September 9, 2019

Contents

1 Configuración del puerto serial para utilizar el modulo

Parámetro	Valor
Baud Rate	115200
Data bits	8
Parity	None
Flow Control	None
End of line	\n (LF)

2 Descripción del protocolo

Al enviar un comando al módulo se verifica primero si se recibió un comando valido. En caso de que no lo sea, la respuesta del módulo es "C", y el módulo espera de vuelta un comando. Una vez que se validó el comando, se verifica que se cuente con la cantidad de parámetros necesarios. En caso de que no sea suficiente, la respuesta del módulo es "P", y el módulo espera de vuelta un comando. Si el comando es válido y se pasaron los parámetros necesarios para ejecutar el comando, se ejecuta el comando y se provee la respuesta dependiendo del resultado. Al final de cada respuesta del módulo le sigue \n , por ejemplo: 0\n

2.1 Comandos

Sintaxis: `NOMBRE_COMANDO,param1,param2, . . ,paramN\n`

- Los nombres de los comandos deben estar en letras mayúsculas, de lo contrario se dará un error de que no se encuentra el comando.
- Comandos que contienen parámetros deben incluir una marca de coma (,) como delimitador entre ellos.
- La definición de cada comando indica cuantos parámetros necesita, en caso de que falte algún parámetro, se dará un error.

2.2 Respuesta de los comandos

Las respuestas de los comandos pueden ser uno de los siguientes casos:

- Comando que retorna con éxito: `0`
- Comando que retorna con error: Indicado con un numero positivo, mayor que cero.
- Comando que retorna con parámetro: `0,parámetro`
- Comando que retorna con parámetro y datos: `0,parámetro,datos`

3 Comandos Básicos

3.1 MIS

Comando utilizado para verificar que el módulo se encuentra funcionando correctamente y esta listo para recibir comandos.

- **Sintaxis**
  `MIS\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  `0\n`

3.2 MRS

Comando que reinicia inmediatamente el módulo. Al iniciar de vuelta el módulo, este envía por el puerto serial una serie de caracteres sin importancia, luego de esto se recibe el carácter `R\n`, el cual indica que el módulo esta listo.

- **Sintaxis**
  `MRS\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  `R\n`

3.3 MVI

Comando que retorna información acerca de la versión actual del firmware que se esta ejecutando. También informa acerca de la versión del Arduino Core utilizado para programar el firmware.

- **Sintaxis**
  `MVI\n`
- **Parámetros**
  Ninguno.

- **Respuesta**
  `0,Firmware:<numero_version>,ArduinoCore:<version>\n`

3.4 MDS

Comando que configura el modo de bajo consumo *Deep-sleep* para el módulo.

- **Sintaxis**
  `MDS,tiempo_dormir,modo_rf\n`
- **Parámetros**
  - `tiempo_dormir`
    El tiempo medido en *us* que el dispositivo estara en deep-sleep.
  - `modo_rf`
    Parámetro que determina el comportamiento de la calibración RF luego de despertarse.
    * **0** , Configuración RF por defecto.
    * **1** , Efectuar calibración RF.
    * **2** , No se realiza calibración RF, esto reduce el consumo de corriente .
    * **3** , Desactiva el sistema de RF al despertarse. Esta opción permite el menor consumo posible de corriente, sin embargo, no se pueden enviar ni recibir datos al despertarse.

3.5 MUC

Comando utilizado para modificar la velocidad de transmisión del periférico UART, utilizado por el módulo para comunicarse con el micro-controlador externo.

- **Sintaxis**
  `MUC,<velocidad>\n`
- **Parámetros**
  - `<velocidad>`
    Velocidad de transmisión deseada. El rango permitido para este parámetro va desde 9600 a 921600.
- **Respuesta**
  - `0\n`
    El cambio de velocidad se realizó con éxito. Es necesario esperar 5 ms para enviar el siguiente comando utilizando la nueva velocidad.
  - `1\n`
    Error, el parámetro `<velocidad>` se encuentra fuera de rango.

3.6 MRP

Comando que configura la potencia de transmisión de la antena de radio frecuencia del módulo.

- **Sintaxis**
  `MRP,potencia_dbm\n`
- **Parámetros**
  - `potencia_dbm`
    Potencia de transmision a ser utilizada, en dBm. El rango de valores perimitido va desde 0 a 20,5.
- **Respuesta**
  - `0\n`
    La configuración fue aplicada con éxito.
  - `1\n`
    El parámetro `potencia_dbm` esta fuera de rango.

3.7 MFH

Comando que retorna la cantidad de Bytes disponibles de la memoria RAM.

- **Sintaxis**
  `MFH\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  `0,cantidad_bytes_disponibles\n`

4 Comandos WiFi

4.1 WFM

Comando utilizado para establecer el modo de funcionamiento WiFi del modulo.

- **Sintaxis**
  `WFM,modo_wifi\n`
- **Parámetros**
  - `modo_wifi`
    Parámetro que determina cual modo sera utilizado. Valores permitidos del 0 al 3.
    - **0** , WiFi apagado.
    - **1** , modo estación (STA).
    - **2** , modo punto de acceso (AP).
    - **3** , modo estación + punto de acceso (STA + AP).

- **Respuesta**
  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, el parámetro `modo_wifi` se encuentra fuera de rango.
  - `2\n`
    Error, no se pudo establecer la configuración.

4.2 WFC

Comando utilizado para conectar el módulo a un punto de acceso (AP, por sus siglas en ingles).

- **Sintaxis**
  `WFC,<ssid>,<contraseña>\n`
- **Parámetros**
  - `<ssid>`
    Nombre del punto de acceso al cual se desea conectar el módulo.
  - `<contraseña>`
    Contraseña del punto de acceso al cual se desea conectar el módulo.
- **Respuesta**
  - `0\n`
    Conexión exitosa.
  - `1\n`
    Error, no se pudo establecer la conexión al punto de acceso.
  - `2\n`
    Error, se alcanzo el tiempo de espera máximo (20 segundos) sin poder establecer la conexión.
  - `3\n`
    Error, contraseña incorrecta.
  - `4\n`
    Error, no se encuentra el punto de acceso.

4.3 WFS

Comando utilizado para escanear los puntos de acceso que se encuentran al alcance del modulo.

- **Sintaxis**
  `WFS\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  `0,ssid_1;rssi_1,ssid_2;rssi_2,ssid_N;rssi_N\n`

- **Respuesta**
  `1\n`
  No se encontró ningún punto de acceso.

- **Ejemplo**
  Comando: `WFS\n`
  Respuesta: `0,DEI-UCA;-81,DICIA-UCA;-70,LED-UCA;-85\n`

4.4 WFD

Comando utilizado para desconectar el modulo del punto de acceso al cual se encuentra conectado actualmente.

- **Sintaxis**
  `WFD,wifi_off\n`

- **Parámetros**

  - `wifi_off`
    Parámetro que determina si se apagara la radio WiFi luego de desconectarse. Valores permitidos: 0 o 1.

    * **0** , la radio WiFi sigue activada.
    * **1** , se desactiva la radio WiFi.

- **Respuesta**

  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, el parámetro `wifi_off` se encuentra fuera de rango.
  - `2\n`
    Error, no se pudo aplicar la configuración.

4.5 WFA

Comando utilizado para configurar el módulo como un punto de acceso (AP, por sus siglas en ingles). El modo de autenticación es WPA2-PSK.

- **Sintaxis**
  `WFA,ssid,contraseña,canal,ssid_oculto,max_con\n`

- **Parámetros**

  - `ssid`
    Nombre del punto de acceso, longitud máxima de 63 caracteres.
  - `contraseña`
    Contraseña del punto de acceso, longitud minima de 8 caracteres.
  - `canal`
    Numero del canal WiFi que utilizara el punto de acceso. Valores permitidos del 1 al 13.

- `ssid_oculto`
  Parámetro que determina si el SSID se mostrará de manera publica. Para publicar, el valor es 0, para ocultar 1.
- `max_con`
  Numero máximo de conexiones simultaneas que permite atender el punto de acceso. Valores permitidos del 1 al 4.

- **Respuesta**
  - `0\n`
    El punto de acceso fue creado correctamente.
  - `1\n`
    Error, el numero de canal esta fuera de rango.
  - `2\n`
    Error, el parámetro `ssid_oculto` esta fuera de rango.
  - `3\n`
    Error, el parámetro `max_con` esta fuera de rango.
  - `4\n`
    Error, no se pudo crear el punto de acceso.

4.6 WSI

Comando para obtener información acerca de los dispositivos conectados a la interfaz del punto de acceso (softAP) del módulo.

- **Sintaxis**
  `WSI\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  - `0,clientes_conectados,ip_cliente1;mac_cliente1,..,ip_clienteN;mac_clienteN\n`
    Comando ejecutado con éxito. Se muestra en primer lugar la cantidad de clientes conectados, luego la dirección IP y MAC de cada cliente.

4.7 WCF

Comando utilizado para configurar de forma manual los parámetros de la interfaz de red de la estación, desactivando la asignación por DHCP.

- **Sintaxis**
  `WCF,ip,dns,gateway,subnet\n`
- **Parámetros**
  Todos los parámetros son en formato de cadena de caracteres.
  - `ip`
    Dirección IP a ser asignada al modulo.

- `dns`
  Dirección del servidor DNS.
- `gateway`
  Dirección de la puerta de enlace.
- `subnet`
  Dirección de la mascara de la red.

- **Respuesta**
  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, dirección IP invalida.
  - `2\n`
    Error, dirección DNS invalida.
  - `3\n`
    Error, dirección Gateway invalida.
  - `4\n`
    Error, dirección Subnet invalida.
  - `5\n`
    Error, no se pudo establecer la configuración deseada.

4.8 WAC

Comando utilizado para configurar de forma manual los parámetros de la interfaz de red del punto de acceso (softAP).

- **Sintaxis**
  `WAC,ip,gateway,subnet\n`
- **Parámetros**
  Todos los parámetros son en formato de cadena de caracteres.
  - `ip`
    Dirección IP a ser asignada al modulo.
  - `gateway`
    Dirección de la puerta de enlace.
  - `subnet`
    Dirección de la mascara de la red.

- **Respuesta**
  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, dirección IP invalida.
  - `2\n`
    Error, dirección Gateway invalida.

- `3\n`
  Error, dirección Subnet invalida.
- `4\n`
  Error, no se pudo aplicar la configuración.

4.9 WMA

Comando utilizado para configurar la dirección de MAC del módulo.

- **Sintaxis**
  `WMA,interfaz,direccion_mac\n`
- **Parámetros**
  - `interfaz`
    Selecciona para cual interfaz se configurara la direccion MAC.
    * **0** , interfaz de estación (STA) .
    * **1** , interfaz de punto de acceso (AP).
- **Respuesta**
  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, el parámetro `interfaz` esta fuera de rango.
  - `2\n`
    Error, el parámetro `direccion_mac` no tiene la longitud correcta.
  - `3\n`
    Error, no se pudo establecer la configuración.

4.10 WSC

Comando utilizado para iniciar el aprovisionamiento de las credenciales del punto de acceso al cual se intentara conectar, utilizando el protocolo Smart-Config. Al utilizar este comando, el único comando que puede ser llamado después es el comando WSD.

- **Sintaxis**
  `WSC\n`
- **Parámetros**
  - Ninguno
- **Respuesta**
  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, la configuración no pudo ser aplicada.

4.11 WSS

Comando utilizado para detener el aprovisionamiento de las credenciales del punto de acceso.

- **Sintaxis**
  `WSS\n`
- **Parámetros**
  - Ninguno
- **Respuesta**
  - `0\n`
    El aprovisionamiento fue detenido con éxito.
  - `1\n`
    Error, no fue posible detener el aprovisionamiento.

4.12 WSD

Comando utilizado para verificar el estado de la conexión luego de utilizar el comando WSC.

- **Sintaxis**
  `WSD\n`
- **Parámetros**
  - Ninguno
- **Respuesta**
  - `0\n`
    Credenciales recibidas con éxito.
  - `1\n`
    Error, aun no se recibió ninguna credencial.

4.13 WAD

Comando utilizado para desactivar el punto de acceso del modulo.

- **Sintaxis**
  `WAD,wifi_off\n`
- **Parámetros**
  - `wifi_off`
    Parámetro que determina si se apagara la radio WiFi luego de desconectarse. Valores permitidos: 0 o 1.
    * **0** , la radio WiFi sigue activada.
    * **1** , se desactiva la radio WiFi.
- **Respuesta**

  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, el parámetro `wifi_off` se encuentra fuera de rango.
  - `2\n`
    Error, no se pudo aplicar la configuración.

- **Ejemplo**
  Comando: `WAD,0\n`
  Respuesta: `0\n`

4.14 WSN

Comando utilizado para establecer el nombre del módulo con el cual se registrará al servidor DHCP.

- **Sintaxis**
  `WSN,nombre\n`

- **Parámetros**

  - `nombre`
    Nombre a ser enviado. Longitud máxima de 32 caracteres.

- **Respuesta**

  - `0\n`
    Configuración exitosa.
  - `1\n`
    Error, la configuración no pudo ser aplicada.

4.15 WRI

Comando utilizado para obtener el RSSI (en dB) del punto de acceso al cual se encuentra actualmente conectado el modulo.

- **Sintaxis**
  `WRI\n`

- **Parámetros**
  Ninguno.

- **Respuesta**

  - `0,rssi\n`
    Comando ejecutado con éxito, se muestra el RSSI en decibelios.
  - `1\n`
    Error al obtener el RSSI.

- **Ejemplo**
  Comando: `WRI\n`
  Respuesta: `0,-80\n`

4.16 WID

Comando para obtener el SSID de la estación a la que se encuentra conectado actualmente el modulo.

- **Sintaxis**
  `WID\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  - `0,ssid\n`
    Comando ejecutado con éxito, se muestra el ssid.
  - `1\n`
    Error, el modulo no se encuentra conectado a ninguna red.
- **Ejemplo**
  Comando: `WID\n`
  Respuesta: `0,LED-UCA\n`

5 Comandos TCP/UDP

5.1 SOI

Comando que retorna información acerca de los sockets utilizados.

- **Sintaxis**
  `SOI,socket\n`
- **Parámetros**
  - `socket`
    Parámetro utilizado para identificar las conexiones. Los valores permitidos para este parámetro van de 0 a 3.
- **Respuesta**
  - `0,protocolo,ip_remota,puerto_remoto,puerto_local,tipo\n`
    Se retorna la información del socket. En primer lugar, se informa el tipo de protocolo utilizado para el socket: TCP o UDP. Luego, se provee la direccion IP y numero de puerto utilizado en el otro extremo del socket, ademas del puerto local utilizado para este socket. Por ultimo, se informa el tipo de socket, que puede ser cliente o servidor. Esto indica si la conexion fue establecida en modo cliente (1) o si el socket fue creado tras aceptar a un cliente en el servidor (0).
  - `1\n`
    Error, el parámetro `socket` esta fuera de rango.
- **Ejemplo**
  Comando: `SOI,0\n`
  Respuesta: `0,TCP,192.168.0.165,49531,15000,1\n`

5.2 IDN

Comando utilizado para resolver la dirección de IP a partir de un nombre de host .

- **Sintaxis**
  `IDN,nombre_servidor\n`
- **Parámetros**
  - `nombre_servidor`
    Nombre del servidor del cual se quiere resolver la direccion IP.
- **Respuesta**
  - `0,direccion_ip_servidor\n`
    Se retorna la direccion IP del servidor.
  - `1\n`
    Error, no se pudo resolver la direccion.
- **Ejemplo**
  Comando: `SOI,0\n`
  Respuesta: `0,TCP,192.168.0.165,49531,15000,1\n`

5.3 CCS

Comando utilizado para establecer una conexión TCP o UDP en modo cliente a un servidor remoto.

- **Sintaxis**
  `CCS,protocolo,ip,puerto\n`
- **Parámetros**
  - `protocolo`
    Parámetro para definir que protocolo se utilizara en la comunicación, puede ser TCP o UDP.
  - `ip`
    Dirección IP del servidor al cual se quiere establecer la conexión, como también puede ser un nombre de host.
  - `puerto`
    Puerto del servidor. Puede tener un valor máximo de 65535.
- **Respuesta**
  - `0,socket\n`
    Se estableció exitosamente la conexión al servidor. Se retorna un numero de `socket` que sera utilizado para otros comandos para identificar la conexión. Los valores permitidos para este numero van de 0 a 3.
  - `1\n`
    Error, no hay una conexión WiFi activa.

- `2\n`
  Error, el parámetro `puerto` esta fuera de rango.
- `3\n`
  Error, no hay recursos disponibles(socket) para establecer la conexión.
- `4\n`
  Error, no se pudo establecer la conexión al servidor.
- `5\n`
  Error, el parámetro `protocolo` es invalido.

- **Ejemplo**
  Comando: `CCS,TCP,192.168.0.35,9500\n`
  Respuesta: `0,0\n`

5.4 SOW

Comando utilizado para enviar datos a través de una conexión TCP. Para utilizar este comando, es necesario primero utilizar el comando CCS, para establecer la conexión a un servidor, y/o el comando SAC, que acepta un cliente que intenta conectarse a un servidor en el modulo.

- **Sintaxis**
  `SOW,socket,cantidad_Bytes,datos\n`

- **Parámetros**
  - `socket`
    Parámetro utilizado para identificar las conexiones. Los valores permitidos para este parámetro van de 0 a 3.
  - `cantidad_Bytes`
    Cantidad de Bytes a ser enviados. El valor máximo permitido para este parámetro es 1460.
  - `datos`
    Es la cadena de datos a ser enviados. La longitud de esta cadena debe ser igual al del parámetro `cantidad_Bytes`, en caso de que no sean iguales, los datos no serán enviados.

- **Respuesta**
  - `0\n`
    Los datos fueron enviados correctamente.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket` se encuentra fuera de rango.
  - `3\n`
    Error, el parámetro `cantidad_Bytes` se encuentra fuera de rango.
  - `4\n`
    Error, el parámetro `socket` no utiliza el protocolo TCP.

- `5\n`
  Error, el parámetro `socket` no tiene una conexión activa.
- `6\n`
  Error, los datos no fueron enviados.

5.5 SOR

Comando utilizado para recibir datos a través de una conexión TCP. Para utilizar este comando, es necesario primero utilizar el comando CCS, para establecer la conexión a un servidor, y/o el comando SAC, que acepta un cliente que intenta conectarse a un servidor en el modulo.

- **Sintaxis**
  `SOR,socket\n`
- **Parámetros**
  - `socket`
    Parámetro utilizado para identificar las conexiones. Los valores permitidos para este parámetro van de 0 a 3.
- **Respuesta**
  - `0,cantidad_Bytes,datos\n`
    Los datos fueron recibidos correctamente.
    * `cantidad_Bytes`
      Cantidad de Bytes que se recibieron.
    * `datos`
      La cadena de datos que fue recibida.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket` se encuentra fuera de rango.
  - `3\n`
    Error, el parámetro `socket` no tiene una conexión activa.
  - `4\n`
    Error, el parámetro `socket` no utiliza el protocolo TCP.

5.6 SOC

Comando utilizado para cerrar las conexiones activas.

- **Sintaxis**
  `SOC,socket\n`
- **Parámetros**
  - `socket`
    Parámetro utilizado para identificar las conexiones. Los valores permitidos para este parámetro van de 0 a 3.

- **Respuesta**
  - `0\n`
    La conexión fue cerrada con éxito.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket` se encuentra fuera de rango.
  - `3\n`
    Error, el parámetro `socket` no tiene una conexión activa.

5.7 WFI

Comando utilizado para obtener la dirección MAC e IP local de la interfaz de red de la estación, además de la mascara de subred, dirección de la puerta de enlace y servidor DNS1.

- **Sintaxis**
  `WFI\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  `0,mac,ip,subnet,gateway,dns\n`
- **Ejemplo**
  Comando: `WFI\n`
  Respuesta: `0,0A:22,192.168.0.12,255.255.255.255,192.168.0.1,156.13.22.2\n`

5.8 WAI

Comando utilizado para obtener la dirección MAC e IP de la interfaz de red del punto de acceso (softAP).

- **Sintaxis**
  `WAI\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  `0,ip,mac\n`
- **Ejemplo**
  Comando: `WAI\n`
  Respuesta: `0,0A:22,192.168.0.12,255.255.255.255,192.168.0.1,156.13.22.2\n`

5.9 SLC

Comando utilizado para crear un servidor TCP en el módulo. Pueden trabajar en simultaneo 4 servidores, como máximo.

- **Sintaxis**
  `SLC,puerto,cantidad_clientes\n`
- **Parámetros**
  - `puerto`
    Puerto a ser utilizado por el servidor. Puede tener un valor máximo de 65535.
  - `cantidad_clientes`
    Especifica la cantidad de conexiones simultaneas que puede aceptar el servidor. Los valores permitidos para ese parámetro va desde 1 hasta 4.
- **Respuesta**
  - `0,socket_pasivo\n`
    El servidor fue creado exitosamente. Se retorna un numero de `socket_pasivo` que sera utilizado para identificar al servidor. El único comando que utiliza este valor como parámetro es el comando SAC. Los valores permitidos para este numero van de 0 a 3.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `puerto` se encuentra fuera de rango.
  - `3\n`
    Error, el parámetro `cantidad_clientes` se encuentra fuera de rango.

5.10 SAC

Comando utilizado para aceptar clientes que desean conectarse a un servidor TCP del modulo. Para utilizar este comando en primer lugar se debe ejecutar el comando SLC, ya que este comando retorna un valor que utiliza el comando SAC como parámetro.

- **Sintaxis**
  `SAC,socket_pasivo\n`
- **Parámetros**
  - `socket_pasivo`
    Parámetro utilizado para identificar de cual servidor se deben aceptar los clientes. Para obtener este parámetro, se debe almacenar el valor de retorno del comando SLC. Los valores permitidos para este parámetro van de 0 a 3.
- **Respuesta**

  - `0,socket\n`
    El cliente fue aceptado con éxito al servidor. Se retorna un numero `socket` de manera tal a identificar al cliente y poder intercambiar datos. Los valores permitidos para este numero van de 0 a 3.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket` se encuentra fuera de rango.
  - `3\n`
    Error, no hay recursos disponibles para aceptar el cliente, se rechaza la conexión.
  - `4\n`
    El servidor no tiene clientes que quieran conectarse.
  - `5\n`
    El servidor `socket_pasivo` no se encuentra activo.
  - `6\n`
    Ya se alcanzo el numero máximo de conexiones simultaneas permitidas para este servidor. Se rechaza el cliente.

5.11 SCC

Comando utilizado para desactivar un servidor TCP.

- **Sintaxis**
  `SCC,socket_pasivo\n`
- **Parámetros**
  - `socket_pasivo`
    Parámetro para indicar cual es el servidor que se desactivara.
- **Respuesta**
  - `0\n`
    El servidor fue desactivado exitosamente.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket_pasivo` se encuentra fuera de rango.

5.12 SVU

Comando utilizado crear un servidor para recibir paquetes UDP en el puerto especificado.

- **Sintaxis**
  `SVU,puerto\n`
- **Parámetros**

- `puerto`
  Puerto a ser utilizado por el servidor. Puede tener un valor máximo de 65535.

- **Respuesta**
  - `0,socket\n`
    El servidor fue creado exitosamente. Se retorna un numero de `socket` que sera utilizado para identificar al servidor. Los valores permitidos para este numero van de 0 a 3.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `puerto` se encuentra fuera de rango.
  - `3\n`
    Error, no hay recursos disponibles para ejecutar el comando.
  - `4\n`
    Error, no fue posible establecer la recepción de paquetes.

5.13 SDU

Comando utilizado para enviar paquetes UDP.

- **Sintaxis**
  `SDU,socket,cantidad_Bytes,datos\n`

- **Parámetros**
  - `socket`
    Parámetro utilizado para identificar las conexiones. Los valores permitidos para este parámetro van de 0 a 3.
  - `cantidad_Bytes`
    Cantidad de Bytes a ser enviados. El valor máximo permitido para este parámetro es 1460.
  - `datos`
    Es la cadena de datos a ser enviados. La longitud de esta cadena debe ser igual al del parámetro `cantidad_Bytes`, en caso de que no sean iguales, los datos no serán enviados.

- **Respuesta**
  - `0,socket\n`
    Los datos fueron enviados correctamente.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket` se encuentra fuera de rango.
  - `3\n`
    Error, el parámetro `cantidad_Bytes` se encuentra fuera de rango.

- `4\n`
  Error, los datos no fueron enviados.
- `5\n`
  Error, el parámetro `socket` fue configurado para ser utilizado con el protocolo TCP.

5.14 RVU

Comando utilizado para recibir datos a través de una conexión UDP. Para utilizar este comando, es necesario primero utilizar el comando SVU, para saber el puerto por el cual se reciben los paquetes.

- **Sintaxis**
  `RVU,socket\n`
- **Parámetros**
  - `socket`
    Parámetro utilizado para identificar las conexiones. Los valores permitidos para este parámetro van de 0 a 3.
- **Respuesta**
  - `0,cantidad_Bytes,datos\n`
    Los datos fueron recibidos correctamente.
    * `cantidad_Bytes`
      Cantidad de Bytes que se recibieron.
    * `datos`
      La cadena de datos que fue recibida.
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, el parámetro `socket` se encuentra fuera de rango.

5.15 STC

Comando utilizado para configurar el servidor SNTP del módulo.

- **Sintaxis**
  `STC,direccion_servidor_sntp,offset_tiempo\n`
- **Parámetros**
  - `direccion_servidor_sntp`
    Direccion del servidor SNTP a ser utilizado.
  - `offset_tiempo`
    Offset de tiempo para configurar la operacion del servidor.
- **Respuesta**
  - `0\n`
    Servidor SNTP configurado correctamente.
  - `1\n`
    Error, no hay una conexión WiFi activa.

5.16 STG

Comando utilizado obtener el tiempo actual del servidor SNTP configurado previamente utilizando el comando STC.

- **Sintaxis**
  `STG\n`
- **Parámetros**
  Ninguno.
- **Respuesta**
  - `0,tiempo\n`
    Informacion sobre el tiempo obtenido correctamente. Se retorna la informacion como una cadena `tiempo`, con el formato `horas:minutos:segundos`
  - `1\n`
    Error, no hay una conexión WiFi activa.
  - `2\n`
    Error, no se pudo obtener la información.