STM32 + HAL

Регулирование скорости двигателя постоянного тока потенциометром

NUCLEO-F413ZH • STM32CubeIDE / CubeMX • ADC1 + TIM3 PWM

ADC1 Измеряет напряжение потенциометра: 0...3,3 В	TIM3 PWM Создаёт управляющий сигнал на PB0
MOSFET Коммутирует ток двигателя от внешнего питания	HAL Измерение и регулировка каждые 10 мс

Что получится

При повороте ручки потенциометра STM32 измеряет аналоговое напряжение на PA0 и изменяет коэффициент заполнения ШИМ на PB0. MOSFET управляет током двигателя. Это открытое управление: фактические обороты не измеряются, а зависят также от нагрузки и напряжения питания.

Компоненты для сборки

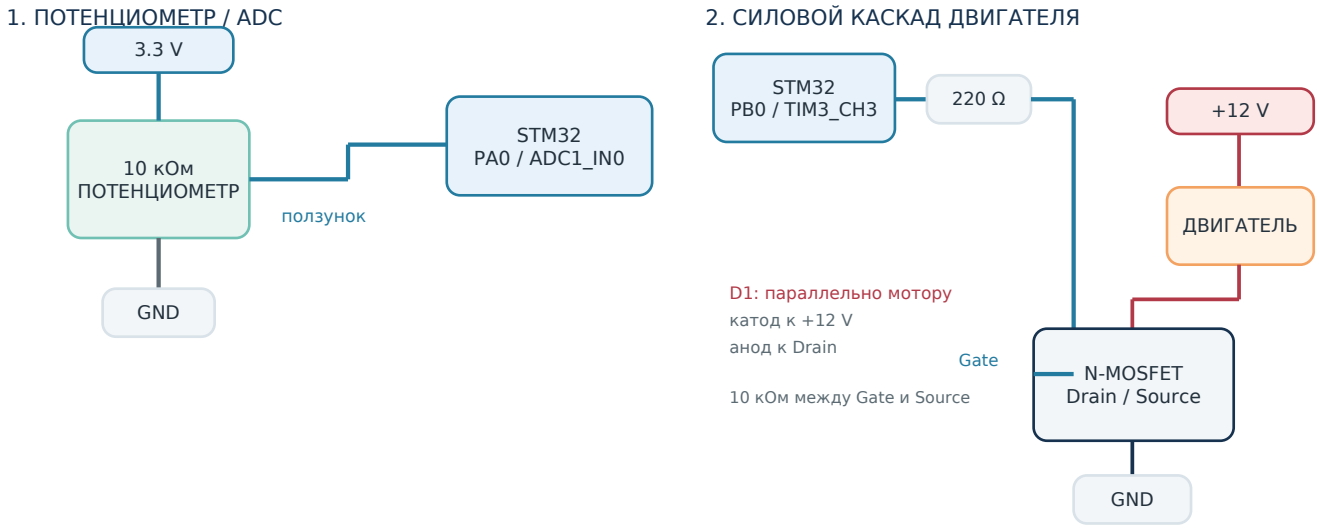
Компонент	Рекомендация
Плата	NUCLEO-F413ZH
Потенциометр	10 кОм, линейный, 3 вывода
Двигатель	Коллекторный DC; пример — 12 В. Источник выбирается по мотору.
Силовой ключ	N-канальный MOSFET, полностью открывающийся при VGS 2,5–3,3 В; ток с запасом относительно пускового/стопорного тока двигателя.
Защита	Диод параллельно двигателю, резистор затвора 100–220 Ом, подтяжка Gate–Source 10 кОм.

Важно. Не подключай двигатель напрямую к выводу STM32 и не питай его от USB платы. В этом проекте PB0 управляет только затвором MOSFET. Минус внешнего питания двигателя нужно соединить с GND платы.

В качестве примера ниже используются PA0 = ADC1_IN0 и PB0 = TIM3_CH3. Эти функции подтверждаются документацией микроконтроллера и руководством платы [1, 2].

1. Подключение компонентов

Электрическая схема: сигнальная часть и силовая часть должны быть разделены. Через MOSFET протекает ток двигателя; STM32 выдаёт только управляющий сигнал.



GND STM32 и минус источника двигателя должны быть соединены.

Подключение по пунктам

Узел	Подключение
Потенциометр	Крайний вывод → 3.3V; второй крайний → GND; средний (ползунок) → PA0. Если вращение работает наоборот, поменяй крайние выводы местами.
Gate MOSFET	PB0 → резистор 100–220 Ом → Gate. Между Gate и Source поставь 10 кОм, чтобы MOSFET оставался закрытым при старте/сбросе MCU.
Двигатель	+ внешнего источника → плюс двигателя; минус двигателя → Drain MOSFET; Source MOSFET → минус источника.
Защитный диод	Параллельно двигателю: катод (полоска) → + питания, анод → узел между двигателем и Drain. Ток диода должен выдерживать выброс тока двигателя.
Общая земля	Минус источника двигателя, Source MOSFET и GND STM32 должны быть соединены. Не проводи ток двигателя через дорожки/провода платы.

Выбор MOSFET. Надписи “logic level” недостаточно: проверь в datasheet сопротивление открытого канала RDS(on) при VGS около 2,5–3,3 В. Номинальный ток двигателя не равен пусковому; подбирай транзистор и охлаждение по пусковому/стопорному току.

Для помехоустойчивости полезно поставить возле двигателя по питанию конденсатор подходящего напряжения (например, электролитический 470–1000 мкФ вместе с керамическим 100 нФ), соблюдая полярность и размещая их рядом с силовой цепью. Это не заменяет защитный диод.

2. Настройка в STM32CubeIDE / CubeMX

Создай проект File → New → STM32 Project, выбери плату NUCLEO-F413ZH, задай имя проекта и открой .ioc. Настрой периферию до генерации кода.

ADC1 — вход потенциометра

Параметр	Значение	Зачем
PA0	ADC1_IN0	Вход напряжения ползунка потенциометра.
Resolution	12 bits	Результат от 0 до 4095.
Scan Conversion	Disabled	Используется один канал.
Continuous Conversion	Disabled	Каждый замер запускается из кода отдельно.
External Trigger	Software Start	Запуск преобразования функцией HAL.
Data Alignment	Right	Значение читается в диапазоне 0–4095.
Rank 1 / Channel	Channel 0	Соответствует ADC1_IN0.
Sampling Time	84 cycles (например)	Для потенциометра обычно достаточно; выбранное время должно соответствовать конфигурации ADC.

TIM3 — генератор ШИМ

Параметр	Значение
PB0	TIM3_CH3 (вывод D33 на разъёме Arduino-совместимой части)
Clock Source	Internal Clock
Channel 3	PWM Generation CH3
Counter Mode	Up
Prescaler (PSC)	4
Counter Period (ARR)	999
Pulse (CCR3)	0
PWM Mode / Polarity	PWM mode 1 / High

Частота PWM зависит от реальной частоты TIM3: fPWM = fTIM3 / ((PSC + 1) × (ARR + 1)). При fTIM3 = 100 МГц, PSC = 4, ARR = 999 получаем 20 кГц. Если CubeMX показывает другую частоту таймера, пересчитай значения по этой формуле.

Проверь Clock Configuration, затем нажми Generate Code. CubeMX должен создать инициализацию MX_ADC1_Init(), MX_TIM3_Init(), а также переменные hadc1 и htim3.

3. Код HAL: переменные и запуск PWM

Ниже приведены фрагменты для вставки в соответствующие пользовательские секции сгенерированного файла Core/Src/main.c. Не удаляй служебный код, который сгенерировал CubeMX.

A. Секция USER CODE BEGIN PV

```
/* USER CODE BEGIN PV */
static uint32_t adc_value = 0;
static uint32_t pwm_value = 0;
#define PWM_MAX 1000U
/* USER CODE END PV */
```

B. Запуск PWM в секции USER CODE BEGIN 2

Этот блок должен выполняться после MX_ADC1_Init() и MX_TIM3_Init(), то есть после инициализации периферии.

```
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, 0U);
/* USER CODE END 2 */
```

C. Содержимое бесконечного цикла while (1)

```
while (1)
{
    if (HAL_ADC_Start(&hadc1) == HAL_OK)
    {
        if (HAL_ADC_PollForConversion(&hadc1, 20) == HAL_OK)
        {
            adc_value = HAL_ADC_GetValue(&hadc1);  // 0..4095
            pwm_value = (adc_value * PWM_MAX) / 4095U;

            if (pwm_value > PWM_MAX)
                pwm_value = PWM_MAX;

            __HAL_TIM_SET_COMPARE(&htim3,
                                  TIM_CHANNEL_3,
                                  pwm_value);
        }
        HAL_ADC_Stop(&hadc1);
    }

    HAL_Delay(10);
}
```

В настройках ADC оставлен программный одиночный запуск, поэтому в каждом проходе цикла вызывается HAL_ADC_Start(), а после чтения — HAL_ADC_Stop(). PWM запускается один раз, а далее меняется только значение сравнения CCR3.

4. Сборка, запуск и диагностика

Порядок проверки

1	Нажми Generate Code и затем Build Project. Исправляй ошибки, если они связаны с настройками ADC/TIM3 или именами периферии.
2	Проверь, что hadc1 и htim3 присутствуют в сгенерированном проекте и функции MX_ADC1_Init() и MX_TIM3_Init() вызываются до USER CODE BEGIN 2.
3	Сначала проверь PWM без двигателя: измерь PB0 осциллографом или логическим анализатором, либо проверь изменение яркости встроенного LD1 (PB0).
4	Затем подключи силовой MOSFET, диод и двигатель с отключённым питанием. Ещё раз проверь полярность диода и общую землю.
5	Подай внешнее питание двигателя и поверни потенциометр. Не удерживай двигатель в остановленном состоянии — стопорный ток может перегреть двигатель, MOSFET или проводку.

Как читать значения

Положение	ADC приблизительно	PWM duty
Минимум	0	0%
Середина	2048	50%
Максимум	4095	100%

Частые проблемы

Симптом	Что проверить
Двигатель всегда выключен	Питание двигателя, общий GND, направление MOSFET D/S, назначение PB0 = TIM3_CH3, наличие сигнала PWM на PB0.
Скорость не меняется	Средний вывод потенциометра должен идти на PA0; PA0 настроен как ADC1_IN0; ADC1 реально запускается и возвращает разные значения.
Двигатель дёргается / STM32 сбрасывается	Проверь просадку питания и помехи, диод, разводку силового тока, конденсаторы у двигателя, общую землю и нагрев MOSFET. Не питать двигатель от USB платы.
Не стартует при малом PWM	У двигателя есть порог пускового момента. Можно добавить минимальный duty, например 15–20%, только когда ручка не в нуле. Это требует дополнительной логики в коде.

Документация

[1] STMicroelectronics, NUCLEO-F413ZH / Nucleo-144 boards user manual (UM1974): https://www.st.com/resource/en/user_manual/dm00244518-stm32-nucleo144-boards-stmicroelectronics.pdf

[2] STMicroelectronics, STM32F413xG/H datasheet: https://www.st.com/resource/en/datasheet/stm32f413zh.pdf

Ограничение примера. Код регулирует мощность двигателю, но не поддерживает постоянные обороты под переменной нагрузкой. Для стабилизации оборотов нужен датчик скорости (энкодер/датчик Холла) и контур обратной связи.