**Communication Standard**

The communications is done on the low lever between the python and Arduino sub modules using serial communications. Each device will be given an ID used to send messages between the modules upon request from the master. The serial communication will be interrupt driven with its replies only, the commands will be stored in the devices serial buffer. Once the device has received a command, it will either acknowledge the packet or send a busy response if it is not ready for the packet.

**Packet Composition:**

**Sent:**

Packet sent from the python script to each Arduino. It consists to 4 main elements, the sender ID, the receiver ID, packet size and then a single command. The packet size is used to determine the length of the command.

The Command – The command is the data sent to the Arduino, this tells the device what operations needs to be done and can consist of multiple operators depending on the command sent. Only one command can be executed at once.

Packet size – this is the size of the packet sent to the device, it is not the number of bytes but the number of operators of the command

Receiver ID – This is the devices ID assigned to each Arduino to ensure that the packet is meant for them

Terminating Character

Start Character

[sID1000 rID1001 PK3 P1 m52.37 D1]

Senders ID – Configured on the Arduino and used to check against here, python ID

Each part of the command is delineated with a space character to make them easier to read as well as easier to decode on the Arduino hardware.

**Received**

The received packets are of similar format to that of the sent packet however with a few more additional operators such as **ACK, BUSY** and **ERR\_.** All of these packets are sent as a reply to a request from the main python script and are only sent as such, there will be no constant writing to the serial lines as this will fill the python buffer. When the Arduino is busy with other operations it will only ever respond with a **ACK** or **BUSY** which will be driven by a timer interrupt 40ms after the receiving of a packet. If the Arduino is **BUSY** it will ignore the latest packet sent and simply reply with a **BUSY**.

The only time when the Arduino is proactively send a message to the python is when the device encounters an error, then the device will send an error message to the python for it to handle and deal with or when the system is free to receive the next command.

Packet size – this is the size of the packet sent to the device, in this case 6 as there are 3 sensors with 3 states.

Senders ID – Same as the Arduinos Device ID to tell the python where the packet is from

Receiver ID – This is the ID of the python serial module, this ensures that the correct device receives the packet

The Command – This is the command for reading all the states of the sensors, it is a list of all the sensors in first instance, it used a letter and number to state which sensor it is which is followed by a state command which denotes the value of the sensor.

[sID1001 rID1000 PK6 T1 S57.3 B1 S1 B2 S0]

Other Packets:

* [sID1001 rID1000 PK1 ACK]
* [sID1001 rID1000 PK1 BUSY]
* [sID1001 rID1000 PK1 ERR0]
* [sID1001 rID1000 PK1 FREE]

|  |  |
| --- | --- |
| Error Number | Fault |
| ERR0 | Incorrect packet format |
| ERR1 | Packet missing items |
| ERR2 | Incorrect Device ID |
| ERR3 | Incorrect Sender ID |
| ERR4 | System is in error state. |

**Command Operators:**

|  |  |
| --- | --- |
| Operator | info |
| sID\_\_\_\_ | Sender ID, this has to match on both devices, has to be unique + ID number |
| rID\_\_\_\_ | Receiver ID, this has to match on both devices, has to be unique + ID number |
| PK\_ | Packet size of the number of command operators, not length |
| ACK | Acknowledgement response |
| BUSY | Busy response when the device is in operation |
| FREE | Send a message to the python handler that it has finished all processes |
| ERR\_ | Error message based on the devices state |
| R | Read sensor operator, sent by the python to indicate the reading of all the sensors, all sent back as a packet as shown above. |
| P\_ | Pump operator followed by the number of the pump |
| m\_\_.\_\_ | number of millilitres to pump, always follows pump command, ml in integer or floating point notation. |
| D\_ | Direction of the pump, 0 pumping away from reactor, 1 pumping into reactor or used for the mixer as 0 clockwise and 1 anticlockwise |
| S\_\_\_\_\_ | State operator, This follows after all the following operators to denote the value that they are or the state in which they will take. 0 or 1 for on and off and either a floating point or integer value for all other devices and sensors. |
| V\_ | Valve operator, followed by the number of the valve and then state. |
| I\_\_\_ | Shutter operator, followed by the number of the shutter and then state. |
| T\_ | Temperature operator, followed by the number of the sensor and then state which is followed by the temperature reading. |
| B\_ | Bubble sensor operator, followed by the number of the sensor and then state which is followed by the value of the sensor. |
| M\_ | Mixer operator, followed by the number of the mixer, State to specify the speed of the mixer and Direction of the mixer |
| L\_ | Liquid level sensor operator, followed by the number of the sensor and then state which is followed by the value of the sensor. |