

**Московский государственный технический
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков
программирования»

Отчет по лабораторной работе №6
“Рефракторинг и оптимизация кода.
Создание мини-приложения”

Выполнил: Сорокин Михаил
Группа ИУ5-33Б
Вариант 15 В

Проверил: Гапанюк Юрий
Евгеньевич
преподаватель каф. ИУ5

Подпись и дата: 19.12.2024



Подпись и дата:

Москва, 2024 г.

Введение

Нашёл старый исходник, который использовался для визуализации данных, поступающих с микроконтроллеров Arduino. В рассматриваемом файле обрабатывались данные с одного микроконтроллера, подключаемого к компьютеру по COM-порту. При помощи библиотеки `pyserial` была организована работа с последовательными портами, `matplotlib` использовалась для визуализации получаемых данных. Данные представляли из себя результат приёма радиопередачи от другого устройства – число принятых пакетов данных (из тысячи отправленных за фрейм времени).

Таким образом, программа состояла из следующих этапов:

1. Инициализация рабочей области `matplotlib` для построения графика;
2. Выбор необходимого COM-порта и количества бодов;
3. Перегрузка найденного микроконтроллера для очистки строки информации;
4. Получение и визуализация поступающей информации.

Первый этап: рефакторинг и оптимизация

Исходный код

```
test_vizual.py  test_data.txt  arduino&python.py  vizual.py  main.py

1  import time
2  import serial
3  import matplotlib.pyplot as plt
4  import matplotlib.animation as animation
5  from functools import partial
6
7  def anmt(i, dataList, serialCom):
8      #print(serialCom)
9      data_befdec = serialCom.readline()
10     arduinoData_string = data_befdec.decode("utf-8").strip('\r\n') # Decode receive Arduino data as a formatted string
11     print(i) # 'i' is a incrementing variable based upon frames = x argument
12
13     try:
14         data_befdec = serialCom.readline()
15         arduinoData_string = data_befdec.decode("utf-8").strip('\r\n') # Decode receive Arduino data as a formatted string
16         arduinoData_float = float(arduinoData_string) # Convert to float
17         dataList.append(arduinoData_float) # Add to the list holding the fixed number of points to animate
18     except: # Pass if data point is bad
19         print("Error encountered, line was not recorded.")
20         pass
21
22     dataList = dataList[-50:] # Fix the list size so that the animation plot 'window' is x number of points
23
24     ax.clear() # Clear last data frame
25     ax.plot(dataList) # Plot new data frame
26
27     ax.set_ylim([0, 1000]) # Set Y axis limit of plot
28     ax.set_title("Robot 5") # Set title of figure
29     ax.set_ylabel("Value") # Set title of y axis
30
31     dataList = [] # Create empty list variable for later use
32
33     fig = plt.figure() # Create Matplotlib plots fig is the 'higher level' plot window
34     ax = fig.add_subplot(111) # Add subplot to main fig window
35
36     serialCom = serial.Serial("COM13", 9600) # Establish Serial object with COM port and BAUD rate to match Arduino Port/rate
37     #print(serialCom)
38
39     time.sleep(2) # Time delay for Arduino Serial initialization
40
41     #Перезапуск ардуино
42     serialCom.setDTR(False)
43     time.sleep(1)
44     serialCom.flushInput()
45     serialCom.setDTR(True)
46
47     # Matplotlib Animation Function that takes takes care of real time plot.
48     # Note that 'fargs' parameter is where we pass in our dataList and Serial object.
49     ani = animation.FuncAnimation(fig, func = partial(anmt, dataList = dataList, serialCom = serialCom), frames=100, interval=300)
50
51     plt.show() # Keep Matplotlib plot persistent on screen until it is closed
52     serialCom.close()
```

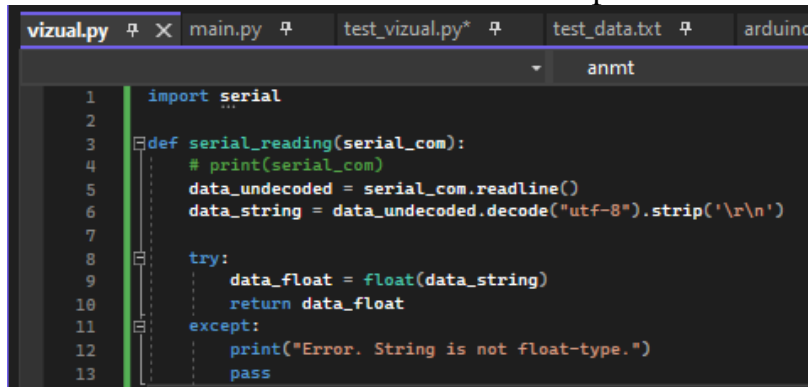
Недостатки

Сейчас, конечно, в глаза сразу бросается множество ошибок: неупорядоченная спагетти-писанина, непонятные переменные, страшные комментарии, где-то лишние, где-то непонятные, использование глобальной области видимости для передачи переменных из `__main__` в функцию обработки

данных, отсутствие единого стиля оформления кода, snake_case вперемешку с camelCase.

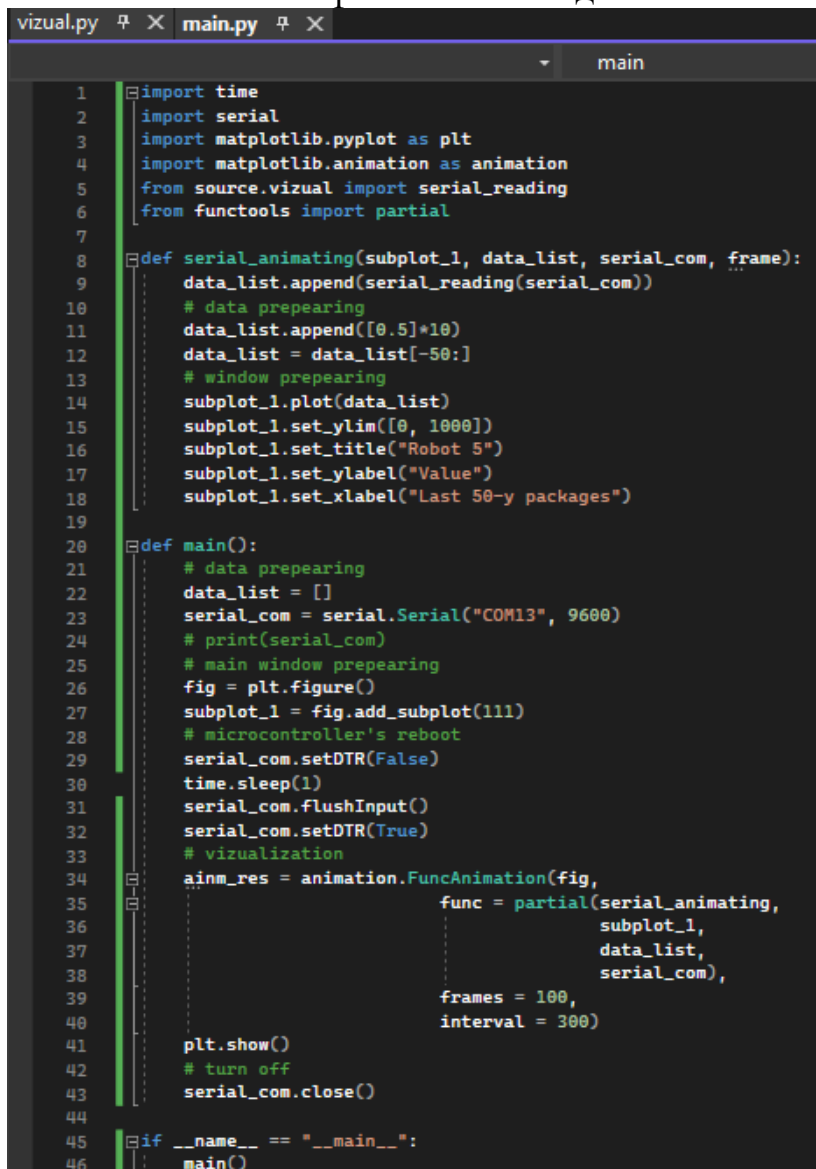
Исправленный код

Для исправления кода пришлось начать с реструктуризации всего проекта. Считывание данных с микроконтроллера и их декодирование, приведение к float типу, было выделено в отдельный модуль для упрощения тестируемости системы и читаемости исполняемого файла.



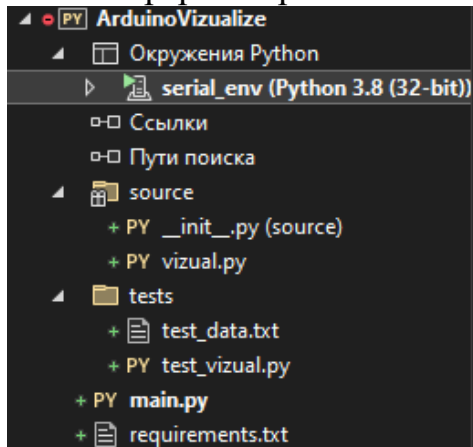
```
1 import serial
2
3 def serial_reading(serial_com):
4     # print(serial_com)
5     data_undecoded = serial_com.readline()
6     data_string = data_undecoded.decode("utf-8").strip('\r\n')
7
8     try:
9         data_float = float(data_string)
10        return data_float
11    except:
12        print("Error. String is not float-type.")
13        pass
```

Исполняемый файл стал выглядеть так:



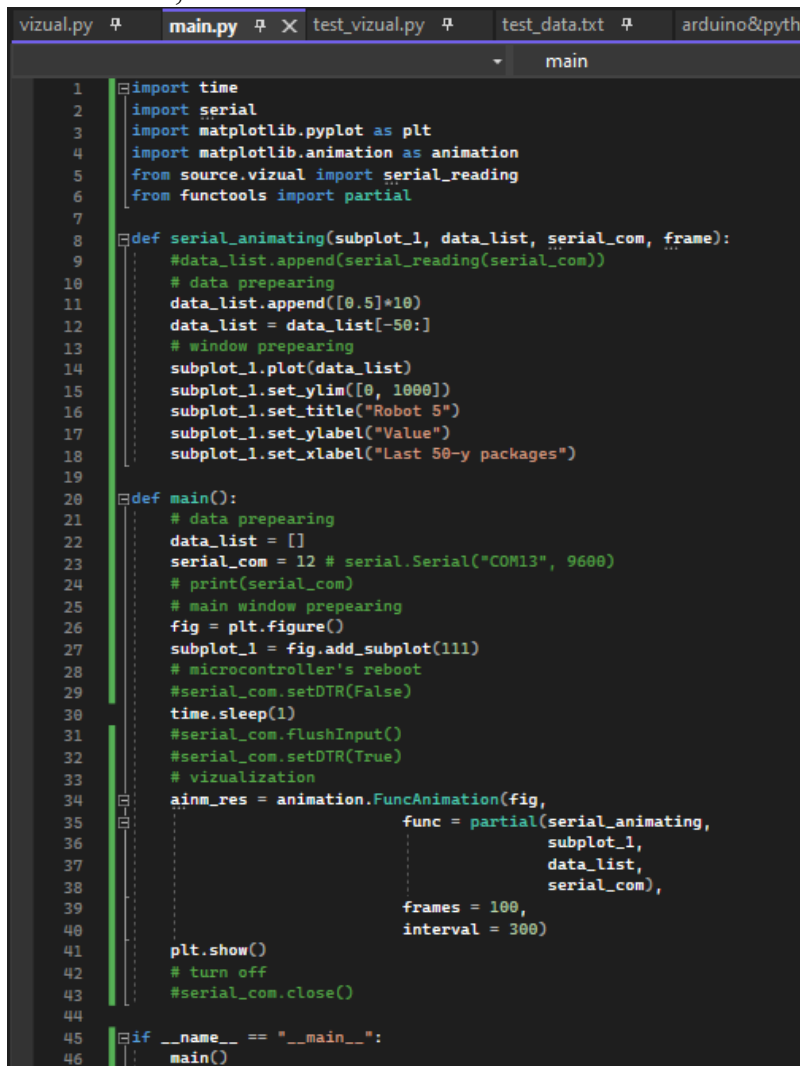
```
1 import time
2 import serial
3 import matplotlib.pyplot as plt
4 import matplotlib.animation as animation
5 from source.vizual import serial_reading
6 from functools import partial
7
8 def serial_animating(subplot_1, data_list, serial_com, frame):
9     data_list.append(serial_reading(serial_com))
10    # data prepearing
11    data_list.append([0.5]*10)
12    data_list = data_list[-50:]
13    # window prepearing
14    subplot_1.plot(data_list)
15    subplot_1.set_ylim([0, 1000])
16    subplot_1.set_title("Robot 5")
17    subplot_1.set_ylabel("Value")
18    subplot_1.set_xlabel("Last 50-y packages")
19
20 def main():
21     # data prepearing
22     data_list = []
23     serial_com = serial.Serial("COM13", 9600)
24     # print(serial_com)
25     # main window prepearing
26     fig = plt.figure()
27     subplot_1 = fig.add_subplot(111)
28     # microcontroller's reboot
29     serial_com.setDTR(False)
30     time.sleep(1)
31     serial_com.flushInput()
32     serial_com.setDTR(True)
33     # vizualization
34     ainm_res = animation.FuncAnimation(fig,
35                                       func = partial(serial_animating,
36                                                       subplot_1,
37                                                       data_list,
38                                                       serial_com),
39                                       frames = 100,
40                                       interval = 300)
41     plt.show()
42     # turn off
43     serial_com.close()
44
45 if __name__ == "__main__":
46     main()
```

Иерархия проектной области:



Затем появилась необходимость протестировать написанное. Было желание написать unit тесты, используя библиотеку pytest, однако проект получился состоящим из двух частей: получения данных с микроконтроллера и визуализации данных. Тестировать визуализацию через assert, мягко говоря, неудобно, а для работы с последовательным портом необходима плата, которой под рукой не оказалось, поэтому от тестирования в “приличном” виде пришлось отказаться.

Чтобы довести работу до разумного конца работа с портами была исключена, вместо этого в список стало записываться просто одно и то же число.



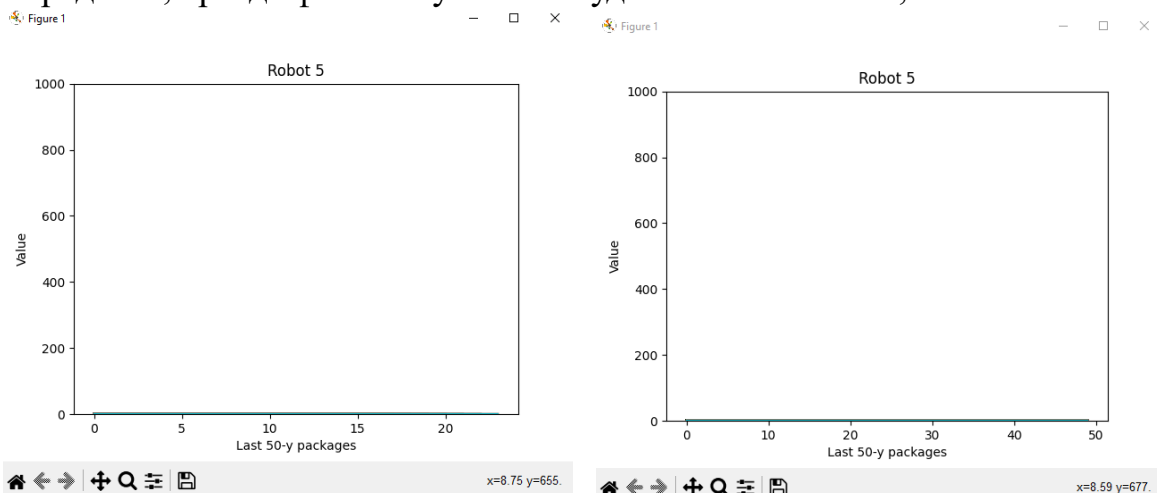
Requirements.txt

```
altgraph==0.17.4
contourpy==1.1.1
cyclер==0.12.1
fonttools==4.55.3
future==1.0.0
importlib-metadata==8.5.0
importlib-resources==6.4.5
iso8601==2.1.0
kiwisolver==1.4.7
matplotlib==3.7.5
numpy==1.24.4
packaging==24.2
pefile==2023.2.7
pillow==10.4.0
pip==19.2.3
pyinstaller==6.11.1
pyinstaller-hooks-contrib==2024.10
pyparsing==3.1.4
python-dateutil==2.9.0.post0
pywin32-ctypes==0.2.3
PyYAML==6.0.2
serial==0.0.97
setuptools==75.3.0
six==1.17.0
zipp==3.20.2
```

Второй этап: подготовка exe-файла

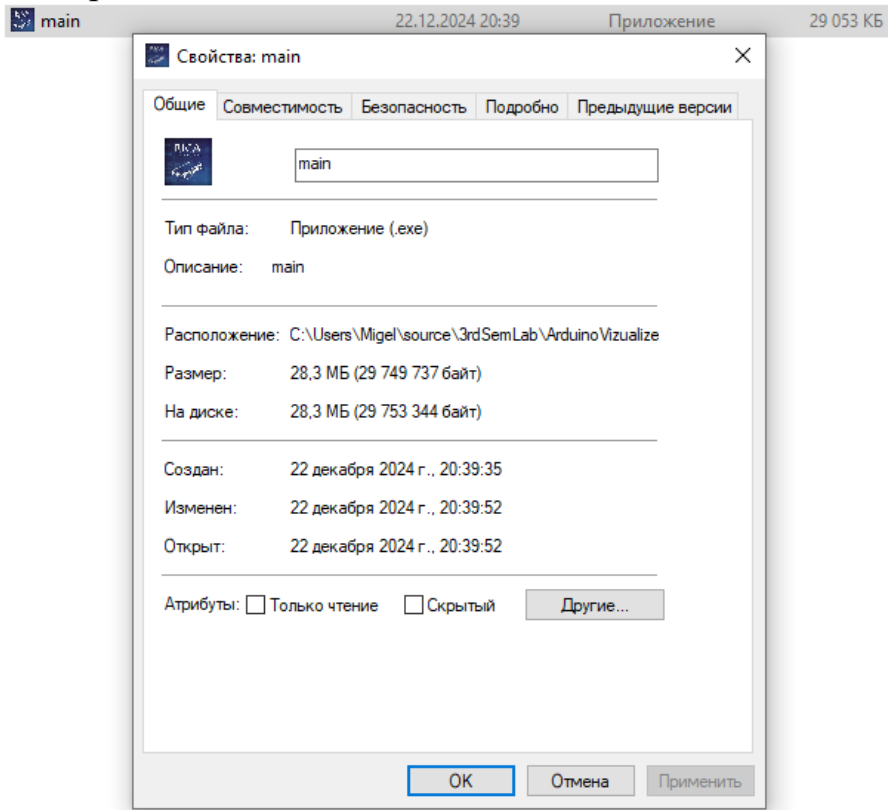
Для использования изучаемого кода раньше приходилось запускать отладчик python'a, искать нужный файл, а затем его запускать, причём вся работа происходила в множество действий и с открытым отладчиком. Сегодня опыт подсказал, что всё смотрелось бы значительно приятнее, будь тот код отдельным мини-приложением — захотелось попробовать.

Был выбран простейший инструмент подготовки запускаемых файлов — pyinstaller. Не отказал себе и в установке иконки “приложения”. Результат порадовал, правда файл получился на удивление большим, почти 30 МБ.



На графиках видно, как постепенно нижняя ось доходит до отметки в 50 последних принятых пакетов и затем перестаёт изменяться (из-за чистки обрабатываемого списка).

Получившееся приложение:



Проверка диспетчера задач показала, что при закрытии окна, поток не виснет фоновым процессом, а полностью удаляется.

Диспетчер задач

Файл Параметры Вид

Процессы Производительность Журнал приложений Автозагрузка Пользователи Подробности Службы

Имя	Состояние	14% ЦП	47% Память	0% Диск	0% Сеть	9% GPU	Ядро GPU	Энергопотребл...	Тенденция эн...
Приложения (5)									
> Microsoft Visual Studio 2022 (8)		0,2%	958,9 МБ	0,1 МБ/с	0 Мбит/с	5,1%	Графический процессор 0 - 3D	Очень низкое	Очень низкое
> Microsoft Word		0,1%	78,3 МБ	0 МБ/с	0 Мбит/с	0%		Очень низкое	Очень низкое
> Yandex with voice assistant Alic...		0%	1 057,8 МБ	0 МБ/с	0,1 Мбит/с	0%		Очень низкое	Очень низкое
> Диспетчер задач		1,2%	31,0 МБ	0,1 МБ/с	0 Мбит/с	0%		Очень низкое	Очень низкое
> Яндекс.Музыка (2)		4,2%	113,6 МБ	0,1 МБ/с	0 Мбит/с	0%	Графический процессор 0 - 3D	Умеренный	Очень низкое