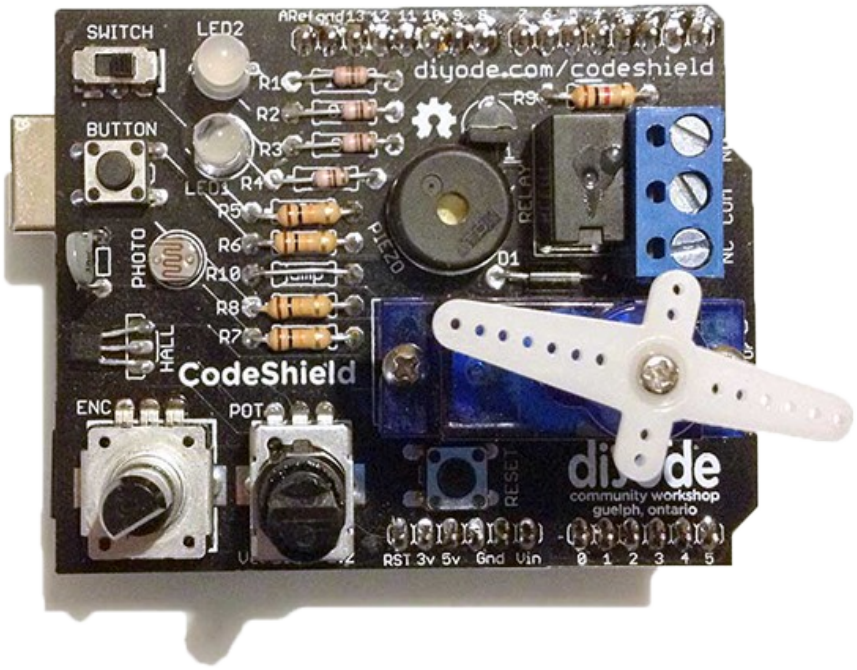
Scratch 2.0 Hardware Extension for CodeShield



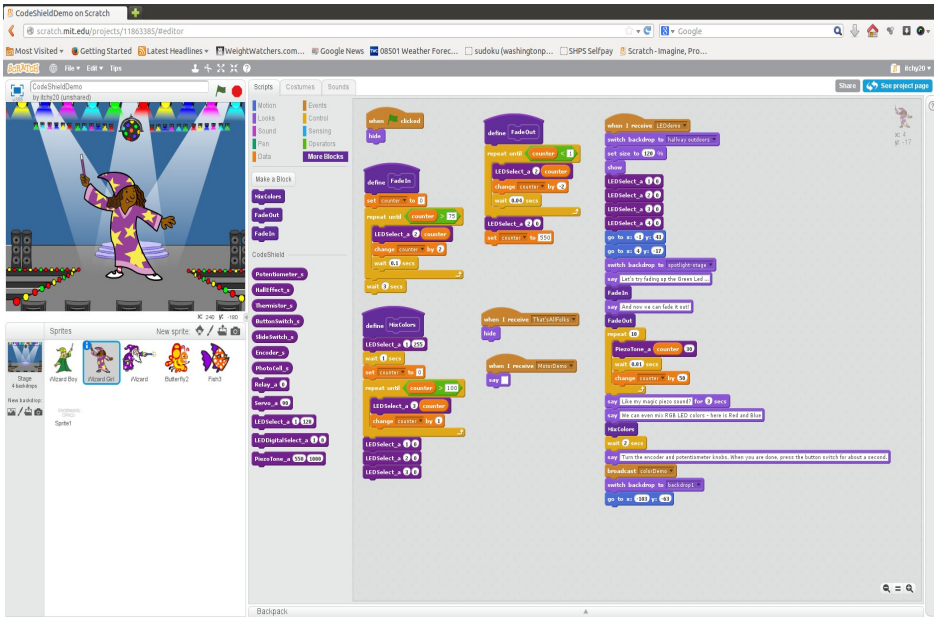
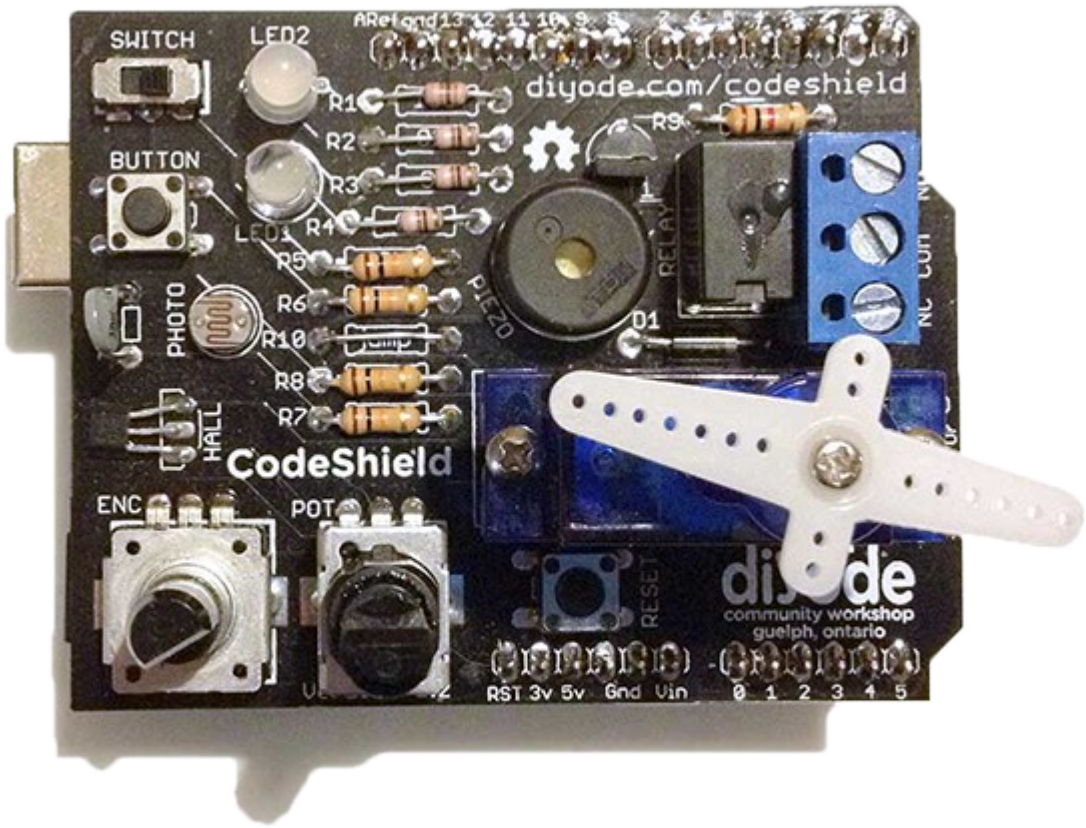


Table of Contents

Scratch 2.0 Hardware Extension for CodeShield

Alan Yorinks
August 16, 2013

CodeShield is a small printed circuit board containing a wide array of sensors and actuators. It plugs directly into an Arduino micro-controller board.



Scratch 2.0 is a programming environment provided by MIT that provides a fun and easy to use coding environment.

This package provides the necessary software to connect the CodeShield to Scratch 2.0 so that the CodeShield can be monitored and controlled by Scratch. CodeShield v.1.2 was used in developing and testing the extension. To see a video demonstration of Scratch and CodeShield in action, click on this link: http://www.youtube.com/watch?v=vdmDf9VvPUs.

Software Architecture

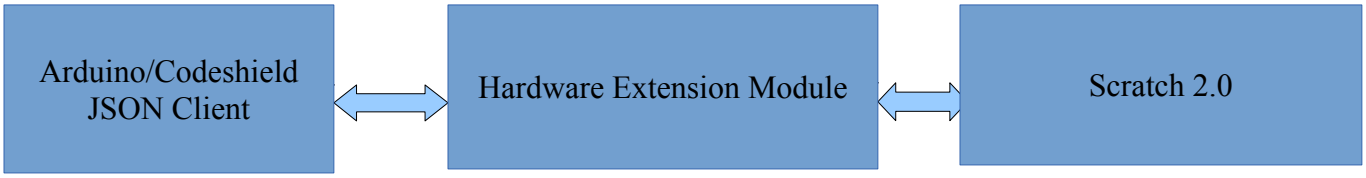


Figure 1 Software Architecture

The software consists of 3 main components:

- A customized JSON client that runs on the Arduino. The original client was written by Chris Warburton and was adapted for this project. It is small, lightweight and easy to customize.
- A Java based Hardware Extension Module (HEM) that runs on the PC. This module utilizes a stable and easy to use library for serial communication called JSSC. It also uses a JSON toolkit provided by json.org. Both libraries are included in the software distribution package as part of the runtime environment. Although JSON is being used by both sides of the HEM interface, each side uses a different JSON syntax structure. HEM provides the translation between these two flavors.
- Scratch 2.0 running on a PC browser.

Sensor Monitoring

Scratch, asynchronously and continuously sends JSON formatted poll requests for sensor data to the HEM via a TCP socket. The poll request is a generalized request and is not sensor specific. Upon receipt, the HEM translates this single poll requests into individual sensor status requests, each with its own JSON string. These individual requests are forwarded to the Arduino board over a USB/serial interface. The Arduino queries the CodeShield sensors and returns a status to the HEM. The HEM translates the replies into a Scratch specific syntax and returns the results to Scratch over the TCP/IP socket.

Actuator Control

To activate a device on the CodeShield board, Scratch sends an individual command to the HEM with user specified parameters. Upon receiving a command, HEM translates the command into an Arduino consumable format and forwards the command to the Arduino using the USB/serial interface. HEM waits for an acknowledgement from the Arduino and then goes on to process the next poll or command. If an acknowledgment is not received in a timely manner, HEM will throw a Java exception and exit.

Reuse of this Software for Other Scratch/Arduino Projects

The software was designed for easy reuse. The only Java file that needs to be modifed is MessageTranslator.java. All other Java files can be used as is. MessageTranslator.java contains all of the JSON to JSON translation as well as the IP port number pre-specified in the JSON code block definition file.

Hardware Requirements

1 Arduino UNO R3 (a Leonardo should work as well but has not been tested).

1 CodeShield plugged into the UNO.

1 PC. This code has been tested using Ubuntu 13.04 and Windows 8. It should run on a Mac but has not been tested with this configuration.

Software Requirements

In addition to the software provided in this package, you will need to have a Java 1.7 Runtime Environment or equivalent installed on your computer.

Software Included With This Distribution

Arduino Specific Files

CodeShieldJSONClient.ino – the Arduino JSON client with CodeShield customization.

Java Specific Files

The Hardware Extension Manager Java source files:

MessageManager.java

MessageTranslator.java

ScratchArduinoExtension.java

SerialManager.java

TCPServerManager.java

A runtime distribution package (.jar files) including the necessary libraries.

Scratch Specific Files

CodeShieldDemo.sb2 – a Scratch 2.0 demo program script

codeShield.json – a JSON code block definition file for Scratch

Why Not Just Share the Scratch Program on the Scratch Website?

The Scratch developers at MIT asked that programs containing customized hardware extensions not be shared until it is determined how best to manage and coordinate these specialized projects. I am

complying with this request, and would ask that you do as well. You can keep an unshared copy on the Scratch website if you wish.

Installation Instructions

Build the Arduino Module

The first step is to build and download the JSON client to the Arduino board. To compile the JSON client, you will need to add the Encoder library to the Arduino IDE. Use the standard Arduino method to add this library before compiling. Once the Encoder library is installed, open up CodeShieldJSONClient.ino and compile and download to the Arduino. You only need to do this once.

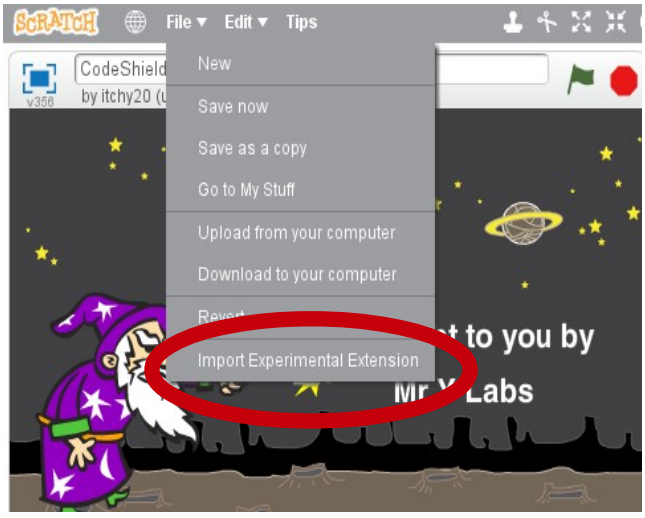
NOTE: The Arduino IDE used for this project was version 1.0.5.

Adding Extension Blocks To Your Scratch Project

If you are using the included Scratch demo project, skip to the next step since it is already included for that application.

For new projects, do the following:

Open up the Scratch program editor, then hold down the Shift key while clicking on File at the upper left of the editor. Select “Import Experimental Extension”. Select the included codeShield.json file. The new code blocks will appear in the “More Blocks” section of scratch.



A description of the imported extension blocks is provided in a section below.

Start the Hardware Extension Module

NOTE: Before starting the Extension Module, the Arduino board with CodeSheild attached, needs to be running and connected to your computer via USB. In addition, the browser needs to be open to your Scratch program containing the installed extension blocks. If either are missing, HWE will detect the missing piece and will not successfully open.

I

n a command window, cd to the included “dist” directory that contains ScratchCodeShield.jar.

Type the following command:

java -jar "ScratchCodeShield.jar" **YOUR_COM_PORT**

YOUR_COM_PORT refers to the com port used for communication between the Arduino IDE and your PC

For example if your com port is specified as “/dev/ttyACM0”, you would type:

java -jar "ScratchCodeShield.jar" /dev/tty/ACM0

Notice there are no quotations on the com part path.

For Windows the command might look like:

java -jar "ScratchCodeShield.jar" COM3

If all is successful, you should see the following output in the command window:

```
Serial Port Opened
TCPServerManager created
ScratchCodeShield started on your machine name here
TCP Server Opened
MessageManager Created
start init
end init - initialized 19 items
Scratch is connected
```

IMPORTANT

If you see “--Closed--” at the end, Scratch closed the connection. This usually happens the first time the extension is started for a Scratch project. Just run the command line again, and the program should run normally.

Now just build and/or run your Scratch program and you should be able to communicate with the CodeShield board.

Also, if you restart the Arduino or reload the Scratch project page while the HWE is connected and

running, HWE will exit because it detects that an interface has disconnected.

Explanation of Extension Blocks

All CodeShield sensors and actuators are supported by this extension.

Reporter Blocks

All reporter blocks end in "_s".

The two switches, **ButtonSwitch_s** and **SlideSwitch_s**, will report a 0 for OFF and a 1 for ON.

Encoder_s will report a value in the range of 1 – 511 for clockwise rotation and -1 to -511 for counter clockwise rotation. It is initially set to zero.

For **all other sensors**, the value returned will be in the range of 0 - 1023.

Actuator Blocks

All actuator blocks end in "_a".

Relay_a activates the relay. A parameter value of Zero causes deactivation and any other value will activate it.

Servo_a accepts a value in the range of 0 -180 and represents the angle of servo rotation.

PiezoTone_a_accepts a frequency in hertz as the first parameter. The second parameter is the tone duration in milliseconds. Frequency has a range of 0-65535 and duration has a range of 0 - 4,294,967,295.

LED Control

LEDs can be controlled either digitally, (either fully on or off), with **LEDDigitalSelect_a** or can can be set to an intermediate illumination with **LEDSelect_a,**

LEDDigitalSelect_a is limited to the red, green, blue and white LEDS. The first block parameter is the LED color selection number and the second parameter is the state. A zero will turn the LED off and any other value will turn it on fully. This block is limited to LED selector numbers 1-4.

LEDSelect_a allows the user to select an illumination level between zero and maximum. The first parameter for this block is the selector number and the second is an illumination value between 0 and 255 (0 =OFF, 255 = Maximum illumination). For selector numbers 5 through 8, multiple LEDs in the RGB device are turned on at different fixed illumination levels to give the effect of additional colors.

NOTE: If the servo has been activated and LEDSelect_a is invoked, the LEDs will activate at full illumination only. This will be the case for selector numbers 5-8 as well. This is the result of a "bug" in the servo library for Arduino and is beyond the control of the author.

For both LED actuator blocks, the first parameter is the LED color selector number and the second is intensity.

LED Color	Selector Number	LEDDigitalSelect Values	LEDSelect_a Values
Red	1	0 or 1	0-255
Green	2	0 or 1	0-255
Blue	3	0 or 1	0-255
White	4	0 or 1	0-255
Indigo	5	x	0-255
Orange	6	x	0-255
Yellow	7	x	0-255
Violet	8	x	0-255

Operational Steps to Use the Hardware Extension Module

1. Press the reset button on the Arduino board.

2. Open the browser to your Scratch project. Do not press the Green Flag until the next step is completed, but you must be on your project page.

3. Start the Hardware Extension Module as described in "Start the Hardware Extension Module" above.

4. Now press the Green Flag to run your Scratch project.

Known Issues

The hardware extension does not stay connected on a first attempt of connection to Scratch. This is under investigation, but appears to be an issue in Scratch. Work around – restart HEM.

LED fading no longer works after operating the servo, even after detach. The Encoder library modifies

the interrupt scheme and as a result, PWM for pins 9 and 10 on the Arduino will no longer be able to control the LEDs in PWM mode. The HEM detects this and operates the LEDs as **LEDDigitalSelect** for all eight LED selections after servo actuation.

If you try rerunning the demo after running it once, you will see this behavior (the LEDs will not dim)

So, to see the LED portion of the demo again:

1. Press the Arduino board reset button.

2. Start the HEM again.

3. Start the demo.

Acknowledgements

Many thanks go to Mitch Resnick and John Maloney both of MIT for providing me with an early copy of the Scratch 2.0 Hardware Extensions specification. Without their support this project would not have been possible.

The good folks at diyode.com for designing and marketing the CodeShield, a quality cost effective kit with a fascinating set of sensors and actuators.

The folks at Arduino who brought back the fun of homebrewing (now called hacking) electronic gadgets.

And last, but certainly not least, my wife, who has both encouraged and allowed me to fully embrace my inner geek.

Alan Yorinks

August 2013

Contact Info:

email: MisterYsLab@gmail.com