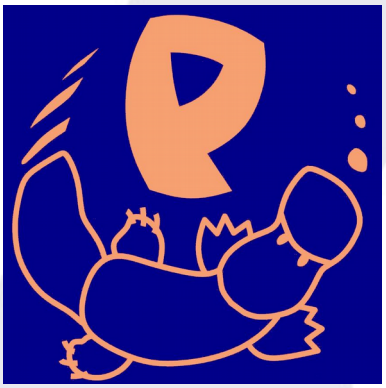
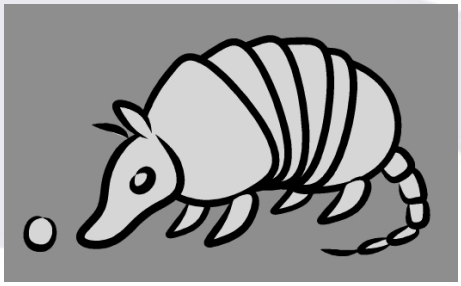
SELETEK DEVELOPER'S GUIDE





1) Introduction

This guide will explain how to control *any of the Seletek family of controllers*. Thus it will cover from the Original Seletek to the latest Dragonfly, including the Armadillos and Platypuses.

The rules, protocol, functions, objects... are always the same, with some commands restricted to the appropriate controllers – there is no stepper motor control commands in the Dragonfly, for instance.

Some conventions general to whatever method.

- Controller ports are numerated from 0 (MAIN) to 2 (THIRD) if available.
- Port pins from 0 to 9.
- Relays (Dragonfly or Firefly alike) and Sensors are numerated from 0 to 7.
- Analog values are 10 bit – so 0 to 1023.

It is important to note the controller will not impose any limits or block access to ports or pins... the software is responsible for that. In practice, this means if port MAIN is being used for moving a stepper motor, there is nothing in the controller's code that will reject pin 3 of that port being used independently – that responsibility relies in the application software.

PWM: all output pins can be PWM-regulated. There are however some limitations due to only existing 4 PWM signals in the microcontroller, and thus some pins are grouped together.

For controllers with 2 ports, the first 2 output pins of each port share a PWM signal, same for the next 2 pins.

For controllers with 3 ports (Platypuses), port MAIN takes 2 PWM (1 for pins 0, 1, next for pins 2, 3), port EXP takes another PWM (all its pins share it), same for port THIRD.

Final note: this document is a work in progress... I'll invest more time in it depending on the requests.

So, should you need more information, whatever, or something that's not clear enough, please contact me at jaime – at – lunatico.es

2) Communications protocol

Whatever the physical interface used to communicate with the controller, the protocol used is always the same, what we call SLP (for Seletek Line Protocol). It is based in plain text, ASCII messages, allowing for easier development and specially debug.

The available physical interfaces are:

- USB 'B' receptacle, device, VID/PID: (hexa) 16c0:09b0.
  - This USB is present in all controllers, but only used as main in the original Seletek. It was semi-exposed in the first generation of Armadillo under the "firmware update" sticker (mini USB receptacle)
- USB 'B' receptacle, device, VID/PID: (hexa) 16c0:09b1.
  - Main USB for the Armadillo and Platypus controllers (whatever version)
- Ethernet 100 Mbps
  - Available in the Platypus and Dragonfly controllers

In order to establish a dialog with the controller, either a serial link is used (the drivers for both USB interfaces will create a virtual COM port), with parameters 115200, n, 8, 1 – or UDP messages are sent to port (default) 10000. The controller will reply to the port originating the UDP packet.

Communications are always to be initiated by the PC, and every message will have one reply from the controller.

All messages will have the format:

```
!group command param1 [param2 [param3]]#
```

For example:

```
!stepper goto 1 23000#          → to command stepper at port 1 to position 23000
```

All replies will have the format:

```
!group command param1 [param2 [param3]]:result#
```

… result being the result code, in general 0 meaning OK. Following with the previous stepper command example, the controller will reply:

!stepper goto 1 23000:0# → meaning message understood

All textual commands have to be sent using lowercase (this includes commands and groups); so:

!Stepper GOTO 1 23000# → is invalid

The controller will ignore improperly formed messages and will reply with a meaningful message in case of wrong command, group, invalid argument, etc.

The system expects **ASCII text caracters**, "#", ":" and "!" being reserved.

Let's briefly enumerate all groups:

Stepper motor control: group name "step".
Available in: Seletek, Armadillo, Platypus
Comments: all commands take the port number (0..2) as the first parameter

Input reading: group name "read"
Available in: Seletek, Armadillo, Platypus
Comments: to read the input pins present in every SubD-9 port, and temperature sensors.

Direct output control: group name "write"
Available in: Seletek, Armadillo, Platypus
Comments: to handle the outputs in all SubD-9 ports, pin by pin.

General system commands: group name "seletek"
Available in: all controllers
Comments: miscellaneous commands. All network related commands (a lot!) are only available in Platypus and Dragonfly.

Relay / Sensor control commands: group name "relio"
Available in: all controllers (as Seletek, Armadillo, etc, can have a Dragonfly attached)
Comments:

- the port parameter is ignored in the Dragonfly
- This group handles the relays and sensors from a Dragonfly or Firefly box

DC motor command: group name "dcmot"
Available in: Seletek, Armadillo, Platypus
Comments: currently not used in any PC program

The following groups add quite powerful features to the controller, but need a bit of explanation. First we have "actions", that are basically full commands. We can add, remove, etc., these actions.

Then we have "triggers", as their name implies are basically sets of conditions.

By associating triggers and actions, we can have any action (again, basically any command of the groups above) executed when certain conditions (as specified by a trigger) are met.

Trigger group, name: "trigger"
Available in: all controllers
Comments: trigger management

Action group, name: "action"
Available in: all controllers
Comments: action management. Actions are specified as a normal commands, replacing the spaces with "_".

As this protocol is continuously evolving (being extended actually) , the current list of commands, with details about all of them, is to be found in a spreadsheet, available here.

You can ignore numbers at the left of some commands (are used as development aids), and in general it's best to ignore the flash memory commands, and the bootfrom, reboot, etc commands. There are also some comments intended for me – as said, this is my working document.

3) HTTP automation

Per some users' request, we have added a http layer so every command in the protocol can be sent to the controller using a simple http request (using curl, maybe).

So, for instance:

http://dragonfly/slp/seletek/version

… will return (plain text) the same we'd get using serial line or UDP, so:

!seletek version:4231#

Similarly:

http://platypus/slp/stepper/goto/0/300

… will move the motor at port MAIN to position 300.

As an aside, and for the dragonfly only at this moment, it's worth checking:

http://dragonfly/info
http://dragonfly/web
