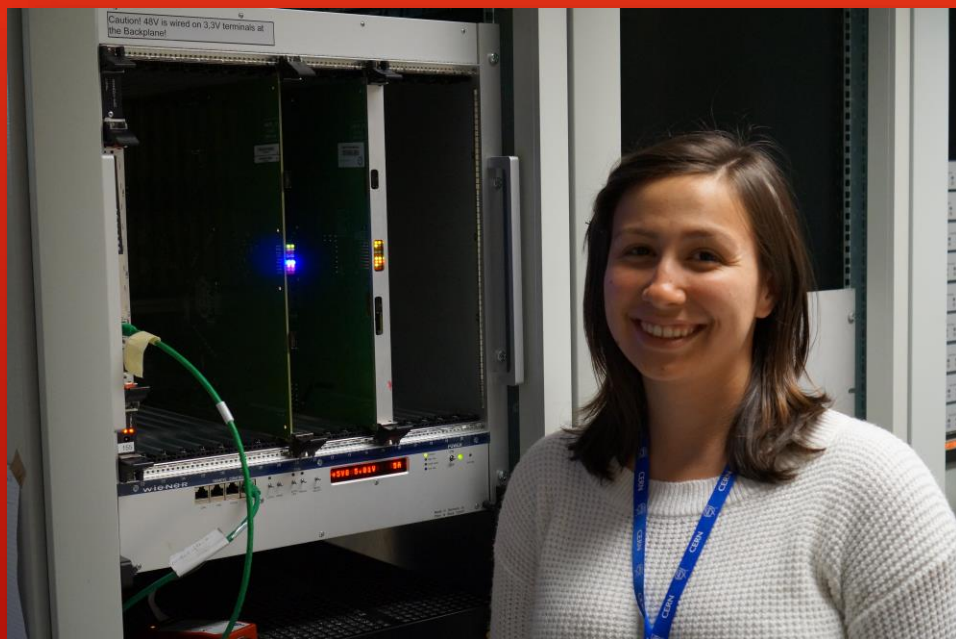


WORKING ON THE FTK: MAKE ATLAS GREAT AGAIN

Haley Marez

Mentor: Guido Volpi

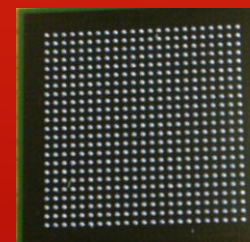
University of Pisa



THE BOARD

The board(s) that I'm going to be working with this summer are currently located in Lab 4.

These boards are AMBs or Associative Memory Boards which are responsible for processing hits and track fitting the data that comes from the ATLAS detectors.



```
f7800000 1  #!/hmarez/public/python
ff0803cf 2  #Processes, sorts, and plots output data from AMBoard
ff2803cf 3  import numpy as np
ff4803cf 4  import matplotlib.pyplot as plt
ff6803cf 5  import sys
f7800001 6  import string as str
ff1c026c 7  import collections as col
ff1e0137 8  #isolates the bb from the chip bytes
ff3c026c 9  def remove_bb(s):
ff3e0137 10  return s[2:]
ff5c026c 11  def remove_chip(s):
ff5e0137 12  return s[:-6]
ff7c026c 13  #select the file that you would like to view
ff7e0137 14  if (len(sys.argv)!=2):
f7800002 15  sys.exit('Please list the file you would like to use in the command-line
ff02051e 16  else:
ff0c0152 17  filename = sys.argv[1]
ff1a0358 18  #open the selected file and read in data
ff1e073b 19  with open(filename) as f:
ff22051e 20  mylist = [n for n in f.read().splitlines() if not n.startswith('f78')]
ff2c0152 21  for line in mylist:
ff3a0358 22  bbdata = []
ff3e073b 23  results = []
ff42051e 24  chiplist = [remove_bb(s) for s in mylist]
ff4c0152 25  for line in chiplist:
ff5a0358 26  chipad = int(line, 16)//0x20000
ff5e073b 27  results.append(chipad)
ff62051e 28  bblast = [remove_chip(s) for s in mylist]
ff6c0152 29  for line in bblast:
ff7a0358 30  bbint = int(line, 16)
ff7e073b 31  bbdata.append(line)
f7800003
ff0200f5
ff0a0489
```

PROGRAMMING: PYTHON

The way that data about each chip in the board outputs is in a byte based format using hexadecimal based variables.

I wrote a program that could break this output apart and analyze where the data is coming from to test the state of the board.

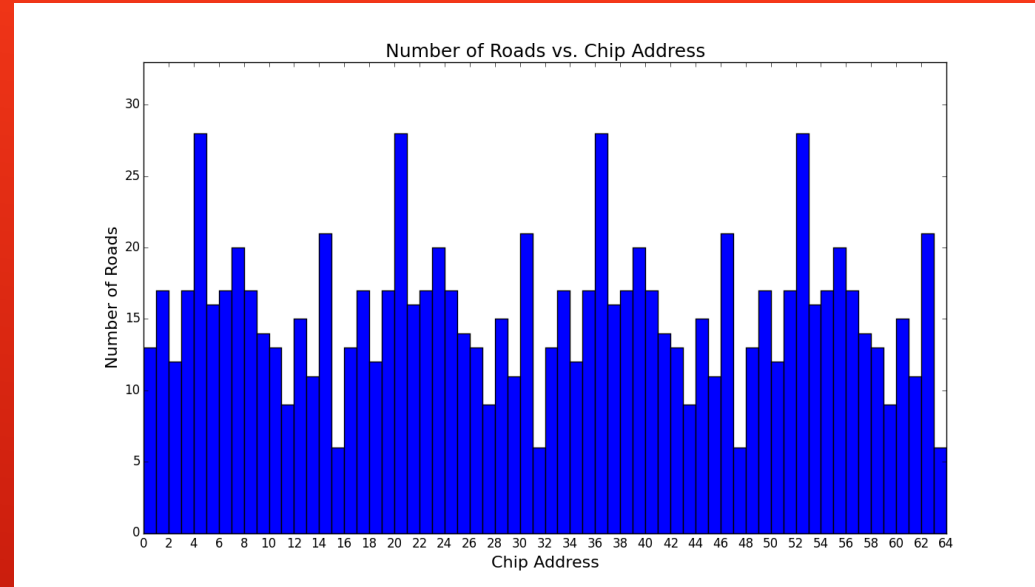
Command Prompt

```
Command Prompt
outroads_sim.out
Counter({28: 7, 53: 7, 5: 6, 8: 6, 14: 6, 37: 6, 20: 5, 41: 5, 4: 4, 6: 4, 18: 4,
, 45: 4, 52: 4, 63: 4, 32: 3, 57: 3, 0: 2, 1: 2, 2: 2, 3: 2, 7: 2, 9: 2, 10: 2,
11: 2, 12: 2, 13: 2, 15: 2, 16: 2, 17: 2, 19: 2, 21: 2, 22: 2, 23: 2, 24: 2, 25:
2, 26: 2, 27: 2, 29: 2, 30: 2, 31: 2, 33: 2, 34: 2, 35: 2, 36: 2, 38: 2, 39: 2,
40: 2, 42: 2, 43: 2, 44: 2, 46: 2, 47: 2, 48: 2, 49: 2, 50: 2, 51: 2, 54: 2, 55
: 2, 56: 2, 58: 2, 59: 2, 60: 2, 61: 2, 62: 2})
Counter({'ff': 174})
Total number of roads: 174
Total time: 1.740000000000001e-06 s | 17.40000000000001 % of max time

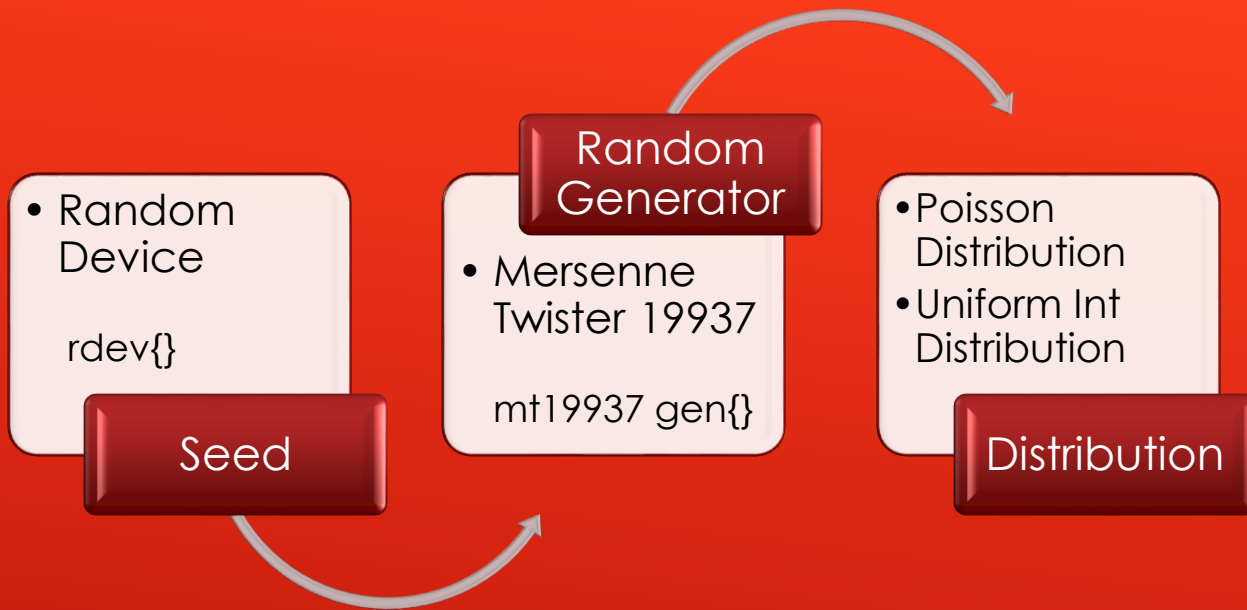
C:\Users\Haley\Documents\FTK Papers & Background Research>python readfile.py 1ou
troads_sim.out
Counter({4: 28, 20: 28, 36: 28, 52: 28, 14: 21, 30: 21, 46: 21, 62: 21, 7: 20, 2
3: 20, 39: 20, 55: 20, 1: 17, 3: 17, 6: 17, 8: 17, 17: 17, 19: 17, 22: 17, 24: 1
7, 33: 17, 35: 17, 38: 17, 40: 17, 49: 17, 51: 17, 54: 17, 56: 17, 5: 16, 21: 16
, 37: 16, 53: 16, 12: 15, 28: 15, 44: 15, 60: 15, 9: 14, 25: 14, 41: 14, 57: 14,
0: 13, 10: 13, 16: 13, 26: 13, 32: 13, 42: 13, 48: 13, 58: 13, 2: 12, 18: 12, 3
4: 12, 50: 12, 13: 11, 29: 11, 45: 11, 61: 11, 11: 9, 27: 9, 43: 9, 59: 9, 15: 6
, 31: 6, 47: 6, 63: 6})
Counter({'ff': 984})
Total number of roads: 984
Total time: 9.840000000000002e-06 s | 98.4000000000003 % of max time

C:\Users\Haley\Documents\FTK Papers & Background Research>
```

Graphing: Matplotlib



- Each road is counted and listed based on its source path in the board as well as how many panels had hits.
- The program will output a graph that displays how many roads are being produced by each chip on the board.



```
#include <random>
int main()
{
    using namespace std;
    random_device rdev{};
    mt19937 gen{};

    int seed = rdev();

    int pattmin, pattmax;
    pattmin = 100;
    pattmax = 20000;

    poisson_distribution<int> distribution(7);
    int nroads_int = distribution(gen);

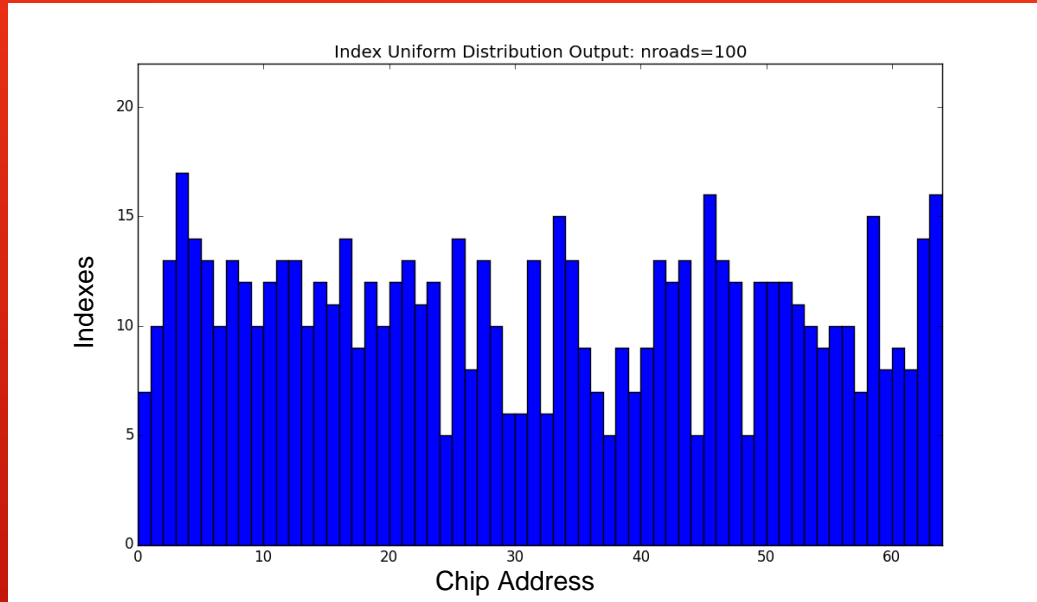
    uniform_int_distribution<int> dist(pattmin,pattmax);
    for (int i=0;i<nroads_int;i++){
        int index = dist(gen);
    }
    return 0;
}
```

PROGRAMMING:C++11

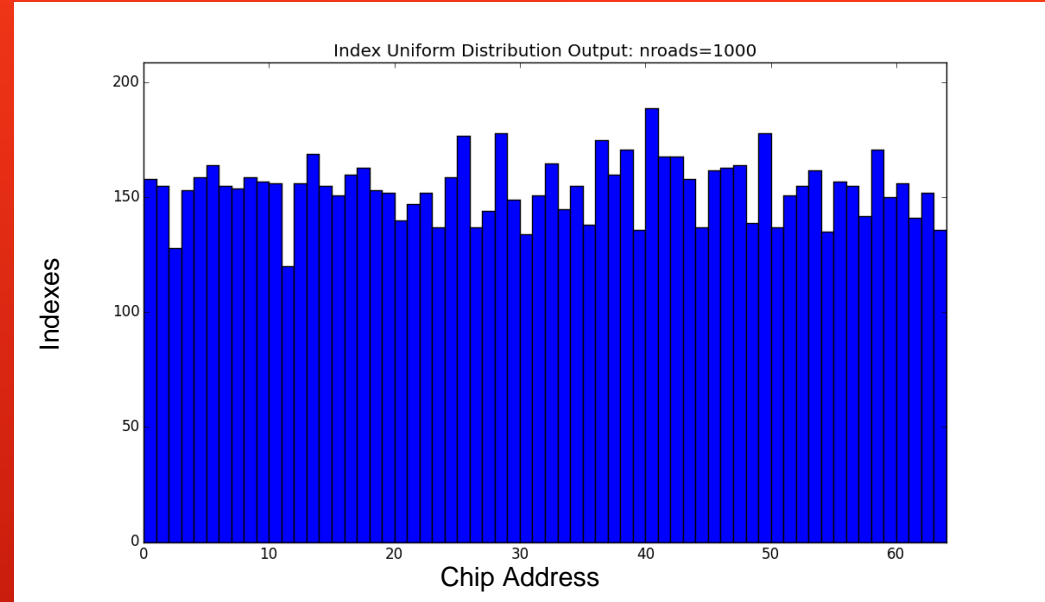
I have been updating the random generating systems used within the mainboard C++ code that is used to produce input files to test the board.

Now that ATLAS uses C++11 I rewrote some of the code to utilize the <random> library to update the seed, random generator (Mersenne Twister) and introduce new Poisson and uniform distributions.

Low Number of Roads



High Number of Roads



LOOKING AT THE UNIFORM DISTRIBUTION