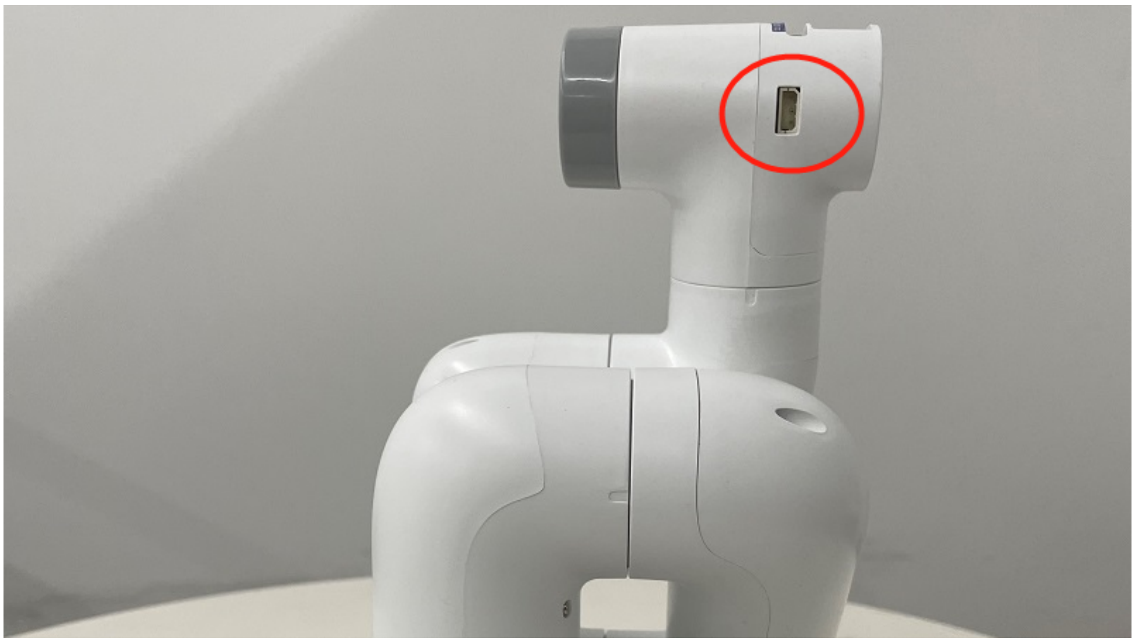
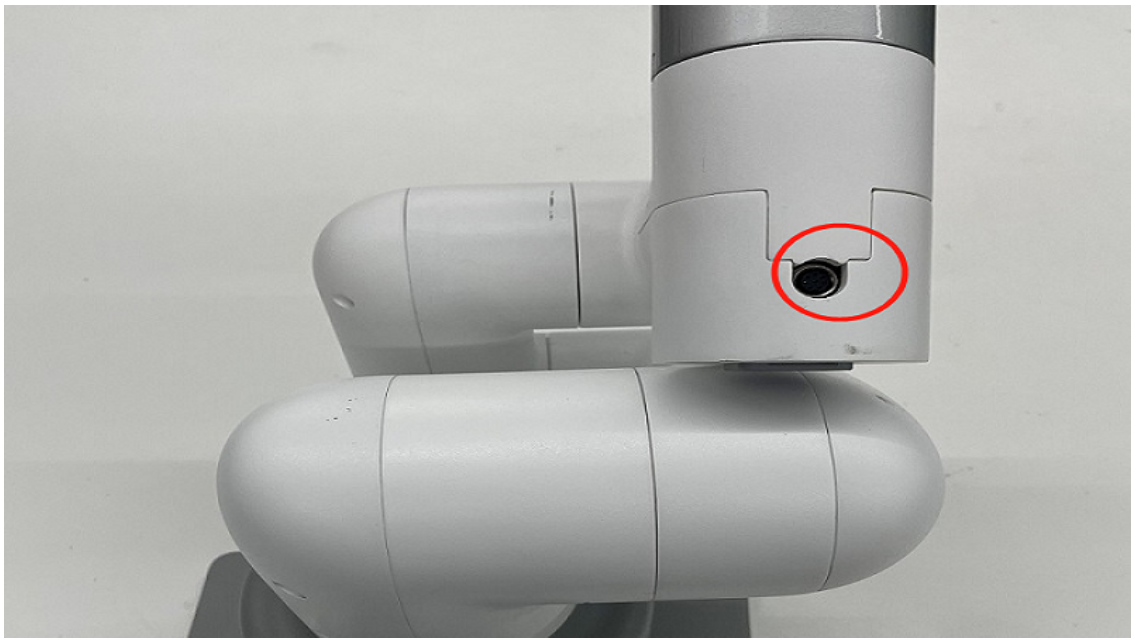
Gripper control

Installing the gripper:

- For an adaptive gripper, insert it on the pin on the atom, as shown in the following figure:

- For an electric gripper, insert it into the 485 interface on the top, as shown in the following figure:
Note: myCobot 280-m5 has no electric gripper, only myCobot 320-m5 has an electric gripper


1 Adaptive gripper control

supports: myCobot280, 320 && myPalletizer 260
SetGriper(int open)
Return value: no
parameter description: gripper switch state (0-off; 1-on)

Case: Due to the delay, when the gripper is controlled for the first time, the gripping may not be successful. Therefore it is recommended to send twice

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

2 Electric gripper control

Available for: myCobot320
SetElectricGriper(int open)
Return value: no
parameter description: gripper switch state (0-off; 1-on)
Case: Due to the delay, when the gripper is controlled for the first time, the gripping may not be successful. Therefore it is recommended to send twice

```
for (int i = 0; i < 2; i++) {<br>
    mycobot::MyCobot::I().SetElectricGriper(1);<br>
    mycobot::MyCobot::I().SleepSecond(1);<br>
    mycobot::MyCobot::I().SetElectricGriper(0);<br>
    mycobot::MyCobot::I().SleepSecond(1);<br>
}
```

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);
    }
}
```