Overview of the Protocol:

Every serial message to the arduino will be a stream of bytes, with a max length of 8 bytes, having the following form:
{function byte, parameter 1, … parameter n, terminator}.

The function byte defines the function to be called on the arduino.
The param bytes are the parameters to the function.
The terminator tells the arduino that the message is over.

The Arduino parses the message, executes the procedure (if possible), and sends a return message of the form: {output, terminator}

If the Arduino encounters an error in executing the procedure, it will return 253 as the output.

For Arduino functions that return an output, such as digitalRead or analogRead, the output will correspond to the output of the function.

For all other functions, the output will be 0, meaning that the function executed properly.

Example 1: You want to set up digital pin 1 on the Arduino as digital output and have it output HIGH.

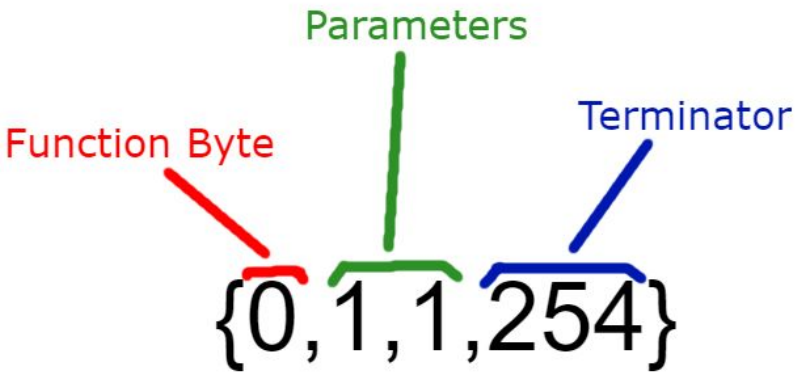
In arduino C++ code that would be done as follows:

```
pinMode(1,OUTPUT);
digitalWrite(1,HIGH);
```

Via the protocol this would be accomplished with the following messages:

{0,1,1,254}
{1,1,1,254}

How does this work? Each message can be divided into three main sections: The function, parameters, and the terminator:



The arduino parses the message {0,1,1,254} as follows:

Section	Value	Meaning
Function Byte	0	pinMode
Parameters	1,1	D1,OUTPUT
Terminator	254	End Message

{0,1,1,254} is interpreted as: `pinMode(1,OUTPUT);`

Special Values:

255 - Null

The max value of a byte, 255, is returned by Arduino's Serial.read() function to signify a lack of available serial input. This means any values of 255 will be ignored.

254 - The Terminator

The terminator was chosen as 254. This value signifies the end of a message.

The Function Byte:

Byte Value	Server side function
0	pinMode(pinNumber, mode)
1	digitalwrite(pinNumber, value)

2	analogwrite(pinNumber,value)
3	digitalRead(pinNumber)
4	analogRead(pinNumber)
5	servo(pinNumber)
6	writeServo(pinNumber,value)
7	encoder(interruptPin,secondaryPin)
8	getEncoderCount(interruptPin)
9	resetEncoder(interruptPin)
253	checkConnection()

Function Usage and Parameters:

Pin numbers:
Any function requiring a pinNumber parameter, interprets the byte as follows:

Byte Value	Pin
0	D0
1	D1
2	D2
3	D3
4	D4
5	D5
6	D6
7	D7
8	D8
9	D9
10	D10
11	D11

12	D12
13	D13
14	A0
15	A1
16	A2
17	A3
18	A4
19	A5

pinMode - Function Byte 0

pinMode(pinNumber, pinType)

Sets up [pinNumber] as type [pinType].

pinType

Byte Value	pinType
0	INPUT
1	OUTPUT
2	INPUT_PULLUP

digitalWrite - Function Byte 1

digitalWrite(pinNumber,value)

Writes [value] to [pinNumber]

value

Byte Value	value
0	LOW
1	HIGH

analogWrite - Function Byte 2

analogWrite(pinNumber,value)

Writes [value] to [pinNumber]

[value] corresponds to a duty cycle between 0 (always off) and 252 (always on).

digitalRead - Function Byte 3

digitalRead(pinNumber)

Returns the state of [pinNumber], as 0 or 1 (LOW or HIGH).

analogRead - Function Byte 4

analogRead(pinNumber)

Returns the state of [pinNumber], as a value between 0 (0v) and 252 (5v or Aref)

servo - Function Byte 5

servo(pinNumber)

Sets up a servo at [pinNumber]

writeServo - Function Byte 6

writeServo(pinNumber,value)

Writes duty cycle, [value] between 0 (minimum duty cycle) and 252 (maximum duty cycle), to servo at [pinNumber].

encoder - Function Byte 7

encoder(interruptPin,secondaryPin)

Sets up an encoder with channel A at interruptPin (must be D2 or D3), and channel B at secondaryPin (any other pin).

getEncoderCount - Function Byte 8

getEncoderCount(interruptPin)

Returns the count of the encoder at [interruptPin]

resetEncoder - Function Byte 9

resetEncoder(interruptPin)

Sets the count of encoder at [interruptPin] to zero.

checkConnection - Function Byte 253

checkConnection()

Returns 0 if the Arduino is connected and working properly.