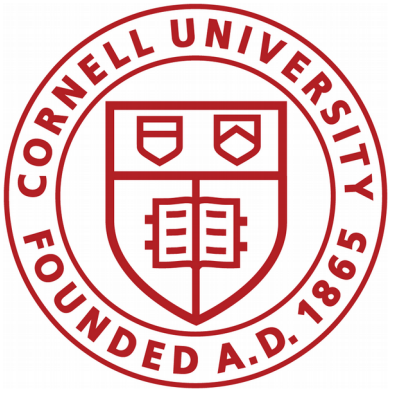


CBPM development

Antoine, Nate

CBPM meeting
December 11, 2019

One of our current goal

We have shown that the spatial resolution can be improved to the micron level if averaging more than a 1,000 turns

Goal: demonstrate with the current CBPM system that we can continuously collect data at 1-10 Hz level to provide a micron resolution

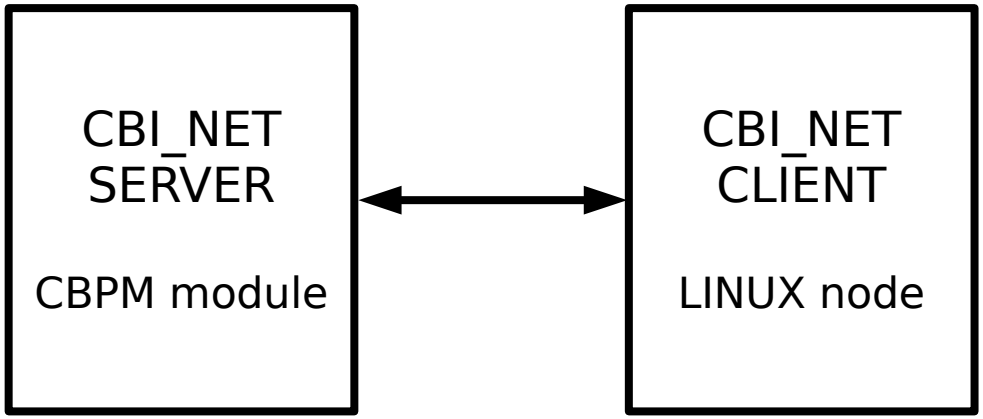
This requires:

- ✗ updating firmware to do processing (averaging) inside the module – Nate
- ✗ write custom software to collect/analyze/display – Antoine, Nate

Let's talk about software written in Python3 – Antoine

CBI_NET

CBI_NET is a custom C library responsible for transfering data and communicating with the CBPM module



The current data acquisition system, CBIC, is a humongous C program that talks to CBI_NET and is quite a mess...

CBI_NET and Python3

Developing a data acquisition software in Python3 to replace CBIC and directly deal with CBI_NET

CBI_NET is compiled as a C shared library and the (debug version) nightly build is available here for instance:

```
/nfs/cesr/online/lib/Linux_x86_64_intel/devel/debug/lib/libcbi_net.so
```

The goal is to call this C library from Python3 to have access to all the functionality

CBI_NET + Python3 example

Simplest possible piece of code to retrieve the FPGA version # from register

```
from ctypes import create_string_buffer, cdll

my_func = cdll.LoadLibrary('/nfs/cesr/online/lib/Linux_x86_64_intel/devel/debug/lib/libcbi_net.so')

Socket = my_func.cbi_net_fdopen(b'192.168.32.191')

address = 150994944 + 2
num_read = 1
data_type = 4

p = create_string_buffer(8)

num_words = fe_fpga_ver = my_func.cbi_net_rd_mem(Socket, address, num_read, data_type, p)

print('FPGA version: ', ord(p[0]))

my_func.cbi_net_close_socket(Socket)
```

Let's run this code:

```
[atc93@lnx6248 cbi_net_python]$
[atc93@lnx6248 cbi_net_python]$ python cbi_net.py
FPGA version:  47
[atc93@lnx6248 cbi_net_python]$
```

Python3 + ctypes → C calls

We are using ctypes to call C from within Python:

“ctypes is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python.”

The biggest time sink in writing Python/C compatible code is to provide the C library with the proper data type. Especially Python and C deal differently with pointers and arrays. Quite a pain...

But the code from the previous page is everything. There is no need for complicated MEX function that MATLAB requires. It looks a lot more straightforward to call CBI_NET in Python.

Test of the data throughput

Using an expanded version of the code I showed previously, I tested how fast we can read the FPGA version # from the module. The code flow is the following:

- ✗ open socket
- ✗ start timer
- ✗ read the FPGA information a 1,000 times
- ✗ stop timer
- ✗ print read rate
- ✗ close socket

The measured rate is about 300 Hz. This is only reading one register.

Next step: try reading out data from module issuing trigger and whatnot and measure the data throughput

MPM_NET + Python3

MPM_NET is an other C shared library. This one act as the data-base and is required alongside CBI_NET.

Currently working on making simple code to work. Fierce battle to create the proper data type in Python3 that works for the C library...

Additional materials