

Embedded Connectivity (SS21)

Prof. Dr. Ing. Andreas Grzemba

Ing. Johann Bretzendorfer

G2WS4

Rashed Al-Lahaseh - 00821573

Automotive Ethernet Industry

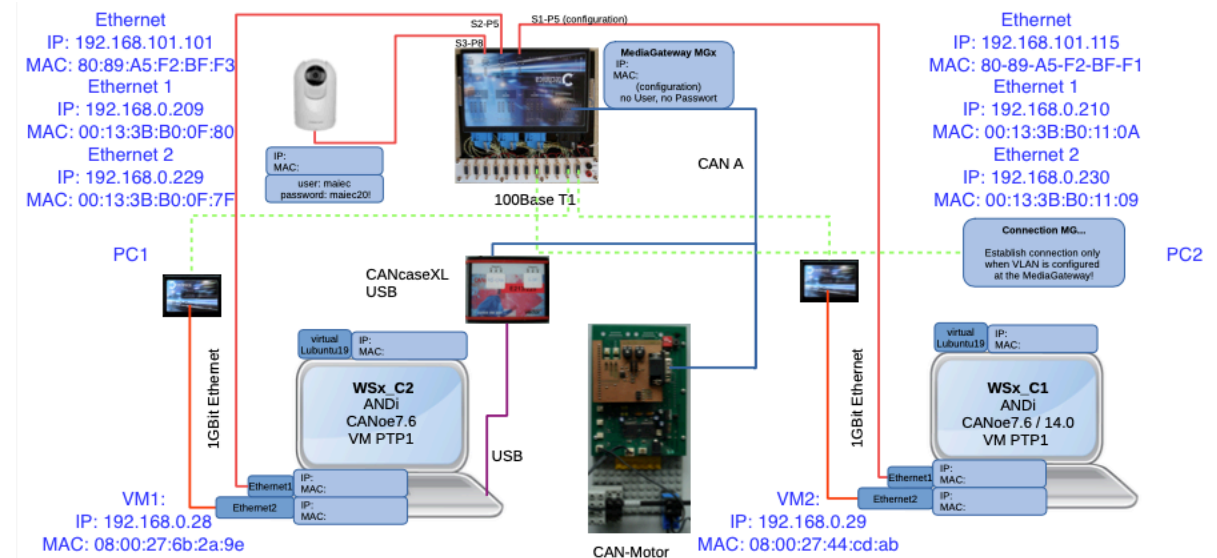
- Regarding Research & Market latest report, the automotive Ethernet market is projected to grow from USD 1.8 billion in 2020 to USD 5.6 billion by 2026, at a Compound Annual Growth Rate (CAGR) of 20.9% from 2020 to 2026.
- Which means major factors expected to drive the growth of the automotive Ethernet market include increasing demand for higher bandwidth, rise in deployment of (Advanced driver-assistance systems) ADAS and infotainment systems, rising vehicle production, and growing demand for passenger and safety and convenience.

Labs

- Now we will take a quick look for the tasks we took during our labs.

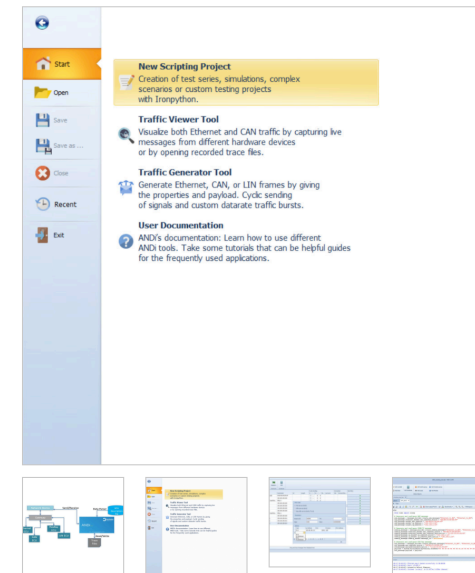
Lab 1

- During the first lab we had the chance to check our work station and we saw that workstation consists of 2 computers.
- The computers have 3 Ethernet ports, which are used for different tasks.



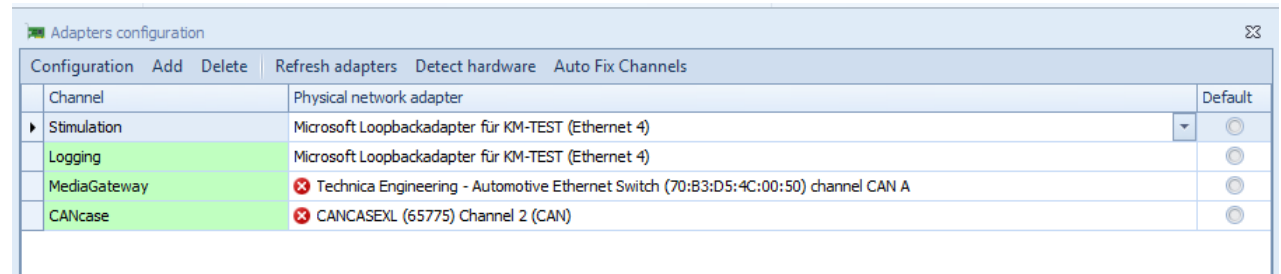
Lab 2

- On the second lab, a new tool where introduced 'ANDi' which is a software developed by Technica-Engineering.
- **The Automotive Network Diagnoser (ANDi)** is a test and simulation environment for electronic control units Automotive Ethernet (100BASE-T1 / 1000BASE-T1), CAN, CAN-FD, LIN and FlexRay bus systems.



Lab 2

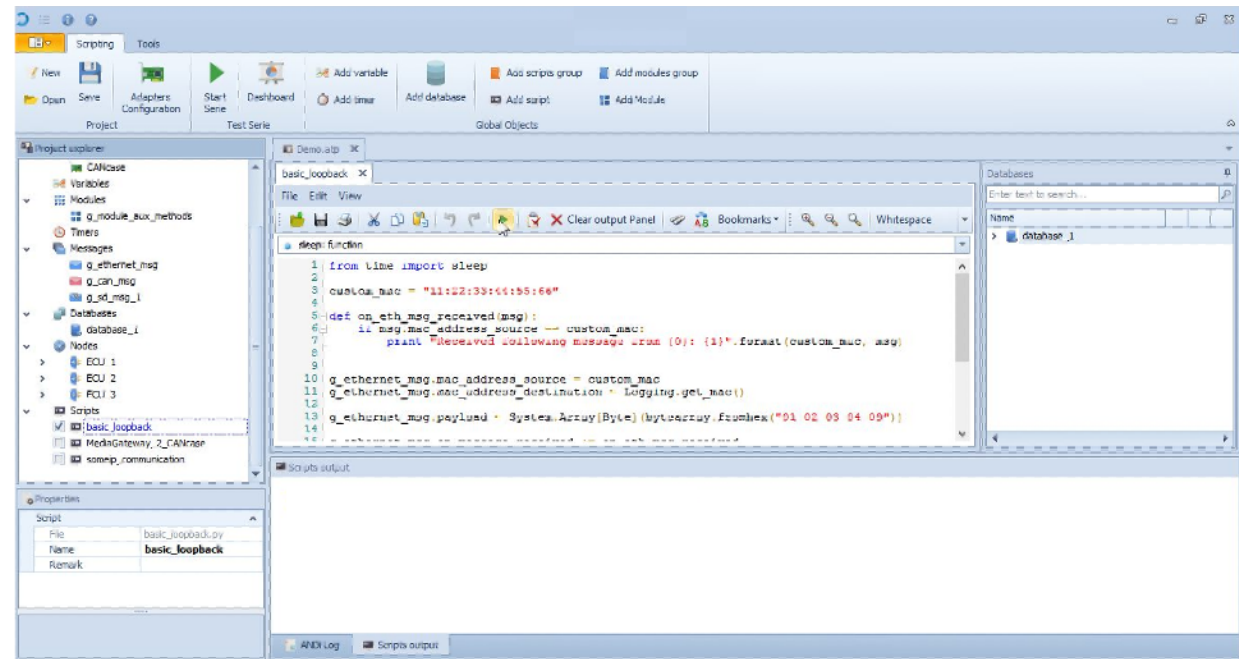
- First we had the chance to check the demo project, which helped us discover the main functionalities in the ANDi tool.
- Adapter configuration, where we set the channels responsible for simulation and logging process, as the demo project was already set up to Loopback adapters.



| Adapters configuration | | | |
|---|---|--|----------------------------------|
| Configuration Add Delete Refresh adapters Detect hardware Auto Fix Channels | | | |
| Channel | Physical network adapter | | Default |
| ► Stimulation | Microsoft Loopbackadapter für KM-TEST (Ethernet 4) | | <input checked="" type="radio"/> |
| Logging | Microsoft Loopbackadapter für KM-TEST (Ethernet 4) | | <input type="radio"/> |
| MediaGateway | Technica Engineering - Automotive Ethernet Switch (70:B3:D5:4C:00:50) channel CAN A | | <input type="radio"/> |
| CANcase | CANCASEXL (65775) Channel 2 (CAN) | | <input type="radio"/> |

Lab 2

- Basic loopback script, the programming language used is Python and the script was provided using custom_mac address and it we saw that if we want to establish a bidirectional data traffic, the sender address must be correct.



Lab 3

- On the third lab we had to establish a send/recive script based on the basic loopback script we tested.
- And from this task we saw that we can modify the adapter configuration and set the logging and simulation to the required enviornment.

```
[2021-04-30 11:45:21]: Start 'receive'
[2021-04-30 11:45:45]: Start 'send'
[2021-04-30 11:45:45]: Sender: Frame 01 has been sent.
[2021-04-30 11:45:45]: Receiver: Received message: [Mac src: 02:00:4C:4F:4F:50 Mac dest: 02:00:4C:4F:4F:50 EtherType: Unknown [2021-04-30 11:45:45]: Payload Data : {1123}]
[2021-04-30 11:45:46]: Sender: Frame 02 has been sent.
[2021-04-30 11:45:46]: Receiver: Received message: [Mac src: 02:00:4C:4F:4F:50 Mac dest: 02:00:4C:4F:4F:50 EtherType: Unknown [2021-04-30 11:45:46]: Payload Data : {2123}]
[2021-04-30 11:45:47]: Sender: Frame 03 has been sent.

[2021-04-30 11:45:47]: Receiver: Received message: [Mac src: 02:00:4C:4F:4F:50 Mac dest: 02:00:4C:4F:4F:50 EtherType: Unknown [2021-04-30 11:45:47]: Payload Data : {3123}]
...
[2021-04-30 11:46:04]: Sender: Frame 20 has been sent.
[2021-04-30 11:46:04]: Receiver: Received message: [Mac src: 02:00:4C:4F:4F:50 Mac dest: 02:00:4C:4F:4F:50 EtherType: Unknown [2021-04-30 11:46:04]: Payload Data : {20123}]
[2021-04-30 11:46:05]: Sender: Frame 21 has been sent.
[2021-04-30 11:46:05]: Sender: Sending End of Transfer message to receiver...
[2021-04-30 11:46:05]: Receiver: Received message: [Mac src: 02:00:4C:4F:4F:50 Mac dest: 02:00:4C:4F:4F:50 EtherType: Unknown [2021-04-30 11:46:05]: Payload Data : {21123}]
[2021-04-30 11:46:05]: Receiver: Received message: [Mac src: 02:00:4C:4F:4F:50 Mac dest: 02:00:4C:4F:4F:50 EtherType: Unknown [2021-04-30 11:46:05]: Payload Data : {0}]
[2021-04-30 11:46:05]: finished 'send' in 20258.7164ms (+193ms cleanup)
[2021-04-30 11:46:06]: Receiver: All frames received successfully. Ending receiver script...
[2021-04-30 11:46:06]: finished 'receive' in 44937.036ms (+145ms cleanup)
```

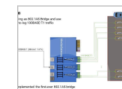
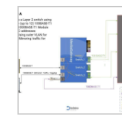

Lab 3

- And modify the source code to initialize the IP for the source and destination instead of using the static custom mac address that was used in the demo script.
- Where PC-1 was the sender and my partner PC-2 was the receiver.

```
send.py
1 #include libraries
2 from time import sleep
3
4 #used to add a sequence number to frames
5 count = 0
6
7 #set source and destination MAC addresses
8 g_ethernet_msg.mac_address_source = "00-13-3B-B0-0F-7F"
9 g_ethernet_msg.mac_address_destination = "00-13-3B-B0-11-09"
10
11 #timer function
12 def on_elapsed(source, event_args):
13     #inc counter
14     global count
15     count += 1
16     #make message using value of count as it's first byte and then send it
17     g_ethernet_msg.payload = System.Array[Byte](bytearray.fromhex('%02X'%count + "01 02 03"))
18     g_ethernet_msg.send()
19     #display a status message
20     print("Sender: Frame" + '%02d'%count + " has been sent.")
21
22 try:
23     #timer setup
24     timer = andi.create_timer()
25     timer.interval = 1000
26     timer.on_time_elapsed += on_elapsed
27     timer.start()
28     #wait 20 seconds
29     sleep(20)
30
31 finally:
32     #send 'end of transfer' message : {0} this message ends receiver script
33     print("Sender: Sending ""End of Transfer"" message to receiver...")
34     g_ethernet_msg.payload = System.Array[Byte](bytearray.fromhex("00"))
35     g_ethernet_msg.send()
36     #different event
37     timer.on_time_elapsed -= on_elapsed
38     timer.stop()
```

Lab 3

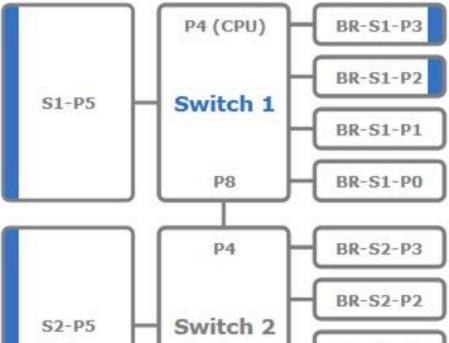
- On the second part from the lab we had to establish a VLAN connecting through MediaGateway.
- **Media Gateway** from [Technica-Engineering](#) is a development tool for testing and analyzing on-board vehicle networks.



Lab 3

- We had the chance in this part to understand how does MediaGateway interface works and what should we do to establish a VLAN connection, access to another gateway.
- Also how to connect the 2 PCs to the local webcam.
- I were responsible for the MediaGateway configuration on PC-1 and my partner for analyzing the connection through wireshark for PC-2.

Switch Status

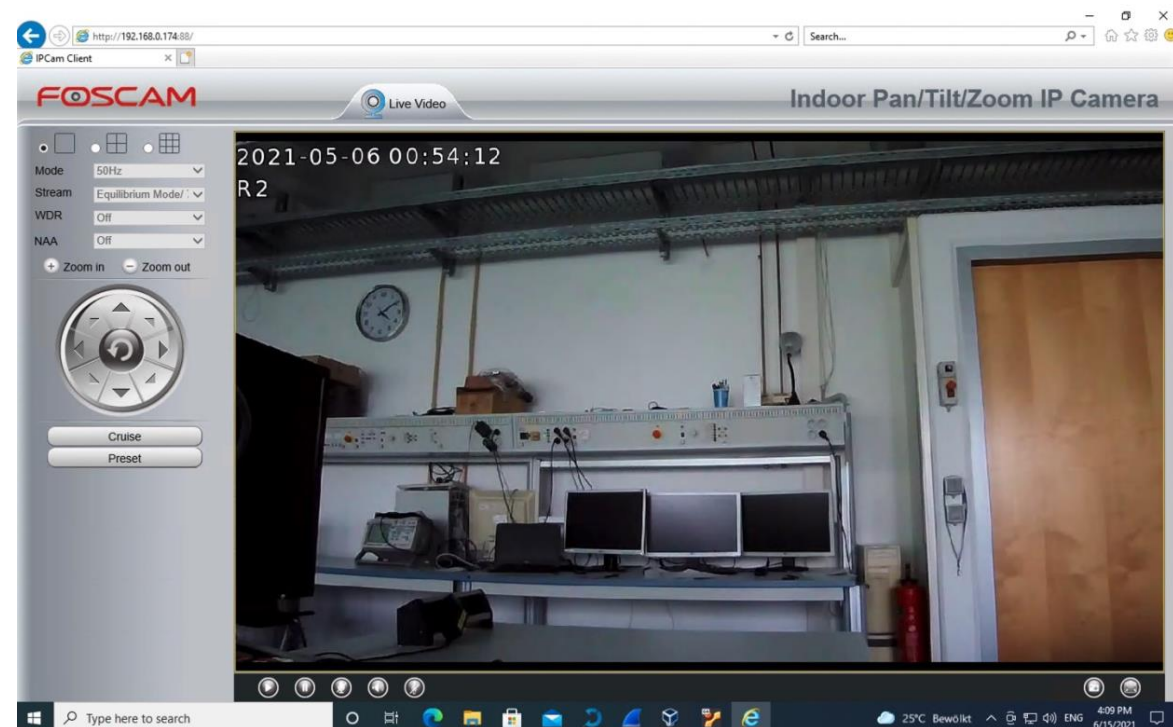


| Switch 1 | | | | |
|---|---------|----------|---------|------------|
| Functional Mode <input checked="" type="radio"/> Dynamic <input type="radio"/> Static | | | | |
| Address Resolution Table Export | | | | |
| MAC address | VLAN ID | Fwd port | Age bit | Static bit |
| 00:13:3B:B0:0F:7F | 085 | 3 | 1 | 0 |
| 70:B3:D5:28:DF:AF | 049 | 4 | 1 | 0 |
| 00:13:3B:B0:0F:80 | 049 | 5 | 1 | 0 |
| 00:13:3B:B0:11:09 | 085 | 2 | 1 | 0 |
| 00:62:6E:93:52:3F | 085 | 8 | 1 | 0 |

In order to gain access to the local webcam, ports S3-P8, S3-P4, S2-P8, S2-P4 and S1-P8 are configured to be a member of VLAN 85.

Lab 3

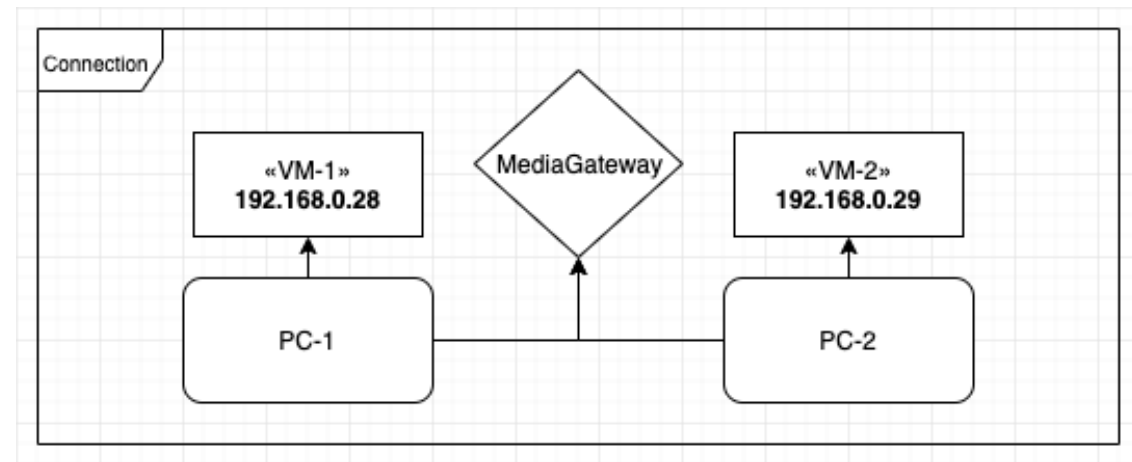
- On the last task we had to establish a connection through MediaGateway and central MediaGateway.
- Where we need to setup PC1: Central Webcam – MediaGateway Configuration and PC2: Central Webcam – Local Webcam.



PC1 access to the Central Webcam

Project

- We have chosen the VSomiIP project, where we had to establish a connection between 2 virtual machines through MediaGateway using SomeIP protocol.
- Both of us were responsible for the work on the project part.



SOME/IP

- SOME/IP is an automotive middleware solution that can be used for control messages.
- Provides service oriented communication over a network.
- It is based on service definitions that list the functionality that the service provides.
- A service can consist of combinations of zero or multiple events, methods and fields.

SOME/IP

- Serialization describes the way data is represented in protocol data units (PDUs) as payload of either UDP or TCP messages, transported over an IP-based automotive in-vehicle network.
- SOME/IP Header Format

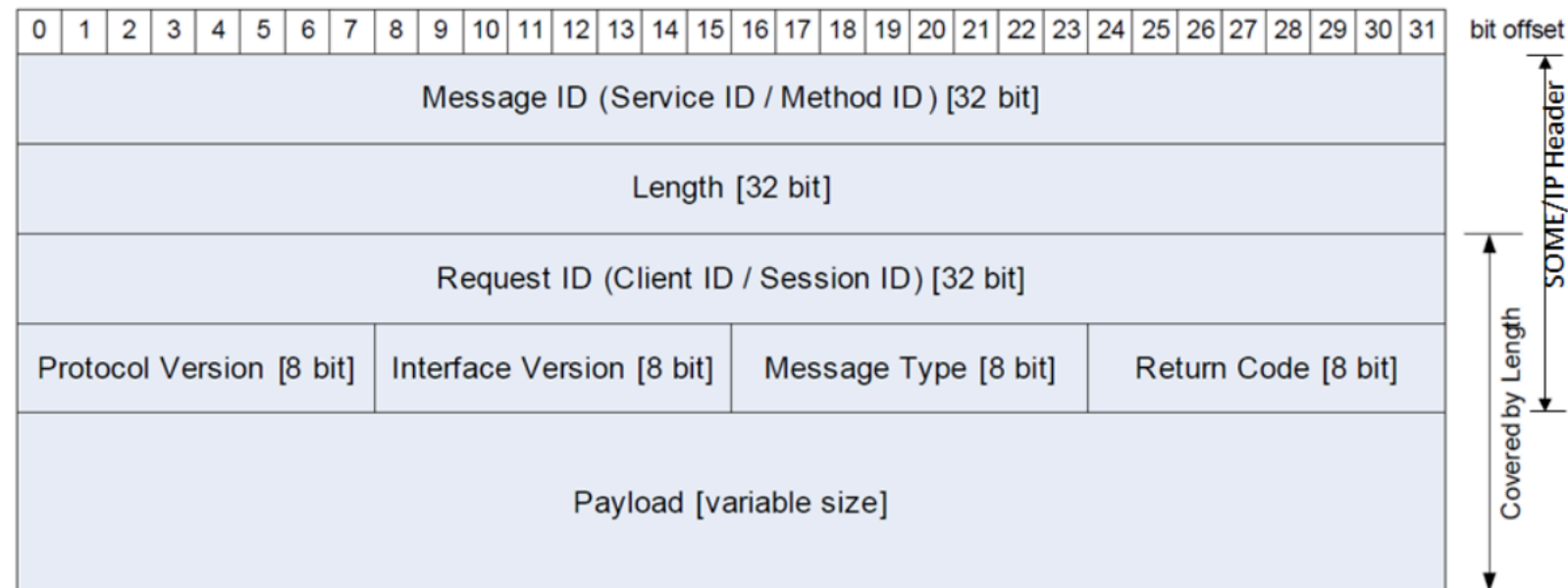


Figure taken from: *SOME/IP Service Discovery* by Dr. Lars Völker

Project

- And the second task was to model one of the nodes using ANDi.
- We need to be able to communicate between Linux computers and Andi on Windows.

The screenshot displays the ANDi 0.15.0 software interface. The main window shows a Python script for creating and sending a SOME/IP message. The script is as follows:

```
1 from globals import *
2
3 someip_msg = message_builder.create_someip_message()
4
5 # set source and destination IP and Mac addresses
6 someip_msg.ip_header.ip_address_source = "192.168.0.210"
7 someip_msg.ip_header.ip_address_destination = "192.168.0.28"
8 someip_msg.ethernet_header.mac_address_source = "00:13:3B:80:11:0A"
9 someip_msg.ethernet_header.mac_address_destination = "08:00:27:6B:2A:9E"
10
11 #set source and destination port numbers
12 someip_msg.transport_header.port_destination = 30509
13 someip_msg.transport_header.port_source = 30610
14
15 #set important header fields
16 someip_msg.someip_header.message_id = 0x11113333
17 someip_msg.someip_header.request_id = 0x44440001
18 someip_msg.someip_header.protocol_version = 0x01
19 someip_msg.someip_header.interface_version = 0x00
20 someip_msg.someip_header.message_type = MessageType.REQUEST
21 someip_msg.someip_header.return_code = ReturnCode.OK
22
23 #set payload data to send "World"
24 someip_msg.payload = System.Array[Byte]([0x57, 0x6F, 0x72, 0x6C, 0x64])
25
26 #send Message
27 someip_msg.send()
28
```

The interface also shows a Project Explorer on the left with a tree view containing Channels, Receiver, Logging, Variables, Modules, Timers, Messages, someip_msg, Databases, Nodes, and Scripts. The Scripts folder is expanded, showing a script named test_vsoneip. The Scripts Output panel at the bottom right shows the execution log:

```
[2021-06-18 15:28:30]: Start 'test_vsoneip'
[2021-06-18 15:28:31]: finished 'test_vsoneip' in 265.204ms (+176ms cleanup)
```


Project Conclusion

- In order to establish service communication:
- The services must be made known in the network where this is done using the service discovery protocol.
- First, a server announces its services where the IP and port of the service are also transmitted.
- If a client needs data from a service, it will subscribe to this at the server.

