3G library for

G-4500 embedded controller

User Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

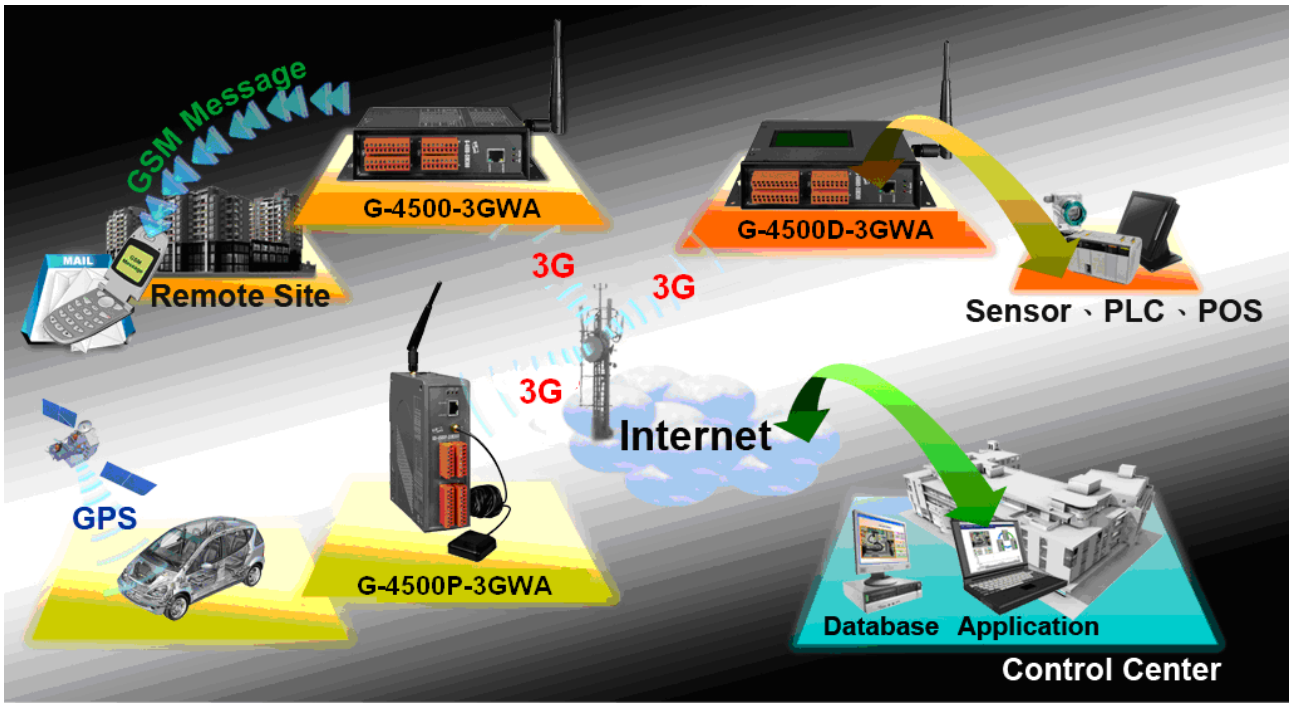
Table of Contents

Chapter 1 Introduction

1.1 3G and G-4500 series embedded controller

The 3G is a new nonvoice value added service that allows information to be sent and received across a mobile telephone network. It supplements today's Circuit Switched Data and Short Message Service. 3G is NOT related to GPS (the Global Positioning System), a similar acronym that is often used in mobile contexts. 3G facilitates instant connections whereby information can be sent or received immediately as the need arises, subject to radio coverage. No dial-up modem connection is necessary. This is why 3G users are sometimes referred to be as being "always connected". Immediacy is one of the advantages of 3G (and SMS) when compared to Circuit Switched Data. High immediacy is a very important feature for time critical applications.

ICP DAS provides the 3G library for G-4500 embedded controller. The library is an easy way to applying the 3G service in the embedded controller. Otherwise, ICP DAS supports many IO modules and GPS modules for users. Therefore, there are many application architectures to apply in the system. Or users can integrate other controller system with 3G library. The follows is a standard application architecture.



Chapter 2 3G Library

2.1 Data structure define

There are some data structure that is useful when you program with 3G library.

SMS:

```
//-- structure for sending/reading SMS
typedef struct STRENCODE_MSG{
    char phoneNumber[30];     //phone number
    char time[20];            //sms_time_stamp
    char msg[161];            //message's content
    unsigned char dataLen;    //Message's length,
                              // Max length: 7-bit=160, UCS2=140(70 word)
    char mode;                //encode style: 0=GSM_7BIT, 8=GSM_UCS2(uni-code)
} strEncode_Msg;
```

GPRS:

```
//-- structure for reading/sending GPRS sockets
typedef struct GPRSDATA{
    char data[1024];          //data
    int dataLen;              //data length
    char IP[16];              //IP，.000.000.000 and '0x00', total 16 byte
    unsigned int port;        //TCP/UDP port
    char link;                //data of link[n]
} GPRSData;

//-- structure for setting network
typedef struct NET_PROFILE
{
    char APN[60];     //APN for network provided by your cellular provider
    char user[32];    //username for network provided by your cellular provider
    char pw[32];      //password for network provided by your cellular provider
} NetProfile;
```

2.2 Function

Before using:

1. the Library use the index[30] of the NVRAM for debug. If you use the NVRAM, please skip index[30]. And index[30] = 0 is turn debug mode off, you could check it by the command “use nvram”. If it isn’t “0”, please set to “0” to be sure your program normal work.
2. If you use multi-client, 2 client link to the same IP server, your port must set different, or Library will couldn't recognize new packet from which link.
3. Server couldn’t send packet data to the client too soon, because the speed of TCP is faster than UART(G-4500 communicate the modem with UART). Suggest you send packet data no more than 1 packet / 1 second.
4. The Library will use a Timer. If you also use LCD function, you will remain Timer 1, 2 to use.

2.2.1 GM_SYS_GetLibVersion

Prototype:

int GM_SYS_LibVersion(void);

Description:

Get Lib. version

Parameter:

no

Return:

version format = A.BC

2.2.2 GM_SYS_GetLibDate

Prototype:

void GM_SYS_GetLibDate(char* libDate);

Description:

Get lib. date

Parameter: None

a string of lib. date, format="Jul 21 2010"

Return:

no

2.2.3 GM_SYS_InitModem

Prototype:

int GM_SYS_InitModem(const char* PinNum);

Description:

initialize Modem, and turn it on

*must use GM_SYS_CheckModemStatus() to check modem status later

Parameter:

PinNum: PIN code, format="xxxx", NULL mean don't send PIN

Return:

GM_NOERROR:	success
GM_COMERROR:	comport error
GM_INITERROR:	init fail error

2.2.4 GM_SYS_CloseModem

Prototype:

int GM_SYS_CloseModem(int mode);

Description:

(1) use it before you want to finish the program

(2) or your case must save the power of the battery during device working

after GM_SYS_CloseModem(1), you must use GM_SYS_InitModem() to wake up modem

Parameter:

mode: 1: close modem and set it power off

0: default, close modem, but maintain it power on

Return:

GM_NOERROR	no error
GM_ERROR:	Error

2.2.5 GM_SYS_CheckModemStatus

Prototype:

int GM_SYS_CheckModemStatus(void);

Description:

check modem status, suggest you check it in your loop every time

Parameter:

no

Return:

GM_NOERROR:	modem register success, can service
GM_NOREG:	modem not registered, can't service

2.2.6 GM_SYS_CheckCmdStatus

Prototype:

int GM_SYS_CheckCmdStatus(void);

Description:

get the status of the command you sent

Parameter:

no

Return:

GM_BUSY:	modem busy, you can't send other command
GM_NOERROR:	success
GM_TIMEOUT:	time out
GM_ERROR:	command fail error
other:	please refer to error codes of HSDPA.h

2.2.7 GM_SYS_CheckSignal

Prototype:

int GM_SYS_CheckSignal(void);

Description:

check signal quality

Parameter:

no

Return:

signal: signal quality

0	-113 dBm or less
1	-111 dBm
2...30	-109... -53 dBm
31	-51 dBm or greater
99	not known or not detectable

2.2.8 GM_SYS_CheckReg

Prototype:

int GM_SYS_CheckReg(void);

Description:

Check register

Parameter:

no

Return:

register flag, the value will fill here, when get value from modem

- 0: not registered
- 1: registered, home network
- 2: not registered, and searching...
- 3: registration denied
- 4: unknown
- 5: registered, roaming

--SMS Function--

2.2.9 GM_SMS_ReadMsg

Prototype:

int GM_SMS_ReadMsg(int index, strEncode_Msg* strMsg);

Description:

Read message(index)

* must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

index: index of sms

strMsg: sms will be put here

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.10 GM_SMS_SendMsg

Prototype:

int GM_SMS_SendMsg(strEncode_Msg* strMsg, int wait);

Description:

Send a message

* must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

strMsg: the message

Return: None

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.11 GM_SMS_GetNewMsg

Prototype:

strEncode_Msg* GM_SMS_GetNewMsg(strEncode_Msg* msg);

Description:

Get new sms message

*this command will read the new message, and delete it

Parameter:

msg:new sms message

Return:

NULL: no new message

msg:new message

2.2.12 GM_SMS_DeleteMsg

Prototype:

int GM_SMS_DeleteMsg(int index);

Description:

delete sms message(index) in SIM card

* must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

index: index of the message you want to delete

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.13 GM_SMS_DeleteAll

Prototype:

int GM_SMS_DeleteAll(void);

Description:

delete all sms message in SIM card

* must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

index: index of the message you want to delete

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

--3G / GPRS data Transmission Function—

2.2.14 GM_NET_SetNet

Prototype:

int GM_NET_SetNet(NetProfile netProfile);

Description:

Set Net profile data

Parameter:

netProfile: include APN(Access Point Name), password and username

Return:

GM_NOERROR	no error
GM_ERROR	error

2.2.15 GM_NET_OpenNet

Prototype:

int GM_NET_OpenNet(void);

* must use "GM_SYS_CheckCmdStatus()" to check status later

Description:

Start TCP/UDP Client NetWork

Parameter:

no

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.16 GM_NET_CloseNet

Prototype:

int GM_NET_CloseNet(void);

* must use "GM_SYS_CheckCmdStatus()" to check status later

Description:

Close NetWork

Parameter:

no

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.17 GM_NET_AutoRelink

Prototype:

void GM_NET_AutoRelink(int autoRelink);

Description:

Auto-Relink to server

Parameter:

autoRelink: 0=not relink, 1=auto-relink

Return:

no

2.2.18 GM_NET_GetIP

Prototype:

int GM_NET_GetIP(char* ipaddr);

Description:

Get IP

Parameter:

ipaddr: IP string, format: char ipaddr[16];

Return:

ipaddr	success
NULL	error, network may be broken

2.2.19 GM_NET_LinkTo

Prototype:

int GM_NET_LinkTo(int n, int tcp, char* ip, unsigned int port);

Description:

Link to the server as TCP or UDP client

*must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

n:	link number, general, you can build 10 link for client, 0~9
tcp:	tcp: client type, tcp=1 for TCP client ; tcp=0 for UDP client
ip:	IP that you want to link, ex: "61.111.222.333"
port:	Port that you want to link, ex: 1234

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.20 GM_NET_CloseLink

Prototype:

int GM_NET_CloseLink(int n);

Description:

Close client link[n]

*must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

n: client link[n]

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.21 GM_NET_GetLinkStatus

Prototype:

int GM_NET_GetLinkStatus(int n);

Description:

get status of Link[n]

Parameter:

n: client link[n]

Return:

status: 0=not link, 1=linked

GM_ERROR: index error

2.2.22 GM_NET_Send

Prototype:

int GM_NET_Send(GPRSData gprsData);

Description:

Send a packet

*must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

gprsData: the data you want to send, the Max. data length = 1024

Return:

GM_NOERROR	no error
GM_NOREG:	not registered, or can't service
GM_BUSY:	modem busy

2.2.23 GM_NET_GetNewPacket

Prototype:

GPRSData* GM_NET_GetNewPacket(GPRSData* gprsData);

Description:

Get the new packet

*must use "GM_SYS_CheckCmdStatus()" to check status later

Parameter:

gprsData: data pointer

Return:

NULL:	no new packet
gprsData:	new packet ,and return data pointer that you give

2.2.24 GM_NET_InstallLink

Prototype:

int GM_NET_InstallLink(NetProfile netProfile, int n, int tcp, char* serverIP, unsigned int serverPort);

Description:

Built TCP/UDP link in auto relink mode

Parameter:

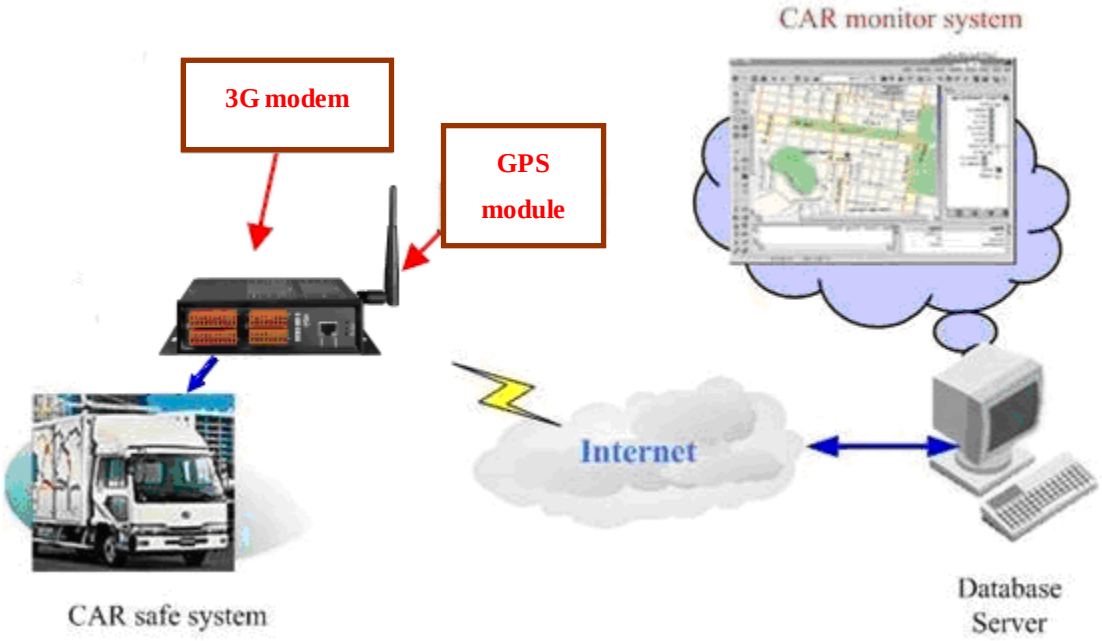
netProfile	include APN(Access Point Name), password and username
n	link number (0~9), generally, you can build 10 link for client
tcp	client type, tcp=1 for TCP client ; tcp=0 for UDP client
serverIP	IP of the server, ex: "61.111.222.333"
serverPort	TCP/UDP Port of the server, ex: 1234

Return:

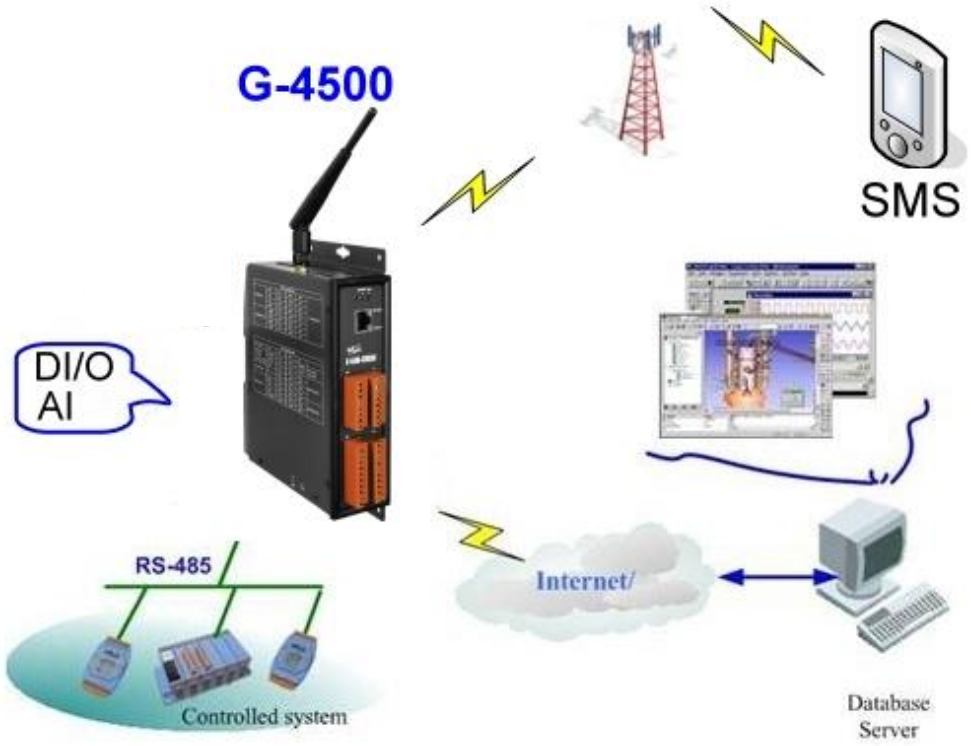
GM_NOERROR	success to built link[n]

Chapter 3 Application architecture

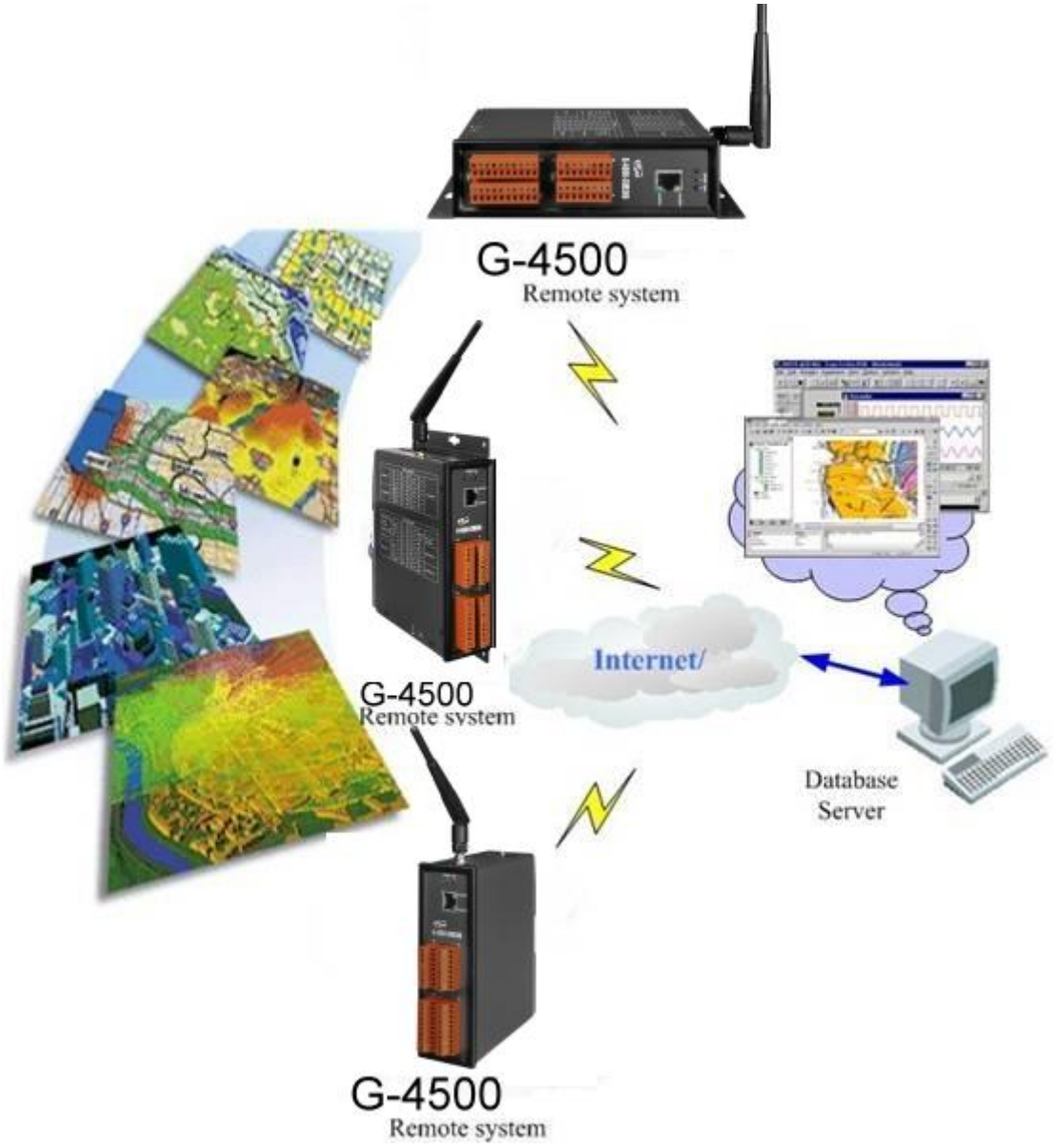
3.1 Car Monitor System



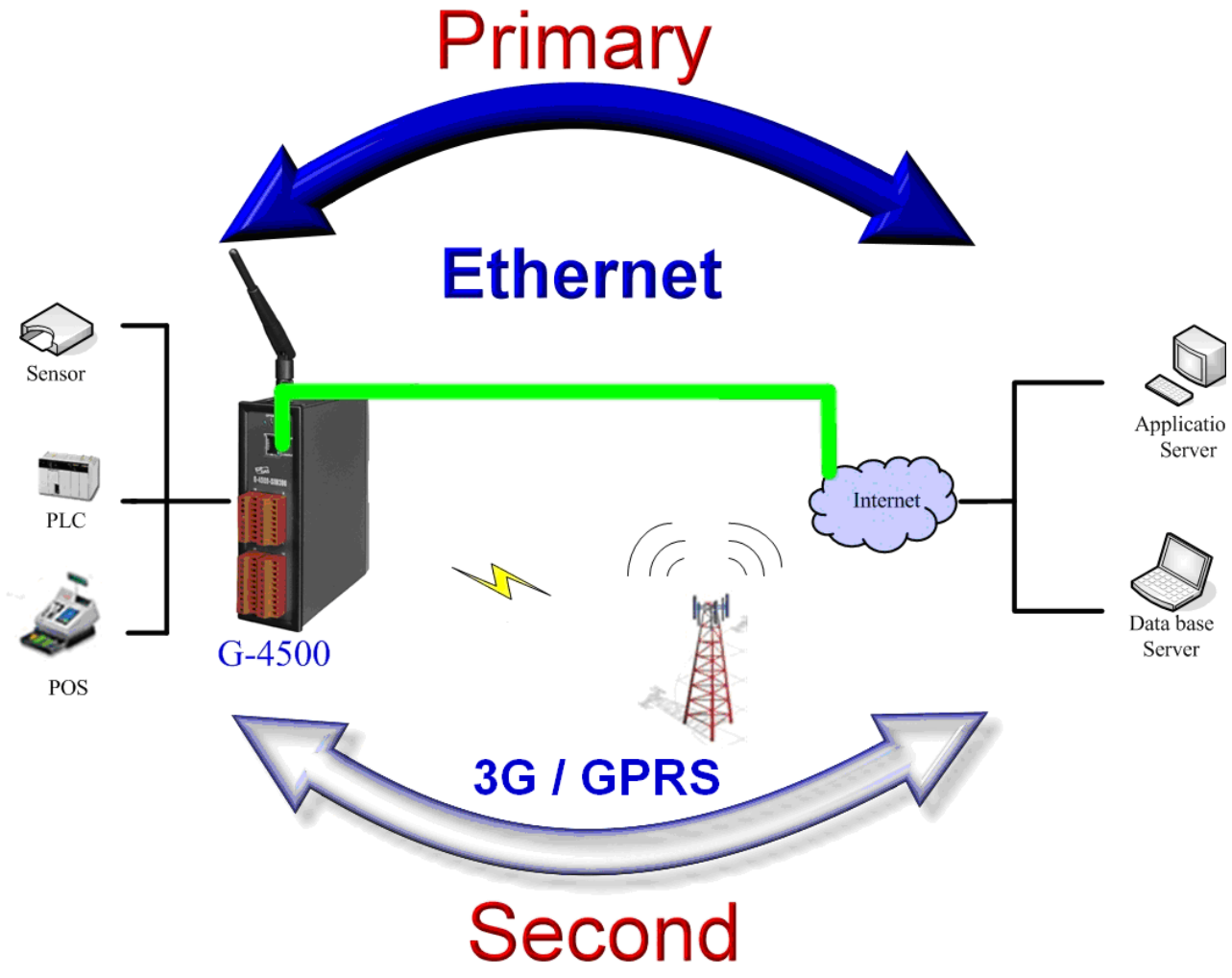
3.2 Remote Control System



3.3 GIS system



3.4 Redundance Communication System



Version Record

Version	By	Date	Description
1.01	Malo	2010/08/11	
1.02	Malo	2010/8/27	Add network function
1.03	Malo	2010/10/6	enhance network function
1.04	Malo	2010/10/29	Enhance SMS function And network function