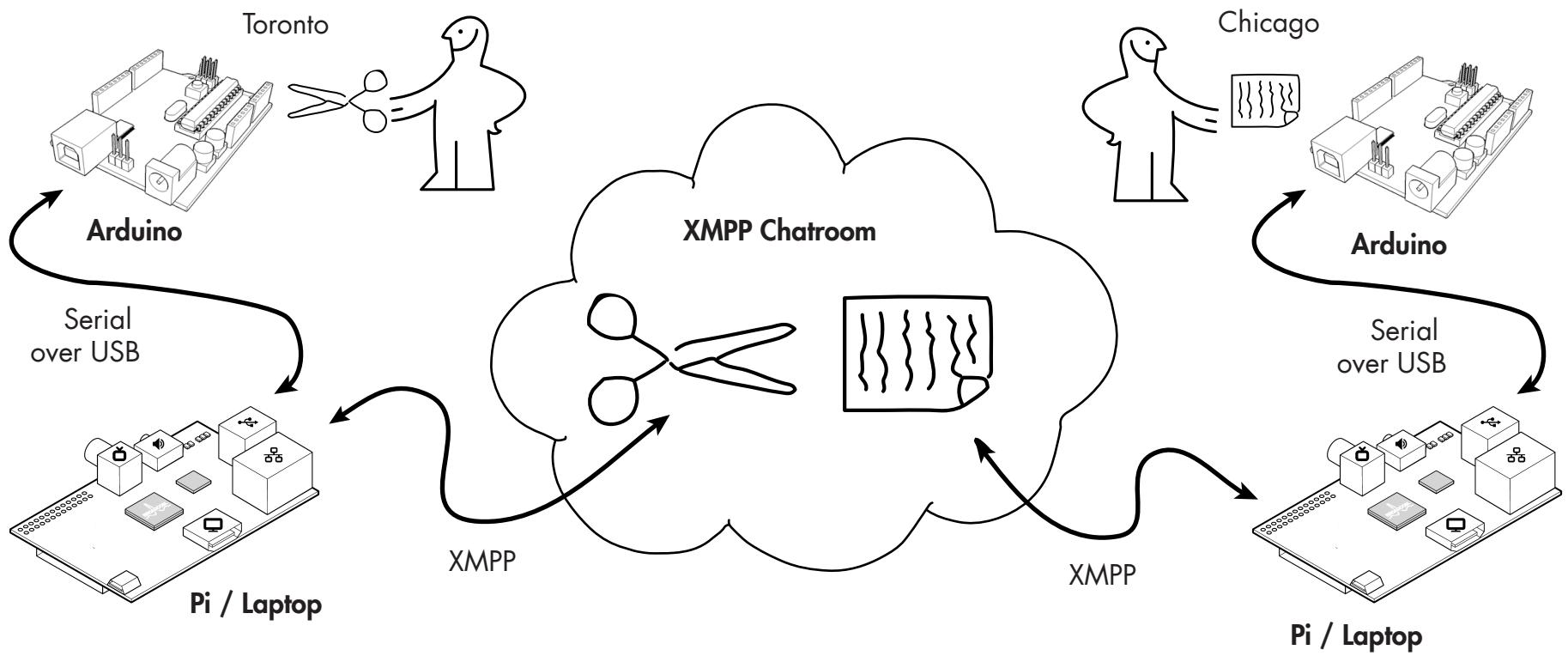
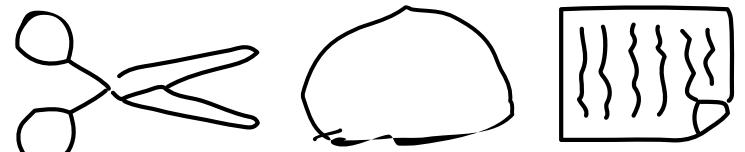
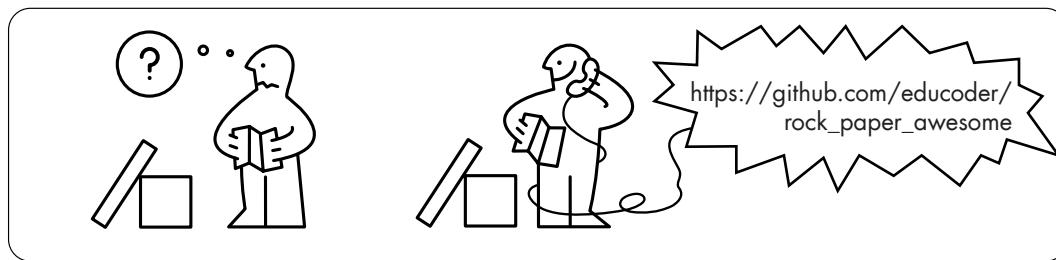


RØCK PÅPER ÄWESOME

https://github.com/educoder/rock_paper_awesome



1 Install the Arduino software

☞ <http://arduino.cc/en/Main/Software>

Install Node.js and npm

```
brew install node  
curl http://npmjs.org/install.sh | sh
```

2 Fork the rock_paper_awesome repo

☞ https://github.com/educoder/rock_paper_awesome



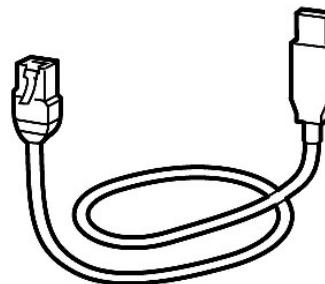
Clone your forked repo

```
git clone git@github.com:username/rock_paper_awesome.git
```

3 Install dependencies

```
cd rock_paper_awesome  
npm install
```

4 Plug the Arduino into your laptop



5 Make a copy of the example Arduino sketch

```
cd arduino/awesome  
cp awesome.example.ino awesome.ino
```

6 Open the copied sketch in the Arduino IDE

A screenshot of the Arduino IDE interface. The title bar says "rock_paper_awesome | Arduino 1.0.1". The top menu bar includes File, Edit, Tools, Sketch, Help, and a language selection dropdown. Below the menu is a toolbar with icons for Open, Save, Print, and other functions. The main workspace shows a tab bar with "rock_paper_awesome" (which is highlighted in red), "Button.cpp", "Button.h", "RPA.cpp", and "RPA.h". The code editor displays the following C++ code:

```
#include <stdio.h>  
#include "Button.h"  
#include "RPA.h"  
  
#define SERIAL_BAUDRATE 9600  
  
// the pins that control the brightness of RED, GREEN, and BLUE LEDs respectively  
#define R 9  
#define G 10  
#define B 11  
  
// pins that receive input for ROCK, PAPER, and SCISSOR  
#define PIN_ROCK 5  
#define PIN_PAPER 7  
#define PIN_SCISSORS 6  
// pin that receives input from the NEW GAME (READY) button  
#define PIN_READY 3  
  
/* before considering the  
 * WAIT 50  
 * then pressed again
```

11

Connect your RPA to the XMPP chatroom and play!

Back in your terminal, run the awesome.js Node service to connect your Arduino to the XMPP chatroom. awesome.js will broadcast events to the chatroom so that other players can receive them.

awesome.js should automatically detect your RPA plugged in to your computer using USB. If for some reason it doesn't, open awesome.js and edit the SERIALPORT value to point to your Arduino serialport dev (e.g. /dev/tty.usbmodemfd131).

```
<< [FROM ARDUINO] >> podoor_mzukowski$ []
XMPP: Connected to toronto@badger.encorelab.org
XMPP: Joined room rock-paper-awesome@conference.badger.encorelab.org
EVENT===== [ connected []
>> [TO ARDUINO] '> connected'
TRANSITION== [ OFFLINE ----connected----> ONLINE
>> [TO ARDUINO] '= ONLINE'
```

Note that all players connected at the chatroom will see your events, but the game is currently only designed to be played one-on-one. Make sure you're only playing against one other RPA, otherwise things can get wonky.

12

Use a Raspberry Pi instead of your computer

See the RPA wiki for instructions:



https://github.com/educoder/rock_paper_awesome/wiki

8

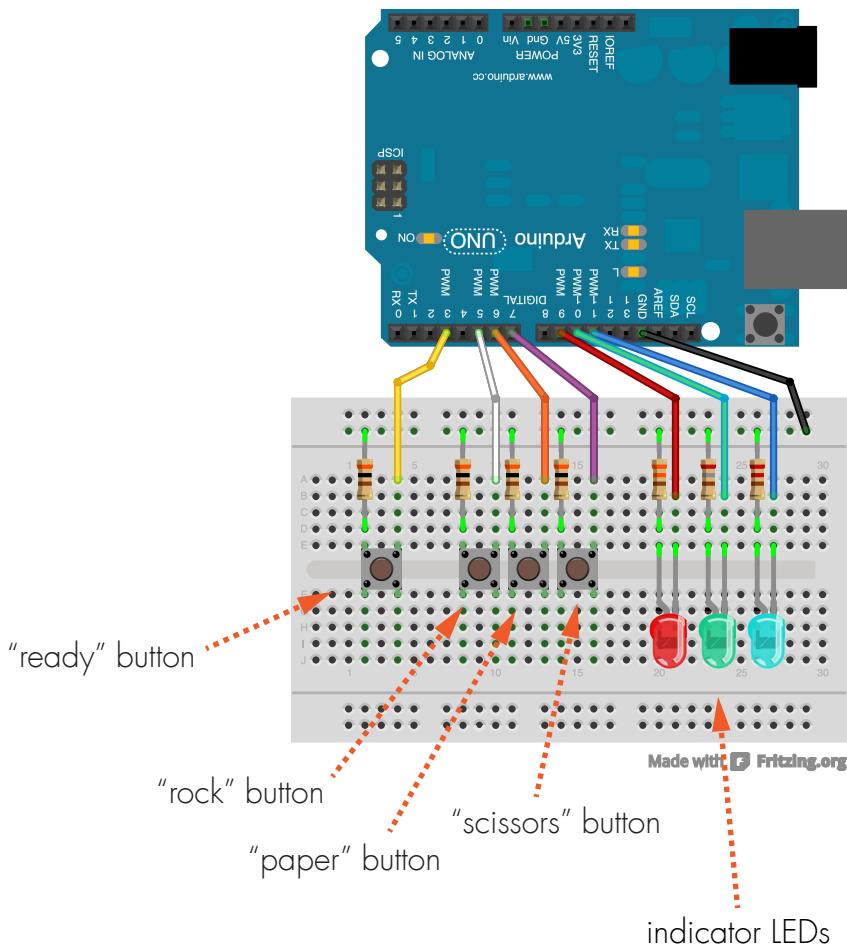
Think of some awesome ways to:

1. Allow the player to choose their weapon: Rock, Paper, or Scissors.
 2. Allow the player to indicate that they are ready to play.
 3. Show the other player's chosen weapon.
 4. Show that the other player is ready.
 5. Show that the other player is in the room.
 6. Show who won, lost, or tied.
 7. Other stuff? Dispense candy when we win? (Lightly) electrocute when we lose? It's up to you... just be awesome.

9

Wire up your Arduino

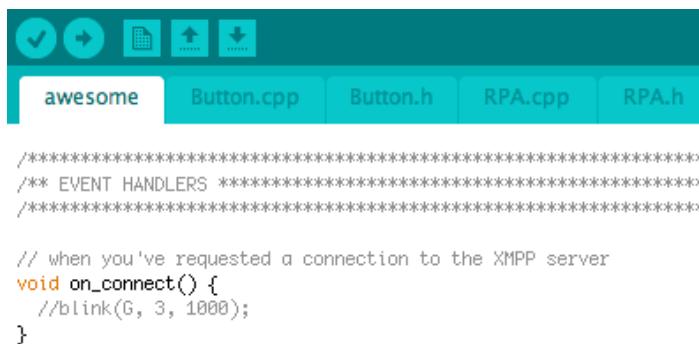
The wiring depends entirely on how you decide to physically represent the game. But the minimal configuration will look something like this:



10

Modify the Arduino code for your physical representation

In the awesome.ino code, look for EVENT HANDLERS and STATE HANDLERS. These functions are automatically executed whenever your Rock Paper Scissors receives events (EVENT HANDLERS) and enters a new state (STATE HANDLERS).



```
*****  
*** EVENT HANDLERS ***  
*****  
  
// when you've requested a connection to the XMPP server  
void on_connect() {  
    //blink(G, 3, 1000);  
}  
  
*****  
*** STATE HANDLERS ***  
*****
```

The Rock Paper Awesome software is structured around a finite state machine. At any point your RPA is in one of several possible states. The RPA receives events — either from the XMPP chatroom or from your physical Arduino inputs — and in response to these events transitions from one state to another.

See the finite state machine UML diagram on the last page for a map of states and possible transitions.

For example:

```
void on_you_lose() {  
    setLED(5, 255);  
}
```

This will turn on the LED connected to pin number 9 (or more specifically, it will set its brightness up to the maximum value of 255).*

```
void while_ready_to_play() {  
    myservo.write(35);  
}
```

This will turn on the LED connected to pin number 9 (or more specifically, it will set its brightness up to the maximum value of 255).*

```
void on_you_win() {  
    for (int n = 0; n < 20; n++) {  
        int dur = 1000/durationsWin[n];  
        tone(13, melodyWin[n], dur);  
        int pause = dur * 1.3;  
        delay(pause);  
        noTone(13);  
    }  
}
```

This will play a song when you win a game.*

* Some additional code and libraries required.

BEHOLD! The Rock Paper Awesome Finite State Machine!

