

# Building a 128/256 SIM card bank with consumer tech on the cheap



Церебрална Романтика

Follow

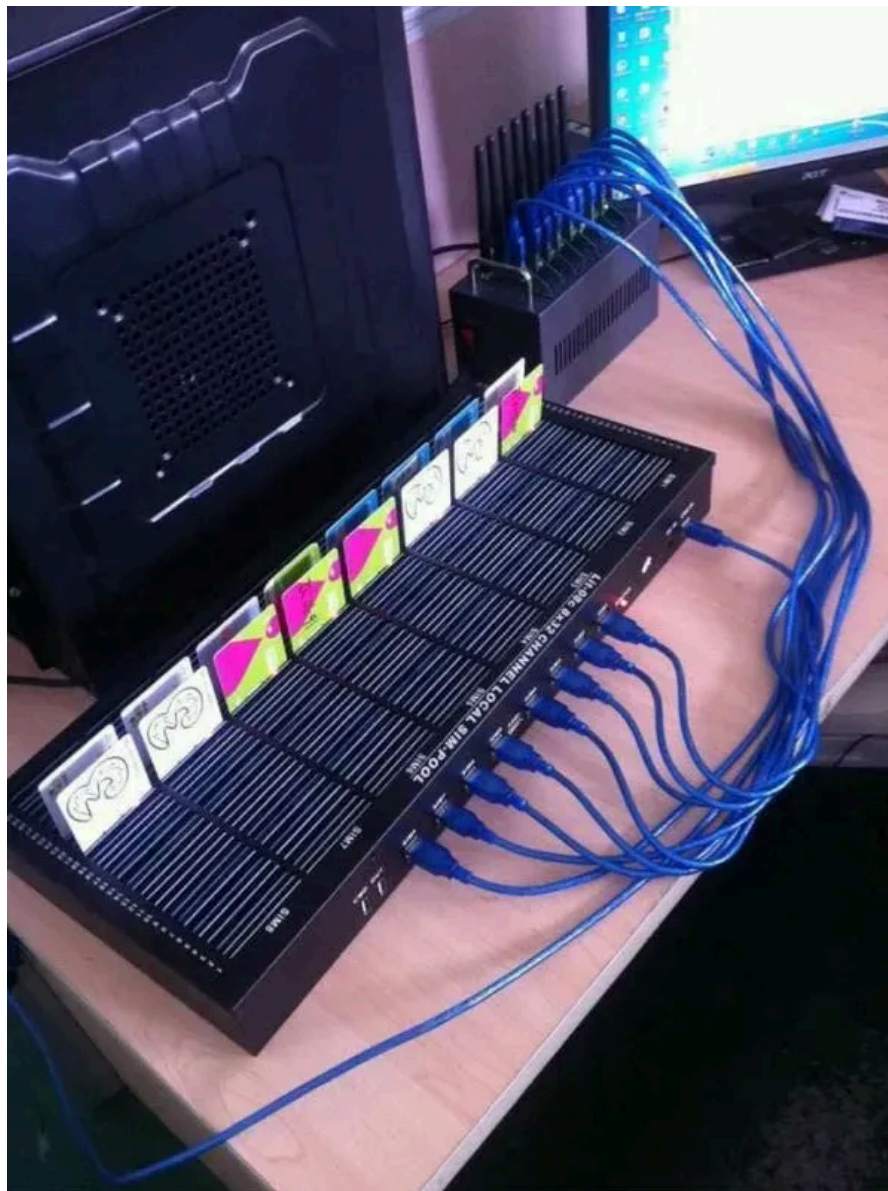
7 min read · Sep 7, 2023



18



4



A SIM pool + SIM bank assembled for less than 600\$. Credit: aliexpress.com

At a point several years prior I was compelled to build an internal company solution for managing multiple SIM cards for receiving SMS codes to our marketing department. Commercial solutions run into thousands easily and a decision was made to go down the economical path. Even though such a solution may be used for sending SMS messages, our case was receiving and routing the messages to our account management platform.

The reason to publish such a document is that a lot of guesswork and reverse engineering has gone into building a working solution. As such it would be magnificent if time on behalf of others might be saved with the findings described.

## Commercial vs consumer grade

The principle behind every SIM pool is the same — the vendor crams as many embedded(industrial) GSM modems as possible onto a chassis along with antennae and SIM slots. A commercial vendor will add an additional transport layer for routing the incoming SMS messages to an endpoint(HTTP/Database/SMTP) for easier integration. Often times the industrial modems are Wavecom or Siemens modules that expose serial ports for running AT commands and/or voice channels. Commercial solutions pack 16/32 or more such modems in a chassis compounding to a high cost of acquisition.

Consumer grade boards include 1/2/4/8/16 modems in a chassis with only a USB connection exposing the serial ports and very little documentation.

Open in app ↗

Sign up

Sign in

Medium

Search

Write





This is a consumer grade SIM pool. Credit : aliexpress.com

## The budget-friendly compromise

I did not want to settle for anything less than 128 SIM cards with a budget under 1000\$. After searching on aliexpress for a while I stumbled upon what vendors call a SIM bank. This is basically a switch with a dummy SIM card connector that you insert into your real modem slot and it extends the connection to an array of 16 physical slots. The drawback is that it is a switch, not a multiplexer. As such only 1 of the 16 slots in the SIM bank is online for a modem port at a given moment. The SIM bank exposes a serial interface via USB that lets you change the port assignments on demand. As one may guess, the documentation for this is scarce if not obscure. So the compromise is availability. One would have to iterate the slots on the SIM bank and look for activity on every slot. This is easy with a few quirks.

## Software

There is a Windows application called SMSCaster that is marked compatible with the SIM pools and possibly the SIM banks. It is on a commercial license

and offers a limited feature set. It provided a basic glimpse into how the pool is working but quickly got discarded as a realistic candidate for SMS management.

The wonderful world of FOSS has provided SMSTools3 SMS Server (<http://smstools3.kekekasvi.com/>). It is configurable and very flexible in terms of managing the modems and the message data. Linux is the platform of choice for me for hosting internal applications. In this instance Raspbian on Raspberry Pi 1 model B+. It was hosted on a RPi 3 initially which I dedicated to another project so switched to the 1 model B+.

I have included a slightly modified smstools3 version in the [github.com repo](#) for this project. As noted in the README.md, it only activates a particular alarm handler that sends a request to a backend that a SIM card is successfully checked for messages( even if no new messages have been received )

You would wanna do :

```
mkdir /home/socks

mkdir /run/smstools/
chown socks /run/smstools

cd /home/socks/

git clone https://github.com/sertys3/sms-receiver.git .

cd smstools3

sudo make && sudo make install

cp /home/socks/smsd.conf/etc/
```

## Hardware

1. A Raspberry Pi ( 1 model B+ in my case) with Raspbian installed
2. 16-port SIM pool — <https://www.aliexpress.com/item/32701584178.html>
3. SIM bank — <https://www.aliexpress.com/item/32947688074.html>

## Drivers

The SIM pool requires a kernel driver that exposes the different modem devices properly. They otherwise show as cdc\_acm / usb2serial devices, but communication is not possible.

Get your driver here:

### XR21V1410

The XR21V1410 is an enhanced 1-channel Universal Asynchronous Receiver and Transmitter (UART) with a USB interface. The...

[www.maxlinear.com](http://www.maxlinear.com)

In my case it was :

Software: Drivers Linux 3.6.x and Newer 1D September 2021 29.9 KB

but version C as it was the latest back then. In order to build you have to :

```
sudo apt-get install build-essential linux-headers screen
```

decompress the archive, go to the decompressed path and

```
make && sudo make install
```

Hopefully that would compile your driver and install it into the modules directory.

The driver is of the CDC class and one would have to blacklist the original cdc\_acm driver to make sure this one loads instead :

```
sudo echo "blacklist cdc-acm" > /etc/modprobe.d/blacklist-cdc.conf
```

After a reboot you will find the corresponding ttyXRUSB devices in your /dev/

```
root@raspberrypi:/# ls /dev/ttyXRUSB*
/dev/ttyXRUSB0 /dev/ttyXRUSB10 /dev/ttyXRUSB12 /dev/ttyXRUSB14
/dev/ttyXRUSB2 /dev/ttyXRUSB4 /dev/ttyXRUSB6 /dev/ttyXRUSB8
/dev/ttyXRUSB1 /dev/ttyXRUSB11 /dev/ttyXRUSB13 /dev/ttyXRUSB15
/dev/ttyXRUSB3 /dev/ttyXRUSB5 /dev/ttyXRUSB7 /dev/ttyXRUSB9
```

## SIM bank switching explained

The SIM bank itself has one USB port to be connected to your host device. It shows up as :

```
Bus 001 Device 005: ID 0403:6001 Future Technology Devices International, Ltd  
FT232 Serial (UART) IC
```

Which is just a serial port exposed on `/dev/ttyUSB0`. It responds to custom AT commands — basically `AT+CWSIM` to check the device status and `AT+SWIT[01-16]-[SLOT]` to switch a port to the desired slot. I do not even remember where the documentation to that was found but certainly was not easily available.

The published [pool\\_switch\\_2.pl](#) script iterates over the ports and changes slots every 200 seconds. Keep in mind that switching is dumb and the modems do not receive a notification that the SIM card has effectively changed.

## SMSTools3 configuration

```
[Port1]  
#init = AT+CMEE=1  
pre_init = no  
device = /dev/ttyXRUSB0  
init = AT+CPMS="SM"  
incoming = yes  
mode = new  
baudrate = 115200  
rtscts = yes  
cs_convert = yes  
report = yes  
memory_start = 1  
check_memory_method = 1  
primary_memory = SM  
pin = ignore  
check_sim = yes  
check_sim_cmd = AT+CUSD=1,"*123#",15  
check_sim_reset = AT+CFUN=1,1  
check_sim_retries = forever  
check_sim_wait = 2
```

```
device_open_retries = -1
#give_up_on_io_error = no
eventhandler = /home/socks/pool_forward_sms.pl
ignore_unexpected_input = +CME ERROR: 10
detect_unexpected_input = no
#pin = 0000
#pinsleeptime = 2
queues = Port1
```

---

This is what port configuration looks like in the bundled [smsd.conf](#). The smsd is spawning threads that run AT commands on the modem devices for each port at regular intervals(configured in the delaytime directive in the global section of smsd.conf [10 seconds in this case] ). Since switching is dumb on the SIM bank we need to make sure every check has a network-registered SIM card. The AT+CPMS="SM" command is trying to set the Message Store to SM. But in case the SIM card is not working or freshly switched, the modem returns an error. smsd will try to reset the modem with AT+CFUN=1,1.

The [smsd.conf](#) file has a USSD command of \*123# that it executes on a SIM card after re-initializing the modem with AT+CFUN=1,1 . This has proven valuable in waking up the SIM cards to receive messages upon reset. Otherwise messages are delivered often times after the 200 seconds checking interval. Put a USSD command that your carrier understands for maximum efficiency.

## Putting it all together

Once everything is connected one would need to start the scripts to iterate over the ports and receive the messages. Luckily SMSTools3 is simple to configure. The provided smsd.conf file in the repository lists all 16 ports on the SIM pool yet only 8 are active in our case. If the need arises we may add another 8x16-slot SIM bank and have 256 SIM cards hosted in total.

There's the peculiarity of installing your perl modules from CPAN for the scripts provided :

---

```
perl -MCPAN -e install Device::SerialPort

perl -MCPAN -e install LWP::UserAgent
```

---

Then it's a matter of doing

```
/etc/init.d/sms3 start
```

This may or may not be created by the make install of smstools3. Otherwise start manually with

```
smsd -c /etc/smsd.conf
```

```
screen /home/socks/pool_switch_2.pl
```

Now smstools3 should be checking enabled ports for SMS messages every 10 seconds and pool\_switch\_2.pl will rotate the switch slots every 200 seconds.

In case an SMS is received, it will be handled by pool\_forward\_sms.pl, which will send a POST request to a web server of your choice. Do not forget to change the \$POST\_ENDPOINT variable.

## Slot fast forward

Since SMS codes are time sensitive, pool\_switch\_2.pl will look for /run/smstools/[PORT]\_next files and switch to the contained slot number upon iteration. The mechanism to write those files in the backend was implemented in our account management platform. So a user expecting an SMS on Port 3, Slot 10 will trigger a check and a /run/smstools/3\_next file will be created with string content "10". And within a maximum of 200 seconds his SMS will be pushed back to him and rendered in the interface.

## A note on SIM Bank location

The current SIM card location is derived from the values in /run/smstools/ which pool\_switch\_2.pl updates upon switching.

```
root@raspberrypi:/home/pi# ls /run/smstools/
```

```
1_current 2_current 3_current 4_current 5_current 6_current 7_current 8_current
```

As such there's a 'location' parameter in the POST request with a string value of [PORT]x[SLOT]. And the ingesting endpoint maps the message to the correct SIM card number based on that parameter.

## The alarmhandler script



In the global section of the `smsd.conf` you will find

```
#alarmhandler = /home/socks/pool_alarm.pl
```

which receives all alarms from `smsd` and looks for a particular one indicating that a check on a SIM card has been successful. It then sends a POST request to an endpoint set in `$POST_ENDPOINT` in `pool_alarm.pl`. This alarm is only available in the modified `smstools3` version bundled.

Once this got deployed to a RPi 1 the script was killing the system with numerous Perl interpretator spawns. So my best man re-wrote it in Go and had it compiled. I would happily share the source to that upon request.

The directive is now commented by default as to ease up initial deployment.

## Goal achieved

This is a solution that we've had in production for several years and with a total purchase price of <600\$. It clearly may be deemed a success and it would be great to see it replicated. Happy SMSing!

[Sms Marketing](#)[Sms](#)[Sms Gateway](#)[Raspberry Pi](#)[Development](#)

Written by Церебрална Романтика

54 Followers · 14 Following

[Follow](#)

## Responses (4)



Write a response

What are your thoughts?



Phdjamel  
Dec 17, 2024



Wow, that's amazing bro. I'm supposed to work on a project that has a lot of similarities with yours. The link for the sim bank is not working. Could you please help me with this. Thank you for sharing your awesome work



[Reply](#)



Michael Nomo  
Dec 17, 2023



I cant find the product for the price the aliexpress links are not available. And I cant find the sim bank for 200 bucks. I only find the same Sim bank for 1300 bucks which is ridiculous



1 reply

[Reply](#)



ouifi  
Dec 1, 2023



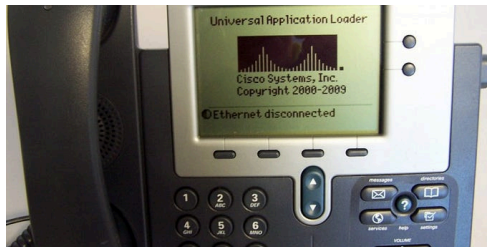
Curious to know are still using it or have you updated to something else ?



[Reply](#)

[See all responses](#)

## More from Церебрална Романтика



Церебрална Романтика

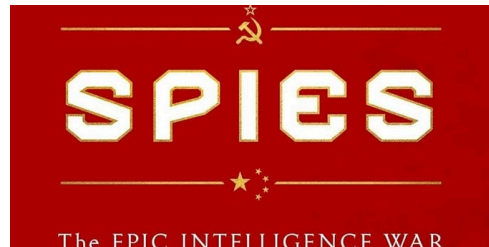
### Running a PBX(FreeSwitch) in Docker as a house intercom

My family recently moved to a three-story flat which was already set and furnished. After...

Jan 23, 2023



63



Церебрална Романтика

### Why did the CIA/NSA create cryptocurrency?

A very, very speculative piece on cryptocurrency and intelligence gathering.

Apr 6, 2024

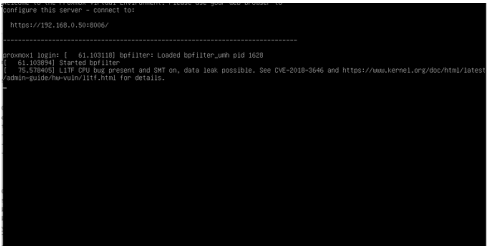



86



3






 Церебрална Романтика

Running a multiple instance Mautic 4 in Docker

Mautic is a great open-source marketing management and automation suite that I ha...

Mar 8, 2024  8 

 Церебрална Романтика

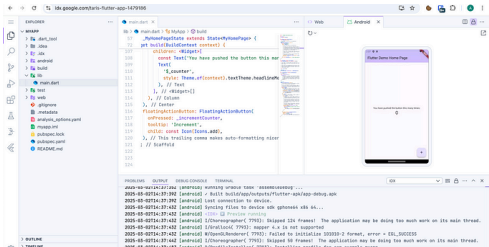
Deploying ProxMox on DL360 G6/G7 with ZFS and NFS root


This is a guide to myself and other who would like to deploy a VM orchestrator on their...

Jun 27, 2023  3  1 

See all from Церебрална Романтика

Recommended from Medium

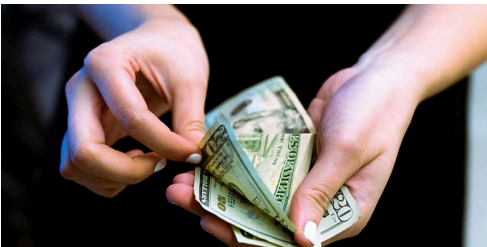



 In Coding Beauty by Tari Ibaba

This new IDE from Google is an absolute game changer

This new IDE from Google is seriously revolutionary.

★ Mar 11  5K  296 

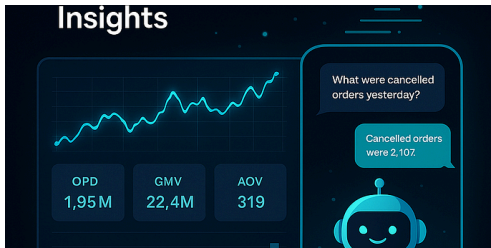


 In Learn AI for Profit by Nipuna Maduranga

You Can Make Money With AI Without Quitting Your Job

I'm doing it, 2 hours a day

★ Mar 24  8.5K  391 

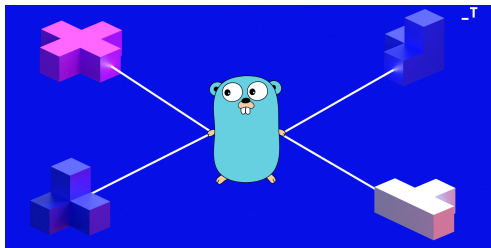


Sundaram Dubey

### Bringing AI to Apache Superset: Integrating Chatbots for Real-Tim...

In today's data-driven world, static dashboards are no longer enough. Business...

★ Apr 12 🖱️ 2 📌<sup>+</sup>



Debugged Thoughts

### Why Big Tech Is Quietly Abandoning Golang

In the early 2010s, Go (Golang) was hailed as the future of systems programming. It was...

★ Apr 21 🖱️ 520 💬 36 📌<sup>+</sup>



In Predict by Will Lockett

### Tesla Is Already Dead

And Musk knows it.

★ Apr 24 🖱️ 6.5K 💬 176 📌<sup>+</sup>



Henrik Hauge Bjørnskov

### TIL: I made a rubygem implementing Cloudflare Turnstil...

For the last couple of weeks I have been fighting against fake signups on some of my...

Feb 27 🖱️ 5 💬 1 📌<sup>+</sup>

See more recommendations