preostalo

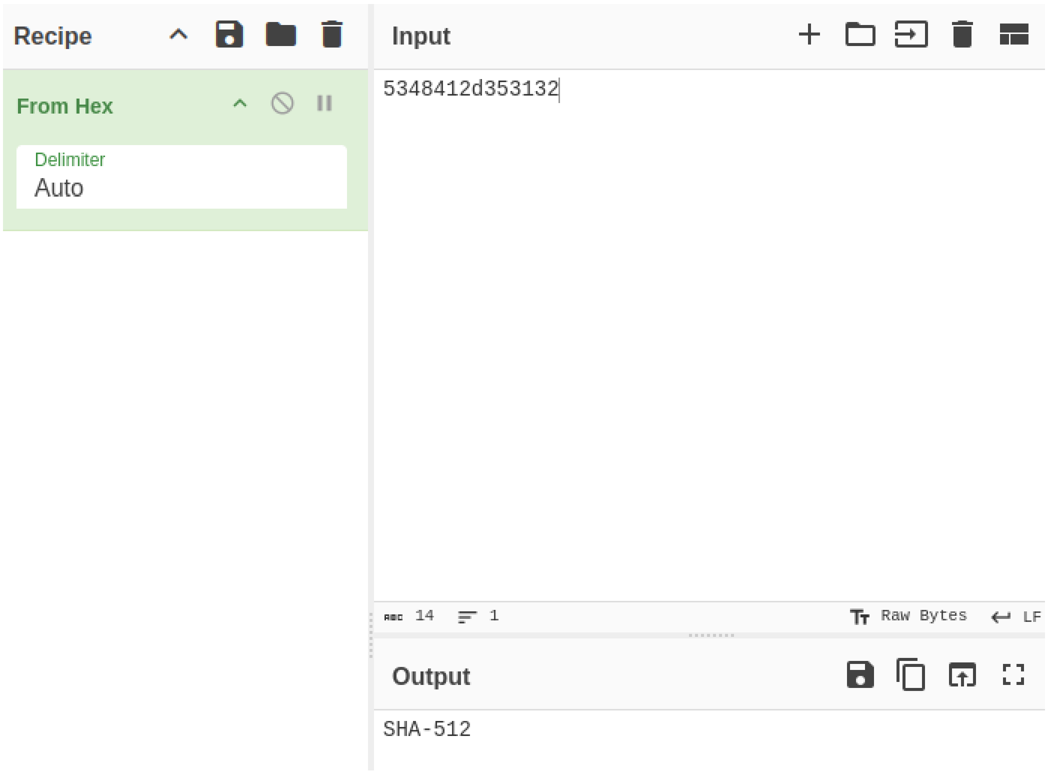
- driver
    - ~~dodat opciju za povezivanje dockercan namespacea na defaultni~~
    - refaktoriraj driver.go u manje funkcije
    - perzistentnost
    - upakirat za linux distribucije
    - github actions za deploy na dockerhub
    - ~~brisanje host interfacea nakon brisanja mreze~~
    - ~~podizanje host interfacea nakon stvaranja mreze~~
    - ~~upload na dockerhub~~
- zadaci
    - napraviti repo koji builda zadatke za dockerhub automatski s github actionsima
    - XCP - dump memory caring caribou
    - UDS Authentication
    - CAN zadatak s dva ECU-a i GUI-jem
        - uds routine control bi se mogao koristiti za ovo
        - upalit zmigavce
        - postic vecu brzinu nego maksimalnu
        - kad uspije, postavit flag na sabirnicu
    - UDS security access MITM izmedju dvije sabirnice
    - UDS zadatak s GUI-jem, routine control
- ecu_template
    - koristit import lib pa po pristunosti datoteka dodat ili maknut neke protokole
    - dodati template za UDS SA
    - povezat s nekim GUI-jem
    - KUKSA server
    - dodati glavni program za slanje repetitivnih poruka umjesto CAN_BCM-a?
    - README
        - čemu svaka od impl datoteka služi
        - kompatibilnost s caring caribouom
        - primjeri kako neke stvari implementirati
    - tagovi za razne varijante templateova
    - XCP
    - DoIP
- dodatni programi:
    - gw program za povezivanje vise dockercan mreza?
    - program za cannelloni udaljeni pristup
- skripta za generiranje docker composeova

30.04.2024.

UDS task 2 otkljucavanje SA

```
❯ caringcaribou uds dump_dids --max_did 0x25 0x100 0x101

-------------------
CARING CARIBOU v0.6 - python 3.11.8 (main, Feb 12 2024, 14:50:05) [GCC 13.2.1 20230801]
-------------------
```

```
Loading module 'uds'

Dumping DIDs in range 0x0000-0x0025

Identified DIDs:
DID    Value (hex)
0x0009 33565746583741543244d363034343934
0x0021 5348412d353132

Done!
```



From Hex

Delimiter
Auto

```
5348412d353132
```

```
SHA-512
```

```
>>> p =s.sr1(UDS()/UDS_SA(securityAccessType=0x1))
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> p =s.sr1(UDS().securitySeed.hex()
'3a1c73eaad1a913dafc452ec332b5185'


>>> import hashlib
>>> hf = hashlib.sha512()
>>> hf.update(p.securitySeed)
>>> key = hf.digest()
>>> key
b'\x145\x84\xac\xeci|p\xf4\xa1+\xcf\xce\x9c\x92\xc9\xdd\x14\xcb_^5\x06\xc2z\x05\xbbU\xe1\n6\xe6\xab\x1c\x
83W#\x13\xd7&-\x96]\x0fW\xf1\xfd1 \x13\xd6>!{\xdd\xbe\xdb\xdcy\xee\xbe\xd7\xf9\xe0'
```

```
>>> s.sr1(UDS()/UDS_SA(securityAccessType=0x2, securityKey=key))
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<UDS  service=SecurityAccessPositiveResponse |<UDS_SAPR  securityAccessType=2 |>>
F)> s.sr1(UDS()/UDS_RMBA(memorySizeLen=0x1, memoryAddressLen=0x4, memoryAddress4=0x00, memorySize1=0xF
...
KeyboardInterrupt
>>> s.sr1(UDS()/UDS_RMBA(memorySizeLen=0x1, memoryAddressLen=0x4, memoryAddress4=0x00, memorySize1=0xFF))
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<UDS  service=ReadMemoryByAddressPositiveResponse |<UDS_RMBAPR
dataRecord='\x7fELF\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00^\x00\x01\x00\x00\x00\\x98\x11
\x08@4\x00\x00\x00\\xbc\\xe2\x04\x00\x00\x03\x00\x004\x00
\x00\x05\x00(\x00\x1c\x00\x1b\x00\x01\x00\x00\x00 \x10\x00\x00 \x00@? \x00@?
\\xbc\\x9f\x00\x00\\xbc\\x9f\x00\x00\x06\x00\x00\x00\x00\x10\x00\x00\x01\x00\x00\x00\x00\\xb0\x00\x00\x00
\x00\\xfb?\x00\x00\\xfb?
T"\x00\x00\x18+\x00\x00\x06\x00\x00\x00\x00\x10\x00\x00\x01\x00\x00\x00\x00\\xe0\x00\x00\x00\x00\x08@\x00
\x00\x08@\x1b\\xc7\x00\x00\x1c\\xc7\x00\x00\x07\x00\x00\x00\x00\x10\x00\x00\x01\x00\x00\x00 \\xb0\x01\x00
\x00\r@
\x00\r@sG\x01\x00sG\x01\x00\x05\x00\x00\x00\x00\x10\x00\x00\x01\x00\x00\x00\\xe8\x0f\x00\x00\\xe8\x1f\x00
P\\xe8\x1f\x00P\x00\x00\x00\x00\x18\x00\x00\x00\x06\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>
```

2.5.2024.

XCP

https://scapy.readthedocs.io/en/latest/layers/automotive.html#universal-calibration-and-measurement-protocol-xcp
https://en.wikipedia.org/wiki/XCP_(protocol)
https://cdn.vector.com/cms/content/application-areas/ecu-calibration/xcp/XCP_Book_V1.5_EN.pdf

XCP ne koristi ISOTP kao i UDS, vec samo obicni CAN (nema dodatnog adresiranja povrh ARB ID-ova)

Standard commands:

Command	PID	Optional
CONNECT	0xFF	No
DISCONNECT	0xFE	No
GET_STATUS	0xFD	No
SYNCH	0xFC	No
GET_COMM_MODE_INFO	0xFB	Yes
GET_ID	0xFA	Yes
SET_REQUEST	0xF9	Yes
GET_SEED	0xF8	Yes
UNLOCK	0xF7	Yes
SET_MTA	0xF6	Yes
UPLOAD	0xF5	Yes
SHORT_UPLOAD	0xF4	Yes
BUILD_CHECKSUM	0xF3	Yes
TRANSPORT_LAYER_CMD	0xF2	Yes
USER_CMD	0xF1	Yes
GET_VERSION	0xC0, 0x00	Yes

Calibration commands:

Command	PID	Optional
DOWNLOAD	0xF0	No
DOWNLOAD_NEXT	0xEF	Yes
DOWNLOAD_MAX	0xEE	Yes
SHORT_DOWNLOAD	0xED	Yes
MODIFY_BITS	0xEC	Yes

Page switching:

Command	PID	Optional
SET_CAL_PAGE	0xEB	No
GET_CAL_PAGE	0xEA	No
GET_PAG_PROCESSOR_INFO	0xE9	Yes
GET_SEGMENT_INFO	0xE8	Yes
GET_PAGE_INFO	0xE7	Yes
SET_SEGMENT_MODE	0xE6	Yes
GET_SEGMENT_MODE	0xE5	Yes
COPY_CAL_PAGE	0xE4	Yes

Cyclic data exchange - basics:

Command	PID	Optional
SET_DAQ_PTR	0xE2	No
WRITE_DAQ	0xE1	No
SET_DAQ_LIST_MODE	0xE0	No
START_STOP_DAQ_LIST	0xDE	No
START_STOP_SYNCH	0xDD	No
WRITE_DAQ_MULTIPLE	0xC7	Yes
READ_DAQ	0xDB	Yes
GET_DAQ_CLOCK	0xDC	Yes
GET_DAQ_PROCESSOR_INFO	0xDA	Yes
GET_DAQ_RESOLUTION_INFO	0xD9	Yes
GET_DAQ_LIST_MODE	0xDF	Yes
GET_DAQ_EVENT_INFO	0xD7	Yes
DTO_CTR_Properties	0xC5	Yes
SET_DAQ_PACKED_MODE	0xC0, 0x01	Yes
Get_DAQ_PACKED_Mode	0xC0, 0x02	Yes

Cyclic data exchange - static configuration:

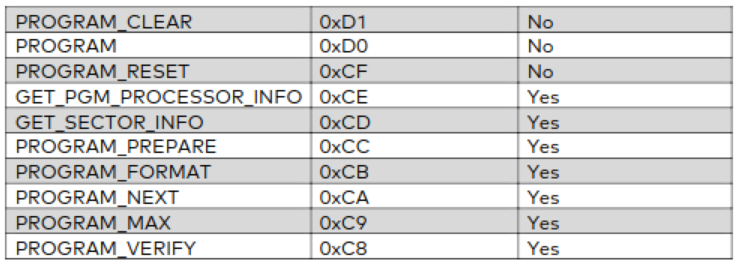
Command	PID	Optional
CLEAR_DAQ_LIST	0xE3	No
GET_DAQ_LIST_INFO	0xD8	Yes

Cyclic data exchange - dynamic configuration:

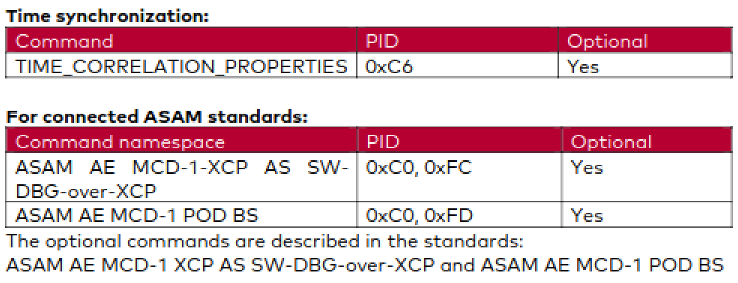
Command	PID	Optional
FREE_DAQ	0xD6	Yes
ALLOC_DAQ	0xD5	Yes
ALLOC_ODT	0xD4	Yes
ALLOC_ODT_ENTRY	0xD3	Yes

Flash programming:

Command	PID	Optional
PROGRAM_START	0xD2	No



PROGRAM_CLEAR	0xD1	No
PROGRAM	0xD0	No
PROGRAM_RESET	0xCF	No
GET_PGM_PROCESSOR_INFO	0xCE	Yes
GET_SECTOR_INFO	0xCD	Yes
PROGRAM_PREPARE	0xCC	Yes
PROGRAM_FORMAT	0xCB	Yes
PROGRAM_NEXT	0xCA	Yes
PROGRAM_MAX	0xC9	Yes
PROGRAM_VERIFY	0xC8	Yes



Time synchronization:

Command	PID	Optional
TIME_CORRELATION_PROPERTIES	0xC6	Yes

For connected ASAM standards:

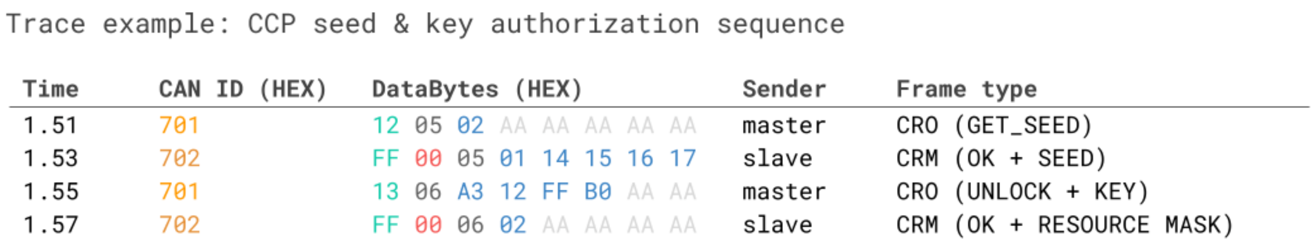
Command namespace	PID	Optional
ASAM AE MCD-1-XCP AS SW-DBG-over-XCP	0xC0, 0xFC	Yes
ASAM AE MCD-1 POD BS	0xC0, 0xFD	Yes

The optional commands are described in the standards:
ASAM AE MCD-1 XCP AS SW-DBG-over-XCP and ASAM AE MCD-1 POD BS

3.5.

https://www.csselectronics.com/pages/ccp-xcp-on-can-bus-calibration-protocol

- postoje GET_SEED i UNLOCK poruke koje su ekvivalenti SecurityAccess porukama na UDS-u



```
Trace example: CCP seed & key authorization sequence

Time     CAN ID (HEX)   DataBytes (HEX)             Sender    Frame type
1.51     701            12 05 02 AA AA AA AA AA     master    CRO (GET_SEED)
1.53     702            FF 00 05 01 14 15 16 17     slave     CRM (OK + SEED)
1.55     701            13 06 A3 12 FF B0 AA AA     master    CRO (UNLOCK + KEY)
1.57     702            FF 00 06 02 AA AA AA AA     slave     CRM (OK + RESOURCE MASK)
```

https://scapy.readthedocs.io/en/latest/layers/automotive.html#xcpscanner

- scapy ima XCPScanner koji bi trebalo prouciti kako funkcionira, jer bi ga natjecatelji mogli koristiti u zadacima

zadatak

- natjecatelju se daje adresa parametra koji se koristi kao materijal za stvaranje kljuca uz seed
    - (kroz opis zadatka ili kroz kod koji generira kljuc)
- koristeci xcp naredbe mora dohvatiti parametar, dohvatiti seed i otkljucati ECU