Rotrics DexArm

Руководство по программированию

Алматы, 2021 (v22102021-1223)

СОДЕРЖАНИЕ:

1. Введение
2. Подготовка робота
3. Движение робота
4. Лазер
5. Пневматика
6. Конвейер
7. Машинное зрение
8. Заключение
9. Приложение

ВВЕДЕНИЕ (1)

Rotrics DexArm – универсальный настольный робот-манипулятор для лазерной гравировки, 3D-печати и машинного зрения. Робот обладает повышенной по сравнению с другими настольными роботами точностью операций – до 0,005 мм. Данное свойство позволяет реализовать множество кропотливых операций, в которых важна точность исполнения.

Изображение 1.1

Робот программируется на языках - G-code, C++, Python. Два последних языка программирования описывают интерфейс-библиотеку для работы с роботом. Робот ориентируется в трех плоскостях – X, Y, Z. При этом, у робота существуют ограничения в координатных плоскостях из-за конструктивных особенностей (*см. Таблица 1.1*).

Ось X	*Ось Y*	*Ось Z*
(-330: - 120) U (120: 330)	(230; 380)	(165; 127)

Таблица 2.1

Спецификации:

1. Точность операций: 0,005 мм;
2. Сборочные габариты: 220 x 155 x 160 мм;
3. Габариты робота: 175 x 128 x 315 мм;
4. Вес робота: 2,4 кг.

Комплектация:

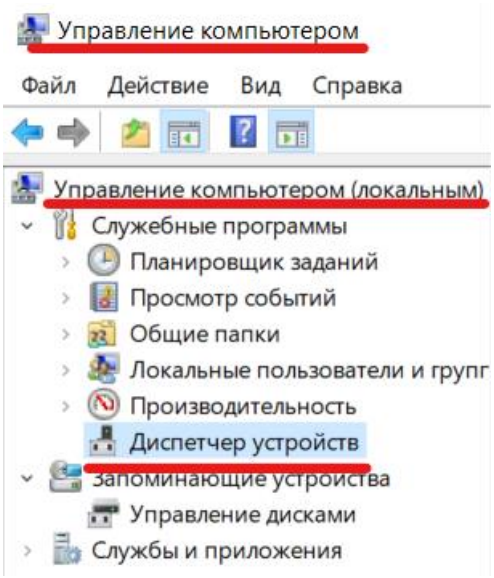
1. Робот;
2. Модуль «Держатель для ручки»;
3. Модуль «Лазер 2,5 Вт»;
4. Модуль «3D-печать»;
5. Сенсорный экран.

Первоначальная настройка робота:

Шаг 1 – Скачайте с официального сайта Rotrics программное обеспечение Rotrics Studio для взаимодействия с роботом – https://rotrics.com;

Шаг 2 – Подключите робота с помощью разъема *Power Adapter* к сети 220В, и с помощью кабеля *USB Type-C <-> USB Type A*, подключите робота к компьютеру;

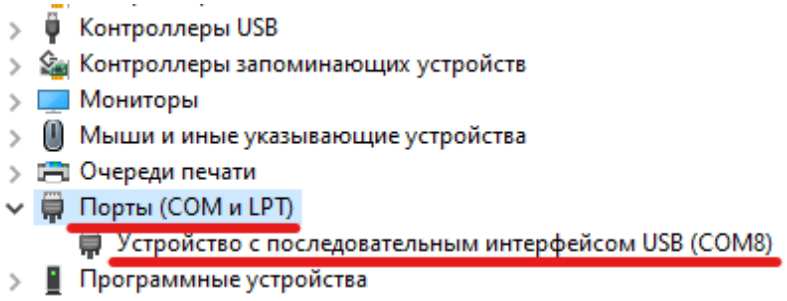
Шаг 3 – После подключения робота, зайдите в меню «*Пуск*», и найдите утилиту «*Управление компьютером*» как показано на изображении ниже: (*см. Изображение 1.2*);



Изображение 1.2

Шаг 4 – В подменю «*Диспетчер устройств*», нажмите на вкладку «*Порты (COM и LPT)*». В выпадающем меню будет показан робот с последовательным портом и номером этого порта.

Например: «*Устройство с последовательным портом USB (COM8)*», где *8* – номер вашего порта. (*см. Изображение 1.3*);



Изображение 1.3

ПОДГОТОВКА РОБОТА (2)

Перед взаимодействием с роботом ознакомьтесь со следующими *системными требованиями*:

- Операционная система – Windows, MacOS, Linux;
- Python – 3.10 и выше (Рекомендуется версия Python не ниже 3.6, однако результат может отличаться, т.к. все действия проводились на ПК с Python 3.10 (Python 3.10 не поддерживает Windows 7);
- Git – 2.20 и выше;
- IDE – VS Code, PyCharm (Рекомендуется использовать Visual Studio Code или PyCharm).

Начало работы:

Шаг 1 – После установки необходимого программного обеспечения, установите утилиту `pip` для дополнительной установки необходимых пакетов Python:

```
python get-pip.py
```

Шаг 2 – После установки утилиты `pip`, установите дополнительные пакеты для корректной работы с роботом:

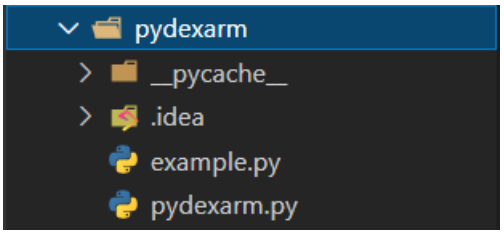
```
pip install opencv-python
pip install pyyaml
pip install pyserial
```

Шаг 3 – С помощью утилиты `git` клонируйте репозиторий по следующему адресу:

```
git clone https://github.com/AndreM07/dex-arm-book
```

Шаг 4 – Открываем IDE, во вкладке на верхней панели, щелкаем на меню «Открыть», выбираем папку с нашим клонированным репозиторием;

Шаг 5 – В открывшемся дереве директории, увидим нашу папку с проектом, в котором есть два файла – «`example.py`» и «`pydexarm.py`»



Изображение 2.1

Шаг 6 – Файл «`example.py`» - содержит своеобразный hello-world для робота, а файл «`pydexarm.py`» - содержит библиотеку-интерфейс для взаимодействия с роботом на языке G-code.

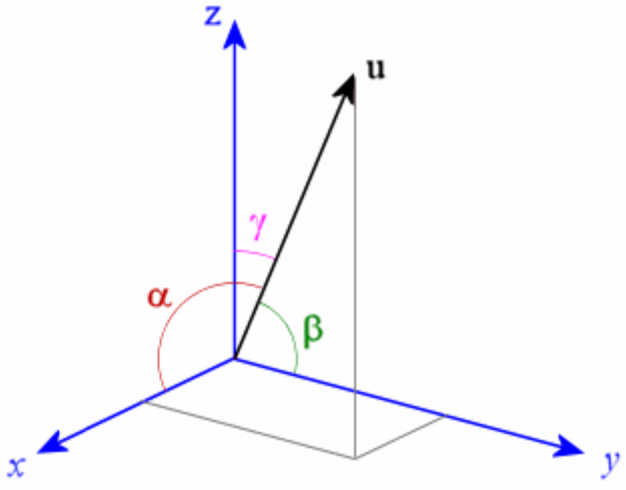
ДВИЖЕНИЕ РОБОТА (3)

Рассмотрим основные правила задания движения робота по координатным осям OX, OY, OZ. На *Изображении 2.1,* изображены координатные оси, а именно:

1. *Ось X*
2. *Ось Y*
3. *Ось Z*
4. *Вектор U*
5. *Углы* α, β *и* γ

Вектор U – это точка в пространстве которая была задана роботу с помощью трех осей, например: (100, 300, -75). Данные координаты расшифровываются так:

1. *100* – точка на координатной оси X
2. *300* – точка на координатной оси Y
3. *-75* – точка на координатной оси Z



Изображение 2.1

Для следующего примера установите модуль «Держатель» и надежно закрепите в данном модуле канцелярскую ручку. С помощью данного модуля будет показана логика ориентирования робота в пространстве с помощью осевых координат.

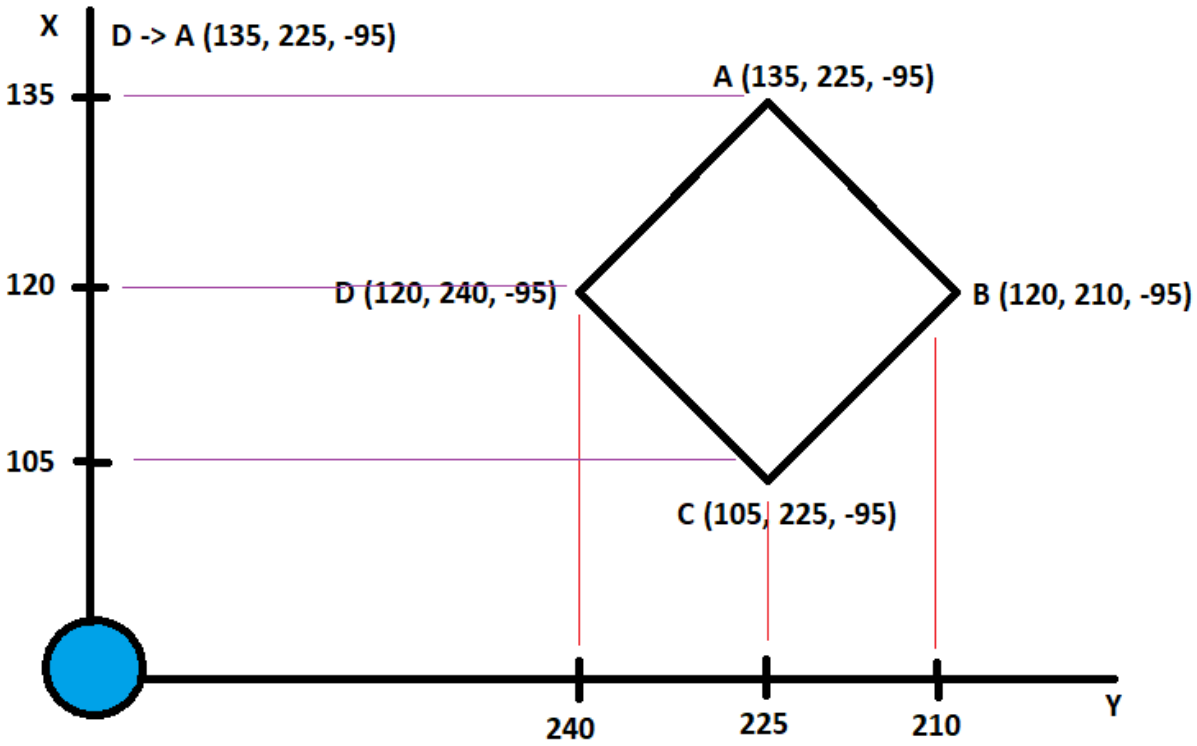
Листинг 2.1:

```
from pydexarm import Dexarm
dexarm = Dexarm("COM8") # Подключение к Windows
# device =
Dexarm("/dev/tty.usbmodem3086337A34381") #
Подключение к MacOS/Linux

dexarm.go_home()

dexarm.move_to(135, 225, -95) # A
dexarm.move_to(120, 210, -95) # B
dexarm.move_to(105, 225, -95) # C
dexarm.move_to(120, 240, -95) # D
dexarm.move_to(135, 225, -95) # D to A

dexarm.go_home ()
```

Рассмотрим *листинг 2.1:*

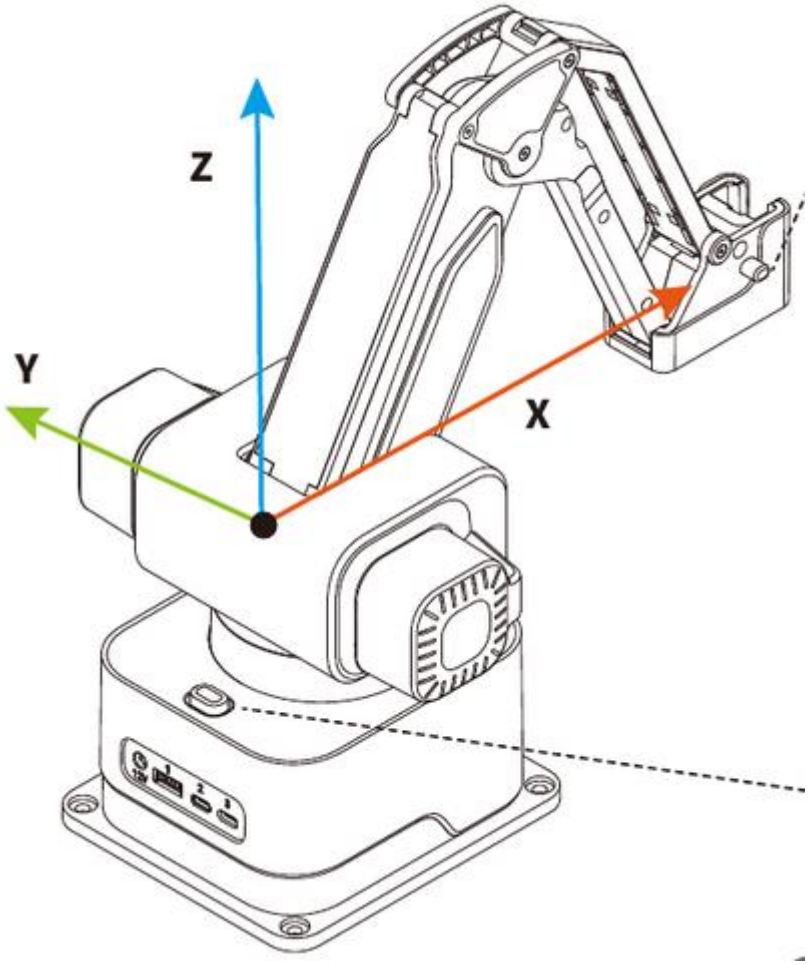
1. Строка « `from pydexarm import Dexarm` » - включает в основной код библиотеку-интерфейс для общения с роботом при помощи связи между Python и G-code;
2. Строка « `dexarm = Dexarm("COM8")` » - отправляет скомпилированный код в робот на указанный COM-порт, а именно COM8. Номер порта можно узнать в главе *«Введение», Шаг 4*
3. Строка « `dexarm.go_home()` » - дает команду роботу вернуться в начальное или исходное положение. Функция не требует аргументов, так как робот настроен на начальное положение в (0; 0; 0)
4. Строка « `dexarm.move_to(135, 225, -95)` » - определяет координаты по трем осям, и отправляет роботу команду для перемещения манипулятор робота в данную точку, а именно – по оси X – 135; по оси Y – 225; по оси Z – -95.

Математическое объяснение листинга 2.1 и операций робота:



Изображение 2.2

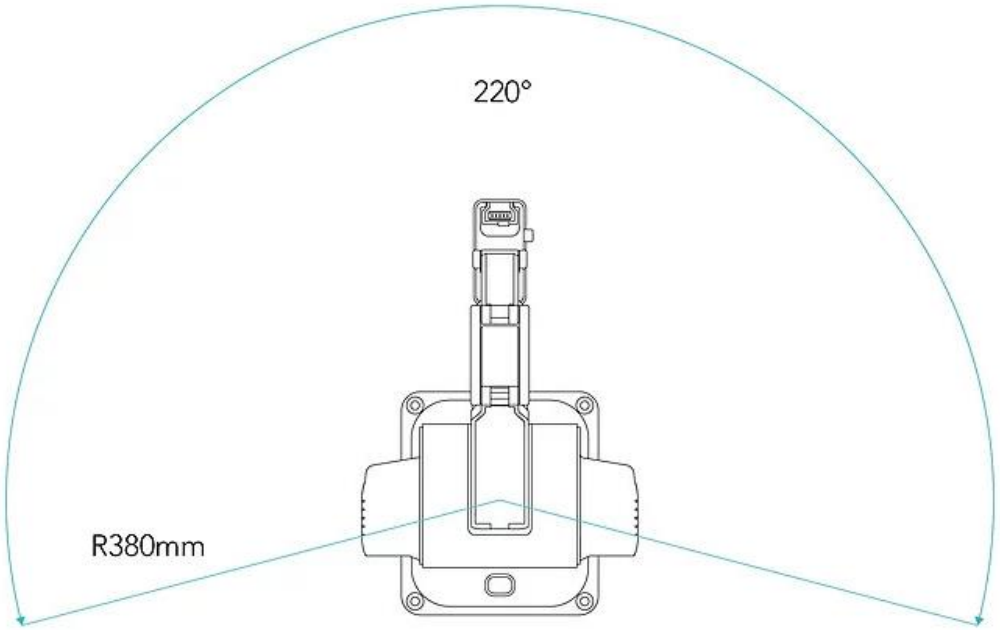
1. Для того чтобы нарисовать квадрат или любую другую фигуру на бумаге, дереве или другом материале, следует помнить то, что для модуля «Ручка» установлено минимальная высота, а именно -95 (минус 95), а для модуля «Лазер» минимальная высота составляет -70 (минус 70);
2. Координатные оси распределены следующим образом:



Изображение 2.3

3. Таким образом, оси X и Y поменялись местами в отличие от математической модели осей, где вверху ось Y, а внизу ось X.

4. При работе с роботом, следует не забывать про конструктивные ограничения, а именно:



- Максимальный угол для движения по оси X и Y составляет 220 градусов;
- При этом максимальный радиус для операций составляет 380 мм;
- Причем, координаты в нашей программе мы принимаем за 1 мм, то есть (100, 300, -75), означает что робот по оси X перейдет в точку 100 (100 мм от начала координат 0; 0; 0), по оси Y в точку 300 (300 мм от начала координат 0; 0; 0) и по оси Z в точку -75 (-75 мм от начала координат 0; 0; 0);

5. При всем этом, мы должны обязательно помнить то, что нельзя выходить за рамки координат!

ЛАЗЕР (4)

ВНИМАНИЕ! ПРИ РАБОТЕ С МОДУЛЕМ «ЛАЗЕР», СОБЛЮДАЙТЕ НЕОБХОДИМУЮ ТЕХНИКУ БЕЗОПАСНОСТИ, А ИМЕННО ПРИ ВКЛЮЧЕНИИ ЛАЗЕРА ОДЕВАЙТЕ СПЕЦИАЛЬНЫЕ ОТРАЖАЮЩИЕ ОЧКИ, ПОСЛЕ ВЫКЛЮЧЕНИЯ ЛАЗЕРА, ОБЯЗАТЕЛЬНО ПРОВЕТРИВАЙТЕ ОБЛАСТЬ И НЕ СОЗДАВАЙТЕ ПОЖАРООПАСНЫХ СИТУАЦИЙ! БУДЬТЕ ОСТОРОЖНЫ!

После установки модуля «Лазер» и его надежном закреплении в месте установки, включится маленький вентилятор, предназначенный для охлаждения лазерного модуля.

ПНЕВМАТИКА (5)

КОНВЕЙЕР (6)

МАШИННОЕ ЗРЕНИЕ (7)

ЗАКЛЮЧЕНИЕ (8)

ПРИЛОЖЕНИЕ (9)

- Приложение № 1 – Полное описание библиотек-интерфейса «pydexarm.py»:

```
import serial
import re

class Dexarm:

    def __init__(self, port):
        self.ser = serial.Serial(port, 115200,
timeout=None)
        self.is_open = self.ser.isOpen()
        if self.is_open:
            print('pydexarm: %s open' %
self.ser.name)
        else:
            print('Failed to open serial port')

    def _send_cmd(self, data):
        self.ser.write(data.encode())
```

```
        while True:
            str =
self.ser.readline().decode("utf-8")
            if len(str) > 0:
                if str.find("ok") > -1:
                    print("read ok")
                    break
                else:
                    print("read:", str)

    def go_home(self):
        self._send_cmd("M1112\r")

    def set_workorigin(self):
        self._send_cmd("G92 X0 Y0 Z0 E0\r")

    def set_acceleration(self, acceleration,
travel_acceleration, retract_acceleration=60):
        cmd = "M204"+"P" + str(acceleration) +
"T"+str(travel_acceleration) + "T" +
str(retract_acceleration) + "\r\n"
        self._send_cmd(cmd)

    def set_module_kind(self, kind):
        self._send_cmd("M888 P" + str(kind) +
"\r")

    def get_module_kind(self):
        self.ser.write('M888\r'.encode())
        while True:
            str =
self.ser.readline().decode("utf-8")
            if len(str) > 0:
                if str.find("PEN") > -1:
                    module_kind = 'PEN'
                if str.find("LASER") > -1:
                    module_kind = 'LASER'
                if str.find("PUMP") > -1:
                    module_kind = 'PUMP'
                if str.find("3D") > -1:
```

```
                    module_kind = '3D'
            if len(str) > 0:
                if str.find("ok") > -1:
                    return module_kind

    def move_to(self, x, y, z, feedrate=2000):
        cmd = "G1"+"F" + str(feedrate) +
"X"+str(x) + "Y" + str(y) + "Z" + str(z) +
"\r\n"
        self._send_cmd(cmd)

    def fast_move_to(self, x, y, z,
feedrate=2000):
        cmd = "G0"+"F" + str(feedrate) +
"X"+str(x) + "Y" + str(y) + "Z" + str(z) +
"\r\n"
        self._send_cmd(cmd)

    def get_current_position(self):
        self.ser.write('M114\r'.encode())
        while True:
            str =
self.ser.readline().decode("utf-8")
            if len(str) > 0:
                if str.find("X:") > -1:
                    temp = re.findall(r"[-
+]?\d*\.\d+|\d+", str)
                    x = float(temp[0])
                    y = float(temp[1])
                    z = float(temp[2])
                    e = float(temp[3])
            if len(str) > 0:
                if str.find("DEXARM Theta") > -
1:
                    temp = re.findall(r"[-
+]?\d*\.\d+|\d+", str)
                    a = float(temp[0])
                    b = float(temp[1])
                    c = float(temp[2])
            if len(str) > 0:
```

```
                if str.find("ok") > -1:
                    return x,y,z,e,a,b,c

    """Delay"""
    def dealy_ms(self, value):
        self._send_cmd("G4 P" + str(value) +
'\r')

    def dealy_s(self, value):
        self._send_cmd("G4 S" + str(value) +
'\r')

    """SoftGripper & AirPicker"""
    def soft_gripper_pick(self):
        self._send_cmd("M1001\r")

    def soft_gripper_place(self):
        self._send_cmd("M1000\r")

    def soft_gripper_nature(self):
        self._send_cmd("M1002\r")

    def soft_gripper_stop(self):
        self._send_cmd("M1003\r")

    def air_picker_pick(self):
        self._send_cmd("M1000\r")

    def air_picker_place(self):
        self._send_cmd("M1001\r")

    def air_picker_nature(self):
        self._send_cmd("M1002\r")

    def air_picker_stop(self):
        self._send_cmd("M1003\r")

    """Laser"""
    def laser_on(self, value=0):
```

```
            self._send_cmd("M3 S" + str(value) +
'\r')

    def laser_off(self):
        self._send_cmd("M5\r")

    """Conveyor Belt"""
    def conveyor_belt_forward(self, speed=0):
        self._send_cmd("M2012 S" + str(speed) +
'D0\r')
    def conveyor_belt_backward(self, speed=0):
        self._send_cmd("M2012 S" + str(speed) +
'D1\r')
    def conveyor_belt_stop(self, speed=0):
        self._send_cmd("M2013\r")
```