Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Томский политехнический университет»

Школа / филиал	Инженерная школа неразрушающего контроля и безопасности
Обеспечивающее подразделение	Отделение электронной инженерии
Направление подготовки	12.03.04 Биотехнические системы и технологии
Образовательная программа	Биомедицинская инженерия
Специализация	Биомедицинская инженерия

ОТЧЕТ О ПРАКТИКЕ

Вид практики	Производственная
Тип практики	Производственно-технологическая
Место практики	г. Томск, ООО «Л. М. Э. «Биоток»»

Выполнил обучающийся	Нгуен Нгок Фу
Группа	1Д01

_____
(подпись обучающегося)

Согласовано:
Руководитель практики от Организации

_____ _____ _____
(должность) (подпись) (Ф. И. О.)

«___»_____ 2023 г.

Руководитель практики ТПУ:
_____к.т.н., доцент ОЭИ_____ _____Дикман Е.Ю._____
(степень, звание, должность) (Ф. И. О.)

Дата проверки ____ _____ 2023 г.

Допустить / не допустить к защите
Подпись _____

Итоговая оценка по практике _____
(традиционная оценка, балл)

Томск 2023

УТВЕРЖДАЮ

Руководитель ООП
Е.Ю. Дикман
_____
«___»____20____г.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на производственно-технологическую практику

Тема задания на производственно-технологическую практику:

Разработка системы управления скоростью движения передвижного компьютерного
томографа

Перечень работ (заданий), подлежащих выполнению:

1. Провести сравнительный анализ шаговых и постоянных двигателей
2. Разработать структурную и принципиальную электрическую схему устройства
3. Разработать алгоритм работы системы управления и код программы

Перечень отчетных материалов и требования к их оформлению:[2]

1. Отчет по производственно-технологической практике (СТО ТПУ 2.5.01-2011)
2. Презентация

Согласовано:
Руководитель практики от организации

_____ _____ _____
(должность) (подпись) (Ф. И. О.)

М.П. «___»_____ 2023 г.

Задание принял к исполнению _____ Нгуен.Н.Ф.
(подпись) (Ф.И.О.)

«___»_____20____г.

Содержание

Индивидуальное задание на практику 4

Работы 4

Результат работы 5

1. Датчик давления RP-L 5
2. ПИД-регулятор 7
3. Мотор с редуктором и энкодером JGB37-520B 12B 8
4. Драйвер DRV8833 10
5. Общая схема системы 13
6. Разработка алгоритма и кода работы системы 13
   - 6.1. Снятие напряжение из датчика схемы 14
   - 6.2. Определение цель скорости по напряжению 14
   - 6.3. Определение реальной скорости 15
   - 6.4. ПИД-регулятор 16
   - 6.5. Подача ШИМ 16
   - 6.6. Результат кода. 17

Список литературы 21

Индивидуальное задание на практику

Разработать систему для регулировки скорости двигателя на основе силы, действующей на датчик давления.

Исходные данные к работе:

- Датчик: Датчик давления RP-L
- Двигатель: Мотор с редуктором и энкодером JGB37-520B 12B
- Микроконтроллер: STM32F103C8T6
- Дривер: DRV8833

Работы

Первая неделя

- Измерьте фактические параметры датчика, чтобы разработать схему делителя напряжения, соответствующую входным параметрам АЦП STM32F103C8T6.
- Узнать об алгоритме ПИД-регулятор
- Узнайте о том, как работает мотор с редуктором и энкодером и дривир DRV8833.
- Общая схема системы.

Вторая неделя

- Завершить общую схему для система (Рисунок 14).
- PCB схема для драйвера DRV8833 (Рисунок 15).

Третья наделя

- Построение алгоритма(Рисунок 16).
- Создание кода для системы.

Четвертая неделя

- Завершить код для система.

Результат работы

1. Датчик давления RP-L

Рисунок 1 – Процесс измерения.

Таблица 1. Релультат измерения

Вес(г)	68.75	137.5	206.25	275	343.75	412.5	481.25	550	650	1200	>1200
кОм	420	90	58	43	38	32	30	28	22	19	4
		86	56	40	33	29	23.5	21.5	18.6	14	
	375	97	82	70	58	41	28.5	23	17	10	
		100	58	39	27	22	18	15	14	11.5	
		98	62	43	33	25	22	21	18	10.5	
З.ср	397.5	94.2	63.2	47	37.8	29.8	24.4	21.7	17.92	13	4

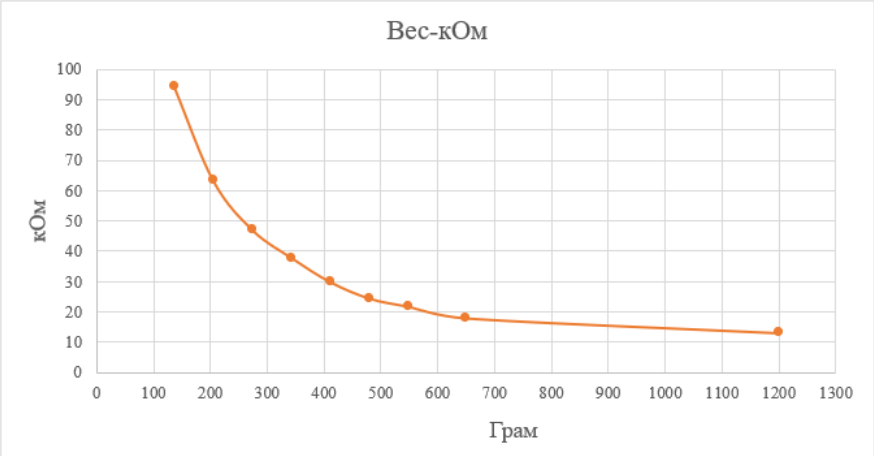

Рисунок 2 – Вес-кОм характеристик датчика.

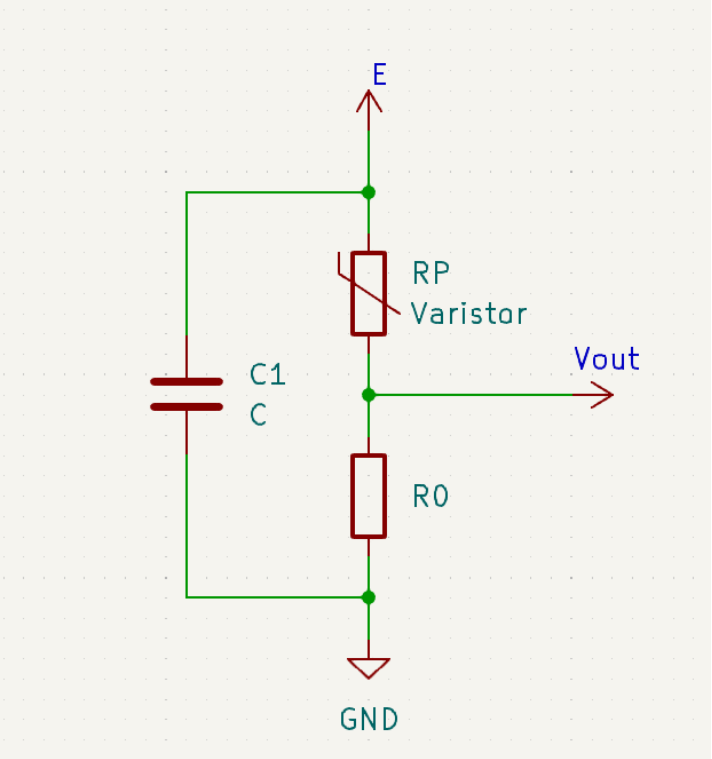

Рисунок 3 – Цепь для датчик с микроконтроллером.

$$RP := \begin{pmatrix} 397.5 \\ 94.2 \\ 63.2 \\ 47 \\ 37.8 \\ 29.8 \\ 24.4 \\ 21.7 \\ 17.92 \\ 13 \\ 4 \end{pmatrix}$$

$E := 3.3 \quad R0 := 1$

$$Vout(RP) := \frac{R0}{RP + R0} \cdot E$$

$Vout(RP) =$

	0
0	$8.281 \cdot 10^{-3}$
1	0.035
2	0.051
3	0.069
4	0.085
5	0.107
6	0.13
7	0.145
8	0.174
9	0.236
10	0.66

Рисунок 4 – Анализ Vout цепи датчика.

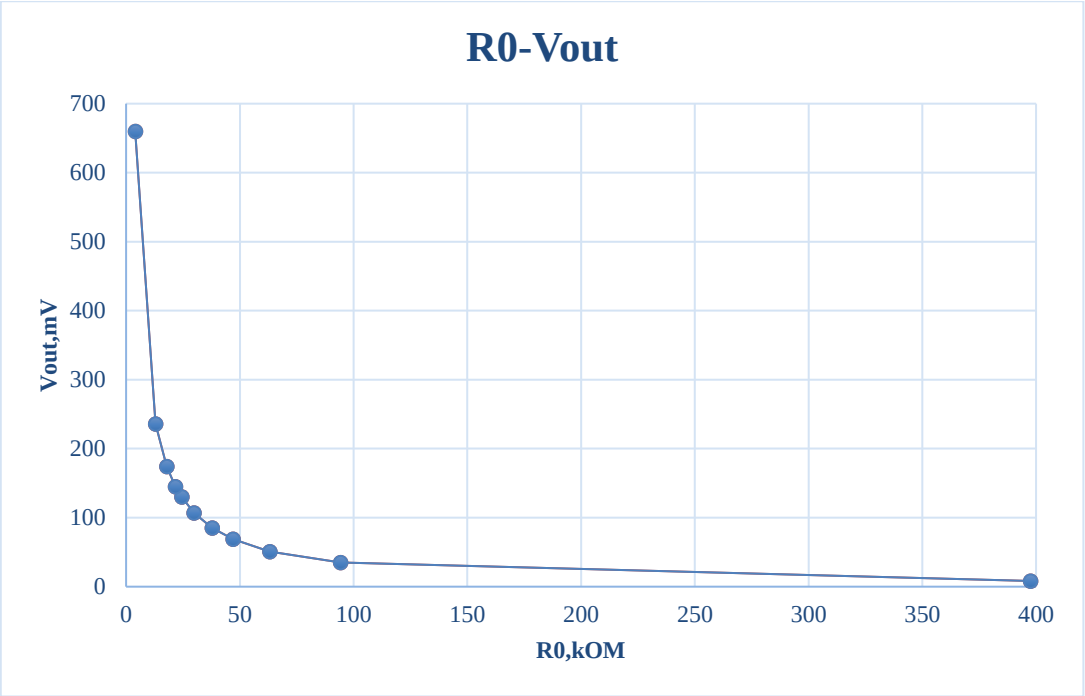

Рисунок 5 – Зависимость R0 от Vвых.

Вывод: Применив к цепи датчика источника E = 3,3 и резистор R0 = 1к, получим наприжение выход Vout на рисунке 4. Шкала результатов от схемы датчиков составляет мВ. Как мы знаем, шкала АЦП STM32 составляет 0,8 мВ. Поэтому эти результаты выше могут быть нормально проанализированы с помощью АЦП STM32.

2. ПИД-регулятор

Пропорционально-интегрально-дифференцирующий (ПИД) регулятор: устройство в управляющем контуре с обратной связью. Используется в системах автоматического управления для формирования управляющего сигнала с целью получения необходимых точности и качества переходного процесса. ПИД-регулятор формирует управляющий сигнал, являющийся суммой трёх слагаемых, первое из которых пропорционально разности входного сигнала и сигнала обратной связи (сигнал рассогласования), второе — интегралу сигнала рассогласования, третье — производной сигнала рассогласования.

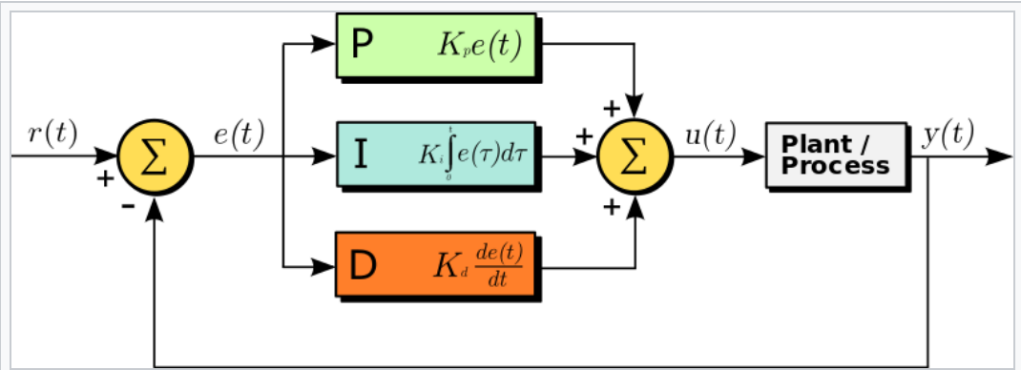

Рисунок 6 – Блок-схема алгоритма PID.

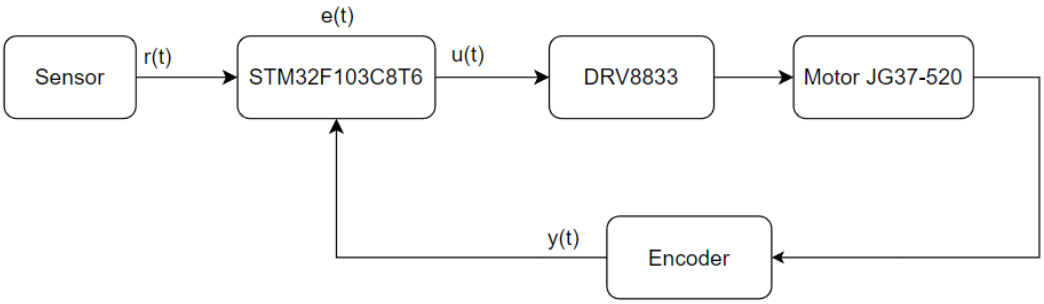

Рисунок 7 – Реальное приложение.

В нашем случае y(t) — это обратная связь по скорости от энкодера. Это реальная скорость. В соответствии со значением датчика мы можем получить целевую скорость - r(t). Тогда мы можем узнать ошибку e(t) = r(t) - y(t). Применяя алгоритм PID, мы можем вычислить управление выходом - u (t), чтобы управлять двигателем.

3. Мотор с редуктором и энкодером JGB37-520B 12B

Энкодер работает, наблюдая за изменениями магнитного поля, создаваемого магнитом, прикрепленным к валу двигателя, когда двигатель вращается, выходы энкодера периодически срабатывают. При вращении магнита по часовой стрелке первым сработает выход «а», а при вращении против часовой стрелки, с другой стороны, сработает выход «б». Таким образом, вы точно знаете, в какую сторону вращается вал двигателя. Это может быть очень удобно в ситуациях, когда вам нужно контролировать прямое и обратное движение DC.

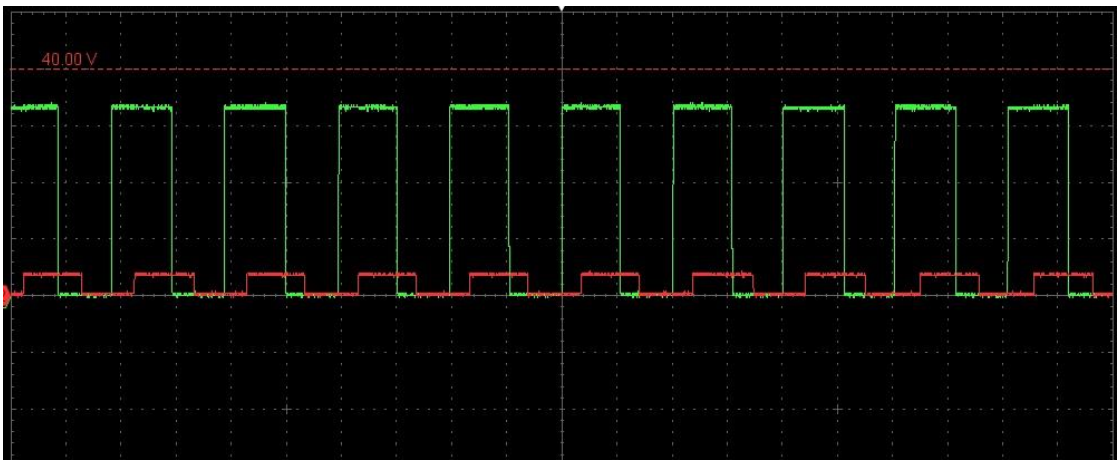

Рисунок 8

Сигнал энкодера «а» (зеленый), «б» (красный) по часовой стрелке.

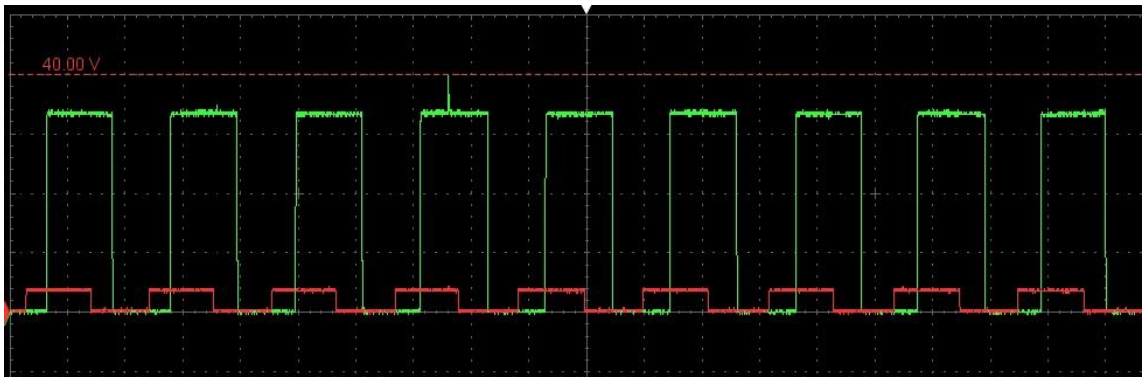

Рисунок 9

Сигнал энкодера «а» (зеленый), «б» (красный) против часовой стрелки.

Рисунок 10 – Распиновка модели.

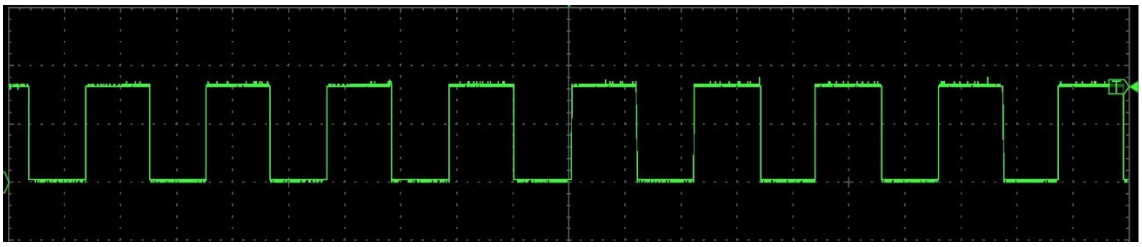

Рисунок 11 – Сигнал «a» по скоростью мотора 930rps.

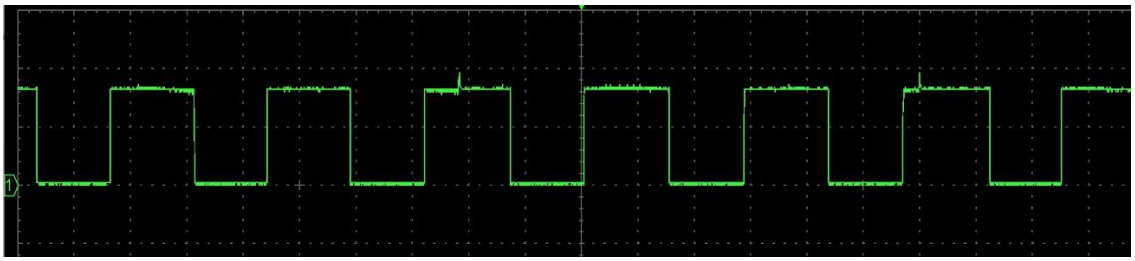

Рисунок 12 – Сигнал «a» по скоростью мотора 730rps.

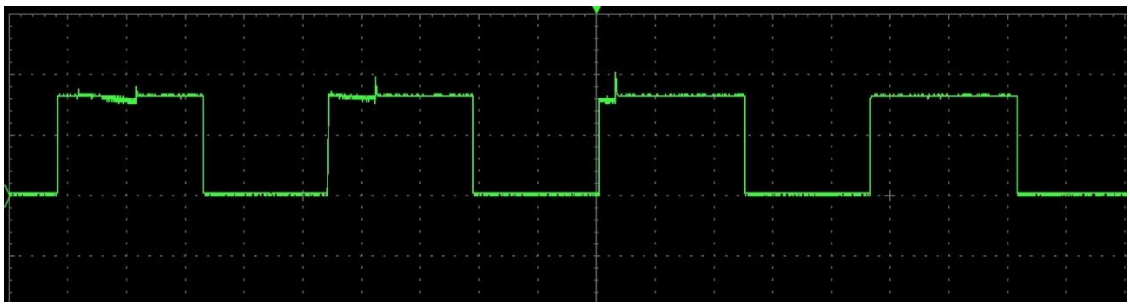

Рисунок 13 – Сигнал «a» по скоростью мотора 300rps.

Выход: Обнаружив нарастающий фронт импульсов, мы можем узнать скорость вала двигателя. Из опыта можно отметить, что чем быстрее вал двигателя, тем короче период сигнала.

4. Драйвер DRV8833

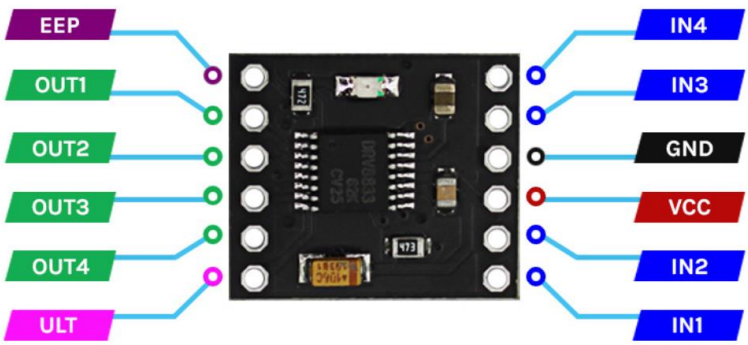

Рисунок 14 – Распиновка модели.

- **VCC** - Вывод подачи питания моторов и драйвера от 2,7 до 10,8 В.
- **GND** - Общий вывод питания.

  Вывод GND драйвера должен быть соединён с выводом GND Arduino!
- **IN1**, **IN2** - Входы управления первым мотором.
- **IN3**, **IN4** - Входы управления вторым мотором.
- **OUT1**, **OUT2** - Выходы подключения первого мотора.
- **OUT3**, **OUT4** - Выходы подключения второго мотора.
- **EEP** - Вход установки спящего режима, задаётся низким логическим уровнем.
- **ULT** - Выход аварийного сигнала. Низкий уровень сигнализирует о перегрузки по току, перегреву чипа, или низкому напряжению. В рабочем режиме вывод отключён (плавающий сигнал).

Управление:

Моторы подключённые к выводам OUT1, OUT2 и OUT3, OUT4 управляются подачей логических уровней на входы IN1, IN2 и IN3, IN4 соответственно. Обратите внимание на то, что выходы инвертируют сигналы своих входов.

Вход		Выход		Описание
IN1	IN2	OUT1	OUT2	
0	0	Z	Z	Отключение выводов мотора (свободное вращение ротора)
0	1	VCC	GND	Вращение в обратном направлении на максимальной скорости.
ШИМ	0	GND	VCC	Вращение в обратном направлении на максимальной скорости.
1	1	GND	GND	Торможение мотора (стопор ротора)

Управление мотором при помощи ШИМ

Вход		Выход		Описание
IN1	IN2	OUT1	OUT2	
ШИМ	1	~ШИМ	GND	Вращение в прямом направлении с линейной зависимостью скорости от инверсного ШИМ (чем выше ШИМ, тем ниже скорость).

1	ШИМ	GND	~ШИМ	Вращение в обратном направлении с линейной зависимостью скорости от инверсного ШИМ (чем выше ШИМ, тем ниже скорость).
ШИМ	0	GND/Z	VCC/Z	Не используйте данные сигналы на входах! Вращение осуществляется с нелинейной зависимостью скорости от инверсного ШИМ и падением крутящего момента.
0	ШИМ	VCC/Z	GND/Z	

Для управления большинством драйверов собранных по схеме H-моста, на один из входов драйвера подают ШИМ (для регулировки скорости мотора), а на второй вход подают логический уровень (для выбора направления вращения мотора). Такая схема управления не подходит для драйвера DRV8833, так как он позволяет переводить свои выходы в состояние высокого импеданса (Z), что приводит к отключению выводов мотора.

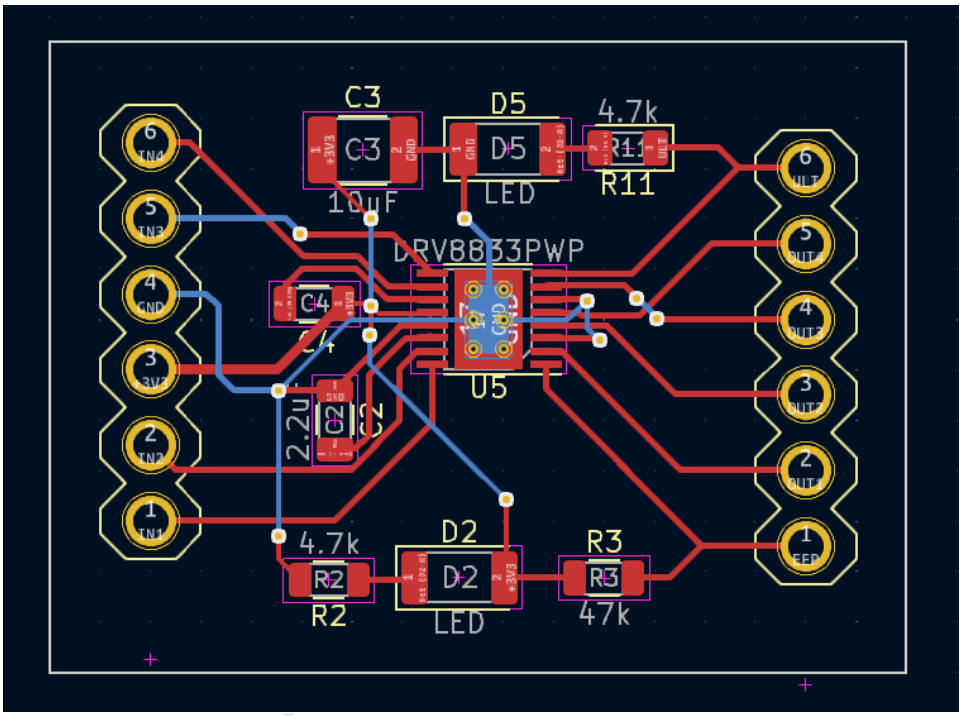

Рисунок 14 – PCB схема для драйвера DRV8833.

5. Общая схема системы

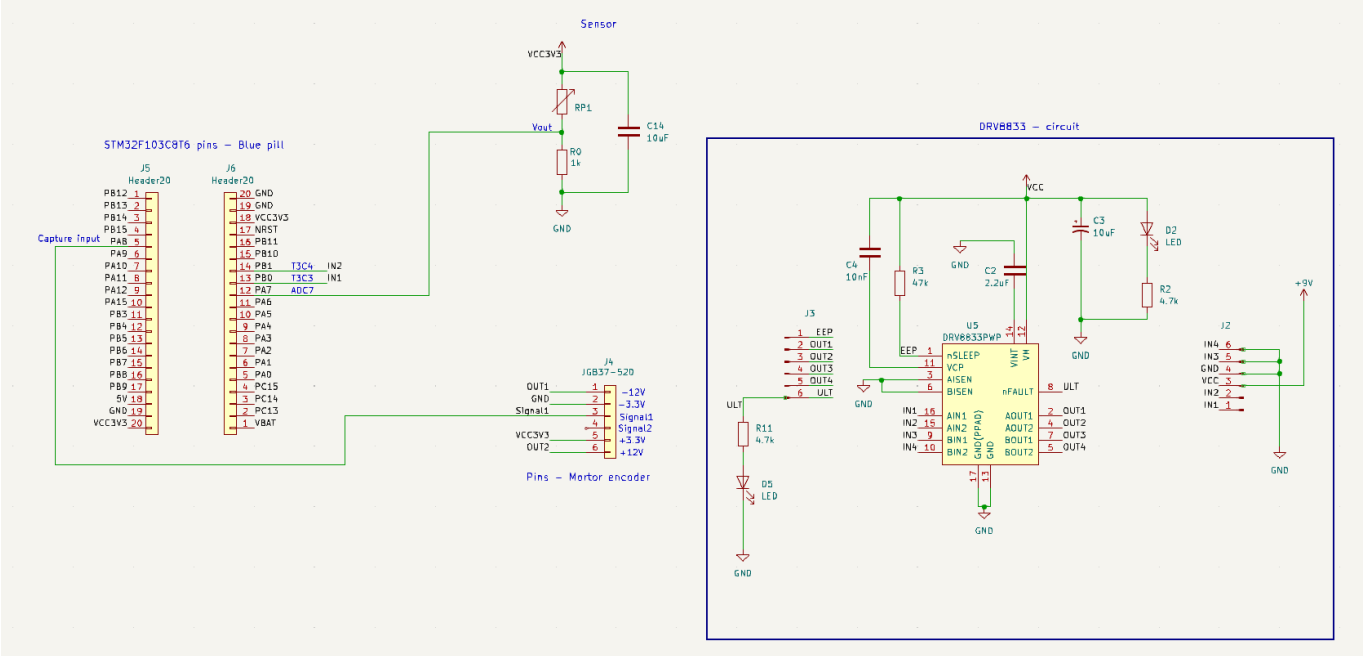

Рисунок 15 – Общая схема системы.

Вывод: на основе тех вещей, которые мы обнаружили выше, выстраивается общая схема. Это поможет нам более четко понять, как подключить и построить настоящую систему.

6. Разработка алгоритма и кода работы системы

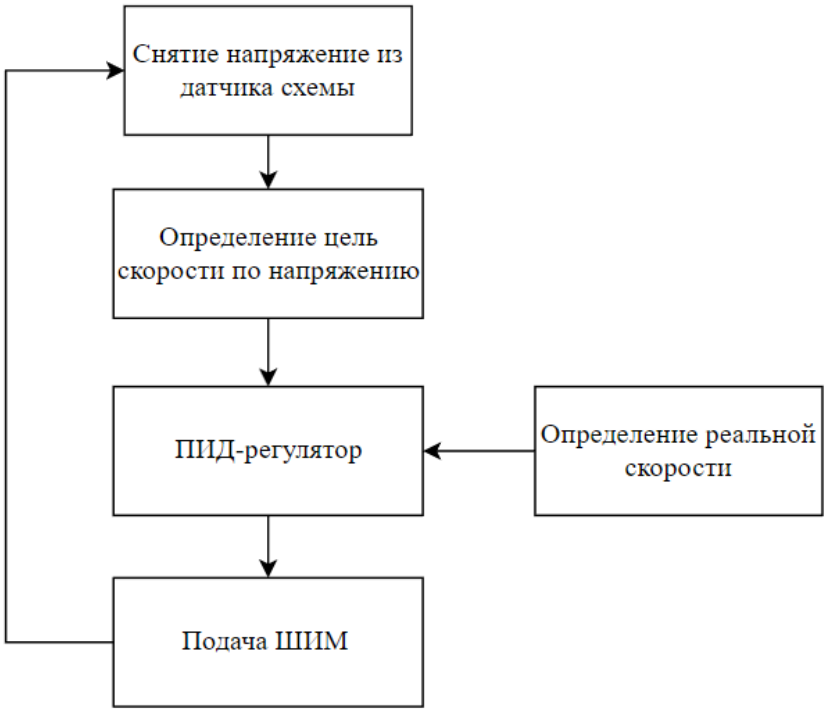

Рисунок 16 – Алгоритм работы программы.

6.1.Снятие напряжение из датчика схемы

```
pre_sensor = Sensor;
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1,1);
Sensor = HAL_ADC_GetValue(&hadc1);
sensor_ave = (pre_sensor + Sensor)/2;
Sensor_10 = sensor_ave/10;
```

Рисунок 17 – Код чтения АЦП.

Я использовал канал 7 АЦП1 для считывания напряжения со схемы датчика. Результат вышел с шумом, и я использовал среднее значение, чтобы уменьшить шум. Он представлен через значение sensor_ave(Рис.17).

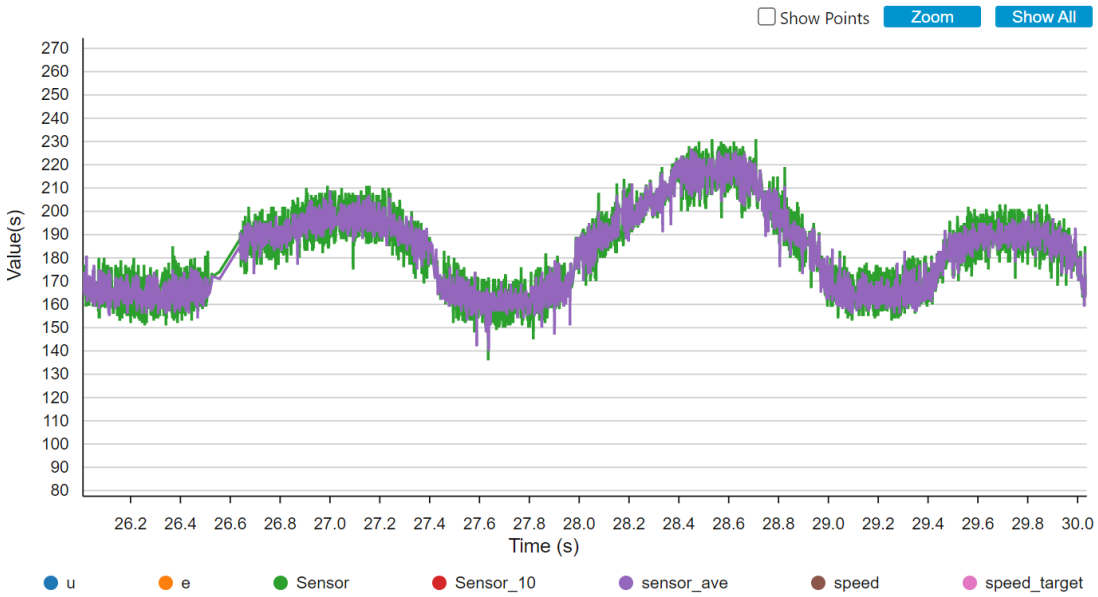

Рисунок 18 – Значение Sensor и sensor_ave

6.2.Определение цель скорости по напряжению

```
int f_speed_target(long a)
{
    int b;
    if ((a >=0 ) && (a < 511))
        {
           b = 310;
        }
        else if ((511 <= a) && (a < 1022))
        {
            b = 450;
        }
        else if ((1022 <= a) && (a < 1533))
        {
            b = 580;
        }
        else if ((1533 <= a) && (a < 2044))
        {
            b = 650;
        }
        else if ((2044 <= a) && (a < 2555))
        {
            b = 790;
        }
        else if ((2555 <= a) && (a < 3066))
        {
            b = 890;
        }
        else if ((3066 <= a) && (a < 3577))
        {
            b = 980;
        } else if (3577 <= a)
        {
            b = 1040;
        }
    return b;
}
```

Рисунок 19 – Код для определения цель скорости по напряжению.

Получаем диапазонное значение sensor_ave от 0 до 4095. Затем разбиваем его на дискретные, и соответственно каждому интервалу задаем целевую скорость. Это делается вышеприведенной функцией f_speed_target().

6.3. Определение реальной скорости

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM1)
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            period = HAL_TIM_ReadCapturedValue(&htim1, TIM_CHANNEL_1);
            speed = 100000/period;
            //pos++;
            TIM1->CCR1 = 0;
            TIM1->CNT =0;
        }
    }
}
```

Рисунок 20 – Код для определения реальной скорости

Используя канал 1 TIM1 для захвата входного сигнала, мы можем получить период, а затем вычислить скорость сигнала. Это означает, что мы получаем скорость вращения вала двигателя. TIM1 инициализируется с предделителем 719, а CLK для TIM1 составляет 72 МГц. Поэтому формула для скорости равна 100000/period. Единицей скорости является раунд в секунду (об/с).

6.4.ПИД-регулятор

```
e = speed_target - speed;
//de = (e - pre_e);
if ((u<=480) ||(e<0))
{
    e_integral = e_integral + 0.1*e;
}
//pre_e = e;
u = 0.19*speed_target + 280 + 0.001*e_integral;//+ 0.1*de;
```

Рисунок 21 – Код для ПИД-регулятор.

Как мы знаем, ПИД представляет собой комбинацию пропорциональной, интегральной и производной частей. В этом случае я просто использую пропорциональную и интегральную части (ПИ). Условие «u<=480 || e<0» направлено на то, чтобы избежать выхода за пределы контрольного значения u. Для пропорциональной части я не использую функцию «Kp*e», вместо нее я использую функцию «0,19*speed_target + 280». Это оптимизирует процесс и требует меньше времени для управления двигателем в соответствии с заданной скоростью.

6.5.Подача ШИМ

Канал 3 TIM3 используется для создания импульса PWM для управления двигателем. Этот шаг легко выполнить, используя код TIM3->CCR3 = u.

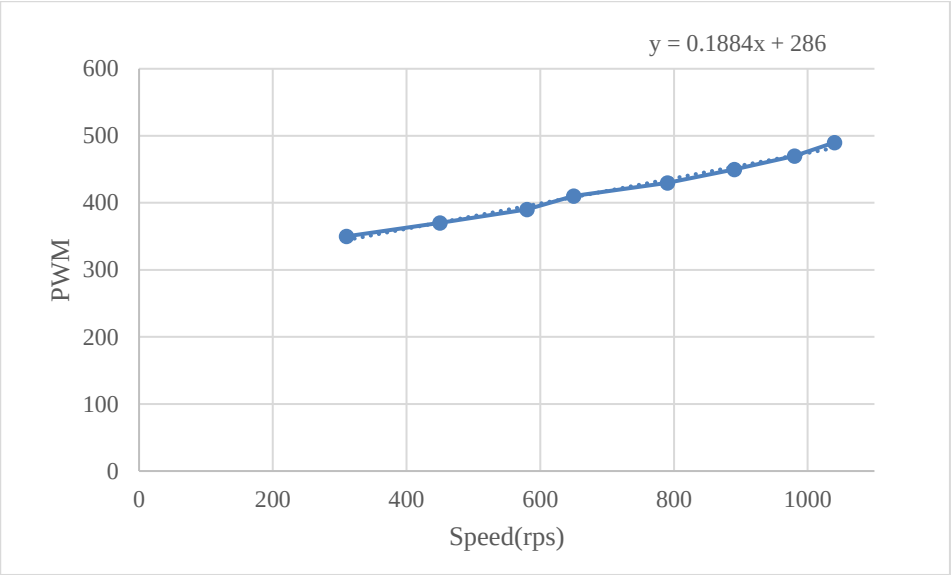

Рисунок 22 – Режмм работы мотора.

Исходя из характеристики TIM3, частота 50 кГц, ARR автоперезагрузки 719, мы получаем зависимость скорости двигателя от ШИМ, как указано выше(Рис.20).

6.6. Результат кода.

Примечание: «sensor_10» — это значение датчика со шкалой, разделенной на 10. «u» — это выходной контроль. "e" - это ошибка между текущей скоростью и заданной скоростью. «Speed» — это текущая скорость, а «speed_target» — целевая скорость.

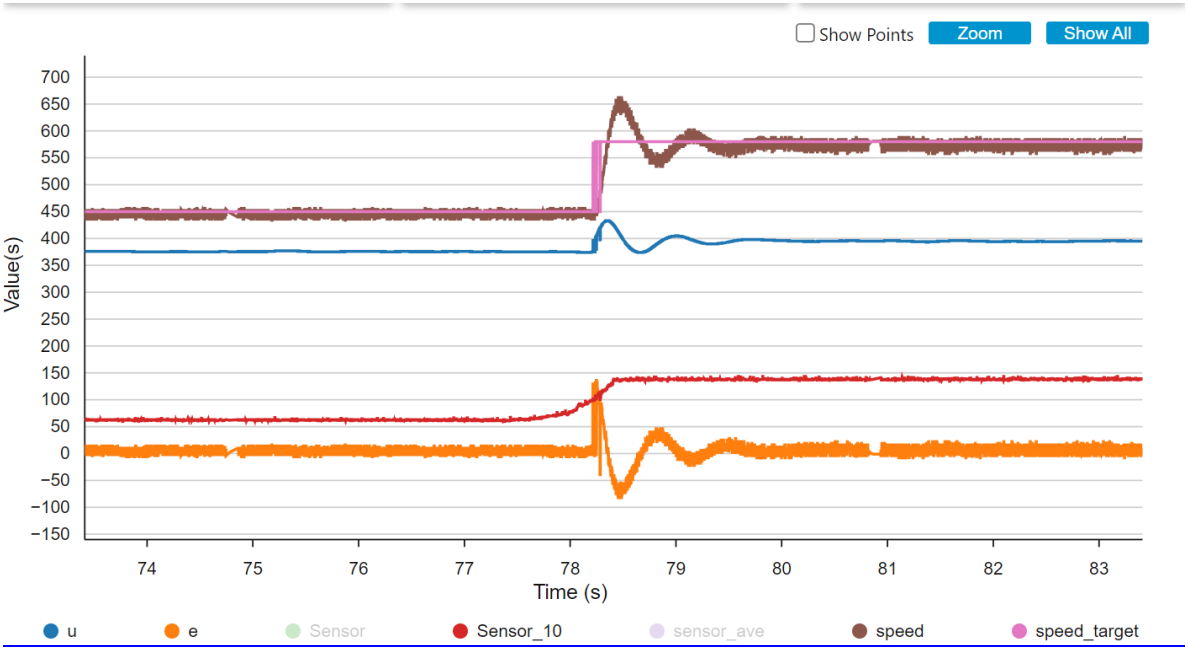

Рисунок 23– Поведение системы при повышении уровня значение от датчика.

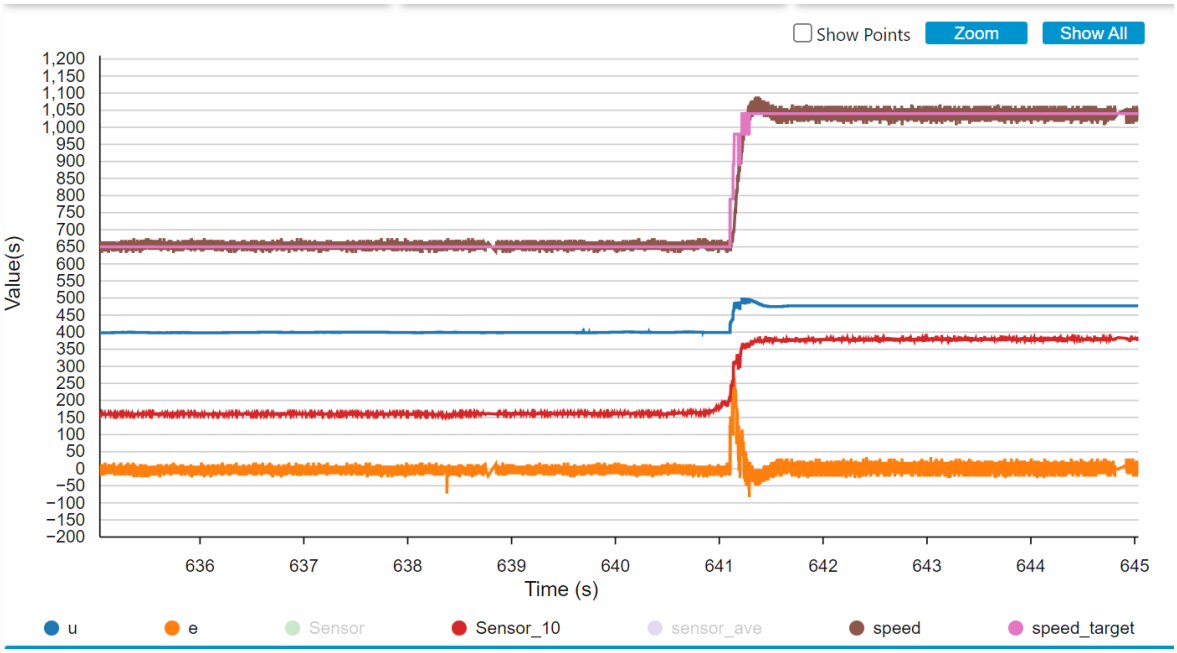

Рисунок 24– Поведение системы при повышении уровня значение от датчика.

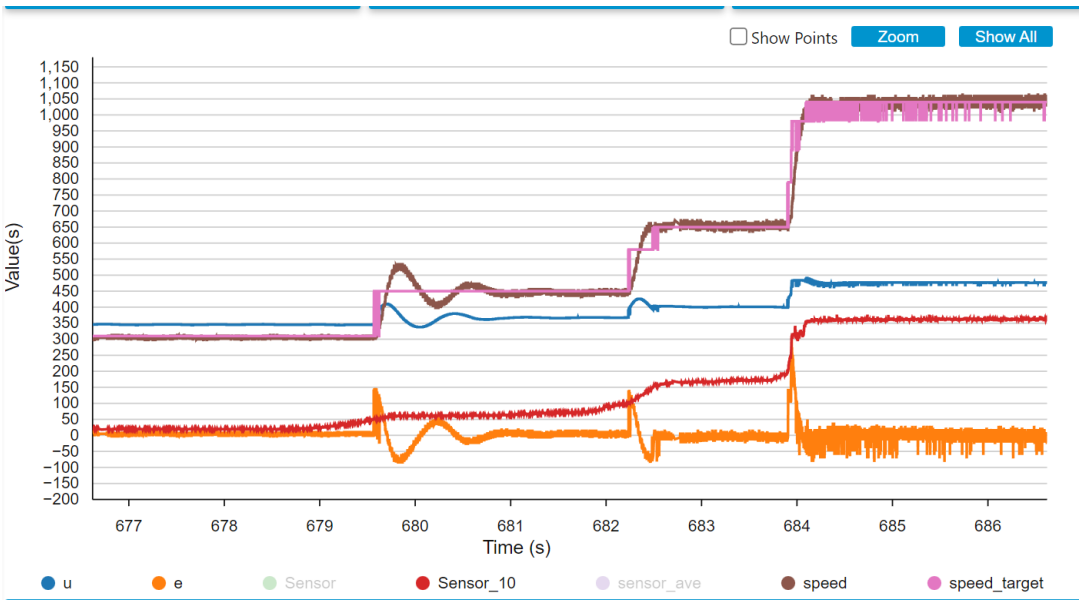

Рисунок 25– Поведение системы при повышении уровня значение от датчика.

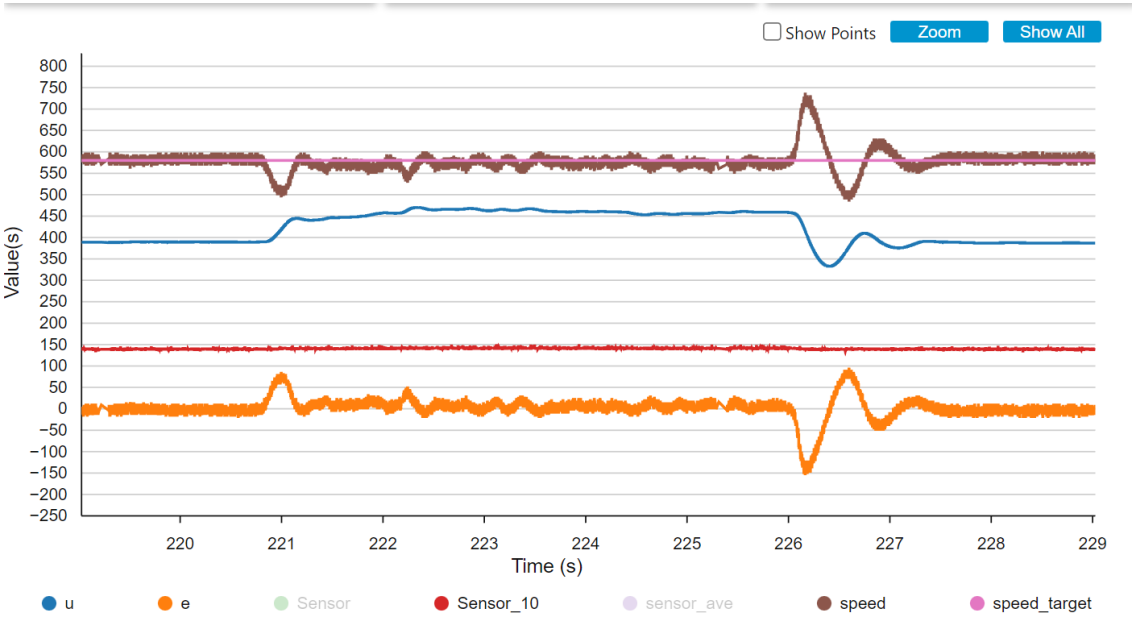

Рисунок 26 – Поведение системы при воздействии нагрузки на двигатель.

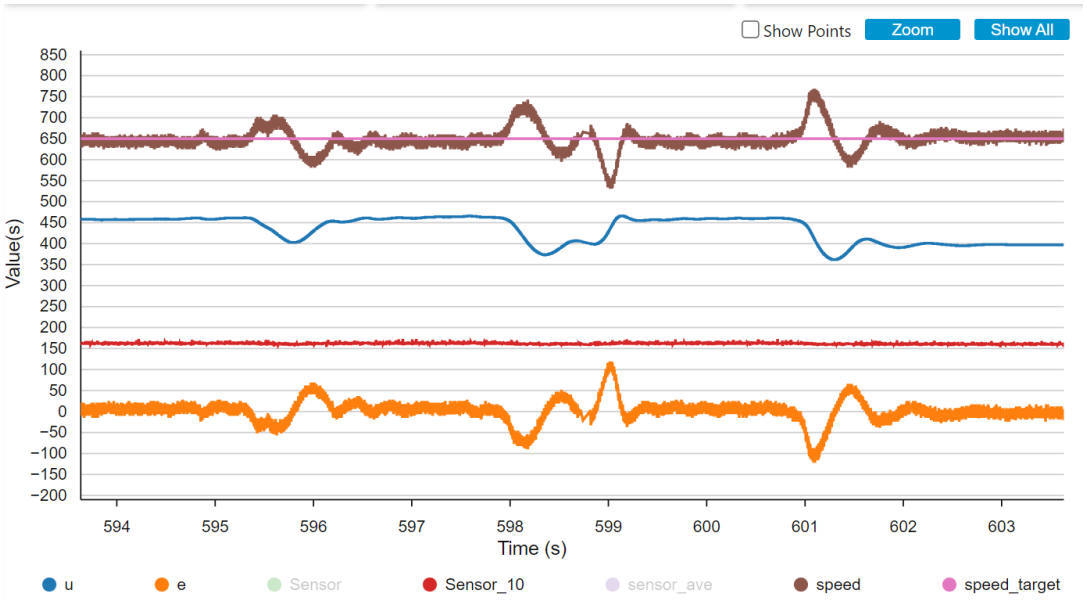

Рисунок 27 – Поведение системы при воздействии нагрузки на двигатель.

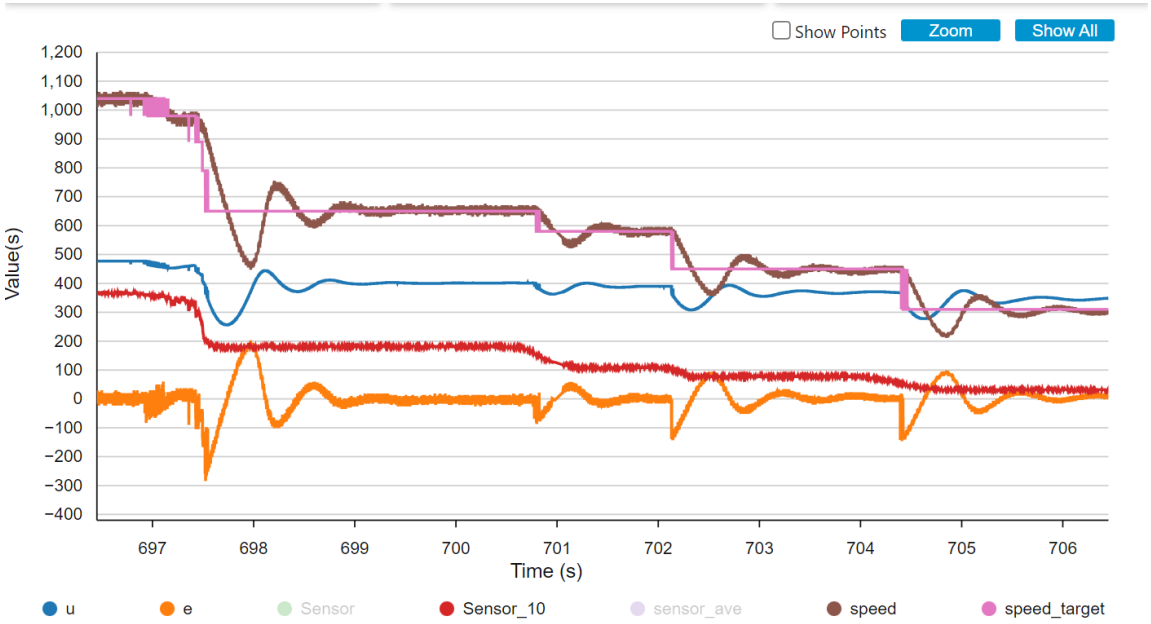

Рисунок 28 – Поведение системы при понижении значения датчика.

Вывод: В целом, система довольно хорошо работает с кодом. Мы можем проверить из этих результатов выше. Тем не менее, остается проблема с шумом и его причиной от сигнала датчика. Мы должны исправить это, чтобы система работала лучше.

На производственно-технологической практике были закреплены знания по разработке принципиальных электрических схем и подбору компонентов. Получены навыки работы с программным обеспечением STM32, в которых производились разработка программного кода, чтение различной документации по работе с микроконтроллером и его устройству, мониторинг работы программы в реальном времени. Появился опыт работы с программой KiCad7.0, использовавшихся для разработки принципиальной электрической схемы и составления отчетной документации, ПИД регулятором, применяемым во многих регулируемых системах. Полученные навыки и знания будут необходимы при написании курсовых и дипломной работ, а так же в последующей инженерной деятельности.

Список литературы

1. Драйвер коллекторного двигателя drv8833 [Электронный ресурс] // alex-exe.ru : информационный портал. URL: https://alex-exe.ru/radio/robotics/driver-collectors-motor-drv8833/
2. Обзор драйвера шагового двигателя A4988 [Электронный ресурс] // robotchip.ru : информационный портал. URL: https://robotchip.ru/obzor-drayvera-shagovogo-dvigatelya-a4988/
3. Как работают резистивные датчики[Электронный ресурс] // rxtx.su: информационный портал. URL:https://rxtx.su/moduli-i-datchiki/kak-rabotayut-rezistivnye-datchiki/
4. Маниш К.С. Разница между шаговым двигателем и двигателем постоянного тока [Электронный ресурс] // translated.turbopages.org: информационный портал.URL:https://translated.turbopages.org/proxy_u/en-ru.ru.fff2b57f-649faa2d-8f5afa2f-74722d776562/https/www.tutorialspoint.com/difference-between-stepper-motor-and-dc-motor
5. ПИД регулятор - принцип работы. [Электронный ресурс] // owen.ru: информационный портал.URL: https://owen.ru/media/video/pid_princip_raboty
6. Robert Keim Choosing the Right Oscillator for Your Microcontroller [Электронный ресурс] // allaboutcircuits.com : информационный портал.
7. URL:https://www.allaboutcircuits.com/technical-articles/choosing-the-right-oscillator-for-your-microcontroller/
8. Кварцевый генератор. Принцип работы.[Электронный ресурс] // principraboty.ru: информационный портал. URL:https://principraboty.ru/kvarcevyy-generator-princip-raboty/
9. Принцип работы стабилизатора напряжения[Электронный ресурс] // principraboty.ru: информационный портал. URL:https://principraboty.ru/princip-raboty-stabilizatora-napryazheniya/