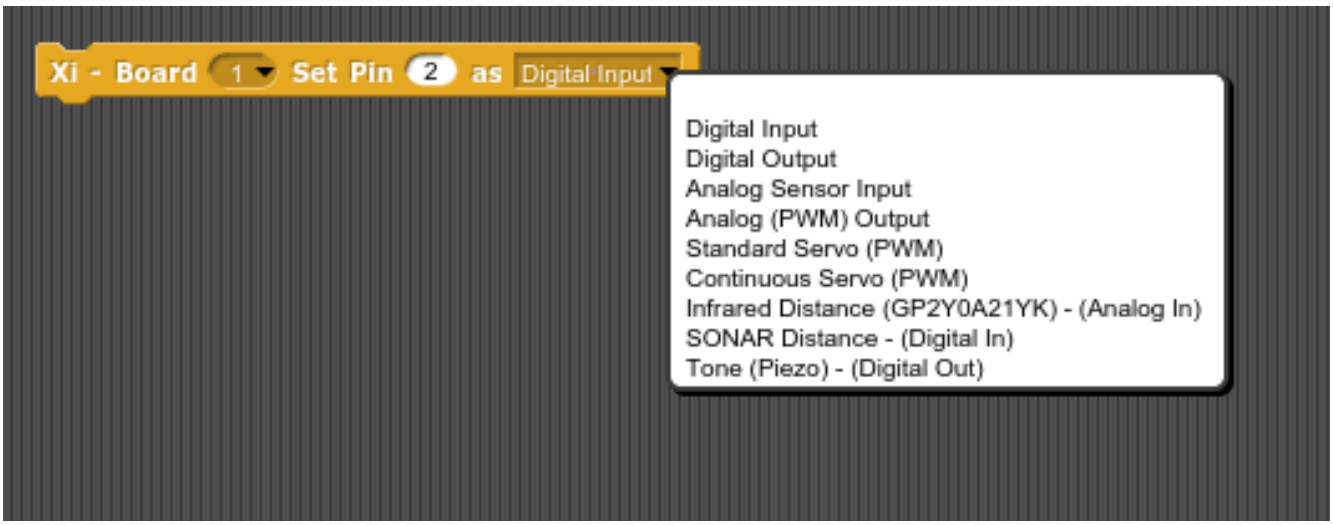
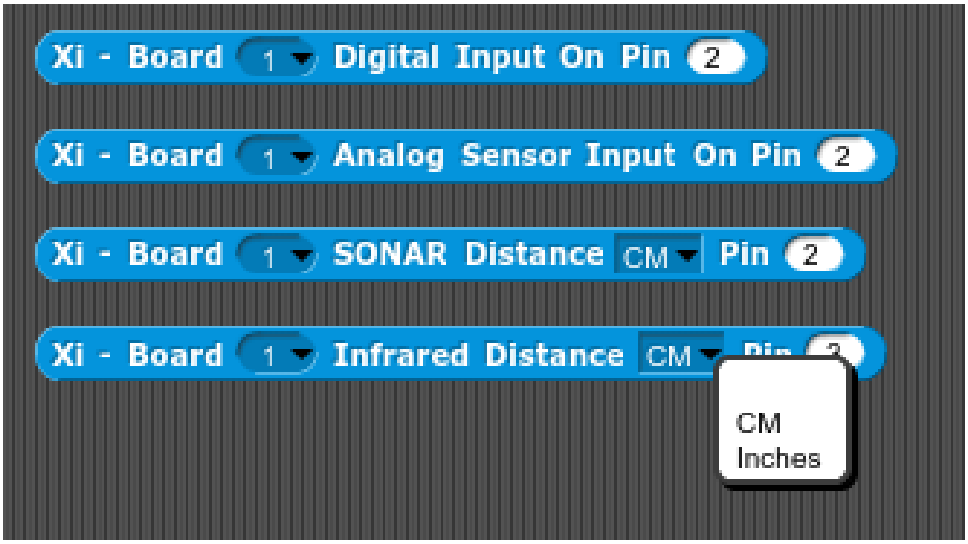
Xi4S / Xi4Snap

Xi For Scratch / Snap

Installation and Usage Guide





The Cross-Platform Interconnect

Copyright © 2014 Alan Yorinks. All rights reserved.

This manual is distributed WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Software Release v.003 (xiserver.js)

Document Release Date: 31 December 2014

Table of Contents

1.Introduction....1
1.1. Supported Operating Systems....2
1.1.1. Linux....2
1.1.2. Windows 8.1....2
1.1.3. MAC....2
1.2. Verified Supported Board Types....2
1.2.1. Arduino....2
1.2.2. BeagleBone Black....2
1.2.3. Raspberry Pi....2
1.3. Functionality Matrix....2
1.4. Browser Support....3
2. Installation....3
2.1. Xi Servers....3
2.1.1. XiDuino....4
2.1.1.1 Experimental Features That Require A Non-Standard Sketch....5
2.1.1.1.1Sonar....5
2.1.1.1.2Stepper Motor Support....5
2.1.1.1.3 Install The Required Development Tools....6
2.1.1.1.4 Installing node.js....6
2.1.1.1.5 Download the Xi Distribution....6
2.1.1.1.6 Run the Install Shell Script....7
2.1.1.2 Windows 7 or 8.1....7
2.1.1.2.1 Install The Optional Development Tools....7
2.1.1.2.2 Installing node.js....8
2.1.1.2.3 Download the Xi Distribution....9
2.1.1.2.4 Run the Install .bat Script....10
2.1.1.2.5 Creating a Desktop Launch Icon....10
2.1.2. XiBone....12
2.1.2.1 Installing The Required Development Tools....12
2.1.2.2 Installing Node.js....12
2.1.2.3 Download the Xi Distribution....12
2.1.2.4 Run the Install Shell Script....13
2.1.3. XiPi....13
2.1.3.1 Installing The Required Development Tools....13
2.1.3.2 Installing Node.js....13
2.1.3.3 Download The Xi Distribution....14
2.1.3.4 Run The Shell Install Script....14
3. Launching a XiServer....14
3.1. Using the startup scripts....14
3.1.1. Launching a XiDuino Server....14
3.1.1.1 Windows....14
3.1.1.2 Linux....15
3.1.2.Launching a XiBone Server....15
3.1.3. Launching a XiPi Server....16

3.2. Custom Launch Using Xiserver Command Line Options....16
3.2.1. boardType....16
3.2.1.1 Default boardType....16
3.2.2. arduinoURL....17
3.2.2.1 Default URLs....17
3.2.2.2 Suppressing Browser Launch....17
3.2.3.debugLevel....17
3.2.3.1Default debugLevel....17
3.2.4. arduinoPort....17
3.2.4.1Default arduinoPort....17
3.2.5.ipPort....17
3.2.5.1Default ipPort....17
4.Using the Xi Clients....18
4.1. The Scratch Xi4S client....18
4.2. The Xi4Snap! Client....19
5.Using the Xi4S and the Xi4Snap! Blocks....20
5.1. Pin Designations....20
5.1.1. Arduino....20
5.1.2. BeagleBone Black....21
5.1.3. Raspberry Pi....22
5.2. Command Blocks....22
5.2.1.Selecting an IP Address and Port for the Connected Boards....22
5.2.2.Setting the Pin Mode....23
5.2.3.Digital Write....24
5.2.4.Analog Write (PWM)....24
5.2.5.Move Standard Servo....24
5.2.6.Move Continuous Servo....25
5.2.7.Stop Servo....25
5.2.8.Play Tone....26
5.2.9.Turn Tone Off....26
5.2.10.Set Debug Level....26
5.2.11.Set Pins For 4 Wire Bipolar Stepper Motor....27
5.2.12.Set Pins for Stepper Driver Board....27
5.2.13.Move Stepper Motor....28
5.2.14.Stop Stepper Motor....28
5.3. Reporter Blocks....28
5.3.1.Digital Input....29
5.3.2.Analog Sensor Input....29
5.3.3.Infrared Distance....29
5.3.4.SONAR Distance....30
6.Some Usage Tips....30
6.1. Keeping Track of Boards While Programming....30
6.2. Create Your Own Custom Blocks for a Higher Level Abstraction of the Boards and Their Connected Devices....30
7.Getting Help....32

1. Introduction

Xi4S is a JavaScript client implementation of the latest Scratch 2.0 Hardware Extension specification. With Xi4S, you will be able to connect one or several hardware boards to Scratch, in any combination, and have Scratch control them all. Scratch can receive data from one board and then use that data to control another board. Each board type has its own server - XiDuino for Arduino, XiBone for BeagleBone Black or XiPi for Raspberry Pi.

Xi4Snap is the Xi JavaScript client for Snap! and it uses the same servers that are used by Xi4S and has the same capabilities.

Each board type requires its own server instance – each Arduino will have an instance of XiDuino running on a PC, each BeagleBone Black will run its own copy of XiBone, and each Raspberry Pi will run its own copy of XiPi.

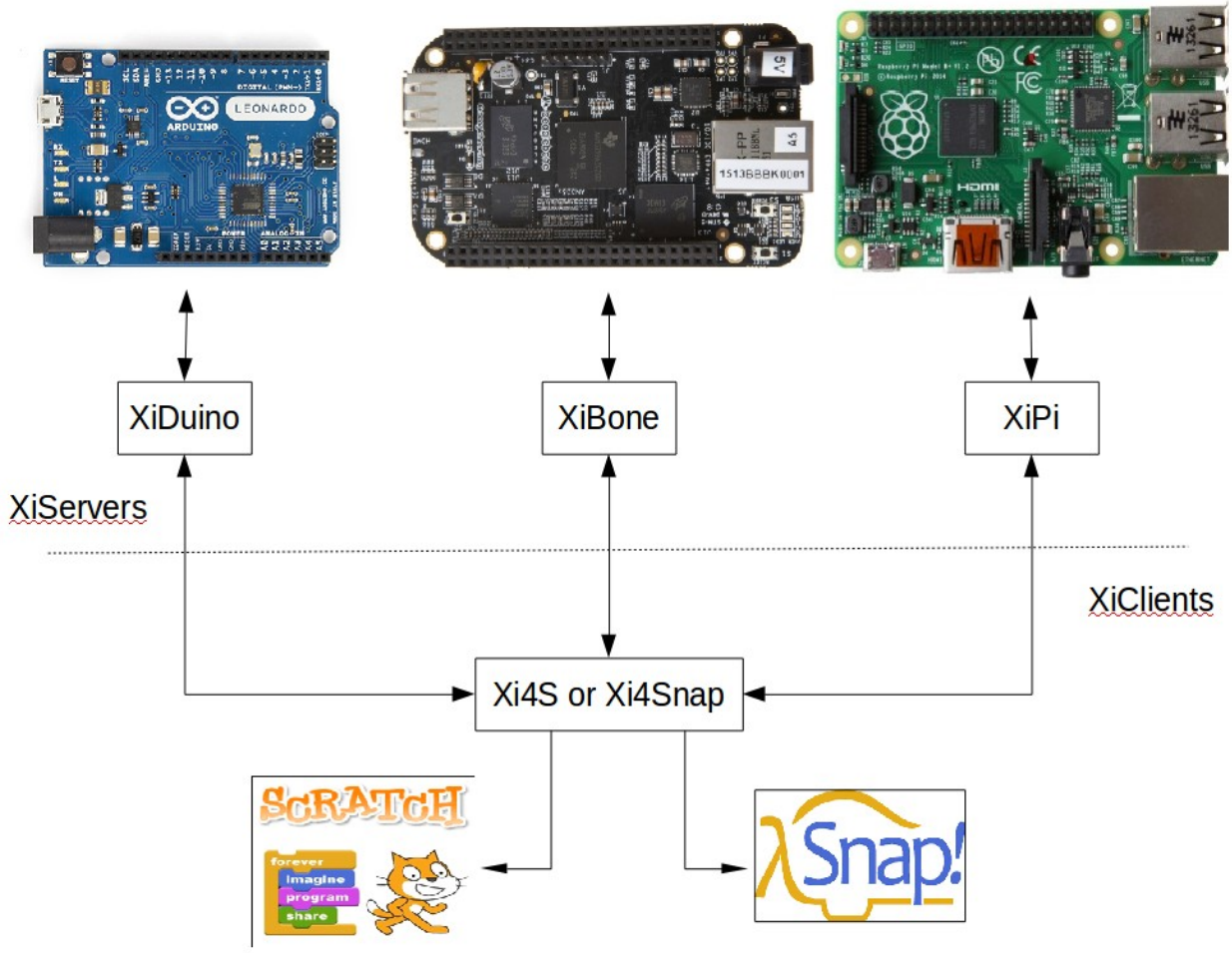


Figure 1: Xi Client/Server Architectural Model

In a nutshell, Xi provides a consistent and easy to use Scratch or Snap! programming interface that supports the most popular micro-controllers and single board computers. It makes programming these devices as easy as creating any other Scratch or Snap! program.

1.1. Supported Operating Systems

Xi is operating system independent and designed to run on:

1.1.1. Linux

- Desktop – tested on Ubuntu 14.10
- BeagleBone Black – tested on the Debian eMMC 2014-05-14 version.
- Raspberry Pi – tested on the Raspbian Debian Wheezy 2014-06-20 version.

1.1.2. Windows 8.1

- Tested on Windows 8.1 32 bit, but Windows 7 should work as well.

1.1.3. MAC

- It has not yet been tested on MAC, but since Xi uses components that are known to work on the MAC, it is fully anticipated that it will work without issue.
- Use the Linux installation instructions as a guide.
- **NOTE:** If you are a MAC user and wish to try Xi4S or Xi4Snap, please send your results, good or bad, to MisterYsLab@gmail.com

1.2. Verified Supported Board Types

1.2.1. Arduino

Tested with Uno and Leonardo (Support for Yun planned for a future release).

1.2.2. BeagleBone Black

Tested with Rev A6A. (Should work for any version running Debian)

1.2.3. Raspberry Pi

Tested with Model B version 2. (Should work with A, B version 1, and B+ as well).

1.3. Functionality Matrix

The following table summarizes the current level of support for each board type. The goal is to have

full and equal functionality across all three board types.

Feature	Arduino	BeagleBone Black	Raspberry Pi
Digital Input	X	X	X
Digital Output	X	X	X
Analog Input	X	X	
Analog Output (PWM)	X	X	X
Standard Servo	X	X	X
Continuous Servo	X	X	X
Infrared Distance Sensor	X	X	
SONAR Distance Sensor	X		
Piezo Tone Generation	X	X	
4 Wire Stepper Motor Control	X		
Driver (EasyDriver) Stepper Motor Control	X		

In addition, XiDuino provides automatic COM port detection (thanks to the Johnny-Five library). It also automatically launches your default browser to the Scratch or Snap! web page when it is started.

1.4. Browser Support

Xi works best with the Google Chrome browser, but works with Firefox as well. It will ***not*** work with Internet Explorer.

2. Installation

This section begins with the Xi server installation procedures. Each server type is addressed separately and for Arduino boards, a separate procedure is provided for Linux and Windows. Mac users can use the Linux procedures as a guide.

Installation procedure is the same whether you will be using the Xi4S Scratch client or Xi4Snap Snap! Client.

2.1. Xi Servers

For Arduino, a XiDuino server needs to be installed on your PC for each Arduino that you wish to

control.

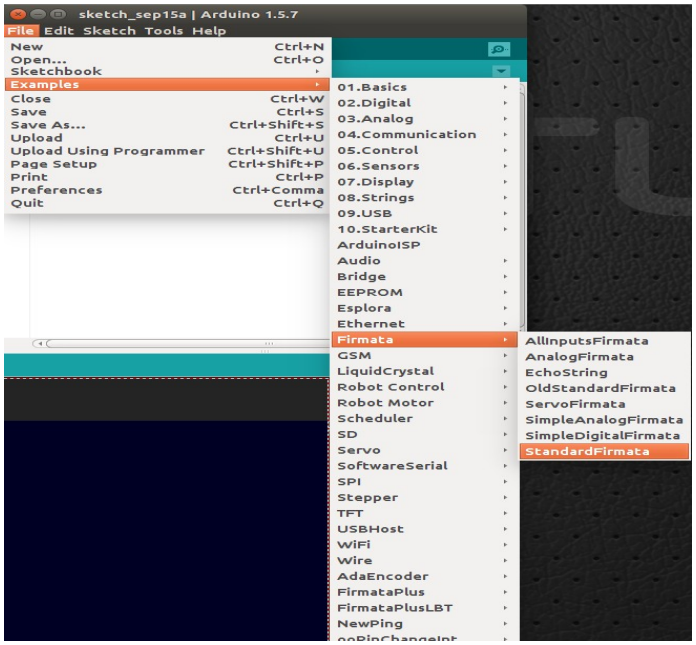
For BeagleBone Black and Raspberry Pi, XiBone or XiPi respectively, needs to be installed on each board you are adding to the system.

2.1.1. XiDuino

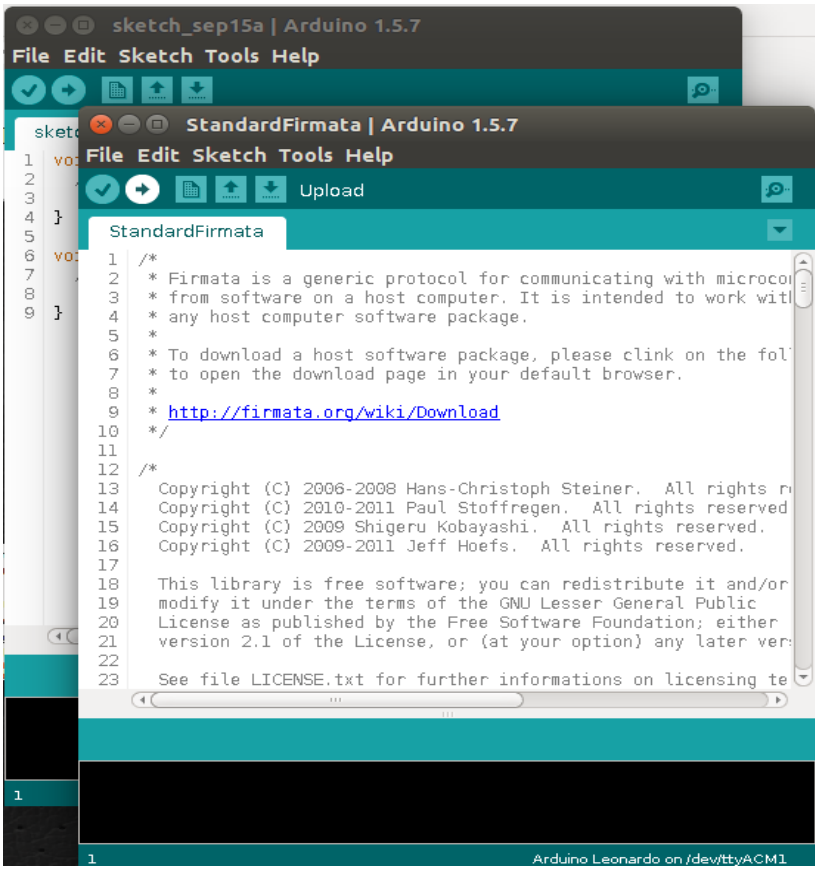
XiDuino requires that either Standard Firmata or FirmataPlus be installed on your Arduino board.

If you need to install StandardFirmata, first install the Arduino IDE for your platform located at this link: http://arduino.cc/en/Main/Software.

Next, open the IDE and then click on File/Examples/Firmata/StandardFirmata.



Finally upload to the Arduino board.



2.1.1.1 Experimental Features That Require A Non-Standard Sketch

2.1.1.1.1 Sonar

The Sonar feature that supports the HC-SR04 type devices, requires a special sketch that replaces StandardFirmata. This sketch does not support stepper motor support, but supports all other features.

Instructions for compiling and installing this sketch may be found at this link: https://github.com/rwaldron/johnny-five/wiki/Ping#setup

2.1.1.1.2 Stepper Motor Support

Stepper motor support requires a special version of StandardFirmata, called AdvancedFirmata.

Instructions for its use may be found at this link: https://github.com/soundanalogous/AdvancedFirmata/

This sketch does not contain SONAR support, but supports all other features

Make sure that the version of Johnny-Five is up to date. You can run “npm update johnny-five” to update to the latest version. Execute this command when in the xiserver directory of the installation.

2.1.1.1.3 Install The Required Development Tools

Make sure that you have the following tools in place on your system. If you are using Ubuntu, they may be installed using the Ubuntu Software Center. Ubuntu package names are given in the table below.

For distributions other than Ubuntu, check your distribution package manager to determine if the required packages have been installed.

Tool	Ubuntu Software Center Package Name
C++ Compiler	build-essential
make	build-essential
curl	Command line tool for transferring data with URL syntax
Python 2.6 or 2.7	Python Interactive high-level object-oriented language (default version)

2.1.1.1.4 Installing node.js

Follow the directions for your distribution on this web page:
https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager

For Ubuntu 14.04, I used the following commands to install node.js

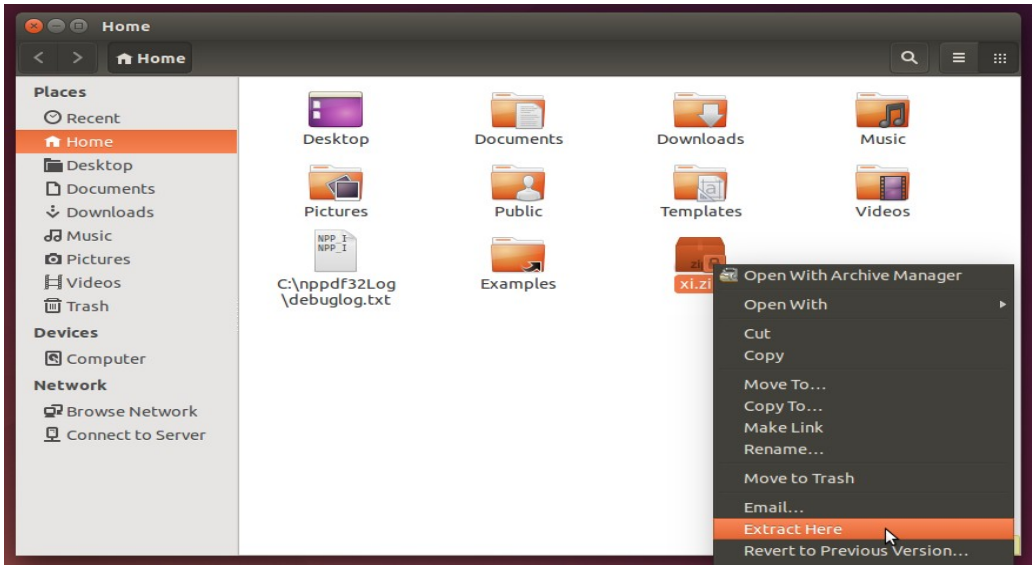
```
curl -sL https://deb.nodesource.com/setup | bash -
sudo apt-get install nodejs
node -v
```

The node -v command will display the version of node that was installed.

2.1.1.1.5 Download the Xi Distribution

Download the Xi distribution from github at https://github.com/MrYsLab/xi.

Place this file in your home directory and extract all of the files.



After extracting, open a command terminal and enter the following command.

```
cd ~/xi/servers/xiduino/linux
```

Next, change the permissions on both files in this directory by typing:

```
chmod ugo+x *
```

2.1.1.1.6 Run the Install Shell Script

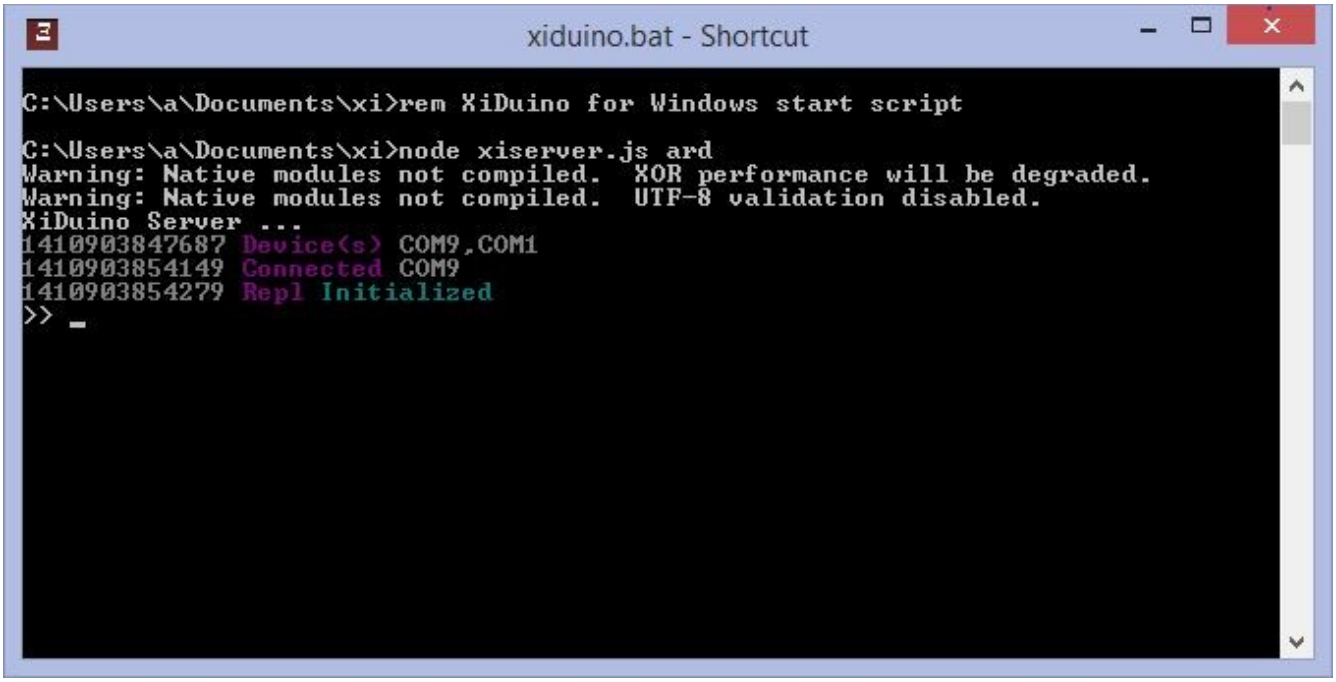
In `~/xi/servers/xiduino/linux`, enter the following command:

```
bash xiduinoInstall.sh
```

2.1.1.2 Windows 7 or 8.1

2.1.1.2.1 Install The Optional Development Tools

This is an optional step. If you choose to skip it, when XiDuino starts, you may see the following warnings:



XiDuino appears to run without problems in spite of these warnings, however if you wish to run a natively compiled version of websocket, you will need to load Python 2.6 or 2.7, and the Visual Express Development Package before running the xiduinoInstall.bat script.

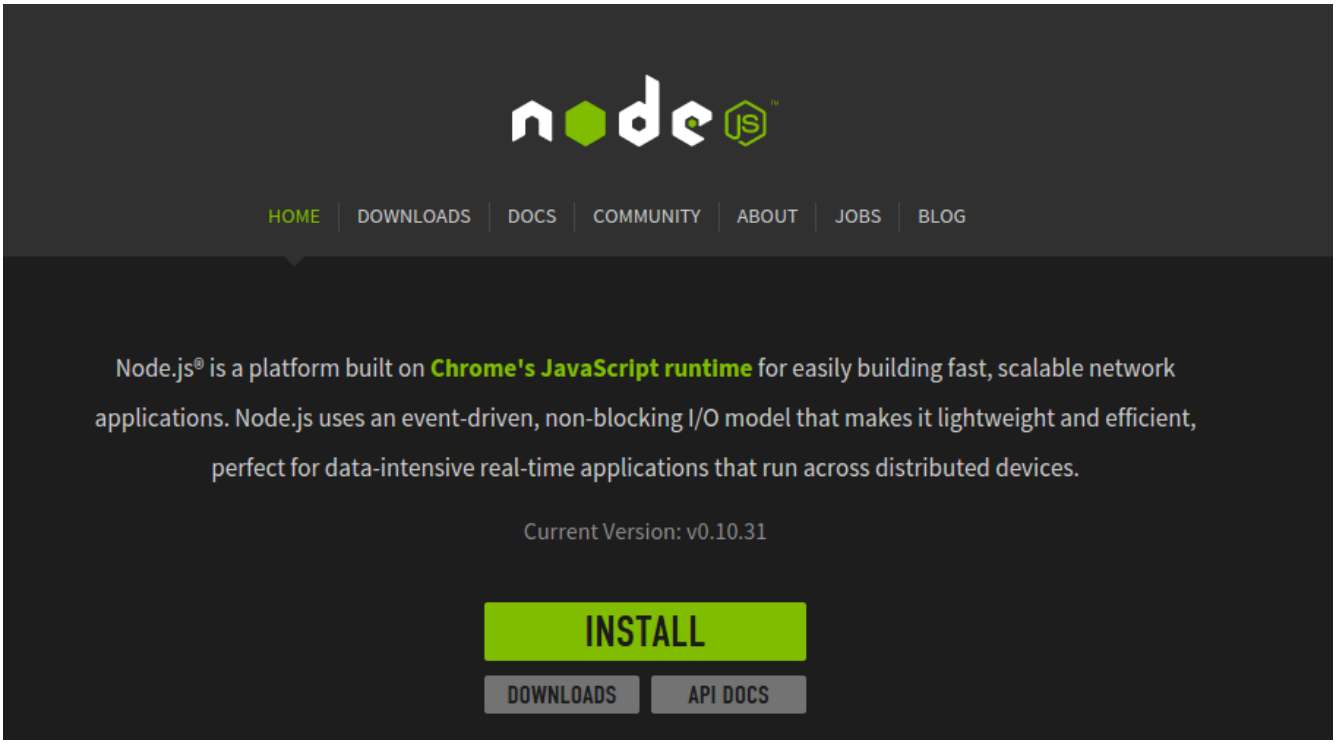
Be warned, the Visual Express Development package takes up a huge amount of disk space when it is installed.

To install Python 2.7, go to this web page, https://www.python.org/download/releases/2.7.8/ and download and install.

To install the Visual Express Package, go to this web page, http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx, download and install.

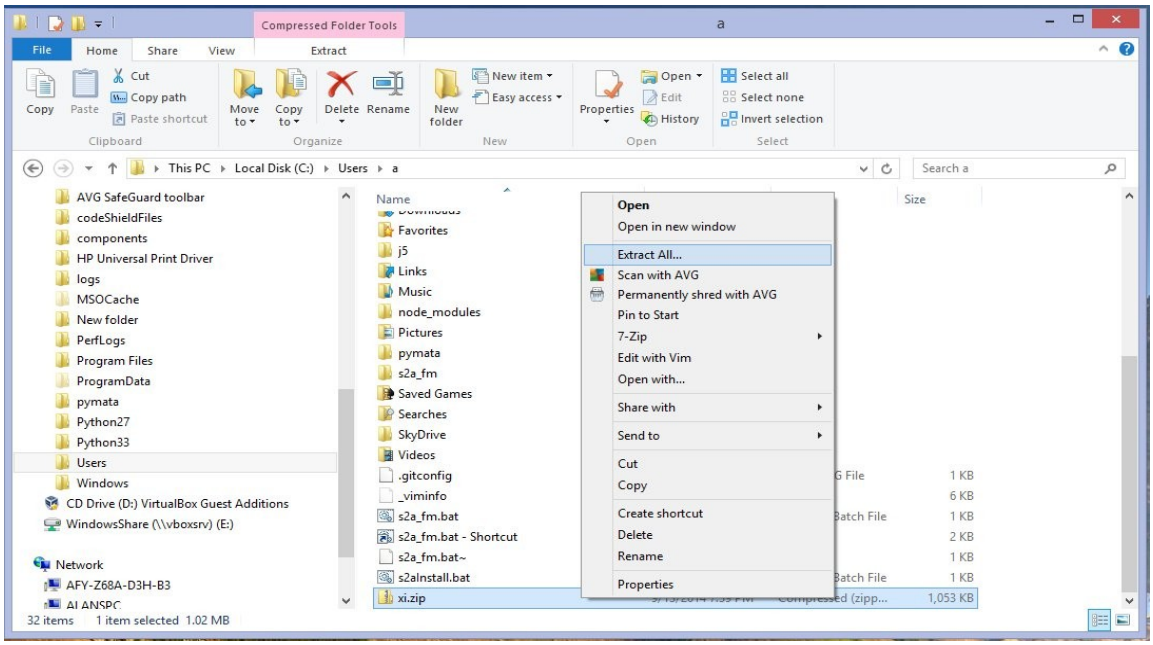
2.1.1.2.2 Installing node.js

To install node.js, go to this web page: http://nodejs.org/ and click on the green install button:



2.1.1.2.3 Download the Xi Distribution

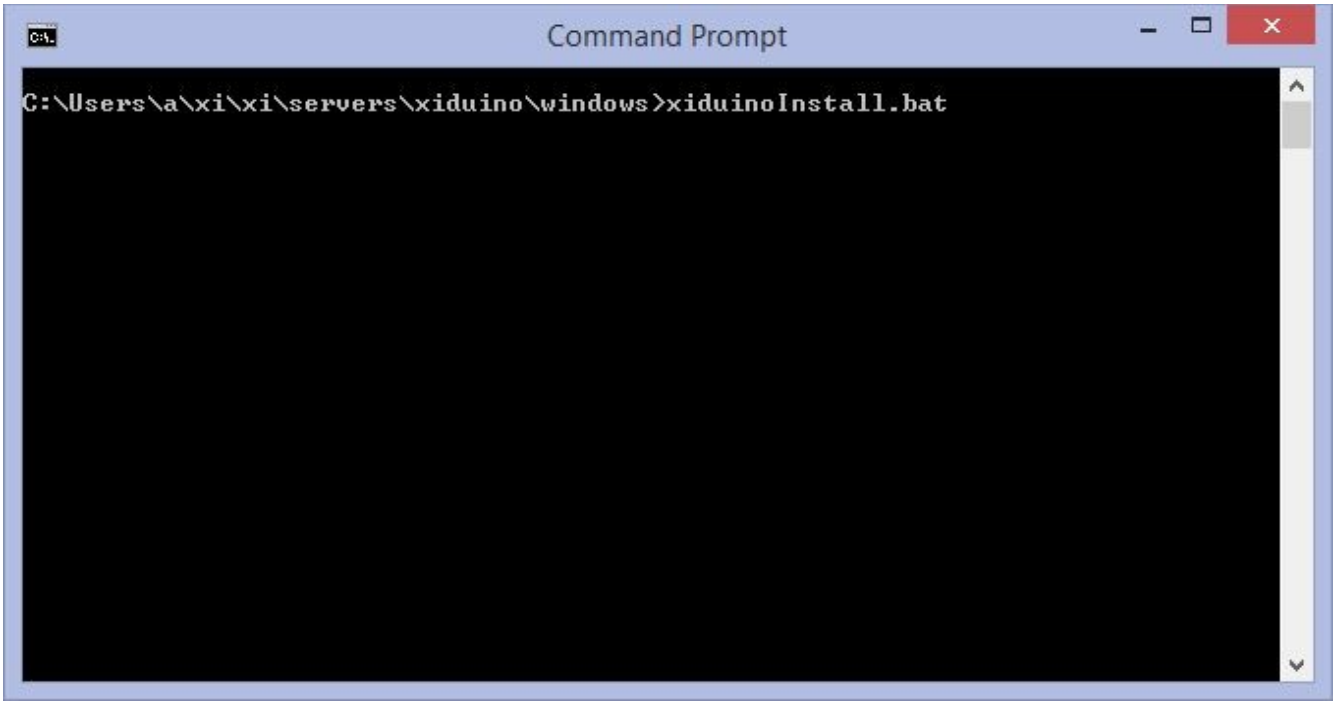
Download the Xi distribution from github at https://github.com/MrYsLab/xi and extract the files from the Xi archive file using the Windows Explorer.



2.1.1.2.4 Run the Install .bat Script

Open a Command window and navigate to the xi\servers\xiduino\windows directory.

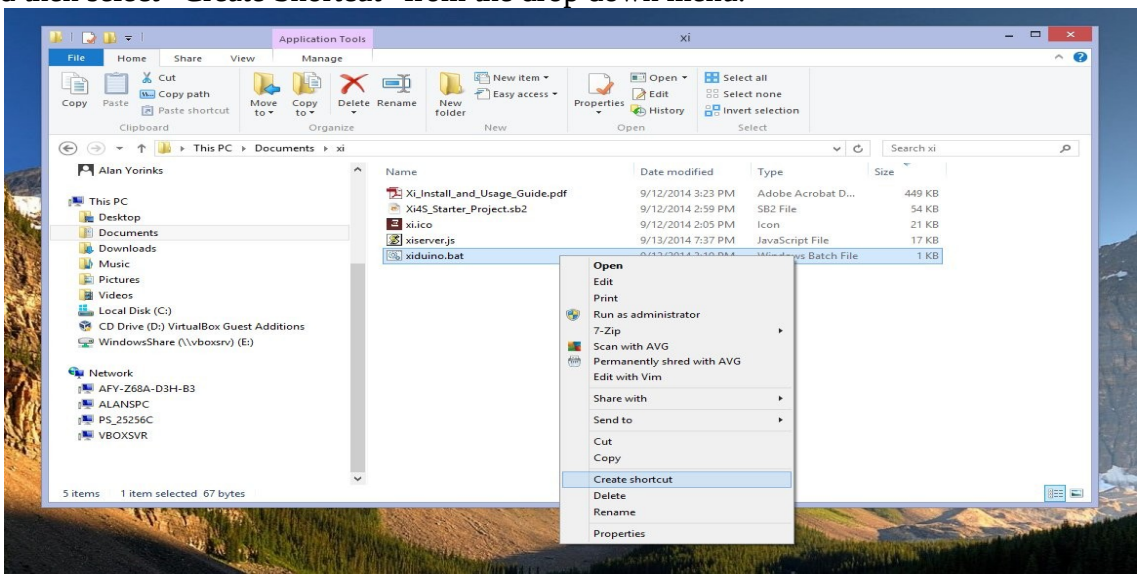
Type xiduinoInstall.bat and press return to begin the installation. This will install the files needed to support both Scratch and Snap!.



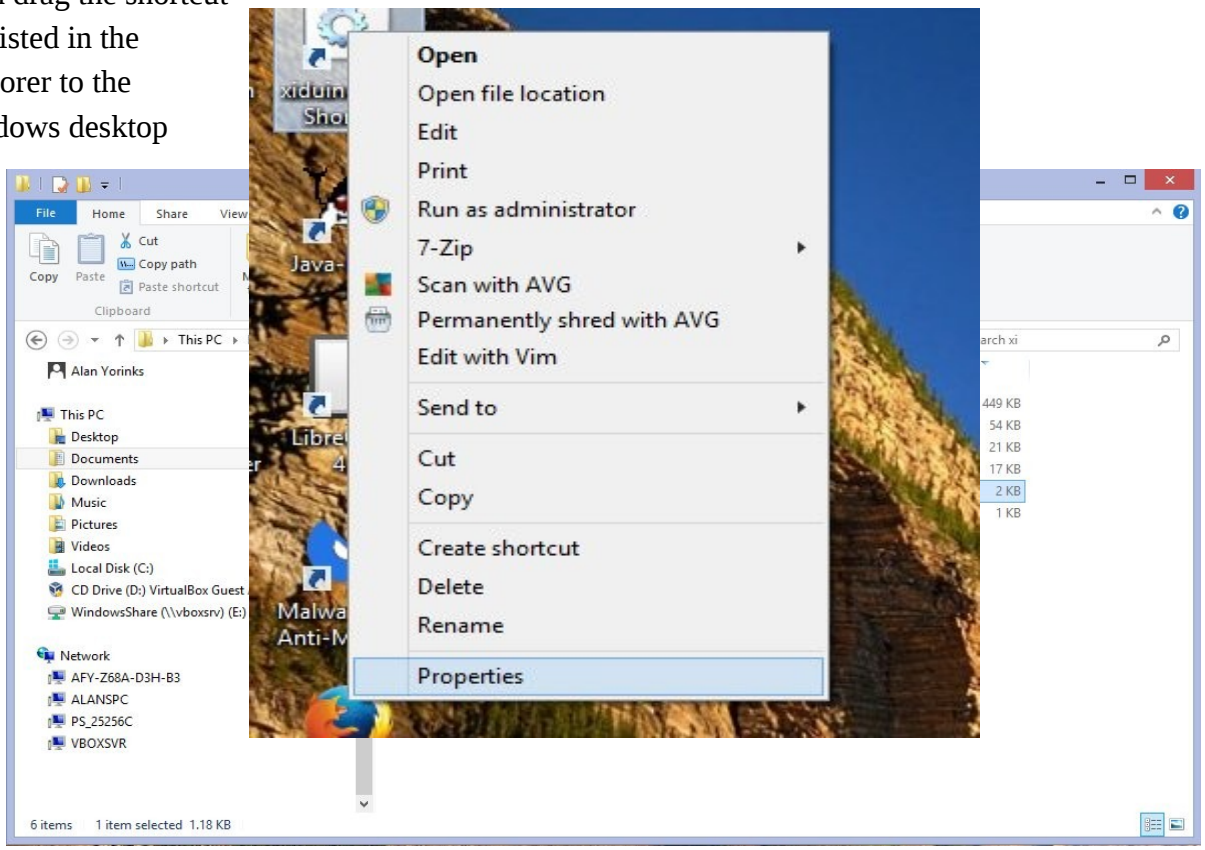
This will take several minutes to complete.

2.1.1.2.5 Creating a Desktop Launch Icon

After the installation scripts completes, open the Windows Explorer and navigate to your home Documents directory, and then select the xi directory. Right click on xiduino.bat or xiduino4snap.bat and then select “Create Shortcut” from the drop down menu.

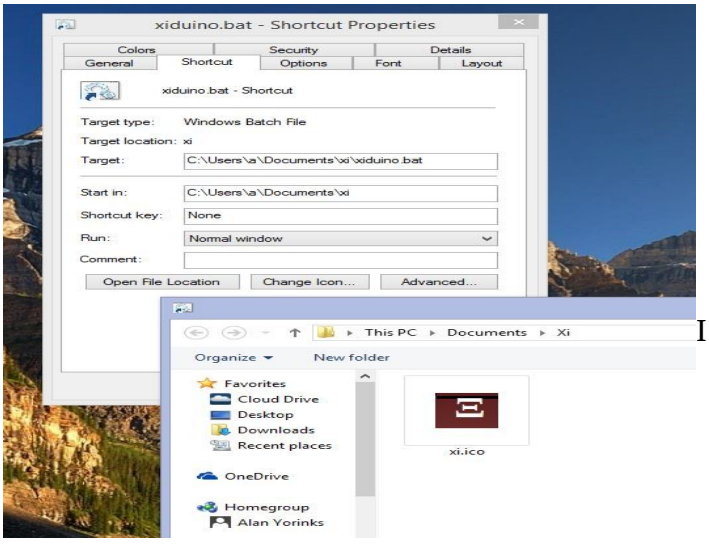


Then drag the shortcut file listed in the Explorer to the Windows desktop



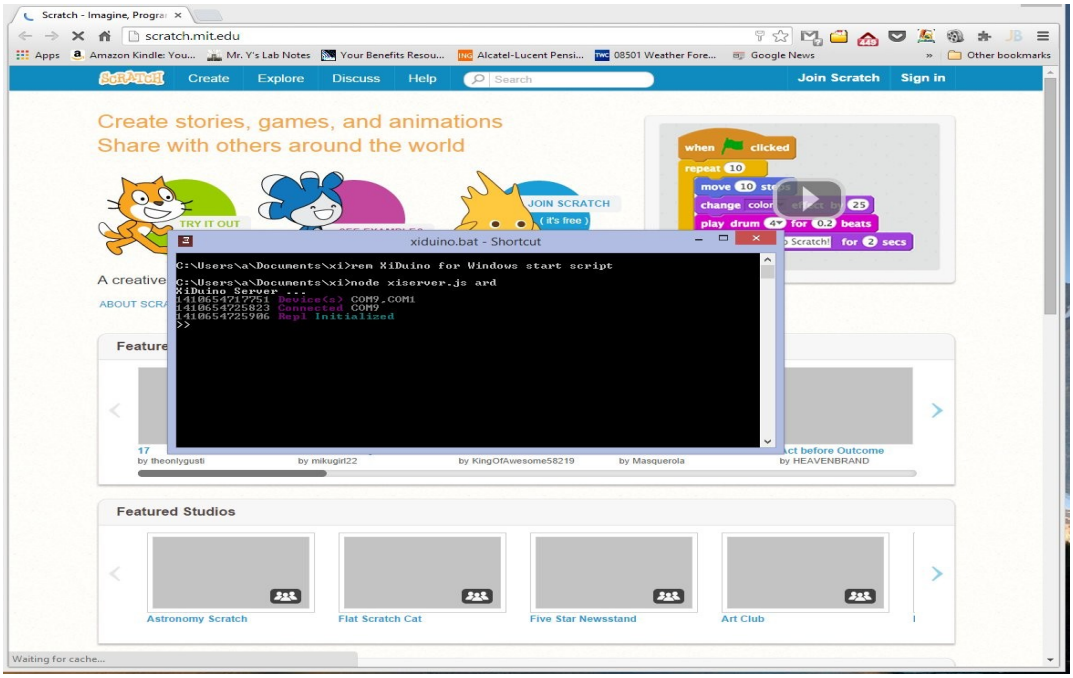
Right click on the default “gears” icon and select Properties, and then select Change Icon …

Go through the menus until you find the xi.ico file and select and apply it to the shortcut.



If you like, right click on the icon again and rename the icon title to XiDuino.

After you are done, when you double click the Xi icon, a command window will open up and the default browser will be launched. Remember to change your default browser to either Chrome or Firefox.



2.1.2. XiBone

XiBone was tested using the Debian 2014-05-15 eMMC image.

To install, it is recommended that you login as a user other than root. Make sure that your BeagleBone Black is connected to the Internet.

2.1.2.1 Installing The Required Development Tools

All supporting packages are already pre-installed.

2.1.2.2 Installing Node.js

Node.js is pre-installed for the BeagleBone Black, so it does not need to be installed.

2.1.2.3 Download the Xi Distribution

Download the Xi distribution from github at https://github.com/MrYsLab/xi and extract the files in your home directory.

2.1.2.4 Run the Install Shell Script

Make sure that your BeagleBone Black is connected to the internet before continuing. Both the Installation and execution of XiBone requires that the date be set to the current date. The installation and execution shell scripts contain a call to ntpdate to make sure that the date is current.

1. In a command terminal, CD to the servers/xibone directory
2. Change the file permissions for xiboneInstall.sh

```
chmod ugo+x xiboneInstall.sh
```

5. Execute the install script by typing:

```
./xiboneInstall.sh
```

6. The installation will take several minutes. If you see some warnings, you can ignore them.

The installation script creates a directory called xibone that is located below the home directory. This directory is populated with all the files you will need to run the XiBone server.

2.1.3. XiPi

XiPi was tested using RASPBIAN Debian Wheezy, version Linux raspberrypi 3.12.22+ #691 PREEMPT Wed Jun 18 18:29:58 BST 2014 armv6l GNU/Linux.

To install XiPi, it is recommended that you login as a user other than root. Make sure that your Raspberry Pi is connected to the Internet.

2.1.3.1 Installing The Required Development Tools

All supporting packages are already pre-installed.

2.1.3.2 Installing Node.js

To install node.js, open an LXTerminal and type in the following commands:

```
sudo apt-get update

sudo apt-get upgrade

sudo wget http://node-arm.herokuapp.com/node_latest_armhf.deb

sudo dpkg -i node_latest_armhf.deb
```

To verify that the package has been installed properly, and to see the node's version type:

```
node -v
```

2.1.3.3 Download The Xi Distribution

Download the Xi distribution from github at https://github.com/MrYsLab/xi and extract the files in your home directory.

2.1.3.4 Run The Shell Install Script

1. In a command terminal, CD to the servers/xipi directory
2. Change the file permissions for xipiInstall.sh

```
chmod ugo+x xipiInstall.sh
```

Execute the install script by typing:

```
./xipiInstall.sh
```

5. The installation will take several minutes. If you see some warnings, you can ignore them.

The installation script creates a directory called xipi that is located below the home directory. This directory is populated with all the files you need to run the XiPi server.

3. Launching a XiServer

3.1. Using the startup scripts

The startup scripts that were installed during system installation are usually sufficient for launching a XiServer.

After a XiServer is launched, you may stop the server by pressing CTRL-C twice.

3.1.1. Launching a XiDuino Server

3.1.1.1 Windows

If you created a desktop shortcut during the installation process, just click on the icon to start XiDuino.

If you want to launch from a command window, go to the xi directory that was created in Documents and for Scratch type:

```
xiduino.bat
```

or for Snap! Type:

```
xiduino4snap.bat
```

After a few moments, you command window should contain something like the following:

```
Starting XiDuino server...
XiDuino Server ...
1410474902988 Device(s) /dev/ttyACM0
1410474908009 Connected /dev/ttyACM0
1410474908009 Repl Initialized
```

3.1.1.2 Linux

To start XiDuino for Scratch, type the following commands:

```
cd  ~/xiduino
bash xiduino.sh
```

or for Xiduino for Snap!, type the following commands:

```
cd  ~/xiduino4snap.sh
bash xiduino.sh
```

A command terminal window will open and your default browser will be launched to the Scratch home page.

After a few moments, you command window should contain something like the following:

```
Starting XiDuino server...
XiDuino Server ...
1410474902988 Device(s) /dev/ttyACM0
1410474908009 Connected /dev/ttyACM0
1410474908009 Repl Initialized
```

3.1.2. Launching a XiBone Server

To start the XiBone server, type the following commands:

```
cd  ~/xibone
bash xibone.sh
```

After a few moments, you command window should contain something like the following:

```
Starting XiBone server...
XiBone Server ...
1410474902988 Device(s) /dev/ttyACM0
```

```
1410474908009 Connected /dev/ttyACM0
1410474908009 Repl Initialized
```

3.1.3. Launching a XiPi Server

To start the XiPi server type the following commands:

```
cd  ~/xipi
bash xipi.sh
```

After a few moments, you command window should contain something like the following:

```
Starting XiPi server...
XiPi Server ...
1410474902988 Device(s) /dev/ttyACM0
1410474908009 Connected /dev/ttyACM0
1410474908009 Repl Initialized
```

3.2. Custom Launch Using Xiserver Command Line Options

A single JavaScript file, xiserver.js, is used to instantiate all Xi board specific servers. And all server instances are configured through command line options when they are launched.

The usage line for xiserver.js is as follows:

```
node xiserver.js [boardType] [arduinoURL] [debugLevel] [arduinoPort] [ipPort]
```

All options are in square brackets and they are positionally dependent. This means that if you wish to include an option, all options to the left of that option must be manually specified.

3.2.1. boardType

The boardType option specifies the target hardware platform.

Board Type	Option Specifier
Arduino	ard
BeagleBone Black	bbb
Raspberry Pi	rpi

3.2.1.1 Default boardType

The default boardType is ard.

3.2.2. arduinoURL

If the board type selected is Arduino, when the XiDuino server is started, it will automatically launche the default web browser to the URL specified by this option.

3.2.2.1 Default URLs

When using the startup script, xiduino.sh or xiduino.sh, the URL is set to http://scratch.mit.edu.

When using the startup script xiduino4snap.sh or xiduino4snap.bat, the URL is set to http://snap.berkeley.edu/snapsource/snap.html, which is the Snap! on-line project editor.

3.2.2.2 Suppressing Browser Launch

When launching a XiDuino server, If you do not wish to automatically launch a browser, use ***null*** as the URL option. This is useful when more than one Arduino will be connected to Scratch or Snap!.

3.2.3. debugLevel

This option sets the amount of debug information sent to the launching command window. For the least amount of debug information set debugLevel to 0. For the most information, set the option to 3.

3.2.3.1 Default debugLevel

The default level is set to 0.

3.2.4. arduinoPort

When using a XiDuino server, you may optionally command XiDuino to manually select a specific COM port. For example /dev/ttyACM0 or COM3.

3.2.4.1 Default arduinoPort

There is no default setting for this option. Xi auto-detects available com ports.

3.2.5. ipPort

The ipPort option allows you to select the IP port used in connecting to a Xi Server. When connecting to a server from either the Scratch Snap! Xi clients, make sure that you specify the ipPort you selected in the Xi server block.

3.2.5.1 Default ipPort

The default value is 1234.

4. Using the Xi Clients

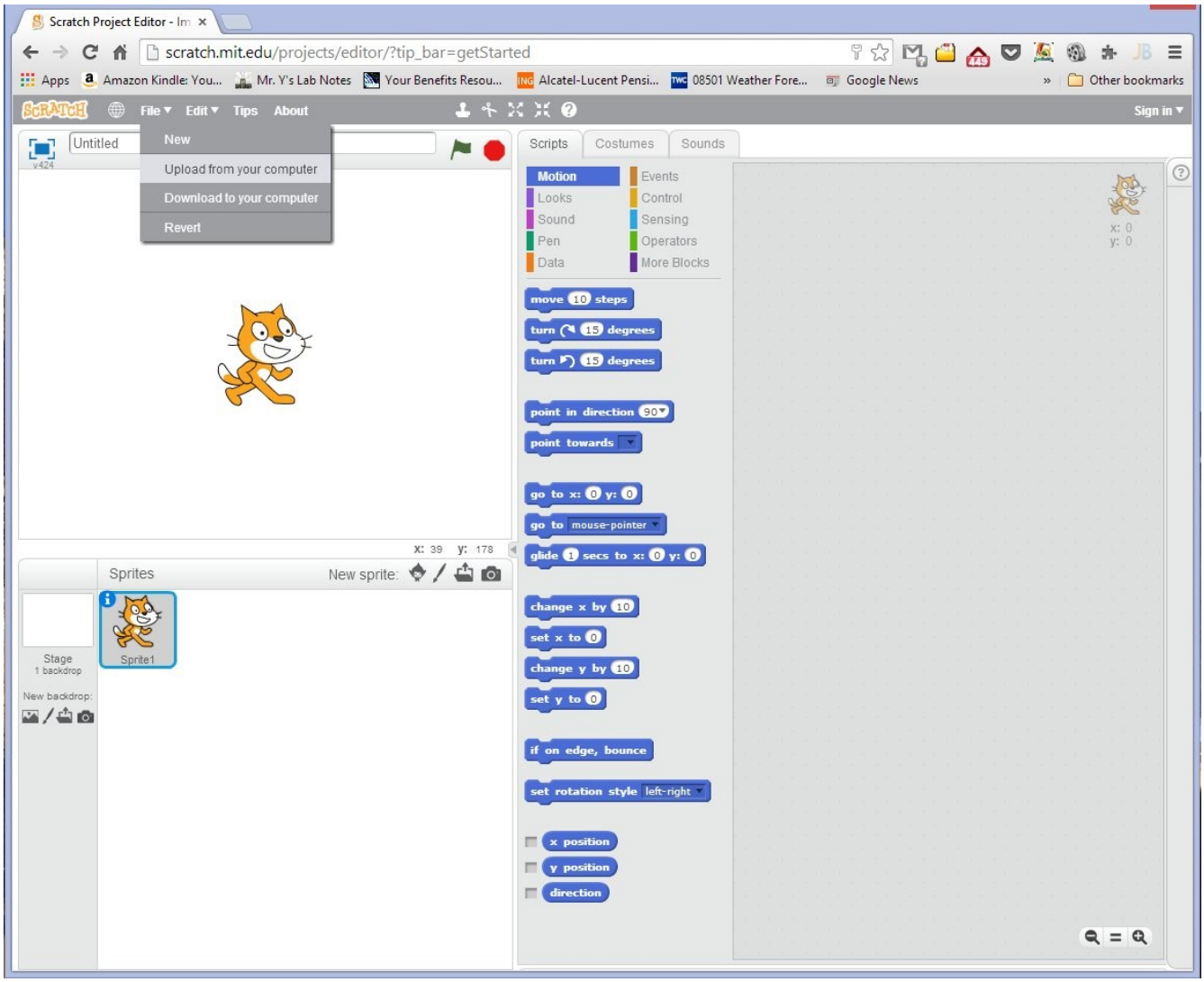
4.1. The Scratch Xi4S client

The Scratch extension client, Xi4S, is automatically loaded into Scratch when an existing project that contains Xi4S is uploaded to the editor. Currently, this is the only option provided by the Scratch team to install the extension.

A starter project, containing the extension and extension blocks, is included with this distribution. It is located in the archived file you extracted earlier. It is located in the xi/clients/scratch/projects directory and the file is called Xi4S_Starter_Project.sb2.

To install it, open the Scratch on-line editor, click on File/Upload from your computer and select Xi4S_Starter_Project.sb2. After uploading the starter project, you probably will want to rename it and save it under a name that is meaningful to you.

You will not be able to share a project containing Xi4S, again a limitation imposed by the Scratch team, but you can save the program to your computer and make a copy of the file if you wish to share it.



The source code for the extension is included in this package and is located in xi/clients/scratch/xi4s and the file is name xi4s.js. It is provided for those interested in gaining an understanding of how Xi4S implements the Scratch JavaScript Hardware Extension Specification.

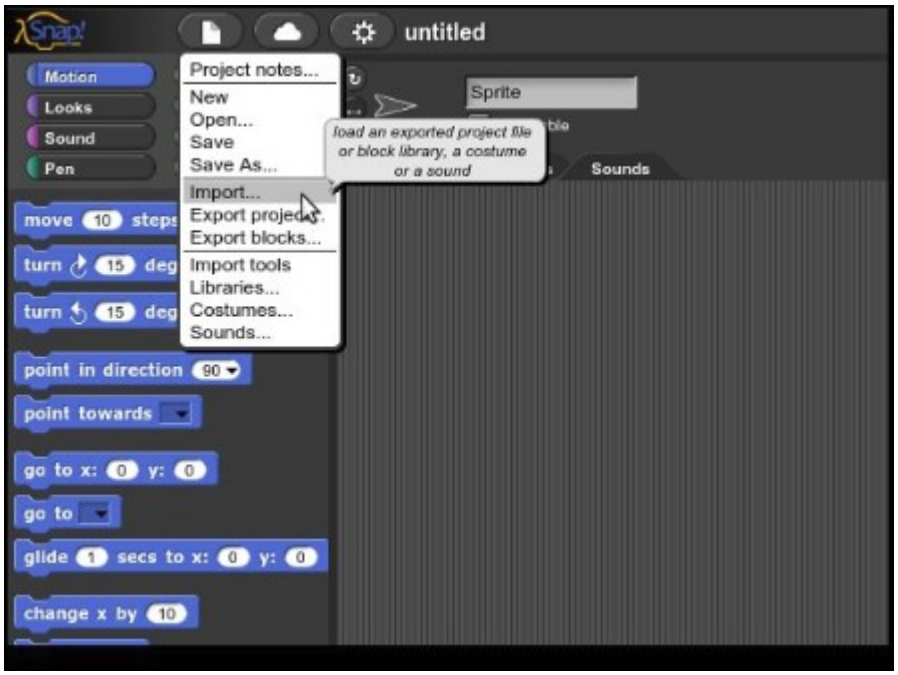
IMPORTANT NOTE: Scratch may prompt you to load a "plugin" when using Xi. **DO NOT LOAD THIS PLUGIN WHILE USING XI.** It is unnecessary to do so, and may in fact interfere with Xi's Arduino support.

4.2. The Xi4Snap! Client

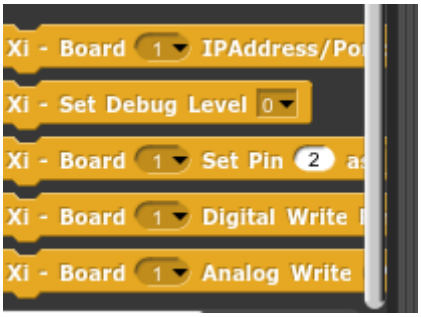
Each custom Snap! block that was created for Xi4Snap contains its own JavaScript extension code.

To load the Xi extension block into Snap!, click the file icon in the upper left and select import. Then

go to the directory where you extracted the Xi download. Go to the clients/snap!/projects Directory and select the xi4snap .xml file. The version number of the file is part of its name.



After the blocks load, select the Control tab in the blocks palette and at the bottom of the list of blocks, you should see the Xi control blocks, and in the Sensing tab, you should the Xi Sensing blocks:



5. Using the Xi4S and the Xi4Snap! Blocks

5.1. Pin Designations

5.1.1. Arduino

The standard pin numbering is used. For Analog input pins, you just type in the pin number and ***not*** the A prefix. So if you want Pin A6 to be an analog input pin, the pin number would be 6 (***not A6***).

5.1.2. BeagleBone Black

The BeagleBone Black uses a mapping of its pins to an Arduino Uno numbering scheme. The mapping matrix is below. I will be investigating if this mapping can be expanded in the future.

BBB Port	Arduino Pin	Type
P8_7	0	Digital
P8_8	1	Digital
P8_9	2	Digital
P8_13	3	PWM
P8_10	4	Digital
P9_14	5	PWM
P8_16	6	PWM
P8_11	7	Digital
P8_12	8	Digital
P9_21	9	PWM
P9_22	10	PWM - Currently not working
P9_28	11	PWM - Currently not working
P8_14	12	Digital
USR3	13	Digital / Default Led
P9_39	A0	Analog Input
P9_40	A1	Analog Input
P9_37	A2	Analog Input
P9_38	A3	Analog Input
P9_35	A4	Analog Input
P9_36	A5	Analog Input

5.1.3. Raspberry Pi

Pin numbers are identified by their pin number on the P1 header, so for example, if you want to use GPIO 17, specify pin 11.

5.2. Command Blocks

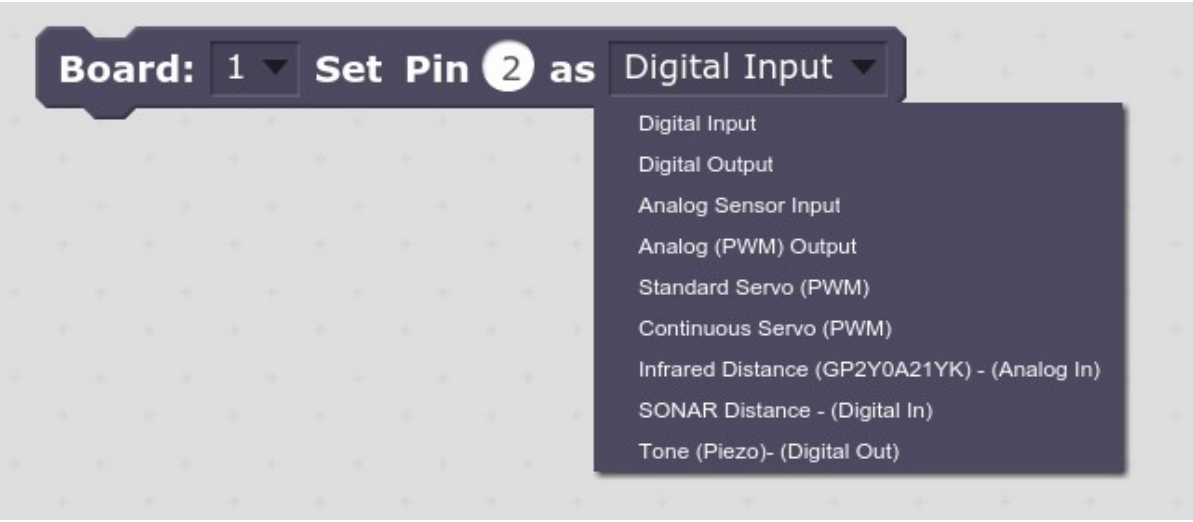
5.2.1. Selecting an IP Address and Port for the Connected Boards

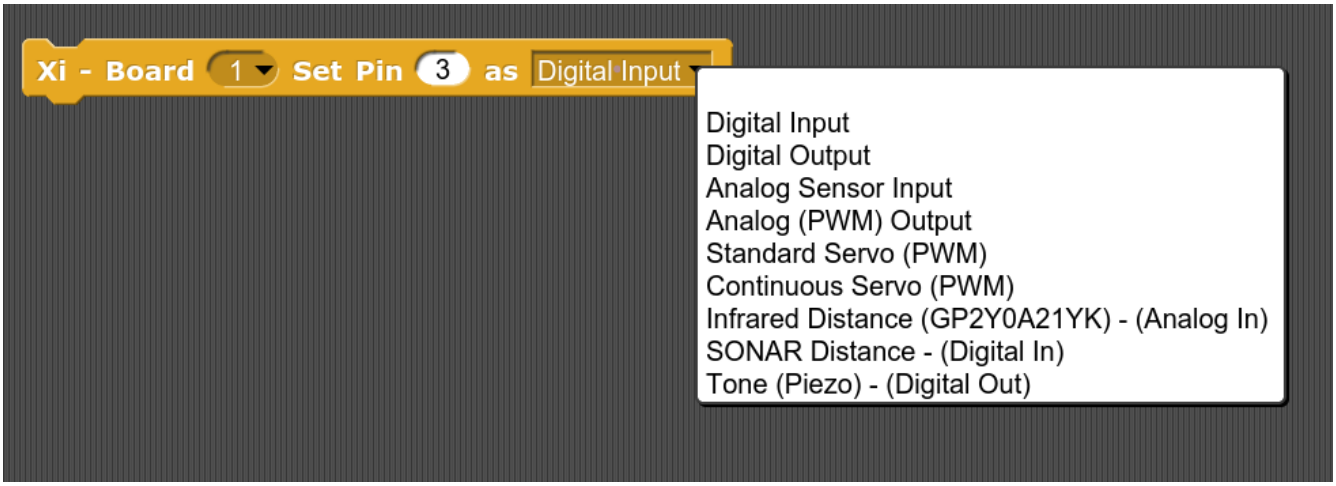
Before using any other blocks for a given board, the board's IP address and port number must be set using this block.

Boards are identified by their Board number and selected from the drop down list. Up to 10 boards can be supported. The *IPAddress* is the address of the board. If you are using Arduino boards, this address is typically “localhost”. For BeagleBone Black and Raspberry Pi, the address is set to the IP address of the board.

The port number, 1234, is the default port number for all of the servers. It typically should not be changed. If you need to change it, you will also need to change the port when invoking the Xi Server by using the appropriate command line option.

5.2.2. Setting the Pin Mode





This block is used to set a *pin* or *device mode* for a selected pin on a specified board.

NOTE: Stepper motors are specified using multiple pins, therefore, stepper motors are defined using their own pin assignment blocks. For Arduino, stepper motor support also requires a special version of Standard Firmata. See sections 2.1.1.1.2, 5.2.11 and 5.2.12 for details.

SONAR support also requires a special version of Standard Firmata (different from the version providing stepper motor support). See section 2.1.1.1.1 for details.

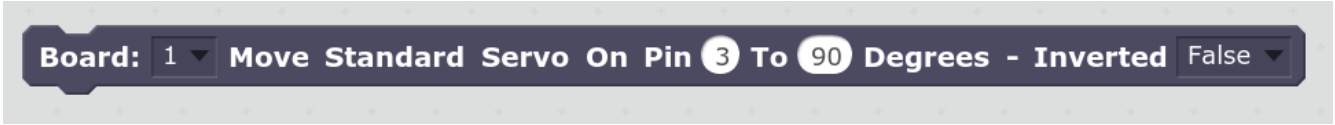
5.2.3. Digital Write

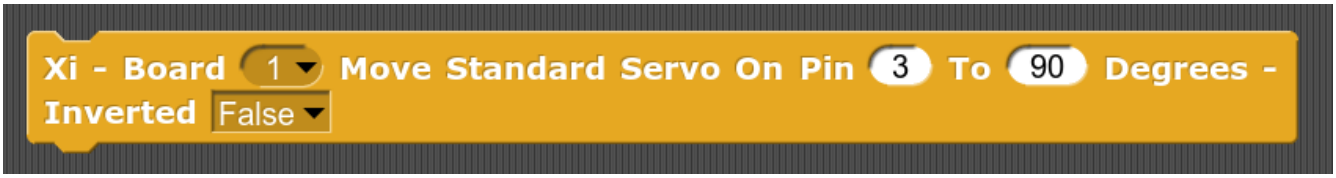
Use this block to set a digital output pin to an “On” or “Off” state.

5.2.4. Analog Write (PWM)

Use this block to set a PWM output pin to a value between 0 and 255.

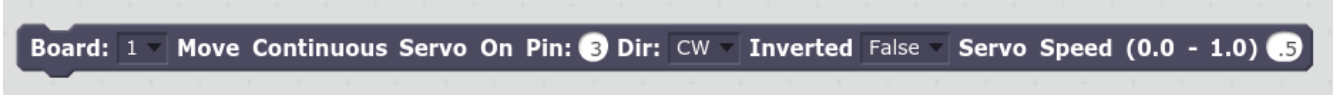
5.2.5. Move Standard Servo

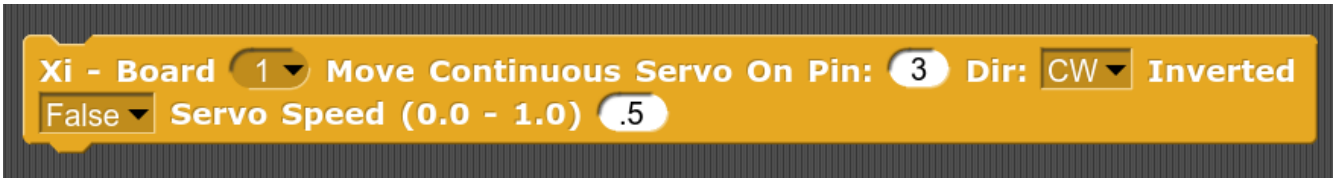




Use this block to set the angular position of a standard servo. The angular range is from 0 to 180 degrees. The *"Inverted"* selector allows you to set 0 degrees to either extreme of the range of motion.

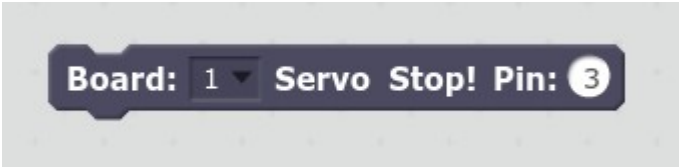
5.2.6. Move Continuous Servo





Use this block to set a continuous servo in motion. You can set the direction, clockwise or counterclockwise, and the speed (from 0.0 to 1.0). If the motor is turning in the opposite direction of what you expect, you can reverse the direction using the *Inverted* field. This allows the direction of the motor to be in sync with the block designator.

5.2.7. Stop Servo



This block allows you to stop the motion of either a standard or continuous servo.

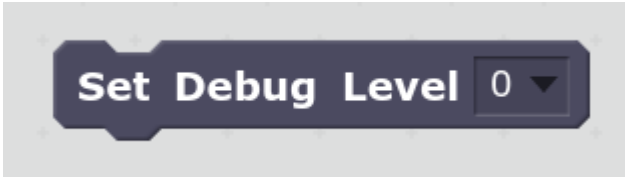
5.2.8. Play Tone

This block will play a tone on a Piezo device at the specified frequency for the specified duration.

5.2.9. Turn Tone Off

This block will turn tone off.

5.2.10. Set Debug Level



This block controls the debug level and the amount of debug statements sent to the browser's JavaScript console.

A value of 0 is the quietest mode and 2 is the most verbose.

5.2.11. Set Pins For 4 Wire Bipolar Stepper Motor

For Arduino boards, this block requires an experimental sketch to be loaded onto the Arduino board., replacing StandardFirmata.

This sketch will support all modes except SONAR.

Go to https://github.com/soundanalogous/AdvancedFirmata/ to obtain the sketch.

This block specifies the 4 motor pins and the number of steps per revolution for the motor.

5.2.12. Set Pins for Stepper Driver Board

For Arduino boards, this block requires an experimental sketch to be loaded onto the Arduino board., replacing StandardFirmata.

This sketch will support all modes except SONAR.

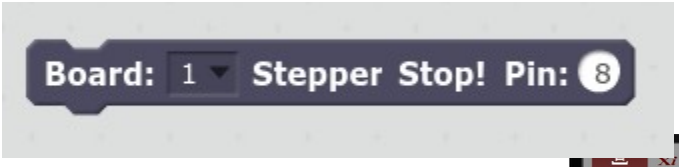
Go to https://github.com/soundanalogous/AdvancedFirmata/ to obtain the sketch.

This block specifies the step pin and motor direction pin when using a stepper driver board such as the Polulu A4988 Stepper Motor Driver.

5.2.13. Move Stepper Motor

This block is used to set either a 4 wire or driver configured stepper motor in motion. The pin number is the first pin specified when the stepper motor pins were defined. This blocks set the motor RPM, Direction, Acceleration, Deceleration in addition to the number of steps to move.

5.2.14. Stop Stepper Motor



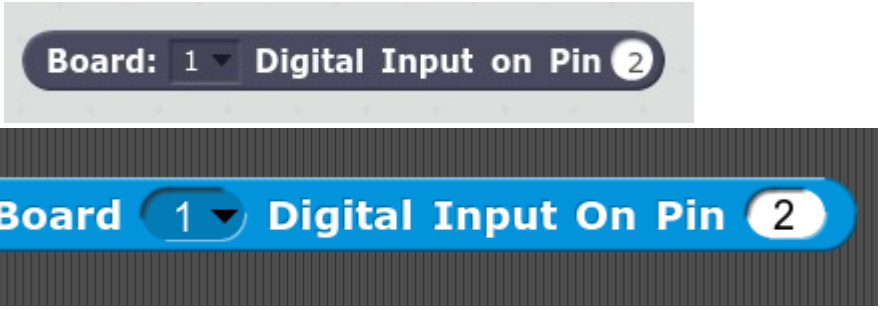
This block will stop a stepper motor specified by the first pin used when the stepper motor pins were defined.

5.3. Reporter Blocks

These blocks retrieve the latest data for the board and pin selected. Data is fetched from a cache located in the in the browser extension. There is no communication between the client and the servers when these blocks are executed.

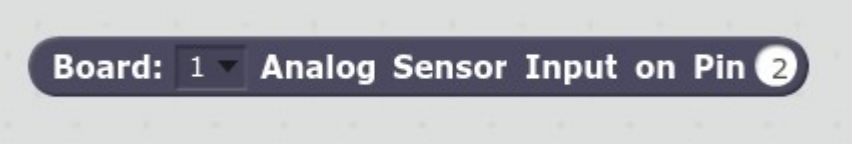
The cache is updated when a server detects a change in input value. The server notifies the client of the change, and the client updates its cache.

5.3.1. Digital Input



This block will report a 0 or 1 for the state of a digital input pin.

5.3.2. Analog Sensor Input

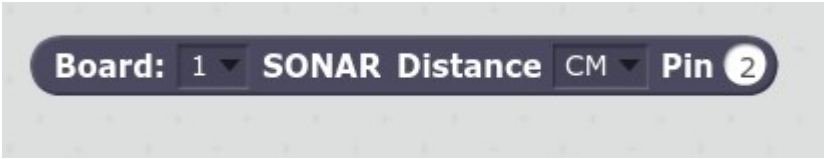


This block reports a value of 0 – 1023 for an analog input pin.

5.3.3. Infrared Distance

This block reports the distance measured using an infrared distance sensor. The unit of measurement may be selected to be in centimeters or inches.

5.3.4. SONAR Distance



This block reports the distance measured using a SONAR distance sensor. The unit of measurement may be selected to be in centimeters or inches.

6. Some Usage Tips

6.1. Keeping Track of Boards While Programming

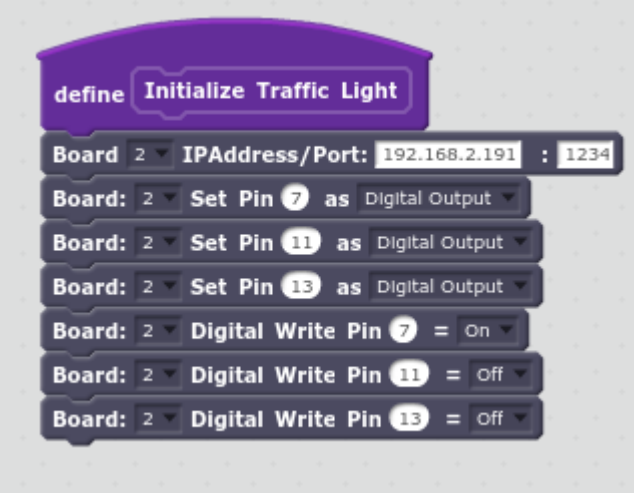
Keeping track of multiple boards when creating a Scratch project can easily lead to confusion. To keep things simple, I suggest creating a separate Sprite for each board connected to Scratch. By naming each Sprite as `BoardName-BoardNumber`, you can easily distinguish between the boards. So for example, if you connect a Raspberry Pi as board #1 and an Arduino as board #2, the Sprite names for these boards might be RPI-1 and ARD-2.

6.2. Create Your Own Custom Blocks for a Higher Level Abstraction of the Boards and Their Connected Devices

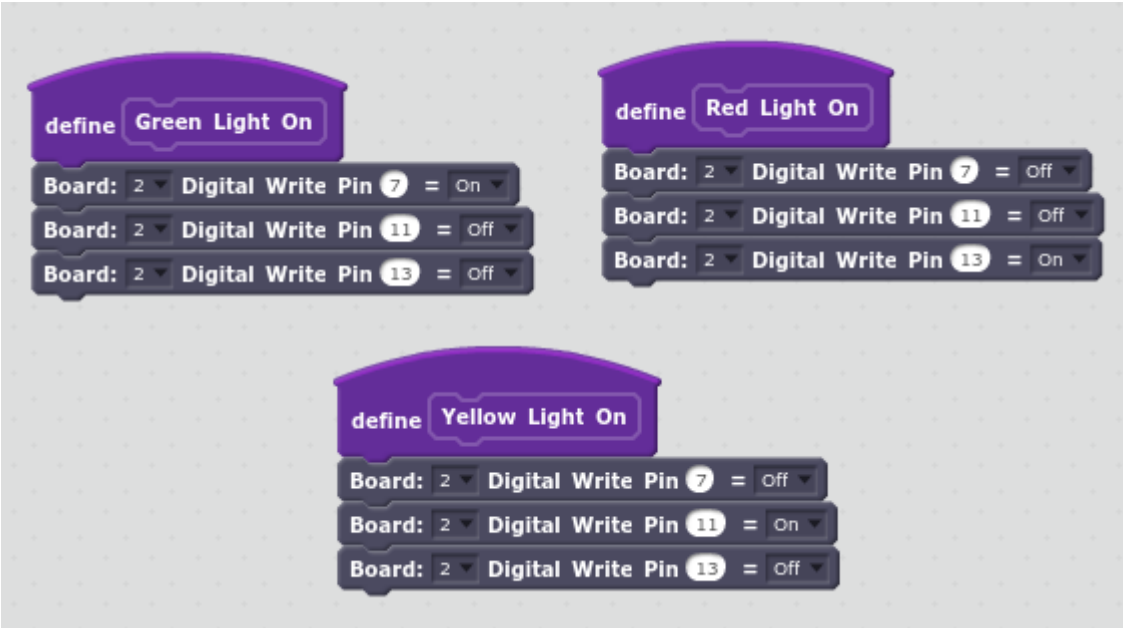
The client block sets were purposefully designed to represent boards at the pin level. This is appropriate for many applications, but may not be desirable in all cases.

For example, you may wish to have a group of students control a “traffic light” that is assembled using a green, yellow and red LED connected to a Raspberry Pi. By creating custom blocks you the students will have a targeted view of the hardware without them having to know the low level details.

In the screen shot below, a block was assembled to create and initialize a “traffic light”. It sets up all the pins for the LEDs and turns the Green LED on.



Additional blocks were assembled to turn on a single light while turning off the other two.



A short script was written to cycle through changing of the traffic signals.

Providing this level of abstraction hides the hardware details and allows users to more readily solve the problem at hand.

In addition, by employing this technique, Xi may be used in conjunction with commercial products that have sensors and actuators that are hardwired to specific pins. By creating custom blocks for these devices, the user can enjoy a plug and play experience, while still having the opportunity to get down to the pin level if so desired.

7. Getting Help

If you have any questions, suggestions, or encounter any problems, please don't hesitate to contact me at:

MisterYsLab@gmail.com

Alan Yorinks