# “Andromeda” Console Implementation Notes

Kjell has created a new PCB for the ANDROMEDA front panel prototype. This has a lot more encoders and pushbuttons, and uses an MCP23017 to interface 16 buttons via I2C.

# Control Layout

# 

Figure 1: Initial V2 Prototype (for test only)



Figure 2: V3 Prototype

## Encoder Functions:

From a software perspective, each dual encoder can have A (upper) and B (lower) encoders plus a “click” function. s/w numbering allows up to 20 numbers making encoders 1-40 and switches 1-20

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Encoder**  **Schematic numbers** | **Prototype V3** | | | **Early Prototype V2** | | |
| **Function** | **Encoder report #** | **Switch**  **s/w#** | **Function** | **Encoder report #** | **Switch**  **s/w#** |
| (2A) | RX1 AF | 1 |  | RX1 AF | 1 |  |
| (2B) | RX1 AGC | 2 |  | RX1 AGC | 2 |  |
| 2 click | RX1 mute |  | 1 | RX1 mute |  | 1 |
| 3A | Filter high | 3 |  | Filter high | 3 |  |
| 3B | Filter low | 4 |  | Filter low | 4 |  |
| 3 Click | Filter reset |  | 3 | Filter reset |  | 3 |
| 4A | Div’ty gain | 5 |  | Div’ty gain | 5 |  |
| 4B | Div’ty phase | 6 |  | Div’ty phase | 6 |  |
| 4 Click | Fast/slow |  | 5 | Fast/slow |  | 6 |
| 5A | Multifunction | 7 |  | Mic gain | 7 |  |
| 5B | N/A | 8 |  | Drive | 8 |  |
| 5 Click | Multi |  | 7 | (none) |  | 7 |
| 6A | RX2 AF | 9 |  | RX2 AF | 9 |  |
| 6B | RX2 AGC | 10 |  | RX2 AGC | 10 |  |
| 6 Click | RX2 mute |  | 9 | RX2 mute |  | 9 |
| 7A | RIT | 11 |  | Compander | 11 |  |
| 7B | N/A | 12 |  | Master AF | 12 |  |
| 7 Click | RIT/XIT step |  | 11 | Comp on/off |  | 11 |
| 8A | N/A |  |  | Multifunction | 13 |  |
| 8B | N/A |  |  | Squelch | 14 |  |
| 8 Click | N/A |  |  | Multi |  | 13 |

(note encoder switch numbers have been done so that encoder report number 13 corresponds to switch 13. If it is a dual encoder that also has encoder 14 on the same footprint, 14 can’t have a switch)

## Pushbutton Functions

(encoder buttons start at 1; pushbuttons start at 21)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pushbutton report number | Prototype V3 | | | Prototype V2 | | |
| s/w scan code | CPU pin | function | s/w scan code | CPU pin | function |
| 1 | 6 | DIG6 | Encoder 1 click | 6 | DIG6 | Encoder 1 click |
| 3 | 17 | DIG53 | Encoder 2 click | 17 | DIG53 | Encoder 2 click |
| 5 | 8 | DIG23 | Encoder 3 click | 8 | DIG23 | Encoder 3 click |
| 7 | 9 | DIG29 | Encoder 4 click | 9 | DIG29 | Encoder 4 click |
| 9 | 27 | DIG61 | Encoder 5 click | 25 | DIG61 | Encoder 5 click |
| 11 | 0 | DIG50 | Encoder 6 click | 26 | DIG62 | Encoder 6 click |
| 13 | - | - | No encoder | 27 | DIG63 | Encoder 7 click |
| 21 | 1 | ANA3 | F1 | 1 | ANA3 | F1 |
| 22 | 2 | ANA4 | F2 | 2 | ANA4 | F2 |
| 23 | 3 | ANA5 | F3 | 3 | ANA5 | F3 |
| 24 | 4 | ANA6 | F4 | 4 | ANA6 | F4 |
| 25 | 5 | ANA7 | F5 | 5 | ANA7 | F5 |
| 26 | 7 | DIG9 | F6 / NR | 30 | DIG66 | PS ON/F6 |
| 27 | 16 | DIG52 | F7 / NB | 31 | DIG67 | SINGLE CAL/F8 |
| 28 | 35 | DIG69 | F8 / SNB | 32 | DIG68 | 2 TONE/F7 |
| 29 | 10 | DIG40 | RIT+ | 0 | ANA2 | RIT- |
| 30 | 11 | DIG41 | BAND- | 7 | DIG9 | NR |
| 31 | 12 | DIG42 | MODE- | 10 | DIG40 | RIT+ |
| 32 | 13 | DIG43 | SDR ON | 11 | DIG41 | BAND- |
| 33 | 14 | DIG44 | BAND+ | 12 | DIG42 | MODE- |
| 34 | 15 | DIG45 | MODE+ | 13 | DIG43 | SDR ON |
| 35 | 18 | ANA6 | FILTER- | 14 | DIG44 | BAND+ |
| 36 | 19 | ANA5 | FILTER+ | 15 | DIG45 | MODE+ |
| 37 | 20 | DIG54 | RIT ON | 16 | DIG52 | NB |
| 38 | 21 | DIG55 | A/B TOGGLE | 18 | DIG54 | RIT ON |
| 39 | 22 | DIG56 | MOX | 19 | DIG55 | A/B TOGGLE |
| 40 | 23 | DIG57 | TUNE | 20 | DIG56 | MOX |
| 41 | 24 | DIG58 | CLICK TUNE | 21 | DIG57 | TUNE |
| 42 | 25 | DIG59 | VFO LOCK | 22 | DIG58 | CLICK TUNE |
| 43 | 26 | DIG60 | A > B | 23 | DIG59 | VFO LOCK |
| 44 | 28 | DIG62 | VFO SYNC | 24 | DIG60 | A > B |
| 45 | 29 | DIG63 | RIT- | 28 | DIG64 | SPLIT |
| 46 | 30 | DIG64 | SPLIT | 29 | DIG65 | B > A |
| 47 | 31 | DIG65 | B > A | 33 | DIG69 | SNB |
| 48 | 32 | DIG66 | PS ON |  |  |  |
| 49 | 34 | DIG68 | 2 TONE TEST |  |  |  |

(DIG67, scan code 33, is unused)

(encoder numbers correspond to the 1st encoder number. So encoder 3,4 has switch scan code 3)

## Indicator Functions

(starting at 1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Indicator report number | Prototype V3 | | Prototype V2 | |
| s/w number | function | s/w number | function |
| 1 | 0 | VFO Lock on | 0 | VFO Lock on |
| 2 | 1 | Puresignal on | 1 | Puresignal on |
| 3 | 2 | MOX | 2 | MOX |
| 4 | 3 | Tune 1 | 3 | Tune 1 |
| 5 | 4 | Click Tune On | 4 | Click Tune On |
| 6 | 5 | VFO Sync On | 5 | COMP On |
| 7 | 6 | NB On | 6 | SQL On |
| 8 | 7 | RIT On | 7 | RIT On |
| 9 | 8 | Tune 2 | 8 | Tune 2 |
| 10 | 9 | NR On | No LED |  |
| 11 | 10 | SNB On | No LED |  |
| 12 | 11 | XIT On | No LED |  |

# Arduino Software Structure

## Concept for Operation

The front panel controller will exchange simple messages with the SDR application (eg Thetis) so that all the “radio” functionality is in one place, to minimise latency. The controller will debounce buttons and encoders and pass the results for processing.

The serial queue to / from the PC will use normal Arduino library code. Simple TX commands will be generated for each pushbutton or encoder event. RX commands will be processed for indicator setting, to allow s/w version to be queried and potentially to allow encoder rates to be adjusted.

As a temporary measure for debug, the older Odin approach and Odin based code is used.

## Serial Command Set

This should be quite simple. CAT compatible commands proposed.

|  |  |  |
| --- | --- | --- |
| **Control type** | **Command** | **Meaning** |
| VFO encoder | Up: ZZZUnn;  Down: ZZZDnn; | nn = number of clicks (0-99) |
| Other encoder | ZZZEnnm; | nn = encoder number:  0: unused  1-20: encoder 1-20, clockwise  51-70: encoder 1-20, anticlockwise  m=number of clicks (0-9) |
| Pushbutton (including encoder) | ZZZPnnm; | nn=button number (1-99, 0=n/a);  m=0: released; m=1: pressed; m=2: long pressed (not supported yet) |
| Indicator | ZZZInnm; | nn=indicator number (1-99, 0=n/a);  m=0: unlit; m=1: lit |
| Query hardware version | ZZZH;  Response ZZZHmm; | mm=0: V2 (early prototype);  mm=1: V3 (current prototype) |
| Query s/w Version | ZZZS;  Response ZZZSmmm; | mmm= s/w version |
| Query product type | CAT message ZZZT;  Response ZZZTn; | n=1: Andromeda  n=2: Aries |
| Query/change encoder increments | ZZZX;  Response ZZZXmmn; | mm = VFO encoder increment;  n= other VFO increment  (typically 1,2 or 4) |

# Arduino Libraries

Arduino Due has the Atmel SAM3X8E ARM Cortex-M3 processor. Any input can have an interrupt and it may be possible to select the h/w input debounce. But needs some specific libraries

* “DueFlashStorage” library is an EEPROM equivalent library for Due
* Timer – Due specific “DueTimer”
* Serial – there seem to be several. “Serial” is the normal one. “SerialUSB” is the “native” port not used in this application.

# Rotary Encoders

Interrupt driven code is poor at debouncing. It does work well with bounce-free optical encoders.

Zacsketches/quadrature works well for the VFO: I’m getting 2400 steps per revolution.

ClickEncoder works well for the other “mechanical” encoders.

# Arduino Hardware



## I2C ports

I2C port 1 (DIG20, DIG21) connect to MCP23017. Chip address = 0x20. Class: Wire

I2C port 2 (SCL1, SDA1) connect to J10 for TX2 interface (note no pullup resistors) Class: Wire1

## CPU Pin Assignments – 2nd Prototype

#define V3HARDWARE

Encoder inputs and LEDs are all connected to Arduino pins. Note that the analogue pins can be assigned digital functions too.

15 pushbuttons are connected to an MCP23017; 20 are directly connected to the Arduino (total=35). The Encoder pushbuttons currently end up with the last s/w numbers, because the EncoderHandleButton function (encoders.c) just subtracts the 1st encoder button number to find which encoder it is assigned to. But that could be done a different way eg with a lookup table.

New hardware pin IDs:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pin Name** | **Function** | **Connector** | **Pin Name** | **Function** | **Connector** |
| Digital 0 / RX0 | Reserved USB | PWML | Digital 31 | LED10 | Digital |
| Digital 1/ TX0 | Reserved USB | PWML | Digital 32 | LED4 | Digital |
| Digital 2 | VFO encoder | PWML | Digital 33 | LED5 | Digital |
| Digital 3 | VFO encoder | PWML | Digital 34 | LED6 | Digital |
| Digital 4 | Encoder 2A | PWML | Digital 35 | SW63 | Digital |
| Digital 5 | Encoder 2A | PWML | Digital 36 | LED13 | Digital |
| Digital 6 | Enc 2 SW | PWML | Digital 37 | LED7 | Digital |
| Digital 7 | Encoder 2B | PWML | Digital 38 | LED8 | Digital |
| Digital 8 | Encoder 2B | PWMH | Digital 39 | Encoder 6B | Digital |
| Digital 9 | SW17 | PWMH | Digital 40 | SW11 | Digital |
| Digital 10 | Encoder 3A | PWMH | Digital 41 | SW12 | Digital |
| Digital 11 | Encoder 3A | PWMH | Digital 42 | SW13 | Digital |
| Digital 12 | Encoder 6B | PWMH | Digital 43 | SW14 | Digital |
| Digital 13 LED | Reserved LED | PWMH | Digital 44 | SW15 | Digital |
| Digital 14 TX3 | Encoder 3B | communication | Digital 45 | SW16 | Digital |
| Digital 15 RX3 | Encoder 3B | Communication | Digital 46 | Encoder 6A | Digital |
| Digital 16 TX2 | SW58 | Communication | Digital 47 | Encoder 6A | Digital |
| Digital 17 RX2 | Encoder 4A | Communication | Digital 48 | Encoder 7A | Digital |
| Digital 18 TX1 | LED12 | Communication | Digital 49 | Encoder 7A | Digital |
| Digital 19 RX1 | (Unused) | Communication | Digital 50 | ENC 7 SW | Digital |
| Digital 20 SDA | Reserved SDA | Communication | Digital 51 | LED11 | Digital |
| Digital 21 SCL | Reserved SCL | Communication | Digital 52 | SW18 | Digital |
| Digital 22 | Encoder 4A | Digital | Digital 53 | Enc3 SW | Digital |
| Digital 23 | Enc4 SW | Digital | Analog 0 | LED2 | Analog In |
| Digital 24 | Encoder 4B | Digital | Analog 1 | LED3 | Analog In |
| Digital 25 | Encoder 4B | Digital | Analog 2 | (unused) | Analog In |
| Digital 26 | SW59 | Digital | Analog 3 | SW46 | Analog In |
| Digital 27 | Encoder 5A | Digital | Analog 4 | (unused) | Analog In |
| Digital 28 | Encoder 5A | Digital | Analog 5 | SW48 | Analog In |
| Digital 29 | Enc5 SW | Digital | Analog 6 | SW49 | Analog In |
| Digital 30 | LED1 | Digital | Analog 7 | SW50 | Analog In |
| Digital 54 | SW9 | MCP23017 | Digital 62 | SW56 | MCP23017 |
| Digital 55 | SW1 | MCP23017 | Digital 63 | SW10 | MCP23017 |
| Digital 56 | SW2 | MCP23017 | Digital 64 | SW8 | MCP23017 |
| Digital 57 | SW3 | MCP23017 | Digital 65 | SW7 | MCP23017 |
| Digital 58 | SW4 | MCP23017 | Digital 66 | SW21 | MCP23017 |
| Digital 59 | SW5 | MCP23017 | Digital 67 | (unused) | MCP23017 |
| Digital 60 | SW6 | MCP23017 | Digital 68 | SW20 | MCP23017 |
| Digital 61 | Enc6 SW | MCP23017 | Digital 69 | SW22 | MCP23017 |

10 mechanical Encoders (4 dual, 2 single) + VFO:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Kjell number** | **A function** | **B function** | **s/w number** | **Enc A pin** | **Enc B pin** |
| 2 | RX1 AF gain | RX1 RF gain | 0, 1 | DIG4, DIG5 | DIG7, DIG8 |
| 3 | Filter High cut | Filter Low cut | 2, 3 | DIG10, DIG11 | DIG14, DIG15 |
| 4 | Diversity gain | Diversity phase | 4, 5 | DIG17, DIG22 | DIG24, DIG25 |
| 6 | RX2 AF gain | RX2 RF gain | 6, 7 | DIG46, DIG47 | DIG12, DIG39 |
| 5 | Multifunction | ---- | 8 | DIG27, DIG28 | ---- |
| 7 | RIT | ---- | 9 | DIG48, DIG49 | ---- |

29 s/w connected pushbuttons + 6 encoder press; 0-19 direct, 20-35 via MCP32017 (in blue):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Kjell SW number** | **s/w** | **Function** | **pin** | **Kjell SW number** | **s/w** | **Function** | **pin** |
| SW1 | 21 | A/B toggle | DIG55 | SW56 | 28 | VFO SYNC | DIG62 |
| SW2 | 22 | MOX | DIG56 | SW20 | 34 | 2 TONE TEST | DIG68 |
| SW3 | 23 | TUNE | DIG57 | SW21 | 32 | PURESIGNAL ON | DIG66 |
| SW4 | 24 | Click TUNE | DIG58 | SW22 | 35 | F8 key / SNB | DIG69 |
| SW5 | 25 | VFO LOCK | DIG59 | SW58 | 1 | F1 key | ANA3 |
| SW6 | 26 | A > B | DIG60 | SW59 | 2 | F2 key | ANA4 |
| SW7 | 31 | B > A | DIG65 | SW63 | 3 | F3 key | ANA5 |
| SW8 | 30 | SPLIT | DIG64 | SW50 | 4 | F4 key | ANA6 |
| SW9 | 20 | RIT ON | DIG54 | SW46 | 5 | F5 key | ANA7 |
| SW10 | 29 | RIT - | DIG63 | ENC2 SW | 6 | RX1 MUTE | DIG6 |
| SW11 | 10 | RIT + | DIG40 | ENC3 SW | 17 | FILTER RESET | DIG53 |
| SW12 | 11 | BAND - | DIG41 | ENC4 SW | 8 | DIVERSITY FINE/COARSE | DIG23 |
| SW13 | 12 | MODE - | DIG42 | ENC5 SW | 9 | Multifunction | DIG29 |
| SW14 | 13 | SDR ON | DIG43 | ENC6 SW | 27 | RX2 MUTE | DIG61 |
| SW15 | 14 | BAND + | DIG44 | ENC7 SW | 0 | RIT/XIT toggle | DIG50 |
| SW16 | 15 | MODE + | DIG45 | SW49 | 18 | FILTER- | ANA6 |
| SW17 | 7 | F6 key / NR | DIG9 | SW48 | 19 | FILTER+ | ANA5 |
| SW18 | 16 | F7 key / NB | DIG52 | SW23 |  | Power (not via Arduino) | N/A |

(DIG67, S/w 33, unused)

12 s/w connected LEDs (note LED9 is directly powered by Vdd, not s/w):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kjell LED number** | **s/w number** | **Function** | **Pin** | **Comment** |
| LED1 | 5 | VFO SYNC | DIG30 |  |
| LED2 | 6 | NB | ANA0 |  |
| LED3 | 2 | MOX | ANA1 |  |
| LED4 | 3 | TUNE on | DIG32 |  |
| LED5 | 4 | CTUNE on | DIG33 |  |
| LED6 | 0 | LOCK on | DIG34 |  |
| LED7 | 1 | PS on | DIG37 |  |
| LED8 | 7 | RIT on | DIG38 |  |
| LED10 | 8 | TUNE on | DIG31 | ATU ready for Andromeda |
| LED11 | 9 | NR | DIG51 |  |
| LED12 | 10 | SNB | DIG18 |  |
| LED13 | 11 | XIT | DIG36 |  |

## CPU Pin Assignments – 1st Prototype

#define V2HARDWARE

Encoder inputs and LEDs are all connected to Arduino pins. Note that the analogue pins can be assigned digital functions too.

16 pushbuttons are connected to an MCP23017; 18 are directly connected to the Arduino (total=34). The Encoder pushbuttons currently end up with the last s/w numbers, because the EncoderHandleButton function (encoders.c) just subtracts the 1st encoder button number to find which encoder it is assigned to. But that could be done a different way eg with a lookup table.

Remember the Ext MOX input with Odin hardware presents itself as an extra button; we don’t have one with V2 hardware.

New hardware pin IDs:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pin Name** | **Function** | **Connector** | **Pin Name** | **Function** | **Connector** |
| Digital 0 / RX0 | Reserved USB | PWML | Digital 31 | LED10 | Digital |
| Digital 1/ TX0 | Reserved USB | PWML | Digital 32 | LED4 | Digital |
| Digital 2 | VFO encoder | PWML | Digital 33 | LED5 | Digital |
| Digital 3 | VFO encoder | PWML | Digital 34 | LED6 | Digital |
| Digital 4 | Encoder 2A | PWML | Digital 35 | Encoder 8B | Digital |
| Digital 5 | Encoder 2A | PWML | Digital 36 | Encoder 8B | Digital |
| Digital 6 | Enc 2 SW | PWML | Digital 37 | LED7 | Digital |
| Digital 7 | Encoder 2B | PWML | Digital 38 | LED8 | Digital |
| Digital 8 | Encoder 2B | PWMH | Digital 39 | Encoder 6B | Digital |
| Digital 9 | SW17 | PWMH | Digital 40 | SW11 | Digital |
| Digital 10 | Encoder 3A | PWMH | Digital 41 | SW12 | Digital |
| Digital 11 | Encoder 3A | PWMH | Digital 42 | SW13 | Digital |
| Digital 12 | Encoder 6B | PWMH | Digital 43 | SW14 | Digital |
| Digital 13 LED | Reserved LED | PWMH | Digital 44 | SW15 | Digital |
| Digital 14 TX3 | Encoder 3B | communication | Digital 45 | SW16 | Digital |
| Digital 15 RX3 | Encoder 3B | Communication | Digital 46 | Encoder 6A | Digital |
| Digital 16 TX2 | Encoder 5B | Communication | Digital 47 | Encoder 6A | Digital |
| Digital 17 RX2 | Encoder 4A | Communication | Digital 48 | Encoder 7A | Digital |
| Digital 18 TX1 | Encoder 7B | Communication | Digital 49 | Encoder 7A | Digital |
| Digital 19 RX1 | Encoder 7B | Communication | Digital 50 | Encoder 8A | Digital |
| Digital 20 SDA | Reserved SDA | Communication | Digital 51 | Encoder 8A | Digital |
| Digital 21 SCL | Reserved SCL | Communication | Digital 52 | SW18 | Digital |
| Digital 22 | Encoder 4A | Digital | Digital 53 | Enc3 SW | Digital |
| Digital 23 | Enc4 SW | Digital | Analog 0 | LED2 | Analog In |
| Digital 24 | Encoder 4B | Digital | Analog 1 | LED3 | Analog In |
| Digital 25 | Encoder 4B | Digital | Analog 2 | SW10 | Analog In |
| Digital 26 | Encoder 5B | Digital | Analog 3 | SW46 | Analog In |
| Digital 27 | Encoder 5A | Digital | Analog 4 | SW47 | Analog In |
| Digital 28 | Encoder 5A | Digital | Analog 5 | SW48 | Analog In |
| Digital 29 | Enc5 SW | Digital | Analog 6 | SW49 | Analog In |
| Digital 30 | LED1 | Digital | Analog 7 | SW50 | Analog In |
| Digital 54 | SW9 | MCP23017 | Digital 62 | Enc7 SW | MCP23017 |
| Digital 55 | SW1 | MCP23017 | Digital 63 | Enc8 SW | MCP23017 |
| Digital 56 | SW2 | MCP23017 | Digital 64 | SW8 | MCP23017 |
| Digital 57 | SW3 | MCP23017 | Digital 65 | SW7 | MCP23017 |
| Digital 58 | SW4 | MCP23017 | Digital 66 | SW21 | MCP23017 |
| Digital 59 | SW5 | MCP23017 | Digital 67 | SW19 | MCP23017 |
| Digital 60 | SW6 | MCP23017 | Digital 68 | SW20 | MCP23017 |
| Digital 61 | Enc6 SW | MCP23017 | Digital 69 | SW22 | MCP23017 |

14 mechanical Encoders:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Kjell number** | **A function** | **B function** | **s/w number** | **Enc A pin** | **Enc B pin** |
| 2 | RX1 AF gain | RX1 RF gain | 0, 1 | DIG4, DIG5 | DIG7, DIG8 |
| 3 | Filter High cut | Filter Low cut | 2, 3 | DIG10, DIG11 | DIG14, DIG15 |
| 4 | Diversity gain | Diversity phase | 4, 5 | DIG17, DIG22 | DIG24, DIG25 |
| 5 | Mic Gain | Drive | 6, 7 | DIG27, DIG28 | DIG16, DIG26 |
| 6 | RX2 AF gain | RX2 RF gain | 8, 9 | DIG46, DIG47 | DIG12, DIG39 |
| 7 | Comp | Master AF gain | 10, 11 | DIG48, DIG49 | DIG18, DIG19 |
| 8 | Squelch | n/a | 12, 13 | DIG50, DIG51 | DIG35, DIG36 |

34 s/w connected pushbuttons:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Kjell SW number** | **s/w** | **Function** | **pin** | **Kjell SW number** | **s/w** | **Function** | **pin** |
| SW1 | 19 | A/B toggle | DIG55 | SW19 | 31 | SINGLE CAL | DIG67 |
| SW2 | 20 | MOX | DIG56 | SW20 | 32 | 2 TONE TEST | DIG68 |
| SW3 | 21 | TUNE | DIG57 | SW21 | 30 | PURESIGNAL ON | DIG66 |
| SW4 | 22 | Click TUNE | DIG58 | SW22 | 33 | SNB | DIG69 |
| SW5 | 23 | VFO LOCK | DIG59 | SW46 | 1 | No function (menu 1) | ANA3 |
| SW6 | 24 | A > B | DIG60 | SW47 | 2 | No function (menu 2) | ANA4 |
| SW7 | 29 | B > A | DIG65 | SW48 | 3 | No function (menu 3) | ANA5 |
| SW8 | 28 | SPLIT | DIG64 | SW49 | 4 | No function (menu 4) | ANA6 |
| SW9 | 18 | RIT ON | DIG54 | SW50 | 5 | No function (menu 5) | ANA7 |
| SW10 | 0 | RIT - | ANA2 | ENC2 SW | 6 | RX1 MUTE | DIG6 |
| SW11 | 10 | RIT + | DIG40 | ENC3 SW | 17 | FILTER RESET | DIG53 |
| SW12 | 11 | BAND - | DIG41 | ENC4 SW | 8 | DIVERSITY FINE/COARSE | DIG23 |
| SW13 | 12 | MODE - | DIG42 | ENC5 SW | 9 | No function | DIG29 |
| SW14 | 13 | SDR ON | DIG43 | ENC6 SW | 25 | RX2 MUTE | DIG61 |
| SW15 | 14 | BAND + | DIG44 | ENC7 SW | 26 | COMP ON/OFF | DIG62 |
| SW16 | 15 | MODE + | DIG45 | ENC8 SW | 27 | SQL ON/OFF | DIG63 |
| SW17 | 7 | NR | DIG9 |  |  |  |  |
| SW18 | 16 | NB | DIG52 | SW23 |  | Power (not via Arduino) | N/A |

9 s/w connected LEDs:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kjell LED number** | **s/w number** | **Function** | **Pin** | **Comment** |
| LED1 | 5 | COMP on | DIG30 | By encoder 7 |
| LED2 | 6 | SQL on | ANA0 | By encoder 8 |
| LED3 | 2 | MOX | ANA1 |  |
| LED4 | 3 | TUNE on | DIG32 |  |
| LED5 | 4 | CTUNE on | DIG33 |  |
| LED6 | 0 | LOCK on | DIG34 |  |
| LED7 | 1 | PS on | DIG37 |  |
| LED8 | 7 | RIT on | DIG38 |  |
| LED9 |  | Power | n/a | Powered directly by Vdd |
| LED10 | 8 | TUNE on | DIG31 | ATU ready for Andromeda |

## Available functions for Each Control

### Indicators and Pushbuttons

For pushbuttons and indicators: try to use the same command set as for the menu buttons. See if any of the following are not available as menu buttons, and add them if needed.

|  |  |
| --- | --- |
| Pushbuttons (including encoder “press”)   * A/B VFO select * MOX * TUNE * AF MUTE (explicit RX1/2 needed) * Filter reset * Band + * Band – * Mode + * Mode – * AGC speed * NB step * NR step * SNB on/off * ANF on/off * Off /RIT on / XIT on * RIT + * RIT – * A>B * B>A * A/B swap * Split * CTUNE * Lock * Radio Start/Stop * Squelch on/off * Attenuation Step * VOX on/off * Diversity fast/slow step * Compander on/off * Puresignal on/off * Puresignal two tones test * Puresignal single cal (deprecated) * MON on / off * Diversity Enable * VFO Sync * Clear RIT/XIT * Filter + * Filter – * VAC1 on/off * VAC2 on/off * Display Centre * F1-F8 button | Indicators   * MOX * TUNE * ATU Ready * RIT on * Split selected * CTune selected * Lock selected * NB off/on * NR off/on * SNB off/on * ANF off/on * Squelch on/off * VFO A/B * Compander on/off * Puresignal on/off * LED lit if encoder 2nd function selected * VFO sync * XIT   Go through and check if these available! |

### Encoders

* RX1 AF gain
* RX2 AF gain
* Sub RX AF Gain
* Master AF gain
* RX1 stereo balance
* RX2 stereo balance
* Sub RX stereo balance
* RX1 AGC
* RX2 AGC
* RX1 step attenuation
* RX2 step attenuation
* RX1 Squelch
* RX2 Squelch
* Selected RX Filter high cut
* Selected RX Filter low cut
* TX Drive
* TX Mic Gain
* VFO A tune
* VFO B tune
* RIT
* VOX gain
* VOX delay
* Compander threshold
* CW sidetone
* CW speed
* Diversity Gain
* Diversity Phase
* Multifunction
* Display Pan
* Display Zoom

# Andromeda Prototype 3

The intention is to replace Arduino Due with a smaller cheaper Arduino. “Arduino Nano Every” has 19 usable I/O: D0-D12, A0-A3, A6-A7 (D13 is LED; A4-5 are I2C; we don’t need to use serial TX1, RX1 so they are available as GPIO) at very low cost.

## Arduino Issues

Arduino Nano Every has an 8 bit ATMEGA4809 processor. I need to change some libraries

* EEPROM.h
* Timer – new code will be needed; no known libraries for ‘4809
* Serial connection to PCB – “Serial” is the normal one.
* You have to install “Arduino Mega AVR Boards” in the Arduino console Boards manager
* Interrupt driven VFO encoder library doesn’t support ‘4809

Interrupt driven code is poor at debouncing. It does work well with bounce-free optical encoders.

Zacsketches/quadrature works well for the VFO: I’m getting 2400 steps per revolution.

Modified ClickEncoder works well for the other “mechanical” encoders. I’m already using modified encoder code, not in a library. Should be simple to change it again to pass 2 bits of data into it, rather than having I/O pin numbers passed to it.

|  |  |
| --- | --- |
|  |  |

We could have up to 8 MCP23017 on one I2C bus (3 ID pins). MCP23017 are 3mA I/O: so not suitable for driving LEDs in a matrix.

The processor is only 8 bit: so change variables wherever possible to be byte (8 bit unsigned) int8\_t (signed 8 bit) or int (16 bit) rather than long (32 bit)

## I/O Approaches

Current I/O requirement:

1 off VFO encoder -> 2 inputs; 2 off single encoder -> 4 inputs; 4 off dual encoder -> 16 inputs; 35 off pushbutton -> 35 inputs; 11 LED -> 11 outputs; total 68 I/O

Alternative:

Add 2xMCP23017 and we have 51 I/O available. 3xMCP23017 -> 67 I/O (not quite enough). 4xMCP23017 -> comfortable enough. But suppose I scanned switches in a matrix –a side benefit would be the debounce code was much easier.

Possible I/O approach:

22 encoder inputs; Encoders in 8x8 matrix; 11 LEDs. Total 49 I/O

### I2C

I2C is normally clocked at 100KHz, but can be faster; use wire.SetClock() to adjust. Could be 400KHz or 1MHz for example. I2C is quite slow: at 400KHz, ~100us per transaction.

In principle the processor doesn’t need to wait for data to come back – could call “wire.requestfrom” and do something else then call “wire.read” later

# Keypad scanning

I will need to use a simple sequencer to scan the pushbuttons. Don’t attempt to cope with more than one button press. Assert a new column low every software tick, then read the rows and look for a row with one or more bits at zero. Only one row driven low at a time. A “helper” function reads the row input and assigns a row code: 0: no button pressed; 1-8: row 0-7 pressed; FF: more than one pressed.

Columns driven using pseudo open drain outputs from MCP23017, so if there are shorted columns it doesn’t matter. Outputs only drive a logic 0 level; inactive column outputs are disabled by turning the pins into inputs.



Figure 3: Suggested keypad scanning sequencer

# Use of Timeslots

Suggest consider 2ms timeslot as the starting point.

@400KHz, byte read over I2C ~45us. @1MHz, will be ~20us. SPI could be 10x faster. Will need to segregate I2C activity to avoid stalling waiting for it.

In one timeslot we need to:

Do one update of the key matrix (involves I2C read, then I2C write);

Update half of the encoders; (I2C 16 bit read);

Update any LEDs.

Keypad – we could initiate the next column write at the end of the time slot, so we don’t wait for it.

So work to an assumption that we will, in one “tick”:

1. Read switch matrix row input, update sequencer
2. Read encoder 16 bit input
3. Update half the encoders
4. Process serial messages
5. Update any LEDs
6. Update Arduino LED
7. Write switch matrix column

# Device, Pin Allocations

|  |  |  |  |
| --- | --- | --- | --- |
| **MCP23017** | **Number:** | **1** | **A2=0 A1=0 A0=0** |
| **Port A** | **Encoder inputs** | **Port B** | **Encoder inputs** |
| GPA7 | Encoder 1 1A (RX1 AF) | GPB7 | Encoder 5 1A (IF Shift High) |
| GPA6 | Encoder 1 1B | GPB6 | Encoder 5 1B |
| GPA5 | Encoder 1 2A (RX1 RF) | GPB5 | Encoder 5 2A (IF Shift Low) |
| GPA4 | Encoder 1 2B | GPB4 | Encoder 5 2B |
| GPA3 | Encoder 3 1A (RX2 AF) | GPB3 | Encoder 7 1A (Diversity Gain) |
| GPA2 | Encoder 3 1B | GPB2 | Encoder 7 1B |
| GPA1 | Encoder 3 2A (RX2 RF) | GPB1 | Encoder 7 2A (Diversity Phase) |
| GPA0 | Encoder 3 2B | GPB0 | Encoder 7 2B |

|  |  |  |  |
| --- | --- | --- | --- |
| **MCP23017** | **Number:** | **2** | **A2=0 A1=0 A0=1** |
| **Port A** | **Switch matrix column OUTPUT** | **Port B** | **Switch Matrix Row INPUT** |
| GPA7 | LED11 | GPB7 | Row 8 |
| GPA6 | LED10 | GPB6 | Row 7 |
| GPA5 | LED9 | GPB5 | Row 6 |
| GPA4 | Column 5 | GPB4 | Row 5 |
| GPA3 | Column 4 | GPB3 | Row 4 |
| GPA2 | Column 3 | GPB2 | Row 3 |
| GPA1 | Column 2 | GPB1 | Row 2 |
| GPA0 | Column 1 | GPB0 | Row 1 |

## Arduino Pins

|  |  |  |  |
| --- | --- | --- | --- |
| **Arduino:** | **Arduino Nano Every** |  |  |
| DIG0 / TX | VFO encoder A | DIG11 | LED5 () |
| DIG1 / RX | VFO encoder B | DIG12 | LED6 () |
| DIG2 | Encoder 9A (RIT) | DIG13 | Arduino LED |
| DIG3 | Encoder 9B | A0 | LED7 () |
| DIG4 | Encoder 10A | A1 | LED8 () |
| DIG5 | Encoder 10B | A2 | Encoder 11A (Multi) |
| DIG6 | LED1 () | A3 | Encoder 11B |
| DIG7 | LED2 () | A4 / SDA | I2C SDA |
| DIG8 | LED3 () | A5 / SCL | I2C SCL |
| DIG9 | LED4 () | A6 | Encoder 12A |
| DIG10 | PWM for display brightness | A7 | Encoder 12B |

## Switch Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Column 5**  **2GPA4** | **Column 4**  **2GPA3** | **Column 3**  **2GPA2** | **Column 2**  **2GPA1** | **Column 1**  **2GPA0** |
| **Row 8**  **2GPB7** | Encoder 5 | Filter + | Band + | SDR on | Softkey 8 |
| **Row 7**  **2GPB6** | Encoder 7 | Filter - | Band - | MOX | Softkey 7 |
| **Row 6**  **2GPB5** | Encoder 9 | SPLIT | A>B | Tune | Softkey 6 |
| **Row 5**  **2GPB4** | SHIFT | RIT + | RIT/XIT | PS on | Softkey 5 |
| **Row 4**  **2GPB3** |  | A/B | Mode + | 2 Tone | Softkey 4 |
| **Row 3**  **2GPB2** |  | VFO Sync | Mode - | Encoder 1 | Softkey 3 |
| **Row 2**  **2GPB1** |  | VFO Lock | B>A | Encoder 3 | Softkey 2 |
| **Row 1**  **2GPB0** |  | CTUNE | RIT- | Encoder 11 | Softkey 1 |

5x8 switch matrix – gives us 5 spare switches

(I’ve assigned this so that only column 1 is used under the display; only column 2 to the left of the display. But make changes if needed, as long as I get a modified table!)

Switch scan code is row number + (Column number-1) \*8

## LEDs

Simple lookup from the software number to a CPU pin

|  |  |  |
| --- | --- | --- |
| **Arduino pin** | **s/w number** | **Function** |
| D6 | 1 | MOX |
| D7 | 2 | ATU Tune |
| D8 | 3 | ATU OK |
| D9 | 4 | Diversity enable |
| D11 | 5 | Puresignal enable |
| D12 | 6 | SHIFT |
| A0 | 7 | CTUNE |
| A1 | 8 | RIT |
| MCP23017 #2. GPA5 | 9 | XIT |
| MCP23017 #2. GPA6 | 10 | SYNC |
| MCP23017 #2. GPA7 | 11 | VFO Lock |

## Switch Matrix Wiring



Every row has a pullup resistor. Columns are driven by the MCP23017; one column will be 0, the others will be 1. If no buttons pressed, the Row word reads out all 1s ie 0xFF

## Main LCD Brightness Control

There is a need to be able to set the brightness of the main LCD display. Implemented with a simple PWM output, which has ~1KHz PRF. Setting changed if a defined encoder turned while a defined pushbutton pressed; setting stored in EEPROM and retrieved at power up. Setting written back ~5s after it was last updated.

# Software Combinations

We have several PCB layouts and several different software programs. This section explains which are which.

All of the software is downloaded from <https://github.com/laurencebarker/Andromeda_front_panel>

(click “clone or download” then “Download zip”). Save to a folder, and you will need code from one of the subfolders.

|  |  |  |
| --- | --- | --- |
| Front panel |  | Andromeda 7 encoder + VFO 1st prototype |
| Processor | Arduino Due | |
| Sketch | andromeda\_panel\_odin\_based.ino | |
| Sub-folder | front panel\odin based sketch\andromeda\_panel\_odin\_based | |
| Customisation steps | This sketch supports both this hardware build and the 6 encoder 2nd prototype hardware build. The sketch needs to be edited slightly to select the target PCB:  Edit the sketch tab “globalinclude.h” to comment out “V3HARDWARE” and include “V2HARDWARE”:  //  // define V2HARDWARE to compile for Kjell's Andromeda early prototype PCB  // design proving initally using Odin software  //  **#define V2HARDWARE 1**  //  // define V3HARDWARE to compile for Kjell's final Andromeda prototype PCB  // design proving initally using Odin software  //  **//#define V3HARDWARE 1** | |
| Works with PC code: | Thetis 2.6.7+  PowerSDR mrx ps v3.4.9+ | |
| Functionality | This sketch is based on the Arduino sketch for Odin. It uses most of the same code as Odin. Much of the logic is provided at the Arduino end; the Arduino controls PowerSDR or Thetis by CAT commands. | |
| Status | This Arduino sketch was intended as an interim build for Andromeda, to be replaced once the radio logic was moved to Thetis. It can be used with PowerSDR. | |

|  |  |  |
| --- | --- | --- |
| Front panel |  | Andromeda 6 encoder + VFO 2nd prototype |
| Processor | Arduino Due | |
| Sketch | andromeda\_panel\_odin\_based.ino | |
| Sub-folder | front panel\odin based sketch\andromeda\_panel\_odin\_based | |
| Customisation steps | This sketch supports both this hardware build and the 7 encoder 1st prototype hardware build. The sketch needs to be edited slightly to select the target PCB:  Edit the sketch tab “globalinclude.h” to comment out “V2HARDWARE” and include “V3HARDWARE”:  //  // define V2HARDWARE to compile for Kjell's Andromeda early prototype PCB  // design proving initally using Odin software  //  **//#define V2HARDWARE 1**  //  // define V3HARDWARE to compile for Kjell's final Andromeda prototype PCB  // design proving initally using Odin software  //  **#define V3HARDWARE 1** | |
| Works with PC code: | Thetis 2.6.7+  PowerSDR mrx ps v3.4.9+ | |
| Functionality | This sketch is based on the Arduino sketch for Odin. It uses most of the same code as Odin. Much of the logic is provided at the Arduino end; the Arduino controls PowerSDR or Thetis by CAT commands. | |
| Status | Support for this hardware written but untested.  This Arduino sketch was intended as an interim build for Andromeda, to be replaced once the radio logic was moved to Thetis. It can be used with PowerSDR. | |

|  |  |  |
| --- | --- | --- |
| Front panel |  | Andromeda 7 encoder + VFO 1st prototype |
| Processor | Arduino Due | |
| Sketch | andromeda\_front\_panel.ino | |
| Sub-folder | front panel\sketch\andromeda\_front\_panel | |
| Customisation steps | This sketch supports both this hardware build and the 6 encoder 2nd prototype hardware build. The sketch needs to be edited slightly to select the target PCB:  Edit the sketch tab “globalinclude.h” to comment out “V3HARDWARE” and include “V2HARDWARE”:  //  // define V2HARDWARE to compile for Kjell's Andromeda early prototype PCB  // design proving initally using Odin software  //  **#define V2HARDWARE 1**  //  // define V3HARDWARE to compile for Kjell's final Andromeda prototype PCB  // design proving initally using Odin software  //  **//#define V3HARDWARE 1** | |
| Works with PC code: | Thetis 2.6.8+  Does not work with PowerSDR | |
| Functionality | This is newly created specifically for Andromeda, removing the radio logic so that it is all in one place. The front panel controls send simple “encoder 3 turned 2 steps clockwise” event messages; these are interpreted to operate the radio by Thetis. | |
| Status | This code works, but will eventually be replaced when the new Andromeda PCB with cheaper Arduino is available. | |

|  |  |  |
| --- | --- | --- |
| Front panel |  | Andromeda 6 encoder + VFO 2nd prototype |
| Processor | Arduino Due | |
| Sketch | andromeda\_front\_panel.ino | |
| Sub-folder | front panel\sketch\andromeda\_front\_panel | |
| Customisation steps | This sketch supports both this hardware build and the 7 encoder 1st prototype hardware build. The sketch needs to be edited slightly to select the target PCB:  Edit the sketch tab “globalinclude.h” to comment out “V2HARDWARE” and include “V3HARDWARE”:  //  // define V2HARDWARE to compile for Kjell's Andromeda early prototype PCB  // design proving initally using Odin software  //  **//#define V2HARDWARE 1**  //  // define V3HARDWARE to compile for Kjell's final Andromeda prototype PCB  // design proving initally using Odin software  //  **#define V3HARDWARE 1** | |
| Works with PC code: | Thetis 2.6.8+  Does not work with PowerSDR | |
| Functionality | This is newly created specifically for Andromeda, removing the radio logic so that it is all in one place. The front panel controls send simple “encoder 3 turned 2 steps clockwise” event messages; these are interpreted to operate the radio by Thetis. | |
| Status | Not tested with this PCB yet. | |

|  |  |  |
| --- | --- | --- |
| Front panel |  | Andromeda 6 encoder + VFO 4th prototype |
| Processor | Arduino Nano Every | |
| Sketch | andromeda\_front\_panel\_nano.ino | |
| Sub-folder | front panel\Arduino Nano Every Sketch\andromeda\_front\_panel\_nano | |
| Customisation steps | None currently required | |
| Works with PC code: | Thetis 2.6.8+  Does not work with PowerSDR | |
| Functionality | This is the newest software, with the radio logic in Thetis and a lower cost Arduino. | |
| Status | Not tested with this PCB yet. | |

# Arduino Software Installation

This guide describes how to download, install and load the Arduino software for the Odin console. The guide assumes that you are using the Arduino Integrated Development Environment (IDE) running on a windows platform. For users with different operating systems, different folder locations will probably apply.

# Install the Arduino IDE

The Arduino IDE is downloaded from the Arduino web page. The download links are on this page:

<https://www.arduino.cc/en/Main/Software>

Download and install the IDE. When you run it for the first time, it will look something like:



This is showing you a new, blank program. Arduino programs are called “sketches”.

# Add Support for the Due and Nano Every Boards

As shipped the Arduino IDE can build code for some of the processor types used in the Arduino range, but not for the Arduino “Due” used in this project. A simple download will add the Due:

1. Open the Arduino IDE
2. Click “Tools|Board|Boards manager” on the menu
3. Scroll down to the entry for “Arduino SAM Boards (32-bit ARM Cortex-M3)” and click “install”
4. Scroll down to the entry for “Arduino Mega AVR boards by Arduino” and click “install”

# Install Libraries into the Arduino IDE

The next step is to install 3 libraries into the Arduino library. This will provide access to the code that we have used as part of the Odin build.

The Arduino system loads libraries into a folder it created on your computer; usually that folder is installed into the “documents” folder called “Arduino\libraries”. On my computer that folder is “C:\Users\loz barker\Documents\Arduino\libraries”. Use windows explorer to find that folder so you know where it is.

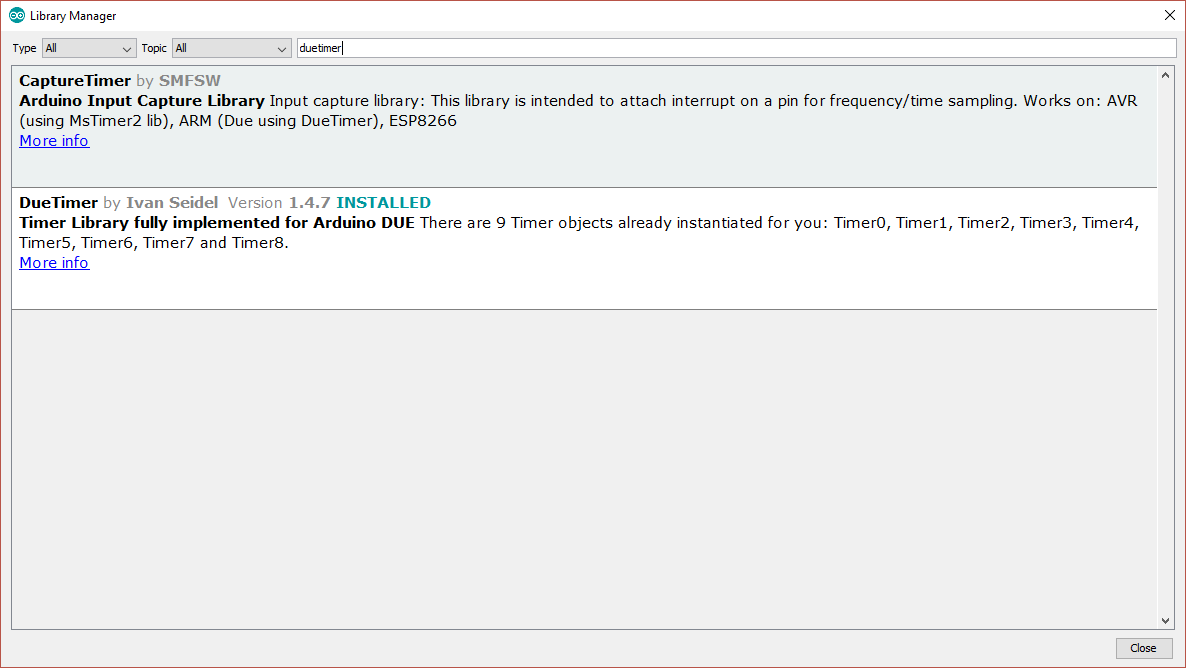
The 3 required libraries are:

|  |  |
| --- | --- |
| DueFlashStorage | Provides persistent storage so settings are kept while power is removed |
| DueTimer | Provides access to the Arduino’s timer, so the code can synchronise to a 1 millisecond time count |
| Encoder | Controls the optical VFO encoder |

The first two are accessed through the Arduino library manager; the last has to be installed manually.

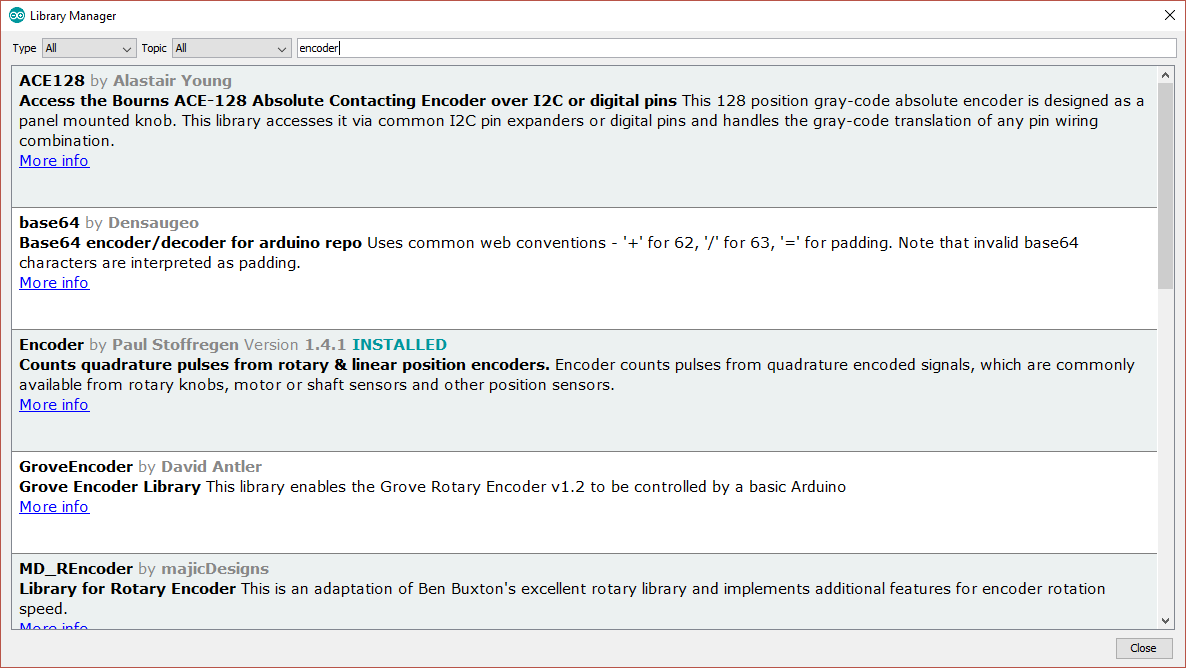
## DueTimer

1. Open the Arduino IDE
2. Click "Sketch | Include Library | Manage libraries..." on the menu
3. In the library manager type "due timer" where it says "filter your search" and hit enter
4. Find "DueTimer" by Ivan Seidel and click "install"
5. You should now have a folder “Documents\Arduino\libraries\DueTimer”
6. Your screen will look something like this:



## Encoder

1. In the library manager type "encoder" where it sayds "filter your search" and hit enter
2. Find "encoder" by Paul Stoffregen and click "install"
3. Your screen will look something like this:



## DueFlashStorage

This needs to be installed manually. The procedure is as follows:

1. Visit the repository on github: <https://github.com/sebnil/DueFlashStorage>
2. Click “clone or download” then “download zip”
3. Store the zip file on your PC for example in the “downloads” folder
4. Open the zip file and extract all files. You will now have a folder “DueFlashStorage-master” which will hold one folder also called “DueFlashStorage-master”
5. Rename the second folder “DueFlashStorage” (remove the “-master” part)
6. Copy that whole folder to your “documents\arduino\libraries” folder

# Download the Andromeda Software Repository

1. Visit the repository on github: https://github.com/laurencebarker/Andromeda\_front\_panel
2. Click “clone or download” then “download zip”
3. Store the zip file on your PC for example in the “downloads” folder
4. Open the zip file and extract to your PC; for example into a folder “SDR” in “documents”
5. There will be a folder called “Andromeda\_front\_panel-master” in your “SDR” folder

There are several folders:

|  |  |
| --- | --- |
| Documentation | The user guide and this installation guide |
| Hardware | The schematics and PCB layouts for the console PCB |
| Nextion Display | Files for 2 things:   * For the Nextion display itself, setting out the layouts of the screens used * Files to patch the Arduino library for the display |
| Sketch | The Arduino program for the console. |

## Build the code

To open the appropriate software sketch (the filenames etc are listed in the tables above)

1. Run the Arduino IDE
2. Use the "File|Open..." menu command
3. Navigate to "andromeda\_front\_panel\_nano.ino" and click "open"
4. you should now see the files listed in tabs above the editor window

You now need to tell the IDE what kind of board it is compiling for, and which serial port to use to connect to it.

1. Connect a USB cable between the Arduino programming port (next to the black power connector) and your PC.
2. It may be necessary to install device drivers at this point – follow any instructions.
3. Click "board" on the "tools" menu and select one of the following depending on sketch:
   1. "Arduino Due (programming Port)” or
   2. “Arduino Nano Every” (for the newest board build)
4. If the sketch is for the Arduino Nano Every, select “register emulation” on the “Tools” menu to say “none (ATMEGA4809)”
5. Click “port” on the “tools” menu and choose the Arduino COM port listed (mine is COM6)
6. Click "Verify/compile" on the "sketch" menu to compile
7. (A message “compiling sketch…” will appear. This will take around a minute and should result in a message saying the % of program space used)

Finally you need to upload the code to your Arduino:

* Click "Upload" on the "sketch" menu to upload to the Arduino
* A simple progress bar will show in the bottom window of the IDE, twice - for each of "programming" and "verify"
* When it has successful finished the last message will be "CPU reset"
* (note that an error message **avrdude: jtagmkII\_initialize(): Cannot locate “flash” and “boot” memories in description** is reported for the Arduino Nano Every but this can be ignored)

Your Arduino should now be executing the Andromeda code!

# To Do List

1. Find out why bottom LED lit in new sketch (still has the debug code – lit while 2ms tick code active)
2. Update scan code to button code table

# Thetis changes still needed

1. SHIFT functionality
2. Different encoder functions during TX (to allow ATU fine tune)
3. User editor for control assignments