Rotary Un-Smartphone

DEVELOPMENT SUPPLEMENT

EDITED 11/24/23, J.Haupt

Flashing firmware via the Arduino IDE

The RUSP firmware can be downloaded from github:
https://github.com/jhaupt/RotaryUnSmartphone/tree/main

- The /RUSP directory is the parent directory for the phone's firmware.
- The /SDCard directory contains the files that go on the MicroSD card.
- The /usbBridge directory contains the firmware for the ATmega8U2 which lets that chip function as a USB-to-UART bridge. This is unlikely ever to need updating by the end user, and it requires an In-System Programmer (ISP) to be flashed, in any case.

1. Download and install the **Arduino Integrated Development Environment**:
   https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE
   NOTE: If you're on Linux (Debian) they have you download the AppImage package, but it's generally available in the apt repositories.

2. Install **MegaCore** by MCUDude:
   Follow his very easy instructions here:
   https://github.com/MCUdude/MegaCore#boards-manager-installation

> NOTE: The Atmega1280 the RUSP runs on needs to have a 16MHz external upload speed of 115200 bauds to communicate with the LARA-R6 cell modem. **This seems to be set this way by default in MegaCore**, but for completeness I will mentioned that the the following line needs to be set as shown in the "boards.txt" file for MegaCore:
>
> ```
> >1280.menu.clock.16MHz_external.upload.speed=115200
> ```
>
> Example location for boards.txt :
> ~/.arduino15/packages/MegaCore/hardware/avr/2.1.3.

3. Install dependent libraries:
   a. **GxEPD2**: This is NOT the official fork from ZinggJM; I haphazardly added support for the 2.9" flexible ePaper display that the Rotary Un-Smartphone uses. To install:

1. Download the modified repository, called GxEPD2-RUSP, here: https://github.com/jhaupt/GxEPD2-RUSP.
2. The GxEPD2-RUSP directory just contains the GxEPD2 directory, which is what we really want. Zip the GxEPD2 directory.
3. In the Arduino IDE go to Sketch>Include Library>Add .ZIP Library. Select the GxEPD2.zip file you just made, and press OK.
4. It's installed now. The files you downloaded, including the ZIP file, can be deleted.

b. **Adafruit GFX Library** by Adafruit: Install the "standard" way in the Arduino IDE by going go to Tools>Manage Libraries, and, after waiting for the library list to load, use the search bar to find this. Click the "install" button next to the search result to install it. If it asks to install dependencies, say yes.

c. **EnableInterrupt** by Mike "GreyGnome" Schwager: Install the standard way as above.

d. **SD** by Arduino: Install the standard way as above.

4. **Configure the board settings for the RUSP:**

In the Arduino IDE, under Tools, select Board > MegaCore > ATmega1280
The default settings are **almost** what we want. Just switch "LTO disabled" to "LTO enabled":

- Clock: External 16MHz
- BOD: BOX 2.7V
- EEPROM: EEPROM retained
- Compiler LTO: LTO enabled
- Pinout: Arduino MEGA pinout
- Bootloader: Yes (UART0)

5. **Flash (upload) the firmware:**
   a. Connect the RUSP to your laptop with a USB to USB-C data cable.
   b. Turn on the RUSP, if not already done.
   c. Under Tools > Port, select the port the RUSP is connected to (it should be the only one there).
   d. To upload, strike "Ctrl + U", or use the little arrow icon at the top of the Arduino IDE.
   e. A green LED behind the bell will flash rapidly and the IDE will tell you the upload is in progress. Do not disturb the USB cable or RUSP until upload is complete.

Done!

Make calls by interacting directly with the cellular radio hardware

The component that actually connects with the cellular network (i.e., the chip that makes the RUSP a phone and not just a handheld rotary thingy) is called a cellular modem, or cellular radio. In the RUSP, the cell modem is a **LARA-R6**, made by uBlox.

The LARA is controlled by the RUSPs microcontroller (an ATmega1280-16), which itself is set up to behave as an Arduino for development ease and tinkering. For brevity below I will call the ATmega1280-16 the **DTE**, which stands for "Device Terminal Equipment".

The way the DTE passes instructions to the LARA (and receives them) is via a serial connection on the RUSP Motherboard using the Hayes command set. These commands are often called "AT commands" because each command starts with "AT", as we'll see below. For example, the command to tell the LARA to **answer** an incoming call is ATA. The command to **hang up** is ATH. There are many Hayes commands, and the complete list of valid ones for a specific device can be found in that device's datasheet. For the **LARA-R6**, the complete AT reference can be found here: LARA-R6 Command Reference

With thheree RUSP, commands sent from the DTE to the LARA, or vice versa, are echoed over the USB connection, wherehich we call a **Serial Console**. Commands entered over the serial console are likewise echoed to the LARA so that we can, essentially, bypass the RUSP and deal directly with the LARA. This is important for development and debugging, giving developers or tinkerers a way to experiment with operating on a cell network directly.

To set up this connection, all that's needed is a data USB cable with a USB-C connector to plug into the port on the bottom of the RUSP. Note that as USB-C has also become ubiquitous for charging (as is also the case for the RUSP), many USB-C cables simply don't have the data wires present, so make sure your cable is in fact a data USB cable.

1. Connect the RUSP to a computer with your data USB cable and turn the RUSP on, if it isn't already.

On Linux or any UNIX-like system:

2. Simply run `$screen /dev/ttyACM0` from a terminal to establish the serial connection.

   As far as I can tell, the USB connection will always get assigned to a ttyACM device node, and almost always /dev/ttyACM0. This can be checked by running `$sudo dmesg` after plugging in the RUSP.

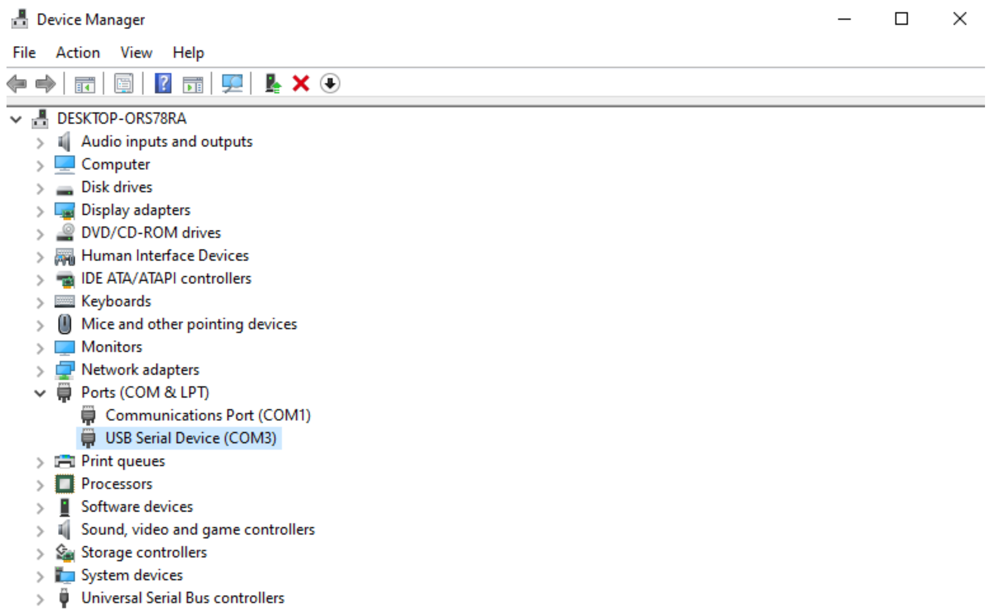
   To close the connection hit [ctrl+a] followed by [k], and then [y] to confirm closing.

On Windows

2. Download and install PuTTY: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

   Most users will want to download the first (64-bit x86) installer on that page.

3. The RUSP will be detected as a USB Serial Device, or similar, and be assigned as a COM port (e.g. COM1, COM2, etc). This can be checked in the Device Manager. Below, the RUSP appears as COM3:

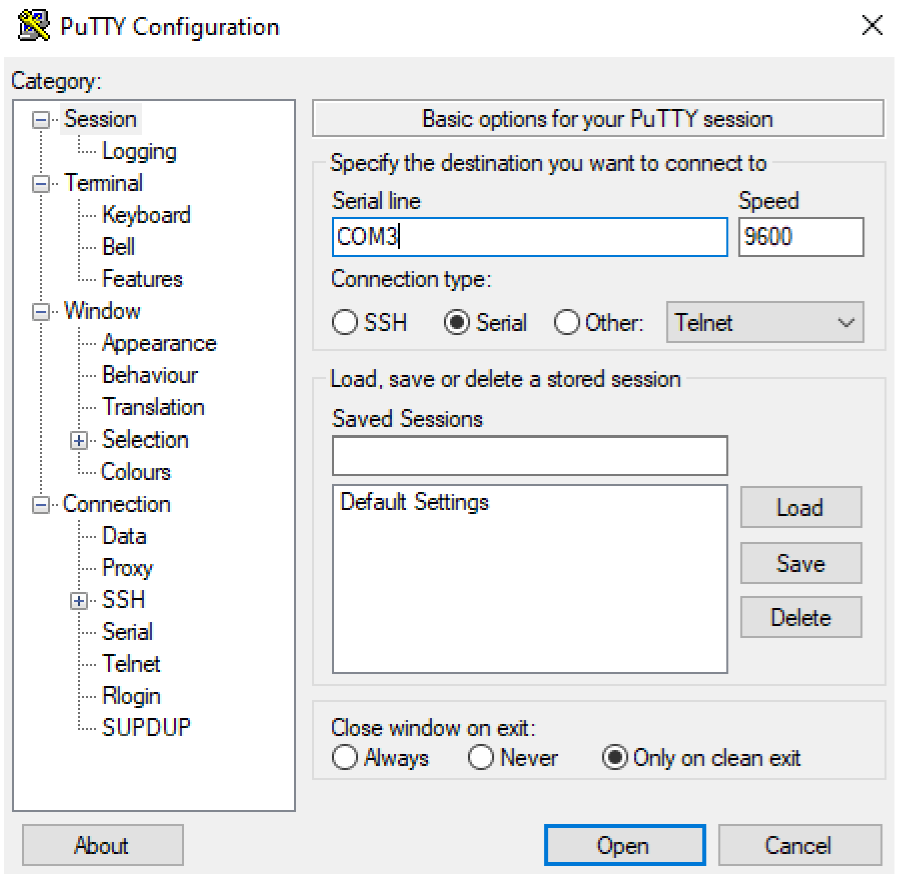


Device Manager

File Action View Help

- DESKTOP-ORS78RA
  - Audio inputs and outputs
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Ports (COM & LPT)
    - Communications Port (COM1)
    - USB Serial Device (COM3)
  - Print queues
  - Processors
  - Software devices
  - Sound, video and game controllers
  - Storage controllers
  - System devices
  - Universal Serial Bus controllers

4. Open PuTTY and select “Serial” as the connection type. The destination should be set to the COM port listed in the device manager (or just try a few different numbers), and the Speed should be 9600:



PuTTY Configuration

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, SSH, Serial, Telnet, Rlogin, SUPDUP

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM3

Speed: 9600

Connection type: SSH [✓ Serial] Other: Telnet

Load, save or delete a stored session

Saved Sessions

Default Settings

Load Save Delete

Close window on exit: Always Never [✓ Only on clean exit]

About Open Cancel

5. After clicking “Open”, the connection window will appear.

The RUSP will restart as soon as the connection is established, and after a number of seconds, some text will appear on the screen. This text may include words like `OK`, or `ERROR`, all of which is normal.

Testing some basic functionality

1. If you type `AT` or `at` (the commands are not case-sensitive), followed by [ENTER], the console should spit back `OK`, which is coming directly from the LARA and affirms you are connected to it.

   If you can't get `OK` to appear, turn off the RUSP, turn it back on, and restart the serial console connection. This will ensure that both the LARA and the DTE are in the ON state.

2. Check the network registration status. Assuming a SIM card is inserted, this will tell you whether the phone is attached to the network, in the process of trying, or has failed to do so:

   Enter `AT+CREG?`. The LARA will quickly send a response that looks like

```
+CREG: 0,0
```

   The second digit is the important one. It means the following:
   0 = not registered and not searching
   1 = registered on home network
   2 = not registered but searching
   3 = registration denied
   4 = unknown
   5 = roaming
   6 = SMS-only, home network
   7 = SMS-only, roaming

   So if the RUSP has successfully connected to the network in your local area, it looks like

```
+CREG: 0,1
```

   It can take a few minutes for the network registration to happen, but if it winds up as 3, or stays as 2, the combination of SIM card and locality is unlikely to work, at least until the RUSP gets PTCRB certification.

3. We can set the LARA to automatically report when a change in network status occurs by turning on Unsolicited Result Codes (URC) for network registration. This is done by sending `AT+CREG=1`. This changes the first digit above to 1, indicating that URCs are turned on.

   If you see the second digit change to the coveted 1, you can…

Receiving a call

1. Turn on URCs for calls by sending `AT+UCALLSTAT=1`

   As with the AT+CREG command above, we can also send `AT+UCALLSTAT?` to directly query this parameter. The response, or the URC, will look something like this:

```
+UCALLSTAT: 0
```

   The meaning of the digit is as follows:
   0 = call active
   1 = call on hold
   2 = dialing out
   3 = alerting (the phone being called is ringing)
   4 = incoming call (the RUSP is “ringing”)
   5 = waiting
   6 = disconnected
   7 = connected

2. Have someone call the RUSP’s phone number (as defined by the SIM card you’re using)

3. A URC should appear indicating an incoming call: `+UCALLSTAT: 4`

4. See caller ID data about the incoming call by sending `AT+CLCC`

5. Answer the call by sending `ATA`

6. You should be able to talk and listen over the RUSP’s built-in speaker and microphone. Audio-related commands are listed below under *other useful Hayes commands*.

Placing a call

1. If not already done, turn on URCs for calls by sending `AT+UCALLSTAT=1`

2. Set the service class to voice: `AT+FCLASS=8`

   Note that setting this parameter to 0 would set the service class to “data”.

3. Make a call. For example, to call 631 965 3409, send `ATD6319653409`

   Note that this command is agnostic to number formatting. Whatever your regional phone number format is (however many or few digits), the numbers simply get appended to the ATD command.

4. As described above, the UCALLSTAT URCs, or direct query using AT+UCALLSTAT?, will indicate the connection status as the call is being placed.

Reading an SMS

1. Switch to text mode: `AT+CMGF=1`
   Note that setting this parameter to 0 switches to PDU mode

2. List messages: `AT+CMGL`

3. Read a message: `AT+CMGR=n`
   Here, *n* is the message number

Other useful Hayes commands

Remember to check the LARA-R6 Command Reference for details. [Ctrl+F] is your friend!

General

- Turn on verbose error reporting: `AT+CMEE=2`

Basic audio settings

This is something I'd like a better handle on. Need to study the command reference more.

- Read audio path settings: `AT+USPM?`
- Speaker gain: `AT+CLVL`
- Codec gain: `AT+UVGC`
- The LARA datasheets say to use `AT+UPAR=2,0,0` to play an audio test file from <audio_resource>, which I interpret to be a location on the LARA, but after much hair-pulling I think <audio_resource> is in fact empty.
- A more crude, but useful, audio test is to play a test tone: `AT+UTGN=1000,1000,100,0`
  The first parameter sets the frequency, which is 1000Hz here. The second parameter is the duration in milliseconds, which here is 1 second. It's not clear to me that durations longer than 1s are possible.

Network activity, registration, etc.

- To shut down cleanly, first issue `AT+CPWROFF`, then turn off the power.
- To deregister from the network: `AT+CFUN=0`
- To make sure the SIM isn't locked out and needing PIN entry, do `AT+CPIN?`. A response of `READY` means it's all good. A response of `SIM PIN` means it needs to be set.
- Display the LARA's IMEI number: `AT+GSN`
- Display the SIM card's ICCID: `AT+CCID`
- To check or confirm registration data and Radio Access Technology, do `AT+COPS?` The last digit indicates whether it's LTE or something else, where LTE is 7.
- A maybe better command then AT+COPS is `AT+URAT?` For this, 3=LTE and 7=LTE Cat M1

Audio CODEC configuration

The RUSP's MAX9860 Codec needs to be configured for first time use, which is done during the programming and quality verification at Sky's Edge. I'm not sure if it's possible for this to get accidentally reset.

- First time only, enable autoconfiguration of the Max9860 Codec: `AT+UEXTDCONF=0,1`
  This command saves to nonvolatile memory on the LARA.
- The Codec will be active the next time the LARA is turned on. Either power the RUSP off and back on, or issue the reset command: `AT+CFUN=16`

Other settings

- Set the module function to airplane mode: `AT+CFUN=4`
  Other functions are available and are described in the command reference.
- Set to normal mode: `AT+CFUN=1`

About the LARA Radio Technology

LARA-R6 versions:

- LARA-R6001: Global
- LARA-R6801: EMEA, APAC, JAPAN, LATAM
- LARA-R6401: N. America

Band frequencies:

Band	Uplink (MHz)	Downlink (MHz)
1	1920 – 1980	2110 – 2170
2	1850 – 1910	1930 – 1990
3	1710 – 1785	1805 – 1880
4	1710 – 1755	2110 – 2155
5	824 – 849	869 – 894
7	2500 – 2570	2620 – 2690
8	880 – 915	925 – 960
12	699 - 716	729 - 746
17	704 – 716	734 – 746
19	830 – 845	875 – 890
20	832 – 862	791 – 821
28	703 – 748	758 – 803

Comments: Band 5 is global. Band 7 is everywhere except Japan. Bands 1, 3, and 5 are everywhere except N. America.

SIM ordering and account activation notes for various carriers

Consumer Cellular:

- Make account online and order SIM card.
  - some basic functionalWill set up with email address and create a pin number (needed for phone service).
  - Will get an account number and phone number. Both will appear on the packing list that comes with the SIM card.
- Later, create online account for billing access.
  - Provide phone number and get password activiation link via email.
  - After logging in, nothing more to do to activate SIM card. Ready to use?
- SIM is already ready to be used in phone when it arrives. Put it in and make a call.

Known issues (firmware)

Problem: Incoming and outgoing call functionality is broken.
Workaround: Call functions can be demoed in certain regions using the AT/serial console (process is described elsewhere in this document)
Planned fix: Will fix firmware when development team gets network access.

Problem: When ISP not connected, debug console stops showing AT commands passings after a short time. Don't know why this happens.

Problem: Erratic behavior that leads to freezing when the phone is operated without either the SIM card or MicroSD card not inserted.
Workaround: Always have both a SIM and MicroSD card inserted, even if the SIM doesn't do anything.

Known issues (mechanical)

Problem: Dial return can stall.
Reason: Hole in reduction gear bearing pad is tight. Also, the Long Spacer and Mallet Clamp Collar should be 0.3mm taller.
Workaround: Trim the hole in the reduction gear bearing pad with a hobby knife. Add brass washers to the Long Spacer and Mallet Clamp Collar to act as spacers.
Planned fix: Increase height of spacers, possibly increasing depth of casing by .2-.4mm to accommodate (retool injection molding). Hole in Reduction gear bearing pad should also be retooled to be looser.

Problem: Lambda and Fn button sometimes get stuck depressed with casing fully tightened.
Reason: Not entirely clear, but extraneous tab features on button assembly is possible, especially on Fn button. Additional complicating factor may be lack of clearance due to depth of Front Casing.
Workaround: Manually trim Lambda and Fn buttons with hobby knife
Planned fix: New mold tooling for lower buttons. Increasing depth of casing is also possible.

Problem: Bell hammer assembly is annoying.
Reason: Hammer mount is combined with a critical spacer between the motherboard and daughterboard.
Workaround: May add spacers during assembly to help set Motherboard-Daughterboard separation.
Planned fix: New spacer design in progress.

Problem: Side window doesn't fully seat in rear casing.
Reason:
Workaround: It's arguably good enough as is.
Planned fix:

Problem: Epoxy needed to glue magnets to finger stop for hall sensor.
Workaround:
Planned fix: Next run of finger stops will include cutout for a single neodymium magnet. May still need a potting process, which would be done in-house.