Security Network Shield
User Guide

Revision	Date	Changes
0.5	23-Aug-2016	Initial Draft

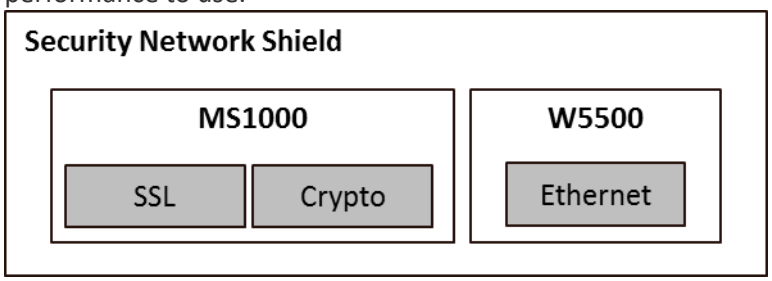
1 Overview

1.1 Security Netwok Shield

Security Network Shield is upgrade version of existing W5500 Ethernet Shield designed using the MS1000 and WIZnet W5500 chip. (Please find the link for further information about W5500 Ethernet Shield - http://wizwiki.net/wiki/doku.php?id=osh:w5500_ethernet_shield:start)

Security Network Shield provides all of existing network function from W5500 Ethernet Shield and specially supports SSL(Secure Sockets Layer) protocol. It ensures that all data passed between server and client.

In Security Network Shield MS1000 take SSL function with HW accelerate security engine for best performance to use.



This Security Network Shield is compatible with Arduino Platform.

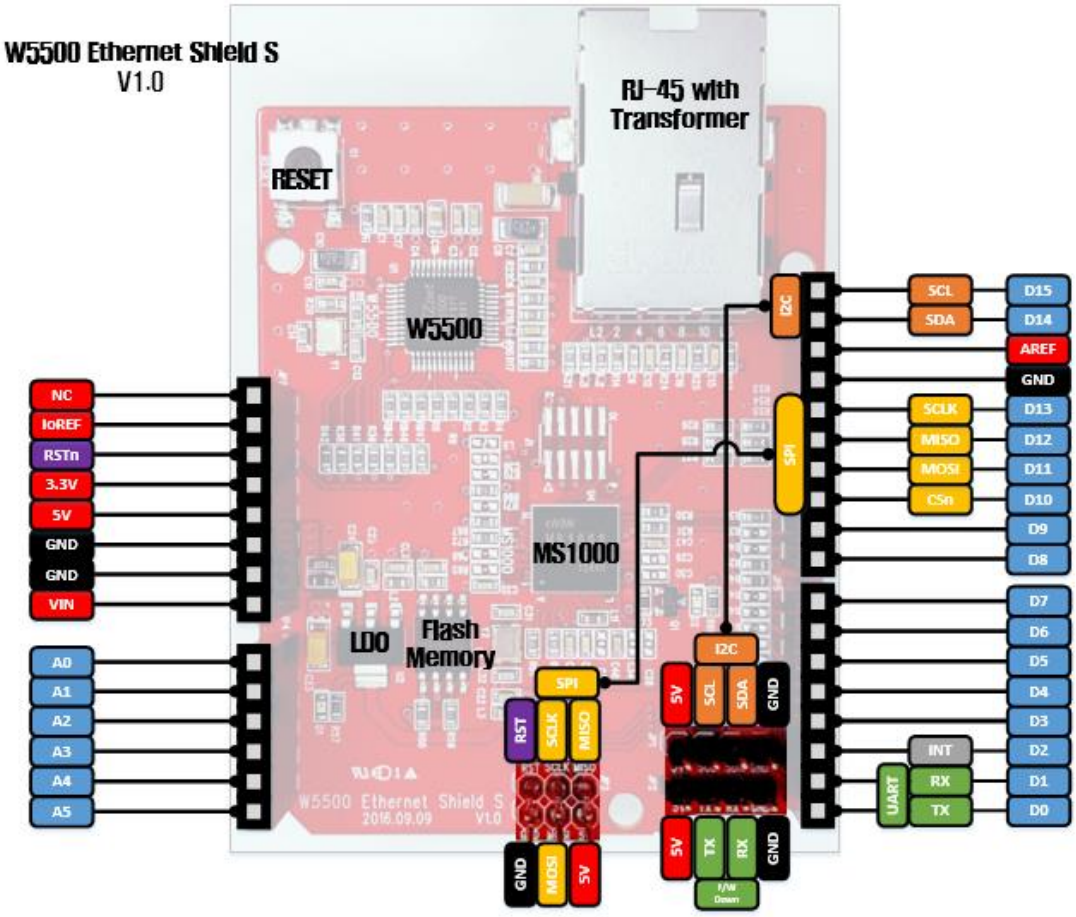
1.2 Available Board list

- Arduino Board (e.g the Uno)

2 Feature

2.1 Hardware Feature

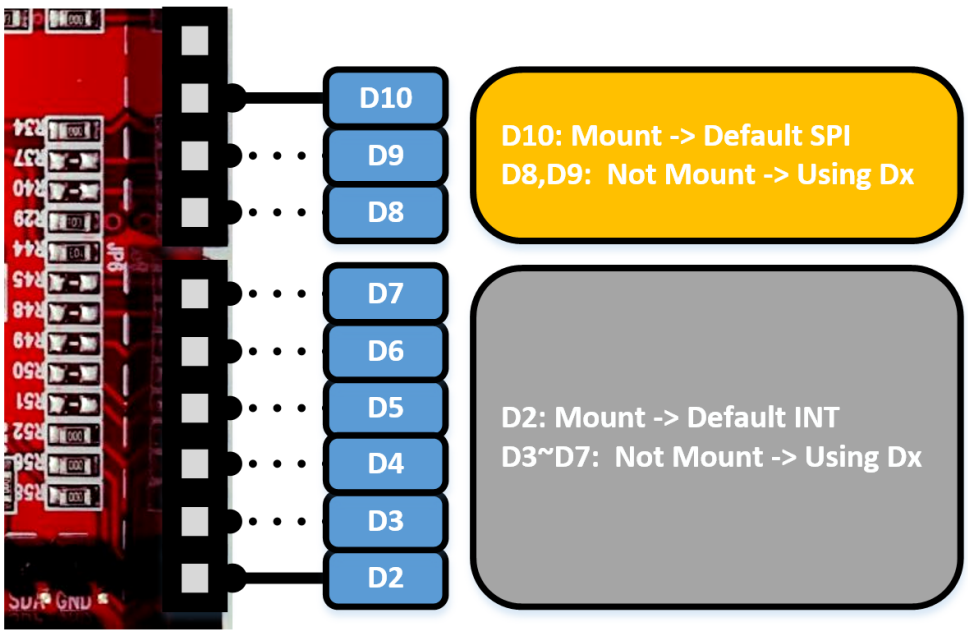
- Support 3.3V
- ARM Cortex-M3 MCU with HW Crypto engine (MS1000)
- High Speed Ethernet controller (W5500)
- 10/100 Ethernet PHY embedded.
- Hardwired TCP/IP Protocols: TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE.
- Support SPI, I2C, UART interface



2.2 Hareware Configuration

- MS1000: ARM® Cortex-M3™ based microcontroller with HW crypto engine.
- W5500: Hardwired TCP/IP Ethernet Controller
- RJ-45 with Transformer: Ethernet Port
- I2C: I2C interface
- UART: UART interface
- SPI: SPI Interface

Pins usage on Arduino



2.3 Software Feature

- Support SSL/TLS 1.2
- SSL Specification

Category	Description	Comment
Cipher Suit - Public Key Algorithm	RSA ECC	TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CCM_8 TLS_RSA_WITH_AES_256_CCM_8 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
Cipher Suit - Block/Stream Ciphers	AES CCM GCM CBC CTR ECB	
Cipher Suit - Hash Functions	SHA1 SHA256	
Side of Connection	Client only	
Client Authentication	APIs support	CA certificate load, Certificate/Private Key load

3 SPI Operation

3.1 Overall SPI Interface

Security Network Shield supports up to 4MHz speed in slave mode.

Function	Interface		GPIO
Security Network Shield	SPI	SCK	PC0
		SSN	PC1
		MISO	PC2
		MOSI	PC3
		OUT_INT	PD6

3.2 SPI Timing

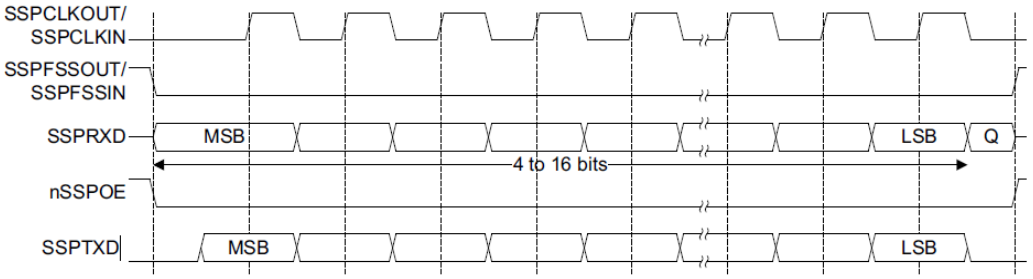
Data Mode
There are four combinations of SCK phase and polarity with respect to serial data.
Which are determined by control bits CPHA and CPOL.
Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize

- By default, Security Network Shield is set to CPOL = 1, CPHA = 1

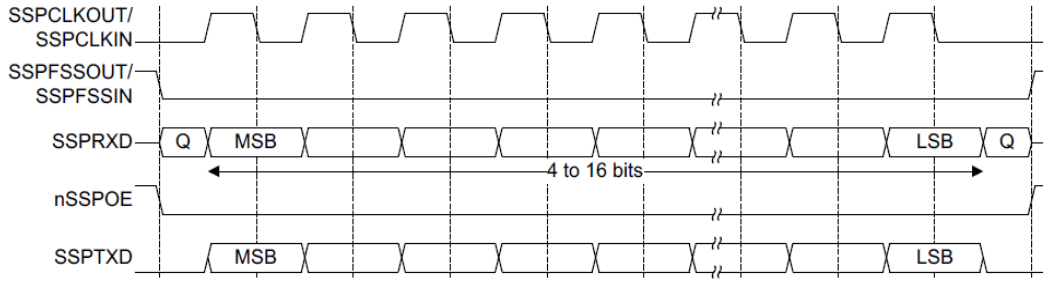
CPOL and CPHA Finctionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	↑		0
CPOL = 0, CPHA = 1		↓	1
CPOL = 1, CPHA = 0	↓		2
CPOL = 1, CPHA = 1		↑	3

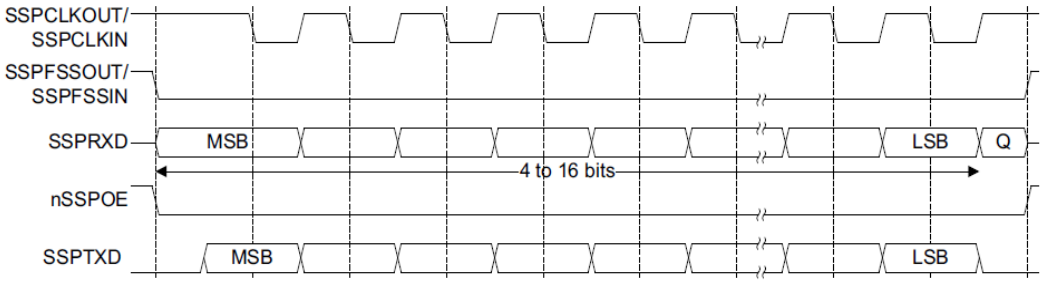
SPI Transfer format with CPOL =0, CPHA = 0



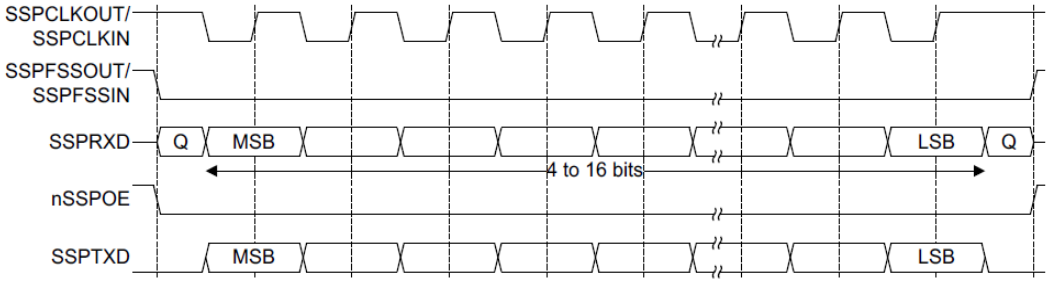
SPI Transfer format with CPOL =0, CPHA = 1



SPI Transfer format with CPOL =1, CPHA = 0

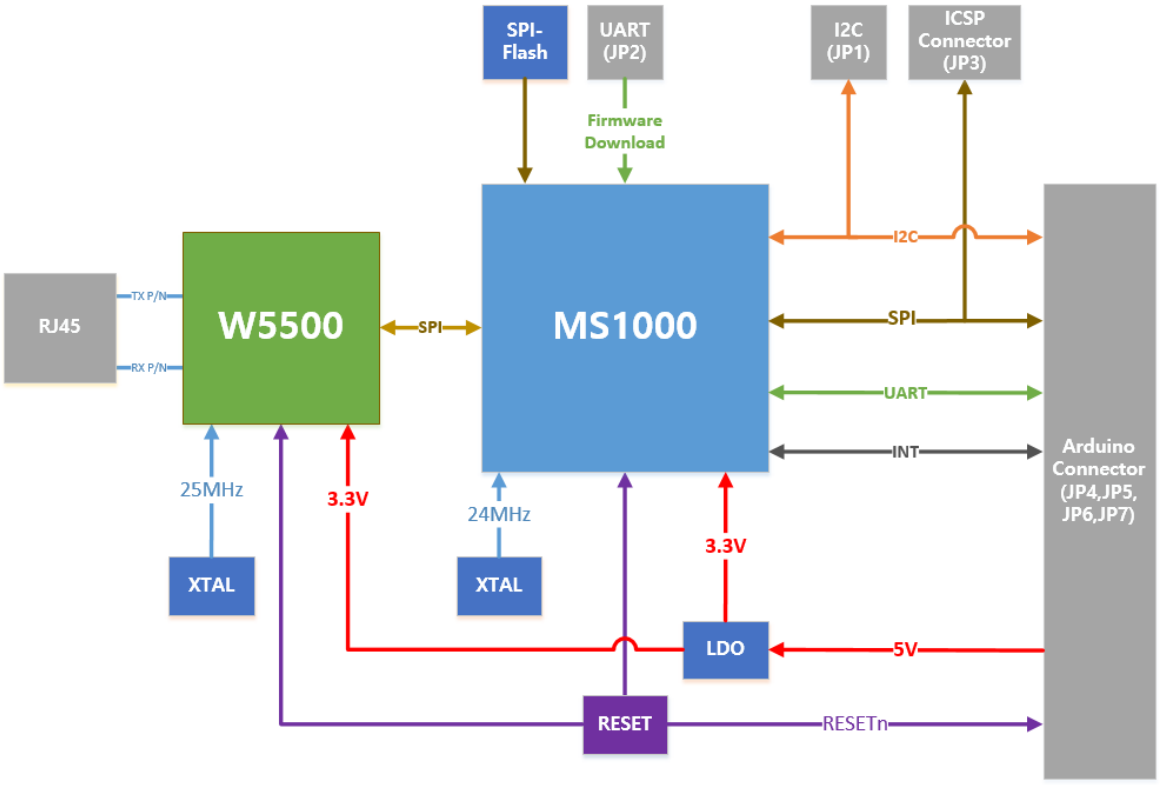


SPI Transfer format with CPOL =1, CPHA = 1



4 Technical Reference

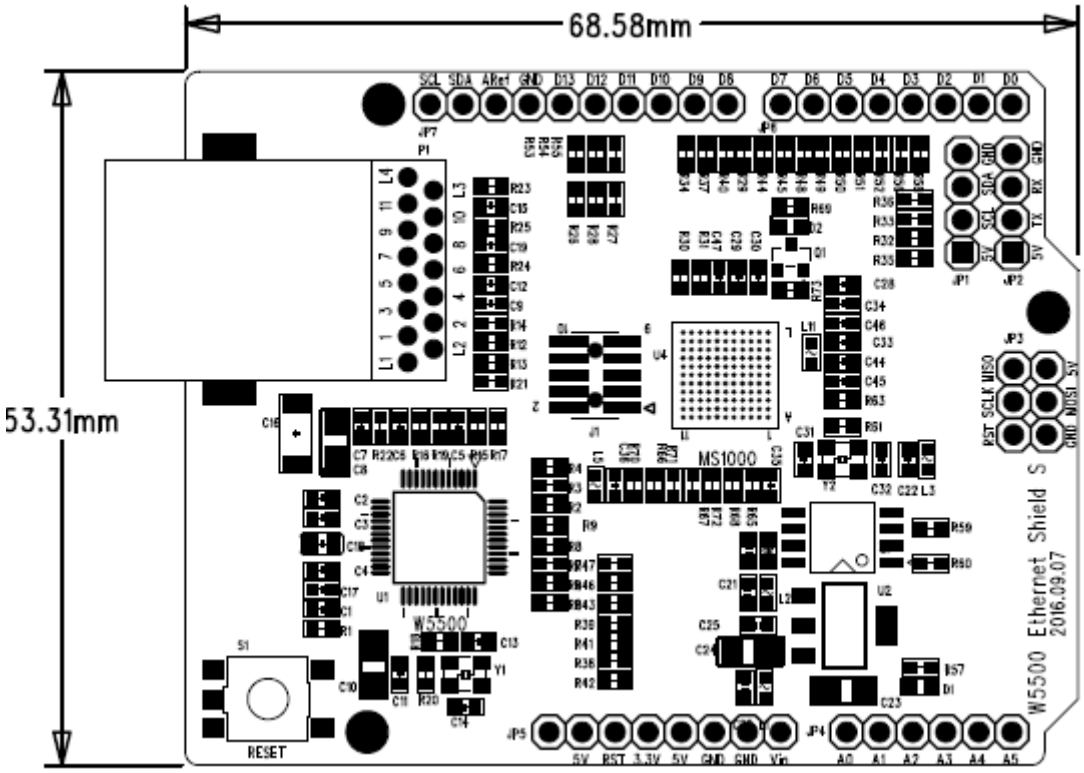
4.1 Block Diagram



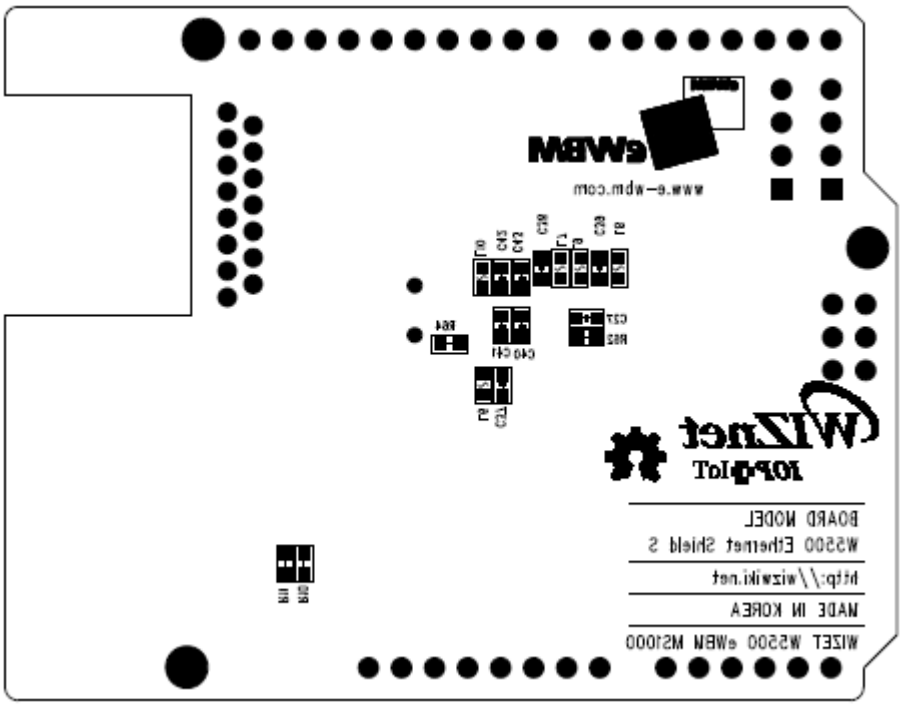
4.2 Schematic

Document Link: w5500_ethernet_shield_s_V1.0.pdf

4.3 Dimension



TOP SIDE



Bottom Side

Security Network Shield Demension

5 Getting Started

5.1 Using WIZ ethernet Library for Arduino Uno

Class	Description
Ethernet Class	Included Class to the Wiz Ethernet library to support internet in Arduino Uno Refer to the Arduino Ethernet library and API Guide at the follow site. WIZ Ethernet Library: https://github.com/Wiznet/WIZ_Ethernet_Library Arduino Ethernet API: https://www.arduino.cc/en/Reference/Ethernet
SSL Class	Added Class to the Wiz Ethernet library to support SSL in the Arduino
MIF Class	Added Class to the Wiz Ethernet library to communicate with Security Network Shield in the Arduino

5.1.1 Description of added SSL class and MIF class

The SSL client works with SSL initialize and connect to server and send/receive data.

- Only SSL Client operation. (SSL Server does not work)
- USE_MS1000_MIF feature is a function for SSL client only on w5500.
- When USE_MS1000_MIF feature is Disable, SSL client does not work.

5.1.2 API Reference of SSL class and MIF class

- **SSL CLASS**

Open()	
Description	Open of SSL Socket
Syntax	SSLClient.Open()
Parameters	None
Returns	If successful the call will return SSL_SUCCESS

Close()	
Description	Close of SSL Socket
Syntax	SSLClient.Close()

Parameters	None
Returns	If successful the call will return SSL_SUCCESS

Connect()	
Description	This function is called on the client side and initiates an SSL/TLS handshake with a server
Syntax	SSLClient.Connect(ip, port) SSLClient.Connect(hostname, port)
Parameters	Ip: connecting to domain ip address hostname: connecting to hostname (ex: www.google.com) port: SSL port
Returns	If successful the call will return SSL_SUCCESS

WriteData()	
Description	This function writes sz bytes from the buffer, data, to the SSL connection, ssl
Syntax	SSLClient.WriteData()
Parameters	buf: data buffer which will be sent to peer size: size, in bytes, of data to send to the peer IsPMEM: the generating data to the Flash (Program) instead of SRAM memory
Returns	If successful the call will return SSL_SUCCESS

ReadData()	
Description	This function reads sz bytes from the SSL session (ssl) internal read buffer into the buffer data. The bytes read are removed from the internal receive buffer.
Syntax	SSLClient.ReadData()
Parameters	buf: data buffer which will be read to peer size: number of bytes to read into data. readsz: getting read size
Returns	If successful the call will return SSL_SUCCESS

SetPeerVerify()	
Description	This function sets the verification method for remote peers and also allows a verify callback to be registered with the SSL session. The verify callback will be called only when a verification failure has occurred. If no verify callback is desired, the NULL pointer can be used for verify_callback
Syntax	SSLClient.SetPeerVerify()
Parameters	verify: enable verify
Returns	If successful the call will return SSL_SUCCESS

SetRootCA()	
Description	This function sets a CA certificate buffer into the SSL. It behaves like the non buffered version, only differing in its ability to be called with a buffer as input instead of a file.
Syntax	SSLClient.SetRootCA()
Parameters	buf: the CA certificate buffer len: size of the input CA certificate buffer

	IsPMEM: the generating data to the Flash (Program) instead of SRAM memory
Returns	If successful the call will return SSL_SUCCESS

GetVersion()	
Description	This function gets the SSL/TLS protocol version for the specified SSL session using the version as specified by version.
Syntax	SSLClient.GetVersion()
Parameters	buf: the version information buffer len: length of buf
Returns	If successful the call will return SSL_SUCCESS

GetCipherName()	
Description	Retrieves the peer's certificate cipher name
Syntax	SSLClient.GetCipherName()
Parameters	buf: the cipher name buffer len: length of buf
Returns	If successful the call will return SSL_SUCCESS

GetX509IssuerName()	
Description	Retrieves the peer's certificate issuer name
Syntax	SSLClient.GetX509IssuerName
Parameters	buf: the issuer name buffer len: length of buf
Returns	If successful the call will return SSL_SUCCESS

GetX509SubjectName()	
Description	Retrieves the peer's certificate subject name
Syntax	SSLClient.GetX509SubjectName
Parameters	buf: the subject name buffer len: length of buf
Returns	If successful the call will return. SSL_SUCCESS

GetX509NextAltName()	
Description	Retrieves the peer's certificate next altname
Syntax	SSLClient.GetX509NextAltName
Parameters	buf: the next altname buffer len: length of buf
Returns	If successful the call will return SSL_SUCCESS

GetX509SerialNum()	
Description	Retrieves the peer's certificate serial number
Syntax	SSLClient.GetX509SerialNum()
Parameters	buf: the serial number buffer len: length of buf OutNumSz: getting a length of serial number
Returns	If successful the call will return SSL_SUCCESS

- **MIF CLASS**

Write()	
Description	This function writes 1 byte to Slave
Syntax	gMIFInfo.Write()
Parameters	w: to write data
Returns	None

Read()	
Description	This function reads 1 byte from Slave
Syntax	gMIFInfo.Read()
Parameters	None
Returns	Read data

WaitCmd()	
Description	This function wait 1 byte command for send to Slave
Syntax	gMIFInfo.WaitCmd()
Parameters	waitcmd: 1byte command
Returns	If successful the call will return 0

StartCmd()	
Description	This function send 1 byte command to Slave
Syntax	gMIFInfo.StartCmd()
Parameters	cmd: 1byte command ctrlb: distinguish from read commad and write command datalen: read/write data length
Returns	If successful the call will return 0

EndCmd()	
Description	This function indicates the end of command
Syntax	gMIFInfo.EndCmd()
Parameters	None
Returns	If successful the call will return 0

Init()	
Description	MIF Class Initialize
Syntax	gMIFInfo.Init()
Parameters	None
Returns	None

WriteData()	
Description	This function writes sz bytes from the buffer, data, to Slave
Syntax	gMIFInfo.WriteData()
Parameters	addr: SSL command set ctrlb: distinguish from read commad and write command pWBuf: data buffer which will be sent to slave len: data buffer size IsPMEM: the generating data to the Flash (Program) instead of SRAM memory
Returns	If successful the call will return 0

ReadData()	
Description	This function reads sz bytes from the Slave.
Syntax	gMIFInfo. ReadData()
Parameters	addr: SSL command set ctrlb: distinguish from read commad and write command pRBuf: data buffer which will be read to slave len: data buffer size
Returns	If successful the call will return 0

IsReady()	
Description	Check the MIF initialize
Syntax	gMIFInfo.IsReady()
Parameters	None
Returns	If MIF initialize successful, the call will return true

5.2 Install the Arduino Software

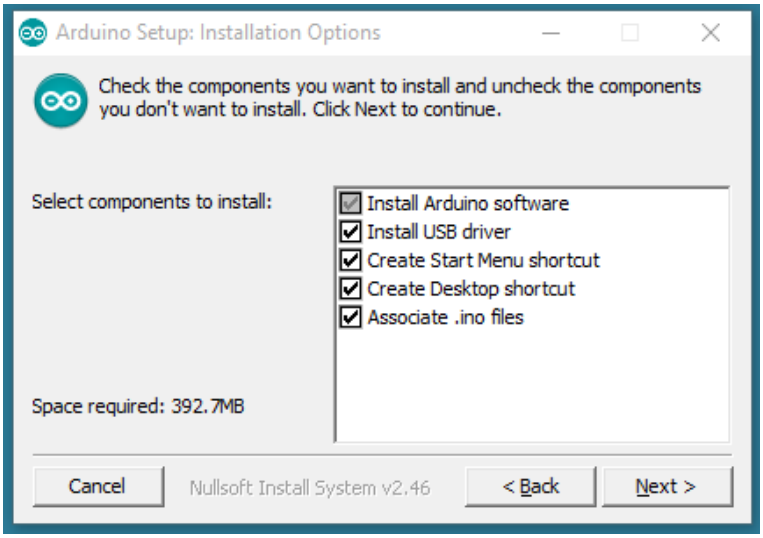
5.2.1 Download IDE

Downloading IDE at Arduino site
https://www.arduino.cc/en/Main/Software

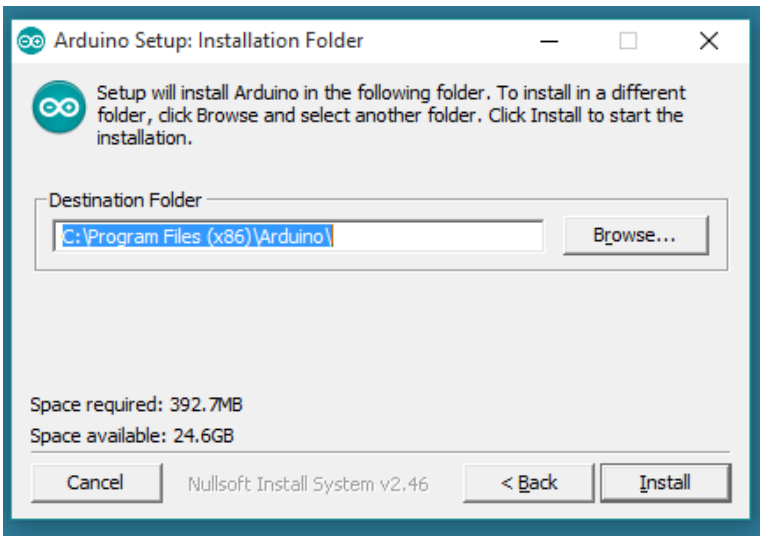
5.2.2 Install IDE

When the download finishes, proceed with the installation and please allow the driver installation process.

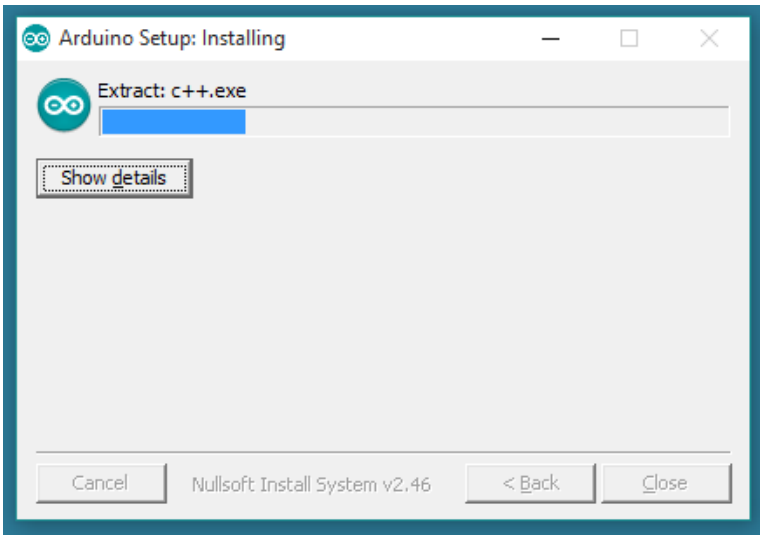
Step 1: Choose the components to install



Step 2: Choose the installation directory (we suggest to keep the default one)

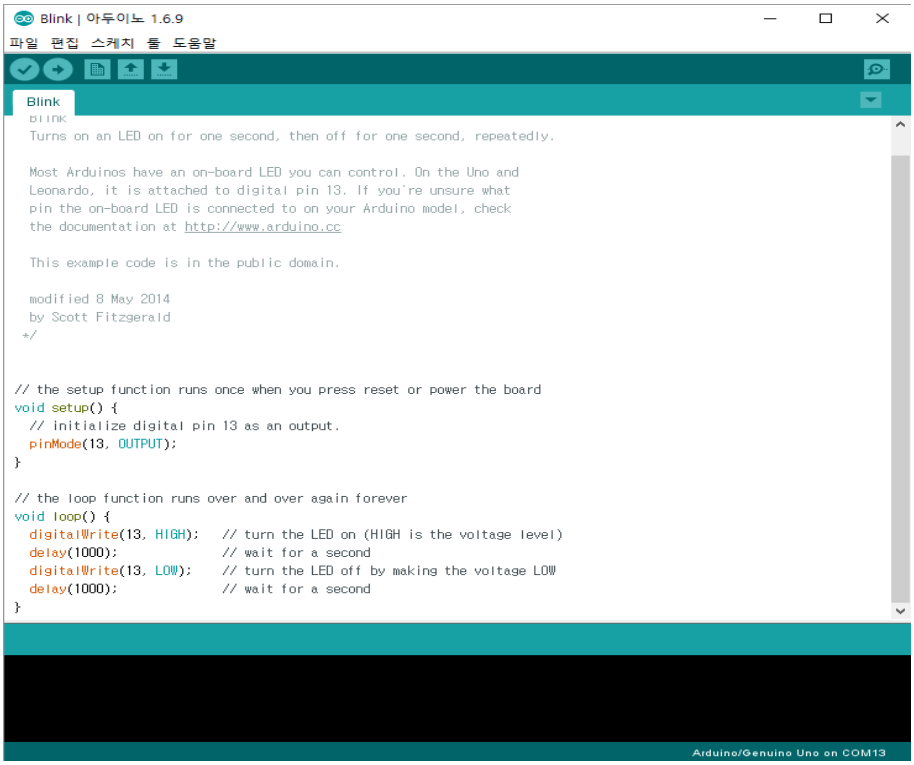


Step 3: The Process will extract and install all the required files to execute properly the Arduino Sortware (IDE)



5.2.3 Launch the IDE

double-Click the Arduino icon (arduino.exe) created by the installation process.
Open the blink example.



5.3 Wiznet Library Update

5.3.1 Getting for Security Network Shield Library

Step 1: Getting the released security network shield source.

- Base source code

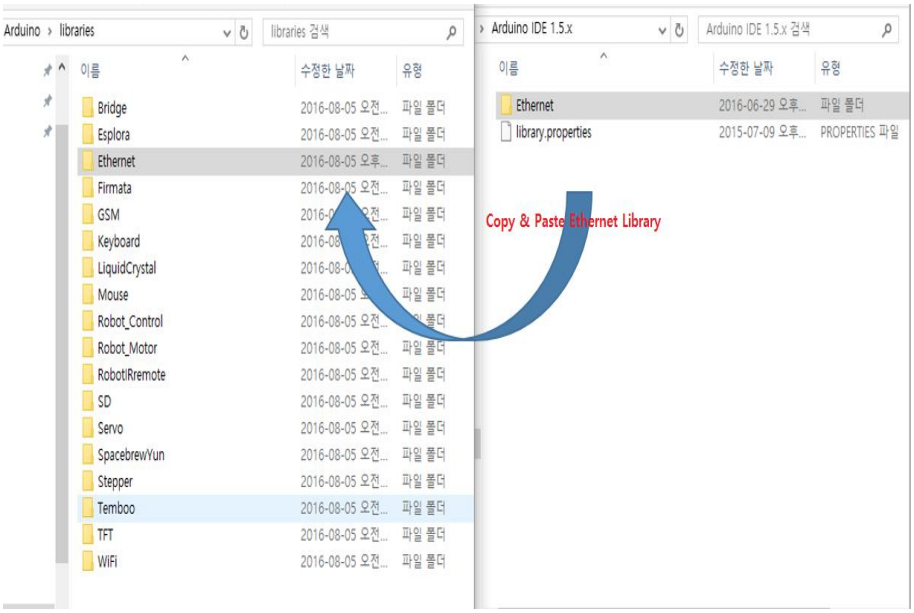
https://github.com/Wiznet/WIZ_Ethernet_Library

5.3.2 Library Update

Step 1: Unzip the ZIP file

Step 2: Go into C:\Program Files\Arduino\libraries

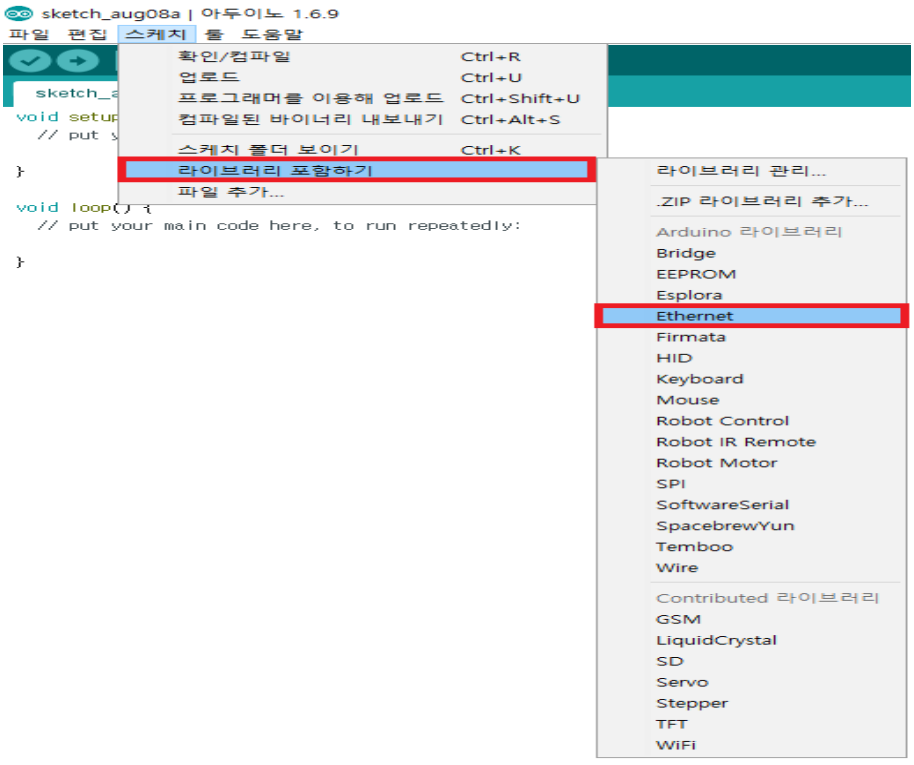
Step 3: Copy & Paste 'Arduino IDE 1.5.x Folder → Ethernet → src' folder to C:\Program Files\Arduino\library\Ethernet folder.



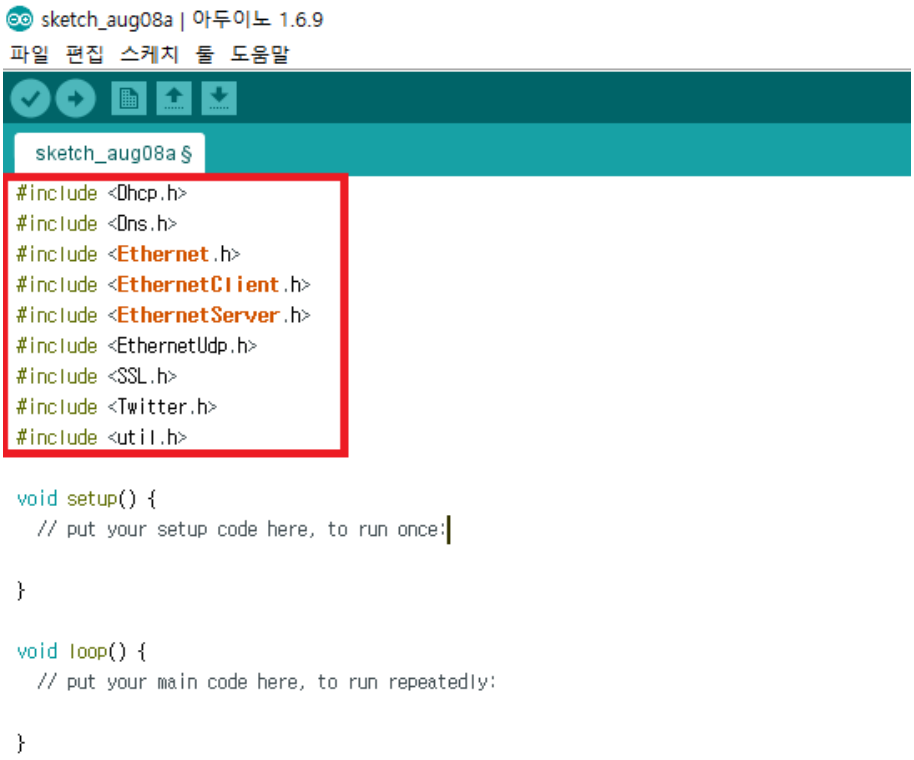
5.3.3 Library Import

Step 1: Use Arduino IDE by importing Library

Step 2: To use library of Arduino Ethernet shield, add header files by selecting Import Library > Ethernet of Sketch menu



▶ Add header file

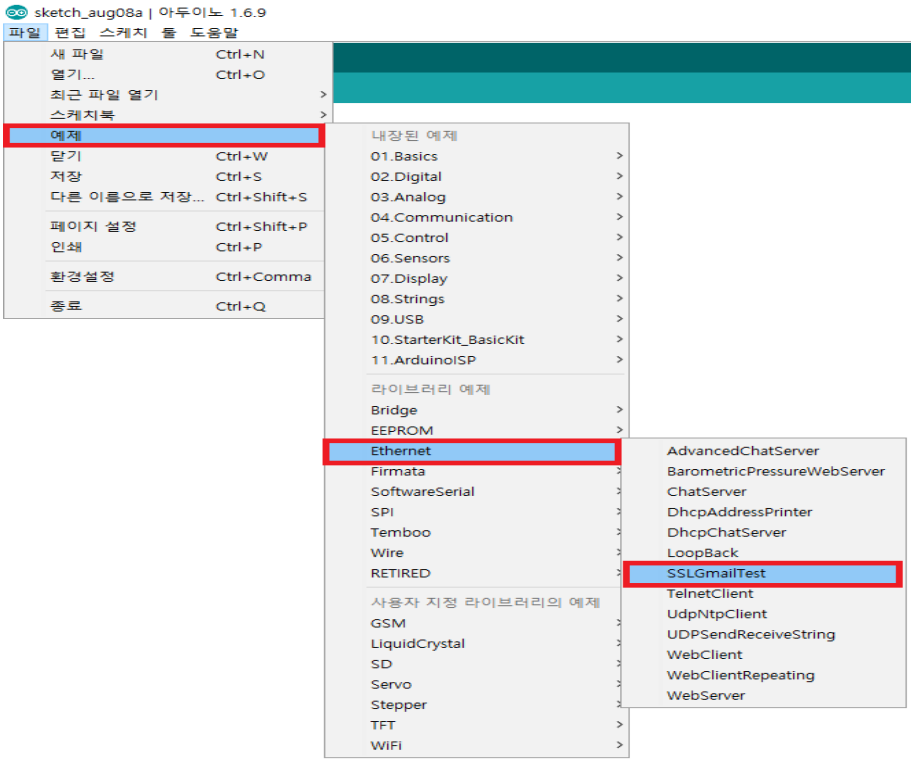


5.4 Arduino Example

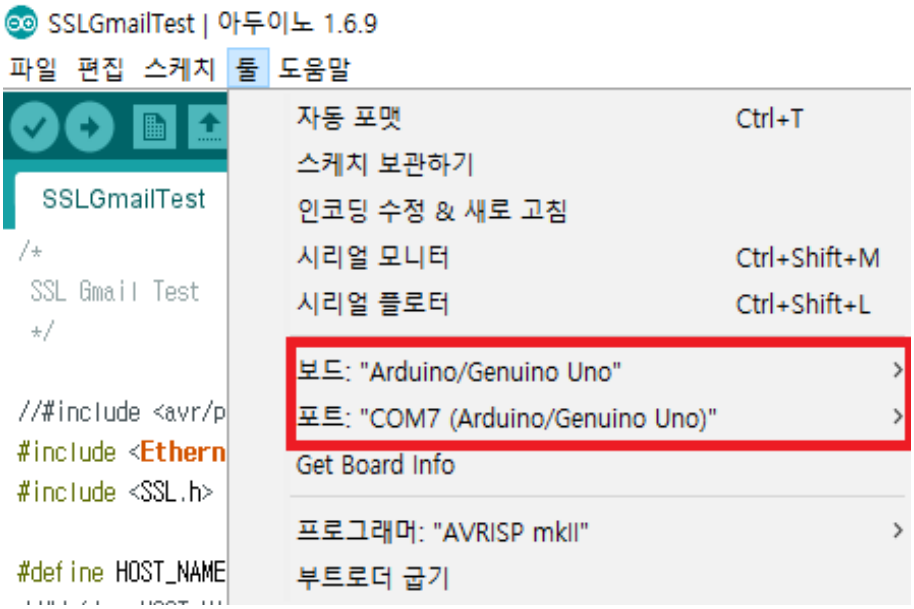
5.4.1 Start Arduino (SSL Example)

Step 1: Run Arduino

Step 2: Select Example -> Ethernet -> SSLGmailTest

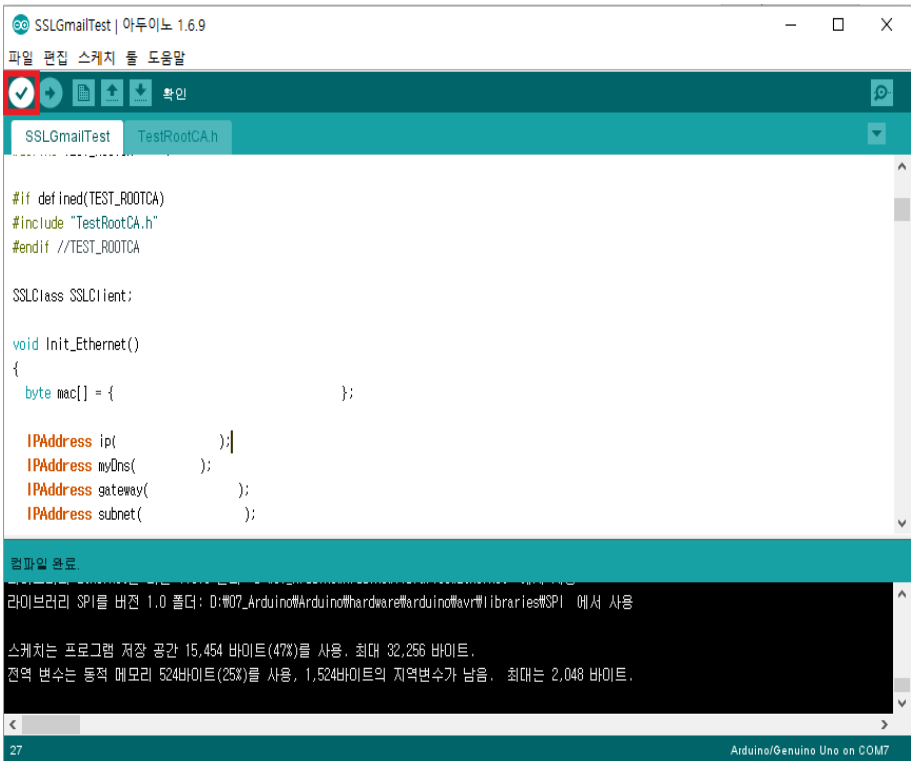


Step 3: Specific setting is needed based on board type such as Uno, Mega, Due... etc
Tool -> board -> Arduino Uno
Tool -> port -> Comx

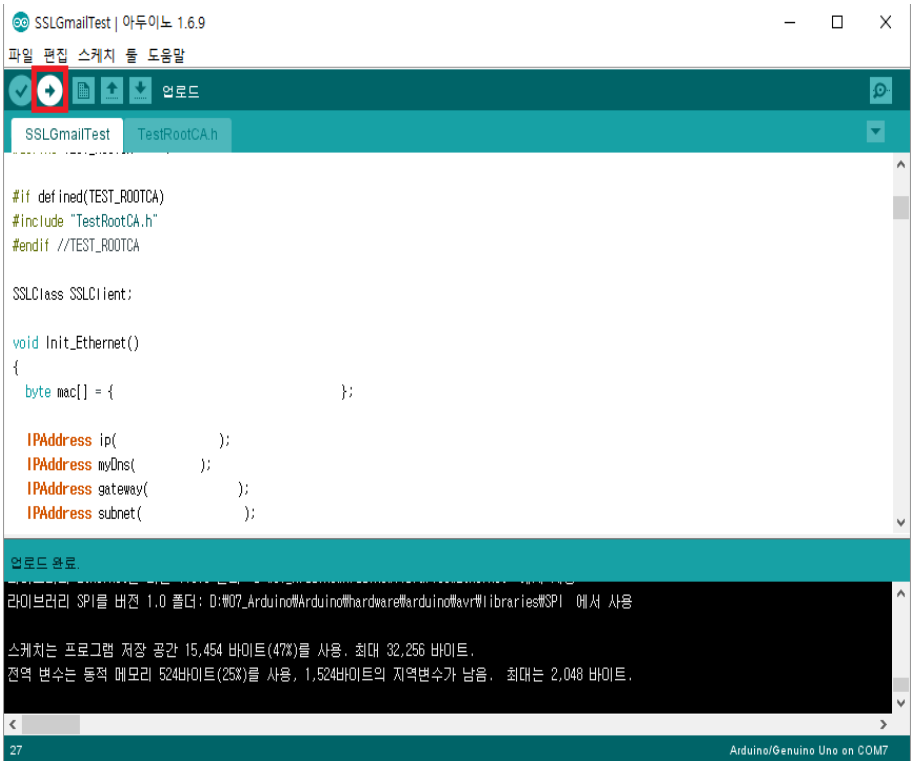


5.4.2 Operation Arduino

Step 1: Click “Verify” to check code error



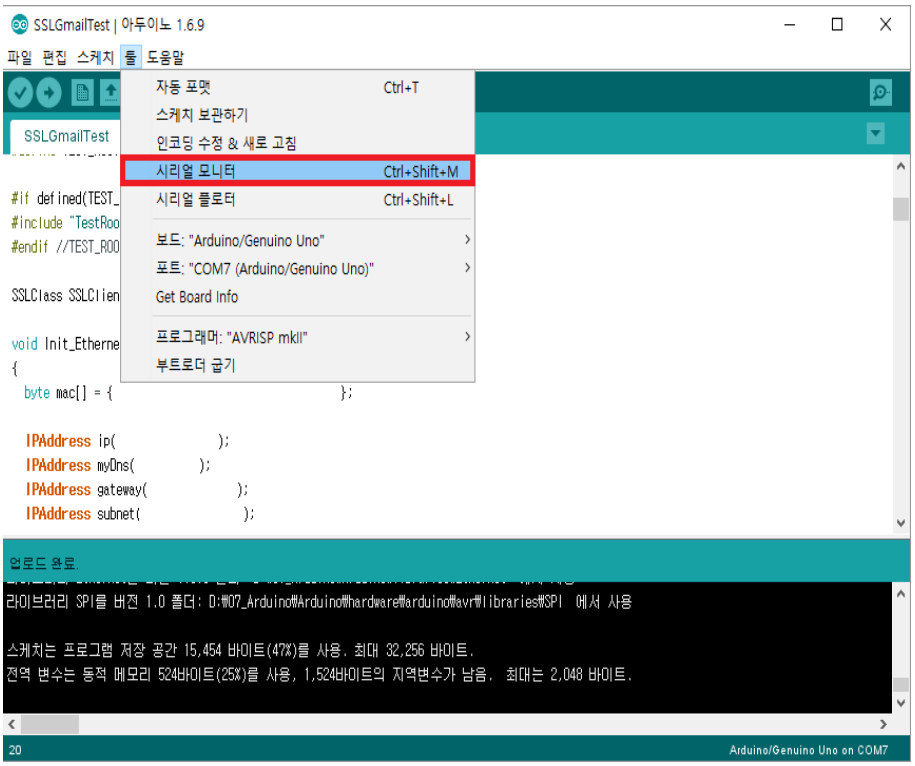
Step 2: Then, Click “Upload” to upload on Arduino board



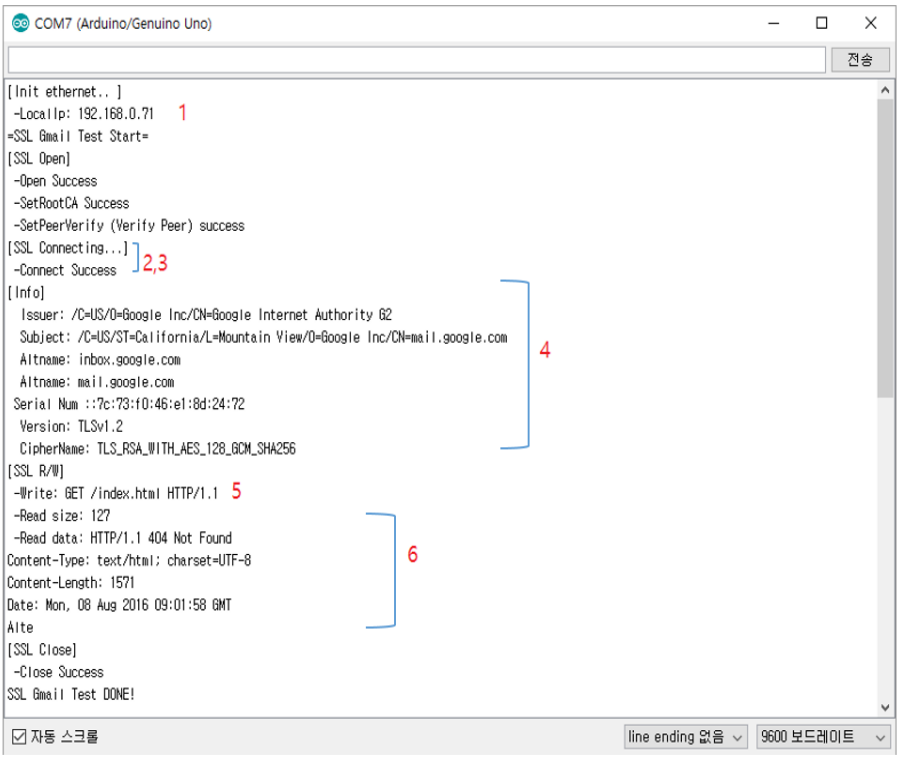
Step 3: Click Serial Monitor when Upload is completed

▶ Flashing

▶ Starting serial monitor



Step 4: Result of Gmail Test



Description:

1) DHCP Initialize and Network Configuration (Allocate IP address)
2) Received Gmail IP via DNS SERVER
3) Connecting Gmail server
4) Received peer info (issuer/subject/altname/serial number)
5) Send data to SSL.
6) Received data from server (SSL Version/Cipher Suite/Content type/Content -Length)