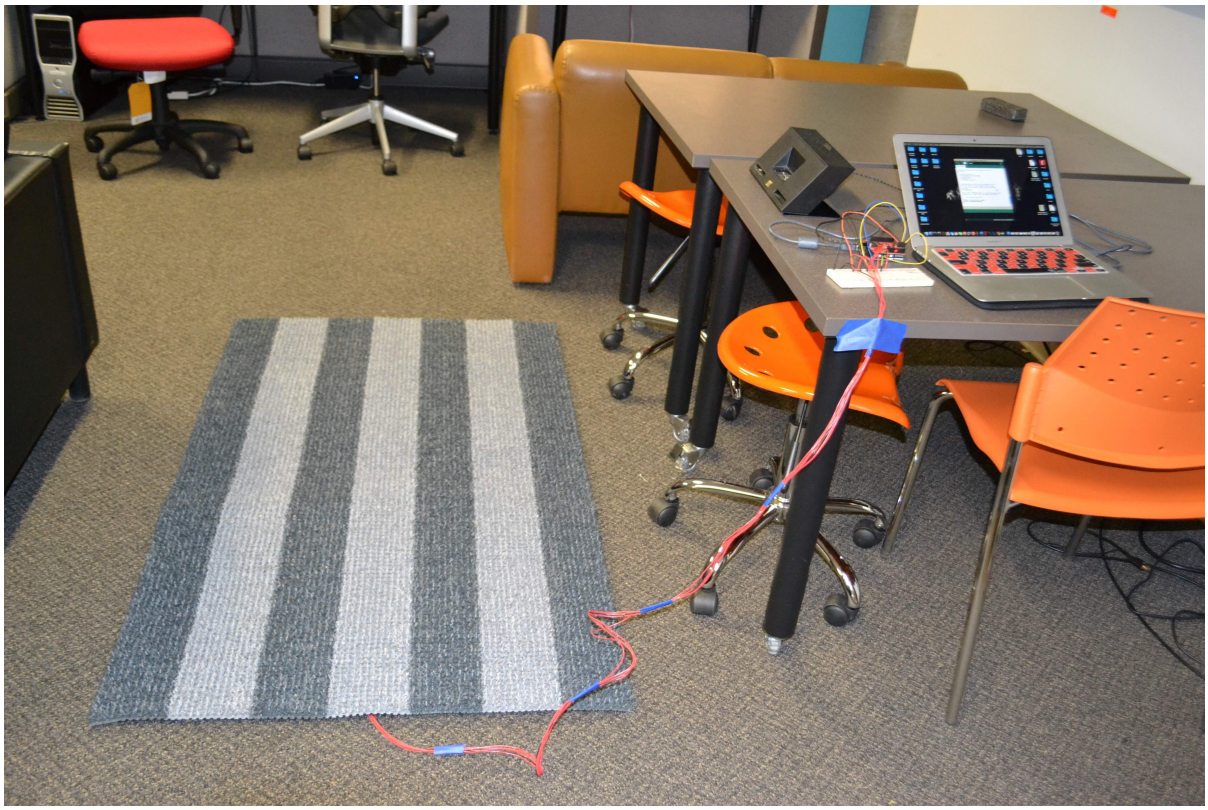
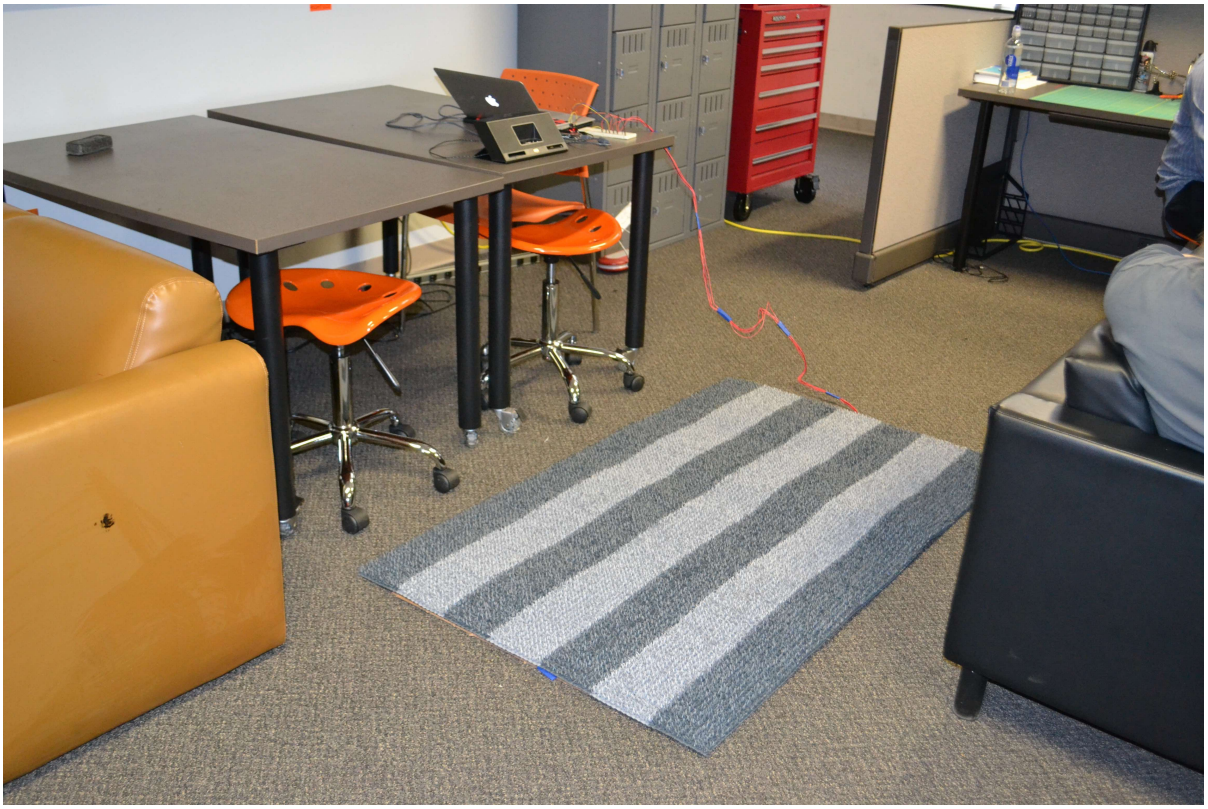
The Ambient Sole: Embodiment, Attunement, and Reciprocity in the Act of Moving through Space

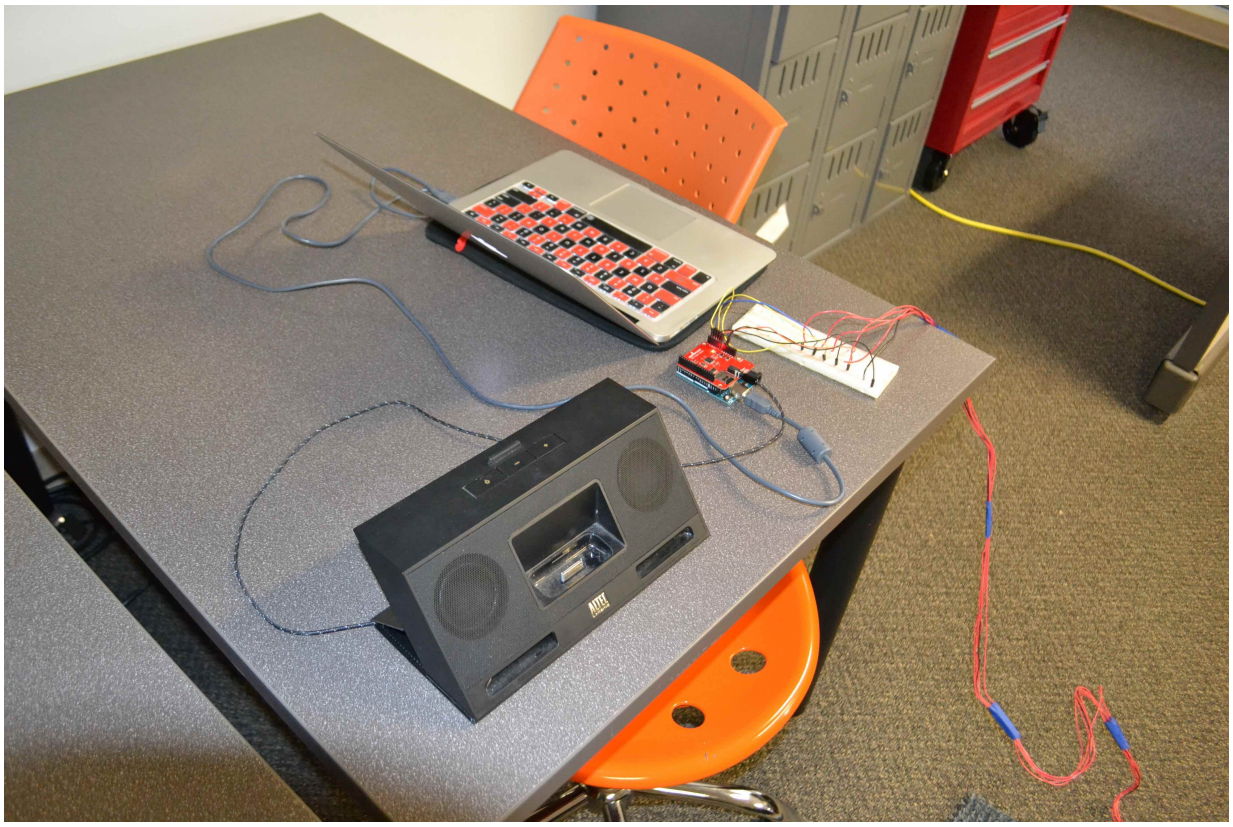
Jessica Handloff, Geoffrey Luurs, and Sarah Evans
North Carolina State University, 2015

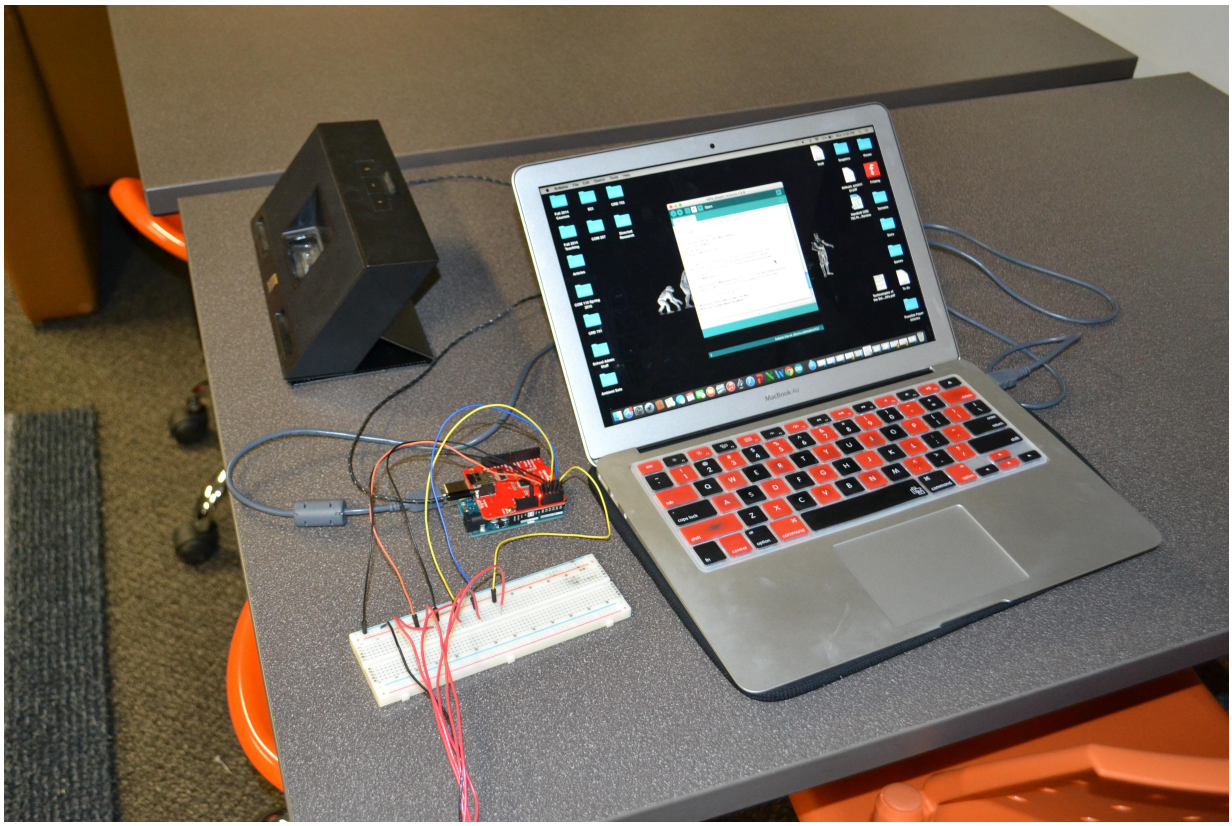
Photos:



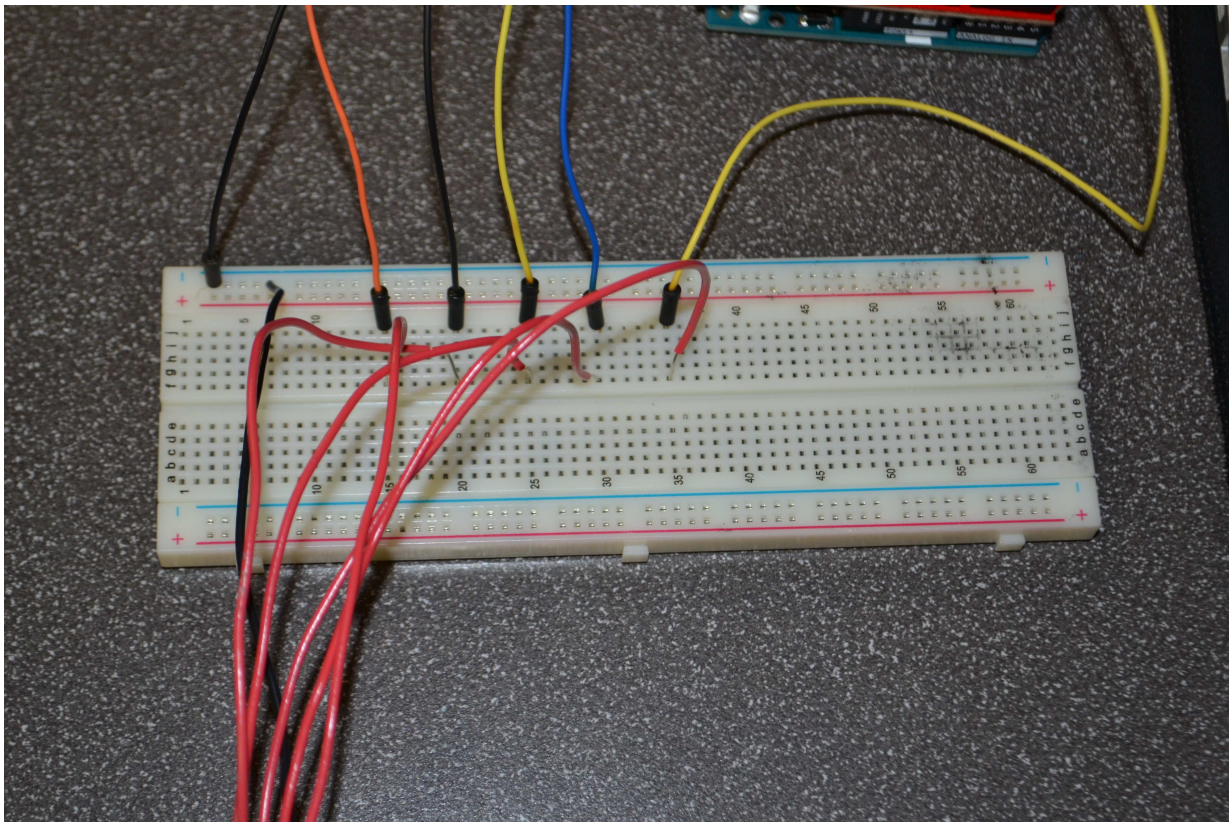


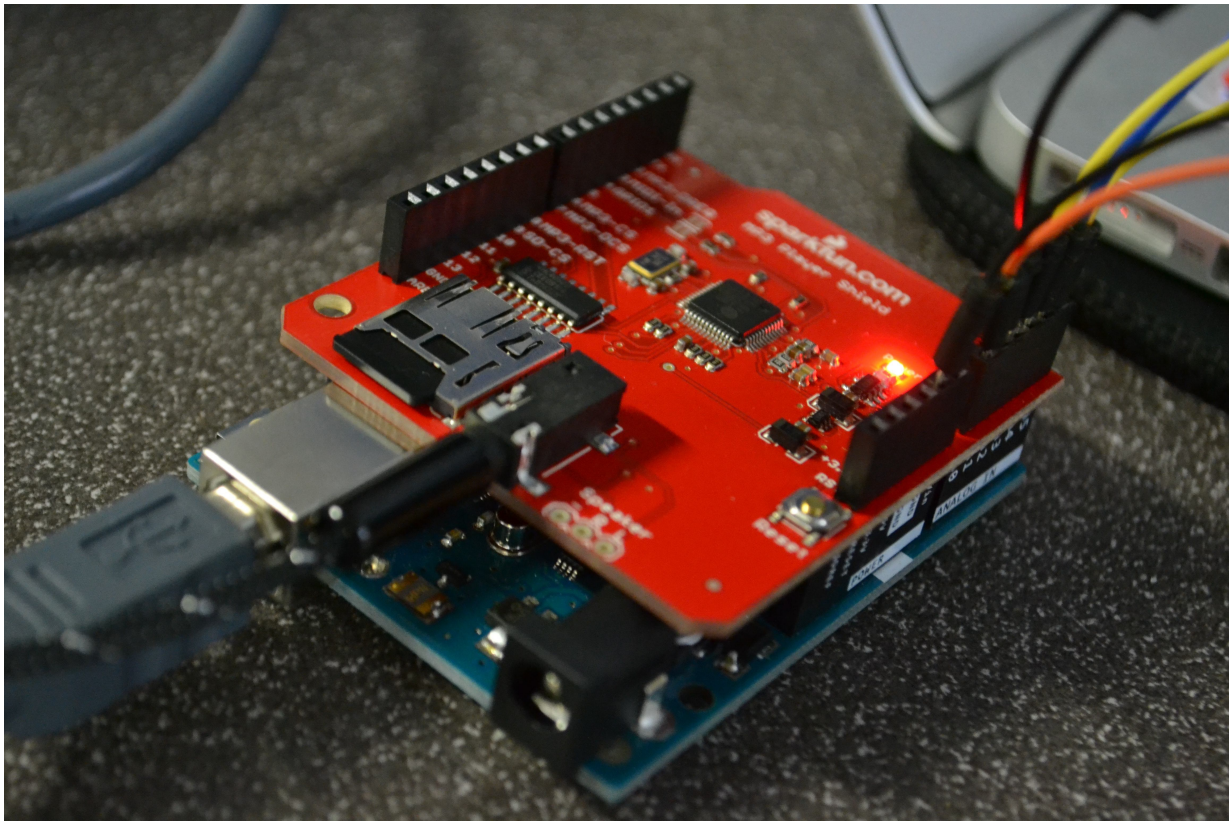
2





3





5

Dimensions:

The mat measures 3ft by 5ft, with 6ft of trailing wire connected to the bread board and arduino, which require a small space to sit.

Technical Specifications:

Materials List
1 - Arduino Uno
1 - soldered MP3 shield
1 - breadboard
6 - 6in leads
5 - 10ft length of insulated wire (red)
5 - 3ft length of insulated wire (black)
1 - 7ft length insulated wire (black)
10 sqft heavy duty aluminum foil
3'x5' cardboard sheet
10 - 1'x6" cardboard cutouts
5 - 1'x6" perforated foam pieces
1 - 3'x5' floor mat
1 - Speaker system
1 - AUX cable
1 - lithium ion battery
1 - roll masking tape
1 – USB cable, Standard AB

Assembly Instructions
1. Attach aluminum foil to one side of each 1'x6" cardboard cutout with masking tape. Secure exposed ends of red insulated wire to 5 foil covered cutouts with masking tape. Secure exposed ends of 3' lengths of black insulated wire to remaining 5 foil covered cutouts.

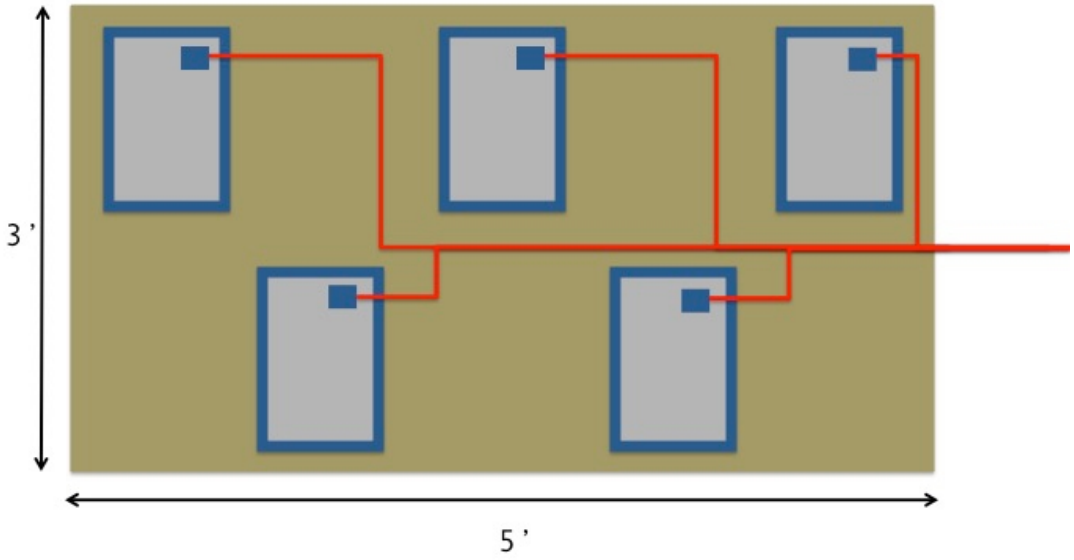
2. Secure the 5 foil covered cutouts with red wire to the 3'x5' cardboard sheet (foil side up), staggered lengthwise. Place a perforated foam piece over each cutout. Place 5 foil covered cutouts with black insulated wire (foil side down) on top of foam pieces. Trailing ends of black wires should be exposed and secured to 5' length of black insulated wire.

3. Attach red wires in sequence on breadboard. Attach black wire to top row of breadboard. Attach leads from breadboard in sequence to each connection. Insert lead from the black wire into the GND pin on the MP3 shield connected to the Arduino Uno. Insert lead in sequence to pins A0-A4.

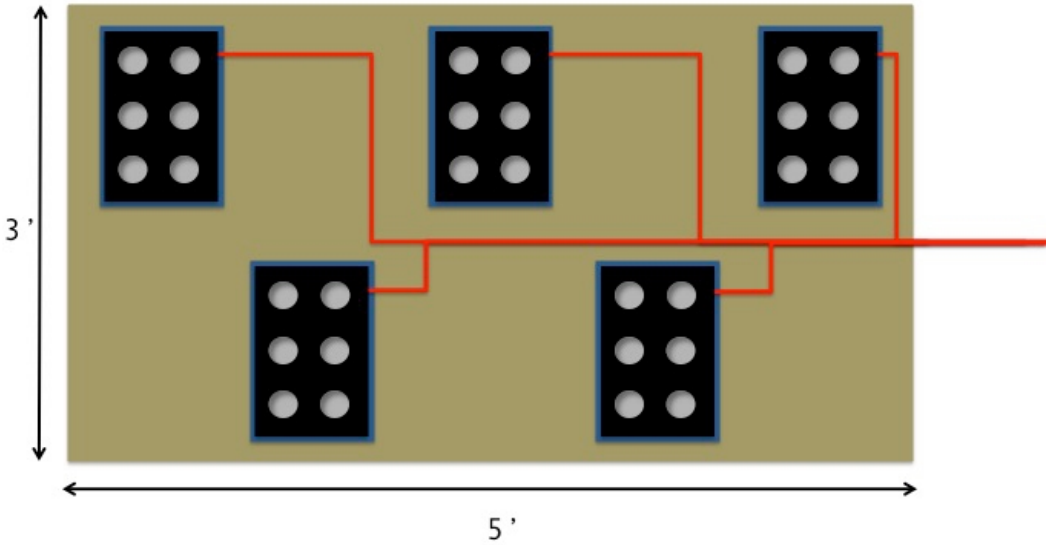
4. Insert USB cable into computer and arduino uno and upload code (below). Remove cable and attach lithium ion battery. Connect AUX cable to MP3 shield and speaker system.

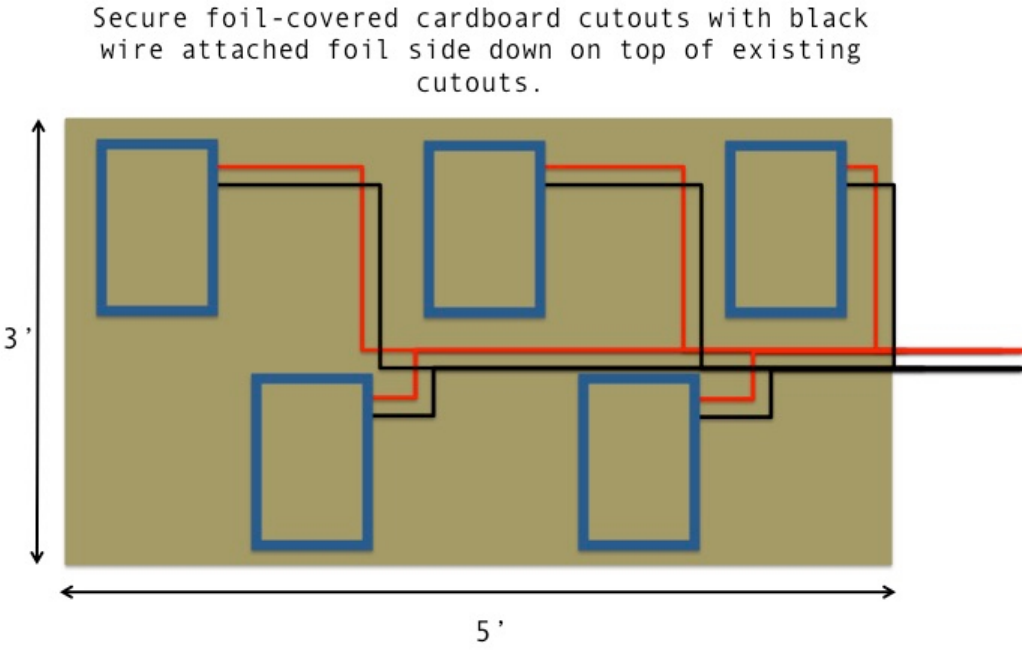
5. Step on mat, and produce some sounds!

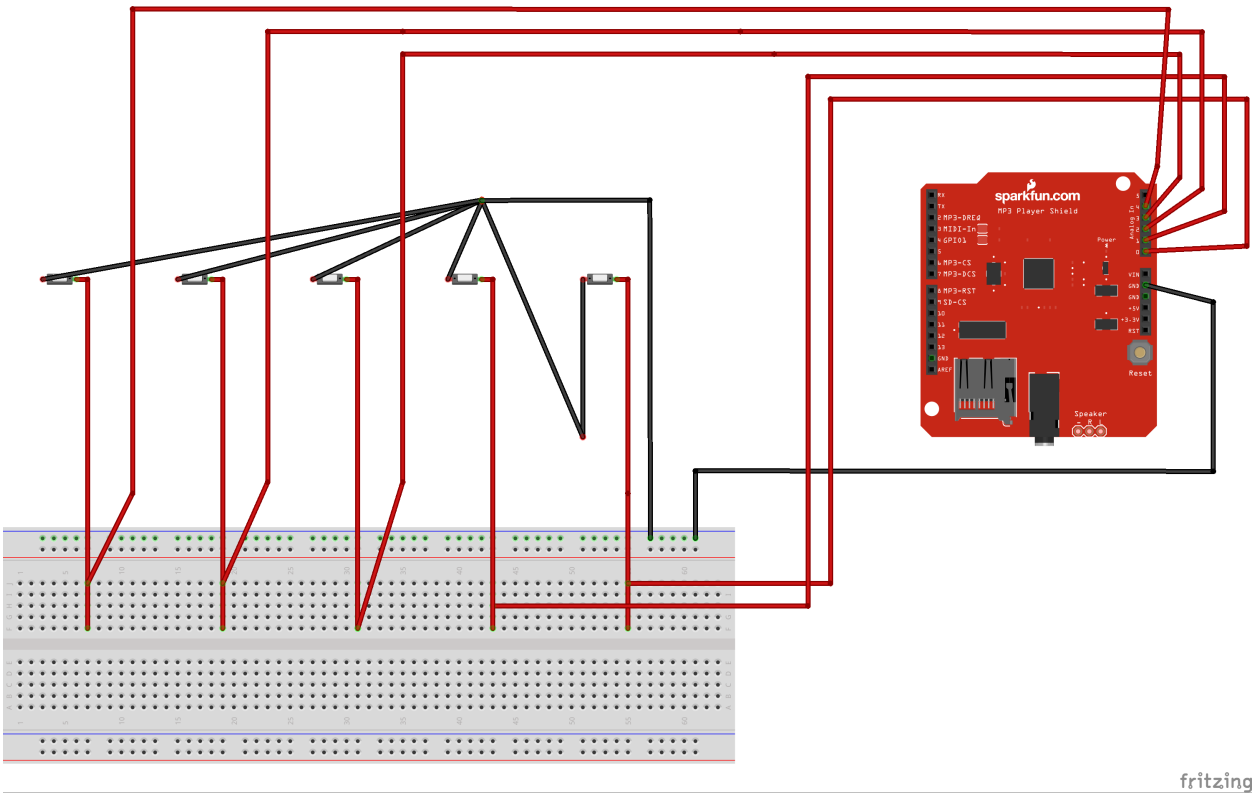
Secure foil-covered cardboard cutouts with red wire attached to 3'x5' cardboard sheet, foil side up.



Place perforated foam pieces on top of cutouts.




Secure foil-covered cardboard cutouts with black wire attached foil side down on top of existing cutouts.
3'
5'


sparkfun.com
MP3 Player Shield
fritzing

Arduino Code

(adapted from SparkFun: https://learn.sparkfun.com/tutorials/mp3-player-shield-hookup)

```
/*
  MP3 Shield Trigger
  by: Jim Lindblom
     SparkFun Electronics
  date: September 23, 2013

  This is an example MP3 trigger sketch for the SparkFun MP3 Shield.
  Pins 0, 1, 5, 10, A0, A1, A2, A3, and A4 are setup to trigger tracks
  "track001.mp3", "track002.mp3", etc. on an SD card loaded into
  the shield. Whenever any of those pins are shorted to ground,
  their respective track will start playing.

  When a new pin is triggered, any track currently playing will
  stop, and the new one will start.

  A5 is setup to globally STOP playing a track when triggered.

  If you need more triggers, the shield's jumpers on pins 3 and 4
  (MIDI-IN and GPIO1) can be cut open and used as additional
  trigger pins. Also, because pins 0 and 1 are used as triggers
  Serial is not available for debugging. Disable those as
  triggers if you want to use serial.

  Much of this code was grabbed from the FilePlayer example
  included with the SFEMP3Shield library. Major thanks to Bill
  Porter and Michael Flaga, again, for this amazing library!
*/

#include <SPI.h>          // SPI library
#include <SdFat.h>        // SDFat Library
#include <SdFatUtil.h>    // SDFat Util Library
#include <SFEMP3Shield.h>  // Mp3 Shield Library

SdFat sd; // Create object to handle SD functions

SFEMP3Shield MP3player; // Create Mp3 library object
// These variables are used in the MP3 initialization to set up
// some stereo options:
const uint8_t volume = 0; // MP3 Player volume 0=max, 255=lowest (off)
const uint16_t monoMode = 1;  // Mono setting 0=off, 3=max

/* Pin setup */
#define TRIGGER_COUNT 9
```

```
int triggerPins[TRIGGER_COUNT] = {0, 1, 5, 10, A0, A1, A2, A3, A4};
int stopPin = A5; // This pin triggers a track stop.
int lastTrigger = 0; // This variable keeps track of which tune is playing

void setup()
{
 /* Set up all trigger pins as inputs, with pull-ups activated: */
 for (int i=0; i<TRIGGER_COUNT; i++)
 {
  pinMode(triggerPins[i], INPUT_PULLUP);
 }
 pinMode(stopPin, INPUT_PULLUP);

 initSD();  // Initialize the SD card
 initMP3Player(); // Initialize the MP3 Shield
}

// All the loop does is continuously step through the trigger
//  pins to see if one is pulled low. If it is, it'll stop any
//  currently playing track, and start playing a new one.
void loop()
{
 for (int i=0; i<TRIGGER_COUNT; i++)
 {
  if ((digitalRead(triggerPins[i]) == LOW)) // && ((i+1) != lastTrigger))
  {
   lastTrigger = i+1; // Update lastTrigger variable to current trigger
   /* If another track is playing, stop it: */
   if (MP3player.isPlaying())
    MP3player.stopTrack();

   /* Use the playTrack function to play a numbered track: */
   uint8_t result = MP3player.playTrack(lastTrigger);
   // An alternative here would be to use the
   //  playMP3(fileName) function, as long as you mapped
   //  the file names to trigger pins.

   if (result == 0)  // playTrack() returns 0 on success
   {
    // Success
   }
   else // Otherwise there's an error, check the code
   {
    // Print error code somehow, someway
   }
  }
 }
```

```
  // After looping through and checking trigger pins, check to
  //  see if the stopPin (A5) is triggered.
  if (digitalRead(stopPin) == LOW)
  {
    lastTrigger = 0; // Reset lastTrigger
    // If another track is playing, stop it.
    if (MP3player.isPlaying())
      MP3player.stopTrack();
  }
}

// initSD() initializes the SD card and checks for an error.
void initSD()
{
  //Initialize the SdCard.
  if(!sd.begin(SD_SEL, SPI_HALF_SPEED))
    sd.initErrorHalt();
  if(!sd.chdir("/"))
    sd.errorHalt("sd.chdir");
}

// initMP3Player() sets up all of the initialization for the
// MP3 Player Shield. It runs the begin() function, checks
// for errors, applies a patch if found, and sets the volume/
// stero mode.
void initMP3Player()
{
  uint8_t result = MP3player.begin(); // init the mp3 player shield
  if(result != 0) // check result, see readme for error codes.
  {
    // Error checking can go here!
  }
  MP3player.setVolume(volume, volume);
  MP3player.setMonoMode(monoMode);
}
```

Biographies

Jessica Handloff (MA, East Carolina University) is a doctoral student at North Carolina State University in the Communication, Rhetoric, and Digital Media program. Her research areas are critical media analysis and cultural studies, and her current interdisciplinary work considers digital engagement and embodiment, and media, technology, and the military.

Geoffrey Luurs (MA, Colorado State University) is a doctoral student at North Carolina State University in the Communication, Rhetoric, and Digital Media program. His program of research explores interpersonal health communication within the digital humanities. His current work focuses on themes of anti and pro-social communication, sexuality, and personal well being.

Sarah Beth Evans (MA, Syracuse University) is a doctoral student in the Communication, Rhetoric, and Digital Media program at North Carolina State University. Her interdisciplinary research focuses on gaming and social networks through various critical, rhetorical, feminist, and ethnographic methods.

12