Presentation

BY WILLIAM LASH

Intro

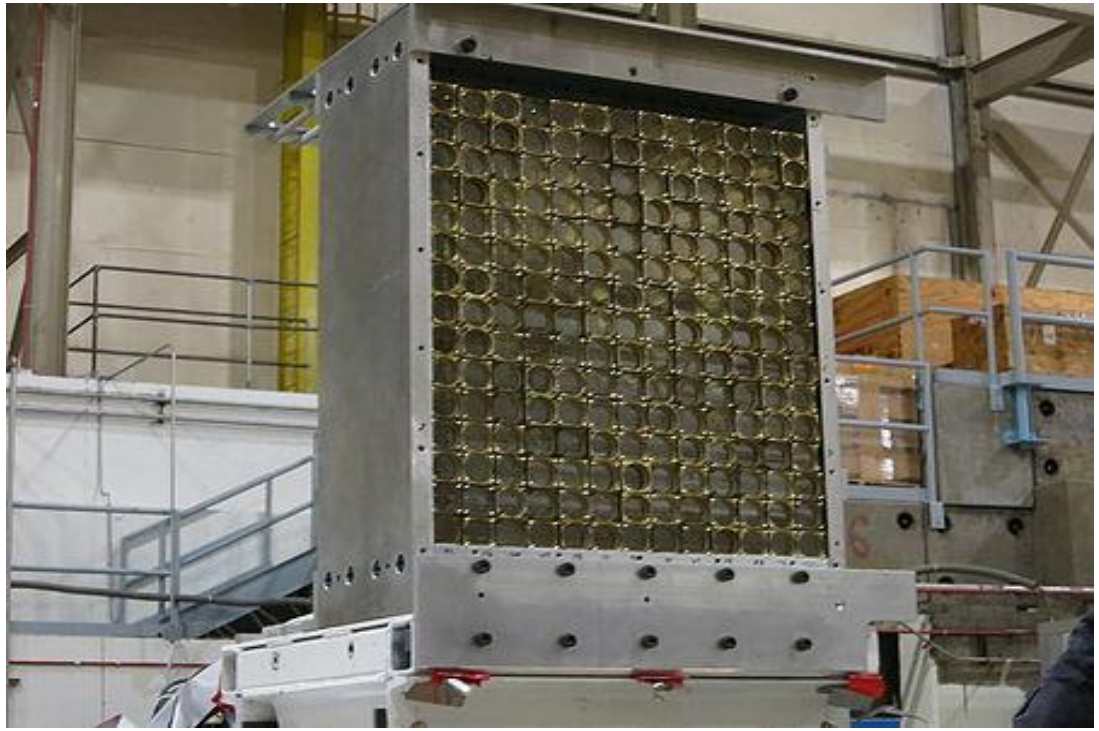
- In this presentation I will talk about my past project a little and describe the my new ideas for my past project.
- On top of that I will discuss the new project that I have be working on

Calorimeter

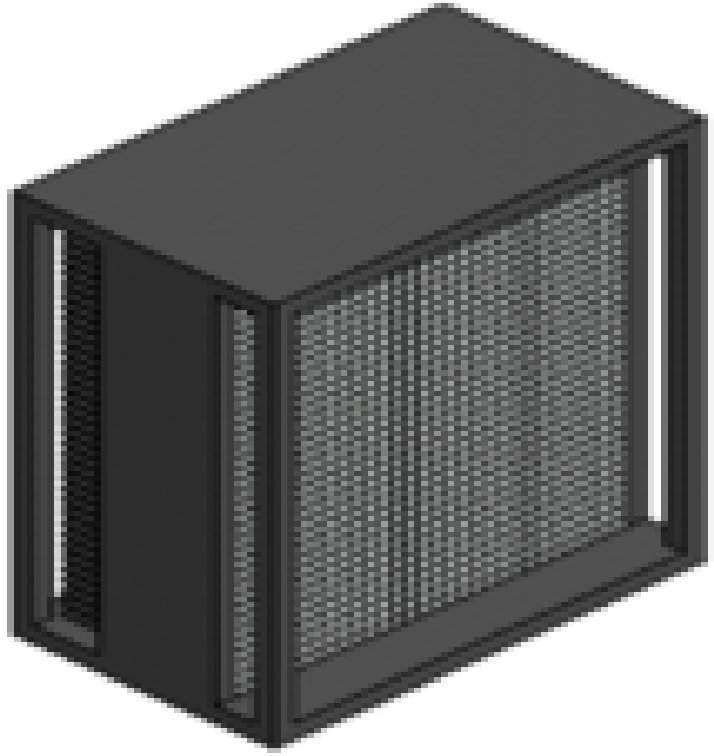
- There is a calorimeter that is currently being used at J-lab for several experiments. The new calorimeter we are planning to construct will contain about a thousand lead tungsten crystals.
- The goal of the new calorimeter is to study properties of neutral particles, Such as studying Pions, Muons and various other particles.

Calorimeter

The J-lab calorimeter



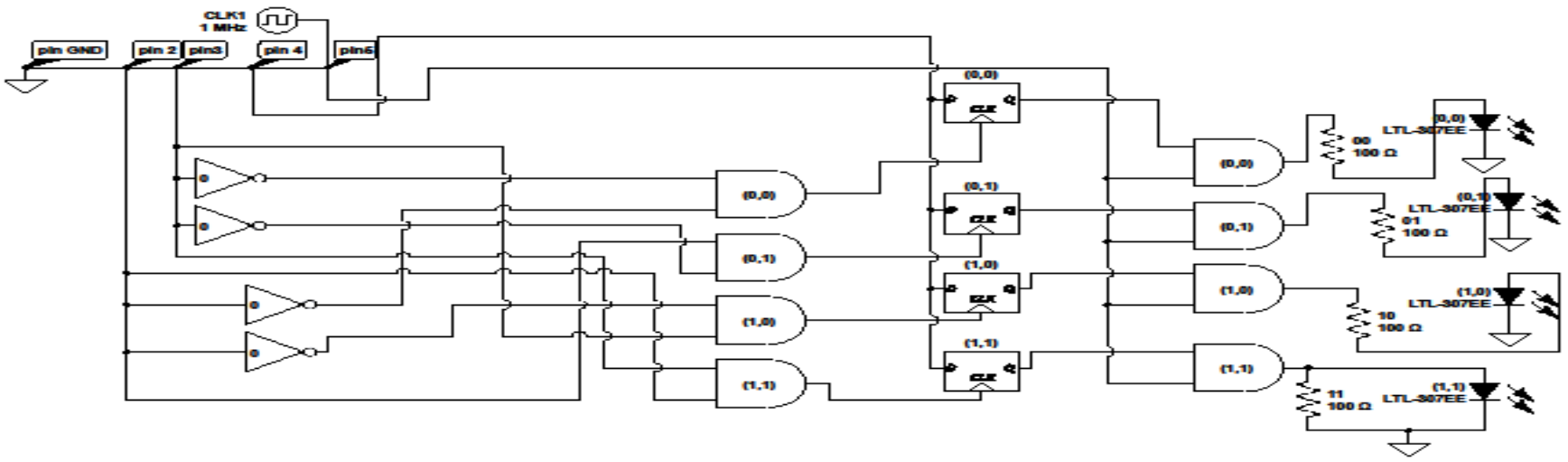
Our calorimeter design



Last Year LED Setup

- At the completion of our project last year we had produced a circuit that addresses LEDs, remembers that LED being addressed, then you can readdressed that LED, so that we can access each LED individually and control the frequency of they're blinks. This is all done through our controller an Arduino Uno, Which we programmed to perform the required functions.
- Our prototype did not scale up very well. This is because their were too many components that took up too much space on the circuit board. To fix this we need to look at a circuit with less components and more software manipulation.

Last Year Circuit

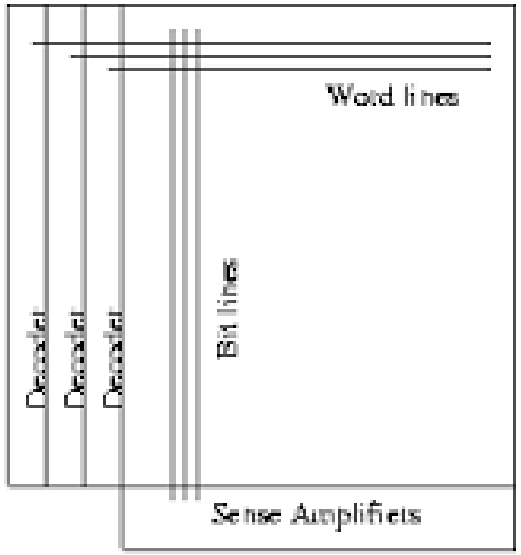


This Year's Objectives

- To redesign the Led circuit so it has the ability to control over a thousand LEDs.
- To design a user interface for a new high voltage board (V6533n)
- This means to use the given C++ code and improve upon it to make it much more user friendly.
- so that anyone can use it, without needing a class to use it.

New LED Design

- My professor suggested looking into Bit-line/Word-line architecture.
- Using an Arduino or other like device, we can send signals down each line.
- This will activate the transistors on each line.
- The spot where the lines intersect is where the LED will be
- This will only be active when both type of lines transistors are on

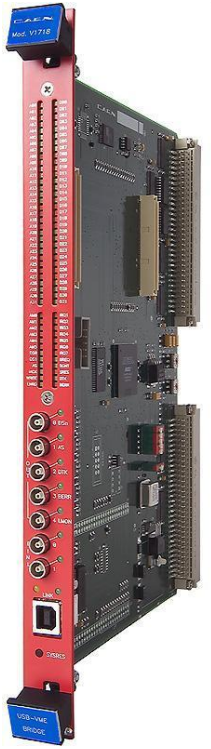


C ++

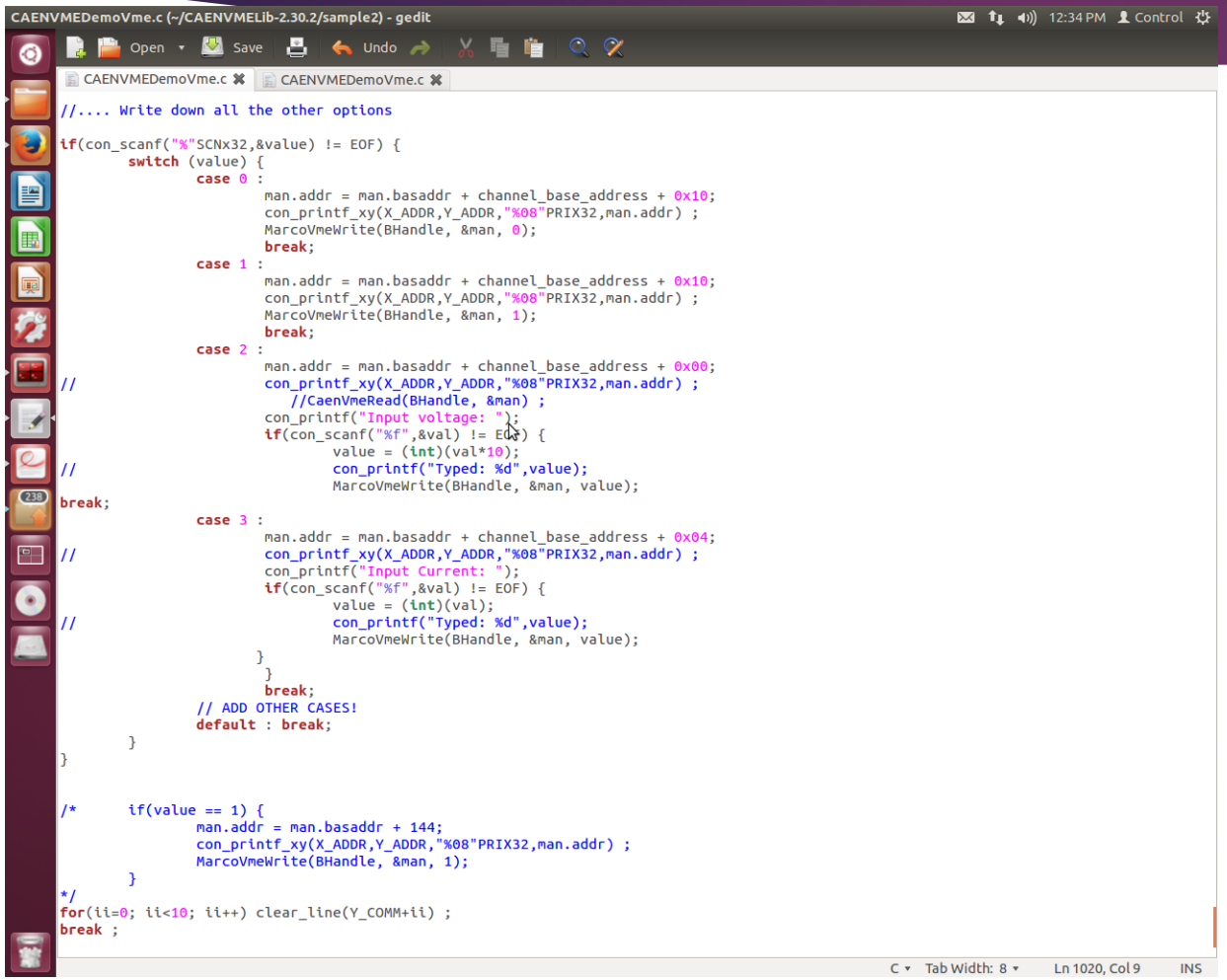
- C ++ is a programing language used to program on many platforms.
- It is a imperative object-oriented and generic programing language.
- It is designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights.
- This is the language in which our high voltage interface is written.

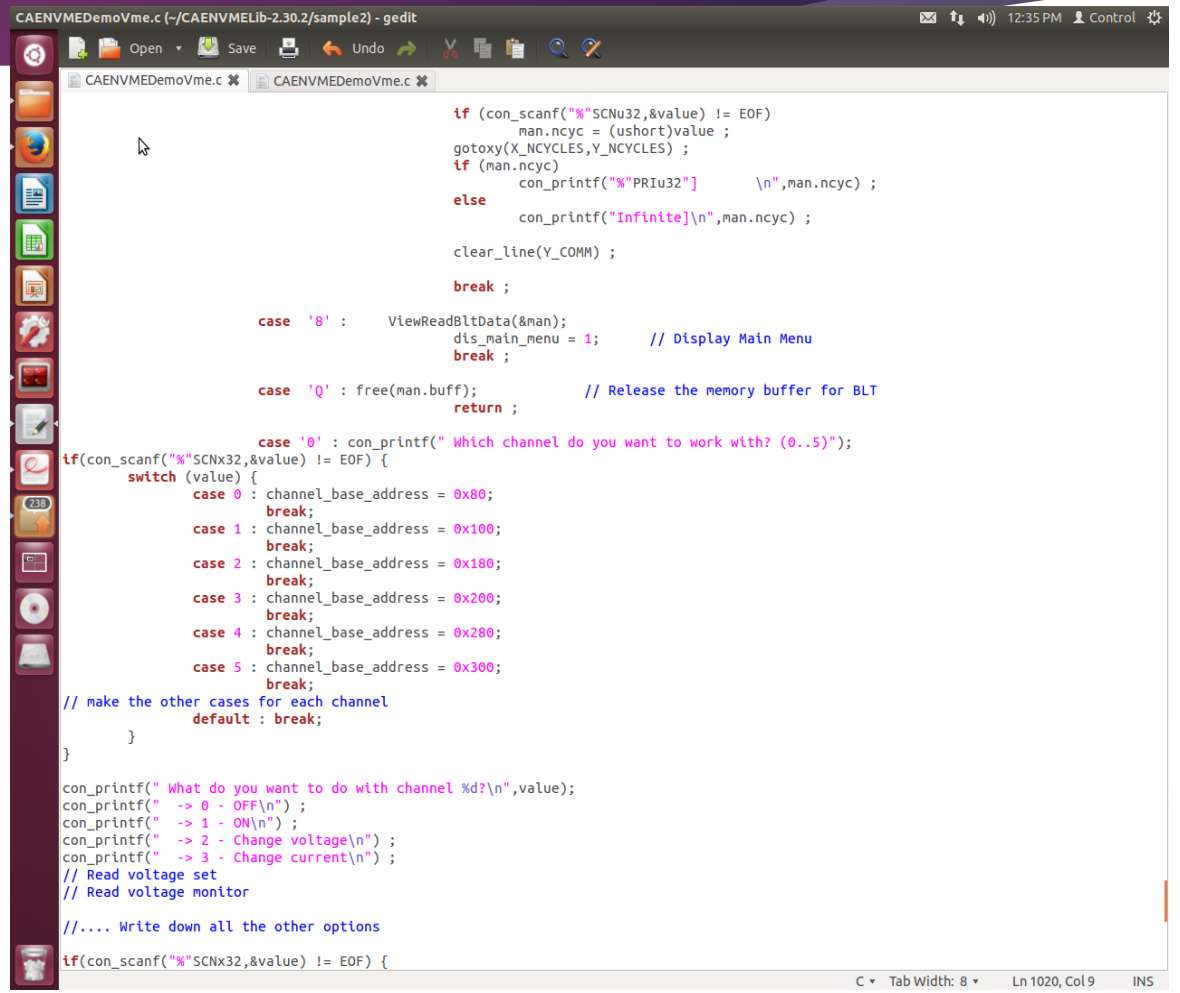
Components For High Voltage

- Board V1718
- This is the system controller for the high voltage supply or V6533N , this board allows us to use our C++ program to manipulate the high voltage source.
- Such as turning channels on and off, change the voltage and current, and so on.
- Board V6533N
- This is the high voltage source which has 6 channels (0 – 5). Up to 4kv and 3 ma



HV Code

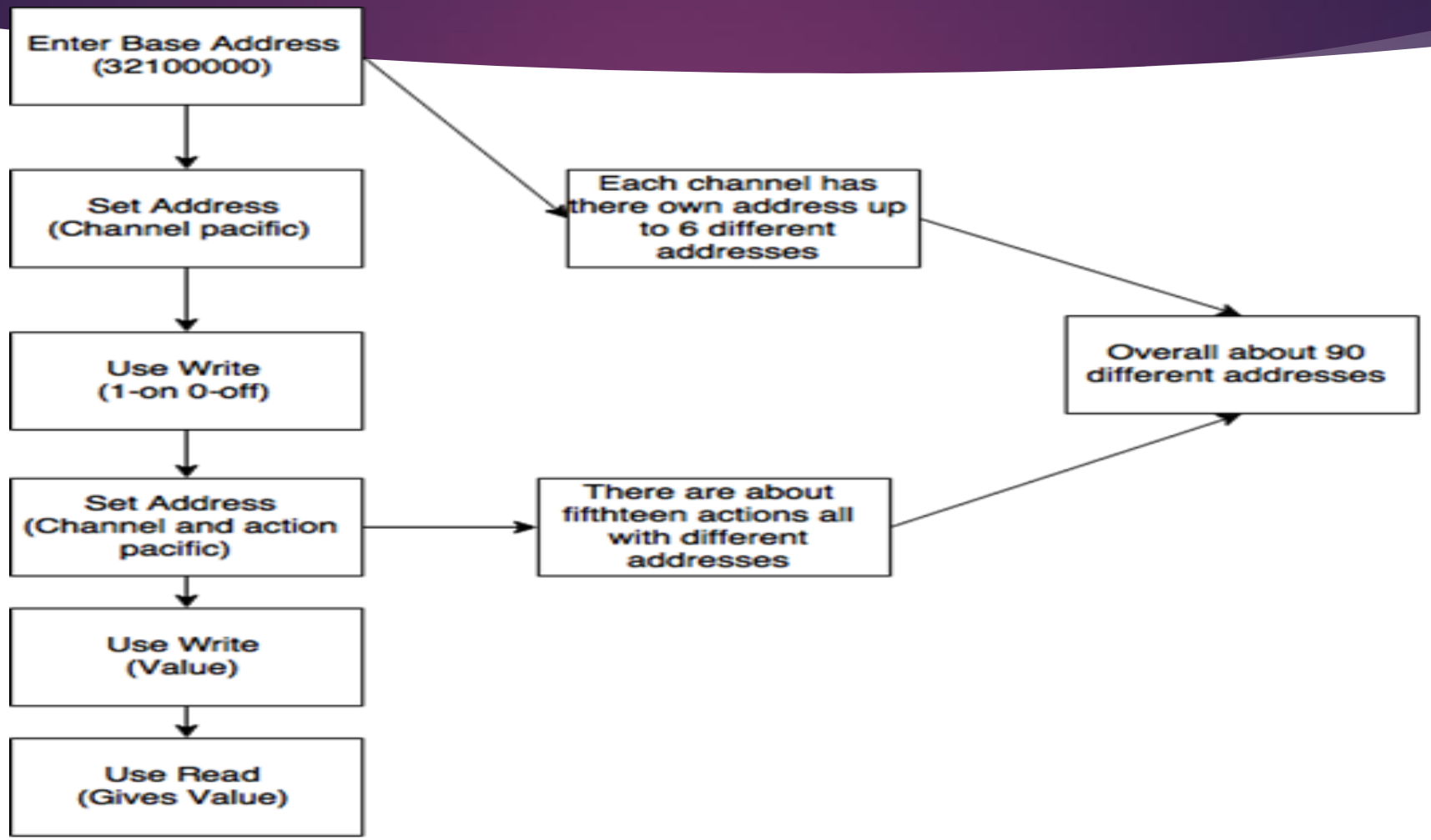

CAENVMEDemoVme.c (~/CAENVMELib-2.30.2/sample2) - gedit
12:34 PM
Control
Open
Save
Undo
CAENVMEDemoVme.c
CAENVMEDemoVme.c
//.... Write down all the other options
if(con_scanf("%"SCNx32,&value) != EOF) {
switch (value) {
case 0 :
man.addr = man.basaddr + channel_base_address + 0x10;
con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
MarcoVmeWrite(BHandle, &man, 0);
break;
case 1 :
man.addr = man.basaddr + channel_base_address + 0x10;
con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
MarcoVmeWrite(BHandle, &man, 1);
break;
case 2 :
man.addr = man.basaddr + channel_base_address + 0x00;
//
con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
//CaenVmeRead(BHandle, &man) ;
con_printf("Input voltage: ");
if(con_scanf("%f",&val) != EOF) {
value = (int)(val*10);
//
con_printf("Typed: %d",value);
MarcoVmeWrite(BHandle, &man, value);
break;
case 3 :
man.addr = man.basaddr + channel_base_address + 0x04;
//
con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
con_printf("Input Current: ");
if(con_scanf("%f",&val) != EOF) {
value = (int)(val);
//
con_printf("Typed: %d",value);
MarcoVmeWrite(BHandle, &man, value);
}
}
break;
// ADD OTHER CASES!
default : break;
}
}
/* if(value == 1) {
man.addr = man.basaddr + 144;
con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
MarcoVmeWrite(BHandle, &man, 1);
}
*/
for(ii=0; ii<10; ii++) clear_line(Y_COMM+ii) ;
break ;
C
Tab Width: 8
Ln 1020, Col 9
INS


CAENVMEDemoVme.c (~/CAENVMELib-2.30.2/sample2) - gedit
12:35 PM
Control
Open
Save
Undo
CAENVMEDemoVme.c
CAENVMEDemoVme.c
if (con_scanf("%"SCNu32,&value) != EOF)
man.ncyc = (ushort)value ;
gotoxy(X_NCYCLES,Y_NCYCLES) ;
if (man.ncyc)
con_printf("%"PRIu32"] \n",man.ncyc) ;
else
con_printf("Infinite]\n",man.ncyc) ;
clear_line(Y_COMM) ;
break ;
case '8' : ViewReadBltData(&man);
dis_main_menu = 1; // Display Main Menu
break ;
case 'Q' : free(man.buff); // Release the memory buffer for BLT
return ;
case '0' : con_printf(" Which channel do you want to work with? (0..5)");
if(con_scanf("%"SCNx32,&value) != EOF) {
switch (value) {
case 0 : channel_base_address = 0x80;
break;
case 1 : channel_base_address = 0x100;
break;
case 2 : channel_base_address = 0x180;
break;
case 3 : channel_base_address = 0x200;
break;
case 4 : channel_base_address = 0x280;
break;
case 5 : channel_base_address = 0x300;
break;
// make the other cases for each channel
default : break;
}
}
con_printf(" What do you want to do with channel %d?\n",value);
con_printf(" -> 0 - OFF\n") ;
con_printf(" -> 1 - ON\n") ;
con_printf(" -> 2 - Change voltage\n") ;
con_printf(" -> 3 - Change current\n") ;
// Read voltage set
// Read voltage monitor
//.... Write down all the other options
if(con_scanf("%"SCNx32,&value) != EOF) {
C
Tab Width: 8
Ln 1020, Col 9
INS

Problems Original Code

- The original code was not very user friendly the problems include:
- you have to enter individual addresses to access each channel and their functions.
- A user could possibly harm the board by entering the wrong address and entering the wrong values.
- Overall it was hard to learn and was completely unintuitive.

Old Code Flowchart

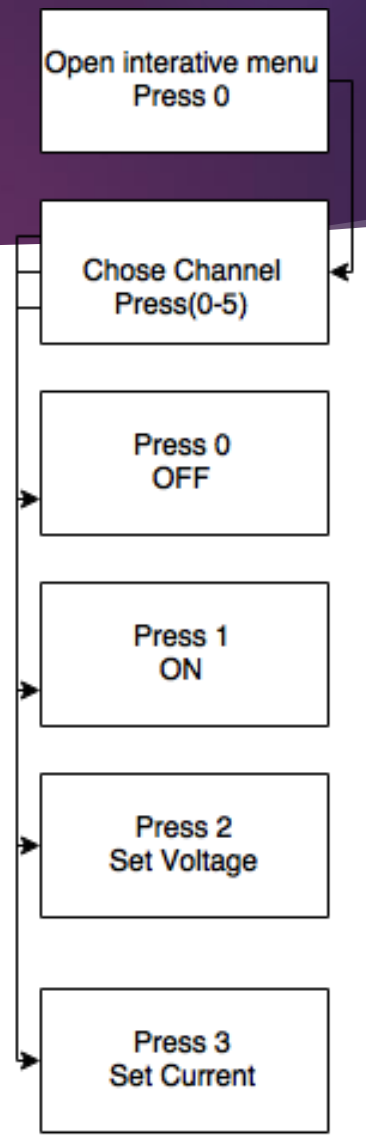


What we set out to do

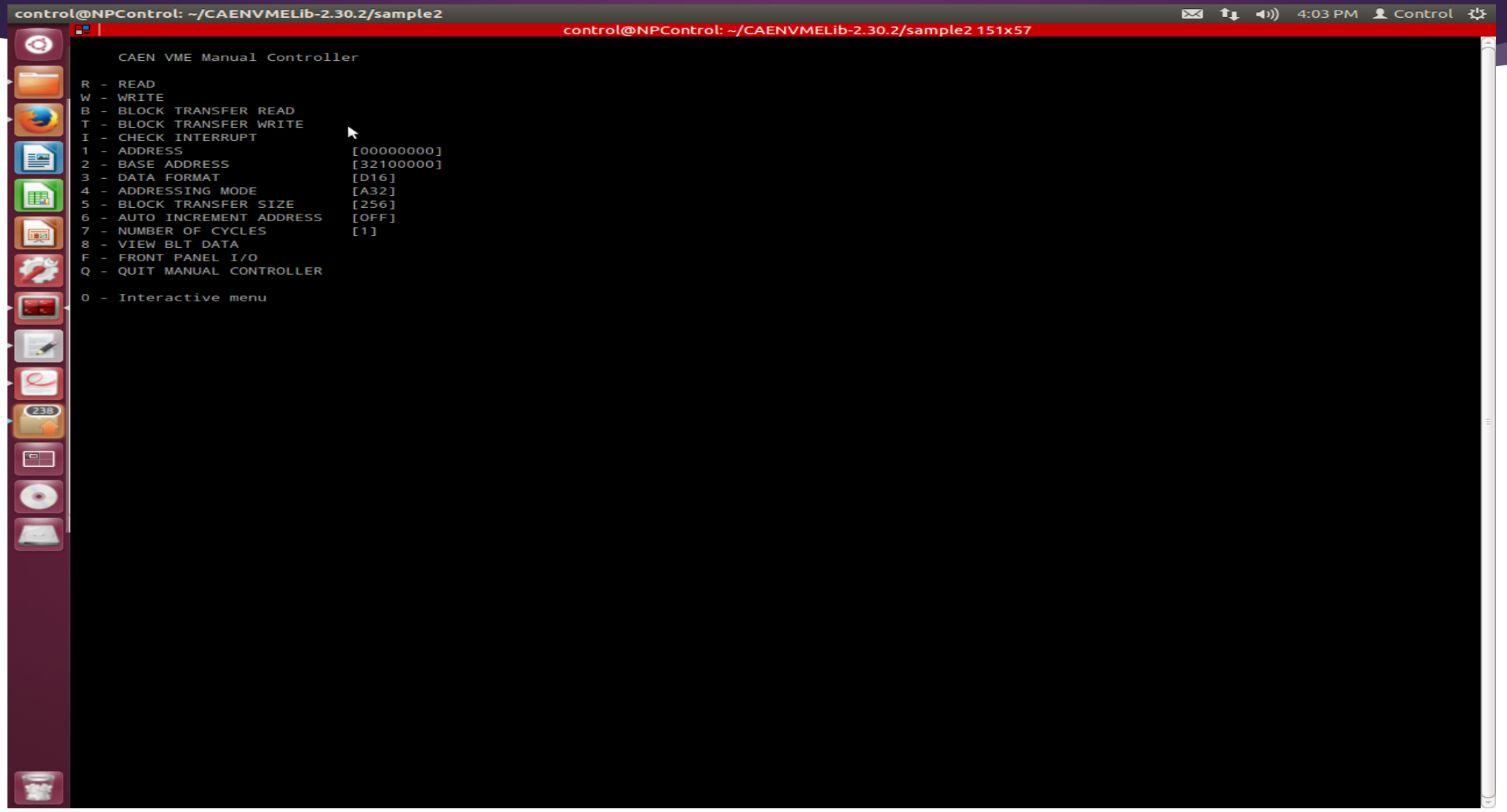
- We wanted to make a user friendly interface where a user could access the HV boards without the need to control the hexadecimal addresses for each action.
- Such as setting the voltage, turning a channel on and off, and reading the voltage.
- To do this we used the original code as a template, but pretty much completely redesign the interface.

Code Flowchart

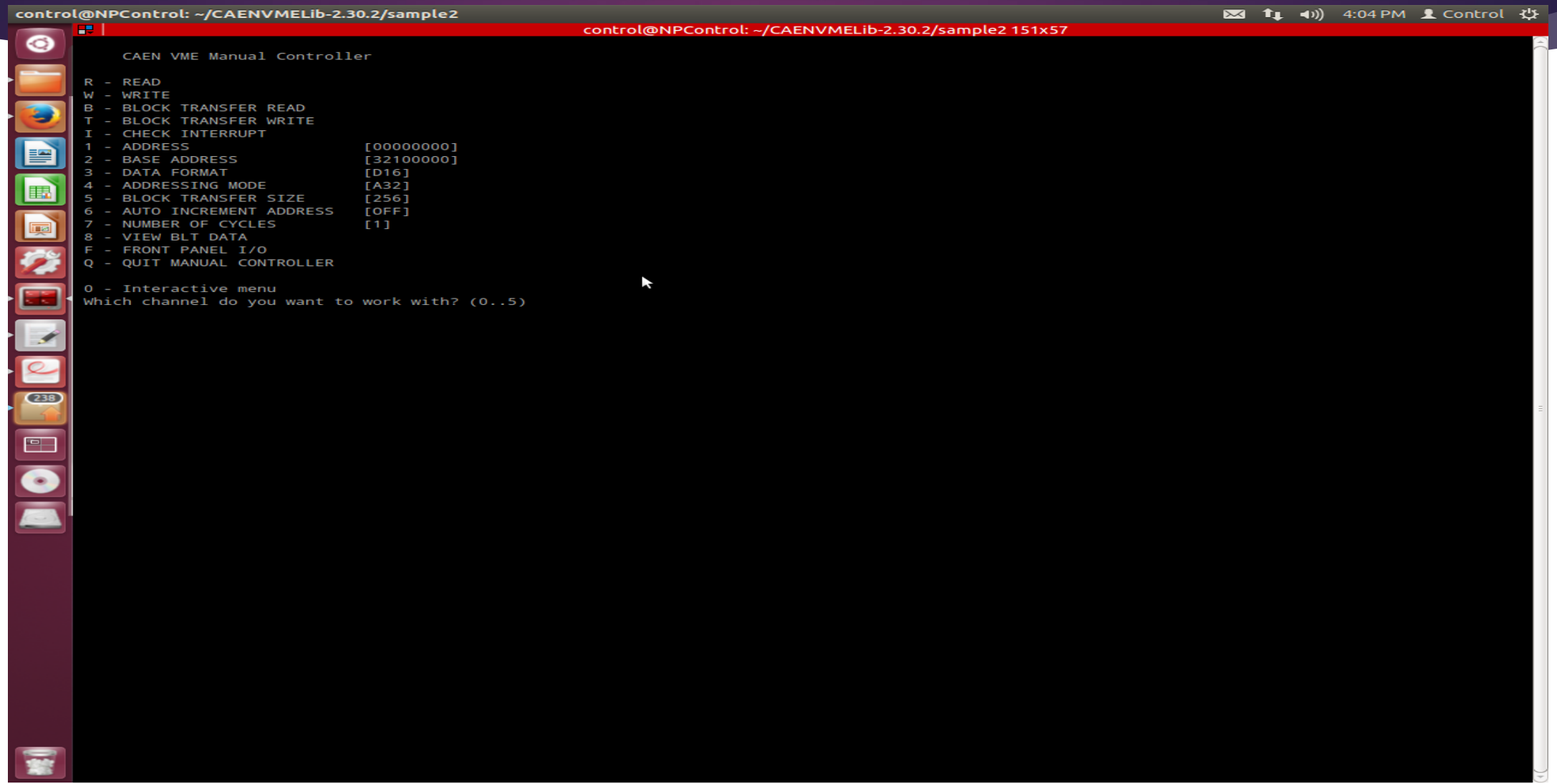
- We created a interactive menu to access it you press zero in the starting screen. Then you get to chose the channel you want to work with by pressing 0-5 representing each channel. Then you select the action press 0-off, 1-on, 2-set voltage, and 3-set current.

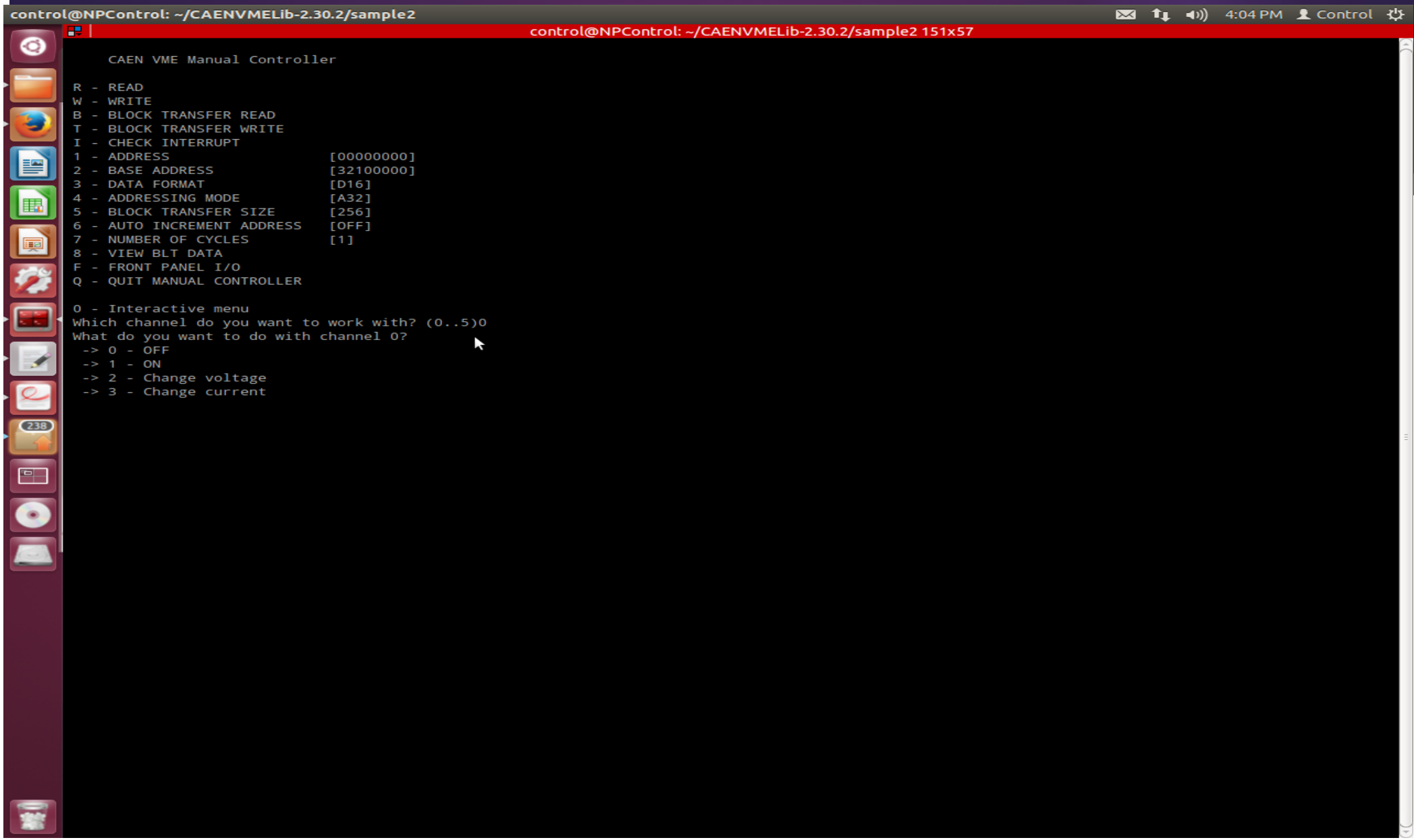


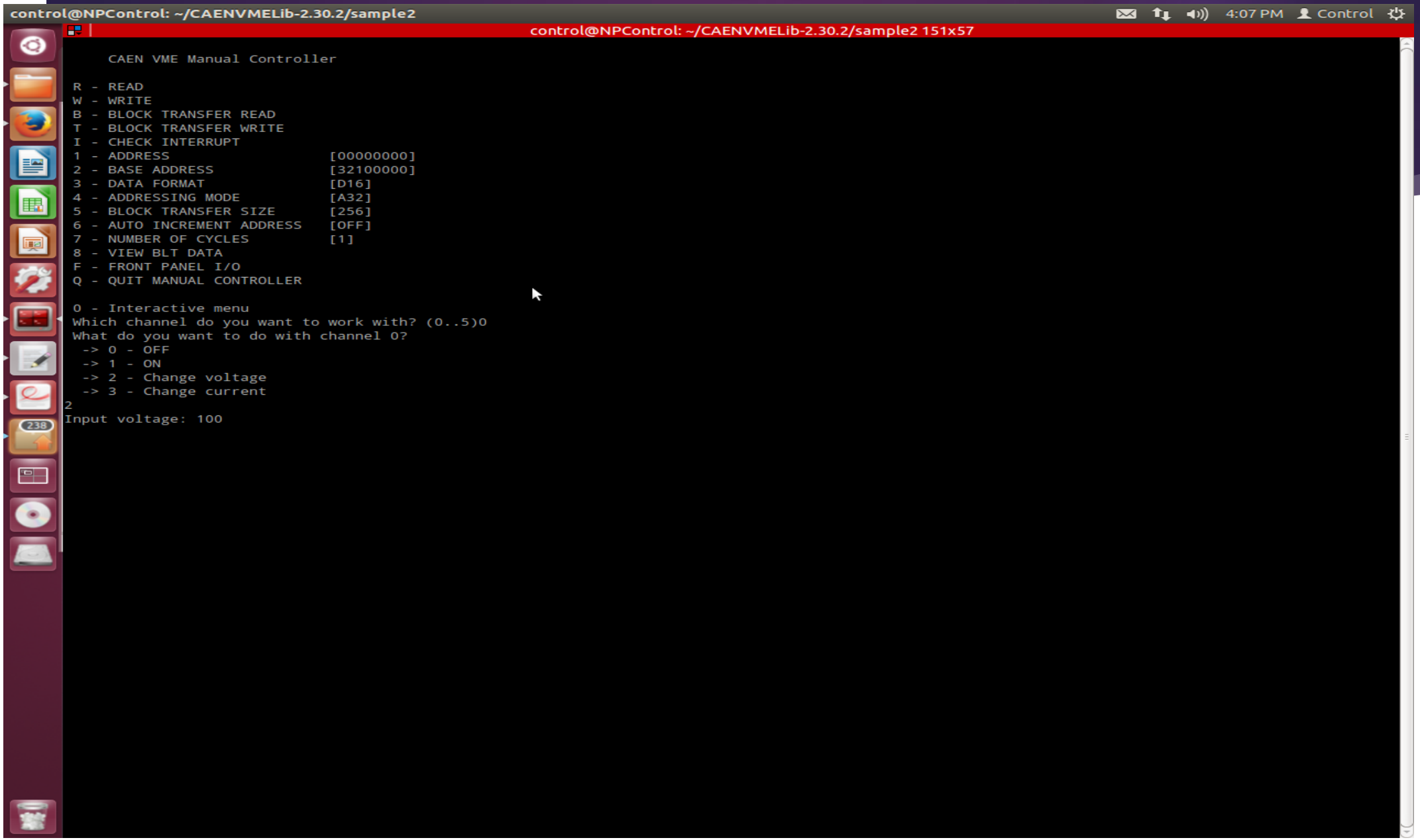
Old interface

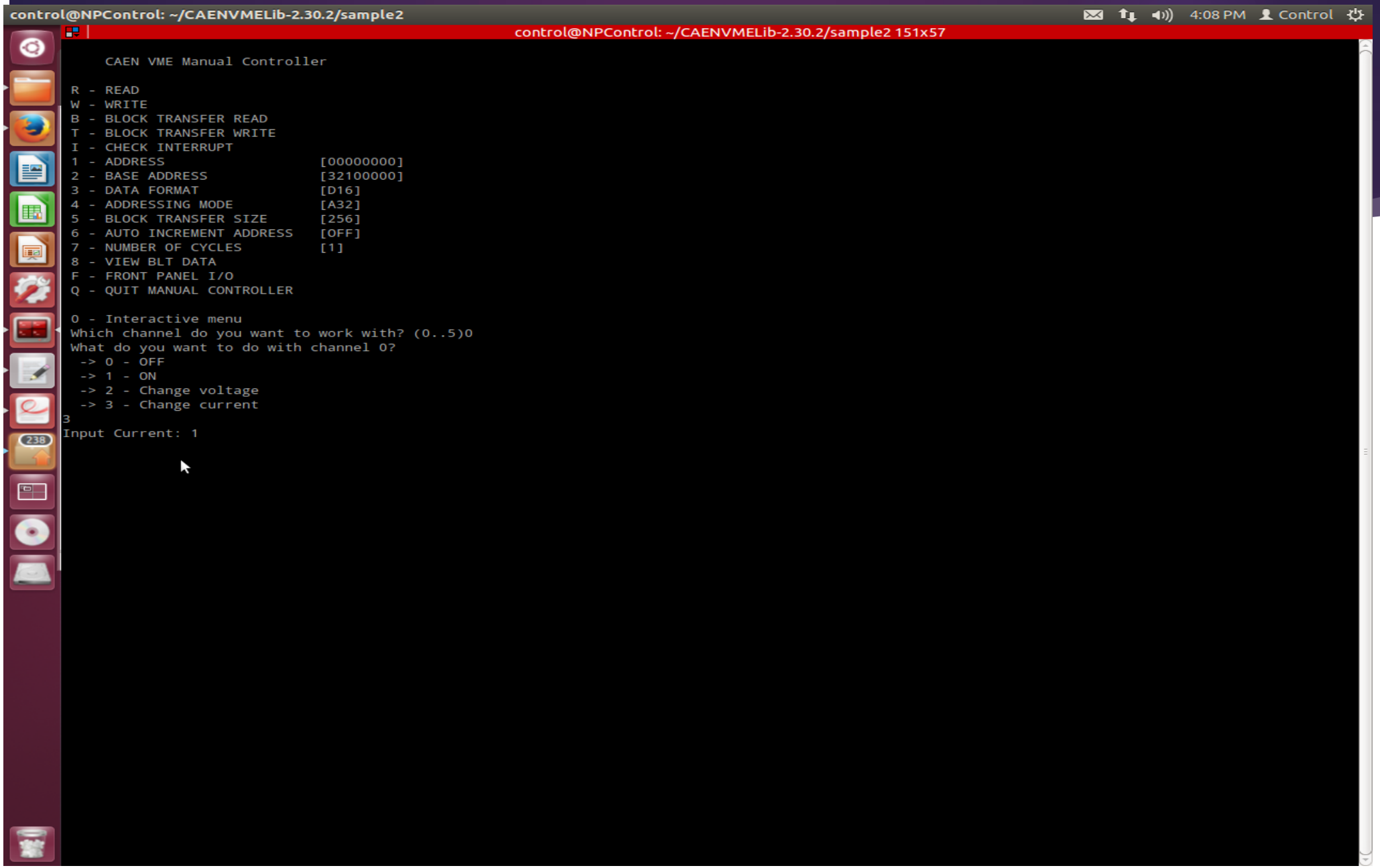


New interface




control@NPControl: ~/CAENVMELib-2.30.2/sample2
4:04 PM
Control
control@NPControl: ~/CAENVMELib-2.30.2/sample2 151x57
CAEN VME Manual Controller
R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS [00000000]
2 - BASE ADDRESS [32100000]
3 - DATA FORMAT [D16]
4 - ADDRESSING MODE [A32]
5 - BLOCK TRANSFER SIZE [256]
6 - AUTO INCREMENT ADDRESS [OFF]
7 - NUMBER OF CYCLES [1]
8 - VIEW BLT DATA
F - FRONT PANEL I/O
Q - QUIT MANUAL CONTROLLER
O - Interactive menu
Which channel do you want to work with? (0..5)0
What do you want to do with channel 0?
-> 0 - OFF
-> 1 - ON
-> 2 - Change voltage
-> 3 - Change current


control@NPControl: ~/CAENVMELib-2.30.2/sample2
4:07 PM
Control
control@NPControl: ~/CAENVMELib-2.30.2/sample2 151x57
CAEN VME Manual Controller
R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS [00000000]
2 - BASE ADDRESS [32100000]
3 - DATA FORMAT [D16]
4 - ADDRESSING MODE [A32]
5 - BLOCK TRANSFER SIZE [256]
6 - AUTO INCREMENT ADDRESS [OFF]
7 - NUMBER OF CYCLES [1]
8 - VIEW BLT DATA
F - FRONT PANEL I/O
Q - QUIT MANUAL CONTROLLER
0 - Interactive menu
Which channel do you want to work with? (0..5)0
What do you want to do with channel 0?
-> 0 - OFF
-> 1 - ON
-> 2 - Change voltage
-> 3 - Change current
2
Input voltage: 100


control@NPControl: ~/CAENVMELib-2.30.2/sample2
4:08 PM
Control
control@NPControl: ~/CAENVMELib-2.30.2/sample2 151x57
CAEN VME Manual Controller
R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS [00000000]
2 - BASE ADDRESS [32100000]
3 - DATA FORMAT [D16]
4 - ADDRESSING MODE [A32]
5 - BLOCK TRANSFER SIZE [256]
6 - AUTO INCREMENT ADDRESS [OFF]
7 - NUMBER OF CYCLES [1]
8 - VIEW BLT DATA
F - FRONT PANEL I/O
Q - QUIT MANUAL CONTROLLER
0 - Interactive menu
Which channel do you want to work with? (0..5)0
What do you want to do with channel 0?
-> 0 - OFF
-> 1 - ON
-> 2 - Change voltage
-> 3 - Change current
3
Input Current: 1

Conclusion

- Using case architecture we were able to set up a interactive menu.
- Second, we changed the write code so you don't have to enter anything all you need to do is press number to turn a channel on or off.
- we entered all the addresses of each channel, so the user doesn't need to enter them.
- We also change the initial base address to 32100000.
- You can now change the voltage and current with the press of a button after the channel is selected.
- The next step is to be able to read the voltage and current values before you enter the new value. Also to make a slew of other actions include in the board, like sensing the temperature and each channels status.