myCobot

This is javascript API for mycobot.

MyCobot

Import to your project:

```
// basic demo
var mycobot = require("mycobot")

// obj Based on SerialPort
var obj = mycobot.connect("COM15",115200)

obj.write(mycobot.powerOn())

// Receive returned data
// obj.on("data",(data)=>{
//     res = mycobot.processReceived(data)
//     console.log("res:", res)
// })
```

Overall status

connect

- **Prototype**: `connect(port, baud)`
- **Description**:Create objects, connect devices (default open).

powerOn

- **Prototype**: `powerOn()`
- **Description**:Atom open communication (default open).

powerOff

- **Prototype**: `powerOff()`
- **Description**: Atom turn off communication.

isPowerOn

- **Prototype**: `isPowerOn()`
- **Description**: Adjust robot arm whether power on.
- **Returns**
  - `1`: power on
  - `0`: power off

releaseAllServos

- **Prototype**: `releaseAllServos()`
- **Description**: Set robot arm into free moving mode.

MDI mode and operation

getAngles

- **Prototype**: `getAngles()`
- **Description**: Get the degree of all joints.
- **Returns**: `object`: A float list of all degree.

sendAngle

- **Prototype**: `sendAngle(id, degree, speed)`
- **Description**: Send one degree of joint to robot arm.
- **Parameters**
  - `id`: Joint id
  - `degree`: degree value(`number`)
  - `speed`: (`number`) 0 ~ 100
- **Example**

```
obj.write(mycobot.sendAngle(1,50,50))
```

sendAngles()

- **Prototype**: `sendAngles(angles, speed)`
- **Description**: Send the degrees of all joints to robot arm.
- **Parameters**
  - `degrees`: a list of degree value(`Array`).
  - `speed`: (`number`) 0 ~ 100
- **Example**

```
obj.write(mycobot.sendAngles([0,0,0,0,0,0],60))
```

getCoords

- **Prototype**: `getCoords()`
- **Description**: Get the Coords from robot arm, coordinate system based on base.
- **Returns**: `object`: A float list of coord - `mycobot:[x, y, z, rx, ry, rz]; mypalletizer: [x, y, z, θ]`

sendCoord

- **Prototype**: `sendCoord(id, coord, speed)`
- **Description**: Send one coord to robot arm.
- **Parameters**
  - `id`: coord id
  - `coord`: coord value(`number`)
  - `speed`: (`number`) 0 ~ 100
- **Example**

```
obj.write(mycobot.sendCoord(x,20,50))
```

sendCoords

- **Prototype**: `sendCoords(coords, speed, mode)`
- **Description**: Send all coords to robot arm.
- **Parameters**
  - `coords`: a list of coords value(`Array`).
  - `speed`: (`number`) 0 ~ 100
  - `mode`: `0` - angular, `1` - linear
- **Example**

```
obj.write(mycobot.sendCoords([160, 160, 160, 0, 0, 0], 70, 0))
```

isInPosition

- **Prototype**: `isInPosition(data, flag)`
- **Description**: Judge whether in the position.
- **Parameters**
  - `data`: A data list, angles or coords.
  - `flag`: Tag the data type, `0` - angles, `1` - coords.
- **Returns**
  - `1` - true
  - `0` - false

JOG mode and operation

jogAngle

- **Prototype**: `jogAngle(jointId, direction, speed)`
- **Description**: Jog control angle
- **Parameters**
  - `jointId`: (`int`) 1 ~ 6
  - `direction`: `0` - decrease, `1` - increase
  - `speed`: 0 ~ 100

jogCoord

- **Prototype**: `jogCoord(coordId, direction, speed)`
- **Description**: Jog control coord.
- **Parameters**
  - `coordId`:(`int`) 1 ~ 6
  - `direction`: `0` - decrease, `1` - increase
  - `speed`: 0 ~ 100

jogStop

- **Prototype**: `jogStop()`
- **Description**: Stop jog moving.

programPause

- **Prototype**: `programPause()`
- **Description**: Pause movement.

programResume

- **Prototype**: `programResume()`
- **Description**: Recovery movement.

stop

- **Prototype**: `stop()`
- **Description**: Stop moving.

setEncoder

- **Prototype**: `setEncoder(jointId, encoder)`
- **Description**: Set a single joint rotation to the specified potential value.
- **Parameters**
  - `jointId`: 1 ~ 6
  - `encoder`: 0 ~ 4096

getEncoder

- **Prototype**: `getEncoder(jointId)`
- **Description**:Obtain the specified joint potential value.
- **Parameters**: `jointId`: 1 ~ 6
- **Returns**: `encoder`: 0 ~ 4096

setEncoders

- **Prototype**: `setEncoders(encoders, speed)`
- **Description**: Set the six joints of the manipulator to execute synchronously to the specified position.
- **Parameters**:

    - `encoders`: A encoder list, length 6.
    - `speed`: 0 - 100

getEncoders

- **Prototype**: `getEncoders()`
- **Description**: Get the all joints of the manipulator.
- **Returns**: the list of encoder (`Array`)

Running status and Settings

getSpeed

- **Prototype**: `getSpeed()`
- **Description**: Get speed.
- **Returns**: speed

setSpeed

- **Prototype**: `setSpeed(speed)`
- **Description**: Set speed.
- **Parameters**: speed: 0 ~ 100

getJointMin

- **Prototype**: `getJointMin(jointId)`
- **Description**: Gets the minimum movement angle of the specified joint
- **Parameters**: `jointId`
- **Returns**: angle value (`float`)

getJointMax

- **Prototype**: `getJointMax(jointId)`
- **Description**: Gets the maximum movement angle of the specified joint
- **Parameters**: `jointId`
- **Returns**: angle value (`float`)

Servo control

isServoEnable

- **Prototype**: `isServoEnable(servoId)`
- **Description**: Determine whether all steering gears are connected
- **Parameters**: `servoId` 1 ~ 6
- **Returns**
    - `0`: disable
    - `1`: enable

isAllServoEnable

- **Prototype**: `isAllServoEnable()`
- **Description**: Determine whether the specified steering gear is connected
- **Returns**
  - `0`: disable
  - `1`: enable

setServoData

- **Prototype**: `setServoData(servo_no, dataId, value)`
- **Description**: Set the data parameters of the specified address of the steering gear.
- **Parameters**:
  - `servo_no`: Serial number of articulated steering gear, 1 - 6.
  - `dataId`: Data address.
  - `value`: 0 - 4096

getServodata

- **Prototype**: `getServodata(servo_no, dataId)`
- **Description**: Read the data parameter of the specified address of the steering gear.
- **Parameters**:
  - `servo_no`: Serial number of articulated steering gear, 1 - 6.
  - `dataId`: Data address.
- **Returns**: `value`: 0 - 4096
  - `0`: disable
  - `1`: enable
  - `-1`: error

setServoCalibration

- **Prototype**: `setServoCalibration(servo_no)`
- **Description**: The current position of the calibration joint actuator is the angle zero point, and the corresponding potential value is 2048.
- **Parameters**:
  - `servo_no`: Serial number of articulated steering gear, 1 - 6.

releaseServo

- **Prototype**: `releaseServo(servoId)`
- **Description**: Power off designated servo
- **Parameters**: `servoId`: 1 ~ 6

focusServo

- **Prototype**: `focusServo(servoId)`
- **Description**: Power on designated servo
- **Parameters**: `servoId`: 1 ~ 6

Atom IO

setColor

- **Prototype**: `setColor(r, g, b)`
- **Description**: Set the color of the light on the top of the robot arm.
- **Parameters**
  - `r`: 0 ~ 255
  - `g`: 0 ~ 255
  - `b`: 0 ~ 255

setPinMode

- **Prototype**: `setPinMode(pin_no, pinMode)`
- **Description**: Set the state mode of the specified pin in atom.
- **Parameters**
  - `pin_no` : Pin number.
  - `pinMode` : 0 - input, 1 - output, 2 - inputPullup

setDigitalOutput()

- **Parameters**
  - `pin_no` :
  - `pinSignal` : 0 / 1

getDigitalInput()

- **Parameters**: `pin_no`
- **Return**: signal value

getGripperValue

- **Prototype**: `getGripperValue()`
- **Description**: Get gripper value
- **Return**: gripper value

setGripperState

- **Prototype**: `setGripperState(flag, speed)`
- **Description**: Set gripper switch state
- **Parameters**
  - `flag` : 0 - open, 1 - close

- `speed` : 0 ~ 100

setGripperValue

- **Prototype**: `setGripperValue(value, speed)`
- **Description**: Set gripper value
- **Parameters**
  - `value` : 0 ~ 100
  - `speed` : 0 ~ 100

setGripperIni

- **Prototype**: `setGripperIni()`
- **Description**: Set the current position to zero, set current position value is `2048`.

isGripperMving

- **Prototype**: `isGripperMving()`
- **Description**: Judge whether the gripper is moving or not
- **Returns**
  - `0` : not moving
  - `1` : is moving

M5Stack-basic

setBasicOutput

- **Prototype**: `setBasicOutput(pin_no, pinSignal)`
- **Description**: Set bottom pin.
- **Parameters**
  - `pin_no` : Pin number.
  - `pinSignal` : 0 / 1

getBasicOutput

- **Prototype**: `getBasicOutput(pin_no)`
- **Description**: get bottom pin.
- **Parameters**
  - `pin_no` : Pin number.