Wireless Energy Controller V1.8

model number：EC-P11

INDEX

IMPORTANT ANNOUNCEMENT

** High voltage in the controller, operators must be with professional knowledge.

** The wire named “E-gnd” must be connected with the earth ground all the time.

FCC Warning Information

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

NOTE:

***WIFI Channel 12 & Channel 13 is not available in final product. The hardware is disabled inside.

***This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

—Reorient or relocate the receiving antenna.

—Increase the separation between the equipment and receiver.

—Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

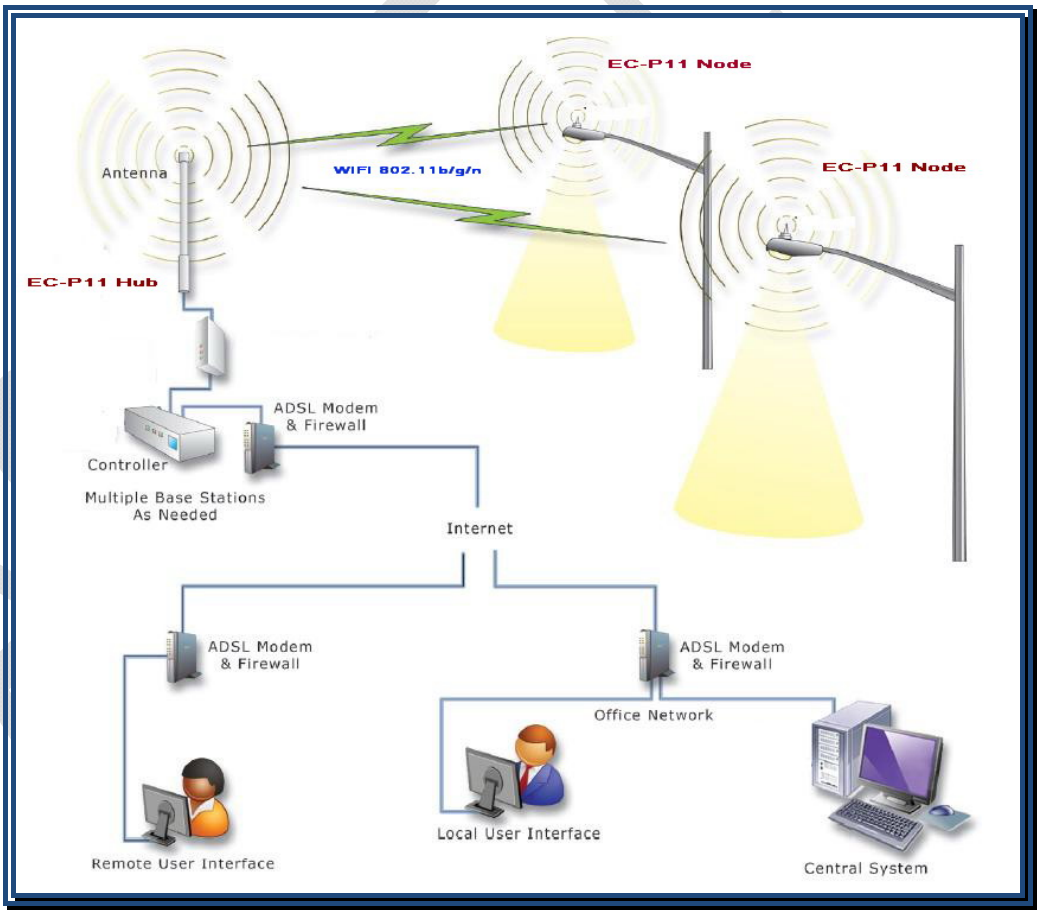
—Consult the dealer or an experienced radio/TV technician for help.

***Be sure to leave the human body more than 20cm when using this product.

***The country code is not supported by this product.

1 DESCRIPTION

The Wireless Energy Controller （model number：EC-P11）works as a controller with LED-fixture, and achieve remote control through WIFI wireless. The controller has 2* 0-10V signal for dimming usage, and current sensor is also integrated.



2 BENEFIT

---- Control and Schedule ON/OFF/DIM for individual fixture or groups of fixtures

---- Remote control and self programmable control mode

---- Customer defined continuous dimming

---- Adjust ON/OFF times and night trimming schedules

---- Monitor the work status & the Energy consumption of every fixture in every minute

---- Track burn hours to verify LED fixture life

---- Detect LED fixture malfunctions remotely

---- Works with any manufactures' outdoor LED lighting fixture

---- Reduce Energy Costs

---- Reduce Carbon Emission

---- Reduce Manpower Costs

---- Reduce Maintenance Costs

3 FEATURES

---- Each device can works in Node/Hub/Mix-mode

---- Wide AC input flexibility, 110V/AC, 277 V/AC, 480V/AC

---- AC switch inside（ Low loss in AC switch, typical 16 mΩ）

---- Max output power 1000W

---- 2* 0-10V Dimming signal for LED-fixture

---- AC Current Sensor integration

---- 1*10M/100M Ethernet port

---- WIFI 802.11b/g/n

---- Ublox GPS

---- 1*Button for customer

---- 1*USB device port (reserved)

---- 4*LED for work status indicated

---- 1* Interface for General Interrupt input (reserved)

---- 1* Interface for ADC signal input (reserved)

---- 12V/0.5A DC power output (reserved)

---- Easy to install & configure

4 HARDWARE CONFIGURATION

HARDWARE CONFIGURATION	
STANDARD LIST	**Remark**
WIFI & WIFI antenna port	
GPS & GPS antenna port	
1 * 0-10V Dimming signal output	
1 * AC switch inside	
OPTIONAL LIST	
2nd * 0-10V Dimming signal output	
2nd * AC switch	
Interface for Motion sensor input	**(reserved , not available now)**
Interface for Light sensor input	**(reserved , not available now)**
1*USB device port	**(reserved , not available now)**

5 ELECTRICAL SPECIFICATION

AC/AC PART

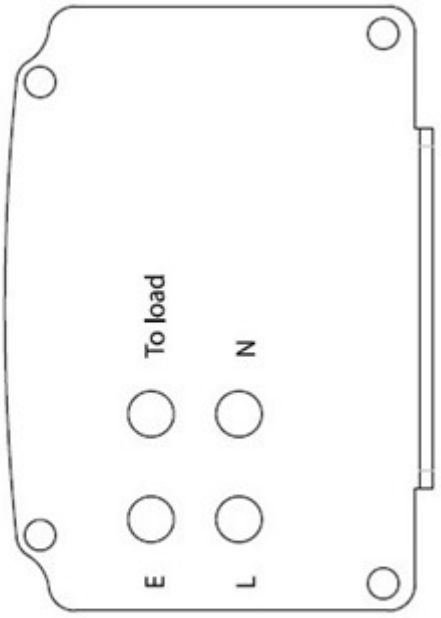
Absolute Maximum Ratings		
Items	**Unit**	**Value**
Max input value	V / AC	500
Max output current	A	2
Max output power	W	1000
Operating Temperature Range	℃	-40℃ to 85℃
Storage Temperature Range	℃	-65℃ to 145℃
Electrical Characteristics		

Items	Unit	Value
Nominal input value	V / AC	277
Input value range	V / AC	100V/AC to 500V/AC
AC Input Frequency range	Hz	50 / 60
Switch Output value	V / AC	same value as input
Accuracy of AC current sensor		±10% RMS (load current<150mA) ±5% RMS (load current>150mA)

GPS SPECIFICATION

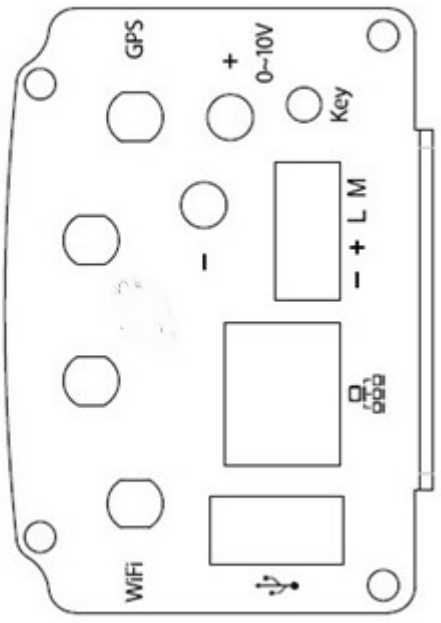
Items	SPEC
Receiver type	50 channel GPS L1 C/A
TTFF	Cold start-----29 s
	Warm start-----28 s
	Hot start-----2 s
Sensitivity	tracking: -160 dBm
	acquisition: -147 dBm
Horizontal position accuracy *	＜ 10 meters
* Test environment: Open sky, 24 hours static, Qty of satellite>6, all of satellite signal strength>-130dBm	

6 SIGNAL DEFINE



E —— GND
L —— AC Line input
N —— AC Neutral input
To load —— AC Line output to load (switch inside)

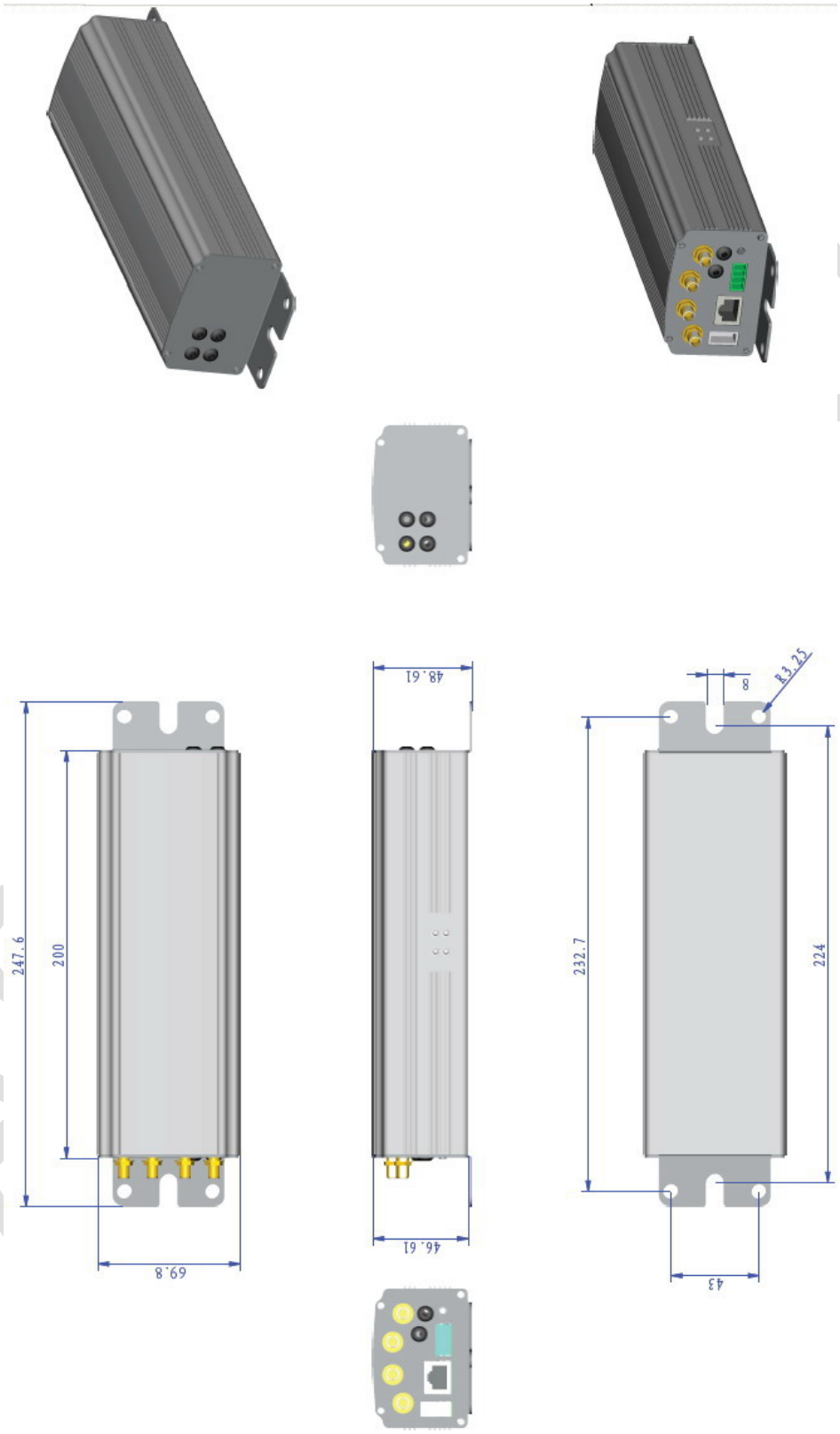
AC supply input range: 100V/AC ~~ 500V/AC
AC Line output(to load) range: the same value as input.



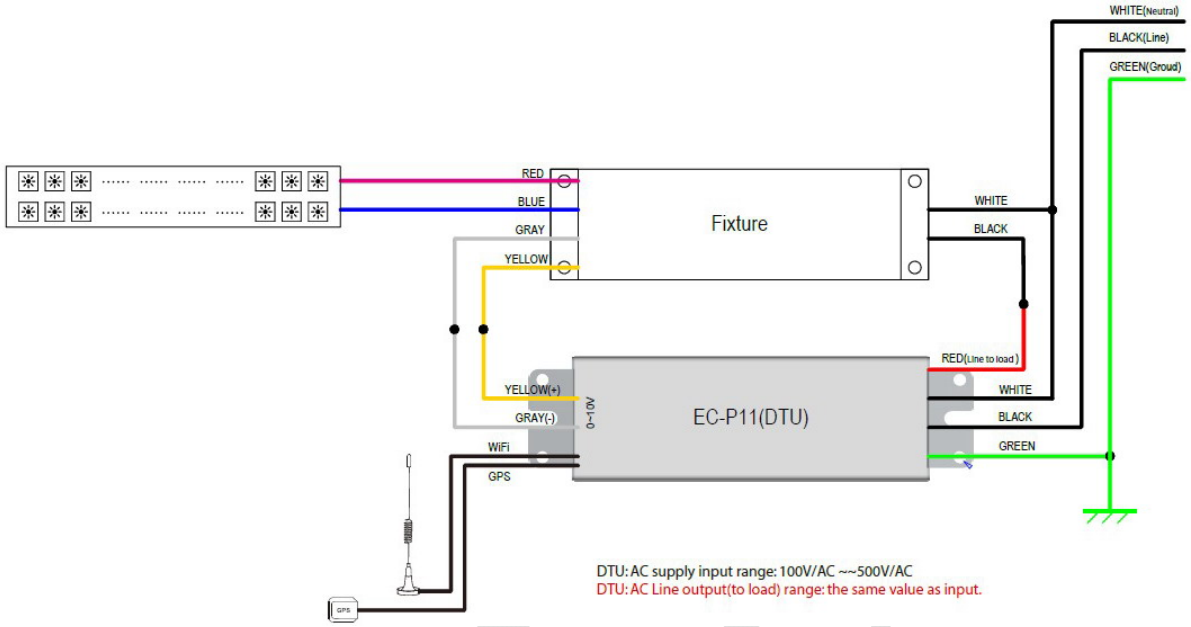
USB device interface (not available now)
Ethernet interface
4-port-connnector (not available now)
– DC signal ground
+ 12V/DC output
L Light sensor analog signal input (CAUTION: voltage of DC signal input range 0~3V)
M Motion sensor digital signal input (CAUTION: voltage of DC signal input range 0~3V)

0~10V / DC wire
– DC signal negative
+ DC signal positive

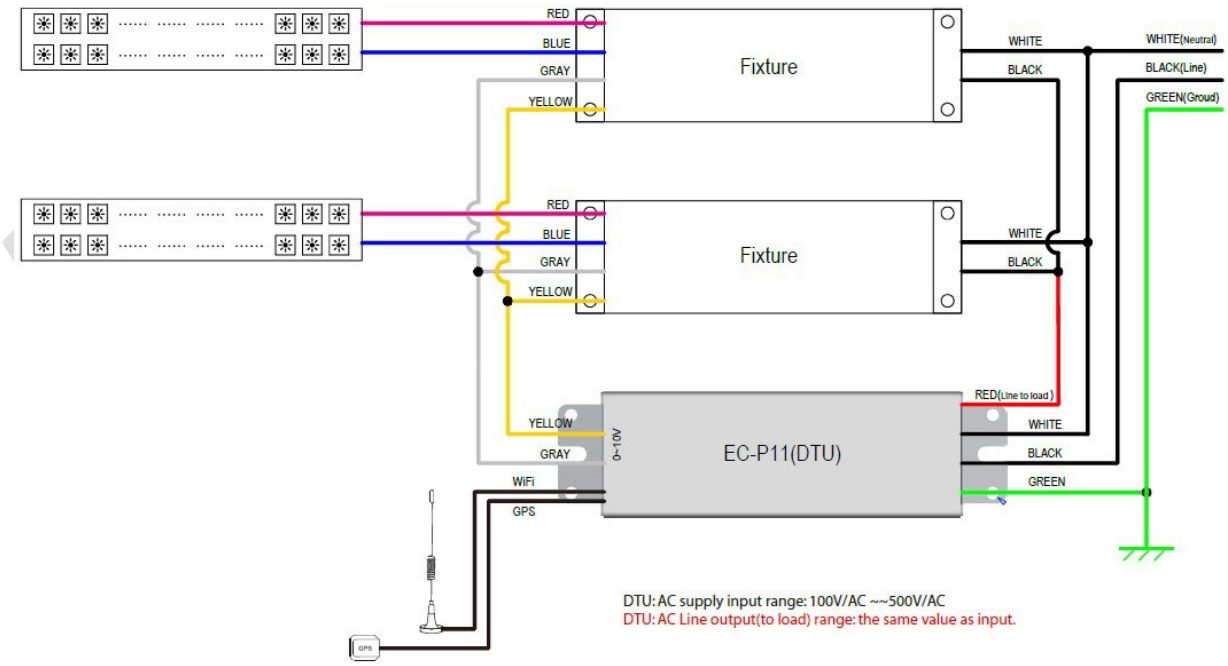
7 DIMENSION



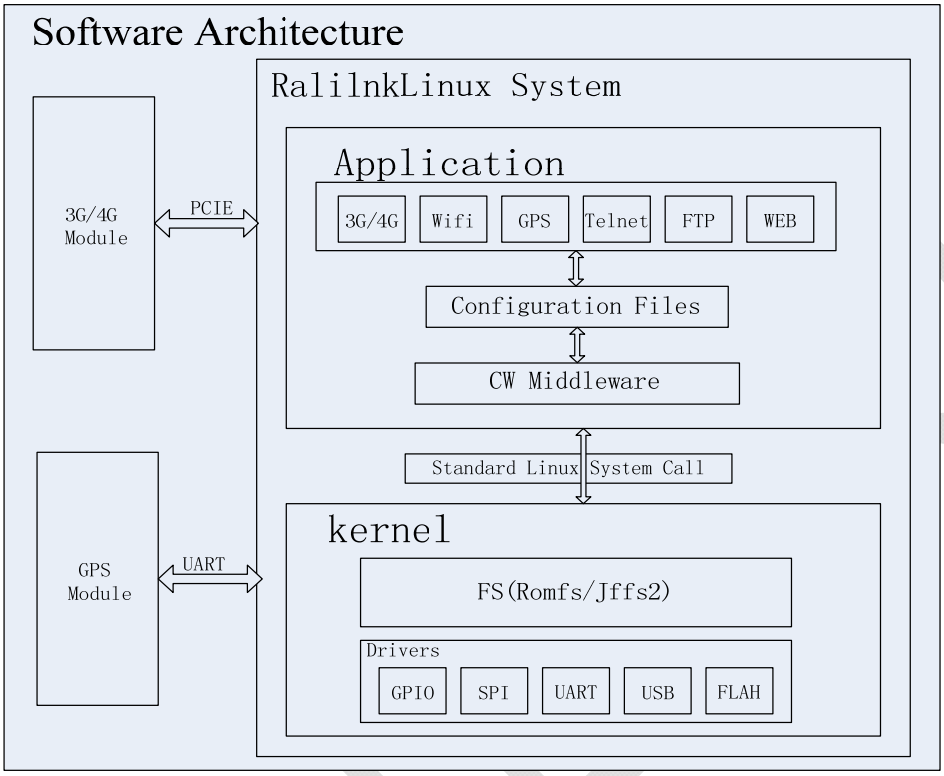
Case—drive 1xFixture



Case—drive 2xFixture



- **Software Architecture**



- **Important Function Using**

  - **system()**

    SYNOPSIS

    #include <stdlib.h>

    int system(const char *command);

  - **ioctl()**

    SYNOPSIS

    #include <sys/ioctl.h>

    int ioctl(int d, int request, ...);

- **UART Interface**

  - **Baud rate setting**

    #include <stdlib.h>

    system("gpio U 115200 "); // Set Baud rate 115200

  - **Read**

    int fd ;

    unsigned char buf[1024];

    fd = open("/dev/ttyS0", O_RDWR);

    read (fd, buf,1024);

    close(fd);

  - **Write**

    int fd ;

    unsigned char buf[1024];

    fd = open("/dev/ttyS0", O_RDWR |O_NONBLOCK | O_SYNC |O_NOCTTY);

```
write (fd, buf,1024);
close(fd);
```

- ## ADC Interface

  - ### Spi read

```
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char* argv[])
{
  unsigned char  ReadByte = 0;
  int fd = open ("/dev/softspi", O_RDONLY);
  if(fd < 0){
    return 1;
  }
  ioctl(fd,0,&ReadByte);
  printf("Read Byte: %d\n",ReadByte);
  close (fd);
  return 0;
}
```

  - ### Spi write

```
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char* argv[])
{
  unsigned char  wirteByte = 0;
  int fd = open ("/dev/softspi", O_RDONLY);
  if(fd < 0){
    return 1;
  }
  ioctl(fd,1,wirteByte);
  close (fd);
  return 0;
}
```

  - ### ADC read code

```
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char* argv[])
{
  unsigned int ADCvalue = 0;
  int fd = open ("/dev/softspi", O_RDONLY);
  if(fd < 0){
    return 1;
  }
  ioctl(fd,31,&ADCvalue);
  /* 'ADCvalue' value is ADC real value */
  printf("Read ADC Value: %d mV\n",ADCvalue);
  close (fd);
  return 0;
}
```

- ## DAC Interface

  - ### Source code

#include <stdlib.h>

Command: system("softspicmd 13 chn value");

Explain:

chn: 0~2 0 select DAC channel A2
1 select DAC channel A1
2 select DAC channel A1&A2

value: 0~4095 0~10V

Command: system("softspicmd 15");

Explain :

Executes a software clear (all CODE and DAC registers cleared to their default values)

Command: system("softspicmd 16");

Explain :

Executes a software reset (all CODE, DAC, and control registers returned to their default values)

Command: system("softspicmd 21 value ");

Explain :

This command will simultaneously writes data to all CODE registers while updating all DAC registers. It means channel A and channel B will simultaneously update to set value.
value: 0~4095 0~3.3V

- **USB Interface**

  - **Read**

    Source Code:
    int fd ;
    unsigned char buf[1024];
    fd = open("/dev/ ttyUSB0", O_RDWR);
    read (fd, buf,1024);
    close(fd);

  - **Write**

    Source Code:
    int fd ;
    unsigned char buf[1024];
    fd = open("/dev/ttyUSB0", O_RDWR);
    write (fd, buf,1024);
    close(fd);

- **GPIO Interface**

  - **Switcher**

    Swith S1 GPIO number is "GPIO9"
    Swith S2 GPIO number is "GPIO13"
    Using methods refer to GPIO write.
    • gpio w [num]: Set gpio high level. GPIO is output mode.
    [num] 0 to 46
    • gpio c [num]: Set gpio low level. GPIO is output mode.
    [num] 0 to 46
    GPS On/Off
    GPIO number is "GPIO60" High Level: Open GPS Power
    Current sensor On/Off
    To achieve in the future
    Light sensor On/Off
    To achieve in the future
    LED fixture On/Off
    Channel 1: GPIO number is "GPIO9" High Level: Open GPS Power
    Channel 2: GPIO number is "GPIO13" High Level: Open GPS Power
    Indicator LED
    Read LED: GPIO number is "GPIO41"
    Yellow LED: GPIO number is "GPIO42"
    Green1 LED: GPIO number is "GPIO43"
    Green2 LED: GPIO number is "GPIO44"

- **NV Interface**

  - **NV Write Interface**

    nvram_set(RT2860_NVRAM, "UserStingID", "UserStingContent");
    Instructions:
    RT2860_NVRAM: fixed value.
    "UserStingID": name of user sting.
    "UserStingContent": content of user sting.

  - **NV Read Interface**

    char * UserStingContent = nvram_get(RT2860_NVRAM, " UserStingID ");
    Instructions:
    RT2860_NVRAM: fixed value.
    "UserStingID": name of user sting.
    "UserStingContent": content of user sting.

- **GPS Interface**

```
//i2ccmd i fd

int open_fd(void)
{
      char nm[16];
int fd;

snprintf(nm, 16, "/dev/%s", I2C_DEV_NAME);
if ((fd = open(nm, O_RDONLY)) < 0) {
perror(nm);
                exit(fd);
      }
      return fd;
}

int ValGetNmeaData (unsigned char * data)
{
      int fd = -1;

      fd = open_fd();
if (fd < 0)
{
                return -1;
}
if (ioctl(fd, RT2880_I2C_READ_GPS_DATA, data) < 0)
{
                perror("ioctl");
                close(fd);
                return -1;
      }
      close(fd);

      return 0;
}
```

Data: input array[] for get data. The max size is 2048.

Example:
$GPRMC,,V,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPGGA,,,,,,0,00,99.99,,,,,,*48
$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30
$GPGSV,1,1,00*79
$GPGLL,,,,,,V,N*64

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ The end, thanks ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

2014/03