**********Draft**********

09/16/20

BW Low Pass, BW High Pass and BW Band Pass
using signal of 50, 120, and 300 Hz
Testing on Raspberry Pi 4 4gB
using Raspberry Pi OS 32 bit and
Raspberry Pi 3B+ Bare Metal

**********Draft**********

https://github.com/develone/filter-c.git which was forked from
https://github.com/adis300/filter-c

git clone https://github.com/develone/filter-c.git

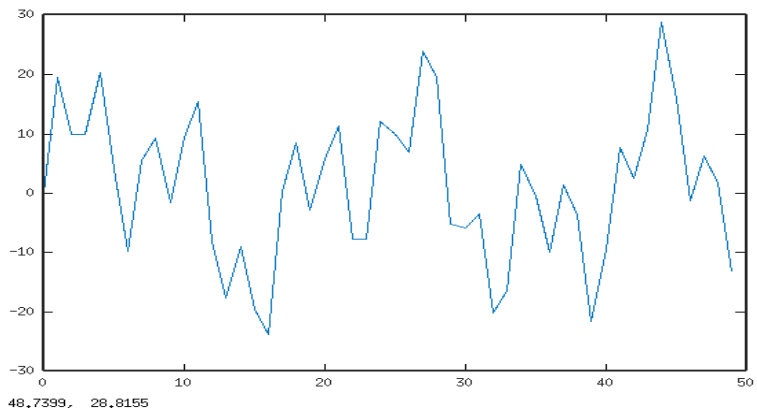
This example will create a BW Low Pass Filter.

cp example.c.low example.c
make clean; make

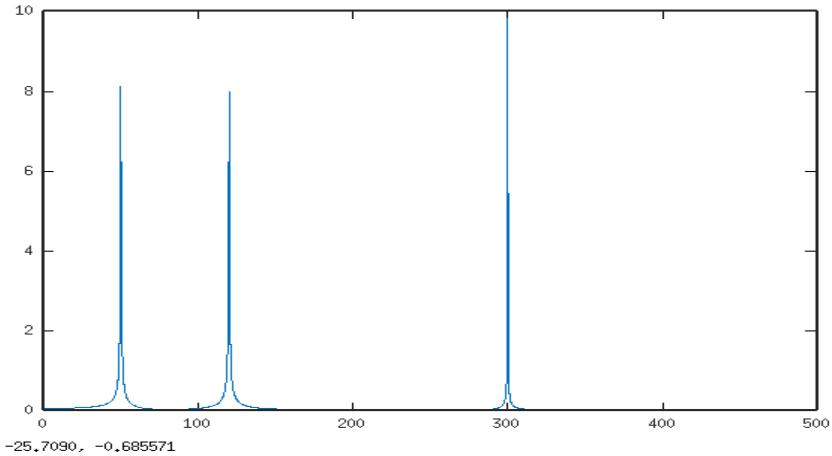
Writes 2 binary files “mysig.bin” & “myfilt.bin”.

ptrsignal = (double)10(sin(2*pi*50*t[i]) + sin(2*pi*120*t[i]) + sin(2*pi*300*t[i])); //no DC
Written to the file “mysig.bin”.

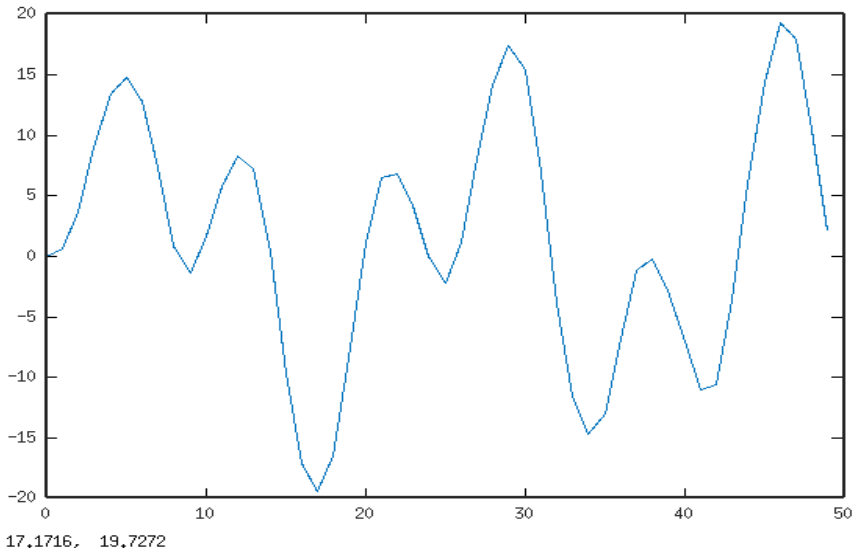
**********Low Pass**********



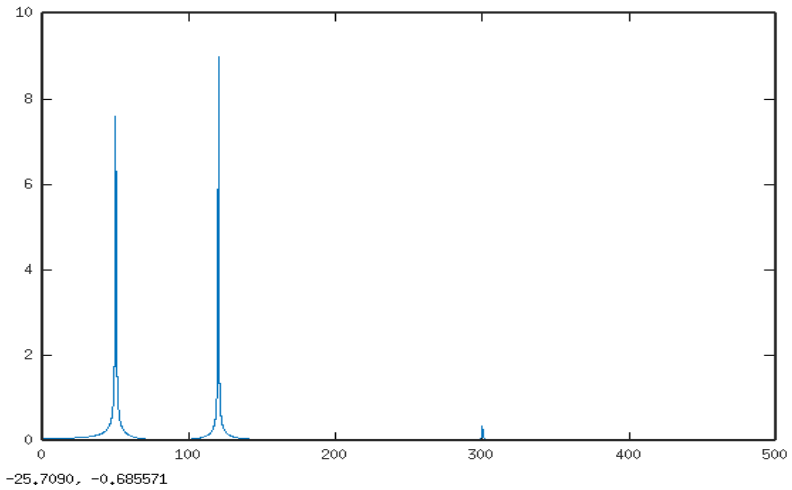
FFT computed in octave from “mysig.bin”.



The filttered data is written to the file “myfilt.bin”.



FFT computed from signal “myfilt.bin” create with example.c
FFT computed in octave from “myfilt.bin”.

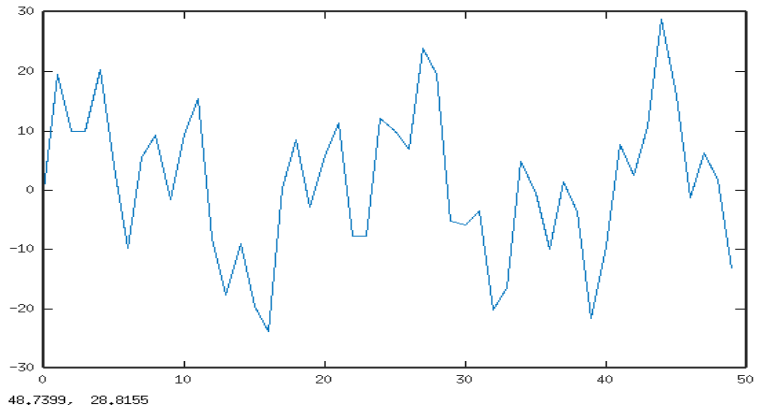


************High Pass************

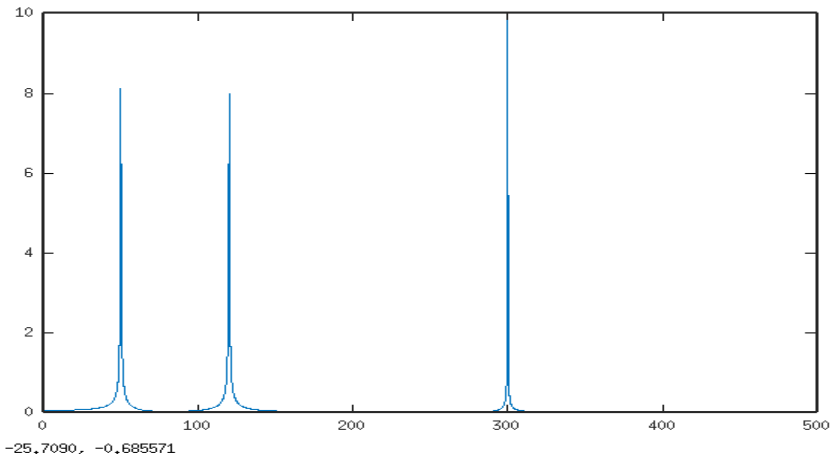
cp example.c.high example.c
make clean; make

ptrsignal = (double)10(sin(2*pi*50*t[i]) + sin(2*pi*120*t[i]) + sin(2*pi*300*t[i])); //no DC

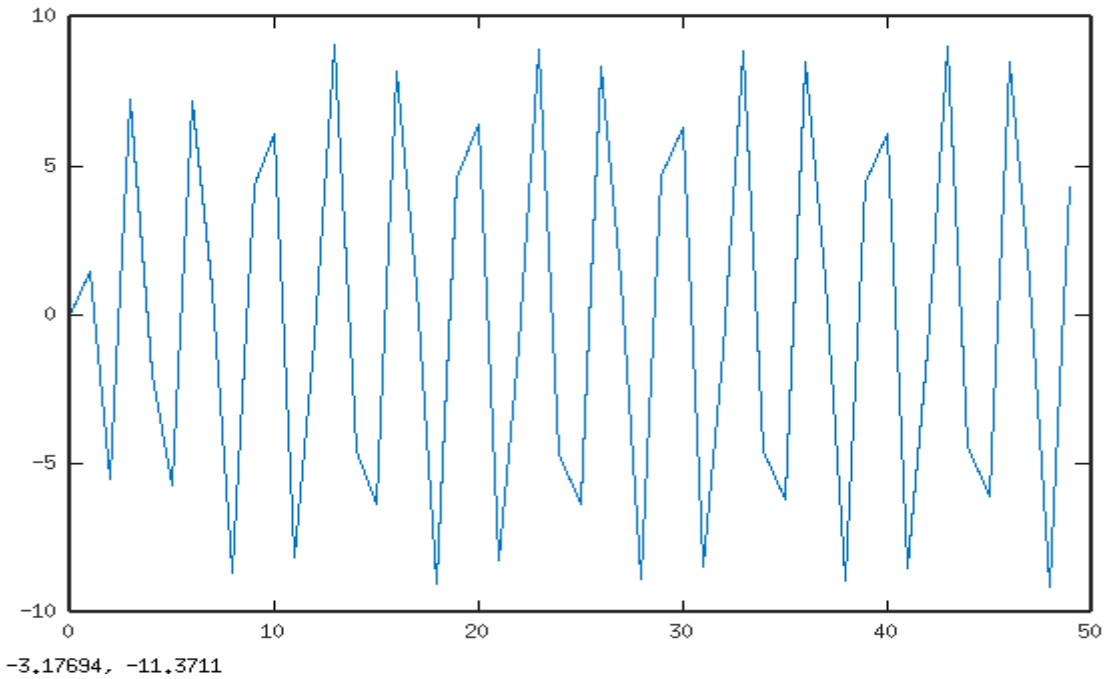
Written to the file “mysig.bin”.



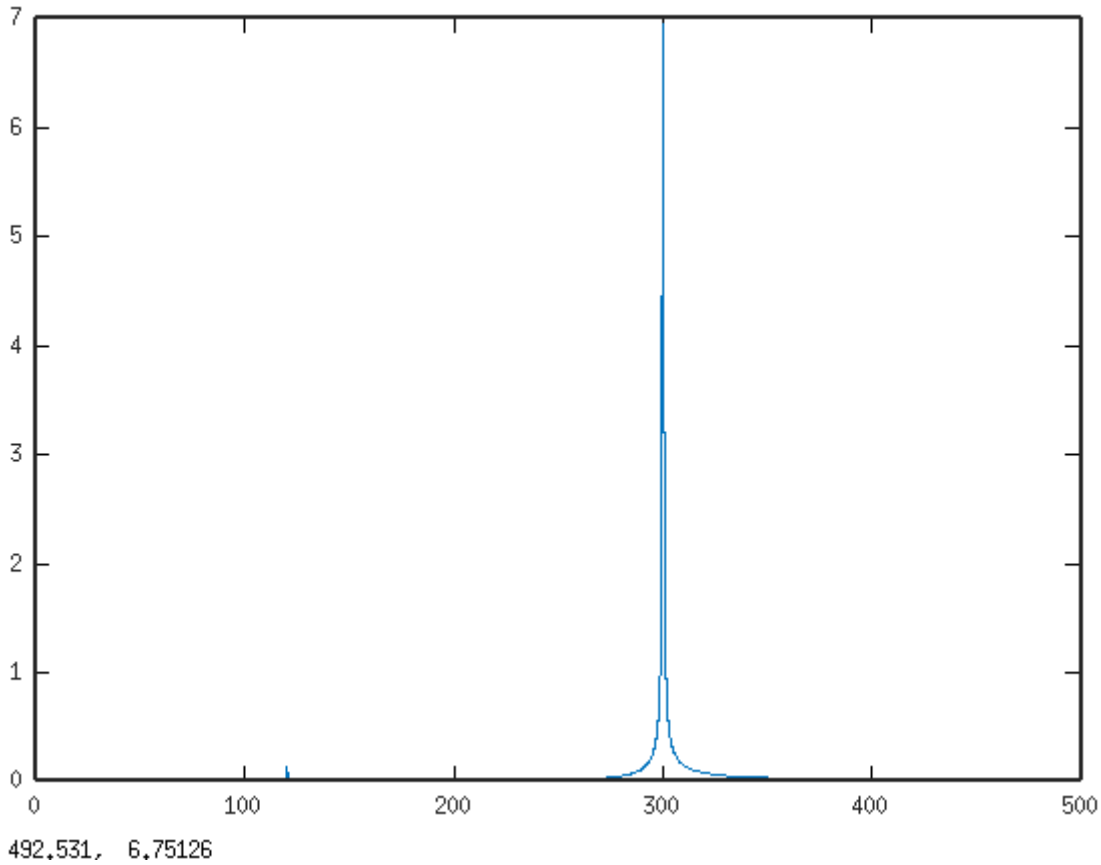
FFT computed from signal “myfilt.bin” create with example.c
FFT computed in octave from “myfilt.bin”.



The filttered data is written to the file “myfilt.bin”.



FFT computed from signal “myfilt.bin” create with example.c
FFT computed in octave from “myfilt.bin”.



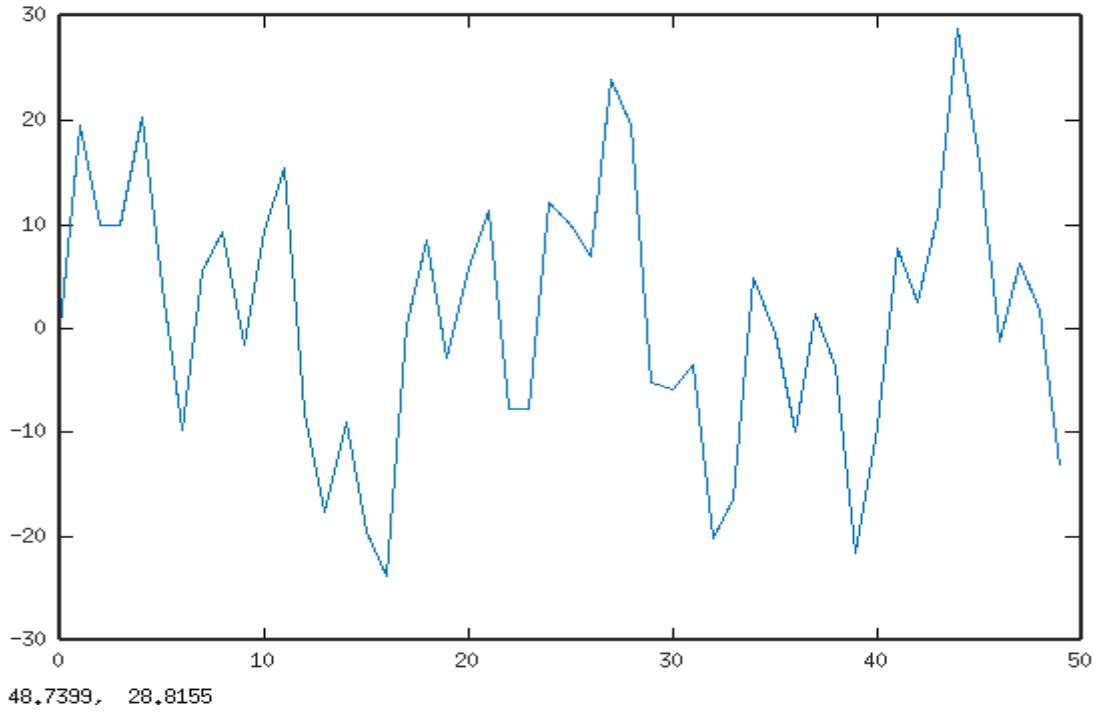
************Band Pass************

cp example.c.band example.c
make clean; make

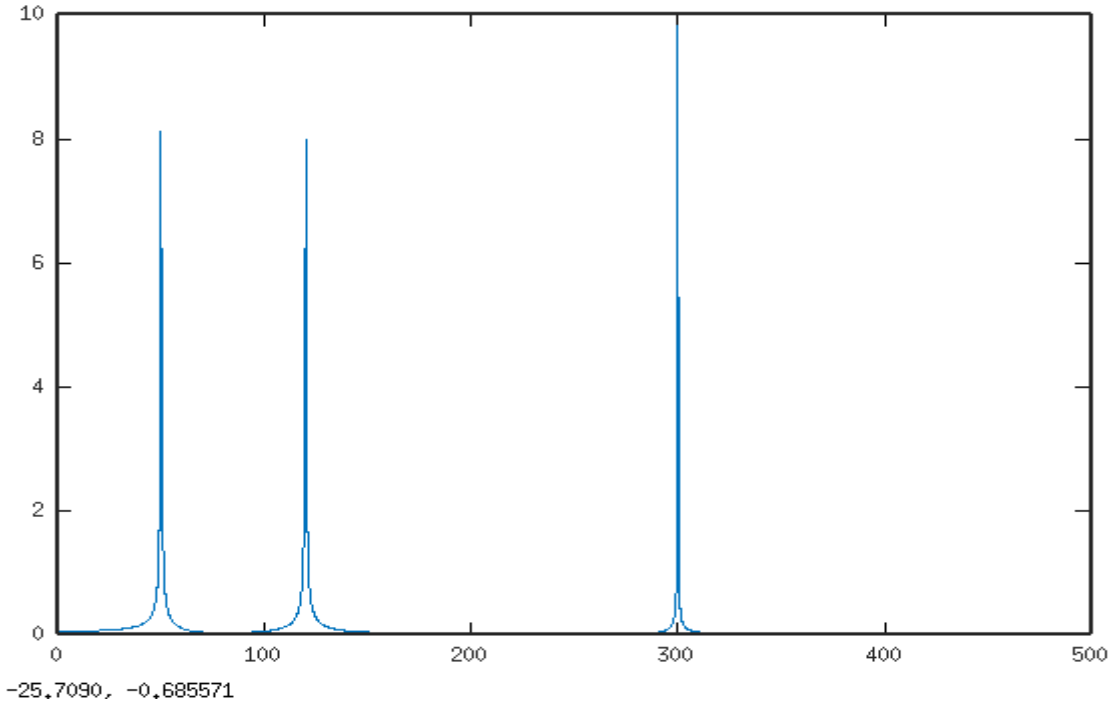
ptrsignal = (double)10(sin(2*pi*50*t[i]) + sin(2*pi*120*t[i]) + sin(2*pi*300*t[i])); //no DC

Written to the file “mysig.bin”.

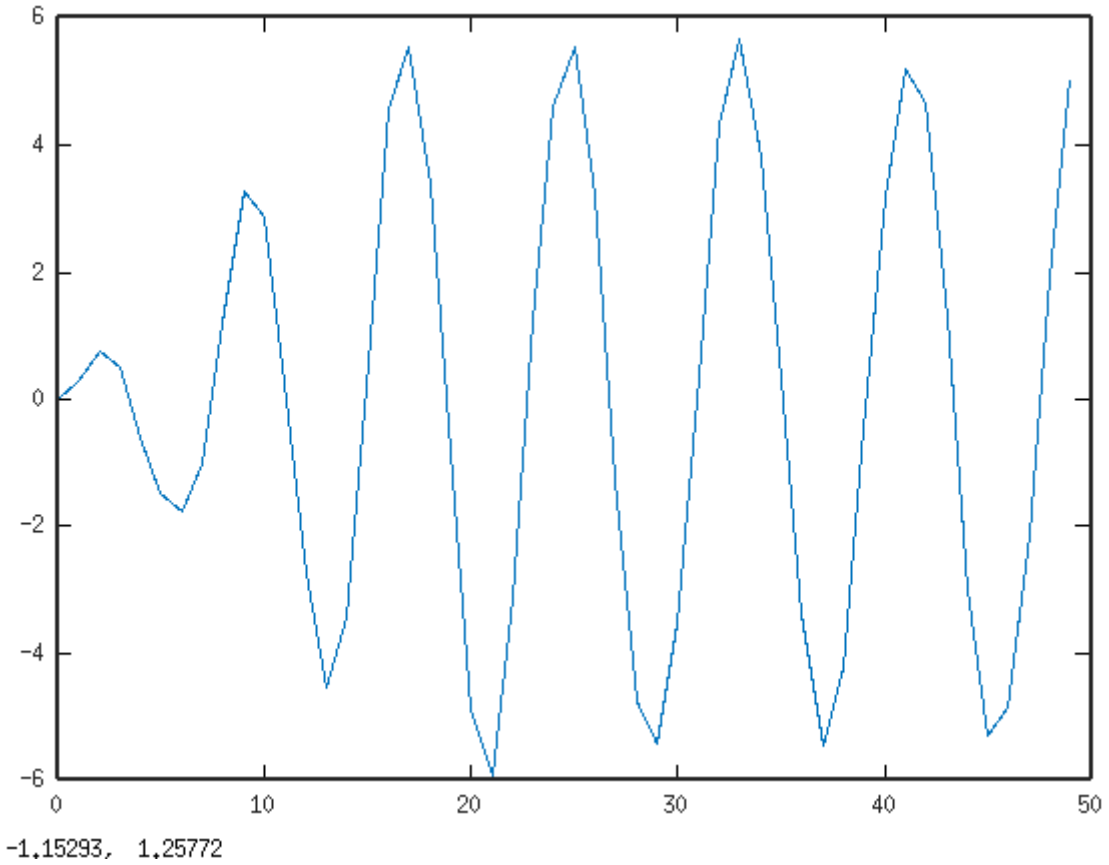
The filttered data is written to the file “myfilt.bin”.



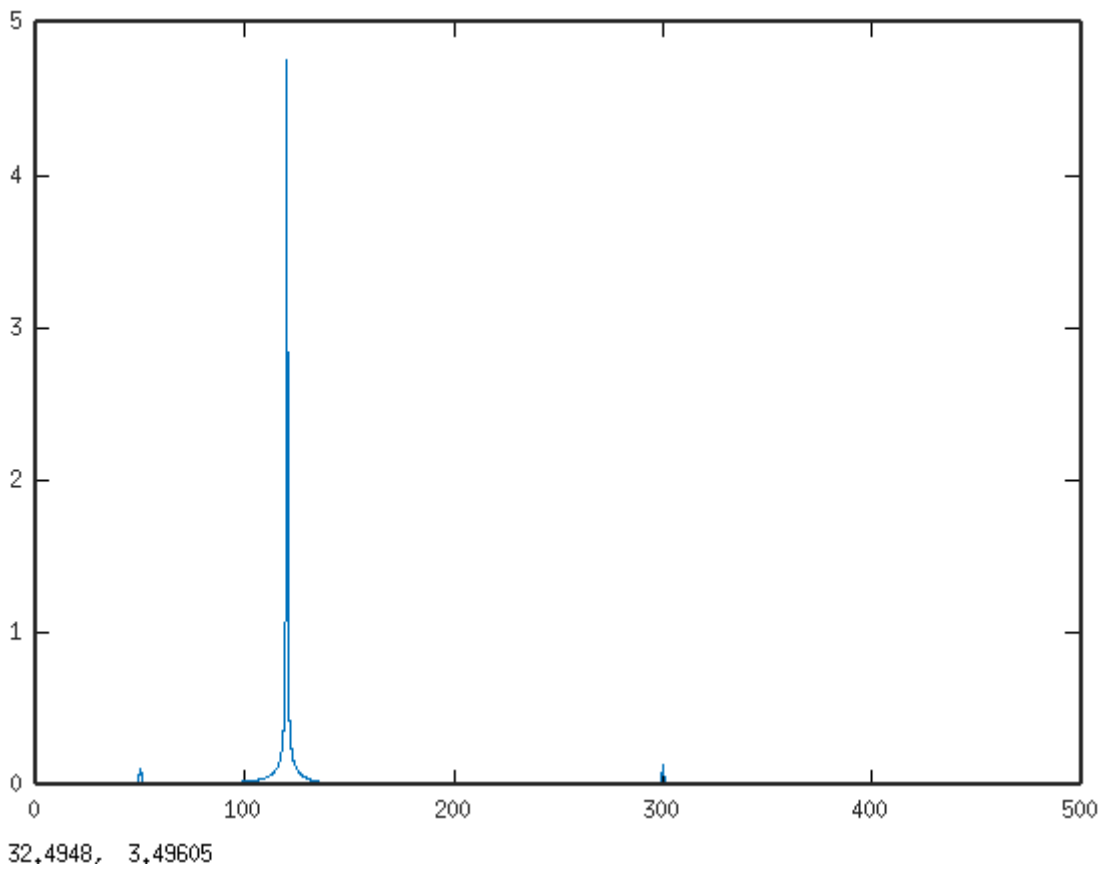
FFT computed in octave from “mysig.bin”.



The band pass signal read from myfilt.bin create_bw_band_pass_filter(4, 120, 15, 20);←-???
Trying to filter the 50 & 300 Hz parts of the signal



FFT computed from signal “myfilt.bin” created with example.c
FFT computed in octave from “myfilt.bin”.



The differences between running the (example.c.low, example.c.high, and example.c.band) in example.c program on the Raspberry Pi OS and Bare Metal
using Ultibo is the

```
diff example.c ~/Ultibo_Projects/FFT/filter-c/example.c
67c67
< int main() {
---
> int test() {
```

In the case of Bare Metal. The code is compiled and library is created.

```
#!/bin/bash
rm -f *.o
arm-none-eabi-gcc -O2 -mabi=aapcs -marm -march=armv7-a -mfpu=vfpv3-d16 \
-mfloat-abi=hard -D__DYNAMIC_REENT__  -c filter.c
arm-none-eabi-gcc -O2 -mabi=aapcs -marm -march=armv7-a -mfpu=vfpv3-d16 \
-mfloat-abi=hard -D__DYNAMIC_REENT__  -c example.c
arm-none-eabi-ar rcs libfilter.a *.o
arm-none-eabi-objdump -d libfilter.a > filter-obj.txt
cp libfilter.a RPi3
```

Lazarus IDE (Ultibo Edition) is used to compile and link to create “kernel7.img”.

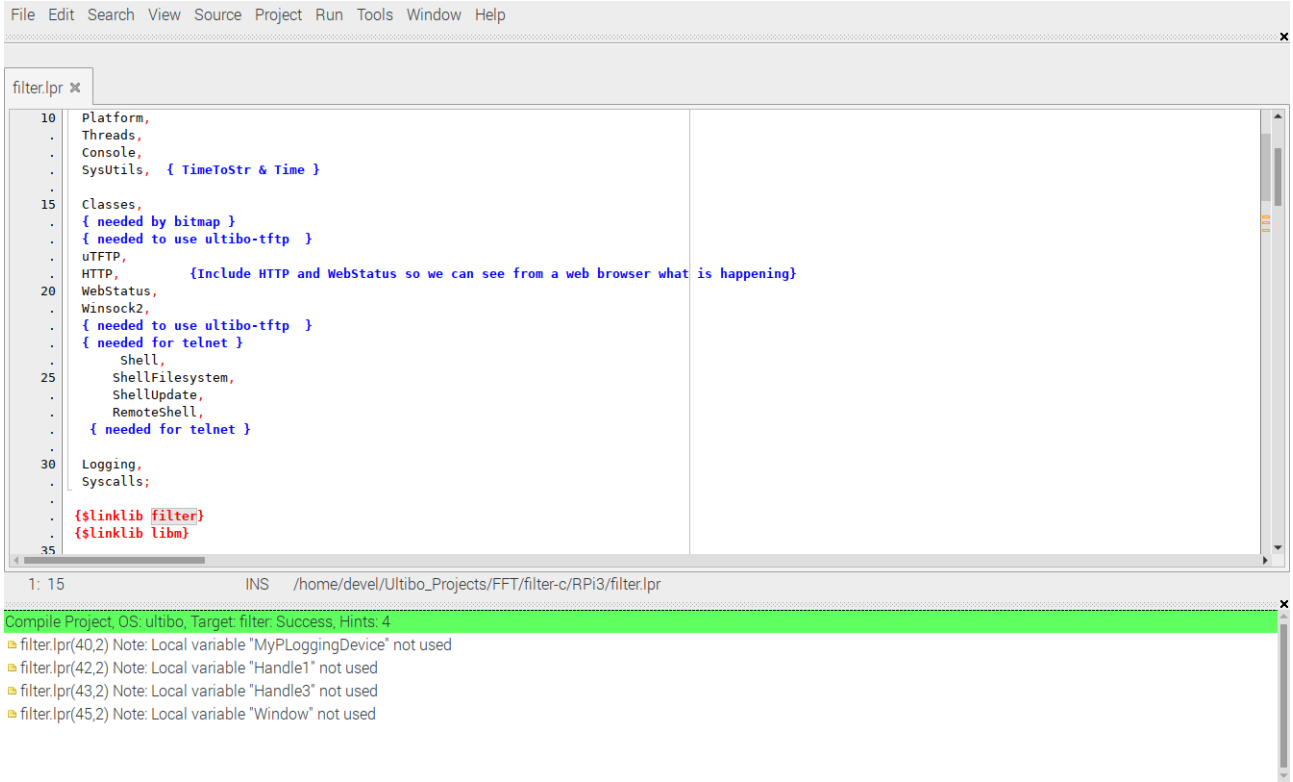
File Edit Search View Source Project Run Tools Window Help

filter

```
program filter;

{$mode objfpc}{$H+}

uses
 RaspberryPi3, {<-- Change this to suit which model you have!!}
 GlobalConfig,
 GlobalConst,
 GlobalTypes,
 Platform,
 Threads,
 Console,
 SysUtils,  { TimeToStr & Time }

 Classes,
 { needed by bitmap }
 { needed to use ultibo-tftp  }
 uTFTP,
 HTTP,        {Include HTTP and WebStatus so we can see from a web browser what is happening}
 WebStatus,
 Winsock2,
 { needed to use ultibo-tftp  }
 { needed for telnet }
      Shell,
     ShellFilesystem,
     ShellUpdate,
```

1: 15 INS /home/devel/Ultibo_Projects/FFT/filter-c/RPi3/filter.lpr

To compile & link “Run/Compile”



The Green bar indicates it was successful.

Then the kernel7.img can be transferred with the command
tftp xx.xx.xx.xx < cmdstftp
tftp> tftp> Sent 2905544 bytes in 8.1 seconds

At this point the Ultibo system reboots with the new kernel7.img

A connection to a remote shell can be made “telnet xx.xx.xx.xx”

```
File  Edit  Tabs  Help
Ultibo Core (Release: Beetroot Version: 2.0.807 Date: 3 September 2020)
 (Type HELP for a list of available commands)
>
```

The dir command provides the files on the Ultibo system.

```
File  Edit  Tabs  Help
 16-8-20 20:24:30              635016  teapot.obj.dat
 16-8-20 20:24:30              230454  tech_char_set_01.bmp
 16-8-20 20:24:32            31008729  test.h264
 2-9-20 23:59:18                36801  test.j2k
 16-8-20 20:24:32                  24  testfile
 16-8-20 20:24:32              921722  test_wr.bmp
 16-8-20 20:24:32               60367  tie04.cob
 24-8-20 19:12:06       <DIR>          extra_font
 27-8-20 00:34:10            55687680  a2002011001-e02.pcm
 26-8-20 21:36:58             9580548  a2002011001-e02.pcm.orig
 16-8-20 20:24:30                1242  configxx.txt
 3-9-20 12:38:26               921654  image2.bmp
 27-8-20 00:16:30            55687680  Crash.pcm.orig
 28-8-20 19:26:24               86960  dump-raw-3.txt
 15-9-20 17:33:34               71680  test-output.txt
 4-9-20 00:43:46                 3897  imagelogging.log
 3-9-20 12:49:54               750054  image3.bmp
 11-9-20 16:02:38                 389  lpcoeffsc
 16-9-20 12:16:51               16384  mysig.bin
 16-9-20 12:16:51               16384  myfilt.bin
          77 file(s) 191512126 bytes
          5 dir(s)

C:\>
```