Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

Проект «Пневматический дозатор»

по дисциплине «Объектно-ориентированное программирование»

Студент Иванов И. В.

Преподаватель Анасьевский М. С.

«___» _____ 2023

Санкт-Петербург
2023 г.

Содержание

1. Цель проекта

Основной целью данного проекта является разработка пневматического дозатора, обладающего функцией вакуумного пинцета, а также обеспечивающего возможность цифрового управления, настройки параметров и вывода информации на интегрированный дисплей.

2. Реализация проекта

Первым этапом проекта было проектирование пневматической схемы дозатора в программе Festo Fluidsim, затем была разработана электрическая схема, корпус был спроектирован в программе Компас 360, в конце были заказаны все компоненты, написана и протестирована программа и произведена сборка прототипа.

Пневматическая схема состоит из пневмораспределителя, пневмоклапана, пневморедуктора, эжектора, различных фитингов и пневмотрубок

Электрическая схема прибора представлена на рисунке 1.

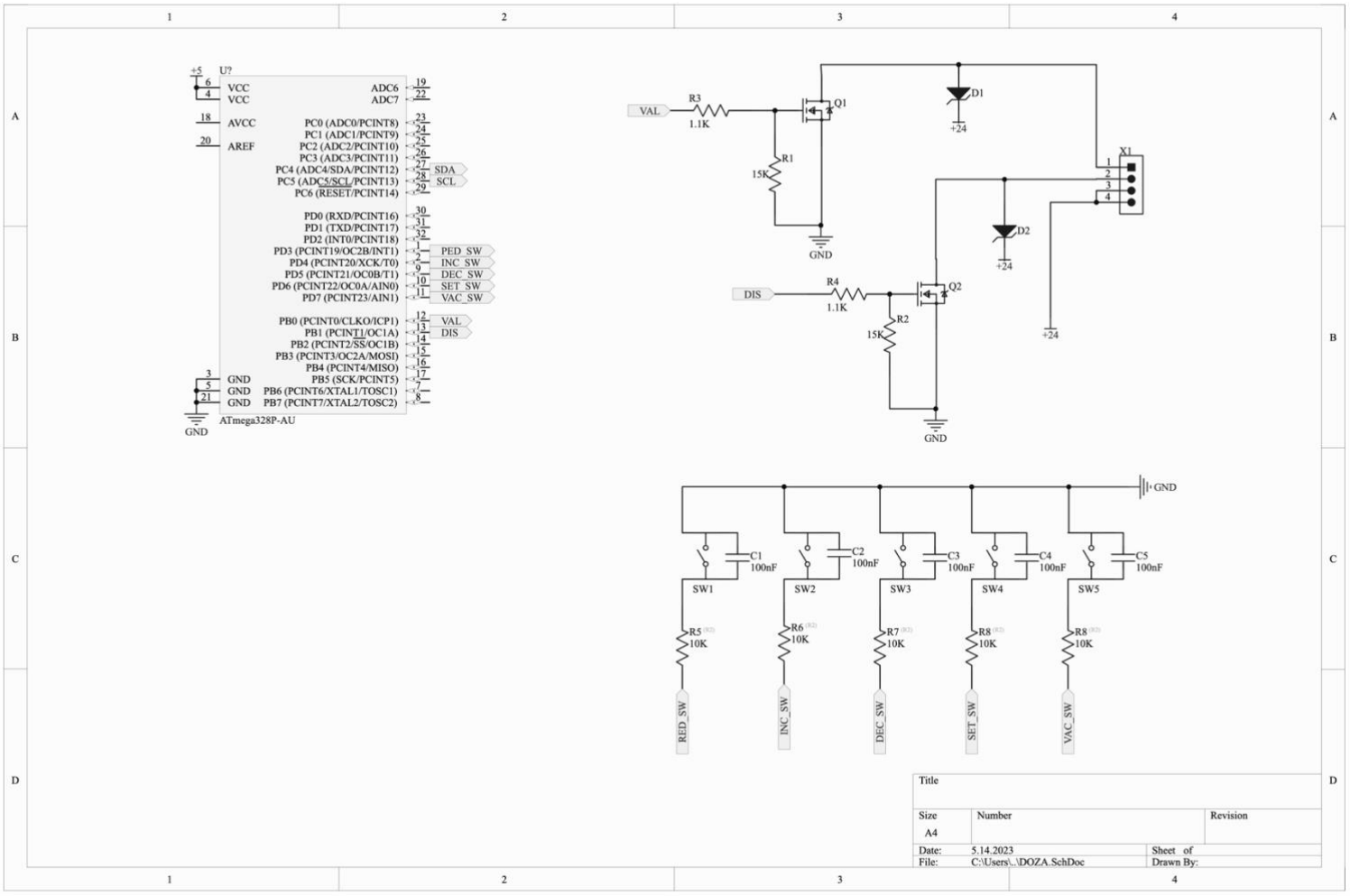


Рисунок 1 – Схема дозатора

Внешний вид корпуса представлен на рисунках 2 и 3.

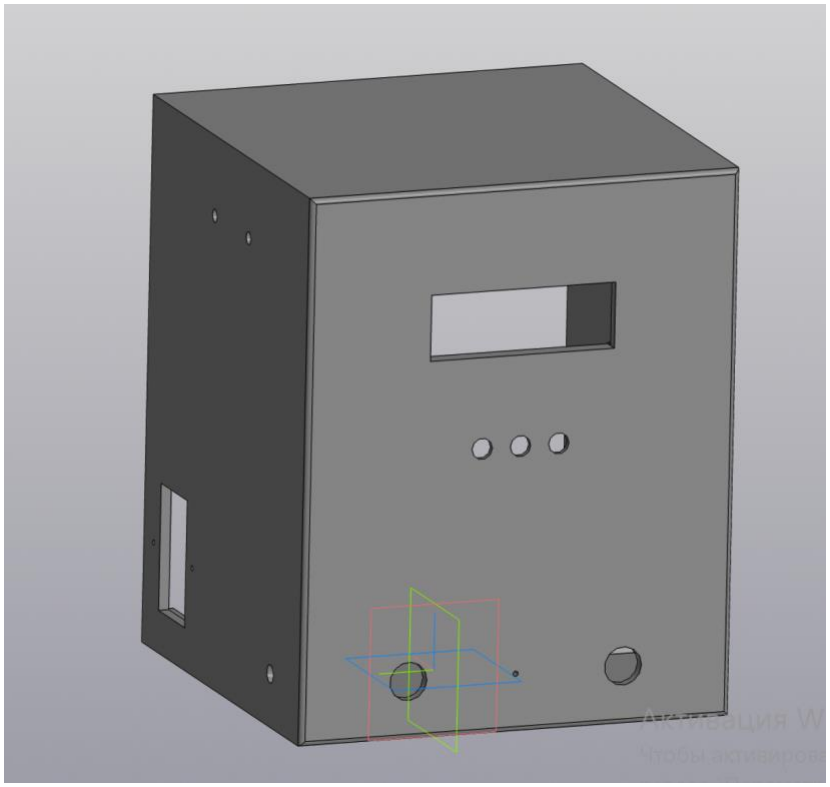


Рисунок 2 – вид корпуса спереди

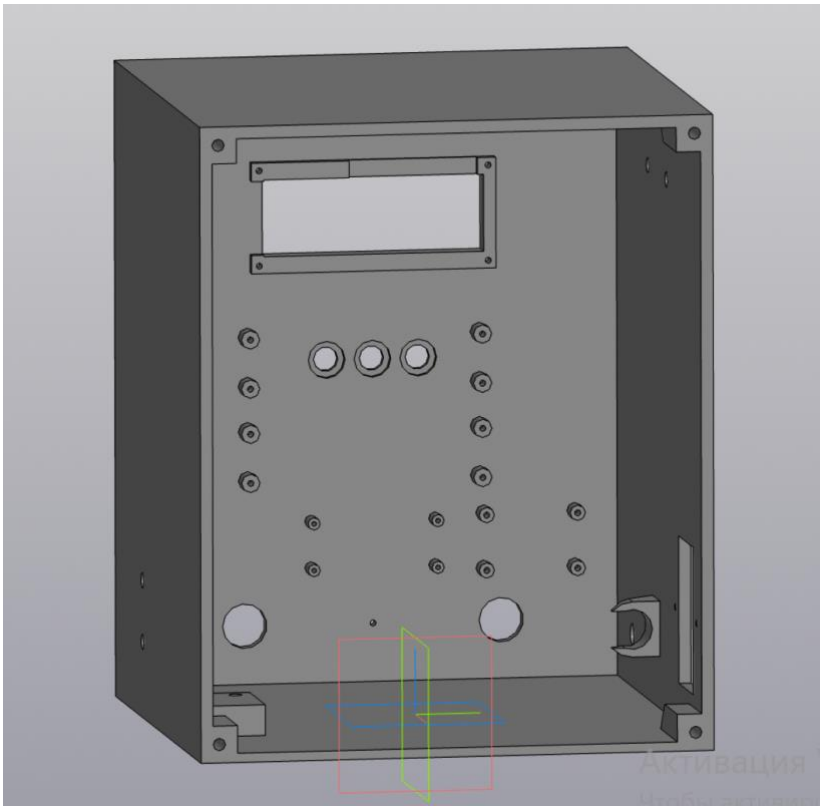


Рисунок 3 – вид корпуса сзади

3. Сборка проекта

Рисунок 4 – Сборка дозатора

Начальным этапом процесса сборки проекта являлась распайка всех элементов на макетных платах. Всего использовалось три платы: плата с кнопками, силовая плата с мосфет-транзисторами и плата с микроконтроллером, конкретно Arduino Nano, оснащенным процессором ATmega328P. Параллельно с этим была собрана и проверена работоспособность пневматической части дозатора. После этого произошло объединение компонентов на столе, где также была проведена проверка функциональности каждого элемента. Наконец, все собранное было помещено в заранее распечатанный корпус, и опять же была выполнена проверка его работоспособности.

4. Код программы

```
#include <LiquidCrystal_I2C.h>
#include <GyverButton.h>
GButton pedal(3, HIGH_PULL, NORM_OPEN);
GButton incr(4, HIGH_PULL, NORM_OPEN);
GButton decr(5, HIGH_PULL, NORM_OPEN);
GButton setting(6, HIGH_PULL, NORM_OPEN);
GButton vacswitch(7, HIGH_PULL, NORM_OPEN);
LiquidCrystal_I2C lcd(0x27, 16, 2);
uint32_t tmr1;          // переменная таймера
int podacha = 300;
int pausa = 300;
boolean setflag, stopflag, podflag;
void setup() {

  pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);

  pedal.setTickMode(AUTO);
  incr.setTickMode(AUTO);
   decr.setTickMode(AUTO);
    setting.setTickMode(AUTO);
      vacswitch.setTickMode(AUTO);

  Serial.begin(9600);
  // put your setup code here, to run once:
  lcd.init();           // инициализация
  lcd.backlight();      // включить подсветку
}

void loop() {
  // put your main code here, to run repeatedly:

  if (pedal.isHold()) {
    Serial.println("press");
    setflag = false;
    lcd.clear();
    lcd.print("press");
    digitalWrite(8, 1);
    Serial.println("ПОДАЧА");
    lcd.setCursor(6, 0);
```

```
   lcd.print("      ");
   lcd.setCursor(6, 0);
   lcd.print("SOLDER");
   delay(podacha);
   digitalWrite(8, 0);
   Serial.println("СТОП");
   lcd.setCursor(6, 0);
   lcd.print("      ");
   lcd.setCursor(6, 0);
   lcd.print("STOP");
   delay(pausa);
 }
   if (setting.isHolded()) {
Serial.println("режим настройки");
setflag = true;
podflag = true;
 lcd.clear();
 lcd.setCursor(13, 0);
 lcd.print("SET");
 lcd.setCursor(0, 0);
 lcd.print("AIR=");
 lcd.setCursor(0, 1);
 lcd.print("STOP=");
 lcd.setCursor(4, 0);
 lcd.print(podacha);
 lcd.setCursor(8, 0);
 lcd.print("mS");
 lcd.setCursor(5, 1);
 lcd.print(pausa);
 lcd.setCursor(9, 1);
 lcd.print("mS");
lcd.setCursor(3, 0);
lcd.blink();}
if (incr.isPress()&& setflag && podflag) {
 podacha = podacha + 10;
 lcd.setCursor(4, 0);
  lcd.print("    ");
  lcd.setCursor(4, 0);
 lcd.print(podacha);
  Serial.println(podacha);
  setflag = true;
```

```
 podflag = true;
   }
   if (decr.isPress()&& setflag && podflag) {
  podacha = podacha - 10;
  lcd.setCursor(4, 0);
   lcd.print("    ");
   lcd.setCursor(4, 0);
  lcd.print(podacha);
  setflag = true;
 podflag = true;

 Serial.println(podacha);
   }
  if (setting.isPress()&& setflag) {
  lcd.setCursor(4, 1);
 stopflag = true;
  podflag = false;
   }
if (incr.isPress()&& setflag && stopflag) {
  pausa =  pausa + 10;
  lcd.setCursor(5, 1);
   lcd.print("    ");
   lcd.setCursor(5, 1);
  lcd.print(pausa);
   Serial.println(pausa);
    stopflag = true;
  podflag = false;
}
if (decr.isPress()&& setflag && stopflag) {
  pausa =  pausa - 10;
  lcd.setCursor(5, 1);
   lcd.print("    ");
   lcd.setCursor(5, 1);
  lcd.print(pausa);
   Serial.println(pausa);
    stopflag = true;
  podflag = false;
}
if (vacswitch.state()&& pedal.state()) {
    lcd.clear();
    lcd.print("vac");
```

```
  while (pedal.state()){
    digitalWrite(9, 1);
  }
  digitalWrite(9, 0);
}
}
```

В данном случае, код должен обрабатывать нажатия кнопок и управлять дисплеем и выводами Arduino на основе этих действий.

1. Подготовка среды разработки.

Чтобы начать разработку, я установил необходимые библиотеки LiquidCrystal_I2C и GyverButton. Затем я создал новый проект в среде разработки Arduino IDE и подключил необходимые библиотеки к проекту.

2. Определение подключений и инициализация объектов.

Для работы с кнопками и дисплеем, я определил необходимые пины ввода/вывода на Arduino. Затем я создал объекты классов GButton и LiquidCrystal_I2C, которые позволяют удобно работать с кнопками и дисплеем соответственно. Также были объявлены переменные и флаги, которые будут использоваться для управления логикой программы.

3. Настройка и инициализация компонентов:

Чтобы корректно работать с кнопками и дисплеем, я настроил режимы работы кнопок и инициализировал дисплей. Настройка кнопок включает выбор типа подключения (например, HIGH_PULL) и тип срабатывания (например, NORM_OPEN).

4. Основной цикл программы

Основной код размещен в функции loop(), которая выполняется в бесконечном цикле. В этой функции происходит обработка действий пользователя и соответствующая реакция Arduino.

5. Разработка логики обработки нажатий кнопок:

Я разработал логику обработки нажатий кнопок с использованием методов предоставляемых библиотекой GyverButton. Для каждой кнопки я определил условия и выполнил соответствующие действия при их

срабатывании. Например, при удержании кнопки "pedal" выводится сообщение на дисплей, устанавливается флаг и происходит задержка.

6. Реализация режима настройки.

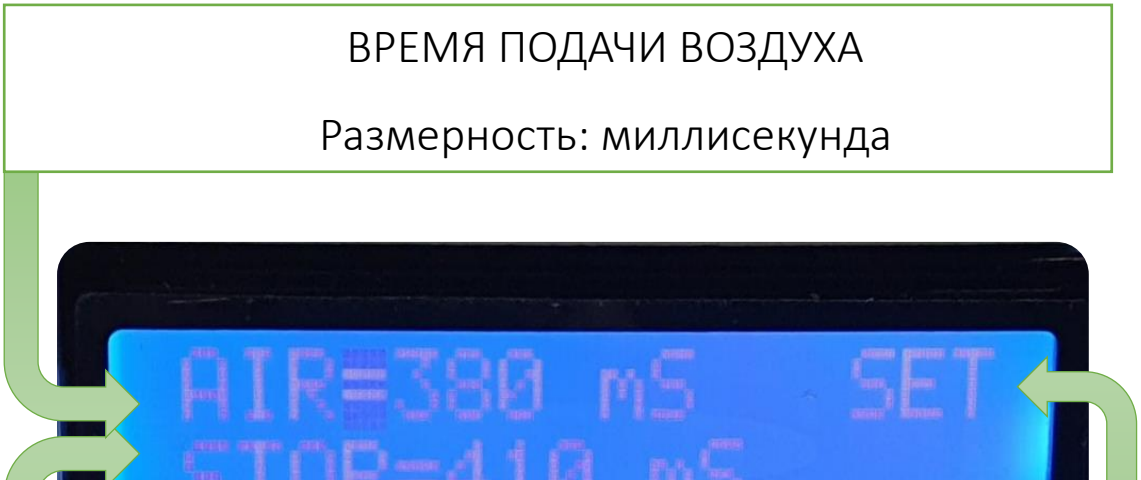
При удержании кнопки "setting" активируется режим настройки. В этом режиме пользователь может изменять значения переменных podacha и pausa с помощью кнопок "incr" и "decr". Я добавил соответствующие условия и логику для изменения значений и отображения их на дисплее. При нажатии кнопки "setting" в режиме настройки "stop", пользователь может изменять значение переменной pausa, которая отвечает за паузу.

7. Обработка состояния переключателя "vacswitch".

Если переключатель "vacswitch" активирован и кнопка "pedal" удерживается, то Arduino устанавливает соответствующий флаг и включает выходной пин для вакуумного насоса. При отпускании кнопки "pedal" выходной пин выключается.

В результате выполнения всех вышеописанных шагов я успешно реализовал требуемую функциональность. Код позволяет обрабатывать нажатия кнопок, управлять дисплеем и включать/выключать выходные пины на основе действий пользователя.

5. Схема интерфейса настроек дозатора



6. Полученные результаты

В конце проекта был создан полностью функциональный прототип дозатора, при невысокой цене он обладает всеми функциями заводского исполнения, превосходящего его в несколько раз по стоимости.

7. Использованная литература

1. Джереми Блюм – Изучаем Arduino. Инструменты и методы технического волшебства.

2. AlexGyver – Шпаргалка по функциям Arduino (составлено по курсу видео-уроков).