FT-817 Buddy

Build Instructions

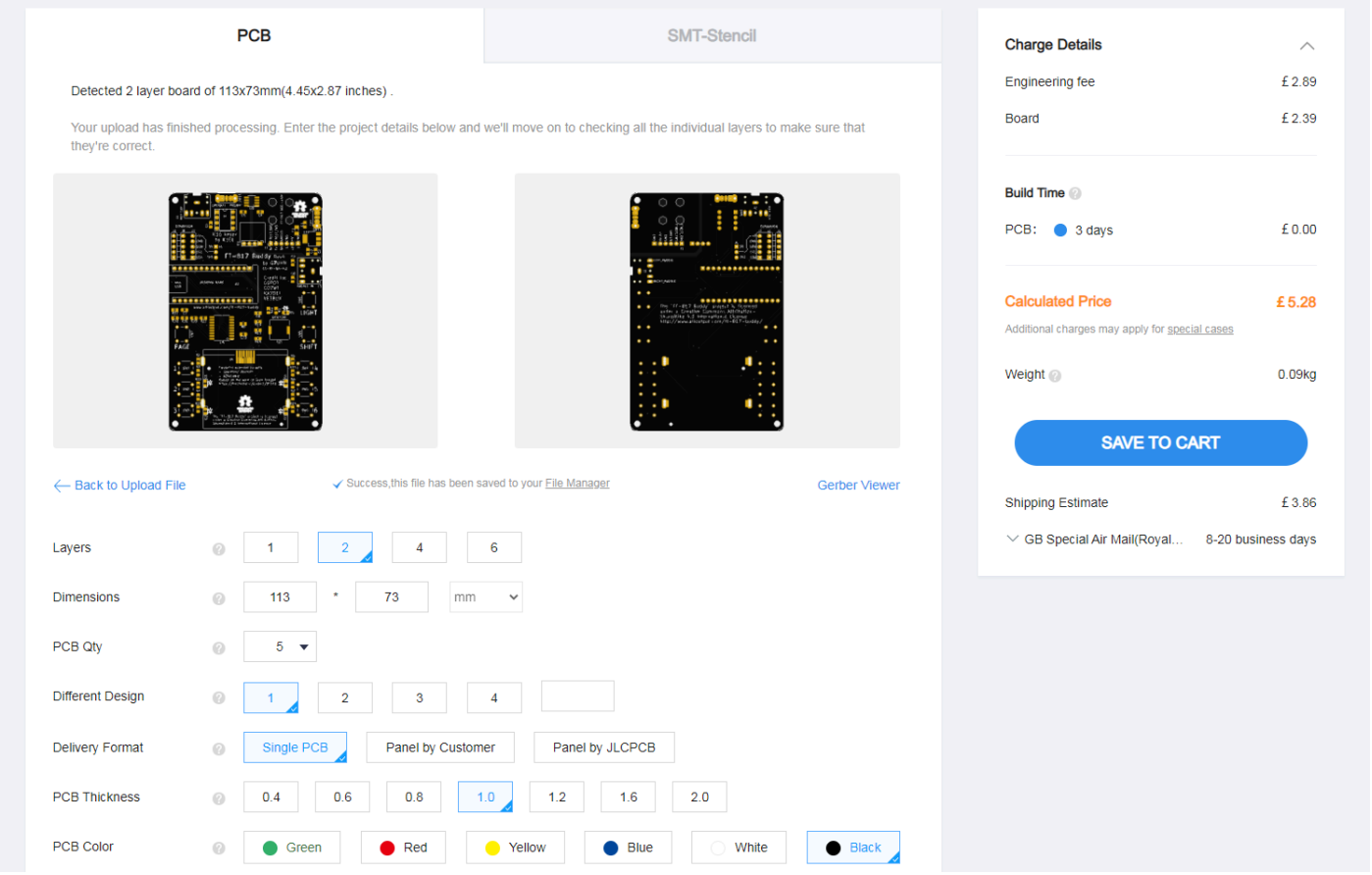
Rev4

Note that because of Minimum Order Quantity (MOQ) against various components, the parts for the FT-817 Buddy are best ordered for multiples of 5 or 10 boards. This makes this build suitable for clubs or groups of friends. It can be done individually but the cost will be higher so try and share the fun!

Acquire the PCB

The PCB KiCAD design and gerber files are available on G7UHN's GitHub repository.

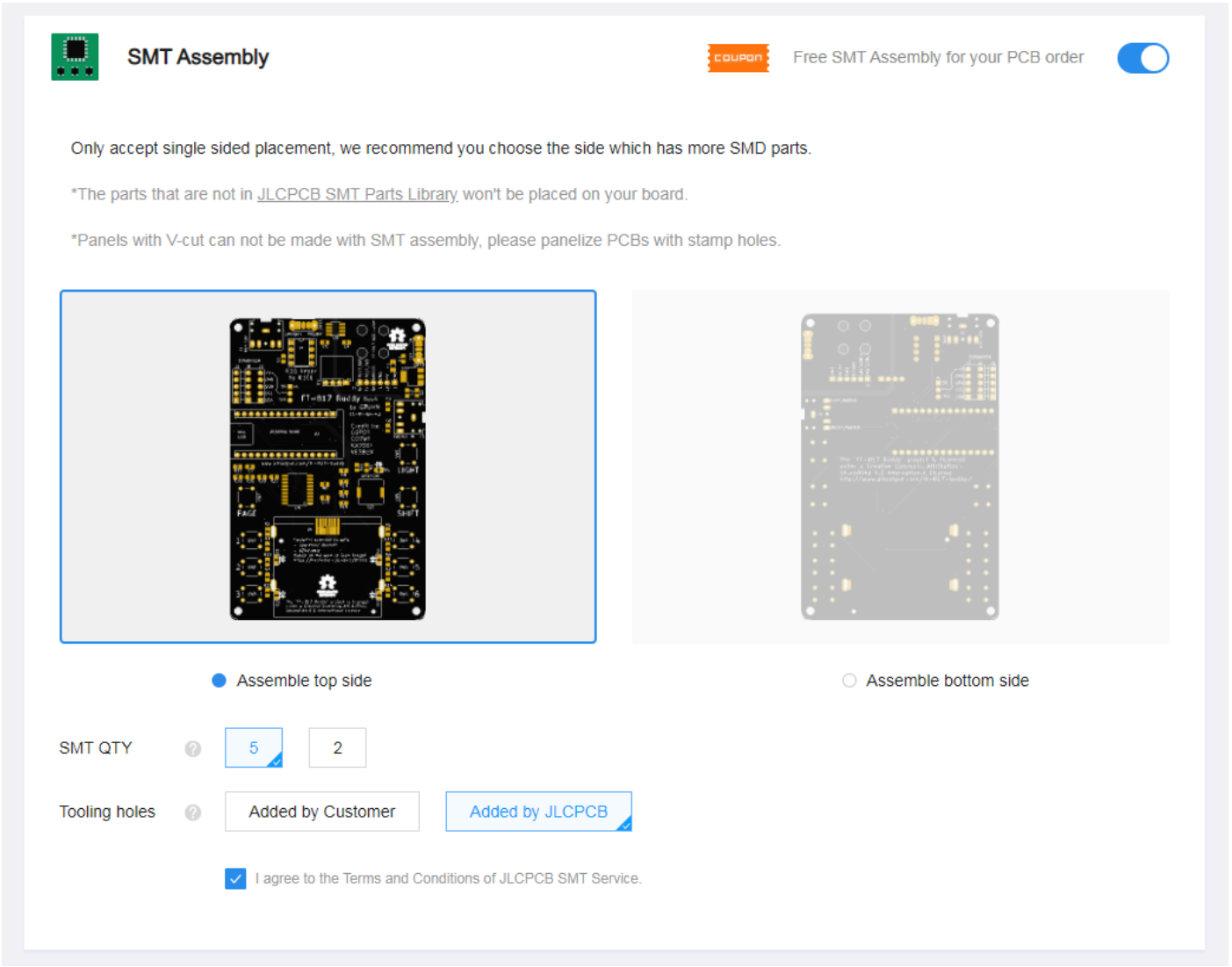
The PCBs made so far have been produced by JLCPCB so the tolerances on the gerber files appear to work with JLC but they should probably work with other fabricators (and you can always produce new gerber files using the KiCAD files if you wish). To order PCBs from JLCPCB just go to the website, hit "Quote Now" and upload the gerber files as a zip file (gerbers.zip is in the GitHub repository along with all the source hardware files).



IMPORTANT: When choosing options for the order **choose a PCB thickness of 1.0mm** to allow the Nokia display to clip nicely onto it. You can pick colour, quantity, etc. to suit. I've used the default options except for the thickness.

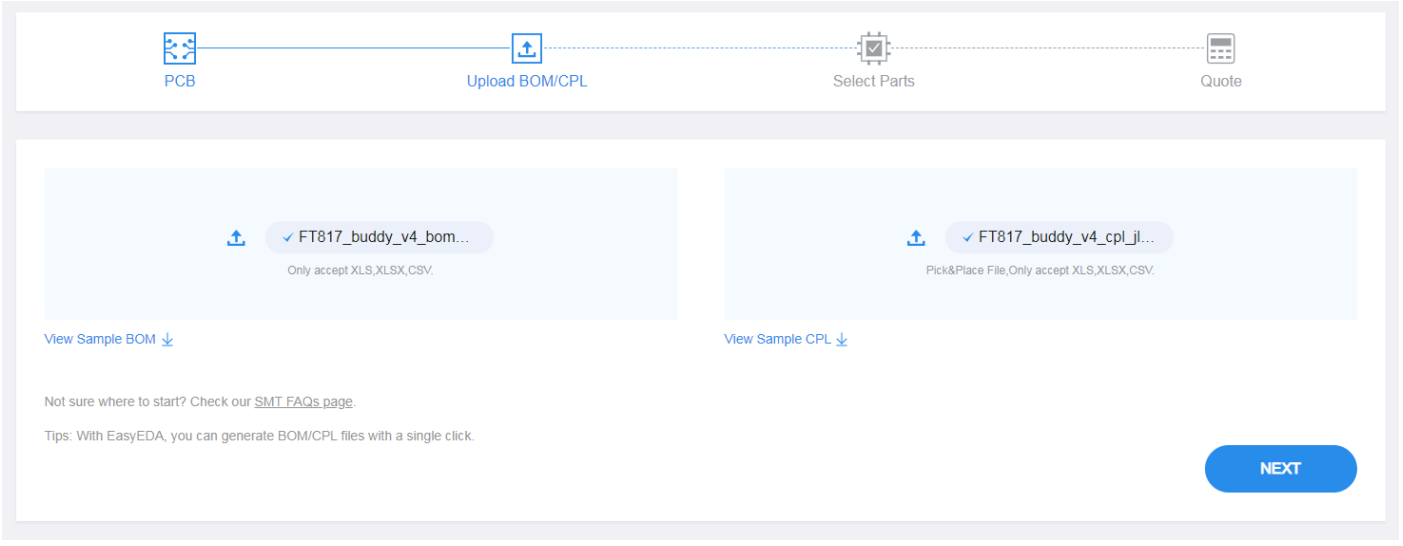
I highly recommend you get JLCPCB to assemble most of the surface mount (SMT) parts for you. You can do it the hard way if you like but this saves so much time and hassle and you still get to wave your soldering iron on a couple of SMT parts later…

Select the SMT Assembly option, click on the picture of the top side of the PCB (the side with all the text on it) to select which side the components will be soldered on to, check the box to agree to T&Cs and hit Confirm:



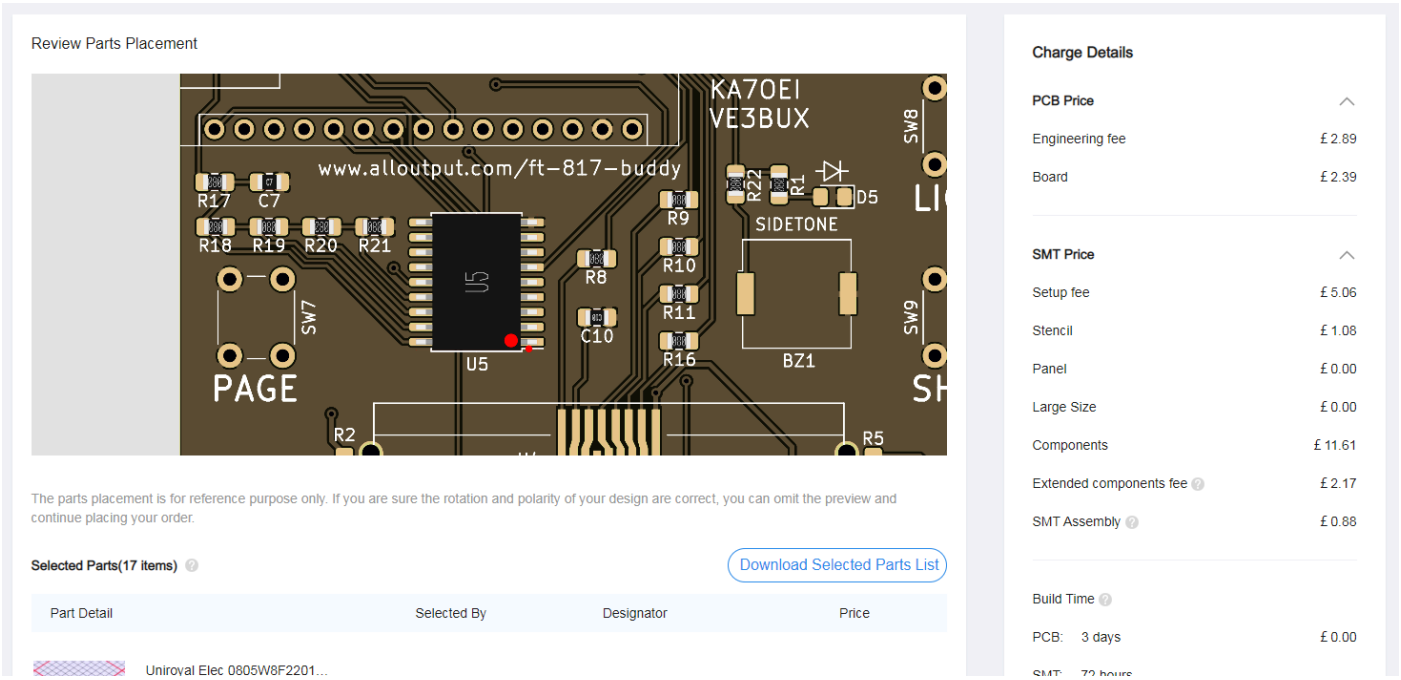
Upload the BOM and CPL files (found in the hardware files in the GitHub repository):

- FT817_buddy_v4_bom_jlc.csv
- FT817_buddy_v4_cpl_jlc.csv



Hit Next. Scan down the list on the next page, I've checked all these parts are in the right orientation so you should be good to confirm them all and hit Next.

Important step, click on the image of the parts placed on the PCB, zoom in and have a close look around at how all of them have been placed… looks good? Go ahead and SAVE TO CART and place your order!



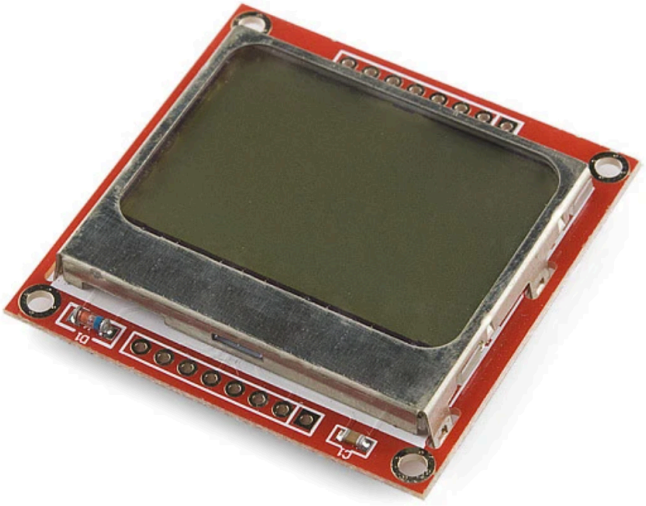
Sit back and wait a week or two for the PCBs to arrive.

Buy the components

While you're waiting for the PCBs to arrive you can buy the rest of the components listed in the Bill Of Materials (BOM). There are a couple of points to note:

- The exact model of the Nokia display is important. There are a few variants of these Nokia-style LCD modules and the differences are subtle.

  The Sparkfun display (model Sparkfun LCD-10168) is slightly smaller than the AZDelivery model that is easy to get on Amazon. Just search around for "Nokia 5110 LCD" and you'll find them. Both have been successfully tested on the Rev4 board. The Adafruit model appears to be the same size as the Sparkfun LCD module but I haven't tested one.

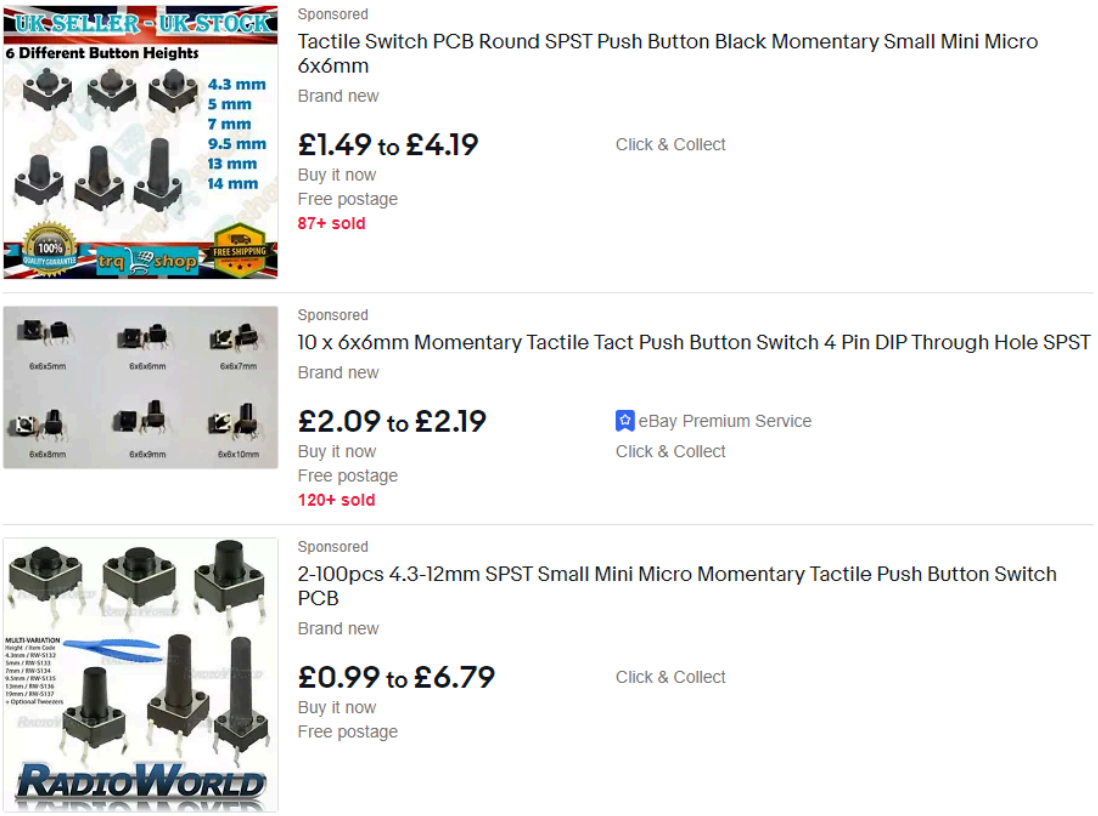


Sparkfun display

AZDelivery display

When unclipped from the PCB, different LCD modules have either 8 or 9 pin contacts, spring or rubber contacts and different sizes. The Rev4 FT-817 Buddy is compatible with Sparkfun and AZDelivery, DEVMO is still to be tested...

- Buttons. You can fit any old 6mm tactile push switches to the Buddy and people often have thousands of these buttons in their workshops. Or you could be really picky and carefully select from the millions of available buttons on suppliers like Mouser, Farnell, etc. choosing actuation force (about 150 grams/gf seems good to me), the button length (height) to suit ideas on enclosures, maybe some matching square or round switch caps. Your call. If I ever decide on the ideal 6mm tactile switch for this project I'll update the BOM. The example part number on the BOM right now is a 12.5mm height switch that is compatible with the current enclosure design (work in progress).



So many options for 6mm tactile switches…!

- LEDs. To keep the Buddy power consumption down I've built this PCB with resistors for **low power LEDs** with a forward current of about 2mA. I've listed APT2012LZGCK from Mouser as the part number in the BOM and these are green, they look pretty good. You're welcome to shop around for different colours or whatever, just make sure they're 0805 size and expect a similar forward current. Because these are slightly exotic they weren't available in the JLCPCB parts library for SMT assembly, you get to solder these ones on yourself. You can do it.
- 8-pin Mini DIN cable, heatshrink, cable ties, velcro… I'll leave these to your personal style. Don't go mad on the velcro though, standard stuff is fine.

The Rev4 Bill Of Materials (BOM) is online here.

For my last order in the UK I got most of the bits from Amazon (for the Arduino Nano, display, cables, bits and bobs), Digikey (for the electronic components) and Kanga Products (for the K16 keyer chip).

Assembly

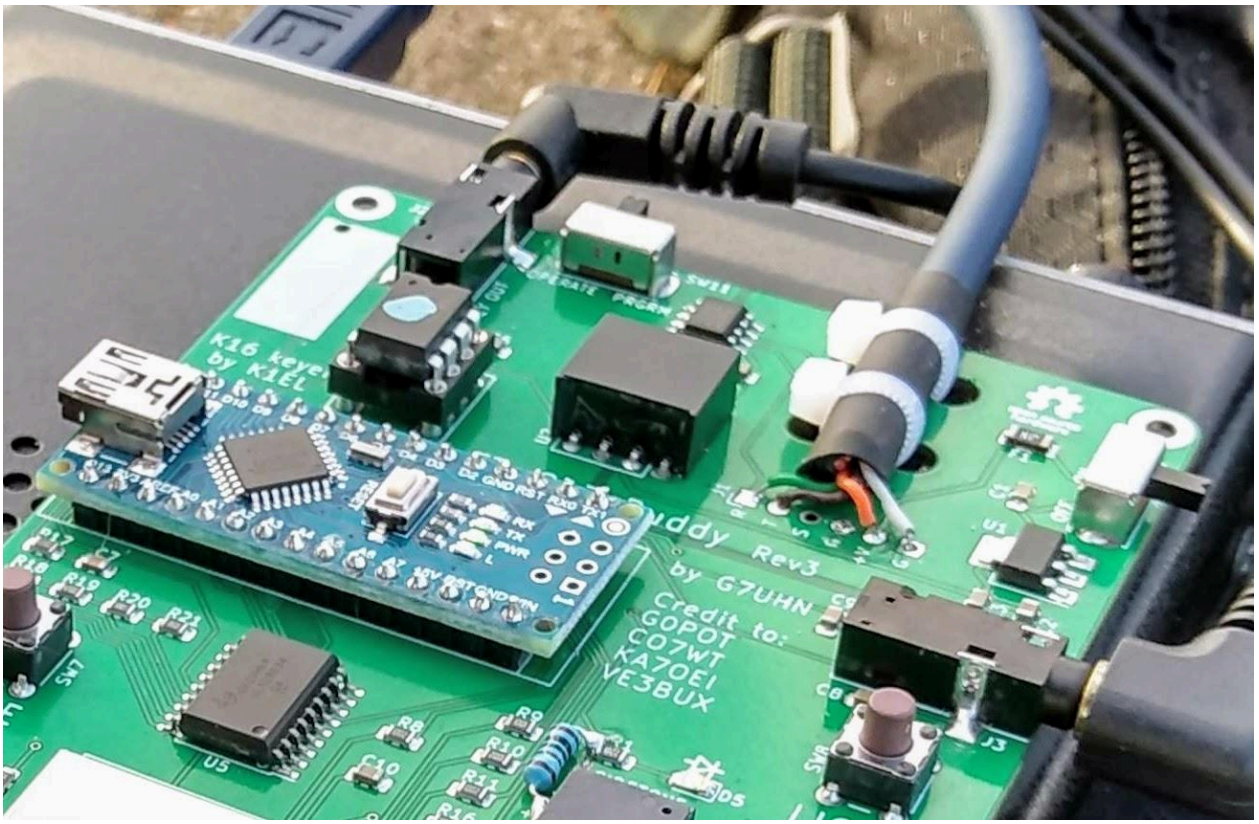
OK, you've got your PCB and you've got your components. Let's do this.

1. Solder on the small SMD components (ADUM1201, LEDs, Resistors, Capacitors, Linear Regulator, Polyfuse, Buzzer). If you've ordered the PCBs with the JLCPCB SMT assembly service you'll only have to solder on the LEDs and the Buzzer. Michael G0POT has produced an excellent video showing how he soldered on the LEDs and buzzer.
    a. LED arrows on the underside of each LED will have to match the direction of the diode symbol printed on the PCB silkscreen.
    b. Keep the LEDs towards the "outside" side edge of the display footprint, too far inwards and they clash with the LCD module

2. Now is a good time to mark and drill the clear lid if you're going down the enclosure route, see the Enclosure Notes.

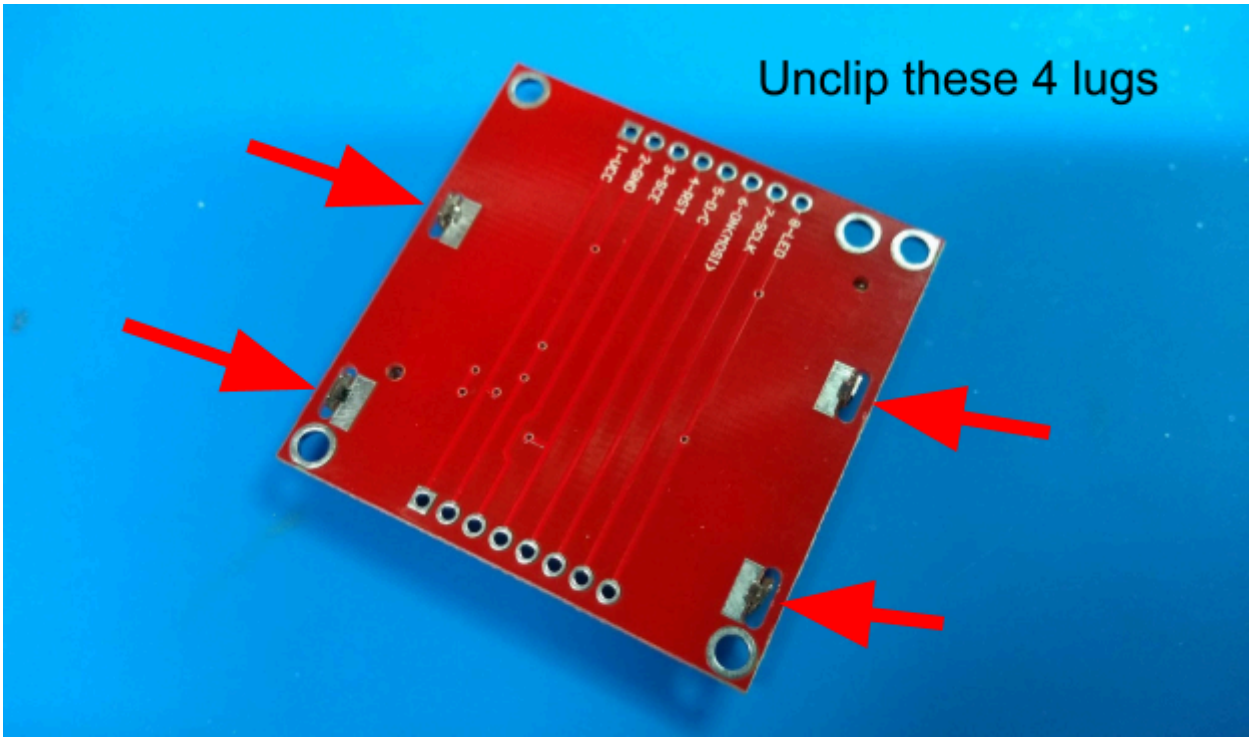
3. Solder on the larger through-hole components (Arduino, slide switches, J2 connector…). If you're going to drill and fit a lid to the Buddy it might be a good idea to leave the push switches to the end, again, see the Enclosure Notes.

4. For the B0505S-1W note the orientation of this on the PCB by referring to the photos below. The pins are to be bent to right-angles such that the B0505S-1W sits flat on the board, matched to the component outline on the PCB.

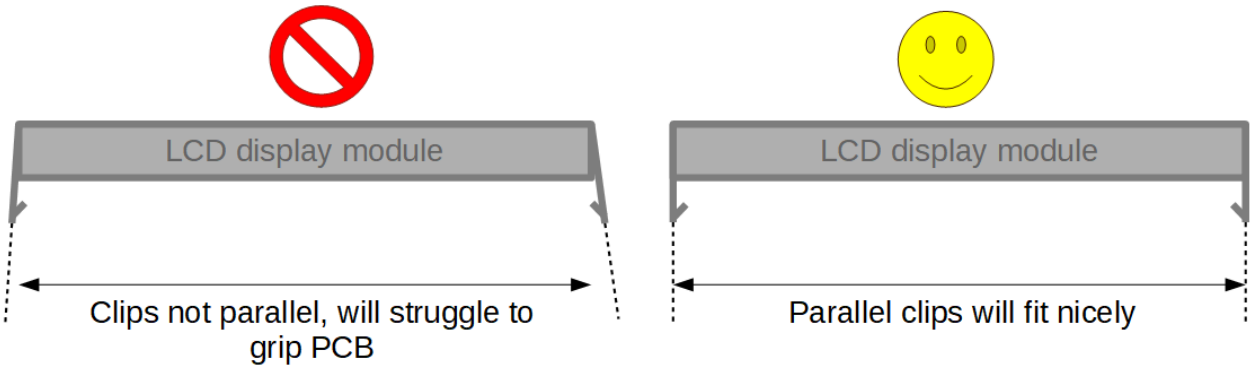


5. Maybe give your board a bit of a clean with IPA or something now...

6. Unclip the LCD assembly from the donor board (photos of Sparkfun, other models are similar) - **be careful not to lose the rubber conductive contact** when the LCD is removed from the donor board. Note the AZDelivery and DEVMO LCD modules use spring contacts, not a rubber contact.



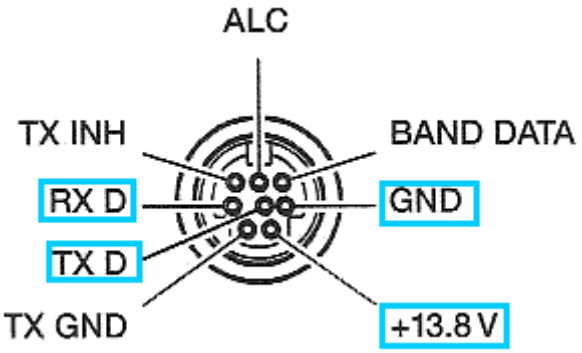
7. After unclipping the LCD, you may need to straighten out the clips so that they provide a nice grip on your PCB:



8. With the rubber contact in place on the LCD module (not required for AZDelivery or DEVMO LCD), carefully clip the LCD assembly to the Buddy PCB using the holes on the Buddy PCB that should match the clip holes on the donor board. Everything should line up, if it doesn't check everything. Note, there may be some fiddling required to get the four lugs to engage on the PCB, using a small precision screwdriver might help lever the lugs into place. You can solder the lugs to the pads on the underside of the PCB if required.

9. Fit FT-817 ACC cable to the Buddy using some heatshrink and cable ties to secure the cable to the PCB after the wires have been soldered. **You'll have to determine your cable pinout/colour code with a multimeter (there is no standard among these cables and the image below is a notional example only)**.

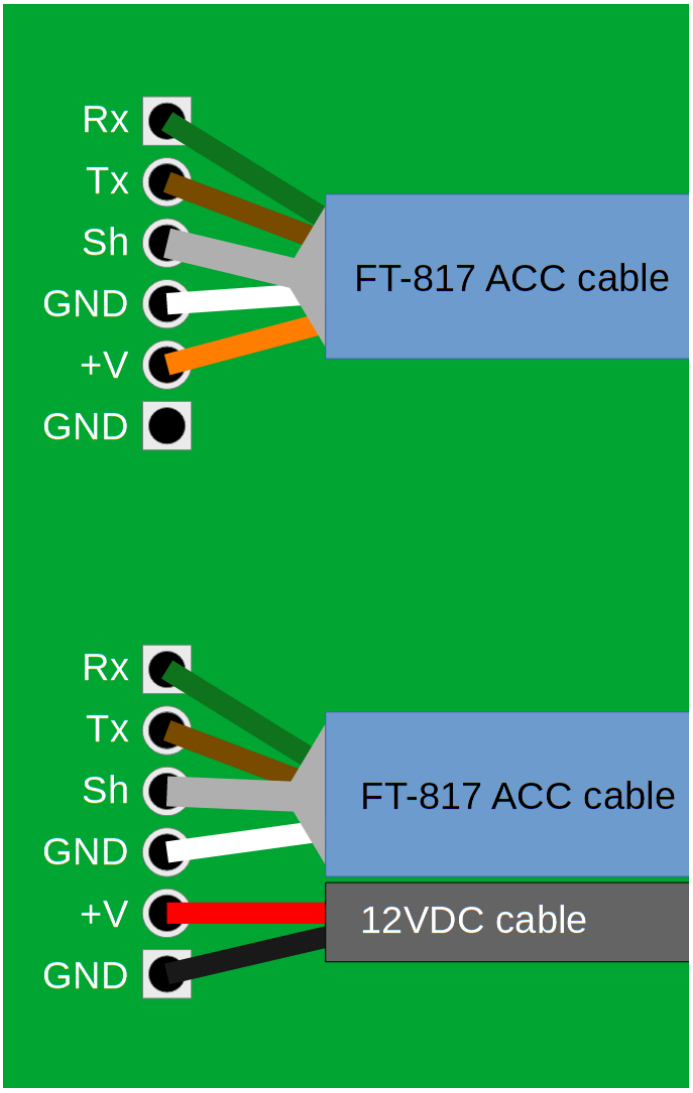
   HELPFUL HINT: Use the gap in the middle row of pins to help orientate yourself as you buzz out your connector cable.



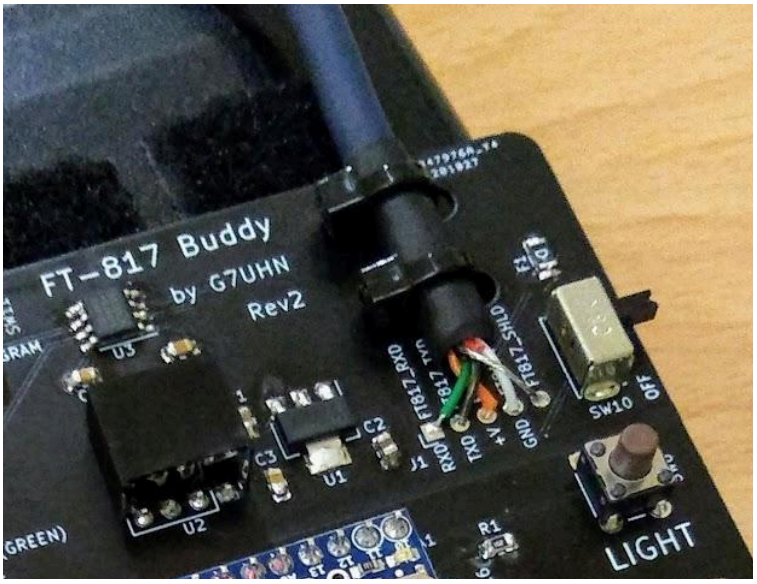
Match the pins to the labels on the PCB i.e. solder the FT-817's "RX D" pin to "Rx" on the PCB and so on.

NOTE OF CAUTION: An extra ground pad is provided on the PCB in case you wish to *power the Buddy with a separate power source*, see figure below. **Cabling in this way lowers the risk of a problem that may damage your FT-817** (i.e. accidental short-circuit on the Buddy blowing the current-limiting resistor in the radio, or worse…). Depends on how confident/ risk-averse you feel. You might like to look at the VK4GHZ +12V Accessory Connector Mod which is quite tricky and also carries some risk. Or play it safe and wire a separate power cable to the Buddy for an external 12V source as shown below.

An optional pad is provided to anchor the cable shield ("Sh") if desired, for neatness only, no electrical connection is made because the shield should be grounded at the FT-817.



Two options for cabling your FT-817 Buddy. Either power from the FT-817 ACC port (top) or use a separate cable to power from external 12V if you prefer (bottom).



Cable fitted to the Buddy. Note the use of heatshrink to beef the cable up a bit before cable-ties secure it to the PCB.

10. **Check there are no shorts between wires on the FT-817 ACC cable.** This is most important as this area does not receive protection from the polyfuse. If there is an accidental short between "+13.8V" and ground here (easily done with an accidental solder bridge or strand of wire) it will blow the current-limiting resistor in your FT-817 which is a difficult part to replace. You have been warned!

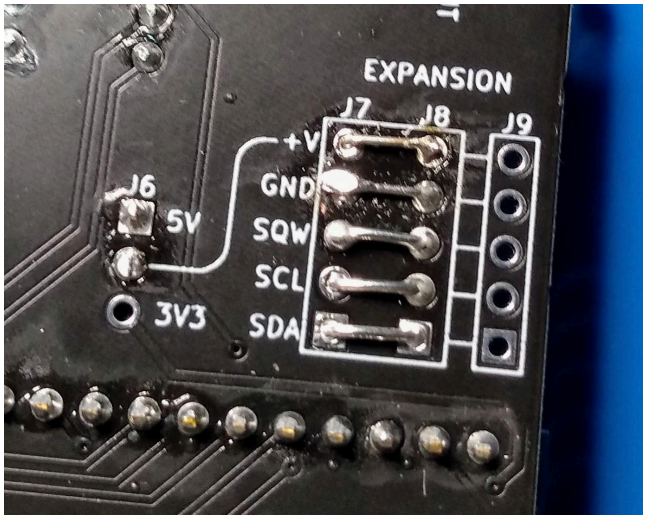
11. Visually inspect your PCB for any obvious shorts, solder bridges, unsoldered pads, wire strands, etc.

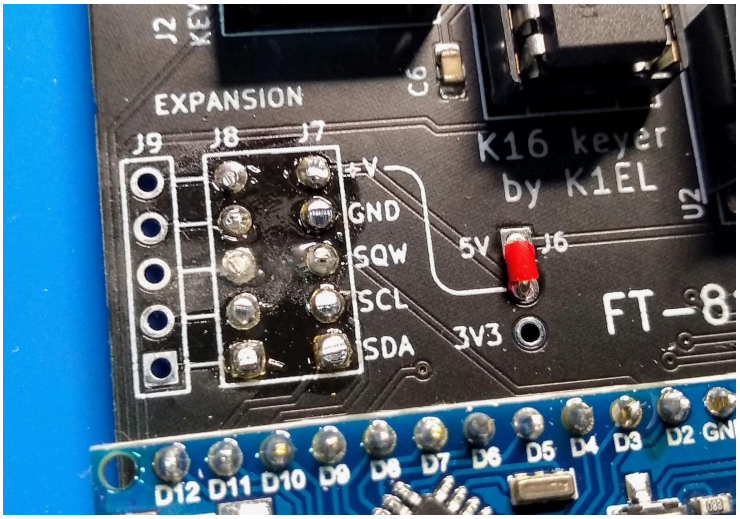
    (A polyfuse is fitted on the Buddy PCB after the cable so this should protect the FT-817 from any shorts further down the line but it's still wise to check).

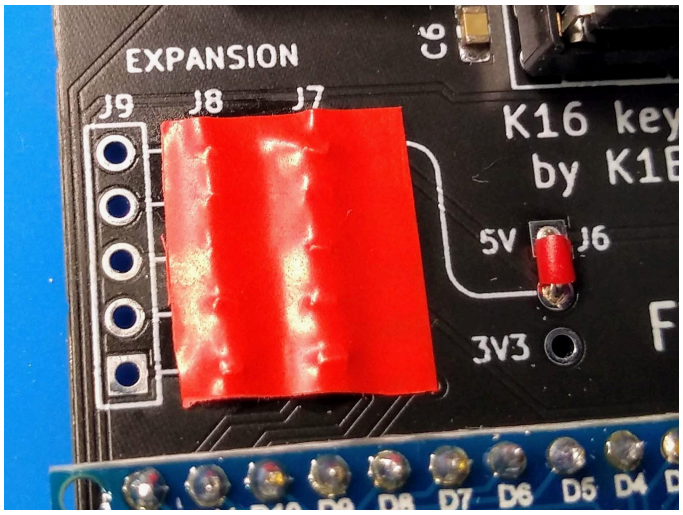
Option: Add a real-time clock

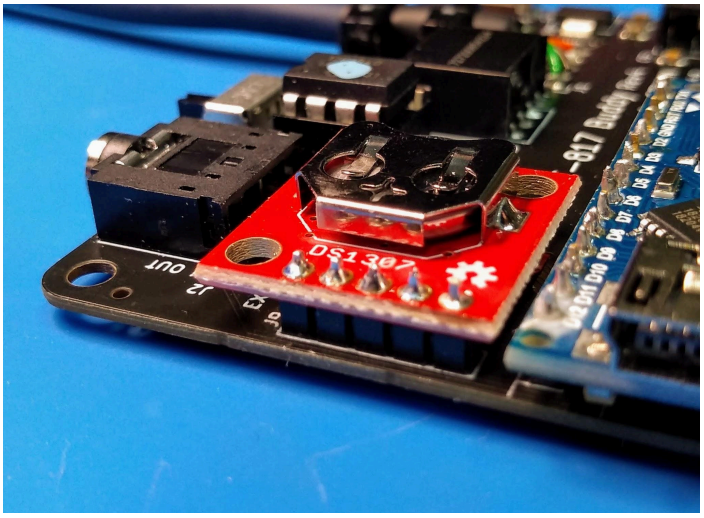
From Rev4 the FT-817 Buddy includes an "expansion header". This is primarily to allow the option to add a real-time clock (RTC) module onto the Buddy, it saves you having to set the time on the clock every time you power up. I could have built the RTC onto the main Buddy board but thought some people might not be bothered, it increases complexity, cost, components… and doing this as an optional expansion also means there are pins available on the Buddy PCB if anyone wants to interface something different to the Buddy's Arduino (e.g. a GPS module to show time and Maidenhead locator…).

The pins and size of the Buddy Rev4 have been made to match Sparkfun's BOB-12708 RTC module that is based on a DS-1307 RTC chip. A patch panel on the Buddy PCB allows you to reconfigure the pinout and supply voltage of the expansion header but for the Sparkfun BOB-12708 you just wire it up with links for 5V and all the patch pins go straight across as shown in the photos below. Links for the patch panel are best made on the underside of the board to give clearance to the module (I had to re-work mine, hence the messy soldering!) and I place some insulating tape over the patch panel on the top side of the board to guard against possible shorting:









Enclosure

Notes on making an enclosure for the Rev4 Buddy can be found here. If you do proceed to load software and test without an enclosure, just be mindful of what you place the Buddy on, don't short-circuit the contacts on the underside :-)

Load the software

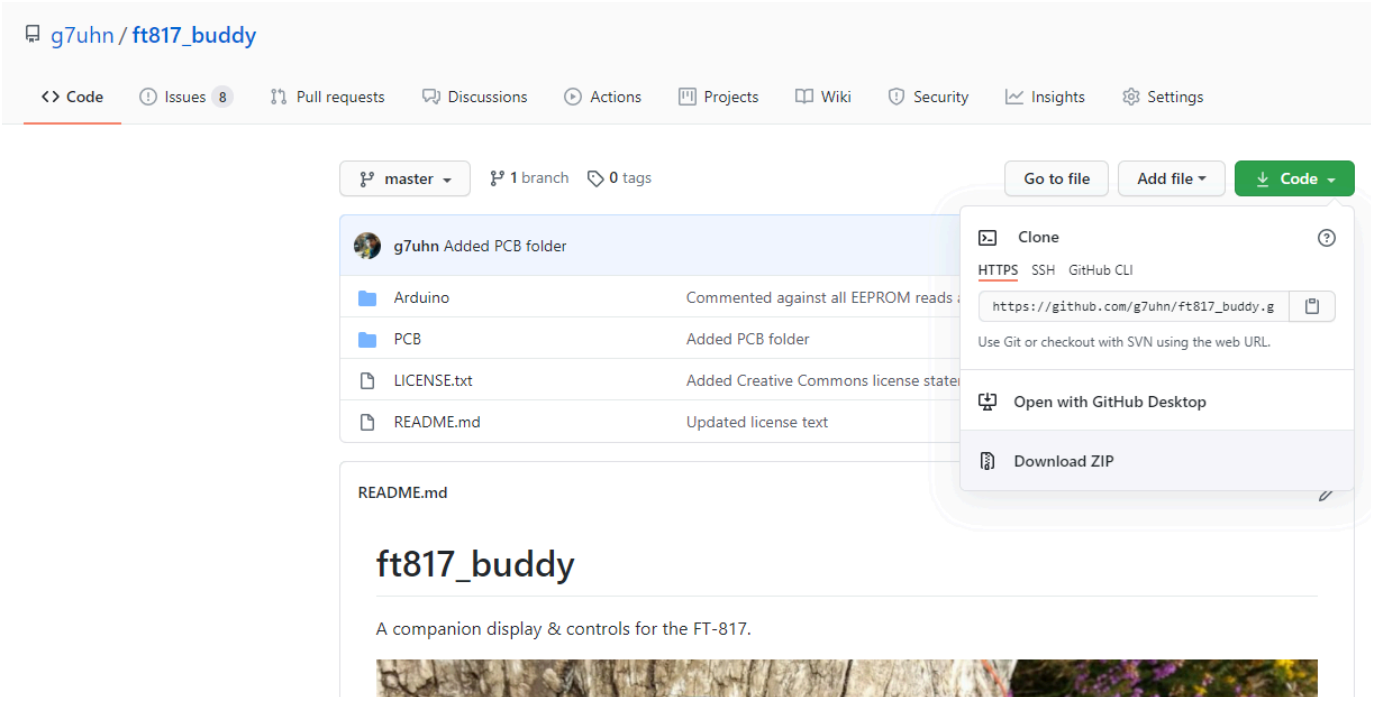
Before you go any further, now is the time to note down the 76 factory settings of your FT-817.

These hidden settings are unique to your particular serial number (everyone's radio is different) and account for the individual component tolerances in your unit. They are set by Yaesu technicians during the factory alignment procedure. Because the FT-817 Buddy interacts with the FT-817 EEPROM there is a chance that a write error (either a hardware glitch or some bad code someone's written) may cause the FT-817 to erase these settings. If this happens you will need to re-enter these factory settings yourself (or send the radio back to Yaesu to re-run the factory alignment procedure, at your cost). Be prepared.

See http://www.ka7oei.com/ft817_calibrate.html for details of how to access these settings, be careful and don't change them, just write them down somewhere double-safe.

OK, now you've recorded your radio's settings, let's proceed!

Download the code from https://github.com/g7uhn/ft817_buddy (click on the green Code button, Download ZIP…)



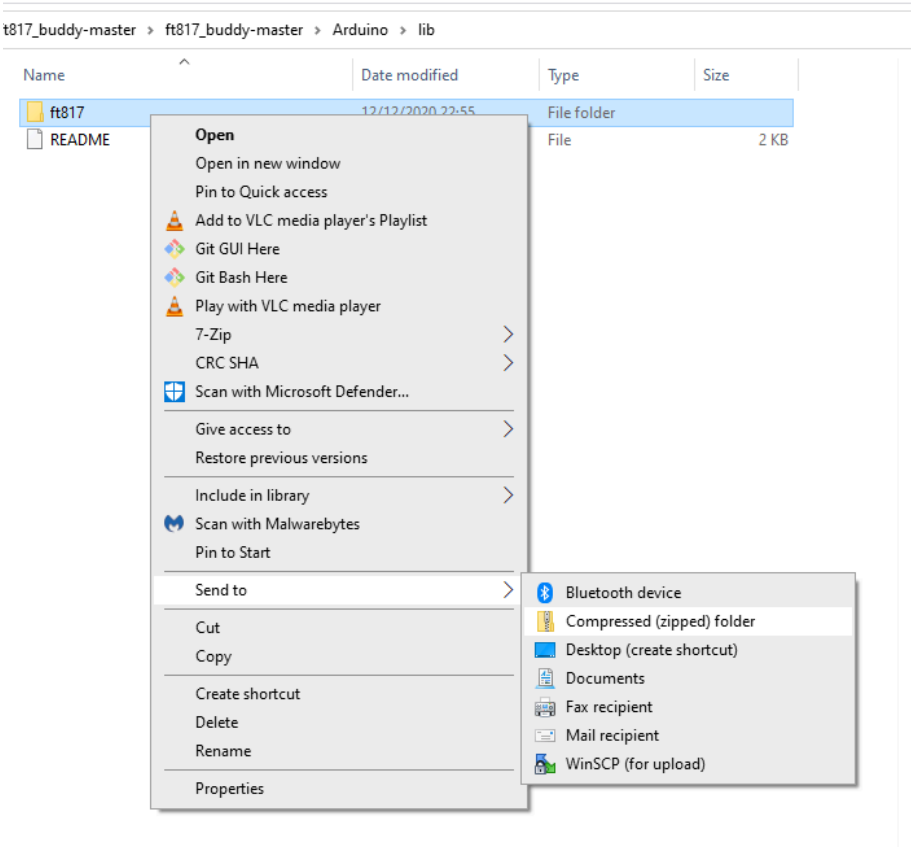
There may be some fiddling here depending on what software you're going to use to load your Arduino. Let's do an example using the standard Arduino IDE (if you're using another IDE you should be able to adapt the steps below)...

Arduino IDE example

Arduino stuff usually refers to a program as a "sketch" so I might start using the words sketch and program interchangeably here to refer to the main block of code that drives the Buddy. The other important blocks of code are in the ft817 library that we will import into Arduino IDE (and are then available for other sketches to use)...

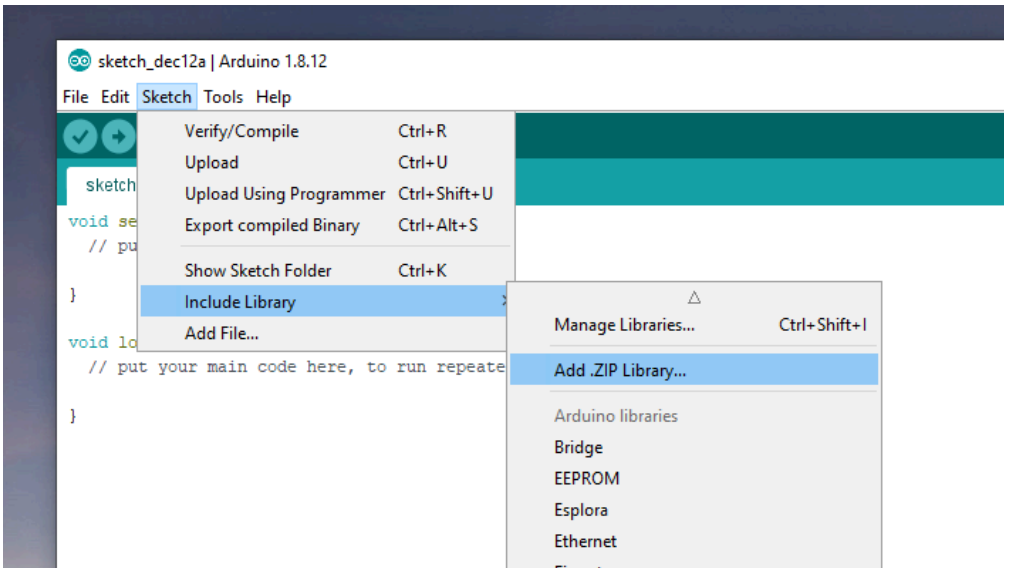
If you haven't already done so, download the Arduino IDE for your operating system from https://www.arduino.cc/en/software and install on your computer.

1. **Prepare the ft817 library to import into Arduino IDE.** Extract the files from the ZIP file you've downloaded from GitHub and head to the Arduino/lib/ folder. There will be a folder called "ft817" that contains the latest version of CO7WT's Arduino library included in this repository. Arduino IDE likes to import non-standard libraries as ZIP files so make a zip file of that ft817 folder (e.g. in Windows, right-click on it, Send to…, Compressed folder).



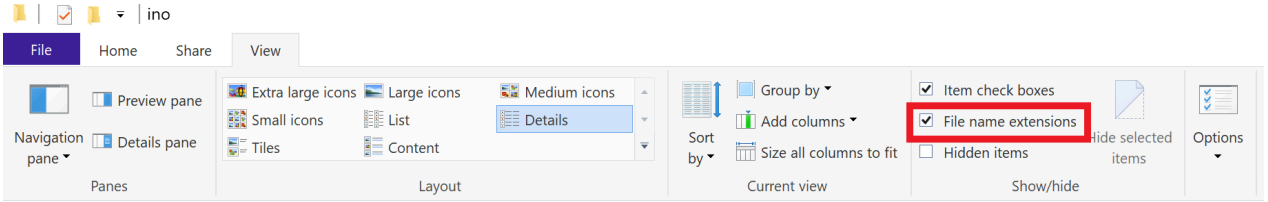
2. **Import the library.** Open up Arduino IDE and from the menu bar, go to Sketch, Include Library, Add .ZIP Library… and find the ZIP file you just created. Hit OK to import the ft817 library into Arduino IDE.

   Note, if you are following this install procedure to update your ft817 library from a previous install then you will need to first go to \Documents\Arduino\libraries and delete the old ft817 folder before restarting the Arduino IDE and following the previous instruction on importing the new ft817.zip library file.

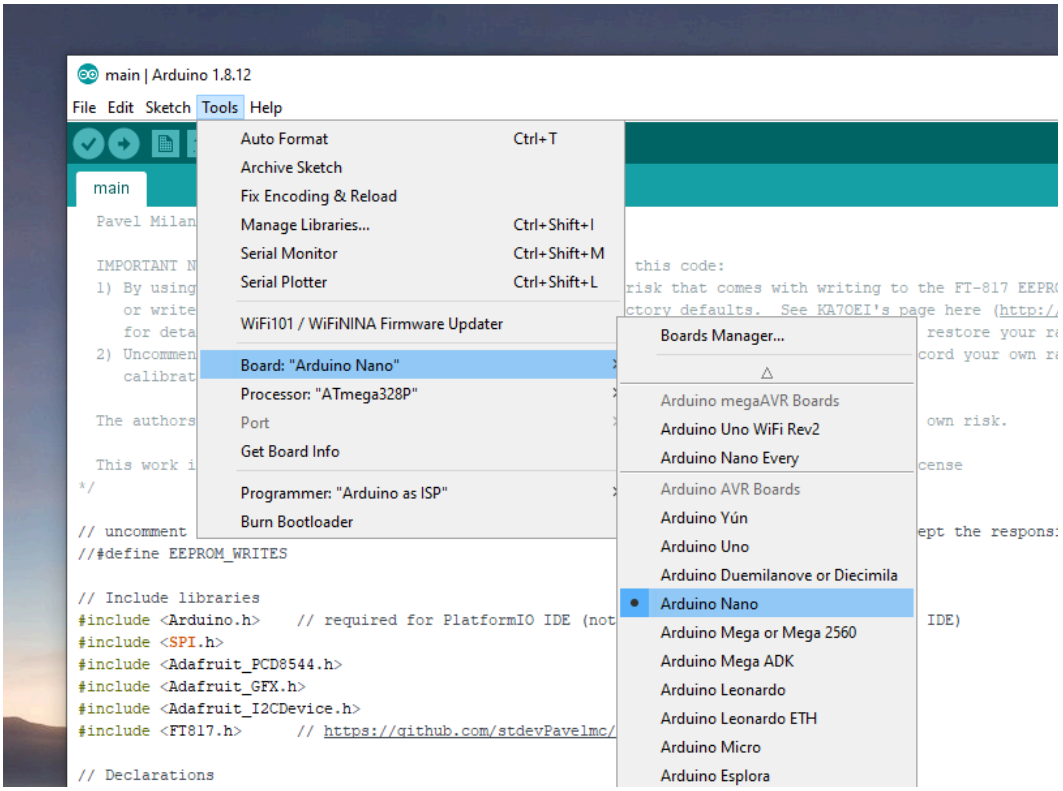


3. **Rename the main program.** Arduino IDE needs the main program to be named as a .ino file to recognise it as a sketch. Back in our files we downloaded and extracted from GitHub, head to the Arduino/src folder. Right-click on the main.cpp file and rename it to main.ino, changing the file extension.

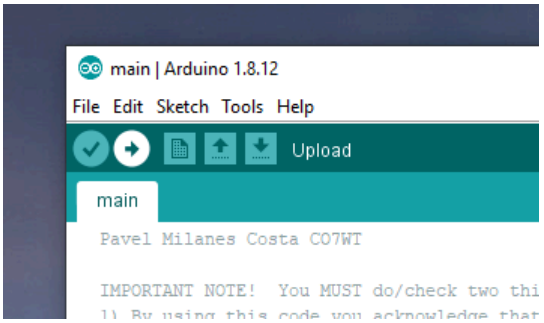
Note, if you are using a Windows machine the option to view and edit the extension may not be selected by default. This can be turned on in Windows Explorer by selecting View from the meny and ticking the 'File Name Extensions' checkbox.



4. **Open the main.ino program with Arduino IDE.** You can probably double-click on main.ino now to open it with Arduino IDE (or do File, Open…). You'll get a message that Arduino IDE requires the main program to be in a folder called main (fussy isn't it!)... hit OK to let Arduino IDE create that folder and move the main program inside.
5. **Remove one line of code.** Most IDEs will need a line of code to tell them to include all the Arduino stuff when they compile but Arduino IDE assumes this already… so if you're using Arduino IDE (as we are in this example) you need to delete (or comment out) the line near the top of the code that says `#include <Arduino.h>`
   To comment out simply add `//` to the start of the line.
6. **Acknowledge responsibility.** Remember, this is DIY and code is provided without warranty or liability. Read the statements at the top of the code and acknowledge you accept by removing the `//` from the line that allows EEPROM writes. Of course you don't have to if you don't want and you could tweak the functions of the Buddy code to only use CAT commands or whatever you feel comfortable with. Up to you.
7. **Option - RTC fitted.** If you've fitted the optional RTC module you'll need to uncomment the line to activate that (`#define RTC_FITTED`). If you don't have the RTC fitted, just leave it commented out and your Buddy will start up showing 00:00 as the time.
8. **Plug the Buddy into your computer** with a mini USB cable.
9. **Slide the Buddy's switch from OPERATE to PROGRAM.** Because we're about to program it.
10. **Select the board type and port in Arduino IDE.** In Arduino IDE, go to Tools, Board… and select Arduino Nano as shown below. You'll also need to go to Tools, Port… and select the right serial/COM port that your Buddy is connected to (if you get it wrong, don't worry, just try the other ports).



11. **Upload the sketch.** With the right board type and port selected, all that's left to do now is upload the sketch to the Arduino on the Buddy. Hit the Upload button, watch the progress. If it was successful Arduino IDE will report that it is 'Done uploading' and also report how much storage space was used to load the sketch and how much was used for global variables. You can now disconnect the Buddy and slide the switch back to OPERATE.



12. **Set your FT-817 CAT baud rate.** Before you connect the Buddy to the FT-817, set your FT-817's CAT baud rate (menu item number 14) to match the baud rate defined in the Buddy software (38400 baud at time of writing).

Operate

Connect the Buddy to your FT-817, slide the switches to ON and OPERATE, turn the FT-817 on.

The functionality of the FT-817 Buddy may have changed by the time you read this (software development is one of the fun parts of this project) but these general principles should help you get going:

- The timing of polling, reading and writing between the Arduino and the radio via the serial interface places limits on how responsive the Buddy display is. On pages where there are higher numbers of items requiring status polling the Buddy may feel sluggish while these status items are being periodically updated. Bear this in mind when customising the pages to suit your own operating (and maybe look for ways to improve the loop timings and contribute back to the project!);

- A function that causes the radio to transmit (e.g. “TUNon” to transmit a low power carrier signal to an ATU) will have a safety feature to prevent accidental transmission when handling the device. Hold down SHIFT at the same time as pressing the function button to activate;
- The current setting of the FT-817’s own softkeys (A,B,C) are reflected at the bottom of the Buddy display to aid operation. Note that, when changing the FT-817’s softkey functions, the Buddy display will not update until after the selection has been confirmed on the FT-817 e.g. by pressing “F” or one of the A,B,C keys… so you will need to look at the FT-817’s own display to change the A,B,C selection.
- The backlight will light for a few seconds on a single press of the LIGHT button. To toggle the backlight to stay on, press SHIFT+LIGHT. A further press of the LIGHT button will turn the backlight off.
- K1EL K16 keyer instructions are in the K16 User Manual. On the Buddy we can trigger the CMD/M1, M2, M3 and M4 buttons via the Arduino code and soft-keys. We don’t have a speed pot on the Buddy (saves space and a real electrical headache on the design) but we can control the CW speed by the K16’s “Fast Speed Change Feature” (see the user manual) which is basically using a paddle+CMD to increment the speed up or down. It works pretty well.

Troubleshooting

Problem	**Common cause**
My Buddy is taking a really long time to start! I’m just seeing a flower logo on the screen and then it’s really unresponsive!	This is what happens when serial lines between the Arduino and the FT-817 are broken. Most likely causes are the OPERATE / PROGRAM switch in the wrong position or you’ve soldered your Tx and Rx wires from the ACC cable on the wrong way around.
I’ve got nothing on the display!	Check the LCD pins are in the right position to make contact with the pads on the PCB. Try pressing the LCD onto the PCB above the contacts in case they aren’t quite making contact.
The contrast on my display is really weak!	This happened to me after fitting LCD modules that has been stored over winter in my freezing garage. In the Arduino code increased the value of `display.setContrast(57)` from 57 to 80 and re-flashed the Arduino… that was way too dark all over the LCD but it seemed to bring the pixels back to life... when I set contrast back down it was OK. I settled on a value of 62 for that one.

If you’ve done all the obvious checks (checked you’ve soldered every pin, no shorts or dry joints, etc.) and your fellow builders around you are stumped, maybe shout out for help on the Discussion page.

Customising the pages (needs improving!)

Every user of the FT-817 Buddy is likely to have different preferences as to what functions they wish to be assigned to the soft-keys on each page. The code that assigns functions to each soft-key is towards the start of the main program. Below is a description of how to customise the code to meet your needs and some [example pages can be found here](#).

Start of the section for Page 0 (there are three pages, 0,1,2)

Start of configuration of Page 0, Soft-key 1 (of 6)

This line labels the soft-key

This line assigns the function to be triggered by the soft-key

This line assigns the function that collects the status of the soft-key if any)

```
// Page0 items
// SOFT-KEY 1
String   page0SoftkeyLabel1 = "TUNon ";           // 6 characters
void  page0SoftkeyFunction1() {tuneSignalOn();}
boolean page0SoftkeyStatus1() {}
// SOFT-KEY 2
String   page0SoftkeyLabel2 = "IPO";              // 3 characters
void  page0SoftkeyFunction2() {radio.toggleIPO();}       // EEPROM write
boolean page0SoftkeyStatus2() {return radio.getIPO();}   // EEPROM read
// SOFT-KEY 3
...
```

Write the soft-key label between the " " marks. Include spaces and pad the label to 6 or 3 characters to maintain display spacing

Place function calls between the { } marks.

tuneSignalOn() is called from the main program

tuneSignalOn() has no corresponding status item

toggleIPO() is called from the ft817 library so it is prefixed as radio.toggleIPO()

The value returned by radio.getIPO() is the status of the IPO setting.

Work in progress, trying to improve this illustration! :-)	// Page0 items
	// SOFT-KEY 1
	String page0SoftkeyLabel1 = "TUNon "; // 6 characters
	void page0SoftkeyFunction1() {tuneSignalOn();}
	boolean page0SoftkeyStatus1() {}
	// SOFT-KEY 2
	String page0SoftkeyLabel2 = "IPO"; // 3 characters
	void page0SoftkeyFunction2() {radio.toggleIPO();} // EEPROM write
	boolean page0SoftkeyStatus2() {return radio.getIPO();} // EEPROM read
	// SOFT-KEY 3
	...	

```
// Page0 items
// SOFT-KEY 1
String   page0SoftkeyLabel1 = "TUNon ";           // 6 characters
void  page0SoftkeyFunction1() {tuneSignalOn();}
boolean page0SoftkeyStatus1() {}
// SOFT-KEY 2
String   page0SoftkeyLabel2 = "IPO";              // 3 characters
void  page0SoftkeyFunction2() {radio.toggleIPO();}          // EEPROM write
boolean page0SoftkeyStatus2() {return radio.getIPO();}      // EEPROM read
// SOFT-KEY 3
...
```