SoRo4

At mission control, the mission control computer and the antenna are connected to a router. Onboard the rover, the antenna, all Arduinos, the GPS, and the rover computer (nvidia jetson) are connected to another router. The antennas act as a network bridge between the two routers. The only communication is over Ethernet or the antennas via the UDP protocol.

**Rover network**

Gateway: 10.0.0.1

Rover Computer: 10.0.0.100, port 1000

Drive subsystem: 10.0.0.101, port 1001

Arm subsystem: 10.0.0.102, port 1002

Drill subsystem: 10.0.0.103, port 1003

Autonomous subsystem: 10.0.0.104, port 1004

GPS module: 10.0.0.223, port 55555

**Message to control a subsystem:**

The Arduinos that control the wheels, arm, and science package will receive these. They can be created by the MC computer (in the case of human operation) or by the rover computer (autonomous operation), but the format will be the same for both.

Format: **[start transmission: control][device id][data (multiple bytes)][hash]**

* Start transmission
  + Value is -127 *(signed 8 bit integer – signed char or int\_8t)*
  + Lets the Arduino know this is the beginning of a new message
* Device id
  + unsigned 8 bit integer – unsigned char or uint\_8t
  + Unique ID of the Arduino the message is intended for
  + Arduinos will ignore the message if the ID does not match
  + Wheels/Drive = 0
  + Arm = 1
  + Science = 2
* Data
  + Multiple bytes
  + Different length and format for each subsystem (see below)
* Hash
  + signed 8 bit integer – signed char or int\_8t
  + The average value of all the data bytes
  + Is used to verify the data. The Arduino calculates the average then compares it to this byte to make sure the data is not corrupted. The Arduino will ignore the message if they don’t match.

**Data bytes formats:** There is a different data format for each subsystem, each being a different amount of bytes.

* Drive/Wheels (also controls the movable camera):
  + **[ [Modifiers][left wheels][right wheels][gimble tilt][gimble pan] ]**
  + Modifiers = each bit is a different boolean *(unsigned char/uint8\_t)*
    - 20 bit: center axle break on(1)/off(0)
    - 21 bit: front wheels modifier. If 1, then only the front wheels will spin
    - 22 bit: rear wheels modifier. If 1, then only the rear wheels will spin
      * If the front and rear wheel modifiers are both 1, then the front and rear wheels will spin in opposite directions (for unfolding the rover)
    - 23 bit: reset camera gimbal. If 1, then the arduino will set the angles of the gimbal servos to default positions. (Gimbal will have to be rebuilt without continuous rotation servos)
  + Left wheels = [-90, 90] speed of the left wheels *(signed char/ int8\_t)*
  + Right wheels = [-90, 90] speed of the right wheels *(signed char/ int8\_t)*
  + Gimble tilt = [-5, 5] speed at which to tilt the camera *(signed char/ int8\_t)*
  + Gimple pan = [-5, 5] speed at which to pan the camera *(signed char/ int8\_t)*
  + **Entire message example:**
    - [-127][0][0][90][70][0][0][36]
    - Drives forward with the right wheels spinning faster than the left. The camera is not moving
      * You would put this data in a QbyteArray and send it to the drive Arduino with the comms.h sendMessage() function
* Arm
  + **[ [base pos][shoulder pos][elbow pos][wrist pitch][buttons] ]**
  + Base position = [0,255] Comes from a potentiometer, 0 corresponds to 0 degrees, 255 corresponds to 270 degrees *(unsigned char/uint8\_t)*
  + Shoulder position = [0,255] Comes from a potentiometer, 0 corresponds to 0 degrees, 255 corresponds to 270 degrees *(unsigned char/uint8\_t)*
  + Elbow position = [0,255] Comes from a potentiometer, 0 corresponds to 0 degrees, 255 corresponds to 270 degrees *(unsigned char/uint8\_t)*
  + Wrist pitch = [0,255] Comes from a potentiometer, 0 corresponds to 0 degrees, 255 corresponds to 270 degrees *(unsigned char/uint8\_t)*
  + buttons = each bit is a different boolean (the 4 buttons are repeated for fun)
    - 20 bit & 24 bit: button 1
    - 21 bit & 25 bit: button 2
    - 22 bit & 26 bit: button 3
    - 23 bit & 27 bit: button 4