

Титульный лист
bldcbot

```
# Файл blcbot.py
import smbus
```

```
class Registers:
    """ Класс, хранящий регистры драйвера """
    REG_WHY_IAM = 0x00 # регистр,
    возвращающий 42
    REG_WORKMODE = 0x01 # режимы
    работы
    REG_PWM_INVERT = 0x02 # инверсия
    шима
    REG_KP = 0x04 # пропорциональный
    коэффициент
    REG_KI = 0x05 # интегральный
    коэффициент
    REG_KD = 0x06 # дифференциальный
    коэффициент
    REG_INT_SUMM = 0x07 # предел
    интегральной суммы
    REG_PID_PERIOD = 0x08 #
    REG_PARROT_A = 0x09 # скорость
    вращения мотора A в попугаях в режиме
    WORK_MODE_PID_I2C
    REG_PARROT_B = 0x0A # скорость
    вращения мотора A в попугаях в режиме
    WORK_MODE_PID_I2C
    REG_DIR_A = 0x0B # направление
    вращения мотора A
    REG_PWM_A = 0x0C # ШИМ задаваемый
    мотору A в режиме WORK_MODE_PWM_I2C
    REG_DIR_B = 0x0D # направление
    вращения мотора B
    REG_PWM_B = 0x0E # ШИМ задаваемый
    мотору B в режиме WORK_MODE_PWM_I2C
    REG_RESET_ALL_MOTOR = 0x0F # сброс
    всех внутренних параметров
```

```
class WorkMode:
    """ Класс, хранящий режимы работы
    драйвера """
    WORK_MODE_PWM = 0x01 # режим
    работы - напрямую от ШИМ вилки на плате
    WORK_MODE_PID = 0x02 # режим работы
    - от ШИМ вилки на плате через ПИД-
    регулятор
    WORK_MODE_PWM_I2C = 0x03 # режим
    работы - от ШИМ, параметры которого
    задаются через i2c
    WORK_MODE_PID_I2C = 0x04 # режим
    работы - через задаваемые по i2c попугаи
```

```
class Direction:
```

```
""" Класс, хранящий возможные
направления """
```

```
FORWARD = 0x00 # вперед
BACKWARD = 0x01 # назад
```

```
class BLDCbot:
    """ Класс работы с драйвером моторов """
```

```
def __init__(self, bus, addr=0x27):
    self._bus = bus # шина i2c
    self._addr = addr # адрес устройства
```

```
def whoIam(self):
    """ Должен вернуть 42 """
    return self._bus.read_byte_data(self._addr,
    Registers.REG_WHY_IAM)
```

```
def setWorkMode(self, mode):
    """ Устанавливает режим работы драйвера
    """
    self._bus.write_byte_data(self._addr,
    Registers.REG_WORKMODE, mode)
```

```
def _setDirectionA(self, dir):
    """ Устанавливает направление вращения
    мотора A """
    self._bus.write_byte_data(self._addr,
    Registers.REG_DIR_A, dir)
```

```
def _setDirectionB(self, dir):
    """ Устанавливает направление вращения
    мотора B """
    self._bus.write_byte_data(self._addr,
    Registers.REG_DIR_B, dir)
```

```
def setParrotA(self, parrot):
    """ Устанавливает скорость вращения
    мотора A в попугаях """
    if parrot < 0:
```

```
self._setDirectionA(Direction.FORWARD)
    else:
```

```
self._setDirectionA(Direction.BACKWARD)
    self._bus.write_byte_data(self._addr,
    Registers.REG_PARROT_A, abs(parrot))
```

```
def setParrotB(self, parrot):
    """ Устанавливает скорость вращения
    мотора B в попугаях """
    if parrot < 0:
```

```
self._setDirectionB(Direction.FORWARD)
    else:
```

```
self._setDirectionB(Direction.BACKWARD)
    self._bus.write_byte_data(self._addr,
Registers.REG_PARROT_B, abs(parrot))

    def setPwmA(self, dir, pwm):
        """ Устанавливает скорость через
параметры шима """
        self._setDirectionA(dir)
        self._bus.write_byte_data(self._addr,
Registers.REG_PWM_A, abs(pwm))

    def setPwmB(self, dir, pwm):
        """ Устанавливает скорость через
параметры шима """
        self._setDirectionB(dir)
        self._bus.write_byte_data(self._addr,
Registers.REG_PWM_B, abs(pwm))

    def setKp(self, kp):
        """ Устанавливает пропорциональный
коэффициент регулятора """
        self._bus.write_byte_data(self._addr,
Registers.REG_KP, abs(int(kp * 10)))

    def setKi(self, ki):
        """ Устанавливает интегральный
коэффициент регулятора """
        self._bus.write_byte_data(self._addr,
Registers.REG_KI, abs(int(ki * 10)))

    def setKd(self, kd):
        """ Устанавливает дифференциальный
коэффициент регулятора """
        self._bus.write_byte_data(self._addr,
Registers.REG_KD, abs(int(kd * 10)))

if __name__ == "__main__":
    bus = smbus.SMBus(1)
    bot = BLDCbot(bus)

    print(bot.whoIam())

bot.setWorkMode(WorkMode.WORK_MODE_P
ID_I2C)
bot.setParrotA(0x05)
```