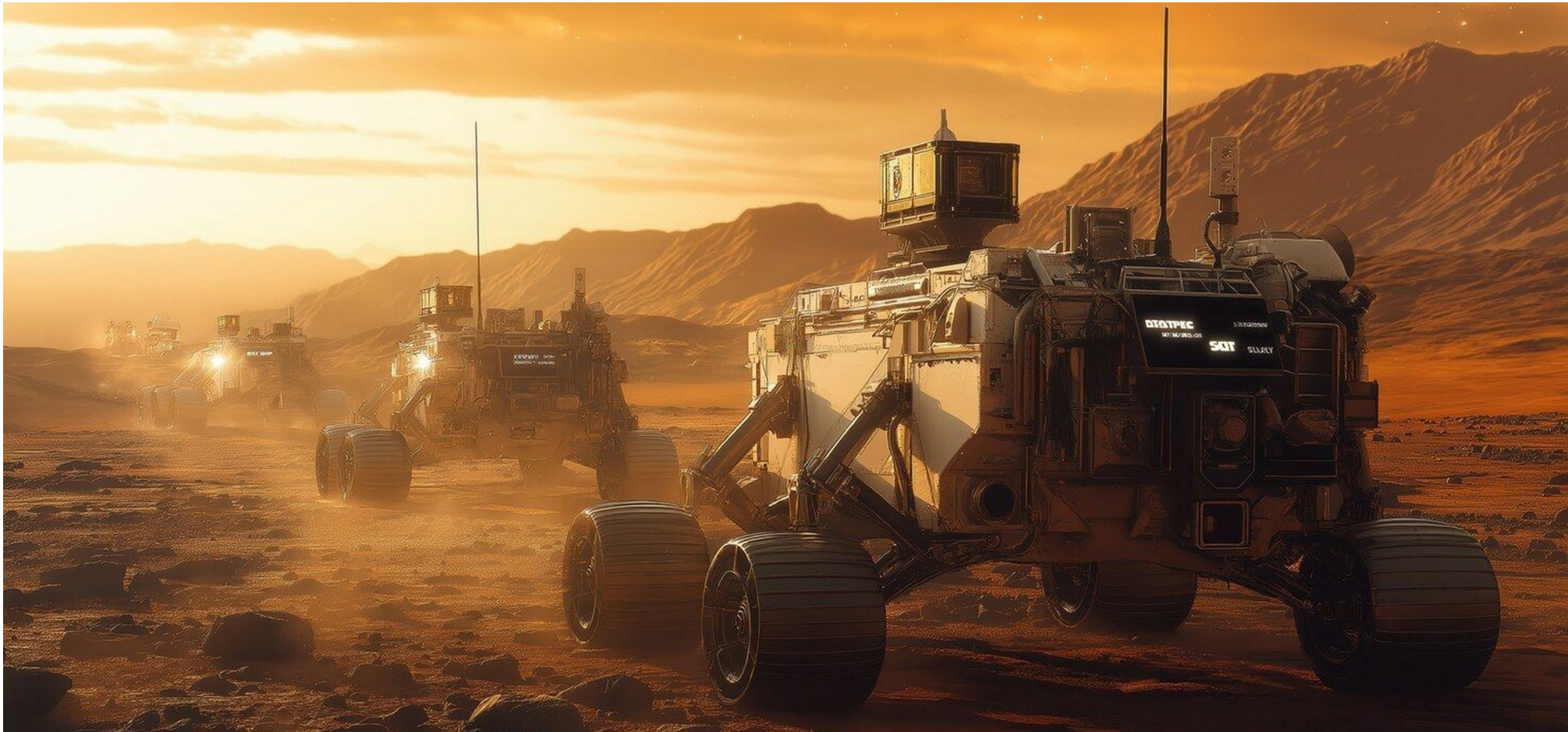
SpaceCAN Tutorial

LIBRE CUBE
open source space exploration

Release date: 2025-05-16



Development Setup

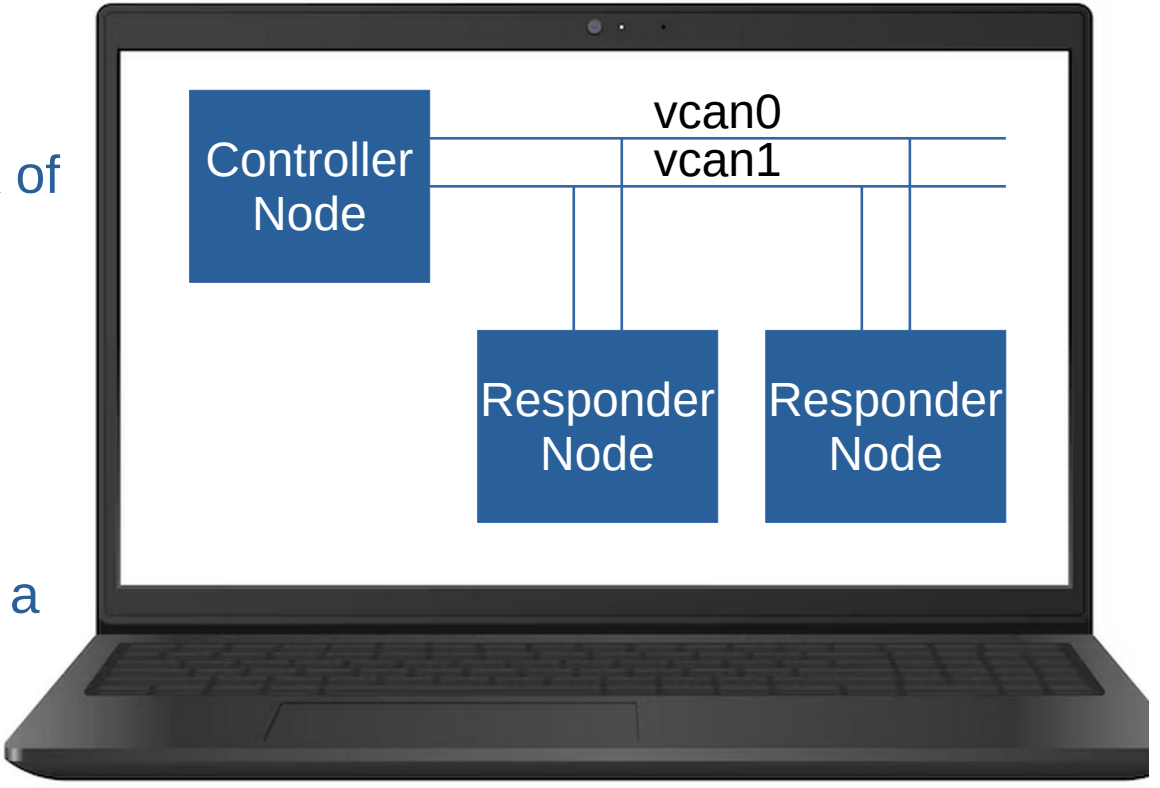
LIBRE CUBE
open source space exploration

- SpaceCAN is a protocol to be used onboard (small) satellites, rovers, drones, and other remote vehicles.
- It is available for Linux and microcontrollers. The following setups are possible:
  - **Virtual Setup**: In this setup the SpaceCAN protocol runs on a single Linux computer and uses the virtual CAN network (vcan). This is easiest to set up.
  - **Bridged Setup**: Here we use a Linux computer (or Raspberry Pi) for the controller node and microcontrollers for the responder nodes. The computer connects to the CAN network through an adapter (USB-CAN).
  - **Embedded Setup**: In this setup we use microcontrollers for the controller node and responder nodes. It uses a physical dual CAN network. This setup reflects a typical setup of small satellite or similar system.
- Note: In this tutorial we will only use the virtual setup.

Virtual Setup Overview

LIBRE CUBE
open source space exploration

- Runs locally on a Linux machine.
- Uses the virtual CAN network of Linux.
- Can be used to test all functionalities of SpaceCAN, including redundancy management.
- Note: You can run this also in a VM that has Linux installed.



Getting Started

LIBRECUBE
open source space exploration

- To run the examples of this tutorial, you need to clone (or download) the python-spacecan to your machine, as it contains the example scripts.
  - git clone https://gitlab.com/librecube/lib/python-spacecan
  - cd python-spacecan
  - python -m venv venv
  - source venv/bin/activate
  - pip install .
- To monitor the traffic on the virtual CAN bus, install “can-utils” (via Linux package manager).

Getting Started (2/3)

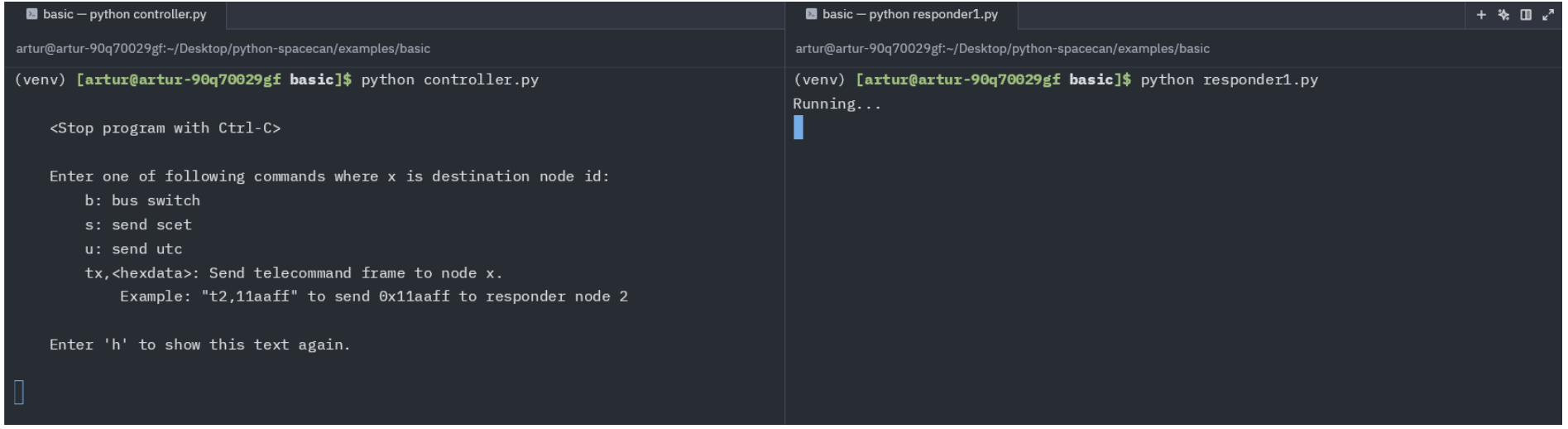
LIBRE CUBE
open source space exploration

- Now, bring up the virtual CAN network. (The network keeps active, until next computer restart).
  - cd examples
  - source vcan.sh
- In the terminal, start the controller from the basic example:
  - source venv/bin/activate
  - cd examples/basic
  - python controller.py
- In another terminal, start the responder 1 from the basic example:
  - source venv/bin/activate
  - cd examples/basic
  - python responder1.py

Getting Started (3/3)

LIBRECUBE
open source space exploration

- You should now have the controller and responder terminals similar to this:



Basics

Basic Features

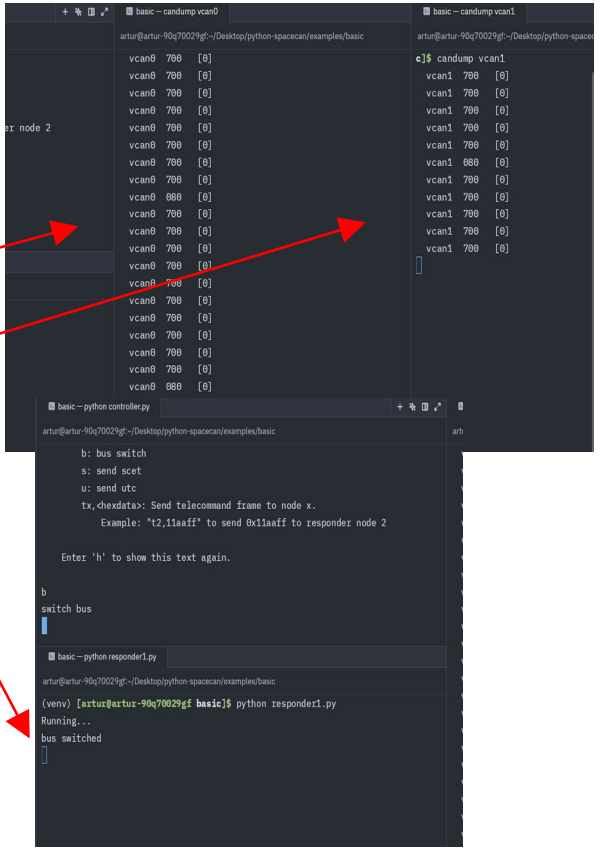
LIBRECUBE
open source space exploration

- Redundancy Management
- Synchronization
- Time Distribution (SCET and UTC)
- Telecommand and Telemetry

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

Redundancy Management

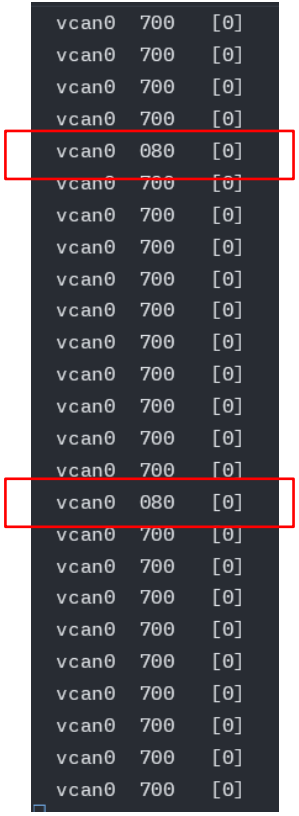
- Open two terminals and start candump for both channels:
  - candump vcan0
  - candump vcan1
- Observe that heartbeats (0x700) are sent on vcan0.
- Now let the controller do a bus switch and observe that the heartbeats are now on vcan1. After 2.5 seconds the responder will print a message that it switched the bus.
- Exercises:
  - Stop the controller and observe how the responder will switch from bus to bus, searching for heartbeats.
  - Change the settings for heartbeat period and max number of missed heartbeats in controller and responder config.



Synchronization

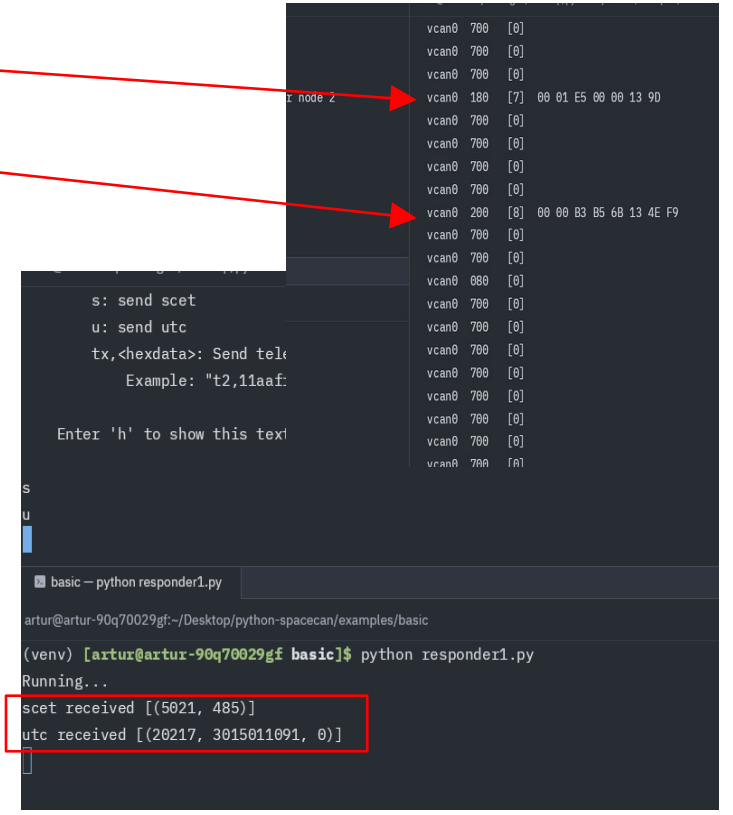
LIBRECUBE
open source space exploration

- Observe in the candump log the appearance of SYNC objects (0x080) every 5 seconds.
- Exercises:
  - Change the sync period in the controller config file.
  - Make the responder print a message every time it receives a sync.



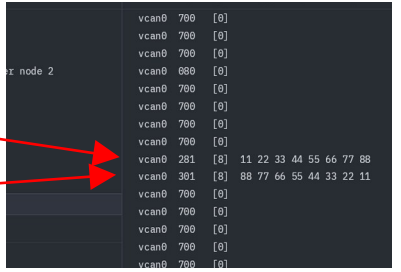
Time Distribution

- Let the controller send a SCET object (0x180) and an UTC object (0x200).
- Observe that it is received by the responder.
- Exercises:
  - Make the responder print the values in a better readable format.
  - Have responder manage a SCET time itself, which shall be printed every 1 second and to be updated with the received SCET time from the controller.
  - Change the absolute time reference for UTC.



Telecommand and Telemetry

- Send a telecommand to node 1 with a data of 0x1122334455667788.
- Observe how the responder replies with a telemetry object that contains the same data but reversed.
- Exercises:
  - Make the responder print out a message “on” or “off” when it receives a telecommand with data “01” or “00”, respectively. For all other data it shall print “invalid TC”.
  - Start a second responder (responder2) and ensure that responders can be commanded individually.




```
       s: send scet
       u: send utc
       tx,<hexdata>: Send telecommand frame to node x.
           Example: "t2,11aaff" to send 0x11aaff to responder node 2

   Enter 'h' to show this text again.

t1,1122334455667788
telemetry received [8877665544332211] from node 1
```

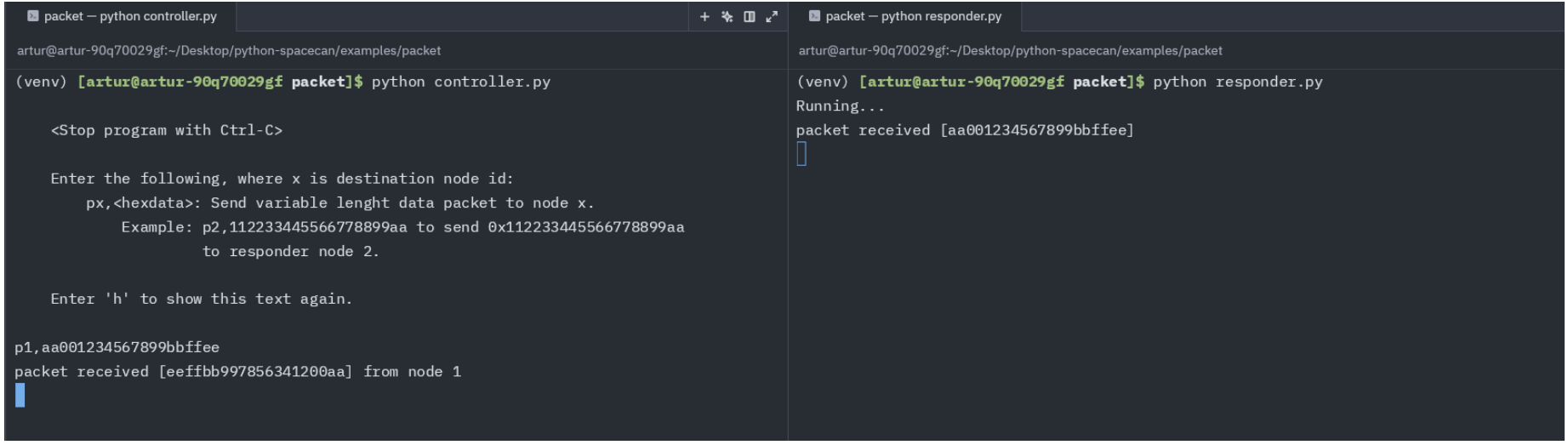
```
basic — python responder1.py
artur@artur-90q70029gf:~/Desktop/python-spacecan/examples/basic
(venv) [artur@artur-90q70029gf basic]$ python responder1.py
Running...
scet received [(5021, 485)]
utc received [(20217, 3015011091, 0)]
t1,112233
telecommand received [112233]
^C(venv) [artur@artur-90q70029gf basic]$ python responder1.py
Running...
telecommand received [1122334455667788]
```

Packets

Packet Example

LIBRE CUBE
open source space exploration

- Follow the similar steps as before, but now go to “examples/packet” folder and start controller and responder. Note that in the config file for controller and responder we see that packets are enabled.
- See how you can now send more than 8 bytes by using the packet protocol.

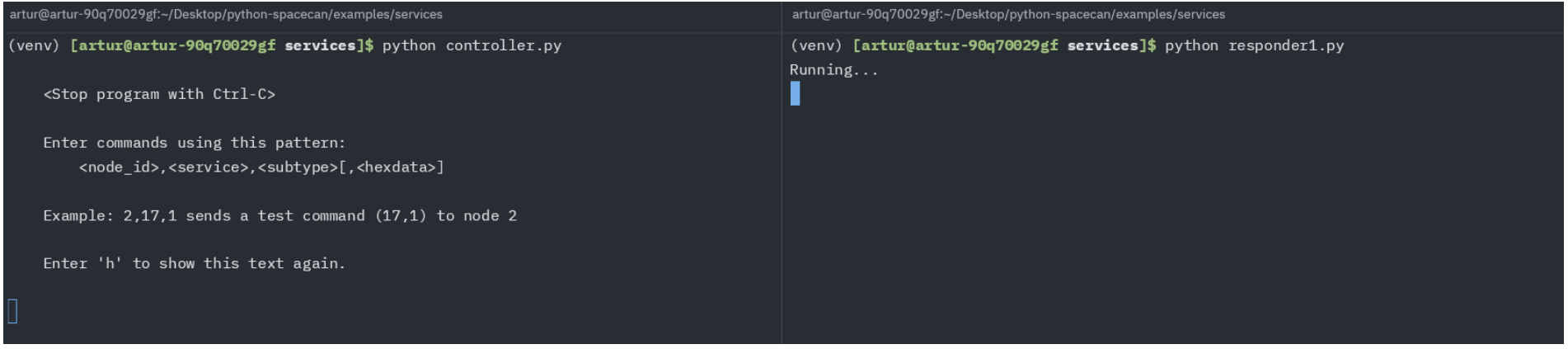


SpaceCAN Service Protocol

Getting Started

LIBRECUBE
open source space exploration

- Follow the similar steps as before, but now go to “examples/services” folder and start controller and one or more responder.



Service 17: Test Service

LIBRECUBE
open source space exploration

- Send to node 1 a TC 17,1 request by entering: 1,17,1
- Observe how the responder node 1 replies with a TM 17,2.

Service 1: Request Verification

LIBRECUBE
open source space exploration

- For each request from controller to responder, a request verification reports will be sent from the responder. In normal case, the controller will receive a TM 1,1 for successful acceptance and a TM 1,7 for successful completion.
- Note that the data field contains the source TC (in this case 17,1 or in hex 0x1101).
- Exercise: Send an invalid request and expect to receive a TM 1,2.

Service 20: Parameter Management (1/2)

LIBRE CUBE
open source space exploration

- Send to responder node 1 a parameter report request TC 20,1 to get the value of parameter with id 1: 1,20,1,0101
- Note how the reply contains the encoded parameter value.
- Then do the same but request report for parameter id 1 and 2: 1,20,1,020102
- Look into the file “config/services.json” to learn more how the parameters are defined and their encoding.

```
    <Stop program with Ctrl-C>

    Enter commands using this pattern:
        <node_id>,<service>,<subtype>[,<hexdata>]

    Example: 2,17,1 sends a test command (17,1) to node 2

    Enter 'h' to show this text again.

1,20,1,0101
001: ServicePacket(1, 1, 1401)
001: ServicePacket(20, 2, 0101000011a9)
001: ServicePacket(1, 7, 1401)
1,20,1,020102
001: ServicePacket(1, 1, 1401)
001: ServicePacket(20, 2, 0201000013470200)
001: ServicePacket(1, 7, 1401)
```

Service 20: Parameter Management (2/2)

LIBRECUBE
open source space exploration

- In the parameter configuration section, each parameter has an id, a name, optionally a comment, an encoding code, and an initial value. The encoding code is listed in the table.
- Exercises:
  - Change the encoding of parameter 2 and see the effect.
  - Add a new parameter. For this, you need to update the config file and also the code in the responder script.

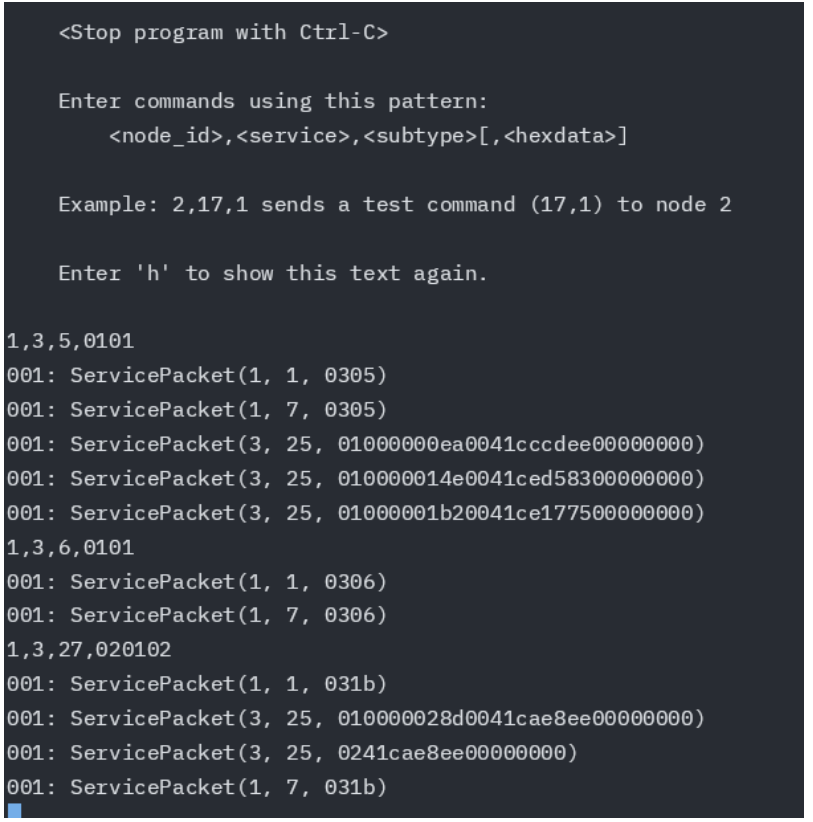
```
"parameters": [
    {
        "parameter_id": 1,
        "parameter_name": "counter",
        "comment": "a monotonic counter",
        "encoding": "i",
        "value": 0
    },
    {
        "parameter_id": 2,
        "parameter_name": "switch_status",
        "encod
        "value
    },
    {
        "param
```

Format	C Type	Python Type	Size
b	signed char	integer	1
B	unsigned char	integer	1
h	short	integer	2
H	unsigned short	integer	2
l	long	integer	4
L	unsigned long	integer	4
f	float	float	4
d	double	float	8

Service 3: Housekeeping Management (1/2)

LIBRECUBE
open source space exploration

- Send a TC 3,5 to enable housekeeping report with id 1: 1,3,5,0101
- Observe how the housekeeping reports TM 3,25 are received every 10 seconds.
- Send a TC 3,6 to disable housekeeping report with id 1: 1,3,6,0101
- Send a TC 3,27 to request a single shot report of housekeeping reports with id 1 and 2: 1,3,27,020102
- Observe how each housekeeping report is sent separately.



Service 3: Housekeeping Management (2/2)

LIBRECUBE
open source space exploration

- In the parameter configuration section, each housekeeping report has a report id, an optional comment, a repetition interval in seconds, a flag to indicate if the report is initially enabled, and the list of parameter ids that it contains.
- Exercise:
  - Change the interval of the reports.
  - Make the report 2 enabled as default.
  - Create new report and check that it can be enabled and disabled.

```
"housekeeping_reports": [
    {
        "report_id": 1,
        "comment": "includes full set of parameters",
        "interval": 10,
        "enabled": false,
        "parameter_ids": [1, 2, 3, 4]
    },
    {
        "report_id": 2,
        "interval": 5,
        "enabled": false,
        "parameter_ids": [3, 4]
```

Service 8: Function Management

LIBRE CUBE
open source space exploration

- Send to responder node 1 a TC 8,1 to request the execution of function with id 1: 1,8,1,01
- Observe the reaction of the responder node.
- Now send a TC 8,1 to request execution of function id 2 with an argument id 1 of value 0: 1,8,1,02010100
- Exercise: Define your own function. You will need to modify the service config file and the responder script. Note also that argument values are sent in encoded form.

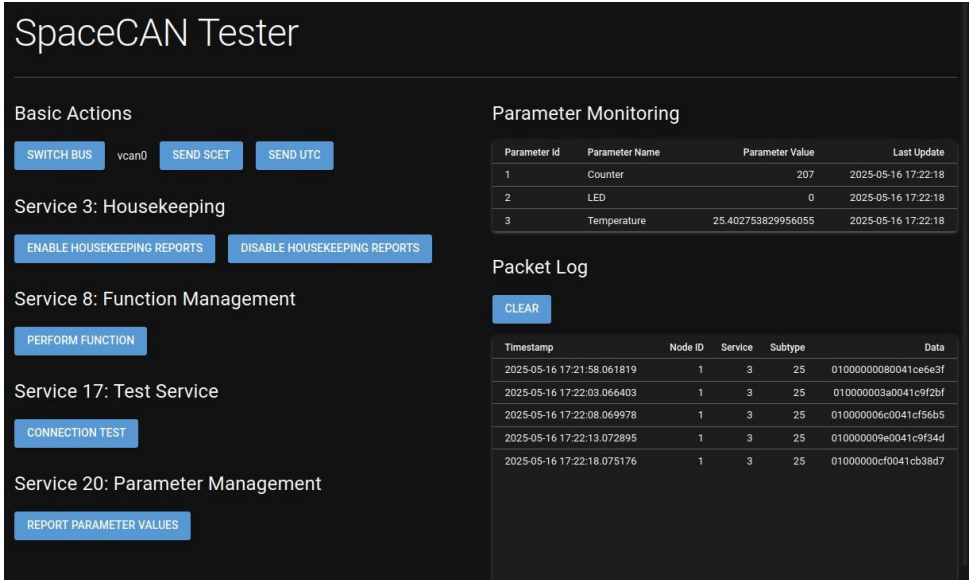
Summary

- These examples were showing the underlying workings of the TC requests and TM reports.
- For typical usage, you will only need to change configuration files and write your code to update parameters and handle functions. The values encoding and decoding is taken care of by the protocol, you do not need to do this by hand.
- Further documentation:
  - The official documentation of the SpaceCAN protocol can be found here: https://librecube.gitlab.io/standards/spacecan/
  - To see how you can use the python-spacecan module in your project have a look here: https://gitlab.com/librecube/lib/python-spacecan/-/blob/main/docs/README.md

SpaceCAN Tester Application

Introduction

- The SpaceCAN Tester application can be used to a test responder node connected over the SpaceCAN bus.
- The SpaceCAN Tester acts hereby as an controller node: it receives the telemetry from the responder nodes, and it can issue commands to it.
- It is much more convenient than using the command line.



Getting Started (1/3)

LIBRECUBE
open source space exploration

- Clone (or download) the SpaceCAN Tester, as it contains the example:
  - git clone https://gitlab.com/librecube/elements/LC3301.git spacecan-tester
  - cd spacecan-tester
  - python -m venv venv
  - source venv/bin/activate
  - pip install .

Getting Started (2/3)

LIBRE CUBE
open source space exploration

- Now we are going to run the example as provided in the SpaceCAN Tester project.
- Bring up the virtual CAN network, if not already:
  - cd examples/virtual
  - source vcan.sh
- In a terminal start the SpaceCAN Tester:
  - source venv/bin/activate
  - cd examples/virtual
  - spacecan-tester config.json
- A browser window will open at http://localhost:8080/
- In another terminal start the responder node:
  - source venv/bin/activate
  - cd examples/virtual
  - python responder.py
- You may also open two other terminals to monitor the traffic on the virtual CAN network.

Getting Started (3/3)

- You should now see a screen as below.
- You can now repeat the exercises from this tutorial using the SpaceCAN Tester by modifying the config files and the responder code.