Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Московский политехнический университет»

**факультет информационных технологий**

**Кафедра СМАРТ-технологий**

Дисциплина: Разработка систем сбора и обработки данных

Отчёт по лабораторной работе №3

«Обработка структурированных пакетов данных и анализ статистики»

Работа выполнена (ФИО):

Ф И О

Научный руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф И О

**Цель работы**

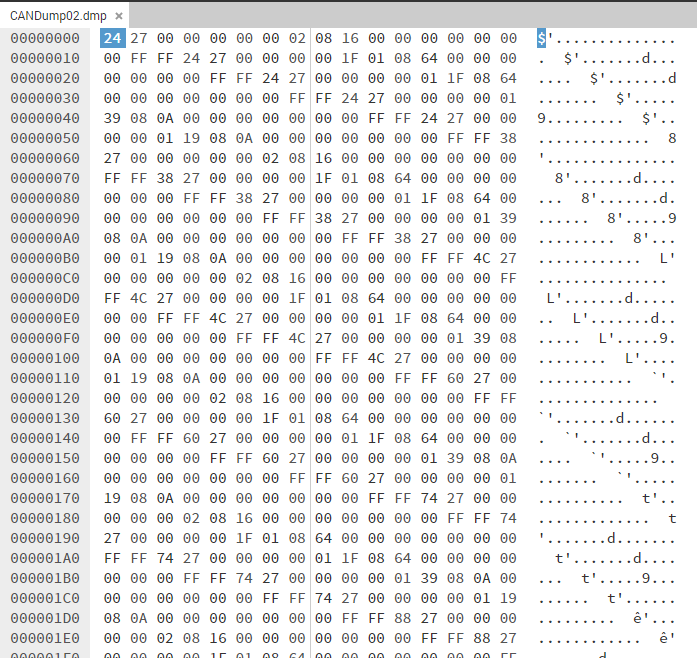
Разработать алгоритм считывания и статистического анализа данных, поступающих по последовательному протоколу J1939 (CAN).

**Задачи**

* Подготовить приложение считывания данных из файла дампа;
* Разработать функцию динамического обновления данных в заданной структуре пакетов и накопления статистики;
* Разработать функцию визуализации связей (графа передачи пакетов) между узлами сети;
* Разработать функцию фильтрации данных по идентификаторам источника и приемника;
* Разработать функцию детектирования изменения значений и указания степени изменений;
* Разработать метод отслеживания взаимодействия между несколькими потоками обмена сообщения;
* Разработать инструмент формирования и серийной отправки запроса (выводить в заданное устройство обмена) подготовленного пакета CAN с введенными пользователем параметрами. Обеспечить копирование приходящих запросов в шаблон для отправки. Обеспечить подтверждение отправки пакетов в текстовом окне.**Ход работы**

Данные из дампа CAN-шины считываются пакетами по 19 байт в каждом. Окончанием пакета служит пара из байтов 0xFF (255)

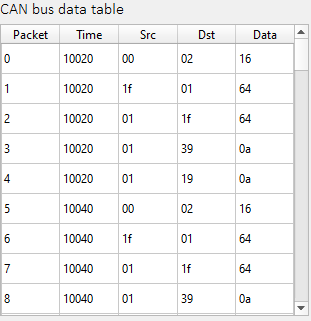
Структура файла выглядит следующим образом:



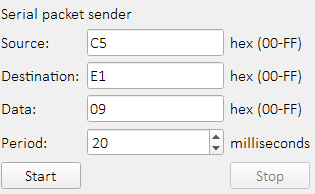
Для запуска потока чтения файла (вместо потока из файла можно использовать, например, UDP порт). На форме имеется кнопка Load data from file

При нажатии на эту кнопку, файл дампа открывается для чтения. А также, запускается отдельный поток, читающий отдельные байты в буфер. Как только будет обнаружена пара из символов 0xFF, буфер будет обработан и передан в функцию proceed\_packet, которая проверит его по таблице разрешённых адресов (Whitelist) и, если адреса отправителя и получателя разрешены, выгрузит в массив данных, содержащий 2 байта времени, байт адреса отправителя, получателя и байт данных

Все разрешённые считанные пакеты (из файла, добавленные вручную или любым иным образом) отображаются в таблице на форме.

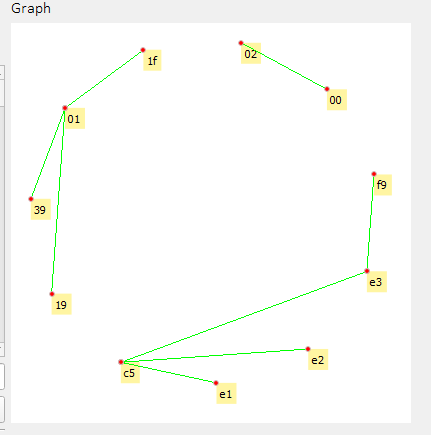


Вторым способом отправки пакетов является автоматическая серийная отправка пакетов с инициализацией на форме. Для этого, на форме, необходимо указать адрес отправителя, получателя, передаваемый байт данных и интервал времени, через который данный пакет будет циклически отправляться.



При нажатии кнопки Start, запускается таймер для циклической отправки пакета через равные промежутки времени.

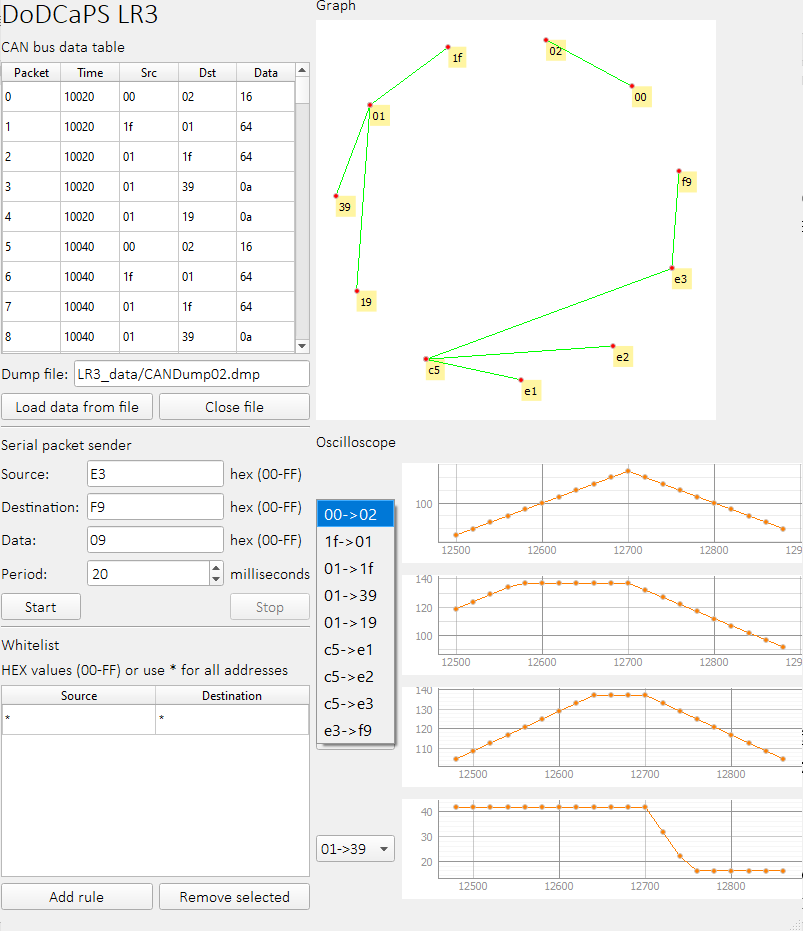
Узлы всех принятых пакетов (после фильтра) и связи между ними отображаются на адаптивном графе:



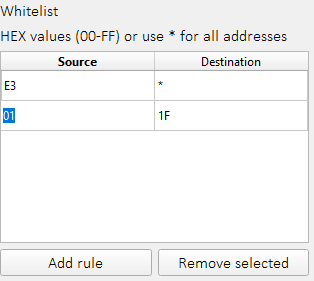
Граф периодически обновляется в процессе работы и самостоятельно распределяет новые точки по окружности, а также рисует связи между узлами и название узлов.

Вместе с графом, автоматически обновляется и «осциллограф». Это 4 графика, задача которых – отображать изменения значений данных во времени.

Около каждого графика имеется соответствующий выпадающий список, который также автоматически обновляется в процессе работы приложения. В этом списке нужно выбрать требуемую связь для визуализации



Для фильтрации получаемых пакетов используется «белый список». Для этого на форме есть соответствующая таблица



Чтобы создать новое правило, нужно нажать на кнопку Add rule. Будет добавлена новая строка, которую можно редактировать, в том числе, копировать и вставлять выделенный текст. Для удаления строки, присутствует кнопка Remove selected.

Чтобы разрешить пакеты, например, на все принимающие узлы, нужно в столбце Destination у конкретной строки поставить символ \*.

Аналогично и для отправляющих устройств (Source).

Если установить и в Source, и в Destination \*, то будут разрешены все пакеты (ситуация по умолчанию)

**Вывод**

В ходе данной работы было создано приложение, реализующее алгоритм считывания и статистического анализа данных, поступающих по последовательному протоколу J1939 (CAN).

**Исходный код**

*"""  
This is free and unencumbered software released into the public domain.  
  
Anyone is free to copy, modify, publish, use, compile, sell, or  
distribute this software, either in source code form or as a compiled  
binary, for any purpose, commercial or non-commercial, and by any  
means.  
  
In jurisdictions that recognize copyright laws, the author or authors  
of this software dedicate any and all copyright interest in the  
software to the public domain. We make this dedication for the benefit  
of the public at large and to the detriment of our heirs and  
successors. We intend this dedication to be an overt act of  
relinquishment in perpetuity of all present and future rights to this  
software under copyright law.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF  
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR  
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,  
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR  
OTHER DEALINGS IN THE SOFTWARE.  
  
For more information, please refer to <https://unlicense.org>  
"""*import math  
import os  
import sys  
import threading  
import time  
  
import pyqtgraph as pg  
from PyQt5 import uic, QtWidgets, QtCore  
from PyQt5.QtWidgets import QApplication, QMainWindow, QTableWidgetItem  
  
  
def rotate2d(pos, rad):  
 *"""  
 Rotates point on angle  
 :param pos: point  
 :param rad: angle in radians  
 :return:  
 """* x, y = pos  
 s, c = math.sin(rad), math.cos(rad)  
 return x \* c - y \* s, y \* c + x \* s  
  
  
def ard\_map(value, in\_min, in\_max, out\_min, out\_max):  
 *"""  
 Arduino's map function  
 :return:  
 """* return out\_min + (out\_max - out\_min) \* ((value - in\_min) / (in\_max - in\_min))  
  
  
class Window(QMainWindow):  
 def \_\_init\_\_(self):  
 super(Window, self).\_\_init\_\_()  
 *# Load GUI file* uic.loadUi(**'LR3.ui'**, self)  
  
 *# System variables* self.dump\_file = None  
 self.reader\_running = False  
 self.packets = []  
 self.manual\_timer = QtCore.QTimer()  
 self.manual\_timer.timeout.connect(self.manual\_send)  
 self.oscilloscope\_timer = QtCore.QTimer()  
 self.oscilloscope\_timer.timeout.connect(self.oscilloscope)  
 self.oscilloscope\_timer.start(100)  
 self.plot\_timer = QtCore.QTimer()  
 self.plot\_timer.timeout.connect(self.update\_plot\_and\_combos)  
 self.plot\_timer.start(500)  
  
 *# Connect GUI controls* self.btn\_load\_data.clicked.connect(self.load\_data)  
 self.btn\_stop\_reading.clicked.connect(self.stop\_reading)  
 self.btn\_add\_row.clicked.connect(self.add\_row)  
 self.btn\_remove\_row.clicked.connect(self.remove\_row)  
 self.btn\_send\_start.clicked.connect(self.send\_start)  
 self.btn\_send\_stop.clicked.connect(self.send\_stop)  
  
 *# Initialize table* self.init\_tables()  
  
 *# Initialize pyQtGraph charts* self.init\_charts()  
  
 *# Show GUI* self.show()  
  
 def init\_tables(self):  
 *"""  
 Initializes table of packets and setup table (whitelist table)  
 :return:  
 """* self.points\_table.setColumnCount(5)  
 self.points\_table.verticalHeader().setVisible(False)  
 self.points\_table.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)  
 self.points\_table.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem(**'Packet'**))  
 self.points\_table.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem(**'Time'**))  
 self.points\_table.setHorizontalHeaderItem(2, QtWidgets.QTableWidgetItem(**'Src'**))  
 self.points\_table.setHorizontalHeaderItem(3, QtWidgets.QTableWidgetItem(**'Dst'**))  
 self.points\_table.setHorizontalHeaderItem(4, QtWidgets.QTableWidgetItem(**'Data'**))  
 header = self.points\_table.horizontalHeader()  
 header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(2, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(3, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(4, QtWidgets.QHeaderView.Stretch)  
  
 self.setup\_table.setColumnCount(2)  
 self.setup\_table.verticalHeader().setVisible(False)  
 self.setup\_table.setEditTriggers(QtWidgets.QAbstractItemView.DoubleClicked)  
 self.setup\_table.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem(**'Source'**))  
 self.setup\_table.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem(**'Destination'**))  
 header = self.setup\_table.horizontalHeader()  
 header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)  
 self.setup\_table.insertRow(0)  
 self.setup\_table.setItem(0, 0, QTableWidgetItem(**'\*'**))  
 self.setup\_table.setItem(0, 1, QTableWidgetItem(**'\*'**))  
  
 def init\_charts(self):  
 *"""  
 Initializes charts  
 :return:  
 """* self.graphWidget.setBackground((255, 255, 255))  
 *# self.gui.graphWidget.getAxis('left').setPen(QtGui.QColor('black'))  
 # self.gui.graphWidget.getAxis('left').setTextPen(QtGui.QColor('black'))  
 # self.gui.graphWidget.getAxis('bottom').setPen(QtGui.QColor('black'))  
 # self.gui.graphWidget.getAxis('bottom').setTextPen(QtGui.QColor('black'))* self.graphWidget.getPlotItem().hideAxis(**'top'**)  
 self.graphWidget.getPlotItem().hideAxis(**'left'**)  
 self.graphWidget.getPlotItem().hideAxis(**'right'**)  
 self.graphWidget.getPlotItem().hideAxis(**'bottom'**)  
 self.graphWidget.showGrid(x=False, y=False, alpha=1.0)  
  
 self.graphWidget\_2.setBackground((255, 255, 255))  
 self.graphWidget\_2.showGrid(x=True, y=True, alpha=1.0)  
 self.graphWidget\_3.setBackground((255, 255, 255))  
 self.graphWidget\_3.showGrid(x=True, y=True, alpha=1.0)  
 self.graphWidget\_4.setBackground((255, 255, 255))  
 self.graphWidget\_4.showGrid(x=True, y=True, alpha=1.0)  
 self.graphWidget\_5.setBackground((255, 255, 255))  
 self.graphWidget\_5.showGrid(x=True, y=True, alpha=1.0)  
  
 def add\_row(self):  
 *"""  
 Adds row to the setup table  
 :return:  
 """* row\_number = self.setup\_table.rowCount()  
 self.setup\_table.insertRow(row\_number)  
 self.setup\_table.setItem(row\_number, 0, QTableWidgetItem(**'00'**))  
 self.setup\_table.setItem(row\_number, 1, QTableWidgetItem(**'00'**))  
  
 *# noinspection PyBroadException* def remove\_row(self):  
 *"""  
 Removes row from the setup table  
 :return:  
 """* selected\_items = self.setup\_table.selectedItems()  
 for selected\_item in selected\_items:  
 try:  
 self.setup\_table.removeRow(self.setup\_table.row(selected\_item))  
 except:  
 pass  
  
 def send\_start(self):  
 *"""  
 Start timer for sending manual packets  
 :return:  
 """* self.btn\_send\_start.setEnabled(False)  
 self.btn\_send\_stop.setEnabled(True)  
 self.manual\_timer.start(self.spin\_send\_period.value())  
  
 def send\_stop(self):  
 *"""  
 Stops timer for sending manual packets  
 :return:  
 """* self.btn\_send\_start.setEnabled(True)  
 self.btn\_send\_stop.setEnabled(False)  
 self.manual\_timer.stop()  
  
 def load\_data(self):  
 *"""  
 Loads dump file  
 :return:  
 """* if not self.reader\_running:  
 if os.path.exists(self.data\_file.text()):  
 print(**'Loading data...'**)  
 self.dump\_file = open(self.data\_file.text(), **'rb'**)  
 self.reader\_running = True  
 thread = threading.Thread(target=self.dump\_reader)  
 thread.start()  
 else:  
 print(**'File'**, self.data\_file.text(), **'doesn**\'**t exist!'**)  
  
 def stop\_reading(self):  
 *"""  
 Stops reading data from dump file  
 :return:  
 """* self.reader\_running = False  
 self.dump\_file.close()  
  
 def manual\_send(self):  
 *"""  
 Sends sequence of packets defined on the form  
 :return:  
 """* packet\_time = 0  
 if len(self.packets) > 0:  
 packet\_time = self.packets[len(self.packets) - 1][0] + self.spin\_send\_period.value()  
 self.proceed\_packet(packet\_time, bytes.fromhex(self.line\_send\_src.text()),  
 bytes.fromhex(self.line\_send\_dst.text()), bytes.fromhex(self.line\_send\_data.text()))  
  
 def dump\_reader(self):  
 *"""  
 Reads dump from file  
 :return:  
 """  
 # Clear table and data arrays  
 # self.points\_table.setRowCount(0)* self.packets = []  
  
 *# Create temp buffers* bytes\_buffer = [b'\x00'] \* 19  
 bytes\_buffer\_position = 0  
 previous\_byte = b'\x00'  
 packets\_read = 0  
  
 *# Continue reading* while self.reader\_running:  
 incoming\_byte = self.dump\_file.read(1)  
 if incoming\_byte is None or len(incoming\_byte) == 0:  
 self.reader\_running = False  
 else:  
 bytes\_buffer[bytes\_buffer\_position] = incoming\_byte  
 if bytes\_buffer[bytes\_buffer\_position] == b'\xff' and previous\_byte == b'\xff':  
 bytes\_buffer\_position = 0  
  
 packet\_time = int.from\_bytes(b''.join([bytes\_buffer[1], bytes\_buffer[0]]),  
 byteorder=**'big'**, signed=False)  
 source = bytes\_buffer[6]  
 destination = bytes\_buffer[7]  
 data = bytes\_buffer[9]  
  
 self.proceed\_packet(packet\_time, source, destination, data)  
  
 packets\_len = len(self.packets)  
 if packets\_len > 1:  
 time\_to\_sleep = self.packets[packets\_len - 1][0] - self.packets[packets\_len - 2][0]  
 if time\_to\_sleep < 10:  
 time\_to\_sleep = 10  
 time.sleep(time\_to\_sleep / 1000)  
  
 packets\_read += 1  
 else:  
 previous\_byte = bytes\_buffer[bytes\_buffer\_position]  
 bytes\_buffer\_position += 1  
 if bytes\_buffer\_position >= 19:  
 bytes\_buffer\_position = 0  
  
 self.dump\_file.close()  
 print(**'File reading stopped. Read'**, packets\_read, **'packets'**)  
  
 def proceed\_packet(self, packet\_time, source, destination, data):  
 *"""  
 Handles packet and adds it to the self.points\_table and table according to the whitelist table  
 :param packet\_time: time mark of the packet  
 :param source: source address  
 :param destination: destination address  
 :param data: payload  
 :return:  
 """  
 # Create whitelists* allowed\_sources = []  
 allowed\_destinations = []  
 for i in range(self.setup\_table.rowCount()):  
 allowed\_sources.append(str(self.setup\_table.item(i, 0).text()).lower())  
 allowed\_destinations.append(str(self.setup\_table.item(i, 1).text()).lower())  
  
 *# Check is packet is allowed* if (str(source.hex()).lower() in allowed\_sources or **'\*'** in allowed\_sources) \  
 and (str(destination.hex()).lower() in allowed\_destinations or **'\*'** in allowed\_destinations):  
 *# Add packet to the list* self.packets.append([packet\_time, source, destination, data])  
  
 *# Add packet to the table* position = self.points\_table.rowCount()  
 self.points\_table.insertRow(position)  
 self.points\_table.setItem(position, 0, QTableWidgetItem(str(position)))  
 self.points\_table.setItem(position, 1, QTableWidgetItem(str(packet\_time)))  
 self.points\_table.setItem(position, 2, QTableWidgetItem(str(source.hex())))  
 self.points\_table.setItem(position, 3, QTableWidgetItem(str(destination.hex())))  
 self.points\_table.setItem(position, 4, QTableWidgetItem(str(data.hex())))  
  
 def oscilloscope(self):  
 *"""  
 Shows last 20 packets on the plots  
 :return:  
 """  
 # Count packets and Initialize arrays* packets\_len = len(self.packets)  
 points\_x = [[], [], [], []]  
 points\_y = [[], [], [], []]  
 if packets\_len > 0:  
 *# Clear plots* self.graphWidget\_2.clear()  
 self.graphWidget\_3.clear()  
 self.graphWidget\_4.clear()  
 self.graphWidget\_5.clear()  
  
 *# Start position* packets\_position = packets\_len - 1  
  
 *# Plot last 20 points* while packets\_position > 0 and (len(points\_y[0]) < 20  
 or len(points\_y[1]) < 20  
 or len(points\_y[2]) < 20  
 or len(points\_y[3]) < 20):  
  
 *# Get channel source and destination from combobox* ch1 = str(self.combo\_ch1.currentText()).split(**'->'**)  
 ch2 = str(self.combo\_ch2.currentText()).split(**'->'**)  
 ch3 = str(self.combo\_ch3.currentText()).split(**'->'**)  
 ch4 = str(self.combo\_ch4.currentText()).split(**'->'**)  
  
 if ch1[0] == self.packets[packets\_position][1].hex() \  
 and ch1[1] == self.packets[packets\_position][2].hex() and len(points\_y[0]) < 20:  
 points\_x[0].append(int(self.packets[packets\_position][0]))  
 points\_y[0].append(int(self.packets[packets\_position][3].hex(), 16))  
  
 if ch2[0] == self.packets[packets\_position][1].hex() \  
 and ch2[1] == self.packets[packets\_position][2].hex() and len(points\_y[1]) < 20:  
 points\_x[1].append(int(self.packets[packets\_position][0]))  
 points\_y[1].append(int(self.packets[packets\_position][3].hex(), 16))  
  
 if ch3[0] == self.packets[packets\_position][1].hex() \  
 and ch3[1] == self.packets[packets\_position][2].hex() and len(points\_y[2]) < 20:  
 points\_x[2].append(int(self.packets[packets\_position][0]))  
 points\_y[2].append(int(self.packets[packets\_position][3].hex(), 16))  
  
 if ch4[0] == self.packets[packets\_position][1].hex() \  
 and ch4[1] == self.packets[packets\_position][2].hex() and len(points\_y[3]) < 20:  
 points\_x[3].append(int(self.packets[packets\_position][0]))  
 points\_y[3].append(int(self.packets[packets\_position][3].hex(), 16))  
  
 packets\_position -= 1  
  
 *# Draw plots* self.graphWidget\_2.plot(points\_x[0], points\_y[0], pen=pg.mkPen((255, 127, 0)),  
 symbolBrush=(255, 127, 0), symbolSize=5)  
 self.graphWidget\_3.plot(points\_x[1], points\_y[1], pen=pg.mkPen((255, 127, 0)),  
 symbolBrush=(255, 127, 0), symbolSize=5)  
 self.graphWidget\_4.plot(points\_x[2], points\_y[2], pen=pg.mkPen((255, 127, 0)),  
 symbolBrush=(255, 127, 0), symbolSize=5)  
 self.graphWidget\_5.plot(points\_x[3], points\_y[3], pen=pg.mkPen((255, 127, 0)),  
 symbolBrush=(255, 127, 0), symbolSize=5)  
  
 def update\_plot\_and\_combos(self):  
 *"""  
 Updates nodes plot and combo boxes  
 :return:  
 """  
 # Create text lists of nodes and links* nodes\_text = []  
 links\_text = []  
 for i in range(self.points\_table.rowCount()):  
 src = self.points\_table.item(i, 2).text()  
 dst = self.points\_table.item(i, 3).text()  
 links\_text.append([src, dst])  
 if src not in nodes\_text:  
 nodes\_text.append(src)  
 if dst not in nodes\_text:  
 nodes\_text.append(dst)  
  
 *# Add links to comboBoxes* for link\_text in links\_text:  
 text\_row = link\_text[0] + **'->'** + link\_text[1]  
 if self.combo\_ch1.findText(text\_row) < 0:  
 self.combo\_ch1.addItem(text\_row)  
 self.combo\_ch2.addItem(text\_row)  
 self.combo\_ch3.addItem(text\_row)  
 self.combo\_ch4.addItem(text\_row)  
  
 *# Draw nodes with text and links* if len(links\_text) > 0:  
 self.graphWidget.clear()  
 nodes = []  
 for i in range(len(nodes\_text)):  
 rotated = rotate2d((1, 1), math.radians(ard\_map(i, 0, len(nodes\_text), 0, 360)))  
 nodes.append([rotated[0], rotated[1]])  
 label = pg.TextItem(text=nodes\_text[i], fill=(255, 245, 162), color=(0, 0, 0))  
 label.setPos(rotated[0], rotated[1])  
 self.graphWidget.addItem(label)  
  
 links = []  
 for i in range(len(nodes)):  
 for k in range(len(nodes)):  
 if k != i:  
 link = [nodes[i], nodes[k]]  
 if link not in links and link[::-1] not in links and \  
 ([nodes\_text[i], nodes\_text[k]] in links\_text  
 or [nodes\_text[k], nodes\_text[i]] in links\_text):  
 links.append(link)  
 for link in links:  
 self.graphWidget.plot([link[0][0], link[1][0]], [link[0][1], link[1][1]], pen=pg.mkPen((0, 255, 0)),  
 symbolBrush=(255, 0, 0), symbolSize=5)  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 app = QApplication(sys.argv)  
 app.setStyle(**'fusion'**)  
 win = Window()  
 sys.exit(app.exec\_())

*<?*xml version="1.0" encoding="UTF-8"*?>  
<!--  
This is free and unencumbered software released into the public domain.  
  
Anyone is free to copy, modify, publish, use, compile, sell, or  
distribute this software, either in source code form or as a compiled  
binary, for any purpose, commercial or non-commercial, and by any  
means.  
  
In jurisdictions that recognize copyright laws, the author or authors  
of this software dedicate any and all copyright interest in the  
software to the public domain. We make this dedication for the benefit  
of the public at large and to the detriment of our heirs and  
successors. We intend this dedication to be an overt act of  
relinquishment in perpetuity of all present and future rights to this  
software under copyright law.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF  
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR  
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,  
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR  
OTHER DEALINGS IN THE SOFTWARE.  
  
For more information, please refer to <https://unlicense.org>  
 -->*<ui version="4.0">  
 <class>MainWindow</class>  
 <widget class="QMainWindow" name="MainWindow">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>801</width>  
 <height>937</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <family>Calibri Light</family>  
 <pointsize>12</pointsize>  
 </font>  
 </property>  
 <property name="windowTitle">  
 <string>DoDCaPS LR3</string>  
 </property>  
 <widget class="QWidget" name="centralwidget">  
 <layout class="QGridLayout" name="gridLayout\_2">  
 <item row="1" column="0">  
 <widget class="Line" name="line">  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="0">  
 <widget class="Line" name="line\_2">  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 <item row="0" column="1">  
 <layout class="QVBoxLayout" name="verticalLayout\_4">  
 <item>  
 <widget class="QLabel" name="label\_26">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Graph</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="PlotWidget" name="graphWidget" native="true">  
 <property name="minimumSize">  
 <size>  
 <width>400</width>  
 <height>400</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>400</width>  
 <height>400</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="0" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_3">  
 <item>  
 <widget class="QLabel" name="label">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>22</pointsize>  
 </font>  
 </property>  
 <property name="text">  
 <string>DoDCaPS LR3</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_4">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>CAN bus data table</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QTableWidget" name="points\_table">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Minimum" vsizetype="Expanding">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="sortingEnabled">  
 <bool>true</bool>  
 </property>  
 <attribute name="horizontalHeaderShowSortIndicator" stdset="0">  
 <bool>true</bool>  
 </attribute>  
 <attribute name="verticalHeaderShowSortIndicator" stdset="0">  
 <bool>false</bool>  
 </attribute>  
 </widget>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout">  
 <item>  
 <widget class="QLabel" name="label\_19">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Dump file:</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="data\_file">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>LR3\_data/CANDump02.dmp</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_5">  
 <item>  
 <widget class="QPushButton" name="btn\_load\_data">  
 <property name="text">  
 <string>Load data from file</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QPushButton" name="btn\_stop\_reading">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Close file</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="1" column="1" rowspan="4">  
 <layout class="QGridLayout" name="gridLayout\_3">  
 <item row="1" column="0">  
 <widget class="QComboBox" name="combo\_ch1">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="0">  
 <widget class="QComboBox" name="combo\_ch2">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="0">  
 <widget class="QComboBox" name="combo\_ch3">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 </widget>  
 </item>  
 <item row="4" column="0">  
 <widget class="QComboBox" name="combo\_ch4">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="1">  
 <widget class="PlotWidget" name="graphWidget\_2" native="true">  
 <property name="minimumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 <item row="4" column="1">  
 <widget class="PlotWidget" name="graphWidget\_5" native="true">  
 <property name="minimumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 <item row="0" column="0">  
 <widget class="QLabel" name="label\_22">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>16777215</width>  
 <height>16777215</height>  
 </size>  
 </property>  
 <property name="text">  
 <string>Oscilloscope</string>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="1">  
 <widget class="PlotWidget" name="graphWidget\_3" native="true">  
 <property name="minimumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="1">  
 <widget class="PlotWidget" name="graphWidget\_4" native="true">  
 <property name="minimumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>400</width>  
 <height>100</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="2" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_2">  
 <item>  
 <widget class="QLabel" name="label\_25">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Serial packet sender</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <layout class="QGridLayout" name="gridLayout">  
 <item row="1" column="1">  
 <widget class="QLineEdit" name="line\_send\_src">  
 <property name="text">  
 <string>00</string>  
 </property>  
 </widget>  
 </item>  
 <item row="4" column="2">  
 <widget class="QLabel" name="label\_17">  
 <property name="text">  
 <string>milliseconds</string>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="2">  
 <widget class="QLabel" name="label\_21">  
 <property name="text">  
 <string>hex (00-FF)</string>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="2">  
 <widget class="QLabel" name="label\_20">  
 <property name="text">  
 <string>hex (00-FF)</string>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="2">  
 <widget class="QLabel" name="label\_18">  
 <property name="text">  
 <string>hex (00-FF)</string>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="1">  
 <widget class="QLineEdit" name="line\_send\_data">  
 <property name="text">  
 <string>00</string>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="1">  
 <widget class="QLineEdit" name="line\_send\_dst">  
 <property name="text">  
 <string>01</string>  
 </property>  
 </widget>  
 </item>  
 <item row="4" column="0">  
 <widget class="QLabel" name="label\_15">  
 <property name="text">  
 <string>Period:</string>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="0">  
 <widget class="QLabel" name="label\_2">  
 <property name="text">  
 <string>Source:</string>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="0">  
 <widget class="QLabel" name="label\_16">  
 <property name="text">  
 <string>Data:</string>  
 </property>  
 </widget>  
 </item>  
 <item row="2" column="0">  
 <widget class="QLabel" name="label\_14">  
 <property name="text">  
 <string>Destination:</string>  
 </property>  
 </widget>  
 </item>  
 <item row="5" column="0">  
 <widget class="QPushButton" name="btn\_send\_start">  
 <property name="text">  
 <string>Start</string>  
 </property>  
 </widget>  
 </item>  
 <item row="4" column="1">  
 <widget class="QSpinBox" name="spin\_send\_period">  
 <property name="minimum">  
 <number>10</number>  
 </property>  
 <property name="maximum">  
 <number>99999</number>  
 </property>  
 <property name="singleStep">  
 <number>10</number>  
 </property>  
 <property name="value">  
 <number>20</number>  
 </property>  
 </widget>  
 </item>  
 <item row="5" column="2">  
 <widget class="QPushButton" name="btn\_send\_stop">  
 <property name="enabled">  
 <bool>false</bool>  
 </property>  
 <property name="text">  
 <string>Stop</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="4" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout">  
 <item>  
 <widget class="QLabel" name="label\_23">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Whitelist</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_24">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>HEX values (00-FF) or use \* for all addresses</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QTableWidget" name="setup\_table">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Minimum" vsizetype="Expanding">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="sortingEnabled">  
 <bool>true</bool>  
 </property>  
 <attribute name="horizontalHeaderShowSortIndicator" stdset="0">  
 <bool>true</bool>  
 </attribute>  
 <attribute name="verticalHeaderShowSortIndicator" stdset="0">  
 <bool>false</bool>  
 </attribute>  
 </widget>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_4">  
 <item>  
 <widget class="QPushButton" name="btn\_add\_row">  
 <property name="text">  
 <string>Add rule</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QPushButton" name="btn\_remove\_row">  
 <property name="text">  
 <string>Remove selected</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </widget>  
 <widget class="QMenuBar" name="menubar">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>801</width>  
 <height>21</height>  
 </rect>  
 </property>  
 </widget>  
 <widget class="QStatusBar" name="statusbar"/>  
 </widget>  
 <customwidgets>  
 <customwidget>  
 <class>PlotWidget</class>  
 <extends>QWidget</extends>  
 <header>pyqtgraph</header>  
 <container>1</container>  
 </customwidget>  
 </customwidgets>  
 <resources/>  
 <connections/>  
</ui>