Unitree A1用
高トルクインテリジェントモーター

Unitree A1担当
Tel: 03-5683-7293
Email: unitree@techshare.co.jp

TechShare

Agenda

1. モーターの構成や基本仕様
2. モーター開発キットの概要と使い方
3. モーター開発用のSDK
   - 3.1 開発用SDKの概要
   - 3.2 サンプルプログラムの説明
   - 3.3 データの読み書き
   - 3.4 減速機の説明
   - 3.5 複数モーターの連動
4. よくある質問

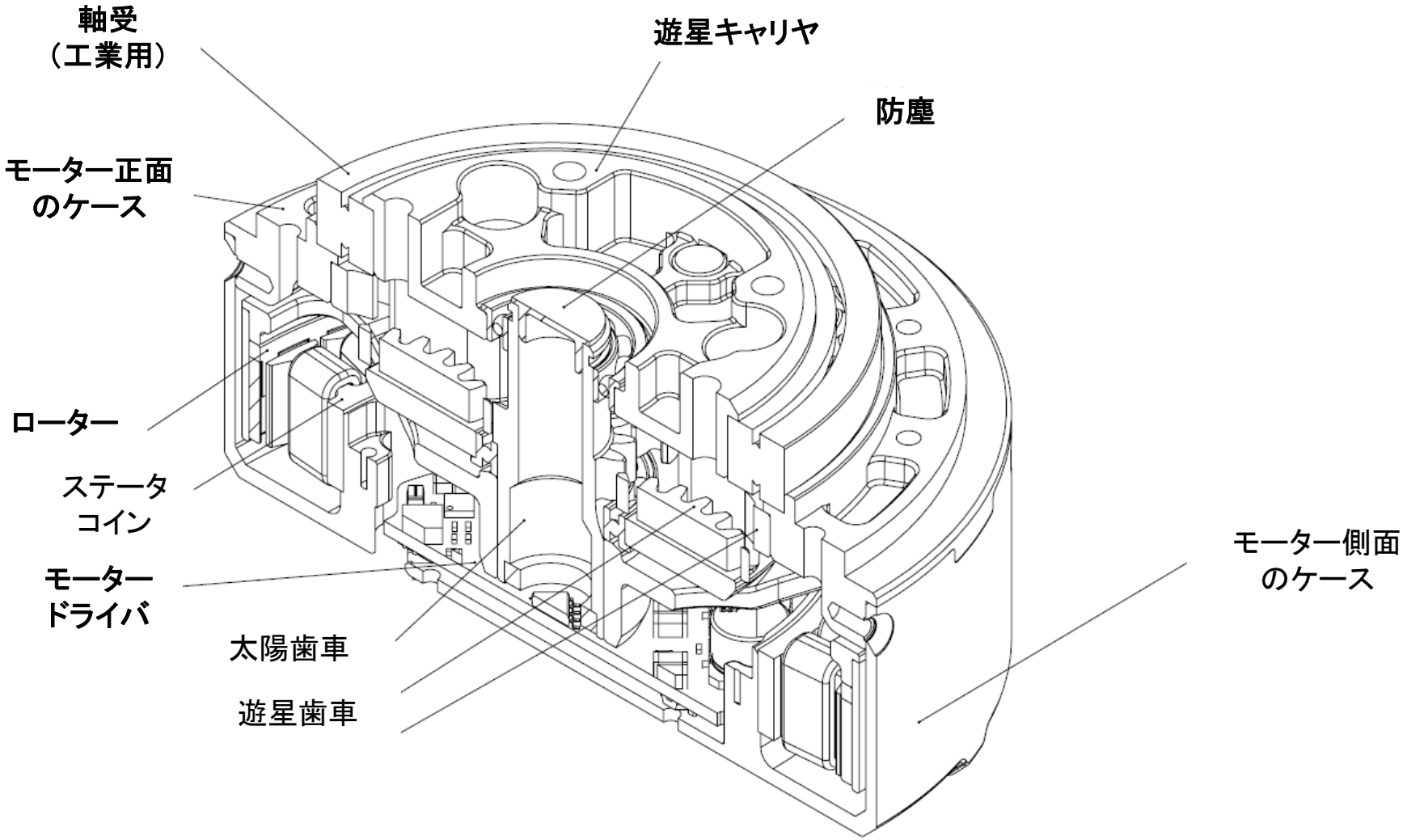
1.1 モーターモジュールの構成

A1モーターモジュール

1. モーター単体（**BLDCモーター**）
2. 減速機
3. ドライバ
4. エンコーダ

2,3,4を含めて
内蔵されています

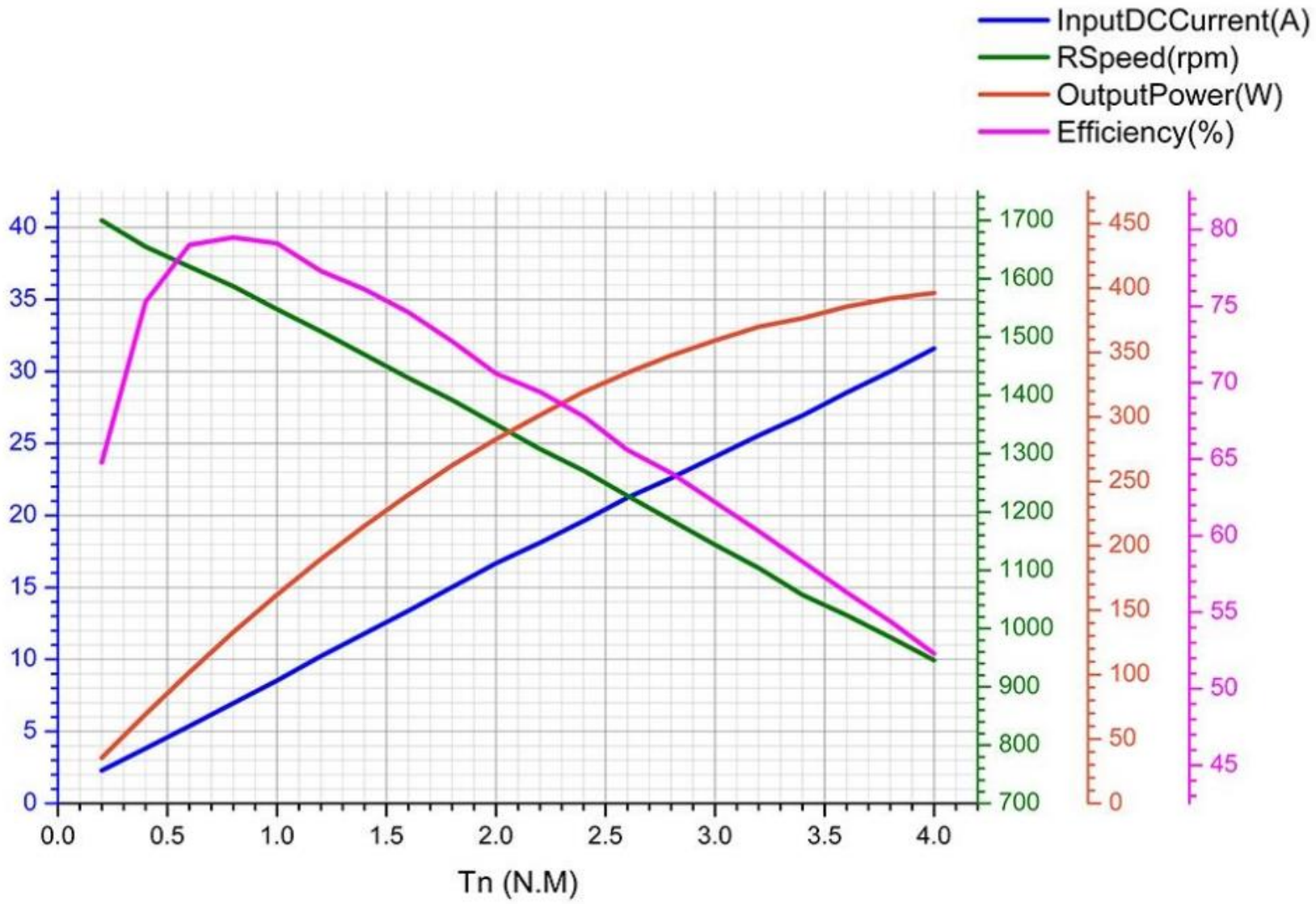
1.2 Unitreeモーターの構造



1.3 モーターの基本仕様

品名		品名	
重量	605 g	制御方式	位置、速度、トルク
定格出力	260 W	インピーダンス制御	OK
最大トルク	33.5 N・m	開発環境	C, C++, Python, ROS
最大角速度	21 rad/s	通信方式	RS485
減速比	約7:1	軸受の種類	工業用軸受を搭載
最大の回転数	1700 rpm	最大の通信周期	1 KHz
動作電圧	12~30V（24V推奨）	エンコーダの分解能	15bit
電流の種類	直流電流　AC	温度センサー	搭載(高温保護機能有)
トルク係数	0.8372N・m/A	冷却ファン	搭載可能(オプション)

1.4 3相ベクトル制御からモーターの性能線図



参考（1）：BLDCモーター単体の概要とメリット

BLDCモーターとは?

BLDC（ブラシレスDC）モータは、のように回転子が永久磁石、固定子がコイルになっています。

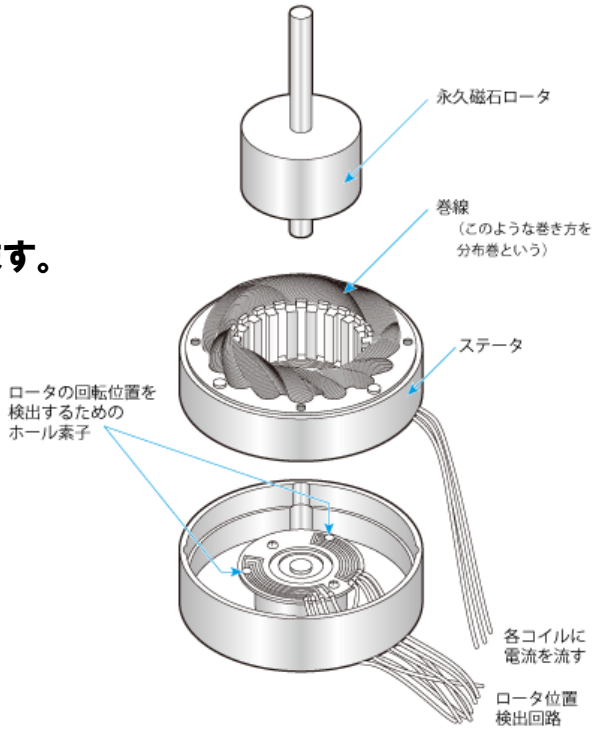
メリット1:　効率が良い

小さなBLDCモータでも力を出せる

メリット2. 制御性が良さ

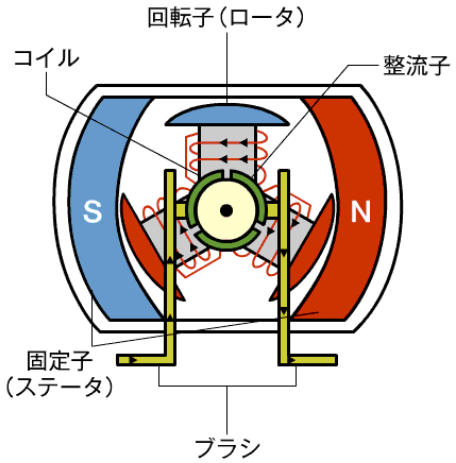
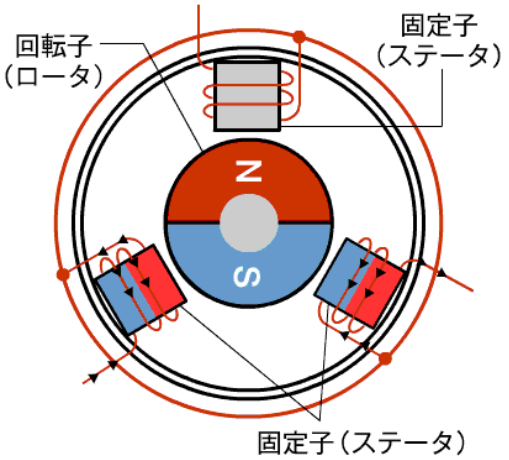
出したいトルク、得たい回転数などをピッタリと得ることができます。

メリット3. 耐久性が高い



BLDCモーターの構造

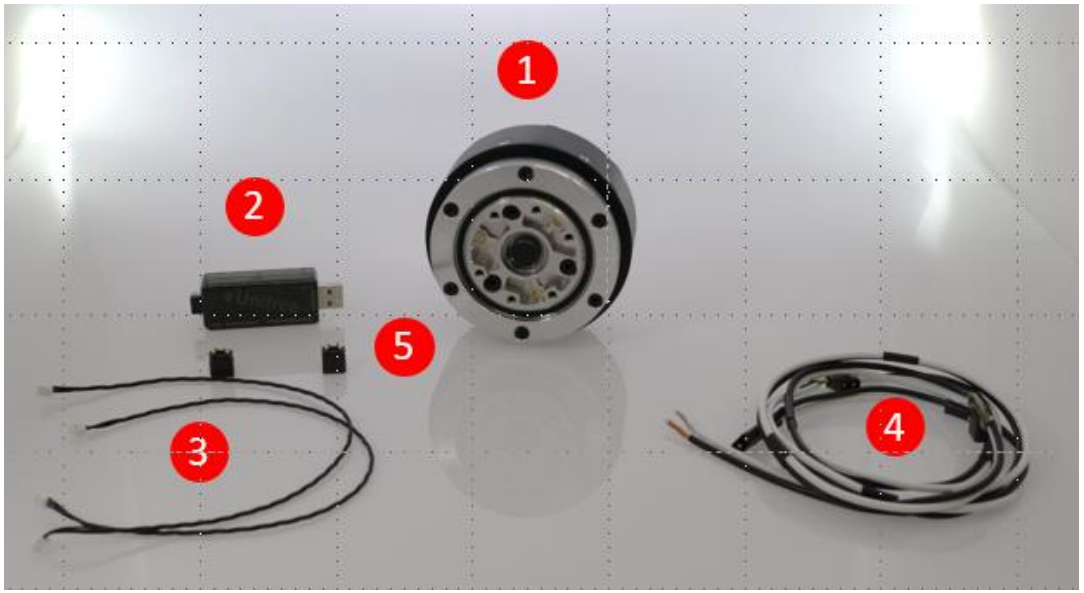
参考（2）：DCモーターとBLDCモーターの比較

	DCモーター	BLDCモーター（Unitree用）
種類		
動作のイメージ		
	直流を入力とする電動機	BLDCモータでは永久磁石が回転子になっていて、回転子にコイルが無いのです

Agenda

1. モーターの構成や基本仕様
2. モーター開発キットの概要と使い方
3. モーター開発用のSDK
   - 3.1 開発用SDKの概要
   - 3.2 サンプルプログラムの説明
   - 3.3 データの読み書き
   - 3.4 減速機の説明
   - 3.5 複数モーターの連動
4. よくある質問

2.1 モーター開発キットの概要



1. モーター本体 x 1
2. 開発用のUSBドングル x 1
3. RS485通信ケーブル x 1
4. 電源供給ケーブル x 1
5. 電源端子（予備用） x 1

2.2 開発部品の説明

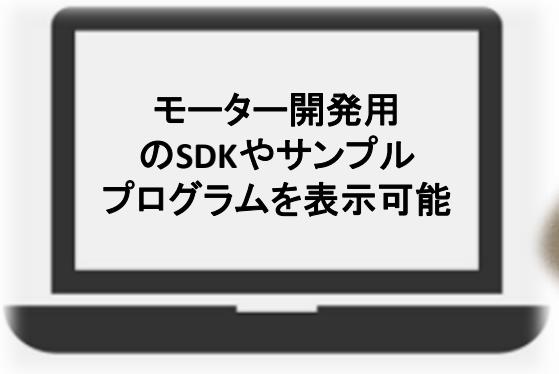
電源供給ケーブル

最大 3つまで モーター給電可能
長さ約1.2メートル

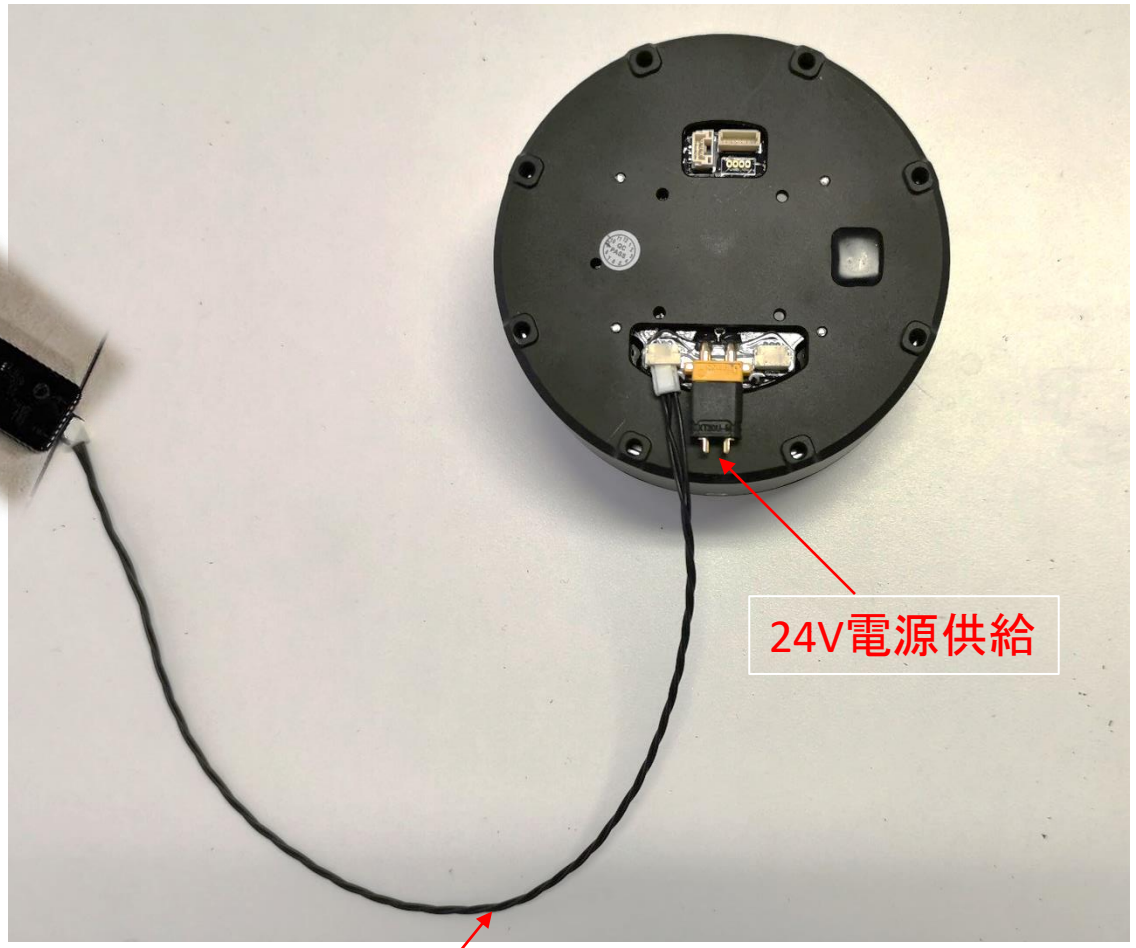
1. RS485からUSBへ変換
2. モーター開発用SDK
3. サンプルプログラム
4. SDKマニュアル

開発用のUSBドングル

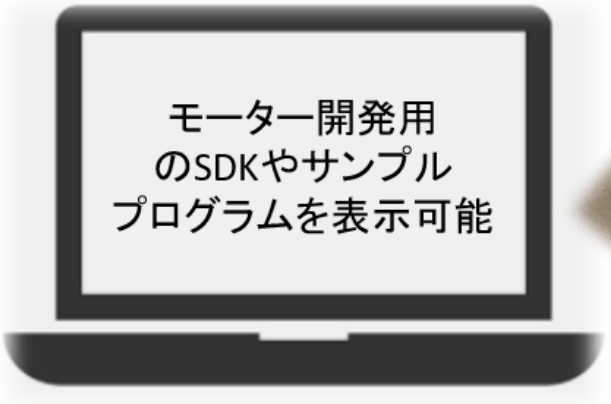
2.3 接続方法



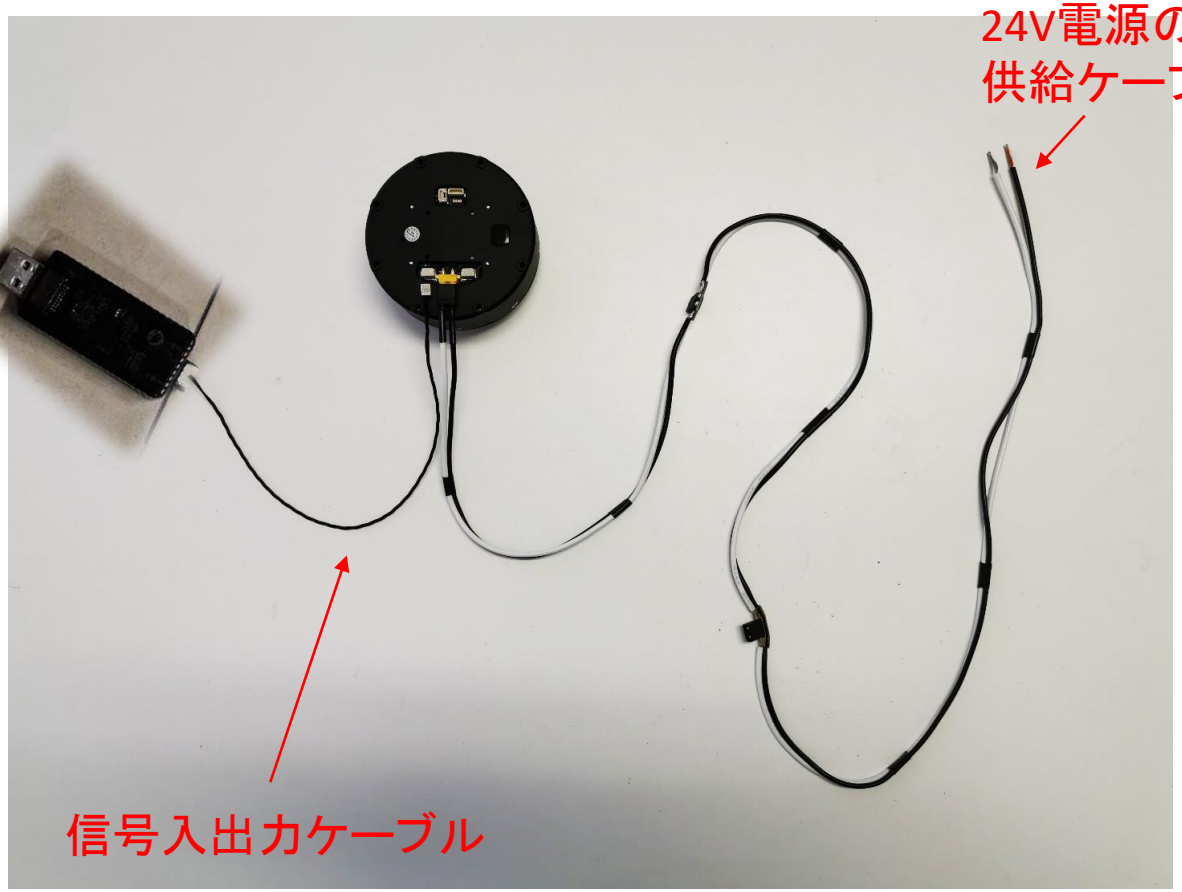
Linux系又はWindows系



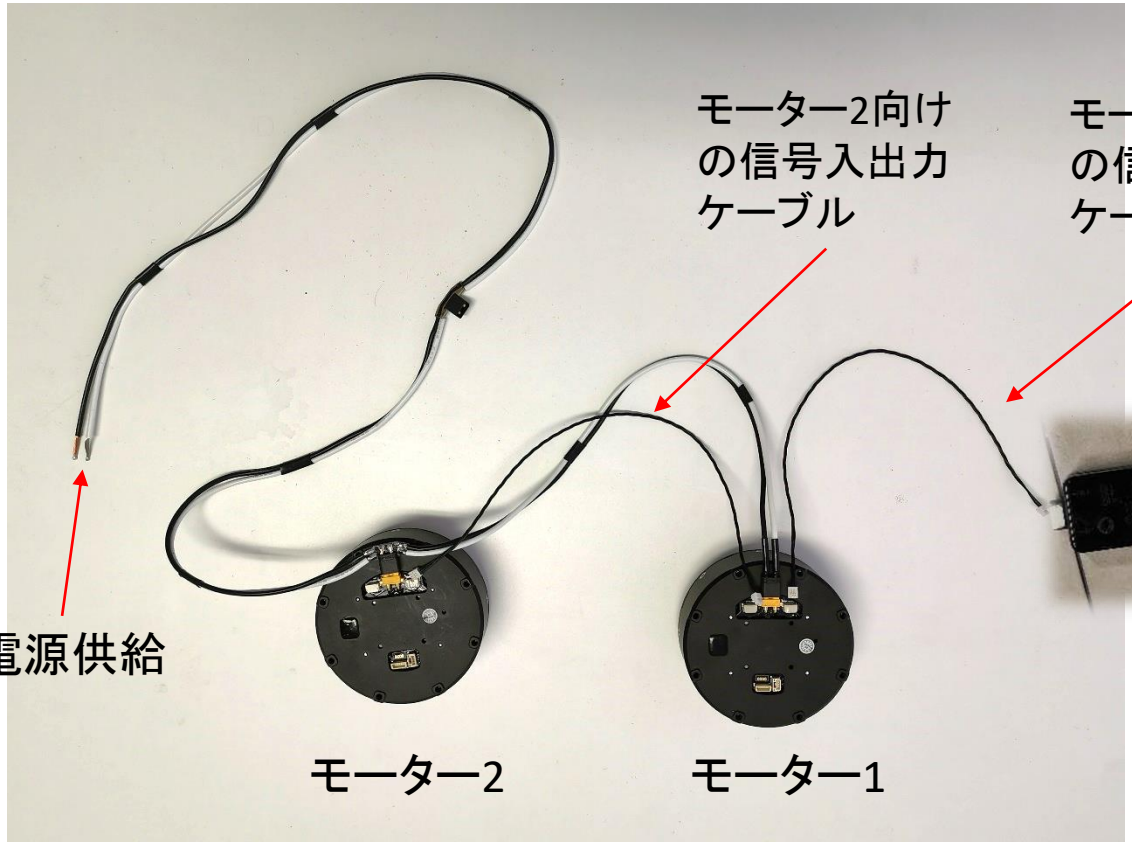
2.4 接続例(１)ーモータ1つを接続する場合

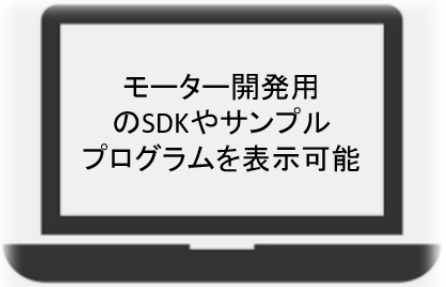


Linux系又はWindows系



2.4 接続例（2）－複数のモータを接続する場合





Linux系又はWindows系

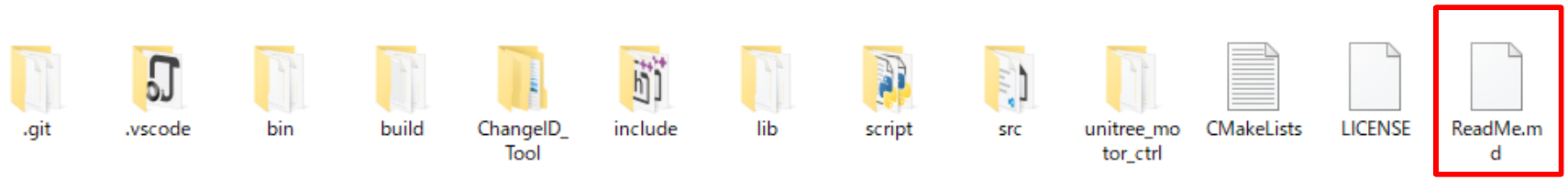
“モーター” 2個 ⇒ 信号入出力ケーブル2本

“直列回路”の感じ

Agenda

1. モーターの構成や基本仕様
2. モーター開発キットの概要と使い方
3. モーター開発用のSDK
   - 3.1 開発用SDKの概要
   - 3.2 サンプルプログラムの説明
   - 3.3 データの読み書き
   - 3.4 減速機の説明
   - 3.5 複数モーターの連動
4. よくある質問

3.1 開発用SDKの概要



大切

1. 利用できるOS : Linux, Windows10
2. 利用できるプロセッサ : intel, amd64, arm32, arm64
3. 開発環境 : C, C++, Python, ROS

headファイル:

“LSerial.h”: シリアルポートの配置に関する変数

“motor_msg.h”: モーターとの通信に関する構造体

“motor_ctrl.h”: encodingやdecodingに関する構造体

ChangeID_Tool: モーターの番号を指定するツール

詳しくは ReadMe.mdファイルをご覧になってください

3.2 モーターとの通信に関する構造体 -- 受信

motor_msg.hで定義されたデータを受け取る構造体

```
typedef struct
{
    ServoComdDataV3 motor_recv_data;     //The data received from motor. Details are shown in motor_msg.h
    int hex_len;                         //The Bytes count of the received message, it should be 78 for this motor
    long long resv_time;                 //The time of receiving this message, microsecond(us)
    int correct;                         //Whether the received data is correct(1:correct, 0:wrong)
    unsigned char motor_id;              //モーターID: 0 or 1 or 2
    unsigned char mode;                  //The control mode, 0:free, 5:Open loop slow turning, 10:close loop control
    int Temp;                            //Temperature
    unsigned char MError;                //Error code

    float T;                             //The output torque of motor
    float W;                             //The motor shaft speed(without filter)
    float LW;                            //The motor shaft speed(with filter)
    int Acc;                             //The acceleration of motor shaft
    float Pos;                           //The motor shaft position(control board zero fixed)

    float gyro[3];                       //The data of 6 axis inertial sensor on the control board
    float acc[3];

}MOTOR_recv;
```

トルク制御：実際の出力トルク

速度制御: モーターの角速度

位置制御: 角度

motorからのデータを受信に関する構造体MOTOR_recv

3.2 モーターとの通信に関する構造体 -- 送信


```
//Define the message to be send.
typedef struct {
    MasterComdDataV3  motor_send_data;  //The data to be sent to motor. Details are shown in motor_msg.h
    int hex_len;                        //The Bytes count of the message to be sent, it should be 34 for this motor
    long long send_time;                //The time that message was sent
    //The values of motor commands
    unsigned short id;                  //Motor ID
    unsigned short mode;                //The control mode, 0:free, 5:Open loop slow turning, 10:close loop control
    //The following parameters are just motor's parameters, do not concern the reducer.
    //the following parameters all related to motor, no related to reducer
    //The real torque command to control board is:  K_P*delta_Pos + K_W*delta_W + T
    float T;                            //Desired output torque of motor
    float W;                            //Desired output speed of motor
    float Pos;                          //Desired shaft position of motor
    float K_P;                          //The position stiffness
    float K_W;                          //The speed stiffness
}MOTOR_send;
```

motor_msg.hで定義されたデータを送る構造体

トルク制御：目標トルク

速度制御: 目標角速度

位置制御: 目標角度

Motorへのデータを送信に関する構造体MOTOR_send

剛性可変
- K_Pは位置に関する剛性係数　K_P = 0.1をお勧め
- K_Wは速度に関する剛性係数　K_W = 3をお勧め

3.3 サンプルプログラムの説明 – 変数の宣言と設定

```c
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>    //usleep()
#include <sys/time.h>
#include "LSerial.h"    //serial port communication
#include "motor_ctrl.h"

enum motor_command_type
{
    TORQUE,
    POSITION,
    VELOCITY
};

int main()
{
    int com_type = VELOCITY;
    float goal_T;
    float goal_W;
    float goal_Pos;
    float goal_K_W;
    float goal_K_P;

    if(com_type == TORQUE)
    {
        goal_T = 0;
        goal_W = 0;
        goal_Pos = 0;
        goal_K_W = 0;
        goal_K_P = 0;
    }
    else if(com_type == POSITION)
    {
        goal_T = 0;
        goal_W = 0;       //have to be 0
        goal_Pos = 0;
        goal_K_W = 3;       //K_W works as a dampping
        goal_K_P = 0.1;
    }
```

```c
    else if(com_type == VELOCITY)
    {
        goal_T = 0;
        goal_W = 50.0;
        goal_Pos = 0;
        goal_K_W = 3;
        goal_K_P = 0;
    }
    else
    {
        printf("motor command type error!\n");
        return 1;
    }

    //Send
    MOTOR_send motor_s, motor_s1;
    // long long start_time = 0;
    motor_s.id = 0;                 //motor ID
    motor_s.mode = 10;              //switch to servo mode
    motor_s.T = goal_T;            //Nm, T<255.9
    motor_s.W = goal_W;              //rad/s, W<511.9
    motor_s.Pos = goal_Pos;          //rad, Pos<131071.9
    motor_s.K_P = goal_K_P;        //K_P<31.9     value should around 0.1
    motor_s.K_W = goal_K_W;        //K_W<63.9     value should around 3

    motor_s1.id = 0;
    motor_s1.mode = 0;
    //Receive
    MOTOR_recv motor_r;
    //over heat protection
    const float safe_torque = 0.7;
    const float cooldown_torque = 0.5;
    const long long overload_duration = 70000000;    //microsecond(us)
    const long long cooldown_duration = 600000000;    //microsecond(us)
    float safe_ratio = 1;
    // float past_torque = 0;
    // int overload_flag = 0;
    long long current = 0;
    long long overload_start = 0;
    long long cooldown_start = 0;
    enum motor_status
    {
        NORMAL,
        OVERHEAT,
        COOLDOWN
    };
printf("87\n");
```

3.3 サンプルプログラムの説明 – 通信を繋げる、安全保護モード


```
95  printf("95\n");
96      int status = NORMAL;
97
98      modify_data(&motor_s);
99      modify_data(&motor_s1);
100
101     send_recv(fd, &motor_s, &motor_r);
102     send_recv(fd, &motor_s1, &motor_r);
103 printf("102\n");
104     extract_data(&motor_r);
105     printf("START\n");
106
107     for(int i=0; i<500; i++)
108     {
109         motor_s.T = goal_T;
110         motor_s.K_P = goal_K_P;
111         motor_s.K_W = goal_K_W;
112         modify_data(&motor_s);
113
114         if(status == NORMAL)
115         {
116             if(motor_r.T > safe_torque)
117             {
118                 overload_start = getSystemTime();
119                 status = OVERHEAT;
120             }
121         }
122         else if(status == OVERHEAT)
123         {
124             current = getSystemTime();
125             if(motor_r.T < safe_torque)
126             {
127                 status = NORMAL;
128             }
129             else if((current - overload_start) > overload_duration)
130             {
131                 cooldown_start = getSystemTime();
132                 status = COOLDOWN;
133             }
134         }
```

```
135         else if(status == COOLDOWN)
136         {
137             current = getSystemTime();
138             if((current - cooldown_start) > cooldown_duration)
139             {
140                 status = NORMAL;
141             }
142             else
143             {
144                 safe_ratio = cooldown_torque / motor_r.T;
145                 motor_s.T = safe_ratio * goal_T;
146                 motor_s.K_P = safe_ratio * goal_K_P;
147                 motor_s.K_W = safe_ratio * goal_K_W;
148                 modify_data(&motor_s);
149             }
150         }
151         printf("*****************\n");
152         printf("Torque command: %f\n", motor_s.T);
153         send_recv(fd, &motor_s, &motor_r);
154         extract_data(&motor_r);
155
156         printf("status: %d\n", status);
157         // printf("pos: %f\n", motor_r.Pos);
158         printf("w: %f\n", motor_r.LW);
159         printf("T: %f\n", motor_r.T);
160         usleep(500000);
161     }
162
163     send_recv(fd, &motor_s1, &motor_r);
164     extract_data(&motor_r);
165     // show_resv_data_hex(&motor_r);
166     printf("END\n");
167     printf("ID: %d\n", motor_r.motor_id);
168     // show_resv_data(&motor_r);
169
170     close_serial(fd);
171 #if defined(__WIN32__)
172     system("pause");
173 #endif
174
175     return 0;
176 }
```

位置に関する剛性係数K_P や 速度に関する剛性係数K_W

モーターのbackdrivable / backdrivability について

https://www.mi.ams.eng.osaka-u.ac.jp/research-mechatronics-e.html

https://www.mi.ams.eng.osaka-u.ac.jp/research-mechatronics-j.html

よくある質問

Q1: モーター内蔵のドライバの制御則や通信周期を編集できますか？
A1: モータードライバ部分の制御則や通信周期を編集できません。

Q2: 最大、何個のモータまで接続可能でしょうか？
A2:四足歩行ロボットA1は12個モーターを入ります。三つモーターが一体を送受信となって、12個モーターや18個モーターの接続や連動は大丈夫と思います。詳しくはincludeフォルダで“motor_msg.h”ファイルのComHead構造体の中に、motorIDという変数の説明をご覧になってください。

よくある質問

Q3: stiffnessとdampingを0にしたとしても，ギアの摩擦やコギングの影響で，外力を加えて出力軸をスムーズに回せるということは無いようなのですが，

もっとローカルな制御の部分を修正をユーザーができるようにするというようなことは今後も想定されていませんか？

A3:

よくある質問

Q4:できればA1 motorを組み合わせて自分でロボットを作成したいと考えていまして，そのような場合に，どこまでプログラムをカスタムして使えるのかというところが気になりました．

A4:

1. Motor_msg.h → ServoComdV3構造体 →
2. 安全保護モードの定義や設定ができます。

よくある質問

Q5: unitree_legged_sdkはA1脚ロボット用のsdkだと思うのですが，今回のようにモータを1つだけ購入した場合には使用できないという理解で正しいでしょうか？

A5: 正しいです。

	unitree_legged_sdk	モーター専用sdk
モーター通信方式RS485に対応する	不可	可
motorIDを自由に設定	不可	可
ハイブリッド制御	可	可
モーターに関する全部のデータを読み込み	不十分	十分
Python	不可	可
Windows	不可	可
最適な開発言語	C++	C
安全保護モードの設定	不可	可