# “Andromeda” Console Implementation Notes

This documentation is for the Andromeda front panel PCB using the Arduino Nano Every processor and 2 MCP23017 ICs.

# Control Layout



## 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 number** | **Function** | **Encoder report #** | **Switch**  **s/w#** |
| ENC1 Top | RX1 AF | 1 |  |
| ENC1 bottom | RX1 AGC | 2 |  |
| ENC1 click | RX1 mute |  |  |
| ENC3 Top | RX2 AF | 3 |  |
| ENC3 bottom | RX2 AGC | 4 |  |
| ENC3 click | RX2 mute |  |  |
| ENC5 Top | Filter high | 5 |  |
| ENC5 bottom | Filter low | 6 |  |
| ENC5 click | Filter reset |  |  |
| ENC7 Top | Diversity gain | 7 |  |
| ENC7 bottom | Diversity phase | 8 |  |
| ENC7 click | Fast/slow |  |  |
| ENC9 Top | RIT | 9 |  |
| ENC9 bottom | n/a | 10 |  |
| ENC9 click | RIT/XIT clear |  |  |
| ENC11 Top | Multifunction | 11 |  |
| ENC11 bottom | Drive | 12 |  |
| ENC11 click | Multi click |  |  |

## Pushbutton Functions



Pushbuttons have two numbers: the software scan code from the matrix algorithm; and the number that is reported to CAT commands. A lookup table from scan code gives the CAT number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s/w scan code | CAT report | function | s/w scan code | CAT report | function |
| 0 | 21 | Softkey 1 | 20 | 39 | User 1 |
| 1 | 22 | Softkey 2 | 21 | 36 | A>B |
| 2 | 23 | Softkey 3 | 22 | 33 | Band - |
| 3 | 24 | Softkey 4 | 23 | 30 | Band + |
| 4 | 25 | Softkey 5 | 24 | 45 | CTUNE |
| 5 | 26 | Softkey 6 | 25 | 44 | VFO lock |
| 6 | 27 | Softkey 7 | 26 | 43 | VFO A/B |
| 7 | 28 | Softkey 8 | 27 | 42 | RIT/XIT |
| 8 | 11 | Encoder 11 | 28 | 41 | User 3 |
| 9 | 3 | Encoder 3 | 29 | 38 | Split |
| 10 | 1 | Encoder 1 | 30 | 35 | Filter - |
| 11 | 50 | Two tone test | 31 | 32 | Filter + |
| 12 | 49 | Puresignal on | 32 | 0 | n/a |
| 13 | 48 | TUNE | 33 | 0 | n/a |
| 14 | 47 | MOX | 34 | 0 | n/a |
| 15 | 46 | SDR on | 35 | 0 | n/a |
| 16 | 40 | User 2 | 36 | 29 | Shift |
| 17 | 37 | B>A | 37 | 9 | Encoder 9 |
| 18 | 34 | Mode - | 38 | 7 | Encoder 7 |
| 19 | 31 | Mode + | 39 | 5 | Encoder 5 |

Encoder buttons start at 1; pushbuttons start at 21; 4x3 keypad starts at 30. They are laid out geographically.

(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 | s/w number | function |
| 1 | 0 | MOX |
| 2 | 1 | ATU ready |
| 3 | 2 | TUNE active |
| 4 | 3 | Puresignal on |
| 5 | 4 | Diversity enabled |
| 6 | 5 | SHIFT on |
| 7 | 6 | CTUNE |
| 8 | 7 | RIT |
| 9 | 8 | XIT |
| 10 | 9 | VFO A |
| 11 | 10 | VFO lock |

## 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

# Arduino Software Structure

A simpler cheaper is being used from the first prototypes: an 8 bit Arduino Nano Every.

## 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 Hardware

“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.

## I2C ports

I2C (A4, A5) connect to MCP23017 devices. Chip address = 0x20, 0x21. Class: Wire

## 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

Add 2xMCP23017 and we have 51 I/O available. Pushbuttons scanned switches in a matrix.

### 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** | CAT:5  Encoder 5 | CAT:32  Filter + | CAT:30  Band + | CAT:46  SDR on | CAT:28  Softkey 8 |
| **Row 7**  **2GPB6** | CAT:7  Encoder 7 | CAT:35  Filter - | CAT:33  Band - | CAT:47  MOX | CAT:27  Softkey 7 |
| **Row 6**  **2GPB5** | CAT:9  Encoder 9 | CAT:38  SPLIT | CAT:36  A>B | CAT:48  Tune | CAT:26  Softkey 6 |
| **Row 5**  **2GPB4** | CAT:29  SHIFT | CAT:41  User3 | CAT:39  User1 | CAT:49  PS on | CAT:25  Softkey 5 |
| **Row 4**  **2GPB3** | N/A | CAT:42  RIT/XIT | CAT:31  Mode + | CAT:50  2 Tone | CAT:24  Softkey 4 |
| **Row 3**  **2GPB2** | N/A | CAT:43  A/B | CAT:34  Mode - | CAT:1  Encoder 1 | CAT:23  Softkey 3 |
| **Row 2**  **2GPB1** | N/A | CAT:44  VFO Lock | CAT:37  B>A | CAT:3  Encoder 3 | CAT:22  Softkey 2 |
| **Row 1**  **2GPB0** | N/A | CAT:45  CTUNE | CAT:40  USER2 | CAT:11  Encoder 11 | CAT:21  Softkey 1 |

5x8 switch matrix – gives us 4 spare switches

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 Combination

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 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.

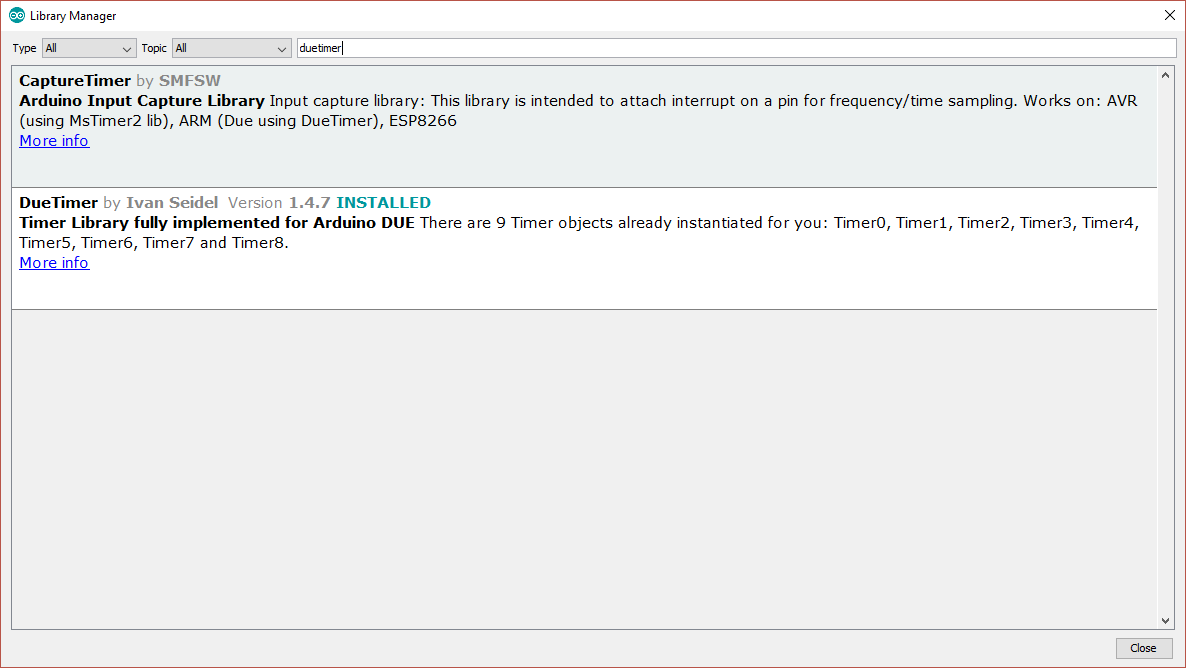
The 3 required libraries are:

|  |  |
| --- | --- |
| 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.

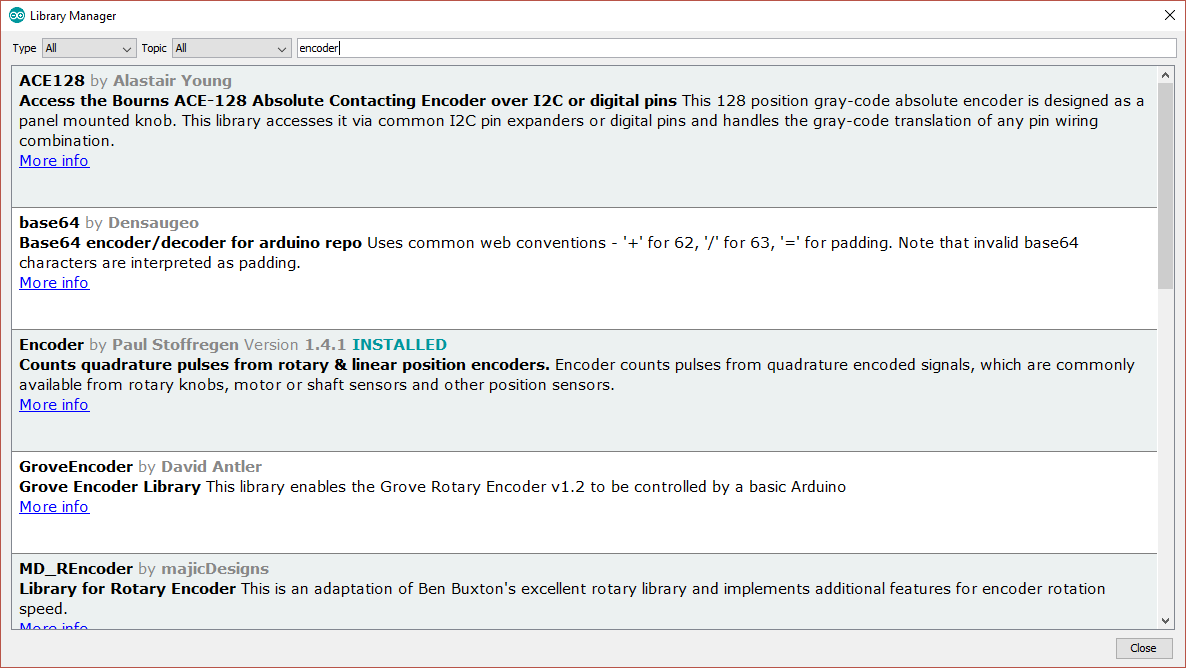
## DueTimer

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