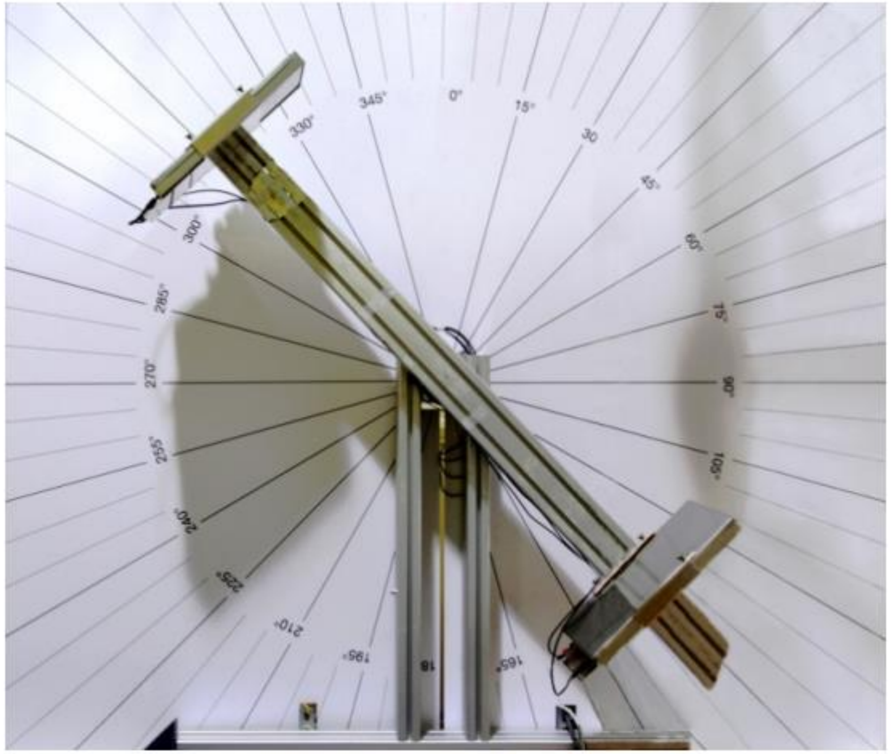
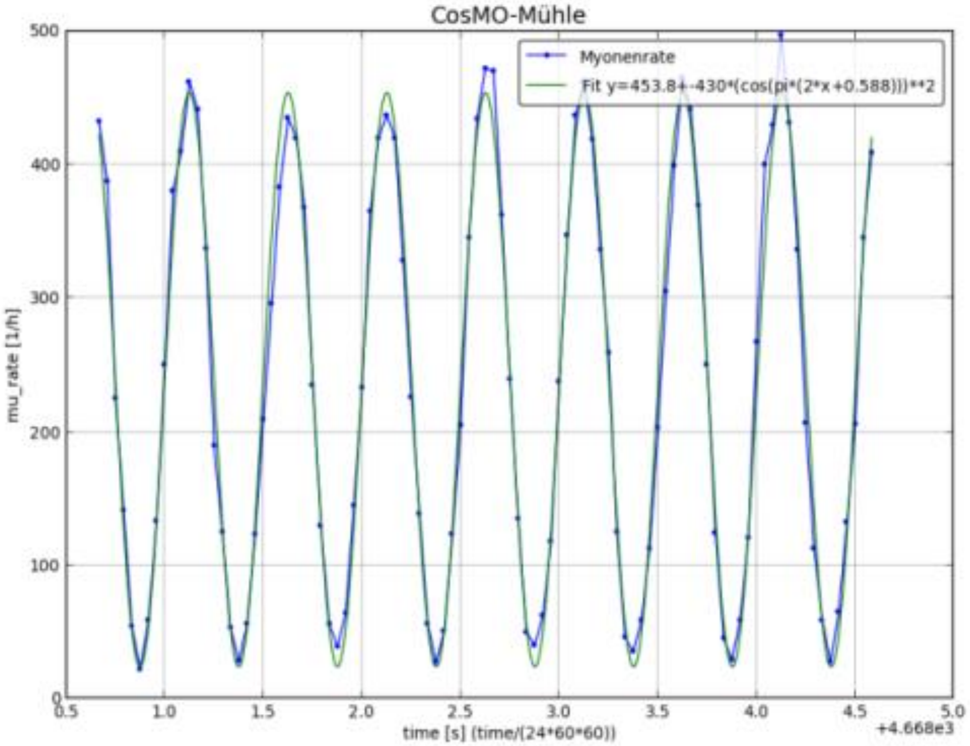
PEDRO COPETO’S NUC-RIA INTERNSHIP

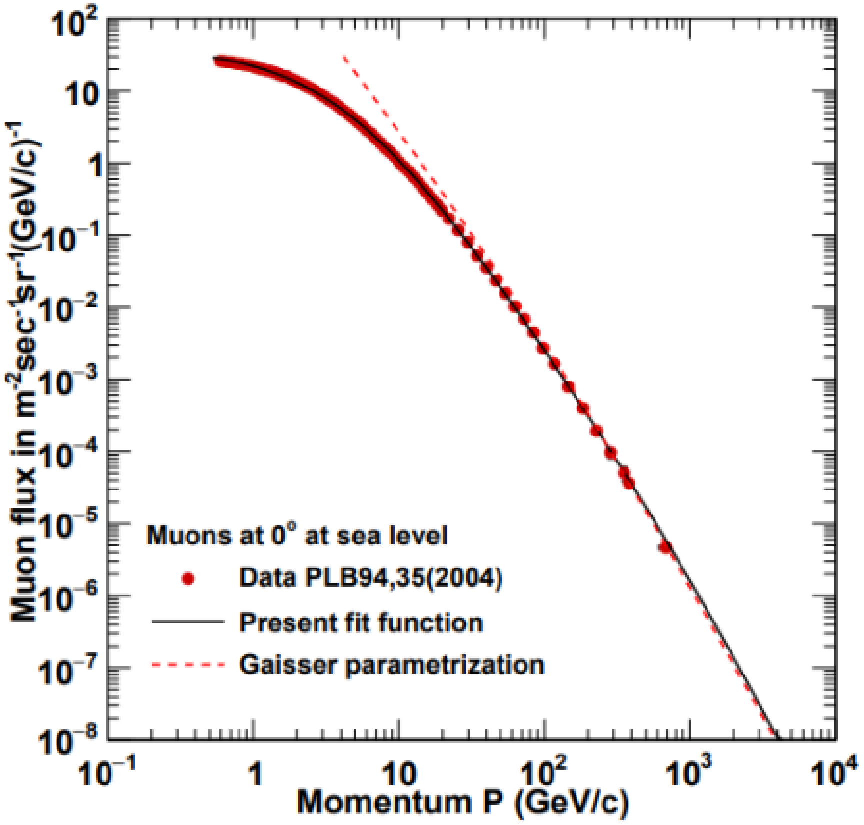
Muon detection with a scintillator-PMT based setup

ANGLE DISTRIBUTION OF MUONS





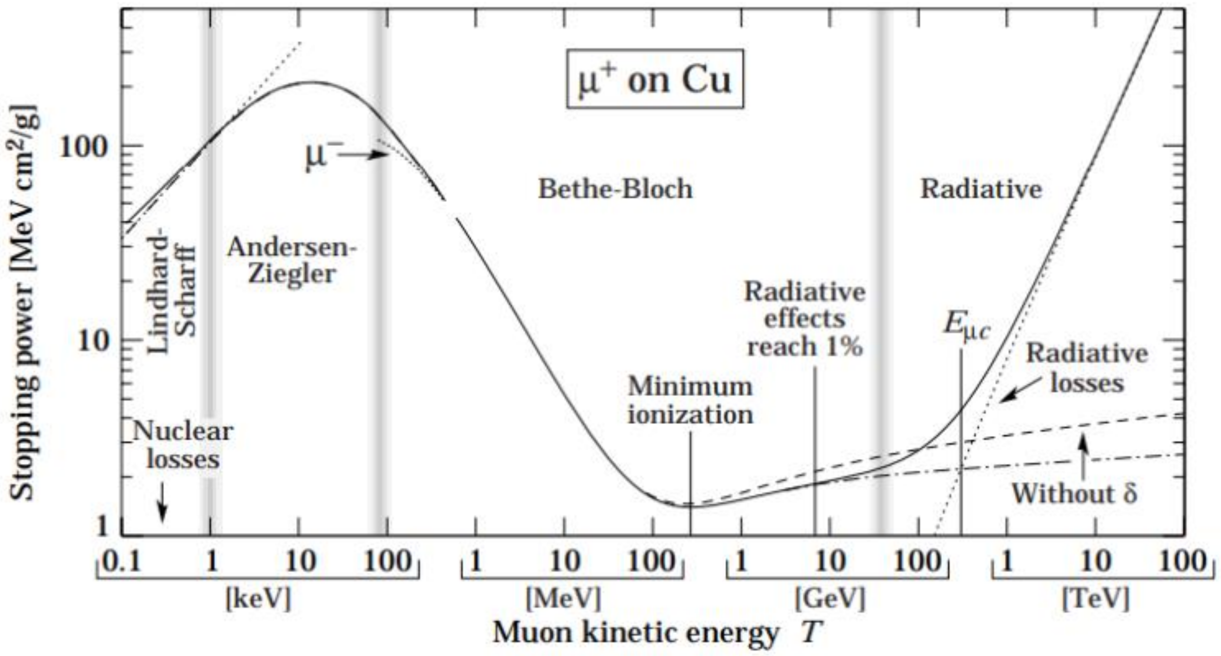
MUON ENERGY



$$E = \sqrt{p^2c^2 + m_0^2c^4}$$

E = 0.6 to 10 GeV

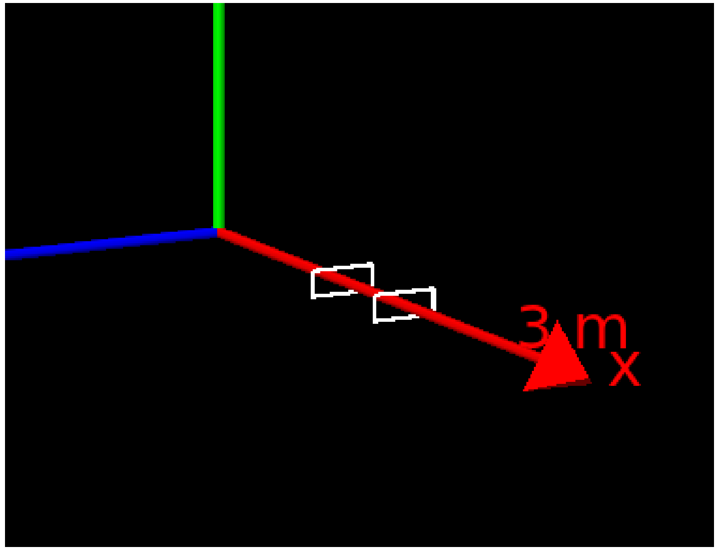
MUON STOPPING POWER

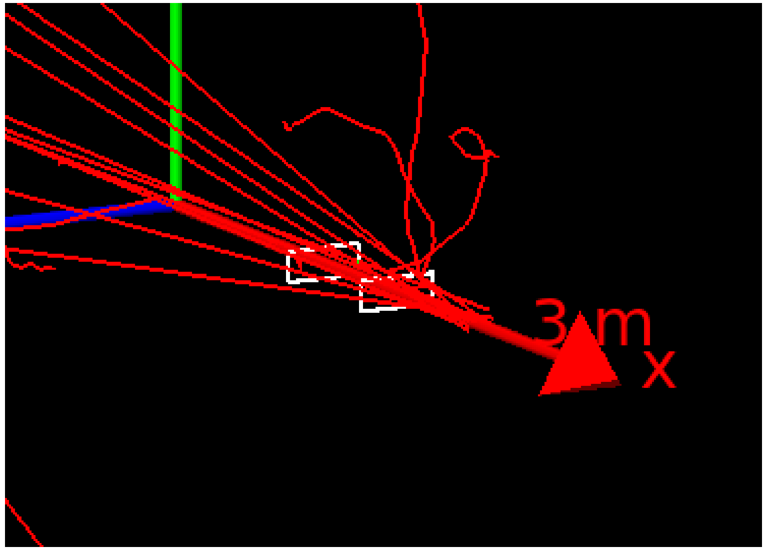


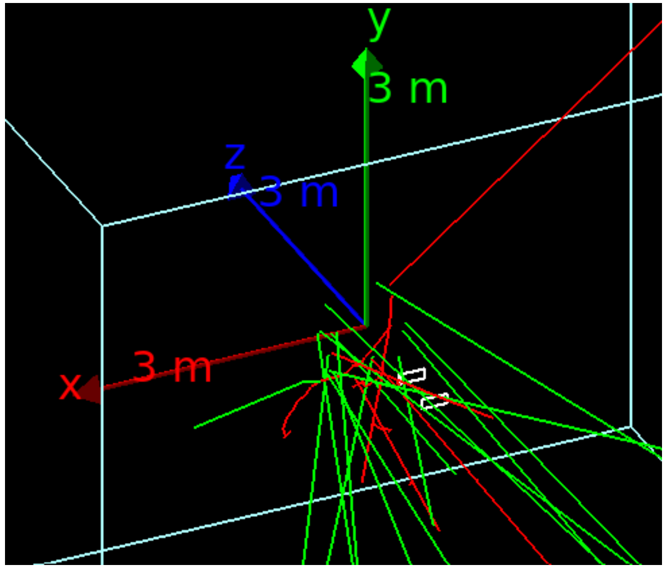
Muons have an average minimum ionizing energy equal to **1.956 MeV/gcm^{-2}**

SIMULATION

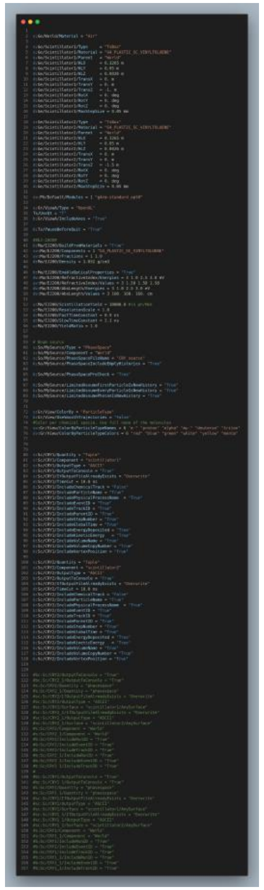
- Phase Space Scorer
- nBio Scorer
- CRY-TOPAS







TOPAS



```

s:Ge/World/Material = "Air"

s:Ge/Scintillator1/Type     = "TsBox"
s:Ge/Scintillator1/Material = "G4_PLASTIC_SC_VINYLTOLUENE"
s:Ge/Scintillator1/Parent   = "World"
d:Ge/Scintillator1/HLX      = 0.1265 m
d:Ge/Scintillator1/HLY      = 0.05 m
d:Ge/Scintillator1/HLZ      = 0.0020 m
d:Ge/Scintillator1/TransX   = 0. m
d:Ge/Scintillator1/TransY   = 0. m
d:Ge/Scintillator1/TransZ   = -1. m
d:Ge/Scintillator1/RotX     = 0. deg
d:Ge/Scintillator1/RotY     = 0. deg
d:Ge/Scintillator1/RotZ     = 0. deg
d:Ge/Scintillator1/MaxStepSize = 0.05 mm

s:Ge/Scintillator2/Type     = "TsBox"
s:Ge/Scintillator2/Material = "G4_PLASTIC_SC_VINYLTOLUENE"
s:Ge/Scintillator2/Parent   = "World"
d:Ge/Scintillator2/HLX      = 0.1265 m
d:Ge/Scintillator2/HLY      = 0.05 m
d:Ge/Scintillator2/HLZ      = 0.0020 m
d:Ge/Scintillator2/TransX   = 0. m
d:Ge/Scintillator2/TransY   = 0. m
d:Ge/Scintillator2/TransZ   = -1.5 m
d:Ge/Scintillator2/RotX     = 0. deg
d:Ge/Scintillator2/RotY     = 0. deg
d:Ge/Scintillator2/RotZ     = 0. deg
d:Ge/Scintillator2/MaxStepSize = 0.05 mm

sv:Ph/Default/Modules = 1 "g4em-standard_opt0"

s:Gr/ViewA/Type = "OpenGL"
Ts/UseQt = "T"
b:Gr/ViewA/IncludeAxes = "True"

b:Ts/PauseBeforeQuit = "True"

#EJ-200
b:Ma/EJ200/BuildFromMaterials = "True"
sv:Ma/EJ200/Components = 1 "G4_PLASTIC_SC_VINYLTOLUENE"
uv:Ma/EJ200/Fractions = 1 1.0
d:Ma/EJ200/Density = 1.023 g/cm3

b:Ma/EJ200/EnableOpticalProperties = "True"
dv:Ma/EJ200/RefractiveIndex/Energies = 3 1.0 2.5 3.0 eV
uv:Ma/EJ200/RefractiveIndex/Values = 3 1.58 1.58 1.58
dv:Ma/EJ200/AbsLength/Energies = 3 1.0 2.5 3.0 eV
dv:Ma/EJ200/AbsLength/Values = 3 380. 380. 380. cm

u:Ma/EJ200/ScintillationYield = 10000.0 #[illegible]
u:Ma/EJ200/ResolutionScale = 1.0
d:Ma/EJ200/FastTimeConstant = 0.9 ns
d:Ma/EJ200/SlowTimeConstant = 2.1 ns
u:Ma/EJ200/YieldRatio = 1.0


# Beam source
s:So/MySource/Type = "PhaseSpace"
s:So/MySource/Component = "World"
s:So/MySource/PhaseSpaceFileName = "CRY_source"
b:So/MySource/PhaseSpaceIncludeEmptyHistories = "True"

b:So/MySource/PhaseSpacePreCheck = "True"

b:So/MySource/LimitedAssumeFirstParticleIsNewHistory = "True"
b:So/MySource/LimitedAssumeEveryParticleIsNewHistory = "True"
b:So/MySource/LimitedAssumePhotonIsNewHistory = "True"


s:Gr/ViewA/ColorBy = "ParticleType"
b:Gr/ViewA/UseSmoothTrajectories = "False"
#Color per chemical specie. Use full name of the molecules
sv:Gr/ViewA/ColorByParticleTypeNames = 6 "e-" "proton" "alpha" "mu-" "neutron" "triton"
sv:Gr/ViewA/ColorByParticleTypeColors = 6 "red" "blue" "green" "white" "yellow" "magenta"



s:Sc/CRY1/Quantity = "Tuple"
s:Sc/CRY1/Component = "scintillator1"
s:Sc/CRY1/OutputType = "ASCII"
b:Sc/CRY1/OutputToConsole = "True"
s:Sc/CRY1/IfOutputFileAlreadyExists = "Overwrite"
d:Sc/CRY1/TimeCut = 10.0 ns
b:Sc/CRY1/IncludeChemicalTrack = "False"
b:Sc/CRY1/IncludeParticleName = "True"
b:Sc/CRY1/IncludePhysicalProcessName  = "True"
b:Sc/CRY1/IncludeEventID = "True"
b:Sc/CRY1/IncludeTrackID = "True"
b:Sc/CRY1/IncludeParentID = "True"
b:Sc/CRY1/IncludeStepNumber = "True"
b:Sc/CRY1/IncludeGlobalTime = "True"
b:Sc/CRY1/IncludeEnergyDeposited = "True"
b:Sc/CRY1/IncludeKineticEnergy   = "True"
b:Sc/CRY1/IncludeVolumeName = "True"
b:Sc/CRY1/IncludeVolumeCopyNumber = "True"
b:Sc/CRY1/IncludeVertexPosition = "True"

s:Sc/CRY2/Quantity = "Tuple"
s:Sc/CRY2/Component = "scintillator2"
s:Sc/CRY2/OutputType = "ASCII"
b:Sc/CRY2/OutputToConsole = "True"
s:Sc/CRY2/IfOutputFileAlreadyExists = "Overwrite"
d:Sc/CRY2/TimeCut = 10.0 ns
b:Sc/CRY2/IncludeChemicalTrack = "False"
b:Sc/CRY2/IncludeParticleName = "True"
b:Sc/CRY2/IncludePhysicalProcessName  = "True"
b:Sc/CRY2/IncludeEventID = "True"
b:Sc/CRY2/IncludeTrackID = "True"
b:Sc/CRY2/IncludeParentID = "True"
b:Sc/CRY2/IncludeStepNumber = "True"
b:Sc/CRY2/IncludeGlobalTime = "True"
b:Sc/CRY2/IncludeEnergyDeposited = "True"
b:Sc/CRY2/IncludeKineticEnergy   = "True"
b:Sc/CRY2/IncludeVolumeName = "True"
b:Sc/CRY2/IncludeVolumeCopyNumber = "True"
b:Sc/CRY2/IncludeVertexPosition = "True"


#b:Sc/CRY2/OutputToConsole = "True"
#b:Sc/CRY2_1/OutputToConsole = "True"
#s:Sc/CRY2/Quantity = "phasespace"
#s:Sc/CRY2_1/Quantity = "phasespace"
#s:Sc/CRY2/IfOutputFileAlreadyExists = "Overwrite"
#s:Sc/CRY2/OutputType = "ASCII"
#s:Sc/CRY2/Surface = "scintillator2/AnySurface"
#s:Sc/CRY2_1/IfOutputFileAlreadyExists = "Overwrite"
#s:Sc/CRY2_1/OutputType = "ASCII"
#s:Sc/CRY2_1/Surface = "scintillator2/AnySurface"
#s:Sc/CRY2/Component = "World"
#s:Sc/CRY2_1/Component = "World"
#b:Sc/CRY2/IncludeRunID = "True"
#b:Sc/CRY2/IncludeEventID = "True"
#b:Sc/CRY2/IncludeTrackID = "True"
#b:Sc/CRY2_1/IncludeRunID = "True"
#b:Sc/CRY2_1/IncludeEventID = "True"
#b:Sc/CRY2_1/IncludeTrackID = "True"
#
#b:Sc/CRY1/OutputToConsole = "True"
#b:Sc/CRY1_1/OutputToConsole = "True"
#s:Sc/CRY1/Quantity = "phasespace"
#s:Sc/CRY1_1/Quantity = "phasespace"
#s:Sc/CRY1/IfOutputFileAlreadyExists = "Overwrite"
#s:Sc/CRY1/OutputType = "ASCII"
#s:Sc/CRY1/Surface = "scintillator1/AnySurface"
#s:Sc/CRY1_1/IfOutputFileAlreadyExists = "Overwrite"
#s:Sc/CRY1_1/OutputType = "ASCII"
#s:Sc/CRY1_1/Surface = "scintillator1/AnySurface"
#s:Sc/CRY1/Component = "World"
#s:Sc/CRY1_1/Component = "World"
#b:Sc/CRY1/IncludeRunID = "True"
#b:Sc/CRY1/IncludeEventID = "True"
#b:Sc/CRY1/IncludeTrackID = "True"
#b:Sc/CRY1_1/IncludeRunID = "True"
#b:Sc/CRY1_1/IncludeEventID = "True"
#b:Sc/CRY1_1/IncludeTrackID = "True"
```

THEORETICAL ENERGY DEPOSITED

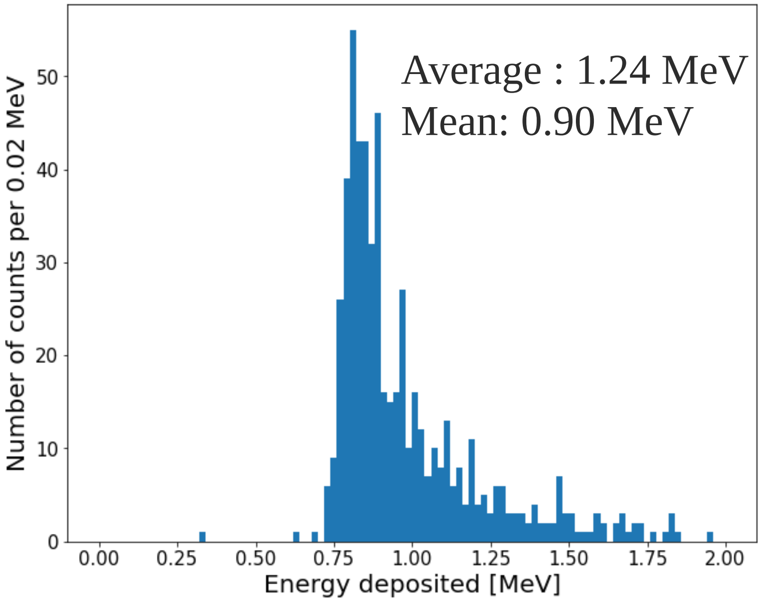
Stopping Power:
$S = 1.956$ MeV cm^2/g

Density of scintillators:
$\rho = 1.032$ g/cm^3

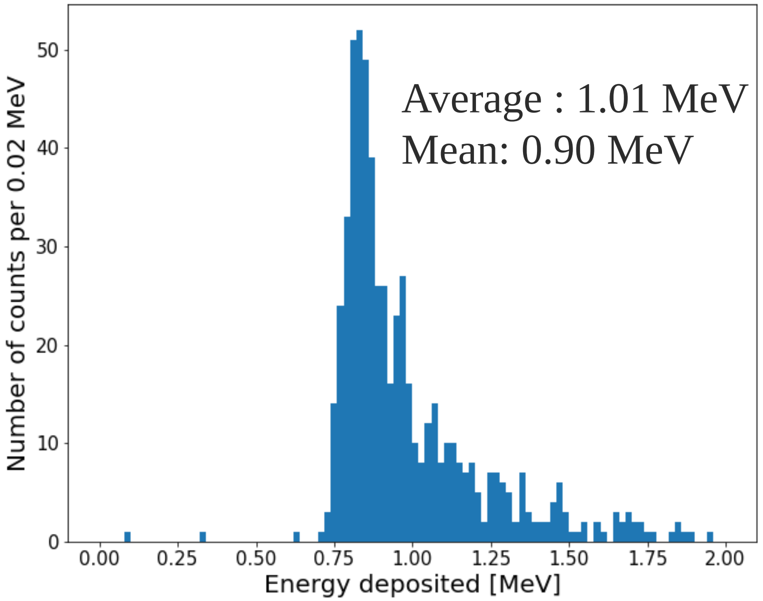
$S\rho = 2.019$ MeV/cm

2.019 x 0.52 = **1.05 MeV**

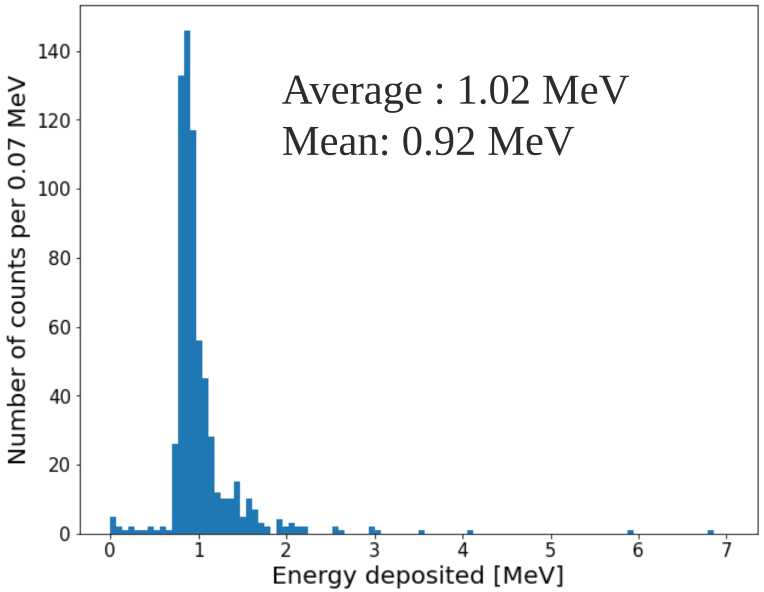
Phase Space



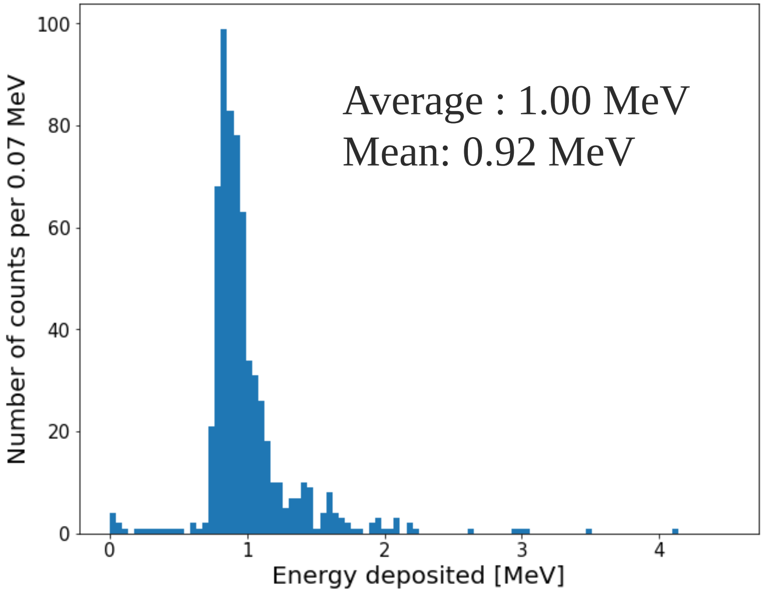
nBio



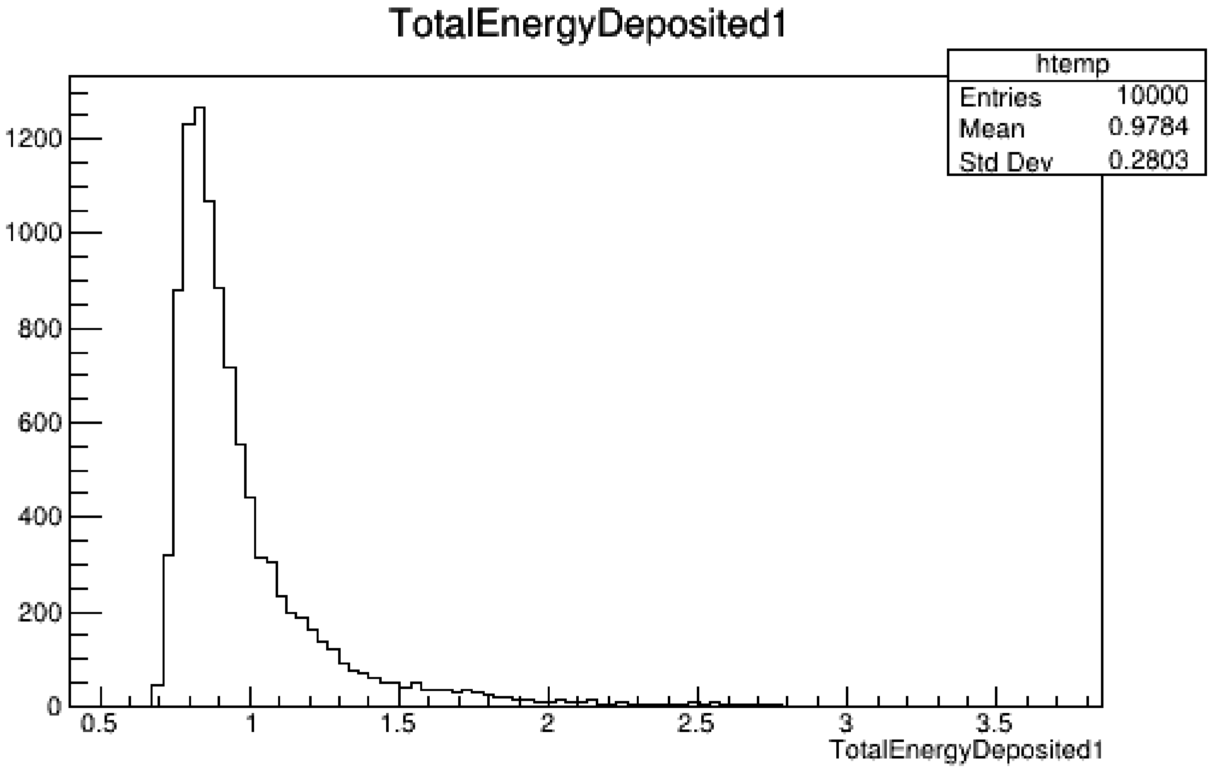
CRY-TOPAS

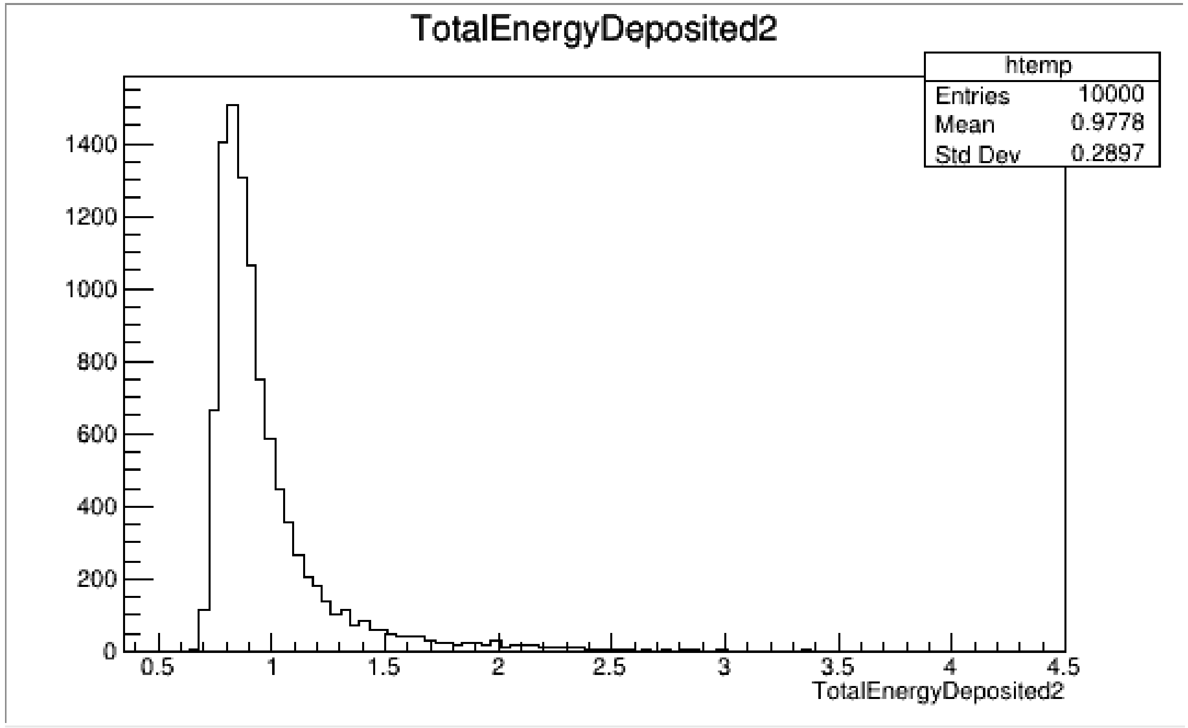


CRY-TOPAS only muons

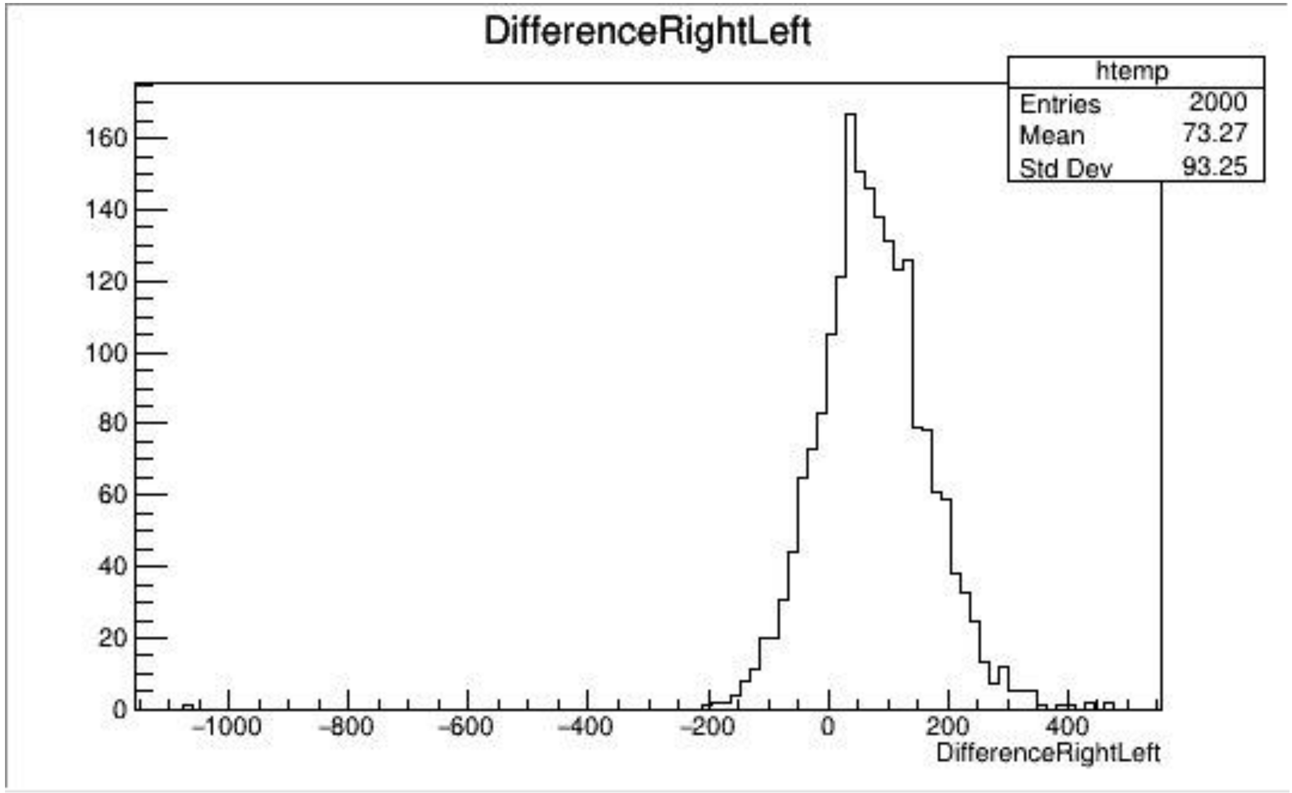


ENERGY DEPOSITED IN SCINTILLATORS

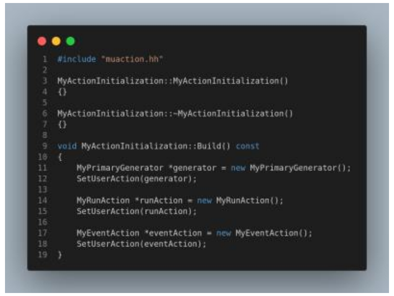




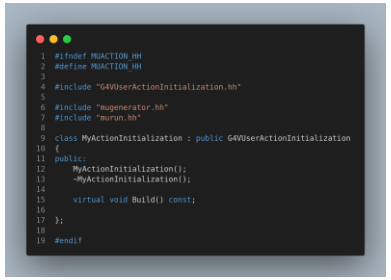
PMT'S PHOTON COUNT DIFFERENCE



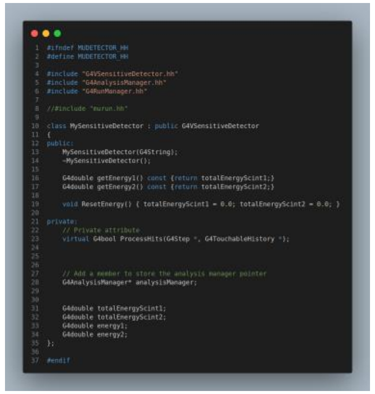
GEANT4



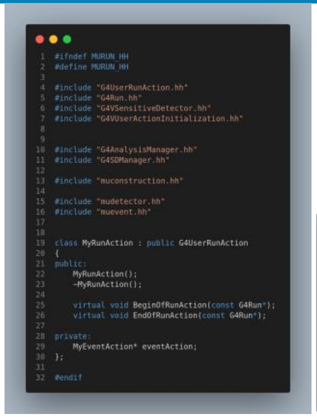
```cpp
#include "muaction.hh"

MyActionInitialization::MyActionInitialization()
{}

MyActionInitialization::~MyActionInitialization()
{}

void MyActionInitialization::Build() const
{
    MyPrimaryGenerator *generator = new MyPrimaryGenerator();
    SetUserAction(generator);

    MyRunAction *runAction = new MyRunAction();
    SetUserAction(runAction);

    MyEventAction *eventAction = new MyEventAction();
    SetUserAction(eventAction);
}
```



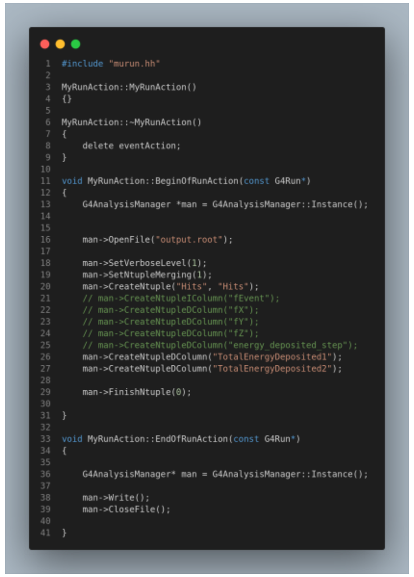
```cpp
#ifndef MUACTION_HH
#define MUACTION_HH

#include "G4VUserActionInitialization.hh"

#include "mugenerator.hh"
#include "murun.hh"

class MyActionInitialization : public G4VUserActionInitialization
{
public:
    MyActionInitialization();
    ~MyActionInitialization();

    virtual void Build() const;

};

#endif
```



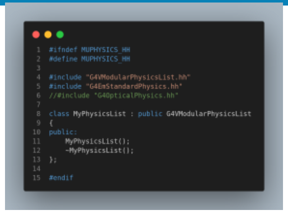
```cpp
#ifndef MUDETECTOR_HH
#define MUDETECTOR_HH

#include "G4VSensitiveDetector.hh"
#include "G4AnalysisManager.hh"
#include "G4RunManager.hh"

//#include "murun.hh"

class MySensitiveDetector : public G4VSensitiveDetector
{
public:
    MySensitiveDetector(G4String);
    ~MySensitiveDetector();

    G4double getEnergy1() const {return totalEnergyScint1;}
    G4double getEnergy2() const {return totalEnergyScint2;}

    void ResetEnergy() { totalEnergyScint1 = 0.0; totalEnergyScint2 = 0.0; }

private:
    // Private attribute
    virtual G4bool ProcessHits(G4Step *, G4TouchableHistory *);



    // Add a member to store the analysis manager pointer
    G4AnalysisManager* analysisManager;


    G4double totalEnergyScint1;
    G4double totalEnergyScint2;
    G4double energy1;
    G4double energy2;
};

#endif
```



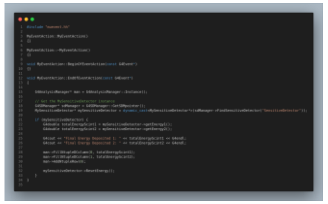
```cpp
#ifndef MURUN_HH
#define MURUN_HH

#include "G4UserRunAction.hh"
#include "G4Run.hh"
#include "G4VSensitiveDetector.hh"
#include "G4VUserActionInitialization.hh"


#include "G4AnalysisManager.hh"
#include "G4SDManager.hh"

#include "muconstruction.hh"

#include "mudetector.hh"
#include "muevent.hh"


class MyRunAction : public G4UserRunAction
{
public:
    MyRunAction();
    ~MyRunAction();

    virtual void BeginOfRunAction(const G4Run*);
    virtual void EndOfRunAction(const G4Run*);

private:
    MyEventAction* eventAction;
};

#endif
```



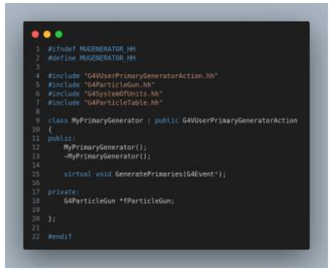
```cpp
#include "murun.hh"

MyRunAction::MyRunAction()
{}

MyRunAction::~MyRunAction()
{
    delete eventAction;
}

void MyRunAction::BeginOfRunAction(const G4Run*)
{
    G4AnalysisManager *man = G4AnalysisManager::Instance();


    man->OpenFile("output.root");

    man->SetVerboseLevel(1);
    man->SetNtupleMerging(1);
    man->CreateNtuple("Hits", "Hits");
    // man->CreateNtupleIColumn("fEvent");
    // man->CreateNtupleDColumn("fX");
    // man->CreateNtupleDColumn("fY");
    // man->CreateNtupleDColumn("fZ");
    // man->CreateNtupleDColumn("energy_deposited_step");
    man->CreateNtupleDColumn("TotalEnergyDeposited1");
    man->CreateNtupleDColumn("TotalEnergyDeposited2");

    man->FinishNtuple(0);

}

void MyRunAction::EndOfRunAction(const G4Run*)
{

    G4AnalysisManager* man = G4AnalysisManager::Instance();

    man->Write();
    man->CloseFile();

}
```



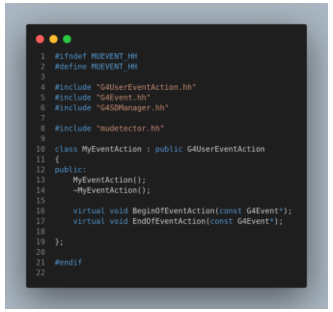
```cpp
#ifndef MUPHYSICS_HH
#define MUPHYSICS_HH

#include "G4VModularPhysicsList.hh"
#include "G4EmStandardPhysics.hh"
//#include "G4OpticalPhysics.hh"

class MyPhysicsList : public G4VModularPhysicsList
{
public:
    MyPhysicsList();
    ~MyPhysicsList();
};

#endif
```



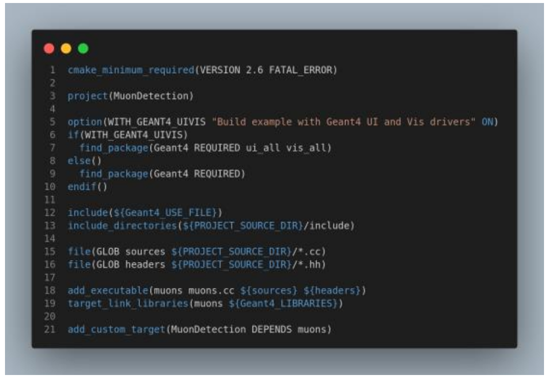
```cpp
#include "muevent.hh"

MyEventAction::MyEventAction()
{}

MyEventAction::~MyEventAction()
{}

void MyEventAction::BeginOfEventAction(const G4Event*)
{}

void MyEventAction::EndOfEventAction(const G4Event*)
{

    G4AnalysisManager* man = G4AnalysisManager::Instance();

    // Get the MySensitiveDetector instance
    G4SDManager* sdManager = G4SDManager::GetSDMpointer();
    MySensitiveDetector* mySensitiveDetector = dynamic_cast<MySensitiveDetector*>(sdManager->FindSensitiveDetector("SensitiveDetector"));

    if (mySensitiveDetector) {
        G4double totalEnergyScint1 = mySensitiveDetector->getEnergy1();
        G4double totalEnergyScint2 = mySensitiveDetector->getEnergy2();

        G4cout << "Final Energy Deposited 1: " << totalEnergyScint1 << G4endl;
        G4cout << "Final Energy Deposited 2: " << totalEnergyScint2 << G4endl;

        man->FillNtupleDColumn(0, totalEnergyScint1);
        man->FillNtupleDColumn(1, totalEnergyScint2);
        man->AddNtupleRow(0);

        mySensitiveDetector->ResetEnergy();
    }
}
```



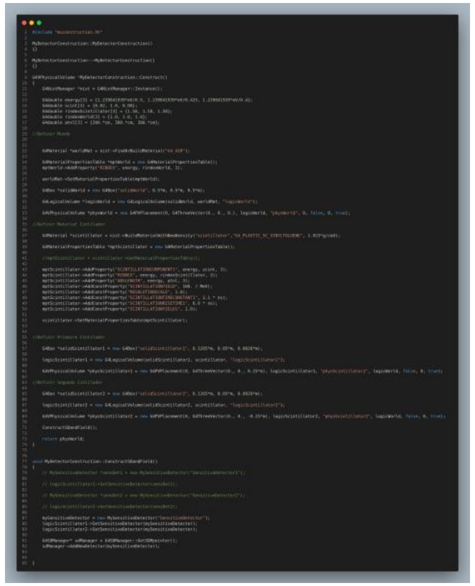
```cpp
#ifndef MUGENERATOR_HH
#define MUGENERATOR_HH

#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4ParticleGun.hh"
#include "G4SystemOfUnits.hh"
#include "G4ParticleTable.hh"

class MyPrimaryGenerator : public G4VUserPrimaryGeneratorAction
{
public:
    MyPrimaryGenerator();
    ~MyPrimaryGenerator();

    virtual void GeneratePrimaries(G4Event*);

private:
    G4ParticleGun *fParticleGun;
};

#endif
```



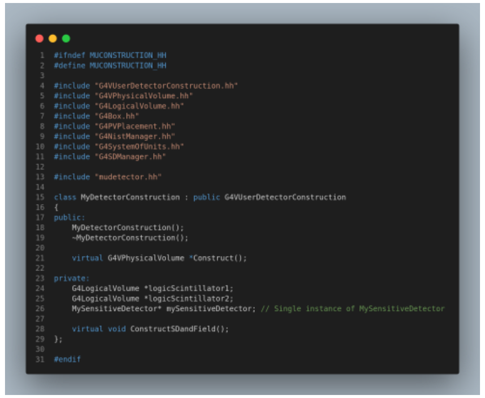
```cpp
#ifndef MUEVENT_HH
#define MUEVENT_HH

#include "G4UserEventAction.hh"
#include "G4Event.hh"
#include "G4SDManager.hh"

#include "mudetector.hh"

class MyEventAction : public G4UserEventAction
{
public:
    MyEventAction();
    ~MyEventAction();

    virtual void BeginOfEventAction(const G4Event*);
    virtual void EndOfEventAction(const G4Event*);

};

#endif
```



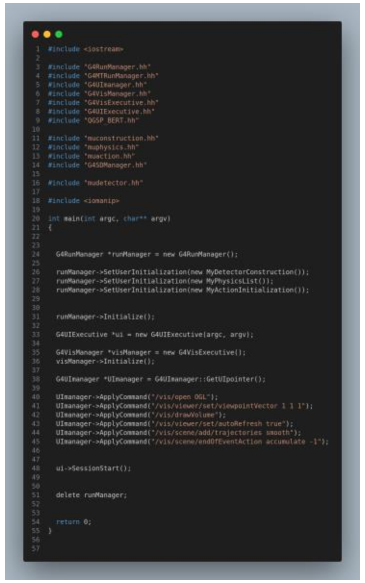
```cmake
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)

project(MuonDetection)

option(WITH_GEANT4_UIVIS "Build example with Geant4 UI and Vis drivers" ON)
if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
endif()

include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)

file(GLOB sources ${PROJECT_SOURCE_DIR}/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/*.hh)

add_executable(muons muons.cc ${sources} ${headers})
target_link_libraries(muons ${Geant4_LIBRARIES})

add_custom_target(MuonDetection DEPENDS muons)
```



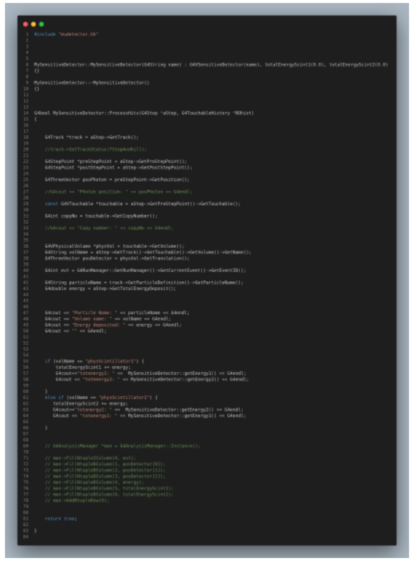
```cpp
#include "muconstruction.hh"

MyDetectorConstruction::MyDetectorConstruction()
{}

MyDetectorConstruction::~MyDetectorConstruction()
{}

G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
    G4NistManager *nist = G4NistManager::Instance();

    G4double energy[3] = {1.239841939*eV/0.9, 1.239841939*eV/0.425, 1.239841939*eV/0.4};
    G4double scint[3] = {0.02, 1.0, 0.00};
    G4double rindexScintillator[3] = {1.58, 1.58, 1.58};
    G4double rindexWorld[3] = {1.0, 1.0, 1.0};
    G4double absl[3] = {380.*cm, 380.*cm, 380.*cm};

//Definir Mundo


    G4Material *worldMat = nist->FindOrBuildMaterial("G4_AIR");

    G4MaterialPropertiesTable *mptWorld = new G4MaterialPropertiesTable();
    mptWorld->AddProperty("RINDEX", energy, rindexWorld, 3);

    worldMat->SetMaterialPropertiesTable(mptWorld);

    G4Box *solidWorld = new G4Box("solidWorld", 0.5*m, 0.5*m, 0.5*m);

    G4LogicalVolume *logicWorld = new G4LogicalVolume(solidWorld, worldMat, "logicWorld");

    G4VPhysicalVolume *physWorld = new G4PVPlacement(0, G4ThreeVector(0., 0., 0.), logicWorld, "physWorld", 0, false, 0, true);

//Definir Material Cintilador

    G4Material *scintillator = nist->BuildMaterialWithNewDensity("scintillator","G4_PLASTIC_SC_VINYLTOLUENE", 1.023*g/cm3);

    G4MaterialPropertiesTable *mptScintillator = new G4MaterialPropertiesTable();

    //mptScintillator = scintillator->GetMaterialPropertiesTable();

    mptScintillator->AddProperty("SCINTILLATIONCOMPONENT1", energy, scint, 3);
    mptScintillator->AddProperty("RINDEX", energy, rindexScintillator, 3);
    mptScintillator->AddProperty("ABSLENGTH", energy, absl, 3);
    mptScintillator->AddConstProperty("SCINTILLATIONYIELD", 100. / MeV);
    mptScintillator->AddConstProperty("RESOLUTIONSCALE", 1.0);
    mptScintillator->AddConstProperty("SCINTILLATIONTIMECONSTANT1", 2.1 * ns);
    mptScintillator->AddConstProperty("SCINTILLATIONRISETIME1", 0.9 * ns);
    mptScintillator->AddConstProperty("SCINTILLATIONYIELD1", 1.0);

    scintillator->SetMaterialPropertiesTable(mptScintillator);


//Definir Primeiro Cintilador

    G4Box *solidScintillator1 = new G4Box("solidScintillator1", 0.1205*m, 0.05*m, 0.0028*m);

    logicScintillator1 = new G4LogicalVolume(solidScintillator1, scintillator, "logicScintillator1");

    G4VPhysicalVolume *physScintillator1 = new G4PVPlacement(0, G4ThreeVector(0., 0., 0.25*m), logicScintillator1, "physScintillator1", logicWorld, false, 0, true);

//Definir Segundo Cintilador

    G4Box *solidScintillator2 = new G4Box("solidScintillator2", 0.1205*m, 0.05*m, 0.0028*m);

    logicScintillator2 = new G4LogicalVolume(solidScintillator2, scintillator, "logicScintillator2");

    G4VPhysicalVolume *physScintillator2 = new G4PVPlacement(0, G4ThreeVector(0., 0., -0.25*m), logicScintillator2, "physScintillator2", logicWorld, false, 0, true);

    ConstructSDandField();

    return physWorld;
}


void MyDetectorConstruction::ConstructSDandField()
{
    // MySensitiveDetector *sensDet1 = new MySensitiveDetector("SensitiveDetector1");

    // logicScintillator1->SetSensitiveDetector(sensDet1);

    // MySensitiveDetector *sensDet2 = new MySensitiveDetector("SensitiveDetector2");

    // logicScintillator2->SetSensitiveDetector(sensDet2);

    mySensitiveDetector = new MySensitiveDetector("SensitiveDetector");
    logicScintillator1->SetSensitiveDetector(mySensitiveDetector);
    logicScintillator2->SetSensitiveDetector(mySensitiveDetector);

    G4SDManager* sdManager = G4SDManager::GetSDMpointer();
    sdManager->AddNewDetector(mySensitiveDetector);


}
```



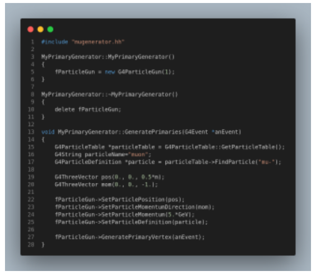
```cpp
#ifndef MUCONSTRUCTION_HH
#define MUCONSTRUCTION_HH

#include "G4VUserDetectorConstruction.hh"
#include "G4VPhysicalVolume.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4PVPlacement.hh"
#include "G4NistManager.hh"
#include "G4SystemOfUnits.hh"
#include "G4SDManager.hh"

#include "mudetector.hh"

class MyDetectorConstruction : public G4VUserDetectorConstruction
{
public:
    MyDetectorConstruction();
    ~MyDetectorConstruction();

    virtual G4VPhysicalVolume *Construct();

private:
    G4LogicalVolume *logicScintillator1;
    G4LogicalVolume *logicScintillator2;
    MySensitiveDetector* mySensitiveDetector; // Single instance of MySensitiveDetector

    virtual void ConstructSDandField();
};

#endif
```



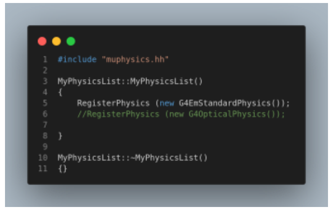
```cpp
#include <iostream>

#include "G4RunManager.hh"
#include "G4MTRunManager.hh"
#include "G4UImanager.hh"
#include "G4VisManager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "QGSP_BERT.hh"

#include "muconstruction.hh"
#include "muphysics.hh"
#include "muaction.hh"
#include "G4SDManager.hh"

#include "mudetector.hh"

#include <iomanip>

int main(int argc, char** argv)
{


  G4RunManager *runManager = new G4RunManager();

  runManager->SetUserInitialization(new MyDetectorConstruction());
  runManager->SetUserInitialization(new MyPhysicsList());
  runManager->SetUserInitialization(new MyActionInitialization());


  runManager->Initialize();

  G4UIExecutive *ui = new G4UIExecutive(argc, argv);

  G4VisManager *visManager = new G4VisExecutive();
  visManager->Initialize();

  G4UImanager *UImanager = G4UImanager::GetUIpointer();

  UImanager->ApplyCommand("/vis/open OGL");
  UImanager->ApplyCommand("/vis/viewer/set/viewpointVector 1 1 1");
  UImanager->ApplyCommand("/vis/drawVolume");
  UImanager->ApplyCommand("/vis/viewer/set/autoRefresh true");
  UImanager->ApplyCommand("/vis/scene/add/trajectories smooth");
  UImanager->ApplyCommand("/vis/scene/endOfEventAction accumulate -1");


  ui->SessionStart();


  delete runManager;


  return 0;
}
```



```cpp
#include "mudetector.hh"




MySensitiveDetector::MySensitiveDetector(G4String name) : G4VSensitiveDetector(name), totalEnergyScint1(0.0), totalEnergyScint2(0.0)
{}

MySensitiveDetector::~MySensitiveDetector()
{}



G4bool MySensitiveDetector::ProcessHits(G4Step *aStep, G4TouchableHistory *ROhist)
{


    G4Track *track = aStep->GetTrack();

    //track->SetTrackStatus(fStopAndKill);

    G4StepPoint *preStepPoint = aStep->GetPreStepPoint();
    G4StepPoint *postStepPoint = aStep->GetPostStepPoint();

    G4ThreeVector posPhoton = preStepPoint->GetPosition();

    //G4cout << "Photon position: " << posPhoton << G4endl;

    const G4VTouchable *touchable = aStep->GetPreStepPoint()->GetTouchable();

    G4int copyNo = touchable->GetCopyNumber();

    //G4cout << "Copy number: " << copyNo << G4endl;


    G4VPhysicalVolume *physVol = touchable->GetVolume();
    G4String volName = aStep->GetTrack()->GetTouchable()->GetVolume()->GetName();
    G4ThreeVector posDetector = physVol->GetTranslation();

    G4int evt = G4RunManager::GetRunManager()->GetCurrentEvent()->GetEventID();

    G4String particleName = track->GetParticleDefinition()->GetParticleName();
    G4double energy = aStep->GetTotalEnergyDeposit();



    G4cout << "Particle Name: " << particleName << G4endl;
    G4cout << "Volume name: " << volName << G4endl;
    G4cout << "Energy deposited: " << energy << G4endl;
    G4cout << "" << G4endl;




    if (volName == "physScintillator1") {
        totalEnergyScint1 += energy;
        G4cout<<"totenergy1: " <<  MySensitiveDetector::getEnergy1() << G4endl;
        G4cout << "totenergy2: " << MySensitiveDetector::getEnergy2() << G4endl;

    }
    else if (volName == "physScintillator2") {
       totalEnergyScint2 += energy;
       G4cout<<"totenergy2: " <<  MySensitiveDetector::getEnergy2() << G4endl;
       G4cout << "totenergy1: " << MySensitiveDetector::getEnergy1() << G4endl;

    }


    // G4AnalysisManager *man = G4AnalysisManager::Instance();

    // man->FillNtupleIColumn(0, evt);
    // man->FillNtupleDColumn(1, posDetector[0]);
    // man->FillNtupleDColumn(2, posDetector[1]);
    // man->FillNtupleDColumn(3, posDetector[2]);
    // man->FillNtupleDColumn(4, energy);
    // man->FillNtupleDColumn(5, totalEnergyScint1);
    // man->FillNtupleDColumn(6, totalEnergyScint2);
    // man->AddNtupleRow(0);


    return true;

}
```



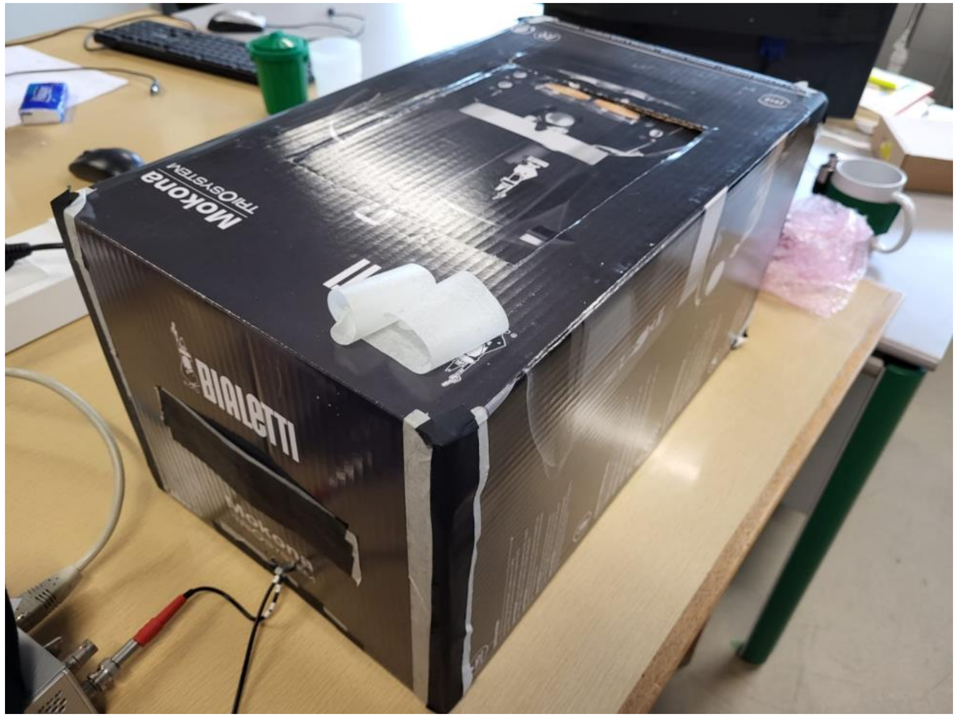
```cpp
#include "mugenerator.hh"

MyPrimaryGenerator::MyPrimaryGenerator()
{
    fParticleGun = new G4ParticleGun(1);
}

MyPrimaryGenerator::~MyPrimaryGenerator()
{
    delete fParticleGun;
}

void MyPrimaryGenerator::GeneratePrimaries(G4Event *anEvent)
{
    G4ParticleTable *particleTable = G4ParticleTable::GetParticleTable();
    G4String particleName="muon";
    G4ParticleDefinition *particle = particleTable->FindParticle("mu-");

    G4ThreeVector pos(0., 0., 0.5*m);
    G4ThreeVector mom(0., 0., -1.);

    fParticleGun->SetParticlePosition(pos);
    fParticleGun->SetParticleMomentumDirection(mom);
    fParticleGun->SetParticleMomentum(5.*GeV);
    fParticleGun->SetParticleDefinition(particle);

    fParticleGun->GeneratePrimaryVertex(anEvent);
}
```



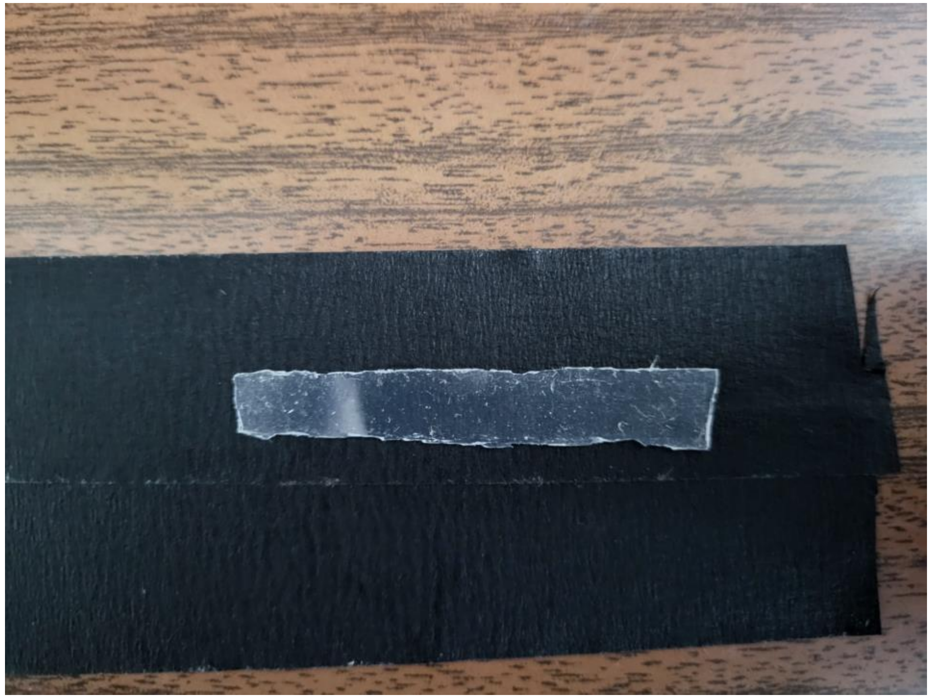
```cpp
#include "muphysics.hh"

MyPhysicsList::MyPhysicsList()
{
    RegisterPhysics (new G4EmStandardPhysics());
    //RegisterPhysics (new G4OpticalPhysics());

}

MyPhysicsList::~MyPhysicsList()
{}
```

EXPERIMENTAL SETUP

DARK BOX

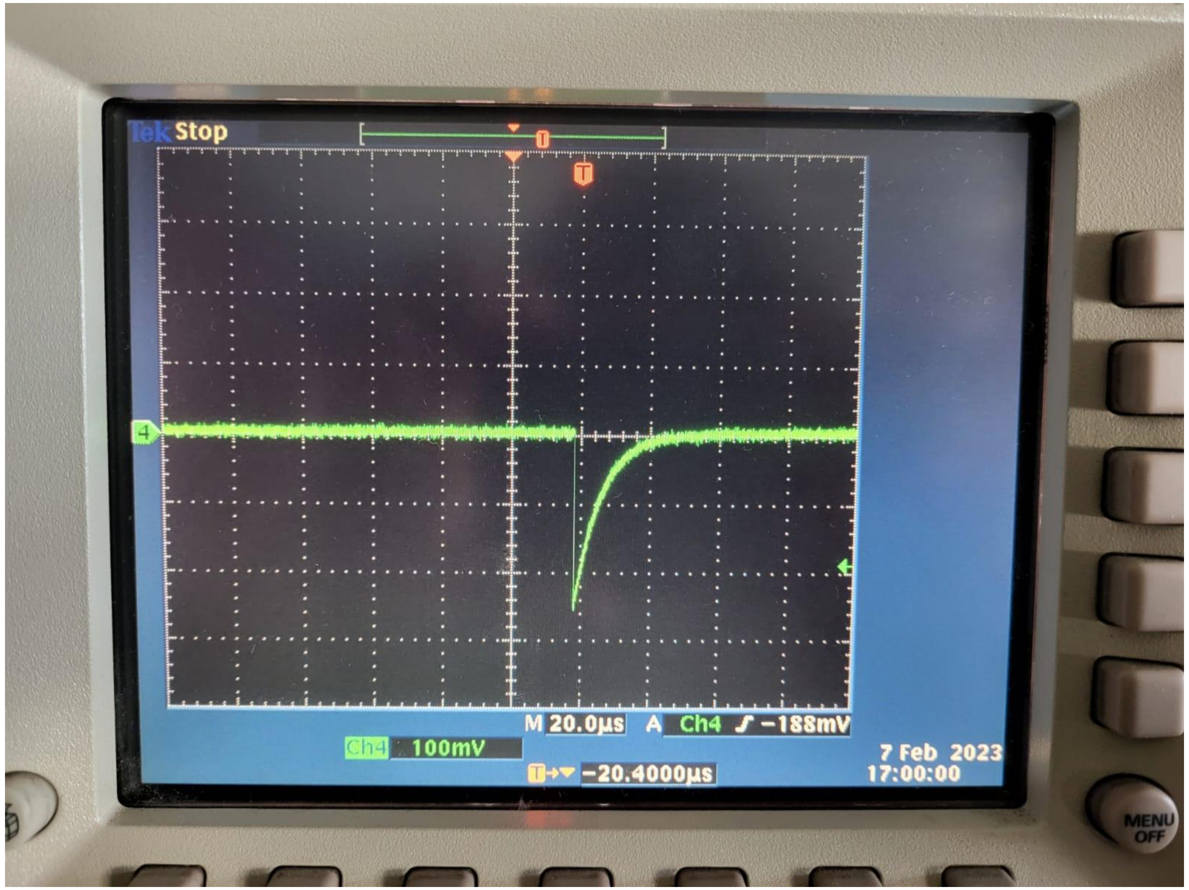


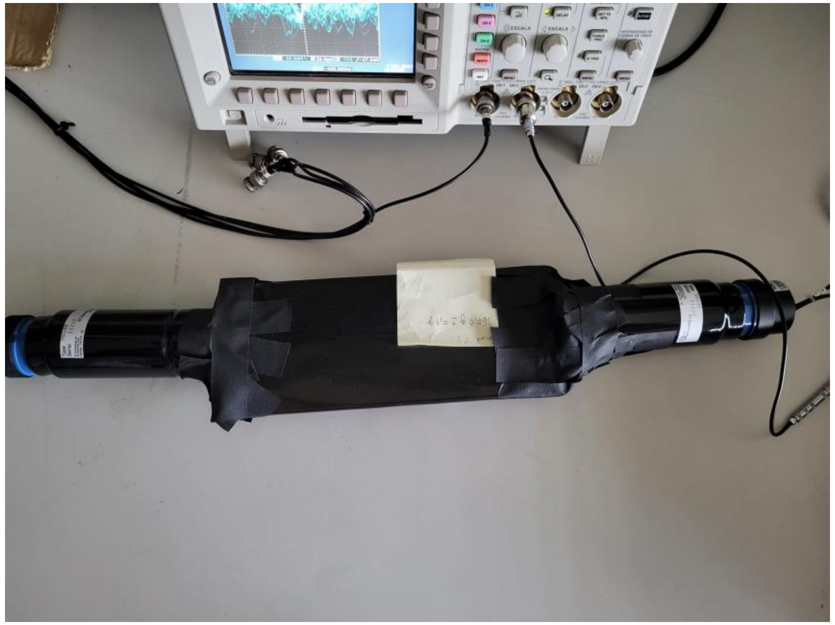
SILICONE GLUE

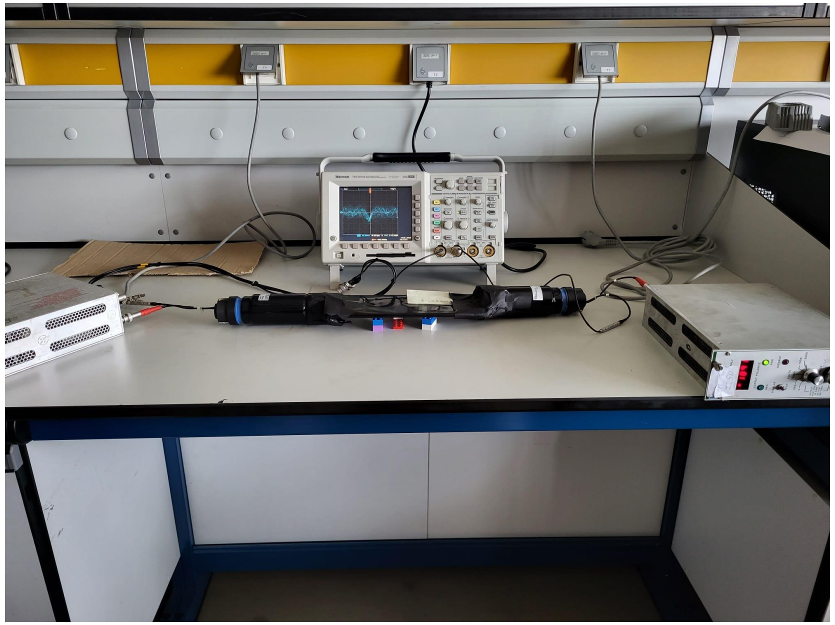


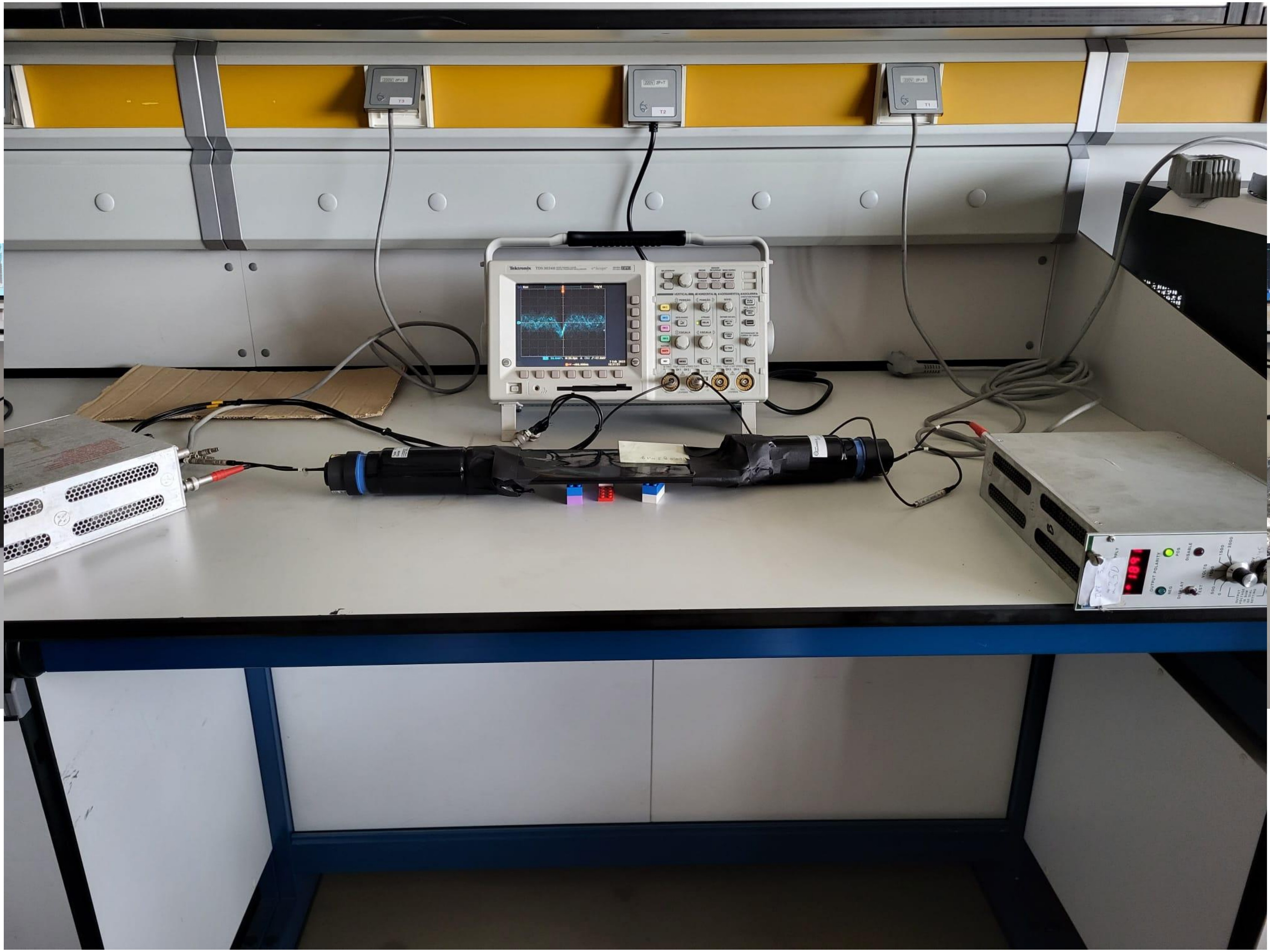
Mokona
TRIOSYSTEM

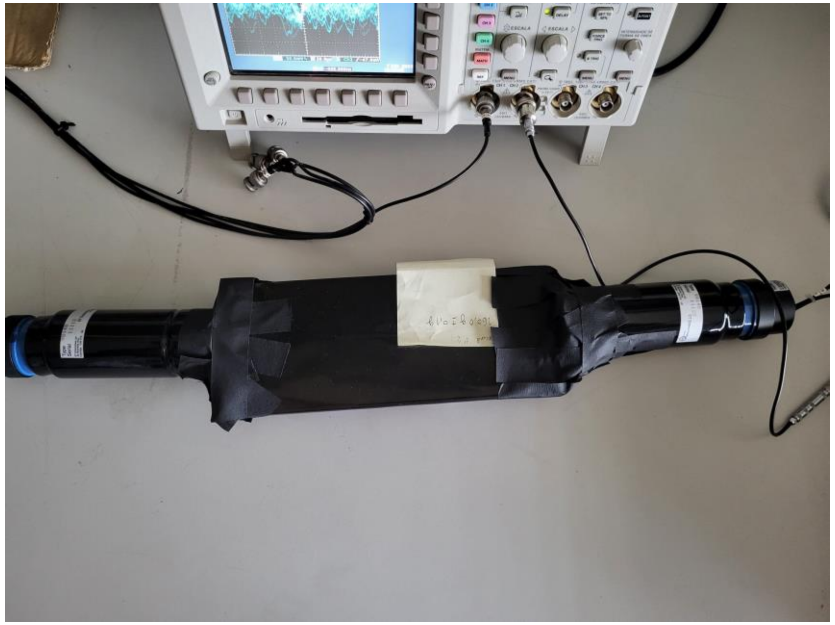
PMT SIGNAL

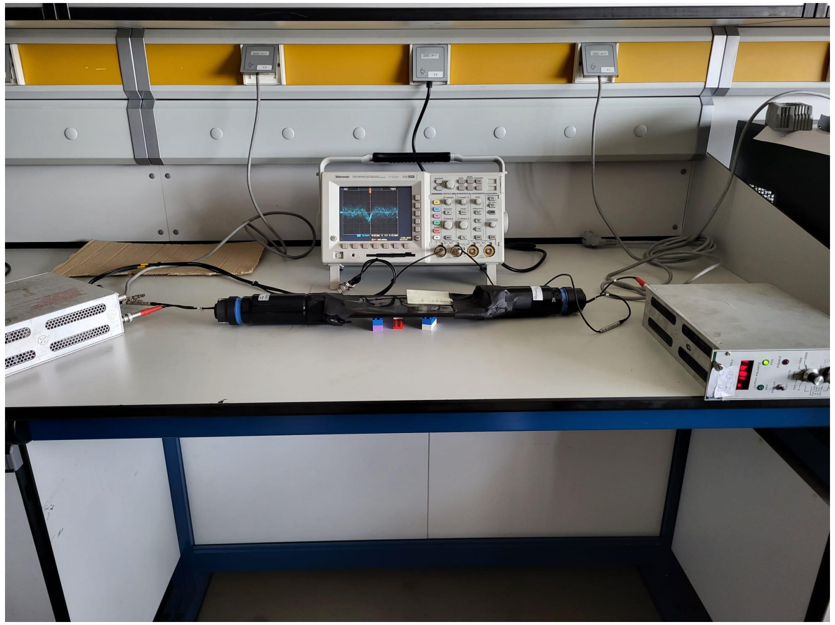

Tek Stop
4
M 20.0µs A Ch4 ƒ −188mV
Ch4 100mV
T→▼ −20.4000µs
7 Feb 2023
17:00:00
MENU OFF

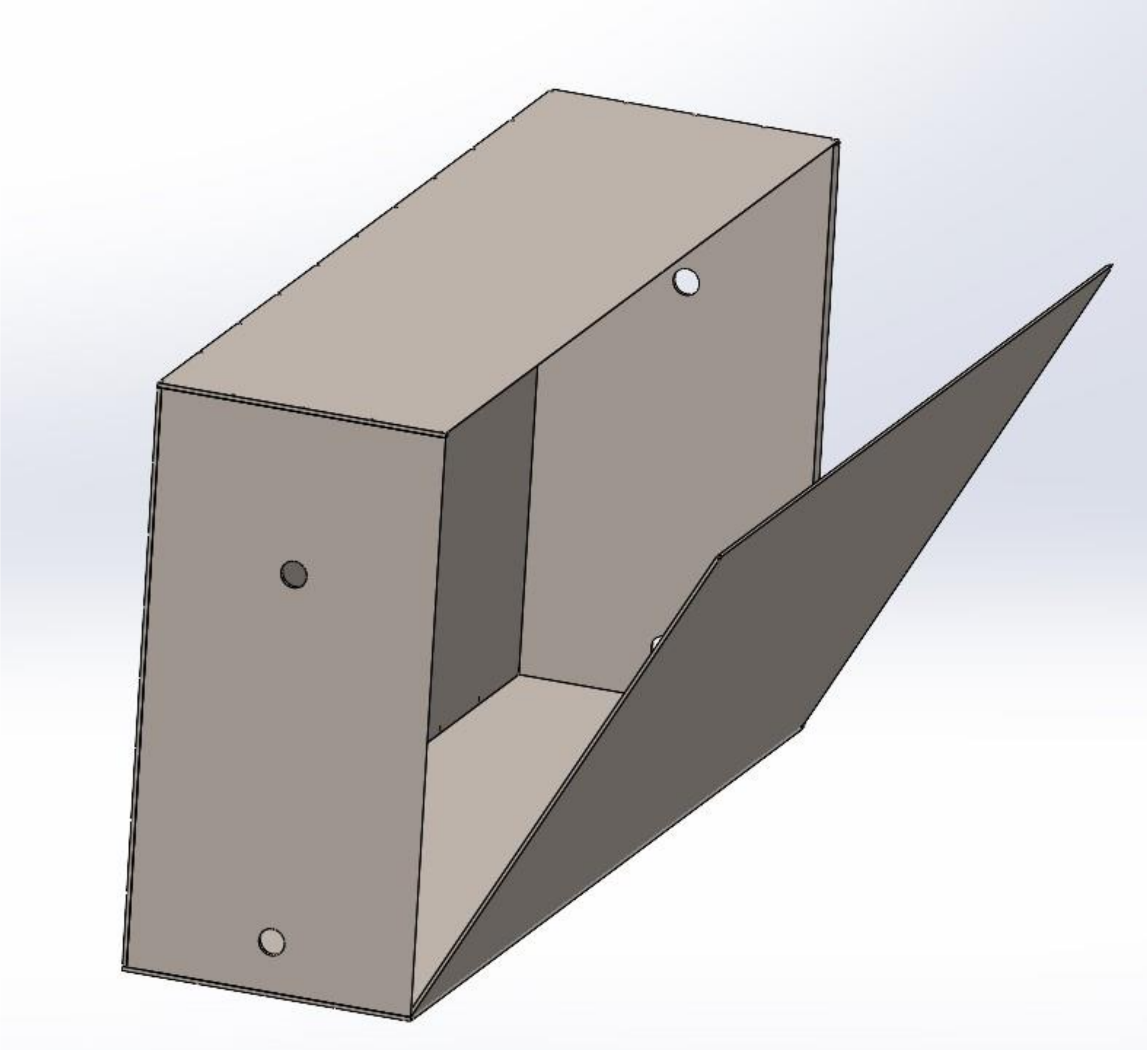












WHAT HAVE I LEARNED?

LaTeX

Pandas

TOPAS

BASH
THE BOURNE-AGAIN SHELL

CRY

Experimental Work

WHAT HAVE I LEARNED?

SOLIDWORKS

C++

ROOT
Data Analysis Framework

GEANT4
A SIMULATION TOOLKIT

FUTURE WORK

QUESTIONS?