Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное  
учреждение высшего образования  
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ЛАБОРАТОРНАЯ РАБОТА №8  
**«Системы с автоматическим управлением»**

по курсу: **«**Введение в инженерную деятельность**»**

Выполнил  
студент группы КТбо1-7 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. А. Бекезин

Принял  
ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. С. Лихтин

СОдержание

[Введение 3](#_Toc103894329)

[Основная часть 4](#_Toc103894330)

[Заключение 8](#_Toc103894331)

Введение

Целью работы является изучение алгоритмов управления устройствами с использованием обратной связи.

Задачами работы являются:

1. Настроить стабилизатор напряжения.
2. Запрограммировать и настроить PID регулятор;
3. Написать программу для автоматического управления стабилизатором напряжения.

Основная часть

В лабораторной работе №5 мы уже работали с электродвигателем. Сейчас же мы подключили его к порту Voltage Regulator. Далее используя PID функцию, мы крутили потенциометр, а скорость вращения я мотора менялась. Этого удалось достичь благодаря данному коду

#include <iostream>

#include "stm32f4xx.h"

float Reg\_P = 0.1; // Коэффициент усиления пропорциональной составляющей

float Reg\_I = 0.05; // Коэффициент усиления интегрирующей составляющей

float Reg\_D = 0.02; // Коэффициент усиления дифференцирующей составляющей

int mV;int mA;

int mD;

void SetAltFunc(GPIO\_TypeDef\* Port, int Channel, int AF)

{

 Port->MODER &= ~(3<<(2\*Channel)); // Сброс режима

 Port->MODER |= 2<<(2\*Channel); // Установка альт. Режима

 if(Channel<8) // Выбор регистра зависит от номера контакта

 {

 Port->AFR[0] &= ~(15<<4\*Channel); // Сброс альт. функции

 Port->AFR[0] |= AF<<(4\*Channel); // Установка альт. функции

 }

 else

 {

 Port->AFR[1] &= ~(15<<4\*(Channel-8)); // Сброс альт. функции

 Port->AFR[1] |= AF<<(4\*(Channel-8)); // Установка альт. функции

 }

}

int AnalogRead(int N) // Функция принимает номер канала для преобразования

{

 ADC1->SQR3 = N; // Выбран полученный из аргумента канал

 for(int a=0; a<100; a++) { asm("NOP"); } // Ожидать больше 100 тактов

 ADC1->CR2 |= ADC\_CR2\_SWSTART; // Начать преобразование

 while(!(ADC1->SR & ADC\_SR\_EOC)) { asm("NOP"); } // Ждать установки бита конца операции

 return ADC1->DR; // Вернуть результат преобразования

}

int PID(int Value, int Current) // Функция ПИД регулятора

{

 static int l=0;

 static float i=0;

 float p, d;

 int e;

 e=Value-Current; // Текущая ошибка регулирования

 p=e\*Reg\_P; // Вычисление пропорциональной составляющей

 i+=e\*Reg\_I; // Вычисление интегрирующей составляющей с накоплением

 d=(e-l)\*Reg\_D; // Вычисление дифференцирующей составляющей

 l=e; // сохраняем текущую ошибку регулирования

 return (int)(p+i+d); // Возвращаем управляющее воздействие

}

extern "C" void TIM2\_IRQHandler()

{

 TIM2->SR=0;

 mV = AnalogRead(3)\*11966/4096; // Значение напряжения на выходе в милливольтах

 mA = AnalogRead(6)\*3300/4096; // Значение тока на выходе в миллиамперах

 mD=AnalogRead(13);

  GPIOF->BSRRL = GPIO\_BSRR\_BS\_11;

  GPIOF->BSRRL = GPIO\_BSRR\_BS\_13;

 for(int i=0;i<10000;i++){

 }

  GPIOF->BSRRH = GPIO\_BSRR\_BS\_11;

  GPIOF->BSRRH = GPIO\_BSRR\_BS\_13;

 for(int i=0;i<10000;i++){

 }

TIM12->CCR1=PID(mD,mV);

}

int main()

{

  RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOFEN; // Порт I задействован

  GPIOF->MODER &= ~GPIO\_MODER\_MODER11;

  GPIOF->MODER &= ~GPIO\_MODER\_MODER13;

  GPIOF->MODER |= GPIO\_MODER\_MODER11\_1;

  GPIOF->MODER |= GPIO\_MODER\_MODER13\_1;

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOHEN; // Порт H задействован

GPIOH->MODER &= ~GPIO\_MODER\_MODER12; // Сброс режима для PH12

GPIOH->MODER |= GPIO\_MODER\_MODER12\_0; // Установка режима на выход PH12

GPIOH->BSRRL = 1<<12; // Установить значение HIGH (3.3V) для PH12

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOHEN; // Порт H задействован

SetAltFunc(GPIOH, 6, 9); // Установка альт. режима AF9 для TIM12\_CH1(PH6)

RCC->APB1ENR |= RCC\_APB1ENR\_TIM12EN; // Таймер 12 задействован (APB1 x2 = 84МГц)

TIM12->CR2 = TIM\_CR2\_MMS\_0 | TIM\_CR2\_MMS\_1; // Генерация пульса сравнения

TIM12->ARR = 1000 - 1; // Диапазон сравнения 100

TIM12->PSC = (84000000/1000/42000)-1; // Задан делитель на 40кГц

TIM12->CCMR1= TIM\_CCMR1\_OC1M\_1 | TIM\_CCMR1\_OC1M\_2 | TIM\_CCMR1\_OC1PE; // Режим PWM 1 для CH1

TIM12->CCER = TIM\_CCER\_CC1E; // Выход CH1 активен

TIM12->CR1 = TIM\_CR1\_CEN; // Таймер запущен

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOCEN; // Порт C задействован

GPIOC->MODER &= ~GPIO\_MODER\_MODER3; // Сброс режима для PC3

GPIOC->MODER |= GPIO\_MODER\_MODER3; // Аналоговый вход для PC3

RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN; // АЦП задействован

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN; // Порт A задействован

GPIOA->MODER &= ~(GPIO\_MODER\_MODER3 | GPIO\_MODER\_MODER6); // Сброс режима для PA3 и PA6

GPIOA->MODER |= GPIO\_MODER\_MODER3\_0 | GPIO\_MODER\_MODER3\_1; // Аналоговый вход PA3

GPIOA->MODER |= GPIO\_MODER\_MODER6\_0 | GPIO\_MODER\_MODER6\_1; // Аналоговый вход PA6

ADC1->CR2 = ADC\_CR2\_ADON; // АЦП активен

RCC->APB1ENR |= RCC\_APB1ENR\_TIM2EN; // Задействовать 2 таймер

TIM2->PSC = ((84000000)/100/1000)-1; // Задать делитель на 100Гц

TIM2->ARR = 1000-1; // Задать диапазон переполнения

TIM2->DIER = TIM\_DIER\_UIE; // Разрешить прерывания по переполнению

TIM2->CR1 = TIM\_CR1\_CEN; // Запустить таймер

NVIC\_SetPriority(TIM2\_IRQn, 2); // Задать приоритет прерывания

NVIC\_EnableIRQ(TIM2\_IRQn); // Прерывание активировано

while(1){}

}

Затем в зависимости от положения потенциометра менялась частота мигания светодиодов. В зависимости от коэффициентов усиления пропорциональной и интегрирующей составляющей менялась длительность циклов, отвечающая за работу светодиодов.

Заключение

В ходе этой лабораторной работы мы познакомились с ПИД-регулятором. Полное название – Пропорционально-интегрально-дифференцирующий регулятор. Такое устройство используется в системах автоматического управления. Для его работы необходима обратная связь.