cmd_response

simple command/response access via USB to Arduino's I/O capabilities

The Arduino family of microcontrollers possess a reasonable collection of input and output capabilities. While there are different versions, each with some unique capabilities, most controllers have a USB port, some digital input and output ports, a few pulse-width modulated (PWM) digital ports, and some analog input ports. This code module programs the Arduino with a simple command/response language via USB providing access to Arduino's I/O capabilities.

Arduino Interface

Table: Communications Parameters

term	value
communications rate	115200
line terminator	`\n`

Table: USB command interface

When present, "#" refers to the Arduino pin number used in the operation

command	value	action
`?ai pin`	0..1023	returns current value of numbered analog input
`?bi pin`	0..1	returns current value of numbered digital input
`!bo pin v`	0..1	sets numbered digital output to value v
`!pwm pin v`	0..255	sets numbered PWM digital output to value v
`!pin pin v`	0..1	sets mode of digital pin to value v (value: 1=OUTPUT, not 1=INPUT)
other		returns "ERROR_UNKNOWN_COMMAND:text"

notes:

1. must use lower case (as shown in table)
2. integers must be specified in decimal (no octal or hex interpreted)
3. pin numbers are not checked for correctness in the current version
4. "?" commands return an integer
5. "!" commands return "Ok"
6. Errors, starting with "ERROR_" will substitute for expected output

?ai command

format:	`?ai pin`
example:	`?ai 0`
returns:	`601`

Reads the analog input *pin* specified in the first argument. The returned value is between 0 and 1023 (10-bit ADC) which represents a measured voltage between 0 and 5 VDC.

> ***Note***
>
> pin number is not checked but will be in a future version (`0..NUM_ANALOG_INPUTS`)

?bi command

format:	`?bi pin`
example:	`?bi 3`
returns:	`1`

Reads the digital input *pin* specified in the first argument. The returned value is either 0 or 1.

> ***Note***
>
> pin number is not checked but will be in a future version (`0..NUM_DIGITAL_PINS`)

!bo command

format:	`!bo pin value`
example:	`!bo 6 0`
returns:	`Ok`

Sets the digital output *pin* specified in the first argument to the *value* specified in the second argument. If the syntax is correct and the value is within range, returns `Ok`. The pin must first be configured for output by calling `!pin pin 1`.

> ***Note***
>
> pin number is not checked but will be in a future version (`0..NUM_DIGITAL_PINS`)

!pwm command

format:	`!pwm pin value`
example:	`!pwm 11 127`
returns:	`Ok`

Sets the PWM-enabled digital output *pin* specified in the first argument to the *value* specified in the second argument. If the syntax is correct and the value is within range, returns `Ok`. The pin must first be configured for output by calling `!pin pin 1`.

The pin number is checked for PWM hardware-support by a call to the `digitalPinHasPWM(pin)` macro (which is defined by the Arduino SDK for each supported board type in <Arduino>/hardware/arduino/variants/*/pins_arduino.h).

!pin command

format:	`!pin pin value`
example:	`!pin 11 1`
returns:	`Ok`

Configures the digital output *pin* specified in the first argument for output as a binary digital output and also as a PWM digital output if the hardware enables this.

value	meaning
0	use pin as input
1	use pin as output

Note

pin number is not checked but will be in a future version (`0..NUM_DIGITAL_PINS`)

Examples

1. Read analog input from pin 0:

```
>>> ?ai 0
41
```

2. Set digital pin 11 for PWM output:

```
>>> !pin 11 1
Ok
```

3. Set PWM output pin 11 to 128:

```
>>> !pwm 11 128
Ok
```

4. Show how a bad command (no space between baseCmd and pin) is handled:

```
>>> !pwm11 128
ERROR_UNKNOWN_COMMAND:!pwm11 128
```

Error messages

This is a list of the possible error messages and their meanings. All error messages begin with the text `ERROR_` and then some terse descriptor of the error. In most cases, the input that triggered the error message is returned. A single ":" is used as the delimiter when the input is appended.

`ERROR_BINARY_RANGE:input`

The value must be either `0` or `1`.

`ERROR_BUFFER_OVERFLOW`

Too many characters were received before the line terminator. All characters received so far will be discarded.

Some Arduinos do not have much available RAM. The current buffer length is 40 characters.

`ERROR_COMMAND_FORMAT:input`

All commands must have at least one space separating the baseCmd from the pin number. This command is generated when no space is detected in the input.

`ERROR_PIN_NOT_PWM:input`

The specified pin is not supported for PWM on this Arduino hardware. This is determined by calling the Arduino system macro `digitalPinHasPWM(pin)` which is defined for each different type of Arduino hardware variation.

`ERROR_PWM_RANGE:input`

The PWM value must be between 0 and 255, inclusive. This error is reported for any value outside this range.

`ERROR_UNKNOWN_COMMAND:input`

The input was not recognized as a valid command. One reason for this might be the use of upper case. Other possibilities exist.

`ERROR_TOO_MANY_ARGUMENTS:input`

The general form for input commands is `baseCmd pin [value]` where `baseCmd` is given in the table above, `pin` is an integer appropriate for the chosen hardware interface, and `value` is only used for "!" (set) commands.

At this time, if `value` is specified for a "?" (read) command, it is ignored. In the future, this will generate an error message.