7.5.

Postavljanje scapy konzole za koristenje XCP-a:

```
load_layer('can')
load_contrib('cansocket')
load_contrib('automotive.xcp')
load_contrib('automotive.xcp.xcp')
sock = CANSocket(bustype='socketcan', channel='vcan0', basecls=XCPOnCAN)
```

Slanje connect paketa

```
>>> pkt = XCPOnCAN(identifier=0x700) / CTORequest() / Connect()
>>> pkt
<XCPOnCAN  identifier=0x700 |<CTORequest  pid=CONNECT |<Connect  |>>>
>>> sock.send(pkt)
16
```

8.5.

https://piembsystech.com/xcp-protocol/

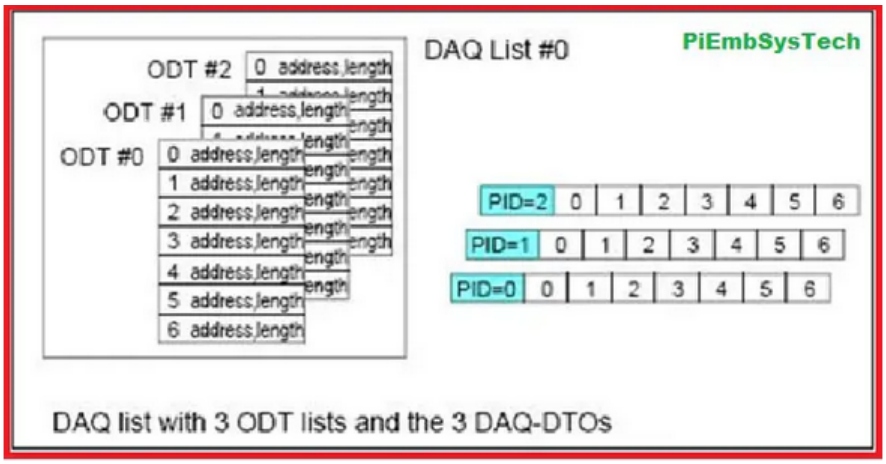
- CTO - poruke
  - CMD/RES/ERR/EV/SERV
- DTO - podaci
  - DAQ/STIM

The DTO has two types of objects that are both used for event-driven reading of variables from, or writing values to, the memory of the slave.

1. DAQ (Data Acquisition).
2. STIM (Stimulation).

Data Acquisition (DAQ): A core feature of the XCP protocol is the DAQ lists. In order to be able to send a large amount of data in a small amount of time and with low bandwidth load desirable, XCP offers the ability to configure lists that take care of the transmitting requested data at a given interval. Each DAQ list (Figure 10) has a number of Object Descriptor Tables (ODTs) that in turn contain Object Descriptor Table Entries (ODT Entries) as described in Figure 11. Each the ODT Entry has an address and a length, these make out the description of the parameter that it represents. When the DAQ list has processed the contents of the list are copied to the corresponding address of each entry in each ODT. The slave doesn’t receive an acknowledgment that the master

has received the data correctly.



XCP DAQ Diagram

preostalo

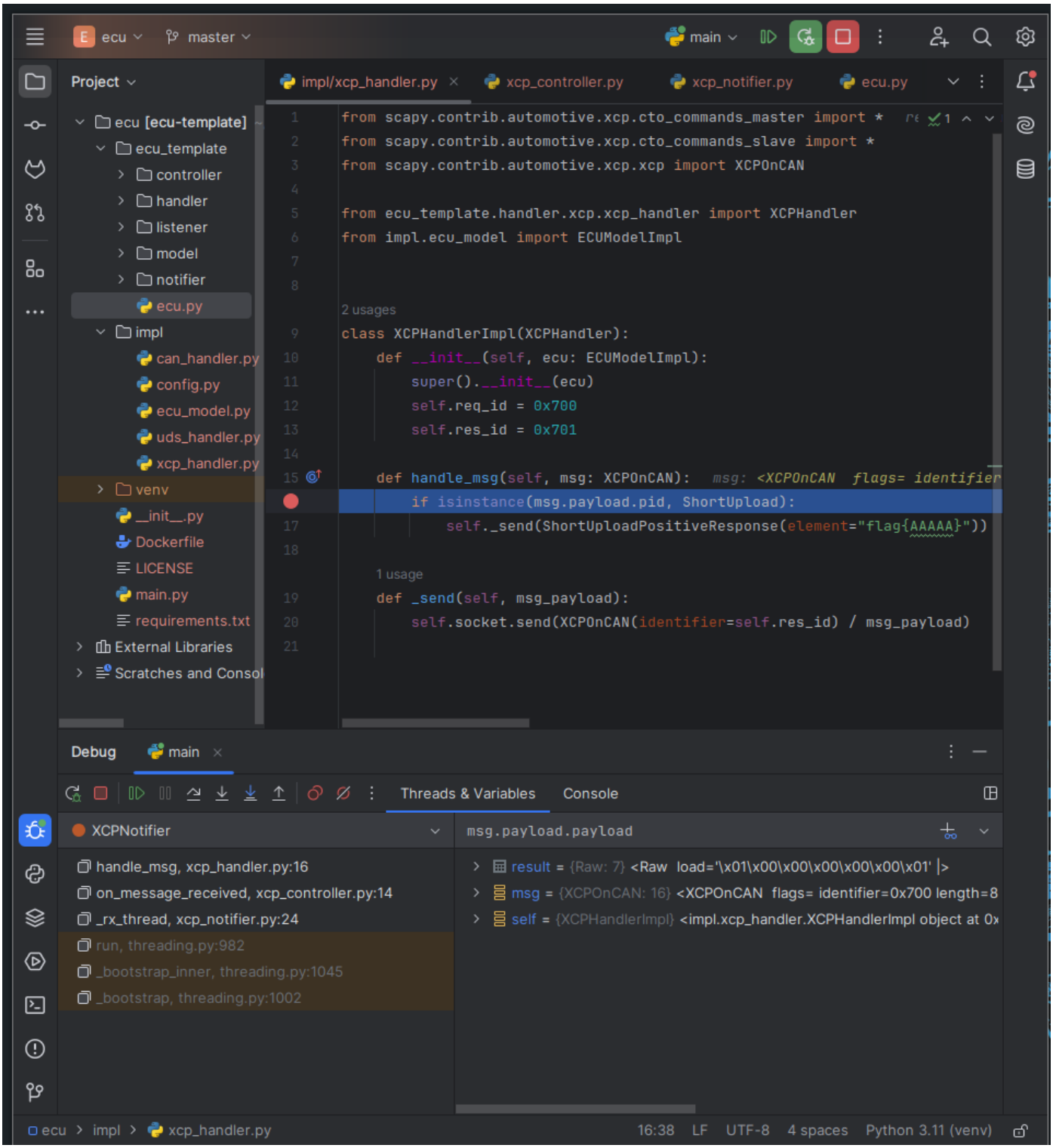
- driver
  - ~~dodat opciju za povezivanje dockercan namespacea na defaultni~~
  - refaktoriraj driver.go u manje funkcije
  - perzistentnost
  - upakirat za linux distribucije
  - github actions za deploy na dockerhub
  - ~~brisanje host interfacea nakon brisanja mreze~~
  - ~~podizanje host interfacea 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

    - parsiranje CAN signala po DBC-u
- dodatni programi:
    - gw program za povezivanje vise dockercan mreza?
    - program za cannelloni udaljeni pristup
- skripta za generiranje docker composeova

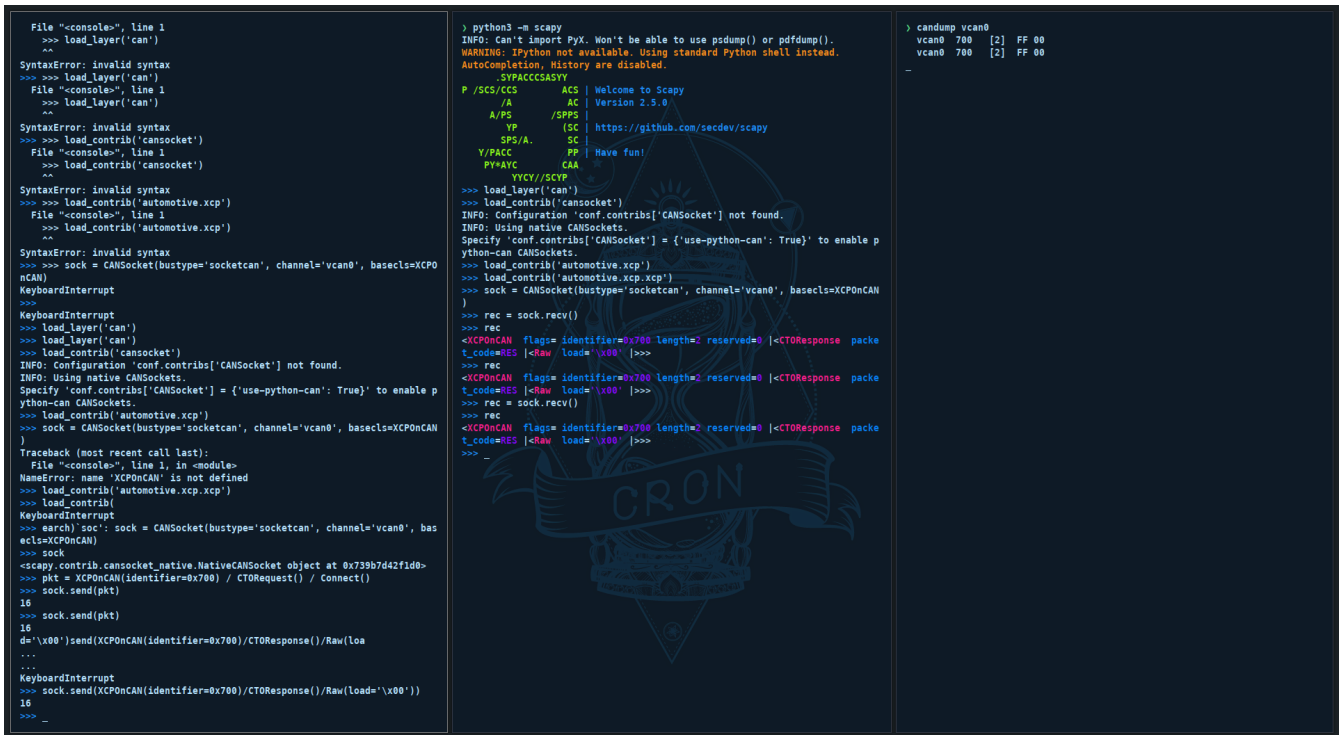
10.5.

Iz nekog razloga scapy ne prepoznaje automatski shortupload paket po primitku.

```
)>> pkt2 = XCPOnCAN(identifier=0x700) / CTORequest() / ShortUpload(nr_of_data_elements=1, address=0x0)
>>> pkt2
<XCPOnCAN  identifier=0x700 |<CTORequest  pid=SHORT_UPLOAD |<ShortUpload  nr_of_data_elements=1 address=1
|>>>    sock.sr1(pkt2)
Begin emission:
Finished sending 1 packets.
^C
Received 0 packets, got 0 answers, remaining 1 packets
>>> sock.sr1(pkt2)
Begin emission:
Finished sending 1 packets.
```



- scapy poslani CTORequest prepoznaje kao CTOResponse?



Sadrzaj paketa za CTOResponse i CTORequest se ne razlikuje vec je kontekstualan. Scapy automatski sve sto procita smatra Responseom, obzirom da je namijenjen za komunikaciju s ECU-ovima.

```
❯ candump vcan0
  vcan0  700   [2]  FF 00
  vcan0  700   [2]  FF 00
```