

**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

|  |
| --- |
| **РТУ МИРЭА** |
|  |
| **Институт кибербезопасности и цифровых технологий (ИКБ)** |
|  |
| КБ-2 «Прикладные информационные технологии» |

**ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОГО ЗАДАНИЯ №11**

**В РАМКАХ ДИСЦИПЛИНЫ «**[**БИОМЕТРИЧЕСКИЕ СИСТЕМЫ АУТЕНТИФИКАЦИИ**](https://online-edu.mirea.ru/course/view.php?id=5957)**»**

Выполнил:

Студент 3-ого курса

Учебной группы БИСО-02-22

Зубарев В.С.

Для анализа клавиатурного почерка была написана программа, которая захватывает входящие нажатия клавиатуры и анализирует их.

Листинг программы

import time  
import random  
import matplotlib.pyplot as plt  
from pynput import keyboard  
  
# Генерация фразы для анализа клавиатурного почерка  
def generate\_analysis\_phrase():  
 return "Сегодня мы изучаем особенности клавиатурного почерка"  
  
# Основной класс для анализа клавиатурного почерка  
class KeyboardAnalyzer:  
 def \_\_init\_\_(self, analysis\_phrase):  
 self.analysis\_phrase = analysis\_phrase  
 self.input\_phrase = ""  
 self.start\_time = None  
 self.key\_times = []  
 self.release\_times = []  
  
 def on\_press(self, key):  
 try:  
 if self.start\_time is None:  
 self.start\_time = time.time()  
  
 char = key.char if hasattr(key, 'char') else str(key)  
 self.input\_phrase += char  
  
 # Записываем время нажатия  
 self.key\_times.append((char, time.time() - self.start\_time))  
 except Exception as e:  
 print(f"Ошибка: {e}")  
  
 def on\_release(self, key):  
 try:  
 char = key.char if hasattr(key, 'char') else str(key)  
 self.release\_times.append((char, time.time() - self.start\_time))  
 except Exception as e:  
 print(f"Ошибка: {e}")  
  
 if key == keyboard.Key.enter: # Завершение ввода при нажатии Enter  
 return False  
  
 def calculate\_metrics(self):  
 dwell\_times = []  
 flight\_times = []  
 for i, (char, press\_time) in enumerate(self.key\_times):  
 # Рассчитываем Dwell Time  
 if i < len(self.release\_times) and self.release\_times[i][0] == char:  
 dwell\_time = self.release\_times[i][1] - press\_time  
 dwell\_times.append(dwell\_time)  
  
 # Рассчитываем Flight Time  
 if i > 0:  
 flight\_time = press\_time - self.key\_times[i - 1][1]  
 flight\_times.append(flight\_time)  
  
 avg\_speed = len(self.key\_times) / (self.key\_times[-1][1] if self.key\_times else 1)  
 rhythm = sum(abs(flight\_times[i] - flight\_times[i - 1]) for i in range(1, len(flight\_times))) / len(flight\_times) if len(flight\_times) > 1 else 0  
  
 return {  
 "dwell\_times": dwell\_times,  
 "flight\_times": flight\_times,  
 "average\_speed": avg\_speed,  
 "rhythm": rhythm,  
 "keys": [item[0] for item in self.key\_times]  
 }  
  
 def plot\_analysis(self, metrics):  
 keys = metrics["keys"]  
 times = [item[1] for item in self.key\_times]  
  
 # График времени нажатий клавиш  
 plt.figure(figsize=(6, 4))  
 plt.plot(times, [i for i in range(len(times))], marker='o', label="Нажатия клавиш")  
 for i, key in enumerate(keys):  
 plt.text(times[i], i, key, fontsize=9, ha='right')  
 plt.xlabel("Время (секунды)")  
 plt.ylabel("Порядок нажатия")  
 plt.title("Нажатия клавиш")  
 plt.legend()  
 plt.grid()  
 plt.show()  
  
 # График Dwell Time  
 plt.figure(figsize=(6, 4))  
 plt.bar(range(len(metrics["dwell\_times"])), metrics["dwell\_times"], label="Dwell Time")  
 plt.xticks(range(len(keys)), keys, rotation=45)  
 plt.xlabel("Символы")  
 plt.ylabel("Время (сек)")  
 plt.title("Dwell Time (Удержание клавиши)")  
 plt.grid()  
 plt.show()  
  
 # График Flight Time  
 plt.figure(figsize=(6, 4))  
 plt.bar(range(len(metrics["flight\_times"])), metrics["flight\_times"], label="Flight Time")  
 plt.xticks(range(len(keys) - 1), keys[:-1], rotation=45)  
 plt.xlabel("Символы")  
 plt.ylabel("Время (сек)")  
 plt.title("Flight Time (Время между нажатиями)")  
 plt.grid()  
 plt.show()  
  
 # Ритмичность и средняя скорость  
 plt.figure(figsize=(6, 4))  
 plt.bar([0, 1], [metrics["average\_speed"], metrics["rhythm"]], tick\_label=["Скорость", "Ритм"])  
 plt.title("Средняя скорость и ритмичность")  
 plt.grid()  
 plt.show()  
  
# Основная функция  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("Анализатор клавиатурного почерка")  
 analysis\_phrase = generate\_analysis\_phrase()  
 print(f"Введите любую фразу. Например: {analysis\_phrase}")  
  
 analyzer = KeyboardAnalyzer(analysis\_phrase)  
  
 with keyboard.Listener(  
 on\_press=analyzer.on\_press,  
 on\_release=analyzer.on\_release) as listener:  
 listener.join()  
  
 metrics = analyzer.calculate\_metrics()  
 analyzer.plot\_analysis(metrics)

На первом скрине показан график нажатия клавиш в зависимости от времени, который иллюстрирует общую скорость печати и позволяет предполагать постановку рук при печати(за чет разницы между нажатиями клавиш) и точность их нажатия(по количествам backspace)

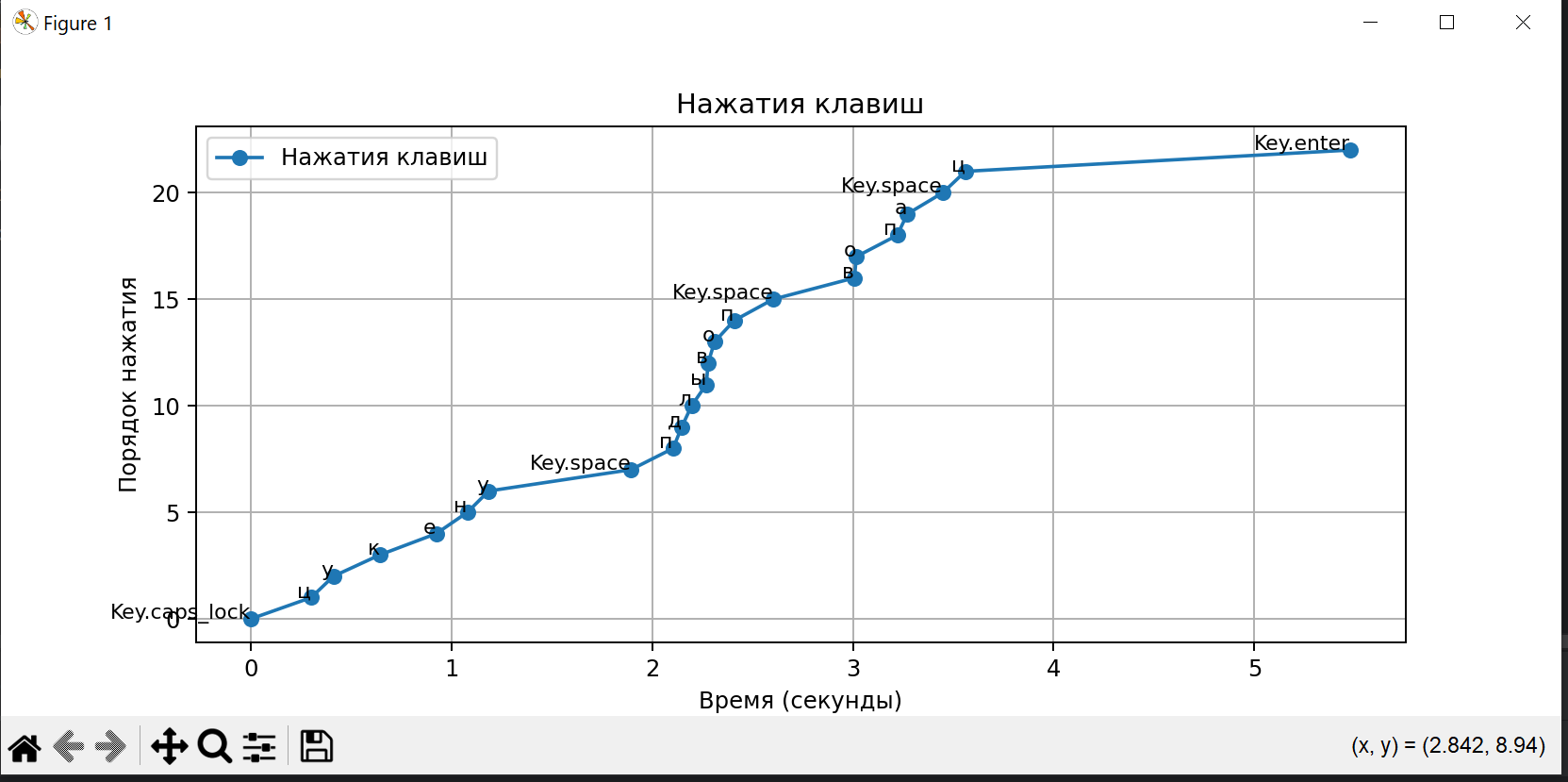


График удержания клавиш, позволяет анализировать общую скорость печати и время реакции субъекта на событие (сколько он удерживает клавишу после нажатия)

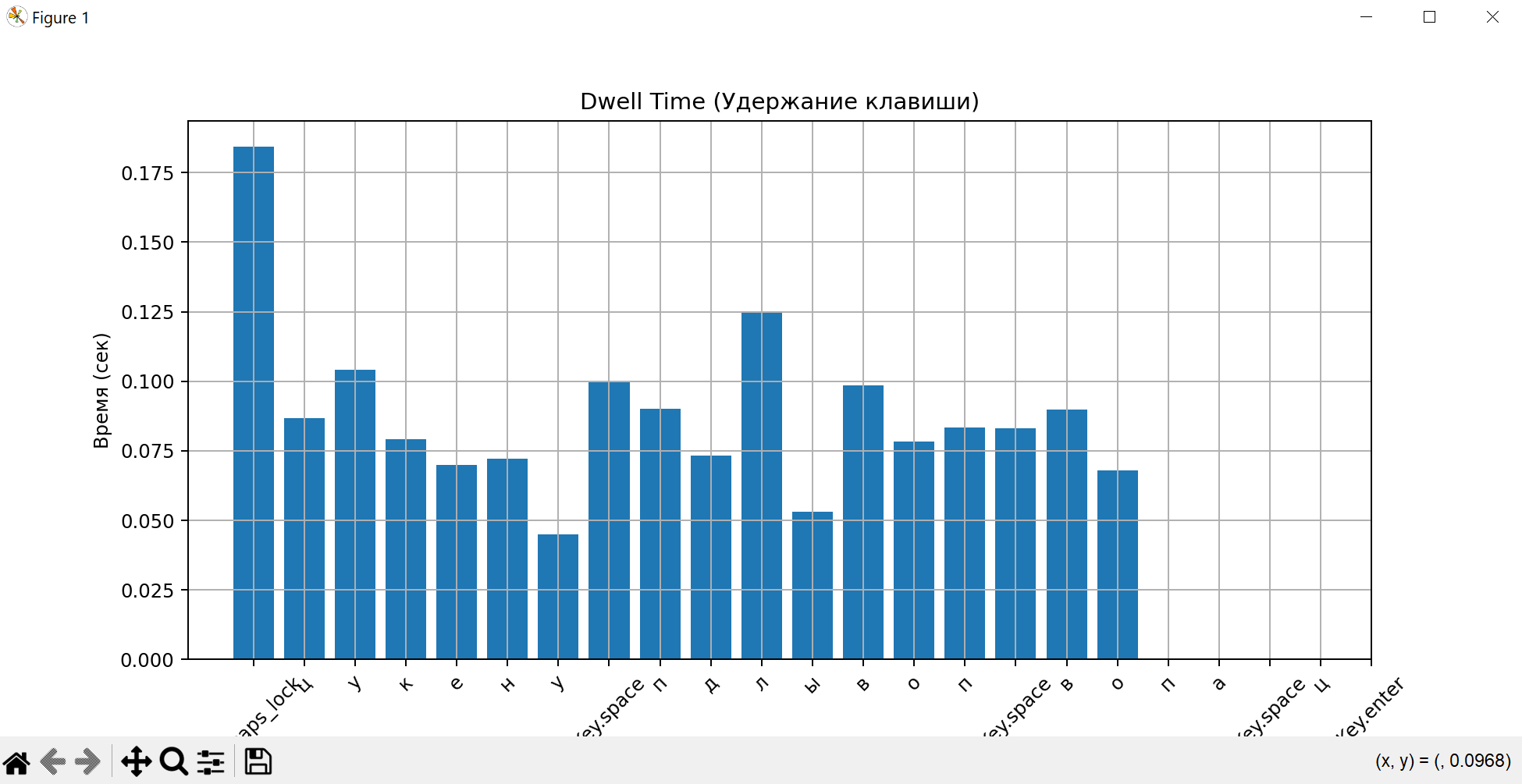
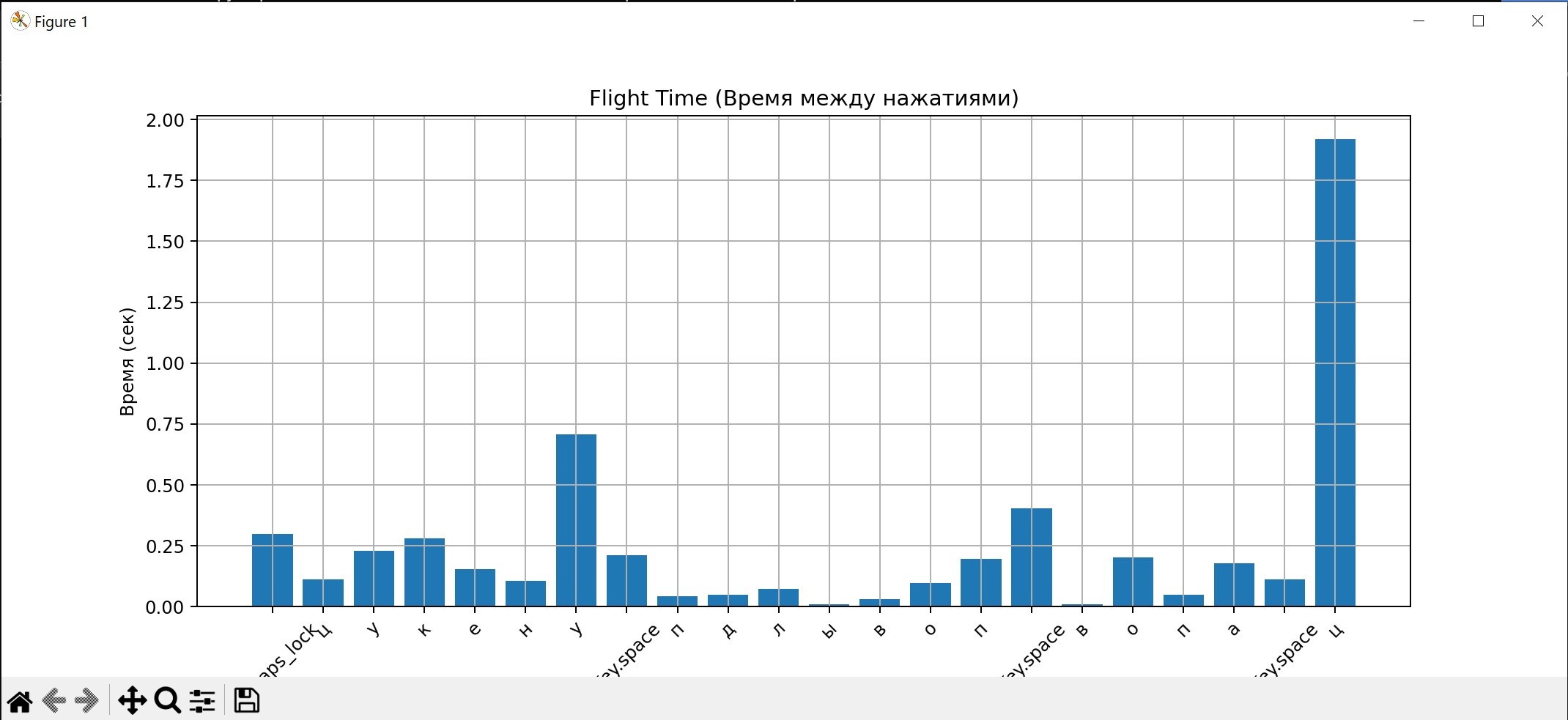


График времени между нажатиями позволяет анализировать насколько адаптивным расположением рук пользуется субъект (большая разница между нажатиями свидетельствует о плохом расположении рук\ отсутствию выработанного навыка печати)



Средняя скорость и ритм – являются статистическими показателями которые можно использовать для сравнения с эталонными значениями и идентификации субъекта (становятся точнее со временем)

