Лабораторная работа №4

Цель работы:

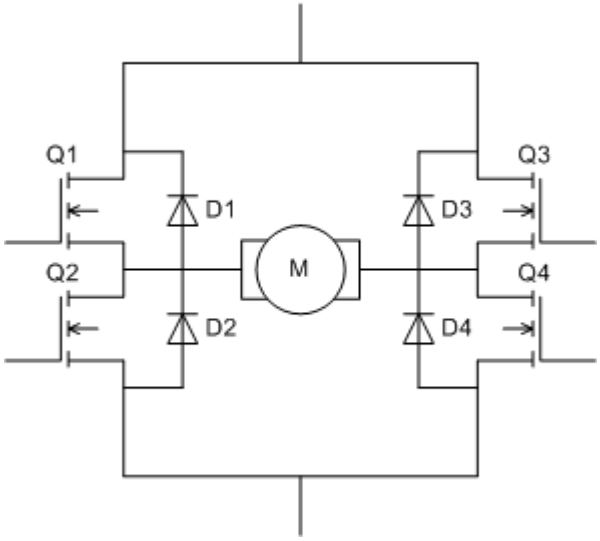
Моделирование управления двигателем постоянного тока при помощи ПИД регулятора реализованного на микроконтроллере.

Программное обеспечение:

Proteus, STM32CubeIDE.

Общие сведения:

Для управления двигателем постоянного тока существует множество вариантов, в данной работе будет рассмотрен принцип управления двигателем при помощи H-моста.



H-мост – это простая схема, состоящая из четырех ключей с нагрузкой между ними. Таким образом, внешне она напоминает букву «H», откуда и получила название.

Ключи Q1…Q4 обычно являются биполярными либо полевыми транзисторами. Если схема работает в высоковольтных сетях, то могут использоваться IGBT-транзисторы.

Диоды D1…D4 называются ограничительными диодами (catch diodes) и чаще всего являются диодами Шоттки. При применении полевых транзисторов дополнительные диоды ставить не нужно, т.к. такие транзисторы по своей структуре уже имеют паразитные диоды.

В общем случае все четыре ключа в схеме могут быть независимо переведены в состояние «включено» либо «выключено» (соответственно, транзисторы открыты либо закрыты).

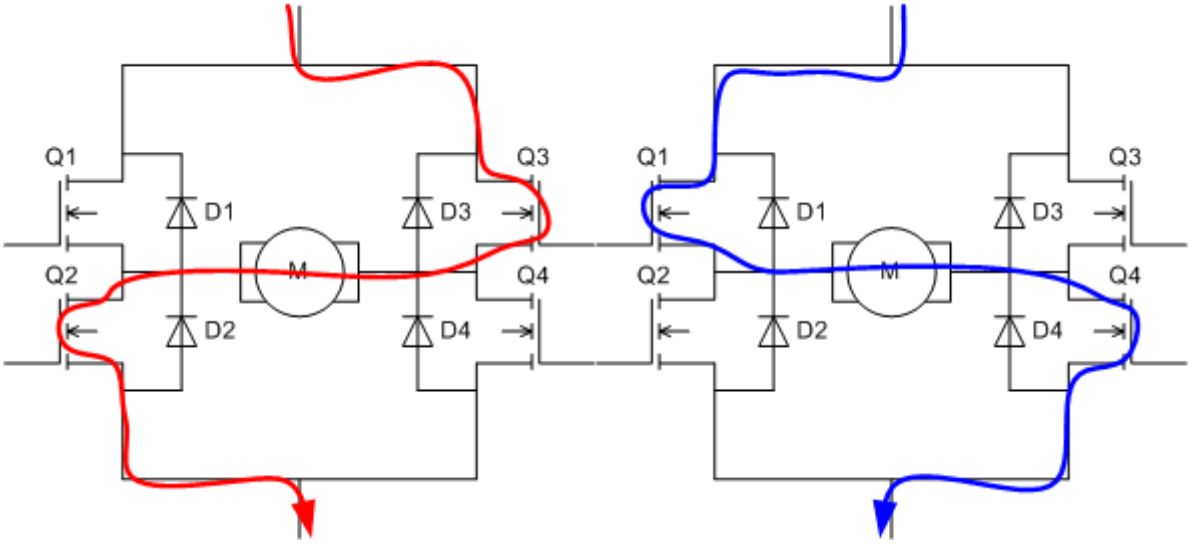
Принцип действия H-моста:

- если Q2 и Q3 открыты, а Q1 и Q4 закрыты, то левый вывод мотора на нашей схеме будет подключен к земле, а правый – к питанию. Через обмотки двигателя будет течь ток, и двигатель будет вращаться вперед (направление выбрано условно).
- если Q1 и Q4 открыты, а Q2 и Q3 закрыты, то левый вывод мотора будет подключен к линии питания, а правый – к земле. Через обмотки двигателя опять же будет течь ток, но на этот раз в противоположном направлении, и двигатель будет вращаться назад.
- Если Q2 и Q4 открыты, а Q1 и Q3 закрыты, то двигатель входит в режим электромагнитного тормоза.

Нельзя допускать открытия транзисторов Q1 и Q2 или Q3 и Q4 одновременно!

Если требуется вращение двигателя со скоростью (или моментом) меньше, чем максимальная, один из ключей управляется ШИМ-сигналом (PWM). При этом среднее

напряжение на моторе будет определяться ШИМ-сигналом, а именно отношением между временем, на протяжении которого ключ будет открыт и закрыт.



Моделирование системы.

Создание схемы в Proteus.

1. Запустите Proteus. По аналогии с лабораторно работой 3 создайте новый проект, задайте имя проекта и расположение (удобнее всего сделать отдельную папку для проекта Proteus и STM IDE), далее следует выбрать: «Create a schematic from selected template – DEFAULT», «Do not create a PCB layout», «No Firmware Project».

2. Для добавления элементов нажмите Library->Pick parts на панели инструментов (или пр.кн.мыши на области схемы Place->Component->From library). В поле поиска введите название необходимого компонента, затем выберите нужный компонент из списка и нажмите кнопку “OK”.

3. Соберите схему управления двигателем.

Для моделирования двигателя используйте «MOTOR-ENCODER», в настройках установите следующие параметры:

- Nominal Voltage: 12V;
- Coil Resistance: согласно варианту;
- Coil Inductance: согласно варианту;
- Zero Load RPM: 1000;
- Load: 10%;
- Effective Mass: 0.001;
- Pulses per Revolution: 180.

Для ключей Q1 и Q3 выберете «IRF4905», а для ключей Q2 и Q4 «IRL3705NS». Для не инвертированного управления транзисторами Q1 и Q3 необходимо добавить биполрные транзисторы BC547.

Линии от энкодера следует “подтянуть” внутренними резисторами к линии питания микроконтроллера, это будет описано далее.

Для отображения данных об угле, ошибке и значении управляющего сигнала добавьте на схему «Virtual Terminal», в настройках необходимо указать Baud Rate = 115200.

Добавьте на схему микроконтроллер STM32F401RE и подключите к нему все элементы, как показано на рисунке ниже.

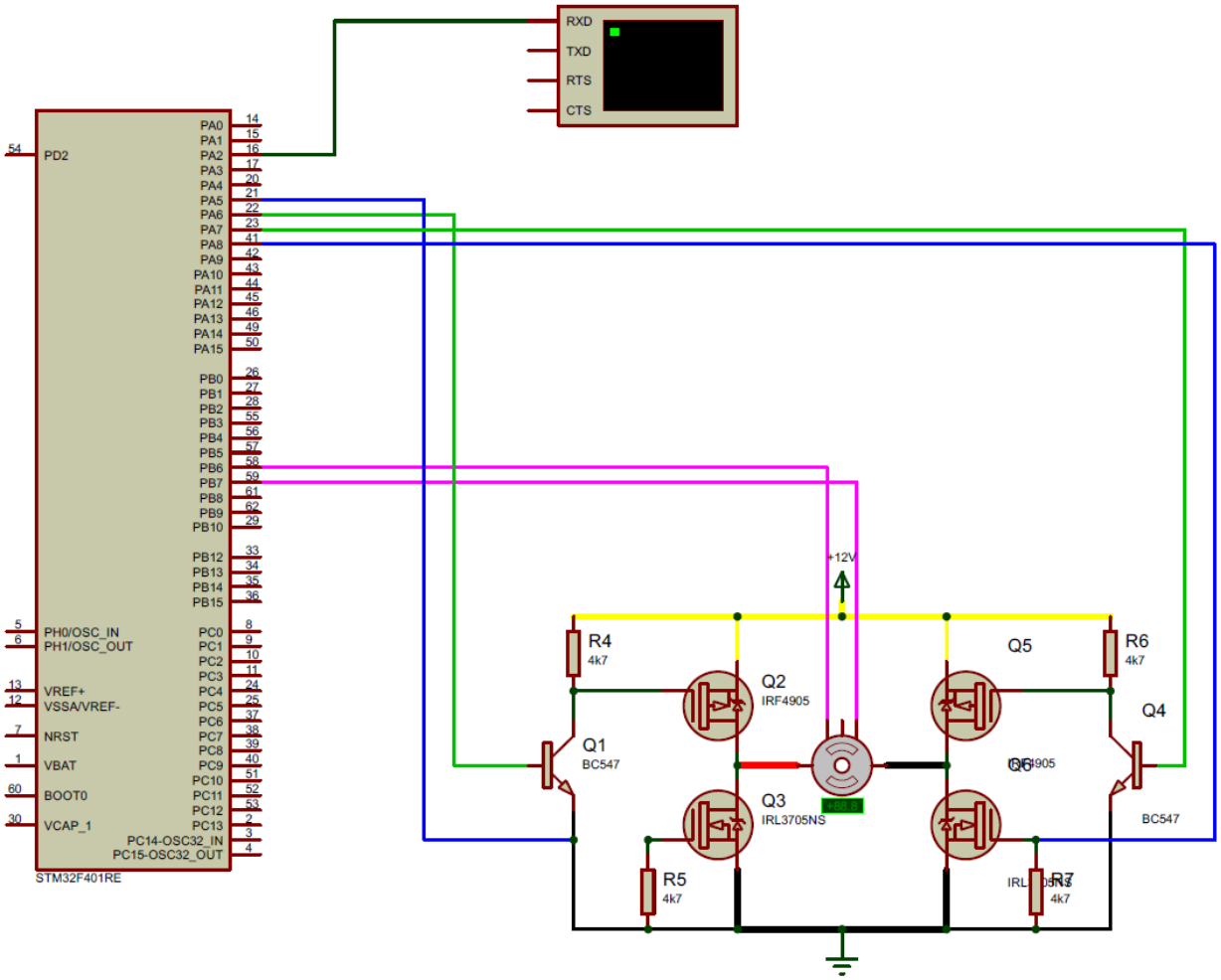
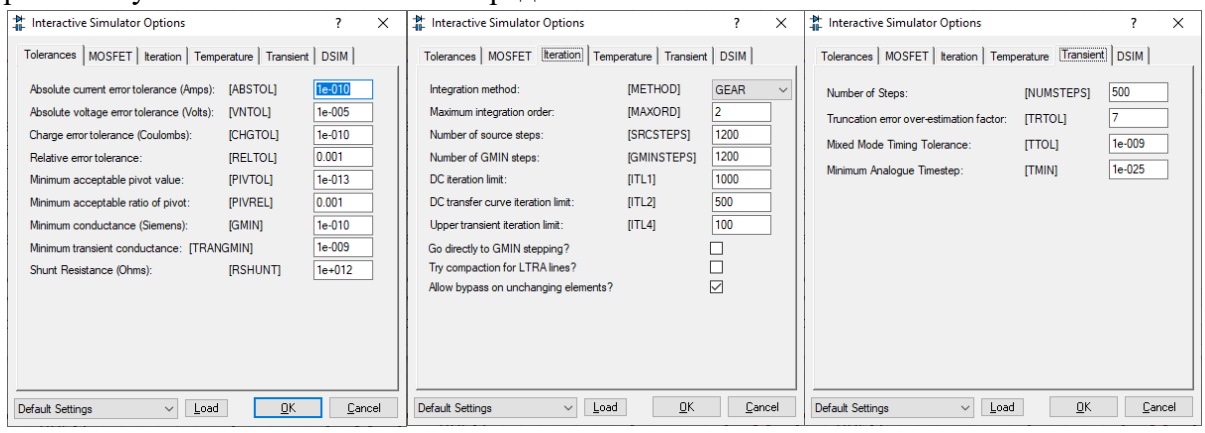


Рисунок 1 – Схема управления двигателем в Proteus

4. После сборки схемы необходимо настроить линии питания, для этого необходимо перейти в «Design->Configure Power Rails». Выбрать имя линии – VDD/VDD и добавить в нее линии 3.3V, и указать напряжение 3.3. Далее создать новую линию питания – 12V, установить напряжение 12 и добавить в нее силовую линию 12V.

5. Что бы Proteus всегда находил решение для данной схемы и запускал симуляцию необходимо изменить настройки. Переходим «System->Set Animation Options -> SPICE Options» и устанавливаем значения представленные ниже.



Создание прошивки для микроконтроллера.

1. Запустите STM32CubeIDE, в качестве workspace выберете папку для лабораторной работы. Создайте новый проект «Start new STM32 project», в поле выбора микроконтроллера выберете STM32F401RE. Нажмите Next, задайте имя проекта, выберете язык – C, типы выходного файла – Executable, тип проекта – STM32Cube; нажмите Finish. В пути и названии проекта не должно быть русских букв.

2. В открывшемся окне появится схематичное изображение микроконтроллера. Необходимо настроить все выводы, к которым подключены элементы.

2.1. Настройка канала передачи данных – UART.

Установите порты PA2 и PA3 как USART2_TX и USART2_RX соответственно. На боковой панели разверните пункт Connectivity, выберите USART2. В части окна – USART2 Mode and Configuration выберите режим работы usart – Asynchronous.

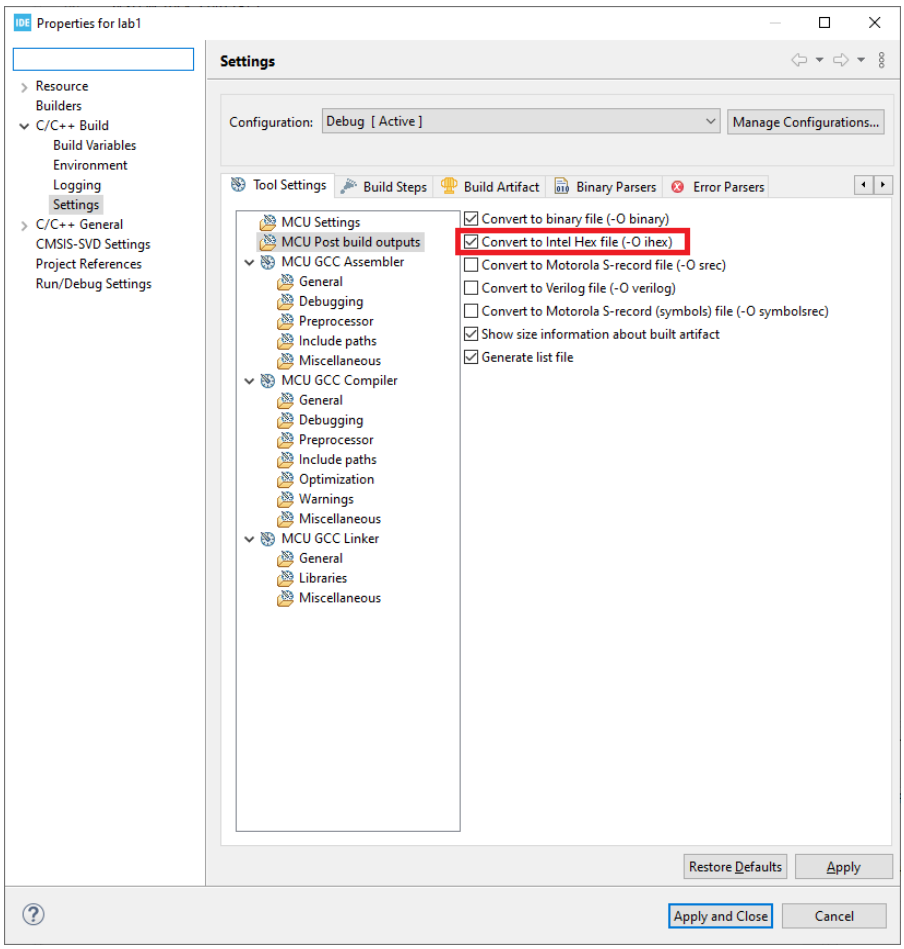
2.2. Настройка линий управления двигателем.

Установите PA5 и PA8 как GPIO_Output, а PA6 и PA7 как TIM3_CH1 и TIM3_CH2 соответственно. На боковой панели раскройте пункт Timers, выберите TIM3. В окне свойств таймера установите источник тактовых импульсов (Clock Source) на внутренний генератор (Internal clock). Для каналов 1 и 2 (Channel1, Channel2) выберите режим PWM (PWM Generation). В нижней части окна свойств установите предделитель и период счета таймера: Prescaler = 159, Counter Period = 100.

2.3. Настройка получения данных от энкодера.

Установите PB6 и PB7 как TIM4_CH1 и TIM4_CH2 соответственно. На боковой панели раскройте пункт Timers, выберите TIM4. В окне настройки таймера выберите комбинированный режим работы – работа с энкодером (Combined Channels – Encoder Mode). На боковой панели раскройте пункт System Core и выберите GPIO. В окне настройки GPIO перейдите на вкладку TIM. Для пинов, к которым подключен энкодер выберете режим работы Pull-up.ы

3. Сгенерируйте код, для этого нажмите Project->Generate Code на панели инструментов. Откроется основной файл программы – “main.c”. Если “main.c” не открылся, то его необходимо открыть вручную, он находится в Project Explorer по пути Core->Src->main.c. Откройте свойства проекта, для этого нажмите Project->Properties, далее перейдите C/C++ Build->Settings->Tool settings->MCU Post build options и установите галочку напротив пункта Convert to Intel Hex file.



Примените и закройте свойства.

4. Работа с периферией.

4.1. Отправка сообщения в терминал.

Для отправки сообщения в терминал используется команда «HAL_UART_Transmit(&huart#, &DATA, SIZE, 1000);», где # - номер используемого USART, DATA – массив с данными для отправки, SIZE – количество элементов в массиве. Для улучшения визуального восприятия передаваемой информации, лучше передавать текстовую строку, содержащую необходимые данные. Для того, что бы преобразовать значения различных переменных можно использовать команду «sprintf(DATA, TE,....data_values)». Для того, что бы можно было построить график разных величин, желательно воспользоваться следующим форматом сообщений: «sprintf(str, "%4d; %6d; %6d; %6d;\n\r", count, alpha, error, u);», где str – массив типа char, count – текущий номер сообщения, alpha – текущее положение ротора двигателя, error – значение ошибки между заданным положением и текущим, u – управляющее воздействие на двигатель; размер данного сообщения 31 символ.

4.2. Для управления двигателем необходимо включить таймер в режим генерации ШИМ и подать логическую единицу на нужный вывод GPIO. Работа с GPIO была разобрана в предыдущей работе. Включение таймера делается командой «HAL_TIM_PWM_Start(&htim#, TIM_CHANNEL_№);», где # - номер используемого таймера, № - номер канала. После этого, необходимо задать скважность шим, это делается командой «__HAL_TIM_SET_COMPARE(&htim#, TIM_CHANNEL_№, %);», где % - значение скважности ШИМ. Таким образом будет управляться скорость вращения

двигателем в процентном соотношении от 0 до 100%. Сформированное управляющее воздействие для двигателя необходимо будет подавать именно в данное место, т.е. изменять скважность ШИМ. (__HAL_TIM_SET_COMPARE(&htim#, TIM_CHANNEL_№, **u**);). Обратите внимание какой таймер и какую ножку GPIO необходимо включить, что бы двигатель начал вращаться.

4.3. Для получения данных о положении вала двигателя необходимо запустить таймер в режиме работы с энкодером командой – «HAL_TIM_Encoder_Start(&htim#, TIM_CHANNEL_ALL);». После запуска таймера, текущее значение энкодера можно получить, прочитав регистр счета соответствующего таймера: «TIM#->CNT;».

5. Используйте область “USER CODE BEGIN 2” для задания переменных и область бесконечного цикла “USER CODE BEGIN 3” для написания кода управления двигателем.

6. После написания кода нажмите Project->Build all для компиляции кода. Если в программе нет ошибок то в Build Console будет надпись: “Build Finished”.

Запуск программы в Proteus.

1. Откройте окно со схемой в Proteus. Перейдите в свойства микроконтроллера дважды на него нажав, в области Program File выберете скомпилированный STM32 IDE файл. Этот файл находится в папке Debug проекта и имеет расширение «ИМЯ ПРОЕКТА.hex».

Ход работы

1. Разработать схему в Proteus.
2. Разработать программу управления двигателем с применением ПИД регулятора.
3. Провести моделирование системы. Из полученных в терминале данных построить графики.

Содержание отчета:

- Титульный лист.
- Цель работы.
- Схема Proteus.
- Код программы управления двигателем.
- Графики работы ПИД регулятора.
- Вывод.

Варианты:

№ вар.	R, Ом	L, мГн	α_{ref}
1	6,6	39,6	200
2	0,75	3,75	300
3	0,85	2,55	600
4	1,5	9	750
5	0,53	4,24	820
6	16,6	116,2	1024
7	0,3	3	128
8	0,95	6,65	550

9	32	192	375
10	0,65	6,5	690
11	4,2	21	2000
12	0,72	1,44	1500