s2a_fm

A Scratch 2.0 and Snap! 4.0 Hardware Extension for Arduino Micro-Controllers

January 1, 2013
s2a_fm Version 1.2

Version 1.2 Changes:

1. **Support for Snap! 4.0 has been added.**
2. **Support for HC-SR04 type Ping devices has been added.**
3. **Translation of Scratch Block Text into Dutch has been provided.**
4. **Facility to translate Scratch/Snap Block Text to any language is included.**

Version 1.1 Changes:

1. **When enabling a digital pin, the pin is verified to support the requested mode.**

2. **Added debugging feature**

Table of Contents

1.Introduction....1
2.What is s2a_fm?....1
3.Installing s2a_fm....2
  3.1.Python and Python Files....2
  3.2.Arduino Sketch....2
4.Running s2a_fm....3
  4.1.Starting a Scratch Project....3
  4.2.Starting a Snap! Project....4
5.The Extension Blocks....4
  5.1.Set Digital Pin Mode....5
    5.1.1.Enable....5
    5.1.2.Disable....5
    5.1.3.Pin Number....5
    5.1.4.Digital Mode....6
  5.2.Set Analog Pin Input Mode....6
    5.2.1.Enable....6
    5.2.2.Disable....7
    5.2.3.Pin Number....7
  5.3.Digital Write....7
    5.3.1.Pin Number....7
    5.3.2.Pin Output Value....7
  5.4.Analog (PWM) Write....7
    5.4.1.Pin Number....7
    5.4.2.PWM Value....8
  5.5.Play Tone....8
    5.5.1.Pin Number....8
    5.5.2.Frequency (HZ)....8
    5.5.3.Duration (ms)....8
  5.6.Turn Tone Off....8
    5.6.1.Pin Number....8
  5.7.Move Servo....9
    5.7.1.Pin Number....9
    5.7.2.Position (Deg)....9
  5.8.Read Digital Pin....9
  5.9.Read Analog Pin....9
  5.10.Debugger....10
  5.11.The Red Stop Button....10
6.Log File....10
7.Example Programs....11
  7.1.A Scratch/Snap! Program to Turn On an LED using Digital Pin 6....11
  7.2. Scratch/Snap Program to Read A Potentiometer on Pin A2 and Spin the Scratch Cat....11
8.Scratch/Snap! Block Text Translation....12
  8.1.Translating to Other Languages....12
9.Project Directory Tree....12
10.References....14
11.Questions, Comments, Bug Reports....14

1. Introduction

Arduino users! Would you like to configure and control your Arduino micro-controller without having to write a single line of Arduino sketch code and at the same time have access to a graphical user interface?

Scratch and Snap! programmers! Would you like to control and communicate with an Arduino board? Imagine, using Scratch or Snap! to control physical devices such as LEDs, motors, and relays while monitoring devices, such as temperature sensors, potentiometers, and light sensors. What would you create?

Look no further, **s2a_fm** is here!

2. What is s2a_fm?

s2a_fm is a Scratch hardware extension written in Python allowing Scratch and an Arduino micro-controller to communicate seamlessly.

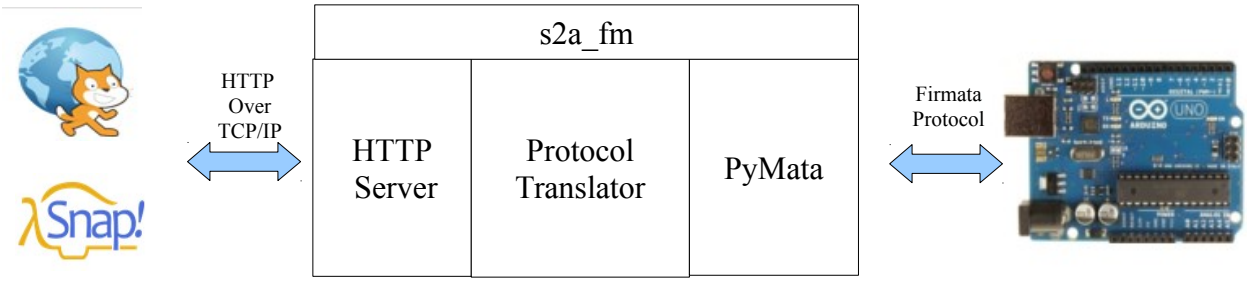


Figure 1 – s2a_fm Architecture

s2a_fm is comprised of three main software components:

1. An HTTP server that communicates with Scratch. The standard Python BaseHTTPServer is used for simplicity and compatibility.
2. The protocol translator translates data between the HTTP and Firmata protocols. This is all done seamlessly and it allows your Scratch/Snap!/Arduino projects to look and work just like any other Scratch or Snap! project.
3. PyMata**,** a Python library that communicates with the Arduino using the Arduino Standard Firmata protocol. This makes for fast, efficient and consistent communications. PyMata handles all of the Firmata protocol details.

To start a new Scratch project, just upload the *s2a_fm_base.sb2* script file supplied with this release. To start a new Snap! project, import s2a_fm_Snap_base.xml. These files contain all of the extension blocks ready to go. After loading the project for your platform, you will find the Scratch extension blocks in the More Blocks tab of the Scratch Project editor ready for use. For Snap! The blocks may be found in the Motion, Sound and Sensor tabs of the blocks palette.

3. Installing s2a_fm

3.1. Python and Python Files

Before installing s2a_fm, you must have the following Python components installed on your computer:

Python version 2.7 or greater. You can download Python from: (http://python.org/)

PySerial (http://pyserial.sourceforge.net/)

PyMata (https://github.com/MrYsLab/PyMata)

To install PySerial and PyMata, you may use the "pip' installation program (http://www.pip-installer.org/en/latest/) if you have it installed on your computer. To use it, open a command window and type: pip install *package_name.*

Alternatively, you can download the packages using the links above, and for both the PySerial and PyMata packages type:

python setup.py install.

Note: you may need administrative privileges to perform the install.

3.2. Arduino Sketch

Load Standard Firmata into the Arduino from the examples directory of the Firmata library included with the Arduino IDE.

If you wish to use the Tone and/or SONAR (ping) functionality provided in S2a_fm, you should load the FirmataPlus sketch included with this distribution. See the Project Directory Map (section 9) for its location.

Make sure that you are using PyMata 1.54 or later with this sketch.

You will need to add the NewPing library to Arduino to successfully compile this sketch. It may be downloaded at: https://code.google.com/p/arduino-new-ping/downloads/detail?name=NewPing_v1.5.zip

NOTE: After installing NewPing, you will need to modify one of its files. Follow the directions provided in this link:
https://code.google.com/p/arduino-new-ping/wiki/HELP_Error_Vector_7_When_Compiling

NotSoStandardFirmata has been replaced by FirmataPlus, but is still provided as part of the distribution for those that wish to use it.

It is recommended that you use Arduino IDE version 1.5.5 or greater. Using an earlier version my result in compile errors.

4. Running s2a_fm

Plug your Arduino into a USB port on your computer.

Go to the directory where you installed the **s2a_fm** package and type:

```
python s2a_fm.py COM_PORT_ID
```

where *COM_PORT_ID* is the name of the serial port that is used to communicate with the Arduino. It will have the same name as the one used when you use the Arduino IDE. For example, for Windows, it might be COM3, for linux, it might be /dev/ttyACM0.

After executing the command, you should see something like the following appear in the terminal window:

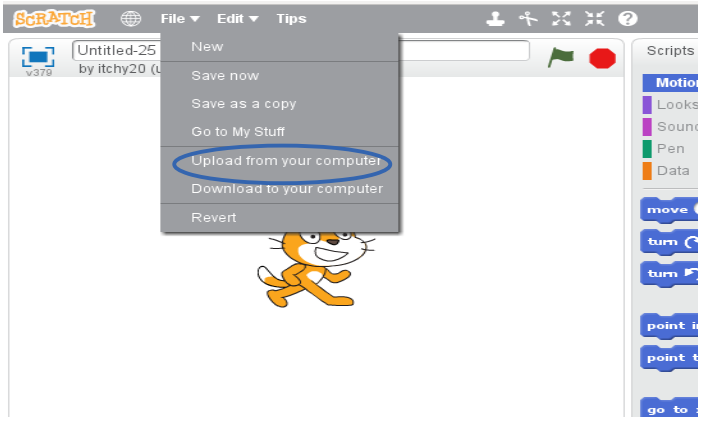
```
s2a_fm version 1.2   Copyright(C) 2013-2014 Alan Yorinks    All Rights Reserved

Opening Arduino Serial port /dev/ttyACM0
Please wait while Arduino is being detected. This can take up to 30 seconds ...
Board initialized in 1 seconds
Total Number of Pins Detected = 20
Total Number of Analog Pins Detected = 6
Please wait for Total Arduino Pin Discovery to complete. This can take up to 30
additional seconds.
Arduino Total Pin Discovery completed in 1 seconds
Starting HTTP Server!
Use <Ctrl-C> to exit the extension

Please start Scratch or Snap!
```

4.1. Starting a Scratch Project

Now start Scratch 2.0. Create a new project and select File/Upload from your computer and select s2a_fm_base.sb2 from the ScratchFiles/ScratchProjects directory included with this distribution.

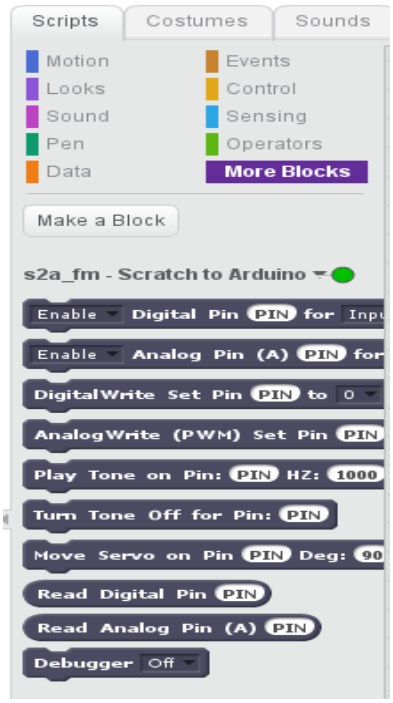


When Scratch is detected the following line is printed to the console:

```
Scratch detected! Ready to rock and roll...
```

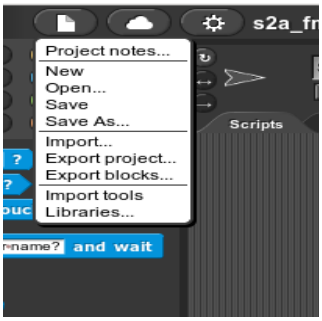
And if you go to the More Blocks tab in Scratch you should see:

The circular indicator next to " s2a_fm – Scratch to Arduino" should be green, indicating that the connection has been successfully established.



4.2. Starting a Snap! Project

After Snap! is started, go to the File menu and select Import.

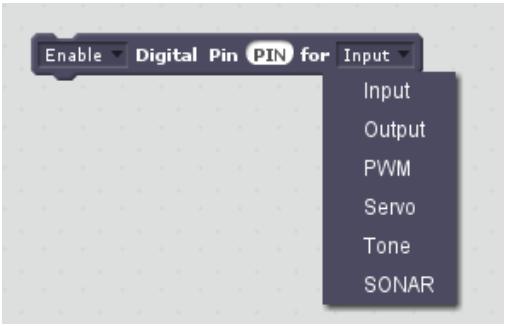


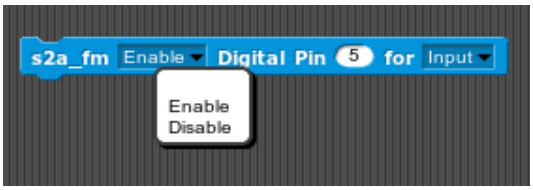
Go to the Snap!Files directory and select the "s2a_fm_Snap_base.xml" file. This is a blank project that will load all the s2a_fm blocks.

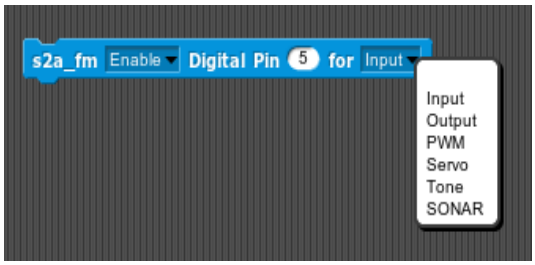
5. The Extension Blocks

The blocks in black and white are the Scratch command and reporter blocks. The blocks in color are the Snap! Blocks. In Snap!, all of the "s2a_fm" blocks start with s2a_fm as part of the identifier string.

5.1. Set Digital Pin Mode







For Snap! this block is located on the Sensing tab of the block palette.

This block enables or disables an Arduino digital pin as an Input, Output, PWM, Servo or Tone or SONAR.

5.1.1. Enable

Before accessing a pin for input or output it must be enabled. To enable, select Enable from the first drop down menu. If **Input** mode is chosen (see below), enabling the pin will automatically instruct the Arduino board to report value changes for the pin.

5.1.2. Disable

If a pin has been previously been enabled, selecting disable will inactivate the pin from its previously enabled mode and reporting will cease for an I**nput** mode pin. Normally this choice is not used, but is provided to give full flexibility in writing Scratch scripts.

5.1.3. Pin Number

The value entered must be within the range of pin numbers for the board. The number of pins detected for the Arduino is shown in the console window when the program first starts up and this information is also placed in the log file (see section 6). If the pin number is outside of the range for the board, an error message is sent to the console and an error is logged. The command is ignored if an error is

detected.

5.1.4. Digital Mode

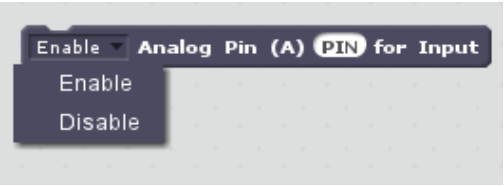
There are 6 digital modes supported.

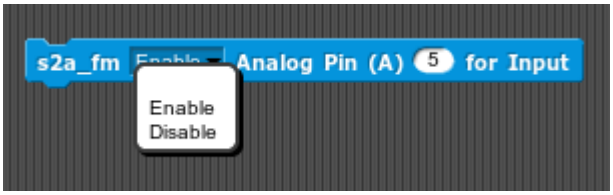
- **Input** for connection to an input device such as a switch.
- **Output** for connection to a device such as an LED.
- **PWM** is an output mode used to control a pin with an AnalogWrite. This can be used to fade the light level on an LED.
- **Servo** configures the pin to operate a servo motor.
- **Tone** configures the pin to call on the Tone library through Firmata.
- **SONAR** configures the pin to work with the NewPing library. Up to a maximum of 6 devices can be monitored simultaneously.

NOTE: The HC-SR04 type device is configured to work in single pin mode (trigger and echo connected together). See the NewPing documentation for details: https://code.google.com/p/arduino-new-ping/wiki/NewPing_Single_Pin_Sketch.

SPECIAL NOTE: Some Arduino boards provide internal pull-up resistors To enable pull-up mode for a pin, refer to the Arduino documentation for instruction.

5.2. Set Analog Pin Input Mode





Analog pins are always input pins and can only be enabled or disabled. They are fixed as inputs.

For Snap! this block is located on the Sensing tab of the block palette.

5.2.1. Enable

Before accessing an analog pin for input it must be enabled. To enable, select Enable from the first drop down menu. Enabling the pin will automatically instruct the Arduino board to report changes in values for the pin.

5.2.2. Disable

If a pin has been previously enabled, selecting disable will inactivate the pin from its previously enabled mode. Data will no longer be reported for the pin while it is disabled.

5.2.3. Pin Number

The value entered must be within the range of analog pin numbers for the board. The number of pins detected for the board is shown in the console window when the program first starts up. If the pin number entered is outside of the range of the board, an error message is sent to the console and the error is logged. If an error is detected, the command is ignored. The pin number usse the Arduino analog pin number scheme. For example to enable analog pin A3, set the pin number in the block to 3.

5.3. Digital Write

For Snap! this block is located on the Sensing tab of the block palette.

5.3.1. Pin Number

The pin number must be set to a pin that was previously enabled as Output and is in the range of digital pin values. If the pin is not enabled as an Output pin or if it is out of range, the request is ignored and an error message is written to the console and the log file.

5.3.2. Pin Output Value

If a digital pin has been enabled for output, you can set its output level to either a one or a zero. Select the value from the drop down list in the block.

5.4. Analog (PWM) Write

For

Snap! this block is located on the Sensing tab of the block pallet.

5.4.1. Pin Number

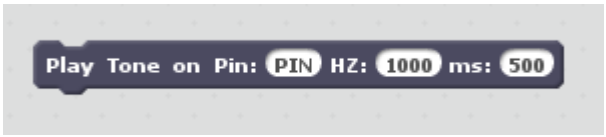
The pin number must be set to a pin that was previously enabled as PWM and is a PWM digital pin. If the pin is not enabled as a PWM pin or if it is out of range, the request is ignored and an error message

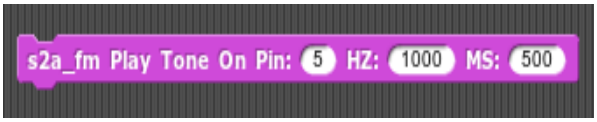
is written to the console and the log file.

5.4.2. PWM Value

Set the pin output value to be between 0-255. If the value is out of range an error message is written to the console and the log file. The request will be ignored.

5.5. Play Tone





This block instructs the Arduino to play a tone on the designated pin.

For Snap! this block is located on the block palette Sound tab.

5.5.1. Pin Number

The pin number must be for a pin that was previously enabled for Tone and is in the range of digital pin numbers. If the pin is not enabled as a Tone pin or if it is out of range, the request is ignored and an error message is written to the console and the log file.

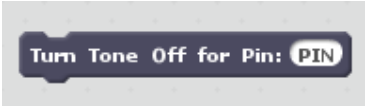
5.5.2. Frequency (HZ)

The tone will be played at the specified frequency. There is no data validation for this entry.

5.5.3. Duration (ms)

The number of milliseconds to sustain the tone. If this value set to zero, the tone will be played indefinitely. It may be turned of my using the Turn Tone Off block.

5.6. Turn Tone Off



This block will turn the tone off. It is used primarily if the duration for Play Tone was set to zero (a continuous tone).

For Snap! this block is located on the block palette Sound tab.

5.6.1. Pin Number

The pin number must be for a pin that was previously enabled for Tone. If the pin is not enabled as a Tone pin or if it is out of range, the request is ignored and an error message is written to the console

and the log file.

5.7. Move Servo

This block will set a servo position to the desired value.

For Snap! this block is located on the block palette Motion tab.

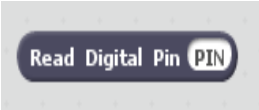
5.7.1. Pin Number

The pin number must be for a pin that was previously enabled for Servo and is in the range of digital pin values. If the pin is not enabled as a Servo pin or if it is out of range, the request is ignored and an error message is written to the console and the log file.

5.7.2. Position (Deg)

This sets the motor position expressed in degrees. The range is 0 to 180. A value outside of this will be ignored and an error message is written to the console and the log file.

5.8. Read Digital Pin



This block is a “reporter” block for a pin that is enabled as a digital input. It is used to gain access to the current value for the pin number specified. It returns either a zero or a one.

For Snap! this block is located on the block palette Sensing tab.

5.9. Read Analog Pin

This block is a “reporter” block for a pin that is enabled as an analog input. It will contain the current value for the pin number specified in the range of 0-1023. Enter the Arduino analog pin number in the PIN field. For example to read analog pin A3, set the pin number in the block to 3.

For Snap! this block is located on the block palette Sensing tab.

5.10. Debugger

Turn debugging On or Off (the default is Off) from the drop down list. If debugging is enabled all Scratch commands are logged to both the log and console. Each command is time stamped with the local time. Reporter blocks (sensor information) do not appear as part of the debug report because Scratch polls for data approximately 30 times per second and this would flood the log.

For Snap! this block is located on the block palette Sensing tab.

Here is a sample of the debug log:

```
DEBUG: 2013-12-19 14:19:06.733833: debugger On
DEBUG: 2013-12-19 14:19:06.768232: digital_pin_mode Enable 48 Output
DEBUG: 2013-12-19 14:19:06.800449: digital_pin_mode Enable 51 Output
DEBUG: 2013-12-19 14:19:06.849955: digital_write 48 1
DEBUG: 2013-12-19 14:19:06.872217: digital_write 51 1
DEBUG: 2013-12-19 14:19:06.906062: digital_write 48 0
```

5.11. The Red Stop Button

Pressing the red stop button on the Scratch player will send a reset command to the Arduino and will reset all internal data structures to their initial values.

6. Log File

A new log file is created each time s2a_fm is started, and the previous log file is discarded. The name of the log file is“s2a_fm_debugging.log” and is located in a directory called “log” (see the Project Directory Tree in section 9). The log file contains “info” records which are considered part of normal operation and it may contain debug records which will help in debugging a Scratch program accessing an Ardunio board.

Here is part of a typical log:

```
INFO:root:s2a_fm version 1.2  Copyright(C) 2013-2014 Alan Yorinks  All Rights
Reserved
INFO:root:com port = /dev/ttyACM0
INFO:root:20 Total Pins and 6 Analog Pins Found
INFO:root:Scratch detected! Ready to rock and roll...
DEBUG:root:digital_pin_mode: The pin number must be set to a numerical value
DEBUG:root:analog_write: The value field must be set to a numerical value
```

In this example, the INFO records indicate normal behavior and provide informational data, and the DEBUG records indicate a problem. In the first DEBUG line, the pin number was not entered properly

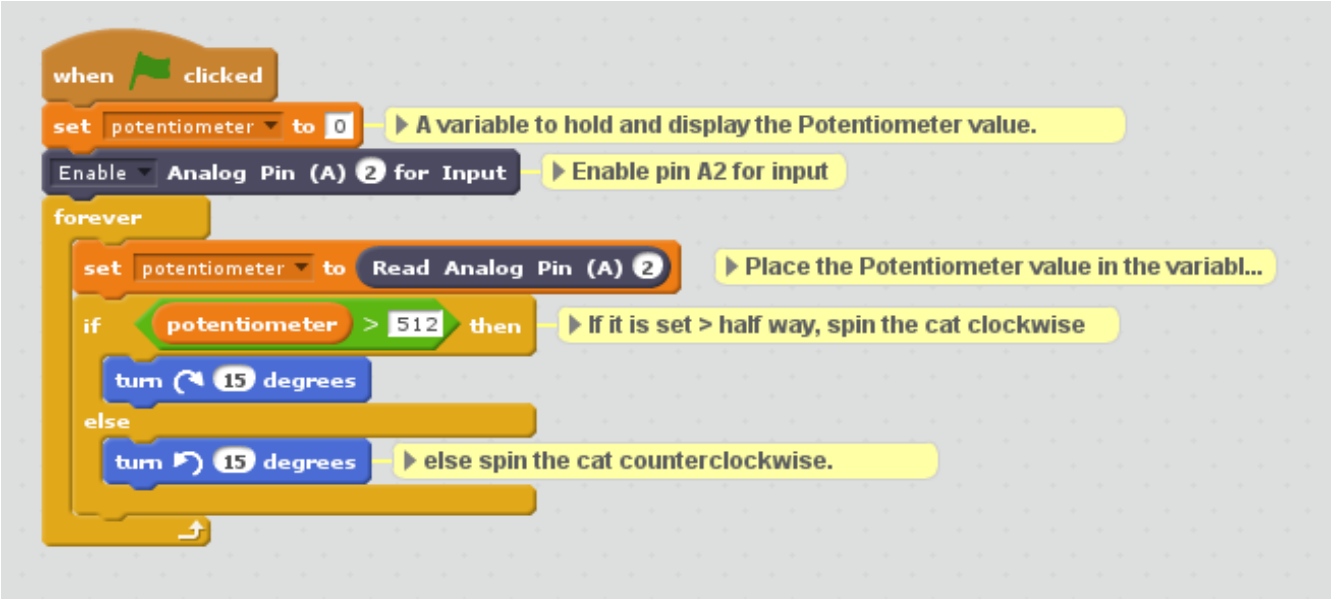
when setting a digital pin mode, and the second DEBUG indicates that the VAL field needs to be properly entered when executing an analog write block.

7. Example Programs

These Scratch .sb2 program files for these examples can be found in ScratchFiles/ScratchProjects directory and the Snap!Files/Snap!Projects directory (see the Project Directory Tree in section 9). Note: the screenshots below show the Scratch blocks.

7.1. A Scratch/Snap! Program to Turn On an LED using Digital Pin 6

7.2. Scratch/Snap Program to Read A Potentiometer on Pin A2 and Spin the Scratch Cat



8. Scratch/Snap! Block Text Translation

Thanks to Sjoerd Dirk Meijer, a Dutch translation has been provided. A blank Scratch project, called s2a_fm_base_nl.sb2 is contained in the ScratchFiles/ScratchProjects directory.

In addition, Sjoerd Dirk has kindly provided a JSON extension specification for Scratch also translated to Dutch. This file is called s2a_fm_NL.s2e and can be found in the ScratchFiles/ExtensionDescriptors directory.

For Snap!, just load the "s2a_fm_Snap_base_dutch.xml" file as described in section 4.2.

8.1. Translating to Other Languages.

If you wish to translate the text to an additional language, two files will need to be modified. The first file is the extensions descriptor file. The other file is called xlate.cfg.

To understand what needs to change in the extension descriptor file, compare the English and Dutch versions provided in this distribution to identify the strings that need to be changed. Then you must add the translation to the xlate.cfg file. For each new keyword, add an entry for your additions. Make sure that there is no space after the comma.

Example:

```
ln_INPUT = Input,ingang,MY_NEW_TRANSLATION
```

Once both files have been modified, load the extension descriptor file into Scratch by selecting holding the Shift Key and selecting File from the Scratch Menu (this only functions with the offline version of Scratch) and then select "Import Experimental Extension" and if the file was modified appropriately, the translated blocks should appear in the More Blocks Section.

If nothing appears, check your modifications in the a JSON lint checker (http://jsonlint.com/) and make any corrections that are necessary.

9. Project Directory Tree

```
s2a_fm
├── ArduinoFiles
│   ├── FirmataPlus
│   │   ├── examples
│   │   │   └── FirmataPlus
│   │   │       ├── FirmataPlus.ino
│   │   │       ├── LICENSE.txt
│   │   │       └── Makefile
│   │   ├── Boards.h
│   │   ├── Firmata.cpp
│   │   ├── Firmata.h
│   │   ├── keywords.txt
```

```
│   │   ├── LICENSE.txt
│   │   └── readme.md
│   └── NotSoStandardFirmata
│       ├── examples
│       │   └── NotSoStandardFirmata
│       │       ├── LICENSE.txt
│       │       ├── Makefile
│       │       └── NotSoStandardFirmata.ino
│       ├── Boards.h
│       ├── Firmata.cpp
│       ├── Firmata.h
│       ├── keywords.txt
│       ├── LICENSE.txt
│       └── readme.md
├── documentation
│   ├── LED_EXAMPLE.png
│   ├── pot1.png
│   ├── s2a_fm_reference.pdf
│   ├── scratch_blocks.png
│   └── snap_blocks.png
├── extra_goodies
│   ├── linux
│   │   └── s2a_fm.sh
│   └── windows
│       └── s2a_fm.bat
├── log
├── ScratchFiles
│   ├── ExtensionDescriptors
│   │   ├── s2a_fm_NL.s2e
│   │   └── s2a_fm.s2e
│   └── ScratchProjects
│       ├── s2a_fm_base_nl.sb2
│       ├── s2a_fm_base.sb2
│       ├── sonarTest.sb2
│       ├── spinning_cat.sb2
│       └── Turn On LED On Pin 6.sb2
├── Snap!Files
│   ├── s2a_fm_Snap_base_dutch.xml
│   ├── s2a_fm_Snap_base.xml
│   ├── spinning_sprite.xml
│   └── Turn On LED On Pin 6.xml
├── license.txt
├── README.md
├── s2a_fm.py
├── scratch_command_handlers.py
├── scratch_http_server.py
└── xlate.cfg
```

-

10. References

Arduino	http://arduino.cc/
Snap!	http://snap.berkeley.edu/
Scratch	http://scratch.mit.edu/
Arduino Standard Firmata	http://playground.arduino.cc/Interfacing/Firmata
PyMata	https://github.com/MrYsLab/PyMata
NewPing Arduino Library	https://code.google.com/p/arduino-new-ping/

11. Questions, Comments, Bug Reports

Please contact us at MisterYsLab@gmail.com