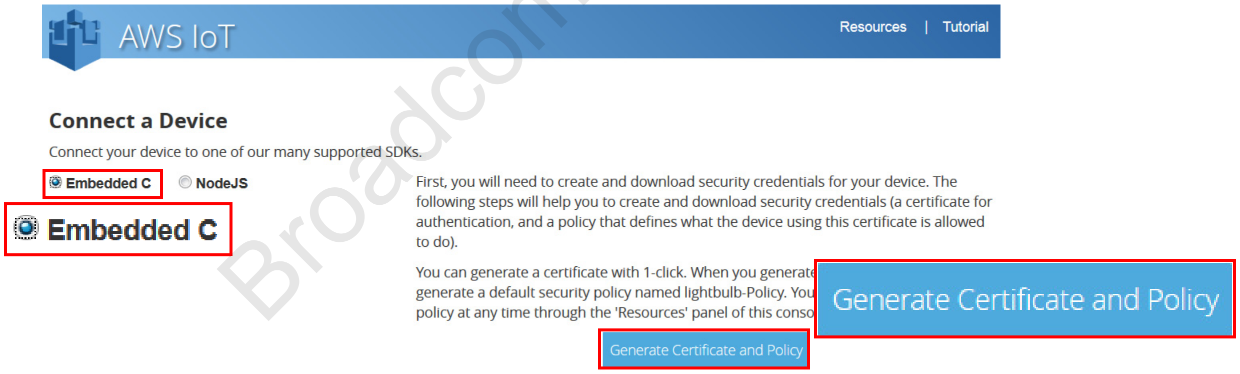
From there, you can connect a device to this thing, or add a rule to take actions when your thing publishes a message.

d. Click **Connect a Device**.


AWS IoT
Resources | Tutorial | Help and details
Resources
Create a resource
Filter by resource names or by resource type (below)
Select all
Actions
All 1/1 Things 0/0 Rules
0/0 Certificates 0/0 Policies
First Previous 1 Next Last
Learn More Detail Update State Edit
Connect a Device
Create a rule
Connect a Device

Name: lightbulb
REST API endpoint: https://A1LGB93OY3EINK.iot.us-east-1.amazonaws.com/things/lightbulb/shadow
MQTT Topic: '$aws/things/lightbulb/shadow/update'
Last update: No State
Attributes: There are no attributes
Linked certificates

e. Click **Embedded C** as the supported SDK and then click **Generate Certificate and Policy**.


AWS IoT
Resources | Tutorial
Embedded C
Generate Certificate and Policy
Generate Certificate and Policy

Connect a Device

Connect your device to one of our many supported SDKs.

[✓ Embedded C] NodeJS

First, you will need to create and download security credentials for your device. The following steps will help you to create and download security credentials (a certificate for authentication, and a policy that defines what the device using this certificate is allowed to do).

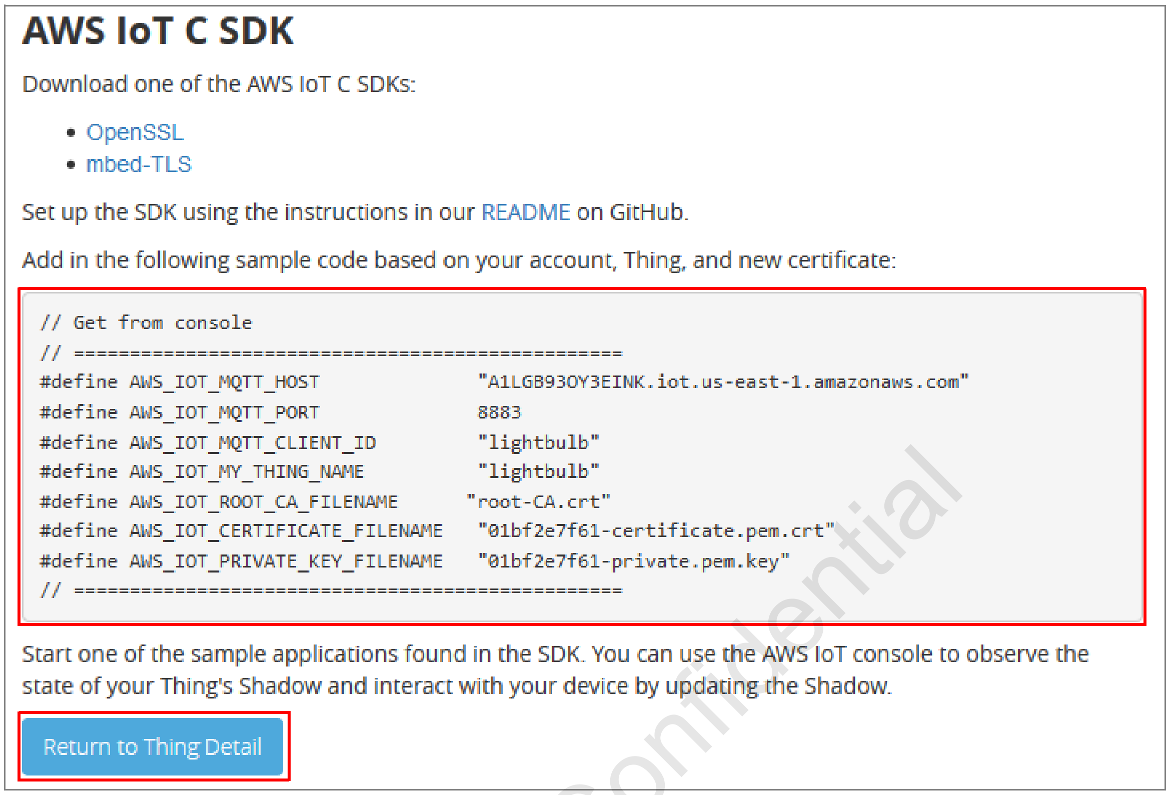
You can generate a certificate with 1-click. When you generate generate a default security policy named lightbulb-Policy. You policy at any time through the 'Resources' panel of this conso

f. Sequentially click on the **Download Public Key**, **Download Private Key**, and **Download Certificate** links and for each click **Save File**, **OK**, navigate to the directory where the credentials should be stored, and then click **Save**.

Note: The certificate and private key PEM files will be needed to configure each IoT device (see "Configuring an IoT Device" on page 53) that will use the AWS IoT service.

g. Click **Confirm & Start Connecting** after saving the keys and the certificate to a known directory.

h. Save the #define's displayed on the page and then click **Return to Thing Detail**.



AWS IoT C SDK

Download one of the AWS IoT C SDKs:

- OpenSSL
- mbed-TLS

Set up the SDK using the instructions in our README on GitHub.

Add in the following sample code based on your account, Thing, and new certificate:

```
// Get from console
// =================================================
#define AWS_IOT_MQTT_HOST              "A1LGB93OY3EINK.iot.us-east-1.amazonaws.com"
#define AWS_IOT_MQTT_PORT              8883
#define AWS_IOT_MQTT_CLIENT_ID         "lightbulb"
#define AWS_IOT_MY_THING_NAME          "lightbulb"
#define AWS_IOT_ROOT_CA_FILENAME      "root-CA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME   "01bf2e7f61-certificate.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "01bf2e7f61-private.pem.key"
// =================================================
```

Start one of the sample applications found in the SDK. You can use the AWS IoT console to observe the state of your Thing's Shadow and interact with your device by updating the Shadow.

Note: At this point, the IoT device (that is, the Thing) that you initially created in Step b on page 49 is provisioned and a certificate and policy have been attached to it.

Next, configure your IoT device (see "Configuring an IoT Device" on page 53).

Resolving a Failed-to-Create-Thing Error Message

If you encounter a *Failed to Create Thing* error message when attempting to create an IoT device, there is a possibility that you did not complete the full sign-up process. To resolve this potential issue:

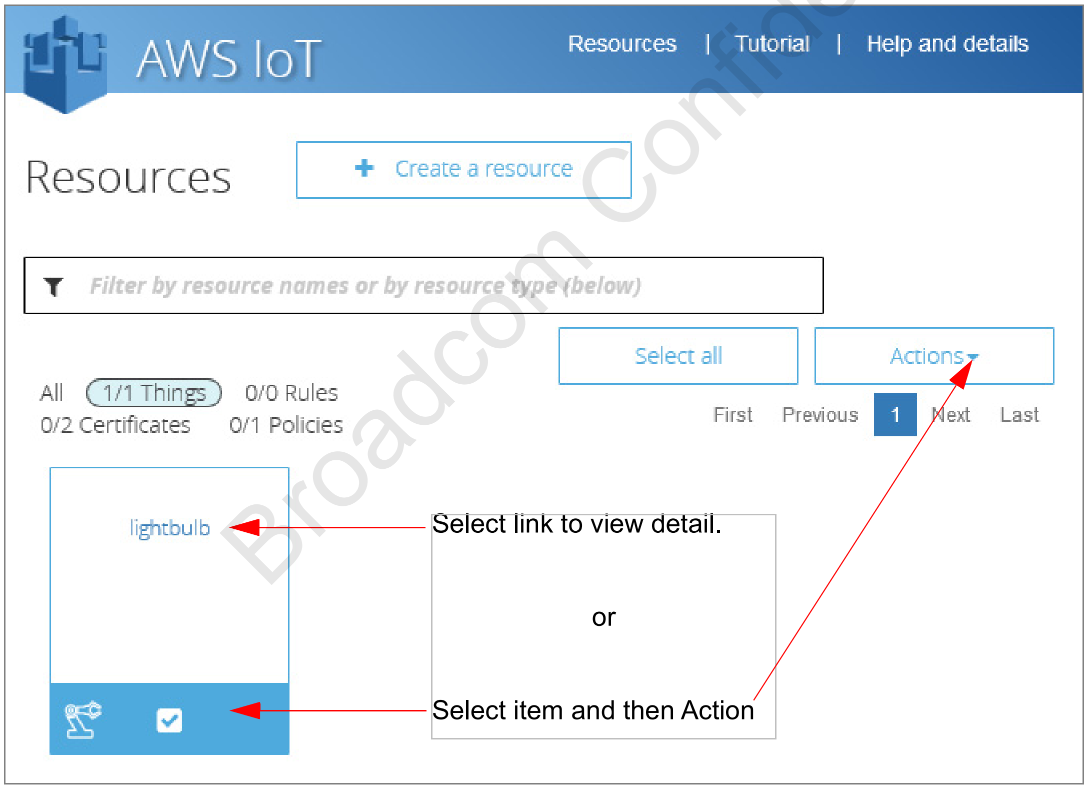
1. Click on or enter http://aws.amazon.com/iot/ in a browser.
2. In the upper-right corner of the displayed webpage, click on **Complete Sign Up**.
3. Follow the onscreen instructions to complete the sign-up process.

Viewing and Editing IoT Device Information

To view and/or edit information associated with a previously provisioned IoT device, such as the device provisioned in "Getting Started with the AWS IoT Service" on page 48:

1. Go to the management console by clicking on or navigating to https://aws.amazon.com/console/ or by clicking on the Console Home button ( ).
2. In the Internet of Things column, click **AWS IoT** **BETA**.

3. In the AWS IoT management display, click on **All** or any combination of **Things**, **Rules**, **Certificates**, and **Policies**.
4. Click on the link in any of the items presented to view the associated detail or select an item (thing, rule, certificate, or policy) by clicking in the shaded area of the item and then clicking **Actions** to take a specific action.

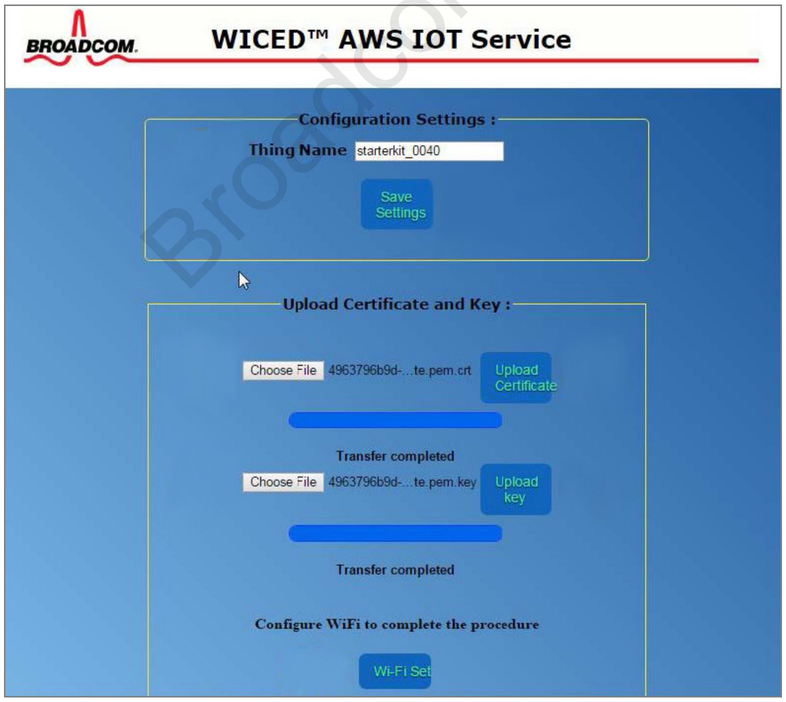

AWS IoT
Resources | Tutorial | Help and details
Resources
+ Create a resource
Filter by resource names or by resource type (below)
Select all
Actions
All 1/1 Things 0/0 Rules
0/2 Certificates 0/1 Policies
First Previous 1 Next Last
lightbulb
Select link to view detail.
or
Select item and then Action

Configuring an IoT Device

Note: In this section, the carrier board provided with the BCM4343W IoT Starter Kit (and actually labeled BCM4343W IoT Starter Kit) will be referred to as the IoT device (or the thing).

To prepare the IoT device so that it can be used with the AWS IoT service, perform the following steps:

1. Connect the USB cable provided with the BCM4343W IoT Starter Kit from the IoT device to a PC. Doing so will cause the 3.3V power LED (D6) to illuminate.
2. Establish a Wi-Fi connection from the PC to the WICED_AWS SoftAP. WICED_AWS is the SSID of the IoT device while it's operating as a SoftAP.
   Note: When prompted, enter **123454678** as the password for the Wi-Fi connection.
3. In the address field of an Internet browser, enter **192.168.0.1** to configure the IoT device.
4. In the WICED AWS IOT Service webpage:
   a. In the Configuration Settings **Thing Name** field, enter the name of the thing (for example, lightbulb) then click **Save Settings**.
   b. In the Upload Certificate and Key portion of the webpage:
   – Click **Choose File** (next to Upload Certificate), browse to and select the certificate downloaded during this step (Step f on page 50), then click **Upload Certificate** and wait for the transfer to complete.
   – Click **Choose File** (next to Upload key), browse to and select the private key downloaded during this step (Step f on page 50), then click **Upload key** and wait for the transfer to complete.
   – Click **Wi-Fi Set**.

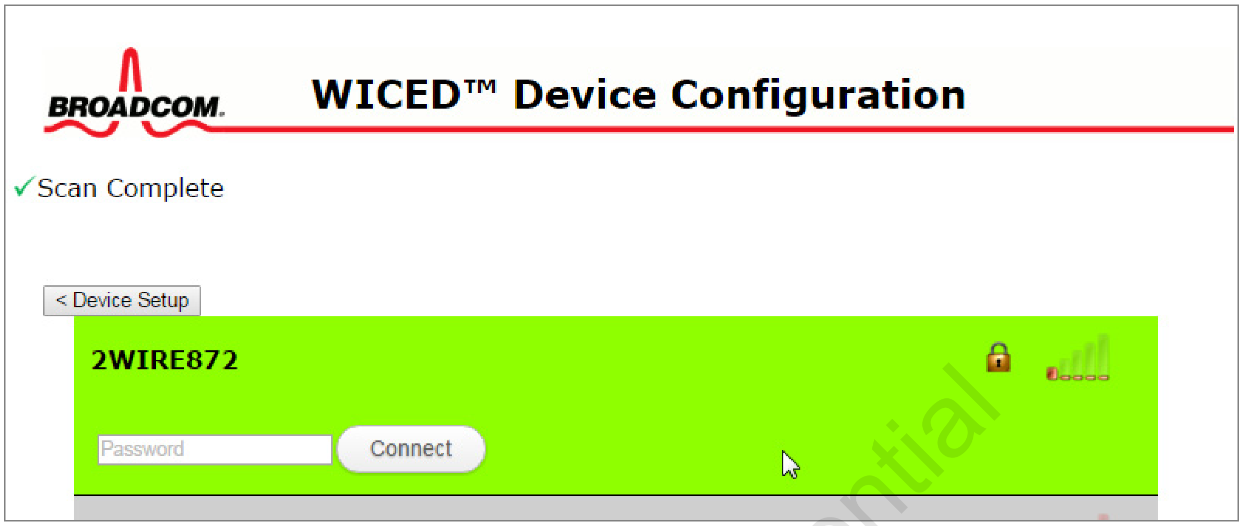

WICED™ AWS IOT Service

Configuration Settings :
Thing Name: starterkit_0040

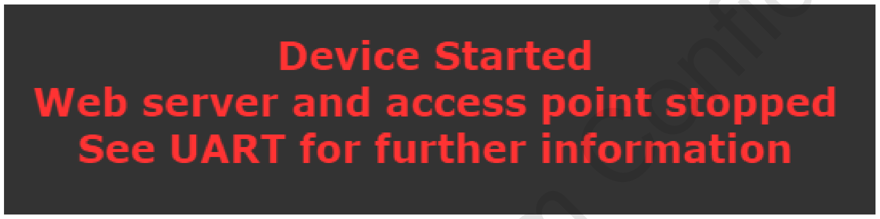
Upload Certificate and Key :
4963796b9d-...te.pem.crt
Transfer completed
4963796b9d-...te.pem.key
Transfer completed
Configure WiFi to complete the procedure

5. In the WICED Device Configuration webpage select the AP that the IoT device will use to connect to the Internet, enter the password, and click **Connect**.

   Note: If connecting to an AP that does not broadcast its SSID, click on **Add network manually**.



6. Observe the following message on successful connection to the selected AP.



Device Started
Web server and access point stopped
See UART for further information

The IoT device is configured to communicate using the AWS IoT service. After the above message is displayed, the Wi-Fi module exits configuration and starts running the preloaded app, which in the case of the BCM4343W IoT Starter Kit, is the Shadow app.

Reconfiguring an IoT Device

To reconfigure an IoT device (upload a new AWS certificate or private key, or connect to another AP):

1. Press **SW1** (the Reset button).
2. Press and hold **SW2** for five seconds.

Performing the above steps clears Wi-Fi module memory and puts the IoT device back into its SoftAP mode where it can once again be configured.

Connecting everything®

Broadcom Corporation

5300 California Avenue

Irvine, CA 92617

Phone: 949-926-5000

Fax: 949-926-5203

E-mail: info@broadcom.com

Web: www.broadcom.com