Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Специальное машиностроение»

КАФЕДРА «Колесные машины»

Отчёт о выполнении лабораторной работы №6

по курсу

«Управление техническими системами»

на тему

«Изучение принципа обратной связи в системах автоматического управления с использованием двигателя постоянного тока»

Студент СМ10-81 _____ В.Б. Сухоносенко
(подпись, дата) *(Ф.И.О.)*

Преподаватель _____ А.А. Смирнов
(подпись, дата) *(Ф.И.О.)*

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задачей лабораторной работы №6 являлось:

1. Изучить теоретические основ работы,
2. Разработка математической модели ДПТ в Matlab,Simulink и её исследование,
3. Построение нагрузочной характеристики ДПТ в координатах момент – обороты вала ДПТ при напряжении питания 12 В,
4. Определение необходимого напряжения питания для поддержания заданной частоты вращения вала ДПТ при нулевом внешнем моменте сопротивления,
5. Построение нагрузочной характеристики ДПТ в координатах момент – обороты вала ДПТ при выбранном напряжении питания,
6. Построение переходного процесса при достижении заданной угловой скорости вала ДПТ
7. Разработка математической модели в Matlab,Simulink системы автоматического управления оборотами вала ДПТ с ПИД-регулятором,
8. Реализация синтезированного ПИД-регулятора в виде управляющей программы для платы Arduino,
9. Проверка работы САУ на лабораторной установке и запись данных для построения графика переходного процесса,
10. Сравнение результатов теоретических расчётов и эксперимента.

1 Выполнение работы

1.1 Модель ДПТ

Двигатель постоянного тока описывается системмой уравнений (в операторной форме, p - оператор дифференцирования)

$$\begin{cases} i_a = \frac{u_a - k_e\omega}{L_a p + r_a} \\ J\frac{d\omega}{dt} = k_M i_a - M_c \end{cases} \tag{1.1}$$

Здесь i_a - ток якоря, u_a - напряжение, подаваемое на цепь якоря, r_a,L_a обозначают сопротивление и индуктивность в схеме замещения ДПТ соответственно, k_e,k_M - коэффициент ЭДС и коэффициент моментный - оба зависят от конструкции, M_c - момент сопротивления, состоящий из внутреннего механического сопротивления двигателя M_int и внешнего сопротивления нагрузки (в лабораторной работе - момент от генератора, к которому подключён ДПТ).

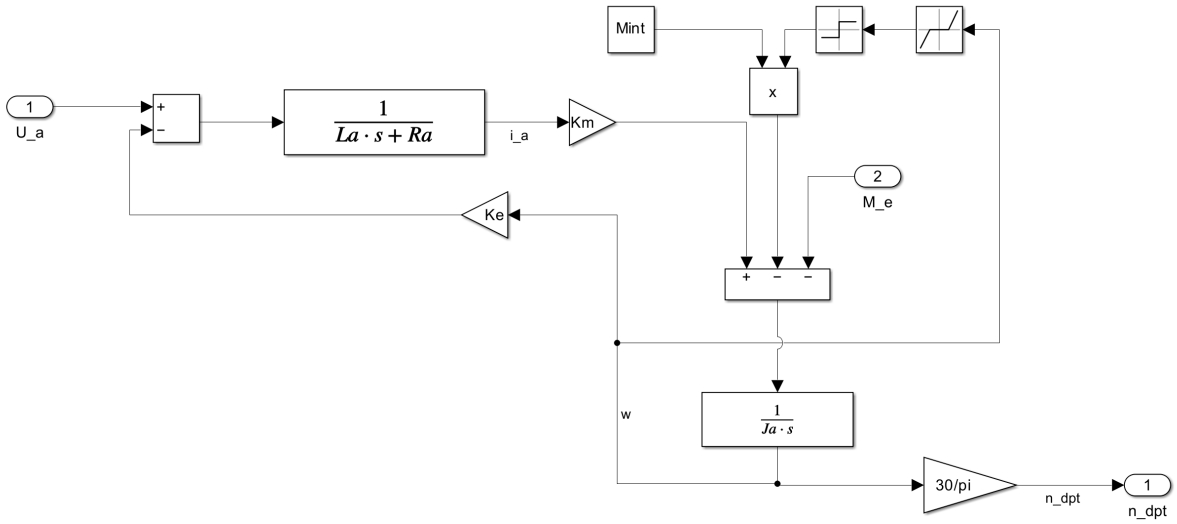

Рисунок 1.1 — Схема Simulink ДПТ

На рисунке 1.1 представлена схема Simulink, решающая систему уравнений выше. Модель ДПТ принимает значения u_a,M_c и выдаёт на выход значение оборотов вала n. Сопротивление M_int реализовано с помощью нелинейных блоков Dead space, Sign.

В .m-файле заданы параметры модели:

```
Mint=1e-3;
V_max = 12;
Ra= 1;
La = 0.018;
Ja=7.3e-5;
Km=0.018;
Ke=27.8e-3;
n_zad = 2750;
nmax= 4103;
Mmax=0.216; % Максимальный момент двигателя
```

При $M_c = 0, u_A = 12V$ определена максимальная частота вращения двигателя (4100 об/мин) - см. рисунок 1.2.

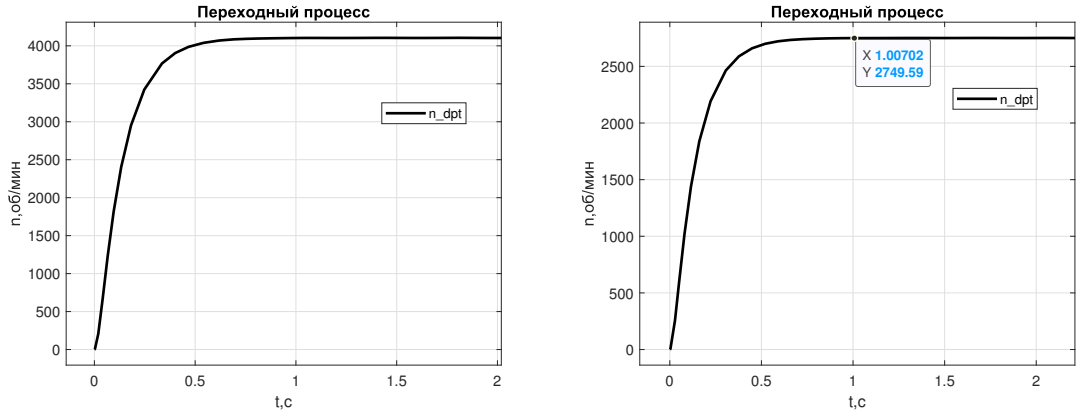

Рисунок 1.2 — Переходный процесс при холостом ходе, $u_a = 12V$ и $u_a = 8.06V$

Определено, что при $u_a = \frac{2750}{4100} \cdot 12 \approx 8.06V$ устанавливаются обороты 2750 об/мин при холостом ходе, что соответствует варианту задания - см. рисунок 1.2

С помощью симуляции модели при различных значениях M_c при заданном u_a строилась нагрузочная характеристика двигателя: (см. рисунок 1.3)

```
Nm=16;
U=12;
for i = 1:Nm+1
    Me=(i-1)*Mmax/Nm;
    M(i)=Me;
    sim('pidanddpt.slx',10);
```

```
Nd(i)=n_dpt(end);
end
U=8.06;
for i = 1:Nm+1
Me=(i-1)*Mmax/Nm;
M1(i)=Me;
sim('pidanddpt.slx',10);
Nd1(i)=n_dpt(end);
end

plot(Nd,M);
hold on
plot (Nd1,M1)
xlim([0 4200])
legend("12V","8,06V")
```

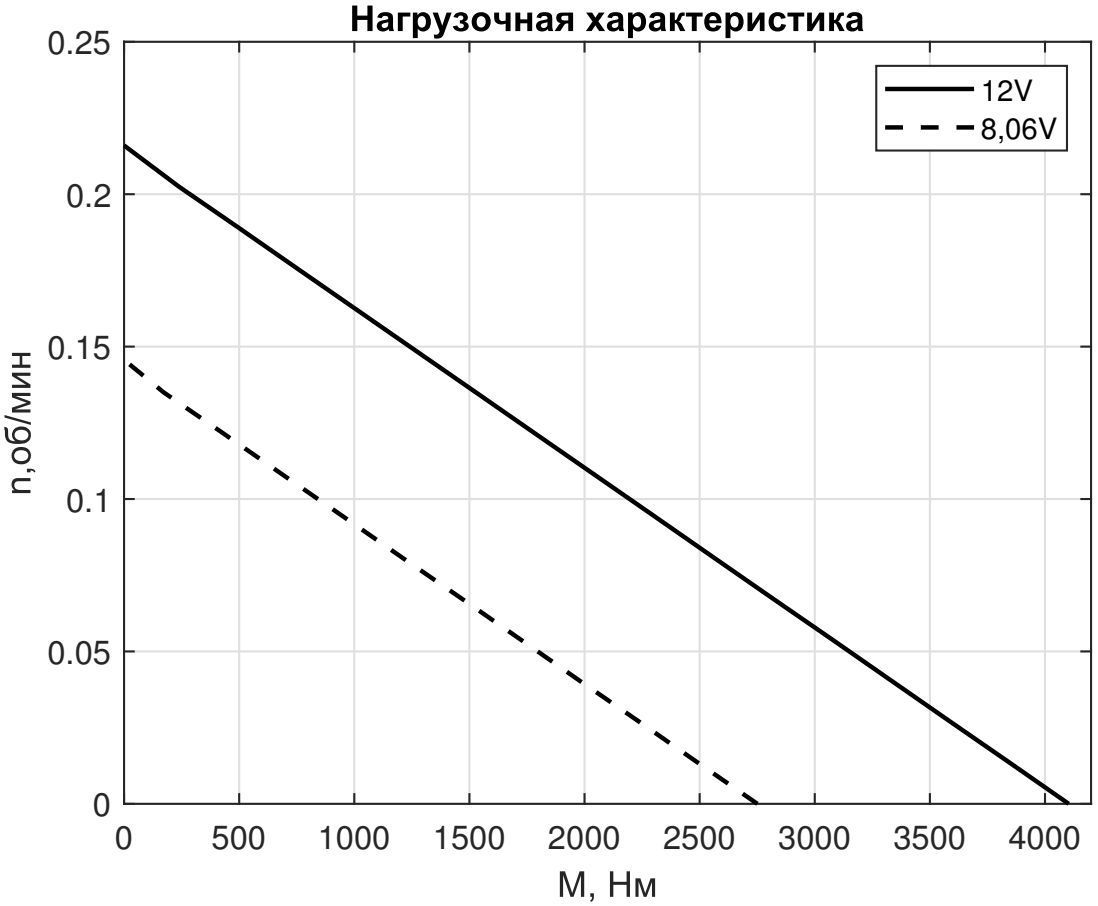

Рисунок 1.3

1.2 Модель системы управления

Управление ДПТ реализуется с помощью ПИД-регулятора, который изменяет воздействие u_a в пределах от 0 до 12 В в зависимости от текущего значения n оборотов вала, получаемых по обратной связи с помощью датчика Холла.

Составлена модель системы управления - см. рисунок 1.4. Установка на обороты равна 2750 об/мин. Введено ограничение на максимальное и минимальное управляющее воздействие - от 0 до 12 В.

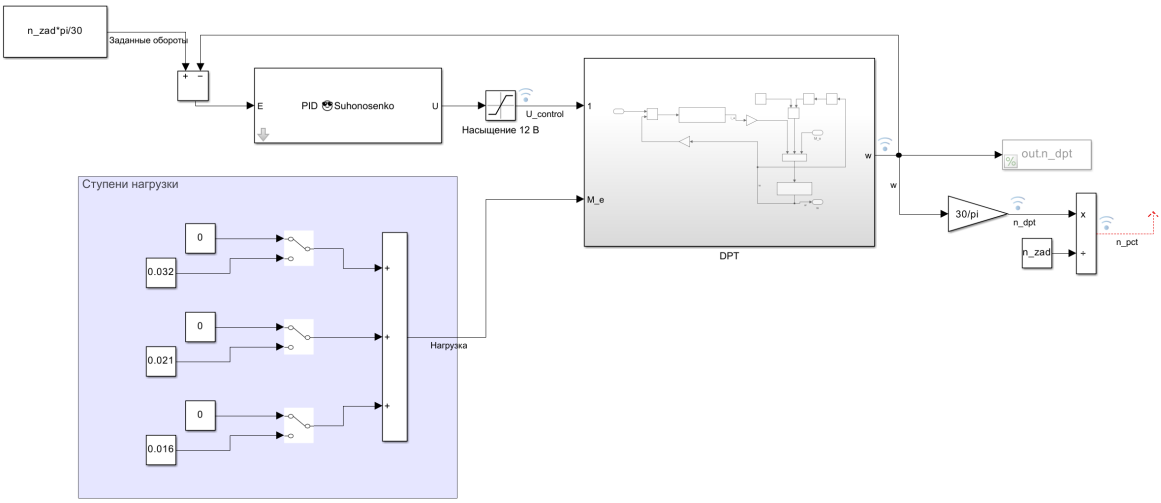

Рисунок 1.4 — Модель ДПТ с системой управления

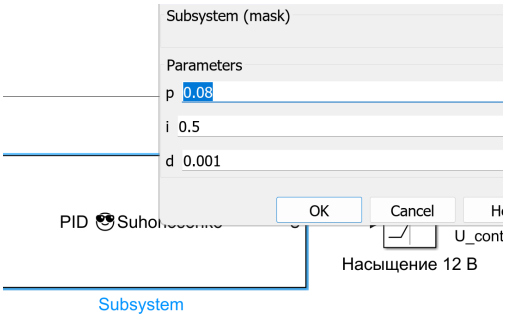

Рисунок 1.5

Приняты коэффициенты для ПИД регулятора: $kp = 0.08, ki = 0.5, kd = 0.001$. Эти коэффициенты обеспечивают перерегулирование менее 20 процентов и время регулирования меньше 2 секунд. Переходный процесс в системе САУ с настроенным регулятором показан на рисунке 1.6 и 1.7.

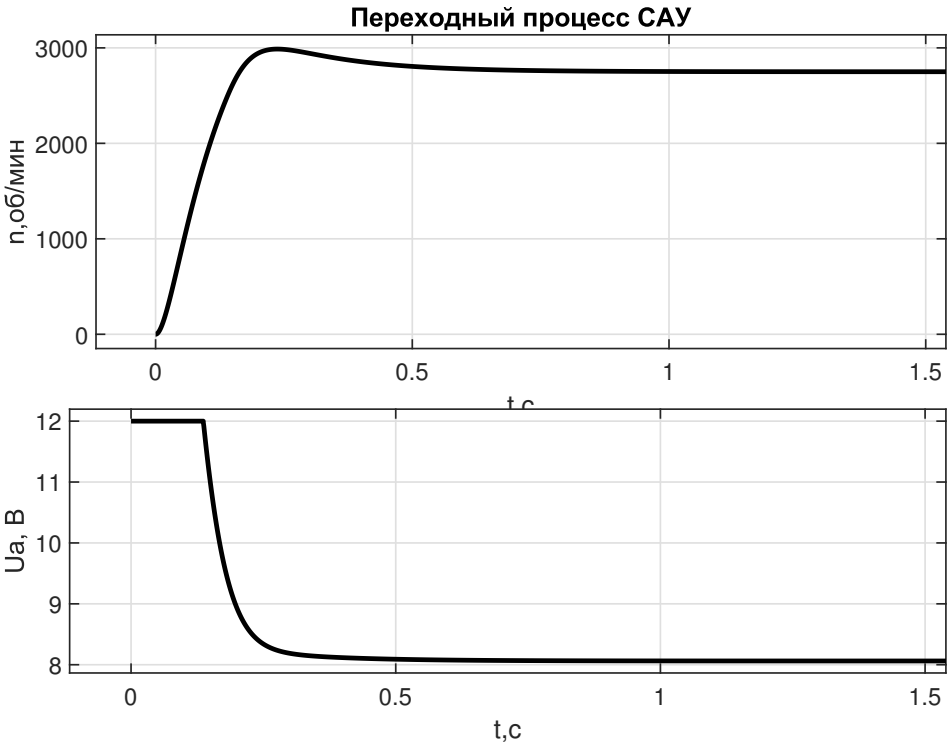

Рисунок 1.6 — Переходный процесс САУ - холостой ход

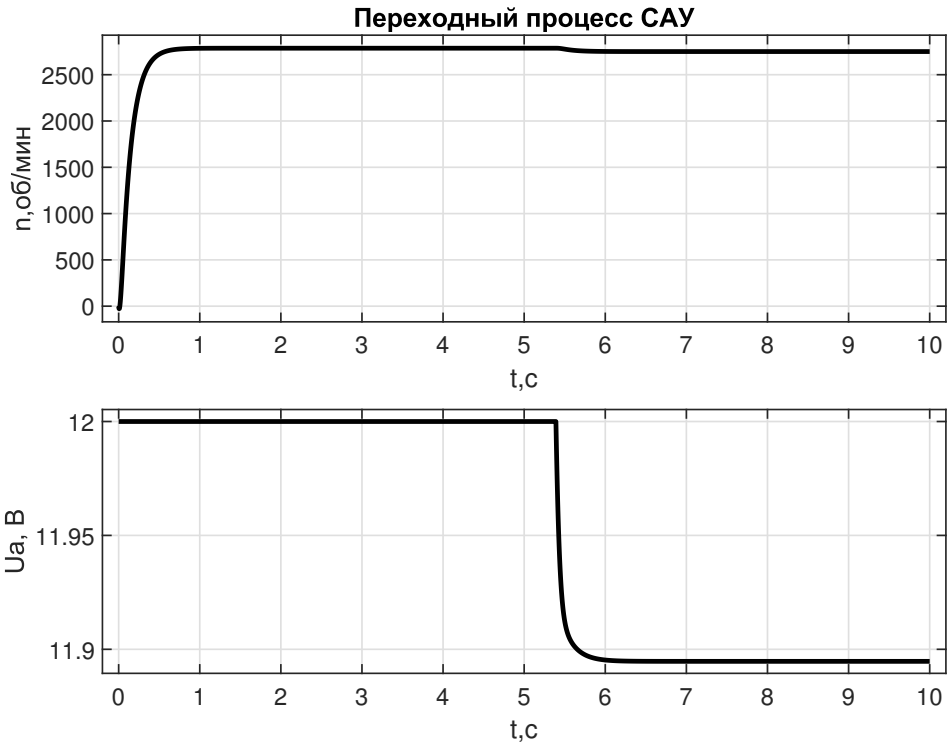

Рисунок 1.7 — Переходный процесс САУ - максимальная нагрузка

1.3 Реализация ПИД в Arduino

Для расчёта интеграла ошибки применена формула прямоугольников. Период срабатывания таймера задан 50000 мкс. Целевое значение частоты вращения 288 рад/с. Программа для Arduino приведена ниже. Обороты определяются с помощью алгоритма фиксированного времени (Считается число импульсов ImpulseCnt и делится на интервал таймера TimerInterval). Выполняется это с помощью прерываний.

Используются библиотеки для реализации программного таймера, вывода на ЖК-дисплей значений оборот в реальном времени.

Во время эксперимента сначала двигатель работал вхолостую, затем в три этапа производилось увеличение нагрузки. В конце нагрузка полностью снималась. Между каждым этапом было выдержано 5 секунд. Сигнал с датчика Холла с помощью АЦП и программы LGraph записан в виде массива значений время-напряжение. Текстовый документ lr.txt импортирован в Matlab и обработан с помощью Simulink-модели (рисунки 1.10 и 1.10).

Модель представляет собой алгоритм фиксированного времени для оценки угловой скорости. В Matlab использованы параметры, соответствующие частоте записи значений АЦП.

```
Ts=50e-3
dt=2e-5;
time = [lr(:,1),lr(:,1)];
signal = [lr(:,1),lr(:,2)-2.0];
```

Видно, что во время эксперимента обороты вала сильно колеблются (240-370 рад/с) около целевых оборотов 288 рад/с (соответствует 2750 об/мин). Во время эксперимента был слышен сильный шум установки, движение нельзя было назвать установившимся. Наиболее плавно вал вращался при максимальной нагрузке, при которой показания датчика в среднем равны $\frac{293.2+284.8}{2} = 289$ рад/с.

```
//Сухоносенко, Хоронский 2026
#include <TimerOne.h> //Предварительно д.б.импортирована библиотека
//TimerOne https://www.pjrc.com/teensy/td_libs_TimerOne.html
#include <Wire.h>
#include <LiquidCrystal_I2C.h> //Предварительно д.б.импортирована
//библиотека LiquidCrystal_I2C
//Объявление используемых функций
void CalcControl(void); // функция определения сигнала управления
void ImpulseCount(void); // функция подсчёта импульсов
//Объявление глобальных констант, переменных
const float pi =3.1416; // и ежу понятно :)
const int ImpulsePerRound = 15; //число имп. датчика Холла на оборот
const float C1 = 2*pi*1000000/ImpulsePerRound; // константа для
//перевода имп/мкс в рад/с
const float C2 = 1000000*60/ImpulsePerRound;
//
//перевода имп/мкс в об/мин
const long TimerInterval = 50000; // период срабатывания таймера в мкс
const int MotorPin = 11;   // выход (с ШИМом) для управления мотором
const int StartMotorPin = 12; // вход для кнопки "Старт" мотора
const int HallSensorPin = 2; // вход для от датчика Холла
const float OmegaTarget = 288; //заданная угловая скорость в рад/с
int ImpCnt = 0; //переменная для подсчета импульсов от датчика за
//период срабатывания таймера
// коэффициенты ПИД-регулятора
const float Kp = 0.08;   // Коэф. ПИД-регулятора Кп
const float Ki = 0.5*TimerInterval/1000000; // Коэф. ПИД-регулятора Ки
//с учетом интервала дискретизации
const float Kd = 0.003*1000000/TimerInterval; // Коэф. ПИД-регулятора Кд
//с учетом интервала дискретизации
float ErrorPrev = 0.0; //значение ошибки во время предыдущ. прерывания
float ErrorIntegral = 0.0;
//интеграл ошибки
float OmegaCur = 0.0;
//текущая частота вращения мотора рад/с
float Error = OmegaTarget - OmegaCur;  //текущая ошибка
float u = Kp*Error;
//сигнал управления
int StartMotor = 1; // состояние кнопки "Старт" мотора (1-Off, 0-On)
int StateMotor = 0; // состояние мотора (1-On, 0-Off)
LiquidCrystal_I2C lcd(0x38, 20, 4); // Экземпляр объекта ЖК-экрана
void setup() {
  // put your setup code here, to run once:
lcd.init();
lcd.backlight();
pinMode(MotorPin, OUTPUT);
pinMode(StartMotorPin, INPUT);
Timer1.initialize(TimerInterval);
Timer1.attachInterrupt(CalcControl);
attachInterrupt(0, ImpulseCount, RISING);
```

Рисунок 1.8 — Код для контроллера Arduino

```
45  lcd.backlight();
46  pinMode(MotorPin, OUTPUT);
47  pinMode(StartMotorPin, INPUT);
48  Timer1.initialize(TimerInterval);
49  Timer1.attachInterrupt(CalcControl);
50  attachInterrupt(0, ImpulseCount, RISING);
51  interrupts();
52  }
53
54  void loop() {
55    // put your main code here, to run repeatedly:
56  int Omega_rpm = 0;
57  StartMotor = digitalRead(StartMotorPin);
58  if (StartMotor == 0 && StateMotor == 0){
59    StateMotor = 1;
60    analogWrite(MotorPin, 50);
61    delay(150);
62  }
63  else if (StartMotor == 0 && StateMotor == 1) {
64    StateMotor = 0;
65    analogWrite(MotorPin, 0);
66    delay(150);
67  }
68  if (u > 12.0) u = 12.0;
69  else if (u < 0) u = 0.0;
70  analogWrite(MotorPin, int(255*u/12));
71  Omega_rpm = int(OmegaCur*C2);
72  lcd.setCursor(1, 1);
73  lcd.print("RPM=     ");
74  lcd.setCursor(6, 1);
75  lcd.print(Omega_rpm);
76  }
77  void ImpulseCount(void) {
78    ImpCnt++;
79  }
80  void CalcControl(void) {
81    if (StateMotor == 0) {
82      OmegaCur = 0;
83      Error = 0;
84      ErrorIntegral = 0;
85      u = 0;
86      return;
87    }
88    OmegaCur = float(ImpCnt)/float(TimerInterval);
89    ImpCnt = 0;
90    Error=OmegaTarget-OmegaCur*C1;
91    ErrorIntegral=ErrorIntegral+Error;
92    u = Kp*Error + Ki*ErrorIntegral + Kd*(Error - ErrorPrev);
93    ErrorPrev = Error;
94  }
```

Рисунок 1.9 — Код для контроллера Arduino. Продолжение

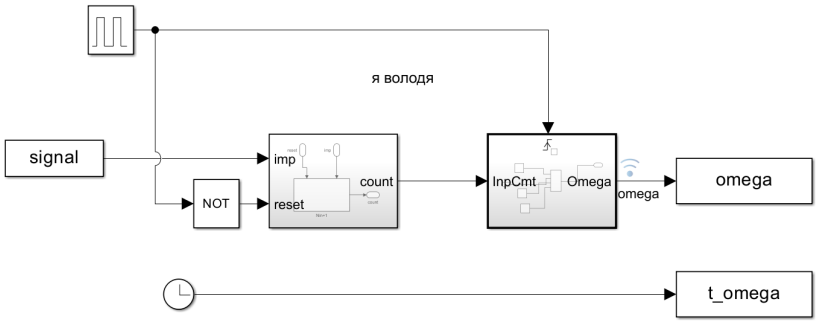

Рисунок 1.10 — Модель для пересчёта импульсов датчика в частоту

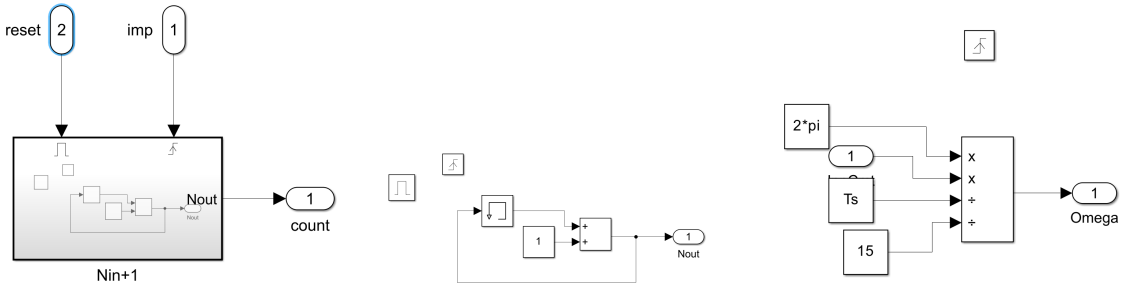

Рисунок 1.11 — Подмодели

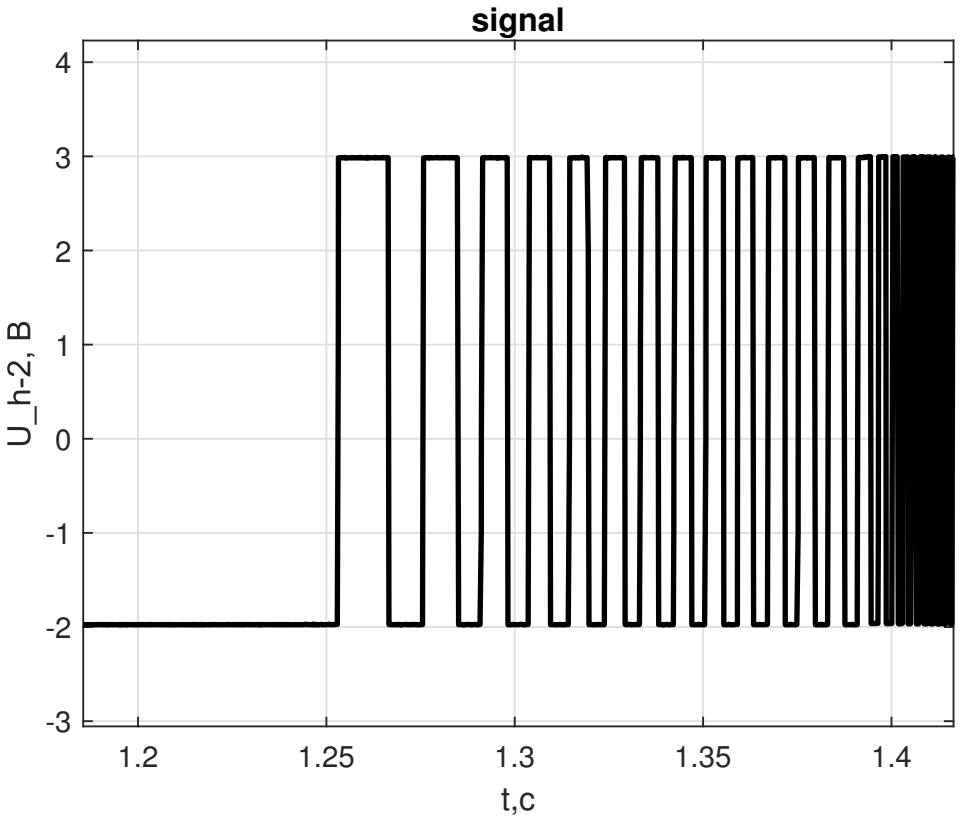

Рисунок 1.12 — Сигнал с датчика (смещён на 2 для обработки) при разгоне

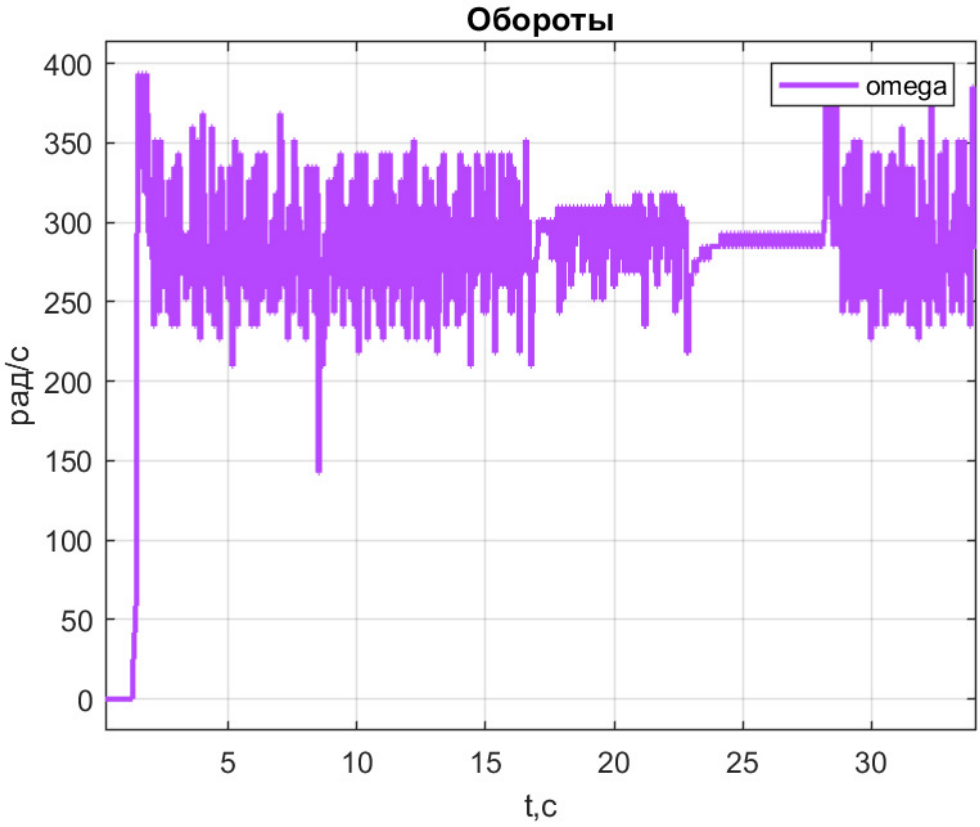

Рисунок 1.13 — Обороты вала (эксперимент)

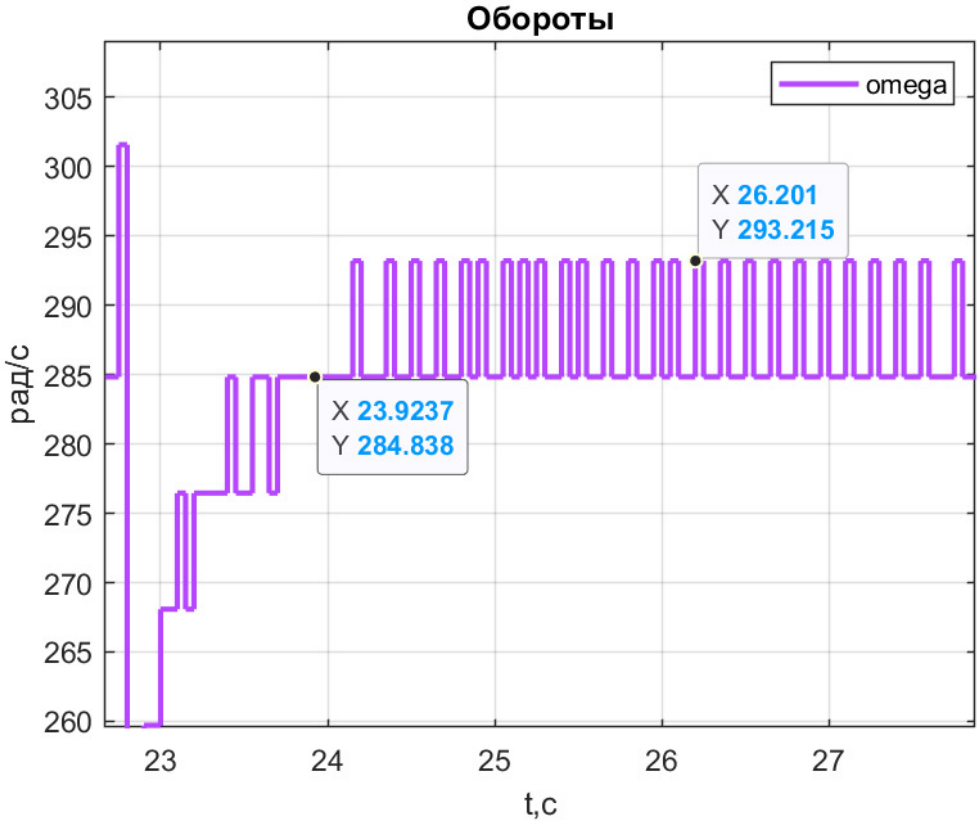

Рисунок 1.14 — Обороты при максимальной нагрузке

ЗАКЛЮЧЕНИЕ

В лабораторной работе разработана модель САУ двигателем постоянного тока, в которой применяется ПИД регулятор. Разработана программа для платы Arduino, управляющей драйвером мотора.

В результате проведения эксперимента можно сделать вывод, что написанная программа управления даёт неудовлетоврительные результаты. При различной нагрузке достигается разброс в 100 рад/с и более относительно целевых оборотов. Стабильная работа наблюдается только при максимальной нагрузке двигателя генератором с лампами.

Работа САУ проверена на двух установках. В обоих случаях качественно результаты были схожи.

Причинами несовпадения расчётной модели с экспериментом могут являться:

— Неучёт работы регулятора с дискретным шагом - в модели Simulink PID-регулятор выполнен как элемент, производящий вычисления непрерывно во времени;

— Наличие нелинейностей в реальной системе, в т.ч. вязкого трения. Внутреннее трение в модели приближалось как линейно зависящая от частоты;

— Подбор коэффициентов PID-регулятора по непрерывной модели, неучёт ограничений установки по реализации управляющего воздействия

— Некачественная обработка программой сигнала датчика Холла, регистрирующего обороты - метод фиксированного времени вносит погрешность;

Наиболее вероятной причиной неоптимальной работы САУ является некачествен ный подбор коэффициентов ПИД-регулятора - сильные колебания говорят о неправильной величине коэффициентов и их соотношении.