ASG200 Datasheet

Version 1.0.0

☞ For more information, visit our website at https://www.wiznet.io/

Contents

Figures

Tables

1. ASG200

1.1. Overview

WIZnet Azure Sphere Guardian 200 (ASG200) is a product which provides Ethernet interfaces to both Public and Private Network. The general Azure Sphere Module supports only one ethernet Interface interacting with Azure Sphere Pluton OS. But ASG200 has an additional Ethernet interface which WIZnet Hardwired TCP/IP is embedded on, so that a legacy device having only ethernet interface can send data to the cloud server in Azure Sphere Security system.

East to apply in brown field system, ASG200 supports a plenty of network application protocol libraries. ASG200 receives data from brown field system in private network and parses it. Then the data is secured and sent to Cloud server by ASG200.

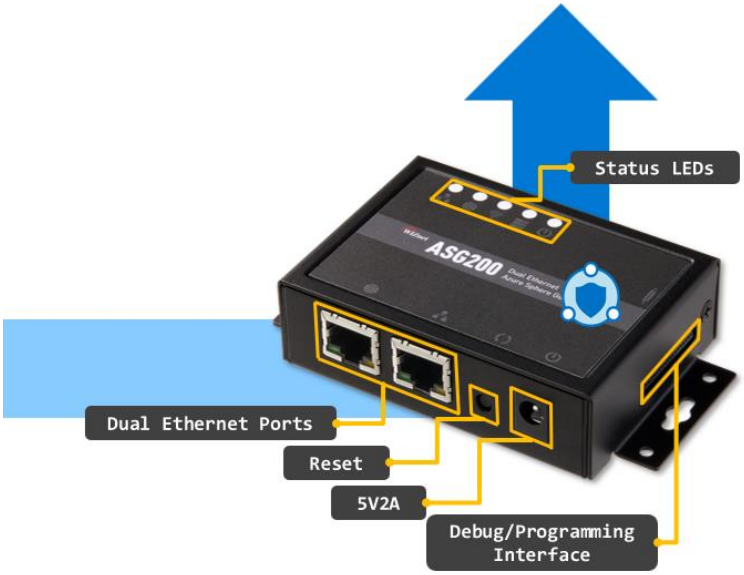

Figure 1. Azure Sphere Guardian 200

1.2. Feaures

- Data transfer between the private network and the public network
- Certificate management
  - By console
  - By Azure Sphere Service
  - By Configuration tool thru Ethernet (***Under development***)
- Support TLS session in private network

- Auto switching between Wi-Fi and Ethernet for public networks optionally
- Support USB interface for debug and programming

1.3. Specification

Item		Description
MCU		MediaTek MT3620 (Single ARM Cortex A7 Core, Dual ARM Cortex M4 Dual Core)
Operating System		Customized Linux Kernel by Microsoft
WAN	HW	Wi-Fi (2.4G/5G Dual band 1T1R) Ethernet (Microchip Ethernet)
	SW	Client application to a Cloud service on Azure IoT
	LEDs	LEDs output: Link, Active
LAN	HW	Ethernet (WIZnet Hardwired TCP/IP)
	SW	Supports following Hardwired TCP/IP protocols: • TCP Server/ TCP Client • DHCP Server/ DHCP Client • SNTP Server • UDP *(To be applied to various brown field network systems, it will be updated a plenty of TCP/IP protocols)*
	LEDs	LEDs output: Link, Active
GPIO	Status LEDs	Five Status LEDs: LAN Ethernet Data communication, WAN Ethernet Link, WAN Wi-Fi Connection, Server Connection, Power
	Input Button	One User Button: Can be set as HW Reset or User-defined Button
	Pin header	18 pin headers are FTDI board connector for Azure Sphere debugging and programing
Power		5V2A (Power Consumption -TBD)
Dimension		90x65x35 mm
Environment		Operating Temperature: -25 ~ 70 Storage Temperature: -40 ~ 85 Operating Humidity: 20 ~ 95 Storage Humidity: 0~95

Table 1. ASG200 Specification

2. System Architecture

System Architecture describes entire system which is ASG200 applied to brown field network and connected to Cloud Server and Management service

2.1. Block Diagram

In ASG200, M4 Core of MT3620 is connected to W5500 which is WIZnet Hardwired TCP/IP chip with SPI interface. Because of Hardwired TCP/IP stack embedded in W5500, software TCP/IP stack is not required on M4 Core for ethernet communication. M4 Core only receives data parsed by W5500 then sends it to A7 Core on Inter-core communication. A7 Core secures this data on Azure Sphere Security System and sends it Azure Cloud via public network.

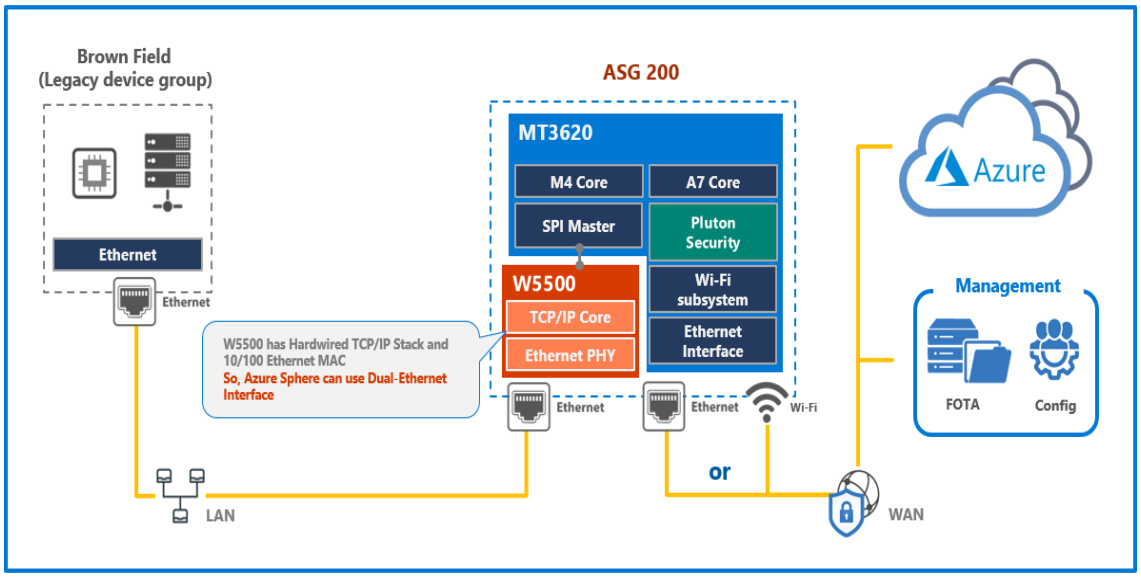

Figure 2. ASG200 Block Diagram

W5500 is connected with only SPI interface to M4 Core. So, the data communication between brown field system and W5500 is out of Azure Sphere Security system. However, W5500 which is Hardwired TCP/IP embedded can filter the ethernet packets used in data communication. It allows reliable ethernet communication even if heavy traffic occurs, like DDoS attack.

3. Installation Overview

ASG200 components consists of two lan cable, micro usb cable, 5V2A power adaptor and debugger board. Debugger board can be connected to ASG200 18pin headers to debug and programing ASG200.

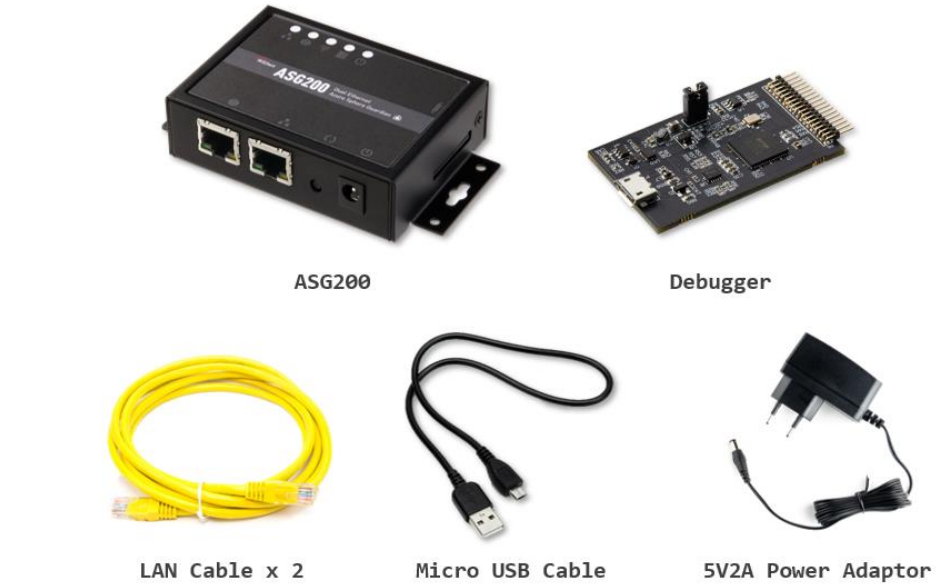

Figure 3. ASG200 Components

Figure 4. ASG200 External description

3.1. ASG200 Connections

An overview of how ASG200 interface to the equipment in local network is as follows:

1. Power provided to ASG200 with 5V2A power adaptor, power status LED turned on.
2. For equipment with as Ethernet interface, connect Ethernet cable from ASG200's LAN port to the equipment.
3. Connect another Ethernet cable from ASG200's WAN port to internet router for public network.
4. Once connected, the LEDs on ASG200 should be as follows:

Status LEDs	Color	Description
Power	Red	Confimation that 5V supply rail voltage is ok

Connection	Green	Ready to communicate with Azure Cloud
Wi-Fi	Green	Activate Wi-Fi
ETH0	Green	Activate WAN port
ETH1	Green	Received data from LAN port

Table 2. LED Status description

4. Azure Sphere Development Environment

Conplete the below steps to develop applications with Azure Sphere on a Windows or Linux system.

4.1. Machine Environment

You can select Azure Sphere development kit for your machine and install software.

- On Windows 10 (1st anniversary update or more)
  - Visual Studio Enterprise, Professional or Community 2019
  - Visual Studio Code
- On Linux
  - Visual Studio Code

4.2. Azure Sphere SDK Installation

Download Azure Sphere SDK and install it.

- Download Azure Sphere SDK

Then complete Azure Sphere SDK Extension Install for development tool.

- For Visual Studio, Azure Sphere SDK Extension Install
- For Visual Studio, Azure Sphere SDK Extension Install

4.3. Azure Sphere Debuger

The MT3620 exposes two dedicated UARTs and SWD interface for debugging. The Azure Sphere PC software tools require the use of a USB-to-UART interface that exposes these interfaces to a PC in away that allows the tools to recognize and interact with them.
For this, ASG200 components has ‘Debugger’ board which can attaches to 18pin headers on ASG200. To use this debugger board, user should init the interface information with FTDI tools.

Please follow these steps decribed in this link:

- FTDI FT_PROG programming tool

5. Development Environment

5.1. Azure Sphere CLI

The azsphere.exe command-line utility supports commands that manage Azure Sphere elements. For the more details, enter the below link:

- azsphere command-line utility

On Azure Sphere Developer Command Prompt Preview, the option, -?, helps to show the command information.

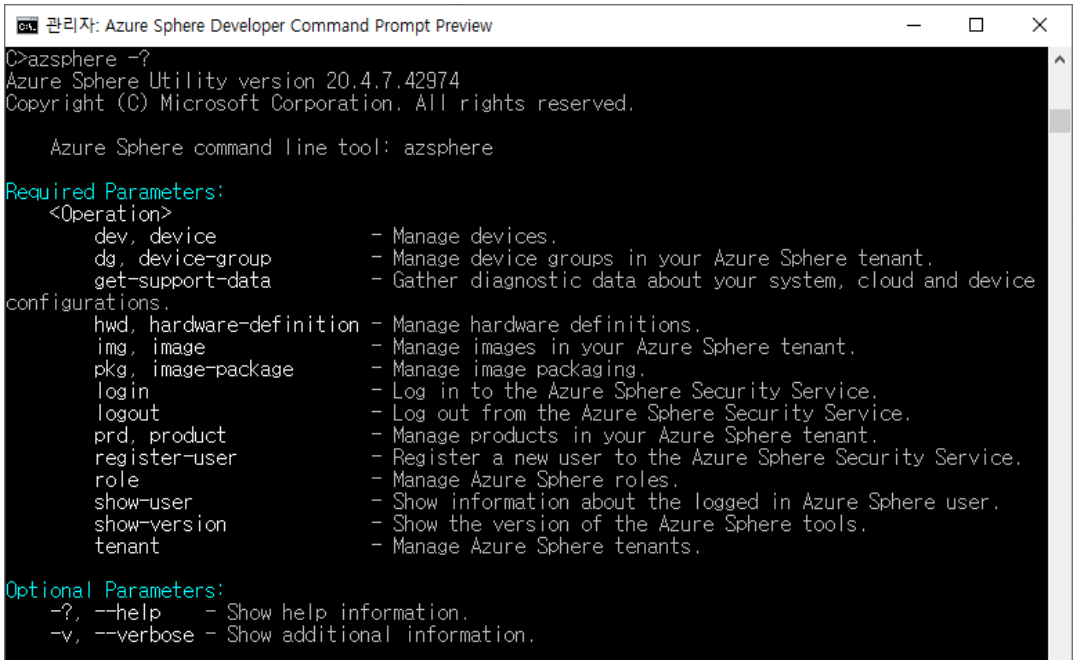

Figure 5. Azure Sphere CLI - -? option

5.2. Register User Account

To manage Azure Sphere elements for development, log in Azure Sphere Developer Command Prompt Preview with Microsoft account. To use Azure Sphere Security Service, Microsoft Account is required.

1. Log in on ‘azsphere login’ command

   (Needed the option, --newuser, with ‘azsphere login’ command to register the account only have to sign in once.)

```
azsphere login –newuser <MS account>
```

5.3. Azure Sphere Tenant

An Azure Sphere tenant provides a secure way for your organization to remotely manage its Azure Sphere devices in isolation from other customer’s devices. And it is accessed based on RBAC (Role Based Access Control). Only people with an account in that directory will be able to manage devices within your Azure Sphere tenant.

5.3.1. Role assigned an account in the Azure Sphere tenant

Follow these steps to select the role assigned Azure Sphere tenant:

1. Search the tenant list.

```
azsphere tenant list
```

2. Select the tenant from the list with tenant id.

```
azsphere tenant select -i <tenant id>
```

3. Check the selected tenant.

```
azsphere tenant show-selected
```

5.3.2. Create new tenant

There is no existed Azure Sphere tenant or assigned role in it. User can create new Azure Sphere tenant.

1. Create new tenant

```
azsphere tenant create -n <tenant name>
```

5.4. ASG200 Claim

Check the selected tenant for development environment. Once ASG200 claimed to the Azure Sphere tenant, claiming to other tenant is prohibited followed Azure Sphere Security policy.

1. Claim ASG200 to the selected tenant

```
azsphere device claim
```

5.5. ASG200 Configuration

5.5.1. Recovery interface

Once ASG200 is connected to the internet, Azure Sphere OS updates are initiated automatically via OTA (Over The Air) Wi-Fi interface. Also, user can manually update Azure Sphere OS with recovery. Recovery is the process of replacing the latest system software on the device using a special recovery bootloader instead of cloud update.

Follow these steps to update the latest Azure Sphere OS:

1. Set Wi-Fi interface

```
azsphere device wifi add –ssid <SSID> --psk <Password>
```

2. Check Wi-Fi Status

```
azsphere device wifi show-status
```

3. Recovery for Azure Sphere OS update

```
azsphere device recover
```

4. Check Azure Sphere OS version

```
azsphere device show-os-version
```

5.5.2. Development Mode

Connect Debugger board which is attached to ASG200 debug interface to PC and set development mode for debugging on In Azure Sphere Developer Command Prompt Preview. On development mode, OTA is inactivated.

1. Development mode for debugging

```
azsphere device enable-development
```

2. Add option for RT App debugging

```
azsphere device enable-development --enablertcoredebugging
```

6. Run Application

For ASG200 application, chapter 5, Development Environment, is preceded.
ASG200 application has two types of applications, High-level application and Real-time capable application.

6.1. Real-time capable Application: W5500 SPI BareMetal

Real-time (RT) capable application run on bare metal or with a real-time operating system on the real-time cores.
In ASG200, RTApp (Real-time capable Application) is 'RTApp_W5500_SPI_BareMetal_WIZASG200' and it controls WIZnet W5500 ethernet chip and provides variety protocol communications with legacy devices on brown field. Also, it performs inter-core communication between RTApp and HLApp (High-level Application).

RTApp_W5500_SPI_BareMetal_WIZASG200 is performed as the followed:

- WIZnet W5500 SPI control
  - Local network communication with brown field
  - Ethernet interface
  - TCP Server for data communication with brown field
  - DHCP Server for local network address configuration of brown field
  - SNTP Server for time information management
- Inter-core communication
  - Send the parsing data from brown field to HLApp

6.2. High-level Application: AzureIoT

High-level (HL) application run containerized on the Azure Sphere OS. In ASG200, HLApp (High-level application) is 'HLApp_AzureIoT_WIZASG200' and it provides whole functions for Azure IoT Cloud service. Also, it automatically switches global interface, Ethernet and Wi-Fi, for network condition.

HLApp_AzureIoT_WIZASG200 is performed as the followed:

- Global network communication with Azure IoT Cloud service
  - Ethernet and Wi-Fi interface
  - Connection and Authentication on IoT Hub or IoT Central
- Inter-core communication
  - Receive the data from RTApp for sending to Azure IoT Cloud

6.3. Configure an IoT Hub

To operate ASG200 application, RTApp and HLApp, Azure IoT Hub or IoT Central configuration is required.

- Set up an Azure IoT Hub for Azure Sphere
- Set up an Azure IoT Central to work with Azure Sphere

Then, user will need to supply the following information in the app_manifest.json file for Azure IoT:

- The Tenant ID for ASG200
- The Scope ID for Azure device provisioning service (DPS) instance
- The Azure IoT Hub URL for user IoT Hub or Central along with the global access link to DPS (global.azure-devices.provisiong.net)

In app_manifest.json, add Azure DPS Scope ID, Azure IoT Hub endpoint URL and Azure Sphere Tenant ID from Azure IoT Hub or Central into the following lines:

```
{
  "SchemaVersion": 1,
  "Name": "HLApp_AzureIoT_WIZASG200",
  "ComponentId": "819255ff-8640-41fd-aea7-f85d34c491d5",
  "EntryPoint": "/bin/app",
  "CmdArgs": ["<Azure DPS Scope ID>"],
  "Capabilities": {
    "AllowedConnections": [
      "global.azure-devices-provisioning.net",
      "<Azure IoT Hub endpoint URL>"
    ],
    "DeviceAuthentication": "<Azure Sphere Tenant ID>",
    "AllowedApplicationConnections": ["005180bc-402f-4cb3-a662-72937dbcde47"],
    "Gpio": [
      "$WIZNET_ASG200_CONNECTION_STATUS_LED",
      "$WIZNET_ASG200_WLAN_STATUS_LED",
      "$WIZNET_ASG200_ETH0_STATUS_LED",
      "$WIZNET_ASG200_ETH1_STATUS_LED"
    ],
    "NetworkConfig": true,
    "WifiConfig": true
  },
  "ApplicationType": "Default"
}
```

6.4. Set up Public Ethernet interface

To enable ethernet interface for public network and communication with Azure IoT, install ethernet imagepackage by deploying a board configuration image to ASG200. The board configuration image contains information that the Azure Sphere Security Service requires to add support for Ethernet to the Azure Sphere OS.

Follow these steps to enable public ethernet interface:

1. Create a board configuration image package

```
azsphere image-package pack-board-config --preset lan-enc28j60-isu0-int5 --output
enc28j60-isu0-int5.imagepackage
```

2. Prepare ASG200 for development mode

```
azsphere device enable-development
```

3. Sideload a board configuration image package

```
azsphere device sideload deploy --imagepackage enc29j60-isu0-int5.imagepackage
```

4. Check the sideloaded imagepackage

```
azsphere device image list-installed
```

```
C>azsphere device image list-installed
Installed images:
 --> lan-enc28j60-is
   --> Image type:    Board configuration
   --> Component ID: 07499982-a803-4b06-9cfc-d39c444e3629
   --> Image ID:     1a60a239-fb1c-4e0b-ab88-90aa7294b3b4
 --> gdbserver
   --> Image type:    Application
   --> Component ID: 8548b129-b16f-4f84-8dbe-d2c847862e78
   --> Image ID:     77717169-c643-4c59-bca3-b79035469ea1
```

Figure 6. Azure Sphere CLI - Image installed list

6.5. Build and Run the Application

The application can be run and developed with Visual Studio and Visual Studio Code.

6.5.1. Run with Visual Studio

Follow these steps to build and run the application with Visual Studio:

1. Start Visual Studio, From the **File** menu, select **Open > Folder…** and navigate to the folder, '**HLApp_AzureIoT_ASG200**'.

2. Open app_manifest.json file and check the information correct.

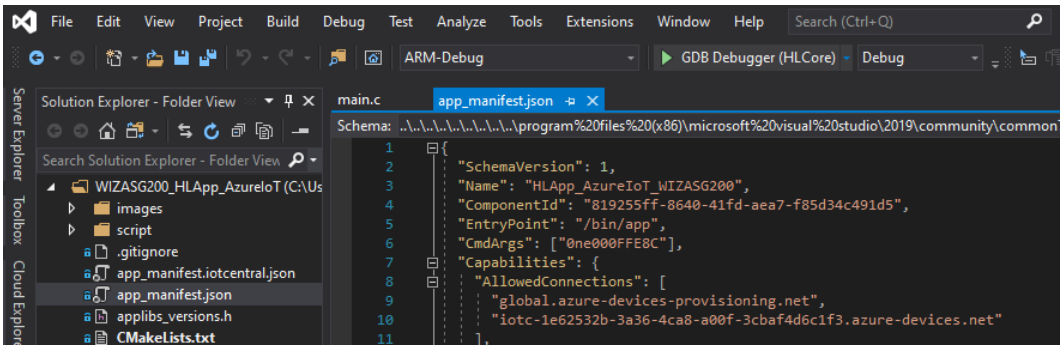

Figure 7. Visual Studio - Open app_manifest.json

3. From the Select Startup Item menu, on the tool bar, select GDB Debugger (HLCore).

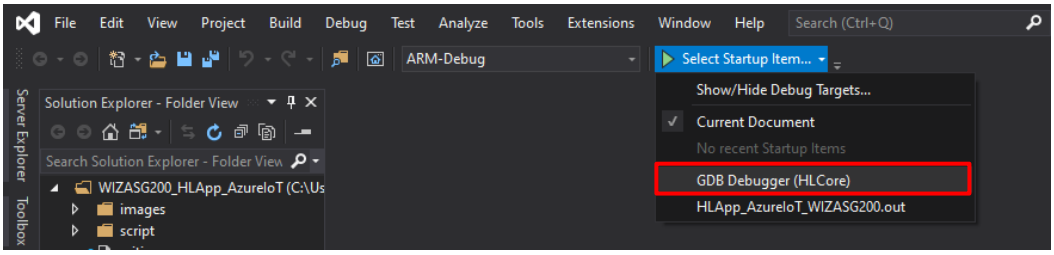

Figure 8. Visual Studio - Select GDB Debugger

4. Click Build>Build All to build the project

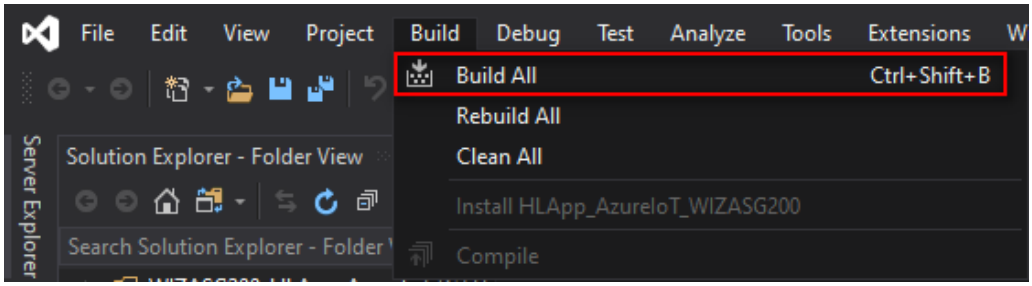

Figure 9. Visual Studio - Build the project

5. Press F5 to start the application with debugging.

6.5.2. Run with Visual Studio Code

Follow these steps to build and run the application with Visual Studio Code:

1. Open 'HLApp_AzureIoT_ASG200' folder.

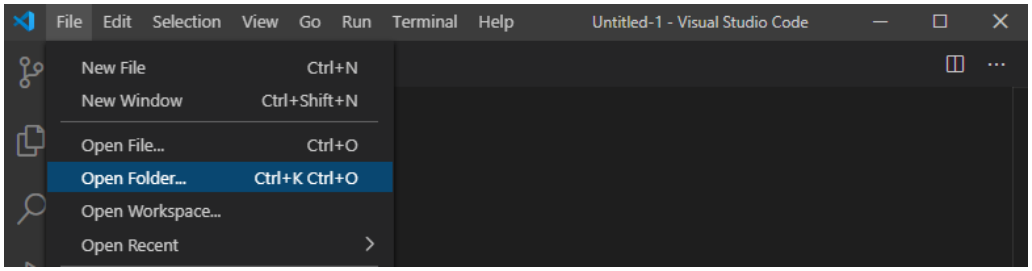

Figure 10. Visual Studio Code - Open Project Folder

2. Press F7 to build the project

3. Press F5 to start the application with debugging

7. Hardware Specification

7.1. Dimensions

Figure 11. ASG200 Dimension

7.2. DC Power Cable Specification

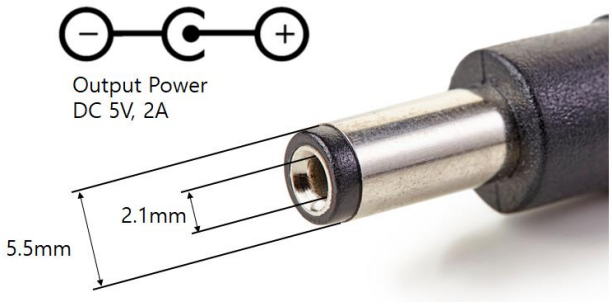

Figure 12. ASG200 DC Power cable specification

8. Resource

8.1. Software Checklist

ASG200 Application	Github Repository
ASG200_App	Github Repository Link

Table 3. ASG200 Application Github Repository

Libraries and Samples	Github Repository

Table 4. Libraries and Samples Github Repository

Document History Information

Version	Date	Description
Ver. 1.0.0	5JUN2020	Initial Release

Copyright Notice

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at https://www.wiznet.io/