How to Run & Use EicRoot *and* Machining Parts Using PEEK

Fall 2018 Report

Matthew Bomberger

December 25, 2018

Contents

General Tip for Reading: A backslash (\) in command line entries connects the line with the backslash to the next line with a space.

1 How to Run & Use EicRoot

1.1 Introduction to EicRoot

EicRoot is a powerful simulation framework based on PandaRoot for emulating and testing detector assemblies. The goal of this software is to provide a simple way to simulate prototype detector designs for a future electron-ion collider (EIC) in the USA. Two methods of running EicRoot are presented in this document. First, the simplest way is addressed, i.e. running EicRoot in a Docker container. After that, a harder approach is described. This method involves installing and setting up the software necessary to run EicRoot on an Ubuntu partition. Finally, a description of functions used in EicRoot C++ scripts is provided.

The motivation behind EicRoot is straight forward. Simply insert multiple detector geometries into common simulation, digitization, reconstruction, display, and analysis scripts and one can see the effect on the whole assembly by modifying an arbitrary detector or detectors. In this sense, EicRoot is very modular. All one needs to do is write or modify C++ scripts to include desired geometries, resolutions, etc.

Since EicRoot is based on PandaRoot, which is itself a version of FairRoot, large directories are necessary for storing configuration files, scripts, and input files. Either one has to have free space on their machine, or one can use a Docker container to run the software. The easiest approach, using a Docker container, is addressed first and then the harder method is presented.

1.2 Running EicRoot Using Docker

Before providing the instructions for running EicRoot in a Docker container, the concept of Docker should be given first. Docker is a software that is used to make and run containers on one's local computer. It is similar to running a virtual machine in that one networks to another computer, but instead of mapping directly to the computer, one maps to an *image* of that machine. This image does not store perminant changes in the computer being mapped to. In other words, once one disconnects from the container, the changes one makes go away.

Since EicRoot works best on Linux or Mac, instructions for installing Docker and running EicRoot using it are given below. The Linux distro used in this example is CentOS 7. The container runs on CentOS, so it makes sense to use a compatible distro.

1.2.1 CentOS 7

The instructions provided for installing Docker on CentOS 7 are based on what is in the Docker manual [1]. First, let's assume that Docker is not installed. Set up the repository to install Docker using the following commands:

```
$ sudo yum install -y yum-utils \
  device-mapper-persistent-data \
  lvm2
$ sudo yum-config-manager \
  --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo
```

Next, install the latest version of Docker CE.

```
$ sudo yum install docker-ce
```

Start Docker, and run a sample image called "hello-world."

```
$ sudo systemctl start docker
$ sudo docker run hello-world
```

Welcome to the Docker world! The next instructions are for running EicRoot.

First, pull and run the EicRoot docker container in a terminal window.

```
$ sudo docker pull ayk1964/eicroot:r940
$ sudo docker run -i -p 127.0.0.1:7777:22 \
  -t ayk1964/eicroot:r940
```

In a new terminal window, run the following commands to connect to the container and set up the environment for EicRoot simulations of 10 rectangular silicon trackers with a pion beam. The password is "test."

```
$ ssh -Y eic@localhost -p 7777
$ . eicroot/build/config.sh
$ cd eicroot/examples/tracking/config.1
```

Next, run the following scripts in this order: tracker.C $\rightarrow$ simulation.C $\rightarrow$ digitization.C $\rightarrow$ reconstruction.C $\rightarrow$ analysis.C. The command line entries are as follows with optional commands in brackets:

```
$ root -l tracker.C
$ root -l simulation.C
$ root -l digitization.C
$ root -l reconstruction.C
$ root -l ../analysis.C
$ root -l eventDisplay.C
```

This concludes the instructions for getting EicRoot working in a Docker container on CentOS . An interesting study is to change detector parameters such as spacing between the silicon wafers, resolution of the wafers, or wafer size or to change the particle beam type, momentum, or angle. After making these changes, one can notice how these parameters effect the momentum resolution of the detectors being simulated. Other directories in the "tracking" directory provide a variety of geometries to work with, from using GEM chambers to implementing calorimeters.

1.2.2 macOS

As with the Ubuntu instructions, the procedure for installing Docker on a Mac is based on the Docker manual [2]. Before installing Docker, make sure that the Mac is at least a 2010 model. Enter the following command in a terminal window for checking for Intel hardware support:

```
$ sysctl kern.hv_support
```

Other requirements: at least 4 GB of RAM, VirtualBox version **no older** than 4.3.30, and preferably the newest macOS.

To install, first visit https://store.docker.com/editions/community/docker-ce-desktop-mac and login (or create an account) to download Docker for Mac. Once that is done, double-click the "Docker.dmg" file. This will start the installer GUI, and all one has to do is drag the whale icon to the Applications folder. Double-click the "Docker.app" icon in Applications to start Docker. Authorize the application via password confirmation and Docker is ready to go. A whale icon should appear in the status bar on the top of one's screen. This indicates that Docker is working and ready to be used in a terminal window. Test it by running the image called "hello-world."

```
$ sudo docker run hello-world
```

Welcome to the Docker world!

Open a terminal window and write the following command to allow graphics to be forwarded:

```
$ defaults write org.macosforge.xquartz.X11 \
  enable_iglx -bool true
```

Next, pull and run the docker container:

```
$ docker pull ayk1964/eicroot:r940
$ docker run -i -p 127.0.0.1:7777:22 \
 -t ayk1964/eicroot:r940
```

In a separate terminal window, run the following sequence of commands to "ssh" to the container (the password is "test") and build the EicRoot environment:

```
$ ssh -Y eic@localhost -p 7777
$ . eicroot/build/config.sh
$ cd eicroot/examples/tracking/config.1
```

Next, run the following scripts in the following order: tracker.C → simulation.C → digitization.C → reconstruction.C → analysis.C. An optional script to execute is "eventDisplay.C," which must be run after the "reconstruction.C" script. This script opens an interactive GUI display of the trackers, hits, and tracks. The command line entries are as follows with optional commands in brackets:

```
$ root -l tracker.C
$ root -l simulation.C
$ root -l digitization.C
$ root -l reconstruction.C
$ root -l ../analysis.C
[$ root -l eventDisplay.C]
```

This concludes the instructions for getting EicRoot working in a Docker container on macOS. An interesting study is to change detector parameters such as spacing between the silicon wafers, resolution of the wafers, or wafer size or to change the particle beam type, momentum, or angle. After making these changes, one can notice how these parameters effect the momentum resolution of the detectors being simulated. Other directories in the "tracking" directory provide a variety of geometries to work with, from using GEM chambers to implementing calorimeters.

1.2.3 Mapping Directories to the Docker Container

If one wishes to make a persistent directory for running simulations in EicRoot, mapping to the container is the logical thing to do. The following lines show how one would pursue this. Leave out sudo if working on a Mac.

```
$ mkdir /home/[username]/[directory name]
$ sudo docker run -i -p 127.0.0.1:7777:22 \
  -t -v /home/[username]/[directory]:/mnt \
  ayk1964/eicroot:r940
```

Then one simply ssh's to the container as before.

To access this directory, one needs to type the following.

```
$ cd /mnt
```

A test of the connection of the directory is to make a file on the local machine and notice if it extends to the container.

1.3 Running EicRoot on a CentOS 7 Machine

A more ambitious method for getting EicRoot to work is to install the image locally on one's machine. The following instructions are for installation on CentOS 7, but similar methods can be extended to macOS if so desired. Note: to install EicRoot in this manner, one needs to apply for BNL RACF login credentials. Visit `https://www.racf.bnl.gov/experiments/rhic/useraccounts` for more information.

```
$ sudo git clone https://git.racf.bnl.gov/gitea/EIC/EicRoot
```

A prompt for username and password will pop up, enter the appropriate information. After a minute, the files should be installed.

Run the following command to install the appropriate packages for a 64-bit system [3]:

```
$ sudo apt-get install  cmake cmake-data g++ \
  gcc gfortran debianutils build-essential make \
  patch sed libx11-dev libxft-dev libxext-dev \
  libxpm-dev libxmu-dev libglu1-mesa-dev \
  libgl1-mesa-dev libncurses5-dev curl \
  libcurl4-openssl-dev bzip2 libbz2-dev gzip \
  unzip tar subversion git xutils-dev flex bison \
  lsb-release python-dev libc6-dev-i386 \
  libxml2-dev wget libssl-dev libkrb5-dev \
  automake autoconf libtool \
```

This long command installs various tools that are needed for installing FairRoot.

Next, FairRoot needs to be installed on the local machine. The following commands will do this:

```
$ mkdir /home/[username]/fairsoft
$ git clone https://github.com/FairRootGroup/FairSoft \
  /home/[username]/fairsoft/may16p1
$ git checkout -b /home/[username]/fairsoft/may16p1 \
  /home/[username]/fairsoft/may16p1
```

Then, run the following commands to finish setting up the software:

```
$ . /home/[username]/fairsoft/may16p1/configure.sh \
$ export SIMPATH = /home/[username]/fairsoft/may16p1-build
$ export ROOTSYS = /home/[username]/fairsoft/may16p1/tools/root
$ cd EicRoot && mkdir build && cd build
$ cmake -DCMAKE_INSTALL_PREFIX=. -Wno-dev -DROOT=yes ..
$ make -j4 install
```

As a tip, if there are any errors along the way in this process, consult the configuration files and comment out the appropriate lines. Another suggestion is to look up forums with related issues.

Once installation is complete, one can execute the following commands to run a simple simulation:

```
$ . /home/[username]/eic/eicroot/build/config.sh
$ cd /home/[username]/eic/eicroot/examples/tracking/config.1
$ root -l tracker.C
$ root -l simulation.C
$ root -l digitization.C
$ root -l reconstruction.C
$ root -l ../analysis.C
$ root -l eventDisplay.C
```

If installation was successful, all of these commands should not produce any errors. Figure 1 is an example of possible geometry configurations being implemented. Visit ROOT forums for any questions regarding errors that come up.

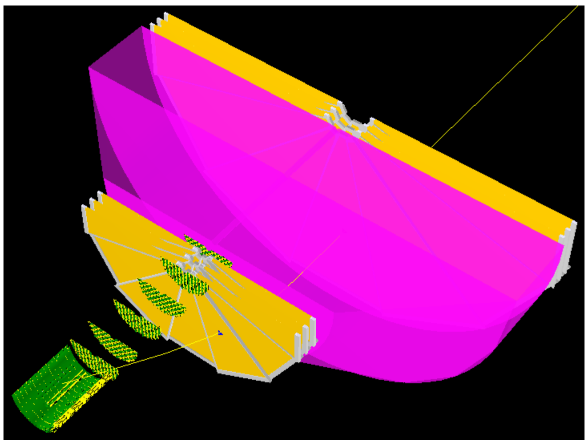

Figure 1: Geometry output from “eventDisplay.C” script showing vertex silicon tracker, silicon trackers, and two sets of three triple gas electron multiplier (GEM) rings sandwiching a ring imaging Cherenkov (RICH) detector

1.4 Scripts

This section is a presentation of a few scripts developed and implemented in the author's studies using EicRoot. Various lines are described for the script describing GEM modules and the script describing the RICH chamber is presented in full. This should give an example of what is necessary for geometry scripts.

1.4.1 GEM Modules

The following section outlines the changes made to the "fbgt.C" script under the "geometry/GEM" directory. Note that the pointer to "GemModule" is changed to "fit." First, under the foil parameters, the Kapton, copper, and G10 thickness values were taken to be 50 μm, 5 μm, and 0 μm, respectively. These lines look as such:

```
fit->mDriftFoilKaptonThickness  =  0.050; // 50 um
fit->mDriftFoilCopperThickness  =  0.005; // 5 um

fit->mGemFoilAreaFraction       =  0.80;
fit->mGemFoilKaptonThickness    =  0.050;
fit->mGemFoilCopperThickness    =  0.005; // GEM Cu layer

fit->mReadoutG10Thickness            =  0.0;
fit->mReadoutFoilKaptonThickness     =  0.050;
fit->mReadoutFoilCopperThickness     =  0.005;
```

One study performed using this type of chamber was to modify the material such that the GEM copper layer is changed to 200 nm of chromium. A realistic application of this modification is to have the first two GEM foils be Cr-GEMs and the last one remain copper. This would reduce the likelihood of burning the readout foil. The following calculation was used to find the equivalent thickness in copper. Each x_0 value is the radiation length of the labeled material, and X is the thickness of the associated material.

$$\frac{X_{equiv}}{x_{0,Cu}} = \frac{2X_{Cr}\,/\,x_{0,Cr} + X_{Cu}\,/\,x_{0,Cu}}{3}$$

$$\begin{aligned} X_{equiv} &= \frac{x_{0,Cu}\,(2X_{Cr}\,/\,x_{0,Cr} + X_{Cu}\,/\,x_{0,Cu})}{3} \\ &= \frac{(1.436\,cm)\,(2(2\cdot 10^{-5}\,cm)\,/\,(2.081\,cm) + (5\cdot 10^{-4}\,cm)\,/\,(1.436\,cm))}{3} \\ &= 1.76\,\mu m \end{aligned}$$

Using this result, the line specifying the thickness of copper is modified as follows.

```
fit->mGemFoilCopperThickness  =  0.00176; // GEM Cr layer
```

Another study with these GEM chambers was to make another set of chambers further in the forward region, right past the RICH chamber (see Fig. 1 for reference). The following code was used to place the first three rings at 1030, 1100, and 1170 mm, and the next three 1500 mm further forward. Note that the second triplet of GEMs are larger.

```
double centerChamDist  = 370. + 50;
double distFirstWheel  = 1030.0;
double distSecWheel    = 1100.0;
double distThiWheel    = 1170.0;

...

fbgt->AddWheel(fit, 12, centerChamDist, distFirstWheel);
fbgt->AddWheel(fit, 12, centerChamDist, distSecWheel);
fbgt->AddWheel(fit, 12, centerChamDist, distThiWheel);

...

distFirstWheel += 1500.0;
distSecWheel += 1500.0;
distThiWheel += 1500.0;

//Proposed GEMs in far-forward region;
fit->mActiveWindowHeight          *= 2;
fit->mActiveWindowTopWidth        *= 2;
fit->mActiveWindowBottomWidth     *= 2;
```

The lines with the "fbgt" and "fit" pointers are enclosed in a loop to make the far-forward GEM rings.

1.4.2 RICH Chamber

The following script was written from scratch to describe a RICH geometry and material. The only portion missing is the mirrors integral to the chamber. The name of the file is "rich.C" and outputs "rich.root" to be used in the "simulation.C" script.

```
// Purpose of script is to emulate a simple
// RICH configuration with fake material for
// approximation of BeAST config.

void rich()
{
  // Load basic libraries;
  gROOT->Macro(``$VMCWORKDIR/gconfig/rootlogon.C");

  EicGeoParData *rich = new EicGeoParData(``RICH", \
  0, 0);
  rich->SetFileName(``rich.root");

  Double_t waferNum = 1.0;
  Double_t rMin             =   850.0;
  Double_t rMax             =  1600.0;
  Double_t wndThickness     =     0.2;
  Double_t wndLocation      =  1265.0;
  Double_t cf4Thickness     =  1200.0;
  //Double_t cf4Location       =  wndLocation + \
  0.5*cf4Thickness;//1605.0;

  // RICH entrance window;
  TGeoTube *wnden = new TGeoTube(``richWndEn",
                             0.1 * 20.0,
                             0.1 * rMin,
                             0.1 * wndThickness/2);
  TGeoVolume *vwnden = new TGeoVolume(``richWndEn", \
  wnden, rich->GetMedium(``carbon")); // Fake material;

  // Make entrance window sensitive;
  EicGeoMap *richmap = rich->CreateNewMap();
  richmap->AddGeantVolumeLevel(``richWndEn", waferNum);
  richmap->SetSingleSensorContainerVolume(``richWndEn");

  rich->AddLogicalVolumeGroup(0, 0, waferNum);

  for(unsigned wf=0; wf<waferNum; wf++)
    {
      double offset = 0.1 * (wndLocation + wf);

      UInt_t geant[1] = {wf}, group = 0, logical[3] = \
      {0, 0, wf};

      if(rich->SetMappingTableEntry(richmap, geant, \
      group, logical)){
        cout << ``Failed to set mapping table entry!" << endl;
        exit(0);
      }//if

      rich->GetTopVolume()->AddNode(vwnden, wf, \
      new TGeoCombiTrans(0.0, 0.0, offset, \
      new TGeoRotation()));
    }// for wf


  // RICH gas volume, a cone followed by a cylinder;
  TGeoCone *cf41 = new TGeoCone(``richVolume1",
                               0.1 * 685/2,
                               0.1 * 20.0,
                               0.1 * rMin,
                               0.1 * 20.0,
                               0.1 * rMax);
  TGeoVolume *vcf41 = new TGeoVolume(``richVolume1", \
  cf41, rich->GetMedium(``CF4"));
  rich->GetTopVolume()->AddNode(vcf41, 0, \
  new TGeoCombiTrans(0.0, 0.0, 0.1 * 1605, \
  new TGeoRotation()));

  TGeoTube *cf42 = new TGeoTube(``richVolume2",
                                0.1 * 20.0,
                                0.1 * rMax,
                                0.1 * 515/2);
  TGeoVolume *vcf42 = new TGeoVolume(``richVolume2", \
  cf42, rich->GetMedium(``CF4"));
  rich->GetTopVolume()->AddNode(vcf42, 0, \
  new TGeoCombiTrans(0.0, 0.0, 0.1 * 2205, \
  new TGeoRotation()));

  // RICH exit window;
  TGeoTube *wndex = new TGeoTube(``richWndEx",
                                 0.1 * 10.0,
                                 0.1 * rMax,
                                        0.1 * wndThickness/2);
  TGeoVolume *vwndex = new TGeoVolume(``richWndEx", \
  wndex, rich->GetMedium(``carbon"));
  rich->GetTopVolume()->AddNode(vwndex, 0, \
  new TGeoCombiTrans(0.0, 0.0, 0.1*(wndLocation + \
  wndThickness + cf4Thickness), new TGeoRotation()));

  rich->GetColorTable()->AddPatternMatch(``richWndEn", kMagenta);
  rich->GetTransparencyTable()->AddPatternMatch("richWndEn", 50);

  rich->GetColorTable()->AddPatternMatch(``richVolume", kMagenta);
  rich->GetTransparencyTable()->AddPatternMatch("richVolume", 50);

  rich->GetColorTable()->AddPatternMatch(``richWndEx", kMagenta);
  rich->GetTransparencyTable()->AddPatternMatch("richWndEx", 50);

  rich->FinalizeOutput();
  exit(0);
}//rich()
```

1.5 Problems Using EicRoot

EicRoot can be finnicky, as one can imagine. Its promise of high modularity results in a system that's rather complicated to understand. For instance, the instructions for installing EicRoot on Linux don't work as is. It depends on earlier versions of dependencies, and for this reason errors corresponding to packages not being loaded, such as EICSMEAR and CLHEP, impede the ability for this software to be loaded properly.

An attempt was made before trying CentOS 7 to install EicRoot on Ubuntu. Although the system was successfully installed on the writer's Ubuntu partition, the process could not be replicated on the lab computer named Beauty. This is a serious flaw since insufficient documentation is available for EicRoot.

1.6 Preliminary Results Using EicRoot: Simulating Chromium GEMs

By comparing GEM chambers with copper foils to those with chromium foils, one can see that the momentum resolution of the chambers is improved by reducing the material in the first two GEM foils. In this study, a 3-1-2-1 mm gap configuration and a beam of 1 GeV electrons at 25° to the beamline are used. Note that this study is performed in the forward region of the interaction point. The geometry of this study is shown in Figure 2.

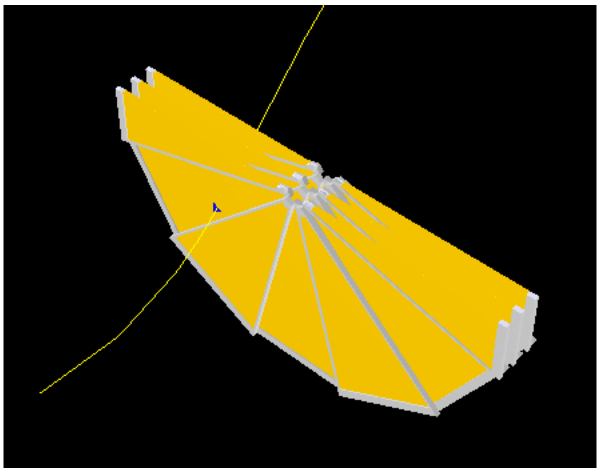

Figure 2: Cross-section of 3 GEM rings in forward region of IP

The first configuration simulated was when all GEM foils were "standard," i.e. 5 μm thick planes of copper on both sides of a 50 μm thick plane of polyimide material. The Guassian plot of events versus percent difference between reconstructed and simulated momenta, momuntum resolution, has a root mean square of ±8.5% (Figure 3).

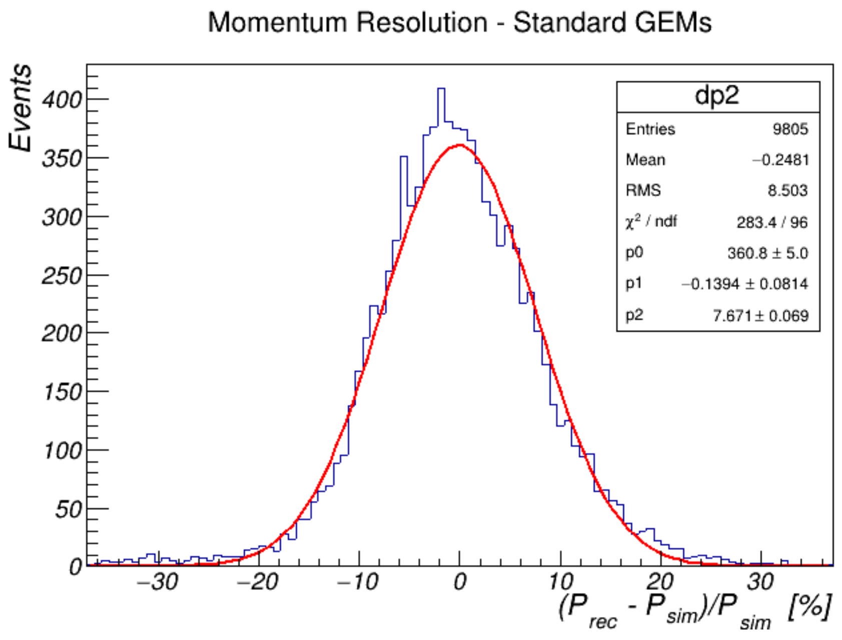

Figure 3: Momentum resolution plot associated with 3 rings of GEMs with copper foils

Next, material was reduced to accomodate for two chromium GEM foils followed by a "standard" GEM foil. This configuration was implemented since this would be a realistic configuration in the actual detectors. Instead of having all three foils be chromium GEMs, only two are chromium so that the readout electronics would not be burnt out by the secondary electrons. Chromium GEM foils are different than "standard" in that a 50 μ thick plane of polymide is sandwiched by two planes of 200 nm thick planes of chromium.

Since no direct method of using chromium foils is possible in EicRoot, the thickness of copper that would be equivalent to 2 foils of 200 nm of chromium and one of 5 μm of copper is calculated as follows.

$$\begin{aligned}
\frac{X_{equiv}}{x_{0,Cu}} &= \frac{2X_{Cr} \,/\, x_{0,Cr} + X_{Cu} \,/\, x_{0,Cu}}{3} \\
X_{equiv} &= \frac{x_{0,Cu}\,(2X_{Cr} \,/\, x_{0,Cr} + X_{Cu} \,/\, x_{0,Cu})}{3} \\
&= \frac{(1.436\,cm)\,(2(2 \cdot 10^{-5}\,cm) \,/\, (2.081\,cm) + (5 \cdot 10^{-4}\,cm) \,/\, (1.436\,cm))}{3} \\
&= 1.76\,\mu m
\end{aligned}$$

This thickness value is plugged into the variable defining thickness of copper in the GEM foils.

```
fit->mGemFoilCopperThickness = 0.00176; // GEM Cr layer
```

In this way, the desired configuration of two chromium GEM foils and one "standard" GEM foil is made.

The Gaussian plot of events versus momentum resolution for this chromium GEM configuration has an associated root mean square of ±8.3% (Figure 4).

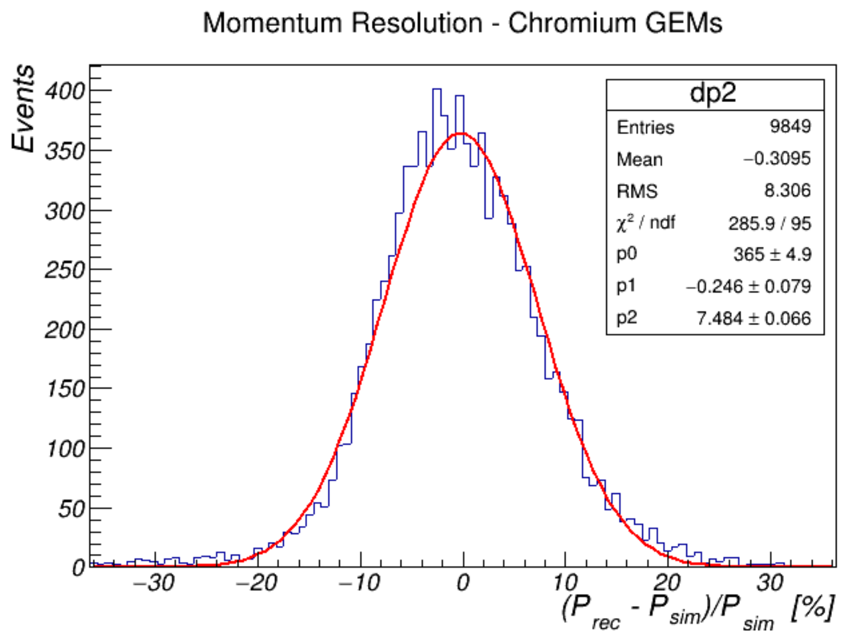

Figure 4: Momentum resolution plot associated with 3 rings of GEMs with 2 chromium foils and 1 copper foil

Comparing these RMS values for "standard" and chromium GEMs, one can see that they differ by only 0.2%, favoring the chromium configuration. Also, the momentum values p_0, p_1, and p_2 have slightly smaller errors for the chromium case. This implies that reducing the material from copper to chromium in two of the GEM foils has a minimal impact on the momentum resolution.

The next step in this study is to measure track resolution of the electron beam through the chambers. This will be implemented by adding a dummy plane about 15 cm past the third GEM ring. The beam of electrons will be focused on either the x or y axis, and the positions of events on the chosen axis will be histogrammed. IErrorn this fashion, one will be able to tell how the reduction of material in the GEM chambers affects the spread of position values read after the chambers. It is expected that the chromium configuration will produce a smaller spread than the “standard” configuration.

2 Machining Parts Using PEEK

2.1 Introduction to PEEK

Polyetheretherketone (PEEK) is a strong plastic material, having great chemical resistance and performing well at high temperatures [4]. As SpecialChem points out, PEEK belongs to the polyketone polymer family. It is the most popular polyketone material to use since it is produced in large quantities and used in various sectors of industry. Some applications include replacement of metals in cars and planes to reduce weight and erosion.

The particular interest in PEEK that the EIC R&D team at Florida Tech has is the potential the material holds for the mechanical stretching method employed for stretching the stack of GEM foils. The parts manufactured using PEEK are the pull-outs and two layers of inner frames. To increase the spacing in the stack, the two layers of inner frames are both 2 mm thick, making the stack a 3 mm, 2 mm, 2 mm, 2 mm stack (instead of the previous 3-1-2-1 setting). This will be a demonstration of the full modularity of the detector.

2.2 Machining Methods & Limiting Factors

For machining the pull-out parts, a 12”x12” plate of about 6 mm thick PEEK was taken to the on campus machine shop and machined by one of the technicians running the facility. A few of the parts were not tapped on one side, so these were tapped by hand. One of the parts, illustrated in Figure 5, was cleaned using acetone.

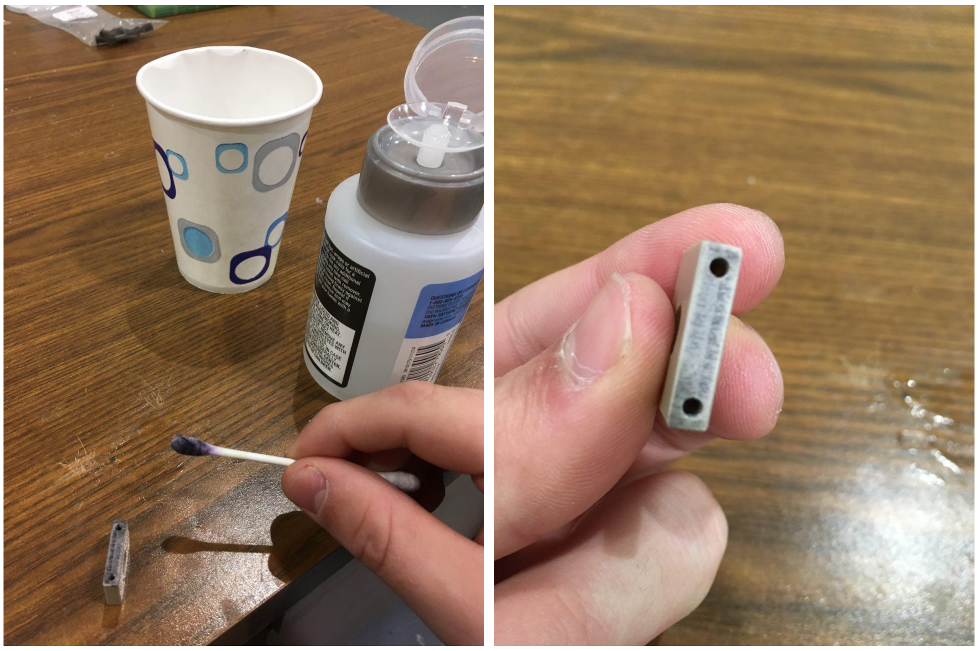

Figure 5: Cleaning of pull-out part with ink mark using acetone

The laser cutter in the Harris Student Design Center (HSDC) is being utilized in the machining of the inner frame pieces using a 12"x6" plate of 2 mm thick PEEK. As can be seen in Figure 6, the laser cutter produced scorch marks. This was cleaned, with increasing success, using water, 71% ethyl alcohol, 91% ethyl alcohol, and acetone.

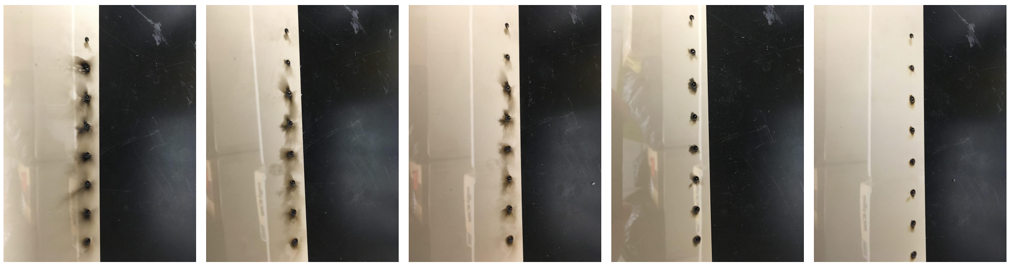

Figure 6: Cleaning of burn marks on PEEK plate; From left to right: original burn marks → clean with water → clean with 71% alcohol → clean with 91% alcohol → clean with acetone

Most likely, the cause of this scoring is a lower power setting. In the beginning of the Spring 2019 semester, these frame pieces will be cut using a higher power setting, such as 90-100%.

2.3 Designs Used for Machining Parts

The following section highlights the designs used to manufacture the parts necessary to update the GEM chamber with PEEK. Figure 7 is the associated design for 60 pull-out parts. Figures 8, 9, and 10 are the designs for the small end and wide end parts, respectively. Figures 11 and 12 are the designs for the side frames.

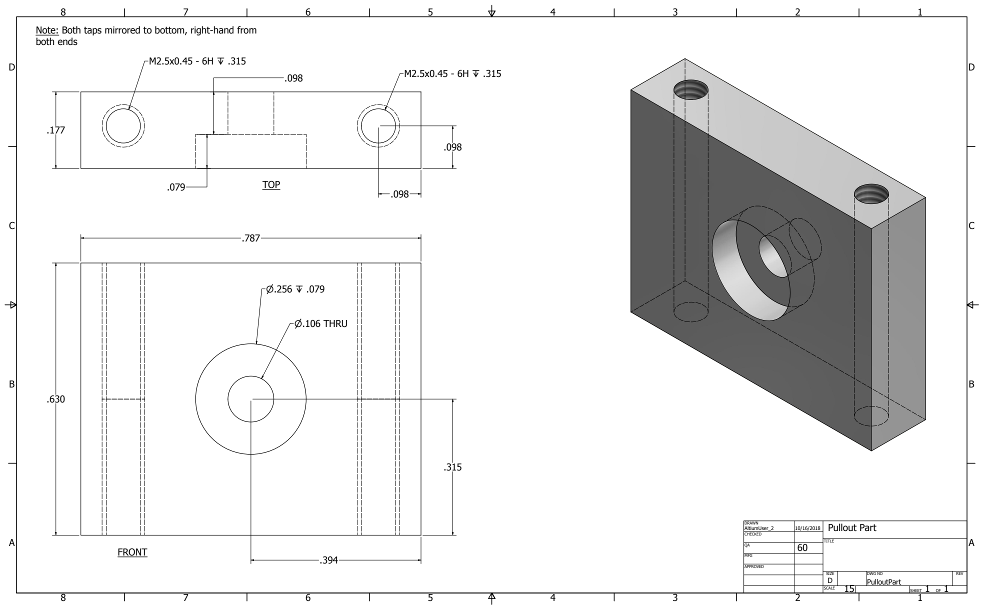

Figure 7: Design of pull-out part using PEEK material; 60 parts are manufactured

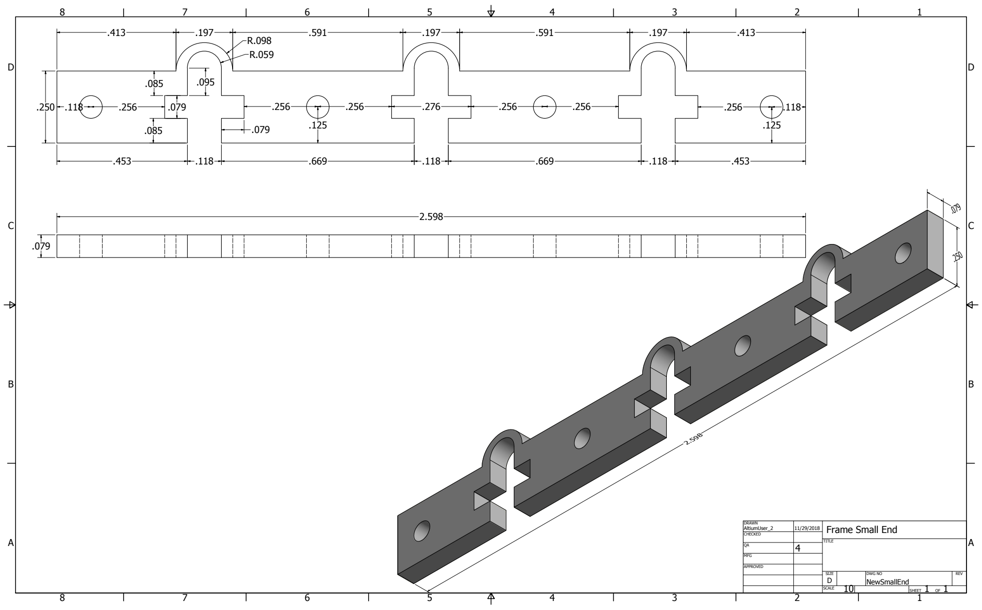

Figure 8: Design of 2 mm small end frame made using PEEK; 4 parts are manufactured

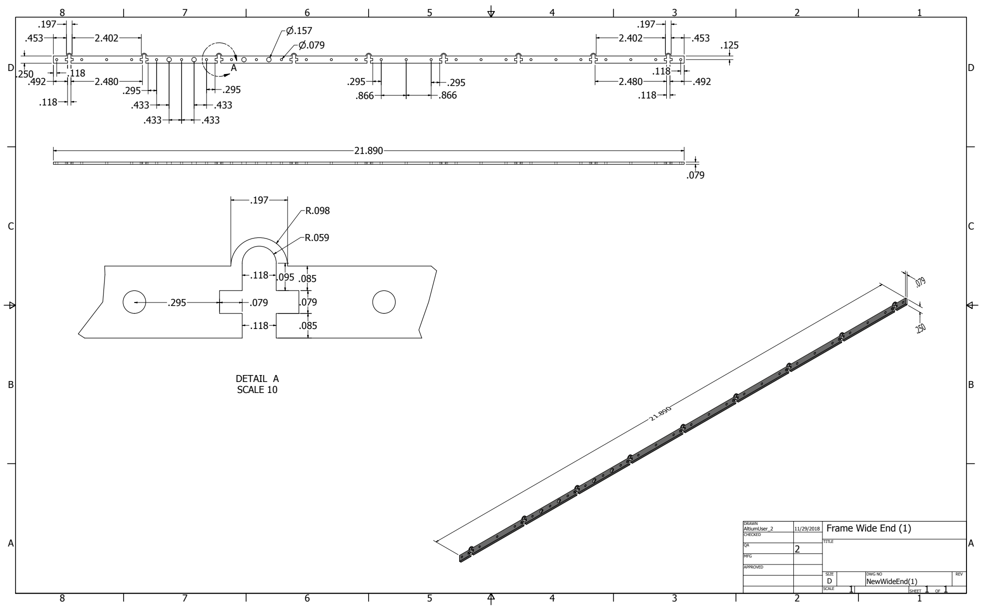

Figure 9: Design of first 2 mm wide end frame made using PEEK; 2 parts are manufactured

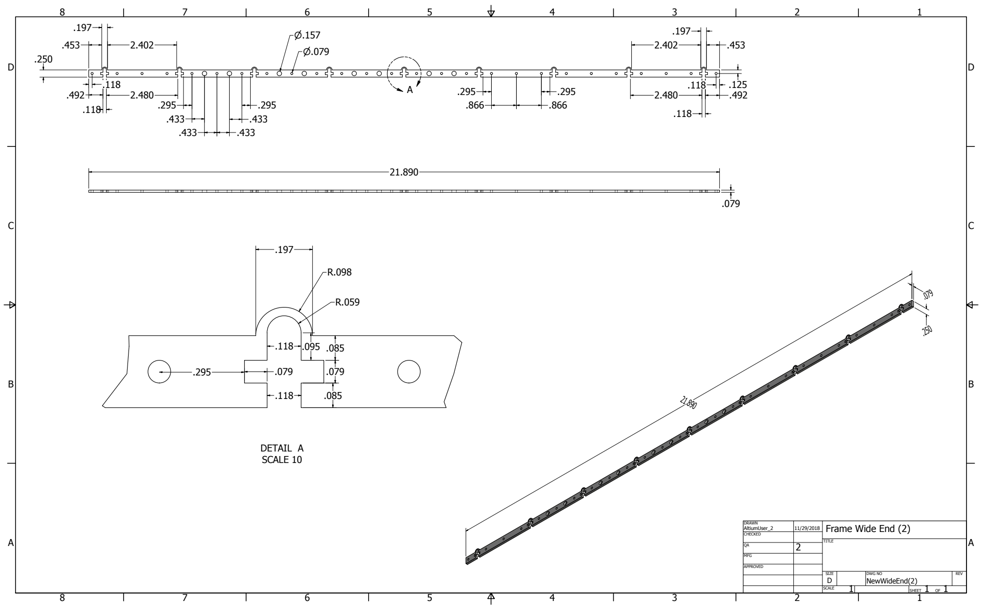

Figure 10: Design of second 2 mm wide end frame made using PEEK; 2 parts are manufactured

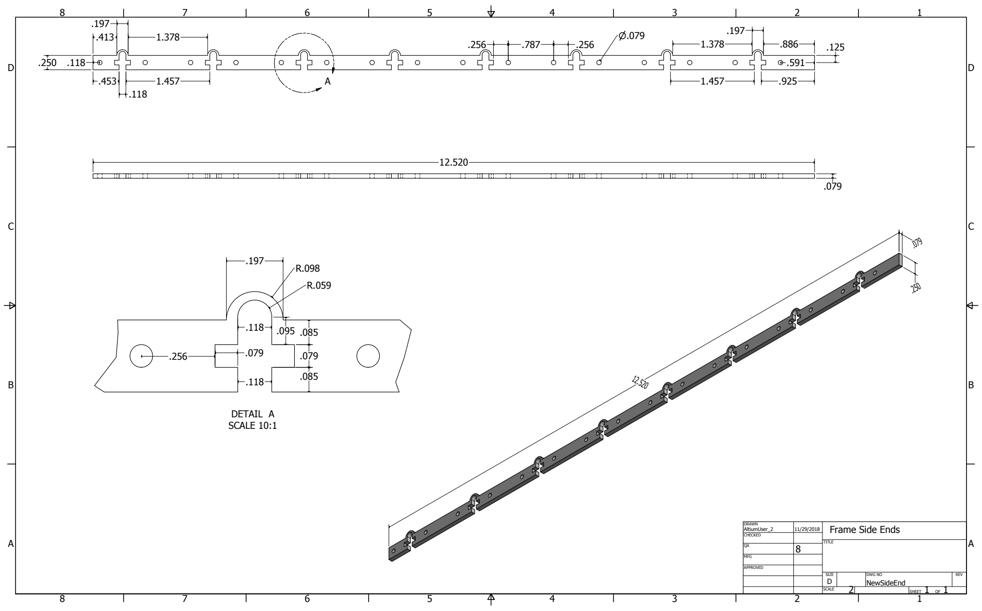

Figure 11: Design of ends of 2mm side frames made using PEEK; 8 parts are manufactured

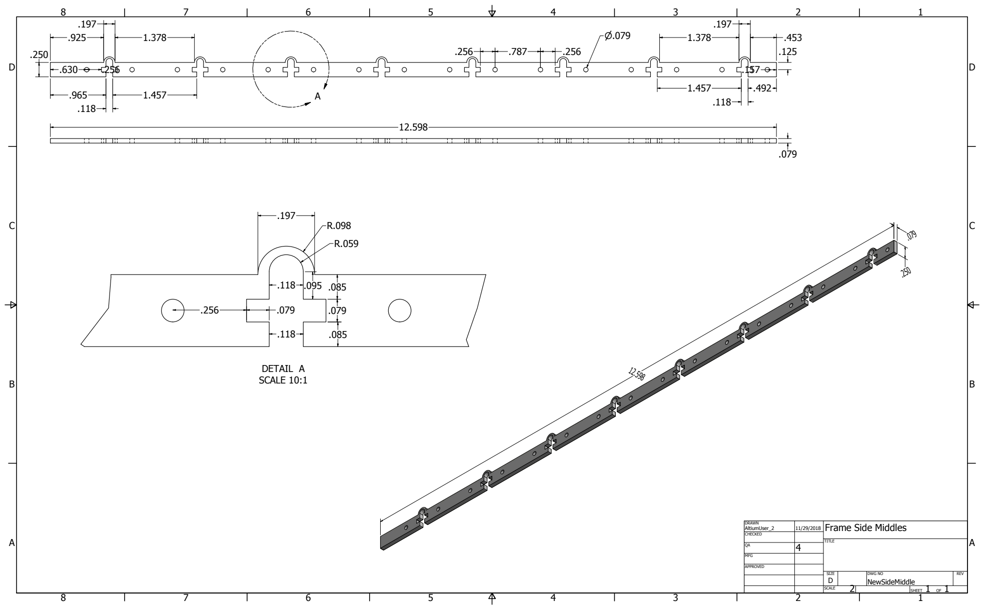

Figure 12: Design of middle sections of 2 mm side frames using PEEK; 4 parts are manufactured

3 Summary

The first section outlined how one would use EicRoot in a Docker container and outlined a strategy for running the sotware locally on one's machine. Various problems were addressed. One such problem is the use of outdated dependencies, i.e. ROOT4 and ROOT5. Preliminary results using EicRoot were presented. Table 1 summarizes these results.

Table 1: Comparison of material budget and impact on momentum resolution for "standard" and chromium GEM chambers, simulated in EicRoot

Parameters	**GEM**	**Cr-GEM**
GEM Conductive Material		
GEM Foil 1	Cu	Cr
GEM Foil 2	Cu	Cr
GEM Foil 3	Cu	Cu
GEM Conductive Material Thickness		
GEM Foil 1	5 μm	200 nm
GEM Foil 2	5 μm	200 nm
GEM Foil 3	5 μm	5 μm
Momentum Resolution		
Mean	-0.25%	-0.31%
RMS	$\pm$8.5%	$\pm$8.3%

The second section presented the advantages of PEEK material and how it will be utilized in the Florida Tech EIC prototype GEM. Machining methods and issues were presented. One such issue is how the material is affected by a low-power laser beam.

In conclusion, this document provides guidelines for using EicRoot and notes on PEEK material for a prototype GEM. When implemented in the GEM chamber, the pull-outs improve the mechanical stretching of the GEM foils. A view of the GEM stack is in Figure 13.

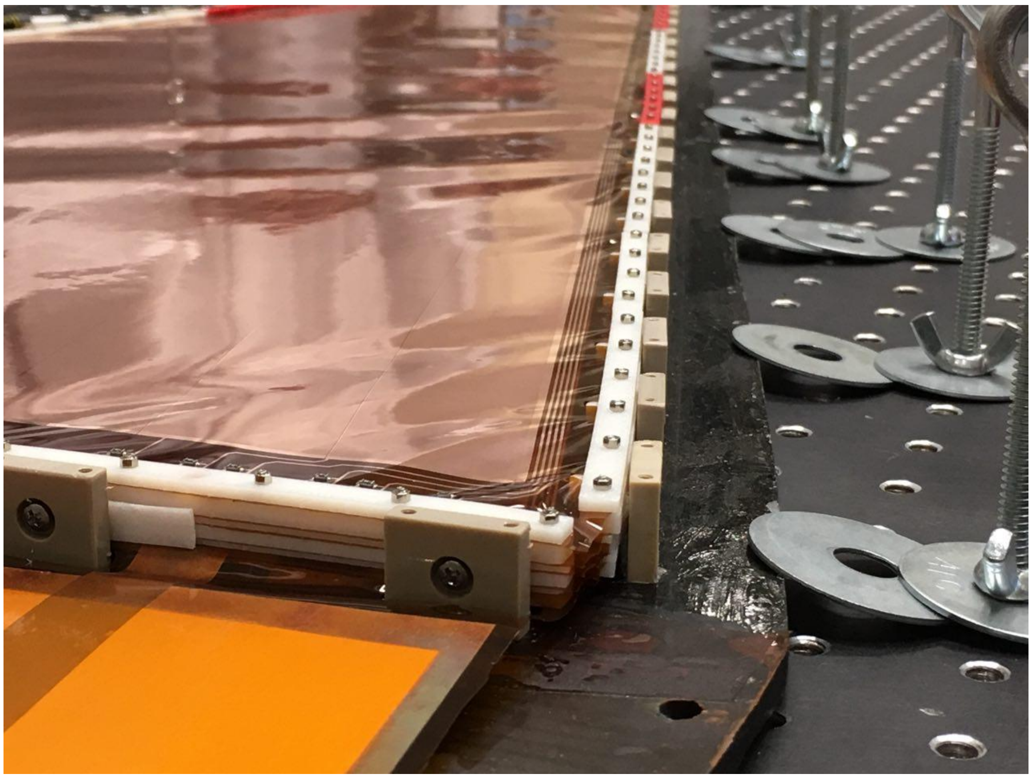

Figure 13: PEEK pull-outs installed in GEM chamber

References

[1] "Get Docker CE for CentOS." Docker. https://docs.docker.com/install/linux/docker-ce/centos/. 2018.

[2] "Install Docker for Mac." Docker. https://docs.docker.com/docker-for-mac/install/. 2018.

[3] "FairSoft/DEPENDENCIES." GitHub. FairRootGroup. https://github.com/FairRootGroup/FairSoft/blob/master/DEPENDENCIES. 2017.

[4] "Polyetheretherketone (PEEK): A Complete Guide on High-Heat Engineering Plastic." SpecialChem. https://omnexus.specialchem.com/selection-guide/polyetheretherketone-peek-thermoplastic#content. 2018.

[5] "Atomic and nuclear properties of materials." Particle Data Group. http://pdg.lbl.gov/2005/reviews/atomicrpp.pdf. 2005.

[6] "Atomic and nuclear properties of chromium (Cr)." Particle Data Group. http://pdg.lbl.gov/2014/AtomicNuclearProperties/HTML/chromium_Cr.html. 2014

[7] "Atomic and nuclear properties of argon gas (Ar)." Particle Data Group. http://pdg.lbl.gov/2017/AtomicNuclearProperties/HTML/argon_gas_Ar.html. 2017

[8] "Atomic and nuclear properties of materials: Carbon dioxide gas (CO_2)." Particle Data Group. http://pdg.lbl.gov/2012/AtomicNuclearProperties/HTML_PAGES/134.html. 2012

A Material Budget

Table 2 shows the materials used in the GEMs simulated in EicRoot. An improvement for future simulations would be to use frame dimensions similar to those implemented in the prototype Florida Tech chamber. Note that although the gas mixture has a high radiation length, its density is very low compared to the other materials.

Table 2: Table of materials used in GEMs for EicRoot simulation with dimensions and radiation lengths [5], [6], [7], [8]

Item	Material	Thickness [μm]	Rad. Length [cm]
Drift Frame (3 cm width)	G10	$1 \cdot 10^3$	19.4
Outer Frame (8 mm width)	G10	$1.6 \cdot 10^4$	19.4
Entrance Window	KAPTON	25	28.6
Drift Foil	Copper	5	1.43
	KAPTON	50	28.6
Cu-GEM Foil	Copper	5	1.43
	KAPTON	50	28.6
	Copper	5	1.43
Cr-GEM Foil	Chromium	0.2	2.08
	KAPTON	50	28.6
	Chromium	0.2	2.08
Readout Foil	Copper	5	1.43
	KAPTON	50	28.6
Exit Window	KAPTON	25	28.6
R/O Frame (3 cm width)	G10	$1 \cdot 10^3$	19.4
Ar/CO_2 (70/30)	Ar	$1.2 \cdot 10^4$	$1.18 \cdot 10^4$
	CO_2	$1.2 \cdot 10^4$	$1.97 \cdot 10^4$