IOT Relay Programing Manual

V1.6

1 Product Overview

1.1 Overview

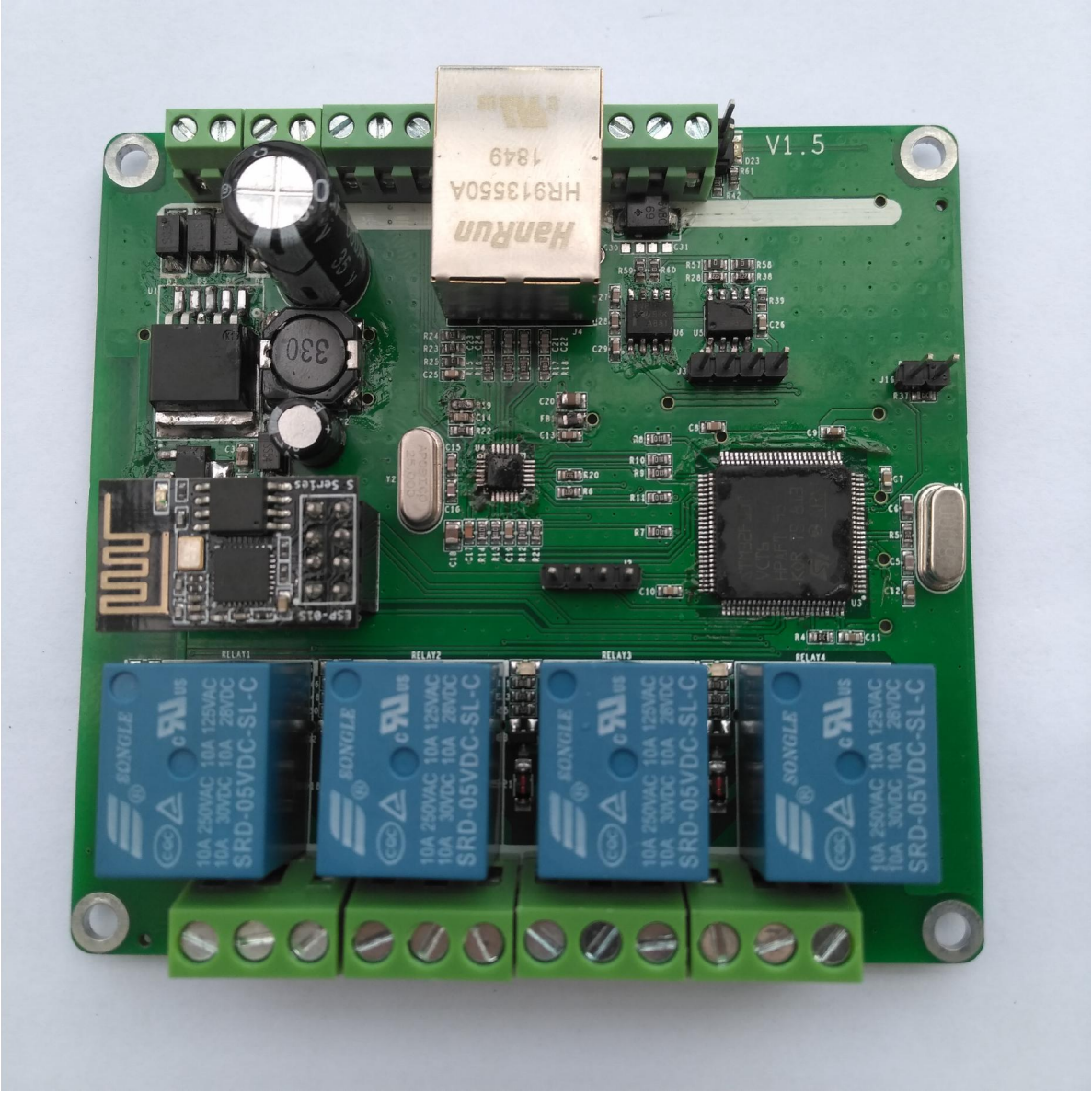
Support multiple channel relay, On/Off,Delay,Momentary
Support multiple interface RJ45/RS485/CAN/WIFI
Support HTTP GET CGI/UDP/TCP Server/TCP Client
Local Button control
PC app config and control
WEB config and control
8KB fifo command buffer
Support password.

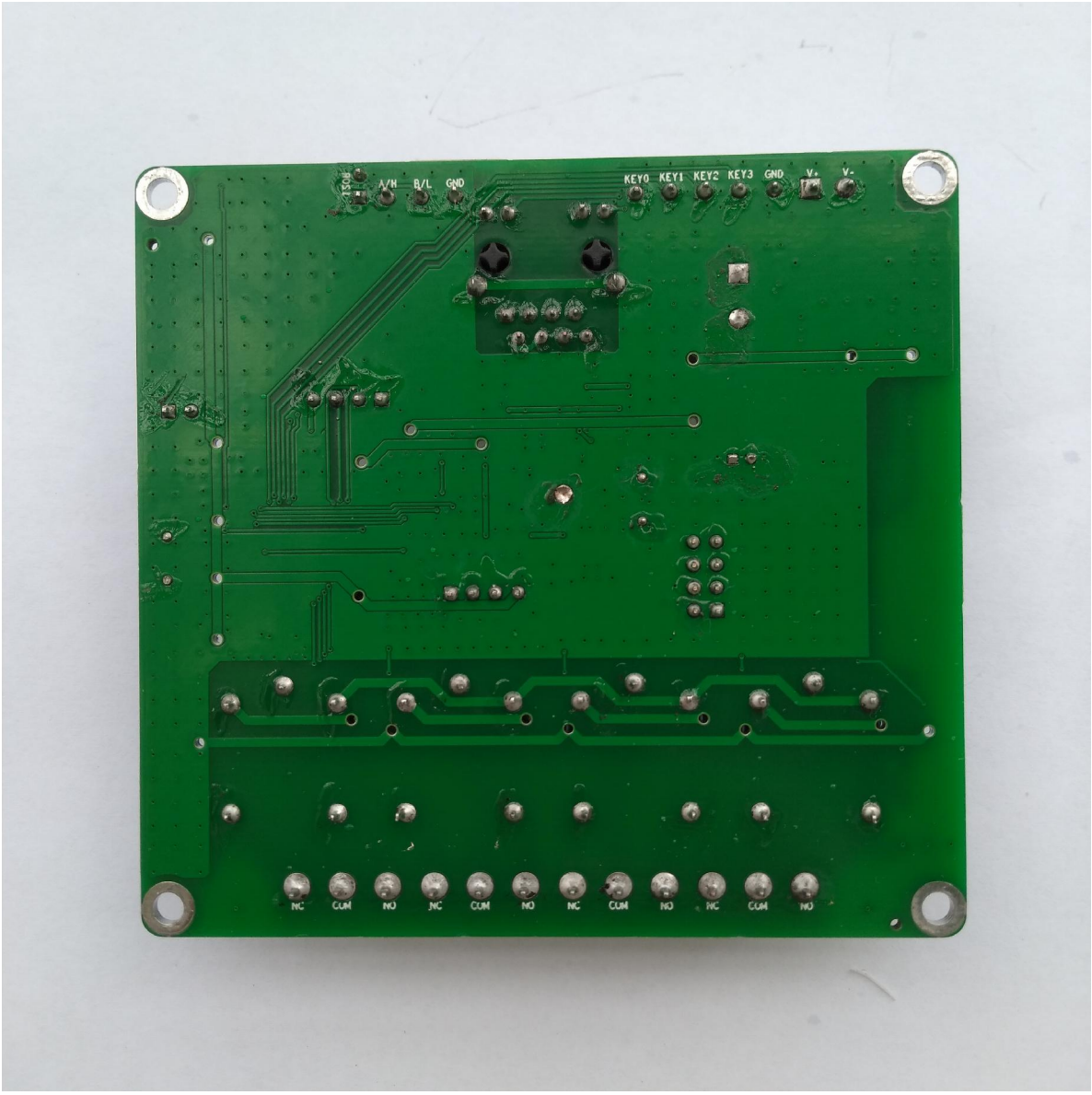
1.2 Technical Parameters

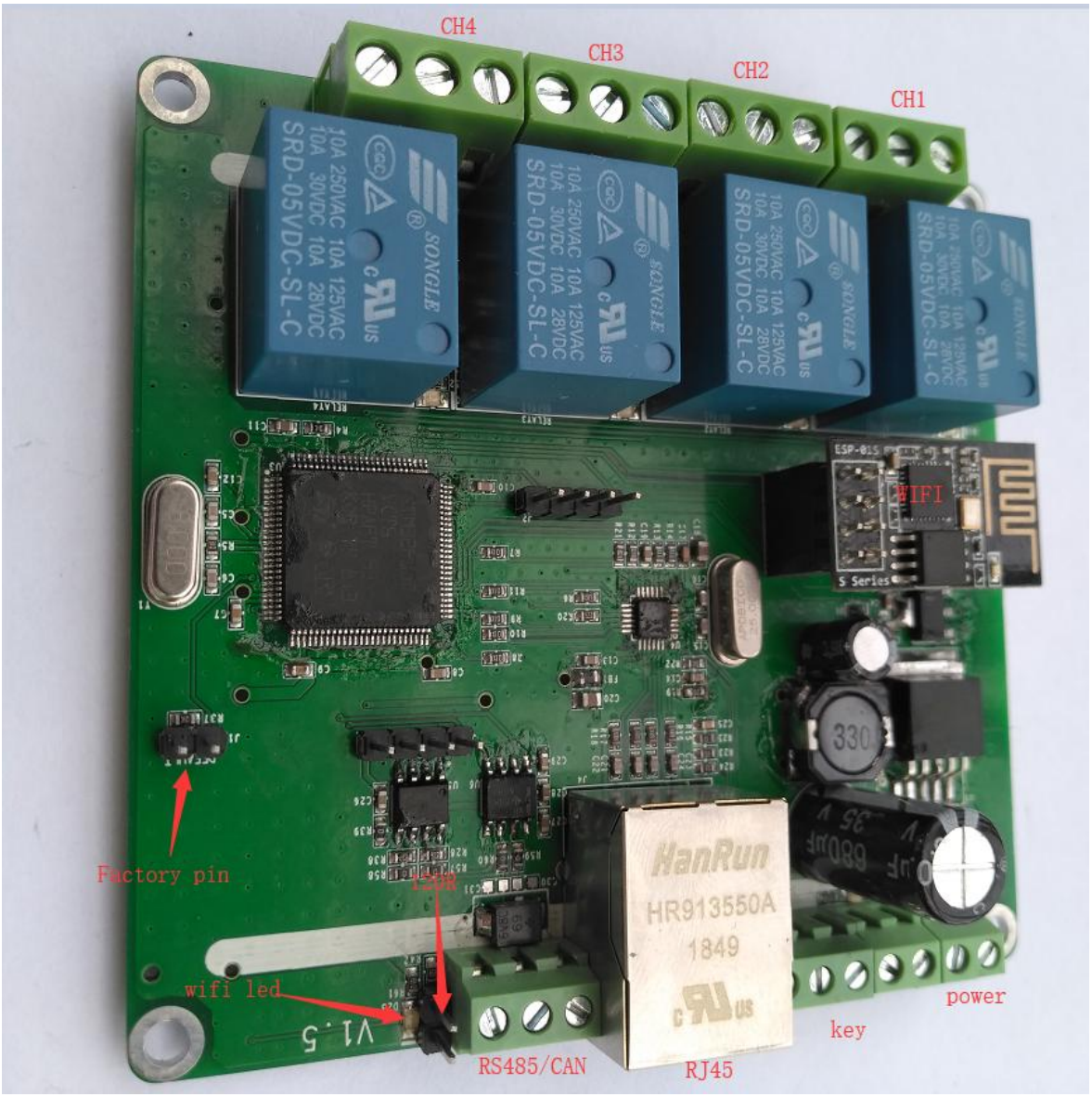
Network	Interface	RJ45/ RS485/CAN/WIFI
	Rate	100M/115200bps/125kbps/150Mbps
	Protocol	TCP server/client, UDP server/client/HTTP GET CGI, RS485, CAN, WIFI
Output	Relay Power	AC 250V/10A,DC 30V/10A
	Contacts	Normally Close Normally Open
	Delay	1~65535 seconds
	Momentary	Pull in 0.5 seconds, automatically release
Working environment	Operating temperature	-40~+85℃
Power	Power Specifications	Power supply 5-40V, Recommended 12V
	Current	200mA@12V DC
	Power consumption	Less than 5W

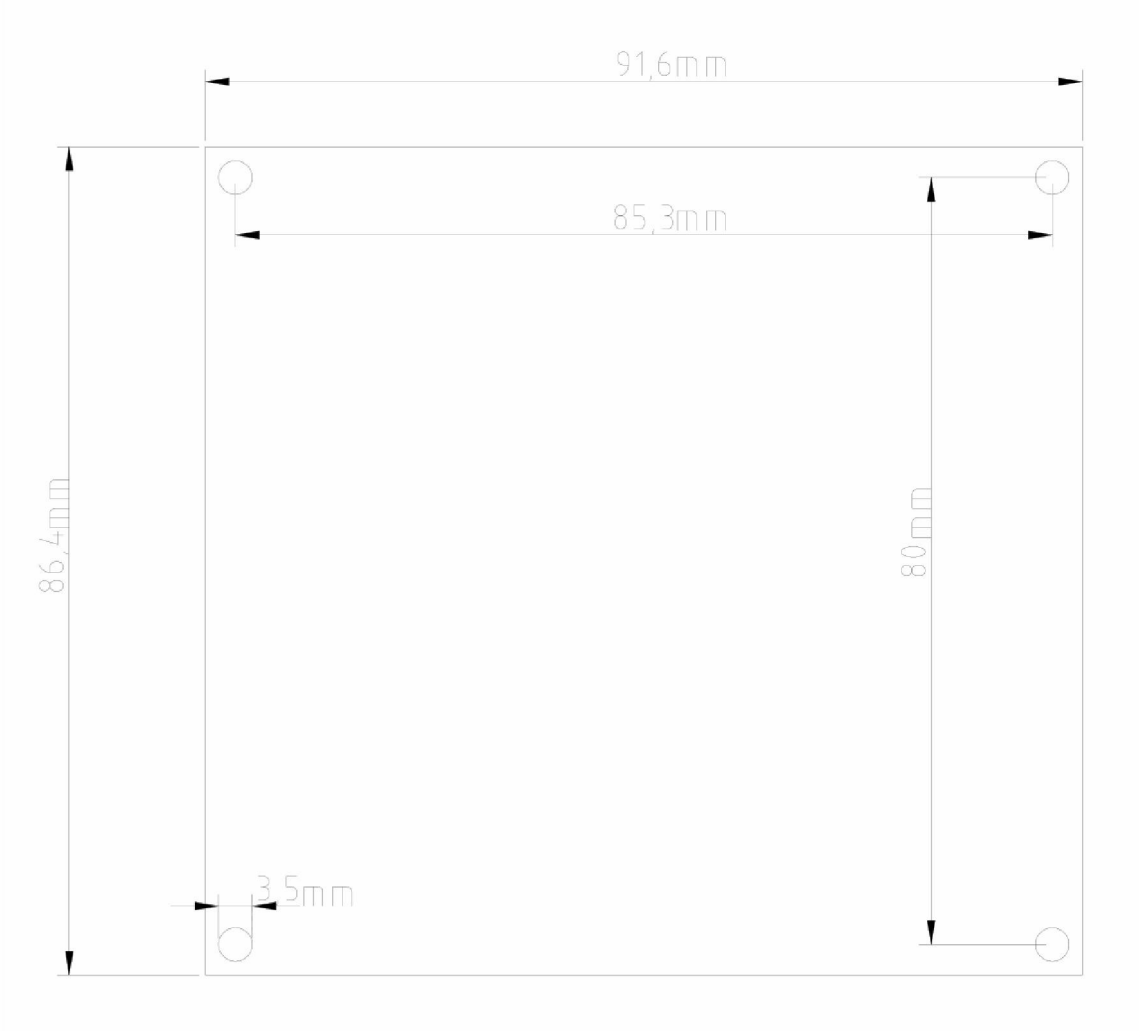
2 Size

2.1 4 Channel Relay




A/H
B/L
GND
KEY0
KEY1
KEY2
KEY3
GND
V+
V-
NC
COM
NO
NC
COM
NO
NC
COM
NO
NC
COM
NO


CH4
CH3
CH2
CH1
SRD-05VDC-SL-C
WIFI
Factory pin
120R
wifi led
V1.5
RS485/CAN
RJ45
key
power



3 Interface Description

3.1 Indictor

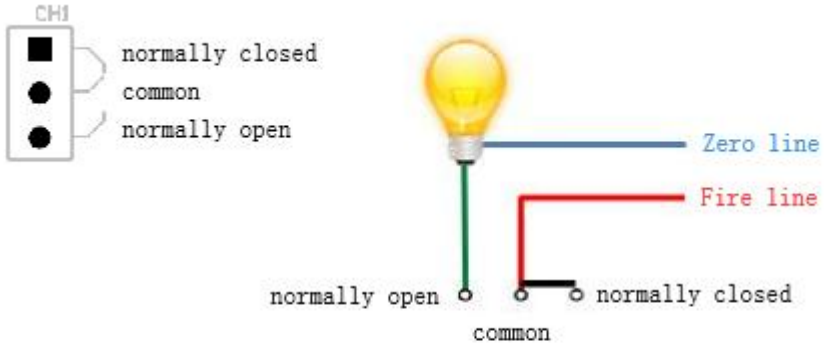
wifi led	on: connect wireless route success off: can not connect wireless router
CH1-CH8 led	on: relay on off: relay off

3.2 Relay contact

Each set of relay outputs has three terminals: normally open contact, common terminal and normally closed contact. The contact capacity is AC 250V10A, DC 30V10A, and the output of controlling higher power requires external contactor.

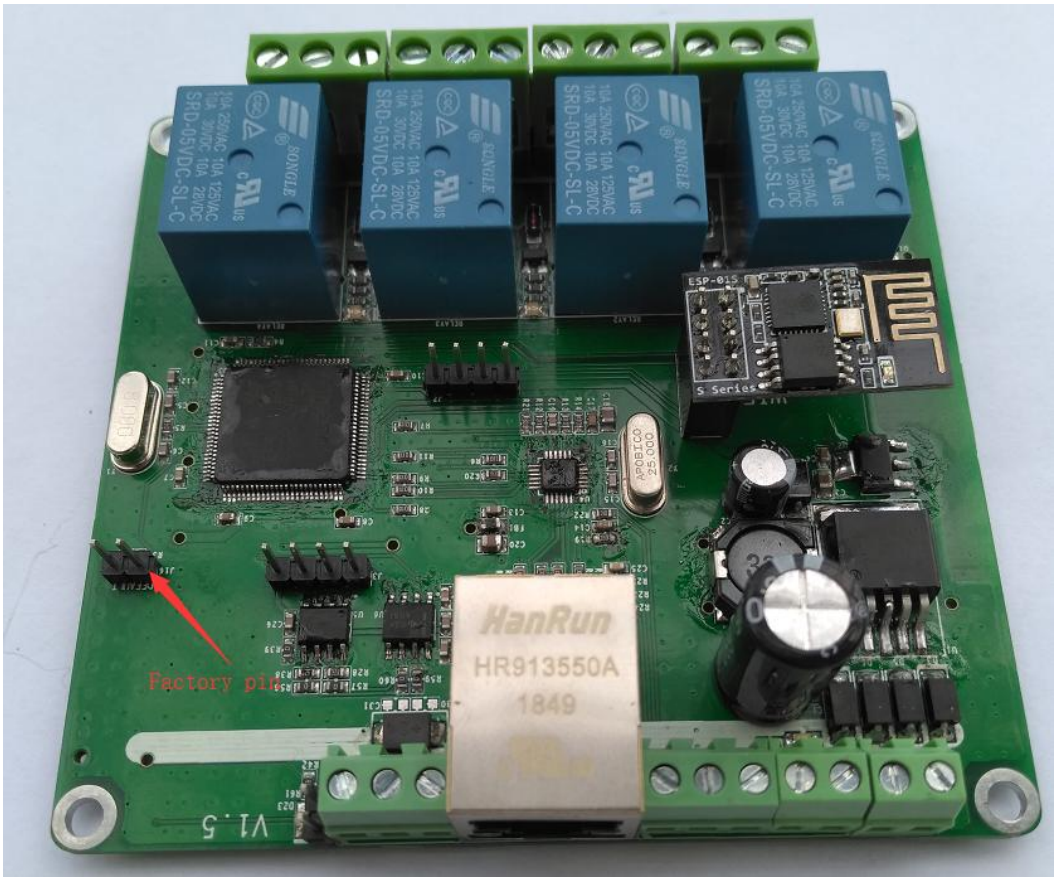
• Normally open contact:
When the relay is released (or the module is powered off), the common terminal is disconnected from the normally open contact. After the suction is closed, the two contacts are closed.
• Common:
Controlled power input
• Normally closed contact:
When the relay is released (or the module is powered down), the common and normally closed contacts are closed. After the pull-in, the two contacts are disconnected.

Connection example



3.3 Reset to factory

1 Short the 2 pin headers under the Default assembly with a jumper cap



2 Turn off the power of the network module, and then power on the module again.
3 Pull out the Default jumper cap

4 Tools

RelayTool	PC app for device
CGITool	PC test HTTP CGI API
relay.sh	unix/linux shell control relay

5 relay protocol(string)

Suport TCP client, TCP server, UDP, CAN/RS485

5.1 default setting

RJ45

IP	192.168.1.100
Netmask	255.255.255.0
Gateway	192.168.1.1
TCP Port	60001
UDP Port	60001

RS485

Baudrate	115200bps
Databits	8bits
Stopbits	1bits
Parity	None

CAN

Baudrate	125Kbps
ID	1

WIFI

IP	DHCP
UDP Port	60001

5.2 Query status command

command code	00(2 character)	return 8 character, Each character may be 0 or 1, representing a relay On or Off The state, such as the return value of 11000000, means that CH1 and CH2 are On, and the other channels are Off

Remarks

1 The command code is a text string and does not need to be followed by a return.

2 UDP mode does not support query instructions

5.3 Basic control command

CH1 On	11	The return value is the same as 5.2 Query status command
CH1 Off	21	
CH2 On	12	
CH2 Off	22	
CH3 On	13	
CH3 Off	23	
CH4 On	14	

CH4 Off	24	
CH5 On	15	
CH5 Off	25	
CH6 On	16	
CH6 Off	26	
CH7 On	17	
CH7 Off	27	
CH8 On	18	
CH8 Off	28	
All On	1X	
All Off	2X	

5.4 Delay command

The delay command consists of the basic command + ":" + delay seconds. The delay time range is 1-65535 seconds, which can be turned Off delay On or the delay is Off after On

E.g

status	Command code	result
CH1 is currently Off	11:30	CH1 On and Off automatically after 30 seconds
CH2 is currently in On	22:30	CH2 Off, automatically On after 30 seconds
CH2 is currently Off	22:30	CH2 Off(no state change), automatically On after 30 seconds

5.5 Momentary command

The momentary command consists of the basic pull-in command + “*”. The effect of the momentary is that the relay is automatically Off after 0.5 seconds of On

6 relay protocol(binary)

Only support UDP
Support Different network segment communication
Mulitcast addr: 224.0.2.11

Support password

6.1 default setting

IP	192.168.1.100
Netmask	255.255.255.0
Gateway	192.168.1.1
UDP Port	60000
Multicast addr	224.0.2.11

6.2 command

data bytes >=2byte store format is LSB
example:0x1234,store format is 0x34,0x12

format

filed	bytes	comment
command	1	0xFF: set relay 0x07: multicast set relay
result(xor 0xAA)	1	pc->device: 0 xor 0xAA device->pc: result xor 0xAA result=0 success result=other fail
session	1	0~255 device reply the same
relay command	1	0: read relay status 1:write relay 2:write relay with delay 3:write relay with momentary 4:relay keep alive
password	2	0~9999 0:no password Password incurrent device no reply
command data	x	

6.2.1 read relay status

pc send

filed	bytes	comment
command	1	0xFF

result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	0: read relay status
password	2	0~9999 0:no password

device reply

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	0: read relay status
Relay status	1	Bit0~7 map to relay relay1~8 Bit=1 relay on Bit=0 relay off

Example:

pc send:

FF AA 00 00 34 12 # password 0x1234

device reply:

FF AA 00 00 01 # relay 1 on

6.2.2 write relay

pc send

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	1:write relay
password	2	0~9999 0:no password
relay mask	1	Bit0~7 map to relay relay1~8 Bit=1,relay need update
relay set	1	Bit0~7 map to relay relay1~8 Bit=1,relay on Bit=0,relay off

device reply

filed	bytes	comment
command	1	0xFF

result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	1:write relay

Example:

pc send:

FF AA 00 01 34 12 05 01 # relay 1 on, rely 3 off

device reply:

FF AA 00 01

6.2.3 write relay with delay

pc send

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	2:write relay with delay
password	2	0~9999 0:no password
relay index and relay on/off	1	Bit0=1 relay on Bit0=0 relay off Bit1~bit7=relay index
Relay delay second	2	1~65535 second

device reply

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	2:write relay with delay

Example:

pc send:

FF AA 00 02 34 12 03 05 # relay 1 on, delay 5 second off

device reply:

FF AA 00 02

6.2.4 write relay with momentary

pc send

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	3:write relay with momentary
password	2	0~9999 0:no password
relay index and relay on/off	1	Bit0=1 relay on Bit0=0 relay off Bit1~bit7=relay index

device reply

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 device not change
relay command	1	3:write relay with momentary

Example:

pc send:

FF AA 00 03 34 12 05 05 # relay 2 on, momentary

device reply:

FF AA 00 03

6.2.5 relay keep alive

device send

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 pc not change
relay command	1	4: relay keep alive
device MAC	6	device MAC address
Relay status	1	Bit0~7 map to relay relay1~8 Bit=1 relay on Bit=0 relay off

pc reply

filed	bytes	comment
command	1	0xFF
result(xor 0xAA)	1	0 xor 0xAA=0xAA
session	1	0~255 pc not change
relay command	1	4: relay keep alive

Example:

device send:

FF AA 00 04 BC 34 88 12 34 56 00 # MAC BC:34:88:12:34:56   00:all relay off

pc reply:

FF AA 00 00

7 config protocol (binary)

Only support UDP

Support Different network segment communication

Mulitcast addr: 224.0.2.11

7.1 default setting

IP	192.168.1.100
Netmask	255.255.255.0
Gateway	192.168.1.1
UDP Port	60000
Multicast addr	224.0.2.11

7.2 config struct

```
#define WEB_USER_LEN            (16)
#define WEB_PASSWORD_LEN        (16)
#define MAC_LEN                 (6)
struct rs232_conf
{
    u32 baudrate;/* 1200/2400/4800/9600/19200/38400/57600/115200bps */
    u32 databits;/* 8:8bit1 */
    u32 stopbits;/* 1:1bits 2:2bits */
    u32 parity;/* 0:none 1:odd 2:even    */
```

```
};
struct can_id
{
	u32 id		:29;
	u32 res		:1;
	u32 remote	:1;
	u32 extid	:1;
};/* 4bytes */
enum CAN_BAUD
{
	CAN_BAUD_START,
	CAN_BAUD_5K = CAN_BAUD_START,
	CAN_BAUD_10K,
	CAN_BAUD_20K,
	CAN_BAUD_25K,
	CAN_BAUD_50K,
	CAN_BAUD_100K,
	CAN_BAUD_125K,
	CAN_BAUD_200K,
	CAN_BAUD_250K,
	CAN_BAUD_500K,
	CAN_BAUD_800K,
	CAN_BAUD_888K,
	CAN_BAUD_1000K,
};
struct can_conf
{
	u32 baudrate;/* enum CAN_BAUD */
	struct can_id id;
};
enum
{
	DNS_START,
	DNS_UDP = DNS_START,
	DNS_TCP,
	DNS_MQTT,

	DNS_END,
};
#define SERVER_URL_MAX	(32)
struct server_port
{
	char server[SERVER_URL_MAX];
	u32 port;
```

```
};
#define RELAY_MAX           (32)

#define ROUTER_SSID_LEN        (64)
#define ROUTER_PWD_LEN         (64)

#define KEEP_ALIVE_MAX         (120)
#define KEEP_ALIVE_DEF         (30)
#define KEEP_ALIVE_CLOSE       (0)

#define MOMENTARY_100MS_MAX         (255)
#define MOMENTARY_100MS_DEF         (5)
#define MOMENTARY_100MS_MIN         (1)

//#pragma pack(1)
struct param
{
	u32 sn;
	u32 sw_ver;
	u32 hw_ver;
	u32 model;

	char web_passwd[WEB_PASSWORD_LEN];

	u32 ip;
	u32 netmask;
	u32 gateway;
	u32 dns;
	b8 dhcp;
	u8 res;
	u8 mac[MAC_LEN];

	struct rs232_conf rs232;

	struct can_conf can;

	struct server_port sp[DNS_END];

	u16 relay_password;
	u8 power_failure_recovery;/* power failure recover relay status */
	u8 relay_cnt;
	u32 relay_status;/* 1bit per relay */
	u8 key_type[RELAY_MAX];
```

```
    char router_ssid[ROUTER_SSID_LEN];
    char router_pwd[ROUTER_PWD_LEN];
    u8 keep_alive_second;/* 1~120 second */
    u8 momentary_100ms;/* momentary 100ms count */
    u8 res2[2];
};/* 360 bytes */
```

7.3 command

data bytes >=2byte store format is LSB
example:0x1234,store format is 0x34,0x12

notice:
multicast command reply to 224.0.2.11 forever

format

filed	bytes	comment
command	1	3: read info+config 4: write config 5: multicast read info+config 6: multicast write config
result(xor 0xAA)	1	pc->device: 0 xor 0xAA device->pc: result xor 0xAA result=0 success result=other fail
command data	xx	

7.3.1 read config

pc send

filed	bytes	comment
command	1	3: read info+config or 5: multicast read info+config
result(xor 0xAA)	1	0 xor 0xAA=0xAA

device reply

filed	bytes	comment
command	1	0x03
result(xor 0xAA)	1	0 xor 0xAA=0xAA
config	360	struct param

7.3.2 write config

pc send

filed	bytes	comment
command	1	3: write config or 5: multicast write config
result(xor 0xAA)	1	0 xor 0xAA=0xAA
config	360	struct param

device reply

filed	bytes	comment
command	1	0x03
result(xor 0xAA)	1	0 xor 0xAA=0xAA

8 relay protocol(HTTP GET CGI)

Relay board as HTTP server, accept HTTP GET CGI request.
Support CGI relay on/off
Support CGI relay momentary
Support CGI relay delay
Support CGI password verification

8.1 load relay status

HTTP GET request

parameter	filed	data	comment
1	CGI API	relay_cgi_load.cgi	cgi changeable suffix relay_cgi_load.cgi, relay_cgi_load.php, relay_cgi_load.cs is work ok

HTTP GET respond

parameter	filed	data	comment
1	result	0	0: ok other fail
2	relay count	2/4/8	
3	relay 1	0/1	0:off

	status		1:on
4	relay 2 status	0/1	0:off 1:on
5	relay 3 status	0/1	0:off 1:on
6	relay 4 status	0/1	0:off 1:on
7	relay 5 status	0/1	0:off 1:on
8	relay 6 status	0/1	0:off 1:on
9	relay 7 status	0/1	0:off 1:on
10	relay 8 status	0/1	0:off 1:on

example(4 channel relay):

HTTP GET request

http://192.168.1.100/relay_cgi_load.cgi # request relay board HTTP CGI API

HTTP GET respond

&0&4&1&0&1&0& # ok,4 relay,relay 1 on,relay 2 off,relay 3 on, relay 4 off

8.2 set relay

HTTP GET request

parameter	filed	data	comment
1	CGI API	relay_cgi.cgi	cgi suffix variable relay_cgi.cgi, relay_cgi.php, relay_cgi.cs is work ok
2	type	0/1/2	0:relay on/off 1:relay momentary 2:relay delay
3	relay	0~8	
4	on	0/1	0:off 1:on
5	time	0 1~255 1~65535	0:type 0:time 1:type 1~255:time(1=100ms)

			2:type 1~65535:time(second)
6	pwd	0~9999	0~9999 Password incurrent device no respond

HTTP GET respond

parameter	filed	data	comment
1	CGI API	relay_cgi.cgi	cgi suffix variable relay_cgi.cgi, relay_cgi.php, relay_cgi.cs is work ok
2	type	0/1/2	0:relay on/off 1:relay momentary 2:relay delay
3	relay	0~7	0:relay 1 1:relay 2 ... 7:relay 8
4	on	0/1	0:off 1:on
5	time	0 1~255 1~65535	0:type 0:time 1:type 1~255:time(1=100ms) 2:type 1~65535:time(second)
6	pwd	0~9999	0~9999 Password incurrent device no respond

example 1(relay on):

HTTP GET request(request relay board HTTP CGI API, set relay 0 on ,time 0,password 0)

http://192.168.1.100/relay_cgi.cgi?type=0&relay=0&on=1&time=0&pwd=0&

HTTP GET respond

&0&0&0&1&0& # ok, type 0 on/off,relay 0 on,time 0

example 2(relay off):

HTTP GET request(request relay board HTTP CGI API, set relay 0 off ,time 0,password 0)

http://192.168.1.100/relay_cgi.cgi?type=0&relay=0&on=0&time=0&pwd=0&

HTTP GET respond

&0&0&0&0&0& # ok, type 0 on/off,relay 0 off,time 0

example 3(relay 1 momentary on):
HTTP GET request(request relay board HTTP CGI API, set relay 1 momentary on ,time 500ms,password 4660)
http://192.168.1.100/relay_cgi.cgi?type=1&relay=1&on=1&time=5&pwd=4660&
HTTP GET respond
&0&1&1&1&5& # ok, type 1 momentary,relay 1 on,time 5(500ms)

example 4(relay 1 momentary off):
HTTP GET request(request relay board HTTP CGI API, set relay 1 momentary off,time 500ms,password 4660)
http://192.168.1.100/relay_cgi.cgi?type=1&relay=1&on=0&time=5&pwd=4660&
HTTP GET respond
&0&1&1&0&5& # ok, type 1 momentary,relay 1 off,time 5(500ms)

example 5(relay 1 on delay 10 second off):
HTTP GET request(request relay board HTTP CGI API, set relay 1 on delay 10 second off ,time 5 second,password 4660)
http://192.168.1.100/relay_cgi.cgi?type=2&relay=1&on=1&time=10&pwd=4660&
HTTP GET respond
&0&2&1&1&10& # ok, type 2 delay,relay 1 on,time 10 second

example 6(relay 1 off delay 10 second on):
HTTP GET request(request relay board HTTP CGI API, set relay 1 off delay 10 second on ,time 5 second,password 4660)
http://192.168.1.100/relay_cgi.cgi?type=2&relay=1&on=0&time=10&pwd=4660&
HTTP GET respond
&0&2&1&0&10& # ok, type 2 delay,relay 1 off,time 10 second