

串口总线舵机SDK使用手册(Arduino Mega2560)

串口总线舵机SDK使用手册(Arduino Mega2560)

安装串口总线舵机的Arduino库

演示例程的操作流程

舵机对象的创建与初始化

舵机通讯检测

API- ping

例程源码

舵机阻尼模式

API- setDamping

例程源码

舵机角度查询

API- queryAngle

例程源码

舵机轮式模式

API- wheelStop

API- wheelRun

API- wheelRunNTime

API- wheelRunNCircle

例程源码

设置舵机角度

API- setAngle

API- setRawAngleByInterval

API- setRawAngleByVelocity

API- isStop

API- setRange

例程源码

舵机阻塞式等待

API- wait

例程源码

设置舵机角度-多圈模式

API- setRawAngleMTurn

API- setRawAngleByInterval

API- setRawAngleMTurnByVelocity

例程源码

舵机扭力开关

API- setTorque

例程源码

舵机标定

API- calibration

API- angleReal2Raw

API- angleRaw2Real

例程源码

舵机转速设置

API- setSpeed

舵机数据读取

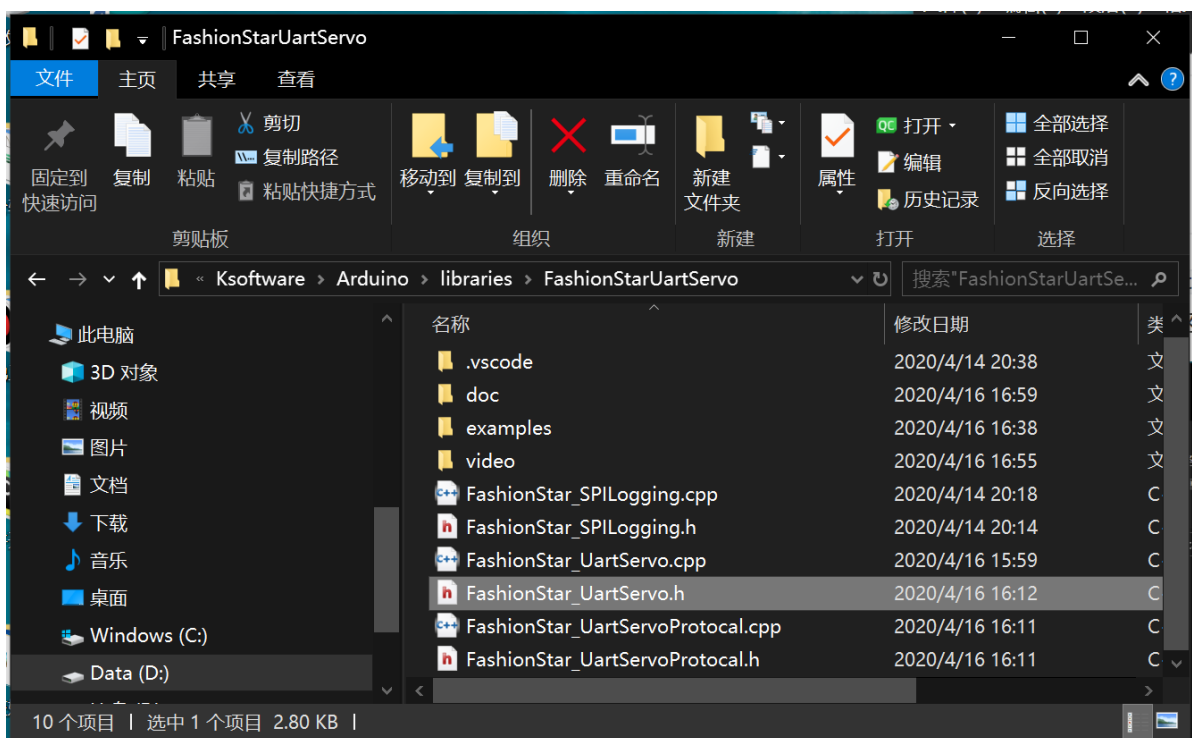
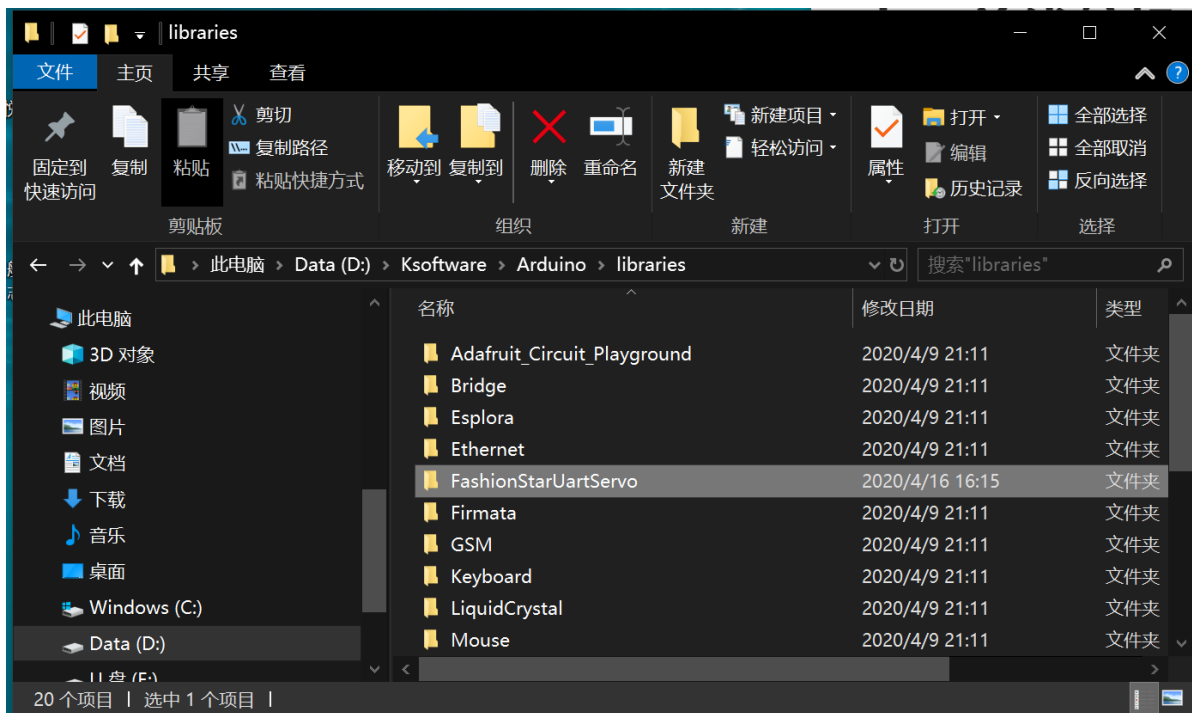
示例源码

邮箱: kyle.xing@fashionstar.com.hk

更新时间: 2021 / 07 / 15

安装串口总线舵机的Arduino库

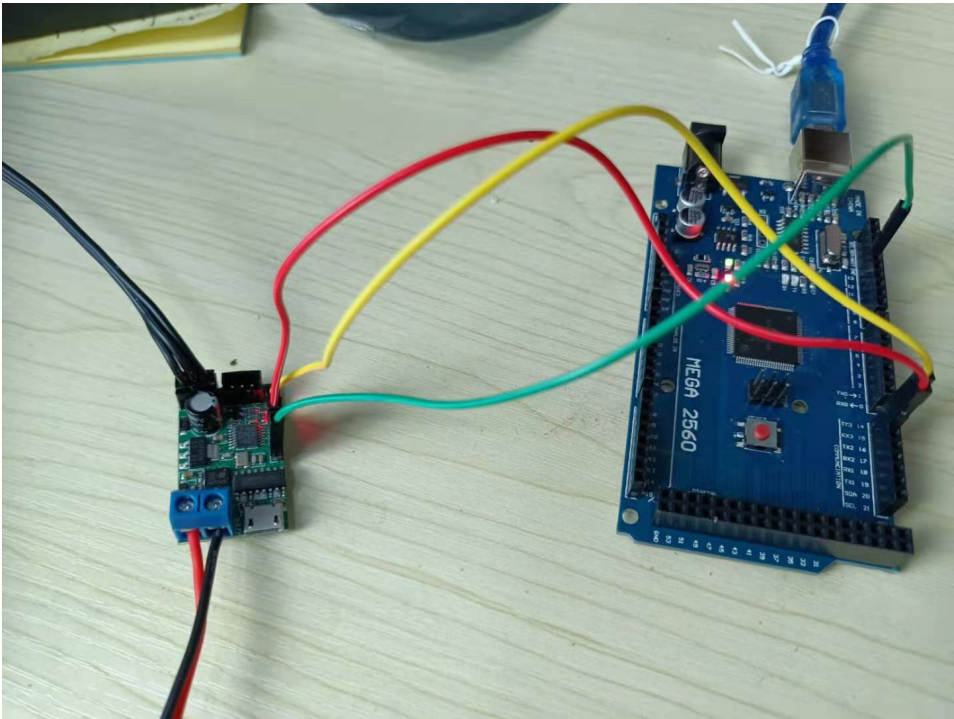
将 FashionStar_UartServo 这个工程文件, 整体拷贝到 Arduino IDE 安装路径下的 libraries 这个文件夹.



演示例程的操作流程

1. 接线 - Arduino Mega2560跟串口转接板UART接口

Arduino Mega2560	USB转TTL模块
D15(串口3 RX 接收端)	Tx (USB转TTL模块的接收端)
D14 (串口3 Tx 发送端)	Rx (USB转TTL模块的发送端)
GND	GND



2. 接线 - Arduino Mega2560与PC 通过USB线相连接

3. 在PC端打开Arduino IDE, 打开FashionStar串口总线舵机的例程文件.

[打开Arduino示例代码-流程演示.mp4](#)



4. 选择开发板型号为Arduino Mega2560, 选择端口号

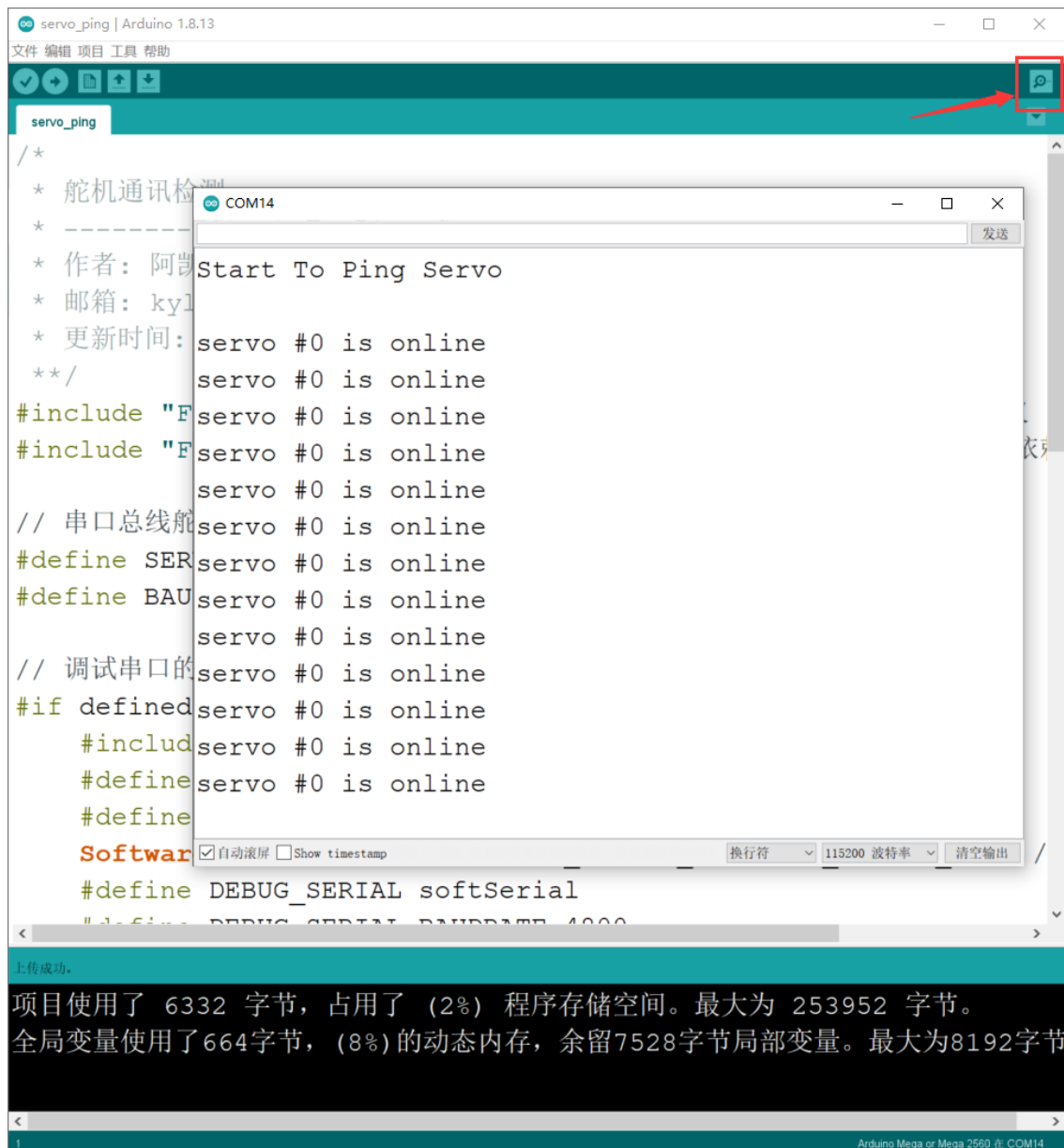


5. 给串口舵机转接板供电，电压7.2V.

6. 编译并烧录固件



7. 查看日志输出



舵机对象的创建与初始化

```
1 #include "FashionStar_UartServoProtocol.h" // 串口总线舵机通信协议
2 #include "FashionStar_UartServo.h" // 串口总线舵机SDK
```

`FashionStar_UartServoProtocol` 用来处理舵机的底层通信协议的逻辑(数据帧的收发, 数据校验等).

`FashionStar_UartServo` 是舵机的SDK, 是在协议上层的更高一级的封装.

创建一个串口总线舵机通信协议对象 `FSUS_Protocol`, 构造器里面需要填写Arduino与串口总线舵机通信的波特率, 默认为 115200.

```
1 #define BAUDRATE 115200 // 波特率
2
3 FSUS_Protocol protocol(BAUDRATE); //协议
```

创建一个 `FSUS_Servo` 舵机对象, 创建的时候需要传入舵机的ID, 以及通信协议对象的指针 `&protocol`.

舵机的ID取值范围为 0-254

```
1 #define SERVO_ID 0 //舵机ID号
2
3 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
```

接下来需要在 setup() 函数里对通信协议对象以及舵机对象进行初始化

```
1 void setup(){
2     ...
3     protocol.init(); // 舵机通信协议初始化
4     uservo.init(); // 串口总线舵机初始化
5     ...
6 }
7
```

舵机通讯检测

API-ping

调用舵机的 ping() 函数用于舵机的通信检测, 判断舵机是否在线.

```
1 bool isOnline = uservo.ping(); // 舵机通讯检测
```

例程源码

servo_ping.ino

```
1  /*
2   * 舵机通讯检测
3   * -----
4   * 作者: 阿凯|kyle
5   * 邮箱: kyle.xing@fashionstar.com.hk
6   * 更新时间: 2021/06/02
7   */
8  #include "FashionStar_UartServoProtocol.h" // 串口总线舵机通信协议
9  #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
10
11 // 串口总线舵机配置
12 #define SERVO_ID 0 //舵机ID号
13 #define BAUDRATE 115200 // 波特率
14
15 // 调试串口的配置
16 #if defined(ARDUINO_AVR_UNO)
17     #include <SoftwareSerial.h>
18     #define SOFT_SERIAL_RX 6
19     #define SOFT_SERIAL_TX 7
20     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
21     #define DEBUG_SERIAL softSerial
22     #define DEBUG_SERIAL_BAUDRATE 4800
23 #elif defined(ARDUINO_AVR_MEGA2560)
```

```

24     #define DEBUG_SERIAL Serial
25     #define DEBUG_SERIAL_BAUDRATE 115200
26 #elif defined(ARDUINO_ARCH_ESP32)
27     #define DEBUG_SERIAL Serial
28     #define DEBUG_SERIAL_BAUDRATE 115200
29 #endif
30
31 FSUS_Protocol protocol(BAUDRATE);          //协议
32 FSUS_Servo uservo(SERVO_ID, &protocol);    // 创建舵机
33
34 void setup(){
35     protocol.init(); // 舵机通信协议初始化
36     uservo.init(); // 串口总线舵机初始化
37     // 打印例程信息
38     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE);
39     DEBUG_SERIAL.println("Start To Ping Servo\n");
40 }
41
42 void loop(){
43     bool isOnline = uservo.ping(); // 舵机通讯检测
44     String message = "servo #" + String(uservo.servoId, DEC) + " is "; // 日志
45     // 输出
46     if(isOnline){
47         message += "online";
48     }else{
49         message += "offline";
50     }
51     // 调试串口初始化
52     DEBUG_SERIAL.println(message);
53     // 等待1s
54     delay(1000);
55 }

```

输出日志

```

1  Start To Ping Servo
2
3
4  servo #0 is online.
5
6  servo #0 is online.
7
8  servo #0 is online.
9
10 servo #0 is online.
11

```

舵机阻尼模式

API- setDamping

设置舵机为阻尼模式.

```
1 void FSUS_Servo::setDamping(FSUS_POWER_T power)
```

输入参数

- `power` 舵机的功率，单位为mW. 功率值越大，旋转舵机的时候阻尼力也就越大

使用示例

```
1 #define DAMPING_POWER 800 // 阻尼模式下的功率(单位mw) 500,800,1000
2
3 uservo.setDamping(DAMPING_POWER);
```

例程源码

servo_dampping.ino

```
1  /*
2   * 设置舵机为阻尼模式
3   * 调整参数`DAMPING_POWER`感受不同的阻尼力
4   * -----
5   * 作者：阿凯|kyle
6   * 邮箱：kyle.xing@fashionstar.com.hk
7   * 更新时间：2021/06/02
8   */
9  #include "FashionStar_UartServoProtocol.h"
10 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
11
12 // 串口总线舵机配置参数
13 #define SERVO_ID 0 //舵机ID号
14 #define BAUDRATE 115200 // 波特率
15 #define DAMPING_POWER 800 // 阻尼模式下的功率(单位mw) 500,800,1000
16
17 // 调试串口的配置
18 #if defined(ARDUINO_AVR_UNO)
19     #include <SoftwareSerial.h>
20     #define SOFT_SERIAL_RX 6
21     #define SOFT_SERIAL_TX 7
22     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
23     #define DEBUG_SERIAL softSerial
24     #define DEBUG_SERIAL_BAUDRATE 4800
25 #elif defined(ARDUINO_AVR_MEGA2560)
26     #define DEBUG_SERIAL Serial
27     #define DEBUG_SERIAL_BAUDRATE 115200
28 #elif defined(ARDUINO_ARCH_ESP32)
29     #define DEBUG_SERIAL Serial
30     #define DEBUG_SERIAL_BAUDRATE 115200
31 #endif
32
```

```

33 FSUS_Protocol protocol(BAUDRATE); //协议
34 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
35
36 void setup(){
37
38     protocol.init(); // 通信协议初始化
39     uservo.init(); // 舵机初始化
40     // 打印日志
41     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE);
42     DEBUG_SERIAL.println("Set Servo Mode To Damping");
43     // 设置电机的阻尼系数
44     uservo.setDamping(DAMPING_POWER);
45 }
46
47 void loop(){
48     // TODO;
49 }
50

```

日志输出

```

1 | Set Servo Mode To Damping

```

舵机角度查询

API-`queryAngle`

查询舵机当前的真实角度，向舵机发送角度查询指令，并将角度值赋值给舵机对象的 `curAngle` 属性

```

1 | FSUS_SERVO_ANGLE_T FSUS_Servo::queryAngle()

```

输入参数

- <无>

输出参数

- `curAngle` 舵机当前的真实角度

使用示例

示例1

```

1 | float curAngle = uservo.queryAngle()

```

示例2

```

1 | // 舵机角度查询（更新角度）
2 | uservo.queryAngle();
3 | // 通过.curAngle访问当前的真实角度
4 | uservo.curAngle

```

例程源码

servo_query_angle.ino

```
1  /*
2   * 舵机角度回读实验
3   * 用手掰动舵机，角度回读并将角度读数通过SPI发送
4   * -----
5   * 作者：阿凯|Kyle
6   * 邮箱：kyle.xing@fashionstar.com.hk
7   * 更新时间：2021/06/02
8   */
9  #include "FashionStar_UartServoProtocol.h"
10 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
11
12 // 串口总线舵机配置
13 #define SERVO_ID 0 //舵机ID号
14 #define DAMPING_POWER 800 // 阻尼模式下的功率(单位mw) 500,800,1000
15 #define BAUDRATE 115200 // 波特率
16
17 // 调试串口的配置
18 #if defined(ARDUINO_AVR_UNO)
19     #include <SoftwareSerial.h>
20     #define SOFT_SERIAL_RX 6
21     #define SOFT_SERIAL_TX 7
22     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
23     #define DEBUG_SERIAL softSerial
24     #define DEBUG_SERIAL_BAUDRATE 4800
25 #elif defined(ARDUINO_AVR_MEGA2560)
26     #define DEBUG_SERIAL Serial
27     #define DEBUG_SERIAL_BAUDRATE 115200
28 #elif defined(ARDUINO_ARCH_ESP32)
29     #define DEBUG_SERIAL Serial
30     #define DEBUG_SERIAL_BAUDRATE 115200
31 #endif
32
33
34
35 FSUS_Protocol protocol(BAUDRATE); //协议
36 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
37
38
39 void setup(){
40     protocol.init(); // 通信协议初始化
41     uservo.init(); // 舵机角度初始化
42     uservo.setDamping(DAMPING_POWER); // 舵机设置为阻尼模式
43     // 打印例程信息
44     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE);
45     DEBUG_SERIAL.println("Query Servo Angle\n");
46 }
47
48 void loop(){
49     // 舵机角度查询 (更新角度)
50     uservo.queryRawAngle();
51     // 日志输出
```

```

52     String message = "Status Code: " + String(uservo.protocol-
>responsePack.recv_status, DEC) + " servo #" + String(uservo.servoId, DEC) + "
, Current Angle = " + String(uservo.curRawAngle, 1) + " deg";
53     DEBUG_SERIAL.println(message);
54     // 等待1s
55     delay(1000);
56 }

```

输出日志

```

1  Query Servo Angle
2
3  Status Code: 0 servo #0 , Current Angle = -99.0
4
5  Status Code: 0 servo #0 , Current Angle = -99.0
6
7  Status Code: 0 servo #0 , Current Angle = -99.0
8

```

舵机轮式模式

API-wheelStop

轮式模式, 停止旋转

函数原型

```

1  void FSUS_Servo::wheelStop()

```

输入参数

<无>

API-wheelRun

轮子持续旋转

函数原型

```

1  void FSUS_Servo::wheelRun(uint8_t is_cw)

```

输入参数

- `is_cw` 轮子的旋转方向
 - 0 : 逆时针
 - 1 : 顺时针

API-wheelRunNTime

轮子旋转特定的时间

函数原型

```
1 void FSUS_Servo::wheelRunNTime(uint8_t is_cw, uint16_t time_ms)
```

输入参数

- `is_cw`: 轮子的旋转方向
 - 0: 逆时针
 - 1: 顺时针
- `time_ms`: 持续旋转的时间, 单位为ms

API-wheelRunNCircle

轮子旋转特定的圈数

函数原型

```
1 void FSUS_Servo::wheelRunNCircle(uint8_t is_cw, uint16_t circle_num)
```

输入参数

- `is_cw`: 轮子的旋转方向
 - 0: 逆时针
 - 1: 顺时针
- `circle_num`: 轮子旋转的圈数

例程源码

servo_wheel_mode.ino

```
1  /*
2   * 测试舵机轮式模式
3   * 提示: 拓展板上电之后, 记得按下Arduino的RESET按键
4   * -----
5   * 作者: 阿凯|kyle
6   * 邮箱: kyle.xing@fashionstar.com.hk
7   * 更新时间: 2021/06/02
8   */
9  #include "FashionStar_UartServoProtocol.h"
10 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
11
12 // 配置参数
13 #define BAUDRATE 115200 // 波特率
14 #define SERVO_ID 0 // 舵机ID号
15
16 FSUS_Protocol protocol(BAUDRATE); // 协议
17 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
18
19 /* 轮子持续旋转指令与停止指令测试 */
```

```

20 void testWheelRunAndStop(){
21     uservo.wheelRun(FSUS_CCW); // 轮子持续旋转，方向为逆时针
22     delay(2000);                // 等待2s
23     uservo.wheelStop();
24     delay(2000);                // 等待2s
25     uservo.wheelRun(FSUS_CW); // 轮子持续旋转
26     delay(2000);                // 等待2s
27     uservo.wheelStop();
28     delay(2000);                // 等待2s
29 }
30
31 /* 测试轮子旋转特定的时间 */
32 void testWheelRunNTime(){
33     uservo.wheelRunNTime(FSUS_CW, 5000); // 轮子持续旋转5s(顺时针)
34     delay(5000);
35     uservo.wheelRunNTime(FSUS_CCW, 5000); // 轮子持续旋转5s(逆时针)
36     delay(5000);
37 }
38
39 /* 测试轮子旋转特定的圈数 */
40 void testWheelRunNCircle(){
41     uint16_t nCircle = 2; // 旋转圈数
42     uint16_t delayMsEstimate = (uint16_t)(360.0 * nCircle / uservo.speed *
1000); // 估计旋转的时间
43     uservo.wheelRunNCircle(FSUS_CW, 2); // 轮子持续旋转2圈(顺时针)
44     delay(delayMsEstimate);            // 等到轮子旋转到特定的位置
45
46     uservo.wheelRunNCircle(FSUS_CCW, 2); // 轮子持续旋转2圈(逆时针)
47     delay(delayMsEstimate);            // 等到轮子旋转到特定的位置}
48 }
49
50 void setup(){
51     protocol.init();                // 通信协议初始化
52     uservo.init();                  // 舵机角度初始化
53     uservo.setSpeed(100);           // 设置转速为20°/s
54
55     // 测试持续旋转与停止
56     // testRunAndStop();
57
58     // 测试旋转特定的时间
59     // testWheelRunNTime();
60
61     // 测试旋转特定的圈数
62     testWheelRunNCircle();
63 }
64
65 void loop(){
66 }

```

设置舵机角度

API-`setAngle`

设定舵机的角度

函数原型

```
1  /* 设置舵机的原始角度 */
2  void FSUS_Servo::setRawAngle(FSUS_SERVO_ANGLE_T rawAngle, FSUS_INTERVAL_T
    interval, FSUS_POWER_T power)
```

```
1  /* 设置舵机的原始角度 */
2  void FSUS_Servo::setRawAngle(FSUS_SERVO_ANGLE_T rawAngle, FSUS_INTERVAL_T
    interval)
```

```
1  /* 设置舵机的原始角度 */
2  void FSUS_Servo::setRawAngle(FSUS_SERVO_ANGLE_T rawAngle)
```

输入参数

- `rawAngle` : 舵机的目标角度, 单位°
- `interval` 舵机旋转的周期, 单位ms
- `power` 最大功率, 单位mW

API-`setRawAngleByInterval`

函数原型

```
1  // 设置舵机的原始角度(指定周期)
2  void FSUS_Servo::setRawAngleByInterval(FSUS_SERVO_ANGLE_T rawAngle,
    FSUS_INTERVAL_T interval, FSUS_INTERVAL_T t_acc, FSUS_INTERVAL_T t_dec,
    FSUS_POWER_T power)
```

输入参数

- `rawAngle` : 舵机的目标角度, 单位°
- `interval` : 舵机旋转的周期, 单位ms
- `t_acc` : 加速时间
- `t_dec` : 减速时间
- `power` : 最大功率, 单位mW

API-`setRawAngleByVelocity`

函数原型

```
1 // 设定舵机的原始角度(指定转速)
2 void FSUS_Servo::setRawAngleByVelocity(FSUS_SERVO_ANGLE_T rawAngle,
    FSUS_SERVO_SPEED_T velocity, FSUS_INTERVAL_T t_acc, FSUS_INTERVAL_T t_dec,
    FSUS_POWER_T power)
```

输入参数

- `rawAngle` : 舵机的目标角度, 单位°
- `velocity` : 舵机旋转的转速, 单位°/s
- `t_acc` : 加速时间
- `t_dec` : 减速时间
- `power` : 最大功率, 单位mW

API- `isStop`

判断舵机是否在旋转, 是否是静止.

改函数在执行的时候, 会先查询舵机当前的角度, 返回对比跟目标角度 `targetAngle` 之间的差值是否小于控制死区.

函数原型

```
1 bool FSUS_Servo::isStop()
```

输入参数

<无>

返回参数

- `is_stop`:
 - `true` 舵机已经到达目标角度, 停下来了
 - `false` 舵机还没有到达目标角度, 正在旋转

API- `setRange`

设置舵机的角度范围

函数原型

```
1 void FSUS_Servo::setAngleRange(FSUS_SERVO_ANGLE_T minAngle,
    FSUS_SERVO_ANGLE_T maxAngle)
```

输入参数

- `minAngle` : 舵机角度下限
- `maxAngle` : 舵机角度上限

输出参数

<无>

例程源码

```
1  /*
2   * 设置舵机的角度(单圈模式)
3   * 提示：拓展板上电之后，记得按下Arduino的RESET按键
4   * -----
5   * 作者：阿凯|kyle
6   * 邮箱：kyle.xing@fashionstar.com.hk
7   * 更新时间：2021/06/02
8   */
9
10 #include "FashionStar_UartServoProtocol.h"
11 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
12
13 // 串口总线舵机配置参数
14 #define SERVO_ID 0 //舵机ID号
15 #define BAUDRATE 115200 // 波特率
16
17 // 调试串口的配置
18 #if defined(ARDUINO_AVR_UNO)
19     #include <SoftwareSerial.h>
20     #define SOFT_SERIAL_RX 6
21     #define SOFT_SERIAL_TX 7
22     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
23     #define DEBUG_SERIAL softSerial
24     #define DEBUG_SERIAL_BAUDRATE 4800
25 #elif defined(ARDUINO_AVR_MEGA2560)
26     #define DEBUG_SERIAL Serial
27     #define DEBUG_SERIAL_BAUDRATE 115200
28 #elif defined(ARDUINO_ARCH_ESP32)
29     #define DEBUG_SERIAL Serial
30     #define DEBUG_SERIAL_BAUDRATE 115200
31 #endif
32
33 FSUS_Protocol protocol(BAUDRATE); //协议
34 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
35
36 uint16_t interval; // 运行周期 单位ms
37 uint16_t t_acc; // 加速时间 单位ms
38 uint16_t t_dec; // 减速时间 单位ms
39 float velocity; // 目标转速 单位°/s
40
41 /* 等待并报告当前的角度*/
42 void waitAndReport(){
43     uservo.wait(); // 等待舵机旋转到目标角度
44     DEBUG_SERIAL.println("Real Angle = " + String(uservo.curRawAngle, 1) + "
45 Target Angle = "+String(uservo.targetRawAngle, 1));
46     delay(2000); // 暂停2s
47 }
48
49 void setup(){
50     protocol.init(); // 通信协议初始化
51     uservo.init(); //舵机角度初始化
52     // 打印例程信息
```

```

53     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE); // 初始化软串口的波特率
54     DEBUG_SERIAL.println("Set Servo Angle");
55 }
56
57 void loop(){
58     DEBUG_SERIAL.println("Set Angle = 90°");
59     uservo.setRawAngle(90.0); // 设置舵机的角度
60     waitAndReport();
61
62     DEBUG_SERIAL.println("Set Angle = -90°");
63     uservo.setRawAngle(-90);
64     waitAndReport();
65
66     DEBUG_SERIAL.println("Set Angle = 90° - Set Interval = 500ms");
67     interval = 1000;
68     t_acc = 100;
69     t_dec = 100;
70     uservo.setRawAngleByInterval(90, interval, t_acc, t_dec, 0);
71     waitAndReport();
72
73     DEBUG_SERIAL.println("Set Angle = -90° - Set Velocity = 200°/s");
74     velocity = 200.0;
75     t_acc = 100;
76     t_dec = 100;
77     uservo.setRawAngleByVelocity(-90, velocity, t_acc, t_dec, 0);
78     waitAndReport();
79 }

```

输出日志

```

1  Set Angle = 90°
2  Real Angle = 89.7 Target Angle = 90.0
3  Set Angle = -90°
4  Real Angle = -89.6 Target Angle = -90.0
5  Set Angle = 90° - Set Interval = 500ms
6  Real Angle = 89.7 Target Angle = 90.0
7  Set Angle = -90° - Set Velocity = 200°/s
8  Real Angle = -89.6 Target Angle = -90.0

```

舵机阻塞式等待

API-wait

等待舵机旋转到目标角度, 阻塞式.

函数原型

```

1  void FSUS_Servo::wait()

```

输入参数

<无>

输出参数

<无>

例程源码

servo_wait.ino

```
1  /*
2   * 测试wait()函数,轮询角度直到舵机旋转到目标位置
3   * 提示: 拓展板上电之后, 记得按下Arduino的RESET按键
4   * -----
5   * 作者: 阿凯|kyle
6   * 邮箱: kyle.xing@fashionstar.com.hk
7   * 更新时间: 2021/06/02
8   */
9  #include "FashionStar_UartServoProtocol.h"
10 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
11
12 // 串口总线舵机配置参数
13 #define SERVO_ID 0 //舵机ID号
14 #define BAUDRATE 115200 // 波特率
15
16 // 调试串口的配置
17 #if defined(ARDUINO_AVR_UNO)
18     #include <SoftwareSerial.h>
19     #define SOFT_SERIAL_RX 6
20     #define SOFT_SERIAL_TX 7
21     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
22     #define DEBUG_SERIAL softSerial
23     #define DEBUG_SERIAL_BAUDRATE 4800
24 #elif defined(ARDUINO_AVR_MEGA2560)
25     #define DEBUG_SERIAL Serial
26     #define DEBUG_SERIAL_BAUDRATE 115200
27 #elif defined(ARDUINO_ARCH_ESP32)
28     #define DEBUG_SERIAL Serial
29     #define DEBUG_SERIAL_BAUDRATE 115200
30 #endif
31
32 FSUS_Protocol protocol(BAUDRATE); //协议
33 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
34
35 void setup(){
36
37     protocol.init(); // 通信协议初始化
38     uservo.init(); //舵机初始化
39     // 打印例程信息
40     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE);
41     DEBUG_SERIAL.println("Test wait");
42 }
43
44 void loop(){
45     DEBUG_SERIAL.println("Set Angle = 90.0");
46     uservo.setAngle(90.0); // 设置舵机的角度
47     uservo.wait();
48     DEBUG_SERIAL.println("Real Angle = "+String(uservo.curAngle, 2));
49 }
```

```

50     DEBUG_SERIAL.println("Set Angle = -90.0");
51     uservo.setAngle(-90);
52     uservo.wait();
53     DEBUG_SERIAL.println("Real Angle = "+String(uservo.curAngle, 2));
54 }

```

输出日志

```

1  Set Angle = -90.0
2  Real Angle = -89.00
3  Set Angle = 90.0
4  Real Angle = 89.80
5  Set Angle = -90.0
6  Real Angle = -89.00
7  Set Angle = 90.0
8  Real Angle = 89.80
9  Set Angle = -90.0
10 Real Angle = -89.00
11 Set Angle = 90.0
12 Real Angle = 89.80
13 Set Angle = -90.0
14 Real Angle = -89.00

```

设置舵机角度-多圈模式

API-`setRawAngleMTurn`

函数原型

```

1  // 设定舵机的原始角度(多圈)
2  void FSUS_Servo::setRawAngleMTurn(FSUS_SERVO_ANGLE_T rawAngle,
    FSUS_INTERVAL_T_MTURN interval, FSUS_POWER_T power)

```

```

1  // 设定舵机的原始角度(多圈)
2  void FSUS_Servo::setRawAngleMTurn(FSUS_SERVO_ANGLE_T rawAngle,
    FSUS_INTERVAL_T_MTURN interval)

```

```

1  // 设定舵机的原始角度(多圈)
2  void FSUS_Servo::setRawAngleMTurn(FSUS_SERVO_ANGLE_T rawAngle)

```

输入参数

- `rawAngle` : 舵机的目标角度, 单位°
- `interval` 舵机旋转的周期, 单位ms
- `power` 最大功率, 单位mW

输出参数

<无>

API- setRawAngleByInterval

函数原型

```
1 // 设定舵机的原始角度(多圈+指定周期)
2 void FSUS_Servo::setRawAngleMTurnByInterval(FSUS_SERVO_ANGLE_T rawAngle,
    FSUS_INTERVAL_T_MTURN interval, FSUS_INTERVAL_T t_acc, FSUS_INTERVAL_T t_dec,
    FSUS_POWER_T power)
```

输入参数

- `rawAngle` : 舵机的目标角度, 单位°
- `interval` 舵机旋转的周期, 单位ms
- `t_acc` 加速时间, 单位ms
- `t_dec` 减速时间, 单位ms
- `power` 最大功率, 单位mW

输出参数

<无>

API- setRawAngleMTurnByVelocity

函数原型

```
1 // 设定舵机的原始角度(多圈+指定转速)
2 void FSUS_Servo::setRawAngleMTurnByVelocity(FSUS_SERVO_ANGLE_T rawAngle,
    FSUS_SERVO_SPEED_T velocity, FSUS_INTERVAL_T t_acc, FSUS_INTERVAL_T t_dec,
    FSUS_POWER_T power)
```

输入参数

- `rawAngle` : 舵机的目标角度, 单位°
- `velocity` : 舵机旋转的速度, 单位°/s
- `t_acc` : 加速时间, 单位ms
- `t_dec` : 减速时间, 单位ms
- `power` : 最大功率, 单位mW

输出参数

<无>

例程源码

servo_set_angle_mturn.ino

```
1  /*
2   * 设置舵机的角度(多圈模式)
3   * 提示：拓展板上电之后，记得按下Arduino的RESET按键
4   * -----
5   * 作者：阿凯|Kyle
6   * 邮箱：kyle.xing@fashionstar.com.hk
7   * 更新时间：2021/06/02
8   */
9
10 #include "FashionStar_UartServoProtocol.h"
11 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
12
13
14 // 串口总线舵机配置参数
15 #define SERVO_ID 0 //舵机ID号
16 #define BAUDRATE 115200 // 波特率
17
18 // 调试串口的配置
19 #if defined(ARDUINO_AVR_UNO)
20     #include <SoftwareSerial.h>
21     #define SOFT_SERIAL_RX 6
22     #define SOFT_SERIAL_TX 7
23     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
24     #define DEBUG_SERIAL softSerial
25     #define DEBUG_SERIAL_BAUDRATE 4800
26 #elif defined(ARDUINO_AVR_MEGA2560)
27     #define DEBUG_SERIAL Serial
28     #define DEBUG_SERIAL_BAUDRATE 115200
29 #elif defined(ARDUINO_ARCH_ESP32)
30     #define DEBUG_SERIAL Serial
31     #define DEBUG_SERIAL_BAUDRATE 115200
32 #endif
33
34 FSUS_Protocol protocol(BAUDRATE); //协议
35 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
36
37 uint32_t interval; // 运行周期 单位ms
38 uint16_t t_acc; // 加速时间 单位ms
39 uint16_t t_dec; // 减速时间 单位ms
40 float velocity; // 目标转速 单位°/s
41
42 /* 等待并报告当前的角度*/
43 void waitAndReport(){
44     uservo.wait(); // 等待舵机旋转到目标角度
45     DEBUG_SERIAL.println("Real Angle = " + String(uservo.curRawAngle, 1) + "
46 Target Angle = "+String(uservo.targetRawAngle, 1));
47     delay(2000); // 暂停2s
48 }
49
50 void setup(){
51     protocol.init(); // 通信协议初始化
52     uservo.init(); //舵机角度初始化
53     // 打印例程信息
```

```

53     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE); // 初始化软串口的波特率
54     DEBUG_SERIAL.println("Set Servo Angle");
55 }
56
57 void loop(){
58     DEBUG_SERIAL.println("Set Angle = 900°");
59     uservo.setRawAngleMTurn(900.0); // 设置舵机的角度
60     waitAndReport();
61
62     DEBUG_SERIAL.println("Set Angle = -900.0°");
63     uservo.setRawAngleMTurn(-900.0);
64     waitAndReport();
65
66     DEBUG_SERIAL.println("Set Angle = 900° - Set Interval = 10s");
67     interval = 10000;
68     t_acc = 100;
69     t_dec = 100;
70     uservo.setRawAngleMTurnByInterval(900, interval, t_acc, t_dec, 0);
71     waitAndReport();
72
73     DEBUG_SERIAL.println("Set Angle = -900° - Set Velocity = 200°/s");
74     velocity = 200.0;
75     t_acc = 100;
76     t_dec = 100;
77     uservo.setRawAngleMTurnByVelocity(-900, velocity, t_acc, t_dec, 0);
78     waitAndReport();
79 }

```

输出日志

```

1  Set Angle = 900°
2  Set Servo Angle
3  Set Angle = 900°
4  Real Angle = 899.0 Target Angle = 900.0
5  Set Angle = -900.0°
6  Real Angle = -899.0 Target Angle = -900.0
7  Set Angle = 900° - Set Interval = 10s
8  Real Angle = 899.0 Target Angle = 900.0
9  Set Angle = -900° - Set Velocity = 200°/s
10 Real Angle = -899.0 Target Angle = -900.0

```

舵机扭力开关

API- setTorque

函数原型

```

1 void FSUS_Servo::setTorque(bool enable)

```

输入参数

- enable: 扭力是否开启

- `true`: 开启扭力
- `false`: 关闭扭力

使用示例

```
1 | uservo.setTorque(true); // 开启扭力
```

例程源码

servo_torque.ino

```
1  /*
2  * 测试舵机扭力开关
3  * 提示：拓展板上电之后，记得按下Arduino的RESET按键
4  * -----
5  * 作者：阿凯|Kyle
6  * 邮箱：kyle.xing@fashionstar.com.hk
7  * 更新时间：2021/06/02
8  */
9  #include "FashionStar_UartServoProtocol.h"
10 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
11
12 // 配置参数
13 #define BAUDRATE 115200 // 波特率
14 #define SERVO_ID 0 // 舵机ID号
15
16 FSUS_Protocol protocol; // 协议
17 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
18 void setup(){
19     protocol.init(); // 通信协议初始化
20     uservo.init(); // 舵机初始
21
22     uservo.setTorque(true); // 开启扭力
23     // uservo.setTorque(false); // 开启扭力
24 }
25
26 void loop(){
27
28 }
```

舵机标定

API-calibration

在 `FSUS_Servo` 类里面, 有两个跟标定相关的参数:


```

1  class FSUS_Servo{
2  public:
3      ...
4      float kAngleReal2Raw; // 舵机标定数据-舵机角度与位置之间的比例系数
5      float bAngleReal2Raw; // 舵机标定数据-舵机角度与位置转换过程中的偏移量
6      ...
7  }

```

舵机真实角度跟原始角度的映射关系如下:

$$angleRaw = kAngleReal2Raw \cdot angleReal + bAngleReal2Raw \quad (1)$$

函数原型

```

1  void FSUS_Servo::calibration(FSUS_SERVO_ANGLE_T rawA, FSUS_SERVO_ANGLE_T
    realA, FSUS_SERVO_ANGLE_T rawB, FSUS_SERVO_ANGLE_T realB)

```

输入参数

- `rawA` 在位置A时刻舵机原始的角度
- `realA` 在位置A时刻舵机真实的角度
- `rawB` 在位置B时刻舵机原始的角度
- `realB` 在位置B时刻舵机真实的角度

使用示例

```

1  // 设置舵机的标定点
2  // 样本1
3  #define SERVO_REAL_ANGLE_A 90 // 舵机真实角度
4  #define SERVO_RAW_ANGLE_A -86.2 // 舵机原始角度
5  // 样本2
6  #define SERVO_REAL_ANGLE_B -90 // 舵机真实角度
7  #define SERVO_RAW_ANGLE_B 91.9 // 舵机原始角度
8
9
10 // 输入舵机标定数据
11 uservo.calibration(
12     SERVO_RAW_ANGLE_A, SERVO_REAL_ANGLE_A, \
13     SERVO_RAW_ANGLE_B, SERVO_REAL_ANGLE_B);

```

函数原型

```

1  void FSUS_Servo::calibration(float kAngleReal2Raw, float bAngleReal2Raw);

```

输入参数

- `kAngleReal2Raw`: 舵机标定数据-舵机角度与位置之间的比例系数
- `bAngleReal2Raw`: 舵机标定数据-舵机角度与位置转换过程中的偏移量

API- angleReal2Raw

舵机真实角度转换为舵机原始角度

函数原型

```
1 // 真实角度转化为原始角度
2 FSUS_SERVO_ANGLE_T FSUS_Servo::angleReal2Raw(FSUS_SERVO_ANGLE_T realAngle);
```

输入参数

- `realAngle`: 舵机真实角度

返回参数

- `rawAngle`: 舵机原始角度

API- angleRaw2Real

舵机原始角度转化为真实角度

函数原型

```
1 // 原始角度转换为真实角度
2 FSUS_SERVO_ANGLE_T FSUS_Servo::angleRaw2Real(FSUS_SERVO_ANGLE_T rawAngle);
```

输入参数

- `rawAngle`: 舵机原始角度

返回参数

- `realAngle`: 舵机真实角度

例程源码

```
1  /*
2   * 测试舵机标定
3   * 提示：拓展板上电之后，记得按下Arduino的RESET按键
4   * -----
5   * 作者：阿凯|kyle
6   * 邮箱：kyle.xing@fashionstar.com.hk
7   * 更新时间：2021/06/02
8   */
9
10 #include "FashionStar_UartServoProtocol.h"
11 #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
12
13
14 // 串口总线舵机配置参数
15 #define SERVO_ID 0 //舵机ID号
16 #define BAUDRATE 115200 // 波特率
17
18 // 设置舵机的标定点
```

```

19 // 样本1
20 #define SERVO_REAL_ANGLE_A 90 // 舵机真实角度
21 #define SERVO_RAW_ANGLE_A -86.2 // 舵机原始角度
22 // 样本2
23 #define SERVO_REAL_ANGLE_B -90 // 舵机真实角度
24 #define SERVO_RAW_ANGLE_B 91.9 // 舵机原始角度
25
26 // 调试串口的配置
27 #if defined(ARDUINO_AVR_UNO)
28     #include <SoftwareSerial.h>
29     #define SOFT_SERIAL_RX 6
30     #define SOFT_SERIAL_TX 7
31     SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
32     #define DEBUG_SERIAL softSerial
33     #define DEBUG_SERIAL_BAUDRATE 4800
34 #elif defined(ARDUINO_AVR_MEGA2560)
35     #define DEBUG_SERIAL Serial
36     #define DEBUG_SERIAL_BAUDRATE 115200
37 #elif defined(ARDUINO_ARCH_ESP32)
38     #define DEBUG_SERIAL Serial
39     #define DEBUG_SERIAL_BAUDRATE 115200
40 #endif
41
42 FSUS_Protocol protocol(BAUDRATE); //协议
43 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
44
45 void setup(){
46     protocol.init(); // 通信协议初始化
47     uservo.init(); //舵机角度初始化
48     // 调试串口初始化
49     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE); // 初始化软串口的波特率
50     DEBUG_SERIAL.println("Set Servo Angle");
51     // 输入舵机标定数据
52     uservo.calibration(
53         SERVO_RAW_ANGLE_A, SERVO_REAL_ANGLE_A, \
54         SERVO_RAW_ANGLE_B, SERVO_REAL_ANGLE_B);
55
56     // 打印舵机标定数据
57     DEBUG_SERIAL.println("kAngleReal2Raw = "+String(uservo.kAngleReal2Raw,2)
+ \
58         "; bAngleReal2Raw = " + String(uservo.bAngleReal2Raw, 2));
59 }
60
61 void loop(){
62     DEBUG_SERIAL.println("Set Angle = 90°");
63     uservo.setAngle(90.0); // 设置舵机的角度
64     uservo.wait();
65     delay(2000);
66
67     DEBUG_SERIAL.println("Set Angle = -90°");
68     uservo.setAngle(-90);
69     uservo.wait();
70     delay(2000);
71 }

```

```
1 Set Servo Angle
2
3 kAngleReal2Raw = -0.99; bAngleReal2Raw = 2.85
4
5 Set Angle = 90
6
7 Set Angle = -90
```

舵机转速设置

API- setSpeed

函数原型

```
1 void FSUS_Servo::setSpeed(FSUS_SERVO_SPEED_T speed)
```

输入参数

- `speed` 舵机的平均转速, 单位°/s

返回参数

<无>

舵机数据读取

示例源码

servo_data_read.ino

```
1  /*
2   * 舵机数据读取实验
3   *  -----
4   *  作者：阿凯|Kyle
5   *  邮箱：kyle.xing@fashionstar.com.hk
6   *  更新时间：2021/06/02
7   */
8  #include "FashionStar_UartServoProtocol.h" // 串口总线舵机通信协议
9  #include "FashionStar_UartServo.h" // Fashion Star串口总线舵机的依赖
10
11  // 配置
12  #define SERVO_ID 4 //舵机ID号
13  #define BAUDRATE 115200 // 波特率
14
15  // 调试串口的配置
16  #if defined(ARDUINO_AVR_UNO)
17      #include <SoftwareSerial.h>
18      #define SOFT_SERIAL_RX 6
19      #define SOFT_SERIAL_TX 7
20      SoftwareSerial softSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // 创建软串口
```

```

21     #define DEBUG_SERIAL softSerial
22     #define DEBUG_SERIAL_BAUDRATE 4800
23 #elif defined(ARDUINO_AVR_MEGA2560)
24     #define DEBUG_SERIAL Serial
25     #define DEBUG_SERIAL_BAUDRATE 115200
26 #elif defined(ARDUINO_ARCH_ESP32)
27     #define DEBUG_SERIAL Serial
28     #define DEBUG_SERIAL_BAUDRATE 115200
29 #endif
30
31 FSUS_Protocol protocol(BAUDRATE); //协议
32 FSUS_Servo uservo(SERVO_ID, &protocol); // 创建舵机
33
34 // 读取数据
35 uint16_t voltage;           // 电压 mV
36 uint16_t current;          // 电流 mA
37 uint16_t power;            // 功率 mW
38 uint16_t temperature;      // 温度 °C
39
40 void setup(){
41
42     protocol.init(); // 舵机通信协议初始化
43     uservo.init(); // 串口总线舵机初始化
44     // 打印例程信息
45     DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUDRATE);
46     DEBUG_SERIAL.println("Start To Test Servo Data Read \n"); // 打印日志
47
48     uservo.setAngle(-25.0, 1000, 200); // 设置舵机角度(限制功率)
49 }
50
51 void loop(){
52     voltage = uservo.queryVoltage();
53     current = uservo.queryCurrent();
54     power = uservo.queryPower();
55     temperature = uservo.queryTemperature();
56     DEBUG_SERIAL.println("voltage: "+String((float)voltage, 1)+" mV\n");
57     delay(100);
58     DEBUG_SERIAL.println("current: "+String((float)current, 1)+" mA\n");
59     delay(100);
60     DEBUG_SERIAL.println("power: "+String((float)power, 1)+" mW\n");
61     delay(100);
62     DEBUG_SERIAL.println("temperature: "+String((float)temperature, 1)+"
Celsius\n");
63     delay(1000);
64 }

```

输出日志

注: 这个数值范围也不对啊, 要么就是测量的数值并不准确。

```

1
2 voltage: 9473.0 mV
3 current: 2.0 mA
4 power: 3.0 mW

```

```
5 temperature: 46340.0 Celsius
6
7 voltage: 9985.0 mV
8 current: 2.0 mA
9 power: 3.0 mW
10 temperature: 46340.0 Celsius
11
12 voltage: 9473.0 mV
13 current: 2.0 mA
14 power: 3.0 mW
15 temperature: 46340.0 Celsius
16
17 voltage: 9985.0 mV
18 current: 2.0 mA
19 power: 3.0 mW
20 temperature: 46340.0 Celsius
21
22 voltage: 9473.0 mV
23 current: 2.0 mA
24 power: 3.0 mW
25 temperature: 46084.0 Celsius
26
27 voltage: 9473.0 mV
28 current: 2.0 mA
29 power: 3.0 mW
30 temperature: 46084.0 Celsius
31
32 voltage: 9473.0 mV
33 current: 2.0 mA
34 power: 3.0 mW
```