Embedded with Go: from an AWK prototype to a gokrazy appliance

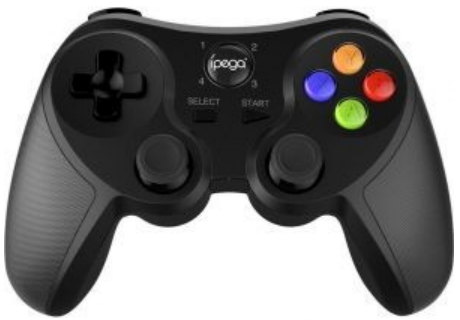
FOSDEM 2019

Whoami

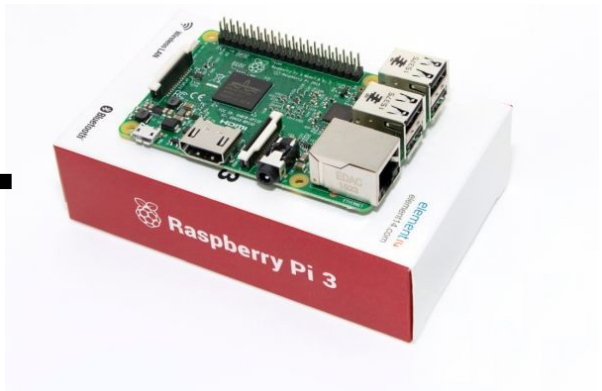
- Anisse Astier
- Father
- Linux Embedded Engineer

https://anisse.astier.eu

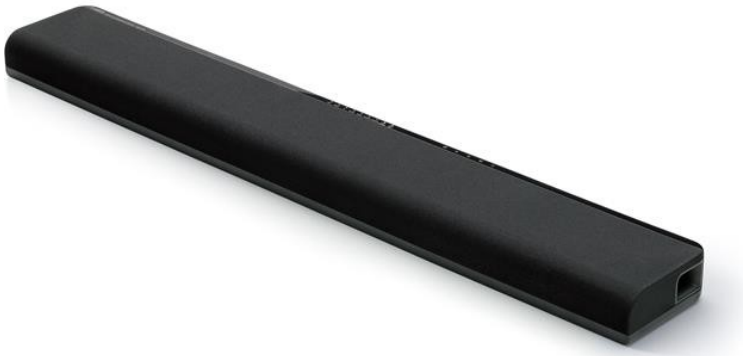
Toy explorations



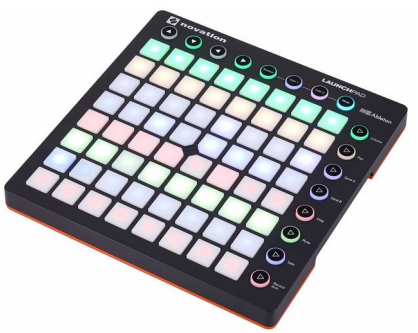
+



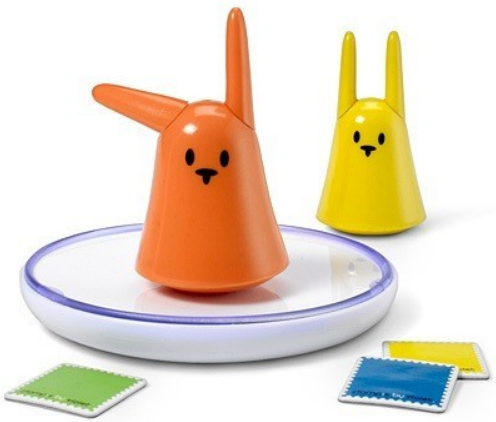
+



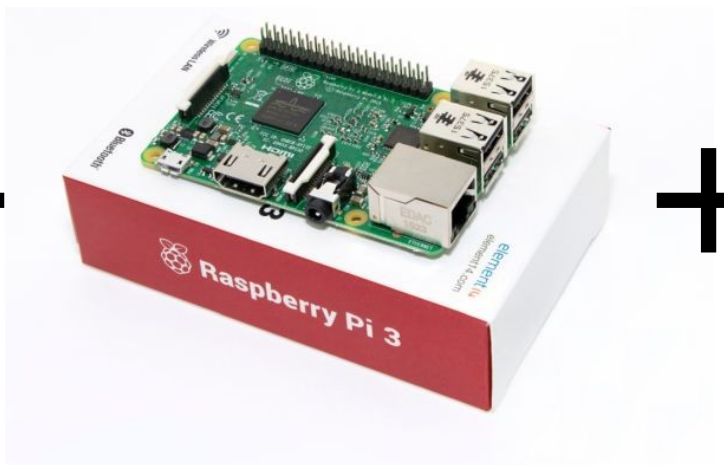
=



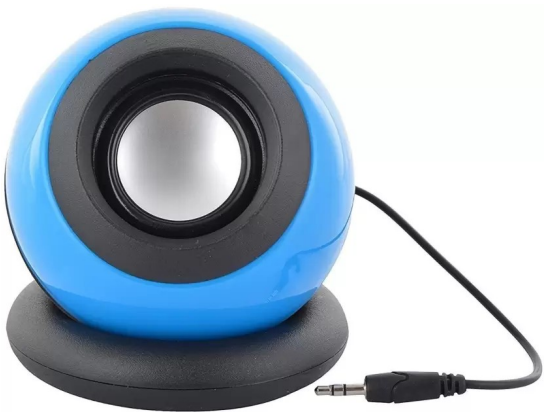
Toy explorations



+



+



= ?

Awk

```
#!/bin/bash
hexdump -C /dev/hidraw0 | awk '
/02 01 00 00 08 d0 02 1a / { system("mpg321 -q file1.mp3 & "); }
/02 01 00 00 04 3f d7 5f / { system("mpg321 -q dir1/*.mp3 & "); }
/  02 02 00 00 0. |01 05 00 00 00/ { system("killall -q mpg321"); }
'
```

Go

- goroutines
- channels
- event loop
- reliable process control with context cancellation

goroutines

```
t := make([]byte, TAGLEN)
c := make(chan string)
go player(c)
for {
    _, err := io.ReadAtLeast(f,
                  t, TAGLEN)
    if err != nil {
        //...
    }
    c <- hex.EncodeToString(t)
}
```

```
w := make(chan error)
for {
    select {
        case s := <-c:
            switch {
            case s == START_TAG:
                 stop = play(w)
            case s == STOP_TAG:
                 stop()
            }
        case err := <-w:
                //...
    }
}
```

Command cancellation

```
ctx, stop := context.WithCancel(context.Background())

cmd := exec.CommandContext(ctx, "mpg321",
playlist[current])

…


stop()
```

Cross-compilation

- GOOS=linux GOARCH=arm go build
- CGO_ENABLED=0

Gokrazy

- Simplified linux system for go appliances written by Michael Stapelberg
- Motivation :
  - @stapelberg spends way more time on C software and their various issues than he would like. Hence, he is going Go-only where feasible.


DOES IT SPARK JOY?

Gokrazy

- Initial target rpi3, now targets x86 for router7
- Router7 pure-go router appliance
- beatbox, this talks's appliance

Gokrazy tour: packages

- gokr-packer : image builder
- Kernel package :
  - kernel image & dtb
  - cmd/gokr-build-kernel : kernel config & build tool
  - gokr-rebuild-kernel : docker wrapper, patch and image copy
- firmware
  - raspberry pi firmware files mirror
  - gokr-update-firmware : updater

Gokrazy tour: inside the OS

- Two partitions : A/B updates
- /boot and /perm
- One init system
- Supervised, remote controlled, password
- goembed for web assets

gokrazy

version 2019-01-27T22:37:00+01:00

host “beatbox”

services

path	last log line
/gokrazy/dhcp	1970/01/01 01:00:09 dhcp.go:211: DHCPACK: IP 192.168.8.131/24, router 192.168.8.…
/gokrazy/ntp	1970/01/01 01:00:09 gokrazy: attempt 2, starting [/gokrazy/ntp]
/gokrazy/randomd	1970/01/01 01:00:07 randomd.go:38: seeding RNG from /perm/random.seed
/gokrazy/breakglass stopped	1970/01/01 01:00:08 gokrazy: process should not be supervised, stopping
/user/hello restarting	Hello, 世界
/user/beatbox	Unknown tag/command: 0201000008d0021a0352c3350b000000

memory

895.0 MiB total, 764.9 MiB available

resident set size (RSS) by service:

Gokrazy tips

- `go.mod`
- go get -u
- Boot control with API (DontStartOnBoot)
- Replace init possible with gokr-packer
- Integrated update system with GOKRAZY_UPDATE env

Breakglass escape hatch

- Why ?
- How ?
- Examples

Breakglass primer

- `scp busybox.tar target: &&`
- `ssh target ./busybox --install -s .`

- `ld-linux-aarch64.so.1` from debian arm64
- https://gokrazy.org/prototyping.html

Constraints

- gokr-packer does a go get directly on a package
- needs to build without CGO
- needs to run without external dependencies

Audio playback

- All go packages I found rely on CGO + alsa-lib (+ something else)
  - oto
  - malgo
  - multiple portaudio bindings
  - etc.

Quick solutions

- Use CGO in /perm
  - `CGO_CPPFLAGS="-I/usr/arm-linux-gnu/include/ -I/usr/arm-linux-gnu/sys-root/usr/include -Wno-error=attributes"  CGO_ENABLED=1 CC=arm-linux-gnu-gcc CXX=arm-linux-gnu-gcc  GOARCH=arm GOOS=linux go build -v -x -work`
- Keep using mpg321, but import all its deps
- Decode mp3 in software, use aplay

aplay

- Alsa player
- Needs working alsa-lib
- Alsa-lib needs alsa.conf and other config files in /usr/share/alsa
  - → modified gokr-packer

```
diff --git a/cmd/gokr-packer/packer.go b/cmd/gokr-packer/packer.go
index a2a7c6d..7303039 100644
--- a/cmd/gokr-packer/packer.go
+++ b/cmd/gokr-packer/packer.go
@@ -371,6 +371,21 @@ func logic() error {
                fromHost: pwPath,
        })

+       for _, dir := range []string{"usr", "usr/share", "usr/share/alsa"} {
+               root.dirents = append(root.dirents, &fileInfo{
+                       filename: dir,
+               })
+       }
+       usr := root.mustFindDirent("usr")
+       usr.dirents = append(usr.dirents, &fileInfo{filename: "share"})
+       share := usr.mustFindDirent("share")
+       share.dirents = append(share.dirents, &fileInfo{filename: "alsa"})
+       alsa := share.mustFindDirent("alsa")
+       alsa.dirents = append(alsa.dirents, &fileInfo{
+               filename: "alsa.conf",
+               fromHost: "./alsa.conf",
+       })
+
        // Determine where to write the boot and root images to.
        var (
```

Mp3 decode

- `github.com/hajimehoshi/go-mp3`
- Pure-go decoder, no CGO
- Very simple API, works well
- Faster on aarch64 rpi3 than 2012 Core i3

Audio on rpi3

- Needs something more recent than 4.20 (was before 5.0-rc1) to have working audio without dt modifications
- Lots of modifications, a big regression was preventing more than one read
- Needs a ~~not-yet merged~~ patch :
  - `Subject: [PATCH] staging: vchiq: Fix local event signalling`
- Update : 5.0-rc4 has the patch

Alsa

- All go packages I found rely on CGO + alsa-lib (+ something else)
  - oto
  - malgo
  - multiple portaudio bindings
  - etc.

github.com/anisse/alsa

- pure go implementation of alsa abi
- very limited scope : playback of stereo interleaved frames, PCM 16 bits LE, 44.1kHz or 48kHz → resample/conversion possibly needed
- unstable API, but for now inspired by oto (partially compatible)

github.com/anisse/alsa

- linux/include/uapi/sound/asound.h
- tinyalsa
- open(), ioctl() → syscall.Open(...), syscall.Syscall(syscall.SYS_IOCTL, ...)
- strace, strace, strace

alsa debugging

- hardware has limited format support – the rest is software (libsamplerate, etc.)
- frame size vs byte size → wrong buffer size, had weird bugs
  - « full read » backup plan

github.com/anisse/alsa

```
type Player


    func NewPlayer(sampleRate, channelNum,
bitDepthInBytes, bufferSizeInBytes int) (*Player,
error)
    func (p *Player) Close() error
    func (p *Player) Write(buf []byte) (int,
error)
```

github.com/anisse/alsa

- Missing features :
  - mmap-based buffer passing (zero copy)
  - non-interleaved
  - hw feature detection
  - capture
  - sample conversion
- PRs welcome

Assembling

- mp3 + alsa
- CopyCtx cancellation

CopyCtx

```
type readerFunc func(p []byte) (n int, err error)
func (rf readerFunc) Read(p []byte) (n int, err error) { return rf(p) }

func CopyCtx(ctx context.Context, dst io.Writer, src io.Reader) (int64, error) {
        n, err := io.Copy(dst, readerFunc(func(p []byte) (int, error) {
                select {
                case <-ctx.Done():
                        return 0, ctx.Err()
                default:
                        return src.Read(p)
                }
        }))
        return n, err
}
```

http://ixday.github.io/post/golang-cancel-copy/

Assembling

```
func playMp3(ctx context.Context, filename string) error {
        f, err := os.Open(filename)
        //...

        dec, err := mp3.NewDecoder(f)
        //...

        sampleRate := dec.SampleRate()
        p, err := alsa.NewPlayer(sampleRate, 2, 2, 4096)
        //...

        _, err = CopyCtx(ctx, p, dec)
        return err
}
```

Demo

Working, now what ?

- ogg
  - `github.com/jfreymuth/oggvorbis`
  - needs sample conversion (float32le → s16le)

Sample conversion reader

```
type resampleReader struct {
        dec *oggvorbis.Reader
}
func (r *resampleReader) Read(p []byte) (n int, err error) {
        fBuf := make([]float32, len(p)/2)
        n, err = r.dec.Read(fBuf)
        for i := 0; i < n; i += 1 {
                val := int16(fBuf[i] * math.MaxInt16)
                binary.LittleEndian.PutUint16(p[i*2:],
                                              uint16(val))
        }
        return n * 2, err
}
```

Working, now what ?

- Playlist and file management is work
- librespot-golang
  - race detector

Demo

Future work

- web interface for controlling data
- more hw platforms with gokrazy
- wireless support with wpa_supplicant
- librespot robustness

Questions ?

References

- https://github.com/gokrazy/gokrazy
- https://github.com/anisse/beatbox
- https://github.com/anisse/alsa
- https://anisse.astier.eu/awk-driven-iot.html
- https://github.com/librespot-org/librespot-golang