IO control

There are pins on the M5Stack-basic at the bottom of the robot arm and on the Atom at the end. The adsorption pump and other tools can be controlled by setting high and low levels of the pins through io control. For the numbers of input and output pins for each type of robot arm, refer to the following description table.
Table of Description of Input and Output Pins on the M5Stack-basic

Robot arm model	Input pin number	Output pin number
myCobot 280-M5	35、36	2、5、26
myCobot 320-M5	35、36	5、15

Table of Description of Input and Output Pins on the Atom

Robot arm model	Input pin number	Output pin number
myCobot 280-M5	19、22	23、33
myCobot 320-M5	No	NO

1 M5Stack-basic io control (m5)

1.1 Setting the high and low levels of output io

SetBasicOut(int pin_number, int pin_signal)
Return value: no
parameter description: Parameter 1: pin number (basic output pin number); Parameter 2: state (0 - low level; 1- high level)
Case: set the output pin 2 to high level

```
mycobot::MyCobot::I().SetBasicOut(2, 1);
```

1.2 Getting the state of input io

GetBasicIn(int pin_number)
Return value: pin state (0-low level; 1-high level)
Parameter description: pin number (basic input pin number)
Case:

```
mycobot::MyCobot::I().GetBasicIn(35);
```

2 Atom io Control

Note: 320m5 has no atom io, so the module API is not used.

2.1 Setting the high and low levels of output io

SetDigitalOut(int pin_number, int pin_signal)
Return value: no
parameter description: Parameter 1: pin number (atom output pin number); Parameter 2: state (0 - low level; 1- high level)
Case:

```
mycobot::MyCobot::I().SetDigitalOut(23, 1);
```

2.2 Getting the state of input io

GetDigitalIn(int pin_number)
Return value: pin state (0-low level; 1-high level)
Parameter description: pin number (atom input pin number)
Case:

```
mycobot::MyCobot::I().GetDigitalIn(19);
```

3 Complete use cases

```
int main(int argc, char* argv[])
{
    try {
        QCoreApplication a(argc, argv);
        using namespace std::chrono_literals;
        if (!mycobot::MyCobot::I().IsControllerConnected()) {
            std::cerr << "Robot is not connected\n";
            exit(EXIT_FAILURE);
    }
    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn();
    mycobot::MyCobot::I().SleepSecond(1);

    mycobot::MyCobot::I().SetBasicOut(2, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(5, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(26, 1);
    mycobot::MyCobot::I().SleepSecond(1);

    /*for (int i = 0; i < 2; i++) {
        std::cout << "35= " << mycobot::MyCobot::I().GetBasicIn(35) <<
std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "36= " << mycobot::MyCobot::I().GetBasicIn(36) <<
std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/

    /*mycobot::MyCobot::I().SetDigitalOut(23, 1);
    mycobot::MyCobot::I().SleepSecond(1);
```

```
    mycobot::MyCobot::I().SetDigitalOut(33, 1);
    mycobot::MyCobot::I().SleepSecond(1);*/

    /*for (int i = 0; i < 2; i++) {
        std::cout << "22= " << mycobot::MyCobot::I().GetDigitalIn(22) <<
std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "19= " << mycobot::MyCobot::I().GetDigitalIn(19) <<
std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/

    /*for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetGriper(1);
        mycobot::MyCobot::I().SleepSecond(3);
        mycobot::MyCobot::I().SetGriper(0);
        mycobot::MyCobot::I().SleepSecond(3);
    }*/

    /*for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetElectricGriper(1);
        mycobot::MyCobot::I().SleepSecond(1);
        mycobot::MyCobot::I().SetElectricGriper(0);
        mycobot::MyCobot::I().SleepSecond(1);
    }*/
    /*mycobot::MyCobot::I().StopRobot();
    std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() <<
"\n";
    mycobot::Angles angles = mycobot::MyCobot::I().GetAngles();
    std::this_thread::sleep_for(200ms);
    mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();
    angles = mycobot::MyCobot::I().GetAngles();
    std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ",
" << angles[mycobot::J3] << ", "
        << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " <<
angles[mycobot::J6] << "]";
    mycobot::Angles goal_angles = { 1, 0, 0, 0, 0, 0 };
    mycobot::MyCobot::I().WriteAngles(goal_angles,180);
    while (!mycobot::MyCobot::I().IsInPosition(goal_angles, false)) {
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] <<
", "
            << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
            << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" <<
std::flush;
        std::this_thread::sleep_for(200ms);
    }

    //mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
    std::this_thread::sleep_for(5000ms);
    mycobot::MyCobot::I().StopRobot();*/

    std::cout << "\n";
    exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
```

```
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
    } catch (...) {
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
    }
}
```