

Embox российская ОС для цифровых устройств

Антон Бондарев

OS Day, 22 июня 2022

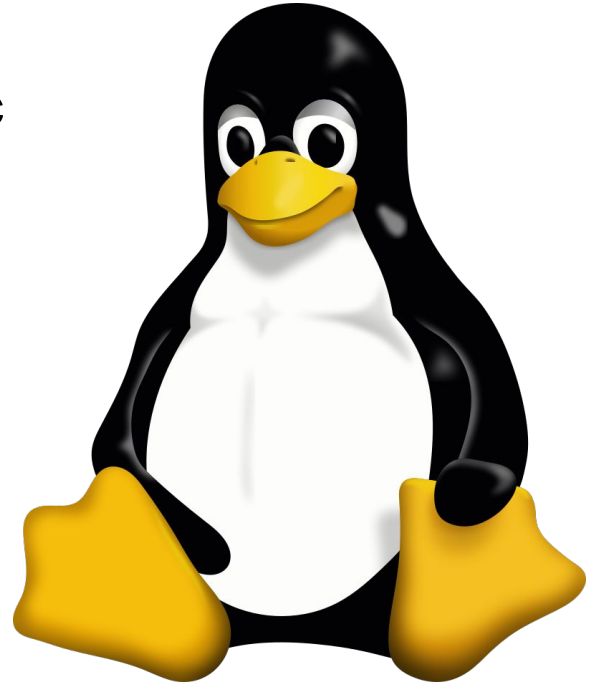
Embox

Embox — свободная операционная система реального времени (RTOS), разрабатываемая для встроенных систем.

Основная идея использование ПО **Линукс** в более безопасном и детерминированном, менее ресурсоемком и энергопотребляющем окружении

Linux

- Величайшая ОС в мире :)
 - Огромное количество ПО на любой вкус
 - Потрясающая поддержка аппаратуры
-
- Но Линукс универсален, поэтому:
 - Требуется мощная аппаратура
 - Трудно обеспечить реал-тайм
 - Подвержен вредоносному ПО



Embox

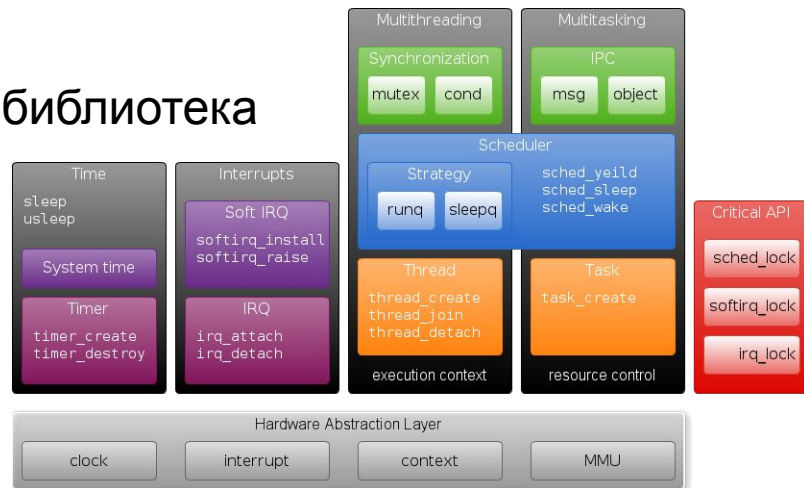
- “Недавно портирован за три недели проект, который разрабатывался два года для Линуха, потом с разными спотыкачами пытались портировать пол-года на NuttX и ещё на пару систем для встроенных решений, которые провозгласили POSIX-совместимость. Проект реализован на Си++.”
- “Для меня было принципиальным моментом когда мне показали обычное ПО на микроконтроллере”
- “Ключевым моментом является, что не нужно сильно переучивать разработчиков”

Средства разработки

- Используются распространенные и открытые средства разработки (gcc, gdb, Eclipse, VSCode, openOCD, ...)
- Ядро написано на **языке C** (gnu99).
- Используется собственная **стандартная библиотека**
- Поддержка распространенных языков: C++, Lua, JS, Tcl, Ruby, Perl, ...

Embox

- Архитектуры ARM, x86, RISC-V, MIPS, SPARC, ...
- Полнофункциональная современная ОС
 - Ядро
 - Вытесняющая многозадачность
 - IPC
 - Управление памятью
 - ...
 - Файловая система на VFS
 - Конфигурируемая стандартная библиотека
 - Собственный TCP/IP
 - Графическая подсистема
 - ...



Простой запуск и отладка

```
XTerm - anton:~/workspace/embox-demo-trunk
anton@twister:~/workspace/embox-demo-trunk$ ./scripts/qemu/auto_qemu
AUTOQEMU_ARCH: arm (guessed)
AUTOQEMU_MEM: 512 (guessed)
AUTOQEMU_NICS: lan9118 (guessed)
AUTOQEMU_NICS_CONFIG: (got)
AUTOQEMU_KVM_ARG: (guessed)
Making uImage...
```

```
source  Common
1
connection
localhost
1234
XTerm - anton:~/workspace/embox-demo-trunk
> route add 10.0.2.0 netmask 255.255.255.0 eth0
> route add default gw 10.0.2.10 eth0
Welcome to Embox and have a lot of fun!
embox>QEMU: Terminated
ioctl(TUNSETIFF): Device or resource busy
./scripts/qemu/stop_script: could not launch network script
anton@twister:~/workspace/embox-demo-trunk$ ./scripts/qemu/auto_qemu -s -S
AUTOQEMU_ARCH: arm (guessed)
AUTOQEMU_MEM: 512 (guessed)
AUTOQEMU_NICS: lan9118 (guessed)
AUTOQEMU_NICS_CONFIG: (got)
AUTOQEMU_KVM_ARG: (guessed)
Making uImage...
32768+0 записей получено
32768+0 записей отправлено
скопировано 32768 байт (33 kB), 0,0663855 с, 494 kB/c
Making nand image...
AUTOQEMU_LOAD_ARG: -mtdblock beagle.nand.img (guessed)
AUTOQEMU_NOGRAPHIC_ARG: -nographic (guessed)
sudo qemu-system-arm -M qemu -mtdblock beagle.nand.img -m 512 -net nic,dlan=
```

Настольная отладка

embox-demo-trunk convertedConfig [C/C++ Remote Application]

embox
Thread [1] 1 (CPU#0 [running]) (Suspended : Breakpoint)

```
main() at route.c:59 0x81017648
├─ cmd_exec() at core.c:35 0x810489bc
├─ diag_shell_exec() at shell.c:111 0x810100dc
├─ shell_exec() at shell.h:54 0x8100fba0
├─ run_script() at start_script.c:49 0x8100fc78
├─ unit_mod_enable() at unit.c:35 0x810345f8
├─ mod_enable() at core.c:162 0x810344c
├─ runlevel_set() at runlevel.c:82 0x8103444c
├─ init() at init.c:57 0x81008370
└─ kernel_start() at init.c:27 0x81008328
```

mods.config route.c

```
47     printf("\n");
48 }
49 return 0;
50 }
51 }
52
53 static void print_help() {
54     printf("route [-hn] [-A family] [add/del] <target> [gw <Gw>]
55           [netmask <Nm>] [dev] if)\n");
56 }
57
58 int main(int argc, char **argv) {
59     struct net_device *netdev = NULL;
60     enum route_action rt_act;
61     in_addr_t target, netmask = INADDR_ANY, gw = INADDR_ANY;
62     enum target_type tar_t = NET;
63     int d_flags = 0x0, i;
64     char **tmp;
65
66     for (tmp = argv; *tmp != NULL; tmp++) {
67         if (!strcmp(*tmp, ".net") || !strcmp(*tmp, ".host")) {
```

Name	Type
argc	int
argv	char **
netdev	struct net_device *
rt_act	enum route_action
target	in_addr_t

```
embox - action //workspace/embox-demo-trunk
test: running embox.test.util.bit_test ..... done
test: running embox.test.util.array_test ..... done
test: running embox.test.stdlib.qsort_test ..... done
test: running embox.test.stdlib.bsearch_test ..... done
test: running embox.test.posix.sleep_test ..... done
runlevel: init level is 1
unit: initializing embox.fs.driver.repo: done
unit: initializing embox.fs.node: done
unit: initializing embox.mem.phymem: start=0x85ae000, end=0xa1009000, size=45440614
done
unit: initializing embox.fs.buffer_cache: done
unit: initializing embox.net.neighbour: done
unit: initializing embox.driver.block: done
unit: initializing embox.fs.rootfs: initfs_mount: unpack initinitfs at 0x810a82b0 in
to /
done
unit: initializing embox.net.tcp: done
unit: initializing embox.cmd.shell: done
unit: initializing embox.driver.net.loopback: done
runlevel: init level is 2
unit: initializing embox.init.start_script:
Started shell [ ] on device [ ]
loading start scripts:
> ifconfig lo 127.0.0.1 netmask 255.0.0.0 up
> route add 127.0.0.0 netmask 255.0.0.0 to
```

Console Tasks Problems Executables Memory

embox-demo-trunk convertedConfig [C/C++ Remote Application] gdb

Embox

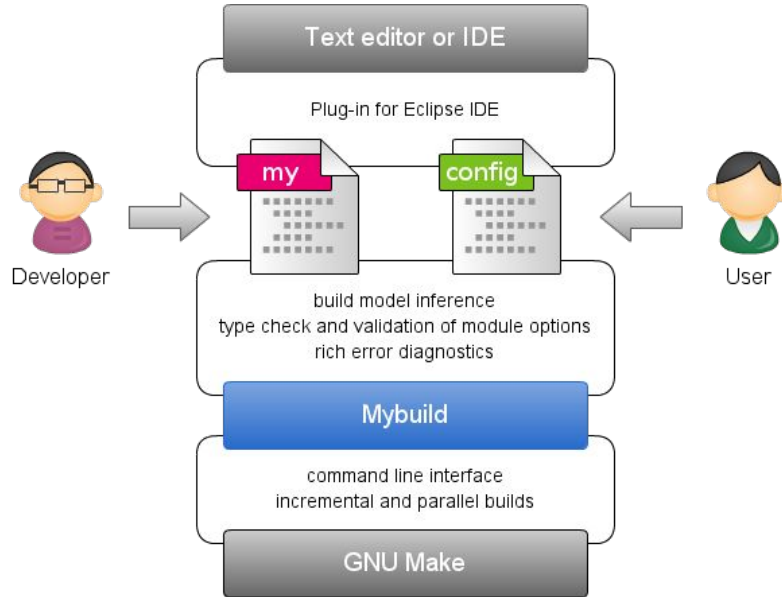
Операционная система под конкретную задачу, основанная на **специальном DSL языке** и использующая:

- Статическую информацию о задачах устройства
- Статическую конфигурацию системы
- Статический анализ зависимостей
- Статическую проверку параметров системы

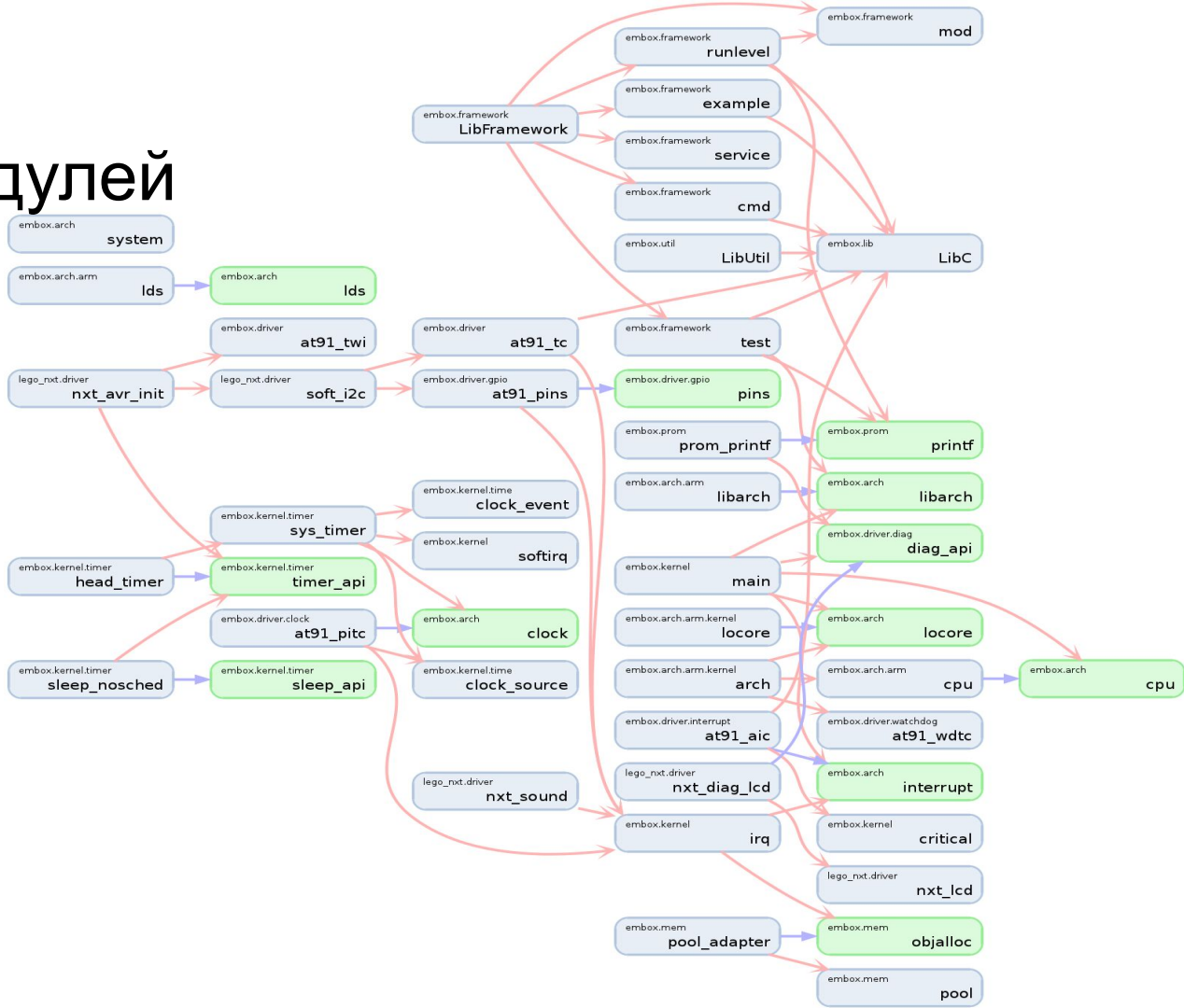
Embox build system:

KBuild + OpenEmbedded + DevTree

Embox. Процесс сборки



Граф модулей



Embox: Hello world (source)

```
#include <stdio.h>
```

```
int main(int argc, char **argv) {
```

```
    printf("Hello, world!\n");
```

```
}
```

Embox: Hello world (Mybuild)

```
package embox.cmd
```

```
@AutoCmd
```

```
@Cmd(name = "hello_world", help="First Embox application")
```

```
module hello_world {
```

```
    source "hello_world.c"
```

```
}
```

Embox: Система Сборки (mods.conf)

```
include embox.kernel.task.resource.idesc_table(idesc_table_size=32)
```

```
include embox.kernel.task.resource.sig_table(sig_table_size=20)
```

```
include embox.kernel.task.resource.env(env_per_task=4,env_str_len=64)
```

```
include embox.kernel.thread.thread_local_none
```

```
include embox.kernel.thread.thread_cancel_disable
```

```
include embox.kernel.thread.signal.siginfoq(siginfo_pool_sz=8)
```

```
@Runlevel(2) include embox.kernel.sched.strategy.priority_based
```

...

```
include embox.cmd.hello_world
```

Embox. Окружение для разработки

- Кросс-компилятор (arm-none-eabi-(gcc))
- Стандартная библиотека (glibc, musl, newlib, ...)
- Стартовый код
- Линкер скрипт
- Прикладные библиотеки

Embox: Mybuild (extbuild)

```
@Build(stage=2,script="$(EXTERNAL_MAKE)")
@Cmd(name = "nano",
...
...
    ")
module nano {
    source "^BUILD/extbld/^MOD_PATH/install/nano.o"
    @NoRuntime depends embox.compat.posix.regex
    depends embox.compat.posix.LibCurses
}
```


Embox: Mybuild (extbuild)

```
PKG_NAME := nano
```

```
PKG_VER := 2.2.6
```

```
PKG_SOURCES := http://www.nano-editor.org/dist/v2.2/$(PKG_NAME)-$(PKG_VER).tar.gz
```

```
...
```

```
$(CONFIGURE) :
```

```
export EMBOX_GCC_LINK=full; \
```

```
cd $(PKG_SOURCE_DIR) && ( \
```

```
    ./configure --host=$(AUTOCONF_TARGET_TRIPLET) \
```

```
        --target=$(AUTOCONF_TARGET_TRIPLET) \
```

```
...
```

```
)
```

Embox: Mybuild (extbuild)

`$(BUILD)` :

```
cd $(PKG_SOURCE_DIR) && ( \
```

```
    $(MAKE) MAKEFLAGS='$(EMBOX_IMPORTED_MAKEFLAGS)'; \
```

```
)
```

```
touch $@
```

`$(INSTALL)` :

```
cp $(PKG_SOURCE_DIR)/src/nano $(PKG_INSTALL_DIR)/nano.o
```

```
touch $@
```

Embox: Mybuild (BuildDepends)

```
@BuildDepends(gcc_build)
```

```
@BuildArtifactPath(cppflags_before="-I$(abspath  
$(EXTERNAL_BUILD_DIR))/third_party/gcc/gcc_build/install/_target/include/c++/_gcc_version  
-I$(abspath  
$(EXTERNAL_BUILD_DIR))/third_party/gcc/gcc_build/install/_target/include/c++/_gcc_version/_target  
")
```

```
static module libstdcxx extends embox.lib.libstdcxx {
```

```
    @AddPrefix("^BUILD/extbld/third_party/gcc/gcc_build/install/libs")
```

```
    source "libstdc++.a"
```

```
    @NoRuntime depends gcc_build
```

```
}
```

Embox: Mybuild (BuildDepends)

```
@AutoCmd
```

```
@Cmd(name="stl_demo_sort1",  
      help="example for stl algo sort",  
      man="")
```

```
@BuildDepends(embox.lib.libstdcxx)
```

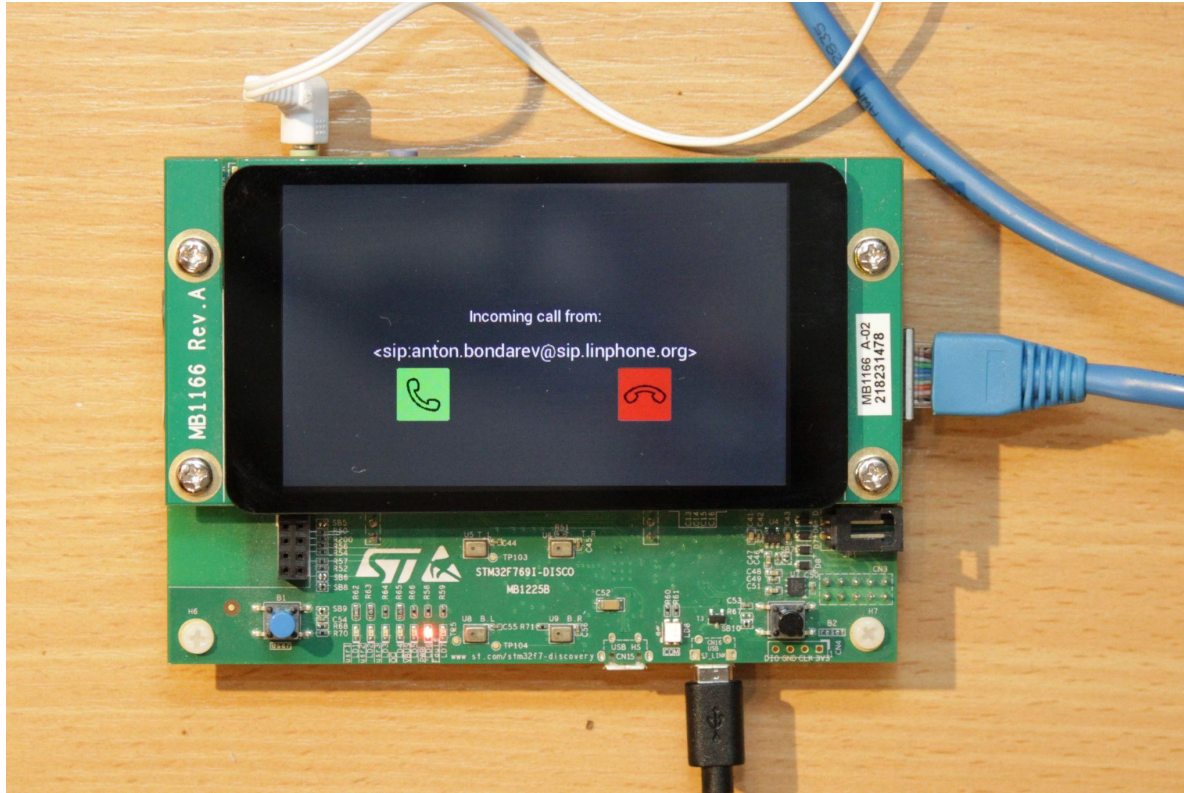
```
@Build(stage=2,script="true")
```

```
module stl_demo_sort1 {  
    source "sort1.cpp"
```

```
@NoRuntime depends embox.lib.libstdcxx
```

```
}
```

Embox: PJSIP



PJSIP на Linux

```
$ ./configure \
```

```
  --prefix=$PREFIX \
```

```
  --disable-l16-codec \
```

```
  --disable-ilbc-codec \
```

```
  --disable-speex-codec \
```

```
  --disable-speex-aec \
```

```
  --disable-gsm-codec \
```

```
  --disable-g722-codec \
```

```
  --disable-g7221-codec \
```

```
  --disable-libyuv \
```

```
  --disable-libwebrtc
```

```
$ make dep && make
```

PJSIP на Linux

- Используем стандартный пример
- Модификация вынос параметров для аккаунта

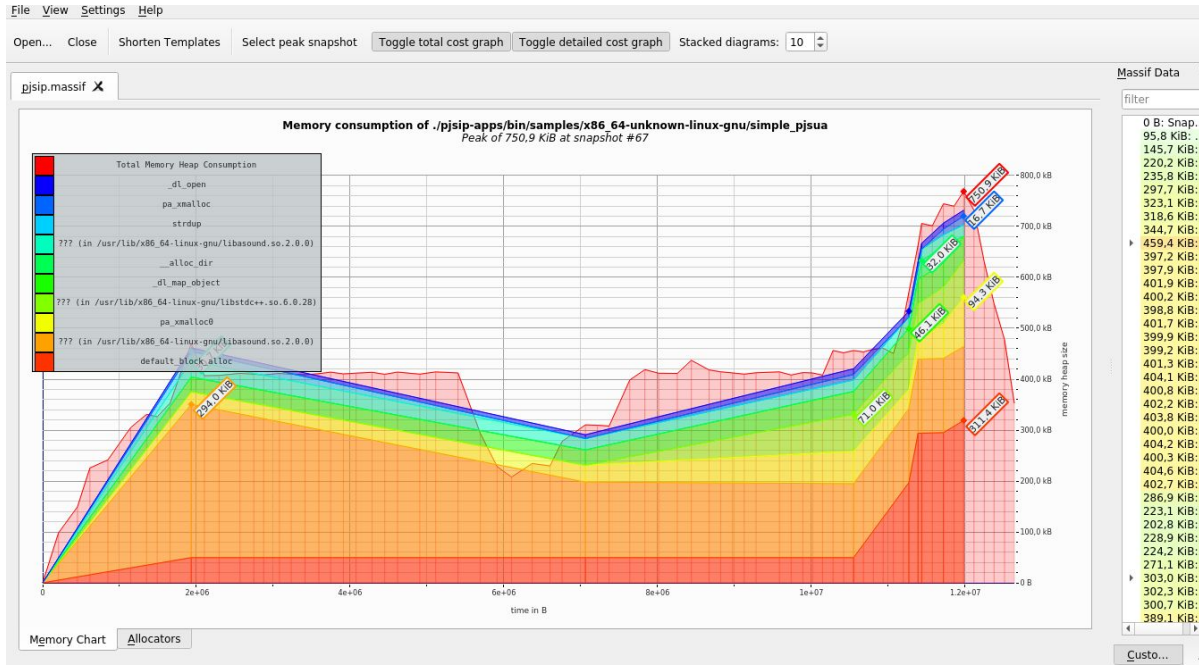
```
#define SIP_DOMAIN      <sip_domain>
```

```
#define SIP_USER        <sip_user>
```

```
#define SIP_PASSWD      <sip_passwd>
```

valgrind on Linux

- `valgrind --tool=massif --time-unit=B --massif-out-file=pjsip.massif`
- `./pjsip-apps/bin/samples/x86_64-unknown-linux-gnu/simple_pjsua`
- `massif-visualizer pjsip.massif`



Embox: PJSIP

```
DISABLE_FEATURES := \
```

```
l16-codec \
```

```
ilbc-codec \
```

```
speex-codec \
```

```
speex-aec \
```

```
gsm-codec \
```

```
g722-codec \
```

```
g7221-codec \
```

```
libyuv \
```

```
libwebrtc
```

```
./configure \
```

```
CC=$(EMBOX_GCC) \
```

```
CXX=$(EMBOX_GXX) \
```

```
--host=$(AUTOCONF_TARGET_TRIPLET) \
```

```
--target=$(AUTOCONF_TARGET_TRIPLET) \
```

```
--prefix=$(PJSIP_INSTALL_DIR) \
```

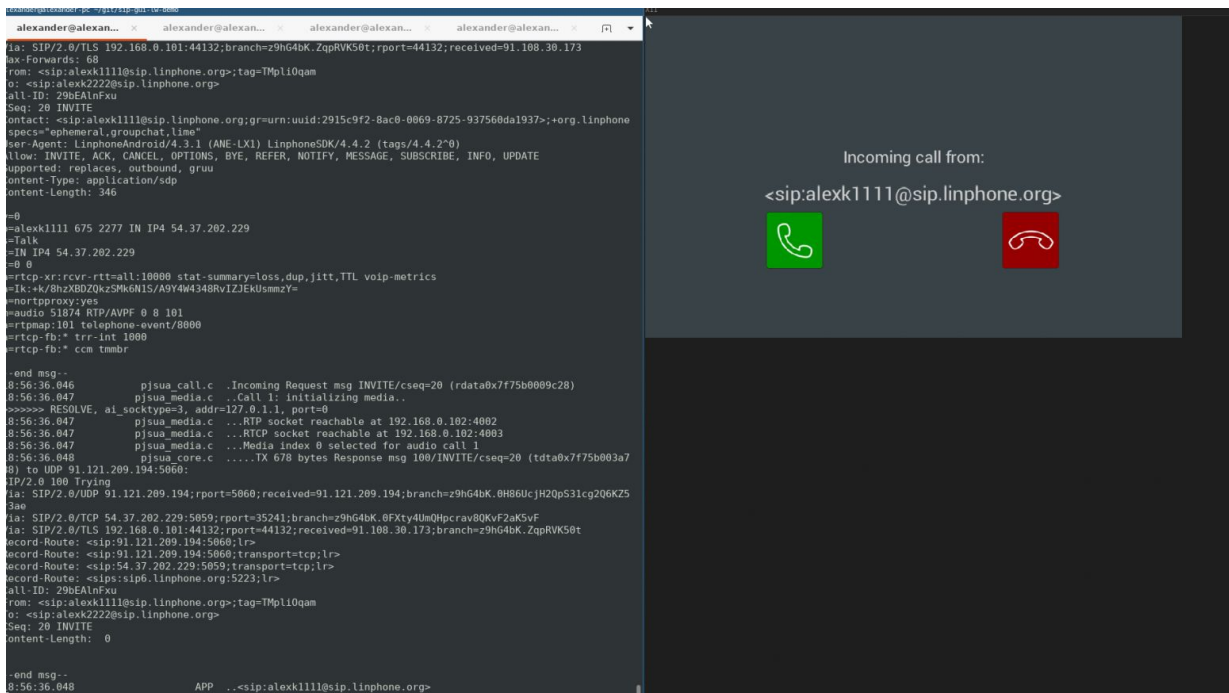
```
$(DISABLE_FEATURES:%=--disable-%) \
```

```
--with-external-pa; \
```

```
)
```

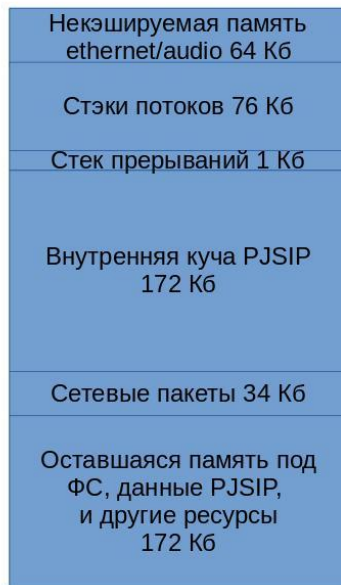
Embox: PJSIP on QEMU

- Добавили графический интерфейс
- Запустили на qemu

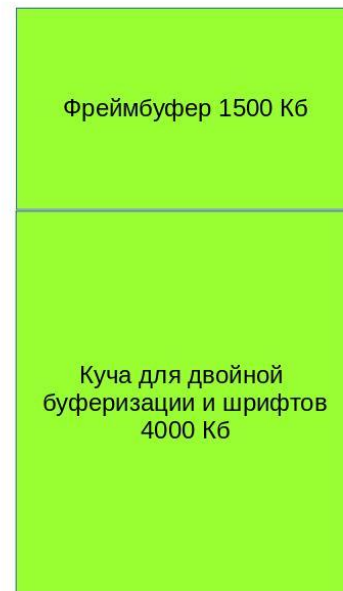


Embox: PJSIP на плате

- карта памяти

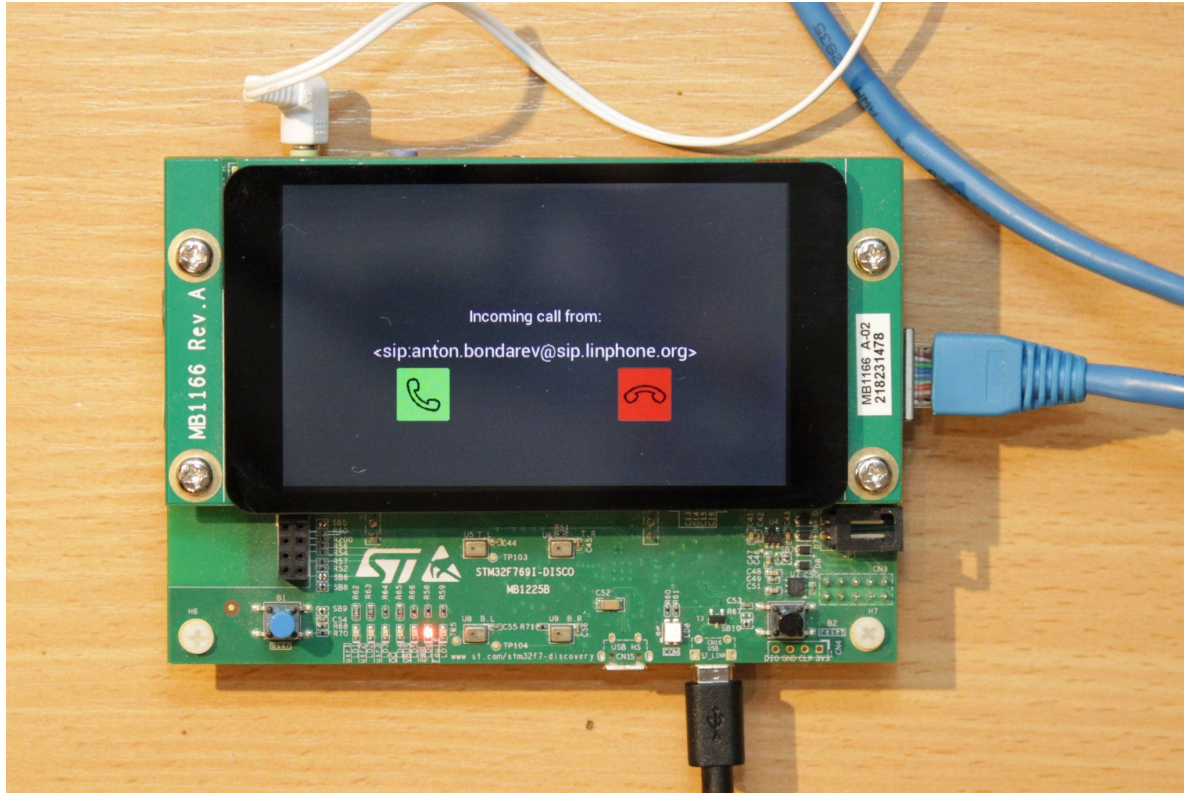


RAM 512 Kb

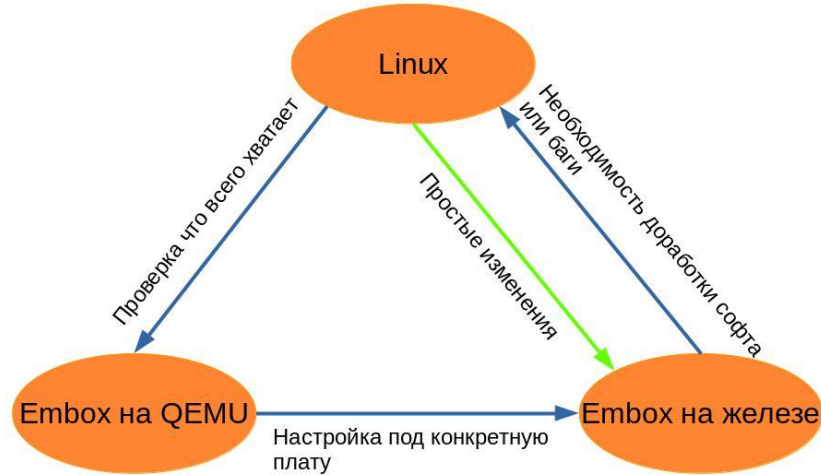


SDRAM 16 Mb

Embox: PJSIP



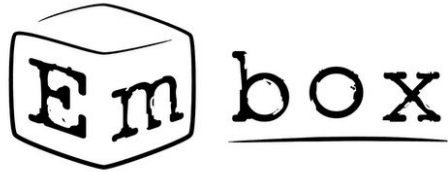
Процесс разработки под Embox



Заключение

- Наиболее развитая экосистема у ОС Linux
- Экосистема должна обеспечивать удобство, качество и скорость разработки под ОС.
- Embox обеспечивает эти свойства, за счет широкого использования экосистемы Linux

Contacts



Essential toolbox for embedded development

youtube

- <https://www.youtube.com/@embox-rtos>

Embox Project Homepage

- <http://embox.github.io/>

Telegram chat

- https://t.me/embox_chat