Титульный лист bldcbot

# Файл bldcbot.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 А в попугаях в режиме WORK\_MODE\_PID\_I2C

REG\_PARROT\_B = 0x0A # скорость вращения мотора А в попугаях в режиме WORK\_MODE\_PID\_I2C

REG\_DIR\_A = 0x0B # направление вращения мотора A

REG\_PWM\_A = 0x0C # ШИМ задаваемый мотору А 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 # адресс 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 А """

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 вращение мотора А 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 вращение 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\_PID\_I2C)

bot.setParrotA(0x05)