By Bread Boards & Bill

Project # 1 Scanning Radio AKA Ghost Box

January 23 2022 Part 2

Ghost Box Part 2

Bill Chappell presents this Project, build at your own risk. The author is not responsible for errors or omissions in this document.
This Project uses open-source software "arduino.org"
Find links at www.digitaldowsing.com/diy/
The Sketch for Part 2 can be downloaded at:
Part 2 Arduino code

Ghost Box Part 2

The goal for part 2, familiarize the reader with the software.
Scanning radios, AKA Ghost Boxes, are based on going from one station to another in increments.

The scan range is typically the whole band the radio can receive.
For example, in the US, the FM broadcast radio band is 87.9 - 108.00.

The FM band is divided into 100 0.2 MHz-wide channels.
They are designated channels 201 through 300 by the FCC.

Each station is 0.2 MHz apart. Though not every channel is active in a local area. 87.7 Is considered FM, but classified as channel 200 TV station by the FCC.

For this project, we will scan the active channels for now. So the Scan range will be 87.9 - 107.90 FM.
The scan increment is 0.2 MHz. Finally, the scan rate will be two-hundredths of a second.

We will discuss changing the radio frequency values in a later discussion taking into consideration how they would affect the Ghost Box functions.

Ghost Box Part 2

FCC regulates stations by minimum distance and frequency.

IE:

Stations on the same channel must be 71 miles apart
Stations separated by 200 kHz must be 45 miles apart
Stations separated by 400/600 kHz must be 19 miles apart
Stations separated by 10.6/10.8 MHz must be 6 miles apart

The closer the stations are to each other, the more significant the frequency of the stations must differ.

Link to FM data
https://en.wikipedia.org/wiki/FM_broadcasting_in_the_United_States

FM Band Plan by area
https://en.wikipedia.org/wiki/FM_broadcasting_in_the_United_States

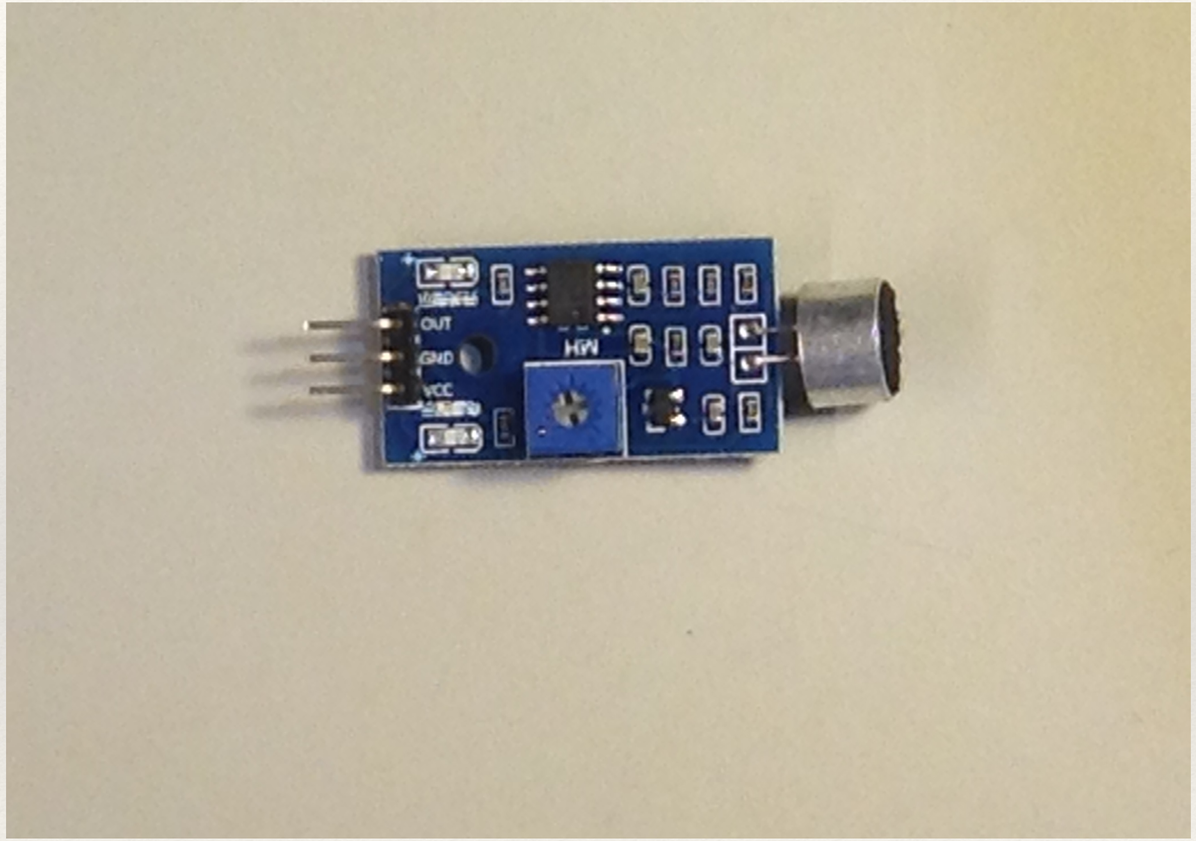
Frequency	Channel	Frequency	Channel	Frequency	Channel	Frequency	Channel
87.9 MHz	200	93.1 MHz	226	98.1 MHz	251	103.1 MHz	276
88.1 MHz	201	93.3 MHz	227	98.3 MHz	252	103.3 MHz	277
88.3 MHz	202	93.5 MHz	228	98.5 MHz	253	103.5 MHz	278
88.5 MHz	203	93.7 MHz	229	98.7 MHz	254	103.7 MHz	279
88.7 MHz	204	93.9 MHz	230	98.9 MHz	255	103.9 MHz	280
88.9 MHz	205	94.1 MHz	231	99.1 MHz	256	104.1 MHz	281
89.1 MHz	206	94.3 MHz	232	99.3 MHz	257	104.3 MHz	282
89.3 MHz	207	94.5 MHz	233	99.5 MHz	258	104.5 MHz	283
89.5 MHz	208	94.7 MHz	234	99.7 MHz	259	104.7 MHz	284
89.7 MHz	209	94.9 MHz	235	99.9 MHz	260	104.9 MHz	285
89.9 MHz	210	95.1 MHz	236	100.1 MHz	261	105.1 MHz	286
90.1 MHz	211	95.3 MHz	237	100.3 MHz	262	105.3 MHz	287
90.3 MHz	212	95.5 MHz	238	100.5 MHz	263	105.5 MHz	288
90.5 MHz	213	95.7 MHz	239	100.7 MHz	264	105.7 MHz	289
90.7 MHz	214	95.9 MHz	240	100.9 MHz	265	105.9 MHz	290
90.9 MHz	215	96.1 MHz	241	101.1 MHz	266	106.1 MHz	291
91.1 MHz	216	96.3 MHz	242	101.3 MHz	267	106.3 MHz	292
91.3 MHz	217	96.5 MHz	243	101.5 MHz	268	106.5 MHz	293
91.5 MHz	218	96.7 MHz	244	101.7 MHz	269	106.7 MHz	294
91.7 MHz	219	96.9 MHz	245	101.9 MHz	270	106.9 MHz	295
91.9 MHz	220	97.1 MHz	246	102.1 MHz	271	107.1 MHz	296
92.1 MHz	221	97.3 MHz	247	102.3 MHz	272	107.3 MHz	297
92.3 MHz	222	97.5 MHz	248	102.5 MHz	273	107.5 MHz	298
92.5 MHz	223	97.7 MHz	249	102.7 MHz	274	107.7 MHz	299
92.7 MHz	224	97.9 MHz	250	102.9 MHz	275	107.9 MHz	300
92.9 MHz	225						

New Hardware

Now it's time to add additional hardware.
This section will require a Microphone
to add the Speak When Spoken to option.

Here's a link to the microphone I'm using.
Allow us to add the speak when spoken to,
feature to the project.

https://www.amazon.com/dp/B00XT0PH10
psc=1&ref=ppx_yo2_dt_b_product_details



New Hardware

The microphone has three leads.

1. VCC, connected to Arduino pin 12
2. Ground, connected to Arduino Uno GND
3. Out, connected to Arduino Uno pin 11.

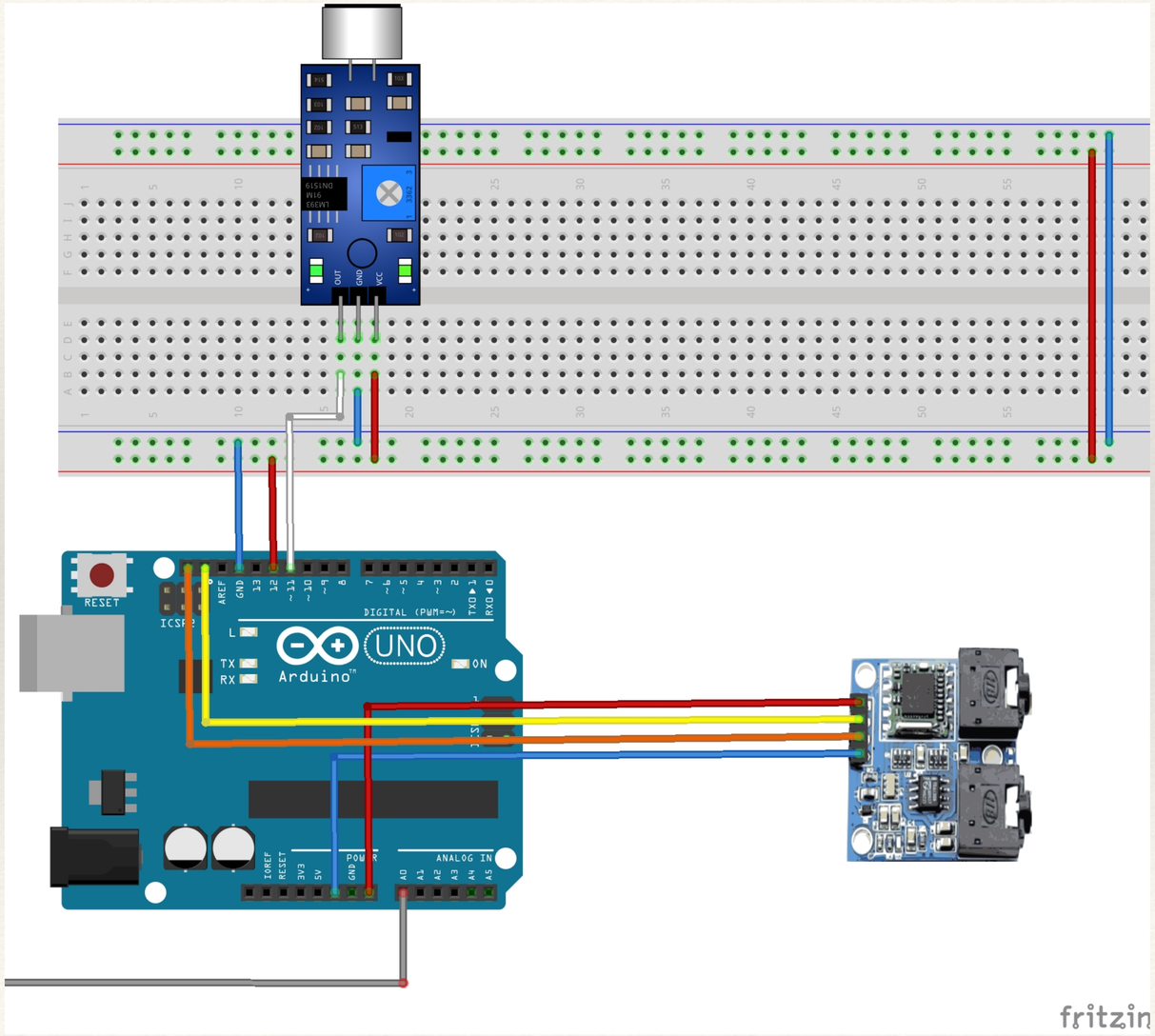
It's time to add a breadboard to the project.

A breadboard will allow for more devices and a cleaner method of connecting items to the Arduino Uno.

In the example code, you will also see an output called BreadBoard. This output is used to power the breadboard when the program is running aids in keeping parts safe from mistakes while setting things up.

One last item needed is a short bare wire 2-3 inches will work fine.

This wire will act as an antenna responding to EMF and Static electricity.



New Hardware

The microphone has three leads.

1. VCC, connected to Arduino Uno 3.3v
2. Ground, connected to Arduino Uno GND
3. Out, connected to Arduino Uno pin 11.

Let's use the breadboard to add this new part.

The Bread Board has power busses top and bottom tied together.

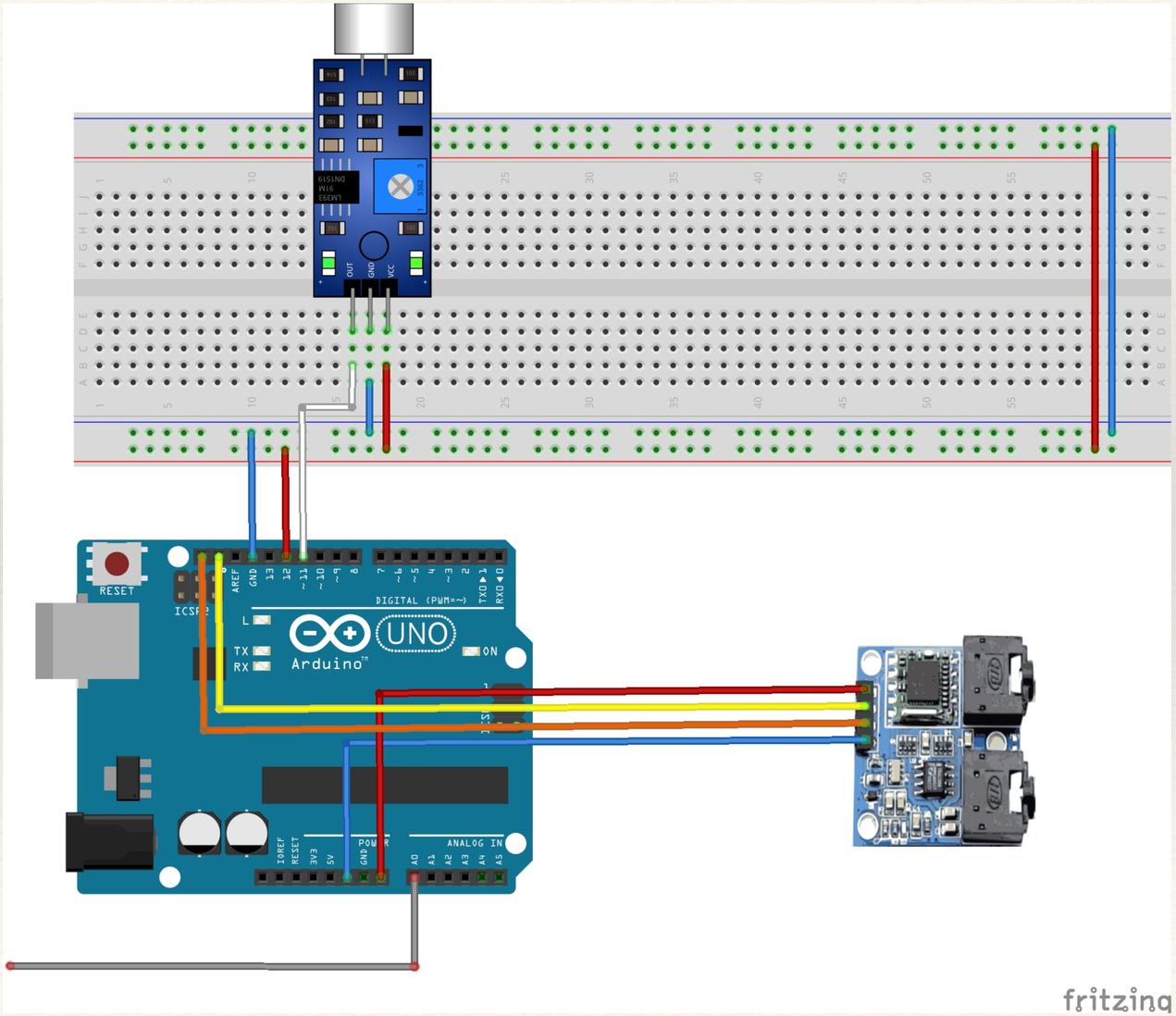
Pin 12 on the Arduino Uno goes to the red bus on the breadboard "+."

GND on the Arduino goes to the Blue bus on the breadboard "GND."

Pin 11 on the Arduino goes to the OUT pin of the Microphone module.

Pin 12 on the Arduino goes to the RED bus on the breadboard "+."

Finally, the short wire 2-3" goes in Arduino pin A0, sticking straight up.

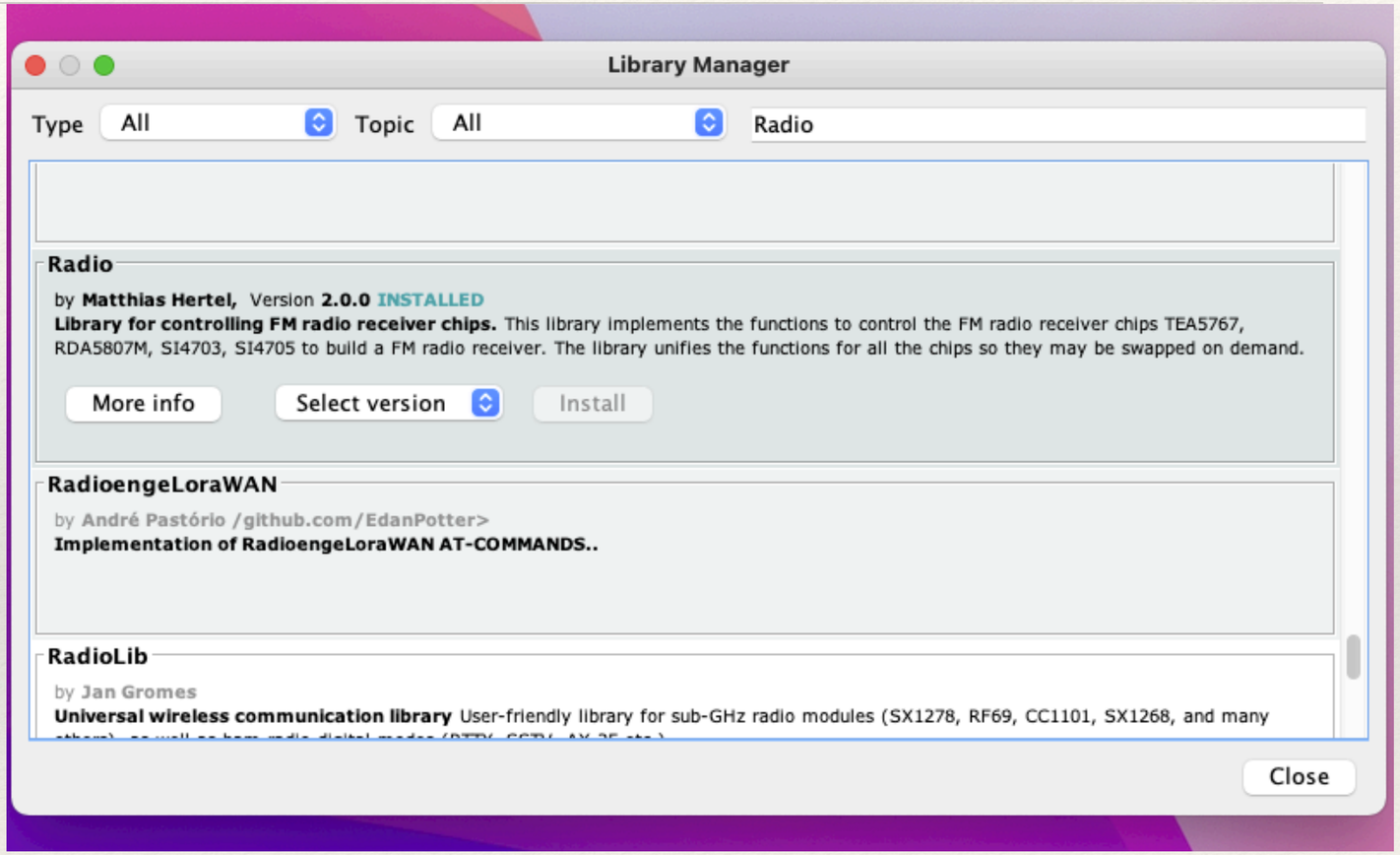


New Software

From the Tools menu, open the Library Manager.
Next, in the Library Manager search bar, enter Radio.
Select the Radio Library and install. Close the Library Manager when done.

Use the link below to download the new Arduino program file
Save it to your computer and open it in the Arduino IDE editor.
The file is in a zip format.

[Part 2 Arduino code]



The Program line by line

It's not my intent to make you a programmer. Instead, I want you to know the program's essential parts and how it works, also, how you can change it!

The sketch is straightforward, so everyone can follow what's going on
for you "gunners" out there, I'll go further from time to time after part 3.
Adding even more features.

Want to learn more about Arduino programming. Youtube has1000's of how-to and instructional videos. Also arduino.org

The Program line by line

```
//*************************************************************************************************
// Ghost Box program scans up then down flashes led on UNO indicates sweeping
// Microphone add and new library Mutes while not scanning
//*************************************************************************************************
/// \file  TestTEA5767.ino
/// \brief An Arduino sketch to operate a TEA5767 chip based radio using the Radio library.
///
/// \author Matthias Hertel, http://www.mathertel.de
/// \copyright Copyright (c) 2014 by Matthias Hertel.\n
/// This work is licensed under a BSD style license.\n
// Added code for simple ghost box project 1.22.22
//*************************************************************************************************
```

The first series of lines have the Library creators copy write and basic information about this program, AKA sketch referred to by the Arduino IDE.

The Program line by line

```
//*****************************************************************************
//  Includes
//*****************************************************************************
#include <radio.h>                  // Library
#include <Wire.h>                   // SLC SDA communications for radio module
#include <TEA5767.h>                // Tells Radio library how to interface the radio module
TEA5767 radio ;                     // Define radio model
```

Includes are files needed to access the library in the sketch the radio library.

The Program line by line

```
//*************************************************************************************************
// Global Variables
//*************************************************************************************************
int LED1 =13;                          // create a name for led1 on the Arduino
int hold =20;                          // hold is the delay time
int inc = 20;                          // How far to move when tuning
int start_FM = 8790;                   // US FM Band start **87.7 excluded
int stop_FM =10790;                    // US FM band stop
int breadboard = 12;                   // power pin to breadboard
int mic = 11;                          // microphone signal
int AntIn = 0;                         // antenna energy value  1 count = 1.1 / 1024 = .0010 volts
```

Global Variables, here we create names for pins on the Arduino or values that we need. This helps make the sketch more readable. read the //comments on each for a brief explanation

The Program line by line

```
//*****************************************************************************************
// Program Setup
//*****************************************************************************************
void setup()
{
  pinMode(LED1, OUTPUT);                    // led1 on board led
  pinMode(bb  , OUTPUT);                    // breadboard power
  Wire.begin();                             // start IC2 communications
  digitalWrite(bb, HIGH);                   // power ON breadboard
  radio.init();                             // Start radio
  radio.setMute(1);                         // Mute ON radio sound off
  delay(100);                               // allow breadboard to power ON allow connected devices time to start
}
```

Program setup, Here we tidy up a bit get things ready for the main section of the program. Setting output pins, starting communications to the radio. Turn on the breadboard power and mute the radio output.

The Program line by line

```
//*************************************************************************************************
// Main program
//*************************************************************************************************
void loop()
{
  delay(10);                                  // allow analog time to settle
  AntIn = analogRead(A0);                     // read energy on antenna
  radio.setMute(1);                           // radio mute ON
  int a = digitalRead(mic);                   // read mic to  variable named a
```

Loop runs the main section of the program. This will take a few pages to get through. At the start, we run a short delay this is required as the program loops to allow the AntIn time to sample. The energy value from the antenna is read and stored in AntIn "Antenna in." Next, the Mute is set to turn off the radio "this will make more sense later.
the last line here looks to see if the microphone is being triggered 1 = yes 0 = no

The Program line by line

```
// to talk or not to talk
if ((AntIn > 340) or (a == 0))              // if AntIn is greater than 340 or a = 1 mic on then scan
{
  while (a == 0)                            // while mic = 1 wait here “mic is hearing sound”
  {
    delay(5);                               // delay 5 ms
    a =  digitalRead(mic);                  // read mic to a
  }
 delay(500);                                // wait 1.5 seconds before responding
 radio.setMute(0);                          // un-mute the radio and allow it to scan
```

We look to see if either the AntIn “Antenna” or the Mic is triggered if both are false. The program resumes at the top of the loop. If either is true, we check to ensure the Mic does not hear sound. Once the Mic is quiet, we delay for a 1/2 second. Next, the radio is UN-muted, and we move to the scan

The Program line by line

```
// for loop and scan from start-FM to stop-FM stepping by inc value in MHz
 for ( int r = start_FM; r <= stop_FM;r = r+ inc)
 {
  digitalWrite(LED1, HIGH);                 // led ON
  radio.setFrequency(r);                    // Set radio to new frequency
  delay(hold/2);                            // wait here for 1/2 of hold
  digitalWrite(LED1, LOW);                 // led OFF
  delay(hold/2);                           // wait here for 1/2 of hold
 }
```

Scan from start-FM to stop-FM scan every .2 MHz the distance each possible channel set for on the FM band. turn on the LED delay for 1/2 the value of hold "time in mills seconds."
Turn off LED delay 1/2 hold again. Allowing the onboard LED to blink every time we scan to the next station. Here we scan up the FM band from the first possible station to the last.

The Program line by line

```
//     for loop and scan from stop-FM to start-FM stepping by inc value in MHz ** -/+ inc to prevent repeat scan
    for ( int r = stop_FM-inc;r >= start_FM+inc;r = r- inc)
   {
    digitalWrite(LED1, HIGH);                    // led ON
    radio.setFrequency(r);                       // Set radio to new frequency
    delay(hold/2);                              // wait here for 1/2 of hold
    digitalWrite(LED1, LOW);                     // led OFF
    delay(hold/2);                              // wait here for 1/2 of hold
   }
  }
}
//*************************************************************************************************
// End of program
//*************************************************************************************************
```

The scan from start-Stop to start-FM scan every .2 MHz, the distance each possible channel is set for on the FM band.
We can scan just like before, except scanning back down the FM band. Turn on the LED delay for 1/2 the hold value "time in mills seconds." Turn off LED delay 1/2 hold again. Allowing the onboard LED to blink every time we scan to the
next station. Here we scan down the FM band, from the last possible station to the first. Finally, the program's end. The loop function will go back to the first line in "Void loop" and repeat without end

Changes

In Variables, FM-Start and FM-Stop can be changed to select just a part of your local FM band.

int hold = 20; // hold is the delay time
Modify this to be how long you want to pause at each scan position IE: hold = 40 or 10

int inc = 20; // How far to move when tuning
Here we can adjust how far we move each scan step. You can tune in-between stations by changing this to .1
int start_FM = 8790 // US FM Band start **87.7 excluded
int stop_FM = 10790 // US FM band-stop
Modify these lines to change the start and stop point IE:9990 “99.9” or 10790 “107.9” to 10390 “103.9”
Allowing the radio to do a shorter scan faster scan. Try picking the most populated second than try the least.
After you have typed your changes, press the right arrow on the upper left of the Arduino IDE to send and start the changes.

Changes

Currently, the sketch will scan from the Fm-Start to Fm-Stop
then scan back up to Fm-Stop. To change this to a simple scan down,
comment this code like bellow: // tells the IDE not to read this, also called a comment.

```
//for loop and scan from stop-FM to start-FM stepping by inc value in MHz

//    for ( int r = stop_FM-inc;r >= start_FM+inc;r = r- inc)
//   {
//    digitalWrite(LED1, HIGH);                    // led ON
//    radio.setFrequency(r);                       // Set radio to new frequency
//    delay(hold/2);                               // wait here for 1/2 of hold
//    digitalWrite(LED1, LOW);                     // led OFF
//    delay(hold/2);                               // wait here for 1/2 of hold
//   }
```

Changes

Currently, the sketch will scan from the Fm-Start to Fm-Stop
then scan back down to Fm-Start. To change this to a simple scan up,
comment this code like bellow: // tells the IDE not to read this, also called a comment.

```
 // for loop and scan from start-FM to stop-FM stepping by inc value in MHz
//  for ( int r = start_FM; r <= stop_FM;r = r+ inc)
//  {
//   digitalWrite(LED1, HIGH);                  // led ON
//   radio.setFrequency(r);                     // Set radio to new frequency
//   delay(hold/2);                             // wait here for 1/2 of hold
//   digitalWrite(LED1, LOW);                   // led OFF
//   delay(hold/2);                            // wait here for 1/2 of hold
//  }
```

Changes

It’s always a good idea to re-save your sketch with a different name when making changes. If something goes wrong, you can get back to what worked and figure out the issue.

Ghost Box Part 2 Wrap Up

At this point, your Ghost Box can scan up and down wait to respond till after you have asked it a question. Respond to the energy around the device.

The sketch can select the range scan from any point on the FM dial: control scan time and scan distance of each scan change.

In Part 3, I'll discuss ways to power your ghost box without plugging it into the computer. Also, discuss adding a display as well.

Having trouble? Did I screw up? Send me a message on Twitter to correct the error for everyone.

Have a great day!
Bill Chappell
BreadBoards & Bill