

Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта

КУРСОВАЯ РАБОТА

Дисциплина: «Объектно-ориентированное программирование»

Тема: «Разработка программы для управления плоттером на Python с графическим интерфейсом»

Выполнил

студент группы 3331506/80401

Кунгурцев Е.В.

Руководитель

Ананьевский М.С.

«__» _____ 2025 г

Санкт-Петербург

2025

Оглавление

1. Введение	3
2. Используемые функции	4
3. Возможности программы.....	5
4. Пример использования	6
5. Конструкция станка	7
6. Вывод	8
7. Список литературы.....	9
8. Приложения	10

1. Введение

Целью курсовой работы являлась разработка программы для управления плоттером с использованием Arduino и библиотеки grbl, интерпретирующей G-код.

Программа реализована на Python с применением библиотеки PyQt6 для создания графического интерфейса при помощи Qt designer.

Дополнительно выполнена сборка проекта в исполняемый файл для удобства использования на компьютерах без установленного Python.

Ключевые задачи:

- Реализация взаимодействия с плоттером через COM-порт.
- Создание интерфейса для ручного и автоматического управления.
- Создание и обработка G-кода, создание рисунков из векторного изображения на бумаге

2. Используемые функции

Библиотеки и инструменты

- PyQt6 (виджеты, сигналы, таймер)
- Qt Designer: Разработка интерфейса (см. приложение файл Plotter.ui).
- Библиотека Serial для работы с последовательным портом для отправки команд и чтения ответов.
- QTimer: Асинхронная обработка отправки G-кода и чтения данных.
- PyInstaller: Сборка проекта в EXE-файл.

2. Основные функции программы

- **Класс PlotterControl:**
 - Управление подключением/отключением COM-порта.
 - Отправка команд движения по осям X/Y.
 - Контроль положения пера
 - Обработка очереди G-кода из файла с поддержкой паузы и остановки.
 - Обновление текущих координат и состояния пера в интерфейсе.
- **Интерфейс:**
 - Кнопки для ручного перемещения (\leftrightarrow ↗ ↘ и др.).
 - Поля ввода шага перемещения и скорости (step_size_xy, feed_rate).
 - Отображение лога работы и загруженного G-кода.
 - Меню для выбора файла с G-кодом.

3. Возможности программы

1. Подключение к устройству:

- Автоматическое обнаружение доступных COM-портов.
- Установка скорости передачи данных

2. Ручное управление:

- Перемещение по осям
- Выбор шага и скорости
- Сброс текущих координат в ноль
- Поднятие/опускание пера

3. Работа с G-кодом:

- Загрузка файлов. gcode или .txt.
- Построчное выполнение кода
- Пауза и аварийная остановка выполнения

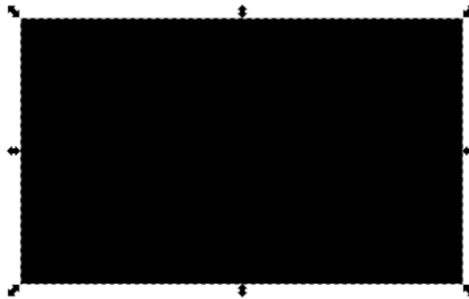
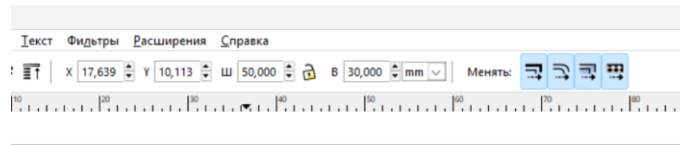
4. Визуализация:

- Отображение текущих координат x, y, состояния пера.
- Логирование команд и ответов устройства в консоли.

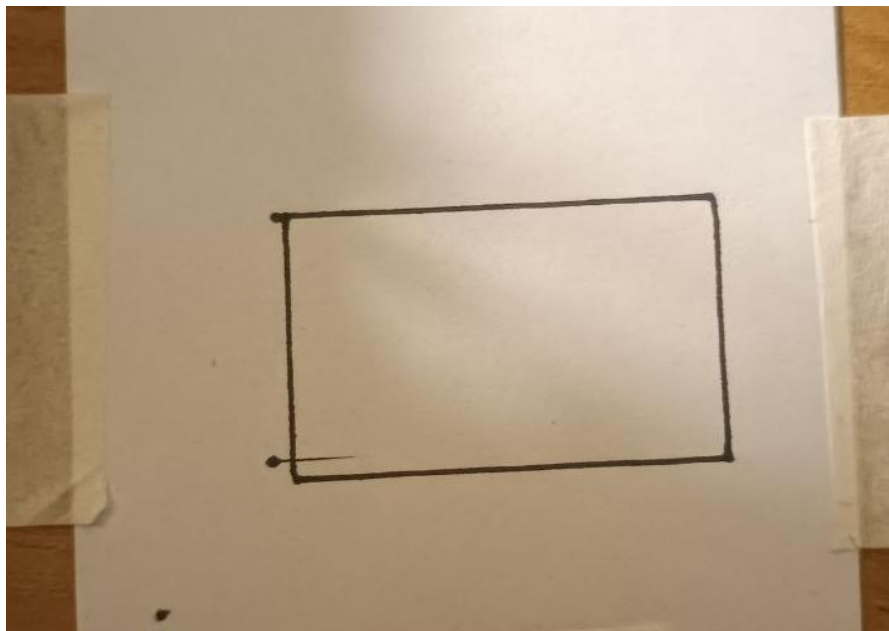
4. Пример использования

Проверка точности станка:

Для проверки был нарисован прямоугольник 50*30:



Получился следующий рисунок:

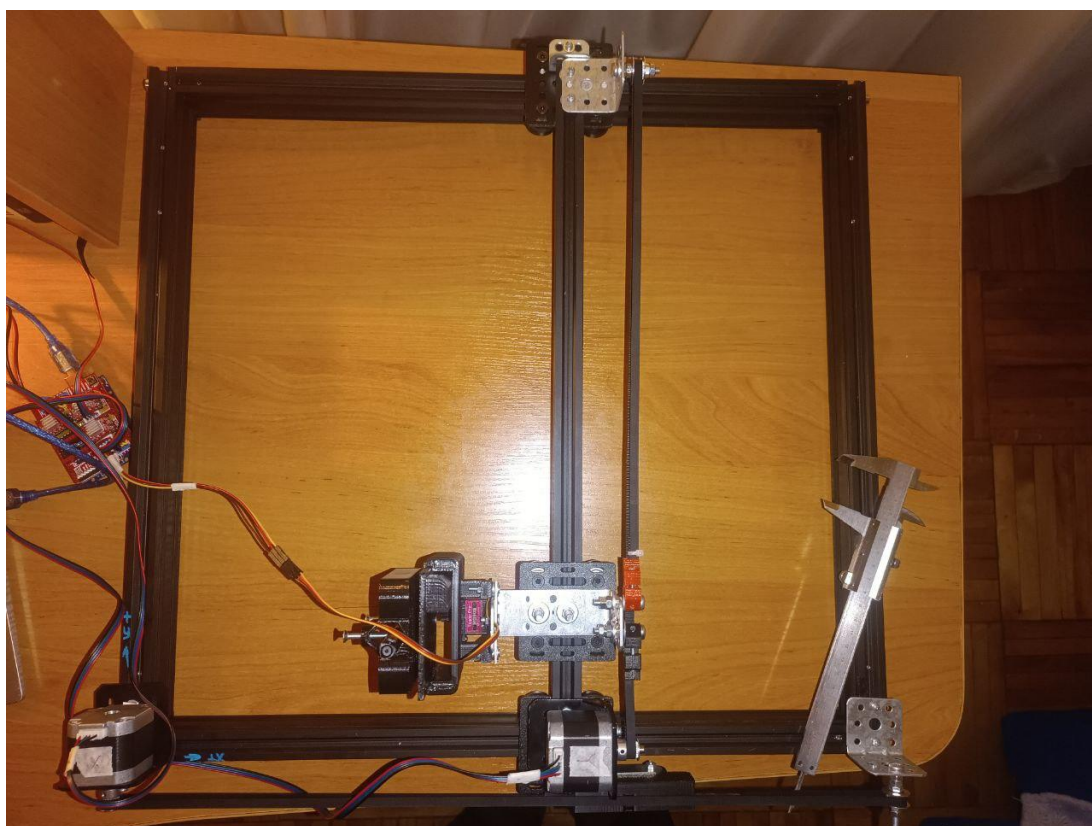
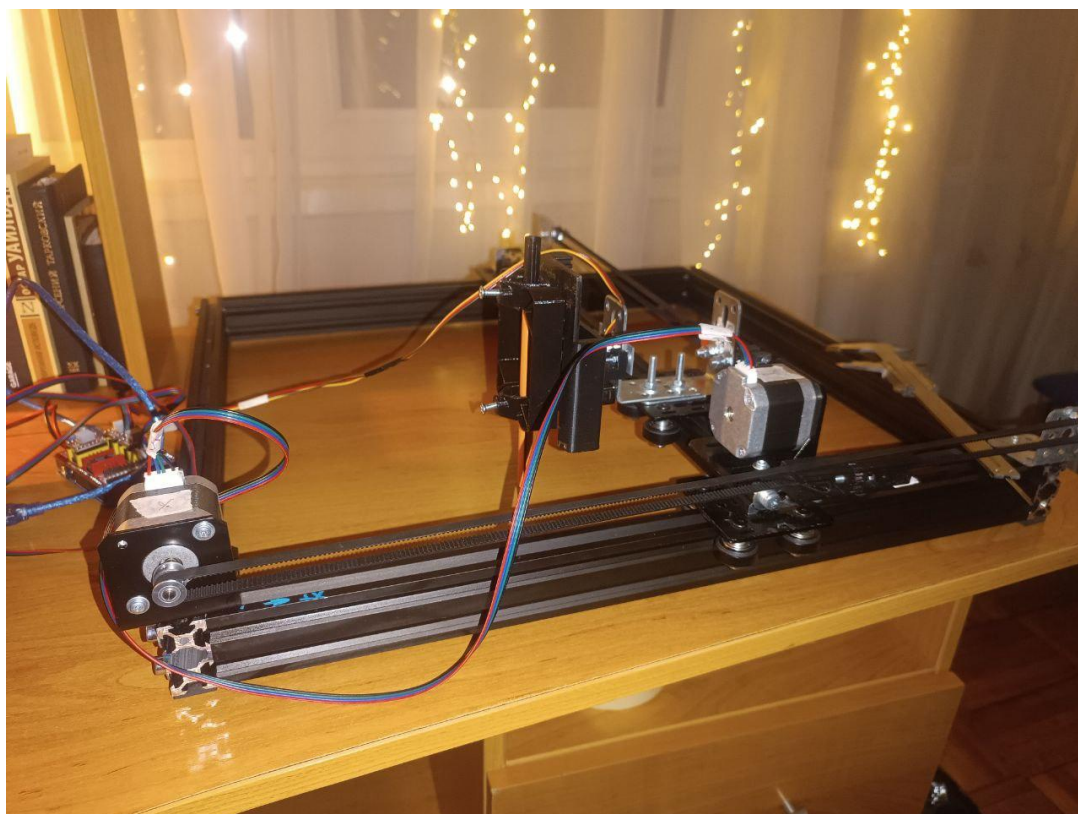


Измеренная штангенциркулем погрешность получилась менее толщины ручки (0,4 мм), размеры прямоугольника по середине линии 50.2*29.9 мм, неровности в углах связаны с проблематичностью закрепления ручки и неровностями стола.

Собственная погрешность станка в каждой из осей порядка 0,04 - 0,08 мм (измерена между конструктивными частями станка). Таким образом, после модификации способа закрепления ручки возможно получить достаточную точность рисунка.

5. Конструкция станка

Фото конструкции станка в 2 проекциях:



6. Вывод

Программа успешно решает поставленные задачи:

- Обеспечивает полный контроль над плоттером через COM-порт.
- Предоставляет интерфейс для ручного и автоматического управления.
- Поддерживает обработку ошибок

Варианты доработки:

- Добавление предпросмотра траектории G-кода.

7. Список литературы

1. **Ikae**. MI-GRBL-Z-AXIS-Servo-Controller [Электронный ресурс]. – URL: <https://github.com/ikaе/MI-GRBL-Z-AXIS-Servo-Controller> (дата обращения: 22.03.2025).
2. **BACK TO BASIC MINI CNC PLOTTER** [Электронный ресурс] // Instructables. – URL: <https://www.instructables.com/BACK-TO-BASIC-MINI-CNC-PLOTTER/> (дата обращения: 22.03.2025).
3. **Хабр** [Электронный ресурс]. – URL: <https://habr.com/ru/articles/432818/> (дата обращения: 22.03.2025).
4. **Thingiverse** [Электронный ресурс]. – URL: <https://www.thingiverse.com/> (дата обращения: 22.03.2025).
5. **Подробное описание G- и M-кодов для программирования ЧПУ CNC станков** [Электронный ресурс] // shopstanki.ru. – URL: <https://shopstanki.ru/blog/podrobnoe-opisanie-g-i-m-kodov-dlya-programmirovaniya-chpu-cnc-stankov/> (дата обращения: 22.03.2025).
6. **Qt Documentation** [Электронный ресурс]. – URL: <https://doc.qt.io/> (дата обращения: 22.03.2025).
7. **PyInstaller** [Электронный ресурс]. – URL: <https://pyinstaller.org/en/stable/> (дата обращения: 22.03.2025).

8. Приложения

1. Исходный код:

```
from PyQt6 import QtWidgets, uic
from PyQt6.QtCore import QTimer
from PyQt6.QtWidgets import QFileDialog
import serial.tools.list_ports
import serial
import sys
import os

class PlotterControl:
    def __init__(self, ui):
        self.ui = ui
        self.serialInst = serial.Serial()

        self.CURRENT_X = 0.0
        self.CURRENT_Y = 0.0
        self.pen_is_down = False

        self.gcode_queue = []
        self.is_running = False
        self.is_paused = False

        self.setup_ui()
        self.setup_serial_monitor()

    def setup_ui(self):
        ports = serial.tools.list_ports.comports()
        self.ui.com_port_choose.clear()
        for port in ports:
            self.ui.com_port_choose.addItem(port.device)

        self.ui.home.clicked.connect(self.home_position)

        self.ui.ButtonUp.clicked.connect(self.move_forward_y)
        self.ui.ButtonDown.clicked.connect(self.move_back_y)
        self.ui.ButtonRight.clicked.connect(self.move_forward_x)
        self.ui.ButtonLeft.clicked.connect(self.move_back_x)

        self.ui.up_right.clicked.connect(self.move_up_right)
        self.ui.up_left.clicked.connect(self.move_up_left)
        self.ui.down_left.clicked.connect(self.move_down_left)
        self.ui.down_right.clicked.connect(self.move_down_right)

        self.ui.zero_x.clicked.connect(self.zero_x)
        self.ui.zero_y.clicked.connect(self.zero_y)

        self.ui.pen_control.clicked.connect(self.toggle_pen)

        self.ui.send_message.clicked.connect(self.send_console_input)
        self.ui.open_port.clicked.connect(self.open_serial)
        self.ui.close_port.clicked.connect(self.close_serial)

        self.ui.start_program.clicked.connect(self.start_gcode_execution)
        self.ui.pause_program.clicked.connect(self.pause_gcode_execution)
```

```

self.ui.stop_program.clicked.connect(self.stop_gcode_execution)

self.ui.actionChoose_file.triggered.connect(self.choose_file)

self.gcode_timer = QTimer()
self.gcode_timer.timeout.connect(self.send_next_gcode_line)

self.timer = QTimer()
self.timer.timeout.connect(self.read_serial_data)

self.response_timer = QTimer()
self.response_timer.timeout.connect(self.check_serial_response)
self.response_timer.start(1)

def open_serial(self):
    try:
        if self.serialInst.is_open:
            self.serialInst.close()

        port = self.ui.com_port_choose.currentText()
        if not port:
            return

        self.serialInst.port = port
        self.serialInst.baudrate = 115200
        self.serialInst.open()
        self.timer.start(100)
        self.ui.console_monitor.append(f"Подключено к {port}")

    except Exception as e:
        self.ui.console_monitor.append(f"Ошибка подключения: {str(e)}")
        self.serialInst.close()

def close_serial(self):
    try:
        if self.serialInst.is_open:
            self.serialInst.close()
            self.timer.stop()
            self.ui.console_monitor.append("Порт закрыт")
    except Exception as e:
        self.ui.console_monitor.append(f"Ошибка: {str(e)}")

def setup_serial_monitor(self):
    self.ui.step_size_xy.setRange(1, 1000)
    self.ui.feed_rate.setRange(1, 10000)

def send_console_input(self):
    text = self.ui.console_input.toPlainText().strip()
    if text and self.serialInst.is_open:
        self.send_command(text)
        self.ui.console_input.clear()
    elif not self.serialInst.is_open:
        self.ui.console_monitor.append("Ошибка: порт не открыт!")

def read_serial_data(self):
    if self.serialInst.is_open and self.serialInst.in_waiting:
        try:
            data = self.serialInst.readline().decode().strip()

```

```

        if data:
            self.ui.console_monitor.append(f"{data}")
        except UnicodeDecodeError:
            self.ui.console_monitor.append("Ошибка декодирования данных")

def get_step_size(self):
    return self.ui.step_size_xy.value()

def get_feed_rate(self):
    return self.ui.feed_rate.value()

def send_command(self, command):
    try:
        self.serialInst.write(f"{command}\n".encode())
        self.ui.console_monitor.append(f"{command}")
    except serial.SerialException:
        self.ui.console_monitor.append("Ошибка отправки команды")

def home_position(self):
    self.send_command("G0X0Y0")
    self.CURRENT_X = 0.0
    self.CURRENT_Y = 0.0
    self.ui.x_current_position.setText("0.00")
    self.ui.y_current_position.setText("0.00")

def move_forward_x(self):
    step = self.get_step_size()
    feed = self.get_feed_rate()
    self.send_command(f"G21G91G1X{step}F{feed}")
    self.CURRENT_X += step
    self.ui.x_current_position.setText(f"{round(self.CURRENT_X, 2)}")

def move_back_x(self):
    step = self.get_step_size()
    feed = self.get_feed_rate()
    self.send_command(f"G21G91G1X-{step}F{feed}")
    self.CURRENT_X -= step
    self.ui.x_current_position.setText(f"{round(self.CURRENT_X, 2)}")

def move_forward_y(self):
    step = self.get_step_size()
    feed = self.get_feed_rate()
    self.send_command(f"G21G91G1Y{step}F{feed}")
    self.CURRENT_Y += step
    self.ui.y_current_position.setText(f"{round(self.CURRENT_Y, 2)}")

def move_back_y(self):
    step = self.get_step_size()
    feed = self.get_feed_rate()
    self.send_command(f"G21G91G1Y-{step}F{feed}")
    self.CURRENT_Y -= step
    self.ui.y_current_position.setText(f"{round(self.CURRENT_Y, 2)}")

def move_up_right(self):
    step_x = self.get_step_size()
    step_y = self.get_step_size()
    feed = self.get_feed_rate()
    self.send_command(f"G21G91G1X{step_x}Y{step_y}F{feed}")
    self.CURRENT_X += step_x

```

```

        self.CURRENT_Y += step_y
        self.update_position()

    def move_up_left(self):
        step_x = -self.get_step_size()
        step_y = self.get_step_size()
        feed = self.get_feed_rate()
        self.send_command(f"G21G91G1X{step_x}Y{step_y}F{feed}")
        self.CURRENT_X += step_x
        self.CURRENT_Y += step_y
        self.update_position()

    def move_down_left(self):
        step_x = -self.get_step_size()
        step_y = -self.get_step_size()
        feed = self.get_feed_rate()
        self.send_command(f"G21G91G1X{step_x}Y{step_y}F{feed}")
        self.CURRENT_X += step_x
        self.CURRENT_Y += step_y
        self.update_position()

    def move_down_right(self):
        step_x = self.get_step_size()
        step_y = -self.get_step_size()
        feed = self.get_feed_rate()
        self.send_command(f"G21G91G1X{step_x}Y{step_y}F{feed}")
        self.CURRENT_X += step_x
        self.CURRENT_Y += step_y
        self.update_position()

    def update_position(self):
        self.ui.x_current_position.setText(f"{round(self.CURRENT_X, 2)}")
        self.ui.y_current_position.setText(f"{round(self.CURRENT_Y, 2)}")

    def zero_x(self):
        self.CURRENT_X = 0.0
        self.ui.x_current_position.setText("0.00")
        self.send_command("G92 X0")

    def zero_y(self):
        self.CURRENT_Y = 0.0
        self.ui.y_current_position.setText("0.0")
        self.send_command("G92 Y0")

    def toggle_pen(self):
        if not self.serialInst.is_open:
            self.ui.console_monitor.append("Ошибка: порт не открыт!")
            return

        try:
            if self.pen_is_down:
                self.send_command("M5")
                self.ui.pen_state.setText("UP")
            else:
                self.send_command("M3 S90")
                self.ui.pen_state.setText("DOWN")

            self.pen_is_down = not self.pen_is_down

```

```

        except Exception as e:
            self.ui.console_monitor.append(f"Ошибка управления пером: {str(e)}")

    def check_serial_response(self):
        if self.serialInst.is_open and self.serialInst.in_waiting:
            try:
                data = self.serialInst.readline().decode().strip()
                if "ok" in data.lower() and self.gcode_queue:
                    self.send_next_gcode_line()
                    self.ui.console_monitor.append(f"{data}")
            except Exception as e:
                self.ui.console_monitor.append(f"Ошибка чтения: {str(e)}")

    def start_gcode_execution(self):
        if not self.serialInst.is_open:
            self.ui.console_monitor.append("Ошибка: порт не открыт!")
            return

        if not self.is_running:
            self.gcode_queue =
self.ui.code_from_file.toPlainText().splitlines()
            self.is_running = True
            self.is_paused = False
            self.ui.console_monitor.append("Старт выполнения G-кода")
            self.send_next_gcode_line()
        elif self.is_paused:
            self.is_paused = False
            self.ui.console_monitor.append("Продолжение выполнения")
            self.send_next_gcode_line()

    def pause_gcode_execution(self):
        if self.is_running and not self.is_paused:
            self.is_paused = True
            self.ui.console_monitor.append("Пауза")

    def stop_gcode_execution(self):
        self.gcode_queue.clear()
        self.is_running = False
        self.is_paused = False
        self.ui.console_monitor.append("Выполнение прервано")

    def send_next_gcode_line(self):
        if self.is_running and not self.is_paused and self.gcode_queue:
            line = self.gcode_queue.pop(0).strip()
            if line:
                self.send_command(line)
            else:
                self.send_next_gcode_line()

    def choose_file(self):
        file_name, _ = QFileDialog.getOpenFileName(None, "Open G-code File",
"", "G-code Files (*.gcode *.txt)")
        if file_name:
            try:
                with open(file_name, 'r') as file:
                    gcode = file.read()
                    self.ui.code_from_file.setPlainText(gcode)
            except Exception as e:

```

```

        self.ui.console_monitor.append(f"Ошибка чтения файла: {e}")

def main():
    app = QtWidgets.QApplication([])

    if getattr(sys, 'frozen', False):
        base_dir = sys._MEIPASS
    else:
        base_dir = os.path.dirname(__file__)

    ui_path = os.path.join(base_dir, 'Plotter.ui')
    ui = uic.loadUi(ui_path)

    ui.setWindowTitle("Plotter Controller")
    controller = PlotterControl(ui)
    ui.show()
    app.exec()

if __name__ == "__main__":
    main()

```

2. Макет интерфейса:

