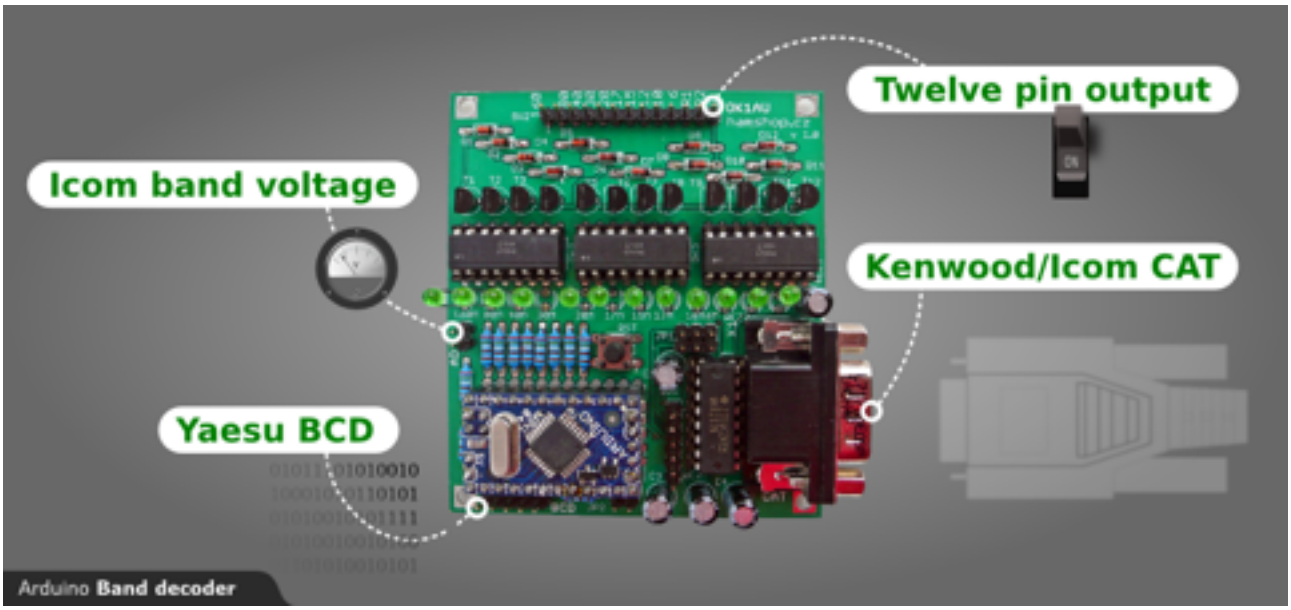
Background.

During HFI2014 in Hyderabad, I came across a ham from Rajkot selling Furuno PA2500 amplifier. This was total VFM. I decided to interface this amplifier to my shack and automate PTT, band switch and Antenna switch operations. The default ALC line from Furuno runs a positive swinging voltage which cannot be connected directly to radio (might even damage the ALC buffer in case of Icom radios) and needs a polarity switching. The following text describes how I went about making these modifications and this documentation became a necessity since many hams were interested in knowing the details and it was practically not possible to type out mails etc to each individual. This project is complete " copy & paste " from different websites/ projects and I would be happy to compile and share my experiences for the sake of other hams.

These are the links…

1. Band decoder is from http://www.remoteqth.com/wiki/index.php?page=Arduino+band+decoder



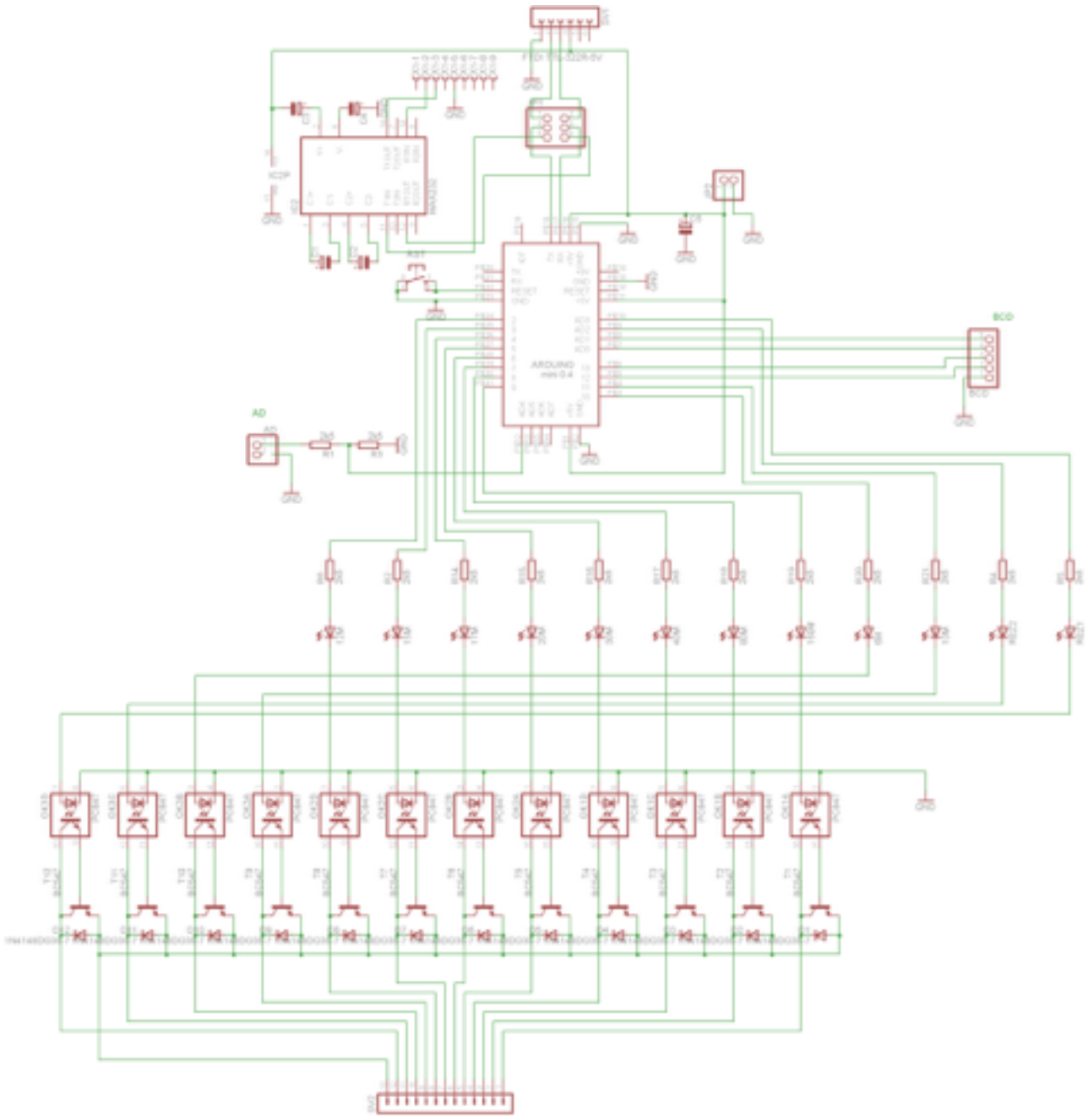
I ordered PCB from hamshop.cz and procured components locally. Arduino used was a cheap 1/2 USD unit from eBay.

Part	Value	Device	Package	Description	PCS
6M, 10M, 12M, 15M, 17M, 20M, 30M, 40M, 80M, 160M, REZ1, REZ2		LED3MM	LED3MM	LED	12
ARDUINO1		ARDUINO-MINI	ARDUINO-MINI	1*6 - 2x / 1*3 - 2x / 1*5 - 3x	1
C1, C2, C3, C4	10u	CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR, European symbol	4
C5	>10u (22u)	CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR, European symbol	1
D1-D12	1N4148	1N4148DO35-7	DO35-7	DIODE	12
IC2	MAX232	MAX232	DIL16	RS232 TRANSEIVER + patice	1
OK1-OK3	PC847	PC847	DIL16	SHARP OPTO COUPLER	3
R1-R21	2k2	R-EU_0204/7	0204/7	RESISTOR, European symbol	14
RST		10-XX	B3F-10XX	OMRON SWITCH	1
T1-T12	BC547	BC547	TO92	NPN TRANSISTOR	12
X1		F09HPS	F09HP	SUB-D	1
BCD, SV2, AD, JP2, SV1, JP1		1X	34	PIN HEADER	34
					1

This decoder works with any ICOM, YAESU or KENWOOD radios.

The default setting is to give an output for each of the 10 bands (160,80,40,30,20,17,15,12,10 & 6 m) and 2 auxiliary outputs.
There is an LED which will light for each band output. This decoder can thus give 12 different outputs as per the code.

Schematic



The default code is here

https://github.com/ok1cdj/ARDUINO-Bandecoder

This decoder is based on arduino mini and I have customised the software for clubbing bands as required for switching furuno LPFs as well as 4 antennas.

This is my modified sketch for switching furuno LPF and then 4 antennas. Currently only 2 antennas are used and with a small modification of code can be implemented for all 4 antenna positions.
****

/* Kenwood Bandecoder
!!! Warning this version was tested only with TS-590 in standalone mode !!!
Please let me know, if works also with other radios....

Tested with: TS-590
Should work with: TS-480, TS-990 and TS-2000
by Ondrej Kolonicny OK1CDJ, ondra@ok1cdj.com, Enric EA3NR
http://www.remoteqth.com
you can get KIT here:
http://hamshop.cz/remoteqth-com-c29/stavebnice-band-dekoderu-s-arduino-i230/
Change log:
v 1.1.4 9.1.2014
- chages by EA3NR TNX
If the radio is powered off and the band decoder remains powered on, the band decoder output remains indefinitely showing the last selected band and never goes out of this status... It happens also if you
power the radio off and then you power the radio on immediately.
v 1.1.3 17.2.2014
- added 29 MHz and 51 and 52 MHz as 10m and 6m output TNX to Adam OK2IPW
v 1.1.2 19.1.2014
- solved bug in BCD output TNX to Frank - IZ4YDF
v 1.1.1 10.12.2O13
- minor change in serial comunication, now is source tested also with TS-590
v 1.1 13.9.2013
- added BCD output like in YAESU RADIO to connect another accessories
WARNING - only standard bands like 80,40,20,15,10 are suported on other bands are all BCD pins in low state

v 1.0 - 20.3.2013

Customisations by VU2CPL 28.10.2015

Band outputs changed to drive LPF of Furuno PA2500 Linear amp and 4:1 Antenna switch.
Yaesu BCD and Icom CIV is retained as it is. Yaesu BCD updated for all bands.

Tested OK with TS990S

```
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
// uncomment next line if band decoder is connected alone to the radio and not sniffing
comunication
#define ALONE_MODE
// YAESU BCD
/* A B C D dec
 160 1 0 0 0 1
 80 0 1 0 0 2
 40 1 1 0 0 3
 20 1 0 1 0 5
 15 1 1 1 0 7
 10 1 0 0 1 9
 */
// IO port definition for BCD output
int D=12;
int C=13;
int B=A0;
int A=A1;
// IO port definition for bands
int b160_1624 = 9;
int b80_2436 = 8;
int b80_3660 = 7;
int b4030_0610 = 6;
int b20_1018 = 5;
int b17151210_1830 = 4;
int ANT1 = 3;
int ANT2 = 2;
int ANT3 = 11;
int ANT4 = 10;
// variables
char inByte;  // incoming byte from serial RX
String buffer = "";  // empty string to store incoming serial data -- a buffer
boolean stringComplete = false;  // whether the string is complete or not

// Setup procedure
void setup()
{
  Serial.begin(57600);   // setup serial speed
  buffer.reserve(64);    // reserve 64 bytes for the incoming data buffer; UARTS's buffer is
only 64 bytes long... :-P
  pinMode(b160_1624, OUTPUT);
  pinMode(b80_2436, OUTPUT);
  pinMode(b80_3660, OUTPUT);
  pinMode(b4030_0610, OUTPUT);
  pinMode(b20_1018, OUTPUT);
  pinMode(b17151210_1830, OUTPUT);
  pinMode(ANT1, OUTPUT);
```

```
  pinMode(ANT2, OUTPUT);
  pinMode(ANT3, OUTPUT);
  pinMode(ANT4, OUTPUT);
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
}

void loop() {
  // send request for frequency only when bandecoder is connected alone to the radio
#ifdef ALONE_MODE
  Serial.print("IF;");
  // wait for the transmission of outgoing serial data from the Arduino to the radio to be
completed
  Serial.flush();
  // we don't have to be always polling the radio frequency... we can wait for a while...XD
  delay(100);
#endif
  if (Serial.available() > 0 ) {
    // get the first byte
    inByte = (char) Serial.read();
    // when the first byte is 'I' then we have to continue reading the input data
    if (inByte == 'I') {
      buffer += inByte;
      while (Serial.available() > 0) {
        inByte = (char) Serial.read();
        buffer += inByte;
        // the reading procedure finishes when we read the ';' character
        if (inByte == ';') {
          stringComplete = true;
        }
      }
    }
    // update band output only if the complete data has been received
    if (stringComplete) {
      writeBand();
      // clear the RX buffer
      buffer = "";
      stringComplete = false;
    }
  }
}
void writeBand() {
  String str_if = buffer;
  String qrg ;
  // tady je kmitocet od 5 do 10 znaku
  qrg = str_if.substring(5,10);
  //na tretim miste ma byt 1 pak do jde na HIGH
  // 160m band
  if ((qrg[0] == '0')&&(qrg[1] == '1'))
  {
```

```
  digitalWrite(b160_1624, HIGH);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, HIGH);
  digitalWrite(ANT2, LOW);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
}
// 80m band
if (qrg[1] == '3')
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, HIGH);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, HIGH);
  digitalWrite(ANT2, LOW);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);;
}
// 40m band
if (qrg[1] == '7')
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, HIGH);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, HIGH);
  digitalWrite(ANT2, LOW);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
}
// 30m band
```

```
if ((qrg[0] == '1')&&(qrg[1] == '0'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, HIGH);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, HIGH);
  digitalWrite(ANT2, LOW);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
}
// 20m band
if ((qrg[0] == '1')&&(qrg[1] == '4'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, HIGH);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);;
}
// 17m band
if ((qrg[0] == '1')&&(qrg[1] == '8'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, HIGH);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
```

```
}
// 15m band
if ((qrg[0] == '2')&&(qrg[1] == '1'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, HIGH);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
}
// 12m band
if ((qrg[0] == '2')&&(qrg[1] == '4'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, HIGH);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
}
// 10m band
if (((qrg[0] == '2')&&(qrg[1] == '8'))||((qrg[0] == '2')&&(qrg[1] == '9')))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, HIGH);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
```

```
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
}
//6m band
if ((qrg[0] == '5')&&(qrg[1] == '0'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
}
if ((qrg[0] == '5')&&(qrg[1] == '1'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
}
if ((qrg[0] == '5')&&(qrg[1] == '2'))
{
  digitalWrite(b160_1624, LOW);
  digitalWrite(b80_2436, LOW);
  digitalWrite(b80_3660, LOW);
  digitalWrite(b4030_0610, LOW);
  digitalWrite(b20_1018, LOW);
  digitalWrite(b17151210_1830, LOW);
  digitalWrite(ANT1, LOW);
  digitalWrite(ANT2, HIGH);
  digitalWrite(ANT3, LOW);
  digitalWrite(ANT4, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
```

```
    digitalWrite(C, LOW);
    digitalWrite(D, HIGH);
  }
}
```

**********’

Just copy everything between stars and paste into arduino compiler.

I had an ABS box which I had procured during HFI2014 in Hyderabad and I decided to use the same box for the decoder and antenna switch.

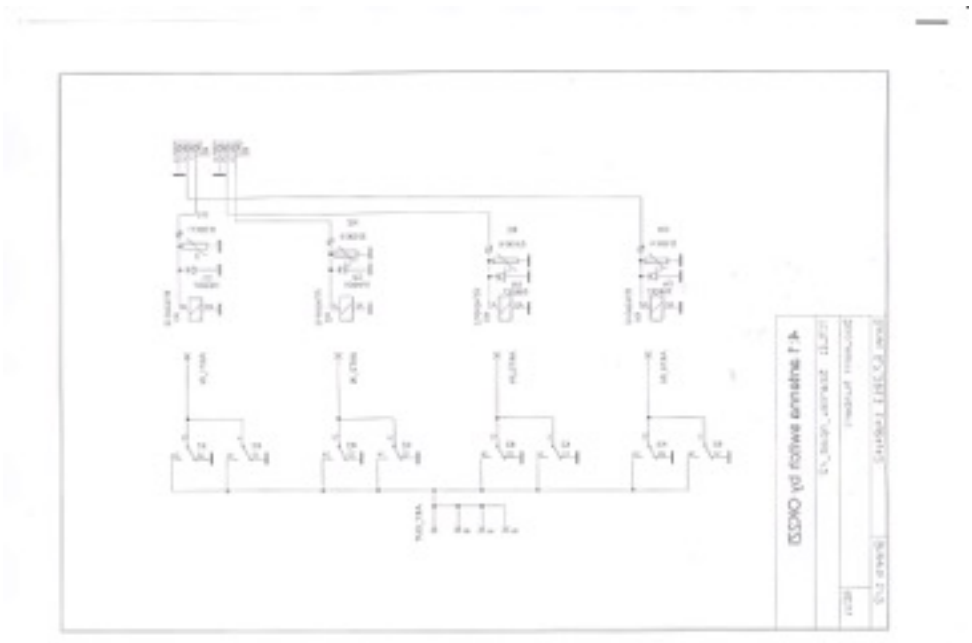
The output from PCB switches (-) of a relay, but furuno switching is with a (+) signal. I had to rewire the output so that when there is an output from decoder, this is in the form of a +12V which can switch a relay with 50ma current. (I used a small plugin PCB on top of the decoder)

The band switch was ordered as a kit form eBay. the link is here.

http://www.ebay.com/itm/KIT-4-1-remote-antenna-switch-DIY-cheap-SO-239-KIT-/261669572320?hash=item3cecb8cee0:g:focAAOSwGvhT~1k7

As per data, this can easily run more than a KW. There is a PCB available form the same seller for a cheaper 2:1 switch.

This is the circuit

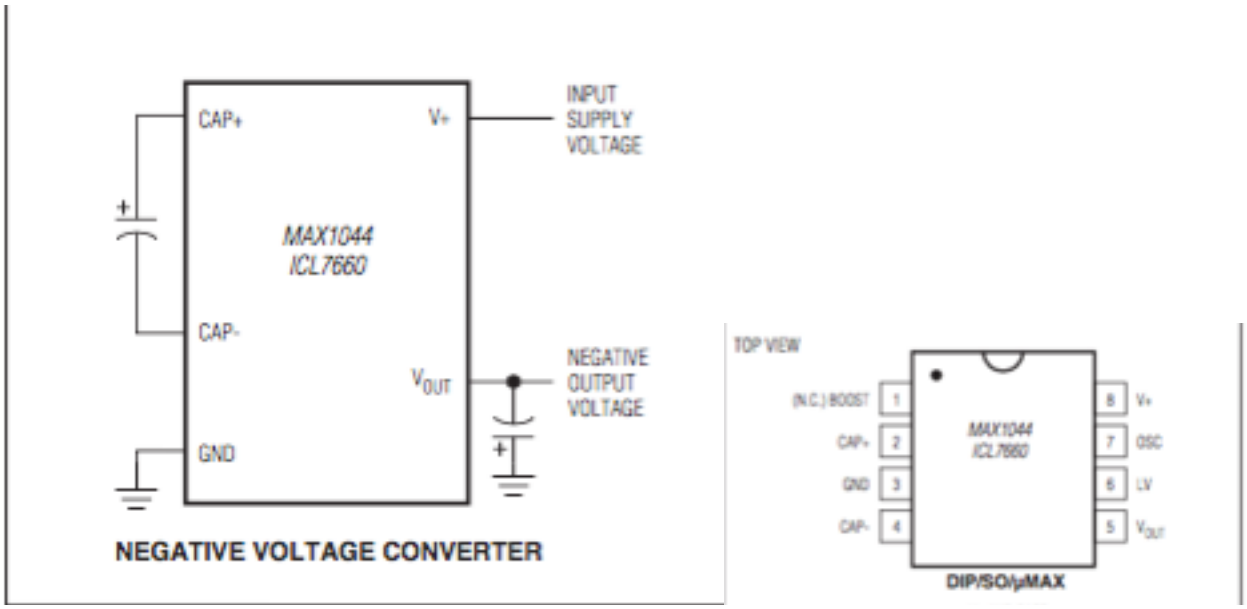


The furuno has a builtin 3 bit BCD decoder which decodes the band signals from a mating transceiver. But this arrangement works only for switching upto 8 bands. Now for our ham bands (HF) we have 10 bands and BCD data from a modern transceiver is in the 4 bit BCD format. There is no way we can retain this arrangement without loosing some data. Hence it was decided to bypass the BCD decoder inside furuno (implemented with CD4028 on the control board) and use direct switching of LPF. I opened up my furuno and

located the 13pin relimate connector from control board to LPF board. all connections for switching BPF was removed from control PCB end and was routed to the back of AMP into an 8 PIN connector. The PTT connection was rewired to an RCA socket with a new hole drilled. This prevents possibility of having 2 LPFs selected at the same time. e.g. through front panel thumb wheel switch selects 40m, and software selects 20m. After this modification, the thumb wheel switch is inactive. If required, this connector can be re wired to work with thumbwheel switch.

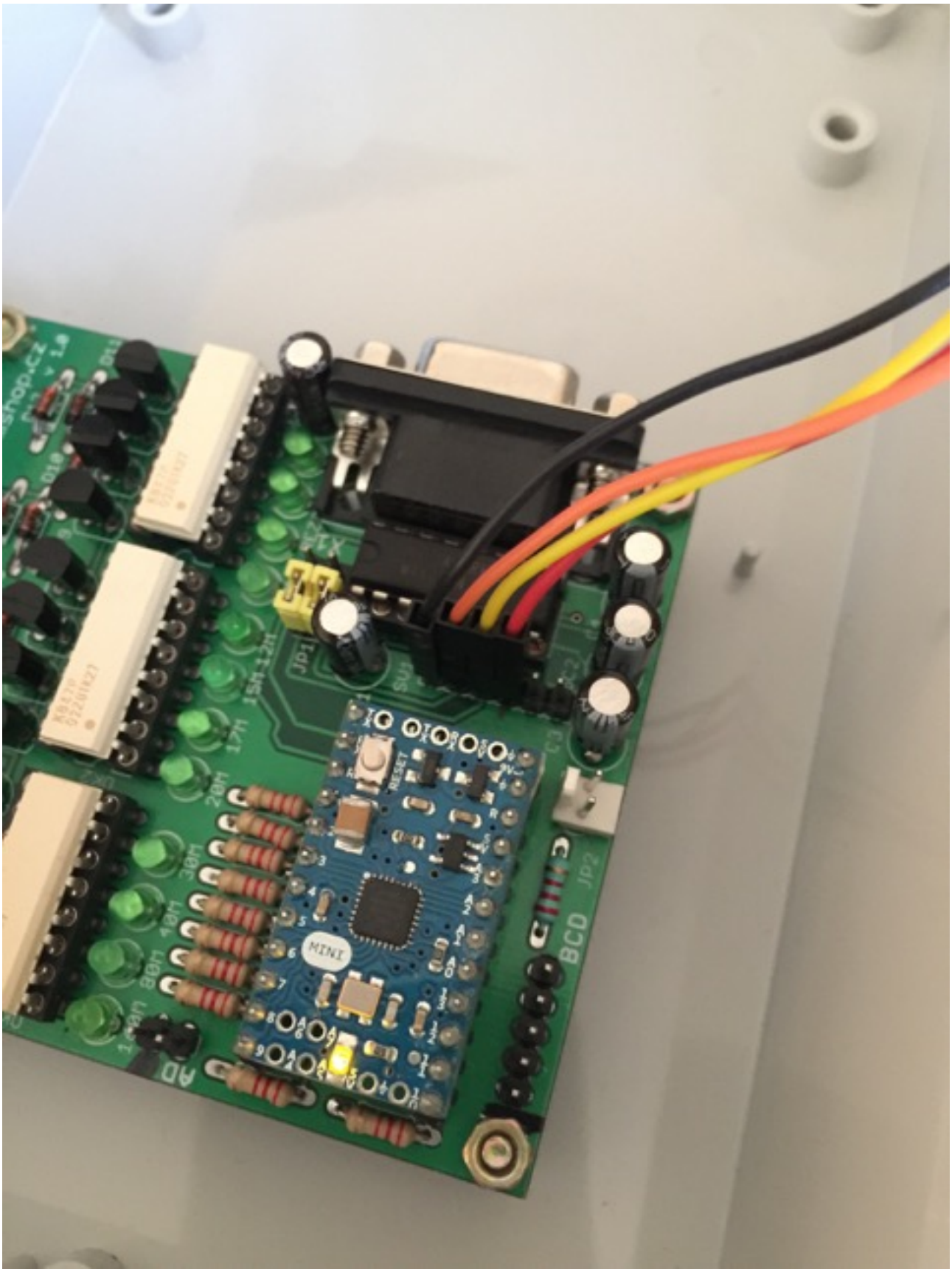
The ALC circuit was modified with an LT7660 which can convert a positive voltage to a negative one with just 2 capacitors in the circuit. This circuit was assembled on a piece of common board. This voltage will have to be calibrated for your rig and set accordingly. Basic idea is to find what negative voltage is required to limit your rigs output power to less than 60 W (Max input for furuno). you can use a 9v battery with a series resistor as a current limiter and feed it through a pot(with + terminal to rigs ground and - terminal to ALC connection). Once you get this value of voltage, set your LT7660 output to this value while furuno is running. My Kenwood radio expects -3 to -7 volts as ALC from amplifier.

This is the circuit I used for ALC conversion.

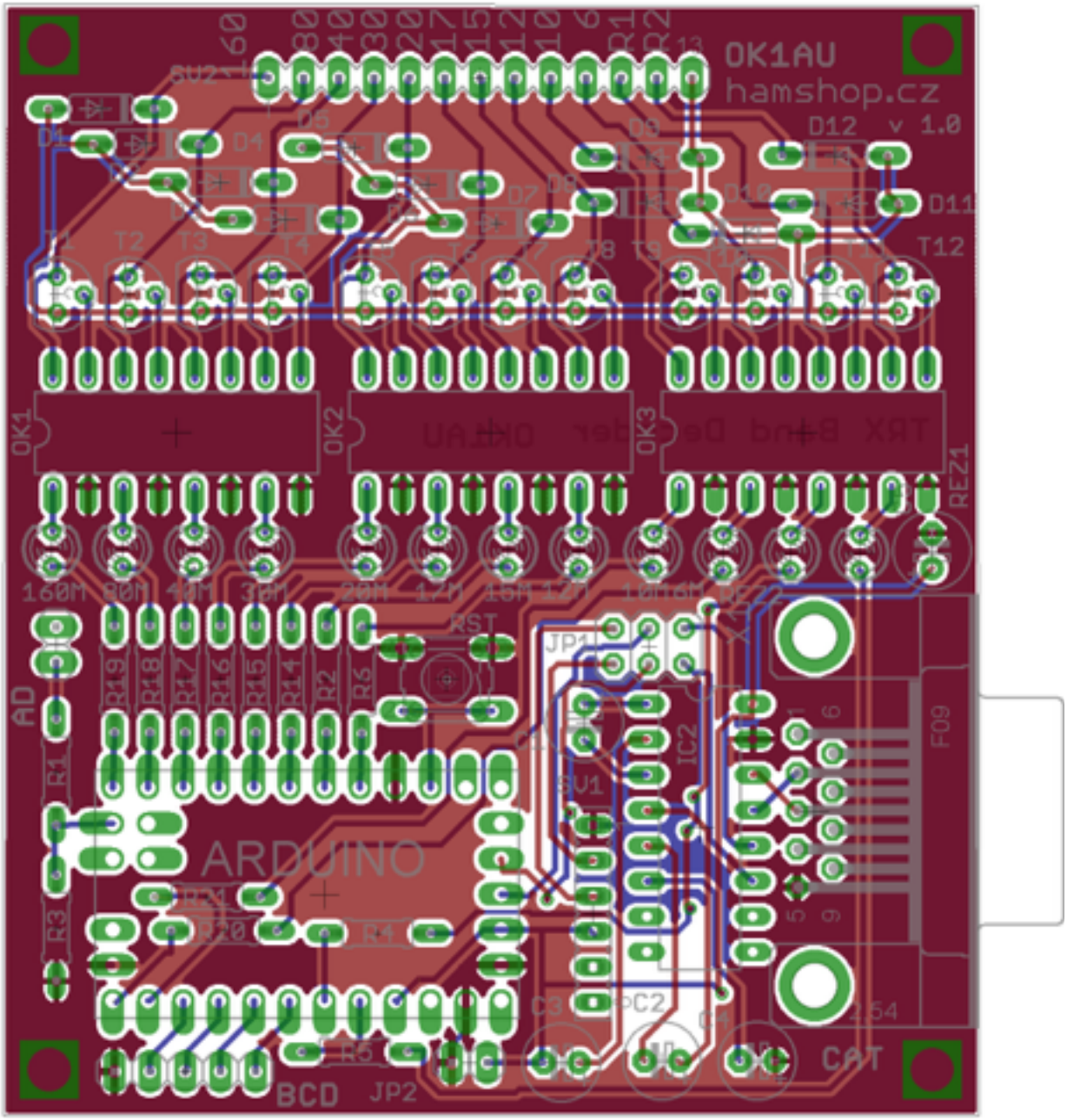


I am attaching some pictures along with this document so that you can see how it was implemented in my shack.

Assembled PCB of band decoder with arduino mini attached.



Band decoder PCB



Pictures of the enclosure with connectors attached.

Another view showing SO-239 connectors attached.

The 9 pin connector is wired to radio with a straight through serial port cable. DC jack accepts 12V DC (consumption is less than 25mA and depends on the relays used). 8 Pin

connector provides switching voltage for Furuno LPFs. Cable has 8 pin female connectors on both sides so that disconnection is easy. In case manual switching of LPFs inside Furuno is required, you can wire a 6 pole rotary switch and fix it in a small box with proper cable terminated in a 8pin socket.

My next step is to attach 2 rotary selectors with sufficient poles in the box incase manual switching of antenna or LPF is required. LEDs for indicating active antenna/ LPF will also be added.

Good luck to all. If you have suggestions/ comments/ queries write to vu2cpl@gmail.com

Best regards

Manoj VU2CPL