Manual Raspberry-Pi para SRT

Configuración del Router

IP: externo

IP interno: 192.168.3.1

Username: admin

Password: admin123

# Configuracion inicial Raspberry-Pi

Configurar de acuerdo a las instrucciones de usuario del manual de la R-Pi.

Configurar interfaz de red usando direccion IP fija

# Conexion SSH

El usuario por default es “pi” y la clave es “raspberry”

“ssh [pi@192.168.3.100](mailto:pi@192.168.3.100)”

MAC: b8:27:eb:88:fc:ff

# Crear usuario SRT con permiso para usar el Puerto serial

“sudo adduser srt”

“sudo adduser srt  dialout”

# Actualizar apt-get

“sudo apt-get update && sudo apt-get upgrade”

# Instalar python-dev

“sudo apt-get install python-dev”

# Instalar py-ephem

sudo apt-get install python-pip

sudo pip install pyephem

~~gunzip < pyephem-3.7.5.1.tar.gz | tar xvf –~~

~~cd pyephem-3.7.5.1/~~

~~sudo python setup.py install~~

# Instalar py-serial

sudo apt-get install python-serial

# Instalar ICE

sudo apt-get install zeroc-ice34 ~~--fix-missing~~

sudo apt-get install python-zeroc-ice

# Instalar Minicom

sudo apt-get install minicom

# Instalar ARI-SRT desde google code

Los siguientes programas deben existir en la carpeta de trabajo:

parametersV01.py

SRTControlClient1.py

SRTcontrol.ice

SRT\_control\_lib\_test.py

SRTcontrolServer.py

SRTControl/\_\_init\_\_.py

SRTcontrol\_ice.py

SRT\_serial\_sim.py

Si se va a controlar desde un mac, el archivo SRTcontrol.ice debe ser compilado con slice2py en el mac para regenerar SRTcontrol\_ice.py y SRTControl/\_\_init\_\_.py

# Test de conversor USB-RS232

Conectar el conversor al puerto USB de Raspberry-Pi

Via SSH ejecutar “lsusb”

Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.

Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter

Bus 001 Device 005: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB

Bus 001 Device 006: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter

Bus 001 Device 007: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter

Los dispositivos bajo la descripcion USB-Serial adapter corresponde a los conversores conectados en la R-Pi. (En esta muestra hay 2 conversores).

Ejecutar “cd /dev” y “ls”

Los dispositivos conectados estaran bajo los nombres “ttyUSB0” o “ttyUSB1”

Test de funcionamiento del conversor se puede realizar usando pyserial y otro dispositivo serial

# Operacion del servidor ICE-SRT en la Raspberry-Pi

Abrir conection SSH con usuaria SRT

“ssh [srt@192.168.0.x](mailto:srt@192.168.0.x)”

Ir a la carpeta:

~/software/acs-ari/trunk/ARI/TCS/rpiSRT/src/iceSRT

ejecutar:

python SRTcontrolServer.py default -h 192.168.0.6 -p 10000

esperar mensaje:

importing SRT library

En caso de necesitar simular el SRT con un Segundo conversor USB-RS232 se debe abrir una segunda session SSH y ejecutar

python SRT\_serial\_sim.py

# Operacion del cliente ICE-SRT en laptop

En la maquina cliente se debe contar con instalacion de Ice y py-ephem.

Los siguientes archivos deben estar instalados en el directorio de trabajo:

SRTcontrol.ice

SRTcontrol\_ice.py

SRTControl/\_\_init\_\_.py

SRTControlClient1.py

Desde la maquina cliente se recomienta utilizar iPython y ejecutar la siguiente sesion:

In [1]: from SRTControlClient1 import \*

In [2]: s = SRT()

s.setIP(‘192.168.0.6 –p 10000’)

In [3]: s.connect()

Connecting to SRTController

In [4]: connected to controller

Proceed with SRT initialization

Serial Port: None

Antenna Initialised (sent to stow): False

In [4]: s.SetSerialPort('ttyUSB0')

Setting serial port

In [5]: Serial port initialized!

In [5]: s.Init('parametersV01')

loading parameters file and sending antenna to stow

In [6]: Done!

In [7]: s.AzEl(20,20)

moving the antenna

Out[7]: <IcePy.AsyncResult at 0x1088cb490>

In [8]: Commanding antenna movement

Movement finished

In [9]: s.status()

getting SRT status

In [10]: SRT antenna Status 2014-12-17 03:15:08

Commanded azimuth: 20.0

Commanded elevation: 20.0

Actual azimuth: 20.0390625

Actual Elevation: 20.1987743378

Slewing antenna: 0

Next axis to move: 1

Commanded to stow: 1

elevation axis at stow: 0

azimuth axis at stow position: 0

Controller Serial Port: ttyUSB0

last SRT command: [1418786051, ' move 0 0\n']

last SRT received message: [1418786051, ['M', '1','3\r']]

# Funciones disponibles

connect() : Conectarse con el servidor

disconnect() : Cerrar conexion al servidor (nota, esta funcion no afecta el estado del servidor)

status() : Obtiene Status del SRT

GetSerialPorts() : Provee los puertos USB-RS232 disponibles en R-Pi

SetSerialPorts*(port)* : configura el Puerto serial con que R-Pi controlara SRT, port es un string con el nombre del dispositivo i.e.: ‘ttyUSB0’

Init*(parameter\_file)* : rutina de inicializacion del SRT mediante carga de archivo de parametros y envio de antenna a posicion de stow (este paso es obligatorio); parameter\_file es un string con el nombre del archivo de parametros, la version actual corresponde al nombre “parametersV01”, este archivo esta guardado en la Raspberry-Pi . Este archiva puede ser editado o versiones diferentes se pueden utilizar para configurar el SRT.

Stow(): Comanda la antenna a posicion de Stow cuando sea necesario

AzEl*(Az, El)* : Envia la antenna a las coordenadas (Az,El). Esta funcion esta desarrollada como thread por lo que el software no es bloqueado mientras se ejecuta el movimiento. (Az,El) debe estar dentro de los limites definidos en el archivo de parametros:

0<Az<355 , 8<El<85

El envio de coordenadas fuera de este rango provocara el termino del servidor como medida de seguridad y el sistema debera reiniciarse.

track\_source*(source)* : Esta funcion esta aun en desarrollo .

Funciones radiometro

setFreq(freq, mode): configure

**1.- Levantar server en SRT2 / R-pi**

ssh 192.168.3.102

login: pi

password: raspberry

cd CURRENTLY\_MOD/rolguin/acs-ari/trunk/ARI/TCS/rpiSRT/src/iceSRT

python SRTcontrolServer.py 192.168.3.102 -p 10000

**2.- Levantar Client Ice en PC de control**

cd /CURRENTLY\_MOD/rolguin/acs-ari/trunk/ARI/TCS/rpiSRT/src/observingModes

python IceSRTClient.py

**3.- Cargar observing mode en PC de control**

cd /CURRENTLY\_MOD/rolguin/acs-ari/trunk/ARI/TCS/rpiSRT/src/observingModes

ipython

from obsmode import \*

om = ARI\_obsmodes()

om.SingleDishSRT('SRT1')

Lectura de datos de espectro

Mode 1:

Single scan, BW app 0.5 MHz

Mode 4:

3 Mode 1 scans patched together with BW app 1.2 MHz

SetFreq(1420.4,4)

* 3 x 500 KHz, 7.81 KHz spacing

Procesos

srt.tracking(source)

* thread ‘tracking’: srt.track\_source(source)
  + srt.track\_source(source):
    - start spectrum acquire thread: srt.GetSpectrum()
    - check source visibility (not visible ends the thread)
    - srt.track = True, srt.OnSource = False, srt.toSource = 0
    - while(srt.track)
      * get Az,El pair
      * compares with current antenna coordinates and decide if movement proceeds

🡪 srt.AzEl for movement

* + - * wait 2 seconds to repeat the loop
* srt.StopTrack() puede interrumpir el loop y finalizar el thread

🡪 srt.AzEl

* + - If serial port is in use ends the routine and back to the track loop
    - If not in use then sets seria port in use, and is Moving

🡪 SRTAzEl (Ice)

Start thread moving for check end of movement

Call AzEl\_thread en R-Pi

🡪 GetSRTThreads

reads Threads at server and look for ‘AzEl’ if not present then movement is finished

if movement is finished then IsMoving=False, portInUse=False

Obtain status to refresh variables (aznow, elnow)

R-Pi AzEl\_thread

Wait for last spectral data to be read by spectra()

Check if antenna is not slewing (self.slew ==0)

Start azel thread ‘AzEl’

🡪 cmd\_azel(az,el)

🡪 check (az,el) values

🡪 calculates counts of az and el encoder for movement

🡪 slew\_antenna

🡪 self.slew = 1

🡪loop until end the movement

Get\_Spectrum