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 №4

«Разработать многоканальную систему накопления и отображения данных измерений»

Работа выполнена (ФИО):

Ф И О

Научный руководитель

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

Ф И О

**Цель работы**

Разработать систему накопления и визуализации данных в виде графиков**Задачи**

* Подготовить приложение, получающее данные с серийного порта и по UDP, по 4 независимым каналам;
* Разработать функцию сохранения данных в файле с временными отметками и настраиваемой фильтрацией (по времени, по реакции на событие – выход за диапазон или быстрое изменение);
* Разработать функцию визуализации данных по заданным параметрам (диапазон времени, масштаб, частота выборки) по накопленным данным;
* Разработать функцию отображения текущих данных с заданными настройками визуализации (масштаб по времени, масштаб по амплитуде);
* Разработать функцию фильтрации шумов с использованием статистических методов (выборочное среднее, скользящее среднее (два варианта), медианную фильтрацию и т.п.);
* Реализовать возможность сохранения изображения графиков. **Ход работы**

В качестве источника данных могут выступать:

1. Последовательный порт

Данные будут считываться по 11 байт из последовательного порта, обрабатываться и передаваться в общую функцию обработки

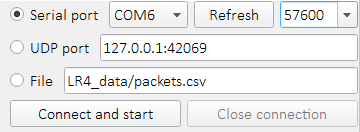
1. UDP – порт

Данные будут считываться по 11 байт из UDP, обрабатываться и передаваться в общую функцию обработки

1. Записанный ранее файл

Данные будут считываться напрямую из CSV файла, и, без парсинга

передаваться в общую функцию обработки



Для проверки функции передачи через Serial порт был написан код на языке Arduino (C++) для микроконтроллера Atmega2560 / Atmega328. (см. в приложении)

Структура пакета данных (для последовательного и UDP порта) выглядит следующим образом

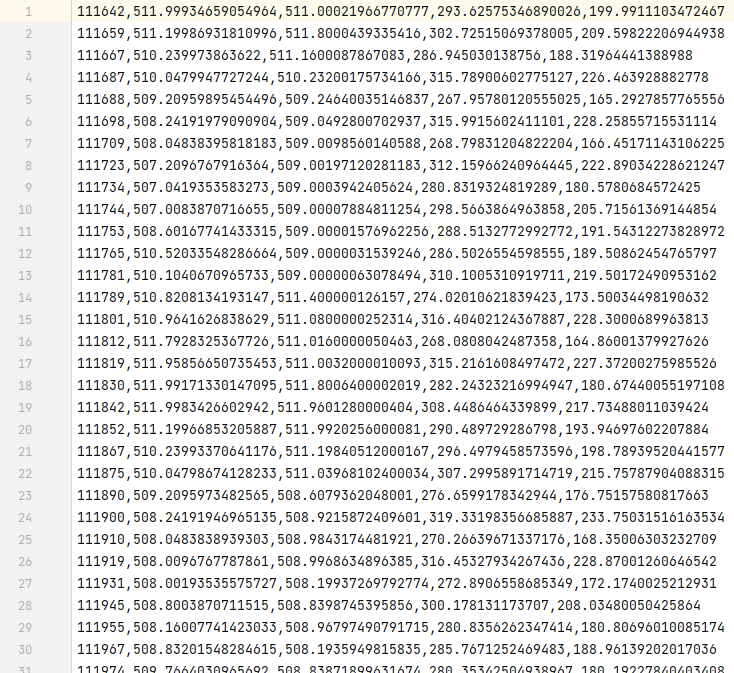
|  |  |  |
| --- | --- | --- |
| № байта | Описание | Пример значения |
| 0 | Значение первого канала | 01 |
| 1 | FE |
| 2 | Значение второго канала | 01 |
| 3 | FE |
| 4 | Значение третьего канала | 01 |
| 5 | F7 |
| 6 | Значение четвёртого канала | 01 |
| 7 | F1 |
| 8 | XOR чек-сумма | 06 |
| 9 | Символ окончания пакета | FF |
| 10 | Символ окончания пакета | FF |

Подразумевается, что значения всех каналов положительные и лежат в диапазоне 0-65534. Для хранения данных используется буфер длинной в 11 байт. Как только обнаруживается пара байтов FF-FF, считается что пакет окончен. После этого вычисляется XOR чек-сумма всех 8-и байтов и сравнивается со значением в пакете. Если чек-суммы совпадают, считается что пакет принят успешно.

Далее из пакета вычисляются значения всех четырёх каналов и подаются в функцию parse\_data, которая вычисляет штамп времени, применяет фильтрацию, отображает значения на графике, а также, при необходимости сохраняет данные в .csv файл.

В случае использования, записанного ранее .csv файла в качестве источника, данные читаются напрямую из файла, без каких-либо проверок.

Структура .csv файла выглядит следующим образом:



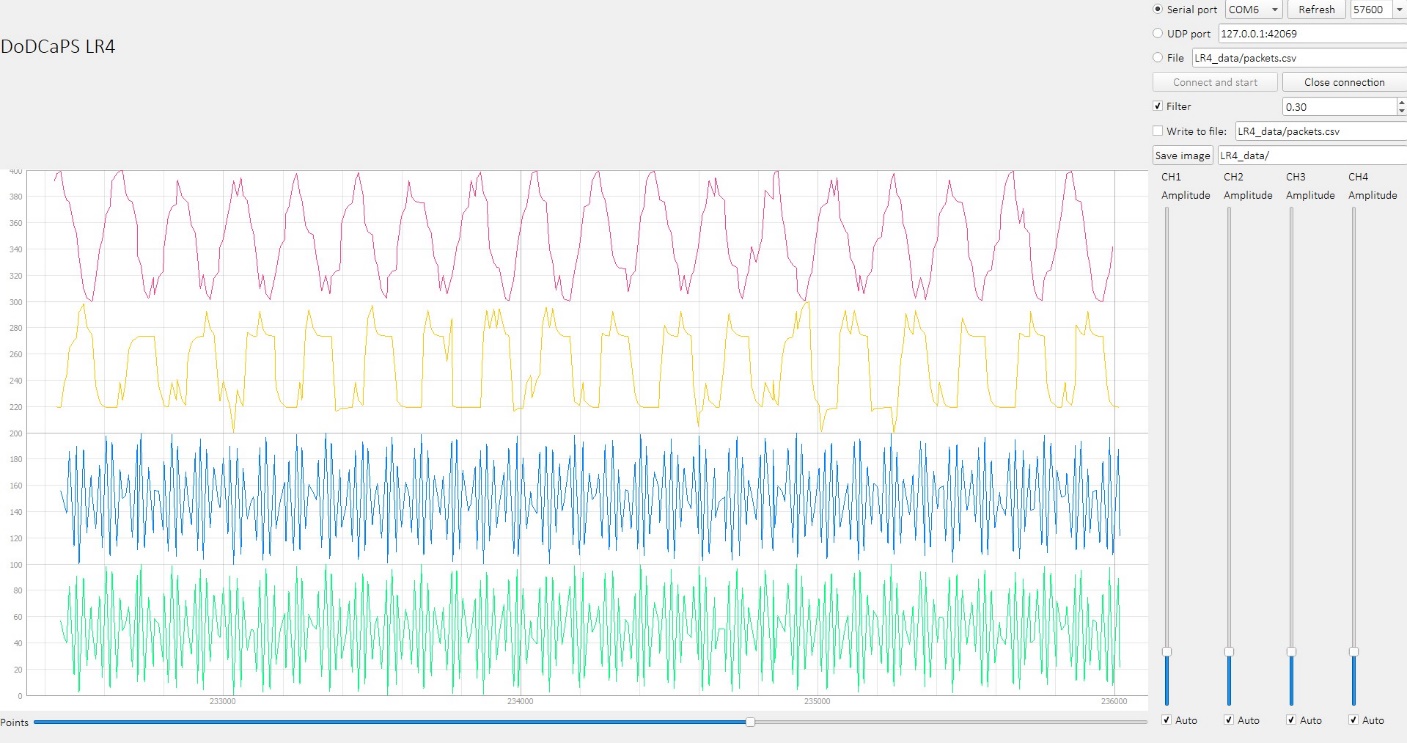
В начале записывается время в миллисекундах, затем 4 значения каналов. Разделитель: запятая (,).

Перед отображением / сохранением в файл к данным можно применить фильтр. Для этого на форме есть соответствующие элементы управления



Фильтр производит сглаживание принятых данных.

Наконец, для отображения данных используется элемент PyQtGraph, на котором отрисовывается 4 графика пропорционально по высоте



Для настройки отображения имеются 5 ползунков: 4 для ручной регулировки амплитуды отображаемого сигнала (имеется чек-бокс Auto, позволяющий выполнять автоматическую настройку) и 1 для регулировки разрешения по оси X (количество точек во времени).

Имеется возможно сохранить снимок графика. Для этого на форме необходимо указать папку, куда будет сохранено изображение и нажать кнопку Save image.

**Вывод**

В ходе данной работы было создано приложение, реализующее систему накопления и визуализации многоканальных данных в виде графиков с возможностью фильтрации, записи данных в файл и экспорта изображений графиков

**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 csv  
import glob  
import socket  
import sys  
import threading  
import time  
  
import numpy as np  
import pyqtgraph as pg  
import pyqtgraph.exporters  
import serial  
from PyQt5 import uic, QtCore  
from PyQt5.QtWidgets import QApplication, QMainWindow  
  
  
class Window(QMainWindow):  
 def \_\_init\_\_(self):  
 super(Window, self).\_\_init\_\_()  
 *# Load GUI file* uic.loadUi(**'LR4.ui'**, self)  
  
 *# System variables* self.serial\_port = None  
 self.udp\_ip = None  
 self.udp\_port = None  
 self.udp\_socket = None  
 self.file\_source = None  
 self.file\_destination = None  
 self.reader\_working = False  
 self.first\_packet\_time = -1  
 self.points = [[0] \* 500, [0] \* 500, [0] \* 500, [0] \* 500, [0] \* 500]  
 self.source\_timestamp = 0  
 self.channel\_1 = 0  
 self.channel\_2 = 0  
 self.channel\_3 = 0  
 self.channel\_4 = 0  
 self.csv\_writer = None  
  
 self.plot\_timer = QtCore.QTimer()  
 self.plot\_timer.timeout.connect(self.update\_plots)  
 self.plot\_timer.start(30)  
  
 *# Connect GUI controls* self.btn\_start.clicked.connect(self.oscilloscope\_start)  
 self.btn\_stop.clicked.connect(self.oscilloscope\_stop)  
 self.btn\_serial\_refresh.clicked.connect(self.refresh\_serial\_ports)  
 self.btn\_save\_image.clicked.connect(self.save\_image)  
  
 *# Initialize pyQtGraph charts* self.init\_charts()  
  
 *# Add serial speeds* self.init\_serial\_bauds()  
  
 *# Show GUI* self.show()  
  
 *# Refresh serial ports* self.refresh\_serial\_ports()  
  
 def init\_serial\_bauds(self):  
 *"""  
 Adds serial speeds (baud rates) to the combo box  
 :return:  
 """* self.combo\_serial\_speed.clear()  
 self.combo\_serial\_speed.addItems([**'9600'**, **'110'**, **'300'**, **'600'**, **'1200'**, **'2400'**, **'4800'**,  
 **'14400'**, **'19200'**, **'38400'**, **'57600'**, **'115200'**, **'128000'**])  
  
 def refresh\_serial\_ports(self):  
 *"""  
 Gets list of the serial ports and adds them to the combo box  
 :return:  
 """* self.combo\_serial.clear()  
 if sys.platform.startswith(**'win'**):  
 ports = [**'COM%s'** % (i + 1) for i in range(256)]  
 elif sys.platform.startswith(**'linux'**) or sys.platform.startswith(**'cygwin'**):  
 ports = glob.glob(**'/dev/tty[A-Za-z]\*'**)  
 elif sys.platform.startswith(**'darwin'**):  
 ports = glob.glob(**'/dev/tty.\*'**)  
 else:  
 raise EnvironmentError(**'Unsupported platform'**)  
 for port in ports:  
 try:  
 s = serial.Serial(port)  
 s.close()  
 self.combo\_serial.addItem(port)  
 except (OSError, serial.SerialException):  
 pass  
  
 def oscilloscope\_start(self):  
 *"""  
 Opens ports / files and starts background reading process  
 :return:  
 """* if self.radio\_serial.isChecked() and len(self.combo\_serial.currentText()) > 0:  
 *# Open serial port* print(**'Using serial port'**, self.combo\_serial.currentText())  
 self.serial\_port = serial.Serial(self.combo\_serial.currentText(),  
 int(self.combo\_serial\_speed.currentText()))  
 self.serial\_port.close()  
 self.serial\_port.open()  
 print(**'Port opened?'**, self.serial\_port.isOpen())  
 self.udp\_ip = None  
 self.udp\_port = None  
 self.file\_source = None  
 elif self.radio\_udp.isChecked() and len(self.line\_udp.text()) > 0:  
 *# Define UDP IP and Port* ip\_port = str(self.line\_udp.text()).split(**':'**)  
 print(**'Using UDP port'**, self.line\_udp.text())  
 self.udp\_ip = ip\_port[0]  
 self.udp\_port = int(ip\_port[1])  
 self.udp\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 self.udp\_socket.bind((self.udp\_ip, self.udp\_port))  
 self.serial\_port = None  
 self.file\_source = None  
 elif self.radio\_file.isChecked() and len(self.data\_file.text()) > 0:  
 *# Open file* print(**'Using file'**, self.data\_file.text())  
 self.file\_source = open(self.data\_file.text(), newline=**''**)  
 self.serial\_port = None  
 self.udp\_ip = None  
 self.udp\_port = None  
 else:  
 *# Nothing to open* self.serial\_port = None  
 self.udp\_ip = None  
 self.udp\_port = None  
 self.file\_source = None  
 self.reader\_working = False  
 print(**'Nothing to open'**)  
  
 if self.udp\_ip is not None or self.serial\_port is not None or self.file\_source is not None:  
 *# If Serial or UDP port* self.btn\_start.setEnabled(False)  
 self.btn\_stop.setEnabled(True)  
 self.reader\_working = True  
 self.source\_timestamp = 0  
 thread = threading.Thread(target=self.async\_data\_reader)  
 thread.start()  
  
 *# noinspection PyBroadException* def oscilloscope\_stop(self):  
 *"""  
 Stops reading data and closes files and ports  
 :return:  
 """* self.btn\_start.setEnabled(True)  
 self.btn\_stop.setEnabled(False)  
 self.reader\_working = False  
 try:  
 if self.file\_destination is not None:  
 self.file\_destination.close()  
 if self.file\_source is not None:  
 self.file\_source.close()  
 if self.udp\_socket is not None:  
 self.udp\_socket.close()  
 if self.serial\_port is not None:  
 self.serial\_port.close()  
 except:  
 pass  
  
 def async\_data\_reader(self):  
 *"""  
 Reads data from the serial port, udp port or file  
 :return:  
 """* data\_buffer = [0] \* 11  
 data\_buffer\_position = 0  
 data\_previous = 0  
  
 while self.reader\_working:  
 if self.file\_source is not None:  
 *# Read from CSV file* csv\_reader = csv.reader(self.file\_source, delimiter=**','**)  
 for row in csv\_reader:  
 if self.source\_timestamp > 0:  
 time.sleep((int(row[0]) - self.source\_timestamp) / 1000)  
 self.source\_timestamp = int(row[0])  
 self.parse\_data(int(float(row[1])), int(float(row[2])), int(float(row[3])), int(float(row[4])))  
 self.reader\_working = False  
 print(**'End of CSV file'**)  
 self.oscilloscope\_stop()  
 break  
 else:  
 *# Read from other sources* incoming\_data = None  
 if self.serial\_port is not None and self.serial\_port.isOpen():  
 incoming\_data = self.serial\_port.read()  
 elif self.udp\_ip is not None:  
 incoming\_data, address = self.udp\_socket.recvfrom(1024)  
 self.udp\_socket.sendto(incoming\_data, address)  
  
 if incoming\_data is not None and len(incoming\_data) > 0:  
 for data\_byte in incoming\_data:  
 data\_buffer[data\_buffer\_position] = data\_byte  
 if data\_buffer[data\_buffer\_position] == 255 and data\_previous == 255:  
 data\_buffer\_position = 0  
  
 check\_sum = 0  
 for i in range(0, 8):  
 check\_sum ^= int(data\_buffer[i] & 0xFF)  
  
 if check\_sum == data\_buffer[8]:  
 *# Parse packet* channel\_1 = (int(data\_buffer[0] & 0xFF) << 8) | int(data\_buffer[1] & 0xFF)  
 channel\_2 = (int(data\_buffer[2] & 0xFF) << 8) | int(data\_buffer[3] & 0xFF)  
 channel\_3 = (int(data\_buffer[4] & 0xFF) << 8) | int(data\_buffer[5] & 0xFF)  
 channel\_4 = (int(data\_buffer[6] & 0xFF) << 8) | int(data\_buffer[7] & 0xFF)  
 self.parse\_data(channel\_1, channel\_2, channel\_3, channel\_4)  
 else:  
 print(**'Wrong checksum'**)  
 else:  
 data\_previous = data\_buffer[data\_buffer\_position]  
 data\_buffer\_position += 1  
 if data\_buffer\_position >= 11:  
 data\_buffer\_position = 0  
  
 print(**'Reader closed'**)  
 self.oscilloscope\_stop()  
  
 def parse\_data(self, channel\_1, channel\_2, channel\_3, channel\_4):  
 *"""  
 Filters channels, plots them on the chart and writes to the file  
 :return:  
 """  
 # Calculate timestamp* if self.first\_packet\_time < 0:  
 self.first\_packet\_time = time.time()  
 packet\_time = int((time.time() - self.first\_packet\_time) \* 1000)  
  
 *# Filter data* if self.checkbox\_filter.isChecked():  
 filter\_factor = self.spin\_filter.value()  
 self.channel\_1 = self.channel\_1 \* filter\_factor + channel\_1 \* (1.0 - filter\_factor)  
 self.channel\_2 = self.channel\_2 \* filter\_factor + channel\_2 \* (1.0 - filter\_factor)  
 self.channel\_3 = self.channel\_3 \* filter\_factor + channel\_3 \* (1.0 - filter\_factor)  
 self.channel\_4 = self.channel\_4 \* filter\_factor + channel\_4 \* (1.0 - filter\_factor)  
 else:  
 self.channel\_1 = channel\_1  
 self.channel\_2 = channel\_2  
 self.channel\_3 = channel\_3  
 self.channel\_4 = channel\_4  
  
 *# Write to file if enabled* if self.checkbox\_write\_file.isChecked() and self.file\_destination is not None:  
 *# Write to file* self.csv\_writer.writerow([packet\_time, self.channel\_1, self.channel\_2,  
 self.channel\_3, self.channel\_4])  
 elif self.checkbox\_write\_file.isChecked() and self.file\_destination is None:  
 *# Create file* self.file\_destination = open(self.data\_file\_out.text(), **'w+'**, newline=**''**)  
 self.csv\_writer = csv.writer(self.file\_destination, delimiter=**','**)  
 elif not self.checkbox\_write\_file.isChecked() and self.file\_destination is not None:  
 *# Close file* self.file\_destination.close()  
 self.file\_destination = None  
 self.csv\_writer = None  
  
 *# Display data on plot* self.points[0] = self.points[0][1:]  
 self.points[0].append(packet\_time)  
 self.points[1] = self.points[1][1:]  
 self.points[1].append(self.channel\_1)  
 self.points[2] = self.points[2][1:]  
 self.points[2].append(self.channel\_2)  
 self.points[3] = self.points[3][1:]  
 self.points[3].append(self.channel\_3)  
 self.points[4] = self.points[4][1:]  
 self.points[4].append(self.channel\_4)  
  
 def save\_image(self):  
 *"""  
 Saves screenshot from the pyQtGraph  
 :return:  
 """* exporter = pg.exporters.ImageExporter(self.graphWidget.plotItem)  
 exporter.export(self.line\_image\_dir.text() + **'image.png'**)  
  
 def init\_charts(self):  
 *"""  
 Initializes charts  
 :return:  
 """* self.graphWidget.setBackground((255, 255, 255))  
 self.graphWidget.showGrid(x=True, y=True, alpha=0.7)  
 self.graphWidget.setYRange(0, 400, padding=0)  
  
 def update\_plots(self):  
 *"""  
 Updates channel plots  
 :return:  
 """* self.graphWidget.clear()  
  
 *# Channel 1* amplitudes = np.array(self.points[1][-self.slider\_points.value():])  
 if self.checkbox\_ch1\_auto.isChecked():  
 if np.max(amplitudes) != 0:  
 amplitudes -= np.min(amplitudes)  
 amplitudes = amplitudes / np.max(amplitudes) \* 100  
 else:  
 amplitudes = amplitudes / 1000  
 amplitudes \*= self.slider\_ampl\_1.value()  
 amplitudes = amplitudes - ((np.max(amplitudes) + np.min(amplitudes)) / 2) + 350  
 self.graphWidget.plot(self.points[0][-self.slider\_points.value():], amplitudes, pen=pg.mkPen((255, 63, 127)),  
 symbolBrush=None, symbolSize=0)  
  
 *# Channel 2* amplitudes = np.array(self.points[2][-self.slider\_points.value():])  
 if self.checkbox\_ch2\_auto.isChecked():  
 if np.max(amplitudes) != 0:  
 amplitudes -= np.min(amplitudes)  
 amplitudes = amplitudes / np.max(amplitudes) \* 100  
 else:  
 amplitudes = amplitudes / 1000  
 amplitudes \*= self.slider\_ampl\_2.value()  
 amplitudes = amplitudes - ((np.max(amplitudes) + np.min(amplitudes)) / 2) + 250  
 self.graphWidget.plot(self.points[0][-self.slider\_points.value():], amplitudes, pen=pg.mkPen((255, 200, 0)),  
 symbolBrush=None, symbolSize=0)  
  
 *# Channel 3* amplitudes = np.array(self.points[3][-self.slider\_points.value():])  
 if self.checkbox\_ch3\_auto.isChecked():  
 if np.max(amplitudes) != 0:  
 amplitudes -= np.min(amplitudes)  
 amplitudes = amplitudes / np.max(amplitudes) \* 100  
 else:  
 amplitudes = amplitudes / 1000  
 amplitudes \*= self.slider\_ampl\_3.value()  
 amplitudes = amplitudes - ((np.max(amplitudes) + np.min(amplitudes)) / 2) + 150  
 self.graphWidget.plot(self.points[0][-self.slider\_points.value():], amplitudes, pen=pg.mkPen((0, 127, 255)),  
 symbolBrush=None, symbolSize=0)  
  
 *# Channel 4* amplitudes = np.array(self.points[4][-self.slider\_points.value():])  
 if self.checkbox\_ch4\_auto.isChecked():  
 if np.max(amplitudes) != 0:  
 amplitudes -= np.min(amplitudes)  
 amplitudes = amplitudes / np.max(amplitudes) \* 100  
 else:  
 amplitudes = amplitudes / 1000  
 amplitudes \*= self.slider\_ampl\_4.value()  
 amplitudes = amplitudes - ((np.max(amplitudes) + np.min(amplitudes)) / 2) + 50  
 self.graphWidget.plot(self.points[0][-self.slider\_points.value():], amplitudes, pen=pg.mkPen((0, 255, 127)),  
 symbolBrush=None, symbolSize=0)  
  
  
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>766</width>  
 <height>730</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <family>Calibri Light</family>  
 <pointsize>12</pointsize>  
 </font>  
 </property>  
 <property name="windowTitle">  
 <string>DoDCaPS LR4</string>  
 </property>  
 <property name="layoutDirection">  
 <enum>Qt::LeftToRight</enum>  
 </property>  
 <widget class="QWidget" name="centralwidget">  
 <layout class="QGridLayout" name="gridLayout">  
 <item row="1" column="1">  
 <layout class="QHBoxLayout" name="horizontalLayout\_11">  
 <item>  
 <widget class="QCheckBox" name="checkbox\_filter">  
 <property name="text">  
 <string>Filter</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QDoubleSpinBox" name="spin\_filter">  
 <property name="layoutDirection">  
 <enum>Qt::LeftToRight</enum>  
 </property>  
 <property name="maximum">  
 <double>1.000000000000000</double>  
 </property>  
 <property name="singleStep">  
 <double>0.050000000000000</double>  
 </property>  
 <property name="value">  
 <double>0.300000000000000</double>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="4" column="1">  
 <layout class="QVBoxLayout" name="verticalLayout\_11">  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_7">  
 <item>  
 <layout class="QVBoxLayout" name="verticalLayout\_6">  
 <item>  
 <widget class="QLabel" name="label\_5">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>CH1</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_9">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Amplitude</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_ampl\_1">  
 <property name="layoutDirection">  
 <enum>Qt::LeftToRight</enum>  
 </property>  
 <property name="maximum">  
 <number>1000</number>  
 </property>  
 <property name="value">  
 <number>100</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Vertical</enum>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QCheckBox" name="checkbox\_ch1\_auto">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Auto</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QVBoxLayout" name="verticalLayout\_7">  
 <item>  
 <widget class="QLabel" name="label\_6">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>CH2</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_10">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Amplitude</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_ampl\_2">  
 <property name="maximum">  
 <number>1000</number>  
 </property>  
 <property name="value">  
 <number>100</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Vertical</enum>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QCheckBox" name="checkbox\_ch2\_auto">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Auto</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QVBoxLayout" name="verticalLayout\_8">  
 <item>  
 <widget class="QLabel" name="label\_7">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>CH3</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_11">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Amplitude</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_ampl\_3">  
 <property name="maximum">  
 <number>1000</number>  
 </property>  
 <property name="value">  
 <number>100</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Vertical</enum>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QCheckBox" name="checkbox\_ch3\_auto">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Auto</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QVBoxLayout" name="verticalLayout\_9">  
 <item>  
 <widget class="QLabel" name="label\_8">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>CH4</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLabel" name="label\_12">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Amplitude</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_ampl\_4">  
 <property name="maximum">  
 <number>1000</number>  
 </property>  
 <property name="value">  
 <number>100</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Vertical</enum>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QCheckBox" name="checkbox\_ch4\_auto">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Auto</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="2" column="1">  
 <layout class="QHBoxLayout" name="horizontalLayout\_10">  
 <item>  
 <widget class="QCheckBox" name="checkbox\_write\_file">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Write to file:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="data\_file\_out">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>LR4\_data/packets.csv</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="4" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_10">  
 <item>  
 <widget class="PlotWidget" name="graphWidget" native="true">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Expanding" vsizetype="Expanding">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>400</width>  
 <height>400</height>  
 </size>  
 </property>  
 <property name="maximumSize">  
 <size>  
 <width>16777215</width>  
 <height>16777215</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_9">  
 <item>  
 <widget class="QLabel" name="label\_3">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Points</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_points">  
 <property name="minimum">  
 <number>5</number>  
 </property>  
 <property name="maximum">  
 <number>500</number>  
 </property>  
 <property name="value">  
 <number>50</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="0" column="1">  
 <layout class="QVBoxLayout" name="verticalLayout\_5">  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_2">  
 <item>  
 <widget class="QRadioButton" name="radio\_serial">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Serial port</string>  
 </property>  
 <property name="checked">  
 <bool>true</bool>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QComboBox" name="combo\_serial">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QPushButton" name="btn\_serial\_refresh">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Refresh</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QComboBox" name="combo\_serial\_speed">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="editable">  
 <bool>true</bool>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_3">  
 <item>  
 <widget class="QRadioButton" name="radio\_udp">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>UDP port</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_udp">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>127.0.0.1:42069</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout">  
 <item>  
 <widget class="QRadioButton" name="radio\_file">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>File</string>  
 </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>LR4\_data/packets.csv</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_6">  
 <item>  
 <widget class="QPushButton" name="btn\_start">  
 <property name="text">  
 <string>Connect and start</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QPushButton" name="btn\_stop">  
 <property name="enabled">  
 <bool>false</bool>  
 </property>  
 <property name="text">  
 <string>Close connection</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="0" column="0">  
 <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="layoutDirection">  
 <enum>Qt::LeftToRight</enum>  
 </property>  
 <property name="text">  
 <string>DoDCaPS LR4</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item row="3" column="1">  
 <layout class="QHBoxLayout" name="horizontalLayout\_8">  
 <item>  
 <widget class="QPushButton" name="btn\_save\_image">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Save image</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="line\_image\_dir">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>LR4\_data/</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </widget>  
 <widget class="QMenuBar" name="menubar">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>766</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>

/\*

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>

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Serial port setup \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Serial port speed

const uint16\_t PORT\_BAUD\_RATE PROGMEM = 57600;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Pins setup \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CH1 analog pin

const uint8\_t CHANNEL\_1\_PIN PROGMEM = A0;

// CH2 analog pin

const uint8\_t CHANNEL\_2\_PIN PROGMEM = A1;

// CH3 analog pin

const uint8\_t CHANNEL\_3\_PIN PROGMEM = A2;

// CH4 analog pin

const uint8\_t CHANNEL\_4\_PIN PROGMEM = A3;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Time settings \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// After this time in milliseconds the buffer will be pushed to the serial port

// Set to 0 to send packets immediately

const uint16\_t SERIAL\_PERIOD PROGMEM = 10;

// System variables

uint8\_t buffer[11];

uint8\_t i;

uint16\_t channel\_1, channel\_2, channel\_3, channel\_4;

uint64\_t serial\_timer;

void setup()

{

// Init serial port

Serial.begin(PORT\_BAUD\_RATE);

// Pre-define buffer ending

buffer[9] = 255;

buffer[10] = 255;

}

void loop()

{

// Read ADC values

read\_channels();

// Fill buffer

fill\_buffer();

if (millis() - serial\_timer > SERIAL\_PERIOD) {

// Push buffer to the serial port

Serial.write(buffer, 11);

// Reset timer

serial\_timer = millis();

}

}

/// <summary>

/// Fills buffer with 4 ADC values

/// </summary>

void fill\_buffer(void) {

// Fill with channel values

buffer[0] = channel\_1 >> 8;

buffer[1] = channel\_1;

buffer[2] = channel\_2 >> 8;

buffer[3] = channel\_2;

buffer[4] = channel\_3 >> 8;

buffer[5] = channel\_3;

buffer[6] = channel\_4 >> 8;

buffer[7] = channel\_4;

// Calculate checksum

buffer[8] = 0;

for (i = 0; i <= 7; i++)

buffer[8] ^= buffer[i];

}

/// <summary>

/// Reads values from analog inputs

/// </summary>

void read\_channels(void) {

channel\_1 = analogRead(CHANNEL\_1\_PIN);

channel\_2 = analogRead(CHANNEL\_2\_PIN);

channel\_3 = analogRead(CHANNEL\_3\_PIN);

channel\_4 = analogRead(CHANNEL\_4\_PIN);

}