

I have taken pwmWrite block for reference and the message is "ANALOG_MESSAGE"

Steps:-

1. Block definition in objects.js (Refer line 81

<https://github.com/bromagosa/Snap4Arduino/blob/master/src/s4a/objects.js>)

```
this.blocks.pwmWrite =  
{  
  only: SpriteMorph,  
  type: 'command',  
  category: 'arduino',  
  spec: 'set pin %pwmPin to value %n',  
  defaults: [null, 128],  
  transpilable: true  
};
```

2. Function definition in thread.js – analogWrite getting called (Refer line 204

<https://github.com/bromagosa/Snap4Arduino/blob/master/src/s4a/threads.js>)

```
Process.prototype.pwmWrite = function (pin, value) {  
  var sprite = this.homeContext.receiver;  
  
  if (sprite.arduino.isBoardReady()) {  
    var board = sprite.arduino.board;  
  
    if (board.pins[pin].mode != board.MODES.PWM) {  
      board.pinMode(pin, board.MODES.PWM);  
    }  
  
    board.analogWrite(pin, value);  
    return null;  
  } else {  
    throw new Error(localize('Arduino not connected'));  
  }  
};
```

3. analogWrite is defined in firmata.js and ANALOG_MESSAGE is passed (Refer line 403 <https://github.com/bromagosa/Snap4Arduino/blob/master/src/platforms/mobile/android/root/mobile/firmata.js>)

```
Board.prototype.analogWrite = function(pin, value) {  
  this.pins[pin].value = value;  
  this.sp.write([ANALOG_MESSAGE | pin, value & 0x7F, (value >> 7) & 0x7F]);  
};
```

4. address for message is given (Refer same as step 3)

```
var PIN_MODE = 0xF4,  
    REPORT_DIGITAL = 0xD0,  
    REPORT_ANALOG = 0xC0,  
    DIGITAL_MESSAGE = 0x90,  
    START_SYSEX = 0xF0,  
    END_SYSEX = 0xF7,  
    QUERY_FIRMWARE = 0x79,  
    REPORT_VERSION = 0xF9,  
    ANALOG_MESSAGE = 0xE0,  
    CAPABILITY_QUERY = 0x6B,  
    CAPABILITY_RESPONSE = 0x6C,  
    SERVO_CONFIG = 0x70,  
    PIN_STATE_QUERY = 0x6D,
```

5. in firmata.h , same address and message is defined which will send data (Refer firmata.h)

```
// message command bytes (128-255/0x80-0xFF)  
#define DIGITAL_MESSAGE 0x90 // send data for a digital port (collection of 8 pins)  
#define ANALOG_MESSAGE 0xE0 // send data for an analog pin (or PWM)  
#define REPORT_ANALOG 0xC0 // enable analog input by pin #  
#define REPORT_DIGITAL 0xD0 // enable digital input by port pair
```

6. in firmata.cpp ANALOG_MESSAGE does two things - 1. it waits for the data needed

```
switch (command) {  
  case ANALOG_MESSAGE:  
  case DIGITAL_MESSAGE:  
  case SET_PIN_MODE:  
  case SET_DIGITAL_PIN_VALUE:  
    waitForData = 2; // two data bytes needed  
    executeMultiByteCommand = command;  
    break;
```

7. second - It attaches the callback function after writing the value (Refer firmata.cpp)

```
* Attach a generic sysex callback function to a command (options are: ANALOG_MESSAGE,
* DIGITAL_MESSAGE, REPORT_ANALOG, REPORT_DIGITAL, SET_PIN_MODE and SET_DIGITAL_PIN_VALUE).
* @param command The ID of the command to attach a callback function to.
* @param newFunction A reference to the callback function to attach.
*/
void FirmataClass::attach(byte command, callbackFunction newFunction)
{
    switch (command) {
        case ANALOG_MESSAGE: currentAnalogCallback = newFunction; break;
```

8. callback is defined here (Refer firmata.cpp)

```
if ( (waitForData == 0) && executeMultiByteCommand ) { // got the whole message
    switch (executeMultiByteCommand) {
        case ANALOG_MESSAGE:
            if (currentAnalogCallback) {
                (*currentAnalogCallback)(multiByteChannel,
                    (storedInputData[0] << 7)
                    + storedInputData[1]);
            }
    }
}
```

9. now , the response is coming back to firmata.js (Refer line 63

<https://github.com/bromagosa/Snap4Arduino/blob/master/src/platforms/mobile/android/root/mobile/firmata.js>)

```
* Handles a ANALOG_MESSAGE response and emits 'analog-read' and 'analog-read-'+n events where n is the pin number.
* @private
* @param {Board} board the current arduino board we are working with.
*/

MIDI_RESPONSE[ANALOG_MESSAGE] = function(board) {
    var value = board.currentBuffer[1] | (board.currentBuffer[2] << 7);
    var port = board.currentBuffer[0] & 0x0F;
    if (board.pins[board.analogPins[port]]) {
        board.pins[board.analogPins[port]].value = value;
    }
    board.emit('analog-read-' + port, value);
    board.emit('analog-read', {
        pin: port,
        value: value
    });
};
```