
00:00:12

TIL using Golang on stage (literally)

👋 Hi, I’m Alex

- Until 05/2020: Alex Pinnecke
- From 05/2020: Alex Klinkert

- Freelancer from Hamburg
- Cloud Software & Infrastructure Architect

- GitHub / GitLab: aklinkert

One More Word


Ausgang
Exit
SLAYER
DESPISED ICON

👀 Situation

- Jan 2017: OMW needed a light show

- Existing: Windows based or ugly
- Or require (expensive) external hardware
- OSS: OLA (open lighting architecture)
  - C++
  - poorly supported golang client (as of 2017)


Created: 16. Feb 2017 Updated: 15. Jan 2020

Controlling Light

Feb 2017:

- Have to buy a micro PC running Windows
- Run a DMX Controller Software on it
- Have that PC on stage. Have Windows on Stage.
- Nope.

💡 The Idea / Inspiration

- Martin Schuhfuß: Let there be light! | JSConf EU 2015
  - Tim Pietrusky: Nerd Disco | JSConf EU 2014
  - Brad Bouse: Usefulness of Uselessness | JSConf EU 2014

→ Do stuff nobody needs, without deadlines.

→ Have fun learning new things.

Decision

- Golang
- Raspberry PI
- DMX over Art-Net

DMX

- Officially DMX512-A
- Physical / electronic protocol
- 512 channels / addresses
  - = 1 Universe
- Uint8 values (0-255) = 1 byte
- controls dimmer channels (PAR cans)
- Escalated over time ☹

Art-Net

- Binary Ethernet protocol
- UDP 😐
- 2.0.0.0/8 or 10.0.0.0/8
- DMX over IP, so to say
  - Plus RDM (Remote Device Management)
- Multiple universes
- Master = Controller | Client = Node
- https://github.com/jsimonetti/go-artnet
- Not all headers are the same …

offset (bytes)	0	1	2	3
0	'A'	'r'	't'	'-'
4	'N'	'e'	't'	0
8	Opcode **ArtDMX** (0x5000) little endian		Protocol Version Hi (0)	Protocol Version Lo (14)
12	Sequence	Physical	Universe little endian	
16	Length Hi	Length Lo (2 to 512, even)	Data	Data
20	Data ...			

(https://en.wikipedia.org/wiki/Art-Net#Packet_format)

https://artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf

Iteration 0: Metronome CLI

- Write metronome sound to STDOUT
- Emit audio signal with configurable tone height

Iteration 1: Simple “sampler”

- Read audio stream
- Detect click track
- Wait for peak, trigger playback

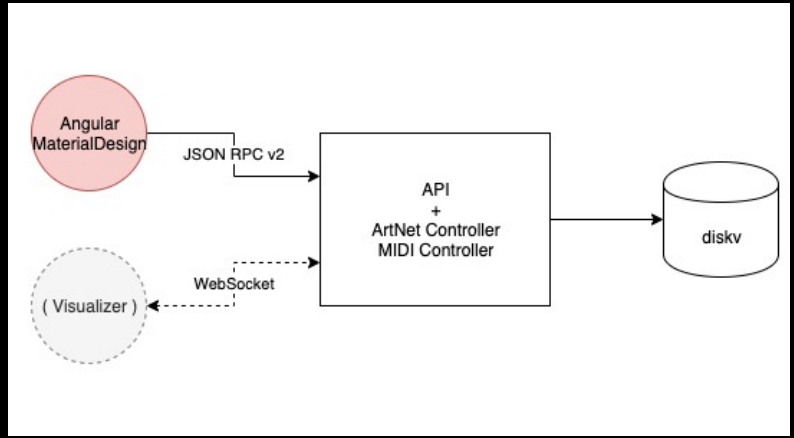
 Iteration 5: LED state change REPL

- Emit Art-Net Commands
- TIL: Art-Net sends full state

...

Iteration 74: Full product

- Diskv storage layer
- Some go-artnet bugfixes later
  - ArtPollReply without version field
  - *Flickering, maximum fps fix*



https://play.golang.org/p/REviGNANWNN

https://github.com/peterbourgon/diskv

DEMO TIME 😎

https://github.com/StageAutoControl/editor

https://github.com/StageAutoControl/controller

https://github.com/OneMoreWord/sac-data

Things I learned

The good, the bad and the stuff I didn’t think about when starting this project. But whom do I tell this, coding is hard sometimes, right?

Flickering: Electrical limits of DMX

- „The maximum packet rate is 44 updates/s if all 513 slots are transmitted (start code + 512 values), but can be higher if fewer slots are transmitted. If packets only consists of a start code + 24 values, up to 830 updates/s can be made.“
- The more addresses, the less fps
- This is the reason why fixtures have a dedicated strobe channel

- → *You cannot outplay physical limitations / timing issues.*

http://www.dmx512-online.com/dmx512_packet.html

https://wiki.openlighting.org/index.php/DMX512-A

Cross compiling is easy. Except when using cgo.

- Cross-compiling go bindings for C libraries is pain.
  - Compiling on AMD64 for ARM requires having the ARM lib installed
- Ansible is still awesome for plain servers / non-cloud environments

Defer is bad. When timing is crucial.

- “calling a function through defer is roughly 100 times slower than inlining it.” - https://bytes-and-bites.com/posts/defer-go-performance/

- (Benchmark)

Channels are bad. When timing is crucial.

- “Channels are the pipes that connect concurrent goroutines. You can send values into channels from one goroutine and receive those values into another goroutine.” - https://gobyexample.com/channels

- In close to real-time communication, use callbacks. This is no joke.
- Try to avoid locks and mutex as long as possible.
- Instead of a single struct holding the data, distribute the data and let the consumer do the building of representational structures. A single routine is way more efficient in (not) syncing access then multiple.

Be aware of go routines.

Golang is a concurrent language, not a parallel one.

“Concurrency is not parallelism, although it enables parallelism.
If you have only one processor, your program can still be concurrent but it cannot be parallel.
On the other hand, a well-written concurrent program might run efficiently in parallel on a multiprocessor.“

Rob Pike @ Google I/O 2012 - Go Concurrency Patterns

https://www.youtube.com/watch?v=f6kdp27TYZs

Be aware of go routines.

- “A single core achieves the illusion of several cores by executing several go-routines in parallel through context switching.”
  - https://medium.com/@vigneshsk/how-to-write-high-performance-code-in-golang-using-go-routines-227edf979c3c

- Be careful with single or slow processors.

Some smaller gotchas

- Dumb storage is insanely fast
- Start interface driven design of code as early as possible
  - Testability will increase a lot with interfaces
  - Wrap libraries with custom types implementing an own interface
- Running go on raspberry is pretty fast
- Compiling golang on raspberry takes some time
- Compiling an angular app on an raspberry takes forever.

Golang isn’t made for real-time.

Do I regret the effort?

No. Go ahead and start writing something that inspires you.

Learn something new. Leave the comfort zone sooner than later.

Write stuff nobody needs. In worst case you learn something, otherwise practice.

Art-Net 4 specification from (c) Artistic Licence Holding Ltd
https://artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf

Art-Net is an Ethernet protocol based on the TCP/IP protocol suite. Its purpose is to allow transfer of large Amounts ofDMX512 data over a wide area using standard networking technology.

🤘 Thank you! 🙏
🧐 Questions? 🤔

Code shown and demo data: https://github.com/aklinkert/godays-lighting-talk
Controller code: https://github.com/StageAutoControl/controller

Mail: alex@klinkert.io

All 📸 images used in these slides are either stock images from pixabay.com , StoryBlocks.com or owned by myself or OneMoreWord.
(except that one Wikipedia Screenshot)