A Homemade LD to DMTF Convertor

Introduction

I've been after a thing like this for a while as I use an ATA which doesn't understand pulse dialling. A few things are commercially available:—

First is the RotaTone. These are fitted inside the telephone so you need one for every 'phone you wish to convert. Great if you only want to do one but it'd soon get expansive if you want to do several.

A better-sounding bet is the DialGizmo which is a box that connects between the socket and 'phone, or 'phones. They cost about the same but come from Australia and are expensive enough to be liable for Import VAT, plus the other charges associated therewith.

A similar thing called a Dialor2 is available from China via AliExpress or e-Bay but these aren't much cheaper and are still enough to get HMRC interested.

I finally came across this design for which I claim no credit. It was first developed by Boris Cherkasskiy and then modified by Arnie Weber.

Both these people have made their code available (the link in Boris Cherkasskiy's page is dead but it's available from here), though sadly they both seem to assume that any reader already knows about programming micro-controllers. Arnie Weber provides a confusing wealth of data, including a double-sided PCB which is beyond the skills of we primitive creatures from the heath, but no instructions at all.

I did find a guide to getting the code onto a µC this but as it was a few years old the software recommended had changed meaning the instructions no longer applied. This is the tale of how I managed it.

I've designed a single-sided circuit board and tweaked the circuit diagram to show how to connect it to a GPO 746 telephone, which is what this guide uses as an example. This model actually involves more work than the older ones. The dial connexions are the same no matter the model of telephone and diagrams for the 232 and 332 are included in `DIAGRAMS.PDF`.

I also added a bridge in case the line polarity is reversed.

The PCBs, diagrams and code (Arnie Weber's version) is in a ZIP file here. If you already know how to program micro-controllers you may as well grab that now and crack on.

Getting the Code onto the Microcontroller

From what I could tell the cheapest way to do this was via an Arduino. I got a Chinese Uno clone for less than three quid.

Assuming you've already got an Arduino and installed it's IDE software here's what to do.

Open the sketch `File > Examples > 11. ArduinoISP > ArduinoISP` and upload it to the Arduino.

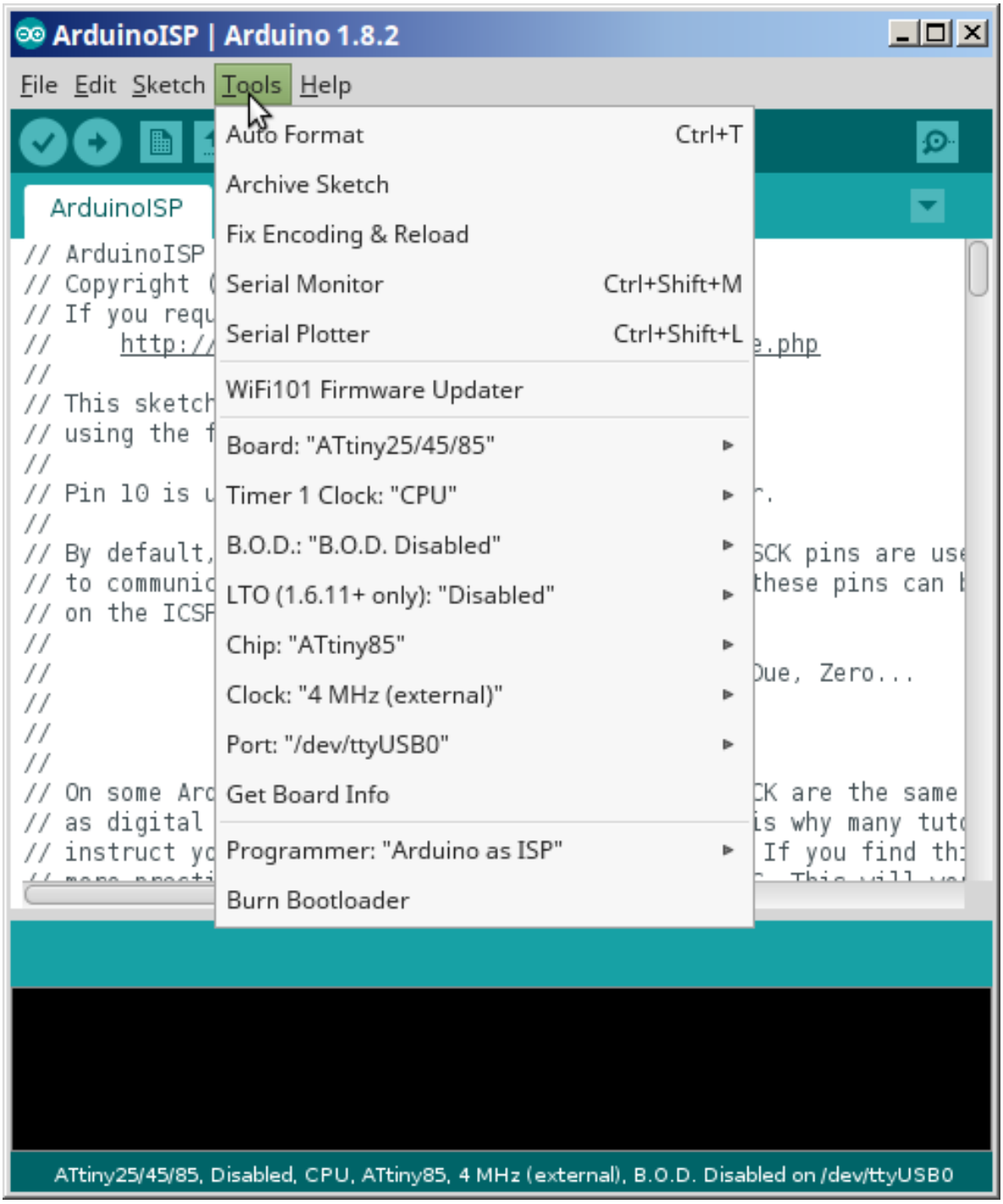
Go to `File > Preferences` and enter `http://drazzy.com/package_drazzy.com_index.json` into the `Additional Boards Manager URLs` box.

Go to **Tools > Board > Boards Manager**. Scroll down to **ATTinyCore by Spence Konde**, click it and then click **Install**.

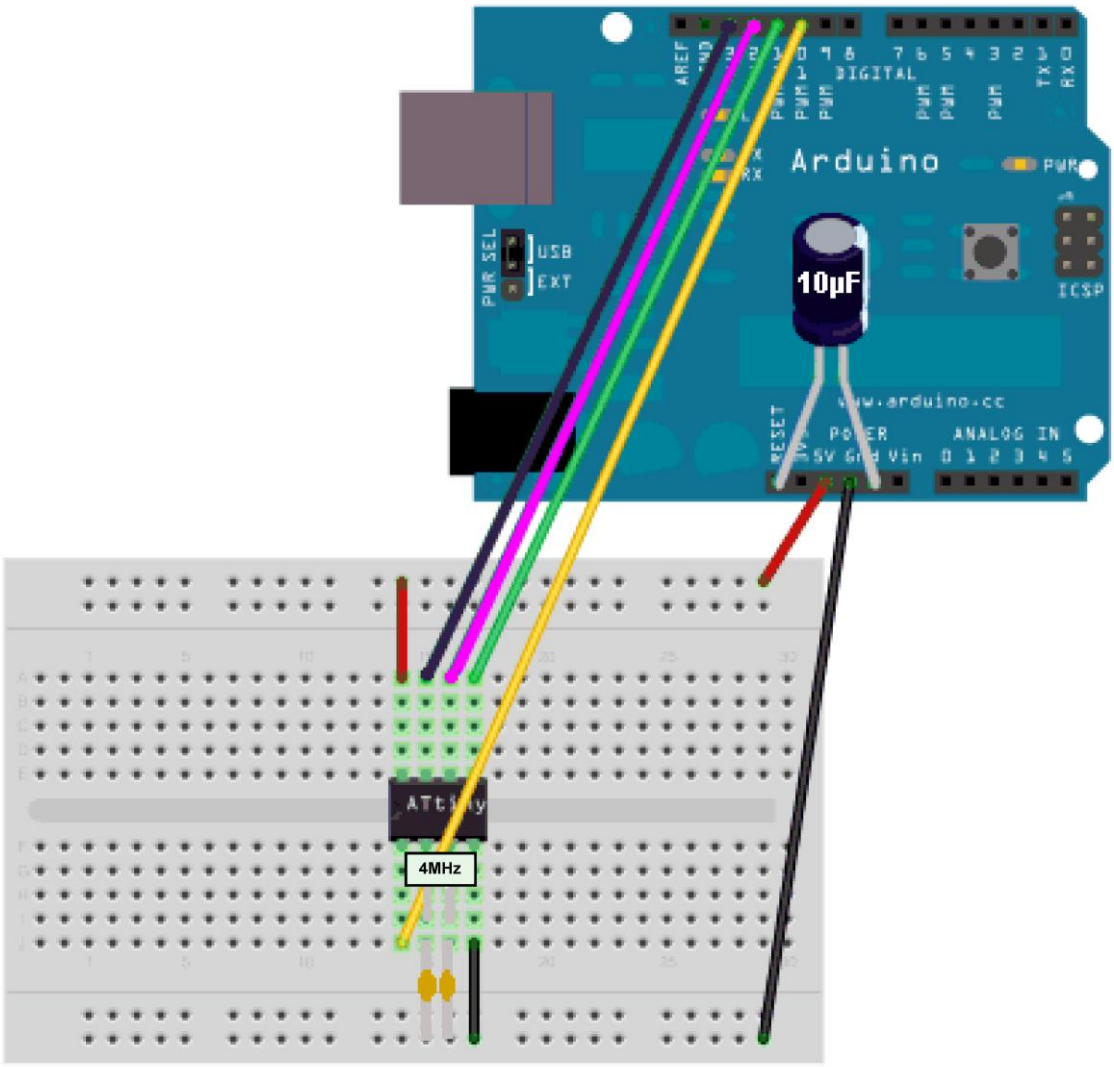
Now go back to **Tools > Board** and choose **ATtiny 25/45/85**. Go to **Tools > Chip** and choose **ATTiny85**. Go to **Tools > Clock** and choose **4 MHz (external)**.

Finally go to **Tools > Programmer** and choose **Aurduino as ISP**.

The **Tools** menu should now look like this, though the **Port** may be different:—



Next connect the ATTiny85, **plus its crystal and capacitors** to the Arduino. The circuit specifies 10pF capacitors but my crystals said to use 20pF so I did. The crystal goes between Pins 2 and 3 and a capacitor is connected between Pin 2 and Ground and another between Pin 3 and Ground. Also connect a 10µF capacitor between RESET and GND on the Arduino.



Now click **`Tools > Burn Bootloader`**.

Extract the file **`main.hex`** from the ZIP file referred to above and put it in your Home directory. On Windows that will be something like **`C:\Users\Joe`** or **`/home/joe`** on Linux.

Open a Command Prompt/Terminal and enter a command **SIMILAR** to one of these; your Home directory isn't going to **`joe`**, the Arduino IDE software may have been installed elsewhere and the port could be different. NB Linux commands are case-sensitive as is AVRDude on either system!

On Windows:—

```
"C:\Program Files\Arduino\hardware\tools\avr/bin/avrdude" -C"C:\Program
Files\Arduino\hardware\tools\avr/etc/avrdude.conf" -v -pattiny85 -cstk500v1 -
PCOM4 -b19200 -Uflash:w:main.hex:i
```

On Linux:—

```
/home/joe/arduino-1.8.2/hardware/tools/avr/bin/avrdude -C/home/joe/arduino-
1.8.2/hardware/tools/avr/etc/avrdude.conf -v -pattiny85 -cstk500v1 -
P/dev/ttyUSB0 -b19200 -Uflash:w:main.hex:i
```

I'll try to explain what this lot means to help making the necessary changes.

`"C:\Program Files\Arduino\hardware\tools\avr/bin/avrdude"` or `/home/joe/arduino-1.8.2/hardware/tools/avr/bin/avrdude` tells the computer to run AVRDude.

`-C"C:\Program Files\Arduino\hardware\tools\avr/etc/avrdude.conf"` or `-C/home/joe/arduino-1.8.2/hardware/tools/avr/etc/avrdude.conf` tells AVRDude to use the configuration file that the Arduino IDE software has already created and which must work or we wouldn't have got this far.

`-v` is Verbosity, ie tell it to let you know what's happening.

`-pattiny85` tells it that we're programming an ATTiny85.

`-cstk500v1` tells it that we're using a Atmel STK500 programmer with Version 1.x firmware, which is what the Arduino is emulating.

`-PCOM4` or `-P/dev/ttyUSB0` tells it which port the programmer (ie the Arduino) is connected to.

`-b19200` sets the baud rate.

`-Uflash:w:main.hex:i` tells it to write `main.hex` to the chip's flash ROM.

Hopefully the Tiny85 is now ready for use. If something has gone wrong it may go into a sulk and refuse to talk to you anymore. If this happens there are details in the ZIP on how to reset it so you can start again.

Construction

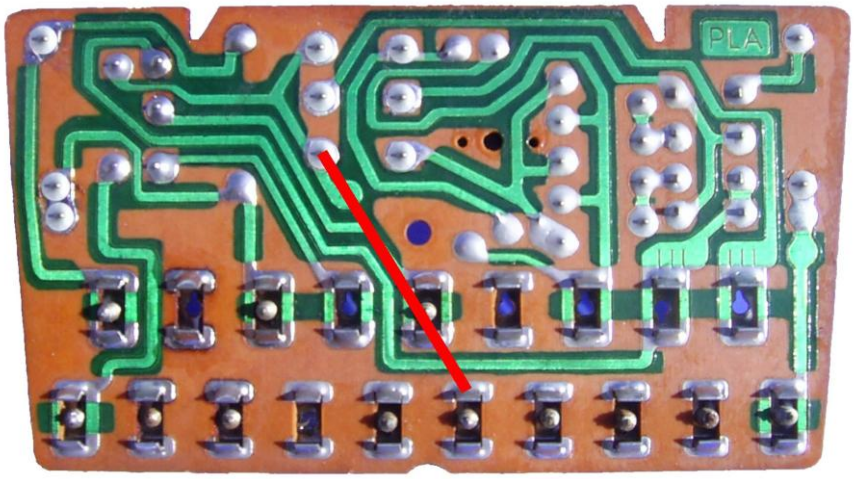
The ZIP contains two PCB versions. One is fairly square and the other long and thin. Choose the one you prefer.

Installation

After building the circuit it needs installing in a telephone.

Firstly remove the dial and its associated wiring. Connect T8 and T10 with a piece of wire.

Later 746s had extra terminals labelled T19A and T19B. Early ones don't. The easiest fix is to use one of the spare terminals, say T14. Solder a bit of wire between the underside of that and Pin 4 of the Induction Coil.



Connect terminal 1 on the converter to terminal 4 of the dial and terminal 2 to terminal 1. Connect dial terminals 2 and 5 together and then connect both of them to terminal 3 of the converter. Finally connect converter terminal 4 to T8 and terminal 5 to T19A or T14. See `DIAGRAMS.PDF` in the ZIP.

Operation

To use this devise you simply dial as normal and when dial returns to rest it will send the appropriate DTMF tone. It will however do more.

To dial a *, dial 1 and hold the dial against the stop until you hear a beep. Let go and when the dial has returned it will send a *.

To dial a #, dial 2 and hold the dial against the stop until you hear a beep. Let go and when the dial has returned it will send a #.

Numbers 3 to 9 can be used for speed dialling. To set these up dial 0 and hold it until you hear a beep. When the dial has returned it will play a tune. Now dial the figure you wish to program; it will then play a shorter tune. Now dial the number you wish to store. When you've finished dial and hold 0 again.

To call a stored number dial and hold the figure you stored it under.