

Easylighting Protocol

Updates:

- Add `Running filter with audio meter` 20180207

There are two types of protocol:

1. Device Protocol
2. Control Protocol

Device Protocol (use udp, port is 48899)

Search device

First, make sure your phone is in the same environment as the device.

Init **UdpSocket** and then enableBroadcast, bind to port **48899**, begin receiving.

Now get your phone IP such as 192.168.0.8, replace the last number with **255**, build the **broadcast IP** as 192.168.0.**255**.

Send search data by UdpSocket 30 times.

iOS example:

```
NSData *data = [@"HF-A11ASSISTHREAD" dataUsingEncoding:NSUTF8StringEncoding];
NSString *host = @"192.168.0.255"; (broadcast IP)
for (int i = 0; i < 30; i++) {
    [udpSocket sendData: data toHost:host port: 48899 withTimeout: -1 tag: 0];
    [NSThread sleepForTimeInterval:0.05];
}
```

If there is some device, you can receive data by udpSocket from address(IP). The data needs to be filtered out which the from address(IP) is the same as the phone address(IP).

iOS example:

```
NSString *receiveString = [NSString stringWithUTF8String:data.bytes];
(receiveString = @"192.168.0.254,ACCF23F3EA1C,")
receiveString separated by ","
NSArray * receiveStringComponents = [receiveString componentsSeparatedByString:@","];
```

```
receiveStringComponents[0] = @"192.168.0.254"; // this is the device' address(IP)
receiveStringComponents[1] = @"ACCF23F3EA1C"; // this is the devices Mac
```

Now you get the device IP and device Mac, connect to **device IP** on port **8899** with TcpSocket, when TcpSocket connected, you can control device with Color Control Protocol.

Connect device to home network

1. Send search device

Make sure phone connect the device WiFi directly.

Init **UdpSocket** and then enableBroadcast, bind to port **48899**, begin receiving.

Now get your phone IP such as 192.168.0.8, replace the last number with **255**, build the **broadcast IP** as 192.168.0.**255**.

Send search data by UdpSocket 30 times.

iOS example:

```
NSData *data = [@"HF-A11ASSISTHREAD" dataUsingEncoding:NSUTF8StringEncoding];
NSString *host = @"192.168.0.255"; (broadcast IP)
for (int i = 0; i < 30; i++) {
    [udpSocket sendData: data toHost:host port: 48899 withTimeout: -1 tag: 0];
    [NSThread sleepForTimeInterval:0.05];
}
```

2. Send "+ok"

When received correct data like (Search Device Protocol), then send "+ok" data 3 times.

iOS example:

```
NSData *okData = [@"+ok" dataUsingEncoding:NSUTF8StringEncoding];
for (int i = 0; i < 3; i++) {
    [udpSocket sendData: okData toHost: host port: 48899 withTimeout: -1 tag: 1];
    [NSThread sleepForTimeInterval: 0.05];
}
```

3. Send search WiFi nearby

After send 3 times okData you need send search WiFi nearby 3 times.

iOS example:

```
NSMutableData *searchWiFiData = [NSMutableData data];
NSData *header = [@"AT+WSCAN" dataUsingEncoding:NSUTF8StringEncoding];
[searchWiFiData appendData:header];

unsigned const char end = 0x0D;
[searchWiFiData appendBytes:&end length:1];

for (int i = 0; i < 3; i++) {
    [udpSocket sendData: searchWiFiData.copy toHost: host port: 48899 withTimeout: -1 tag: 2];
    [NSThread sleepForTimeInterval: 0.05];
}
```

You'll received data string like:

```
NSString *receiveString = [NSString stringWithUTF8String:data.bytes];
```

receiveString:

```
1,ChinaNet-sunricher,0c:4c:39:2e:6c:3d,WPA1PSKWPA2PSK/AES,29,11b/g/n,NONE,In,NO,
1,EASYCOLOR,ac:cf:23:02:b6:e8,WPA2PSK/AES,100,11b/g/n,NONE,In,NO,
3,szsunricher,00:11:f7:07:0e:74,WPA2PSK/TKIPAES,55,11b/g/n,ABOVE,In,NO,
3,sunricher88,00:11:f7:07:0e:75,WPAPSK/TKIPAES,34,11b/g/n,ABOVE,In,NO,
```

```
NSArray *wifiInfoStringArray= [receivedString
componentsSeparatedByCharactersInSet:[NSCharacterSet newlineCharacterSet]];
```

```
NSString *firstWiFiInfoString = wifiInfoStringArray[0];
(1,ChinaNet-sunricher,0c:4c:39:2e:6c:3d,WPA1PSKWPA2PSK/AES,29,11b/g/n,NONE,In,NO,)
```

```
NSArray *comps = [receiveString componentsSeparatedByString:@" "];
```

comps[0]	comps[1]	comps[2]	comps[3]	comps[4]	...
Channel	SSID	BSSID	Security Type	Signal(%)	...

SSID: WiFi name

Security Type: WiFi encryption type, there are 4 types

- 1: OPEN
- 2: SHARED
- 3: WPAPSK
- 4: WPA2PSK

4. Selected the WiFi which want to connect, mark the SSID and Security

5. Send WiFi Name (SSID)

```
NSMutableData *wifiNameData = [NSMutableData
NSData *headerData = [@"AT+WSSSID=" dataUsingEncoding:NSUTF8StringEncoding];
NSData *nameData = [wifiName dataUsingEncoding:NSUTF8StringEncoding];
unsigned const char end = 0x0D;
[wifiNameData appendData:headerData];
[wifiNameData appendData:nameData];
[wifiNameData appendBytes:&end length:1];

for (int i = 0; i < 3; i++) {
    [udpSocket sendData: wifiNameData.copy toHost: host port: 48899 withTimeout: -1 tag: 2];
    [NSThread sleepForTimeInterval: 0.05];
}
```

6. Send Password

```
NSArray *securityComponents = [securityType componentsSeparatedByString:@" /"];

NSMutableData *result = [NSMutableData data];

NSData *header = [@"AT+WSKEY=" dataUsingEncoding:NSUTF8StringEncoding];

NSString *xxx = [NSString stringWithFormat:@"%s", securityComponents.firstObject];
NSString *yyy = [NSString stringWithFormat:@"%s", securityComponents[1]];

if ([xxx rangeOfString:@"OPEN"].length > 0 || [yyy rangeOfString:@"NONE"].length > 0) {
    xxx = @", ";
}

if ([xxx rangeOfString:@"WPA2PSK"].length > 0) {
    xxx = @"WPA2PSK, ";
}

if ([yyy rangeOfString:@"AES"].length > 0) {
    yyy = @"AES, ";
}
```

```
NSData *xxxData = [xxx dataUsingEncoding:NSUTF8StringEncoding];
NSData *yyyData = [yyy dataUsingEncoding:NSUTF8StringEncoding];
NSData *zzzData = [password dataUsingEncoding:NSUTF8StringEncoding];
```

```
unsigned const char end = 0x0D;
```

```
[result appendData:header];
[result appendData:xxxData];
[result appendData:yyyData];
[result appendData:zzzData];
[result appendBytes:&end length:1];
```

```
for (int i = 0; i < 3; i++) {
    [udpSocket sendData: result toHost: host port: 48899 withTimeout: -1 tag: 2];
    [NSThread sleepForTimeInterval: 0.05];
}
```

7. Send AT+WMODE

```
NSMutableData *result = [NSMutableData data];
```

```
NSData *header = [@"AT+WMODE=STA" dataUsingEncoding:NSUTF8StringEncoding];
[result appendData:header];
```

```
unsigned char end = 0x0D;
[result appendBytes:&end length:1];
```

```
for (int i = 0; i < 3; i++) {
    [udpSocket sendData: result toHost: host port: 48899 withTimeout: -1 tag: 2];
    [NSThread sleepForTimeInterval: 0.05];
}
```

8. Send AT+Z (end)

```
NSMutableData *result = [NSMutableData data];
```

```
NSData *header = [@"AT+Z" dataUsingEncoding:NSUTF8StringEncoding];
[result appendData:header];
```

```

unsigned char end = 0x0D;
[result appendBytes:&end length:1];

for (int i = 0; i < 3; i++) {
    [udpSocket sendData: result toHost: host port: 48899 withTimeout: -1 tag: 2];
    [NSThread sleepForTimeInterval: 0.05];
}

```

Restore device settings to factory default settings

1. Send search device

Make sure phone connect the device WiFi directly.

Init **UdpSocket** and then enableBroadcast, bind to port **48899**, begin receiving.

Now get your phone IP such as 192.168.0.8, replace the last number with **255**, build the **broadcast IP** as 192.168.0.**255**.

Send search data by UdpSocket 30 times.

iOS example:

```

NSData *data = [@"HF-A11ASSISTHREAD" dataUsingEncoding:NSUTF8StringEncoding];
NSString *host = @"192.168.0.255"; (broadcast IP)
for (int i = 0; i < 30; i++) {
    [udpSocket sendData: data toHost:host port: 48899 withTimeout: -1 tag: 0];
    [NSThread sleepForTimeInterval:0.05];
}

```

2. Send "+ok"

When received correct data like (Search Device Protocol), then send "+ok" data 3 times.

iOS example:

```

NSData *okData = [@"+ok" dataUsingEncoding:NSUTF8StringEncoding];
for (int i = 0; i < 3; i++) {
    [udpSocket sendData: okData toHost: host port: 48899 withTimeout: -1 tag: 1];
    [NSThread sleepForTimeInterval: 0.05];
}

```

3. Send Restore

```

NSData *restoreData = [[SRLEDData sharedInstance] dataForRestoreLED];
for (int j = 0; j < 90; j++) {
    [self sendDataToUDP:restoreData times:1];
    [NSThread sleepForTimeInterval:0.02];
}

```

The interval between 1 to 3 steps is 0.5 seconds.

```

send search device -> [NSThread sleepForTimeInterval: 0.5] -> send "+ok" ->
[NSThread sleepForTimeInterval: 0.5] -> send restore -> end

```

Control Protocol (use tcp, port is 8899)

You need search device at first, when you get the device IP and device Mac, connect to **device IP** on port **8899** with TcpSocket, when TcpSocket connected, you can control device with Color Control Protocol.

Protocol Formatter

1	2	3	4	5	6	7	8	9	10	11	12
0x55	device id			device type	room selected	data type	key	value	sum	0xAA	0xAA

12 bytes per data for Color Control Protocol;

1, 11 and 12 is const value;

device id: Randomly generated device ID when first run App, total 3 bytes. Later you need to use this number as device ID. It is the device used to distinguish between different App.

device type: Always set to 0x01;

room selected: 1 byte, each bit for a room. For example you selected the first, third and the fourth room then this value will be 0b1301 (0x0D), if you selected all rooms then this value will be 0b11111111 (0xFF). If you want to learning lamps, you need and can only selected one room.

data type: Used to distinguish between functions.

key: The corresponding remote control key.

value: The corresponding remote control value.

sum: For check. $\text{sum} = (\text{deviceType} + \text{roomSelected} + \text{dataType} + \text{key} + \text{value}) \& 0xFF$;

RGB

dataType: 0x01

key: 0x01

value: 0x01 ~ 0x60 (decimalism: 1 ~ 96)

The correspondence between color and value as follows image, for example the Red color's value is 44 (0x2C)

Red Slider

dataType: 0x08

key: 0x48

value: 0x00 ~ 0xFF (0 ~ 255)

Green Slider

dataType: 0x08

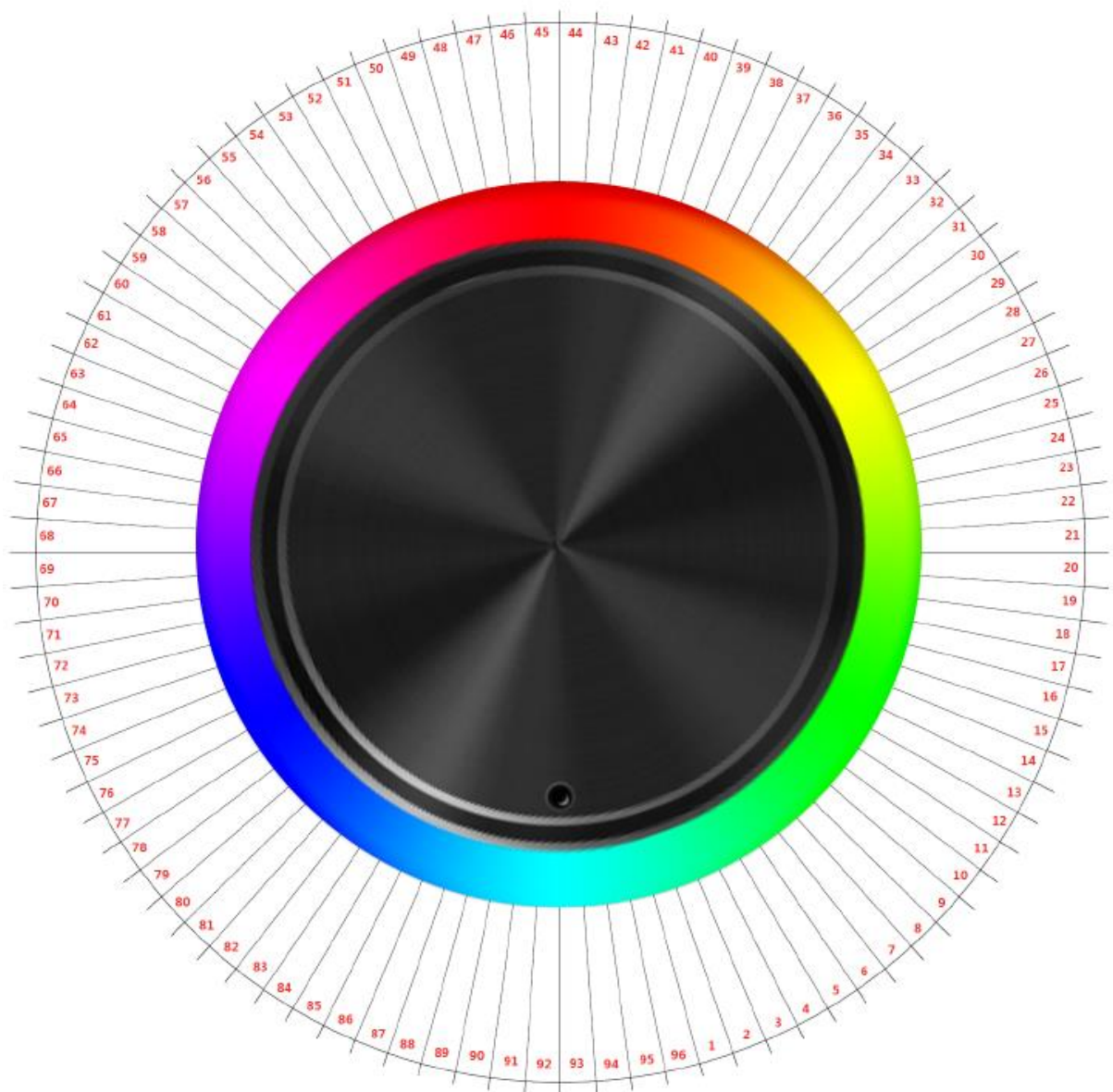
key: 0x49

value: 0x00 ~ 0xFF

Blue Slider

dataType: 0x08

key: 0x4A



value: 0x00 ~ 0xFF

White Slider

dataType: 0x08

key: 0x4B

value: 0x00 ~ 0xFF

Brightness Slider for RGB

dataType: 0x08

key: 0x4C

value: 0x01 ~ 0x40

Run Start for RGB

dataType: 0x02

key: 0x4E

value: 0x01 ~ 0x0A

There are ten modes of running.

Run Stop for RGB

dataType: 0x02

key: 0x4F

value: 0x01

Running filter with audio meter

When LED is running, if the meter of audio meter reaches a certain value, we can send this command to change the color at the moment, make LED looks like It's change according to audio meter.

datatype: 0x03

key: 0x50

value: 0x01

Run Speed for RGB

dataType: 0x08

key: 0x22

value: 0x01 ~ 0x0A

There are ten levels of speed adjustment.

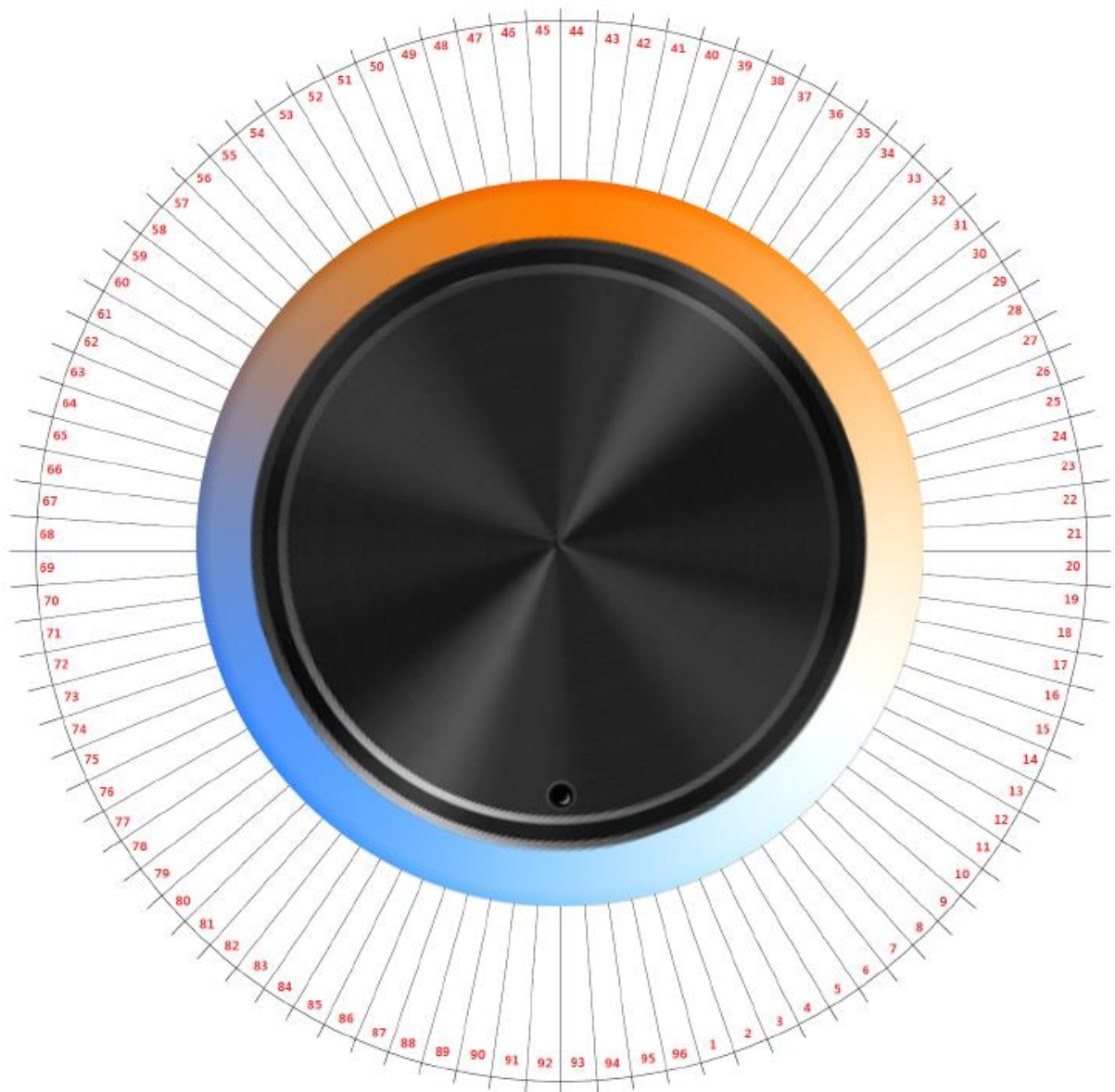
CDW

dataType: 0x01

key: 0x01

value: 0x01 ~ 0x60 (decimalism: 1 ~ 96)

(Just like RGB)



Cold Slider

dataType: 0x08

key: 0x48

value: 0x00 ~ 0xFF (0 ~ 255)

(Just like red slider)

D Slider

dataType: 0x08

key: 0x49

value: 0x00 ~ 0xFF

(Just like green slider)

Warm Slider

dataType: 0x08

key: 0x4A

value: 0x00 ~ 0xFF

(Just like blue slider)

Brightness Slider for CDW

dataType: 0x08

key: 0x4C

value: 0x01 ~ 0x40

(Just like brightness slider for RGB)

Run Start for CDW

dataType: 0x02

key: 0x4E

value: 0x01 ~ 0x0A

There are ten modes of running.

(Just like run start for RGB)

Run Stop for CDW

dataType: 0x02

key: 0x4F

value: 0x01

(Just like run stop for RGB)

Run Speed for CDW

dataType: 0x08

key: 0x22

value: 0x01 ~ 0x0A

There are ten levels of speed adjustment.

(Just like run speed for RGB)

CCT

dataType: 0x08

key: 0x36

value: 0x00 ~ 0x20

The value is from cold to warm.



Brightness Slider for CCT

dataType: 0x08

key: 0x33

value: 0x01 ~ 0x40

DIM

dataType: 0x08

key: 0x38

value: 0x01 ~ 0xFF



Start Save Scene

roomSelected: 0xFF

dataType: 0x02

key: 0x14

value: 0xB1

Save Scene

Can save up to 8 scenes.

When you want to save scene you need send Start Save Scene first then send Save Scene.

Send these two data intervals 0.5 second.

sceneIndex ≥ 0 and **sceneIndex** ≤ 7

roomSelected: 0xFF

dataType: 0x02

key: 0x0A + **sceneIndex**

value: 0x91 + (3 × **sceneIndex**)

for example sceneIndex is 5 than:

dataType: 0x02

key: 0x0A + 5 = 0x0F

value: 0x91 + (3 × 5) = 0xA0

Start Call Scene

roomSelected: 0xFF

dataType: 0x02

key: 0x14

value: 0xB0

Call Scene

When you want to call scene you need send Start Call Scene first then send Call Scene.

Send these two data intervals 0.05 second.

sceneIndex ≥ 0 and **sceneIndex** ≤ 7

roomSelected: 0xFF

dataType: 0x02

key: 0x0A + **sceneIndex**

value: 0x91 + (3 × **sceneIndex**)

for example sceneIndex is 5 than:

dataType: 0x02

key: 0x0A + 5 = 0x0F

value: 0x91 + (3 × 5) = 0xA0

(Just like save scene)

All ON

dataType: 0x02

key: 0x12
value: 0xAB

All OFF

dataType: 0x02
key: 0x12
value: 0xA9

Room ON

room 1:
roomSelected: 0b0001 (binary)
dataType: 0x02
key: 0x0A
value: 0x93

room 2:
roomSelected: 0b0010 (binary)
dataType: 0x02
key: 0x0A
value: 0x96

room 3:
roomSelected: 0b0100 (binary)
dataType: 0x02
key: 0x0A
value: 0x99

room 4:
roomSelected: 0b1000 (binary)
dataType: 0x02
key: 0x0A
value: 0x9C

room 5:
roomSelected: 0b00010000 (binary)
dataType: 0x02
key: 0x0A

value: 0x9F

room 6:

roomSelected: 0b00100000 (binary)

dataType: 0x02

key: 0x0A

value: 0xA2

room 7:

roomSelected: 0b01000000 (binary)

dataType: 0x02

key: 0x0A

value: 0xA5

room 8:

roomSelected: 0b10000000 (binary)

dataType: 0x02

key: 0x0A

value: 0xA8

Room OFF

room 1:

roomSelected: 0b0001 (binary)

dataType: 0x02

key: 0x0A

value: 0x92

room 2:

roomSelected: 0b0010 (binary)

dataType: 0x02

key: 0x0A

value: 0x95

room 3:

roomSelected: 0b0100 (binary)

dataType: 0x02

key: 0x0A

value: 0x98

room 4:

roomSelected: 0b1000 (binary)

dataType: 0x02

key: 0x0A

value: 0x9B

room 5:

roomSelected: 0b00010000 (binary)

dataType: 0x02

key: 0x0A

value: 0x9E

room 6:

roomSelected: 0b00100000 (binary)

dataType: 0x02

key: 0x0A

value: 0xA1

room 7:

roomSelected: 0b01000000 (binary)

dataType: 0x02

key: 0x0A

value: 0xA4

room 8:

roomSelected: 0b10000000 (binary)

dataType: 0x02

key: 0x0A

value: 0xA7

Learning

data1:

dataType: 0x09

key: 0x37

value: 0x01

data2:

(Room ON Protocol or other control protocol)


Send data1 first, after 0.5 second then send data2 more than three times.

This function is valid only ten seconds after the lamp is energized.

note:

Learning can only selected one room, just like 0b0001 or 0b0010 etc.

Protocol Formatter



1	2	3	4	5	6	7	8	9	10	11	12
0x55	device id			device type	room selected	data type	key	value	sum	0xAA	0xAA