

Engineering Tools

Table of Contents

Overview	2
Text Editors	3
Notepad++	3
Kate	4
Gedit	5
Nano	6
vi, vim, or gvim	7
Eclipse	7
Visual Studio Code	8
DVT	9
Sigassi	9
Spyder IDE	9
Arduino IDE	10
AsciiDoctor	11
Sed and Awk	12
Grep	12
Graphic Tools	12
DrawIO	12
Wavedrom	13
Gimp	14
Inkscape	15
Autocad Fusion 360	16
Cura	16
Source Control	16
Git	16
Subversion	17
Compare tools	17
Kdiff3	17
Meld	18
Math Tools	19
Matlab	19
SciLab	20
Wolfram Alpha	20
Programming Languages	21
Python	21
C or C++	23

Ruby	25
Tcl	25
Linux Shell	25
Communication Tools	26
SSH	26
Virtual Network Computing, VNC	26
PuTTY	26
Minicom	27
Remote Desktop Protocol	27
Circuit Design	27
Kicad	27
FPGA Tools	28
Ecellium Logic Simulator	28
Modelsim	29
Aldec	29
Synplify Pro	29
Vivado	29
Libero	29
Audio Tools	29
Audacity	29
Virtualization	30
VirtualBox	30
Docker	30
Windows Subsystem for Linux, WSL	30
Operating Systems	31
Microsoft Windows	31
Ubuntu Linux	31
Security	31
Bitwarden	31
Protocols	31
YAML	31
MQTT	32
Domain Name Server	32
Consistent Overhead Byte Stuffing	32

Overview

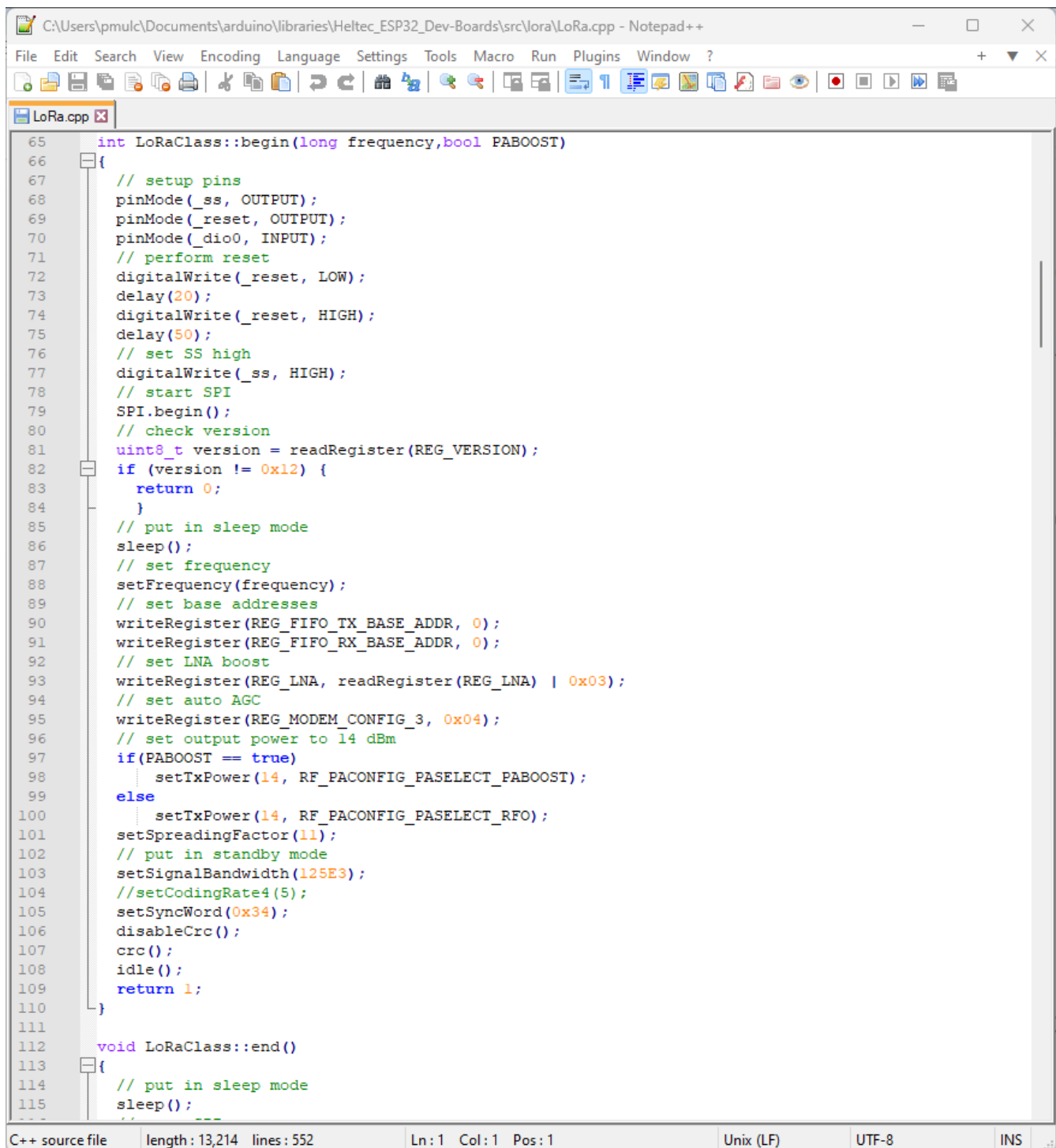
This is a list of tools I was able to list in short order. Please note that this is not complete as new tools are created all the time and I may have missed helpful tools in my haste. Please be patient as I have written this during my off time.

Text Editors

Notepad++

This is a windows based text editor with some nice features like:

- Code Highlighting (color coded text based on type of file)
- Block mode selection
- Enhanced search and replace
- Macro record and playback



```
65 int LoRaClass::begin(long frequency, bool PABOOST)
66 {
67     // setup pins
68     pinMode(_ss, OUTPUT);
69     pinMode(_reset, OUTPUT);
70     pinMode(_dio0, INPUT);
71     // perform reset
72     digitalWrite(_reset, LOW);
73     delay(20);
74     digitalWrite(_reset, HIGH);
75     delay(50);
76     // set SS high
77     digitalWrite(_ss, HIGH);
78     // start SPI
79     SPI.begin();
80     // check version
81     uint8_t version = readRegister(REG_VERSION);
82     if (version != 0x12) {
83         return 0;
84     }
85     // put in sleep mode
86     sleep();
87     // set frequency
88     setFrequency(frequency);
89     // set base addresses
90     writeRegister(REG_FIFO_TX_BASE_ADDR, 0);
91     writeRegister(REG_FIFO_RX_BASE_ADDR, 0);
92     // set LNA boost
93     writeRegister(REG_LNA, readRegister(REG_LNA) | 0x03);
94     // set auto AGC
95     writeRegister(REG_MODEM_CONFIG_3, 0x04);
96     // set output power to 14 dBm
97     if(PABOOST == true)
98         setTxPower(14, RF_PACONFIG_PASELECT_PABOOST);
99     else
100         setTxPower(14, RF_PACONFIG_PASELECT_RFO);
101     setSpreadingFactor(11);
102     // put in standby mode
103     setSignalBandwidth(125E3);
104     //setCodingRate4(5);
105     setSyncWord(0x34);
106     disableCrc();
107     crc();
108     idle();
109     return 1;
110 }
111
112 void LoRaClass::end()
113 {
114     // put in sleep mode
115     sleep();
116 }
```

Figure 1. Notepad++ Example`

<https://notepad-plus-plus.org/downloads/>

Kate

Linux graphical text editor with enhanced features like [Notepad++](#).

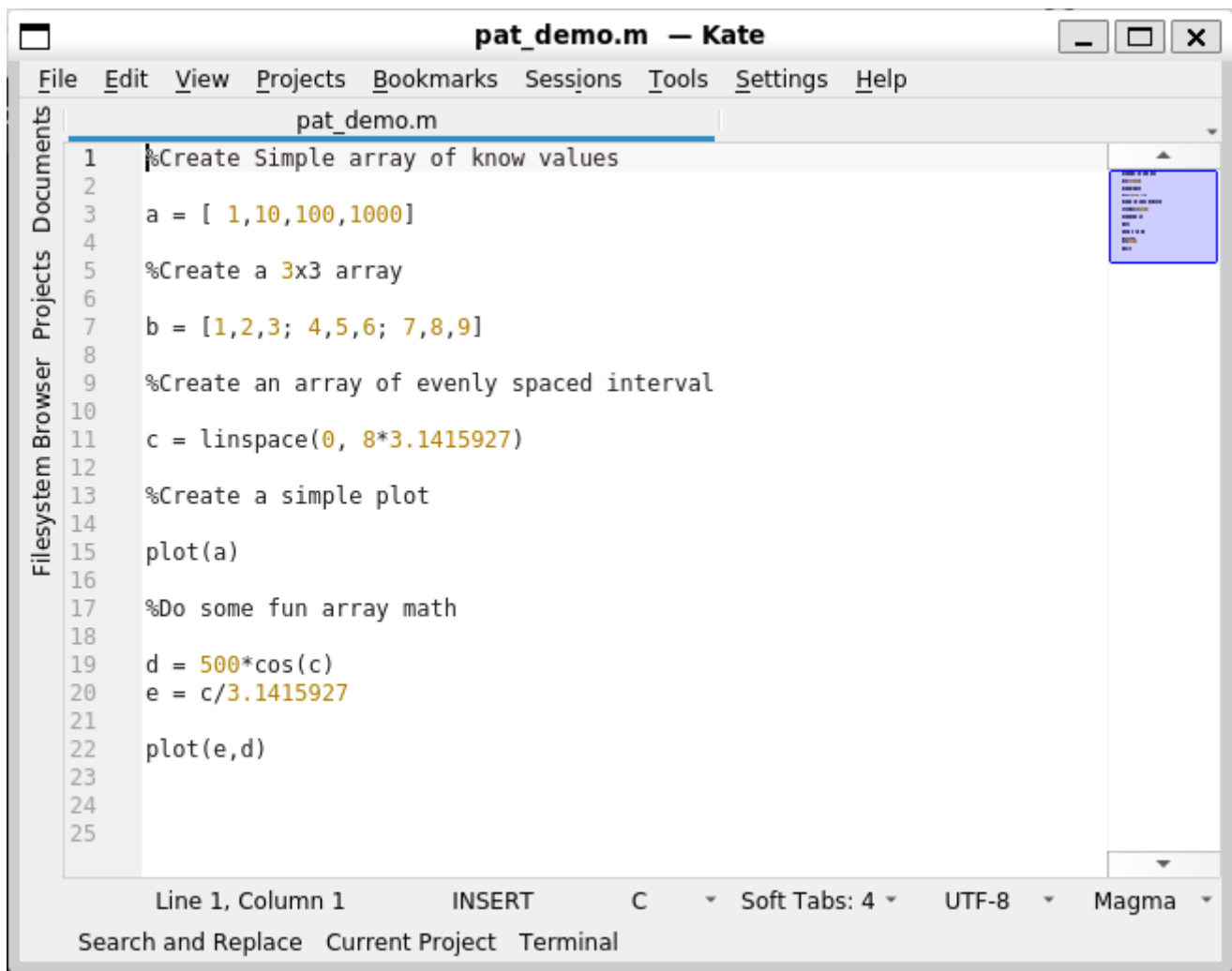
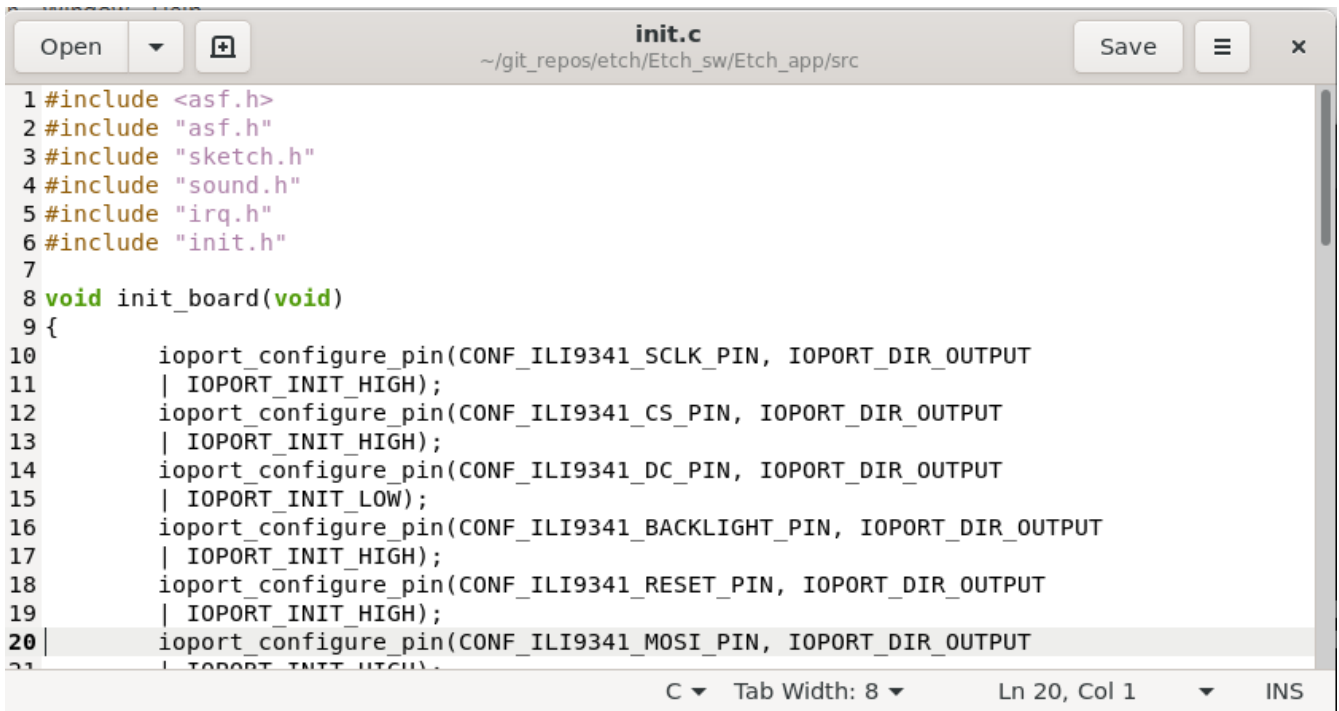


Figure 2. Kate Example`

<https://kate-editor.org/>

Gedit

Linux graphical text editor. This supports basic features.



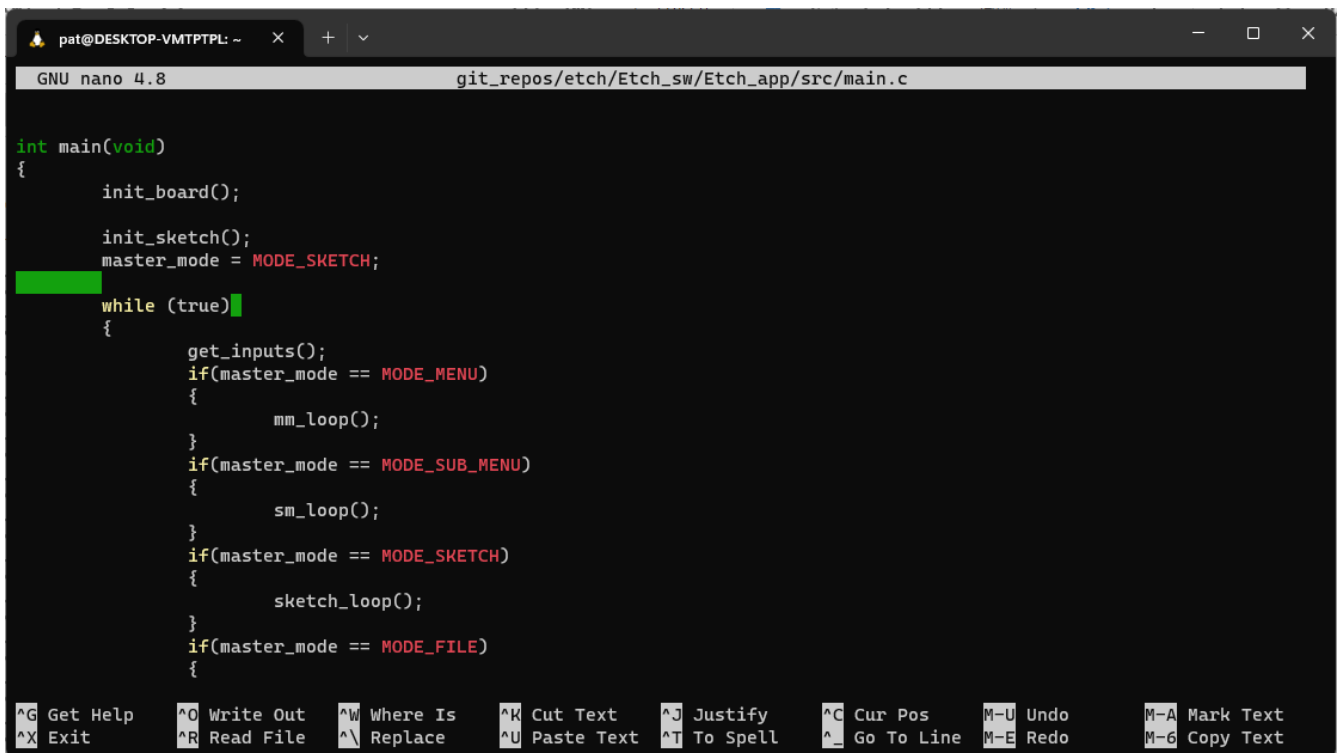
```
1 #include <asf.h>
2 #include "asf.h"
3 #include "sketch.h"
4 #include "sound.h"
5 #include "irq.h"
6 #include "init.h"
7
8 void init_board(void)
9 {
10     ioport_configure_pin(CONF_ILI9341_SCLK_PIN, IOPORT_DIR_OUTPUT
11     | IOPORT_INIT_HIGH);
12     ioport_configure_pin(CONF_ILI9341_CS_PIN, IOPORT_DIR_OUTPUT
13     | IOPORT_INIT_HIGH);
14     ioport_configure_pin(CONF_ILI9341_DC_PIN, IOPORT_DIR_OUTPUT
15     | IOPORT_INIT_LOW);
16     ioport_configure_pin(CONF_ILI9341_BACKLIGHT_PIN, IOPORT_DIR_OUTPUT
17     | IOPORT_INIT_HIGH);
18     ioport_configure_pin(CONF_ILI9341_RESET_PIN, IOPORT_DIR_OUTPUT
19     | IOPORT_INIT_HIGH);
20     ioport_configure_pin(CONF_ILI9341_MOSI_PIN, IOPORT_DIR_OUTPUT
21     | IOPORT_INIT_HIGH);
22 }
```

Figure 3. Gedit Example`

<https://wiki.gnome.org/Apps/Gedit>

Nano

Linux terminal text editor. This editor is an easy to use basic text editor. This can be used when a graphical environment is not available, and is simple and intuitive to use. You navigate your text file with the arrow keys, when you are ready to save type [CTRL-O], to exit type [CTRL-X].



```
GNU nano 4.8 git_repos/etch/Etch_sw/Etch_app/src/main.c

int main(void)
{
    init_board();

    init_sketch();
    master_mode = MODE_SKETCH;

    while (true)
    {
        get_inputs();
        if(master_mode == MODE_MENU)
        {
            mm_loop();
        }
        if(master_mode == MODE_SUB_MENU)
        {
            sm_loop();
        }
        if(master_mode == MODE_SKETCH)
        {
            sketch_loop();
        }
        if(master_mode == MODE_FILE)
        {
        }
    }
}
```

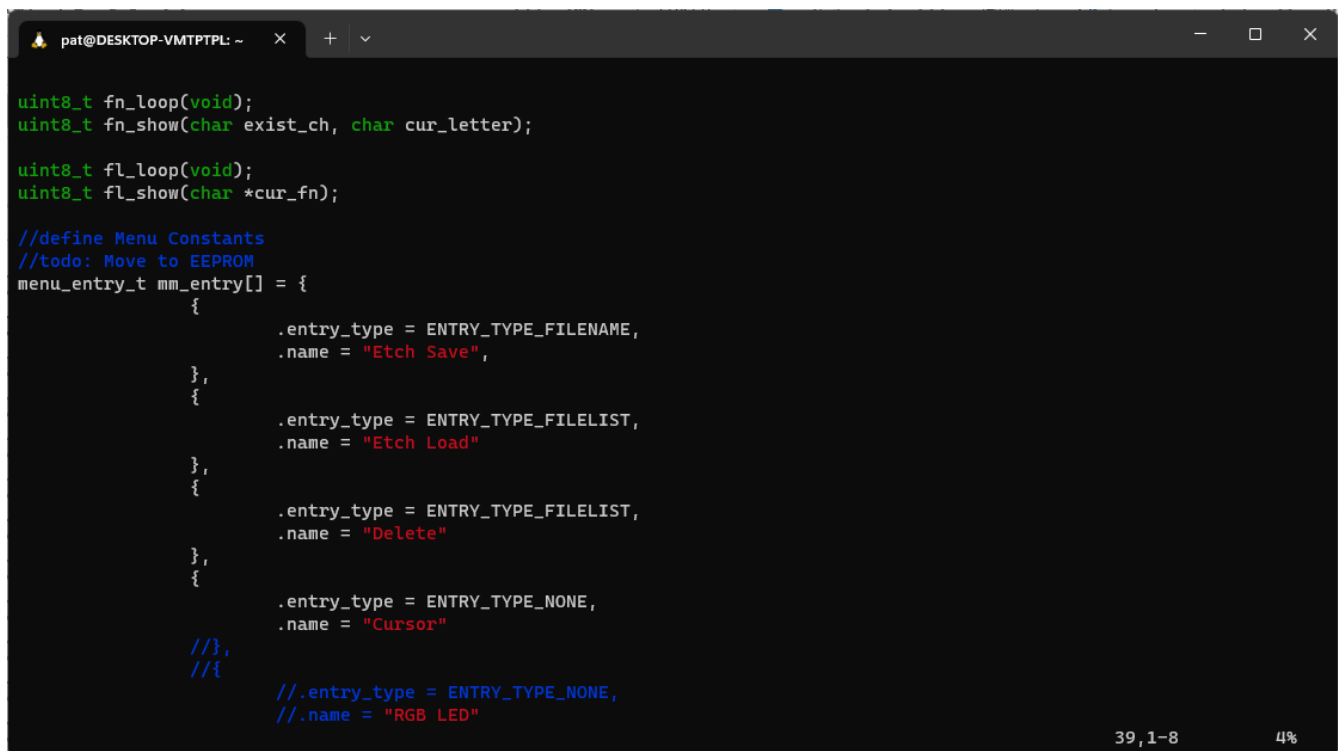
Figure 4. Nano Example`

<https://www.nano-editor.org/>

vi, vim, or gvim

Linux terminal and graphical text editor. This is a very full featured text editor that has a complex interface. There are two modes to the environment: command and insert. Insert mode is entered on a few commands, to exit insert mode and return to command mode use the [ESC] key. If you find yourself in a vi type environment the following commands will help:

- :w - Write to disk
- :wq - Write and exit
- :q! - Exit NOW without writing
- :i Insert text (change to insert mode)
- :set nu - Turn on line numbers

A screenshot of a terminal window with a dark background. The window title bar shows 'pat@DESKTOP-VMTPTPL: ~' and standard window controls. The terminal displays C code with syntax highlighting: green for keywords, blue for comments, and red for string literals. The code includes function declarations, a menu structure definition, and comments. The bottom right of the terminal shows '39, 1-8' and '4%'.

```
uint8_t fn_loop(void);
uint8_t fn_show(char exist_ch, char cur_letter);

uint8_t fl_loop(void);
uint8_t fl_show(char *cur_fn);

//define Menu Constants
//todo: Move to EEPROM
menu_entry_t mm_entry[] = {
    {
        .entry_type = ENTRY_TYPE_FILENAME,
        .name = "Etch Save",
    },
    {
        .entry_type = ENTRY_TYPE_FILELIST,
        .name = "Etch Load"
    },
    {
        .entry_type = ENTRY_TYPE_FILELIST,
        .name = "Delete"
    },
    {
        .entry_type = ENTRY_TYPE_NONE,
        .name = "Cursor"
    },
    //},
    //{
        .entry_type = ENTRY_TYPE_NONE,
        .name = "RGB LED"
    }
};
```

Figure 5. Vim Example`

For more look command information here: <https://web.mit.edu/merolish/Public/vi-ref.pdf>

<https://www.vim.org/>

Eclipse

Eclipse is an open source development environment that is used by many software disciplines. This integrated environment allows full software development including lint (syntax checking) and compiling. There are many language specific perspectives that streamline the environment for specific tasks. Eclipse uses a workbench that contains projects. Each project contains all the files used to build your project. This allows you to organize your source files in an easy to use format; it also allows Eclipse to search only your project. Each project defines how the code will be processed. This allows you to create an environment that knows how to build your project and can present a

debug view of your code.

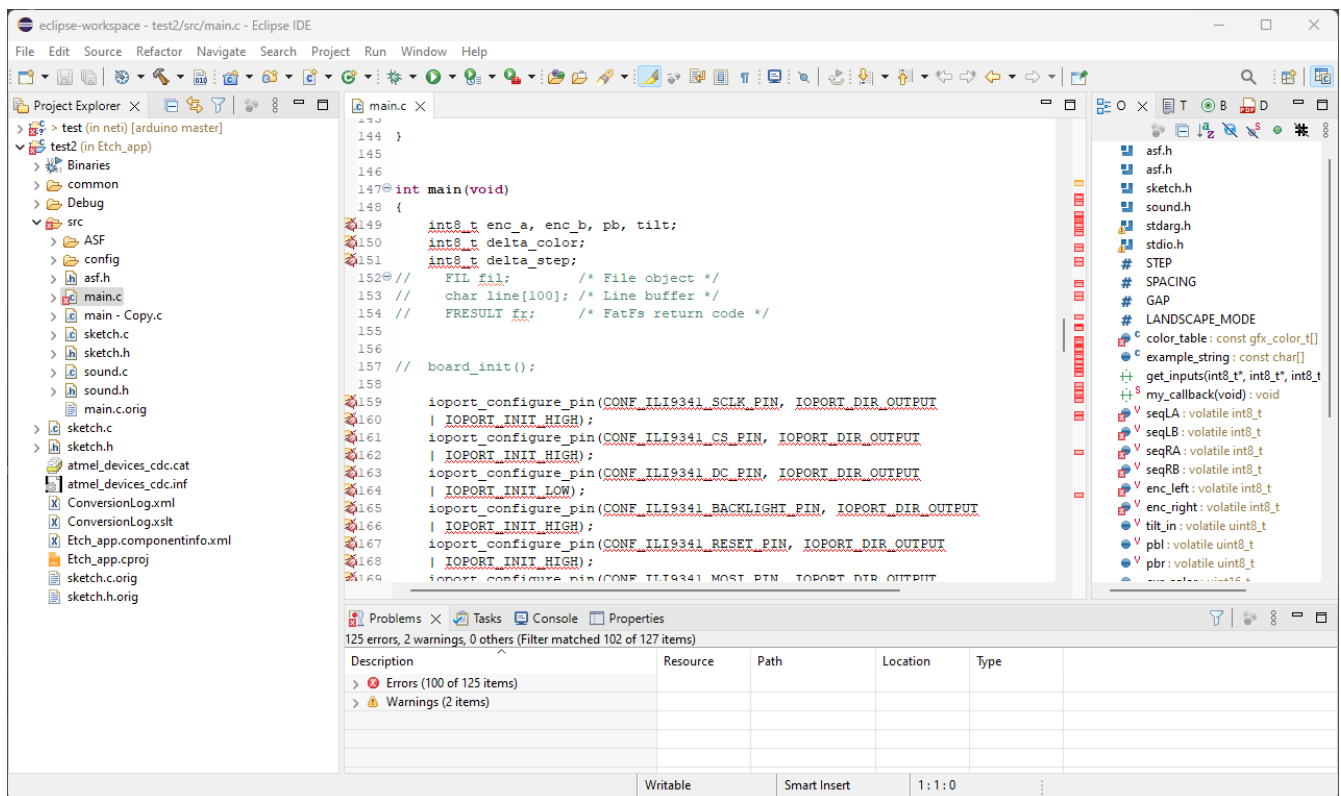


Figure 6. Eclipse Example`

<https://www.eclipse.org/ide/>

<https://www.eclipse.org/home/>

Visual Studio Code

Visual Studio is a free IDE from Microsoft for software development similar to [Eclipse](#). This editor is well supported with many extensions. For example I am using the Asciidoctor extension to write this document. PlatformIO gives you the ability to develop microcontrollers in this nice to use environment.

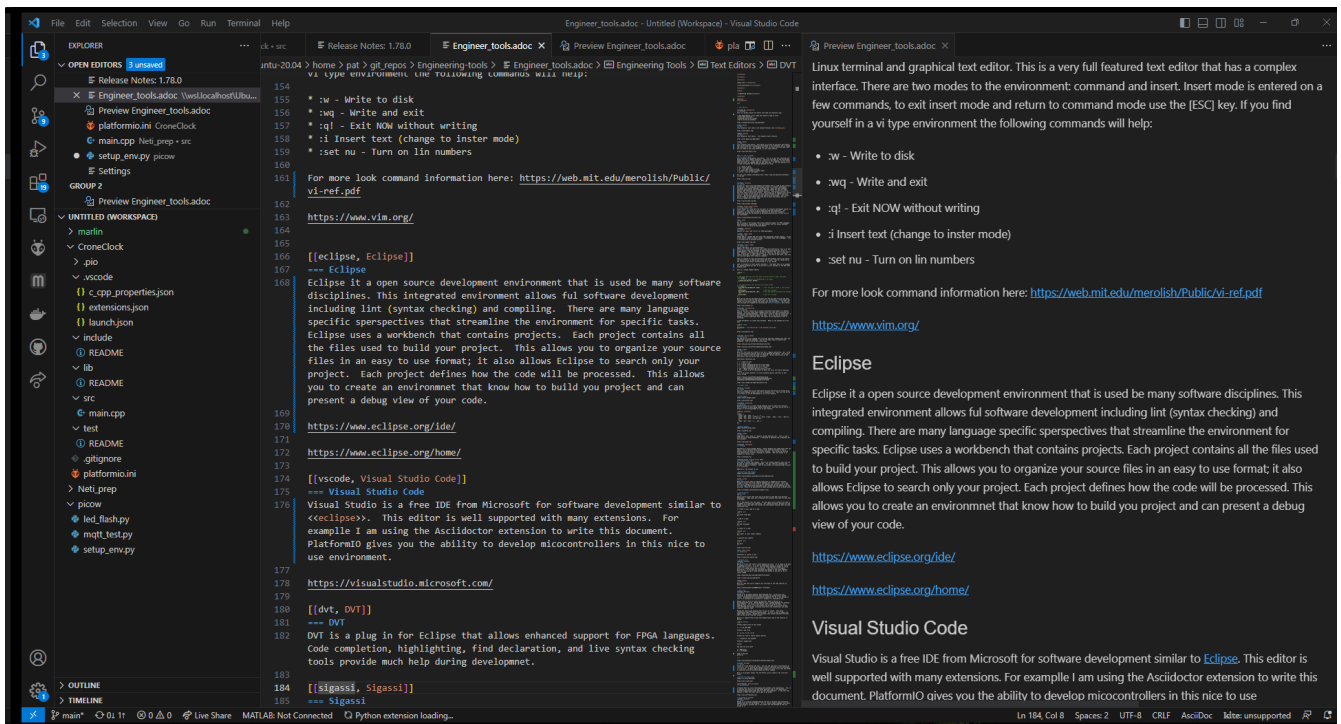


Figure 7. Visual Studio Example`

<https://visualstudio.microsoft.com/>

DVT

DVT is a plug in for Eclipse that allows enhanced support for FPGA languages. Code completion, highlighting, find declaration, and live syntax checking tools provide much help during developmnet.

Sigassi

Sigassi is a tool like [DVT](#) for FPGA development.

Spyder IDE

Spyder IDE is a python IDE that helps when developing a Python program. It has a code editor, python terminal window, a variable viewer and a debugger. This is included with an Anaconda install.

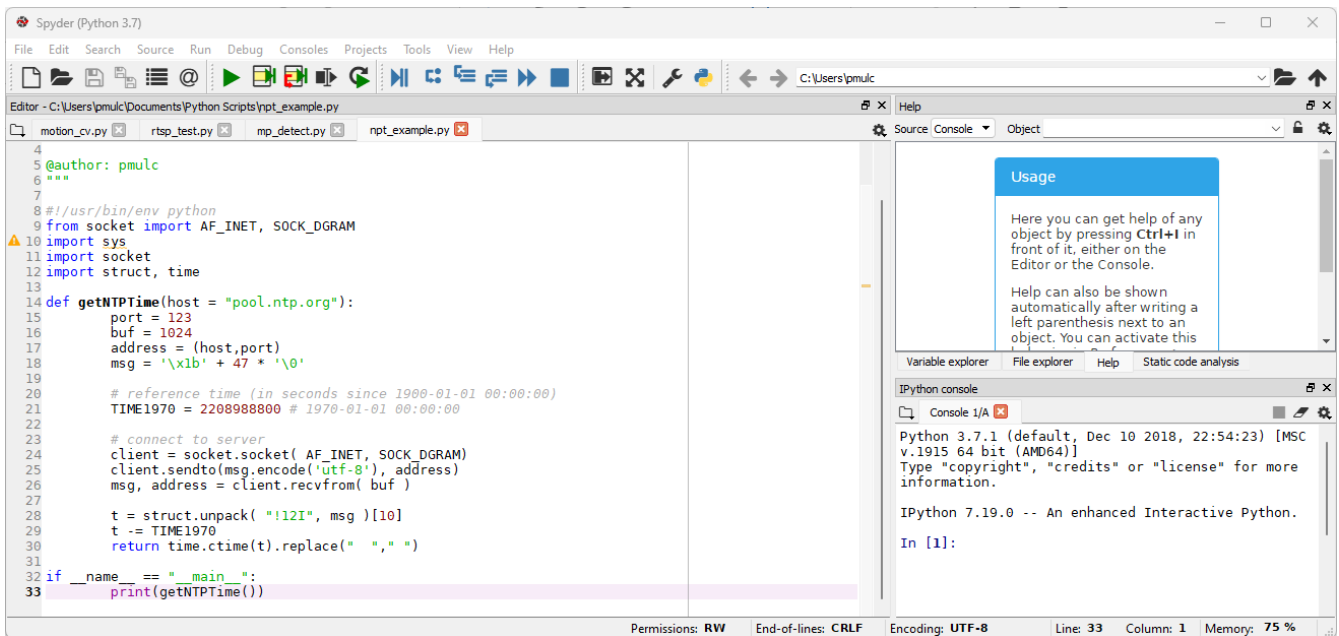


Figure 8. Spyder Example

<https://www.spyder-ide.org/>

Arduino IDE

Simple code editor for microcontrollers. Arduino is a free IDE that works with simple microcontrollers. This is an open source project that was designed to make using microcontrollers easy for gradeschoolers. Because of this we have a powerfull tool that is very easy to use. Arduino uses a simplified version of c that has two basic functions: setup() and loop() in every sketch. Setup is run once at the start. Loop runs over and over again after setup is complete.

There is support for most microntrollers and the examples library has almost any type of sensor or device your would want to use. This is a great way to see something work before incorporating it in your design.

Code is available for most sensors actuators. This means there is an example for almost anything you want to do and this is alll availble in the IDE under Examples menu.

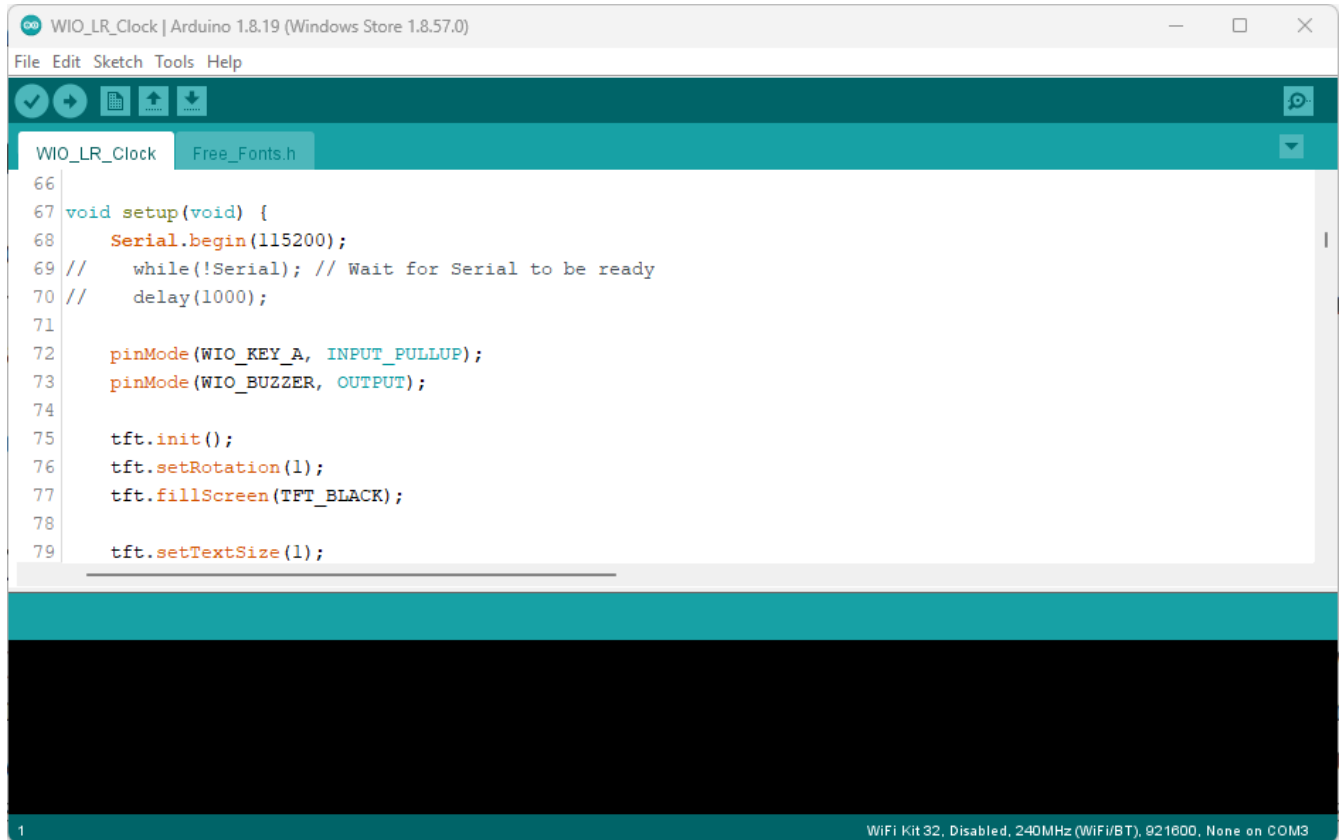
Here is a simple example sketch:

```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
    delay(1000);                        // wait for a second
    digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
    delay(1000);                        // wait for a second
```

```
}
```

Here you can see how the setup function sets the pin mode once and the loop function turns on and off the output with a 1 second delay. This example was copied from the blink example that comes with the environment. You can find this and many more examples by selecting the **File** → **Examples** menu choice.



The screenshot shows the Arduino IDE interface. The title bar reads 'WIO_LR_Clock | Arduino 1.8.19 (Windows Store 1.8.57.0)'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for saving, running, and other functions. The main editor window shows the 'Free_Fonts.h' file with the following code:

```
66
67 void setup(void) {
68     Serial.begin(115200);
69     // while(!Serial); // Wait for Serial to be ready
70     // delay(1000);
71
72     pinMode(WIO_KEY_A, INPUT_PULLUP);
73     pinMode(WIO_BUZZER, OUTPUT);
74
75     tft.init();
76     tft.setRotation(1);
77     tft.fillScreen(TFT_BLACK);
78
79     tft.setTextSize(1);
```

The status bar at the bottom indicates 'WiFi Kit 32, Disabled, 240MHz (WiFi/BT), 921600, None on COM3'.

Figure 9. Arduino Example

<https://www.arduino.cc/>

AsciiDoctor

This is the tool I would use to create a doctorate level paper or a dynamic website. This tool takes human readable text with some easy formatting and created nice looking documents. Since this is easy to read and create text, I can automate the creation without much pain. So if you have an automated method to collect experiment data, the report can be populated as data is collected.

I used AsciiDoctor to create this document. Below is the command run to do that:

```
asciidoctor -r asciidoctor-pdf -b pdf Engineer_tools.adoc
```

<https://asciidoctor.org/>

Sed and Awk

Sed and Awk are linux command line utilities that have spawned into their own simplified scripting languages. You can use these tools to modify text documents. This is use in very clever ways.

<https://www.gnu.org/software/sed/manual/sed.html>

<https://www.gnu.org/software/gawk/manual/gawk.html>

Grep

grep is a tool that will search a file for a "Regular Expression", RE. An RE is a search pattern protocol that can have advanced attributes. You can use grep as a find in files by using a * wildcard. RE are used in many of the text editors and the sed and awk utilities.

Some Useful characters are:

- '^' = Start of line
- '\$' = End of line
- '*' = Match preceding RE term 0 or more times
- '+' = Match preceding RE term 1 or more times
- '?' = Match preceding RE term 0 or 1 times
- '{m}' = Match preceding RE term exactly m times
- '[' = Create a set of characters to match ie. [a-z] will match lowercase letters
- '\' = An escape character to allow preceding special characters in your search string

<https://www.gnu.org/software/grep/manual/grep.html#>

[:~:text=grep%20searches%20the%20named%20input,grep%20searches%20the%20working%20directory%20.](https://www.gnu.org/software/grep/manual/grep.html#text=grep%20searches%20the%20named%20input,grep%20searches%20the%20working%20directory%20)

<https://www.rexegg.com/regex-quickstart.html>

Graphic Tools

DrawIO

This is a simple tool to use that gives the user the ability to create Visio like block diagrams without costs of an expensive tool. This tool can be run in a browser or can be downloaded to run on your machine.

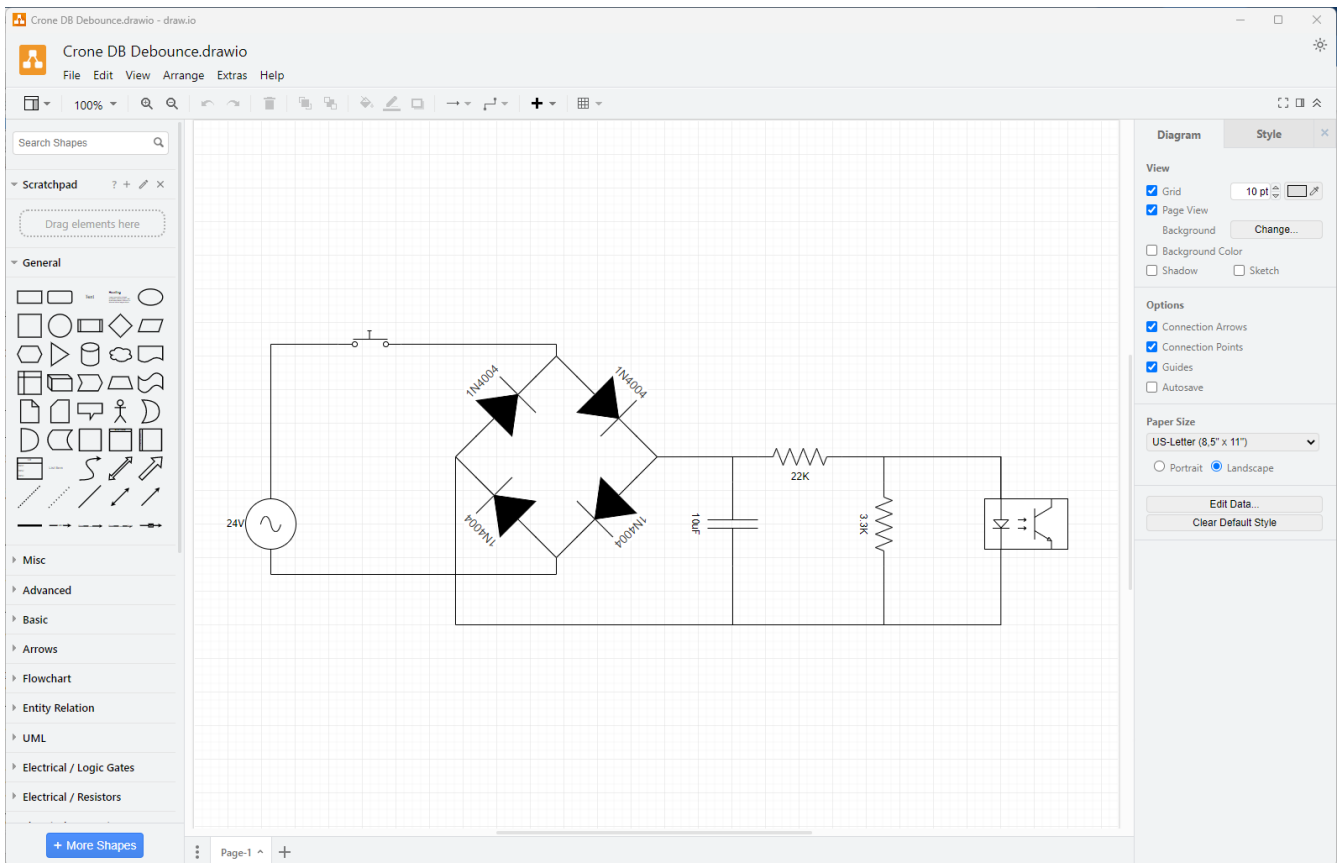


Figure 10. DrawIO Example`

<https://www.drawio.com/>

Wavedrom

Wavedrom allow you to create timing diagrams using a simple text protocol. This is helpful in describing digital interfaces. This tool can be run in a browser or can be downloaded to run on your machine.

```
{signal: [
  {name: 'clk', wave: 'p.....|...'},
  {name: 'dat', wave: 'x.345x|=.x', data: ['head', 'body', 'tail', 'data']},
  {name: 'req', wave: '0.1..0|1.0'},
  {}},
  {name: 'ack', wave: '1.....|01.'}
]}
```

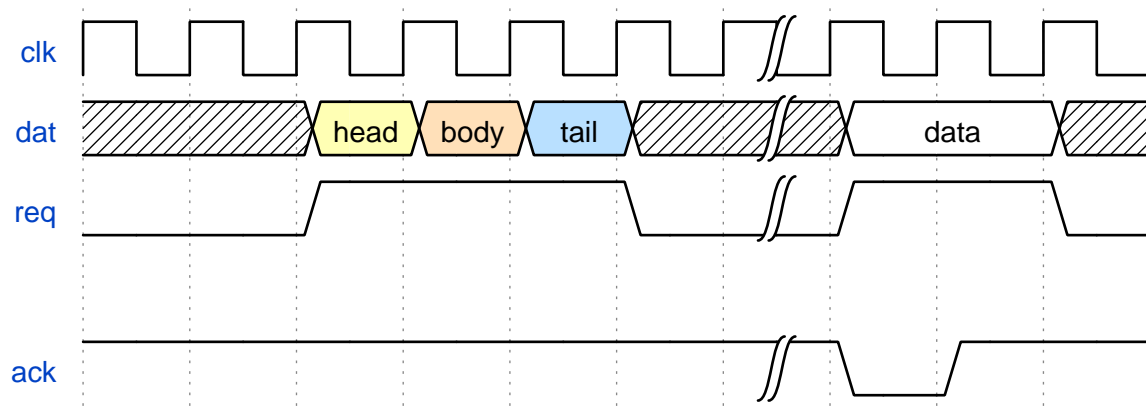


Figure 11. Wavedorm Example

<https://wavedrom.com/>

Gimp

Image editor tool. Gimp is a powerful bitmap editing tool. This is like a free version of Adobe Photoshop. I used gimp to generate the screenshots for this document.

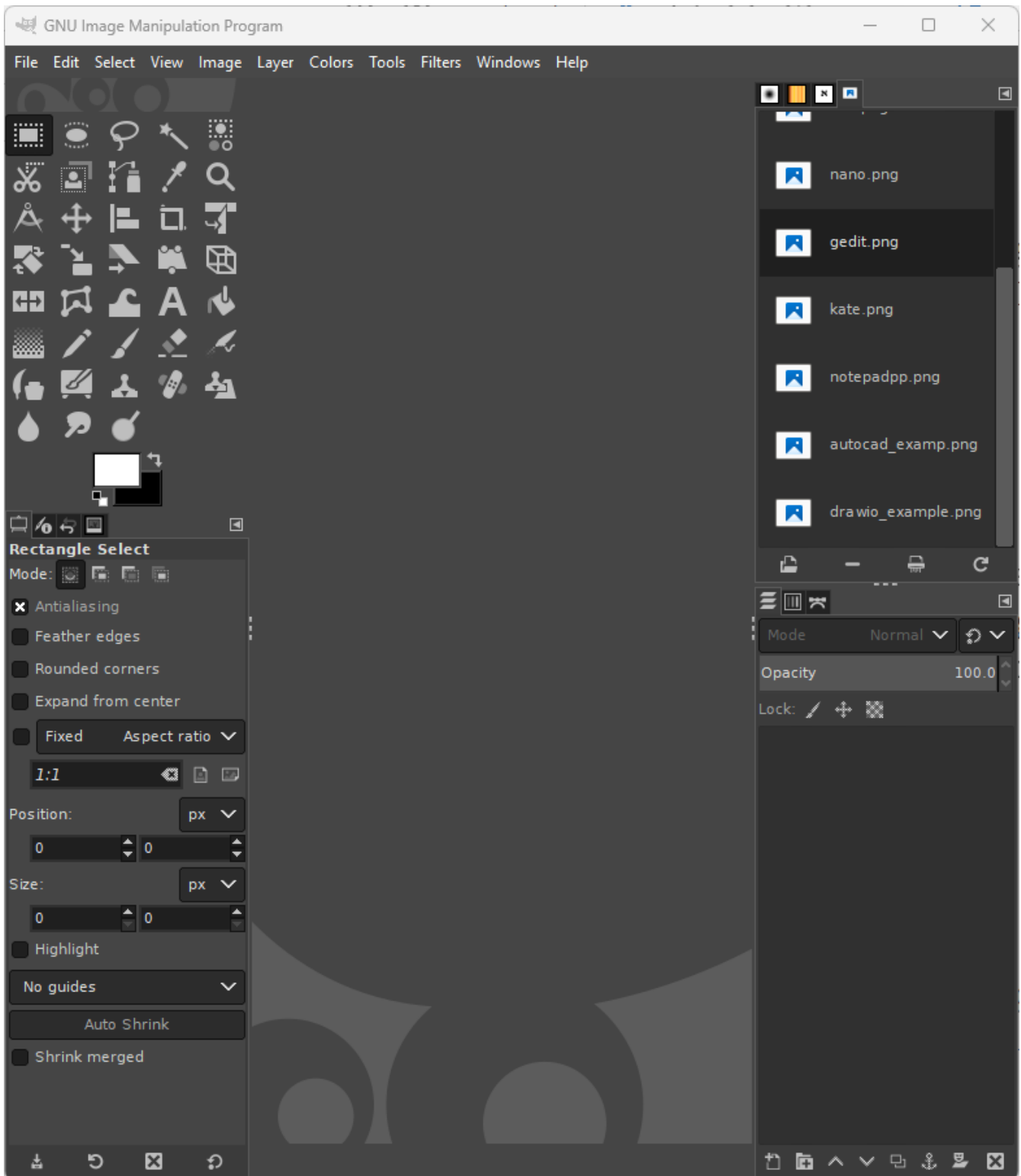


Figure 12. Gimp Example

<https://www.gimp.org/>

Inkscape

Inkscape is a vector graphic editor. Vector graphics are image files that can scale. This is because vector graphic files store directions to make a drawing verses the vause of every pixel (bitmap). This format was derived from instructions to print a drawing on a plotter. PDF file format is an example of a vector file.

<https://inkscape.org/>

Autocad Fusion 360

Free tool to create 3-D drawings. This is an extremely powerful tool that can be used to create 3-dimensional drawings. You can create 2-D sketches and extrude to make a 3-d object. This is very useful in creating things to print on a 3-D printer.

Requires a free account to use.

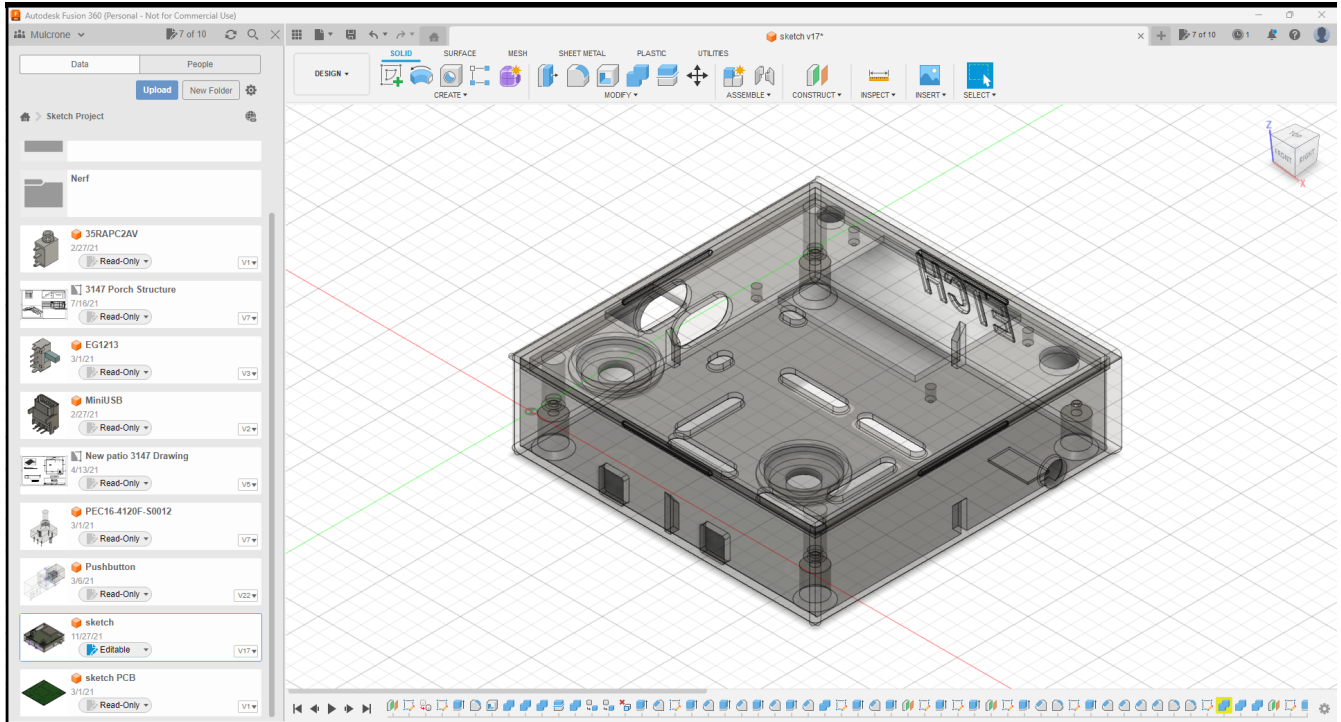


Figure 13. Autocad Fusion 360 Example

<https://www.autodesk.com/products/fusion-360/overview>

Cura

Open source Slicer tool to create G-code that is needed for 3-D printer. G-code is a text file that contains commands for a 3-D printer. This tool converts a drawing ie *.stl file to the commands your printer needs to make it. This tool should have knowledge of your printer and the material you are printing. There is an opportunity to tweak settings here to get a better print.

<https://ultimaker.com/software/ultimaker-cura/>

Source Control

Git

Source control tools that allow you to go back in time and allow parallel development. These tools are a lifesaver for projects that last for more than a few weeks.

Git is the popular tool this week. For most open source software the code is available in a public git repository. For simple one person projects you can create a local repo that will provide the time machine function that can save significant work.

To create a local copy of a repo:

```
git clone [repo URL]
```

To add to a repo:

```
git add [filename]
```

To commit to a repo:

```
git commit -m [your commit comment]
```

To publish your commits:

```
git push
```

<https://git-scm.com/>

Subversion

Subversion is similar to git.

<https://subversion.apache.org/>

Compare tools

Kdiff3

Kdiff3 is a tool that works in both Windows and Linux. It is handy to be able to compare two files. It is key to understand version in a version control system like [Git](#) or [Subversion](#). This can help highlight changes to a file after it is copied before any alterations. This tool also can compare directories. If you want you can merge the changes in the files together. This tool test you go to each difference and decide if you want A, B or a custom change.

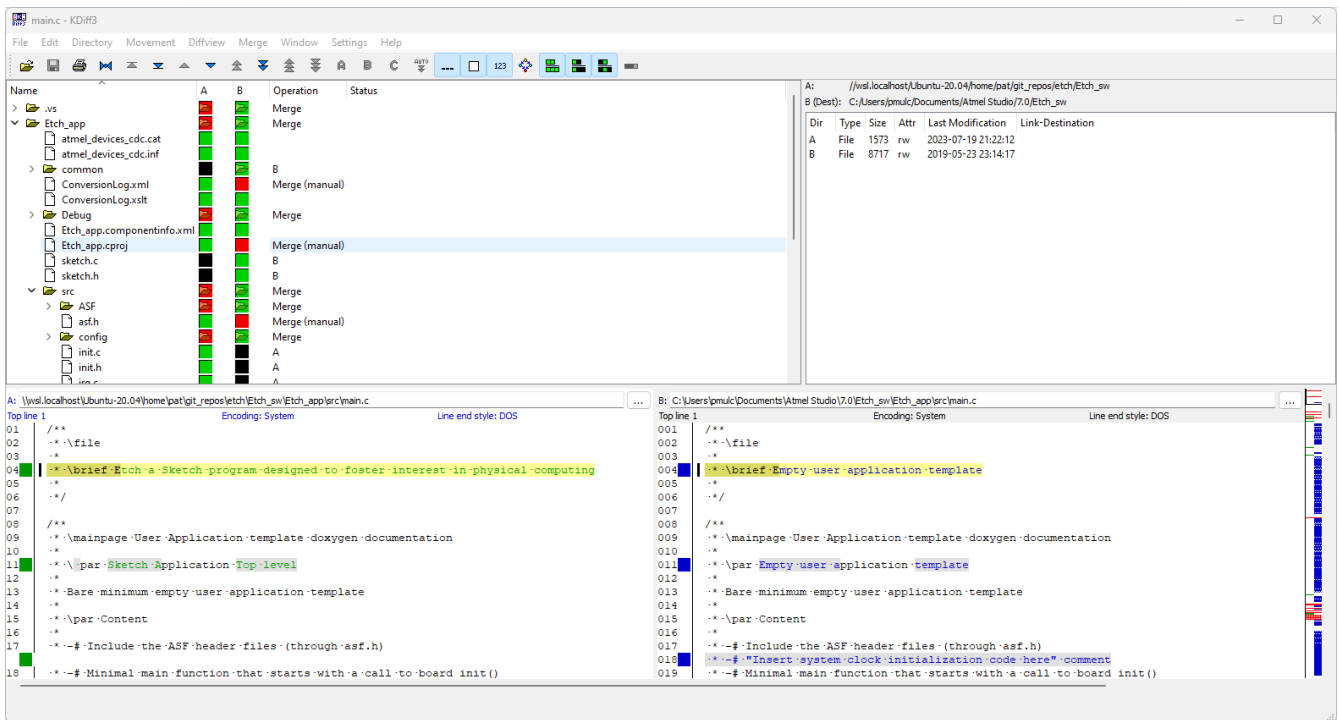


Figure 14. Kdiff3 Example

<https://download.kde.org/stable/kdiff3/?C=M;O=D>

<https://invent.kde.org/sdk/kdiff3>

Meld

Meld is a gnu open source compare tool with many of the same features as [Kdiff3](#).

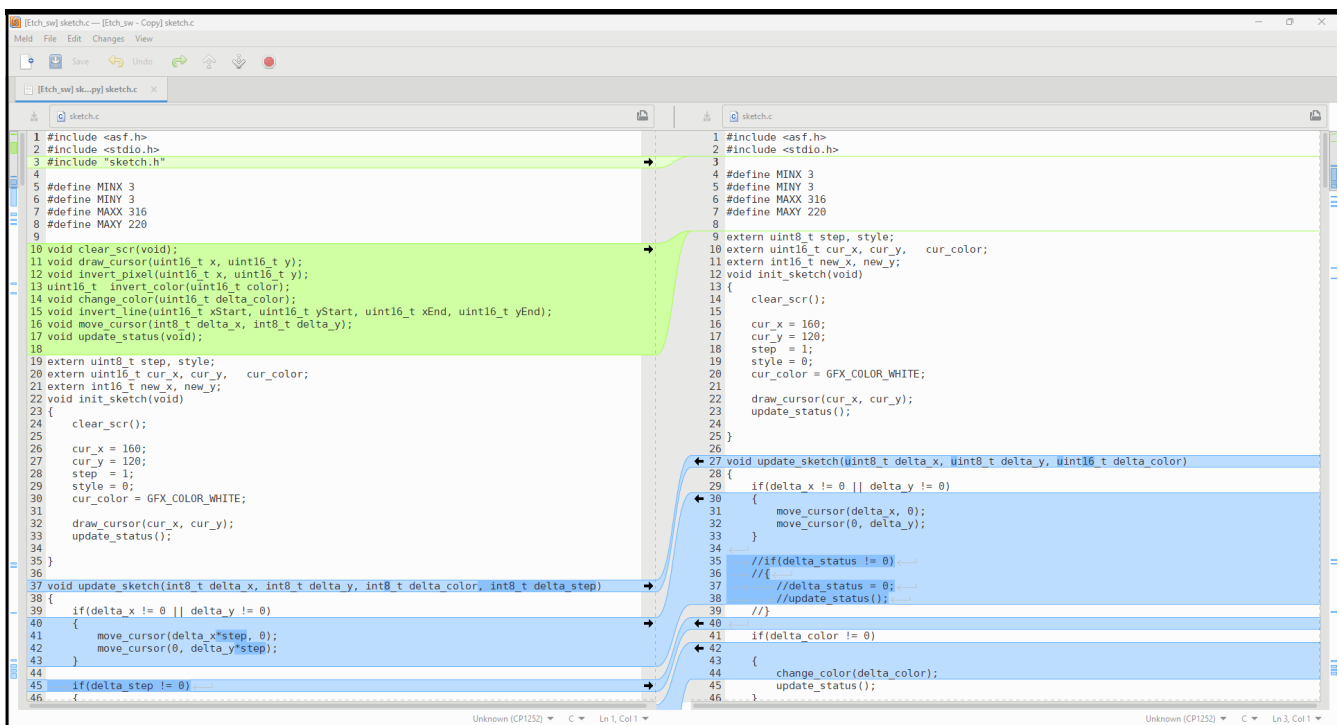


Figure 15. Meld example

<https://gitlab.gnome.org/GNOME/meld/-/tree/main/>

Math Tools

Matlab

Matlab is an extremely powerful math modeling tool. It has very nice libraries of advanced functions that can easily be called from a matlab script. Although Matlab is prohibitavly expensive for me to use outside of work it is available for purchase as a student at a reasonable rate.

Matlab likes to work on arrays and matrices; where a matrice is a multidimensional array. Matlab defaults to a C double datatype for floating point numbers. The scripts that Matlab likes are called M-Files. All commands that can be run in the terminal can be used in an M-File scripts. This combined with normal scripting constructs like conditionals and loops creates a powerful tool.

Matlab has really good graphing that is easy to create. This helps tremendously when trying to visualize data. It also is easy to import and export data to files. Matlab has Toolboxes that contain advanced functions that can be called from your scripts.

Below is a sample M-File script that demonstrates some of the features in Matlab.

```
%Create Simple array of know values  
  
a = [ 1,10,100,1000]  
  
%Create a 3x3 array  
  
b = [1,2,3; 4,5,6; 7,8,9]  
  
%Create an array of evenly spaced interval  
  
c = linspace(0, 8*3.1415927)  
  
%Create a simple plot  
  
plot(a)  
  
%Do some fun array math  
  
d = 500*cos(c)  
e = c/3.1415927  
  
%add to the plot  
plot(e,d)
```

<https://www.mathworks.com/products/matlab/student.html>

SciLab

Scilab is a free Matlab like tool. This has the same basic functionality as [Matlab](#), but is missing the advance function libraries that are available in [Matlab](#). So most simple matlab scripts will just run in this tool after a conversion with the `mfile2sci(path_to_M-File)` command.

Here is the graphix output from the M-File script listed in the [Matlab](#) section.

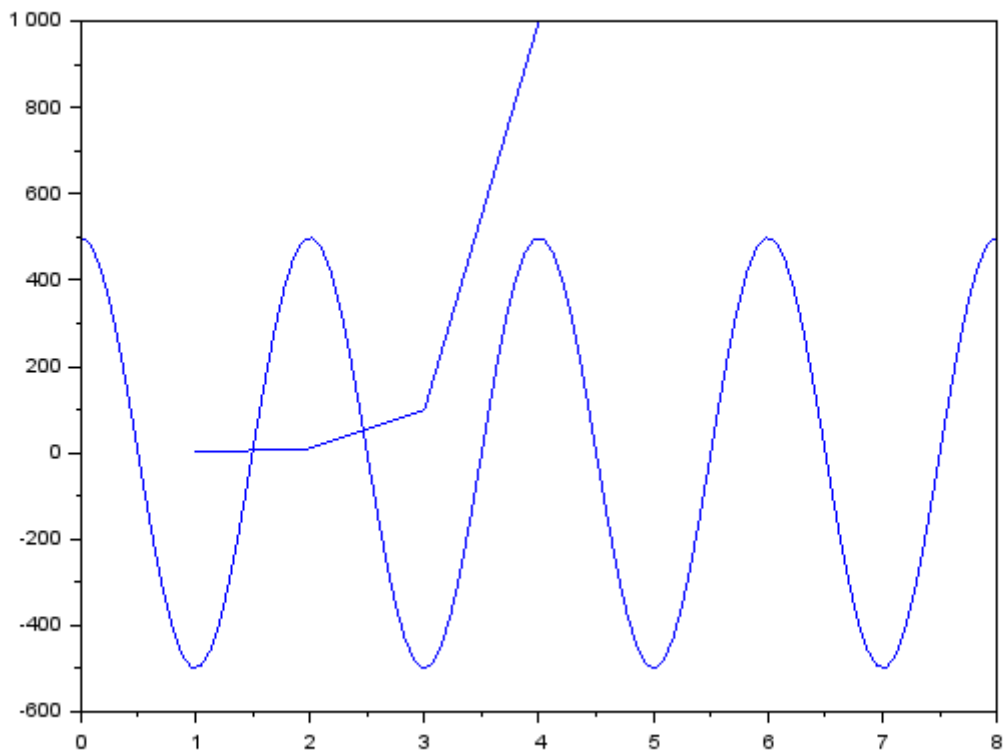


Figure 16. SciLab M-file demo graph

<https://www.scilab.org/>

Wolfram Alpha

A website that can solve challenging math problems with explanations. This is an amazing tool that you can ask wonky questions and it will generate an accurate response more often than not (for math problems). I like this tool because it shows the steps used to get to an answer.

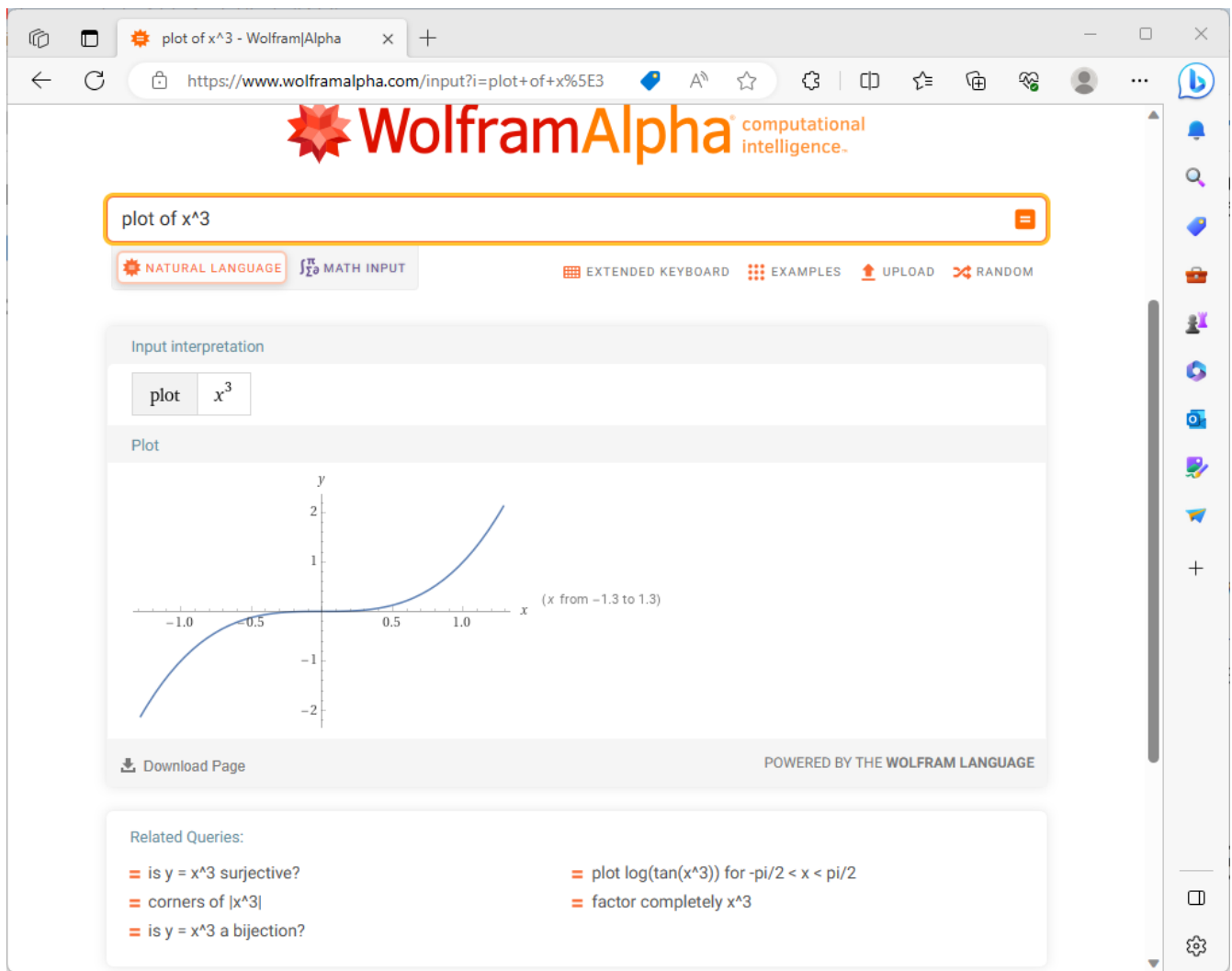


Figure 17. WolframAlpha Example

<https://www.wolframalpha.com/>

Programming Languages

Python

Python is a scripting language meaning the text to machine code conversion happens when you run the program. It is a loosely typed language; this means you can define a variable and it will figure out the datatype by what you assign to it

```
my_int = 10
my_float = 10.5
my_string = "This is my string of characters"
my_char = 'a'
my_int_list = [10, 20, 30, 40]
```

Here is an example that shows a complex function easily done in python. This program will find a NPT server on the internet and get accurate time to print

```
#!/usr/bin/env python
from socket import AF_INET, SOCK_DGRAM
import sys
import socket
import struct, time

def getNTPTime(host = "pool.ntp.org"):
    port = 123
    buf = 1024
    address = (host,port)
    msg = '\x1b' + 47 * '\0'

    # reference time (in seconds since 1900-01-01 00:00:00)
    TIME1970 = 2208988800 # 1970-01-01 00:00:00

    # connect to server
    client = socket.socket( AF_INET, SOCK_DGRAM)
    client.sendto(msg.encode('utf-8'), address)
    msg, address = client.recvfrom( buf )

    t = struct.unpack( "!12I", msg )[10]
    t -= TIME1970
    return time.ctime(t).replace(" ", " ")

if __name__ == "__main__":
    print(getNTPTime())
```

Python is capable of object oriented programming. This allows you to assign functions to a data type. Lets say you have a integer that is a distance; you can use a class to treat this as a radius that has a functions that reurn the circumfrence, area, and volume. This class can be used to inherit those functions to define a class that calculates the weight of a sphere of iron. This allows significant code reuse in ways that are not obvious at the start.

The main strength of Python is the code base that is contained in open source libraries. This means you can do very complex things with very little code since you only have to call the functions someone else has written. Python with a few libraries (numpy, matplotlib) allow manipulation an graphing at a level on par with Matlab. The libraries allow python to talk internet protocols and can talk to hardware interfaces without much effort. There is also significant support for Artificial Inteligence in python.

Python has a package manager to install the libraries you require called **pip**. sometimes you will need to force it to update the python3 install with the **pip3** command

<https://www.python.org/>

Anaconda is a packaged version of Python that will help setup your environment. This includes helpful libraries for math and science.

<https://www.anaconda.com/>

Python has even been ported to many 32bit microcontrollers and is gaining support in the Arduino ecosystem:

<https://micropython.org/>

<https://circuitpython.org/>

C or C++

C is a basic language that is much closer to assembly language than [Python](#). This usually means the execution will be faster when your app is developed in C. C is the base language that defines most of the syntax used. C++ is an extension that allows object oriented programming through classes that can be inherited from. [Arduino IDE](#) uses a subset of C that is easier for beginners. In non [Arduino IDE](#) apps the entry point is a function called main.

Here is a table of basic datatypes in C. Here I give the explicit names of the types instead of the general forms like int, short, long, longlong...

Table 1. Datatypes in C/C++

Datatype	Size	Notes
char	1 Byte	Ascii coded character: https://www.ascii-code.com/
int8_t	1 Byte	Signed byte value capable of storing an integer in the range of [-128 - 127]
uint8_t	1 Byte	Unsigned byte value capable of storing an integer in the range of [0 - 255]
int16_t	2 Bytes	Signed integer capable of storing an integer in the range [-32,768 - 32,767]
uint16_t	2 Bytes	Unsigned integer capable of storing an integer in the range [0 - 65,535]
int32_t	4 Bytes	Signed integer capable of storing an integer in the range [-2,147,483,648 - 2,147,483,647]
uint32_t	4 Bytes	Unsigned integer capable of storing an integer in the range [0 - 4,294,967,295]
float	4 byte	Floating point number in the range [1.2E-38 to 3.4E+38] with 6 decimal places of precision. Here we have 1 bit for sign, 8 bits for exponent, and 23 bits for mantissa.
double	8 byte	Floating point number in the range [2.3E-308 to 1.7E+308] with 15 decimal places of precision. Here we have 1 bit for sign, 11 bits for exponent, and 52 bits for mantissa.
long double	10 byte	Floating point number in the range [3.4E-4932 to 1.1E+4932] with 19 decimal places of precision. Here we have 1 bit for sign, 15 bits for exponent, and 112 bits for mantissa.

Floating point numbers are special and there are a few things that should be known. Floating point numbers in C are in a defined standard from IEEE: IEEE-754. This defines the Most significant bit as the sign bit, Then there is an exponent, and a mantissa. The mantissa is a fractional number less

than or equal to 1 (without the leading 1); the length of the mantissa determines the precision. The exponent part can specify a positive or negative exponent of base 2; the exponent determines the range of the floating point number. Since floating point numbers can have uncertainty, care must be used in comparisons. I would avoid absolute comparisons `A == 3.14159` since this test can fail because of a rounding error. If you need an 'equal' comparison, I would limit the precision of the thing I am comparing and then round the float to the exact precision that is less than the float precision with a known rounding algorithm.

Complex datatypes are available in C/C++. These include arrays, character strings, structures, unions, and classes. Arrays are a list of the datatype the array is defined with. Array elements can be addressed by using an index to point to the element of interest. Character strings are essentially arrays of character elements that are terminated by a NULL character, `0x00`. This allows the programmer to allocate a maximum space, but only use a subset of that space for the string data when it is shorter than the maximum. Structures allow a grouping of many types of data together. This is helpful if there is a relation between the different data elements stored in a struct. Unions are similar to a structure except that all elements share the same memory. This is a clever way that a single memory location can be treated as an integer byte or a character byte in your program. Classes were added to allow C++ object-oriented design. They can be thought of as a structure with the addition of functions that use the data.

One of the major strengths of C/C++ is the ability to use pointers. Pointers are variables that point to a place in system memory. Pointers have a datatype associated with it to tell the compiler how to treat the memory element. Pointers are declared much like regular variables only pointers have a '*' character before the variable name. Extreme care must be used when working with pointers.

The primary compiler for C/C++ programs is the GNU GCC compiler. This is open source and has been ported to nearly every platform you will find.

The following example asks for the number of terms and then calculates and prints that number of Fibonacci terms.

```
#include <stdio.h>
int main() {

    int i, n;

    // initialize first and second terms
    int t1 = 0, t2 = 1;

    // initialize the next term (3rd term)
    int nextTerm = t1 + t2;

    // get no. of terms from user
    printf("Enter the number of terms: ");
    scanf("%d", &n);

    // print the first two terms t1 and t2
    printf("Fibonacci Series: %d, %d, ", t1, t2);
```



```
// print 3rd to nth terms
for (i = 3; i <= n; ++i) {
    printf("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
    nextTerm = t1 + t2;
}

return 0;
}
```

<https://gcc.gnu.org/>

Ruby

Ruby is a popular language for web facing applications. Asciidoctor was written in it.

<https://www.ruby-lang.org/en/>

Tcl

Tcl (pronounced tickle) is a simple scripting language that had very loose licensing that allowed many companies to incorporate it into their tools. I see this in almost all of the FPGA tools that I use.

<https://www.tcl.tk/about/language.html>

Linux Shell

The Linux shell is the command prompt in Linux. There are many varieties of shell including bash, dash, and c shell. Bash is the default in most Linux systems except Ubuntu since that uses dash. C shell has some c like commands built into the prompt that make some tasks easier. Some basic linux commands are:

- pwd → print path of current directory
- ls → list the files in the current directory
- alias → create a macro ie **alias ll='ls -l'**
- cat → prints the contents of a file
- less → interactively prints the content of a file (allow keys navigate)
- cd → change directory
- mkdir → make directory
- cp → copy a file
- mv → move a file
- ln -s → create a symbolic link to a file

Many Linux utilities take a file or stream in and output one or more streams. This allows piping on programs output to another program. Some special characters to pipe are:

- `|` connect a program's output to another program's input. Use this `cmd1 [args] | cmd2 [args]`
- `>` redirect output to a file
- `>>` redirect output to an appended file
- `tee` This allows you to do more than one operation on the output of a program

<https://www.gnu.org/software/bash/>

<https://wiki.archlinux.org/title/Dash>

Communication Tools

SSH

This is an amazing tool to communicate between devices. This tool has two parts, a server and a client. The server runs on the machine you connect to, and the client is the tool you use to connect. Once a connection is made you are presented with a prompt from the remote device; this allows remote control of devices. Graphics can be forwarded through the SSH tunnel if you enable that feature. The best part of SSH is the level of security it can provide.

SSH can be set up to avoid using passwords by using asymmetric keys.

<https://www.openssh.com/>

Virtual Network Computing, VNC

VNC allows a remote graphical connection. The VNC protocol compresses the graphical content so the latency is improved even if the connection is slow. This tool allows you run a computer remotely even though the interface is graphical.

<https://www.tightvnc.com/>

PuTTY

Putty is a serial terminal for windows. This tool allows you to save sessions and easily connect later. Supports serial and ssh connections. Please note that this tool does support SSH keys, but it stores the keys in a different format.

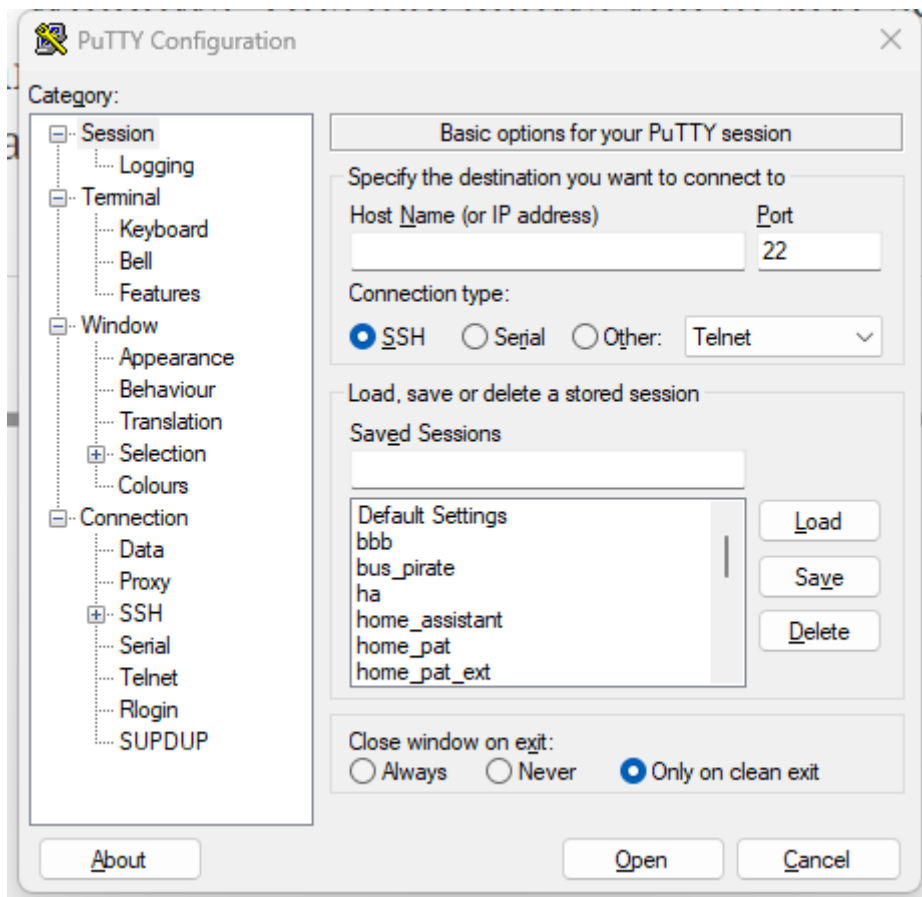


Figure 18. Putty Example

<https://www.putty.org/>

Minicom

a Linux serial terminal program. To start **minicom -s** will start with a menu to configure the port. In linux the serial ports are found in `/dev/tty*`.

<https://help.ubuntu.com/community/Minicom>

Remote Desktop Protocol

Remote desktop protocol is a Windows supported method for remote control similar to [Virtual Network Computing](#), VNC. Now Linux distributions are adding support for this protol.

<https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol>

Circuit Design

Kicad

Kicad is an open source schematic capture and layout tool for printed circuit boards. This is a free tool that allows you to design a printed circuit board. You can add new parts to the library and

model them both in the schematic and in the layout. This means create a picture for the schematic and a footprint for the layout. The gerber outputs from this tool are acceptable to circuit board fabrication companies. The cost of creating a PCB has come down to ~\$1.00 per board making this tool useful.

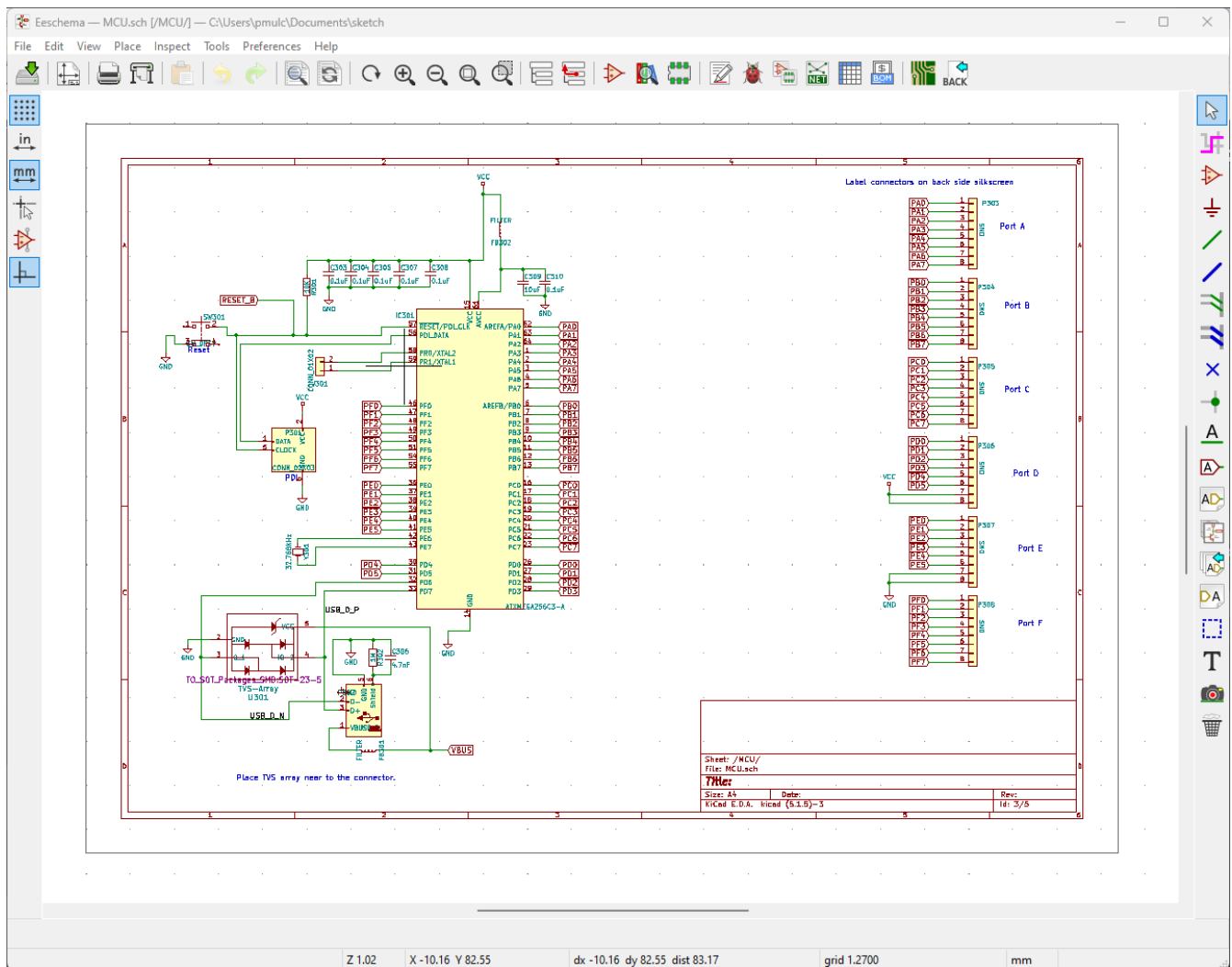


Figure 19. Kicad example

<https://www.kicad.org/>

FPGA Tools

FPGA are field programmable Gate arrays. These programmable devices allow you to create custom logic circuits that can be changed just by loading a new bitfile. FPGA's use LookUp Tables or LUT's to implement logic. Other logic element commonly available in FPGA's are: * Blockram * Hard Multipliers * PCIE interfaces * Hard processor cores * ...

Ecellium Logic Simulator

Excelium is a FPGA simulator from Cadence. This tool allows you to fully see how a FPGA design in a way that is not available once the FPGA runs in hardware.

https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-

Modelsim

Modelsim is a FPGA simulator with features similar to [Ecellium Logic Simulator](#). This is the most popular tool.

<https://eda.sw.siemens.com/en-US/ic/modelsim/>

Aldec

Aldec is a simulator/editor environment from a smaller company. I like this tool because I have found it to adhere to the language standard better than other tools.

<https://www.aldec.com/en>

Synplify Pro

Synplify Pro is a synthesis tool that can take the text of FPGA code and map it to the basic hardware components like LUTs, registers, and blockram.

<https://www.synopsys.com/implementation-and-signoff/fpga-based-design/synplify.html>

Vivado

Vivado is the Xilinx tool to create a bitfile for Xilinx parts. This tool comes with an editor, simulator, block diagram editor, and tools to create FPGA bitfiles. The bitfile is the file that gets loaded.

<https://www.xilinx.com/products/design-tools/vivado.html>

Libero

Libero is the vendor tool for Microsemi/Microchip parts. This toolk is trying to be like Vivado.

<https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/fpga/libero-software-later-versions>

Audio Tools

Audacity

Audacity is a visual audio editor. It contains audio filters that are handy. I like to normalize audio from different sources to have a uniform volume.

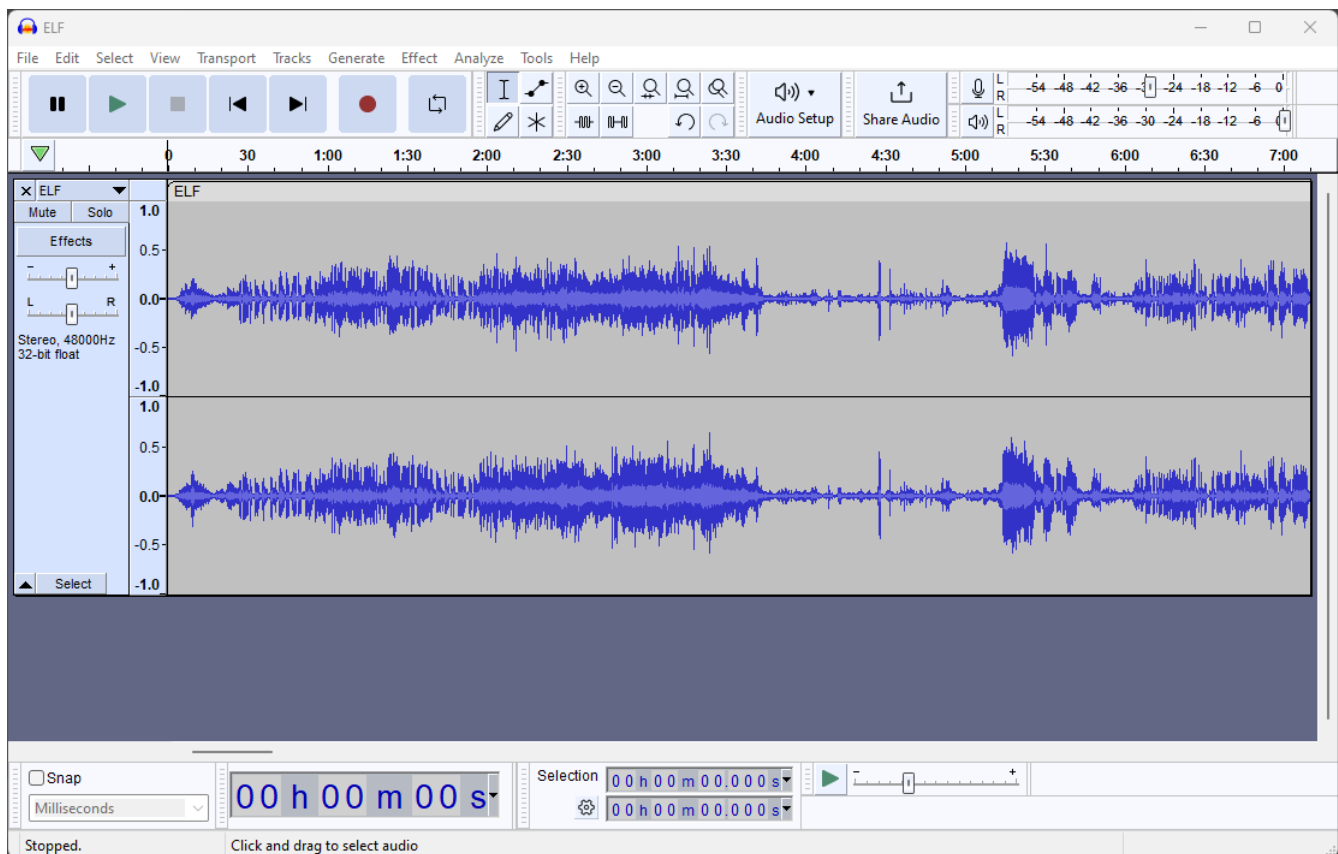


Figure 20. Audacity Screenshot

<https://www.audacityteam.org/>

Virtualization

VirtualBox

Oracle Virtualbox will allow you to run a virtual system without exiting your main OS. These tools allow sharing of development environments since the entire environment can be saved as a file. Gross version control can be had by saving snapshots of your environment. You can configure this tool to share resources from the host OS. This includes a virtual disk drive that you can create.

<https://www.virtualbox.org/>

Docker

Docker is a lightweight virtual environment since it uses the host kernel for low level operations. Many companies are providing docker containers. This allows full control of the environment. There is a penalty to any virtual environment in performance. I also feel like I am managing more and more environments due to docker.

Windows Subsystem for Linux, WSL

Starting in Windows 11, an option was added to Windows OS that allows you to run Ubuntu in a virtual machine that was part of windows. This is called WSL (Windows Subsystem for Linux).

Although this is not a full Linux environment, it does allow you to do most Linux things without booting in Linux. Now even graphical programs are supported. [Ubuntu Linux](#) is supported.

Operating Systems

Microsoft Windows

Microsoft Windows is the default operating system for most Personal Computers. Therefore you are probably familiar with it.

Microsoft has also updated their shell command prompt to something called powershell. This will allow more Linux-like commands.

Ubuntu Linux

Popular Linux distribution with community support for non expert users. Many development environments force you to use Linux. The default desktop is similar to Windows where there is a menu to select what program you want to run.

For terminal operations, I like using a program called byobu. Other similar programs are GNU screen and tmux. These tools allow you to switch between multiple terminal sessions and keep the sessions active even after I loose connection. This is why I use these tools when I am remoting in over an SSH connection. The [Linux Shell](#) section has information of what to type in a terminal session.

<https://www.byobu.org/>

<https://ubuntu.com/>

Security

Bitwarden

Bitwarden is an open wource tool that can save your passwords. You will find that you end up having passwords for too many things to keep track of. I like this tool since it is cross platform and far more secure than a browser or phone OS.

<https://bitwarden.com/>

Protocols

YAML

YAML is a human readable file format that can be used for configurations. It is based off of XML that Microsoft uses for many application files. I like the ease of creating and the complex data

structures that are created from the contents of this type of file. My one complaint is that it uses invisible characters (spaces) in its interpretation of the file.

<https://yaml.org/>

MQTT

MQTT is a simple internet protocol like TCP/IP. The benefit of MQTT is that simple microcontrollers have good libraries to talk this protocol. So you can have a tiny ESP32 like microcontroller that can read a temperature sensor and have it broadcast it over WIFI to redundant data logging computers. This means you can automate lab tests in a way that can scale up with the complexity.

There are apps for your phone that can talk this protocol.

<https://mqtt.org/>

Domain Name Server

This is what your browser uses to find an internet address for a human readable address (ie. `www.google.com` \Rightarrow `143.244.220.150`). There are clever tricks that can be used with this tool. Your computer usually assumes it is the first DNS server it will check. That means you can assign addresses to named entities just by editing a simple hosts file.

<https://www.cloudflare.com/learning/dns/what-is-dns/>

Consistent Overhead Byte Stuffing

Consistent Overhead Byte Stuffing can be used in serial communications to detect packet boundaries. This allows you to know when a message starts and ends. The message is altered by replacing all `0x00` bytes with a count to the next `0x00`. The message will always end with a `0x00` signifying the end of packet.

Simple parsing can remove the counts and return those bytes to `0x00` when the data is received.

https://en.wikipedia.org/wiki/Consistent_Overhead_Byte_Stuffing