Bluebox AVR: Esquire Edition
Assembly and Operation Manual
Board Revision 4.2
Manual Revision 7

David Griffith

Monday September 6, 2021

This page is intentionally left blank.

Contents

1 Introduction **4**
- 1.1 What is This Thing? 4
- 1.2 About This Device 4

2 Use **5**
- 2.1 Button Layout 5
- 2.2 Power Up 5
- 2.3 Setting Default Mode 5
- 2.4 Modes 5
- 2.5 Memory 8

3 Construction **9**
- 3.1 Bill of Materials 9
  - 3.1.1 Onboard Parts 9
  - 3.1.2 Other Parts 9
- 3.2 Preparing the Enclosure 9
- 3.3 Starting Assembly 10
- 3.4 Buttons 10
- 3.5 Batteries 10
- 3.6 Testing 11
- 3.7 LED Resistor Choice 11

4 Firmware **13**
- 4.1 Build Tools 13
  - 4.1.1 Unix, Linux, BSD, etc 13
  - 4.1.2 Windows 13
  - 4.1.3 Macintosh 13
- 4.2 Compilation 13
- 4.3 Programming 14

5 Installation **15**
- 5.1 External Power 15
  - 5.1.1 Voltage Converters 15
- 5.2 Audio Output 15
- 5.3 Power Switch 17

6 Revision History **18**
- 6.1 Board Revisions 18
- 6.2 Manual Revisions 18

APPENDIX **20**

A Resistor Color Code Chart **20**

B Schematic **21**

C Grille Pattern **22**

D Assorted Webpages **23**

COLOPHON **24**

1 Introduction

1.1 What is This Thing?

This is a reimplementation in KiCad of Don Froula's PIC-based bluebox. The circuit was by Don Froula and the board layout was by Phil Lapsley, author of "Exploding the Phone". It is so named because of Don's use of his design to produce a close replica to the bluebox pictured in the October 1971 Esquire article "Secrets of the Little Blue Box".

What's a bluebox? A bluebox is a device originally designed to generate in-band signalling tones to control long-distance telephone exchanges. Blueboxes no longer work because modern switching systems no longer use in-band signalling. The purpose of this board is for historic interest and for use on privately owned switching systems set up specifically to allow blueboxes to function. Project MF is a set of patches to the open-source Asterisk PBX software to support in-band signaling such that a bluebox will work.

More history on blueboxes can be found at Don Froula's website at http://projectmf.org/, on Wikipedia (see https://en.wikipedia.org/wiki/Blue_box), and in the book "Exploding the Phone" by Phil Lapsley. There are many other resources too numerous to list here.

1.2 About This Device

This device uses an AVR microcontroller (ATtiny85) to monitor buttons, generate tones, and blink LEDs. There are thirteen buttons – one for the 2600 Hz tone and the rest as a standard telephone keypad. There are twelve memory locations for storing tone sequences, one for each of the regular buttons. Each memory location stores the tone mode along with a sequence of keystrokes.

There are ten tone modes:

1. Bluebox
2. DTMF
3. Redbox
4. Greenbox
5. 2600 Pulse
6. AC1
7. AC9
8. IMTS ANI
9. IMTS Dial
10. MTS

2 Use

This section details how this particular device operates. How to actually make use of a bluebox and the other modes that this device implements is beyond the scope of this document.

2.1 Button Layout

The device has thirteen buttons. Twelve are arranged in four rows of three columns each. These buttons are used to play tones according to whatever the current tone mode is. They are are also used to save a sequence into memory locations and play back the sequences.

Above the second button in the top row is a button by itself. This is the 2600 button, which is used to play the 2600 Hz tone for seizing a trunk and is also used to change the power-up default tone mode and to toggle from normal and memory playback modes.

2.2 Power Up

When powered up with no buttons being held, the device will emit a tone of 1004 Hz[1]. This indicates that the default tone mode is active. Holding buttons 1 through 5 on power-up will select a tone mode. Holding the star button (*) will select a slow tone length (120 milliseconds). Holding the hash button (#) will select a fast tone length (75 milliseconds). Holding down any button while powering up the device will also cause a 1400 Hz tone to play. Holding a button corresponding to a mode that isn't implemented will cause two 880 Hz beeps to be played and the device will continue on as if no button was held.

2.3 Setting Default Mode

If the 2600 button is held when the device is powered up, a 1700 Hz tone is played. The next button pressed will set a default tone mode or tone length and play a 1400 Hz tone. Buttons 1 through 5 will select a tone mode. The star button (*) will select a slow tone length (120 milliseconds). The hash button (#) will select a fast tone length (75 milliseconds). Holding a button corresponding to a mode that isn't implemented will cause two 880 Hz beeps to be played and the device will continue on as if no button was held. The default tone mode is stored in EEPROM location 0x01. The default tone length is stored in EEPROM location 0x02. Whenever EEPROM is written to, a short high-low chirp is heard.

2.4 Modes

There are ten tone modes:

1. **MF**: A standard bluebox. KP appears on the * key. ST appears on the # key.

1	2	3
4	5	6
7	8	9
KP	0	ST

[1] This is a reference to the telco digital milliwatt test signal frequency.

2. **DTMF**: Standard DTMF dialing tones. All landline phones with buttons use these tones to signal the central office which number the customer wishes to call. These tones are also used to interact with a computer on the other end of the line that's set up for such interaction. This mode is useful when using a rotary phone or any other application that uses DTMF tones to control something. For that reason, handheld devices that emit DTMF tones were sold when rotary phones were still common to allow people using rotary phones to use such applications.

1	2	3
4	5	6
7	8	9
*	0	#

3. **Redbox**: Redbox tones. These modes were used for fooling a payphone into thinking that coins had been deposited when they really weren't. This worked because payphones themselves would play these tones to the central office to indicate what coins were deposited. This mode stopped working when payphones didn't open up the microphone until the call was actually started. Also, COCOTs (Customer Owned Coin Operated Telephone) were never prone to this because they always connected to regular phone lines rather than special payphone-only lines.

US nickel	US dime	US quarter
Canadian nickel	Canadian dime	Canadian quarter
UK 10 pence	UK 50 pence	

4. **Greenbox**: Greenbox tones. These were for manipulating payphones from the phone called with said payphone. The first two rows have a 2600 Hz wink. The next two rows have a 900 Hz + 1500 Hz wink.

Coin Collect	Coin Return	Ringback
Operator Attach	Operator Release	Operator Release and Coin Collect
Coin Collect	Coin Return	Ringback
Operator Attach	Operator Release	Operator Release and Coin Collect

5. **2600 Hz pulse**: Emits 2600 Hz pulses according to the number as on a rotary dial (0 is 10 pulses). This mode predates MF tones. This was how John Draper (aka Cap'n Crunch) and Joe Engressia Jr. (aka Joybubbles) were able to phreak using a whistled 2600 Hz tone.

1	2	3
4	5	6
7	8	9
	0	

6. **AC1**: AC1 is a very old pulse dialing tone system used in the UK. In general, UK trunk systems used short spurts of tone to signal supervision, rather than the continuous 2600Hz tone used in the US. he AC1 system clears the trunk by sending two sequential tones (the * key on the box), then reseizes the trunk with a short burst of one of the tones (the # key on the box). Thereafter, digits are sent using pulses of tone. There is no KP or ST tone equivalent. The end of dialing is determined at the switch by timing the delays. In the UK, the system would split the line during trunk signalling so that the caller's forward audio was blocked from the system during trunk signalling. However, hybrid circuit leakage was usually sufficient to allow the tones to be heard by the distant switch.

1	2	3
4	5	6
7	8	9
Seize	0	Clear Forward

7. **AC9**: AC9 is a somewhat newer pulse tone dialing system used in the UK. It uses a single tone for all signals, unlike the AC1 system which used two tones. Otherwise, operation is identical to the AC1 system.

1	2	3
4	5	6
7	8	9
Seize	0	Clear Forward

8. **IMTS – ANI (Mobile Authentication**: This was the system used by the pre-cellular IMTS system to authenticate a mobile to the base station, which would then return a dial tone to the phone. The user would then used the IMTS Dialed Digit Pulsing mode to dial the number. The authentication number was simply the area code plus the last four digits of the mobile's telephone number. In some areas, this was changed to use all ten digits of the mobile's number. The system used a parity systen, encoded into tones, which allowed for error checking of the ANI sequence by the base station. On an error, the caller would be routed to an operator. Spoofing an IMTS system involved playing the tones into a radio transmitter microphone at the correct times. Typically, one would tune his radio transmitter to the receive frequency of the IMTS base station channel that was transmitting an idle tone on its transmit frequency (the system was full-duplex, with separate frequencies for receive and transmit – no push to talk operation). He would then key his transmitter, transmit the ANI sequence, unkey, and listen for a dial tone. He would then re-key the transmitter and send the dialed digit tones to complete the call. After the call, the user would send the disconnect tone (* key on the box) to hang up.

   It is almost impossible to manually dial an IMTS ANI sequence manually with acceptable timings. Therefore, it is necessary to store the sequence to a box memory. When played back, the box assures all timing are correct. The sequence to store is D+7 or 10 digit ANI number.

1	2	3
4	5	6
7	8	9
Hangup	0	Seize

9. **IMTS - Dialed Digit Pulsing** This is the mode used to send the dialed digits after IMTS authentication, as described above. The # key plays the channel seizure sequence (not really useful except when played before an ANI sequence). The * key plays the disconnect tones, which will end the call.

1	2	3
4	5	6
7	8	9
Hangup	0	Seize

10. **MTS – Dialed Digit Pulsing (Secode/GE System) – No authentication**

    This was the half-duplex, push to talk pre-IMTS radio telephone system. It was completely unauthenticated. The user simply keyed the radio and played the connect tone (# key on the box). The base station then returned a dial tone. The user would then key the transmitter and dial the digits. When the call was finished, the mobile user could send a disconnect tone (* key on the box) to end the call.

1	2	3
4	5	6
7	8	9
Disconnect	0	Connect

2.5 Memory

The device tracks the last 40 keystrokes after powerup or toggle from playback mode. That is, to clear the buffer, powercycle the box or toggle in and then out of playback mode. There are twelve memory locations – one for each of the numeric keys and the star and hash keys.

To toggle to or from playback mode, press and hold the 2600 key for two seconds. A low-high chirp will be played when going into playback mode and a high-low chirp will be played when going back into normal mode. The bluebox always powers up in normal mode.

To save a sequence of keystrokes, first make sure you're not in playback mode. Enter the desired sequence, then press and hold a key other than 2600 for two seconds. A short-long chirp will be played to indicate that the sequence and tone mode has been saved. This means you can have an MF sequence in one memory, a DTMF sequence in another and so on. Sequences cannot be saved when in playback mode.

To play back a sequence, first toggle the bluebox into playback mode. Then press the key for the desired memory location. The sequence will then be played back using the tone mode the bluebox was in when the sequence was saved. Tone length is not saved – it will be whatever length has already been set. See Section 2.2.

To clear a memory location, first clear the keystroke buffer, then press and hold the key for the memory location you want to clear.

3 Construction

This board uses all through-hole parts. Parts are mounted on both sides, which can be a bit confusing, particularly because some parts must be soldered in a specific order or else other parts will be impossible to mount.

3.1 Bill of Materials

3.1.1 Onboard Parts

Location	Description	Value
R1 - R14	0.25W resistor [a]	1KΩ
R15	0.25W resistor	100KΩ
R16, R17	0.25W resistor	150Ω
R18, R19	0.25W resistor	See 3.7 on choosing values
C1	radial capacitor [b]	22μF
C2	ceramic capacitor [c]	0.1μF
C3, C4	ceramic capacitor	See notes on crystal in Section 3.3
C5, C6, C7	ceramic capacitor	0.33μF
X1	HC49-U low-profile crystal	20 MHz
D1	DO-41 rectifier diode	1N4005
IC1	DIP8 AVR microcontroller	ATtiny85
LED1, LED2	1.8mm miniplast LED	See 3.7
SW0 - SW12	6mm tactile switch, 4.4mm stem height	TE/Alco FSM11JH or similar
B1, B2	CR2032 cell holder (see Section 3.5)	Keystone 103, 1026 or MPD BH800S
CONN1 - CONN3	2x1 0.1-inch pitch right angle header	
CONN4	2x3 0.1-inch pitch vertical header	

[a] ordinary axial package
[b] electrolytic or tantalum
[c] all ceramics have 0.1 inch lead pitch

3.1.2 Other Parts

Designation	Description	Notes
Enclosure	Hammond 1591 XXM	
AAA Battery Holder	Keystone 2482 or equivalent	Optional (see Section 3.5)
Power Switch	Miniature or sub-mini SPST switch	E-Switch EG1201 or similar
Audio Jack	3.5mm switched mono phone jack	Switchcraft 35PM2A or similar
Internal Speaker	Small flat speaker about 16mm dia.	
External Speaker	"500" type handset speaker (see Section 5.2)	NS-E004, HHMR-80 or equivalent

3.2 Preparing the Enclosure

Before you attach anything to the board, you must first drill holes in the lid of the Hammond 1591 XXM enclosure for the buttons. Notice that there are holes in the very middle of each of the button footprints. These are guides for drilling pilot holes for the buttons. Screw the board to the lid of the enclosure using 4-40 1/4" self-tapping screws. Mount a 1/16" drill bit in a Dremel or drill press and carefully pass the bit through the hole in the center of a button footprint and drill through the enclosure lid. It's very important

that this be done accurately for the actuators of the buttons to not bind. Otherwise you'll have to carve at the holes or end up with a way too large a hole than you'd otherwise need. Once the pilot holes are drilled, enlarge the holes one step at a time until you reach a diameter of 9/64". At this point, the board can be populated. Any adjustments to the holes should be made as each individual button is soldered. See Section 3.4 for details.

3.3 Starting Assembly

Start with soldering resistors R1 through R10, R12, and R13. These are all 1KΩ and mounted on the "back" where the buttons will by mounted. Clean off flux. Flip the board over to the "front" and solder R11 and R14. These are also 1KΩ.

Solder the resistors R15, R16, and R17 next followed by all the ceramic capacitors (C2 through C7). Then solder the crystal at X1. Capacitors C3 and C4 must match the load capacitance for the crystal. Solder the diode at D1 and the polarized capacitor at C1. Position C1 such that it can be laid down slightly if necessary. Add the connectors CONN2 and CONN3. Then add a socket for IC1. Finally add the 2x3 header at CONN4.

Clean flux off again and flip the board over to the "back". Now the buttons will be mounted.

3.4 Buttons

The buttons are standard 6mm tactile switches with a 4.4mm stem height. The body plus the stem should be about 8mm. When mounted in the lid of the enclosure, the tips should protrude about 1.5mm through 9/64" holes (about 3.6mm). The holes may need to be enlarged slightly to 5/32" if they were not drilled accurately enough. The switches should all be inserted, but not soldered, so as to double check that the holes in the lid were drilled accurately. Slight play may allow bound-up switches to unbind. Don't screw the board to the lid at this point, as you will be constantly removing the board to check for fit. When satisfied that a switch can be pressed smoothly without binding, solder it, and continue with the rest of the switches.

If you will be using CR2032 cells to power the device, clean flux off the side opposite the switches now because you will not be able to do so after cell holders are installed.

3.5 Batteries

The board was designed to accept power either through CR2032 cells mounted onboard or from a 6-volt supply mounted offboard. You have a choice of using CR2032 cell holders that accept one or two cells each. A Keystone 103 accepts a single CR2032 cell, is inexpensive, is very easy to get, and several drop-in replacements are available. A Keystone 1026 or MPD BH800S will each accept two cells and have the same footprint as a Keystone 103.

If Keystone 1026 holders are used, you will need to file or grind down the edges where they face each other and the sides of the enclosure as they are slightly too wide to fit. This is not a problem for the MPD part.

If using cells in holders mounted on the board, J1 must be set according to the number of cells in each holder you use. Header CONN1 should not be installed. For one cell in each holder (two cells total), solder jumper wires between the middle pins to each of the upper pins. For two cells in each holder, solder jumper wires between the middle pins to the lower pins. See Figure 1.

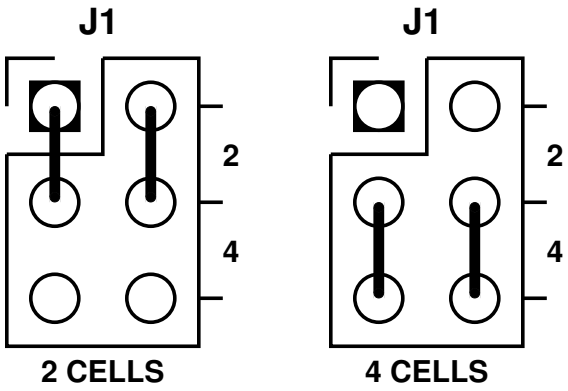


Figure 1: Cells Select Jumpers

The cell holders are mounted on the side opposite the switches, covering some of the switch pins. This means that the switches must be installed first. The pins for the cell holders appear between switches. For this reason, a narrow soldering tip is strongly recommended.

If you want to supply power from offboard, do not mount B1 or B2. Instead, install CONN1 and connect power there, making sure correct polarity is observed. Do not add jumper wires to J1. See Section 5.1 for details on preparing an external cell holder.

3.6 Testing

Now is time to check your work. You should not have applied power to the board yet. Clean the board of flux, taking care not to get any liquids on the bodies of the switches. Let it dry. Then examine the board under magnification for soldering errors. Using a multitester and referring to the schematic in Appendix B, look for shorts between ground and Vcc. Look for pairs of holes where there is a connection where there should not be and vice versa.

Now apply power using whatever means you choose and turn on the switch or put a jumper across the switch header. Don't plug the microcontroller into its socket yet! Clip the ground of your multitester to the ground of the board and check the voltage at various points with the positive probe. The most important of these is pin-8 of IC1 (the upper-right). That one should be close to five volts. You should see six volts before the diode.

3.7 LED Resistor Choice

Stop assembly here. By this point, you should have mounted everything except for the LEDs and resistors R18 and R19. These should not be mounted until after the AVR microcontroller has been programmed. Otherwise, you won't have an easy way to determine if you have a combination of LED and limiting resistor that you like. Go now to Section 5 and prepare an off-board battery holder (if that's what you want), a power switch, and a speaker (jack, internal speaker, or both). Proceed through that and on through Section 4 to build and install the firmware. Then return to this section.

The LEDs can be mounted on either side of the board, depending on personal taste. If mounted on the front, that is, facing towards the user, they must be short enough to fit between the board and the lid of the case. I found that a 1.8mm miniplast LED fits the bill perfectly. The choice of current limiting resistor

is very subjective and varies according to LED choice. Below is a table of characteristic on some LEDs I bought off eBay.

Color	**wavelength at 20mA**	**Forward V Typ / Max**	**Viewing Angle**	**Resistor Value**
Red	624nm	2.0 / 2.4	30	330Ω
Yellow	589nm	2.0 / 2.4	30	330Ω
Green	525nm	3.0 / 3.4	30	1kΩ
Blue	468nm	3.0 / 3.4	30	4kΩ
White	x=0.29, y=0.28	3.0 / 3.4	30	not tested

Experiment with different resistor values to determine the ideal values for R18 and R19. This can be done with a pair of resistor substitution boxes. If you have just one, just testing one LED should give reasonable results.

Connect one end of your substitution box or boxes to pin 6 of the ISP connector (CONN4). Then connect the short pins of the LEDs to your boxes. Then connect the long pins of the LEDs to pin 1 of the ISP connector.

If you have no substitution boxes, here's a simple setup on a small breadboard:

Connect a wire or male-to-male jumper from the positive rail to one of the horizontal rails. Connect a resistor from the negative rail to the horizontal rail just below the first horizontal rail. Plug in an LED across these two rails with the flat side facing the resistor. Set up another LED just like this a short distance below the first.

Connect pin 1 of the ISP connector to the positive rail of the breadboard with a female-to-male jumper wire. Do the same for pin 6 but connect it to the negative rail.

Now apply power to the board and press some buttons. Store something into memory and play it back. Check to see if the brightness of the LEDs satisfy you. If you're using the suggested translucent blue case, hold the case over the LEDs to make sure it looks good. Switch out the resistors for different values if you don't like the brightness. Higher resistance will make the LEDs dimmer.

When you have arrived at the ideal resistance, take two 1/4 watt axial lead resistors and solder them into positions R18 and R19. Then insert your choice of LEDs into positions LED1 and LED2 taking care that the short lead goes in the square pad. If the LEDs will be on the same side as the buttons, check to make sure they will fit. Now solder them into place.

4 Firmware

The ATtiny85 microcontroller needs to be programmed with the Bluebox AVR program in order for the board to work as a multifunction bluebox.

4.1 Build Tools

4.1.1 Unix, Linux, BSD, etc

Git is required to download the firmware to your computer for compilation. GCC-AVR, libusb-dev, avr-libc, and make are required to compile the firmware. Avrdude is required to control an AVR programmer to get the compiled binary into the AVR microcontroller on the circuit board. If you're on a modern Unix computer, you can probably have the package management system install all of these tools for you.

4.1.2 Windows

If you're on Windows, you should install WinAVR from https://sourceforge.net/projects/winavr/ and git from https://git-scm.com/download/win.

4.1.3 Macintosh

Yes, MacOS is a Unix now (well, BSD, really). All the packages described above in Section 4.1.1 can be installed through Fink (see http://finkproject.org/), which uses the same tools as are used in several Linux distributions

If you don't want to do that, you have other options.

First, download and install Git from https://git-scm.com/download/mac

Then choose one of these which appear in order of recommendation:

1. Install Crosspack for AVR from https://www.obdev.at/products/crosspack/index.html
2. Install XCode from your MacOS X install CD or from http://developer.apple.com/tools/. Then install OSX-AVR from https://sourceforge.net/projects/osxavr/.
3. "By hand" compilation. See http://www.ladyada.net/learn/avr/setup-mac.html

4.2 Compilation

Using Git, clone the git repository for Bluebox AVR: https://gitlab.com/DavidGriffith/bluebox-avr.git

This will download the source code and the entire history of the development of the firmware.

Go into the `bluebox-avr` directory that was created and type `make hex`. This should compile the firmware without complaint. You will get two files: `bluebox.hex` and `bluebox.eep.hex` containing the program binary and the initial contents of the EEPROM respectively.

4.3 Programming

You will need a hardware device to write the compiled AVR program into the AVR chip. This board and firmware were designed using the USBtinyISP from Adafruit https://learn.adafruit.com/usbtinyisp). That one is easily available and inexpensive. Sparkfun has something similar at https://www.sparkfun.com/products/9825. Atmel previously offered the STK500 and AVRISP series of programmers, which are now widely cloned. A newer one, the STK600 is currently offered by Microchip, the new owners of Atmel. All of these should be usable. The important thing is that `avrdude` must be able to use it. You can get a list of programmers supported by `avrdude` by issuing the command `avrdude -c?`. If you are using a programmer, you will need to edit the `ISP` line to match your programmer.

If you put cells into the holders or have connected an external battery holder to the board, remove them now. The board should not have power applied to it when the microcontroller is being programmed.

Connect the 6-pin cable to the ISP header, taking care to make sure pin 1 of the cable goes to pin 1 of the header. Then type `make program`. You may want to disconnect the speaker cable or insert a 3.5mm cable with nothing on the other end while doing this because a loud and continuous screeching noise will issue from the speaker during the programming.

After the build process reports success, add your batteries and connect the power switch and speaker. Turn on the unit and you should hear a 1004 Hz tone. This indicates that the unit has started up successfully into the default tone mode (right after programming, this would be MF) and is waiting for a button to be pressed. Now go back to Section 2 and go over how to use your Esquire Edition Bluebox AVR. Once you're satisfied that the device is working properly, you can go back to Section 3.7 and try out LEDs and resistors to arrive at the perfect combination.

5 Installation

When the device is mounted in its enclosure, the "front" is the lid. The "top" is the face adjacent to the edge closest to the 2600 button. The "top" will be one of the shorter faces with the "bottom" opposite. The "sides" will be one of the longer faces perpendicular to the "top" and "bottom". The "floor" is where the PCB mounting risers are.

5.1 External Power

If you will be supplying power to the board from an external power source, this is typically from a plastic 4-cell AAA holder glued to the bottom of the enclosure. The PCB mounting posts in the box should be cut off within 2mm or so of the floor and leveled off with a Dremel using an appropriate routing or grinding bit.

The cell holder should be mounted on the floor close to the bottom pair of posts for screwing down the lid. They should be far enough away from these posts such that the cell holder does not touch any of the components mounted below the 6mm tactile switches.

Some plastic 4-cell AAA holders are slightly too long to neatly fit between the sides of the Hammond 1591 XXM enclosure. If this is the case, use a disc sander or a sheet of sandpaper laid upon a hard flat surface to sand off just enough width to allow the holder to slide into place. A Dremel with a sanding drum is usually incapable of doing a good job and trying to use a file is infuriatingly ineffective.

A Keystone 170 is made mostly of aluminum and will readily slip into place. However, each cell has its own terminals. This means you will need to wire up the holder such that cells are in series. It is also fairly expensive.

To whatever kind of cell holder you choose, you may need to add two wires, black (-) and red (+), and then crimp on female header socket terminals and insert them into a two position header socket. This can then mate with CONN1.

The cell holder may be secured with screws or adhesive, preferably epoxy or silicone RTV. Hot melt or cyanoacrylates may work, but are not recommended. Thin double-sided tape may also work. Double and triple check the position of the holder with cells inside will not touch anything on the board.

5.1.1 Voltage Converters

In the interest of keeping this project's board strictly through-hole, I did not consider using DC-DC voltage converters to allow for fewer than four AAA cells and thus save space. There exist very tiny breakout boards that can be had for a dollar or two. One of these can be inserted in the positive wire going from the cell holder to the main board. The converter can then be secured in the enclosure with glue or double-sided tape.

5.2 Audio Output

A 3.5mm (1/8-inch) jack can be installed in the wall of the enclosure and connected to CONN3 to provide audio output. The most convenient point for putting the jack is in the middle of the top face. A monaural jack is preferable, but a stereo one will work if you solder the tip and ring connections together. A monaural plug will work in both a stereo or monaural jack.

For an internal speaker, a flat one measuring around 16mm in diameter is ideal. This should be mounted on the floor near the top and pointing downward. Take care to avoid contact with the output jack (if present)

or parts of the board. Drill some holes in the enclosure where the speaker will be mounted to allow sound to exit. See Appendix C for a grille pattern. A rubber gasket with an inner diameter of 29mm, outer diameter of 42mm, and thickness of 3.5mm can be glued to the outside around the holes to imitate similar rubber rings found on pocket dialers.

If you use a switched jack, then you can have an internal speaker installed at the same time. A wiring diagram for that is in Figure 2.

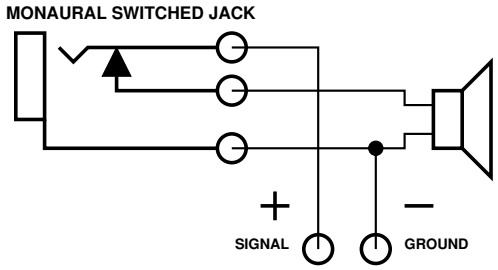


Figure 2: Audio Wiring

Historically, the preferred external speaker has been a 150-ohm earpiece from an old "500" type telephone handset. These were used by Western Electric, ITT and other phone manufacturers for decades. These are still commonly available with part numbers like "NS-E004" and "HHMR-80". A small component called a varistor is commonly found soldered between the contacts. Be sure to remove this or else sound will be distorted.

Take a 1 meter (6 feet) cable with two-conductor 3.5mm plugs on both ends. Cut the cable in the middle. Strip the cable and wires within and attach to your speaker. Apply silicone RTV over the contacts and rear of the speaker to protect the connections and secure the cable. Save the other half of the cut cable for making something else

The volume coming from a 500-type telephone speaker is quite adequate for playing into a telephone receiver. Depending on your choice of internal speaker, some amplification may be necessary. A small board can be made up from perfboard and connected between the 3.5mm jack and the internal speaker. It can then be affixed somewhere convenient within the bluebox's enclosure. There's plenty of room. It uses an LM386 op-amp, a 10KΩ variable resistor, and a 1000 μF capacitor. The schematic is shown in Figure 3.

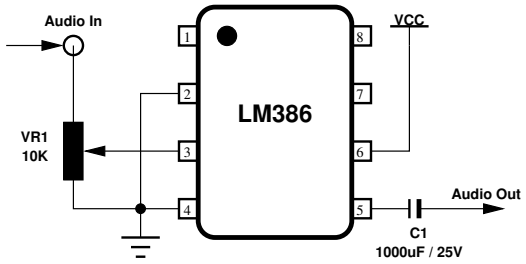


Figure 3: Simple Amplifier

Adding a capacitor of no more than 10μF across pins 1 and 8 will boost the gain to almost 200. When doing this, a 0.1μF bypass capacitor should be placed between pin 7 and ground to avoid problems with oscillating and clipping. This slightly more complex amplifier schematic is seen in Figure 4.

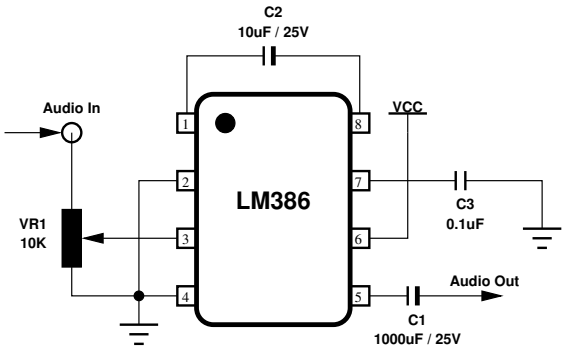


Figure 4: Slightly More Complex Amplifier

5.3 Power Switch

The power switch can be a simple toggle switch or a slightly fancier slide switch. A toggle switch requires just a simple hole drilled in the side. A slide switch requires more effort to first drill a hole and then expand it to a rectangle of the correct shape and size with needle files. In any case, the switch should be an SPST type. Take care to position the switch such that it won't touch the circuit board or anything on it.

If you use a slide switch, the best means of mounting it seems to be with a fast-setting glue. I tried 5-minute epoxy, but ended up with epoxy flowing into the switch and ruining it. Then I tried a gelled cyanoacrylate applied very sparingly. This set instantly and avoided problems with glue getting where it wasn't wanted.

The current going through the switch at any one time is no more than 30mA. Connect the switch to CONN2.

6 Revision History

6.1 Board Revisions

Board revisions prior to 4.0 were never actually produced.

4.0
- Ten copies.
- First run of boards.

4.1
- Ten copies.
- Re-done because PCB manufacturer botched the silk screening.
- Fixed error with J1 “cells select” jumpers.

4.2
- Thirty copies.
- Rework trace routing to keep traces farther away from board edges.
- Scoot parts to the left of CONN4 further to the left to allow for more room for a programming connector to be attached.
- Rework silk layers.

6.2 Manual Revisions

0 October 3, 2018.

- Initial release

1 October 17, 2018.

- Minor typos fixed.
- Briefly expanded on a few things.

2 November 18, 2018.

- Minor typos fixed.
- Added missing reference to C7 in BOM.
- Mention voltage converters.

3 November 25, 2018.

- Added memory subsection forgotten by accident.
- Mention avr-libc for required packages.

4 May 20, 2019.

- Clarify LED package type.

5 June 20, 2019.

- Now using integers for manual releases.
- Change startup tone from 440 Hz to 1004 Hz.
- Mention internal speaker in BOM.

- Added amplifier circuit.
- Assorted cleanup.

6 May 2, 2020.

- Add speaker grille drill pattern to appendix.
- Adjust centering of appendicies.
- Slightly reword button cell holder discussion.
- Add documentation for five new modes: AC1, AC9, IMTS-ANI, IMTS-dial, and MTS
- Tweak and expand explanations of modes.

7 September 6, 2021

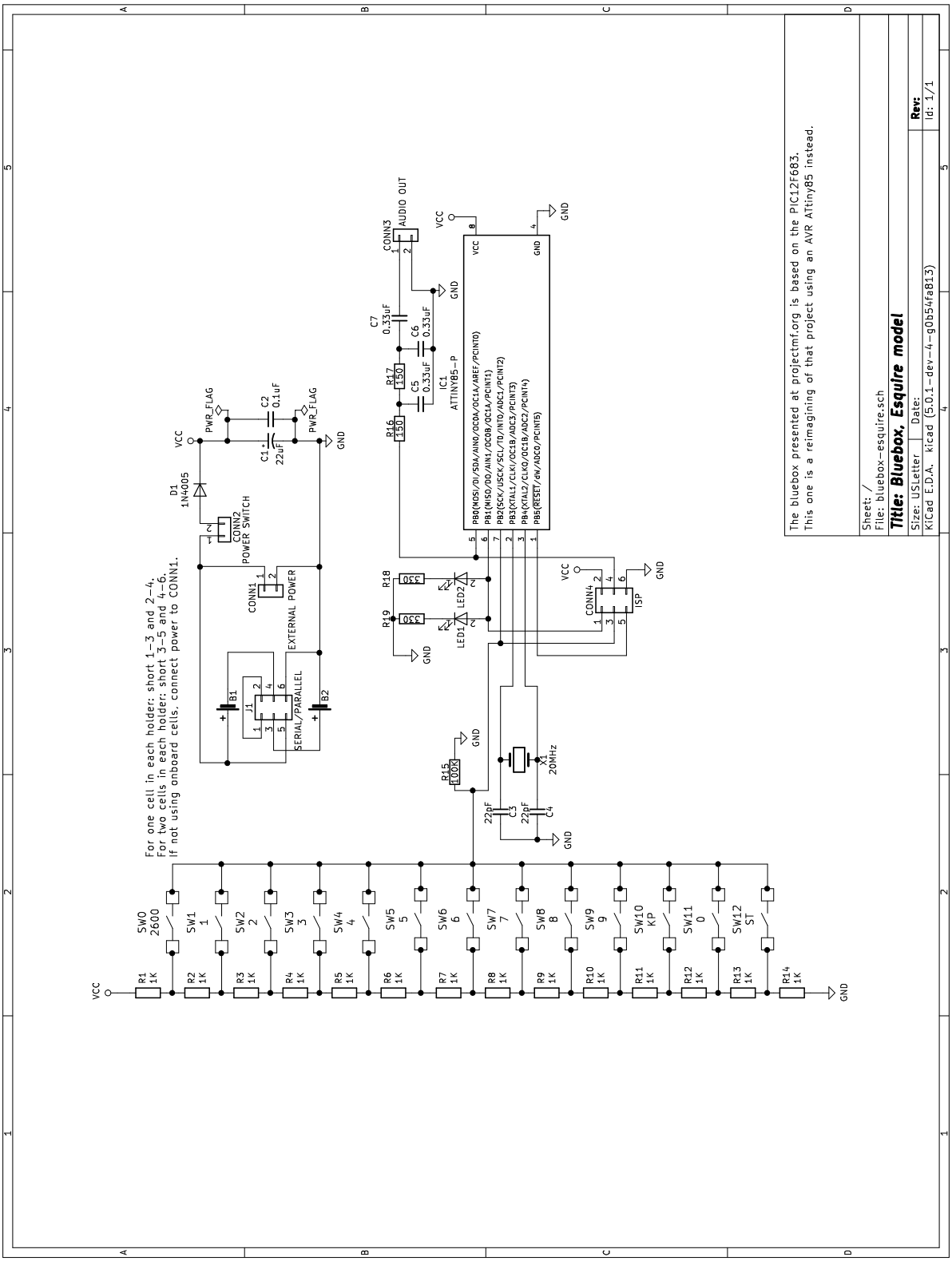
- Corrected mistaken references to Hammond enclosure model 1592 XXM. Should be model 1591 XXM.

APPENDIX

A Resistor Color Code Chart

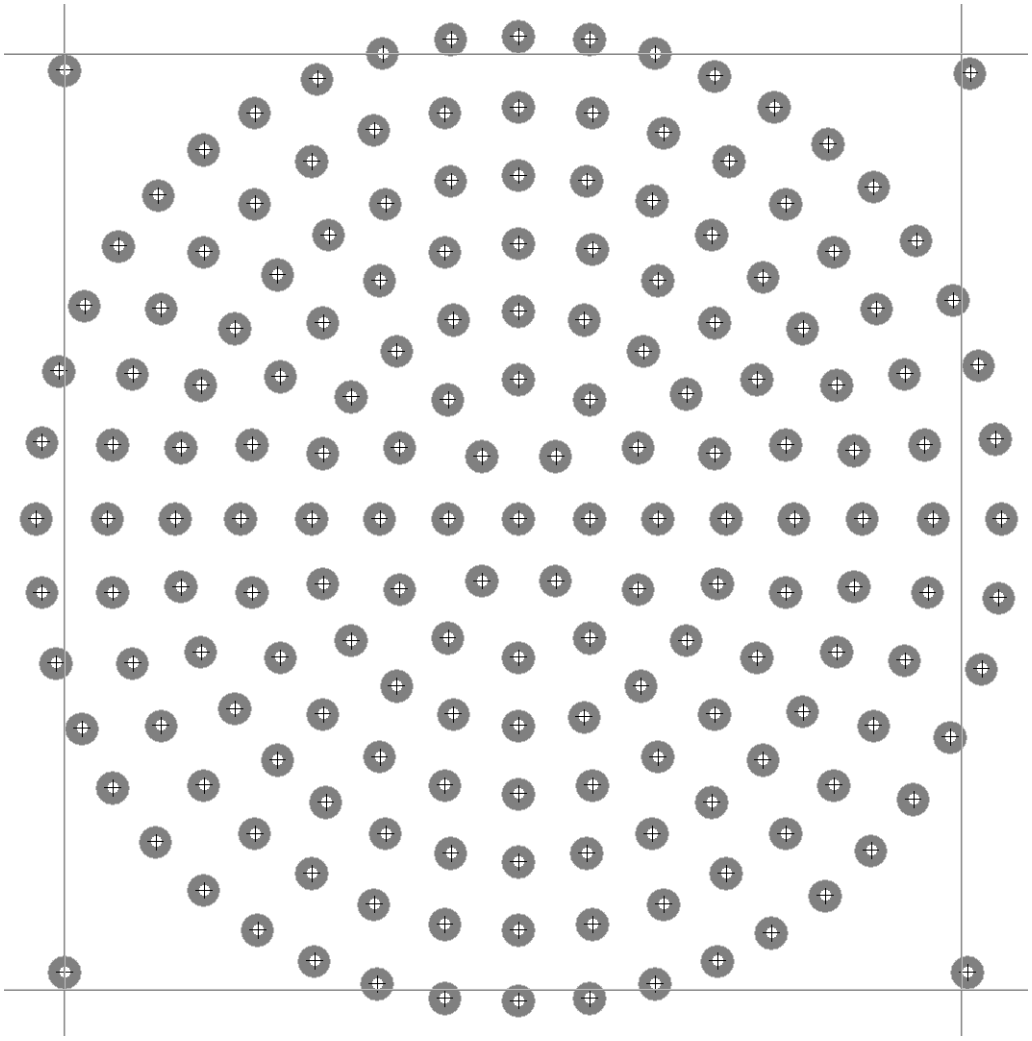
Color	1st band	2nd band	3rd band	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	
Red	2	2	2	100Ω	
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	
Blue	6	6	6	1MΩ	
Violet	7	7	7	10MΩ	
Grey	8	8	8	100MΩ	
White	9	9	9		
Gold				0.1	± 5%
Silver				0.01	± 10%

B Schematic


For one cell in each holder: short 1–3 and 2–4.
For two cells in each holder: short 3–5 and 4–6.
If not using onboard cells, connect power to CONN1.
The bluebox presented at projectmf.org is based on the PIC12F683.
This one is a reimagining of that project using an AVR ATtiny85 instead.
Sheet: /
File: bluebox-esquire.sch
Title: Bluebox, Esquire model
Size: USLetter
Date:
Rev:
KiCad E.D.A. kicad (5.0.1-dev-4-g0b54fa813)
Id: 1/1
VCC
GND
PWR_FLAG
D1
1N4005
CONN2
POWER SWITCH
C1
22uF
C2
0.1uF
CONN1
EXTERNAL POWER
SERIAL/PARALLEL
J1
B1
B2
IC1
ATTINY85-P
PB0(MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)
PB1(MISO/DO/AIN1/OC0B/OC1A/PCINT1)
PB2(SCK/USCK/SCL/T0/INT0/ADC1/PCINT2)
PB3(XTAL1/CLKI/OC1B/ADC3/PCINT3)
PB4(XTAL2/CLKO/OC1B/ADC2/PCINT4)
PB5(RESET/dW/ADC0/PCINT5)
CONN3
AUDIO OUT
C7
0.33uF
C6
0.33uF
C5
0.33uF
R16
150
R17
150
R18
330
R19
330
LED1
LED2
CONN4
ISP
R15
100K
X1
20MHz
C3
22pF
C4
22pF
SW0
2600
SW1
1
SW2
2
SW3
3
SW4
4
SW5
5
SW6
6
SW7
7
SW8
8
SW9
9
SW10
KP
SW11
0
SW12
ST
R1
1K
R2
1K
R3
1K
R4
1K
R5
1K
R6
1K
R7
1K
R8
1K
R9
1K
R10
1K
R11
1K
R12
1K
R13
1K
R14
1K

C Grille Pattern

This pattern of holes resembles what is found on Western Electric 500 telephones. Print or photocopy this page and adjust the scale until you get something that fits what you've selected for your internal speaker. You don't have to use all the holes. It's best to make the grille's diameter slightly smaller than that of the speaker. Cut out the pattern and glue it to the case with white glue or glue stick. School glue is perfect. Drill through the pattern and then wash off the paper and glue. Clean up any stray bits of plastic, take down hard edges, and you've got a very neat grille. I find that a scale of 25% and an 1/16" (1.5mm) drill bit give the best results.



D Assorted Webpages

- http://projectmf.org/
  Don Froula's project to preserve the history of phone phreaking. Includes patches to the Asterisk open source PBX software to allow for MF phreaking on private exchanges.
- http://explodingthephone.com/
  Author Phil Lapsley traces the birth of the telephone, the rise of AT&T's monopoly, the discovery of Ma Bell's Achilles heel, and follows the kids and outlaws who used it for fun and profit.
- https://www.asterisk.org/
  Open source PBX software.
- https://en.wikipedia.org/wiki/Blue_box
  Wikipedia article on blue boxes.
- https://661.org/proj/bluebox
  The homepage for this project.
- https://gitlab.com/DavidGriffith/bluebox-avr
  Git repository for the firmware for this board.
- https://gitlab.com/DavidGriffith/bluebox-esquire
  Git repository for this board.

COLOPHON

This manual was typeset using LaTeX version 2e. Xfig was used to create drawings. A Samsung J7 Star (SM-J737T) mobile phone was used to take photographs. Gimp and ImageMagick were used for further processing and conversion. Some material was copied, with permission, from Don Froulas' writings on blueboxes.