Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
УЧЕБНЫЙ ЦЕНТР ОБЩЕЙ ФИЗИКИ ФТФ

УНИВЕРСИТЕТ ИТМО

Группа М32021 **К работе допущен** _____

Студент Лопатенко Георгий Валентинович **Работа выполнена** _____

Преподаватель Тимофеева Э.О. **Отчет принят** _____

Рабочий протокол и отчет по лабораторной работе №3.00

Изучение электрических сигналов с помощью лабораторного осциллографа

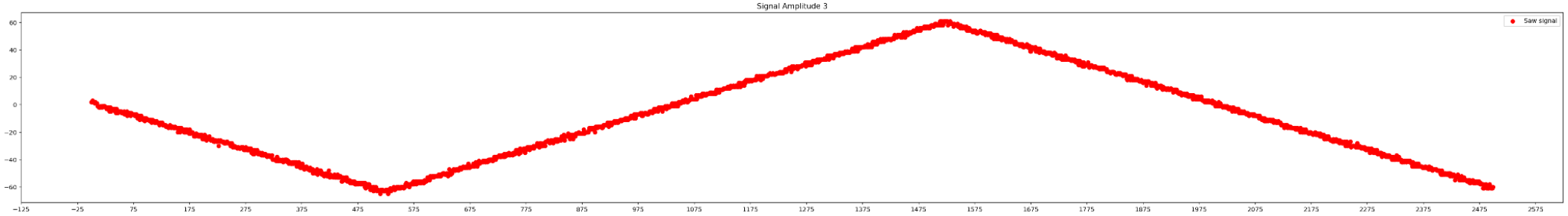
1. Цель работы:
Ознакомление с устройством осциллографа, изучение с его помощью процессов в электрических цепях.

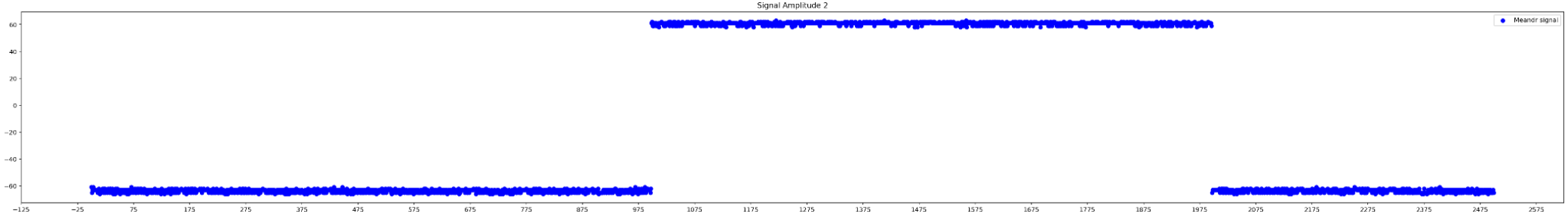
2. Задачи, решаемые при выполнении работы:
1. Снять формы сигналов при прямом подключении;
2. Снять формы сигналов при подключении двух каналов колебательных сигналов;
3. Получить фигуры Лиссажу с разными входными данными по фазам;
4. Получить конфигурацию наслоения шума на сигнал, изучить сущность такого сигнала.

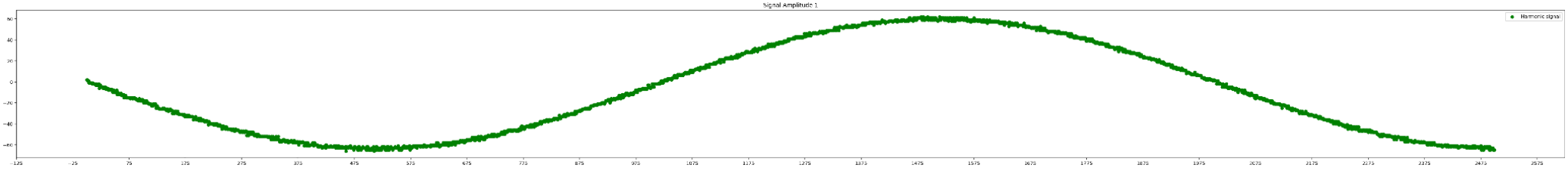
3. Приборы:
1. Цепь, собранная на стенде СЗ-ЭМ01;
2. Цифровой осциллограф;
3. Функциональный генератор

Часть 1: при прямом подключении генератора сигналов специальной формы по шлангу на осциллограф последний выступает в роли высокоомного высокочастотного дифференциального вольтметра. Виды сигналов могут получаться разные за счет изменения функционального закона изменения напряжения. Для фиксированной частоты и амплитуды напряжения были получены данные для трех видов сигналов - гармонический, меандр и пилообразный.

	Амплитуда, В	RMS, мВ	Частота, кГц
sin	0.976	350	1.001
меандр	1.00	498	1.000
пила	1.02	286	1.002

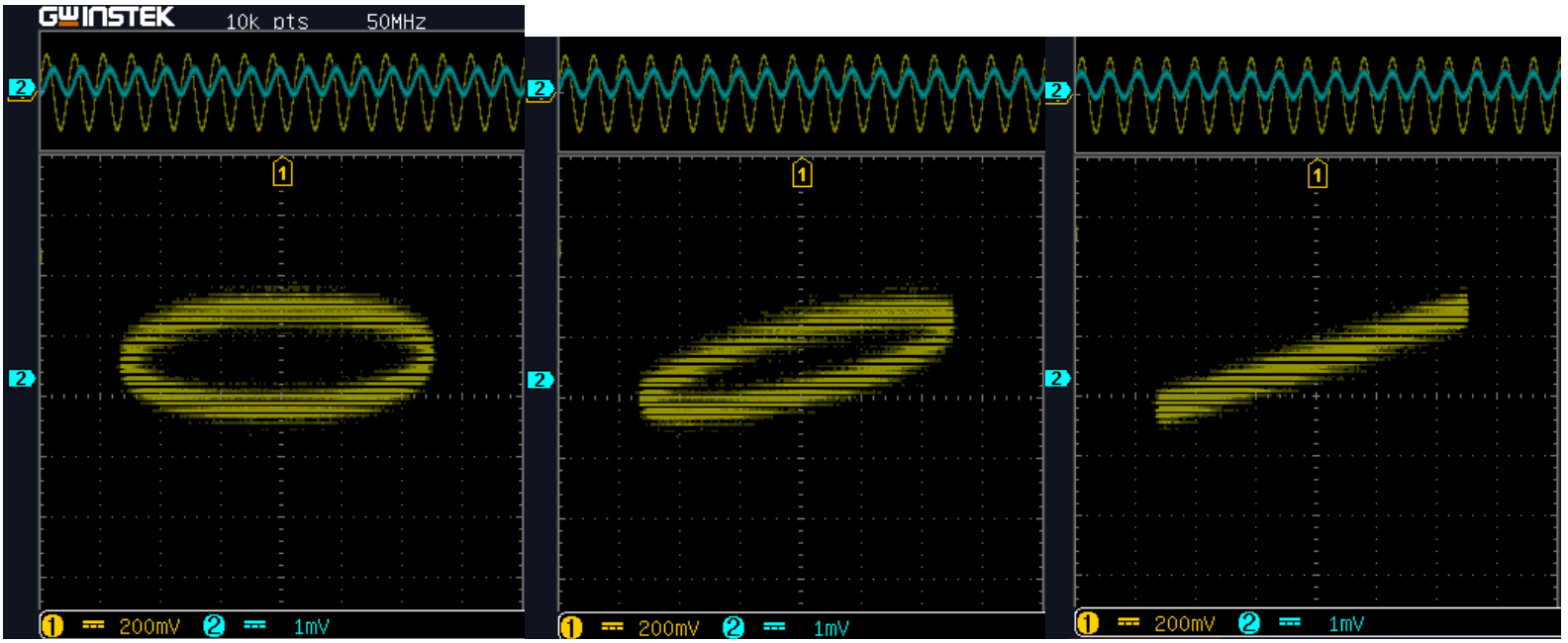

Signal Amplitude 3
Saw signal


Signal Amplitude 2
Meandr signal


Signal Amplitude 1
Harmonic signal

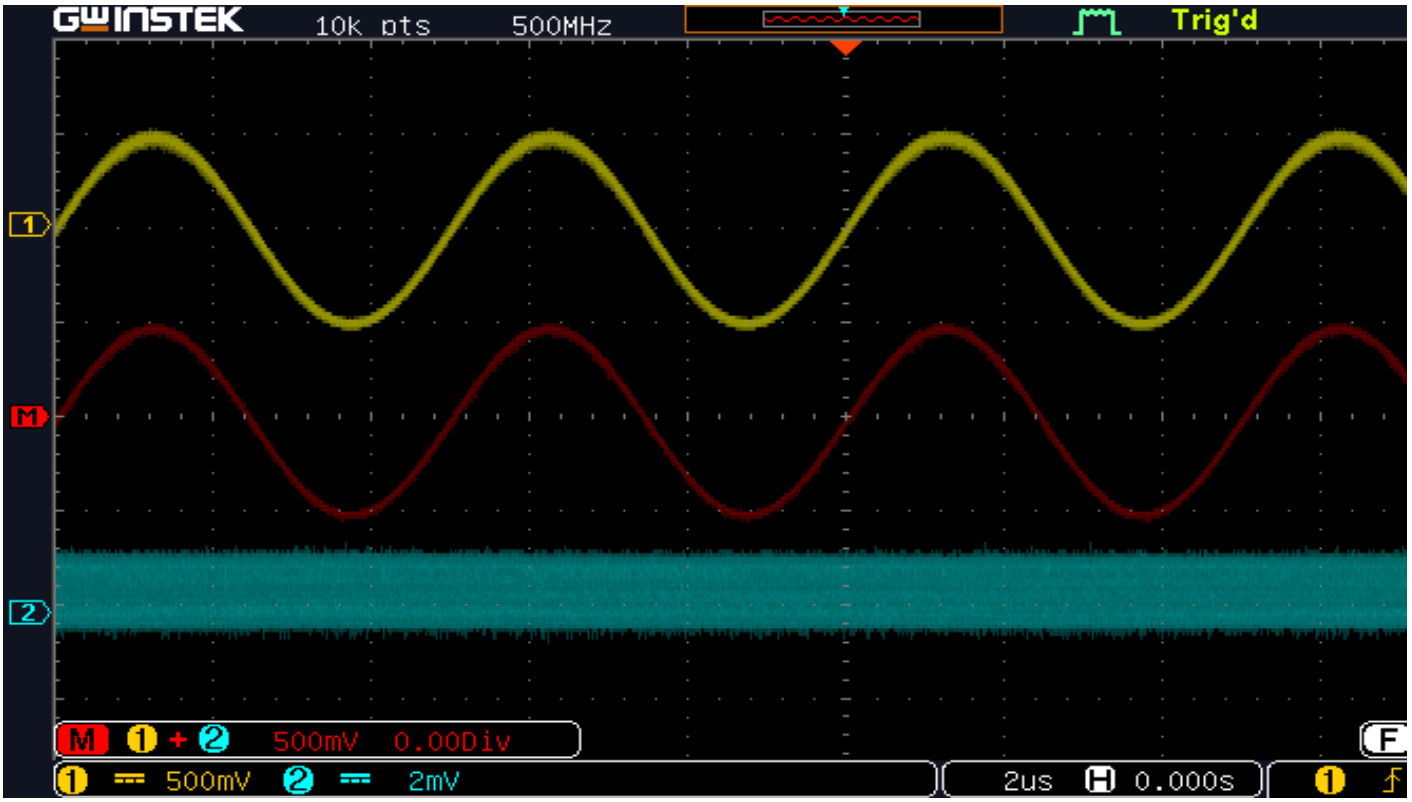
Часть 2: при подключении к осциллографу через два шланга (по двум каналам) будем регистрировать принцип Гюйгенса сложения колебаний на X-Y входах. При этом получение фигур Лиссажу нам гарантируют кратные значения частот (в нашем случае 1:1). Заметим, что теория подтверждается экспериментом с той только поправкой, что при сдвиге фаз на $\frac{\pi}{4}$ не было получено окружности (потому что входы X и Y не согласованы хотя бы по амплитуде)

График 2: фигуры Лиссажу при разности фаз 90°, 45° и 0°



Часть 3: при наложении шума на сигнал наблюдаем сложение значений сигнала и случайной величины на отрезке - белого шума.

График 3: картина наложения шума на гармонический сигнал



12. Выводы и анализ результатов работы:
Самое занимательное в этой лабораторной остается студенту на самостоятельное усвоение, можно найти информацию о нескольких методах фильтрации шума сигнала и попробовать некоторые идеи воплотить в жизнь.
Я реализовал модель средних измерений, получив для каждого значения измерения среднее арифметическое из нескольких файлов. То есть фактически обрабатываемый сигнал стал средним (но именно средним арифметическим) по 9 формам, что получилось сохранить в течение выполнения лабораторной. Синусоида чуть сгладилась, но эффект не радовал из-за больших скачков.
Далее применил медианный подход, когда для отфильтрованных данных считается медиана из трех значений в каретке (то есть мы проходимся по измерениям буффером в три элемента и затираем итерируемое значение на медианное).
Естественно, можно ввести веса, которые в сумме дают единицу, для нормализации синусоиды. Легко вводим адаптивный коэффициент, параметры для расчета которого приходилось искать эмпирическим путем. Если рекурсивно запуститься таким фильтром на одних и тех же измерениях, то можно получить довольно гладкую моду.
Немного прочитал про фильтр Калмана, но его ругают за неприятные вычисления для значений с плавающей точкой.

реализацию смотреть ниже

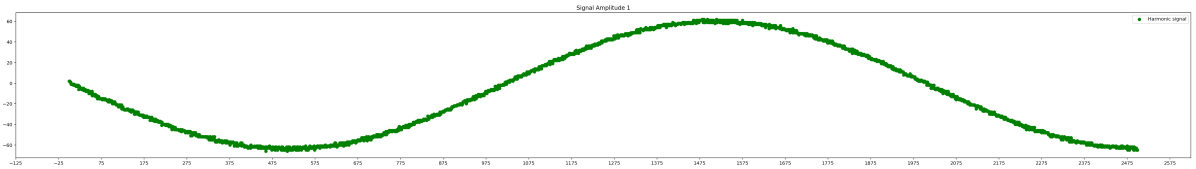
In [ ]:
```
%config InlineBackend.figure_format = 'retina'
```

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
```

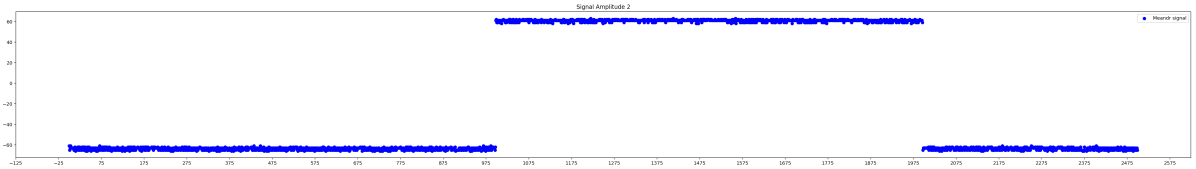
In [ ]:
```
def show_graph(path: str, id: int, signal_type: str, mode: str):
    data_unfiltered = pd.read_csv(f'/content/drive/MyDrive/data_lab3_00/{path}', us
    full_data = np.array([int(elem[0]) for elem in data_unfiltered.values[25:]])
    data = []
    for i in range(0, len(full_data) // 4, 20):
        data.append(full_data[i])

    # print(data[:100])
    data_cut = full_data[:2500]
    fig, ax = plt.subplots(1, 1, figsize=(35, 5))
    ax.scatter(x=[i for i in range(len(data_cut))], y=data_cut, c=mode, label=f'{si
    ax.set(title=f'Signal Amplitude {id}')
    ax.legend()
    start, end = ax.get_xlim()
    ax.xaxis.set_ticks(np.arange(start, end, 100))
    fig.tight_layout()
```

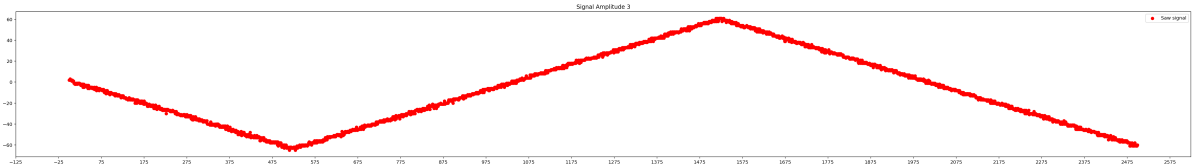
In [ ]:
```
show_graph("DS0001.CSV", 1, "Harmonic", 'g')
```



In [ ]:
```
show_graph("DS0002.CSV", 2, "Meandr", 'b')
```



In [ ]:
```
show_graph("DS0003.CSV", 3, "Saw", 'r')
```

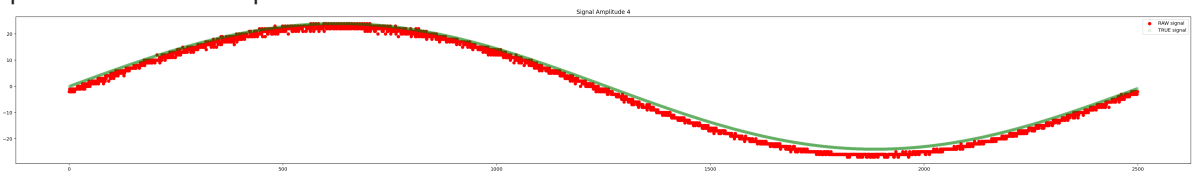


In [49]:
```
def show_minimize_graph(path: str, id: int, signal_type: str, mode: str):
    data_unfiltered = pd.read_csv(f'/content/drive/MyDrive/data_lab3_00/{path}', us
    full_data = np.array([int(elem[0]) for elem in data_unfiltered.values[25:]])

    data_cut = full_data[:2500]
    sampling_period = 2 ** (-9)
    firmware = 1.28
    amplitude = max(data_cut)
    # amplitude = (max(data_cut) + abs(min(data_cut))) // 2
    fig, ax = plt.subplots(1, 1, figsize=(35, 5))
    lin_space = [i for i in range(len(data_cut))]
    ax.scatter(x=lin_space, y=data_cut, c=mode, label=f'{signal_type} signal')
    ax.scatter(x=lin_space, y=[np.sin(elem * sampling_period * firmware) * amplitu
```

```
    ax.set(title=f'Signal Amplitude {id}')
    ax.legend()
    fig.tight_layout()
    print(f"plotted for {len(data_cut)} points")
show_minimize_graph("DS0008.CSV", 4, "RAW", 'r')
```

```
plotted for 2500 points
```

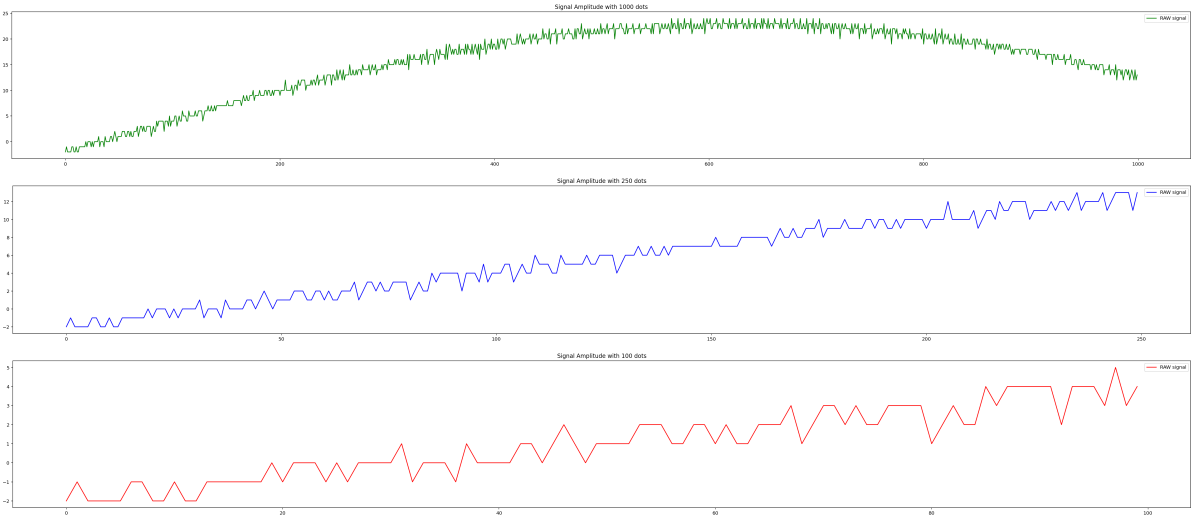


In [119…

```
def show_cleared_graph(path: str, id: int, signal_type: str, mode: str):
    data_unfiltered = pd.read_csv(f'/content/drive/MyDrive/data_lab3_00/{path}', us
    data_cut = np.array([int(elem[0]) for elem in data_unfiltered.values[25:]])[:id
    sampling_period = 2 ** (-9)

    fig, ax = plt.subplots(1, 1, figsize=(35, 5))
    lin_space = [i for i in range(len(data_cut))]
    for dot in lin_space[:-1]:
        ax.plot(dot, data_cut[dot], dot + 1, data_cut[dot + 1])
    plt.plot(lin_space, data_cut, label=f'{signal_type} signal', c=mode)
    ax.set(title=f'Signal Amplitude with {id} dots')
    ax.legend()
    fig.tight_layout()
    plt.show()

show_cleared_graph("DS0008.CSV", 1000, "RAW", 'g')
show_cleared_graph("DS0008.CSV", 250, "RAW", 'b')
show_cleared_graph("DS0008.CSV", 100, "RAW", 'r')
```



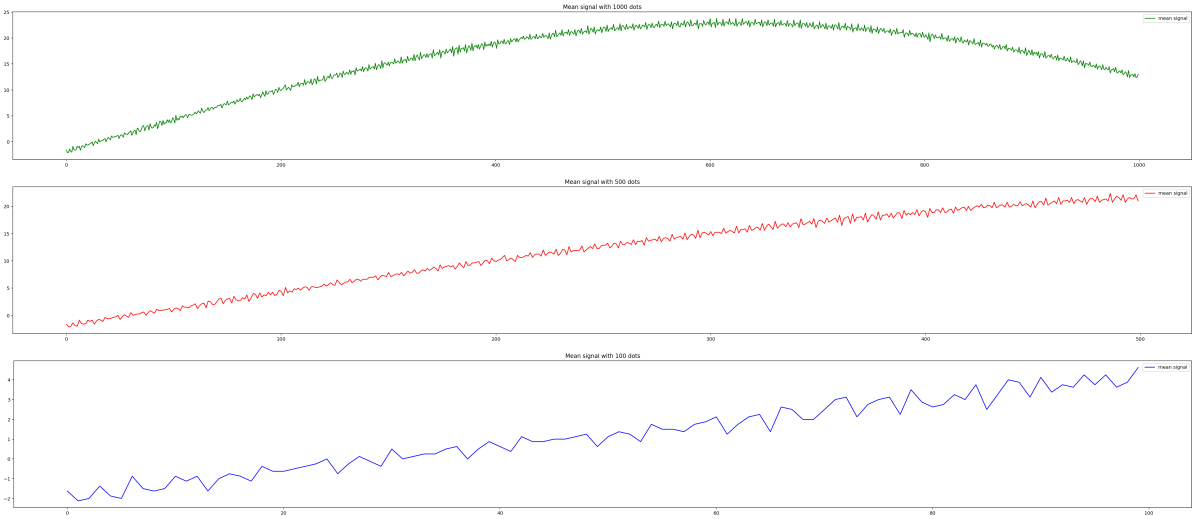
In [84]:

```
def get_data(first_file_num: int, last_file_num: int, dots_n: int):
    data_bank = []
    for file_num in range(first_file_num, last_file_num + 1):
        data_unfiltered = pd.read_csv(f'/content/drive/MyDrive/data_lab3_00/DS000{f
        data_bank.append([int(elem[0]) for elem in data_unfiltered.values[25:]][:dc
    data_mean = [0 for _ in range(len(data_bank[0]))]
    for pos in range(len(data_bank[0])):
        for arr_num in range(len(data_bank)):
            data_mean[pos] += data_bank[arr_num][pos] / len(data_bank)
    return data_mean

def show_mean_graph(dots_n: int, color: str):
    data_cut = get_data(8, 15, dots_n)
    sampling_period = 2 ** (-9)
```

```
    fig, ax = plt.subplots(1, 1, figsize=(35, 5))
    lin_space = [i for i in range(len(data_cut))]
    for dot in lin_space[:-1]:
        ax.plot(dot, data_cut[dot], dot + 1, data_cut[dot + 1])
    plt.plot(lin_space, data_cut, label='mean signal', c=color)
    ax.set(title=f'Mean signal with {dots_n} dots')
    ax.legend()
    fig.tight_layout()
    plt.show()

show_mean_graph(1000, 'g')
show_mean_graph(500, 'r')
show_mean_graph(100, 'b')
```



In [122…

```
def get_data(first_file_num: int, last_file_num: int, dots_n: int) -> list:
    data_bank = []
    for file_num in range(first_file_num, last_file_num + 1):
        data_unfiltered = pd.read_csv(f'/content/drive/MyDrive/data_lab3_00/DS000{f
        data_bank.append([int(elem[0]) for elem in data_unfiltered.values[25:]][:dc
    data_mean = [0 for _ in range(len(data_bank[0]))]
    for pos in range(len(data_bank[0])):
        for arr_num in range(len(data_bank)):
            data_mean[pos] += data_bank[arr_num][pos] / len(data_bank)
    return data_mean

def median_adaptive_filter(mean_data: list, s_k: float, max_k: float, d: float) ->
    median_data = []
    for i in range(len(mean_data) - 2):
        a, b, c = mean_data[i:i+3]
        median_data.append(max(a, c) if (max(a, b) == max(b, c)) else max(b, min(a,
    for _ in range(100):
        for i in range(len(median_data) - 1):
            normalised, new = median_data[i:i+2]
            k = s_k if (abs(new - normalised) < d) else max_k
            median_data[i] += (new - median_data[i]) * k
    return median_data

def show_mean_graph(dots_n: int, color: str, s_k: int, max_k: int, d: int):
    data_cut = median_adaptive_filter(get_data(8, 15, dots_n), s_k, max_k, d)
    sampling_period = 2 ** (-9)

    raw_data_unfiltered = pd.read_csv(f'/content/drive/MyDrive/data_lab3_00/DS0008.
    raw_data_cut = np.array([int(elem[0]) for elem in raw_data_unfiltered.values[25

    fig, ax = plt.subplots(1, 1, figsize=(35, 5))
    lin_space = [i for i in range(len(data_cut))]
    raw_lin_space = [i for i in range(len(raw_data_cut))]
```

```
    for dot in lin_space[:-1]:
        ax.plot(dot, data_cut[dot], dot + 1, data_cut[dot + 1])
    plt.plot(lin_space, data_cut, label='mean/median signal', c=color)
    for dot in raw_lin_space[:-1]:
        ax.plot(dot, raw_data_cut[dot], dot + 1, raw_data_cut[dot + 1])
    plt.plot(raw_lin_space, raw_data_cut, label=f'raw/unfiltered signal', c='r', al
    ax.set(title=f'Mean, Recursive Adaptive Median signal VS Raw signal with {dots_
    ax.legend()
    fig.tight_layout()
    plt.show()

show_mean_graph(1000, 'b', 0.1, 0.8, 1)
show_mean_graph(100, 'b', 0.1, 0.8, 0.9)
show_mean_graph(2500, 'b', 0.1, 0.8, 1)
```