Lego Infrared documentation

# Purpose

This documentation has the purpose to explain how to pilot and interact with the board and pilot either a Lego train, or any other FPS Lego element, change a switch and a signal.

V1 is using a Netduino code running .NET Microframework

V2 is using a RaspberryPi 2 running Windows 10 IoT Core

Most of the documentation is working for both versions

# Introduction

The board can be piloted either thru HTTP or serial (Netduino version only). Both are using a very similar protocol and are command (URL) based with arguments.

Most of the settings can be personalized like the train names, the switches names. This feature is available using a config file. A full Excel file is available to create the right configuration file. This document also explain how to use it.

# Web server configuration

The webserver is configured by default on port 80. The only way to change it is in the code. If you need to change the port because you want to expose it on the internet, I recommend you to change the route in your router and not the port on the device.

# Serial port configuration

This is only for the Netduino version. Not for the Windows 10 IoT Core.

The serial port is configured with the following settings:

MySerialPIC.BaudRate = 9600;

MySerialPIC.DataBits = 8;

MySerialPIC.Parity = Parity.None;

MySerialPIC.StopBits = StopBits.One;

MySerialPIC.Handshake = Handshake.None;

MySerialPIC.ReadTimeout = 10;

As for the web server, there is no way to change the settings in a configuration file. To change them, you’ll have to change the values in the code and recompile the project.

The serial port has to be activated in the config file by setting the ‘dbg’ argument to 1. See the config file section for more details. If the ‘dbg’ argument is not present or with another value, the serial port won’t be activated and it won’t be possible to pilot the board via serial port.

Please note that activating the serial port does not deactivate the web server.

# Piloting and accessing switches

You can pilot 1 switch at a time and change the position either from straight to turned or turned from straight.

## HTTP Communication protocol

The communication protocol looks like: <http://ipaddress/switch.aspx?si=0&md=0&sec=key123>

The answer looks like:

Status of Switch

Switch 0: False

ipaddress: This is the IP Address or name of your device.

switch.aspx: This is the name of the page called to manage the switches

Here is the list of all available arguments. Please note that none of the arguments or the page name is case sensitive. I do recommend to use lowercase.

### si

this argument determine which switch you want to control. Values starts at 0 and finish at 7 maximum.

Any other value will give you the status of all switches like:

Status of Switch

Switch 0: False  
Switch 1: False  
Switch 2: True  
Switch 3: False  
Switch 4: True  
Switch 5: False  
Switch 6: False  
Switch 7: False

### md

accepted values are 0 for straight and 1 for turned.

### sec

this is the security key. It can be any string made with text except non ANSI characters. If the key is not present, you’ll get the following error:

Invalid authentication Key

### no

use this argument with 1 as a value to return only the status of the switches. Returned value looks like with no HTTP header.

False False True False True False False False

Order of the values is always from switch 0 to 7. If you’ve setup a maximum number of switches, only values related to the number of switches will be returned.

## Serial communication protocol

The serial protocol is very similar to the HTTP one. Main difference is that only the page and the arguments are send and no security key is required.

So a command looks like switch.aspx?si=0&md=0\r

Return values is either OK\r or Problem\r

The \r is the return character which mark the end of the line. All arguments are the same as for the http protocol.

The only difference is that when using the ‘no’ argument and a value different from 0 to 7 for ‘si’, the return is 00101000\r which does represent the positioning of the switches. Be careful as if you are not using the ‘no’ argument, you’ll get an HTML answer similar to the one from HTTP protocol.

# Piloting and accessing signals

You can pilot 1 signal at a time. Red = 0 and Green = 1.

## HTTP Communication protocol

The communication protocol looks like: <http://ipaddress/signal.aspx?si=0&md=1&sec=key123>

The answer looks like:

Status of Signal

Signal 0: False

ipaddress: This is the IP Address or name of your device.

signal.aspx: This is the name of the page called to manage the signals

Here is the list of all available arguments. Please note that none of the arguments or the page name is case sensitive. I do recommend to use lowercase.

### si

this argument determine which signal you want to control. Values starts at 0 and finish at 15 maximum.

Any other value will give you the status of all switches like:

Status of Signal

Signal 0: False  
Signal 1: False  
Signal 2: True  
Signal 3: False  
Signal 4: True  
Signal 5: False

In this case, only 6 signals has been declared in the config file. You can have more info regarding the config file in the dedicated section.

### md

accepted values are 0 for red and 1 for green.

### sec

this is the security key. It can be any string made with text except non ANSI characters. If the key is not present, you’ll get the following error:

Invalid authentication Key

### no

use this argument with 1 as a value to return only the status of the switches. Returned value looks like with no HTTP header.

False False True False True False False False

Order of the values is always from signal 0 to 15. If you’ve setup a maximum number of signals, only values related to the number of signals will be returned.

## Serial communication protocol

The serial protocol is very similar to the HTTP one. Main difference is that only the page and the arguments are send and no security key is required.

So a command looks like signal.aspx?si=0&md=0\r

Return values is either OK\r or Problem\r

The \r is the return character which mark the end of the line. All arguments are the same as for the http protocol.

The only difference is that when using the ‘no’ argument and a value different from 0 to 15 for ‘si’, the return is 00101000\r which does represent the signal status. Be careful as if you are not using the ‘no’ argument, you’ll get an HTML answer similar to the one from HTTP protocol.

# Piloting a Lego FPS

The Lego communication is quite complex and there are many options. I do recommend you first get familiar with the Lego communication protocol before trying to pilot your own elements.

## Pre reading and important notice

Please read the Lego protocol before reading the next section. <http://www.philohome.com/pf/LEGO_Power_Functions_RC.pdf>

The infrared protocol is over the air and it may happen that the receptor won’t see it correctly. **You are never sure that the order you send will arrive**. This is not due to my system but due to the fact that infrared communications are very sensitive with other infrared activities. That as it’s lights, even if they may be reflected on walls, an obstacle can stop it and avoid the signal to be received.

Due to the Lego protocol itself any order takes about 1 second to be output totally. So don’t try to pilot too many elements with single commands at the same time or with a short timing. I’ve implemented special functions to be able to pilot up to 8 elements during the same command (so in 1 second).

You can also read those 2 blog posts to understand the way the board is working: <http://blogs.msdn.com/b/laurelle/archive/2012/04/07/using-netduino-and-net-microframework-to-pilot-any-lego-power-function-thru-infrared-part-1.aspx> and <http://blogs.msdn.com/b/laurelle/archive/2012/04/17/using-netduino-and-net-microframework-to-pilot-any-lego-power-function-thru-infrared-part-2.aspx>

They both also explain a bit how the Lego protocol is working.

## Differences between HTTP and Serial

As for the signals and switches piloting there are only few differences between the web server and serial piloting.

A web server request looks like <http://ipaddress/combo.aspx?rd=0&bl=0&ch=0&sec=key123>

The same serial request looks like combo.aspx?rd=0&bl=0&ch=0

In both cases the Return values is either OK\r or Problem\r as pure text without any http header. \r is the return character.

OK is returned if everything was correct. Problem if a value is not correct, or any other error appeared.

As for signals and switches protocols, the command and arguments are not case sensitive. I do recommend to use lowercase characters.

### sec

this is the security key and it’s the only argument that is used only for http. It can be any string made with text except non ANSI characters. If the key is not present, you’ll get the following error:

Invalid authentication Key

In the description of following sections, I will use the simplified serial version as the ‘sec’ argument has to be present in all the calls for HTTP.

## Combo Mode

This mode is able to control: Two outputs float/forward/backward/brake.

This is a combo command controlling the state of both output Red and Blue at the same time.

This mode has timeout for lost IR.

Usage: combo.aspx?rd=1&bl=8&ch=2

### rd

this argument control the red output. Accepted values are the following:

* No change = 0 (0x0)
* Forward = 1 (0x1)
* backward = 2 (0x2)
* Break = 3 (0x3)

### bl

this argument control the blue output. Accepted values are the following:

* No change = 0 (0x0)
* Forward = 4 (0x4)
* backward = 8 (0x8)
* Break = 12 (0xC)

### ch

this argument selects the channel of the receptor. Accepted values goes from 0 to 3 for channel 1 to 4.

## Single pin continuous Mode

This mode is able to control: Clear/set/toggle of an individual pin C1 or C2 on output red or blue.

This mode has no timeout for lost IR.

Usage: continuous.aspx?fc=2&op=2&ch=2

### fc

this argument select what you want to do with the specific pin output, accepted values are the following:

* No change = 0 (0x0)
* Clear = 1 (0x1)
* Set = 2 (0x2)
* Toggle = 3 (0x3)

### op

this argument select the output and pin you want to control, accepted values are the following:

* Red PINC1 = 0 (0x0)
* Red PINC2 = 1 (0x1)
* Blue PINC1 = 2 (0x2)
* Blue PINC2 = 3 (0x3)

### ch

this argument select the channel of the receptor. Accepted values goes from 0 to 3 for channel 1 to 4.

## Single pin timeout Mode

This mode is able to control: Clear/set/toggle of an individual pin C1 or C2 on output red or blue.

This mode has timeout for lost IR.

Usage: timeout.aspx?fc=1&op=1&ch=1

### fc

this argument select what you want to do with the specific pin output, accepted values are the following:

* No change = 0 (0x0)
* Clear = 1 (0x1)
* Set = 2 (0x2)
* Toggle = 3 (0x3)

### op

this argument select the output and pin you want to control, accepted values are the following:

* Red PINC1 = 0 (0x0)
* Red PINC2 = 1 (0x1)
* Blue PINC1 = 2 (0x2)
* Blue PINC2 = 3 (0x3)

### ch

this argument select the channel of the receptor. Accepted values goes from 0 to 3 for channel 1 to 4.

## Single output Mode with PWM

This mode is able to control: One output at a time with PWM.

This mode has no timeout for lost IR.

This function is most likely use to set a nominative speed to a motor.

Usage: singlepwm.aspx?pw=4&op=1&ch=3

### pw

this argument indicate which PWM value the motor (or lamp) has to take, accepted values are the following:

* Float = 0 (0x0)
* PWM forward step 1 = 1 (0x1)
* PWM forward step 2 = 2 (0x2)
* PWM forward step 3 = 3 (0x3)
* PWM forward step 4 = 4 (0x4)
* PWM forward step 5 = 5 (0x5)
* PWM forward step 6 = 6 (0x6)
* PWM forward step 7 = 7 (0x7)
* Brake = 8 (0x8)
* PWM backward step 7 = 9 (0x9)
* PWM backward step 6 = 10 (0xA)
* PWM backward step 5 = 11 (0xB)
* PWM backward step 4 = 12 (0xC)
* PWM backward step 3 = 13 (0xD)
* PWM backward step 2 = 14 (0xE)
* PWM backward step 1 = 15 (0xF)

### op

this argument select the output either red =0 or blue = 1.

### ch

this argument select the channel of the receptor. Accepted values goes from 0 to 3 for channel 1 to 4.

## Single output Mode Clear Set Toggle

This mode is able to control: One output at a time with clear/set/toggle control pins.

This mode has no timeout for lost IR on all commands except “full forward” and “full backward”.

Usage: singlecst.aspx?pw=4&op=1&ch=2

### pw

this argument is used to select the right behavior on pins. The following values are accepted:

* Clear C1 + Clear C2 = 0 (0x0)
* Set C1 + Clear C2 = 1 (0x1)
* Clear C1 + Set C2 = 2 (0x2)
* Set C1 + Set C2 = 3 (0x3)
* Increment PWM = 4 (0x4)
* Decrement PWM = 5 (0x5)
* Full forward (timeout) = 6 (0x6)
* Full backward (timeout) = 7 (0x7)
* Toggle full forward/backward (default forward)

### op

this argument select the output either red =0 or blue = 1.

### ch

this argument select the channel of the receptor. Accepted values goes from 0 to 3 for channel 1 to 4.

## Combo PWM Mode

This mode is able to control: Two outputs with PWM in 7 steps forward and backward.

This is a combo command controlling the state of both output red and blue at the same time.

This mode has timeout for lost IR.

Usage: combopwm.aspx?p1=4&p2=10&ch=1

### p1

this argument indicate which PWM value the motor (or lamp) has to take, accepted values are the following:

* Float = 0 (0x0)
* PWM forward step 1 = 1 (0x1)
* PWM forward step 2 = 2 (0x2)
* PWM forward step 3 = 3 (0x3)
* PWM forward step 4 = 4 (0x4)
* PWM forward step 5 = 5 (0x5)
* PWM forward step 6 = 6 (0x6)
* PWM forward step 7 = 7 (0x7)
* Brake = 8 (0x8)
* PWM backward step 7 = 9 (0x9)
* PWM backward step 6 = 10 (0xA)
* PWM backward step 5 = 11 (0xB)
* PWM backward step 4 = 12 (0xC)
* PWM backward step 3 = 13 (0xD)
* PWM backward step 2 = 14 (0xE)
* PWM backward step 1 = 15 (0xF)

### p2

this argument indicate which PWM value the motor (or lamp) has to take, accepted values are the following:

* Float = 0 (0x0)
* PWM forward step 1 = 1 (0x1)
* PWM forward step 2 = 2 (0x2)
* PWM forward step 3 = 3 (0x3)
* PWM forward step 4 = 4 (0x4)
* PWM forward step 5 = 5 (0x5)
* PWM forward step 6 = 6 (0x6)
* PWM forward step 7 = 7 (0x7)
* Brake = 8 (0x8)
* PWM backward step 7 = 9 (0x9)
* PWM backward step 6 = 10 (0xA)
* PWM backward step 5 = 11 (0xB)
* PWM backward step 4 = 12 (0xC)
* PWM backward step 3 = 13 (0xD)
* PWM backward step 2 = 14 (0xE)
* PWM backward step 1 = 15 (0xF)

### ch

this argument select the channel of the receptor. Accepted values goes from 0 to 3 for channel 1 to 4.

### Combo Mode All

This special function allow to control at the same time all 4 channels and all red and blue output at 1 time! This is something you cannot do with regular remote control.

As for the normal Combo Mode, this function has timeout on lost IR.

Usage: comboall.aspx?rd0=1&bl0=0&rd1=2&bl1=4&rd2=3&bl2=8&rd3=0&bl3=12

## rdX

X from 0 to 3 representing channel 1 to 4.

this argument control the red output. Accepted values are the following:

* No change = 0 (0x0)
* Forward = 1 (0x1)
* backward = 2 (0x2)
* Break = 3 (0x3)

### blX

X from 0 to 3 representing channel 1 to 4.

this argument control the blue output. Accepted values are the following:

* No change = 0 (0x0)
* Forward = 4 (0x4)
* backward = 8 (0x8)
* Break = 12 (0xC)

## Single Countinuous Mode All

As for the Single Continuous Mode, this mode is able to control: Clear/set/toggle of an individual pin C1 or C2 on output red or blue but on 4 channels in 1 time. This is something you cannot do with a regular remote control. Like for the normal function, there is no timeout for lost IR.

Usage: continuousall.aspx?fc0=2&op0=0&fc1=1&op1=1&fc2=0&op2=2&fc3=3&op3=2

### fcX

X from 0 to 3 representing channel 1 to 4.

this argument select what you want to do with the specific pin output, accepted values are the following:

* No change = 0 (0x0)
* Clear = 1 (0x1)
* Set = 2 (0x2)
* Toggle = 3 (0x3)

### opX

X from 0 to 3 representing channel 1 to 4.

this argument select the output and pin you want to control, accepted values are the following:

* Red PINC1 = 0 (0x0)
* Red PINC2 = 1 (0x1)
* Blue PINC1 = 2 (0x2)
* Blue PINC2 = 3 (0x3)

### Single PWM Mode All

As for the normal function, this mode is able to control: One output at a time with clear/set/toggle control pins but on 4 channels at the same time. This is something you can’t do with a regular remote control.

This mode has no timeout for lost IR on all commands except “full forward” and “full backward”.

Usage: singlepwmall.aspx?pw0=0&op0=0&pw1=6&op1=1&pw2=12&op2=1&pw3=2&op3=0

### pwX

X from 0 to 3 representing channel 1 to 4.

this argument is used to select the right behavior on pins. The following values are accepted:

* Clear C1 + Clear C2 = 0 (0x0)
* Set C1 + Clear C2 = 1 (0x1)
* Clear C1 + Set C2 = 2 (0x2)
* Set C1 + Set C2 = 3 (0x3)
* Increment PWM = 4 (0x4)
* Decrement PWM = 5 (0x5)
* Full forward (timeout) = 6 (0x6)
* Full backward (timeout) = 7 (0x7)
* Toggle full forward/backward (default forward)

### opX

X from 0 to 3 representing channel 1 to 4.

this argument select the output either red = 0 or blue = 1.