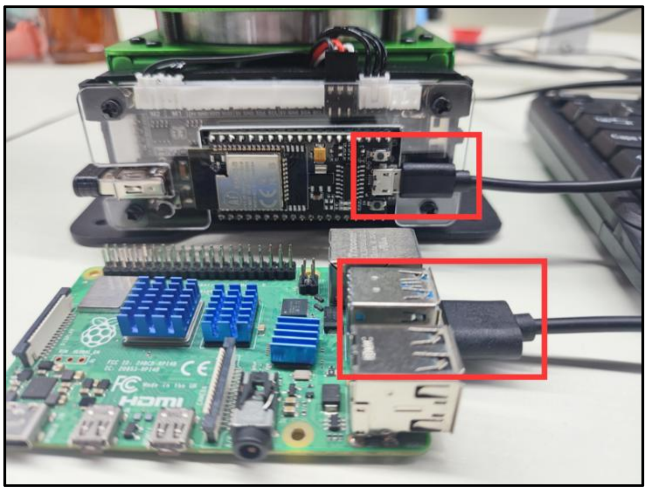
Raspberry Pi Communication Routine Analysis

1. Communication Connection

1.1 Hardware Wiring

Please ensure the MaxArm robotic arm connects to the USB port of Raspberry Pi.



1.2 Communication Protocol

The communication protocol for both the master and slave devices in the MaxArm communication routines follows the following format:

Frame header	**Function code**	**Data length**	**Data information**	**Check bit**
0xAA 0x55	func	len	data	check

The annotations of each part of the protocol are as follows:

Frame header: if 0xAA and 0x55 are received sequentially, it indicates that there is data to be received, consisting of a fixed 2 bytes.

Function Code: Used to indicate the purpose of an information frame, consists of 1 byte.

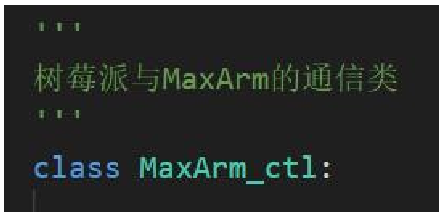
Data Length: Indicates the number of data bits carried by the data frame.

Check Bit: Verifies the correctness of the data frame. If correct, the corresponding function is called; otherwise, the data frame is skipped.

The calculation method for the check bit is: calculate the sum of the function code, data length, and data, then take the complement, and finally, take the low byte, which serves as the checksum.

2. Program Interface Parsing

The routines analyzed in this document are located in folder of “Raspberry Pi Routines”, utilizing examples of communication via microUSB interface. This section mainly analyzes the “**MaxArm_ctl.py**” file, which is program file for the Raspberry Pi’s communication underlying implementation and has been encapsulated into Raspberry Pi communication class MaxArm_ctl:

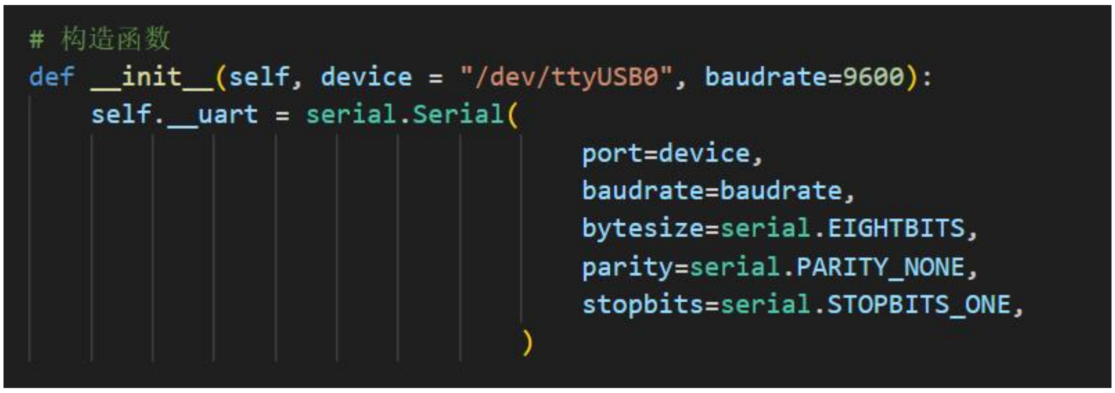


```
'''
树莓派与MaxArm的通信类
'''
class MaxArm_ctl:
```

The usage and parsing of the member functions is as follow:

2.1 Constructor

The constructor of MaxArm_ctl class can takes two parameters. Parameter 1 is the port number for the connection between Raspberry Pi and MaxArm, and parameter 2 is the communication baud rate, which is set to 9600.



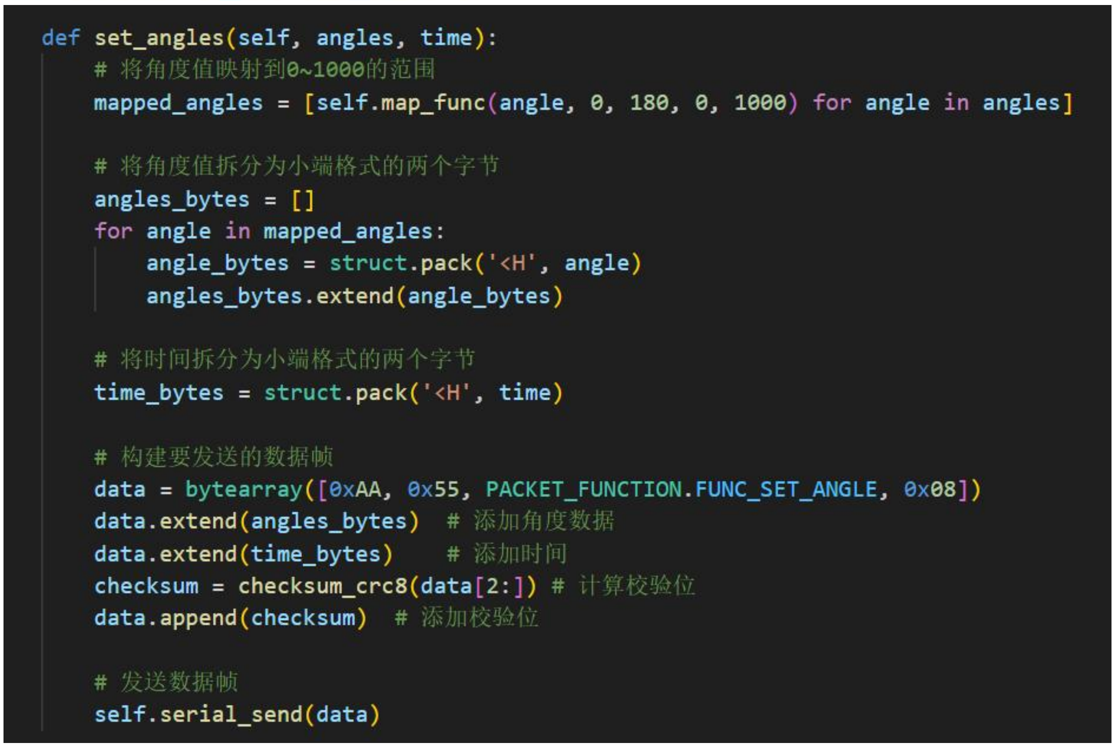
```
# 构造函数
def __init__(self, device = "/dev/ttyUSB0", baudrate=9600):
    self.__uart = serial.Serial(
                                port=device,
                                baudrate=baudrate,
                                bytesize=serial.EIGHTBITS,
                                parity=serial.PARITY_NONE,
                                stopbits=serial.STOPBITS_ONE,
                            )
```

You can enter the command "ls /dev/ttyUSB*" in terminal of Raspberry Pi to view the port number for the connection between Raspberry Pi and MaxArm. When initializing this communication class, it is necessary to first select the appropriate port number for communication, otherwise, the communication fails.

2.2 Function to Set the Angles of Three Bus Servos

The **set_angles()** function is used to set the angles of three bus servos on MaxArm, requiring two parameters. Parameter one is a collections of angles for three servos, and parameter 2 is the servo runtime.

This function decomposes the passed three angle values and runtime into control data of 8 bytes. It construct a complete control frame by adding a frame header, function code 0x01, data length 0x08, control data, and the calculated checksum to a byte array. Consequently, this frame data is sent to MaxArm via a serial port transmission function, enabling control of MaxArm movement.

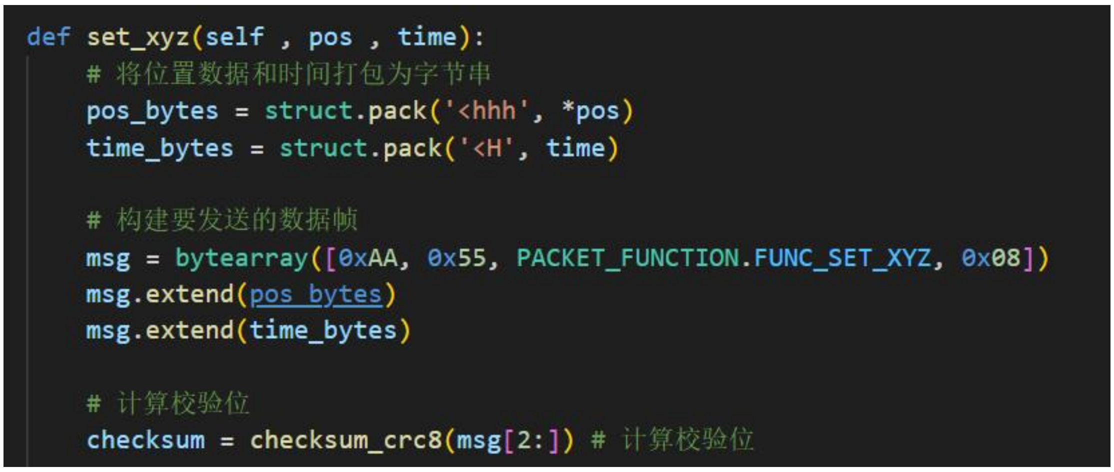


```python
def set_angles(self, angles, time):
    # 将角度值映射到0~1000的范围
    mapped_angles = [self.map_func(angle, 0, 180, 0, 1000) for angle in angles]

    # 将角度值拆分为小端格式的两个字节
    angles_bytes = []
    for angle in mapped_angles:
        angle_bytes = struct.pack('<H', angle)
        angles_bytes.extend(angle_bytes)

    # 将时间拆分为小端格式的两个字节
    time_bytes = struct.pack('<H', time)

    # 构建要发送的数据帧
    data = bytearray([0xAA, 0x55, PACKET_FUNCTION.FUNC_SET_ANGLE, 0x08])
    data.extend(angles_bytes)  # 添加角度数据
    data.extend(time_bytes)    # 添加时间
    checksum = checksum_crc8(data[2:]) # 计算校验位
    data.append(checksum)  # 添加校验位

    # 发送数据帧
    self.serial_send(data)
```

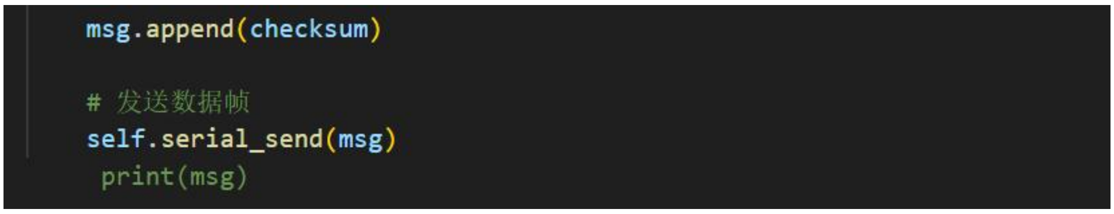
2.3 Function to Set End-effector’ XYZ Coordinates

The **set_xyz()** function sets the movement of MaxArm to specific XYZ coordinates, requiring two parameters. Parameter 1 is a collection of XYZ coordinates, and Parameter 2 is the duration of servo motion.

The function decomposes the passed XYZ coordinates and motion time into control data of 8 bytes. It constructs a complete control frame by adding a frame header, function code 0x03, data length 0x08, control data, and the calculated checksum to a byte array. This frame data is then sent to MaxArm via a serial port transmission function, enabling control of MaxArm movement to the corresponding XYZ axis coordinates.



```python
def set_xyz(self , pos , time):
    # 将位置数据和时间打包为字节串
    pos_bytes = struct.pack('<hhh', *pos)
    time_bytes = struct.pack('<H', time)

    # 构建要发送的数据帧
    msg = bytearray([0xAA, 0x55, PACKET_FUNCTION.FUNC_SET_XYZ, 0x08])
    msg.extend(pos_bytes)
    msg.extend(time_bytes)

    # 计算校验位
    checksum = checksum_crc8(msg[2:]) # 计算校验位
```



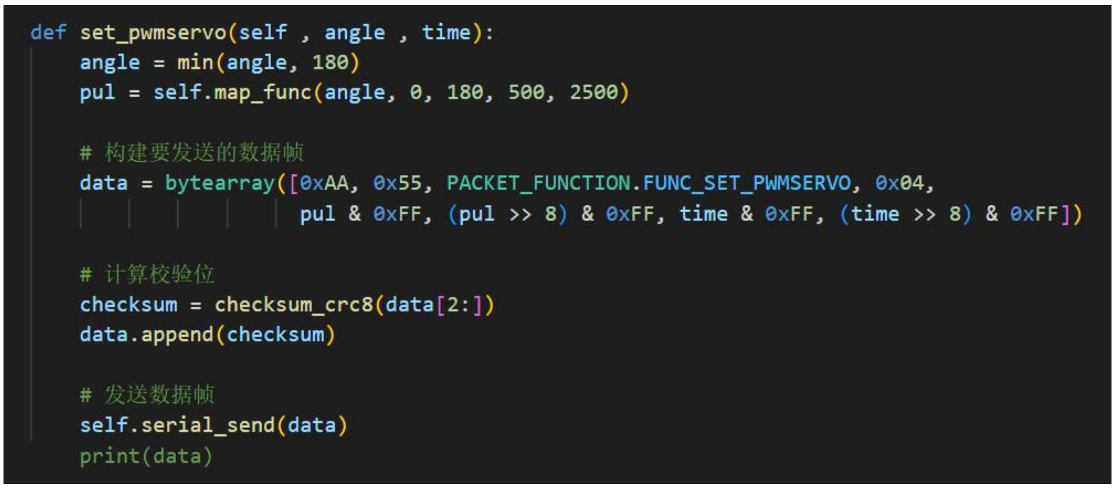
```python
    msg.append(checksum)

    # 发送数据帧
    self.serial_send(msg)
     print(msg)
```

2.4 Function to Set the Position of PWM Servo at End-effector

The **set_pwmservo()**function sets the movement of end-effector's PWM servo to the specific angle. It requires two parameters. Parameter 1 is represents the servo angle value, and Parameter 2 denotes the duration of servo motion.

The function transforms the passed servo angle value into the pulse width of the servo, and decomposes the duration of servo motion into control data of 4 bytes.

It constructs a complete control frame by adding a frame header, function code 0x05, data length 0x04, control data, and the calculated checksum to a byte array. This frame data is then sent to MaxArm via a serial port transmission function, enabling control of MaxArm to rotate to the corresponding angle.



```python
def set_pwmservo(self , angle , time):
    angle = min(angle, 180)
    pul = self.map_func(angle, 0, 180, 500, 2500)

    # 构建要发送的数据帧
    data = bytearray([0xAA, 0x55, PACKET_FUNCTION.FUNC_SET_PWMSERVO, 0x04,
                      pul & 0xFF, (pul >> 8) & 0xFF, time & 0xFF, (time >> 8) & 0xFF])

    # 计算校验位
    checksum = checksum_crc8(data[2:])
    data.append(checksum)

    # 发送数据帧
    self.serial_send(data)
    print(data)
```

2.5 Function to Control the Working Status of the Suction Cup at End-effector

The **set_SuctioNnozzle()** function is used to control the working status of the end-effector’s suction cup. It requires one parameter only, which represents the sub-function code value of the working status of the suction cup. The value involves 1, 2 and 3 corresponding to opening the air pump, closing the air pump and opening the air valve, and closing the air value, respectively.

The sequence of operation the air pump is as follow: when the pump is opened, the suction cup will generate suction force. Then, when the pump is closed and the valve is opened, negative pressure still exists inside the suction cup after the pump is turned off. Opening the valve allows external air to enter, eliminating the negative pressure and completely eliminating the suction force. Finally, the valve is closed.

The function checks whether the passed sub-function code is correct. If correct, it constructs a complete control frame by adding a frame header, function code 0x07, data length 0x01, sub-function code, and the calculated checksum to a byte array. This frame data is then sent to MaxArm via a serial port transmission function, enabling control of the working status of the end effector's suction nozzle on MaxArm.

```python
def set_SuctioNnozzle(self , func):
    if func not in [1,2,3]:
        return
    # 构建要发送的数据帧
    data = bytearray([0xAA, 0x55, PACKET_FUNCTION.FUNC_SET_SUCTIONNOZZLE,
                      0x01, func & 0xFF])

    crc = checksum_crc8(data[2:])
    data.append(crc)
    self.serial_send(data)
     print(data)
```

2.6 Function to Read Bus Servo Angle

The **read_angles()** function reads the angles of the bus servos on MaxArm. This function returns a collection of angles containing three elements, each representing the angle of one bus servo.

The program flow of this function is as follows: first, it sends a command to read the angle values. After waiting for 0.1 seconds, it receives the data frame sent through the serial port. Then, it extracts and parses the angle values from the received data frame, and finally returns the results.

```python
def read_angles(self):
    # 构建要发送的读取角度的命令帧
    command = bytearray([0xAA, 0x55, 0x11, 0x00 , 0xEE])
    # 发送读取角度的命令帧
    self.serial_send(command)

    time.sleep(0.1)

    # 从串口接收数据
    response = self.__uart.read(12)

     print(response)

    # 调用解析函数处理接收到的数据
    rec = self.rec_handle(response , 0x11)

    if rec:
        angles = struct.unpack('<hhh', rec)
    else:
        return
    return angles
```

2.7 Function to Read XYZ Coordinates of End-effector

The **read_xyz()** function reads the XYZ coordinates of the end effector on MaxArm. This function returns a collection with three elements representing the values of x, y, and z.

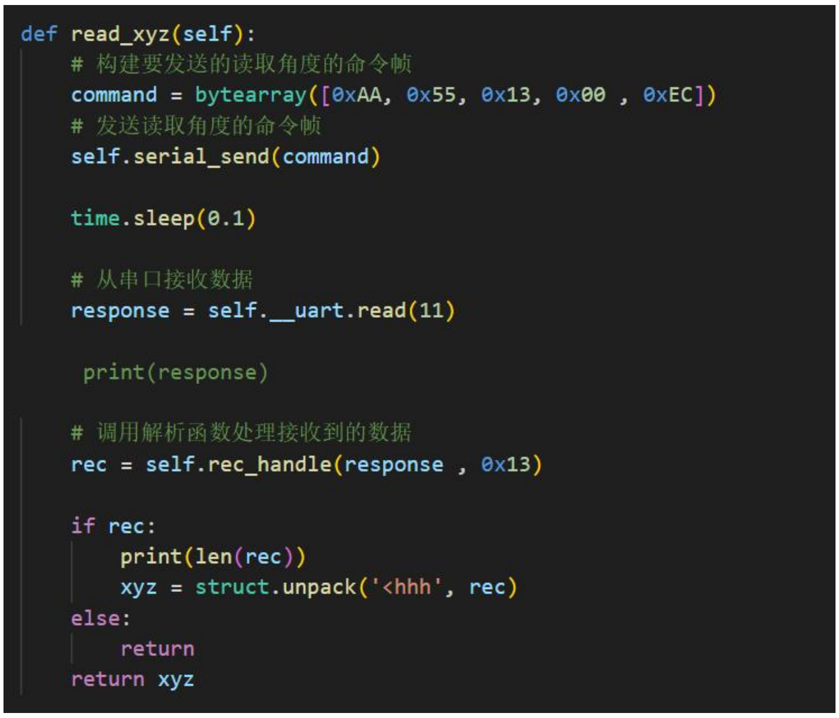
The program of this function is as follow:

1) Send a command to request the XYZ coordinates from MaxArm.

2) Wait for 0.1 seconds, and then receive the data frame sent via the serial port.

3) Extract and parse the XYZ coordinates from the received data frame.

4) Return the XYZ coordinates as a collection of three elements.



```
def read_xyz(self):
    # 构建要发送的读取角度的命令帧
    command = bytearray([0xAA, 0x55, 0x13, 0x00 , 0xEC])
    # 发送读取角度的命令帧
    self.serial_send(command)

    time.sleep(0.1)

    # 从串口接收数据
    response = self.__uart.read(11)

     print(response)

    # 调用解析函数处理接收到的数据
    rec = self.rec_handle(response , 0x13)

    if rec:
        print(len(rec))
        xyz = struct.unpack('<hhh', rec)
    else:
        return
    return xyz
```