**Telit UL865**

Objetivos:

* Recibir un SMS: modo texto, modo PDU
* Mandar un SMS: modo texto, modo PDU
* TCP cliente: no puedo poner mi compu como servidor porque necesito abrir los puertos necesarios en el router del LAC, al que no tengo acceso. Debo instalar un servidor TCP en mi casa.
* TCP servidor: Según el manual es posible, pero en la práctica no funcionó
* GET HTTP: Funciona bien
* POST HTTP: No se probó?
* Subida/descarga FTP: Funcionan bien, el servidor es el de la Universidad de Miami
* SMTP (enviar mail): Con conexión segura SSL. No se ha probado enviar un adjunto todavía.
* IMAP (leer mail): Con conexión segura SSL. La lectura de archivos adjuntos también funciona
* Grabar llamada: Funciona, pero no cualquier reproductor de audio puede reproducir el archivo resultante.
* Transmitir audio durante llamada: Funciona, pero se debe transmitir en el mismo formato que el que usa para grabar llamadas.
* DTMF (comandos por tono)
* Scripts de Python en memoria interna

AT Command Tester: <http://m2msupport.net/m2msupport/module-tester>

Drivers USB Serial PL2303: <http://www.ifamilysoftware.com/news37.html>

Los ejemplos de uso están en Telit\_IP\_Easy\_User\_Guide\_r19.pdf

AT+CMEE=2 sirve para que me devuelva los códigos de error

Serial 115200 8N1

Socket TCP

Como cliente no es posible hacerlo desde el Lac porque mi computadora no puede actuar como servidor, hay que hacer port forwarding en el router. Como servidor

AT+CGDCONT = 1,"IP","internet","0.0.0.0",0, AT#SCFG=1,1,300,90,600,50AT#SGACT=1,1

AT#FRWL=1,”0.0.0.0”,”0.0.0.0” //Configuración de las reglas de acceso del firewall

AT#SL=1,1,21

No parece funcionar. Ni siquiera funciona el ping, por lo que se descarta el uso de la placa como servidor TCP.

Como cliente

AT+CGDCONT = 1,"IP","internet","0.0.0.0",0, AT#SCFG=1,1,300,90,600,50AT#SGACT=1,1

AT#SD=1,0,23,”hostname”,0,0 //Modo online

Blablablá

+++ //Con esto podemos volver a enviar comandos AT

AT#SO=1 //Pone el socket en modo online, listo para enviar datos otra vez

+++

AT#SH=1 //Cierra el socket

AT#SD=1,0,23,”hostname”,0,1 //Modo commando

AT#SSEND=1

>blablabla CTRL-Z //Hasta 1500 bytes

AT#SH=1 //Cierra le socket

GET HTTP:

AT+CGDCONT = 1,"IP","internet","0.0.0.0",0, AT#SCFG=1,1,300,90,600,50AT#SGACT=1,1

AT&K=0 //Esto desactiva el control de flujo, sino el socket nunca se conecta

AT#SD=1,0,80,”www.google.com”,0,0 //Modo en línea

GET / HTTP/1.1<cr><lf>

Host: www.google.com <cr><lf>

Connection: keep-alive<cr><lf>

<cr><lf>

AT#SD=1,0,80,”www.google.com”,0,0,1 //Modo commando

AT#SSEND=1

>blablabla CTRL-z

//Si el servidor me responde me avisa con SRING:1

AT#SRECV=1,1500 //Leo 1500 bytes del buffer de recepción

//Leo los datos recibidos

AT#SH=1 //Cierro el socket

POST HTTP:

Descargar un archivo de un servidor FTP:

Como prueba vamos a subir un archivo de texto y descargarlo.

AT+CGDCONT = 1,"IP","internet","0.0.0.0",0, AT#SCFG=1,1,300,90,600,50AT#SGACT=1,1

AT&K=0

AT#FTPOPEN="ftp.swfwmd.state.fl.us","anonymous",[juanrom02@gmail.com](mailto:juanrom02@gmail.com)

AT#FTPTYPE=0

AT#FTPPUT="/pub/incoming/FTP.txt",1

AT#FTPAPPEXT=50

>…

OK

#FTPAPPEXT: 50

AT#FTPGET="/pub/incoming/FTP.txt"

CONNECT

(El archivo)

NO CARRIER

AT#FTPCLOSE

Mandar un Email (SMTP):

<http://www.samlogic.net/articles/smtp-commands-reference.htm>

Mandar un mail con adjunto:

Toda la conexión es idéntica al anterior

MAIL FROM:<[usuario@dominio.com](mailto:usuario@dominio.com)>

> 250 2.1.0 OK v186sm6902425qkb.16 - gsmtpRCPT TO:<[usuario@dominio.com](mailto:usuario@dominio.com)>

> 250 2.1.5 OK v186sm6902425qkb.16 - gsmtpDATA

> 354 Go ahead v186sm6902425qkb.16 - gsmtpDate: Mon, 06 Nov 2016 16:33:29 -0300

From: Telit <[usuario@dominio.com](mailto:usuario@dominio.com)>

Subject: Asunto del mensaje

To: [destino@domini.com](mailto:destino@domini.com)

MIME-Version: 1.0

Content-Type: multipart/mixed; boundary=”delimitador”

--delimitador

Content-Type: text/plain

Cuerpo del mensaje

--delimitador

Content-Type: image/jpeg;

Content-Disposition: attachment; filename=”imagen.jpg”

Content-Transfer-Encoding: base64



--delimitador--

.

Recibir un Email (IMAP):

<http://systemadmin.es/2009/05/conectar-a-un-servidor-imap-mediante-telnet>

Me puse a ver si la conexión SMTP, osea desde el EHLO, lo podía hacer con el código ya hecho (que usa smtplib). El problema es que esa librería llama a socket.py, y yo acabo de abrir el socket en la Telit asique no me hace falta. No podría usar smtplib. Lo mismo para imaplib.

Grabar llamada:

Audio\_Settings\_Application\_Note\_Rev\_5.pdf

ATD\*611 //Siendo \*611 el número que quiero llamar

AT#SPCM = 2,1,0 //Me manda el audio en forma de flujo PCM por el puerto serie

Guardo el flujo recibido en un archivo. Para escucharlo, uso el software Audacity.

Archivo > Importar > Datos en bruto…

Codificación: Signed 8-bit PCM

Orden de byte: Sin endianness

Canales: 1 canal (mono)

Frecuencia de muestreo: 8 kHz



Enviar audio grabado:

AT#SPCM=1,1

El archivo PCM que le mande en python se lee con ‘rb’

Recibir comandos por tono (DTMF):

AT#DTMF=1

Cuando le mando un tono por llamada me responde:

#DTMFEV: X //Siendo X el número

Guardar y ejecutar un script de Python en la placa:

Vamos a probar un script simple. Los print los manda al segundo puerto serie, por lo que hay que usar SER.sen() para mandar por el puerto que uso

import SER, MDM

import sys

SER.set\_speed('115200')

SER.send("START\r\n")

at = 'AT\r'

MDM.send(at, 5)

r = MDM.read()

SER.send(at + " > " + r)

Dado que el script que se encuentre habilitado se va a ejecutar cada vez que se inicie la placa, estaría bueno hacer un script de seteo de parámetros. El script lo envío con RealTerm.

import SER, MDM

SER.set\_speed('115200')

def mdm\_command(comando):

MDM.send(comando, 5)

r = MDM.read()

SER.send(comando + " > " + r)

mdm\_command('AT+CGDCONT = 1,"IP","internet","0.0.0.0",0\r')

mdm\_command('AT&K=0\r')

mdm\_command('AT#SSLSECCFG=1,0,0\r')

mdm\_command('AT+CMEE=2\r')

mdm\_command('AT#DTMF=1\r')

Hasta ahora hicimos todo con el Manhattan. Hay que probar la placa con un USB. Primero, hay que modificar unos jumpers en la placa EVK2. No hace falta nada más.

El puerto USB tiene 4 COM, pero solo uno funciona para el puerto serie principal. El puerto de debugging no es accesible por USB según parece.

**SIM800**

Objetivos:

* Recibir un SMS: modo texto, modo PDU
* Mandar un SMS: modo texto, modo PDU
* TCP cliente:
* TCP servidor:
* GET HTTP:
* POST HTTP:
* Subida/descarga FTP:
* SMTP (enviar mail):
* IMAP (leer mail):
* Grabar llamada:
* Transmitir audio durante llamada:
* DTMF (comandos por tono)

La UART de la Raspberry Pi funciona a 3.3V. Un programador CP2102 también tiene una UART TTL a 3.3V. Para conectarlos con el chip SIM800 hace falta el siguiente circuito, porque este chip tiene un puerto serie a 2.8V:



Sin embargo, en la tesis de los chicos no usan ninguna resistencia, lo cual me parece más lógico. Quizás el módulo ya viene con las resistencias.

La placa EVK2 trabaja a tensión de RS232, por lo que haría falta un MAX232 como interfaz.

Signal strength: AT+CSQ

Operator Info: AT+COPS?

Registration: AT+CREG?

Available Networks: AT+COPS=?

Preferred Operators: AT+CPOL?

Para forzar el registro en la red:

AT+COPS=1,1,”UNIFON”

Realizar llamada

Atender llamada

Enviar un SMS

Recibir un SMS

El chip de Claro causa problemas??

Conexión Bluetooth

Este chip puede usar conexiones Bluetooth. Para conectarse a un celular:

AT+BTPOWER=1 //Activa la radio BT

AT+BTSCAN=1,20 //Busca dispositivos. Me devuelve una lista

AT+BTPAIR=0,6 //Me conecto al sexto dispositivo de la lista

AT+BTPAIR=1,1 //Realizo el emparejamiento

AT+BTHOST=<name> //Cambio el nombre del dispositivo (para no confundir los dos chips SIM800)

Para conectarse en modo serie (SPP), luego de emparejarse:

AT+BTSTATUS? //Estado de la conexión. 0 es inicial, 5 es en espera

AT+BTSCAN=1,20

AT+GETPROF=1 //Me devuelve los modos de conexión del primer dispositivo. Cada modo tiene su ID asociado.

+BTCONNECTING: //Intento conectarme desde el cliente

AT+BTACPT=1 //Acepto la conexión

AT+BTSPPCFG? //Con este comando puedo ver en que modo se encuentra conectado el SIM800

+BTSPPCFG: S,1,1 //S: modo servidor, 1: ID de la conexión, 1: modo APP, por lo que puedo enviar datos al cliente (en modo 0 o AT solo puedo recibir)

En modo SPP, los paquetes son enviados y recibidos por el puerto serie del SIM800

AT+BTSPPSEND //Procedo a enviar datos, finalizo con 0x1A

>

+BTSPPDATA: 1,10,soy xiaomi //Recibo datos

Solo se puede conectar un cliente a la vez en este modo

Para conectar el SIM800 con la RPi por BT, es necesario habilitar el modo SPP. Por defecto, la Raspberry solo tiene habilitado el modo AVRCP, que se utiliza para controles remotos multimedia.

<https://www.raspberrypi.org/forums/viewtopic.php?p=919420#p919420>

sudo rfcomm watch hci0 en la RPi mientras

AT+BTCONNECT=x,4 en el SIM800

Firmware

1418B04SIM800C24\_BT: No se registra

1418B04SIM800C24: No se registra

1418B04SIM800C32\_BT: No se puede flashear

1418B02SIM800C32\_BT: No se puede flashear. Ningún firmware que termina en C32 debe ser flasheable.

**Dongle ZTE MF193**

Objetivos:

* Recibir un SMS: modo texto, modo PDU
* Mandar un SMS: modo texto (incompatible), modo PDU
* TCP/HTTP/FTP/SMTP/IMAP: El dongle no posee pila TCP/IP integrada, por lo que se utiliza el código ya escrito para la Raspberry.
* Grabar llamada: Mismo formato que para la Telit.
* Transmitir audio durante llamada: Transmitiendo por el mismo puerto que recibo no funciona. Probar otros puertos.
* DTMF (comandos por tono)

Encontré en el LAC este Dongle de Claro.

COM17, autobaud

Realizar llamada

Por el COM17 mando el ATD\*611. Por el COM15 recibo el audio de la llamada, como flujo PCM, igual que para la Telit. En Audacity:

Archivo > Importar > Datos en bruto…

Codificación: U-law

Orden de byte: Little Endian

Canales: 1 canal (mono)

Frecuencia de muestreo: 8 kHz

Para finalizar una llamada envío el comando AT+CHUP

Atender llamada

Recibo +CLIP: en COM17, atiendo con ATA. Recibo el audio por COM15, lo mando por COM15 pero no parece andar.

Enviar un SMS

AT+CMGF=0 //No soporta modo texto, solo PDU. Ojo que si hago AT+CMGF=? me devuelve +CMGF=(0-1) como si soportara modo texto

La tesis del comunicador está hecha para modo texto nomás

AT+CMGS=29

> 08915+3402000000F111000D91453915572013F70000FF10D4F29C0E8AE15EB1182806D3D170

+CMGS: 102 //Me devuelve un ID de mensaje.

OK

Es importante tener en cuenta que el paquete PDU se envía como caracteres ASCII, aunque use solamente dígitos hexadecimales.

Dado que no encontré un datasheet del dongle, tuve que hacer ingeniería inversa. Usando el software Advanced Serial Port Monitor (<http://www.aggsoft.com/serial-port-monitor/download.htm>) pude observar los paquetes entre el software de Claro y el dongle.

http://www.developershome.com/sms/cmgsCommand4.asp

<http://www.smartposition.nl/resources/sms_pdu.html>

Transferencia FTP

AT+CLAC me devuelve una lista de los comandos disponibles en el dispositivo. Sin embargo, gracias a la observación de paquetes del paso anterior, me di cuenta de que acepta otros comandos propios de ZTE. No se porqué no figuran en esta lista

<http://forum.xda-developers.com/showthread.php?t=2396752>

$QCSIMSTAT Get/Set SIM status (init completed?)

$QCCNMI Similar to 27.005 +CNMI except for the behavior with $QCCNMI=1,2

$QCCLR Clears mobile error log

$QCDMG Transitions to Diagnostics Monitor (DM) operation

$QCDMR Sets DM baud rate

$QCDNSP Sets primary DNS IP address

$QCDNSS Sets secondary DNS IP address

$QCTER Sets TE-DCE baud rate; baud rates supported are identical to +IPR command

$QCSLOT Sets SIM card on which slot commands will operate

$QCPINSTAT? Sends to the ME the status of all PINs for all cards

$QCPDPP Sets authentication for PDP-IP packet data calls

$QCPDPLT Enables/disables tolerance to long delays in PDP call setup

$QCPWRDN Power Down the UE

$QCDGEN Generates data over +CGACT activated PDP context

$BREW ?? Start a "Brew MP" serial session (by entering the BrewMP Command Processor)

$QCSYSMODE Get hardware available network modes (e.g. WCDMA + HSDPA + HSUPA)

Basandome en la lista que me devolvió, no posee pila TCP/IP. Se usa el software ya escrito que implementa la pila, para FTP, IMAP, SMTP y HTTP.

**Dongle ZTE MF626**

Objetivos:

* Recibir un SMS: modo texto, modo PDU
* Mandar un SMS: modo texto, modo PDU
* TCP/HTTP/FTP/SMTP/IMAP: El dongle no posee pila TCP/IP integrada, por lo que se utiliza el código ya escrito para la Raspberry.
* Grabar llamada:
* Transmitir audio durante llamada:
* DTMF (comandos por tono): No posee decodificador integrado

Se conecta por ZTE NMEA Device.

En la RPi me envía continuamente +ZUSIMR: 2, se corrige mandando AT+CPMS?

Recibir SMS

AT+CPMS=”ME” no anda, solo “SM”

El resto funciona bien, igual que para la Telit

El envío y recepción de instancias entre la Raspberry y un celular no funciona, por un tema de decodificación de caracteres, entre GSM 03.38 y UNICODE UTF-16.

Xiaomi/Tuenti > Moto/Claro: \_ (0x11/0x5F) enviado se vuelve ¿ (0x60/0xBF)

Dongle/Claro > Moto/Tuenti: \_ se lee bien

Moto/Tuenti > Dongle/Claro: \_ enviado se vuelve ¿

Moto/Claro > Xiaomi/Tuenti: \_ se lee bien

Moto/Claro > Xiaomi/Personal: \_ se lee bien

Xiaomi/Tuenti > Xiaomi/Personal: \_ enviado se vuelve ¿

Xiaomi/Tuenti > Xiaomi/Tuenti: \_ se lee bien

El envío de Tuenti parece ser el problemático.

Enviar un SMS

Funciona en ambos modos, igual que para la Telit.

En Windows funciona, porque no requiere el SMSC. Quizás se configure con el software incluido.

En Linux, es necesario agregar el SMSC a la PDU.

Además, por más que configure con CNMI para que me muestre los SMS recibidos y no los guarde, cuando hago AT+CMGL, me tira el último mensaje recibido. La Telit no lo hace; debe ser por un comando propietario de configuración de ZTE.

Llamadas

No puedo recibir ni realizar llamadas. No tengo otro chip Claro para probar, y los chips de otras empresas no funcionan porque el dongle está bloqueado.

**Comandos AT en celulares**

Objetivos:

* Recibir un SMS: No existen el modo texto o PDU, porque se usa adb
* Mandar un SMS: No existen el modo texto o PDU, porque se usa adb
* TCP/HTTP/FTP/SMTP/IMAP: Se hacen con el código de la Raspberry, habilitando la opción de compartir internet por USB (USB Tethering).
* Grabar llamada: Es posible pero no es útil.
* Transmitir audio durante llamada: Android no permite la transmisión de audio por otro canal que no sea el del micrófono (<https://developer.android.com/guide/topics/media/mediaplayer.html>).
* DTMF (comandos por tono): No vale la pena ahondar en esto si no se pueden transmitir los audios con las opciones.

Parece que es necesario tener celular rooteado

Se intenta enviar comandos usando adb (Android Debug Bridge).

*sudo apt-get install android-tools-adb*

para instalar adb en la Raspberry Pi

Hacemos una llamada con

*adb shell am start -a android.intent.action.CALL -d* [*tel:+19709228605*](tel:+19709228605)

Envío de SMS (el cellular debe estar desbloqueado):

adb shell am start -a android.intent.action.SENDTO -d sms:CCXXXXXXXXXX --es sms\_body "SMS BODY GOES HERE" --ez exit\_on\_sent true

adb shell input keyevent 22 #Selecciona el botón de enviar

adb shell input keyevent 66 #Presiona el botón

Para desbloquear el celular (varía entre modelos)

adb shell input keyevent 26 #Tecla de desbloqueo

adb shell input swipe 540 1900 540 1000 #Simula un movimiento del dedo sobre la pantalla. Los números representan las coordinadas (x,y) iniciales y finales del movimiento, respectivamente. Las coordenadas dependen de la resolución de la pantalla (1080x1920 en mi caso)

adb shell input text XXXX #Introduzco el código de desbloqueo

IMPORTANTE: Desactivar el texto predictivo. No es molestia en los SMS de texto plano pero si en las instancias

Lectura de SMS:

adb shell

su

# cd /data/data/com.android.providers.telephony/databases

sqlite3 mmssms.db

SQLite version 3.8.2 2013-12-06 14:53:30

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> SELECT \* FROM sms WHERE read=0; read=1 para mensajes leídos

905|84|**+5491128456083**||1488255895941|1488255893000|0|0|-1|1|0||**WhatsApp code 797-799.**|+543200000811|0|0|1|0|0|2|67928222826792704|Mjc1ODQxNDk2ODE4ODEyMTY6MDorNTQ5MTEyODQ1NjA4MzoxNDg4MjU1ODk1OTQx|27584149681881216|0||0|1668338029|2|0|3|0||0|0|

909|85|**+18503210527**||1488393825446|1488393814000|0|0|-1|1|0||**Nah I'm at Snowmass :/**|+16363848922|0|0|1|0|0|2|67969328229156608|Mjc1ODQxNDk2ODE4ODEyMTY6MDorMTg1MDMyMTA1Mjc6MTQ4ODM5MzgyNTQ0Ng|27584149681881216|0||0|1668338029|2|0|3|0||0|0|

Si cuando quiero ejecutar sqlite3 me dice not found es porque no está instalado en el dispositivo. Para instalarlo:

<https://forum.xda-developers.com/android/general/how-to-install-sqlite3sql-database-t3283409>

Quiero ver los nombres de las columnas, por lo que voy a pasar la base de datos a la compu. Para ello, debo copiarla a una carpeta que me sea accesible desde Windows, con ES Explorer. Pude abrirla con SQLite Administrator (<http://sqliteadmin.orbmu2k.de/>).

read: 0 (no leído), 1 (leído)

status: -1 (mensajes recibidos y leidos?), 0 (mensaje enviado), 32 (no se pudo enviar)

type: 1 (smssubmit), 2 (smsdeliver), 3 (mensaje almacenado pero no enviado), 4 (enviando), 5 (no se pudo enviar?)

Logcat SMS enviado

adb locgat Microlog:D \*:S –v long

Exitosamente:

06-13 15:29:03.885 7851 8297 D Microlog: 8346273 Mms [DEBUG] 06-13 15:29:03 p.m. SmsMessageSender, message queued, uri:content://sms/1209

06-13 15:29:06.961 7851 8297 D Microlog: 8349348 Mms [DEBUG] 06-13 15:29:06 p.m. SmsReportService, runnable mErrorStartId is -1

**06-13 15:29:08.501 7851 8297 D Microlog: 8350889 Mms [DEBUG] 06-13 15:29:08 p.m. SmsReceiver, sms ack received, uri: content://sms/1209, result: -1**

06-13 15:29:08.741 7851 8297 D Microlog: 8351129 Mms [DEBUG] 06-13 15:29:08 p.m. MessageStatusService, sms delivered, uri: content://sms/1209

06-13 15:29:11.538 7851 8297 D Microlog: 8353926 Mms [DEBUG] 06-13 15:29:11 p.m. SmsReportService, runnable mErrorStartId is -1

Fallo:

06-13 15:30:24.150 7851 8297 D Microlog: 8426538 Mms [DEBUG] 06-13 15:30:24 p.m. SmsMessageSender, message queued, uri:content://sms/1210

06-13 15:30:27.267 7851 8297 D Microlog: 8429655 Mms [DEBUG] 06-13 15:30:27 p.m. SmsReportService, runnable mErrorStartId is -1

**06-13 15:30:28.261 7851 8297 D Microlog: 8430649 Mms [DEBUG] 06-13 15:30:28 p.m. SmsReceiver, sms ack received, uri: content://sms/1210, result: 1**

06-13 15:30:31.309 7851 8297 D Microlog: 8433697 Mms [DEBUG] 06-13 15:30:31 p.m. SmsReportService, runnable mErrorStartId is -1

Logcat SMS Recibido

06-12 19:14:07.045 2874 3209 D InboundSmsHandler: writeMessageToInbox uri is not null

06-12 19:14:07.045 2874 3209 D InboundSmsHandler: writeMessageToInbox insert uri is successfully

06-12 19:14:07.045 2874 3209 D InboundSmsHandler: writeMessageToInbox finally

06-12 19:14:07.096 2874 3209 D InboundSmsHandler: writeMessageToInbox args is wrong

06-12 19:14:10.191 4390 4535 D Microlog: 14313498 Mms [DEBUG] 06-12 19:14:10 p.m. SmsReportService, runnable mErrorStartId is -1

Con subprocess no puedo ejecutar commandos posteriores a “adb shell”, porque no puede interpretar la línea de comandos del dispositivo. Se usa la librería pexpect para lograrlo.

El grabado y envío de audio debe hacerse con un cable 3.5mm.

FTP, HTTP, IMAP y SMTP se hacen con el código de la Raspberry, habilitando la opción de compartir Internet por USB en el celular.

Transmitir audio durante llamada

Android no permite el reemplazo del micrófono como entrada de audio durante una llamada. Una solución sería reproducir el audio por el parlante externo del teléfono durante la llamada; esto resulta en una mala calidad de la grabación. Enviar el audio por cable de 3.5 mm no es posible, ya que la ficha que poseen por ejemplo los auriculares manos libres es distinta, y eso les permite transmitir audio en ambas direcciones (para auriculares y micrófono).

Grabar llamada

Enviar el audio de la llamada por cable 3.5 no es posible porque la Raspberry no tiene entrada de audio. Utilizar una aplicación que grabe la llamada es posible; pero como no se puede transmitir el audio desde la Raspberry, no se pueden dar las opciones disponibles, por lo que pierde utilidad el poder grabar la llamada.

Envío de mensajes en modo PDU

El programa está hecho para mandar SMS en modo texto solamente. No todos los dongles soportan este modo (el ZTE no lo soporta), por lo que hay que mandar SMS siempre en modo PDU.

modemClass.Gsm.sendMessage(self, plainText, telephoneNumber)

modemClass.Gsm. sendMessageInstance(self, message, telephoneNumber)

Usando la librería messaging (<https://github.com/pmarti/python-messaging>) se procede a modificar el código.

También se deben modificad la función de recepción

modemClass.Gsm.receive(self)

Detección de dispositivos conectados

El comunicador debe consultar el modelo del dispositivo de redes móviles antes de mandar un mensaje o un archivo por GPRS o EMAIL.

transmitterClass.Transmitter.send(self, messageInstance)

networkClass.Network.send(self, message, destinationHost, destinationTcpPort, destinationUdpPort)

Hace falta testear el envío de texto y archivos sobre sockets TCP/UDP en la Telit.

No se puede usar la Telit como servidor TCP/UDP, no hay siquiera ping; por lo que los receive quedan como están, usando networkClass.Network.receiveTCP() y …receiveUDP().

Se crea una nueva función para el envío de archivos, cuando se encuentre conectado un módulo GSM que posea pila TCP/IP

networkClass.Network.sendFileAT(self, fileName)

Instalación pantalla para Raspberry

<http://www.jeffgeerling.com/blog/2016/review-elecrow-hdmi-5-800x480-tft-display-xpt2046-touch-controller>

sudo apt-get install xinput-calibrator

sudo nano /boot/config.txt

En el archivo agregar:

hdmi\_group=2

hdmi\_mode=1

hdmi\_mode=87

hdmi\_cvt 800 400 60 6 0 0 0

dtparam=spi=on

dtparam=i2c\_arm=on

dtoverlay=ads7846,cs=1,penirq=25,penirq\_pull=2,speed=50000,keep\_vref\_on=0,swapxy=0,pmax=255,xohms=150,xmin=200,xmax=3900,ymin=200,ymax=3900

dtoverlay=w1-gpio-pullup,gpiopin=4,extpullup=1

Y reiniciar la Raspberry.

Como no puedo conectar la pantalla sobre la Raspberry (me quedo sin GPIO y puertos), no anda la pantalla táctil.

Redes 2G

En EEUU no tengo cobertura 2G, por lo que no puedo hacer nada con el SIM800C o el Dongle

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SIM800C** | **MFRC522** | **ESP8266** |
| **Tecnología** | GSM/GPRS | NFC | WiFi 802.11b/g/n |
| **Tensión (V)** | 3.7-4.2 | 3.3/5 | 3.3/5 |
| **Corriente máxima (A)** | 2 | 0.1 | 0.17 |
| **Puertos Serie** | UART | SPI/I2C/UART? | SPI/I2C?/UART |
|  |  |  |  |

Conexiones Serie

Raspberry Pi (/dev/serial0) <-- --> CP2102

ESP8266 <-- --> CP2102

Raspberry Pi <-- --> ESP8266 (usando comandos)

<http://www.instructables.com/id/Connect-an-ESP8266-to-your-RaspberryPi/>

echo –ne ‘AT\r\n’ > /dev/serial0

cat -v < /dev/serial0

AT+CWMODE=1 //Operating Mode 1. Client 2. Access Point 3. Client and Access Point

AT+CWLAP //View available networks

AT+CWJAP=ssid,password // Join network

AT+CIPSTART=”TCP”,”www.google.com”,80

AT+CIPSEND=16 //Cantidad de caracteres a enviar

> GET / HTTP/1.1\r\n\r\n //Toda consulta HTML termina con una línea en blanco

AT+CIPCLOSE

Raspberry Pi <-- --> ESP8266 (con Minicom)

Puertos SPI e I2C de ESP8266

Los puertos se habilitan desde Preferences > Raspberry Pi Configuration > Pestaña Interfaces. Es necesario reiniciar para aplicar los cambios.

Para probar el puerto SPI de la Raspberry: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md#troubleshooting>

<http://docs.micropython.org/en/latest/esp8266/esp8266/quickref.html#pins-and-gpio>

Probé el puerto SPI del ESP8266 en modo loopback y parece funcionar.

Ni el ESP8266 (con MicroPython) ni la RPi pueden funcionar en modo SPI Slave, por lo que no se pueden conectar entre si.

Puerto SPI en MFRC522

<https://github.com/raspberrypi/linux/issues/1547>

Basándome en las instrucciones del link anterior, pareciera que el hardware está dañado. Se descarta, voy a pedir otro.

Instalar microPython en ESP8266

<https://github.com/pfalcon/esp-open-sdk>

$ sudo apt-get install make unrar-free autoconf automake libtool gcc g++ gperf \

flex bison texinfo gawk ncurses-dev libexpat-dev python-dev python python-serial \

sed git unzip bash help2man wget bzip2

$ sudo apt-get install libtool-bin

$ git clone --recursive <https://github.com/pfalcon/esp-open-sdk.git>

$ cd esp-open-sdk/

~/esp-open-sdk/ $ make

~/esp-open-sdk/ $ PATH=/home/pi/esp-open-sdk/xtensa-lx106-elf/bin/:$PATH

<https://github.com/micropython/micropython/tree/master/esp8266>

~/esp-open-sdk/ $ cd

$ git clone <https://github.com/micropython/micropython>

$ cd micropython/

~/micropython/ $ git submodule update –init

~/micropython/ $ make –C mpy-cross

~/micropython/ $ cd esp8266/

~/micropython/esp8266/ $ make axtls

~/micropython/esp8266/ $ make

Luego de esto debo asegurarme que tengo los permisos para acceder al Puerto serie (/dev/ttyAMA0)

ls -l /dev/ttyAMA0

crw-rw---- 1 root **tty** 204, 64 Jul 16 19:44 /dev/ttyAMA0

Agrego mi usuario al grupo tty

sudo adduser pi tty

Además, debo acceder a

sudo nano /boot/cmdline.txt

y borrar la línea "console=serial0,115200"

esptool.py --port /dev/ttyAMA0 erase\_flash

Después de reiniciar el ESP8266

make PORT=/dev/ttyAMA0 deploy

Recordemos que el modo flash se active poniendo GPIO0 en bajo.

Para mostrar en pantalla las excepciones del código,

import traceback

…

print traceback.format\_exc()

Conexión a internet con ppp

Es necesario agregar el usuario al grupo dip. Para eso

sudo nano /etc/group

dip:x:30:pi

El debugging del proceso de conexión mediante

bash -c "tail -f /var/log/messages"

Testing conexiones controllerClass

android gprs

android wifi

android gprs

android wifi

android nada

nada

android wifi

nada

dongle wifi

dongle wifi + dongle 3g

dongle 3g

nada > Hay que desconectar el GPRS primero

ConfigPPP

es llamado desde connectAT(). Es necesario modificar el archivo /etc/ppp/chatscripts/mobile-modem.

chat, reemplazando

'OK-AT-OK' @/etc/ppp/chatscripts/apn.ar.personal por

'OK-AT-OK' @/etc/ppp/chatscripts/apn.operator

Asume que el dispositivo GSM se registra automáticamente.

Github commit

git add .

git commit –m “comentario”

git push origin master

Audio durante llamadas

<https://github.com/pndurette/gTTS>

Al principio tira error al importar, se soluciona con

sudo pip install -U requests

esto rompe el pip. Para arreglarlo

sudo apt-get remove python-pip

sudo easy\_install pip

<https://github.com/jiaaro/pydub>

apt-get install libav-tools

pip install pydub

Para convertir con avconv, el formato es:

avconv –i inputfile –f s8 –ar 8000 outputfile