МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Инженерная школа информационных технологий и робототехники

Направление 15.04.06 «Мехатроника и робототехника»

Отделение автоматизации и робототехники

Курсовой проект по дисциплине

«Встраиваемые системы»

Тема курсового проекта:

«**Синтез системы управления для холодильной установки**»

Выполнил ст. гр. 8EM51 _____ _____ Якушев Н. Е.

Подпись Дата Ф.И.О.

Проверил доцент _____ _____ Ланграф С.В.

Подпись Дата Ф.И.О.

Томск 2025

Оглавление

1. Цель курсового проекта

Изучение принципов управления определенным объектом с дополнительными условиями и реализация алгоритма управления на микроконтроллере.

2. Перечень подлежащих разработке вопросов

В курсовой работе требуется привести необходимое и достаточное по своему содержанию текстовое описание разделов с приложением схем, таблиц, блок-схем алгоритмов и графиков с выводами по итогам выполненной работы.

1. Описание объекта регулирования (краткое описание объекта регулирования, основное назначение и режимы работы, анализ характеристик и динамических свойств),

2. Разработка системы управления (структурная схема встраиваемой системы управления, функциональная схема встраиваемой системы управления, таблица сигналов и переменных),

3. Система автоматического управления (определение контуров регулирования и каналов для организации сигналов обратной связи, оптимизация контуров регулирования, имитационная модель линеаризованной системы управления, корректирующие цепи для соответствия дополнительным требованиям, программная реализация разработанной системы управления и объекта регулирования, реализация системы управления с ограничениями в контуре),

4. Элементы логического контроля (машина состояний системы управления, защитные функции),

5. Программные алгоритмы для микроконтроллера (распределение сигналов на внешних портах микроконтроллера, перенос на микроконтроллер программных алгоритмов разработанной встраиваемой системы управления и объекта регулирования, экспериментальные исследования алгоритмов управления разработанной встраиваемой системы на микроконтроллере).

3. Задание

По варианту необходимо спроектировать систему управления для двигателя постоянного тока. Его заданные характеристики и условия выполнения курсовой работы представлены в таблице 1.

Таблица 1 - Параметры и условия для выполнения курсовой работы

№ вар.	K_1	K_2	K_3	K_4	T_1	T_2	T_3	T_4	$X_1^{макс}$	$Y_2^{макс}$	Возм.	$Y_4^{макс}$
18	6.74	3.12	0.874	-	0.0013	0.0165	-	0.207	±3	±250	17	90

Частота обновления сист. управления, Гц	2-ой порядок астатизма по управлению	Нулевая ошибка по возмущению	Защиты
1000	**Да**	Нет	от превышения максимального уровня Y_4 от снижения ниже минимального уровня Y_4

Схема варианта размещения звеньев с возмущением представлена на рисунке 1.

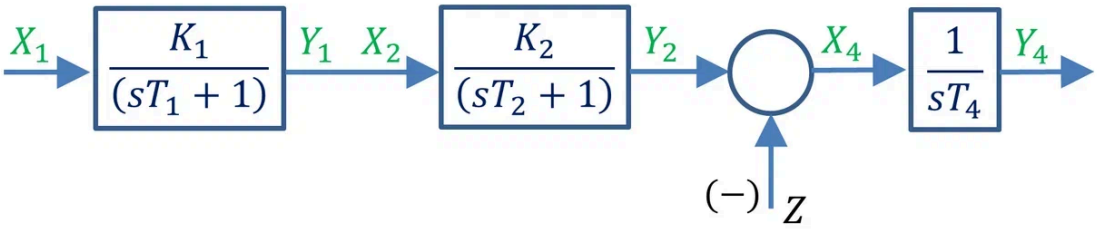


Рисунок 1 – Схема размещения звеньев с возмущением

Все файлы проекта также сохранены в GitHub. QR-код представлен на рисунке 2.

Рисунок 2 – QR-код GitHub с проектом

4. Ход работы

4.1. Описание объекта регулирования

4.1.1. Краткое описание объекта регулирования

Объект регулирования, представленный на рисунке 1, состоит из двух апериодических звеньев первого порядка, интегрирующего звена и учета возмущения. Данную структуру можно проиллюстрировать на примере системы управления холодильником на элементах Пельтье. Разберём сигналы, указанные на рисунке 1:

- входной сигнал X_1 представляет собой напряжение, подаваемое на систему,
- выходной сигнал Y_1 (X_2) соответствует току, протекающему через элементы Пельтье,
- выходной сигнал Y_2 характеризует полезную мощность охлаждения внутри холодильника,
- входной сигнал X_4 моделирует внешнее возмущение — мощность теплопритоков, учитывающую, например, потери при открытой двери холодильника,
- выходной сигнал Y_4 соответствует температуре внутри холодильника.

Данная аналогия соответствует дополнительным условиям задачи. Элементы Пельтье обычно работают при низком напряжении, но требуют значительного тока, а установка из нескольких модулей может потреблять очень большой ток, что актуально для проектирования системы управления и защиты. Кроме того, температура внутри холодильника имеет определённый диапазон, что удовлетворяет условиям по необходимым ограничениям и защите.

4.1.2. Основное назначение и режимы работы

Данная система регулирования предназначена для поддержания заданной пользователем температуры, необходимой для хранения различных объектов (например, продуктов, лекарств). Холодильники на элементах Пельтье также могут использоваться в стендах/установках для создания постоянной температурной среды, как отрицательной, так и положительной.

Данная установка может работать в нескольких режимах:

- активное охлаждение (используется при смене температуры на более низкую, чем в внутри холодильника),
- поддержание постоянной температуры (при открытии дверцы),
- режим обогрева (для разморозки),

Дополнительно для холодильной установки рассмотрим функции, необходимые для оповещения и защиты:

- световая индикация с оповещением о состоянии установки,
- контроль состояния двери для учёта возмущений в объекте регулирования,
- контроль исправной работы датчика температуры и правильной работы системы охлаждения.

К преимуществам элементов Пельтье по сравнению с компрессорными системами (фреоновыми) относятся возможность как охлаждения, так и нагрева, а также бесшумность работы. Основным недостатком является необходимость эффективного отвода тепла с горячей стороны модуля.

4.1.3. Анализ характеристик и динамических свойств объекта регулирования

На основании имеющихся данных и параметров объекта регулирования можно сделать следующие выводы:

- быстрое нарастание тока (переходный процесс около 0,004 секунды) с коэффициентом 6,74 (на один вольт приходится 6,74 А),
- нарастание мощности медлительней (переходный процесс около 0,05 секунд) с коэффициентом 3,12 (связано с КПД системы и нелинейными процессами),
- наличие постоянного возмущения говорит о необходимости постоянной работы системы на элементах Пельтье,
- медленное нарастание температуры относительно тока (на одну единицу мощности (Вт) приходится 4,83 единиц температуры).

Анализ характеристик показывает, что данная система обладает малой инерционностью по сравнению с другими типами холодильных установок.

Следовательно, полученные параметры объекта регулирования характерны для холодильных установок с малым полезным объёмом.

4.2. Разработка системы управления

4.2.1. Структурная схема системы управления

Для построения структурной схемы использовано приведенное выше описание объекта и общие принципы устройства холодильных систем. Структурная схема представлена на рисунке 3.

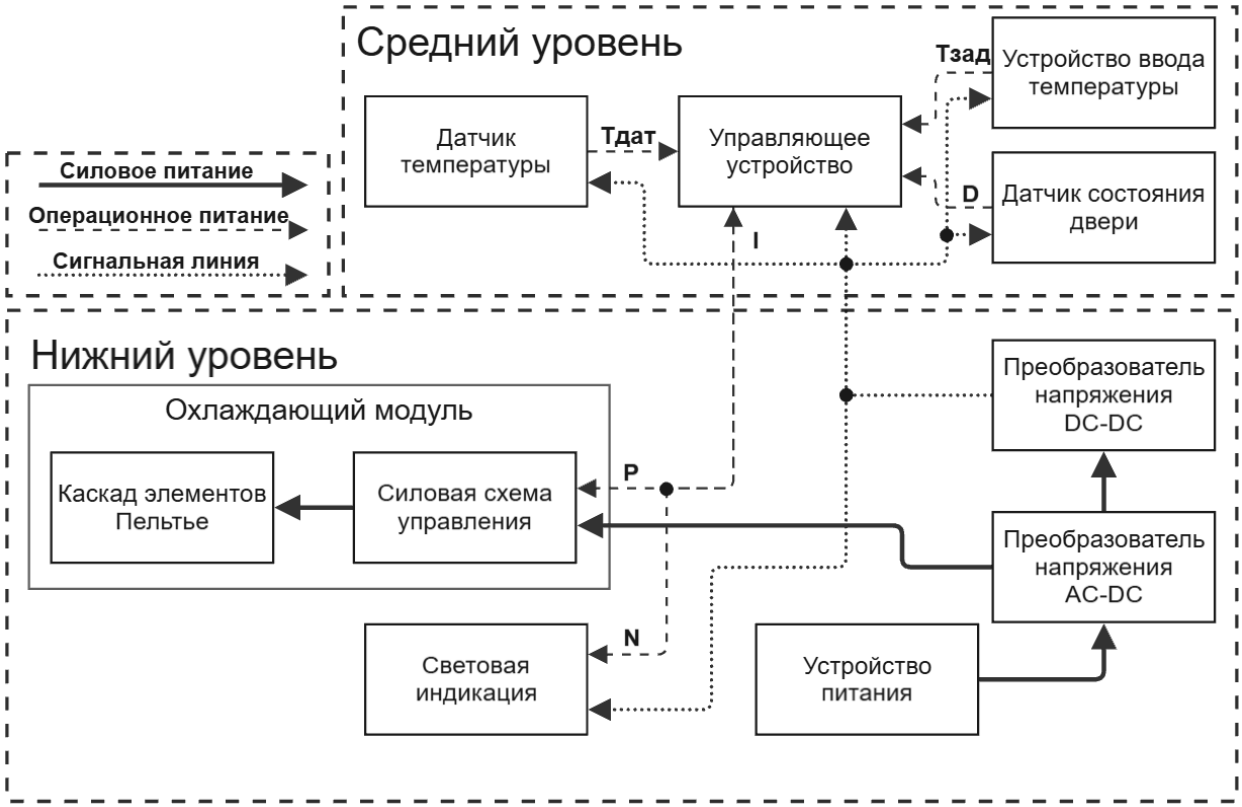


Рисунок 3 – Структурная схема объекта регулирования с системой управления

На данной структурной схеме обозначены блоки и связи между ними. К управляющему устройству подаются сигналы с датчика температуры, датчика состояния двери и значение с блока ввода температуры. После управляющее устройство подает сигналы для работы охлаждающей системы и подсветки. Вся система питается с устройства питания. На рисунке имеются следующие обозначения:

- $T_{\text{ЗАД}}$ – заданная температура с регулировщика, ℃.
- $T_{\text{ДАТ}}$ – температура с датчика температуры, ℃.

- D – состояние двери (открыта, закрыта).
- I – данные с датчика тока при работе компрессора, А.
- P – сигнал, генерируемый управляющим контроллером для управления силовой схемой управления и, соответственно, компрессором.
- N – сигнал, генерируемым управляющим контроллером для управления лампой.

4.2.2. Функциональная схема системы управления

Для разработки функциональной схемы требуется выбрать компоненты системы. Выбранные компоненты представлены в таблице 2.

Таблица 2 – Сводная таблица по компонентам

Устройство	Наименование	Цена, руб.
Управляющее устройство	STM32 Bluepill	350
Датчик состояния двери	Механический концевик NC	50
Датчик температуры	NTC датчик	470
Блок ввода температуры	3310C-001-502L	740
Элемент Пельтье	TEC1-12706 (55 Вт)	300
Силовая схема управления	Транзисторный мост	150
Трансформатор AC-DC 220-12В с мостовым выпрямителем	Понижающий трансформатор APEYRON 12В, 250Вт	1015
Преобразователь DC-DC 12-5В	Понижающий преобразователь LM2596	110
Питание сети	220 В	–
Переключатель	KCD3-101N-C2-R/3P	30
Светодиоды	Трехцветный светодиод	130

Результат разработки функциональной схемы представлен на рисунке 4.

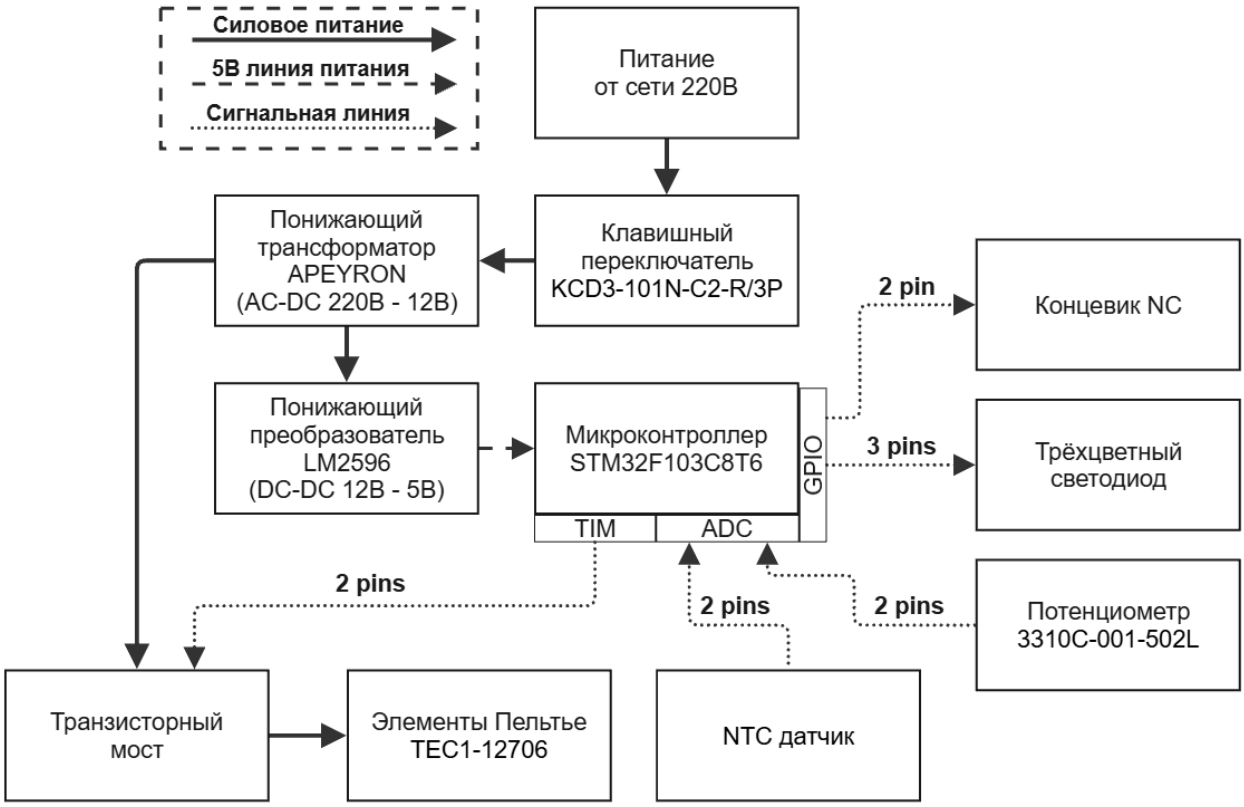


Рисунок 4 – Функциональная схема объекта регулирования с системой управления

Питание системы осуществляется от сети 220 В. Напряжение преобразуется AC-DC преобразователем в 12 В, а затем DC-DC преобразователем — в 5 В для питания логической части и микроконтроллера. Управление элементами Пельтье осуществляется через транзисторный мост, управляемый ШИМ-сигналами с микроконтроллера. В качестве органов управления используются концевой выключатель нормально-замкнутого типа (NC), потенциометр и клавишный переключатель. Для индикации режимов работы применяется трёхцветный светодиод.

4.2.3. Информационные каналы системы управления

В функциональной схеме используются интерфейсы GPIO, таймеры (TIM) и АЦП (ADC). Выводы GPIO задействованы для управления светодиодами и опроса состояния концевого выключателя, входы АЦП — для подключения датчика температуры и потенциометра, а таймеры — для формирования ШИМ-сигналов управления транзисторным мостом.

В таблице 3 представлены входы и выходы системы.

Таблица 3 — Таблица входов и выходов системы управления

№	Название	Диапазон	Знак	Тип сигнала	Назначение
1	Состояние потенцио-метра	0…4095	Без знака	Вход	Установка заданной температуры
2	Состояние концевого выключателя	0 – не нажата 1 - нажата	Без знака	Вход	Определение состояния двери
3	NTC датчик темпера-туры	0…4095	Без знака	Вход	Измерение температуры внутри холодильника
4	Цвета RGB светодиода	(0,0,0) - выкл. (1,0,0) - красный (0,1,0) - зелёный (0,0,1) - синий (0,1,1) - голубой	Без знака	Выход	Индикация состояния системы: - мигающий красный: авария - красный: режим нагрева - синий: режим охлаждения - зеленый: режим поддержания температуры - голубой: дверь открыта
5	Транзистор №1-2 (первая диагональ)	0 - закрыты 1 - открыты	Без знака	Выход	Включение режима охлаждения элементов Пельтье
6	Транзистор №3-4 (вторая диагональ)	0 - закрыты 1 - открыт	Без знака	Выход	Включение режима нагрева элементов Пельтье

Далее необходимо определить переменные, которые будут использоваться в системе управления объектом регулирования. Эти переменные представлены в таблице 4.

Таблица 4 — Таблица переменных

№	Обозна-чение	Название	Тип данных	Диапазон	Назначение
1	X_1	Управляющее напряжение	float	-3…3В	Напряжение управления транзисторным мостом, определяющее полярность и величину напряжения на элементах Пельтье
2	Y_1/X_2	Потребляемый ток	float	А	Измеренный ток через элементы Пельтье
3	Y_2	Полезная мощность	float	Вт	Полезная тепловая мощность, выделяемая / поглощаемая элементами Пельтье
4	X_4	Суммарная мощность	float	Вт	Результирующая мощность с учётом возмущений
5	Y_4	Температура холодильной установки	float	–	Измеренная температура внутри холодильной установки
6	T_{SET}	Заданное значение температуры	float	$°C$	Заданное значение температуры, установленное пользователем
7	Z	Состояние двери	uint8_t	0/17Вт	0 Вт - дверь открыта, 17 Вт - дверь закрыта
8	P_N	Входная полезная мощность	float	Вт	Полезная мощность, необходимая для охлаждения / нагрева холодильной установки
9	L_{STATE}	Режим светодиода	uint8_t	0..4	Код цвета индикации

4.3. Система автоматического управления

4.3.1. Контуры регулирования

Ранее в отчете были рассмотрены сигналы и их математические модели. Для обеспечения устойчивого переходного процесса температуры необходимо разработать структуру системы управления.

Управление элементами Пельтье осуществляется путём регулирования потребляемого тока в соответствии с требуемой полезной мощностью. Данный принцип реализуется через внутренний контур управления (контур мощности), в котором:

- на вход подаётся ошибка по полезной мощности,
- на выходе формируется управляющее напряжение для транзисторного моста,
- обратная связь организована по полезной мощности, рассчитываемой на основе данных датчика тока и напряжения.

Для преобразования сигнала с датчиков тока и напряжения в полезную мощность вводится коэффициент преобразования K_{KP}, который пересчитывает показания датчиков в мощность в Ваттах. В целом, данный коэффициент можно принять равным умножению КПД элементов Пельтье на их количество.

Расчёт полезной мощности осуществляется по формуле:

$$P_{\text{ПОЛ}} = k_{pr} \cdot n \cdot \eta \cdot I_{\text{Д}} \cdot U_{\text{Д}} = K_{KP} \cdot I_{\text{Д}} \cdot U_{\text{Д}},$$

где $U_{\text{Д}}$ – напряжение на модуле Пельтье, В;

$I_{\text{Д}}$ – ток через модуль, А;

k_{pr} – пропорциональный коэффициент преобразования показаний датчиков тока и напряжения.

η – КПД одного элемента Пельтье;

n – количество элементов Пельтье.

Внешний контур управления формирует задание по полезной мощности для внутреннего контура на основе:

- измеренной температуры (обратная связь с датчика),

- заданного значения температуры,
- расчёта ошибки по температуре.

Для преобразования сигнала датчика температуры в физические единицы измерения вводится коэффициент преобразования K_{KT}, который пересчитывает показания датчика в температуру в градусах Цельсия.

На рисунке 5 представлена структурная схема системы управления.

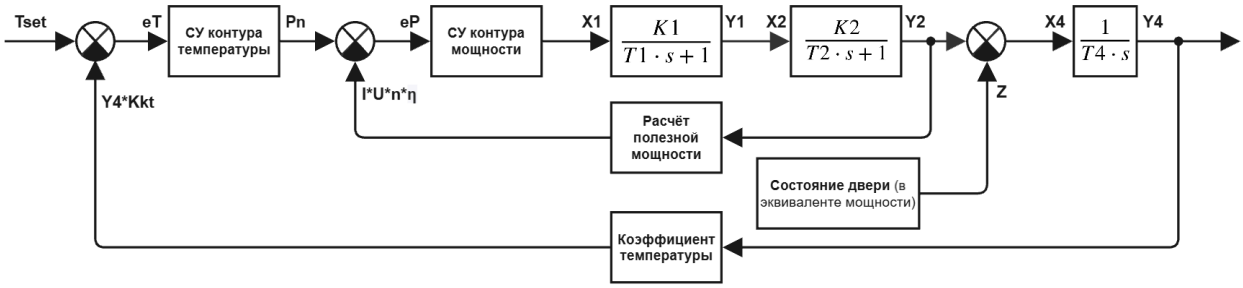


Рисунок 5 – Структурная схема системы управления объектом регулирования

4.3.2. Разработка системы управления

Во внутреннем контуре присутствует два апериодических звена первого порядка, так как дополнительные требования к внутреннему контуру отсутствуют (кроме ограничения управляющего напряжения), произведем синтез системы управления по модульному оптимуму.

Преимуществами синтеза систем по модульному оптимуму является простота реализации и расчёта, обеспечение астатизма первого порядка (статическая ошибка стремятся к нулю) и известные параметры переходного процесса (перерегулирование равное 4,3% и время переходного процесса в 4,1 раз больше минимальной постоянной времени знаменателя объекта управления).

Математическое описание разомкнутой системы объекта с системой управления с настройкой на модульный оптимум имеет вид:

$$W_{\text{Р}}^{MO}(s) = \frac{1}{2 \cdot s \cdot T_{\mu P} \cdot (T_{\mu P} \cdot s + 1)} = W_{\text{РЕГ}}(s) \cdot W_{\text{О}}(s),$$

где $T_{\mu P} = T_1$ – постоянная времени, равная минимальной постоянной времени объекта регулирования во внутреннем контуре, с;

$W_{\text{РЕГ}}(s)$ – передаточная функция системы управления;

$W_{0}(s)$ – передаточная функция объекта регулирования (внутренний контур).

На рисунке 6 представлен вид схемы с настройкой на модульный оптимум. Коэффициент K_{KP} примем равным единице для соблюдения входных условий курсовой работы.

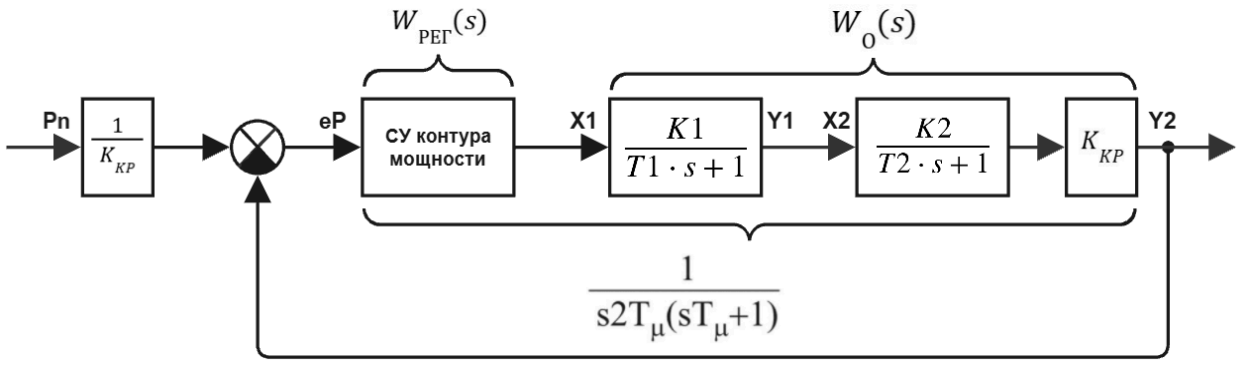


Рисунок 6 – Структурная схема системы управления внутренним контуром с настройкой на модульный оптимум

Далее произведем вывод передаточной функции регулятора:

$$\frac{1}{2 \cdot s \cdot T_{\mu P} \cdot (T_{\mu P} \cdot s + 1)} = W_{\text{РЕГ}}(s) \cdot W_{0}(s) \rightarrow W_{\text{РЕГ}}(s) = \frac{1}{2 \cdot s \cdot T_{\mu P} \cdot (T_{\mu P} \cdot s + 1)} \cdot \frac{1}{W_{0}(s)},$$

$$W_{\text{РЕГ}}(s) = \frac{1}{2 \cdot s \cdot T_{1} \cdot (T_{1} \cdot s + 1)} \cdot \frac{(T_{1} \cdot s + 1) \cdot (T_{2} \cdot s + 1)}{K_{1} \cdot K_{2} \cdot K_{KP}},$$

$$W_{\text{РЕГ}}(s) = \frac{1}{2 \cdot s \cdot T_{1}} \cdot \frac{(T_{2} \cdot s + 1)}{K_{1} \cdot K_{2} \cdot K_{KP}} = \frac{T_{2}}{2 \cdot T_{1} \cdot K_{1} \cdot K_{2} \cdot K_{KP}} \cdot \frac{(T_{2} \cdot s + 1)}{T_{2} \cdot s} = K_{P} \cdot (1 + \frac{K_{I}}{s}),$$

где K_{P} – пропорциональный коэффициент регулятора внутреннего контура;

K_{I} – интегральный коэффициент регулятора внутреннего контура;

В результате вывода передаточной функции регулятора был получен ПИ-регулятор в классической форме для внутреннего контура.

Далее необходимо вывести регулятор для внешнего контура температуры. Из дополнительный условий следует, что в системе необходимо обеспечить астатизм второго порядка (статическая и скоростная ошибки стремятся 0).

Для решения данной задачи воспользуемся настройкой системы управления на симметричный оптимум. Преимуществами данного синтеза является простота реализации и расчёта, обеспечение астатизма второго порядка и известные параметры переходного процесса (перерегулирование равное 43% и время переходного процесса в 29,4 раз больше минимальной постоянной времени знаменателя объекта управления).

Однако, ранее настроенный внутренний контур мощности в дальнейшем может вызвать сложности из-за сложности математического описания. Далее упростим внутренний контур мощности для настройки системы на симметричный оптимум:

$$W_{\text{ЗС}}^{MO}(s) = \frac{W_{\text{Р}}^{MO}(s)}{1 + W_{\text{Р}}^{MO}(s)} = \frac{\dfrac{1}{2 \cdot s \cdot T_{\mu P} \cdot (T_{\mu P} \cdot s + 1)}}{1 + \dfrac{1}{2 \cdot s \cdot T_{\mu P} \cdot (T_{\mu P} \cdot s + 1)}},$$

$$W_{\text{ЗС}}^{MO}(s) = \frac{1}{2 \cdot s \cdot T_{\mu P} \cdot (T_{\mu P} \cdot s + 1) + 1} = \frac{1}{T_{\mu P}^2 \cdot s^2 + 2 \cdot s \cdot T_{\mu P} + 1},$$

$$W_{\text{ЗС}}^{MO}(s) = \frac{1}{T_1^2 \cdot s^2 + 2 \cdot s \cdot T_1 + 1} \approx \frac{1}{2 \cdot s \cdot T_1 + 1}.$$

Данное упрощение допустимо, так как $T_1^2 \approx 0$. Далее можем приступить к математическому описанию регулятора.

Математическое описание разомкнутой системы объекта с системой управления с настройкой на симметричный оптимум имеет вид:

$$W_{\text{Р}}^{MO}(s) = \frac{(4 \cdot T_{\mu T} \cdot s + 1)}{8 \cdot T_{\mu T}^2 \cdot s^2 \cdot (T_{\mu T} \cdot s + 1)} = W_{\text{РЕГ}}(s) \cdot W_{\text{О}}(s),$$

где $T_{\mu T} = 2T_{\mu P} = 2T_1$ – постоянная времени, равная удвоенной минимальной постоянной времени объекта регулирования во внешнем контуре, с;

При настройке регуляторов все возмущения зануляются ($Z = 0$ Вт).

На рисунке 7 представлен вид схемы с настройкой на симметричный оптимум. Коэффициент K_{KT} примем равным единице для соблюдения входных условий курсовой работы.

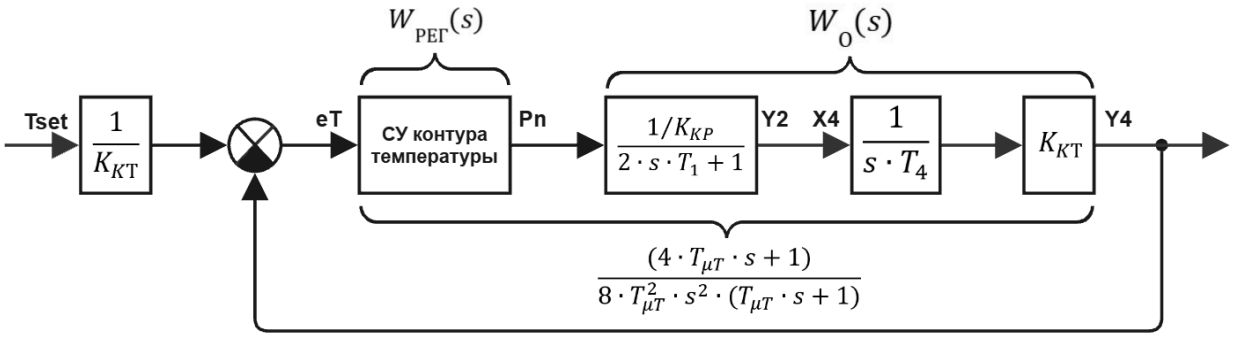


Рисунок 7 – Структурная схема системы управления внешним контуром с настройкой на симметричный оптимум

Далее произведем вывод передаточной функции регулятора:

$$\frac{(4 \cdot T_{\mu T} \cdot s + 1)}{8 \cdot T_{\mu T}^2 \cdot s^2 \cdot (T_{\mu T} \cdot s + 1)} = W_{\text{РЕГ}}(s) \cdot W_{\text{О}}(s),$$

$$W_{\text{РЕГ}}(s) = \frac{(4 \cdot T_{\mu T} \cdot s + 1)}{8 \cdot T_{\mu T}^2 \cdot s^2 \cdot (T_{\mu T} \cdot s + 1)} \cdot \frac{1}{W_{\text{О}}(s)},$$

$$W_{\text{РЕГ}}(s) = \frac{(8 \cdot T_1 \cdot s + 1)}{64 \cdot T_1^2 \cdot s^2 \cdot (2 \cdot T_1 \cdot s + 1)} \cdot \frac{K_{КР} \cdot T_4 \cdot s \cdot (2 \cdot s \cdot T_1 + 1)}{K_{КТ}},$$

$$W_{\text{РЕГ}}(s) = \frac{K_{KP} \cdot T_4}{8 \cdot T_1 \cdot K_{КТ}} \cdot \frac{(8 \cdot T_1 \cdot s + 1)}{8 \cdot T_1 \cdot s} = K_{PT} \cdot (1 + \frac{K_{IT}}{s}),$$

где K_{PT} – пропорциональный коэффициент регулятора внешнего контура;

K_{IT} – интегральный коэффициент регулятора внешнего контура;

В результате вывода передаточной функции регулятора был получен ПИ-регулятор в классической форме для внешнего контура.

Рассчитаем все значения коэффициентов регулятора в таблице 5.

Таблица 5 — Таблица коэффициентов регулятора

Контур	Коэффициент	Формула	Значение
Внутренний контур мощности	Пропорциональный	$K_P = \frac{T_2}{2 \cdot T_1 \cdot K_1 \cdot K_2 \cdot K_{KP}}$	0,302
	Интегральный	$K_I = \frac{1}{T_2}$	60,606
Внешний контур температуры	Пропорциональный	$K_{PT} = \frac{K_{KP} \cdot T_4}{8 \cdot T_1 \cdot K_{KT}}$	39,808
	Интегральный	$K_{IT} = \frac{1}{8 \cdot T_1}$	96,154

Представим полученную систему в математической среде моделирования (см. рисунок 8).

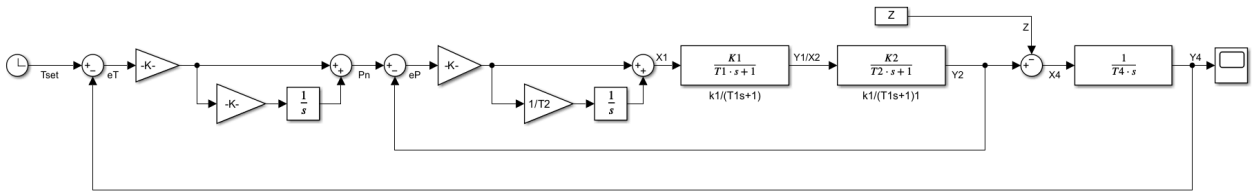


Рисунок 8 – Структурная схема системы в математической среде моделирования

4.3.3. Дополнительные и защитные механизмы

В этой главе разберем указанные в задании дополнительные и защитные механизмы. Для начала перечислим эти механизмы (согласно таблице 1):

- значения X_1 в диапазоне от -3 до 3,
- значения Y_2 в диапазоне от -250 до 250,
- значение возмущения равно 17,

- максимальное значение Y_4 равно 90,
- частота обновления систем управления (внутренний и внешний контура) равен 1000 Гц,
- обеспечение астатизма второго порядка всей системы,
- реализация защиты от превышения максимального значения Y_4 и снижения ниже минимального значения Y_4.

Астатизм второго порядка обеспечивается на внешнем контуре за счёт симметричного оптимума, а частота обновления системы управления настраивается при программной реализации алгоритма. Возмущение в данном случае заводится в систему в виде коэффициента и не требует отдельной настройки. Остальные условия будут рассмотрены ниже.

Для реализации ограничений в переменных X_1, Y_2 и Y_4 воспользуемся ограничения значений сверху и снизу (функция насыщения). Алгоритм работы заключается в принудительном приравнивании выходного значения верхней или нижней границе, если число вне диапазона.

Если с переменными Y_2 и Y_4 никаких проблем нет, то с X_1 является выходом ПИ-регулятора. Установка функции насыщения после ПИ-регулятора может вызвать проблемы в связи с тем, что при насыщении интегральная составляющая продолжит накапливать ошибку, что может привести к неустойчивости или очень долгому переходному процессу. Для борьбы с этим явлением существуют алгоритмы под названием “анти-виндап”, которые борются с накоплением ошибки интеграторов в случае насыщения.

Существуют два метода:

1. Анти-виндап с обратной связью по насыщению. Вычисляется разница между сигналами до и после функции насыщения, после чего они умножаются на коэффициент (K_{AW}) и взаимодействуют с интегральной составляющей.

2. Анти-виндап по условию (отключение интеграции). Данный метод при насыщении отключает расчёт интегральной составляющей. В связи с этим в нём остаётся то значение, которое было до этого.

Для выбора одного из вариантом была смоделирована система на рисунке 6 для двух вариантов анти-виндапа. Сами методы были протестированы с помощью блока “*PID Controller*”. Было выявлено, что отключение насыщения по второму методу не приводит систему к уставке (при больших входных параметрах переходный процесс становится неустойчивым). При моделировании с первым методом (при коэффициенте 1000) переходный процесс стал более устойчивым по отношению к большим входным сигналам. Поэтому далее будет использоваться анти-виндам с обратной связью по насыщению.

Структурная схема интеграции анти-виндапа и ПИ-регулятора представлена на рисунке 9.

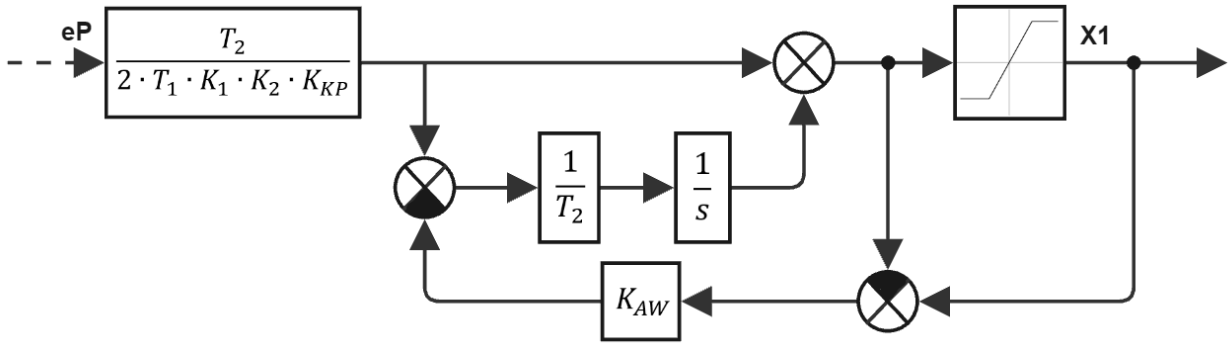


Рисунок 9 – Структурная схема ПИ-регулятора с анти-виндапом

Также по условию задания необходимо реализовать функцию защиты по переменной Y_4. Примем, что минимальное значение равно -90 (по условию не было задано). Суть функции защиты заключается в отключении управления, подаваемого на вход первому звену объекта регулирования (имитация выключения установки). Для этого необходимо дополнить ранее разработанные алгоритмы.

Предположим, что если значение температуры стоит на границы больше 0,5 секунд, то вся установка экстренно отключается. Для этого введем переменную, которая будет умножаться на выходной сигнал внутреннего контура мощности. В случае, если график находится на границе более 0,5 секунд, переменная приравнивается к нулю, в остальных случаях – единице.

Представим полученную систему в математической среде моделирования (см. рисунок 10). Система защиты будет представлена при дальнейшей реализации алгоритма.

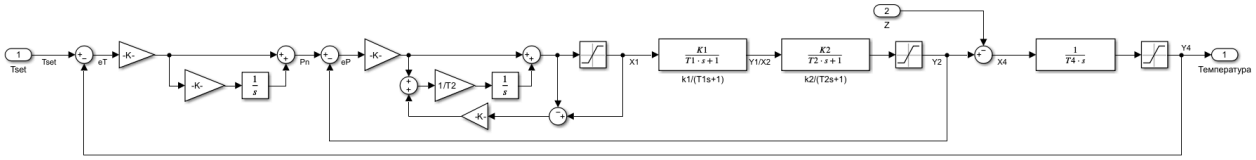


Рисунок 10 – Структурная схема системы в математической среде моделирования с функциями насыщения

4.3.4. Программная реализация с учетом дискретности

В реальных системах процессы моделируются дискретно, поэтому все ранее введённые передаточные функции необходимо преобразовать из непрерывного вида в дискретный с учётом выбранного шага дискретизации. Это требуется для корректной цифровой реализации алгоритмов управления.

Структурная схема, представленная на рисунке 10, включает следующие блоки: ПИ-регуляторы (с защитой от насыщения и без неё), апериодическое звено первого порядка, интегрирующее звено с коэффициентом, ограничители (функции насыщения) и сумматоры.

В то время как реализация сумматоров и функций насыщения в дискретной системе не вызывает сложностей, алгоритмы для остальных блоков требуют особого рассмотрения с учетом дискретности времени.

ПИ-регулятор в классической форме имеет вид:

$$y = K_p * (err + \frac{1}{T_и * s} * err),$$

Далее произведем преобразования, чтобы получить ПИ-регулятор в дискретной форме

$$y_i = K_p * err_i + K_p * err_i * \frac{1}{T_и * s},$$

$$I = K_p * err_i * \frac{1}{T_и * s},$$

$$T_и * \frac{dI}{dt} = K_p * err_i,$$

$$dI = K_p * err_i * \frac{dt}{T_и},$$

$$I_{i+1} = I_i + dI = I_i + K_p * err_i * \frac{dt}{T_и},$$

$$y_{i+1} = K_p * err_{i+1} + I_{i+1},$$

$$y_{i+1} = K_p * err_{i+1} + I_i + K_p * err_{i+1} * \frac{dt}{T_и},$$

Данное математическое описание системы готово к программной реализации, поскольку все переменные представлены в формате чисел с плавающей точкой (float/double), а также учтен шаг дискретизации, необходимый для работы в реальном времени.

Учёт механизма анти-виндапа сводится к добавлению дополнительной функции расчёта добавочного значения к звену насыщения. Разработанные программные алгоритмы для ПИ-регулятора представлены в листинге 1.

Листинг 1 – программная форма записи дискретного ПИ-регулятора

```
1.  float Kp;          // пропорциональный коэффициент
2.  float Ki;          // интегральный коэффициент
3.  float Prop;        // пропорциональная часть
4.  float Int_pr;      // предыдущее интегральное значение
5.  float Int;         // нынешнее интегральное значение
6.  float ErrSat;      // разница до и после функции насыщения
7.  float dt;          // шаг дискретизации системы
8.  float X1_Pre_Sat;  // неограниченный выход функции
9.  float X1;          // ограниченный выход функции
10. float X1_lim;      // лимит выходного значения
11.
12. float PI_reg(float error)
13. {
14.   // расчёт пропорциональной части
15.   Prop = Kp * error;
16.   // расчёт интегральной части без учёта анти-виндапа
17.   Int = Int_pr + Prop * Ki * dt;
18.   Int_pr = Int;
19.   // расчёт итогового значения выхода
20.   X1 = Prop + Int;
21.   return X1
22. }
23. float PI_reg_AW(float error)
24. {
```

```
25.     // расчёт пропорциональной части
26.     Prop = Kp * error;
27.     // расчёт интегральной части с учётом анти-виндапа
28.     Int = Int_pr + Prop * Ki * dt + ErrSat * Ki * dt;
29.     Int_pr = Int;
30.     // расчёт итогового неограниченного значения выхода
31.     X1_Pre_Sat = Prop + Int;
32.     // реализация функции насыщения
33.     if (X1_Pre_Sat > X1_lim)
34.     {
35.             X1 = 3;
36.     }
37.     else if (X1_Pre_Sat < -X1_lim)
38.     {
39.             X1 = -3;
40.     }
41.     // вычисление разницы до и после функции насыщения
42.     ErrSat = X1 - X1_Pre_Sat;
43.     return X1
44.   }
```

Аналогичную процедуру дискретизации применим к апериодическому звену первого порядка. Его передаточная функция в непрерывной форме может быть представлена дифференциальным уравнением:

$$T_1 * \frac{dy}{dt} + y = k * u,$$

где y – выходная величина (значение);

T_1 – постоянная времени звена;

k – коэффициент передачи;

u – входное значение.

При переходе к дискретной форме производная аппроксимируется разностью прямого метода Эйлера:

$$\frac{dy}{dt} = \frac{y_{i+1} - y_i}{\Delta t},$$

где y_{i+1} – выходное значение;

y_i – предыдущее значение;

Δt – шаг дискретизации.

Далее произведем замену и выведем выходное значение y_{i+1}:

$$T * \frac{y_{i+1} - y_i}{\Delta t} + y_{i+1} = k * u,$$

$$y_{i+1} = \frac{\Delta t \cdot k}{\Delta t + T_1} * u + \frac{T_1}{\Delta t + T_1} \cdot y_i.$$

Разработанный программный алгоритм для апериодического звена первого порядка представлены в листинге 2.

Листинг 2 – программная форма записи дискретного апериодического звена первого порядка

```
1.  float T;        // постоянная времени
2.  float dt;       // шаг дискретизации системы
3.  float K;        // коэффициент усиления
4.  float X2;       // входной сигнал
5.  float Y2;       // выходной сигнал
6.  float K1;       // коэффициент перед входном
7.  float K2;       // коэффициент перед предыдущим значением
8.  float Y2_pred;  // предыдущее выходное значение сигнала
9.
10. float func_first(float X2)
11. {
12.         // расчёт коэффициентов
13.         K1 = dt * K / (dt + T);
14.         K2 = T / (dt + T);
15.         // расчёт выходного значения
16.         Y2 = K1 * X2 + K2 * Y2_pred;
17.         // для реализации следующей итерации
18.         Y2_pred = Y2;
19.         return Y2
20. }
```

Аналогичную процедуру дискретизации применим к интегральному звену. Его передаточная функция в непрерывной форме может быть представлена дифференциальным уравнением:

$$T_1 * \frac{dy}{dt} = k * u,$$

Далее произведем замену и выведем выходное значение y_{i+1}:

$$T * \frac{y_{i+1} - y_i}{\Delta t} = k * u,$$

$$y_{i+1} = y_i + \frac{\Delta t \cdot k}{T_1} * u.$$

Разработанный программный алгоритм для интегрального звена представлен в листинге 3.

Листинг 3 – Программная форма записи дискретного интегрального звена

```
1.  float T;        // постоянная времени
2.  float dt;       // шаг дискретизации системы
3.  float K;        // коэффициент усиления
4.  float X4;       // входной сигнал
5.  float Y4;       // выходной сигнал
6.  float K1;       // коэффициент перед предыдущим значением
7.  float Y4_pred;  // предыдущее выходное значение сигнала
8.
9.  float func_int(float X4)
10. {
11.         // расчёт коэффициента
12.         K1 = dt * K / T;
13.         // расчёт выходного значения
14.         Y4 = Y4_pred + K * X4;
15.         // для реализации следующей итерации
16.         Y4_pred = Y4;
17.         return Y4
18. }
```

Дополнительно стоит учесть, что в программной реализации будут присутствовать функции индикации (на основе RGB-светодиода). В листинге 4 представлена программная реализация функции индикации.

Листинг 4 – Программная форма записи функции индикации

```
1.  float T;       // температура в настоящий момент времени
2.  float Tmax;    // максимальное значение температуры
3.  float Tmin;    // минимальное значение температуры
4.  float T_pr;    // режим нагрева/охлаждения (X1 > 0 || abs(T - T_pr) > T_dz)
5.  float T_dz;    // мёртвая зона для режима поддержания
6.  float Z;       // открыта/закрыта древь
7.  int red_l;     // состояние красного светодиода
8.  int blue_l;    // состояние синего светодиода
9.  int green_l;   // состояние зеленого светодиода
10. float t;       // текущее время
11. float t_st;    // время начала выхода температуры за допустимые пределы
12. float error;   // флаг ошибки системы
13. int counter;   // счетчик для реализации мигания светодиодом при ошибке
```

```
14.    float X1;      // управляющее воздействие или флаг режима работы
15.
16.    // Функция управления светодиодной индикацией при ошибке
17.    void err_light()
18.    {
19.        blue_l = 0;
20.        green_l = 0;
21.        counter++;
22.
23.        // Реализация мигания с периодом
24.        if (counter >= 50000 && red_l == 0)
25.        {
26.            red_l = 1;
27.        }
28.        else if (counter >= 100000 && red_l == 1)
29.        {
30.            red_l = 0;
31.            counter = 0;  // сбрасываем счетчик
32.        }
33.    }
34.
35.    // Функция определения состояния системы (стационарного режима)
36.    void def_st()
37.    {
38.        if ((T >= Tmax || T <= Tmin) && t_st == 0)
39.        {
40.            t_st = t;
41.        }
42.        else if ((t - t_st) > 5)
43.        {
44.            error = 1;
45.            X1 = 0;
46.            err_light(); // запускаем индикацию ошибки (мигание красным)
47.        }
48.        else if ((T < Tmax || T > Tmin) && error == 0)
49.        {
50.            t_st = 0;
51.        }
52.    }
53.
54.    // Функция управления светодиодами при открытой двери
55.    void door_state()
56.    {
57.        if (Z > 0 && error == 0)
58.        {
```

```
59.             blue_l = 1;
60.             green_l = 1;
61.         }
62.     }
63.
64.     // Функция управления светодиодами в зависимости от состояния объекта
65.     void obj_state()
66.     {
67.         if ((T - T_pr) > T_dz)
68.         {
69.             red_l = 1;
70.             blue_l = 0;
71.             green_l = 0;
72.         }
73.         else if (abs(T - T_pr) < T_dz)
74.         {
75.             red_l = 0;
76.             blue_l = 1;
77.             green_l = 0;
78.         }
79.         else
80.         {
81.             red_l = 0;
82.             blue_l = 0;
83.             green_l = 1;
84.         }
85.     }
86.
87.     // Основная функция управления индикацией и состоянием переменных
88.     void func_light(float Z, float T, float t, int on)
89.     {
90.         if (on == 1 && error == 0)
91.         {
92.             // Проверяем стационарное состояние температуры
93.             def_st();
94.
95.             if (error == 0)
96.             {
97.                 // Проверяем состояние двери и управляем индикацией
98.                 door_state();
99.             }
100.
101.            if (error == 0 && Z == 0)
102.            {
103.                // Управляем индикацией состояния объекта (температуры)
```

```
104.              obj_state();
105.          }
106.      }
107.      // Если система выключена (on == 0) и была ошибка
108.      else if (on == 0 && error == 1)
109.      {
110.          error = 0;
111.      }
112.  }
```

Данные алгоритмы в дальнейшем будут интегрированы в машину состояний и рабочий код для функционирования системы.

4.4. Элементы логического контроля

4.4.1. Переход в машину состояний

В условии задания необходимо моделировать объект регулирования и систему управления с разными частотами обновления. В математической среде моделирования эффективным средством реализации данного требования, максимально приближенным к реальной программной реализации, является машина состояний

Машина состояний позволяет четко разделить алгоритм на дискретные этапы (состояния) и определить условия перехода между ними. Это позволяет достичь структурированной и наглядной реализации математических систем.

Уже разработанные программные коды (листинг 1-3) перенесем в машину состояний. На рисунках 11,12 представлены алгоритмы для ПИ-регуляторов. На рисунках 13,14 представлены алгоритмы для апериодического звена первого порядка. На рисунках 15,16 представлены алгоритмы для интегрального звена.

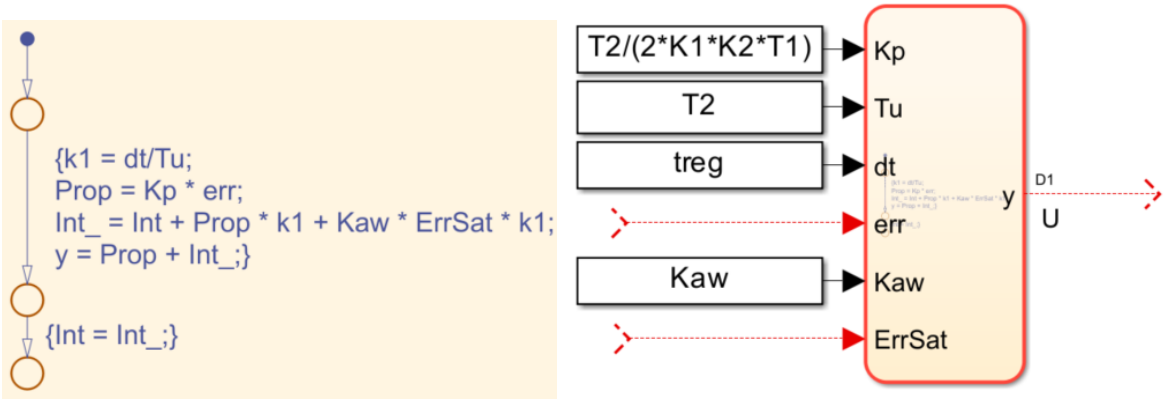


Рисунок 11 – Внешний вид машины состояний для ПИ-регулятора

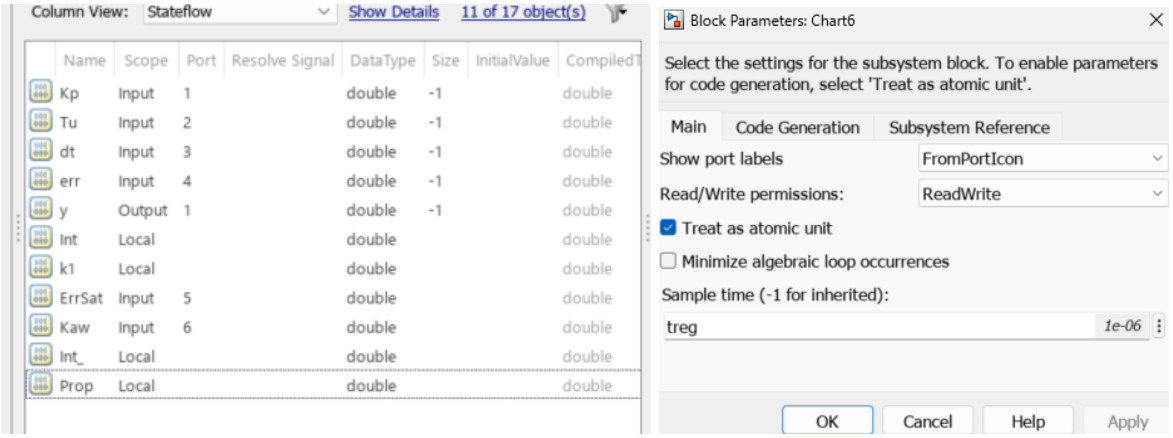


Рисунок 12 – Настройка блока машины состояний для ПИ-регулятора

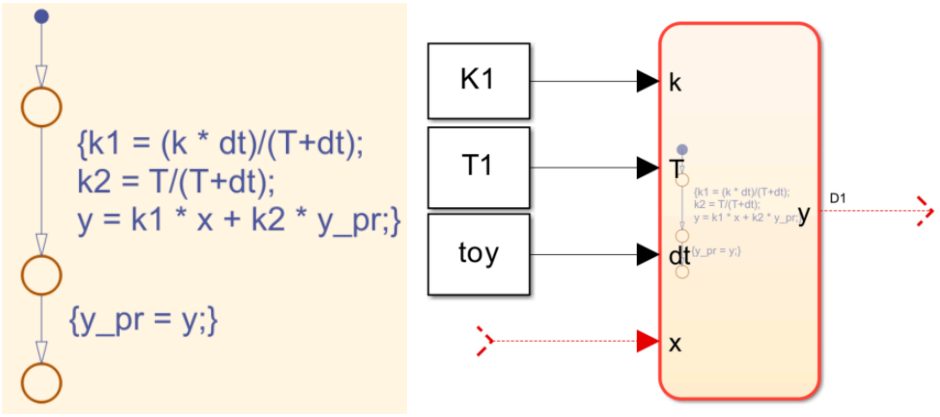


Рисунок 13 – Внешний вид машины состояний для апериодического звена первого порядка

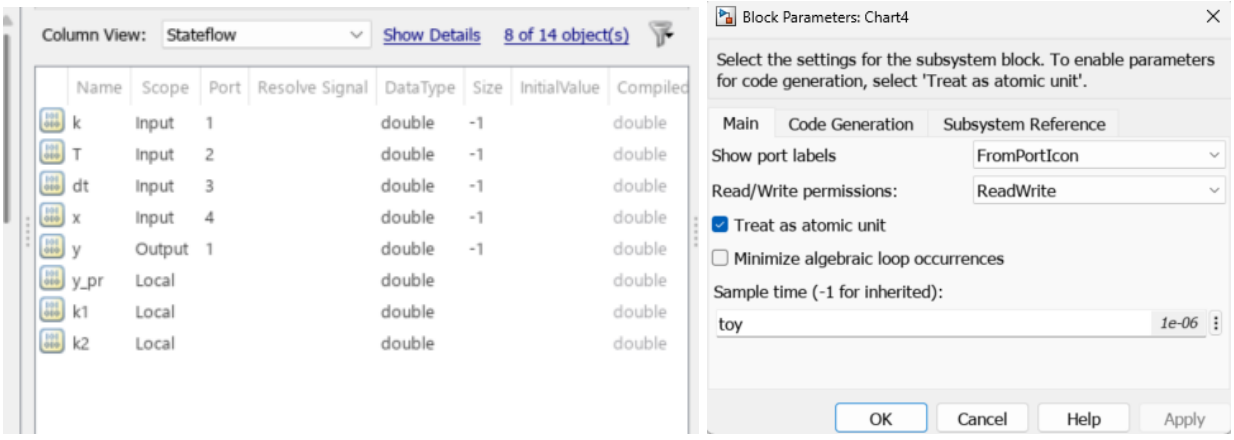


Рисунок 14 – Настройка блока машины состояний для апериодического звена первого порядка

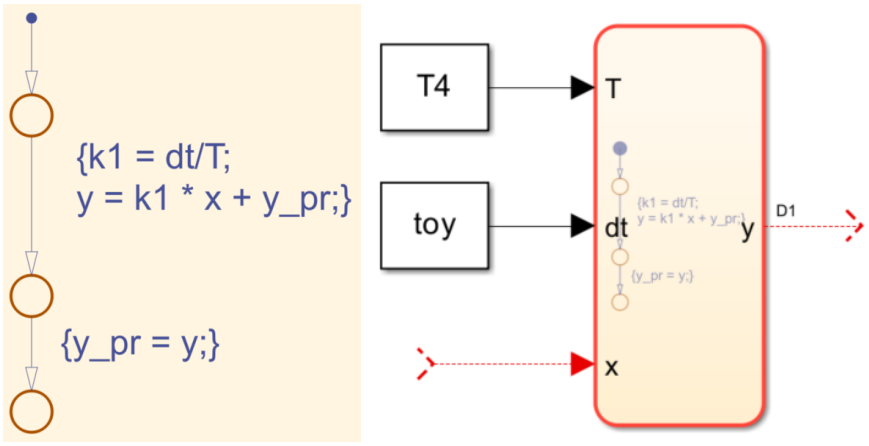


Рисунок 15 – Внешний вид машины состояний для интегрального звена

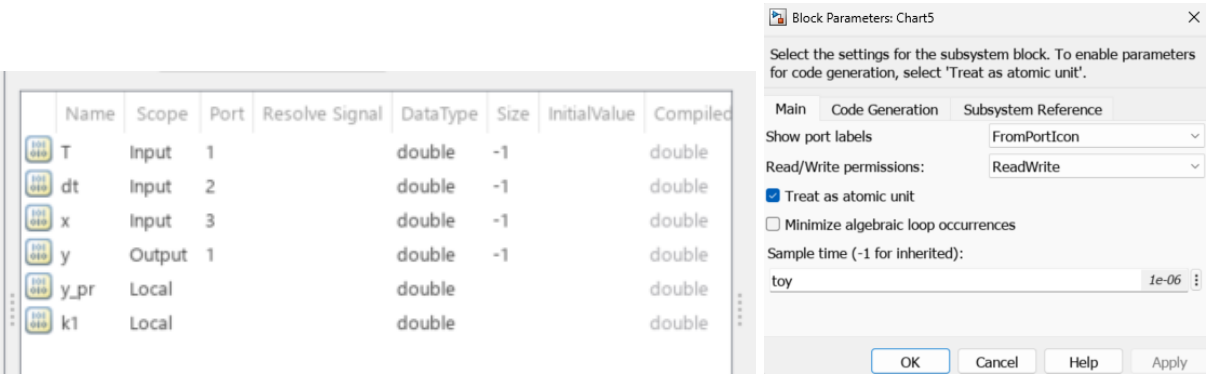


Рисунок 16 – Настройка блока машины состояний для интегрального звена

Дополнительно стоит сказать, что для моделирования ПИ-регулятора без механизма анти-виндапа стоит использовать зануление коэффициента *ErrSat*.

Далее произведем реализацию дополнительных условий (ограничения) в формате машины состояния (защитный механизм также является ограничением). На рисунках 17, 18 представлена машина состояний для функции насыщения с возможностью анти-виндапа (можно не использовать).

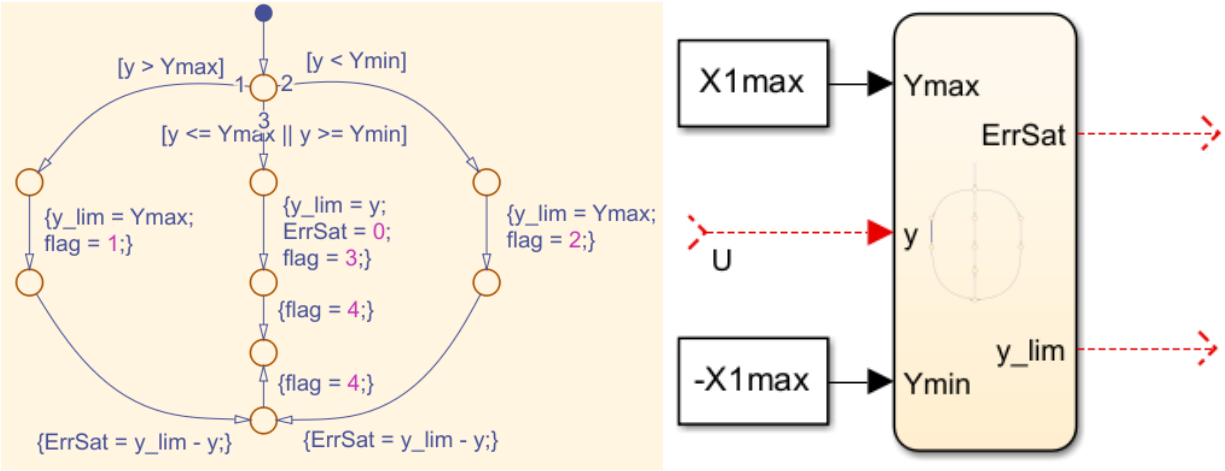


Рисунок 17 – Внешний вид машины состояний для функции насыщения с механизмом анти-виндапа

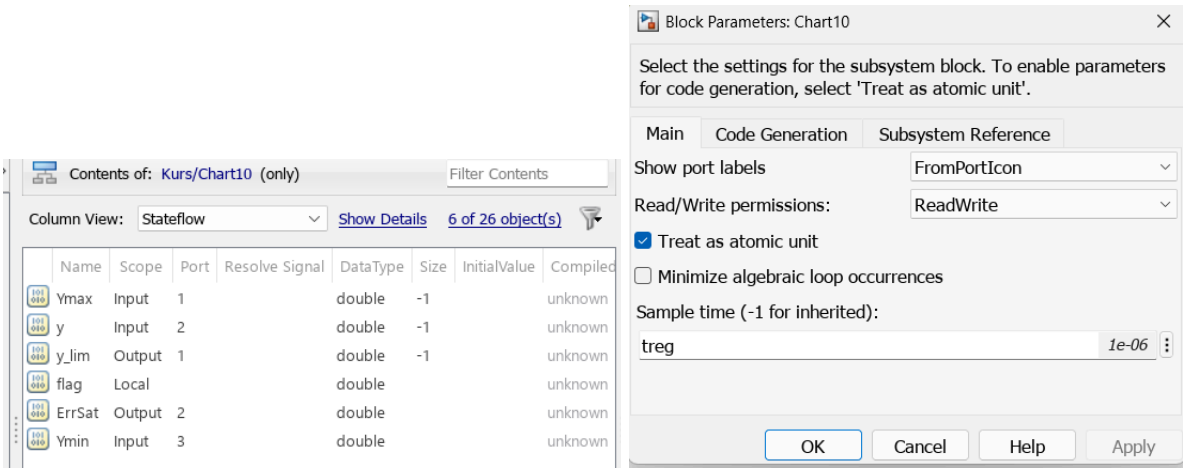


Рисунок 18 – Настройка блока машины состояний для функции насыщения с механизмом анти-виндапа

Конечный вариант схемы всей системы с использованием машины состояний представлен на рисунке 19.

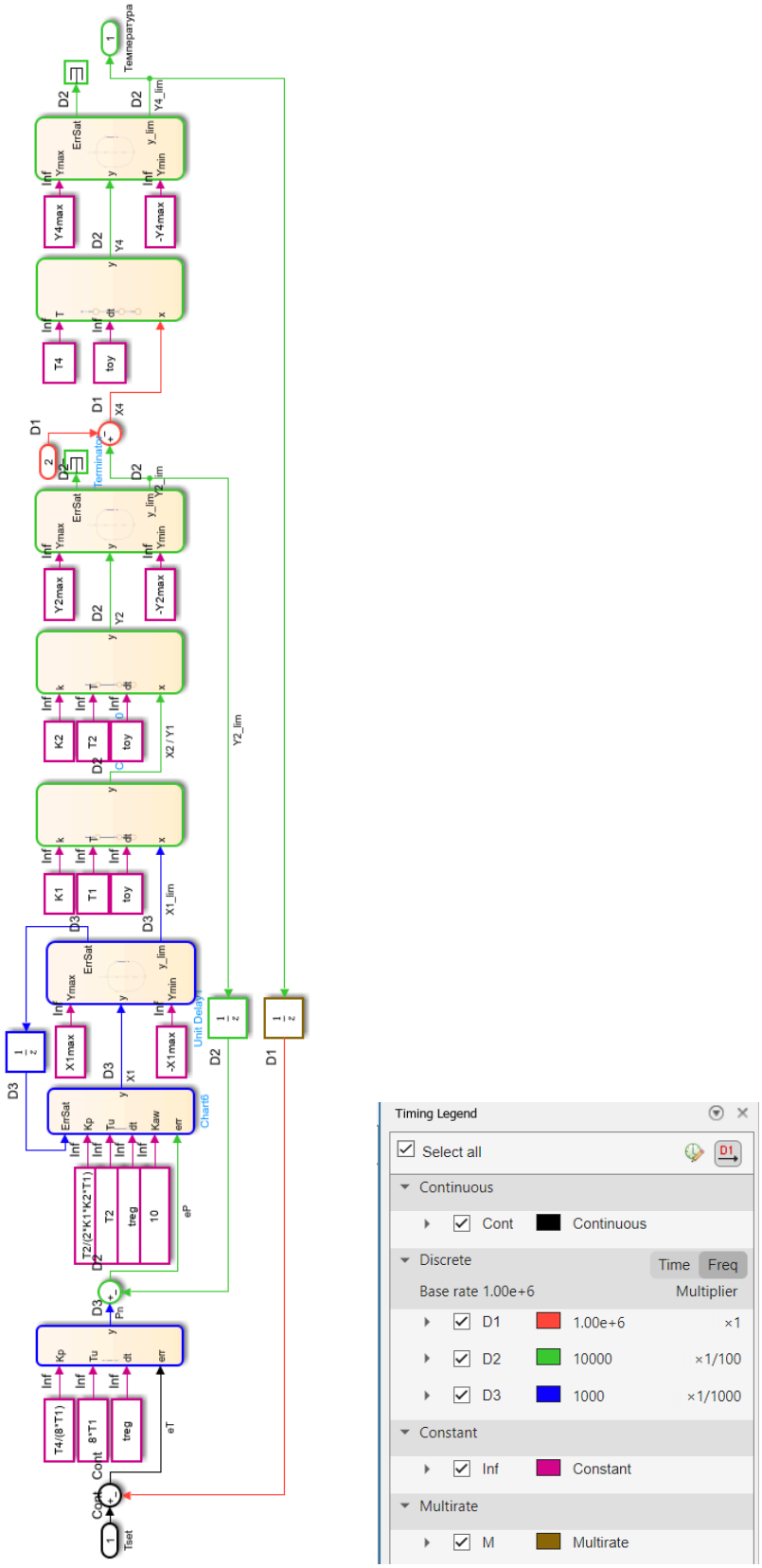


Рисунок 19 – Схема системы с использованием машины состояний

4.4.2. Машина состояния системы индикации и контроля

Для реализации системы индикации и контроля были использованы дополнительные блоки и настройки, которые позволят управлять входной температурой, включением/выключением установки и возмущением в режиме реального времени. Для разработки машины состояний был использован вариант кода из листинг 4 в доработанном варианте (см. рисунок 20-21). Настройки машины состояний представлены на рисунке 22.

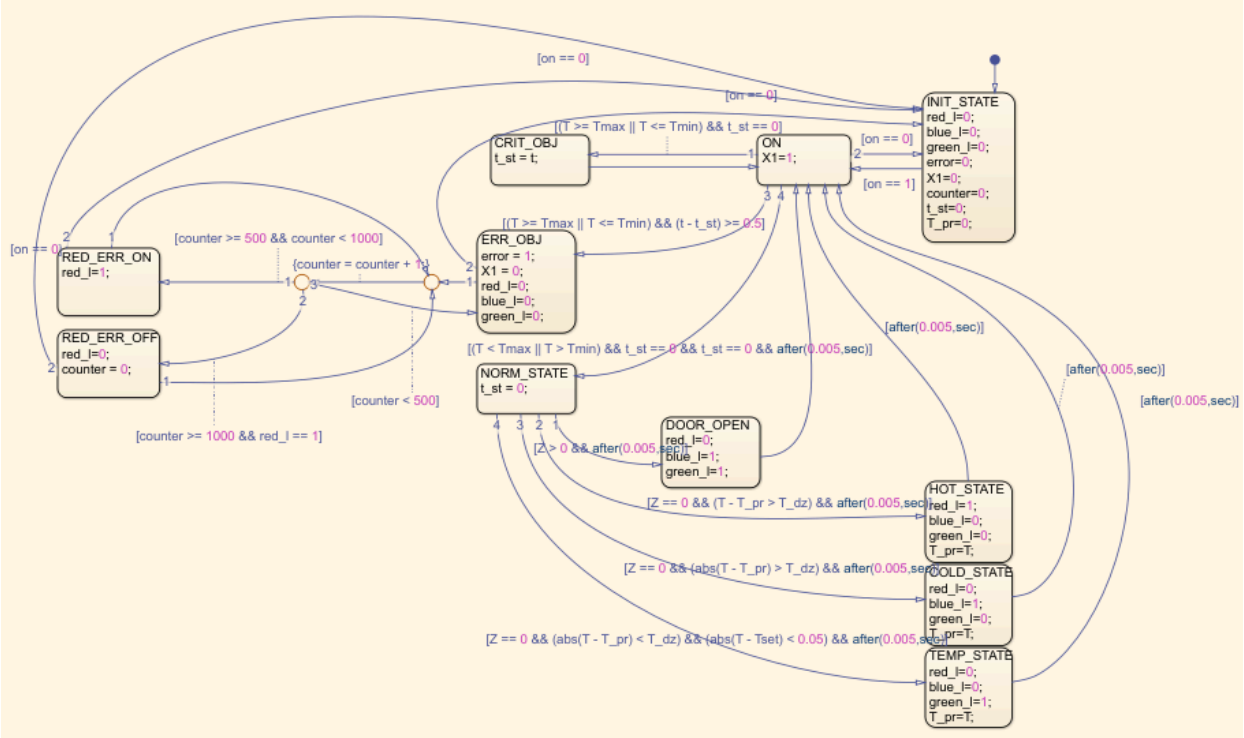


Рисунок 20 – Внутренний вид машины состояний системы индикации и контроля

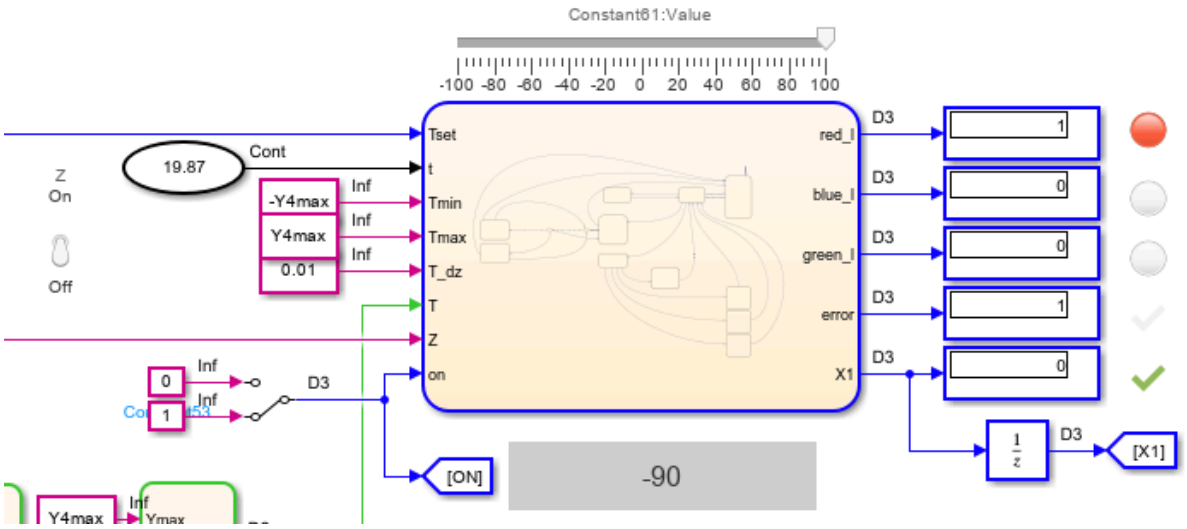


Рисунок 21 – Внешний вид машины состояний для системы индикации и контроля

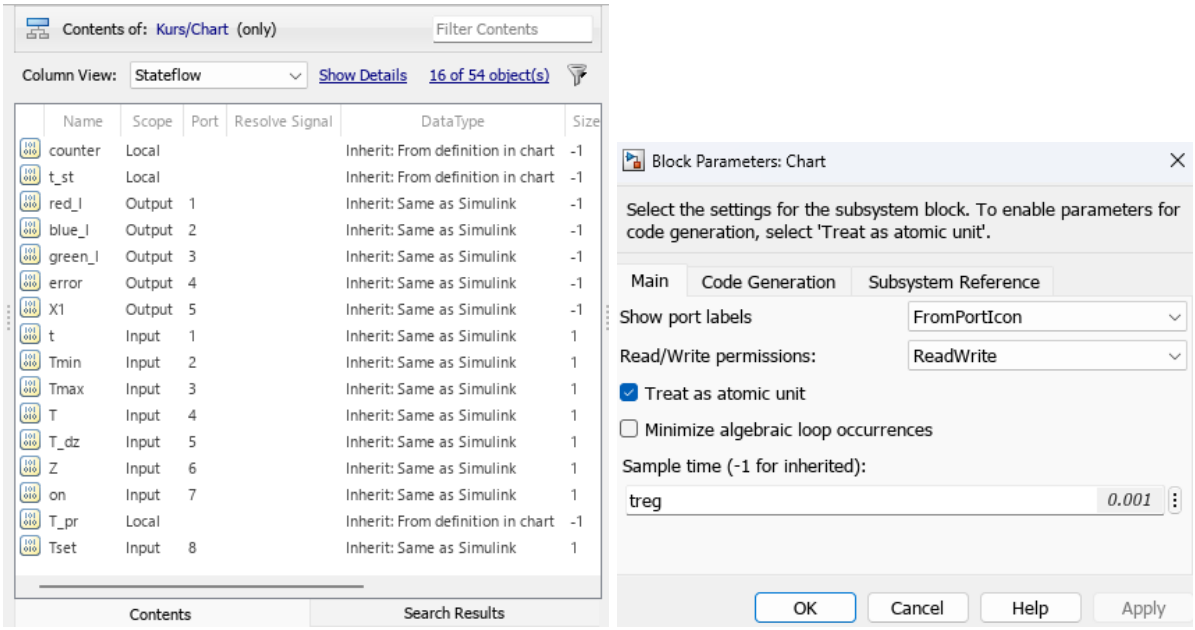


Рисунок 22 – Настройка блока машины состояний для функции насыщения с механизмом анти-виндапа

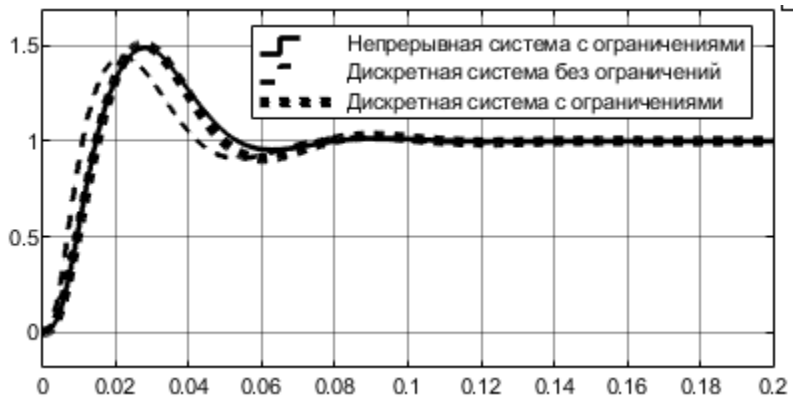
4.4.3. Тестирование разработанных алгоритмов

После нахождения математического описания системы управления, внедрения функций защиты и программной реализации математических уравнений необходимо произвести моделирование с проверкой работы. Моделирование работы будем производить в режимах, представленных в таблице 6.

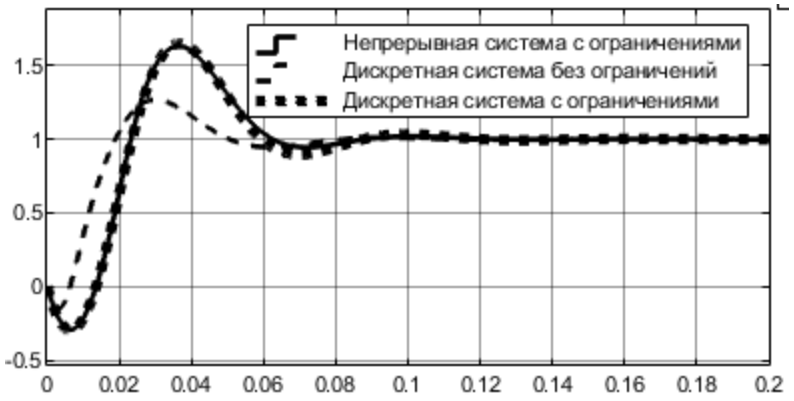
Таблица 6 — Таблица параметров для моделирования

Характер входа	Вход T_{SET}, $°C$	Время моделирования, сек	№ рис.
Ступенчатый	1	0,2	23
Ступенчатый	30	1,2	24
Линейный	$1 \cdot t$	0,1	25
Линейный	$100 \cdot t$	1,2	26

Далее представим графики после моделирования.



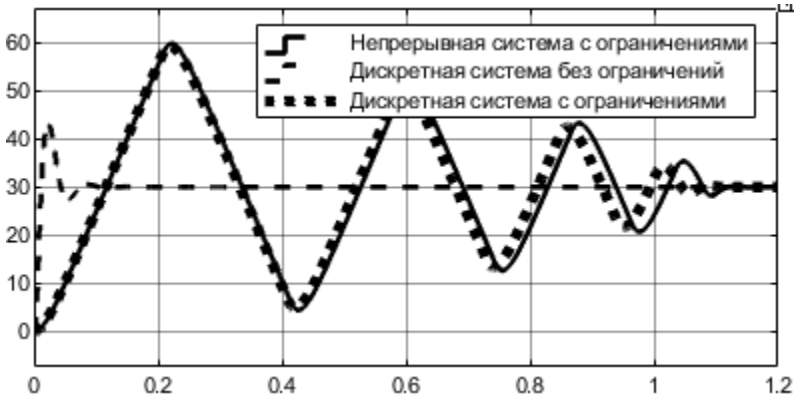
а



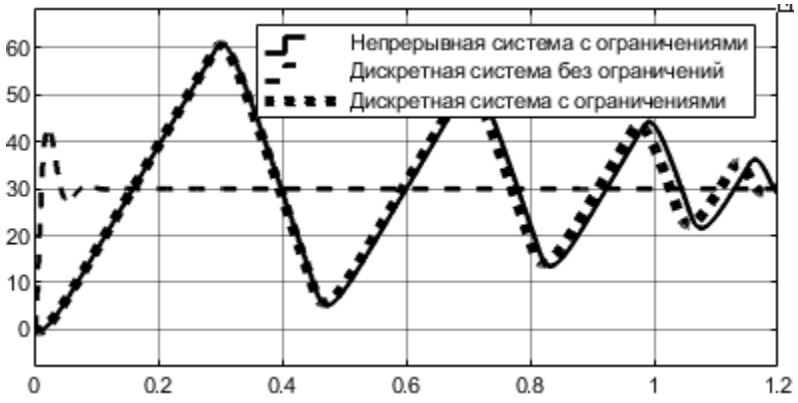
б

а – без возмущения; б – с возмущением

Рисунок 23 – Графики температуры при входных параметрах $T_{SET} = 1$



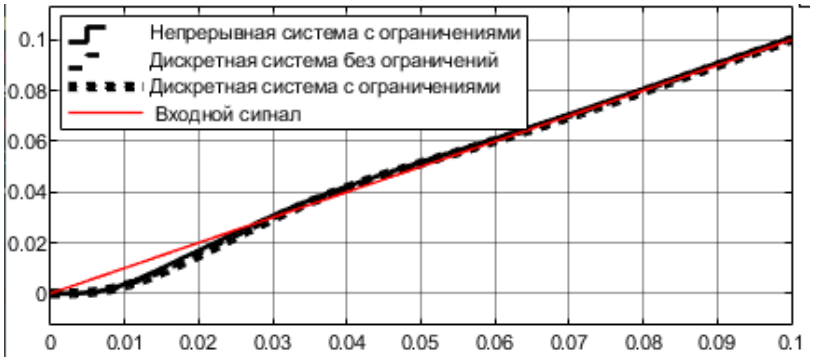
а



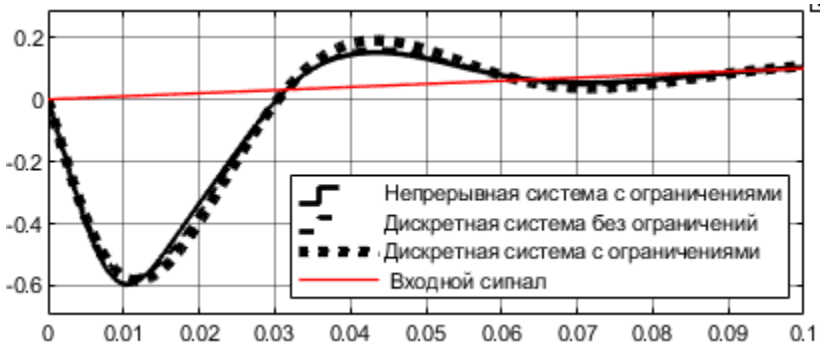
б

а – без возмущения; б – с возмущением

Рисунок 24 – Графики температуры при входных параметрах $T_{SET} = 30$



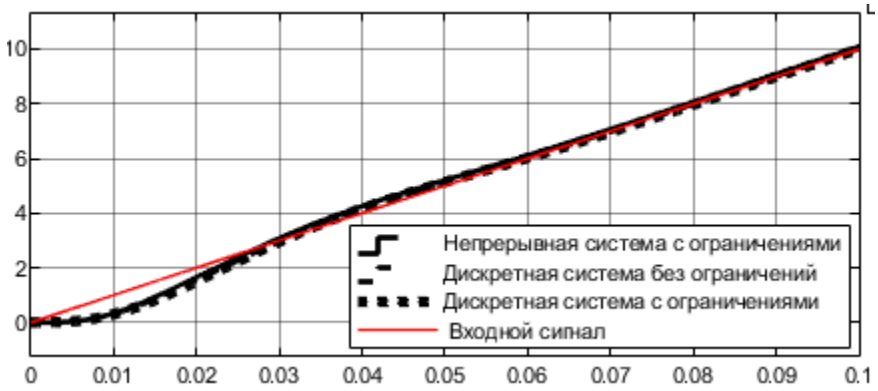
а



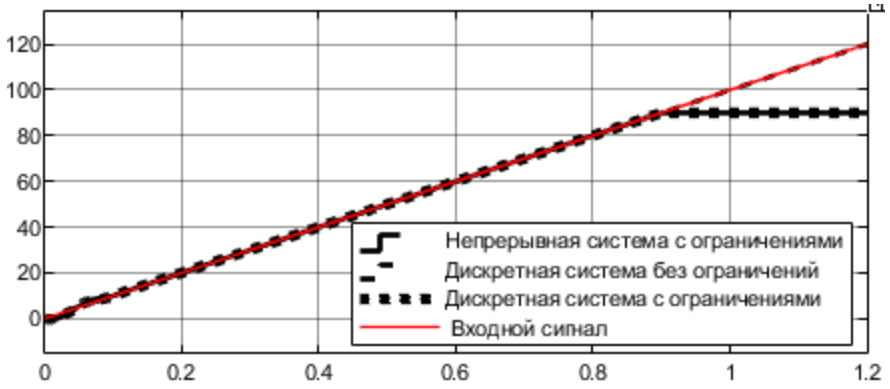
б

а – без возмущения; б – с возмущением

Рисунок 25 – Графики температуры при входных параметрах $T_{SET} = 1 \cdot t$



а



б

а – без возмущения; б – с возмущением

Рисунок 26 – Графики температуры при входных параметрах $T_{SET} = 100 \cdot t$

Тестирование работы алгоритма индикации и контроля происходило в режиме настоящего времени. Все видео загружены в GitHub.

4.5. Программные алгоритмы для микроконтроллера

4.5.1. Распределение внешних портов микроконтроллера

Далее необходимо произвести перенос разработанного алгоритма на микроконтроллер. На рисунке 27 представлен вид STM32F446RET6 с распределенными портами под каждую задачу.

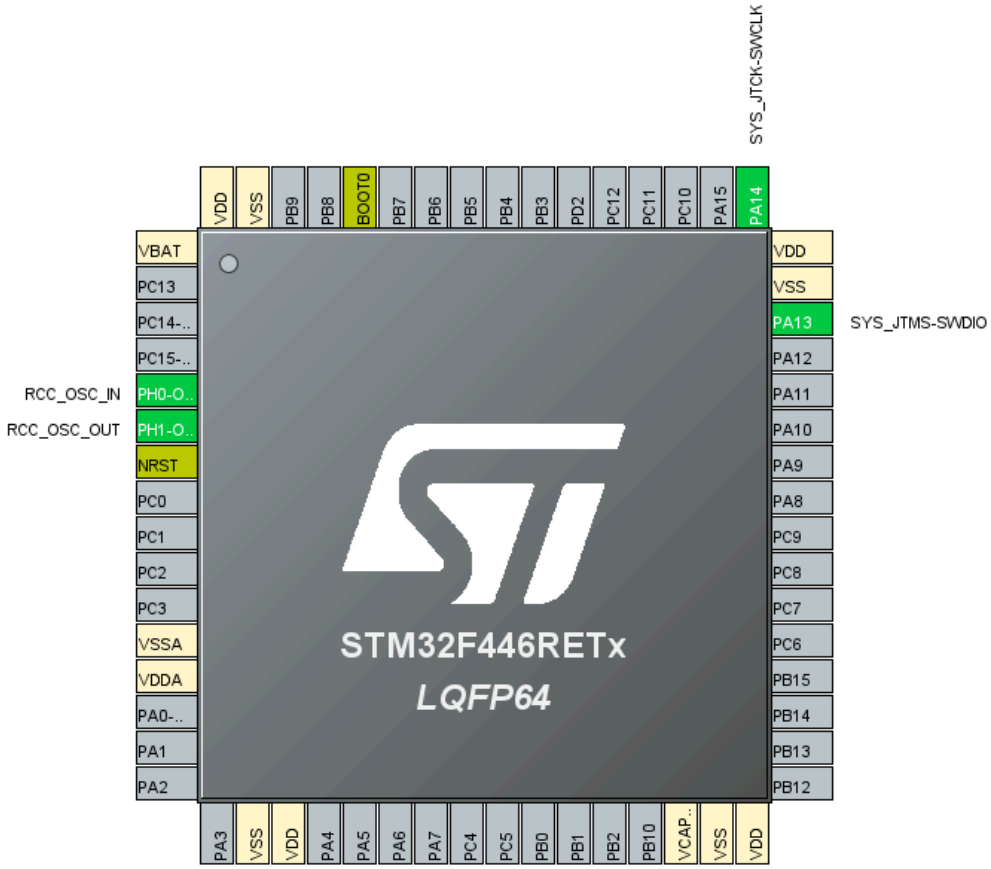


Рисунок 27 – Распределение внешних портов микроконтроллера STM32F446RET6

Также в дополнении используется два таймера для расчёта всей схемы (TIM8 – система управления, TIM10 – объект регулирования). По условию задания необходимости переноса функций индикации и управления (ручная настройка входной температуры). В приложении А представлен код реализации всего разработанного алгоритма на языке С.

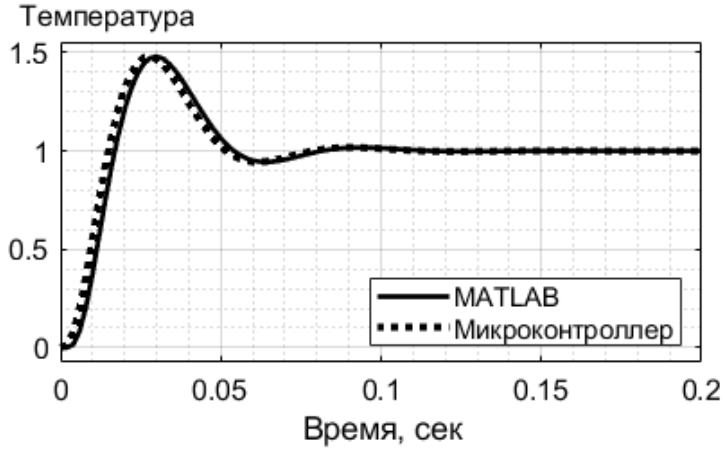
4.5.2. Тестирование работы разработанного алгоритма

Моделирование работы будем производить в режимах, представленных в таблице 7.

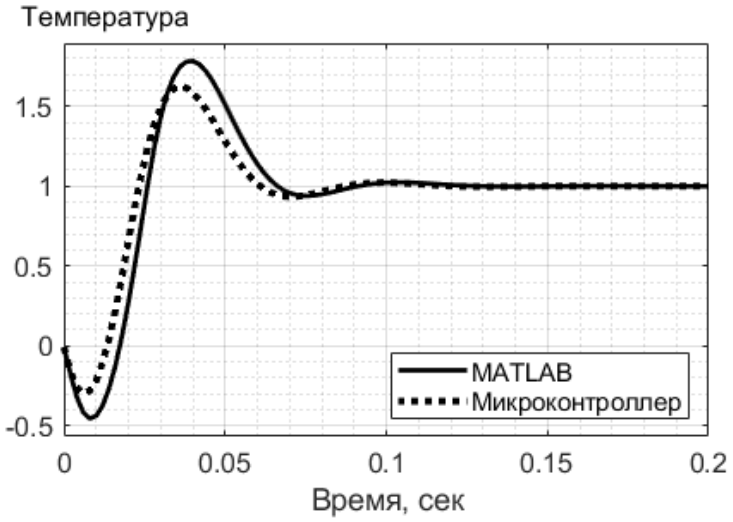
Таблица 7 — Таблица параметров для моделирования

Характер входа	Вход T_{SET}, $°C$	Время моделирования, сек	№ рисунка
Ступенчатый	1	0,2	28
Ступенчатый	30	1,2	29
Линейный	$1 \cdot t$	0,2	30
Линейный	$100 \cdot t$	1,2	31

Далее представим графики после моделирования в математической среде моделирования и расчёте на микроконтроллере.



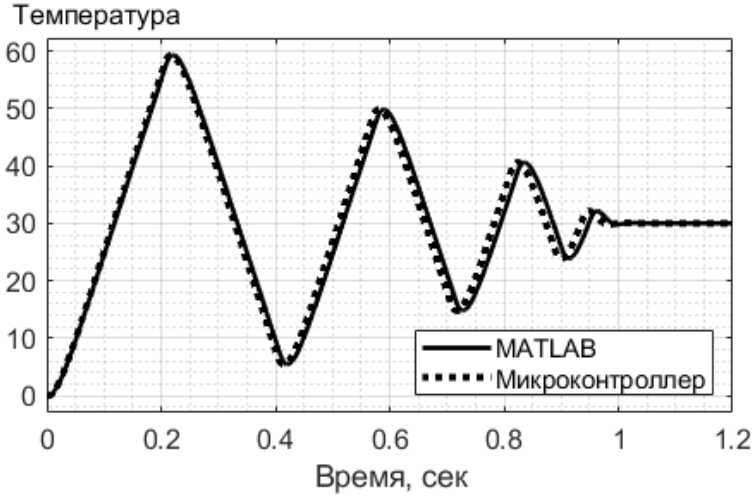
а



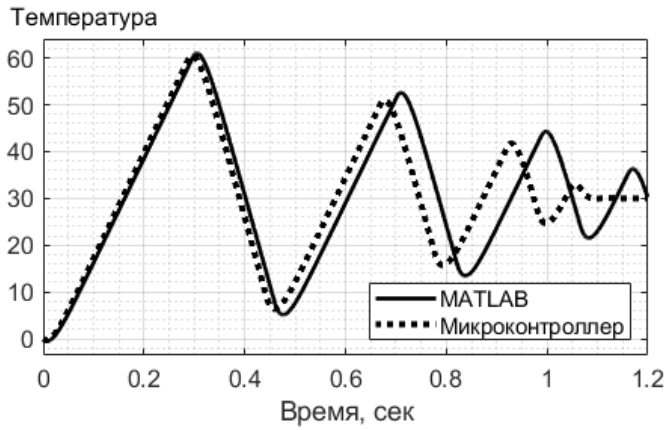
б

а – без возмущения; б – с возмущением

Рисунок 28 – Графики температуры при входных параметрах $T_{SET} = 1$



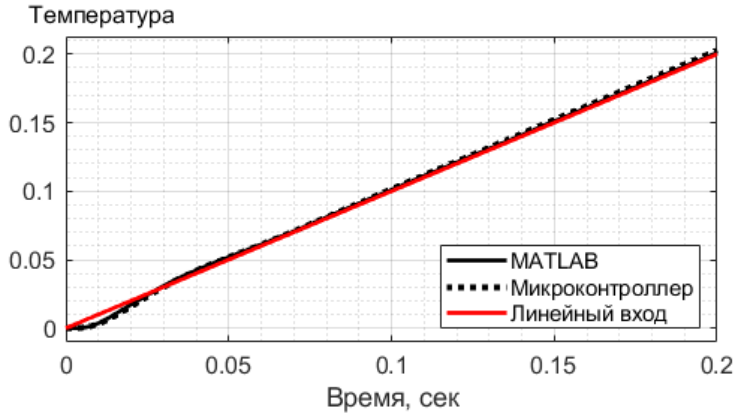
а



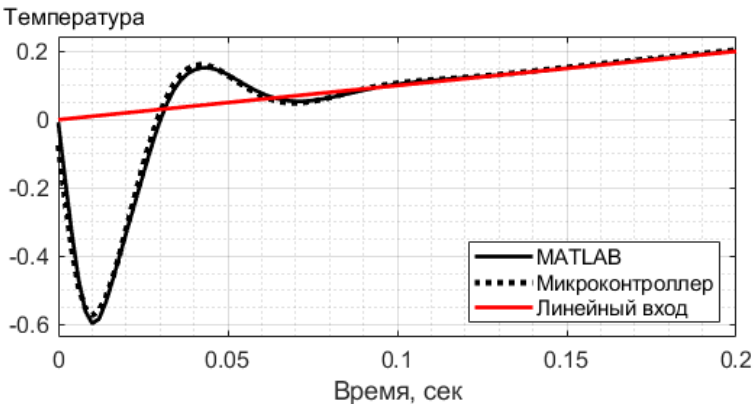
б

а – без возмущения; б – с возмущением

Рисунок 29 – Графики температуры при входных параметрах $T_{SET} = 30$



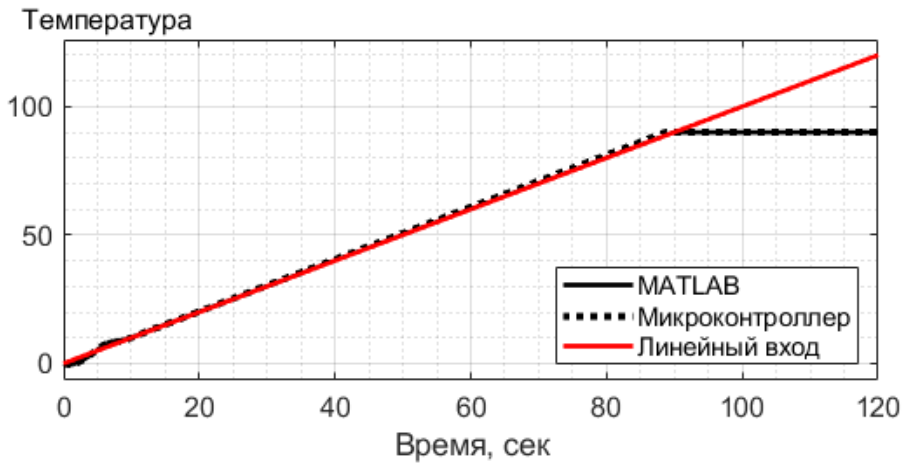
а



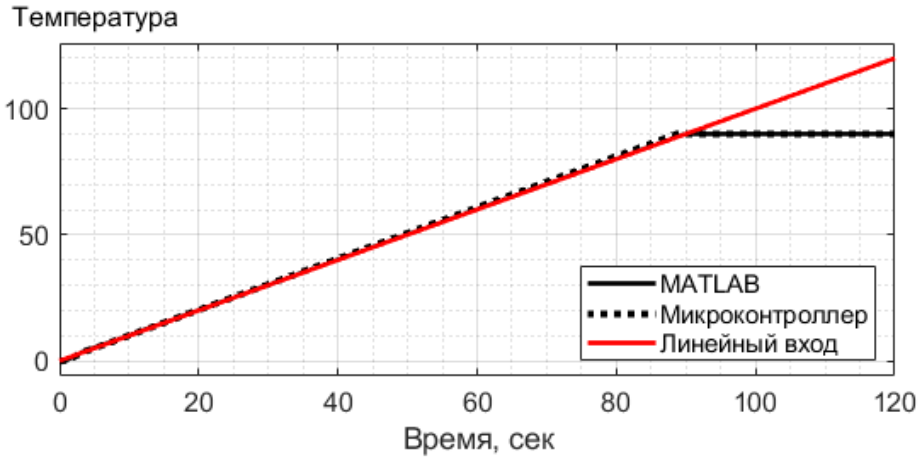
б

а – без возмущения; б – с возмущением

Рисунок 30 – Графики температуры при входных параметрах $T_{SET} = 1 \cdot t$



а



б

а – без возмущения; б – с возмущением

Рисунок 31 – Графики температуры при входных параметрах $T_{SET} = 100 \cdot t$

По рисункам можно сделать вывод, что динамика процессов практически совпадает с большой точностью, однако присутствует запаздывание со стороны микроконтроллера. Однако, такое же запаздывание наблюдалось при переходе с непрерывных функций к дискретным. Данное запаздывание может быть связано с тем, что микроконтроллер считывает данные с запаздыванием (проверено на переменной счётчике). Из-за чего с течением времени запаздывание нарастает.

5. Заключение

В ходе выполнения курсовой работы по разработке встраиваемой системы управления холодильной установкой были последовательно выполнены следующие этапы:

- описание объекта управления – проведена аналогия с холодильной установкой на элементах Пельтье, описаны все сигналы и функции системы, проанализированы статические и динамические характеристики объекта регулирования.

- разработка системы управления – составлены структурная и функциональная схемы системы управления с учётом заданного функционала, определены характеристики всех сигналов.

- система автоматического управления – построена математическая модель объекта регулирования, выполнен синтез регуляторов (на модульный оптимум для внутреннего контура и на симметричный оптимум для внешнего контура температуры), введены необходимые ограничения и разработана программная реализация алгоритмов.

- логический контроль и индикация – все алгоритмы переведены в формат машины состояний, разработана система индикации состояния. Проведено сравнение непрерывной и дискретной реализаций, показавшее незначительное запаздывание дискретной системы при сохранении динамического подобия.

- программная реализация на микроконтроллере – выполнена настройка микроконтроллера и написан программный код для моделирования системы управления. Тестирование подтвердило работоспособность системы с незначительным временным запаздыванием относительно математической модели.

Все поставленные задачи были выполнены.

Приложение А
Спецификация

Листинг А.1 – программная реализация алгоритма всей системы

```
1.  // Структура ПИ-регулятора с механизмом антивиндапа (anti-windup)
2.  typedef struct
3.  {
4.      float dt;       // период дискретизации системы управления
5.      float Kp;       // коэффициент пропорциональной составляющей
6.      float Ki;       // коэффициент интегральной составляющей
7.      float y;        // выходное значение регулятора (ограниченное)
8.      float y_unl;    // выходное значение регулятора (неограниченное)
9.      float Int;      // текущее значение интегратора
10.     float Int_pr;   // предыдущее значение интегратора
11.     float err;      // текущая ошибка регулирования
12.     float u;        // задающее воздействие (вход регулятора)
13.     float Kaw;      // коэффициент компенсации антивиндапа
14.     float ErrSat;   // ошибка насыщения для компенсации
15.     float Ymax;     // максимальное ограничение выхода регулятора
16.     float Ymin;     // минимальное ограничение выхода регулятора
17.     float counter;  // счетчик
18. } Regulator;
19.
20. // Структура апериодического звена первого порядка
21. typedef struct
22. {
23.     float dt;       // период дискретизации объекта регулирования
24.     float K;        // коэффициент усиления
25.     float T;        // постоянная времени
26.     float k1;       // коэффициент для дискретной реализации
27.     float k2;       // коэффициент для дискретной реализации
28.     float y;        // выходное значение звена
29.     float y_pr;     // предыдущее выходное значение
30.     float Z;        // внешнее возмущение
31. } Object;
32.
33. // Структура интегратора
34. typedef struct
35. {
36.     float dt;       // период дискретизации объекта регулирования
37.     float K;        // коэффициент усиления
38.     float T;        // постоянная времени
39.     float k1;       // коэффициент для дискретной реализации
40.     float y;        // выходное значение интегратора
41.     float y_pr;     // предыдущее выходное значение
```

```
42.  } Integrator;
43.
44.  // Структура функции насыщения
45.  typedef struct
46.  {
47.      float Ymax;     // верхний предел насыщения
48.      float Ymin;     // нижний предел насыщения
49.  } Saturation;
50.
51.  // Объявление глобальных переменных
52.  Regulator piP; // регулятор ПИ для контура мощности
53.  Regulator piT; // регулятор ПИ для контура температуры
54.  Object f1;     // первый объект (преобразователь напряжения)
55.  Object f2;     // второй объект (элементы Пельтье)
56.  Integrator F;  // третий объект (модель тепловой инерции)
57.  Saturation S2; // функция насыщения для второго объекта
58.  Saturation S4; // функция насыщения для третьего объекта
59.
60.  void init_str()
61.  {
62.    // Параметры первого объекта (преобразователь напряжения)
63.    f1.dt = 0.0001;
64.    f1.K = 6.74;
65.    f1.T = 0.0013;
66.    f1.k1 = (f1.dt * f1.K) / (f1.dt + f1.T);
67.    f1.k2 = f1.T / (f1.dt + f1.T);
68.    f1.y = 0.0;
69.    f1.y_pr = 0.0;
70.
71.    // Параметры второго объекта (элементы Пельтье)
72.    f2.dt = f1.dt;
73.    f2.K = 3.12;
74.    f2.T = 0.0165;
75.    f2.k1 = (f2.dt * f2.K) / (f2.dt + f2.T);
76.    f2.k2 = f2.T / (f2.dt + f2.T);
77.    f2.y = 0.0;
78.    f2.y_pr = 0.0;
79.    S2.Ymax = 250;
80.    S2.Ymin = -250;
81.    f2.Z = 0.0;
82.
83.    // Параметры третьего объекта (тепловая инерция)
84.    F.dt = f1.dt;
85.    F.K = 1.0;
86.    F.T = 0.207;
```

```
87.     F.k1 = (F.dt * F.K) / F.T;
88.     F.y = 0.0;
89.     F.y_pr = 0.0;
90.     S4.Ymax = 90.0;
91.     S4.Ymin = -90.0;
92.
93.     // Параметры ПИ-регулятора для контура мощности
94.     piP.dt = 0.001;
95.     piP.Kp = f2.T / (2 * f1.T * f1.K * f2.K);
96.     piP.Ki = 1 / f2.T;
97.     piP.y = 0.0;
98.     piP.y_unl = 0.0;
99.     piP.Int = 0.0;
100.    piP.Int_pr = 0.0;
101.    piP.err = 0.0;
102.    piP.Kaw = 10.0;
103.    piP.ErrSat = 0.0;
104.    piP.Ymax = 3;
105.    piP.Ymin = -3;
106.
107.    // Параметры ПИ-регулятора для контура температуры
108.    piT.dt = 0.001;
109.    piT.Kp = F.T / (8 * f1.T);
110.    piT.Ki = 1.0 / (8 * f1.T);
111.    piT.y = 0.0;
112.    piT.Int = 0.0;
113.    piT.Int_pr = 0.0;
114.    piT.err = 0.0;
115.    piT.u = 0.0;
116.    piT.counter = 0.0;
117.  }
118.
119.  // Функция насыщения
120.  float sat_func(Saturation* S, float x)
121.  {
122.    if (S->Ymax < x)
123.    {
124.            return S->Ymax;
125.    }
126.    else if (S->Ymin > x)
127.    {
128.            return S->Ymin;
129.    }
130.    else
131.    {
```

```
132.            return x;
133.    }
134. }
135.
136. // Функция ПИ-регулятора без антивиндапа
137. float reg_func(Regulator* pi, float err)
138. {
139.    // Вычисление нового значения интегральной составляющей
140.    pi->Int = pi->Int_pr + pi->Kp * err * pi->Ki * pi->dt;
141.    pi->Int_pr = pi->Int;
142.
143.    // Выход регулятора
144.    pi->y = pi->Kp * err + pi->Int;
145.
146.    return pi->y;
147. }
148.
149. float reg_func_sat(Regulator* pi, float err)
150. {
151.    // Вычисление интегральной составляющей с компенсацией антивиндапа
152.    pi->Int = pi->Int_pr + pi->Kp * err * pi->Ki * pi->dt + pi->ErrSat * pi->Kaw *
       pi->Ki * pi->dt;
153.    pi->Int_pr = pi->Int;
154.
155.    // Вычисление неограниченного выхода регулятора
156.    pi->y_unl = pi->Kp * err + pi->Int;
157.
158.    // Применение функции насыщения и вычисление ошибки насыщения
159.    if (pi->y_unl > pi->Ymax) {
160.            pi->y = pi->Ymax;
161.            pi->ErrSat = pi->y - pi->y_unl;
162.    } else if (pi->y_unl < pi->Ymin) {
163.            pi->y = pi->Ymin;
164.            pi->ErrSat = pi->y - pi->y_unl;
165.    } else {
166.            pi->y = pi->y_unl;
167.            pi->ErrSat = 0;
168.    }
169.
170.    return pi->y;
171. }
172.
173. // Функция апериодического звена первого порядка
174. float obj_func(Object* f, float x)
175. {
```

```
176.    f->y = f->k1 * x + f->k2 * f->y_pr;
177.
178.    f->y_pr = f->y;
179.
180.      return f->y;
181. }
182.
183. // Функция интегратора
184. float obj_int(Integrator* f, float x)
185. {
186.    f->y = f->y_pr + f->k1 * x;
187.
188.    f->y_pr = f->y;
189.
190.      return f->y;
191. }
192.
193. // Обратный вызов по прерыванию от таймера
194. void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
195.     // Обработка прерывания от TIM8 (1 кГц)
196.    if (htim->Instance == TIM8) {
197.              // Вычисление ошибки регулирования температуры
198.              piT.err = piT.u - F.y;
199.              // Вычисление выходного значения регулятора температуры
200.              piT.y = reg_func(&piT, piT.err);
201.              // Вычисление ошибки регулирования мощности
202.              piP.err = piT.y - f2.y;
203.              // Вычисление выходного значения регулятора мощности с
      антивиндапом
204.              piP.y = reg_func_sat(&piP, piP.err);
205.    }
206.
207.    // Обработка прерывания от TIM10 (10 кГц)
208.    if (htim->Instance == TIM10) {
209.              // Установка уставки температуры (линейный вход)
210.              //piT.counter++;
211.              //piT.u = piT.counter / 1000.0;
212.
213.              // Установка уставки температуры (константа)
214.              piT.u = 1.0;
215.
216.              // Установка внешнего возмущения
217.              f2.Z = 0.0;
218.              // Вычисление результата первого апериодического звена
219.              f1.y = obj_func(&f1, piP.y);
```

```
220.            // Вычисление результата второго апериодического звена с
    функцией насыщения
221.            f2.y = sat_func(&S2, obj_func(&f2, f1.y));
222.            // Вычисление результата интегрального звена с функцией
    насыщения
223.            F.y = sat_func(&S4, obj_int(&F, f2.y - f2.Z));
224.            }
225. }
```