



Si4707 Hookup Guide

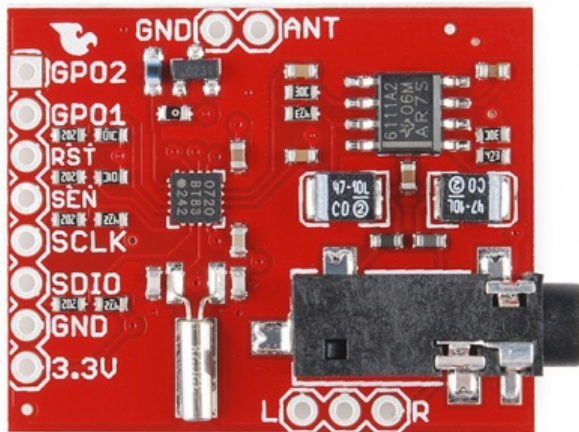
CONTRIBUTORS:  JIMBO

Introduction

Weather-band radio is an awesome public service provided in the US, Canada, and Bermuda. With hundreds of transmitting stations dotting the country, weather radio acts as the “voice of NOAA” (National Oceanic and Atmospheric Administration). In addition to spouting out weather forecasts, weather radio also implements a messaging protocol for emergency weather alerts called **SAME** (Specific Area Message Encoding).

An audio snippet recorded out of the Si4707. Winter storm advisory in Boulder...at the end of April!






The Silicon Labs Si4707 is capable of both receiving weather radio broadcasts and decoding SAME messages. It’s also able to check for 1050Hz alert tones. It’s a unique chip, and we liked it so much we stuck it on a breakout board. In addition to the Si4707, the breakout also includes supporting passive components and a headphone stereo amplifier.



There are a variety of consumer-grade radio receivers with weather band capabilities built-in. But what's the fun in that? With the Si4707 you can make your own weather watching radio!

Required Materials

In this tutorial, we'll explain how to hook-up to the Si4707 breakout using the **Arduino** development platform. To follow along, you'll need these materials:

Si4707 Simple HookUp SparkFun Wish List	
	<p>SparkFun Weather Band Receiver Breakout - Si4707 WRL-11129 [Weather-band radio](https://en.wikipedia.org/wiki/Weather_radio) is a...</p>
	<p>Break Away Headers - Straight PRT-00116 A row of headers - break to fit. 40 pins that can be cut to any size. Us...</p>
	<p>RedBoard - Programmed with Arduino DEV-11575 At SparkFun we use many Arduinos and we're always looking for the...</p>
	<p>Breadboard - Translucent Self-Adhesive (Red) PRT-11317 **Description**: How could anyone improve on the clear plastic bread...</p>
	<p>Jumper Wires Premium 6" M/M Pack of 10 PRT-08431 This is a SparkFun exclusive! These are 155mm long jumpers with m...</p>

We've suggested the RedBoard for all of your Arduino needs, but any of the development boards should work.

You'll also need a listening device – either **headphones or speakers** – with a **3.5mm stereo jack** connector. You want to hear what the radio has to say, don't you?!

Advanced (Optional)

If you'd like to add your own external antenna and/or connect a small speaker to the board, some of these will come in handy:

- A couple feet of wire - This is used to make a really basic antenna.
- Audio speaker - There are a variety of speakers out there including thin and PCB mount packages.

Required Tools

This assembly will require some simple soldering, so you'll need a soldering iron and a bit of solder.

Advanced (Optional)

If you're into the more advanced hookup, you'll need something to cut the jumper on the board. For that, a hobby Knife should be all you need.

Recommended Reading

This tutorial assumes some previous electronics knowledge to get along. If you're not familiar with these concepts, consider checking out our tutorials on the subject!

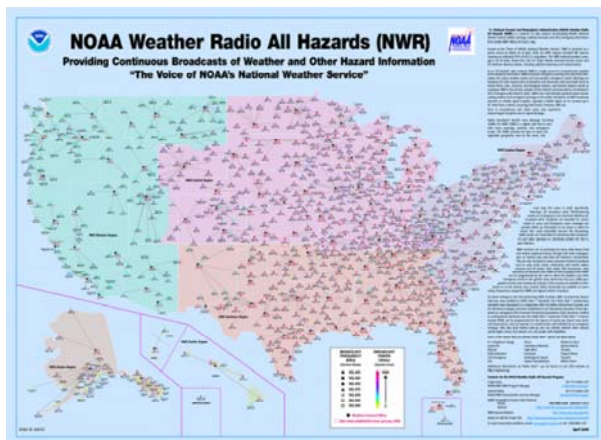
- How to Solder - You'll need some solder points somewhere between the breakout board and your Arduino.
- What is an Arduino? - Be familiar with Arduino. We'll be talking about it a lot in this tutorial.
- I²C Communication - This is the communication standard the Arduino uses to talk to the Si4707.
- How to Use a Breadboard - This guide will use a breadboard to connect between the Arduino and Si4707 breakout.
- Working With Wire - If you want to add an antenna, you'll need to cut and strip some wire.

So grab an Arduino and an Si4707 Breakout, follow along, and you'll be listening to the soothing robotic-weather-person voice in no time!

What's Weather Band

Weather radio (known as *national weather radio* or NWR) is a nationwide, 24/7 weather forecasting, watching, and alert system. Weather band transmissions are frequency-modulated (FM) at one of seven VHF

frequencies (in MHz): 162.400, 162.425, 162.450, 162.475, 162.500, 162.500, 162.525, or 162.550. With broadcast stations all over the country, upwards of 97% of the U.S. population is covered by NWR.



Click to embiggen this informative map of the many many NWR transmitting stations. (Image courtesy of nws.noaa.gov).

The radio broadcasts usually consist of an automated voice (either Donna, Tom, or Javier) speechifying a weather forecast. They'll inform you of current temperatures, wind conditions, precipitation, etc., and they'll give you an outlook for the coming hours and days. The weather-band-radio-robot-voices are really very relaxing, almost hypnotic.

And remember NOAA Weather Radios (very dramatically) SAVE LIVES!!!

What's SAME?

One particularly critical job of NWR is alerting us to immediate weather emergencies. Those emergency weather alerts are reinforced by **Specific Area Message Encoding (SAME)**. SAME is a digital code transmitted under cover of the normal weather radio audio broadcasts. SAME messages are short, but they convey important information like what weather event triggered the alert (winter storm warning, tornado warning, hurricane, etc.) and the area affected.

The SAME messages are accompanied by a lovely 10 second 1050Hz tone, alerting listeners to the incoming warning message.



The Si4707 is fully capable of interfacing with SAME messages and the 1050 Hz tone. Want the radio to turn on only when an alert is incoming? Or blink an LED when catastrophic weather is headed your way? It's possible thanks to SAME.

For more information on National Weather Radio head over to NOAA's

homepage.

Board and IC Overview

In the words of Silicon Labs, the Si4707 is “the industry’s first weather band (WB) radio receiver to include a specific area message encoding (SAME) processor.” It’s a really neat chip. Aside from being able to tune to all weather band frequencies and process SAME messages, the Si4707 also hosts a variety of cool features. It’s versatile: it can be interfaced with using either I²C, SPI, or a 3-wire interface. And, it’s got a really clean-sounding audio quality.

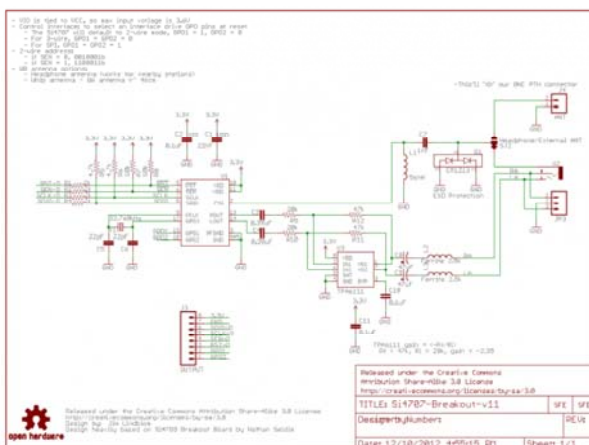
Silicon Labs has produced a wealth of supporting material for the chip. All of it hosted on their website. Some of the really useful documents include:

- Si4707-B20 - The datasheet.
- AN332 - The Si4707 programming guide, where you’ll find information about all of the Si4707’s registers and settings.

It’s important to point out that the Si4707 has a **maximum interface supply voltage of 3.6V**. When we connect it to an Arduino, it’ll be powered by the 3.3V line. Voltage on the input lines should also be limited to about 3.3V as well.

Breakout Board Schematic

The Si4707 Breakout Board surrounds the Si4707 with all of the supporting circuitry one might need to hit the ground running with the chip.



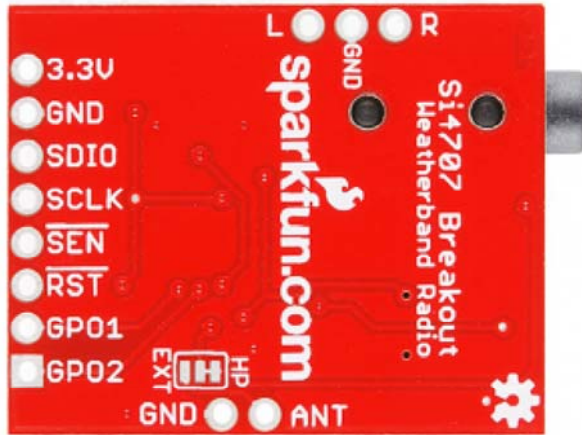
Si4707 Breakout Board schematic. Click to view a larger PDF version.

One of the more important parts surrounding the Si4707 is the 32.768kHz crystal, which provides the IC’s reference clock. All of the I/O lines are pulled up to 3.3V through 4.7kΩ resistors. The antenna portion of the circuit has a filtering 56nH inductor and an ESD protecting diode.

The left and right analog audio outputs of the Si4707 are sent through a headphone amplifier (a TPA6111A2), which boosts the signal. The output of the amplifier is sent to a 3.5mm audio jack.

Board Pin-Out

The Si4707 Breakout has one main interface header and two smaller optional headers on the side of the board.



A view of the bottom of the Si4707 breakout board.

On the main header are the following pins:

- **3.3V** - The supply voltage input. This will go to both the V_{DD} and V_{IO} pins of the Si4707. The supply voltage should be between 1.5 and 3.6V.
- **GND** - This supplies both the digital ground and RF ground of the Si4707.
- **SDIO** - Serial data input/output for all interfaces.
- **SCLK** - Serial clock input for all interfaces.
- **SEN** - Serial enable input. If used in I²C mode, this pin selects the 7-bit address of the IC. In SPI mode, this is the chip-select input.
- **RST** - The reset input. It's active-low, so connecting this pin to ground will turn the chip off.
- **GP01** - General purpose output 1.
- **GP02** - General purpose output 2. This pin also serves as an interrupt output if so configured.

Other pins not on the main 8-pin header are:

- **ANT** - This is connected to the **FMI** input of the Si4707—the WB RF input. If you'd like to use an external antenna, select this output pin on the jumper (on the bottom of the board, near the pin).
- **L** and **R** - These are the analog audio outputs from the headphone amplifier. If you'd rather not use the 3.5mm audio jack, you can use these headers instead.

Well, that explains the board. Now we can move on to hooking it up!

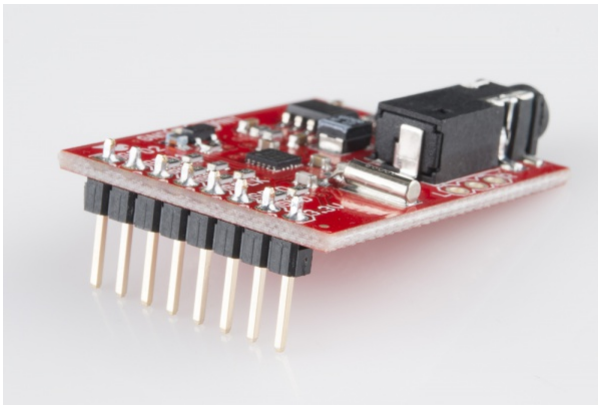
Hardware Hookup

In this section we'll go from soldering up the Si4707 breakout to wiring it up to your Arduino. After that, on the next page, we'll get into programming the Arduino to interact with the weather band radio chip.

Soldering Headers

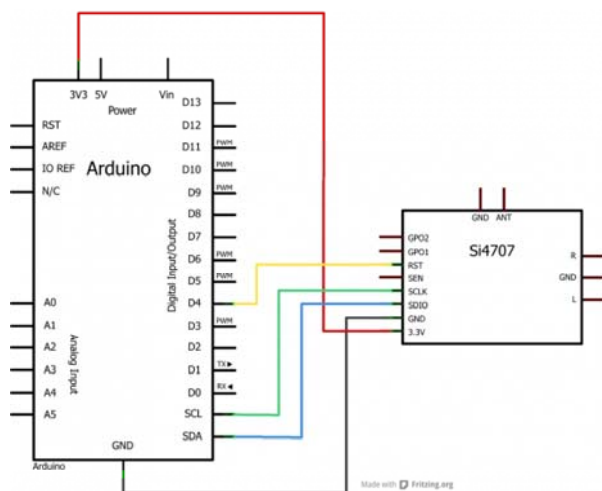
Soldering headers to the Si4707 will create a solid physical and electrical connection to the board. If you've never soldered before, don't worry! We've got a tutorial for that. These solder points are among the easier you'll ever have to make.

After soldering some headers into the breakout board, it should look a little something like this:

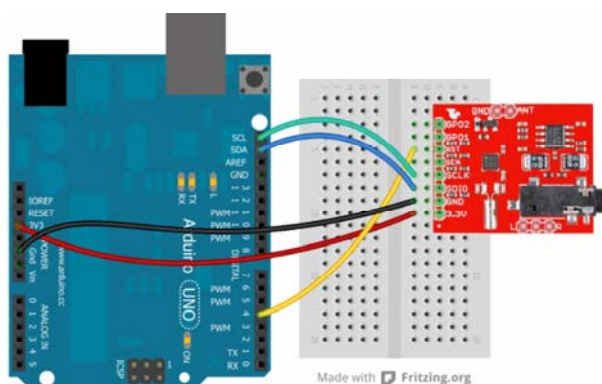


Connecting to the Arduino

We'll be using the Si4707's I²C interface for communication between the Arduino and Si4707. This interface requires connecting to the **SDIO** and **SCLK** pins to send data and a clock signal, respectively. Having control over the Si4707's reset pin (**RST**) is also helpful. Aside from that, all we need is 3.3V and ground. Here's a schematic:



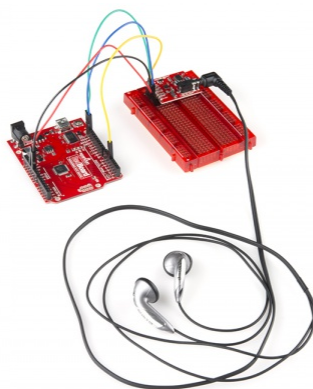
Or, for the more visually inclined, a Fritzing diagram:



The SDA and SCL pins should be present on most Arduinos. Older, pre-rev3 Arduinos might not have SCL and SDA pins. In that case, connect SDIO to **A4** and SCLK to **A5**.

Plug In Headphones/Speakers

No doubt you want to pick up what the weather radio station is putting down! The Si4707 Breakout has an on-board 3.5mm stereo jack, plug some headphones, or your favorite powered speakers, into that jack.



The breakout board comes configured to use the headphone cord as an antenna. So it helps to have headphones or speakers with about 1.5-3ft long, untangled wires.

Once the connections have been made, you can move on to programming the Arduino!

Firmware Upload

Click [here](#) to download our Si4707 I²C example code. You can also grab the code from the product's github repository.

This example code comes with three different files:

- **si4707_example_code_i2c.ino** - The **main** Arduino file. This is where `setup()` and `loop()` are defined, as well as a few other global variables and useful functions. Mess with this one as you please.
- **si4707_system_functions.ino** - This is where an array of “system” functions—nitty-gritty functions that create commands, and receive responses—are kept. It should be safe to ignore this file.
- **si4707_definitions.h** - This file defines all of the Si4707's register and property addresses. Don't change these definitions.

Open up **si4707_example_code_i2c.ino**, and head down to line 62.

There's just one global variable that needs editing before you upload the sketch:

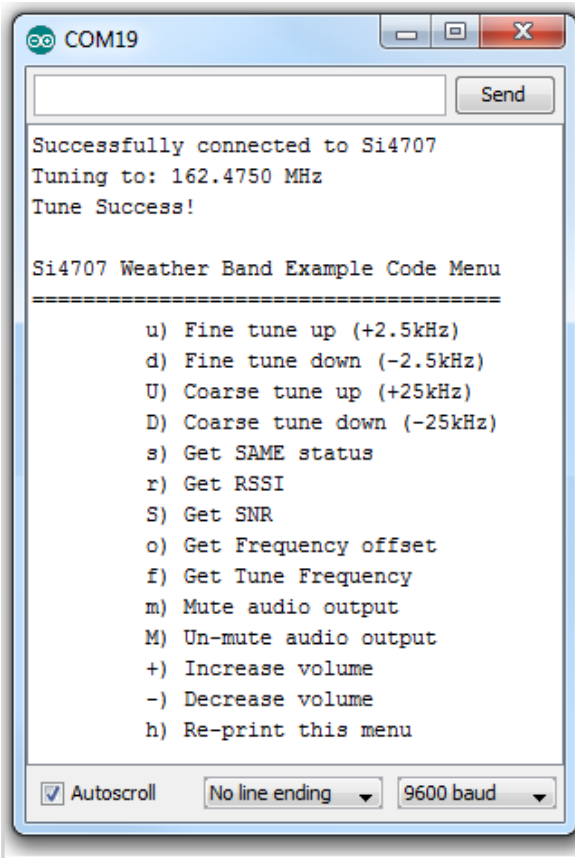
```
// Put the WB frequency you'd like to tune to here:
// The value should be in kHz, so 162475 equates to 162.475 MH
z
// The sketch will attempt to tune to this frequency when it s
tarts.
// Find your frequency here: http://www.nws.noaa.gov/nwr/index
nw.htm
unsigned long tuneFrequency = 162475; // 162.475 MHz
```

Edit the `tuneFrequency` variable to match the frequency of your nearest weather-band station. If you're not sure what that is, consult NOAA's Station Listing page. That variable should be your station's frequency in kHz. For example, if your local weather station is found at 162.500 MHz, set that variable to `162500`.

With that, go ahead and upload your sketch. Then, open up the serial monitor, and make sure the baud rate is set to 9600 bps.

Upon reset, the Arduino will attempt to tune to your weather band station. Have a listen to your headphones. You should at least hear static, if not a weather station.

A configuration menu should also be printed out to the serial monitor:



Try interacting with the config menu. If the volume is too loud, send a few `-`'s. Or just mute the volume by sending a lowercase `m` (an uppercase `M` will un-mute). You can also get the **received signal strength** (RSSI) and **signal-to-noise ratio** (SNR) of the received signal by sending `r` and `S`. Knowing those, you can try tuning up or down finely (`u` or `d`) or coarsely (`U` or `D`) to optimize the signal.



Example of how one might use the config menu to tune the radio and read RSSI/SNR. GIF made using LICEcap, which is the coolest thing ever.

Understanding the Sketch

There are a few functions in the main example sketch that might come in handy, so we'll give a quick overview of them here.

Each of these functions has reliance on definitions in the other two files (*si4707_definitions.h* and *si4707_system_functions.ino*). Make sure each of those files are in the same directory as the main sketch and `#include` the definitions file near the top of the sketch. *Wire.h* should also be included:

```
#include "si4707_definitions.h"
#include <Wire.h>
```

initSi4707()

This function definition begins on line 194. Its purpose is to initiate communication with the Si4707 and to power up the device. This should be called at the beginning of any sketch that uses the IC.

```

// initSi4707 performs the following functions, in sequence:
// * Initialize all pins connected to Si4707 (SEN, SCLK, SDIO, and RST)
// * Starts up the Wire class - used for two-wire communication
// * Applies a reset signal to the Si4707
// * Sends the Power Up command to the Si4707
// * Sends the GET REV command to verify communication, returns the final
//   two digits of the Part Number (should be 7)
byte initSi4707()
{
  // Set initial pin value: RST (Active-low reset)
  pinMode(rstPin, OUTPUT); // Reset
  digitalWrite(rstPin, LOW); // Keep the SI4707 in reset

  // Set initial pin value: SEN (I2C Address select)
  pinMode(senPin, OUTPUT); // Serial enable
  if (SEN_ADDRESS)
    digitalWrite(senPin, HIGH);
  else
    digitalWrite(senPin, LOW);

  // Set initial pin values: SDIO and SCLK (serial data and clock lines)
  // Wire.begin() will take care of this
  Wire.begin();
  delay(1); // Short delay before we take reset up

  // Raise RST, SCLK must not rise within 300ns before RST rises
  digitalWrite(rstPin, HIGH);
  delay(1); // Give Si4707 a little time to reset

  // First, send the POWER UP command to turn on the Si4707.
  // The Si4707 must be powered up before sending any further commands.
  powerUp();

  // See Si4707_system_functions.ino for info on command_Get_Rev
  return command_Get_Rev(1);
}

```

setWBFrequency() and tuneWBFrequency()

These two functions can be used to change the received frequency of the Si4707. `setWBFrequency(long freq)` (definition begins on line 227) should be used to set the Si4707 to a specific WB frequency. The given parameter should be the desired frequency in kHz.

Directly below that function definition is one for `tuneWBFrequency(signed char increment)`. This function can be used to change the radio's frequency in relation to the currently tuned station. The increment parameter can be a positive or negative number, telling the IC how many 2.5kHz increments you'd like to travel from the current frequency.

```

// This function sets the Si4707 to the freq it receives.
// freq should represent the frequency desired in kHz.
// e.g. 162400 will tune the radio to 162.400 MHz.
byte setWBFrequency(long freq)
{
  // Keep tuned frequency between valid limits (162.4 - 162.5
  5 MHz)
  long freqKHz = constrain(freq, 162400, 162550);

  Serial.print("Tuning to: ");
  Serial.print((float) freqKHz / 1000.0, 4);
  Serial.println(" MHz");

  // See si4707_system_functions.ino for info on command_Tune_
  Freq
  return command_Tune_Freq((freqKHz * 10)/25);
}

// This function does incremental tunes on the Si4707. Send a
// signed value representing how many increments you'd like t
o go
// (up is positive, down is negative). Each increment is 2.5kH
z.
void tuneWBFrequency(signed char increment)
{
  unsigned int freq = getWBFrequency();
  freq += increment;
  freq = constrain(freq, 64960, 65020);

  Serial.print("Tuning to: ");
  Serial.print((float) freq * 0.0025, 4);
  Serial.println(" MHz");

  // See si4707_system_functions.ino for info on command_Tune_
  Freq
  command_Tune_Freq(freq);
}

```

setMuteVolume() and setVolume()

These two functions are useful if you want to implement a volume knob of some sort. `setMuteVolume(boolean mute)` either turns the mute on or off. If the boolean parameter is 1, the Si4707 audio output is muted, 0 turns audio back on.

The `setVolume(int vol)` function sets the volume to any value between 0 (muted) and 63 (max volume).

```

// Depending on the value of the mute boolean, this function will either
// mute (mute=1) or un-mute (mute=0) the Si4707's audio output.
void setMuteVolume(boolean mute)
{
  // Mute left (bit 1) and right (bit 0) channels
  setProperty(PROPERTY_RX_HARD_MUTE, (mute<<1) | (mute<<0));
}

// This function interacts with the RX_VOLUME property of the Si4707.
// Send a volume value (vol) between 0 (mute) and 63 (max volume).
void setVolume(int vol)
{
  vol = constrain(vol, 0, 63); // vol should be between 0 and 63
  setProperty(PROPERTY_RX_VOLUME, vol);
}

```

Advanced Features

The breakout board allows for some customization on the Si4707. You can connect your own speaker and/or add an external antenna.

Using an External Antenna

A jumper on the underside of the Si4707 Breakout Board allows you select between a headphone-wire-antenna, or your own external antenna. By default, the jumper comes configured to select the headphone antenna. If you'd like to switch the jumper, you'll need something sharp to cut the trace—a hobby knife should do the job.

Once the jumper between the middle and *HP* pads is cut, apply a blob of solder between the middle and *EXT* pads.



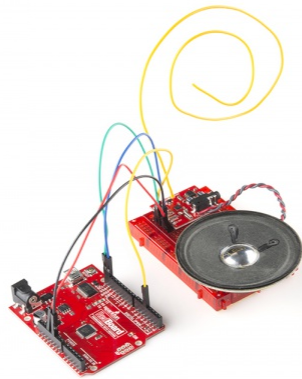
After setting the jumper correctly, you'll have to add an antenna. The simplest antenna—a quarter-wave whip—can be constructed out of a piece of about 46cm-long (18.1 in) wire.

The WB frequency band is just above that of FM radio, so if you can get an FM antenna connected (or even a VHF TV antenna) to the *ANT* pin, that could work too.

Or, a more complex ground plane antenna could be designed following NOAA's specifications.

Using an External Speaker

Both the left and right audio outputs of the Si4707 are broken out to a 0.1"-spaced header. This allows you to bypass the 3.5mm headphone jack and connect a speaker of your own.



The *L* and *R* pins on the side of the board are the outputs of the on-board amplifier. An 8 Ohm speaker gets pretty loud, but you may need to further amplify the signal.

If you're using anything besides headphones (or something connected to the audio jack), you'll need to add an external antenna as described in the section above. By default the board relies on a headphone-wire antenna to bring in the radio signal.

Resources and Going Further

Now that you know how to hook up the Si4707, what project will you be implementing a weather radio into? Need some inspiration? Check out some of these products:

- FabFM Radio Kit - This is a really cool DIY radio kit. Maybe you could swap out the FM module with the Si4707? Or implement both!
- Si4703 FM Tuner Evaluation Board - The Si4703 is the FM-tuning sibling of the Si4707, their breakout boards share a lot of similarities. If you like the Si4707, consider checking out this IC as well.

- USB Weather Board - Why not just make your own weather station?! Compare your measurements with the radio's report. The Weather Board features tons of weather sensors: barometric pressure, humidity, temperature, light, and it connects to our Weather Meter to measure wind speed/direction, and rainfall.

Or some of our other tutorials:

- Using the OpenSegment - The OpenSegment displays are really easy-to-use 4-digit 7-segment displays. You could use one of these displays to print the SAME messages received by the Si4707.
- Heating Pad Hand Warmer Blanket - The weather radio will tell you when to expect the cold. Be ready for it with our very own DIY heating blanket.
- Creating a Humidor Control Box - Be the master of your own weather domain, with a humidor!
- GPS Basics - You could connect the Si4707 to a GPS module and have it tune to stations based on it's current latitude and longitude.

Or, check out our how to use GitHub tutorial and help us make the Si4707 example code better by adding new features!