Manual

XELA Sensor Server
Standalone for Linux and Windows
v.1.4.0

Table of contents

Using the Server

Prerequisites:

Primary requirement is to run the code with **Python 3.7.5** as the files have been pre-compiled and therefore might give an error if running on different Python version

The following packages are required to run the sensor service and tools:

1) Tkinter
2) numpy
3) matplotlib (version 3+ recommended)
4) python-can
5) psutil (normally installed)
6) python-socketio
7) asyncio (requires Python 3.7+)
8) uvicorn

Set up

Manual

Unpack the files to suitable location.

Run configuration tool for can
- Windows PowerShell: ./xela_conf.exe
- Windows cmd: xela_conf.exe
- Linux: ./xela_conf

Run the server
- Windows PowerShell: ./xela_server.exe
- Windows cmd: xela_server.exe
- Linux: ./xela_server

Run the visualizer
- Windows PowerShell: ./xela_viz.exe
- Windows cmd: xela_viz.exe
- Linux: ./xela_viz

Run the logger
- Windows PowerShell: ./xela_log.exe
- Windows cmd: xela_log.exe
- Linux: ./xela_log

Arguments

Following is the list of arguments that can be given to the executables (green – used, red – not used):

-f *filename* The name of configuration file (with .ini extension) to be used (full path or just name if in same folder) (default: */etc/xela/xServ.ini* (Linux) or *xServ.ini* (Windows)
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_server -f myconf.ini

--ip *ip* The IP address for the computer running the server (if same computer, it would be *127.0.0.1* aka *localhost*)
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_viz --ip 192.168.0.107

-p *port* The port for communicating the server (default: *5000*)
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_viz -p 5007

--viz *sensor_number* The sensor number to display in visualization (throws an error if sensor doesn't exist)
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_viz --viz 3

-s *sensor_config* The string for configuring logging script (default: *1:100*)
more info in How to use XELA Logger
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_log -s 1-3:1000

--log *log_file* The path to log file (default: */etc/xela/LOG/app.log* in Linux or *LOG/app.log i*n Windows where app corresponds to app name)
more info in How to use XELA Logger
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_server --log my_server_log.log

-d *device* The can device name (default: *socketcan* in Linux or *esd* in Windows)
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_conf -d pcan

-c *channel* The can device channel (default: *can0* in Linux or *0* in Windows)
Usage: [xela_conf, xela_log, xela_server, xela_viz]
Example: ./xela_conf -c PCAN_USBBUS1

How to use XELA Logger

Under the -s flag (stands for sensor config), you can write the info in following formats that can be combined:

Data	Meaning
1	Sensor* 1 will be read (default 100 broadcast cycles*)
1,2,3	Sensors 1, 2 and 3 will be read (default 100 broadcast cycles)
1-3	Sensors 1,2 and 3 will be read (default 100 broadcast cycles) (shortcut)
1:150	Sensor 1 will be read 150 broadcast cycles
1:150,2	Sensor 1 will be read 150 broadcast cycles and sensor 2 for default 100
1-3:150	Sensors 1, 2 and 3 will all be read for 150 broadcast cycles
1-5,2:150	Sensor 2 will be read 150 broadcast cycles and sensors 1, 3, 4 and 5 for default 100
1:60t	Sensor 1 will be read for 60 seconds
1-3:30t	Sensors 1, 2 and 3 will all be read for 30 seconds
1-5,2:30t	Sensor 2 will be read for 30 seconds and sensors 1, 3, 4 and 5 for default 100 broadcast cycles
1-5:200,3:30t	Sensor 3 will be read for 30 seconds and sensors 1, 2, 3 and 5 will all be read for 200 broadcast cycles

Note

*If sensor is defined multiple times, latest entry counts
*Sensors can be from 1 to 16
*Broadcast cycles depend on the sensor and computer configuration and is defined as sensor read cycle and backup update cycle combined

Configuration file format

Due to the limitations of the esd CAN-USB device driver you are unable to use automatic configuration tool

```
[CAN]
bustype = socketcan *1
channel = can0 *2
[viz]
max_offset = 40 *3
max_size = 50 *4
[sensor]
num_brd = 2 *5
brd_height = 4 *6
brd_width = 4 *7
ctrl_id = 6 *8
m = 5 *9
chan = 0 *10
[sensor2]
brd_height = 4
brd_width = 6
ctrl_id = 7
m = 5
chan = 0
mirror_x = 1 *11
mirror_y = 1 *12
rotate = 2 *13
```

Sections in config file:
[CAN] – defines section for CAN bus
[viz] – defines settings for visualization
[sensor] – defines primary sensor settings and number of total boards
[sensor*N*] – (*N* is a number above 2, sequential) – defines Nth sensors settings, if different from primary

Notes:
*1 Bus types for Linux are socketcan, slcan and pcan and for Windows esd, pcan
*2 Channel for **pcan** is *PCAN_USBBUS1*, for **socketcan** *can0* and for **esd** *0*
*3 Defines how far the bubbles will go in visualization
*4 Defines how big the pressure bubble will be
*5 Defines the total number of boards
*6 Defines the number of rows on the sensor (visually only)
*7 Defines the number of columns on the sensor
*8 Defines controller ID
*9 Defines sensor mode (default: 5)
*10 Defines channel (default 0, in multi-mode (2 sensors on one controller) use 2 for second sensor)
*11 Defines for visualizer to mirror display of taxels on x-axis (optional)
*12 Defines for visualizer to mirror display of taxels on y-axis (optional)
*13 Defines for server to rotate the sensor by 180 degrees before outputting the data (optional)

Data format

As the data is provided in a JSON string, there are few ways to access it and it has following format:

{'data':'FA00,FA00,FA00,FB00,FB00,FB00...' , 'sensor': '1'}

data contains X, Y and Z values in HEX and they are comma-separated for all taxels on a sensor

sensor has the number of the sensor in string format (important when requesting multiple sensors from server)

Supported USB-CAN devices

Following USB-CAN devices are supported in Linux:

- esd CAN-USB/2 (bus: esd)
- PEAK USB-CAN (bus: pcan, default channel: CAN_USBBUS1)
- CANable and CANable Pro (with candlelight firmware on Linux)

Example usage:

```
#!/usr/bin/env python3

import asyncio
import socketio
loop = asyncio.get_event_loop()
sio = socketio.AsyncClient()

@sio.event
async def connect():
    print("Connection established")

@sio.event
async def sensor_data(data):
    #normally the taxel data is in data element as list type string
    print(data[u"data"])

@sio.event
async def disconnect():
    print("Disconnected from server")

async def start_server():
    #using loop makes the application wait for connection in case server takes too long to start
    ncd = True
    while ncd:
        try:
            await sio.connect("http://localhost:5000", namespaces=["/sensor1"])
        except socketio.exceptions.ConnectionError:
            pass
        else:
            ncd = False
            break
    try:
        await sio.wait()
    except KeyboardInterrupt:
        exit()
loop.run_until_complete(start_server())
```

Common errors

Error	Reason
Module not found	Use pip to install the mentioned module and try again. If the module is already installed, try reinstalling. On systems with several Python versions, you might need to specify which pip you are using
Could not start CAN: OSError: [Errno 19] No such device	Make sure to pull up the network with one of the following commands, if using Linux: user@localhost:~$ sudo ip link set can0 type can bitrate 1000000 user@localhost:~$ sudo ip link set up can0 or user@localhost:~$ sudo slcand -o -s8 -t hw -S 3000000 /dev/ttyUSB0 user@localhost:~$ sudo ifconfig slcan0 up In case of Windows, make sure the adapter is connected and using specified channel (esd channel might not be 0)
The MATPLOTLIBDATA environment variable was deprecated in Matplotlib 3.1 and will be removed in 3.3	This can be ignored as it is caused by incompatibility between PyInstaller and MatPlotLib For future builds we will try to remove the warning by modifying the libraries involved.

If you find errors, not listed in this file, please send an email regarding it to info@xelarobotics.com
Do not forget to attach files from /etc/xela/LOG folder in Linux or LOG folder in Windows (if using default log folder) with the terminal log (if you copy-paste, there might be some strange characters, don't worry about them, we can read them) and description of the failure. If you have made your own code that is causing errors, please attach it as well.