# Protocollo di base di configurazione

Questo protocollo server per identificare i devices presenti sulla local subnet. Utilizza un annuncio broadcast per segnalare la presenza del device. Il protocollo invia e riceve sulla porta UDP 5232

E' possibile inviare comandi ad un device usando questo protocollo.

Questo protocollo utilizza pacchetti UDP

I comandi possono essere inviati in broadcast o unicast, il device verifica la corrispondenza del MAC address presente nel pacchetto per determinare se deve eseguire il comando.

DEFINIZIONI:

Tutti i dati sono in trasmessi in ‘network order’

SERVER: applicazione o sistema da cui è possibile configurare le unità radio

DEVICE: le unità radio stesse

## Pacchetti inviati dal SERVER

|  |  |  |
| --- | --- | --- |
|  |  |  |
| DISCOVERY | 0xAF34 | Richede l'invio di un pacchetto di annuncio |
| SET\_IP | 0x5E01 | Permette di impostare indirizzo IP, Netmask e Gateway del device. I nuovi dati sono salvati in e2prom |
| START\_LOAD | 0x6A32 | Inizia il download del firmware da server TFTP indicato nel pacchetto |
| REBOOT\_REQ | 0xE2F6 | Forza il reboot del device (non gestito dal bootloader) |

I pacchetti possono essere unicast o broadcast.

Dove presente il campo MAC questo identifica il device a cui è indirizzato il comando oppure 00:00:00:00:00:00 per il comando DISCOVERY

Contenuto campi:

* Magic Number = 0xc0defeed
* Ver = Protocol version ( 0x10 for 1.0 )
* Id = device identifier ( 0x01 for bootloader, other TBD )
* Command = MUST be 0 for announce packet

## Pacchetto DISCOVERY

Il messaggio viene inviato con pacchetto broadcast per poter essere ricevuto anche da device non ancora configurati. I device che ricevono questo messaggio rispondono con un messaggio broadcast ANNOUNCE.

|++++++++|++++++++|++++++++|++++++++|

| magic number |

|++++++++|++++++++|++++++++|++++++++|

| VER | Id | 0xAF 0x34 |

|++++++++|++++++++|++++++++|++++++++|

| MAC address |

|++++++++|++++++++|++++++++|++++++++|

| MAC address | Padding |

|++++++++|++++++++|++++++++|++++++++|

typedef struct \_\_attribute\_\_((\_\_packed\_\_)) {

uint32\_t magic;

uint8\_t version;

uint8\_t id;

uint16\_t command;

uint8\_t mac[6];

} command\_t ;

## Pacchetto SET\_IP:

Il messaggio viene inviato con pacchetto broadcast e viene processato SOLO dal device il cui MAC address corrisponde a quello indicato all’interno del pacchetto. Non è prevista risposta, ma il device salva il nuovo indirizzo e si reinizializza.

|++++++++|++++++++|++++++++|++++++++|

| magic number |

|++++++++|++++++++|++++++++|++++++++|

| VER | Id | 0x5E 0x01 |

|++++++++|++++++++|++++++++|++++++++|

| MAC address |

|++++++++|++++++++|++++++++|++++++++|

| MAC address | Padding |

|++++++++|++++++++|++++++++|++++++++|

| new IP address |

|++++++++|++++++++|++++++++|++++++++|

| new Netmask |

|++++++++|++++++++|++++++++|++++++++|

| new Gateway address |

|++++++++|++++++++|++++++++|++++++++|

typedef struct \_\_attribute\_\_((\_\_packed\_\_)) {

uint32\_t magic;

uint8\_t version;

uint8\_t id;

uint16\_t command;

uint8\_t mac[6];

uint8\_t nu1[2];

uint32\_t new\_ip;

uint32\_t new\_mask;

uint32\_t new\_gw;

} set\_address\_t ;

## Pacchetti inviati dal DEVICE

Contenuto campi:

* Magic Number = 0xdeadbeef
* Ver = Protocol version ( 0x10 for 1.0 )
* Id = device identifier ( 0x01 for bootloader, other TBD )
* Command = MUST be 0 for announce packet
* MAC address = MAC of sending device
* IP address = IP of sending device
* Device Data = String for device identification ( depends on Id ) TBD

## Pacchetto ANNOUNCE:

Il messaggio viene inviato con pacchetto broadcast.

|++++++++|++++++++|++++++++|++++++++|

| magic number |

|++++++++|++++++++|++++++++|++++++++|

| VER | Id | command |

|++++++++|++++++++|++++++++|++++++++|

| MAC address |

|++++++++|++++++++|++++++++|++++++++|

| MAC address | Padding |

|++++++++|++++++++|++++++++|++++++++|

| Device IP address |

|++++++++|++++++++|++++++++|++++++++|

| Device data |

| |

| max 128 byte |

| NULL terminated |

| |

| |

| |

| |

|++++++++|++++++++|++++++++|++++++++|

typedef struct \_\_attribute\_\_((\_\_packed\_\_)) {

uint32\_t magic;

uint8\_t version;

uint8\_t id;

uint16\_t command;

uint8\_t mac[6];

uint8\_t nu1[2];

uint32\_t ip;

uint8\_t data[128];

} announce\_t ;

# Protocollo configurazione unità ROIP

Questo protocollo server per monitorare e configurare i device presenti in rete

Questo protocollo utilizza una connessione TCP con il device sulla porta 8032.

Una volta stabilita la connessione il SERVER

DEFINIZIONI:

Tutti i dati sono in trasmessi in ‘network order’

SERVER: applicazione o sistema da cui è possibile configurare le unità radio

DEVICE: le unità radio stesse

## Pacchetto COMMAND

Questo pacchetto viene inviato dal SERVER per eseguire un comando sul DEVICE

Contenuto campi:

* Ver = Protocol version ( 0x10 for 1.0 )
* Command = vedere tabella di seguito
* Data = dati specifici del comando

/\*\*

\* Elenco valori del campo <command>

\*/

enum

{

command\_status = 0,

command\_save = 1,

command\_set\_ip,

command\_set\_sysmode,

command\_set\_name,

command\_set\_in1\_gain,

command\_set\_in2\_gain,

command\_set\_out1\_gain,

command\_set\_out2\_gain,

command\_set\_remotes,

command\_set\_single\_remote,

} command\_packet\_commands;

/\*\*

\* struttura dati del messaggio

\*

\*/

typedef struct \_\_attribute\_\_((\_\_packed\_\_)) {

uint8\_t sync[16]; // 16 byte sync = "DST-ROIP-AXDDRFS"

uint8\_t version;

uint8\_t command;

} command\_packet\_t;

## Pacchetto STATUS

Questo pacchetto viene inviato dal DEVICE al SERVER in risposta al comando STATUS.

/\*\*

\* Structure of the status packet sent over TCP connection

\* All fields are in network order

\*/

typedef struct \_\_attribute\_\_((\_\_packed\_\_)) {

uint8\_t sync[16]; // 16 byte sync = "DST-ROIP-AXDDRFS"

uint8\_t version; // 1 for now

uint8\_t system\_mode; // from config system-mode

uint8\_t name[CONFIG\_NAME\_LEN]; // string to identify the unit

uint8\_t system\_status[16]; // string with state machine info

uint8\_t radio\_rx;

uint8\_t radio\_tx;

uint16\_t audio\_in1\_rms; // percentuale rispetto a sinusoide 1Vpp x10

uint16\_t audio\_in1\_peak; // percentuale rispetto a clipping x10

uint16\_t audio\_in2\_rms; // percentuale rispetto a sinusoide 1Vpp x10

uint16\_t audio\_in2\_peak; // percentuale rispetto a clipping x10

uint16\_t audio\_out1\_rms; // percentuale rispetto a sinusoide 1Vpp x10

uint16\_t audio\_out1\_peak; // percentuale rispetto a clipping x10

uint16\_t audio\_out2\_rms; // percentuale rispetto a sinusoide 1Vpp x10

uint16\_t audio\_out2\_peak; // percentuale rispetto a clipping x10

uint8\_t audio\_in1\_gain;

uint8\_t audio\_in2\_gain;

uint8\_t audio\_out1\_gain;

uint8\_t audio\_out2\_gain;

uint32\_t remotes[MAX\_REMOTE\_ADDRESS];

incoming\_stream\_status\_t remoteStatus[MAX\_REMOTE\_ADDRESS];

} status\_packet\_t ;

Valori dei campi dati:

system\_mode: 0 = MASTER, 1 = SLAVE

name: stringa con nome device, max 16 byte

system\_status : non usare

radio\_tx: 0 = tx inattivo, 1 = tx attivo

radio\_rx: 0 = rx inattivo, 1 = rx attivo

audio\_inX\_rms:

audio\_inX\_peak:

audio\_outX\_rms:

audio\_outX\_peak:

audio\_inX\_gain:

audio\_outX\_gain:

remotes[]: array di 4 elementi contenenti gli indirizzi delle unità remote connesse

remoteStatus[]: array di 4 elementi <incoming\_stream\_status\_t> che rappresentano lo stato di connessione dei corrispondenti device remoti.

typedef enum {

RemoteNotPresent = 0,

RemoteConnected,

RemoteLostConnection,

} incoming\_stream\_status\_t;