GUI for a Microgrid's Control System

Shouran Mu

supervised by Edwin Mora

TECHNISCHE UNIVERSITÄT DARMSTADT

EINS
ENERGY INFORMATION NETWORKS AND SYSTEMS

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1 **Introduction**

2 **Methodology**

3 **Implementation**

4 **Outlook**

Introduction - Motivation

GUI Design

LabVIEW
Implemetation

OPC Unified
Architecture

Real-time
Simulation

Introduction - Challenges

TECHNISCHE UNIVERSITÄT DARMSTADT

1

GUI designed in LabVIEW

How to present comprehensive information with GUI?

2

OPC Unified Architecture

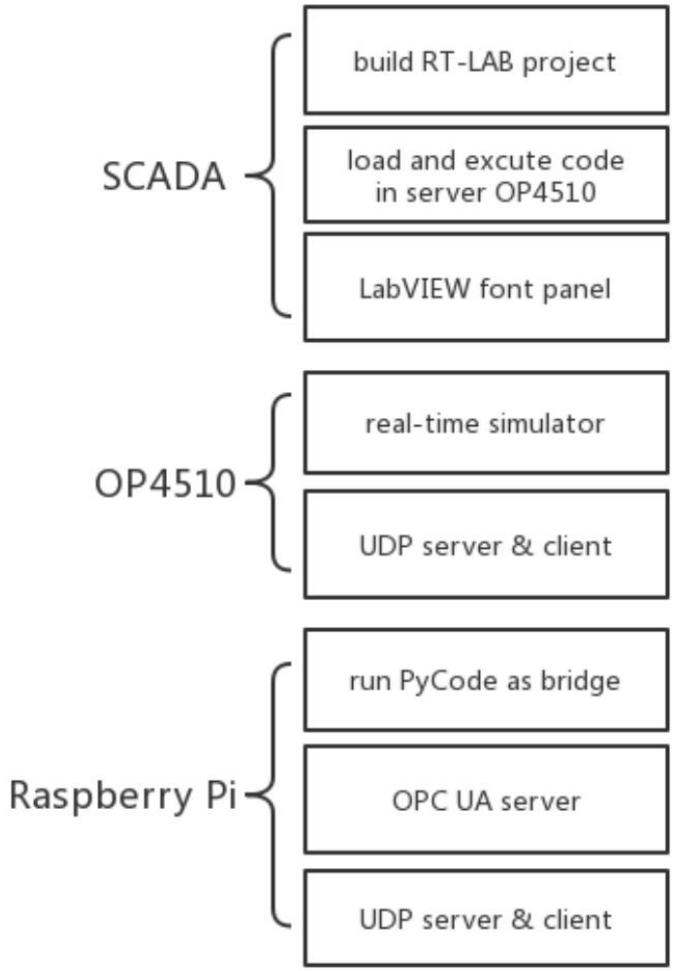
How to build communication based on OPC UA?

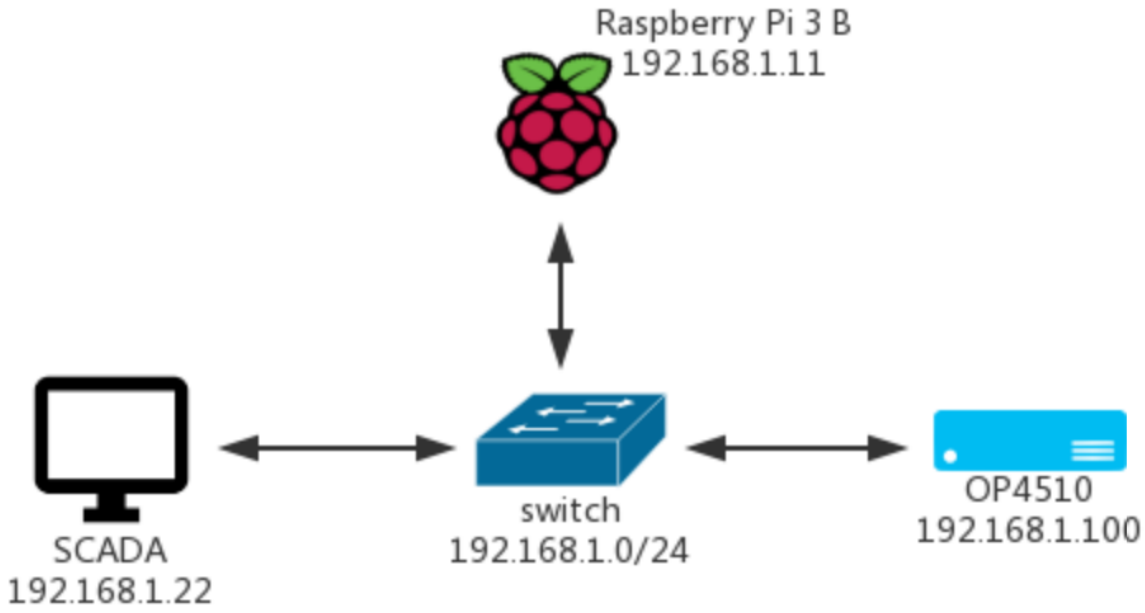
3

Raspberry Pi as Bridge

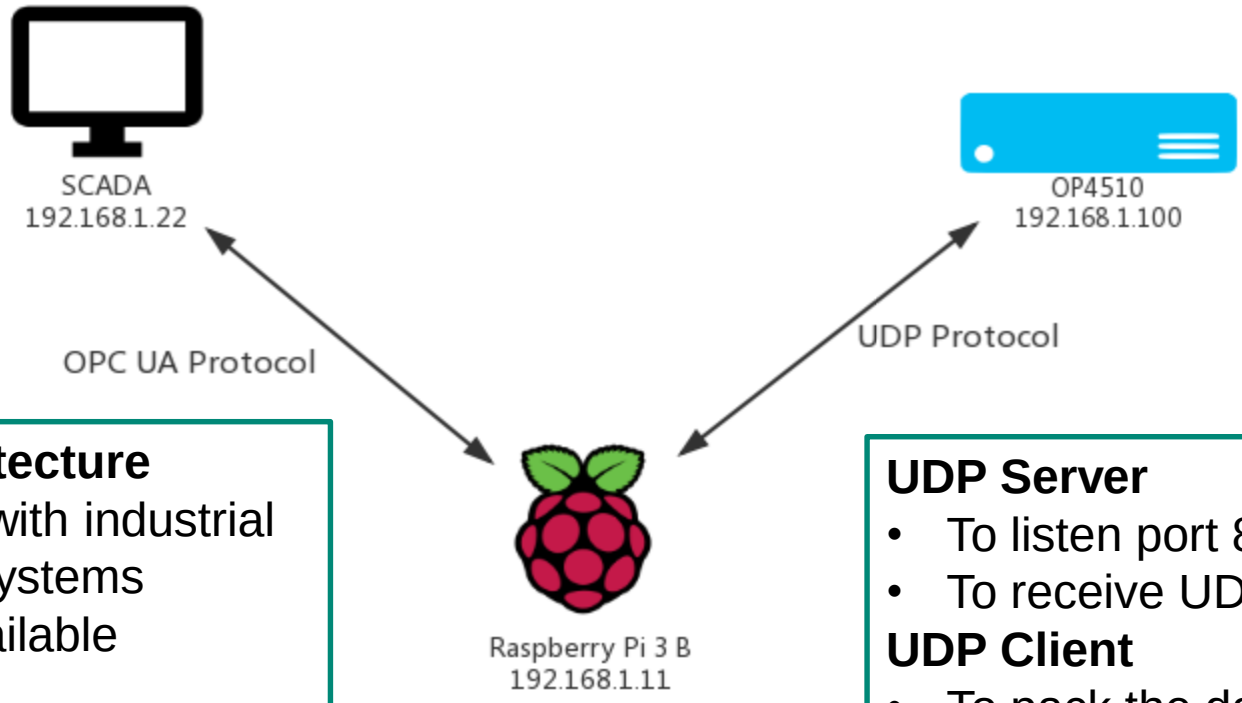
How to coordinate different communication protocols?

Methodology - Main Topology





Methodology – Transmission Protocol



OPC Unified Architecture

- Communication with industrial equipment and systems
- Open - freely available
- Cross-platform
- Robust security
- ....

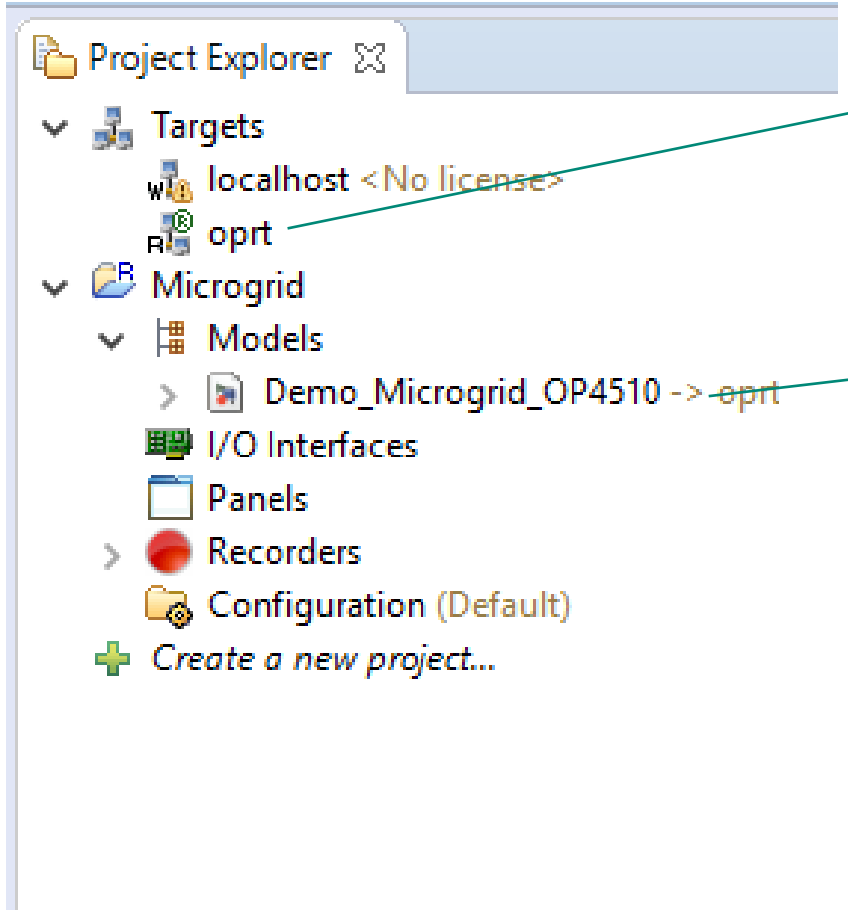
UDP Server

- To listen port 8001
- To receive UDP packet

UDP Client

- To pack the data
- To send UDP packet

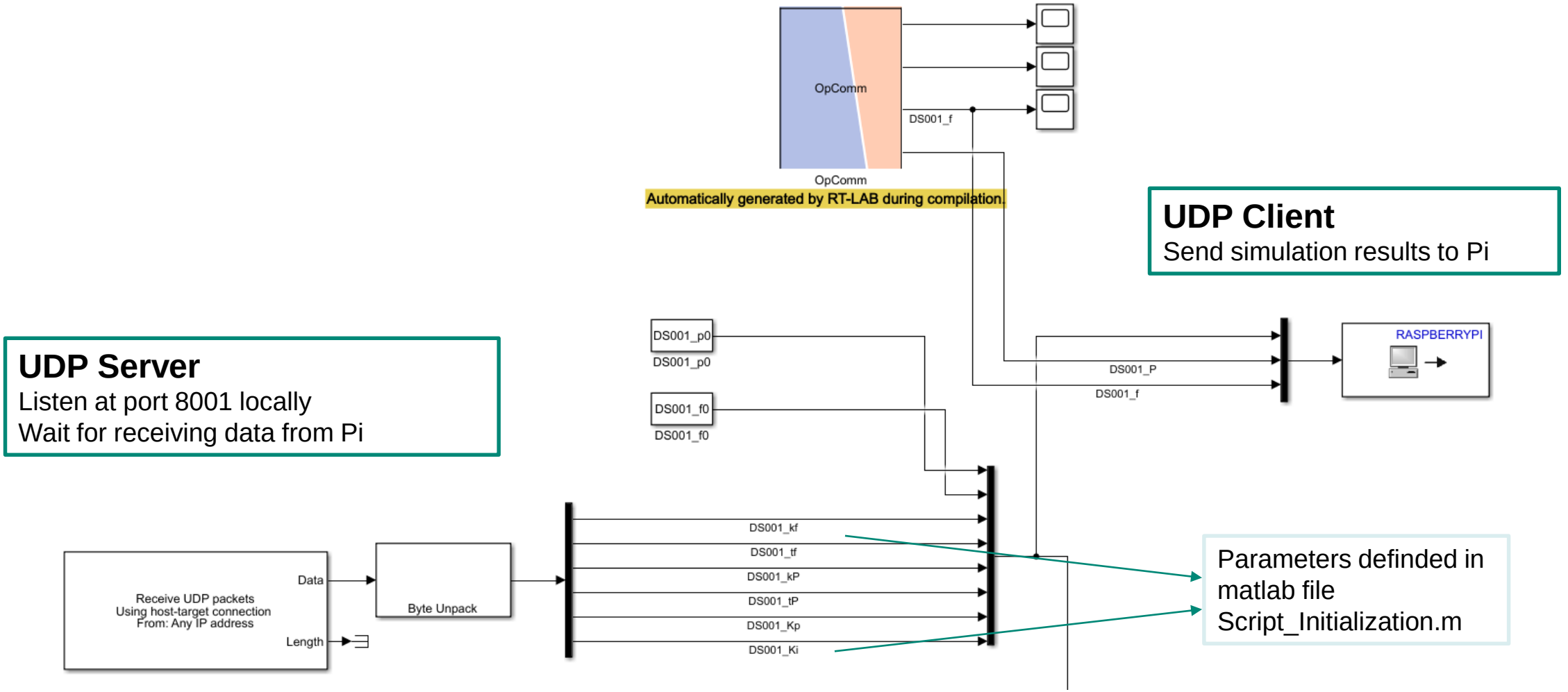
Implementation - RT-LAB Project



OPAL real-time server
IP: 192.168.1.100

- Assigned to server target
- Load and excute model
- Simulation in real-time

Implementation - Matlab Simulink



Implementation – Service Configuration

```
############################################## OPC Servi
param_list = read_param_from_txt()

server = Server()
# start opc port in pi
#opc_url = "opc.tcp://192.168.1.11:4840/"
opc_url = "opc.tcp://" + Local_IP + ":4840/"
server.set_endpoint(opc_url)

# register namespaces for 4 diff devices
namespace_list = ["DS001", "DS002", "Battery", "Photovol
object_list = create_namespace_for_object(namespace_list

#start opcua server
server.start()
print("OPCUA Server started at {}".format(opc_url))
```

Load the parameters name from txt file
e.g. GasLevel, P, f, kf, tf, kP, tP...

Create namespace for each device
e.g. DS001, DS002, BT001, PV001

Start OPC server at port 4840 locally

Bind UDP server with port 8001 for listening

```
############################################## UDP Server Config #
# UDP Server used for listening
#IP = "192.168.1.11"
#PORT = 8001
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
address = (Local_IP, Local_Port)
server_socket.bind(address)
print("UDP server start at {}".format(Local_IP))

############################################## UDP Client Config #
# UDP target ip and port for sending data
Target_IP = "192.168.1.22"
PORT_send = 8002
client=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
```

Send UDP packet to server with host IP & port 8002

Implementation – Service

TECHNISCHE UNIVERSITÄT DARMSTADT

```
############ read changed parameters from Labview ############
kf_var = server.get_node("ns=2;i=8").get_value()
tf_var = server.get_node("ns=2;i=9").get_value()
kP_var = server.get_node("ns=2;i=10").get_value()
tP_var = server.get_node("ns=2;i=11").get_value()
Kp_var = server.get_node("ns=2;i=12").get_value()
Ki_var = server.get_node("ns=2;i=13").get_value()

print("changed kf_node: ", kf_var, " tf_var: ", tf_var, " kP_

############ send the changed value back to op4510 ############
#client.sendto(struct.pack('d',kf_var),(Target_IP,PORT_send))
client.sendto(struct.pack('dddddd',kf_var,tf_var,kP_var,tP_va

############ receive data from op4510 server ############
receive_data, client_address = server_socket.recvfrom(80)
#print(receive_data)
#unpack the data with double format
receive_data = struct.unpack('ddddddddddd', receive_data)
#print("received data from UDP : ", receive_data)
```

Read value by specified namespace and node id

Pack and send updated value to server OP4510 in UDP packet for simulation

Receive simulation results from server OP4510

Implementation – Service

TECHNISCHE UNIVERSITÄT DARMSTADT

Unpack and prepare the values for building parameters

Insert some constant values (which are not provided by RT server) to build parameters list

Set complete parameters list to LabVIEW to display on each control panel

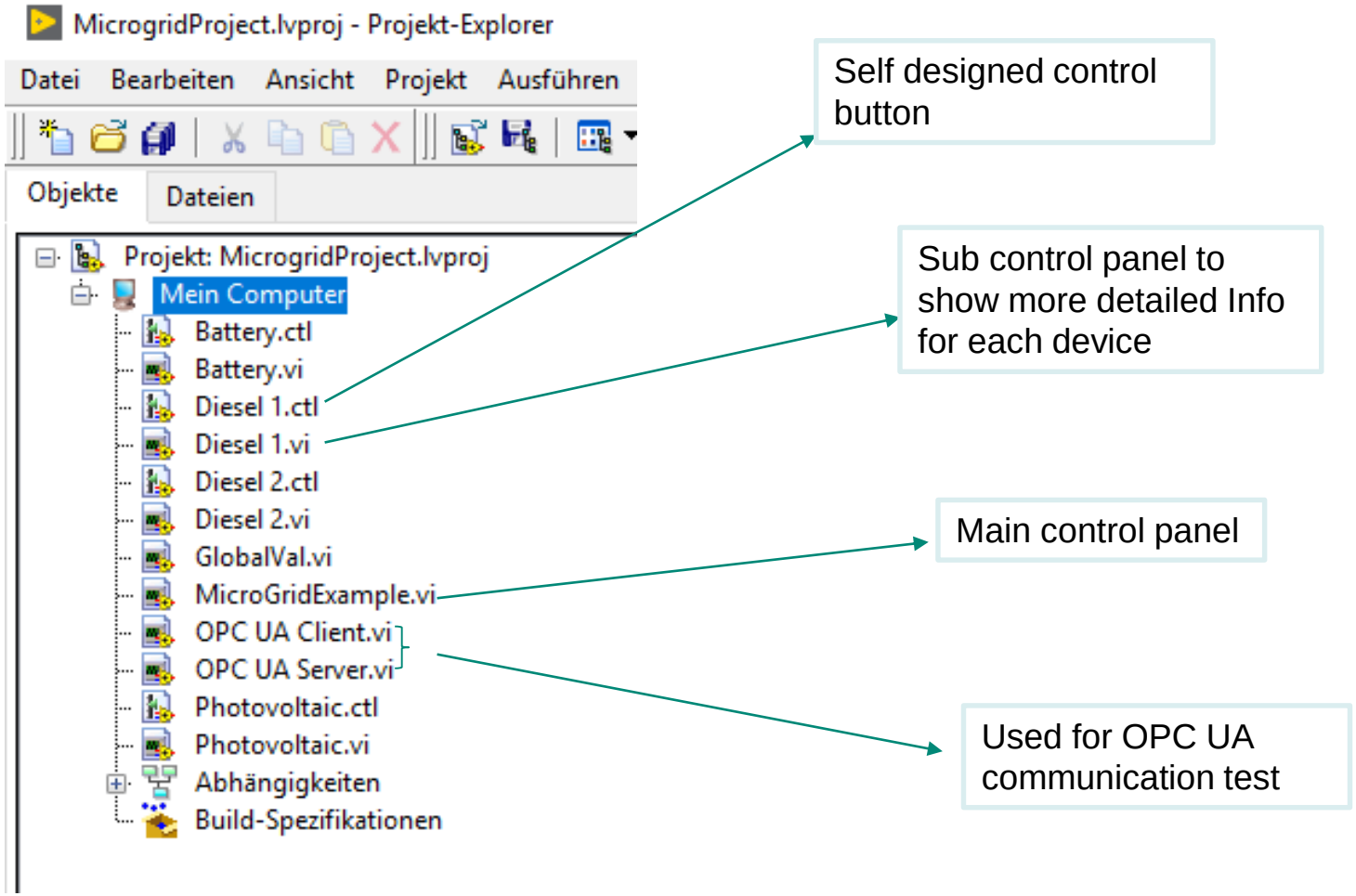
```
############ prepare the param_list ############
#prepare parameters sent to the Labview
data_list=[]
#gas level
data_list.append(randint(0,100))
for item in receive_data:
    #2 digits after dot
    data_list.append(float('%.3f' % (item)))

#insert some missing parameters
#Pmin, Pmax
data_list.insert(3, 500)
data_list.insert(4, 500)
#opMode
data_list.insert(5, 0)

#print ("parametsers data : ", data_list)

############ send updated param_list to Labview ##
for object_item in object_list:
    var_list = object_item.get_variables()
    for var in var_list:
        # set node value
        index = var_list.index(var)
        var.set_value(float(data_list[index]))
        # get node info and node value
        #print(str(var) + " ---- " + str(var.get_v
    #print(object_item.get_display_name())
```

Implementation - LabView Project



Self designed control button

Sub control panel to show more detailed Info for each device

Main control panel

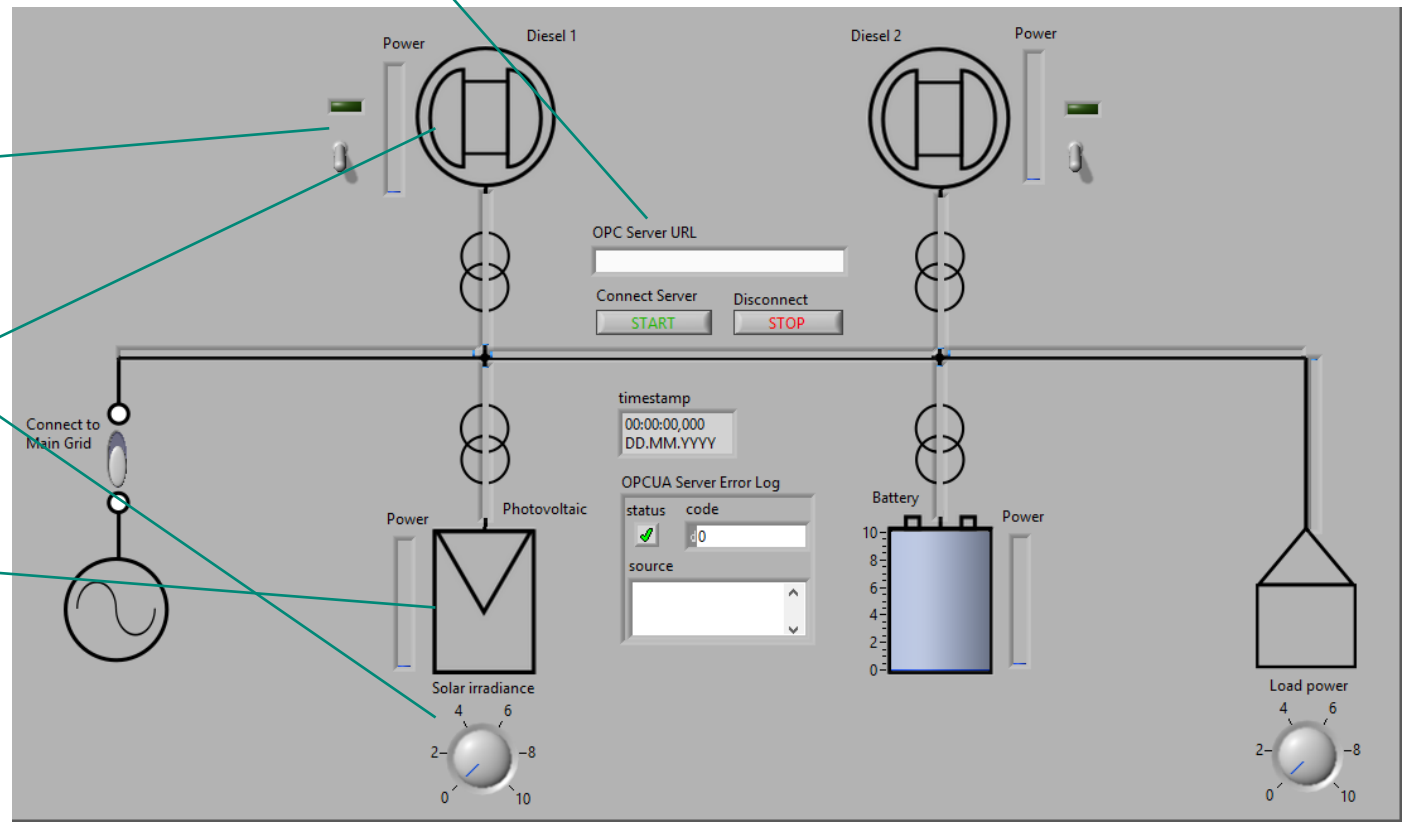
Used for OPC UA communication test

Implementation - Main Control Panel

URL used to connect to OPC server in Pi

Connect the device to microgrid by switch

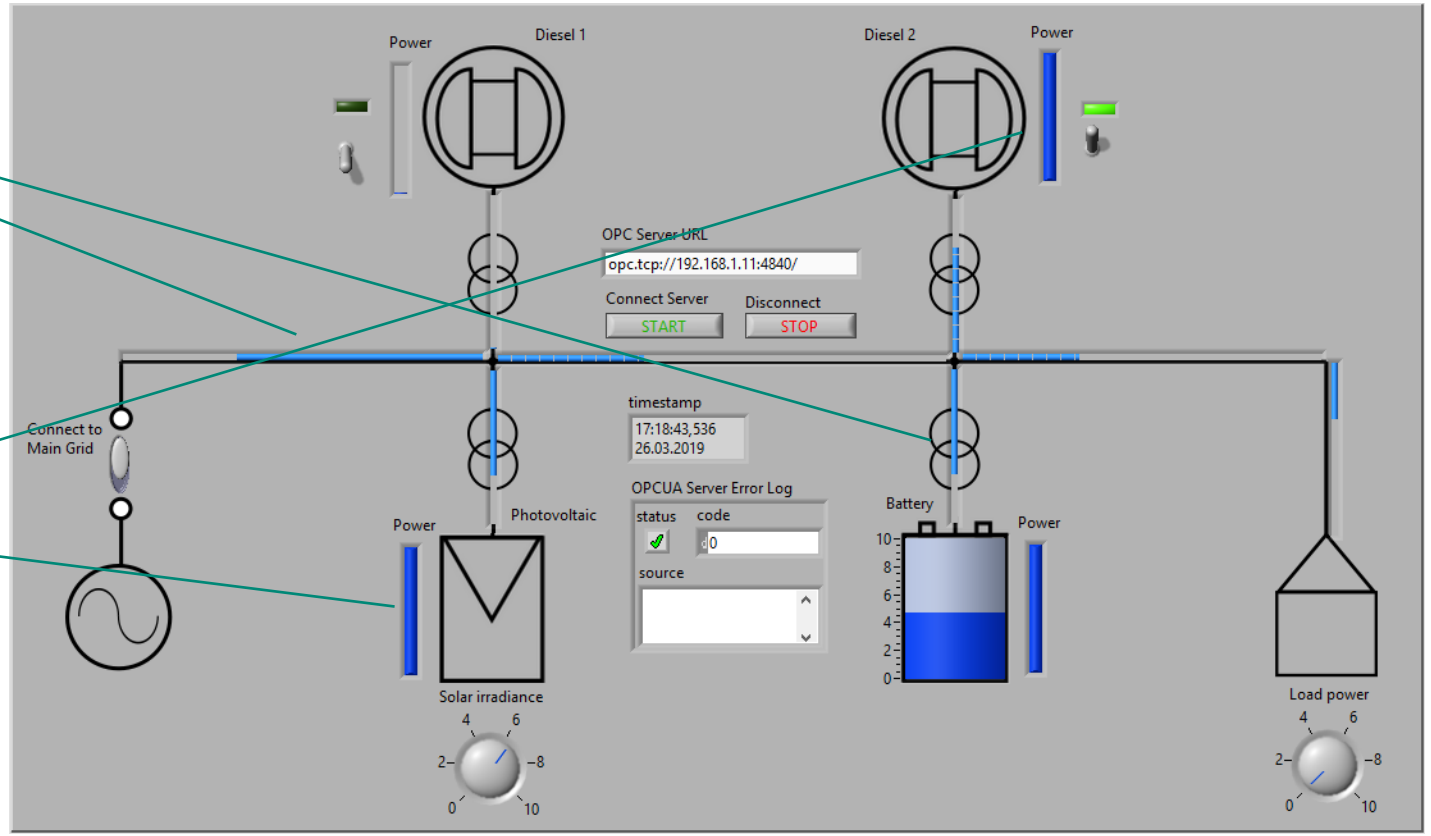
Image button to open sub control panel



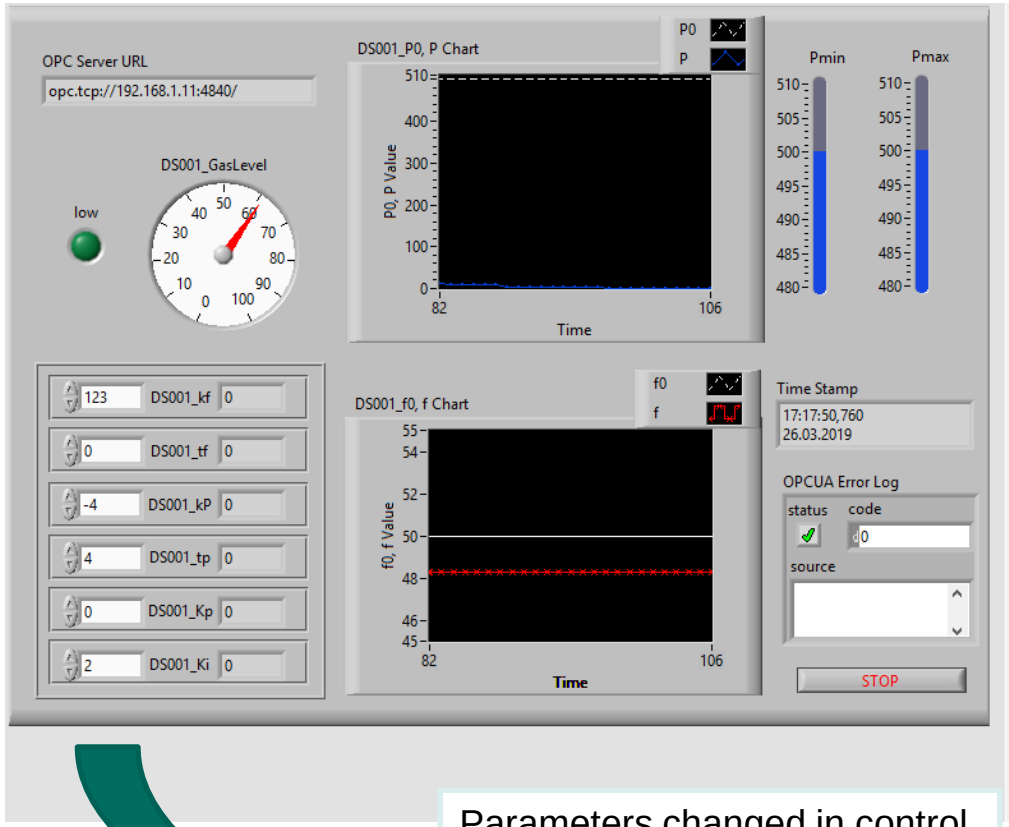
Implementation - Main Control Panel

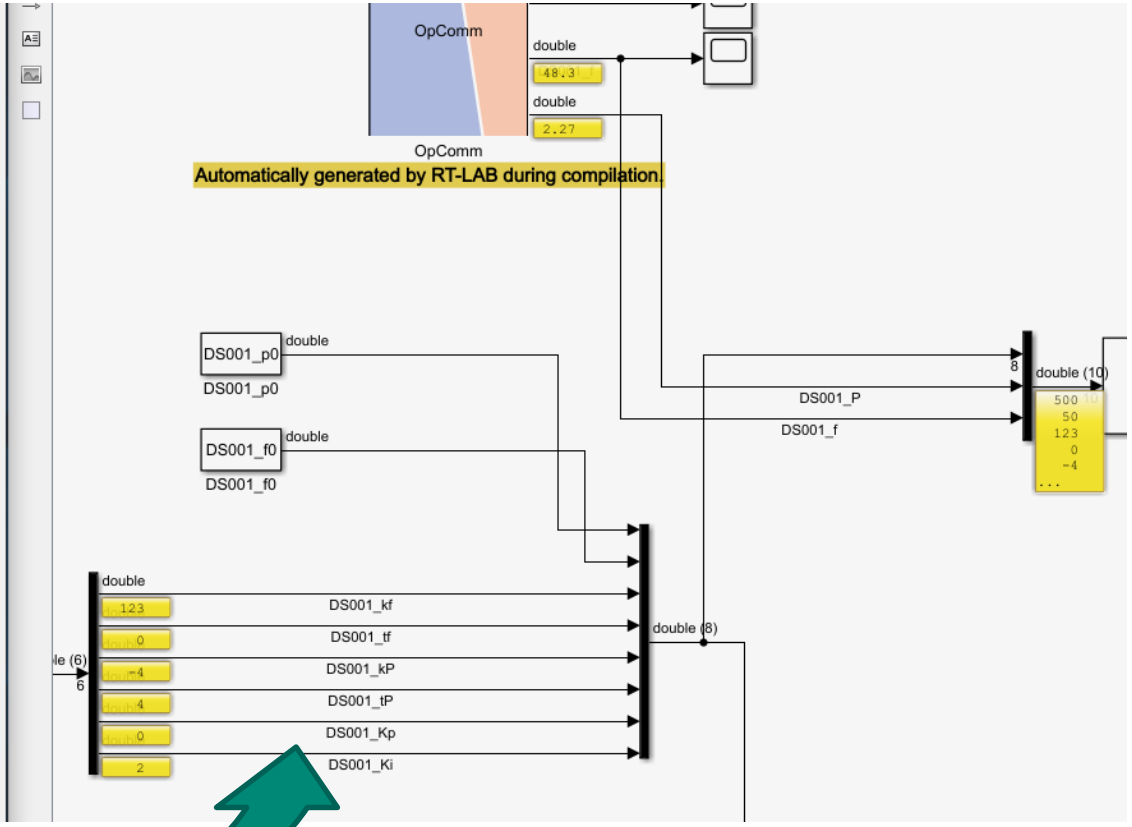
Energy flow animation to show more information dynamiccally

Power slide shows power capacity of the PV device



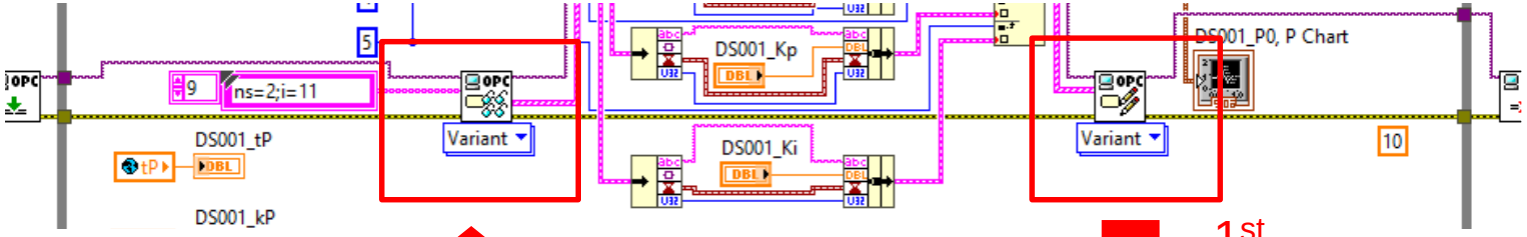
Implementation - Sub Control Panel





Parameters changed in control panel will be sent to RT server online (or in real-time)

Implementation - Example



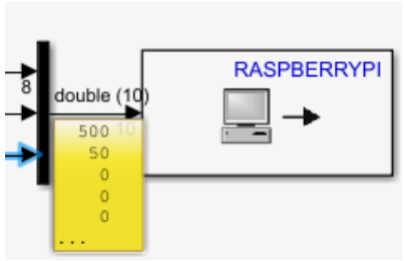
1st

Changed value will be updated by LabVIEW OPC UA Toolkit

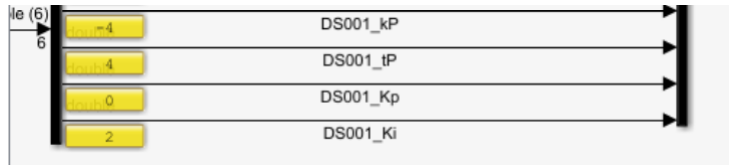
```
receive_data, client_address = server_socket.recvfrom(80)
#print(receive_data)
#unpack the data with double format
receive_data = struct.unpack('ddddddddddd', receive_data)
```

```
tP_var = server.get_node("ns=2;i=11").get_value()
```

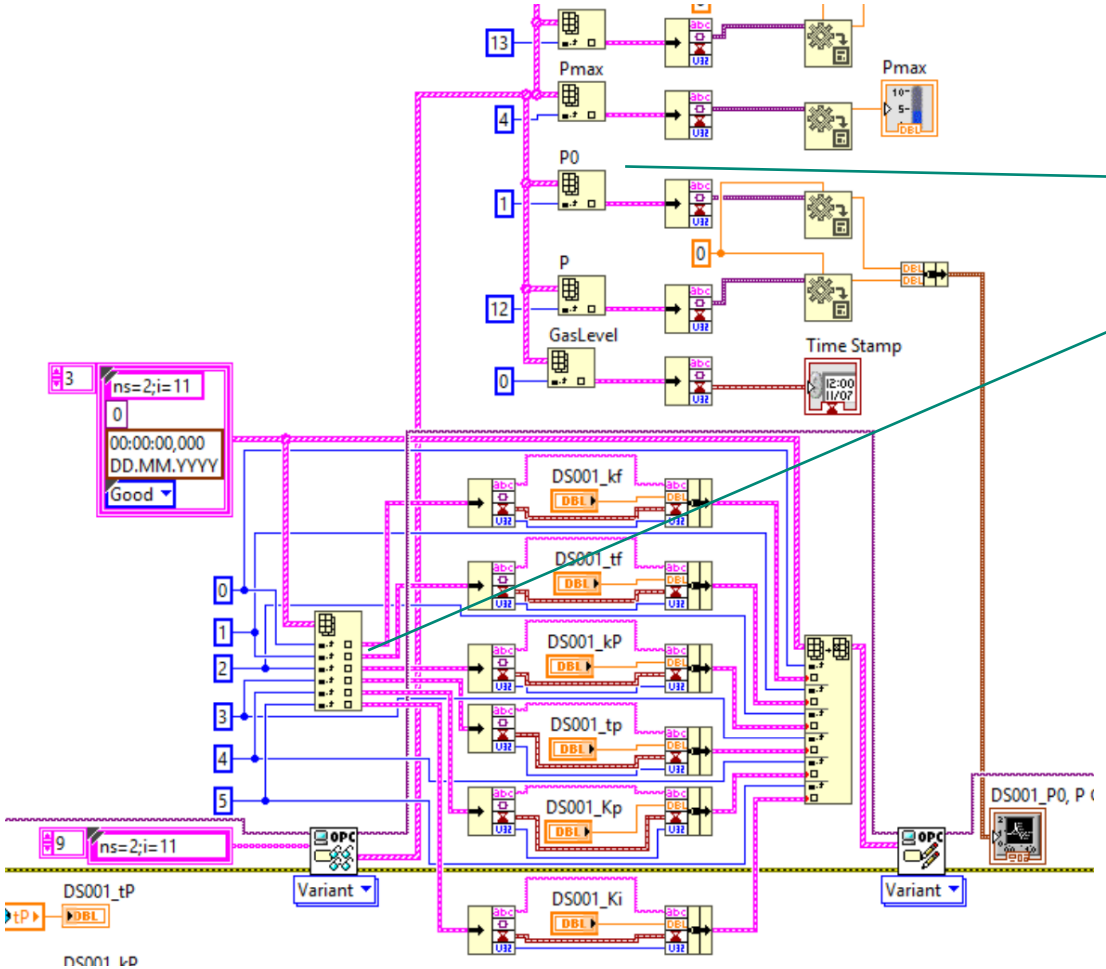
```
client.sendto(struct.pack('dddddd',kf_var,tf_var,kP_var,tP_var,Kp_var,Ki_var),(Target_IP,PORT_send))
```



Simulation and sent results with Simulink UDP Transmission Element



Outlook – to be better...



How to improve the data flow for a big set of parameters rather than using index?

Test this application with a real microgrid model?

Question about ...