ESTUDIO DE LA RESOLUCION ESPACIAL DE LOS DETECTORES RPC DEL EXPERIMENTO CMS

DEXY YAMILE VILLARREAL BENAVIDES

UNIVERSIDAD DE NARIÑO
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE FISICA
PROGRAMA DE FISICA
SAN JUAN DE PASTO
2009

ESTUDIO DE LA RESOLUCION ESPACIAL DE LOS DETECTORES RPC DEL EXPERIMENTO CMS

DEXY YAMILE VILLARREAL BENAVIDES

Trabajo de grado para optar el título de Físico

Director
JAIME ALFREDO BETANCURT MSc.

Codirector
JUAN CARLOS SANABRIA PhD.

UNIVERSIDAD DE NARIÑO
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE FISICA
PROGRAMA DE FISICA
SAN JUAN DE PASTO
2009

Nota de aceptación

_____

_____

_____

_____

_____

_____

Director

_____

Jurado

_____

Jurado

San Juan de Pasto, Noviembre de 2009

Las ideas y conclusiones aportadas en este trabajo de grado, son responsabilidad exclusiva del autor

Artículo primero del acuerdo No. 324 de Octubre 11 de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

AGRADECIMIENTOS

Deseo expresar mis más sinceros agradecimientos, a todas aquellas personas que hicieron posible la realización de este trabajo.

A mis padres por su amor y sacrificio, por regalarme sus sabios consejos, por creer en mí ciegamente, por apoyarme en el transcurso de mi carrera universitaria. A mi hermano por su apoyo, comprensión y amor.

Al profesor Juan Carlos Sanabria y Jaime Betancourt por confiar en mis capacidades, por su paciencia, por brindarme sus conocimientos, sugerencias y correcciones en la elaboración de este trabajo.

A los integrantes del grupo de investigación de altas energías experimental de la Universidad de los Andes y al grupo de altas energías de la universidad de Nariño por compartirme algo de sus conocimientos y experiencia.

A la Universidad de Nariño en especial al departamento de Física, por formarme como profesional.

A mis amigos y compañeros que estuvieron siempre conmigo en el transcurso de mis estudios.

DEDICATORIA

Este trabajo esta dedicado a las personas más importantes de mi vida: Mis padres Roman y Susana, quienes con su ejemplo y dedicación me han demostrado que los sueños se pueden convertir en realidad.

A Mi hermano Holmer por brindarme su apoyo, comprensión y amistad.

A Mis tíos.

RESUMEN

El experimento CMS (Compact Muon Solenoid) es uno de los detectores multipropósito del gran colisionador de protones LHC (Large Hadron Collider); Uno de los componentes esenciales en el detector CMS es el sistema de detección de muones y dentro de él las cámaras RPCs. Debido a que los muones interactúan con los materiales físicos del detector CMS habrá producción de partículas dentro del mismo las cuales afectaran la resolución espacial del detector. En este trabajo se determinó el efecto que tiene la presencia de barras de aluminio en frente de las cámaras RPC y se investigó el efecto que producen los rayos delta en la producción de señales alejadas de la trayectoria original por donde el muon incidente atraviesa la RPC; para ello se llevó acabo una simulación del paso de muones a través de una RPC usando el código de Monte Carlo GEANT4 en el cual se a implementado un modelo realista de una cámara RPC del detector CMS.

ABSTRACT

The Compact Muon Solenoid (CMS) experiment is one of large detectors on the proton-proton LHC. One of the components of CMS detector is the muon system inside of Resistive Plate Chambers RPCs. Due the muones act whith other physics materials of CMS detector, there will be production of particles inside of itself which will change the space resolution from RPCs. This study determined the effect of the presence of aluminum bars in front of the chambers RPCs and investigated the effect produced by delta rays in the production of signals away from the original path where the incident muon transverses the RPC. In order to do this study were simulated muones traversing an RPC whit a block Al in front, using the GEANT 4 Montecarlo code in which it has implemented a realistic model of a RPC chamber of the CMS detector.

TABLA DE CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

Pág.

GLOSARIO

ALICE: Experimento que estudiara las colisiones de iones pesados en el acelerador LHC.

ATLAS: Experimento de propósito general en el LHC que estudiara la física tanto de las nuevas partículas y las interacciones implicadas.

CERN: Organización Europea para la investigación nuclear.

CMS: El Compact Muon Solenoide, es uno de los detectores de propósito general del acelerador LHC diseñado especialmente para detectar muones, cuyo objetivo principal es detectar el bosón de Higgs.

GEANT: Paquete de software de libre distribución capaz de simular los procesos dominantes que gobiernan las interacciones de las partículas e iones con la materia, en un rango que van desde keV a TeV. Este código Monte-Carlo esta escrito en C++, utiliza avanzadas técnicas de software e ingeniería y tecnología orientada a objetos.

LHC: Acelerador de partículas que hace colisionar dos haces de protones cada uno con una energía de 7 TeV, alcanzando una energía en el centro de masa de 14 TeV y una luminosidad de $10^{34} cm^{-2} s^{-1}$.

LHC-B: experimento del LHC que estudiara la física del quark b analizando la violación CP para estas partículas.

MODELO ESTÁNDAR DE PARTÍCULAS ELEMENTALES: Es una teoría gauge local que describe las interacciones fuertes y electrodébiles en un rango de energía de unos cientos de GeV; tiene una extraordinaria precisión al explicar la mayoría de las observaciones experimentales realizadas en procesos de partículas elementales.

RPC: Las Cámaras de Placas Resistivas son detectores gaseosos que combinan una buena resolución espacial y temporal.

1. INTRODUCCIÓN

Este documento muestra los resultados obtenidos de la simulación hecha con el código de Montecarlo Geant4 cuando algunas partículas cargadas (muones) pasan a través de una cámara de placas resistivas (RPC) y como la presencia de algunos materiales del detector afectan la resolución espacial del mismo. Las RPCs hacen parte del sistema de detección de muones del detector CMS en el gran colisionador de protones LHC (Large Hadron Collider). El objetivo principal de este acelerador es encontrar el bosón de Higgs con el cual se puede dar respuesta al origen de la masa o en su defecto encontrar nuevas pistas que permitan una generalización del Modelo Estándar de Partículas Elementales. Para conseguir este gran objetivo LHC utiliza cuatro grandes detectores ALICE, CMS, ATLAS y LHCb, dos experimentos más pequeños también hacen parte de este TOTEM y LHCf. Cada uno de estos detectores tiene una misión en especial por lo que se diferencian en su diseño y caracterización.

CMS es un detector de propósito general diseñado para medir la energía y el momentum de partículas tales como electrones, muones, fotones, mesones, etc. que resulten de la colisión protón-protón, pero dentro de su diseño se ha enfatizado en la detección de muones [1], debido a que Uno de los canales predichos para el decaimiento del bosón de Higgs tienen como subproductos, muones. Los muones se caracterizan por perder poca energía al atravesar materiales, por lo tanto se pueden detectar en regiones en las que otras partículas no alcanzan a llegar. Teniendo en cuenta esta característica, los detectores de muones se encuentran en las partes exteriores de CMS.

Para identificar los muones y medir su momento CMS emplea tres tipos de sub-detectores los tubos de deriva, las cámaras catódicas y las cámaras de placas resistivas. Las cámaras de placas resistivas RPCs, son muy susceptibles al paso de muones en un amplio radio y son utilizadas en experimentos de altas energías debido a su alta eficiencia y velocidad de respuesta.

En el proceso de detección de muones se producen electrones, fotones, rayos delta etc. Los rayos delta ó electrones altamente energéticos dentro de las cámaras RPCs pueden provenir de la interacción de muones con los materiales físicos de CMS tales como barras de aluminio que se encuentran enfrente de las cámaras RPCs. Los muones producidos en la colisión ionizan átomos y los electrones resultantes de esta ionización pueden alcanzar energías suficientes para producir ionizaciones de átomos cercanos . Estos electrones libres no se distinguen de los electrones producidos por el paso de muones y pueden llevarnos a obtener señales equivocas alejadas de la zona por donde el muon a travesó la RPC. Por otro lado lo anterior se debe a que los productos de la colisión protón-protón pueden ser tan energéticos que al impactar sobre los componentes del detector darán origen a más

[1] CMS Collaboration. CMS Thechnical Design Report: The Muon Proyect. CERN/LHC

partículas y estas a su vez, pueden repetir el proceso aumentando el número de partículas en un fenómeno conocido como cascada.

Para estudiar el efecto que produce la presencia de barras de aluminio en frente de las cámaras RPCs e investigar el efecto que producen los rayos "delta"en la producción de señales alejadas de la trayectoria original de un muon incidente, se llevó acabo una simulación del paso de muones a través de una RPC doble-gap como las que se han implementado en el CMS usando el software Geant4. En la simulación se trabajaron dos tipos diferentes de RPCs, la primera sin presencia de barras de aluminio y la segunda con presencia de barras de aluminio en frente, tomando en cuenta todos los procesos físicos que se presentan y con el fin de determinar el efecto que tiene la presencia de barras de Aluminio en la resolución espacial de las cámaras RPCs del experimento CMS.

En el presente trabajo en el capitulo 2, se realiza una breve descripción de la geometría, diseño y sub-detectores de CMS especialmente el sistema detector de muones. En el capitulo 3 se describe la física de los detectores de ionización gaseosos, también se hace una breve descripción de los fundamentos de las RPCs y finalmente se describe los detectores de placas resistivas RPC del CMS, teniendo en cuenta su funcionamiento y como estas se encuentran distribuidas dentro de él. En el capitulo 4 se describe la simulación implementada en el software Geant4, dando a conocer brevemente el software utilizado, luego se describe la geometría de la RPC doble-gap implementada y la física que se tuvo en cuenta para la simulación. En el capitulo 5 se presentan los resultados obtenidos y se realiza una discusión de los mismos. En el capitulo 6 se encuentran las conclusiones.

2. SISTEMA DE MUONES DEL DETECTOR CMS

El Compact Muon Solenoid (CMS), es un detector de propósito general para la física de partículas, construido en el Large Hadron Collider (LHC) en CERN cerca de Ginebra, en la frontera Franco-Suiza, una vista aerea del acelerador LHC se muestra en la figura 1.En este capítulo se hace una descripción general de la geometría del detector CMS y sus componentes, en especial se describe el sistema de detección de muones.

Figura 1. Vista aerea del acelerador LHC

2.1. El CMS

El Compact Muon Solenoid es un detector que investigara la física a escala energética de Teraelectronvolts, incluye la búsqueda del Boson de Higgs, búsqueda de física mas allá del modelo estándar, como supersimetría, dimensiones extras y partículas que podrían constituir la materia oscura.[2]

Con el fin de obtener estos resultados científicos, CMS colocó ciertos requisitos en su diseño:

[2]Compact Muon Solenoide, Disponible en: http://lhc.web.cern.ch/cms/.

• Un sistema de alto rendimiento para detectar y medir muones

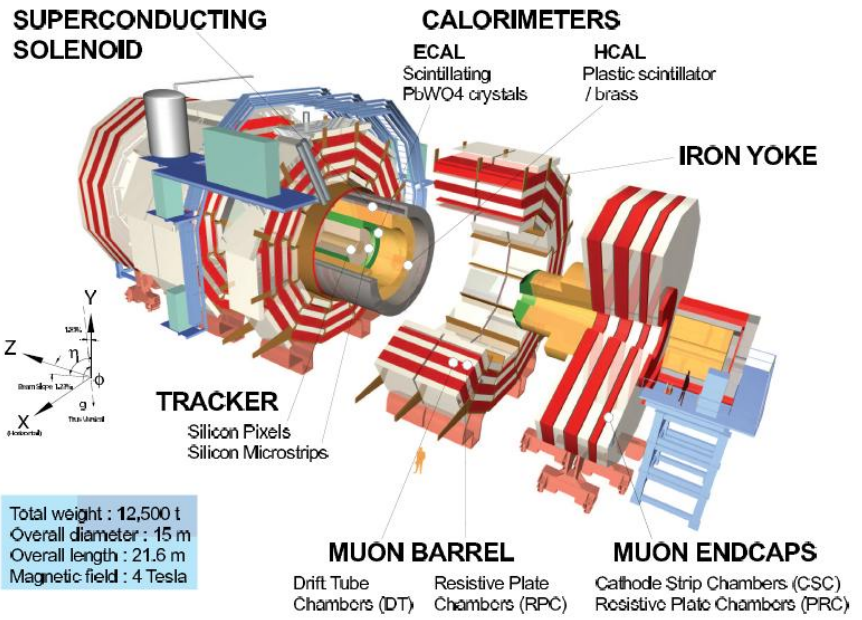
• Un método de alta resolución para detectar y medir electrones y fotones (calorímetro electromagnético).

• Alta calidad del sistema central tracking para dar mediciones precisas de momentum y un hermético calorímetro hadronico, diseñado para rodear por completo la colisión y evitar que escapen las partículas.

El detector CMS tiene la forma de un cilindro alrededor de uno de los puntos de colisión del LHC. El detector esta dividido en una región central o "barril" y dos "tapas".[3]

La estructura general de CMS se muestra en la figura 2. Tiene 15 m de diámetro, 21.6 m de largo y 12500 toneladas de peso. Dentro del detector se encuentra el solenoide superconductor que proporciona un campo magnético de 4 teslas.

Figura 2. Detector CMS

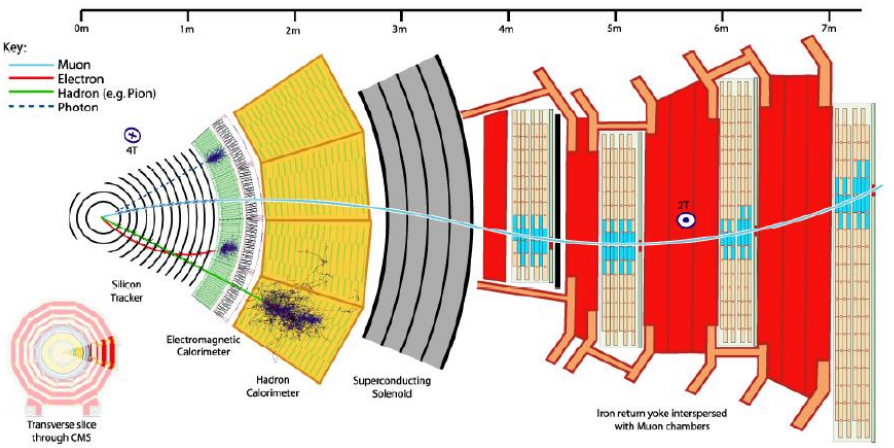


El campo magnético es uno de los elementos esenciales en CMS, sirve como herramienta para identificar partículas cargadas, ya que estas deflectan su trayectoria en presencia del campo. Cuanto mayor sea el momentum de una partícula cargada, menos se curva su trayectoria en el campo magnético, de modo que si se conoce su trayectoria se puede medir su momentum (Ver figura 3). Es por esto que el detector más interno de CMS

[3]CMS Collaboration. CMS Thechnical Design Report: The Muon Proyect. CERN/LHC.

conocido como tracker se encarga de reconstruir las trayectorias de las partículas producidas en la colisión. Para determinar la energía se utiliza los calorímetros Electromagnético y Hadronico. Estos detectores detienen las partículas y permiten medir la energía depositada por estas al ser detenidas. Los calorímetros y el tracker se encuentran al interior del solenoide. Fuera de este están las cámaras de muones. Todo el dispositivo se encuentra soportado por una estructura de hierro llamada yugo que también sirve para desviar el campo magnético fuera del solenoide.[4]

Figura 3. Detección de particulas en el CMS.



Como se describió anteriormente el detector consiste de diferentes sub-detectores, cada uno con unas propiedades de medición determinadas para cada evento físico. Todas las medidas son realizadas tan exactamente como sea posible y con la mínima interrupción de los procesos subyacentes de la física. Cada sub-detector visto en la figura 3 será discutido de forma general dentro de esta sección.

2.1.1. Tracking: El Tracker es el dispositivo encargado de obtener las trazas de las partículas producto de la colisión para asociarlas con señales obtenidas por detectores más externos. El tracker se divide en tres sub-detectores: el sistema de detectores de silicio, las microcintas de silicio y las microcamaras de gas. [5]

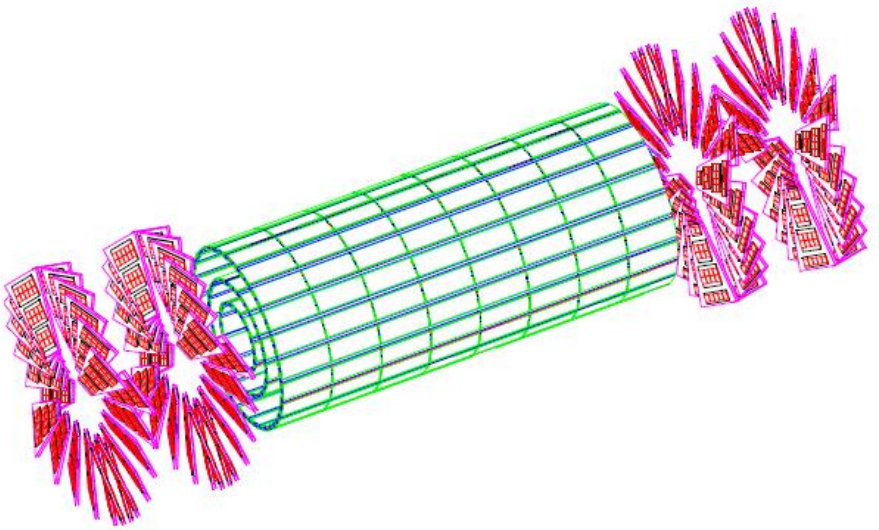
Los pixeles son detectores de estado sólido, se colocan lo más cerca posible al punto de interacción donde el flujo de partículas es alto. En la figura 4 se muestra la distribución

[4]Matthew Wingham. CMS tracker and preparing for Early physics at the LHC,CERN-THESIS-2008-081.

[5]CMS Collaboration.CMS Physics Thechnical Design Report. Vol I: Detector Performance and Software. CERN/LHC.

de los detectores de pixeles para el Tracking en la zona del Barril y en las Endcaps.[6]

Figura 4. Detectores de pixeles para el barril se muestran en verde, para las encaps se muestran en rojo.



2.1.2 El Calorímetro Electromagnético: La tarea del calorímetro electromagnético (ECAL) es medir la energía de los fotones, electrones y positrones que resulten de la colisión protón-protón. Con esto se intenta identificar el bosón de Higgs, cubriendo los canales que resulten en decaimientos lepto-nicos o en fotones.[7]

Los componentes principales del calorímetro electromagnético son los cristales centelladores de tungstenato de plomo ($PbWO_4$). En el barril cada cristal esta conectado a un fotodiodo de avalancha, en el caso del endcap los cristales están conectados a tríodos de vacio.

El barril se compone de 61.200 cristales y las endcaps de casi 15.000, para mayor precisión espacial el calorímetro electromagnético también contiene el detector "presho-wer"en la región de ángulos pequeños que tiene como función principal distinguir los dos fotones producto del decaimiento de mesones π^0 de otros fotones.[8]

2.1.3 El Calorímetro Hadronico: El Calorímetro Hadronico (HCAL) mide la energía de los Hadrones por ejemplo, los protones, neutrones, piones y kaones. Además, proporciona

[6]Matthew Wingham. CMS tracker and preparing for Early physics at the LHC,CERN-THESIS-2008-081.

[7]Compact Muon Solenoide. Disponible en: http://lhc.web.cern.ch/cms/.

[8]Compact Muon Solenoide. Disponible en: http://lhc.web.cern.ch/cms/.

una medición indirecta de la presencia de partículas sin carga, como los neutrinos.

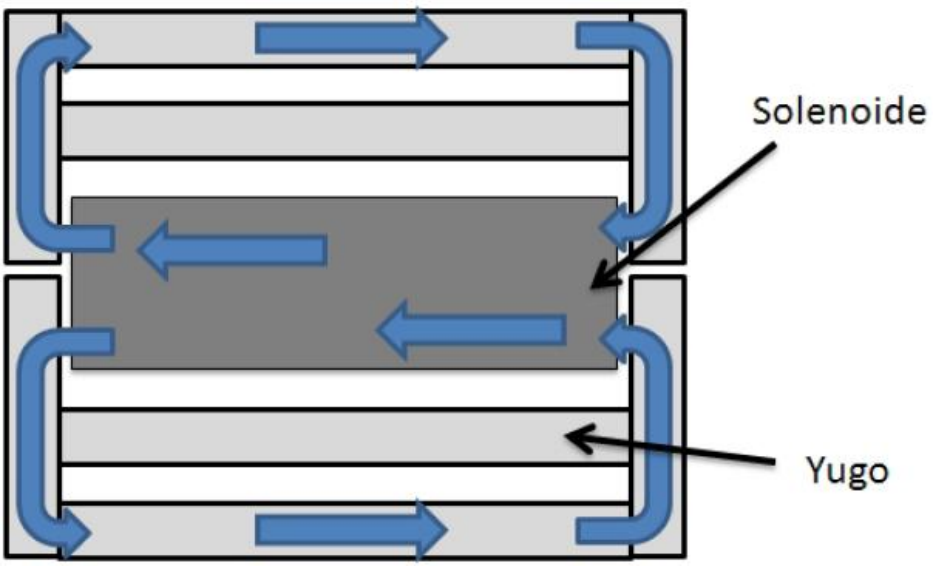
La medición de estas partículas es importante, ya que nos puede decir si las nuevas partículas como el bosón de Higgs ó partículas supersimétricas (mucho más pesada que las versiones estándar de las partículas que conocemos) se han formado.

Este se divide en cinco piezas importantes: el barril, los endcaps y los calorímetros hadronicos frontales. Los detectores consisten en varias placas de aleación de cobre y zinc de espesores de alrededor de 5 cm. En los espacios entre placas y placas se encuentran lozas de un polímero que sirve de material centellador, con un espesor de alrededor de 4 cm.

2.1.4 Solenoide y Yugo: El campo magnético del CMS es producido por un magneto solenoidal de radio 3 m y 14 m de largo en cuyo interior se encuentra el sistema de trazas y los calorímetros electromagnético y hadronico, produce un campo magnético de 4 Tesla. La corriente electrica es transmitida por un entramado de cables hechos de aleación de niobio-titanio con un recubrimiento de aluminio, formando una bobina que se encuentra sumergida en helio líquido, dentro de un criostato contenedor de acero inoxidable. El solenoide funciona en régimen superconductor.

En la parte exterior del solenoide se encuentra un yugo que consiste en un armazón de hierro que además sirve para recolectar las líneas externas de campo magnético, buscando que la intensidad del campo magnético en la región en la cual se encuentran los detectores de muones sea lo mas alta posible. La figura 5 nos muetra la distribución de las líneas de campo magnético desviadas por el yugo.

Figura 5. Líneas de campo magnético desviadas por el yugo.



Para construir el yugo se usan estructuras de hierro dispuestas de modo tal que conforman anillos con doce lados, la masa total del yugo es 3050 toneladas. Los endcaps del yugo son cuatro estructuras de hierro con forma dodecagonal que alcanzan el mismo diámetro que el barril y tienen espesores entre los 10 cm y los 65 cm. Incrustadas entre el yugo se encuentran las cámaras de muones.

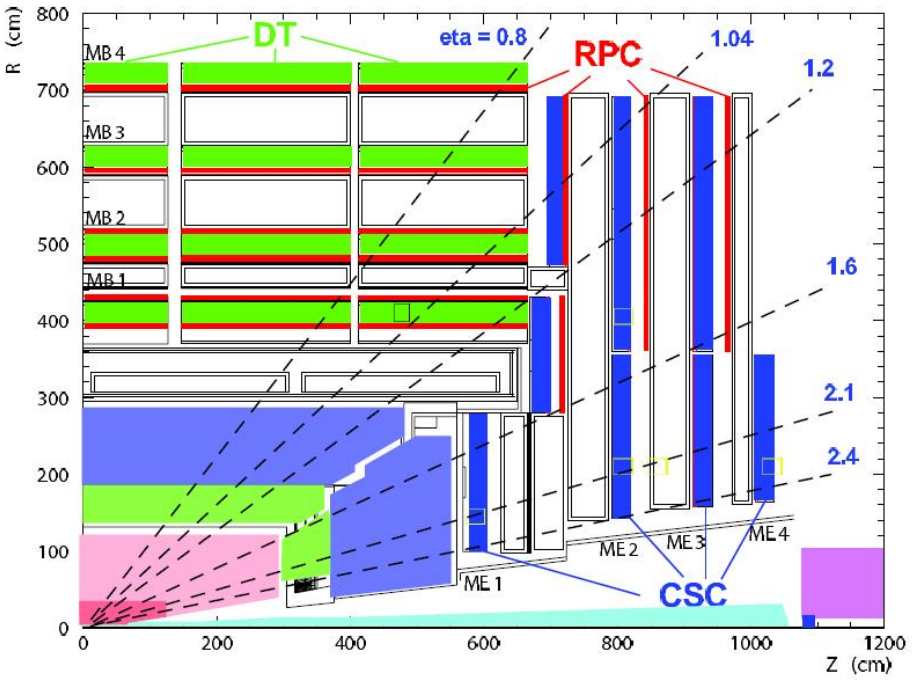
2.1.5 Sistema de Muones: Para identificar los muones y medir su momentum, el detector CMS emplea tres tipos de detectores: los tubos de deriva, las cámaras catódicas y las cámaras de placas resistivas. En total hay 1.400 cámaras de muones: 250 tubos de deriva (DT) y 540 cámaras catódicas (CSC), encargadas de realizar mediciones de la trayectoria de muones y dar una señal de activación, mientras que 610 cámaras de placa resistiva (RPC) forman un sistema redundante de activación del tracker, que rápidamente deciden almacenar los datos adquiridos o no. Debido a las muchas capas de detectores y diferentes especialidades para cada tipo, el sistema es naturalmente robusto y capaz de filtrar el ruido de fondo. En la sección 2.2 se realizará una descripción de estos detectores.

2.2 EL SISTEMA DE MUONES DEL DETECTOR CMS

El experimento CMS esta diseñado especialmente para detectar muones, debido a que estas partículas son los únicos productos de la colisión que alcanzan el exterior del solenoide por tal razón resultan fáciles de distinguir. Debido a que los muones son partículas mínimo ionizantes, y son 200 veces mas pesado que un electrón, su pérdida de energía por bremsstralung es muy pequeña. Los muones pueden atravesar los calorímetros y alcanzar las regiones exteriores al solenoide. Por tal razón en la parte exterior de CMS se han colocado los detectores de muones, para hacer un seguimiento de su trayectoria y medir su momentum. Para la detección de muones se utilizan tres tipos de dispositivos. Los tubos de deriva, las cámaras catódicas y las cámaras de placas resistivas. En la región del barril se combinan tubos de deriva DT y cámaras de placas resistivas RPCs, estos están dispuestos en forma de cilindros concéntricos alrededor de la línea del haz. En los endcaps se colocan las cámaras catódicas CSC y las cámaras de placas resistivas RPCs, las cuales conforman discos que abarcan los extremos del barril.[9]

[9]CMS Collaboration.CMS Physics Thechnical Design Report. Vol I: Detector Performance and Software. CERN/LHC.

Figura 6. Esquema de un cuadrante del sistema de muones. Drift Tubes (DT), Resistive Plate Chambers (RPC) y Cathode Strip Chambers (CSC).



2.2.1 Tubos de Deriva: Los tubos de deriva son cavidades de 42 x 13 mm^2 de frente y entre 2 y 3 metros de largo. Las paredes son de aluminio de 1.5 mm de espesor, formadas por dos varillas en forma de I (Catodos) y que soportan dos placas. En el centro hay un alambre de acero de 50 micrómetros de diámetro que sirve de ánodo. Se llena con una mezcla de Ar y CO2. Los voltajes colocados son de 3600 V en el ánodo, 1800 V en las placas de aluminio y -1200 V en las varillas. El modulo completo tiene 2.5 metros de ancho con varias capas de tubos de deriva. El tiempo máximo de deriva hacia el alambre es de 400 ns.[10]

Los 250 tubos de deriva se organizan en 4 capas (estaciones MB1, MB2, MB3 Y MB4). Cada una se divide en 12 mecanismos, dispuestos en tres grupos de cuatro, cada uno con hasta 60 tubos: el grupo intermedio mide las coordenadas en la dirección paralela al haz y los dos grupos exteriores miden las coordenadas perpendiculares.

Cada tubo de deriva DT tiene 1 o 2 RPCs acopladas a él dependiendo de la estación. En las estaciones MB1 y MB2, cada paquete consiste de 1 tubo de deriva DT en medio de 2 RPCs. En las estaciones MB3 y MB4, cada paquete comprende 1 tubo de deriva DT y 1

[10]Matthew Wingham. CMS tracker and preparing for Early physics at the LHC,CERN-THESIS-2008-081.

RPC colocado en la parte mas interna de la estación.

2.2.2 Cámaras catódicas: Las cámaras catódicas tienen una forma trapezoidal como lo muestra la figura 7 cada cámara consisten en una serie de alambres paralelos colocados a 3mm uno del otro.

Figura 7. Esquema de una CSC

Estos alambres están agrupados en paquetes y todo se encuentra encapsulado entre dos placas que sirven de cátodos. Una de estas placas esta formada por cintas que se extienden perpendicularmente a los alambres. Los cátodos consisten en placas de un agregado de fibra de vidrio y cobre. La distancia entre las placas es de 9.5 mm. La cámara completa consiste de siete placas entre las cuales se introduce el gas. El gas es una mezcla de 40 % de Ar, 50 % de CO_2 y 10 % de CF_4.

Hay 468 CSCs en los 2 Endcaps de CMS. Cada Endcap consiste de 4 estaciones de cámaras, comenzando desde ME1 a ME4 en orden de distancia creciente desde el punto de la colisión.

2.2.3 Cámaras de Placas Resistivas RPCs: Las cámaras de placas resistivas combinan una buena resolución espacial (30 μm) con una resolución temporal comparable a los detectores de centelleo (ns). Además son indiferentes ante el intenso campo magnético en el que se encuentran. Por otro lado son detectores de bajo costo y fácil ensamblaje. Debido a su rápida respuesta permiten asociar la trayectoria de un muon con el cruce de haces que lo originó. La sección 2.3 profundiza en la geometría de estas.

2.3 CÁMARAS DE PLACAS RESISTIVAS RPCs

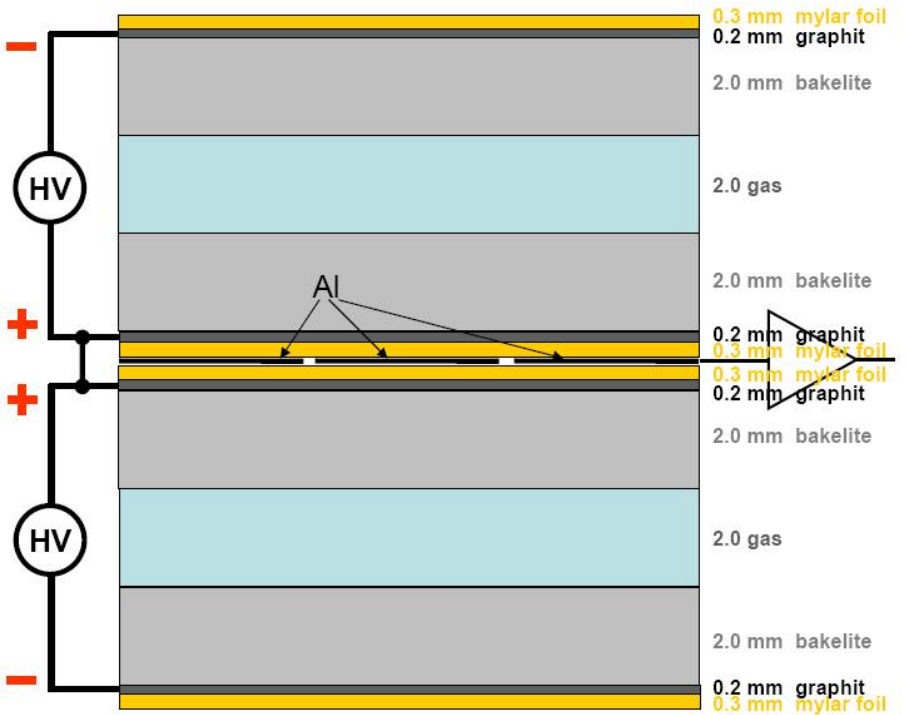
Las cámaras de placas resistivas (RPCs) son detectores gaseosos de placas paralelas, que

combinan una buena resolución espacial (30 μm) y temporal comparable con los detectores de centelleo. Estas se han desarrollado para responder rápidamente al paso de una partícula, dar información de espacio-tiempo como es requerido por el muon trigger en el experimento LHC.

Una RPC consiste en dos placas paralelas, hechas de una resina fenólica (bakelita) la cual es altamente resistiva de 10^{10} - 10^{11} Ωcm, separadas por un gap de gas de unos pocos milímetros. El espesor de las placas es de 2 mm y el espacio entre ellas es también 2 mm. El gap se llena con una mezcla de gas, el 95 % de Tetra-Fluoro-Etano ($C_2H_2F_4$), el 5 % de Iso-Butano ($i - C_4H_{10}$). Este gas no inflamable es escogido debido a que el ancho de la acumulación de cargas es pequeño (tamaño del cluster).

La estructura general esta encerrada de gas. Las superficies exteriores del material resistivo están cubiertas por un material conductor Grafito de 0.02 mm de espesor que sirve para distribuir el campo eléctrico, a este se conecta el HV haciendo la una placa de electrodo y la otra de tierra. Por encima del grafito se coloca una película de polietileno de 0.1 mm de espesor. La información es obtenida por medio de las strips de aluminio.

Figura 8. RPC de doble gap.



Para incrementar la señal obtenida por medio de las strip se aumenta el número de cámaras. Para lograr esto se colocan las cámaras en una distribución conocida como dis-

tribución de doble gap, como se muestra en la figura 8. En este caso el paso de un muon simultáneamente activa dos cámaras en lugar de una sola. El tamaño de una cámara total armada es de 256 cm de ancho, con una longitud entre 2 a 4m y una altura de 5,5 cm.

Una RPC es capas de etiquetar el tiempo de una ionización incluso en tiempos tan pequeños del orden de 25 ns entre dos sucesos bunch crossings (BX). Por tal razón un muon veloz que activa el trigger del detector basado en las RPCs puede ser identificado inequívocamente. Señales de tales detectores proporcionan directamente el tiempo y posición del paso de un muon con la exactitud requerida.[11]

[11]CMS Collaboration.CMS Physics Thechnical Design Report. Vol I: Detector Performance and Software. CERN/LHC.

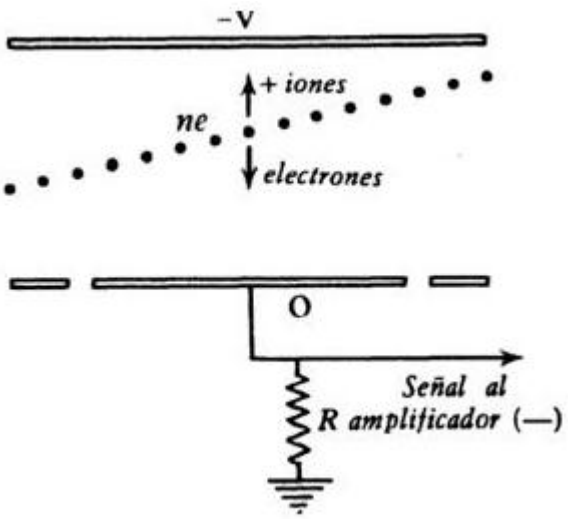
3. DETECTORES DE PLACAS RESISTIVAS RPCs

Una RPC es un detector de ionización gaseoso formado por dos placas resistivas paralelas que actúan como electrodos, entre los se aplica una diferencia de potencial constante, de manera que aparece un campo eléctrico uniforme. El espacio entre electrodos o gap se llena con una mezcla gaseosa adecuada para producir una señal rápida y limpia, al paso de una partícula ionizante. Dentro de este capitulo se describe la física de los detectores de ionización gaseosos, luego se describe los detectores de placas resistivas y finalmente se describe en detalle las RPCs del CMS.

3.1. FÍSICA DE LOS DETECTORES DE IONIZACION

Los detectores de ionización son dispositivos diseñados para medir la ionización producida cuando una partícula atraviesa un material. El objetivo del detector es la colección directa de electrones e iones producidos en un fluido (gas o líquido) por el paso de la radiación, para producir un pulso eléctrico como señal de salida (esta señal de salida es proporcional a la energía de radiación incidente). Si la radiación penetra en el detector se crea un cierto número de pares electrón-ion. El número promedio de pares creados es proporcional a la energía depositada en el detector de ionización. Uno de los componentes básicos de un detector es el campo eléctrico debido a que bajo la acción de este los electrones se aceleran hacia el ánodo y los iones se recolectan en el cátodo.[12]

Figura 9. Detector de ionización.



[12]W.R. Leo. Thechniques for Nuclear an Particles Physics Experiments, Springer-Verlag.

El modelo más básico de un detector de ionización (Ver figura 9), consiste en un contenedor lleno con un material fácil de ionizar. Sin embargo este material debe ser químicamente estable para evitar que los electrones producidos por la ionización sean nuevamente capturados por las moléculas del material. El detector también debe tener un ánodo y un cátodo que se mantienen a una cierta diferencia de potencial (V) como se muestra en la figura 9. Además el material debe ser poco sensitivo a daños con radiación, es decir que su respuesta a partículas ionizantes no cambie mucho con el tiempo y adicionalmente debe poseer una baja energía de ionización para maximizar la tasa de ionización producida por energía depositada por una partícula incidente. Cuando una partícula atraviesa el detector ioniza el medio y se producen pares electrón-ion, los electrones viajan hacia el ánodo y los iones positivos hacia el cátodo induciendo señales en los electrodos que producen pequeñas corrientes que son detectadas por un amplificador (A). El pulso que produce el amplificador puede ser ana-lizado para relacionarlo con la cantidad de ionización producida por la partícula incidente.

Los procesos físicos que permiten determinar que tipo de partícula pasó por el detector son diferentes para partículas neutras o cargadas. A energías mayores a 10 MeV, los fotones pueden interactuar con la materia por efecto Fotoeléctrico, efecto comptom o por creación de pares. Los positrones y electrones resultantes se pueden detectar por su interacción electromagnética. Los procesos físicos mas comunes que se presentan en un detector de ionización son: excitación de un átomo, ionización la más importante, efecto Penning, creación de pares electrón-ion, recombinación, electrón adjunto, difusión, deriva, movilidad.

3.2 FÍSICA DE LOS DETECTORES DE PLACAS RESISTIVAS RPCs

Las RPCs son detectores gaseosos compuestos por dos placas paralelas que actúan como electrodos, entre los que se aplica una diferencia de potencial constante, de modo que se produce en el espacio entre electrodos (gap) un campo eléctrico uniforme. Al menos uno de estos electrodos tiene que ser de un material con una resistividad característica (alta, pero no excesivamente), que es lo que dió origen al nombre de "placas resistivas". El gap, cuyo rango varía entre centenas de micras y pocos milímetros, dependiendo de la aplicación. Los electrones y los iones creados por una partícula cargada que atraviese el detector son acelerados hacia el ánodo y el cátodo respectivamente. Cuando la ionización primaria adquiere suficiente energía ioniza otras moléculas del gas y se crean electrones secundarios. Estos nuevos electrones son acelerados de nuevo, originando una avalancha.

El número total de electrones creados en un cierto recorrido x es:

$$n = n_0 e^{\alpha x} \tag{1}$$

donde n_0 es el número inicial de electrones y α es el primer coeficiente de Townsed. Al factor $e^{\alpha x}$ se le llama factor de multiplicación y en las cámara proporcionales está limi-

tado a 10^8 o α x = 20 (limite de Raether)[13] ya que, por encima de este valor, se puede entrar en régimen de descarga (streamer), que es el equivalente al modo Geiger-Muller. A medida que los electrones y los iones derivan hacia el ánodo y el cátodo respectivamente, se induce un pulso en los electrodos. Este pulso producido puedes ser leído de dos modos, directamente del electrodo en que se aplica el voltaje usando un condensador para desacoplar el pulso de la alta tensión, o bien a partir de la señal inducida en un electrodo de recogida, separado del de alimentación por un material aislante.

La alta resistencia de los electrodos, que evita la producción de chispas y otros procesos peligrosos para el detector (por ejemplo descargas permanentes), representa por otra parte una de las principales limitaciones de este tipo de detectores. Tras la producción de la señal, la carga producida por la avalancha permanece en la superficie del electrodo, y durante este tiempo, el campo efectivo en la región en que se ha producido la descarga es menor. La consecuencia más importante es una reducción de la eficiencia de detección y de la resolución temporal. Las RPCs pueden trabajar en dos modos de funcionamiento, modo avalancha y modo streamer.

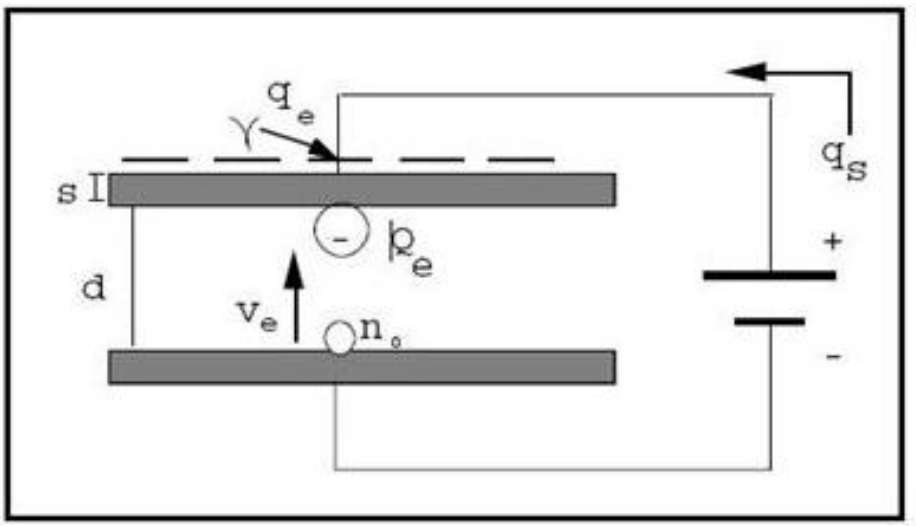
3.3. LAS RPCs DEL CMS

Las cámaras de placas resistivas RPC juegan un papel importante en el muon trigger del LHC.

3.3.1. Principio de Operación: La resistividad de los electrodos determinan principalmente la taza de conteo de una RPC, mientras que la anchura del gap de gas determina el rendimiento en tiempo de la cámara. Otros parámetros, como la densidad del gas y el espesor de los electrodos, también son importantes para alcanzar el mejor rendimiento. Al pasar una partícula cargada por una RPC esta ioniza el gas del gap produciendo un cumulo de electrones n^0 lo que produce una avalancha de electrones debido al campo eléctrico presente en el gas. En la figura 10 se presenta esquemáticamente un modelo simple de la formación de carga en una RPC. Esta avalancha genera una carga eléctrica $Qe(d)$ dentro del gap de gas de espesor d. El viaje de tal carga hacia el ánodo induce rápidamente en uno de los electrodos la carga qe, que representa la señal útil que recibe la RPC de la ionización. La fuente tiene que mover la carga qs en el circuito fuera del gap para compensar la carga colectada en los electrodos. Si α es el número de ionizaciones encontradas por unidad de longitud para un electrón y el coeficiente de acoplamiento β es el número de enlaces encontrados por unidad de longitud, entonces el coeficiente efectivo de ionización puede definirse como $\eta = \alpha - \beta$.

[13] W.R. Leo. Thechniques for Nuclear an Particles Physics Experiments, Springer-Verlag.

Figura 10. Formación de carga en una RPC.



Se dice que una RPC trabaja en modo "*avalancha*" si cumple la condición ηd <20. Se ha mostrado en "Properties of $C_2H_2F_4$ based gas mixture for avalanche mode operation of Resistive Plate Chambers"[14] que, el promedio de carga inducida en uno de los electrodos (qe) se calcula de la siguiente forma:

$$q_e = \frac{k}{\eta d} Qe(d) = q_{el} n_0 \frac{k}{\eta d} \frac{\lambda}{\eta + \lambda} e^{\eta d}. \tag{2}$$

Donde:

- k es una constante que depende de los parámetros del material.

- q_{el} es la carga del electrón.

- n_0 es el tamaño promedio del cluster de carga que origino la avalancha.

- λ es la densidad del cumulo en el gas es decir el número primario de clusters/unidad de longitud producida por una partícula ionízante.

- d es el espesor del gap.

[14]M. Abbrescia."Properties of $C_2H_2F_4$ based gas mixture for avalanche mode operation of Resistive Plate Chambers", CMS NOTE 97/004.

• s es el espesor del electrodo

Para un valor dado de ηd, los factores k y λ deben ser lo mas grande posibles para maximizar la señal que se obtiene en la strip.

3.3.2 Requerimientos y Condiciones Especificas: La RPCs debe cumplir algunos requisitos específicos básicos: buen tiempo de resolución, pequeño tamaño del cluster, buena taza de conteo, es más, deben responder con alta eficiencia y deben resistir el largo tiempo de funcionamiento en condiciones de ruido alto. La identificación de Muones dentro de una ventana de 25 ns requiere no solo resolución de nanosegundos si no también que las colas de la señal de distribución de tiempo estén dentro de la ventana. Esto implica que el paso de tiempo debido a la propagación de la señal a lo largo de las strips debe guardarse dentro de unos nanosegundos.[15]

En CMS las strips largas se usan en la región del barril dónde los efectos de ruido son despreciables, mientras que se usan las strips muy cortas en las endcap dónde el problema de ruido es más severo. El tamaño del cluster (el número de strips contiguas las cuales dan la señal de una partícula ionizada) debe ser pequeño (≤ 2), para alcanzar la resolución del momentum requerido y minimizar el número de posibles procesos asociados. El cuadro 1 lista los requisitos principales.

Cuadro 1. Requisitos principales.

Eficiencia	$> 95\,\%$
Tiempo de resolución	≤ 3 ns 98 % dentro de 20 ns
Tamaño promedio del cluster	$\leq 2 strips$
Taza de conteo	$\geq 1 KHz/cm^2$

Los electrodos usualmente están hechos de placas de Bakelita (resina fenolica) cubiertas con una delgada capa de aislante. La alta resistividad de las placas de la bakelita ρ se elige en conformidad con los requerimientos de conteo necesario. Uno de ellos es la constante de tiempo $\tau = \varepsilon_0(\varepsilon_r + 2)\rho$ de una RPC. De la calidad de la superficie de los electrodos depende que se pueda reducir las descargas espontáneas que puedan afectar el conteo de la cámara.[16]

La densidad del cúmulo de gas λ es crucial para explotar al máximo el detector. En principio λ puede ser tan grande como sea posible para maximizar la señal y alcanzar una eficiencia alta (ver ecuación 2). Las RPCs del CMS utilizan una mescla de $C_2H_2F_4$ ($\lambda \sim 5$ clusters/mm) debido a que últimamente RPCs con gap de 2mm han operado

[15]CMS Collaboration. CMS Thechnical Design Report: The Muon Proyect. CERN/LHC.
[16]CMS Collaboration. CMS Thechnical Design Report: The Muon Proyect. CERN/LHC.

exitosamente utilizando estas mezclas; Una baja densidad de mescla de gas (por ejemplo, la mezcla de argón) tiene $\lambda \sim 2.5$ clusters/mm no permite alcanzar la eficacia requerida en medio de la contaminación alta presente.

Las RPCs de CMS están hechas de dos gaps con strips receptoras comunes en el medio llamado doble-gap RPC. Un esquema simplificado del plan del doble-gap se muestra en figura 8; La señal total inducida en los electrodos es la suma de las señales de cada gap.

Cuadro 2. Parámetros de construcción y operación para una RPC doble-gap del CMS.

Bakelita	2mm
Coeficiente de resistividad de la Bakelita	1-2x10^{10} Ω cm
ancho del gap	2 mm
Voltaje de Operacion	8.5-9.0 kV
Número de gaps	2
Mescla de Gas	95 % $C_2H_2F_4$, 5 % $i - C_4H_{10}$

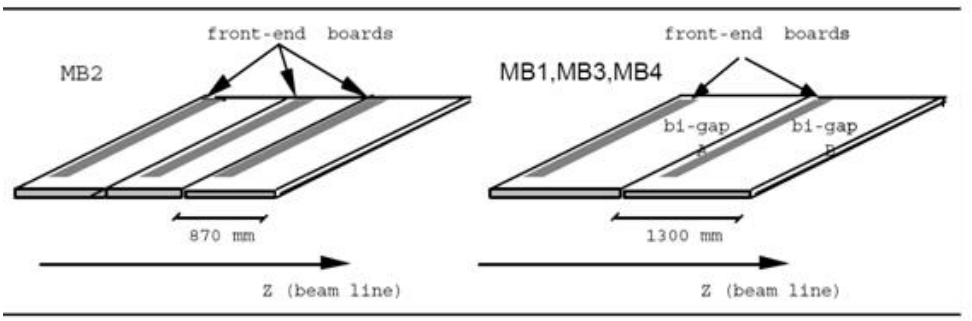
3.3.3 Las RPCs del Barril: En el hierro del barril, las RPCs son arregladas en seis capas (estaciones). Cada capa tiene una cobertura completa de 2π. Dos capas están localizadas en MB1, dos en MB2, una en MB3 y una en MB4. Hay un total de 360 estaciones rectangulares, cada una con una longitud de 2560 mm en la dirección del haz (Z) y un espesor de la rueda de 2000 (MB1) a 4000 (MB4) mm.

Los requerimientos de la física exigen que en cada estación las strips funcionen siempre en la dirección del haz, por tal razón se han dividido en dos partes para las estaciones MB1, MB3 y MB4, la estación MB2 tendrá strips en tres partes ya que esta representa un caso especial para el algoritmo del trigger. En cada estación se a desarrollado dos (o tres) módulos doble-gap montados secuencialmente a lo largo de la dirección del haz para cubrir toda el área. En el caso de dos RPC doble-gap las strips tendrán 1300 mm de longitud, en el caso de tres tendrán una longitud de 850 mm. La figura 11 muestra la estación del barril hecha de dos (o tres) modulos doble-gap.[17]

En cada modulo doble-gap, el circuito electrónico se situo anteriormente en la strip donde se reduce al mínimo el tiempo de llegada de la señal. Cada doble-gap tiene 96 strips.Se desarrollaron un total de 288 canales electrónicos para la estación MB2 y 192 para las otras. La anchura de la strip aumentara por consiguiente desde las estaciones internas hasta las externas (cada strip cubre 5/16 grados en ϕ).

[17]CMS Collaboration. CMS Thechnical Design Report: The Muon Proyect. CERN/LHC.

Figura 11. Modulos.



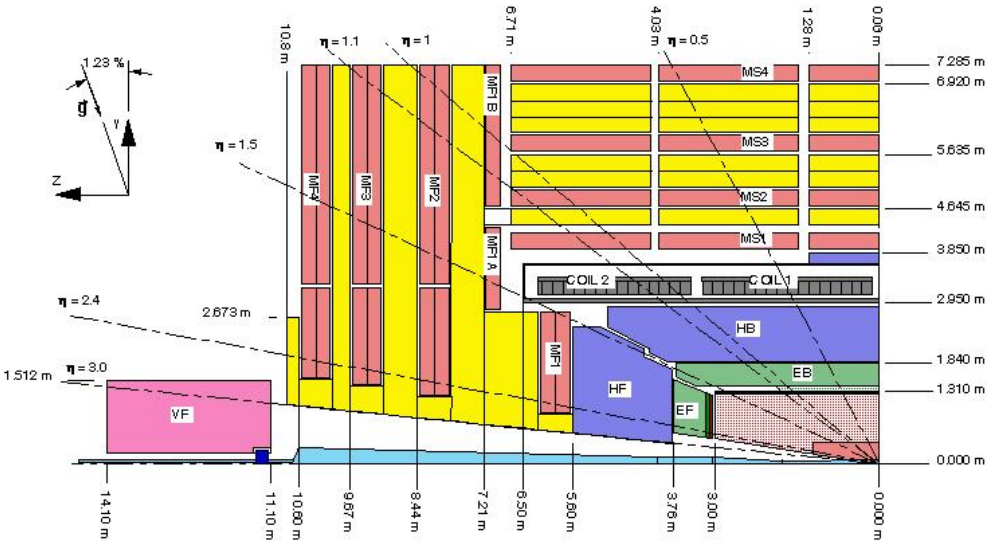
Cuadro 3. Información sobre RPCs en el barril

Número de estaciones	360
Area superficial total	2400 m^2
Número de doble-gaps	840
Número de strips	80640

3.3.4 Las RPCs de las Endcap: Un esquema de la vista R-Z del sistema de RPC Endcap y la localización con respecto al hierro se muestra en la figura 12. Cuatro estaciones de RPCs se planearon en la parte delantera de CMS (ME1, ME2, ME3, ME4) para cubrir regiones hasta η= 2.1. Las estaciones tienen una forma trapezoidal y las strips funcionan a lo largo de la dirección radial.

Para mantener la proyección las strip tienen forma trapezoidal de manera que en cada región η la región cubierta es 5/16 grados en ϕ. También la longitud de las strips varia según la región desde $\sim$ 25 cm a $\sim$ 100 cm. Las estaciones de endcap RPC también se construyen utilizando el gap doble. Sin embargo en el caso de strips muy cortas (en especial para ME1 y en general para alta η), la utilización del gap-doble estándar donde las strips encajan entre los gaps tiene problemas en que la señal no puede ser extraida a no ser de que la segmentación de las cámaras siga la longitud de las strip.

Figura 12. RPCs en las Endcap



4. SIMULACIÓN DEL DESEMPEÑO DE LAS RPCs DEL CMS

Para estudiar el efecto que tiene la presencia de barras de aluminio en frente de las camaras RPC se utilizo el paquete simulador Geant4, herramienta desarrollado en CERN que utiliza algoritmos de Monte Carlo para reproducir los procesos que sufren las partículas al atravesar materiales. En este capitulo primero se dara a conocer de manera general el sofware Geant4 y luego se describirá en detalle la simulación implementada en este para los dos tipos de cámara RPC doble-gap simuladas.

4.1. GEANT

El código de Geant4 es un paquete de software de libre distribución capaz de simular los procesos dominantes que gobiernan las interacciones de las partículas e iones con la materia, en un rango que van desde keV a TeV.[18]

Geant4 un código Monte-Carlo escrito en C++, utiliza avanzadas técnicas de software e ingeniería y tecnología orientada a objetos capaz de simular una configuración experimental al completo con todos los detectores y seguir a las partículas en sus trayectorias dentro de ellos.[19]

Todos los aspectos de los procesos de simulación han sido incluidos en las herramientas:

- Geometría del sistema Detector
- Materiales
- Partículas elementales de interés
- Generación de eventos primarios
- Seguimiento de la trayectoria de partículas a través de los materiales y de los campos electromagnéticos
- Procesos físicos que gobiernan las interacciones entre partículas
- Respuesta de los componentes sensibles del detector
- Generación de datos de los eventos

[18] sitio oficial Disponible en: http://geant4.web.cern.ch/geant4/
[19] Manual de Geant4. Geant4 User's Guide for Aplication Developers.

• Almacenamiento de eventos y trazas

• Visualización del detector y de la trayectoria de las partículas

• Interfaz con el usuario

• Rutinas de construcción

En el interior de Geant4 hay una abundante cantidad de modelos físicos para manejar las interacciones de las partículas con la materia para un amplio rango de energías. Datos y experiencias han sido recogidos desde muchos lugares de todo el mundo y en este sentido, Geant4 actúa como un repositorio que incorpora una gran cantidad de información acerca de todo lo que se conoce en cuanto a interacciones entre partículas. En particular GEANT4 proporciona paquetes especializados para modelar las interacciones físicas electromagnética y hadronica. Esto lo hace por medio del paquete estándar y paquete de bajas energías.

El paquete estándar de GEANT4 proporciona una variedad de modelos basados en análisis que describen de manera aproximada las interacciones de electrones, positrones, fotones y hadrones cargados en el rango de energía de 1.0 KeV a 100TeV. El paquete estándar de GEANT4 esta principalmente destinado para la física de altas energías. Los procesos físicos que este paquete activa son efecto fotoeléctrico, dispersión Compton, producción de pares (e-/e+, mu-/mu+), ionización y producción de rayos delta, bremsstrahlung, aniquilación e+/e- radiación de sincrotrón.

El paquete de bajas energías de GEANT4 contiene las librerías EPDL97, EEDL EADL las cuales proporcionan datos para la calculación de cross-section y los datos del estado final para la interacción del fotón y electrón con la materia.

Este paquete es empleado para activar procesos que se presentan a bajas energías, Rayleigh Scattering, efecto fotoeléctrico, efecto Compton, conversión gama, ionización y Bremstrahlung. Adicionalmente es importante porque la detección de electrones con muy baja energía (menos de 1 eV) esta incluida en este paquete.

Las áreas de aplicación de Geant4 incluyen altas energías, física nuclear, física de aceleradores, estudios de ciencias médicas y espaciales.

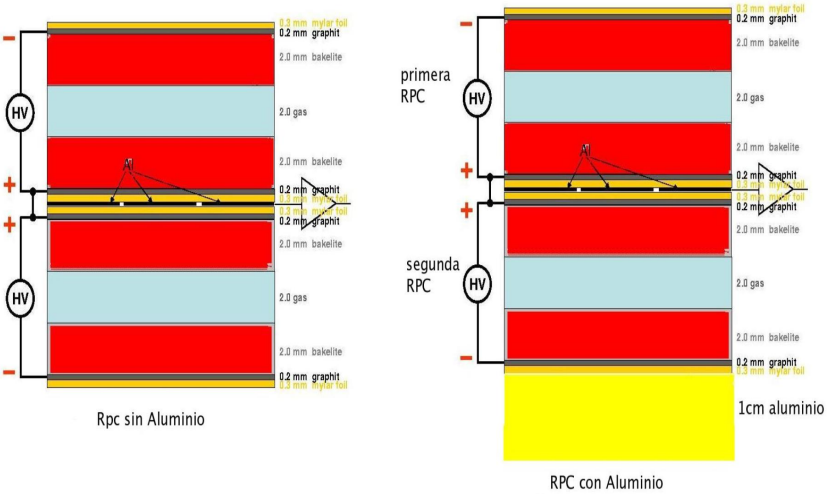
Geant 4 puede ser descargado de la pagina oficial de Geant4[20], una descripción del método Montecarlo utilizado por geant4 se encuentra en los manuales de aplicación[21] y de he-rramientas[22].

[20]Geant4 sitio oficial Disponible en: http://geant4.web.cern.ch/geant4/
[21]Manual de Geant4. Geant4 User's Guide for Aplication Developers.
[22]Manual de Geant4. Geant4 User's Guide for Tooltit Developers.

4.2. IMPLEMENTACION DE UNA RPC EN GEANT

Se realizaran dos simulaciones de RPC doble-gap una con presencia de barras de Aluminio y otra sin presencia de estas. De las simulaciones se obtiene información de posición, energía y procesos físicos de las partículas, esta información se guardó en archivos de datos que luego fueron utilizados para obtener graficas de estas magnitudes en Root[23]. Los tipos de RPCs simulados se muestran en la figura 13.

Figura 13. RPCs Simuladas.



La figura 14 muestra la implementación general que se hizo en Geant4. La geometría del detector se implementa en la clase DetectorConstruction (ver apéndice B.3.1 y B.4.1). En Geant4 la geometría de un detector se implementa por medio de varios volúmenes, primero se define un volumen madre llamado "world volume" (color negro en la figura 15), el cual contiene en su interior los demás volúmenes que conforman el detector, cada volumen es creado definiendo su forma, sus características físicas y el lugar donde se debe ubicar dentro del volumen madre.

Para describir la forma de un volumen, utilizamos el concepto de un sólido. Un sólido es un objeto geométrico que tiene forma y valores específicos para cada una de las dimensiones. Para describir las características completas de un volumen, utilizamos un volumen lógico en el que se incluye las características geométricas del sólido y se agrega características físicas del mismo como es el material; para ubicar el volumen creamos el Volumen Físico.

[23]ROOT Pagina Oficial. Disponible en: http://root.cern.ch/

Figura 14. Esquema de simulación de las RPCs de CMS en Geant4.

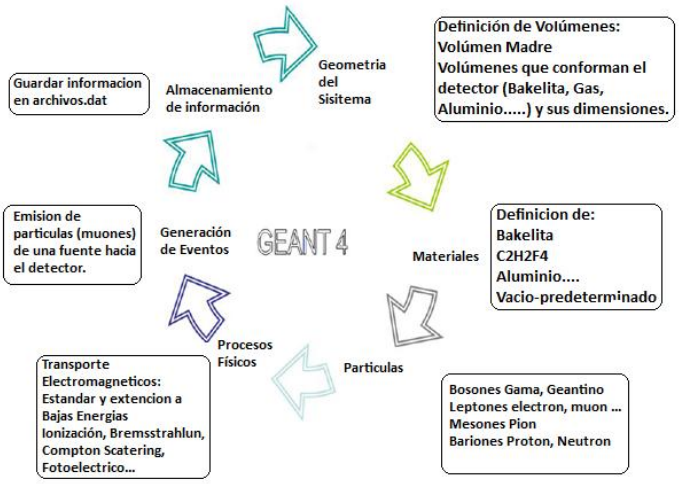
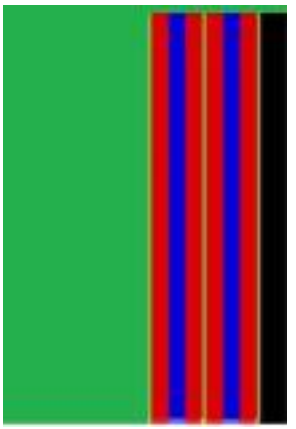


Figura 15. RPC de CMS en geant4. De color verde Barra de Al, rojo Bakelita, Azul C2H2F4, negro volumen madre.



Para nuestra simulación el volumen madre consistió de una caja de 12cm de espesor, 60 cm de ancho y 60 cm de largo lleno de aire.Dentro de este se definió los siguientes volúmenes.

- Mylar de 0.3 mm de espesor, 60 cm de ancho y 60 cm de largo
- Grafito de 0.2 mm de espesor, 60 cm de ancho y 60 cm de largo
- Bakelita de 2 mm de espesor, 60 cm de ancho y 60 cm de largo

• Gas (C2H2F4) de 2mm de espesor 60 cm de ancho y 60 cm de largo

• Aluminio central de 0.2 mm de espesor, 60 cm de ancho y 60 cm de largo

• Barras de aluminio de 1 cm de espesor, 60 cm de ancho y 60 cm de largo

Para especificarlos materiales del detector se utiliza la clase G4Element quien describe las características de los átomos tales como: número atómico, número de nucleones, masa atómica etc. y la clase G4Material quien describe las características macroscópicas de la materia tales como: densidad, estado, etc. Por ejemplo para el gas Tetrafluoretano $C_2H_2F_4$, primero se debe crear cada uno de los elementos que lo constituyen por separado utilizando la clase G4Element; para crear el elemento Carbono se debe especificar su nombre "Carbono", su símbolo "C", su número atómico "z=6" y su masa atómica "a=12.01 g/mol"; luego se crea el gas tetrafluoretano por medio de la clase G4Material especificando su nombre "tetrafluoretano", su densidad "density= 4.25 mg/cm^3" y el número de elementos que lo componen "nel=3" y al adicionar cada elemento se especifica cuantos átomos de este están presentes en el material (ver apéndice B.3.1 y B.4.1). Los demás materiales utilizados como Bakelita, Mylar, Grafito, Aluminio y Aire se toman de la base de materiales que tiene Geant4 por medio de la clase G4NistManager y G4Material.

La física del detector se implementa por medio de la clase PhysicsList (ver apendice B.3.3 y B.4.3) en la que se define los procesos físicos para muones, electrones, iones, gamas, hadrones etc. pero antes se construyen todas las partículas que son posible construir en la versión de Geant4 utilizada.

Las partículas se implementan por medio de la clase G4particleDefinition, esta contiene las características individuales de cada partícula tales como nombre, masa, carga, spin, tiempo de vida, modos de decaimiento, paridad, conjugación de carga, iso-spin, 3ra componente de iso-spin, tipo de partícula, sub-tipo de partícula, número leptonico, número barionico, por ejemplo para crear el electrón se llama la clase G4Electron la cual nos crea la partícula con las características que se listan en el cuadro 4.

Las particulas que se implementaron son:

• Bosones: gama

• Leptones: electron, neutrino electronico, positron, antineutrino electronico, muon, neutrino muonico, antimuon, antineutrino muonico

• Mesones: π^0, π^+, π^-, eta, $kaon^+$, $kaon^-$, $kaon^0$, antikaon

• Baryones: proton, antiproton, netron, antineutron

Cuadro 4. Argumentos para la construcción de un electrón

mass	0.51099906*MeV	width	0.0*MeV
charge	-1	2*spin	1
parity	0	C-conjugation	0
2*Isospin	0	2*Isospin3	0
G-parity	0	type	lepton
número leptonico	1	número barionico	0
lifetime	-1	decay table	NULL
shortlived	false	subType	e

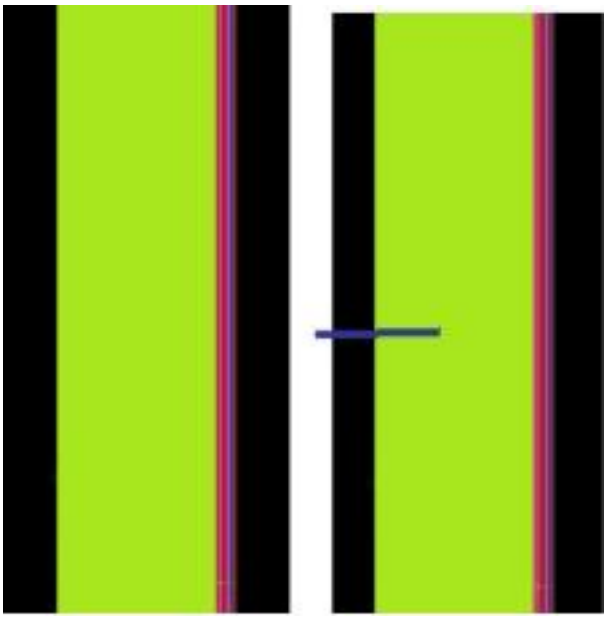
Los procesos físicos que describen cómo las partículas interactán con los Materiales en GEANT se derivan de la clase de G4VProcess. En el caso para nuestra simulación se tomaron en cuenta: proceso de transporte, procesos electromagnéticos típicos, decaimiento y también procesos a bajas energías para iones y hadrones, todos proporcionados en el paquete de librerías de Geant4 (ver apendice B.3.3 y B.4.3). En concreto, los procesos electromagnéticos que se tomaron son los proporcionados por el paquete de Geant4, el cual incluye:

- Dispersión Compton, Efecto fotoeléctrico y conversión gama para fotones
- Producción de pares para electrones y muones
- Dispersión múltiple e ionización para electrones y muones
- Bremsstrahlung para electrones y muones
- Aniquilación para positrones

Los procesos nucleares que se tubieron encuenta fueron colisiones elásticas e inelásticas para los neutrones, fisión, captura y decaimiento.

Para crear las partículas incidentes (en este caso muones) se implemento la clase PrimaryGeneratorAction (ver apendice B.3.6 y B.4.7). La fuente de las partículas esta ubicada en las coordenadas (x=0,y=0,z= -40mm) a una distancia de 28 mm de la RPC doble-gap en direccion z, las partículas siempre inciden perpendicularmente sobre el plano (x-y) ver figura 16, con una energía inicial de 500 GeV, en total se se hacen incidir un millón de muones; Geant4 presenta en su entorno grafico de diferentes colores la traza que dejan de las partículas al atravesar materiales en la figura 15 la traza que deja el muon se identifica de color azul.

Figura 16. Vista de la incidencia de 1 muon de 500 GeV sobre una RPC.



Los parámetros como que partícula incide, la energía inicial con la que son disparadas pueden modificarse externamente una vez compilado el programa la manera más comoda de ejecutarlo es a través de un macro (ver apendice B.5) en donde se le pasan todos estos parámetros con los valores deseados.

Las siguientes imágenes figuras 17,18,19,20 y 21 nos muestran como geant4 presenta en su forma grafica las trazas de diferentes partículas atravesando materiales, las trazas de color rojo corresponde a electrones y las de color verde fotones.

Figura 17. Vista de la incidencia de 100 muones de 500 GeV sobre una RPC.

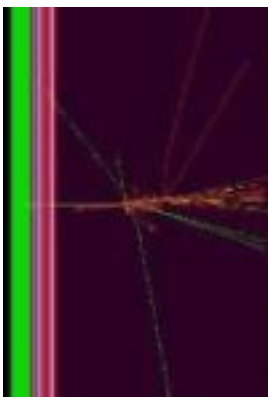


Figura 18. Vista de la incidencia de 1000 muones de 500 GeV sobre una RPC.

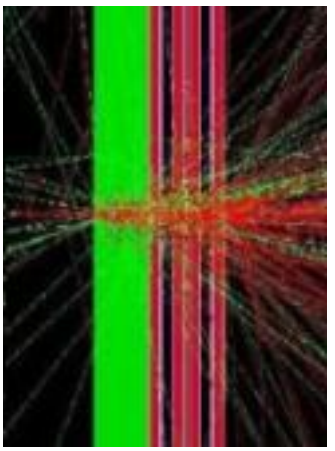


Figura 19. Vista de la incidencia de 10000 muones de 500 GeV sobre una RPC.

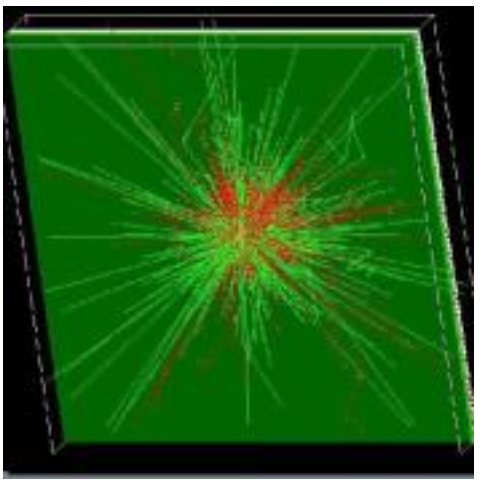


Figura 20. Vista de la incidencia de 10000 muones de 500 GeV sobre una RPC

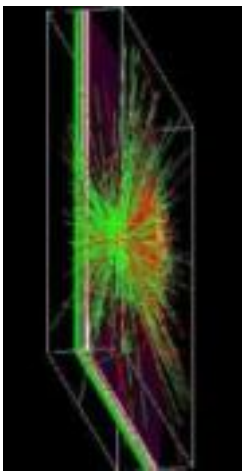
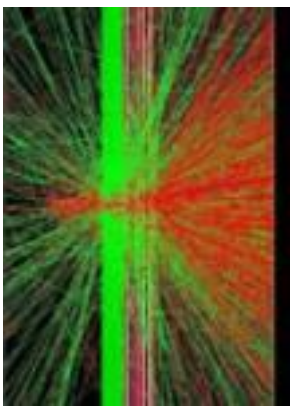


Figura 21. Vista de la incidencia de 20000 muones de 500 GeV sobre una RPC



También se creo la clase SteppingAction (ver apédice B.3.12 y B.4.13) encargada de obtener información de energía y coordenada (x) para los electrones que llegan a la zona del gas, así como también nos da información de la creación de rayos delta y los procesos físicos producidos al pasar las partículas a traves de los diferentes materiales. Toda esta información se almacenó en un archivo de datos, para luego obtener histogramas de las variables en Root.

5. RESULTADOS

En este capítulo se comentan los resultados obtenidos para cada uno de los casos de RPCs simulados en Geant4.

5.1. SIMULACIÓN DE UNA RPC SIN PRESENCIA DE BARRAS DE ALUMINIO

Con las energías de los electrones producidos como consecuencia del paso de muones por la RPC que alcanzan a llegar al gas se construyo un histograma de la distribucion de Energía ver Figura 22. También se construyo una gráfica con la posición X de estos electrones, ver Figura 23

Figura 22. Histograma de Energía de los electrones que se obtienen en el gas de la RPC debido a la incidencia de un millón de muones de 500 GeV

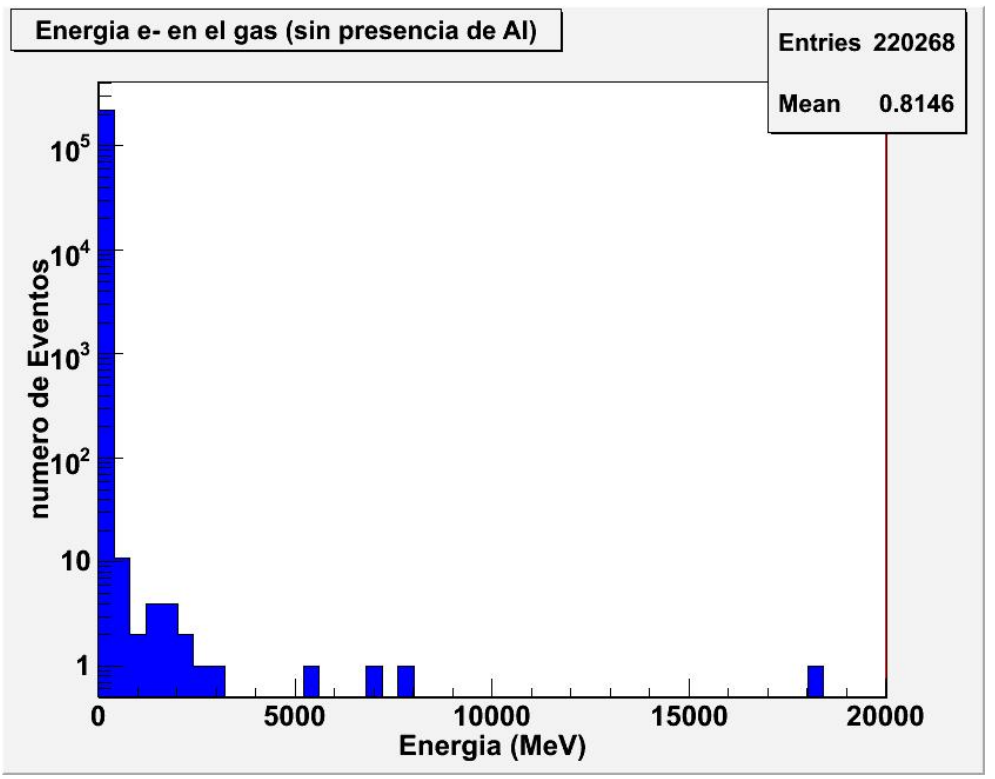
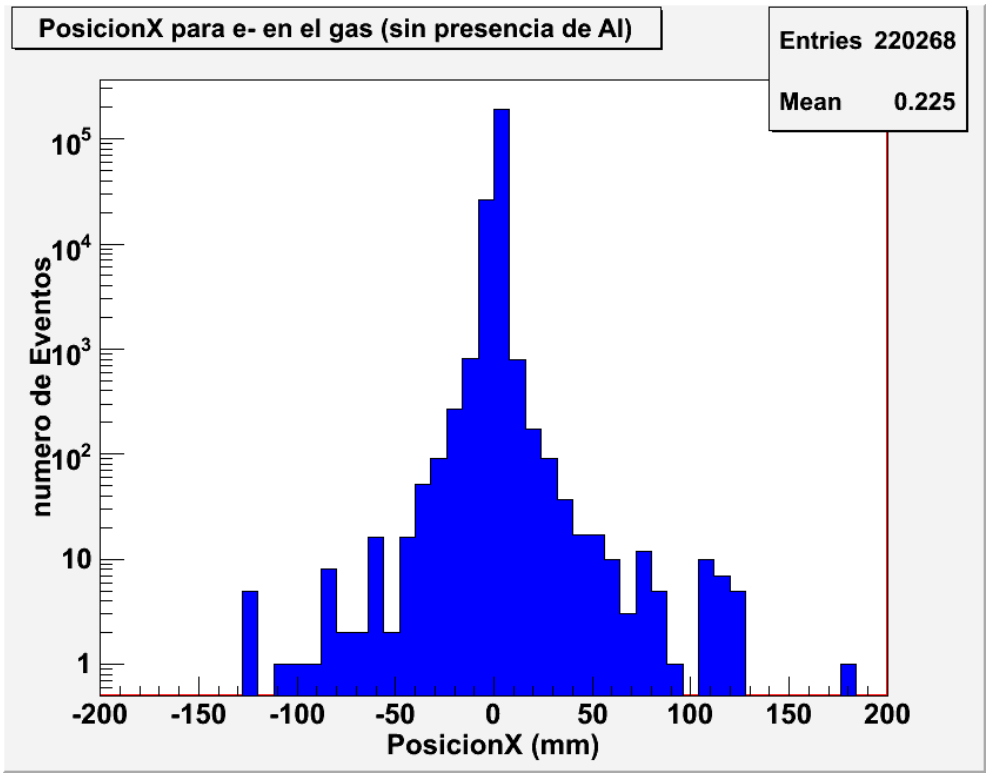


Figura 23. Posicion X de los electrones que alcanzan a llegar al gas de la RPC debido a la incidencia de un millón de muones de 500 GeV



Debido a que la señal de las RPCs se obtiene por medio de las strip de Aluminio en el cuadro 5 se da conocer como se ha tomado la distribución de strips alrededor de la strip por donde el muon atravesó originalmente.

Cuadro 5. Distribución de strips alrededor de la strip por donde el muon atravesó originalmente.

Strip #	Rango (mm)
4*	< -120.5
3*	-120.5 a -85.5
2*	- 85.5 a 50.5
1*	- 50.5 a -17.5
0	- 17.5 a 17.5
1	17.5 a 50.5
2	50.5 a 85.5
3	85.5 a 120.5
4	> 120.5

Cuadro 6. Distribución del número de electrones para cada una de las strip contiguas a la strip por donde el muon atravezo originalmetnte strip número 0.

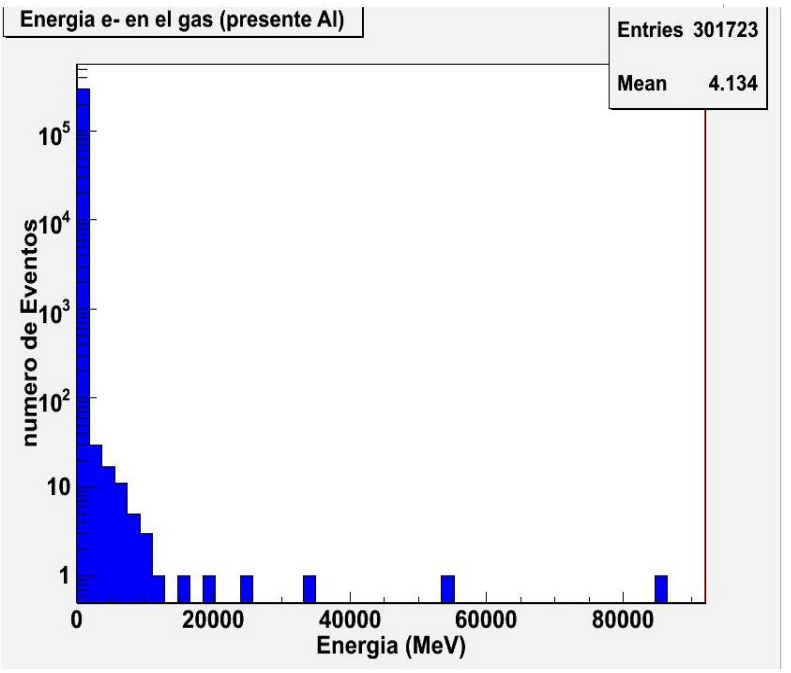
Strip #	Strip 3*	Strip 2*	Strip 1*	Strip 0	Strip 1	Strip 2	Strip3
# electrones	1	1	100	220102	60	2	2
%	0.0005	0.0005	0.0454	99.9246	0.0272	0.0009	0.0009

Para la incidencia de un millón de muones de 1GeV sobre la RPC sin presencia de barras de aluminio se encontró que el número de electrones producidos que alcanzan a llegar a la zona del gas es de 220268 con una energía media de 0,225 MeV, es decir se producen alrededor de 0,22 electrones por incidencia de un muon sobre la RPC. Cerca del 99.9 % de electrones que alcanzan a llegar al gas dejarían su señal en la strip número 0, el 0.1 % restante dejaría su señal en las strips vecinas.

5.2 SIMULACION DE UNA RPC CON PRESENCIA DE BARRAS DE ALUMINIO

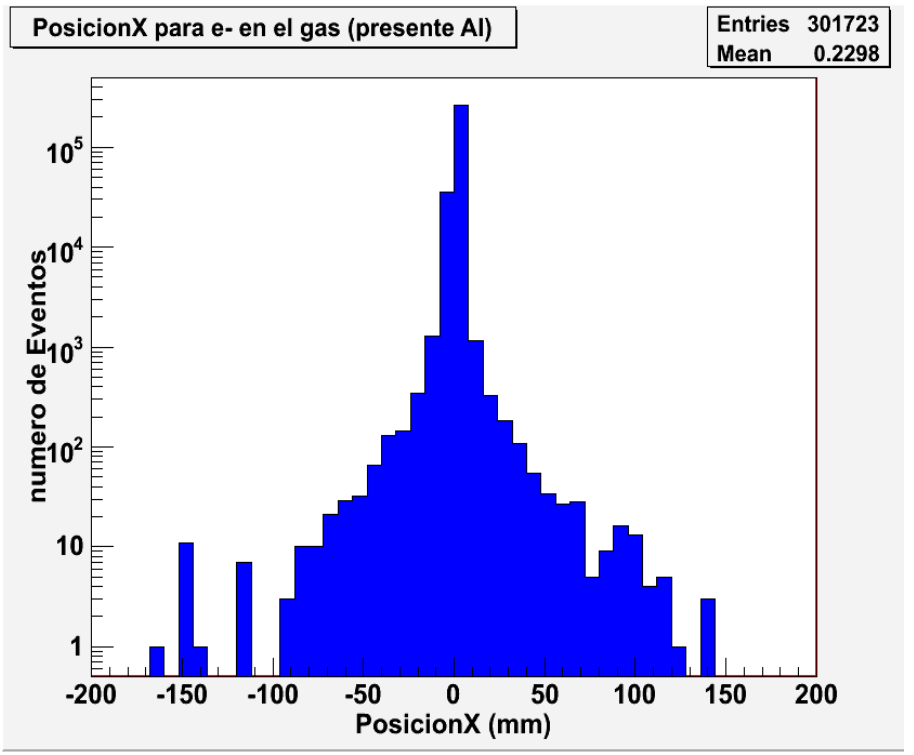
Con las energías de los electrones producidos como consecuencia del paso de muones por la RPC que alcanzan a llegar al gas se construyo un histograma de la distribucion de Energía ver Figura 24.

Figura 24. Energía de los electrones que alcanzan a llegar al gas de la RPC debido a la incidencia de un millón de muones de 500 GeV



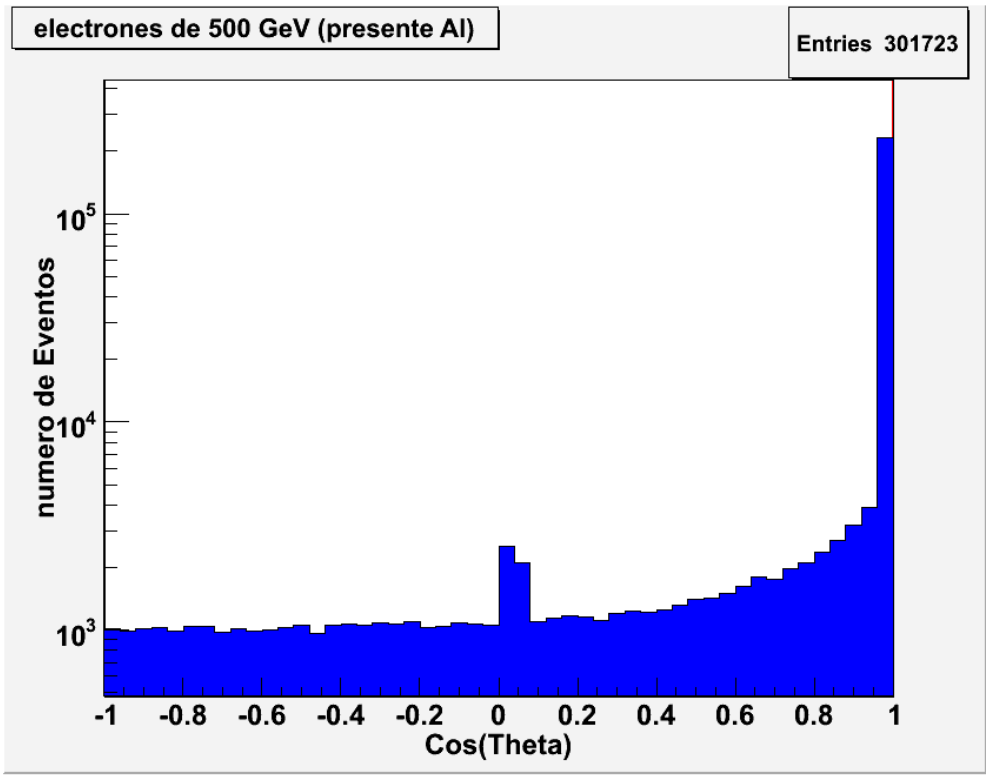
También se construyo una gráfica con la posición X de estos electrones, ver Figura 25.

Figura 25. Posición X de los electrones que alcanzan a llegar al gas de la RPC debido a la incidencia de un millón de muones de 500 GeV



Adicionalmente se construye una gráfica teniendo en cuenta la distribución del ángulo θ de los electrones obtenidos Figura 26, debe tenerse en cuenta que θ es el ángulo polar, lo que significa que θ=0 grados (cosθ=1) significa seguir en línea recta hacia adelante, y θ=180 grados (cosθ=-1) significa rebotar que la partícula se esta devolviendo.

Figura 26. Distribución angular ($\cos\theta$) de los electrones que alcanzan a llegar al gas de la RPC debido a la incidencia de un millón de muones de 500 GeV



Debido a que la señal de las RPCs se obtiene por medio de las strip de Aluminio, tambien se realizo el cuadro 7 en el cual se mira en que strip dejaria un muon la señal.

Cuadro 7. Distribución del número de electrones para cada una de las strip contiguas a la strip por donde el muon atravezo originalmetnte strip # 0.

Strip #	Strip4*	Strip3*	Strip2*	Strip1*	Strip0	Strip1	Strip2	Strip3	Strip4
# e-	67	2	173	265	300615	426	148	2	25
%	0.0222	0.0007	0.0573	0.0878	99.6328	0.1412	0.0491	0.0007	0.0083

Para incidencia de un millón de muones de 1GeV sobre la RPC con presencia de barras de aluminio se encontró que el número de electrones producidos que alcanzan a llegar a la zona del gas es de 301723 con una energía media de 4.134 MeV, es decir se producen alrededor de 0,301 electrones por incidencia de un muon sobre la RPC.

Cerca del 99.6 % de electrones que alcanzan a llegar al gas dejarían su señal en la strip #0, el 0.4 % restante dejaría su señal en las strips vecinas.

Para el estudio de los rayos delta electrones altamente energéticos se obtuvo archivos de datos en los cuales se presenta información de cuantos deltas producidos en las barras de aluminio y los electrones producidos por estos alcanzan a llegar a la zona de gas, encontrando aproximadamente 3020 eventos, correspondiendo esto al 1 % del número total de electrones que llegan a la zona del gas; La gran mayoría de estos eventos se encuentran concentrados en la zona comprendida entre -15 mm y 15 mm es decir dejarían señal en la strip número 0 y número 1.

5.3 ANALISIS DE RESULTADOS

• Debido a la presencia de Barras de Aluminio enfrente de las RPCs el número de electrones producidos que alcanzan a llegar a las franjas de gas, se incrementan cerca del 37 % pasa de 220268 (sin presencia de barras de Aluminio) a 301723 (con presencia de barras de aluminio) y la energía media con la que son producidos es mucho mas grande 4.134 MeV cuando hay presencia de barras de aluminio y 0.225 MeV sin presencia de barras de Aluminio.

• El número de rayos delta electrones altamente energéticos producidos en las barras de Aluminio y los electrones originados por estos que alcanzan a llegar a las franjas de gas es aproximadamente de 3020 correspondiendo este valor al 1 % del número total de electrones que llegan a la zona del gas. Esto quiere decir que el efecto de los rayos delta producidos por la presencia de barras de aluminio incrementa en muy poca medida la producción del número de electrones en las franjas de gas.

• Las graficas de distribución de posición x en la zona del gas para los electrones producidos por el paso de mu- muestran que la mayor parte de los electrones que alcanzan a llegar al gas cerca del 99.9 % para el caso de la RPC sin presencia de barras de aluminio y del 99.6 % para el caso de la RPC con presencia de barras de Aluminio se encuentran a una distancia comprendida entre -25 mm y 25 mm al rededor del punto por donde el muon atraviesa para los dos casos de RPCs simulados es decir atraviesan la strip número 0 y parte de la strip número.

• Teniendo en cuenta todo lo anterior la presencia de Barras de Aluminio en frente de las RPC aunque incrementan el número de electrones producidos que alcanzan a llegar a la franja de gas no son las causantes de producir señales alejadas de la trayectoria original por donde el muon atravesó.

CONCLUSIONES

Geant4 es una herramienta muy poderosa en la investigación de altas energías, aporta valiosa información sobre el comportamiento que las partículas tienen al interaccionar con la materia. Con los resultados obtenidos de la simulación RPC doble-gap de CMS se puede concluir que la presencia de barras de aluminio en frente de las cámaras RPC aunque incrementan el número de electrones producidos que alcanzan a llegar a la franja de gas no son las causantes de producir señales alejadas de la trayectoria original por donde el muon atravesó.

El número de strips contiguas que darían la señal de activaciíon de que por ese lugar paso una partícula, seria de 2 strips para los dos casos de RPCs simuladas, cumpliendo así con el requisito para las RPCs de CMS.

El número de rayos delta electrones altamente energéticos producidos en las barras de Aluminio y los electrones originados por estos que alcanzan a llegar a las franjas de gas corresponde al 1 % del número total de electrones que llegan a la zona del gas. Esto quiere decir que el efecto de los rayos delta producidos por la presencia de barras de aluminio incrementa en muy poca medida la producción del número de electrones en las franjas de gas.

BIBLIOGRAFÍA

CMS Collaboration.CMS Physics Thechnical Design Report. Vol I: Detector Performance and Software. CERN/LHC,CMS TDR 8.1, 2 February 2006.

CMS Collaboration. CMS Thechnical Design Report: The Muon Proyect. CERN/LHC, 97-32,CMS TDR 3, 15 December 1997.

Compact Muon Solenoide. Disponible en: http://lhc.web.cern.ch/cms/.

FERRER SORIA Antonio. Física Nuclear y de Particulas,Segunda Edición, Universidad de Valencia, 2006.

Geant4. Sitio oficial Disponible en: http://geant4.web.cern.ch/geant4/

M. ABBRESCIA. "Properties of C2H2F4 based gas mixture for avalanche mode operation of Resistive Plate Chambers", CMS NOTE 97/004.

Manual de Geant4. Geant4 User's Guide for Aplication Developers Version , geant4 9.0, 2007. Disponible en: http://geant4.web.ch/geant4/UserDocumentation/UserGuides/forAplicationDeveloper/BackupVersions/V9.0/

Manual de Geant4. Geant4 User's Guide for Tooltit Developers Version , geant4 9.0, 2007. Disponible en: http://geant4.web.ch/geant4/UserDocumentation/UserGuides/forToolkitDeveloper/BackupVersions/V9.0/

MATTHEW WINGHAM. CMS tracker and preparing for Early physics at the LHC,CERN-THESIS-2008-081, 09/09/2008.

ROOT. Pagina Oficial. Disponible en: http://root.cern.ch/

W.E. Burchan. Física Nuclear (Parte B), Editorial Reverte, Buenos Aires, 1979.

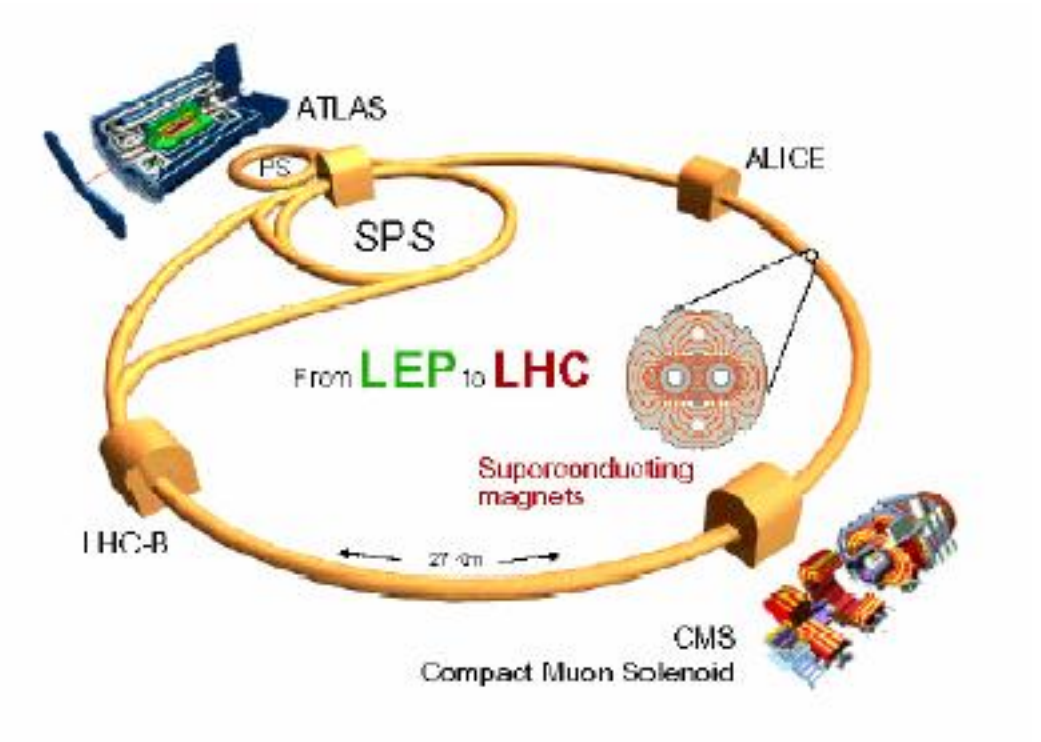
W.R. LEO. Thechniques for Nuclear an Particles Physics Experiments, Springer-Verlag, 1987.

ANEXOS

ANEXO A. El Gran Colisionador de Hadrones LHC.

El LHC está formado por un conjunto de dos anillos sincrotrón instalados en los 26.6 km de circunferencia del túnel que anteriormente fue utilizado por el colisionador electrón-positrón LEP. Además, aprovecha el complejo de aceleradores ya existente para conseguir llegar, de forma progresiva, a las altas energías requeridas. Primero las partículas son aceleradas en el acelerador lineal (Linac 2) que genera protones de 50 MeV que alimenta el PSB (proton Synchrotron Booster). Después los protones son inyectados a 1.4 GeV en el Proton Synchrotron (PS) que los acelera a 26 GeV (ver figura 27). Luego son inyectados al SPS (Super Proton Synchrotron) que es empleado para incrementar la energía de los protones por encima de los 450 GeV energía requerida para que sean inyectados en los dos anillos de LHC con sentido de giro inverso.

Figura 27. LHC



Los haces, una vez que se han inyectado en los dos anillos de LHC, se hacen girar en sentido inverso, uno respecto de otro, efectuando las revoluciones necesarias hasta alcanzar los 7 TeV y con cuya energía ya son dirigidos a los distintos puntos de colisión donde se localizan los experimentos. Los parámetros que caracterizan al colisionador LHC se

muestran en el cuadro 8.

Cuadro 8. Resumen de los parámetros característicos de colisionador LHC.

Parámetro	Valor nomonal
Energía centro de masa por haz	7TeV
Luminosidad máxima p-p	$10^{34} cm^{-2} s^{-1}$
Energía de inyección el LHC	450 GeV
Corriente DC	0.56 A
Número de paquetes por anillo	2835
Número de protones por paquete	10^{11}
Separación entre paquetes	25ns (7.48m)
Frecuencia de interacción	10^{9}Hz
Temperatura de imán	$\leq$ 2K

El LHC esta provisto de cuatro puntos de colisión, como se aprecia en la Figura 27. En cada uno de ellos se localiza un detector con propósitos de investigación concretos, en función de los cuales se diseña. Los experimentos localizados en dichos puntos de colisión son:

• LHCb (Large Hadron Collider beauty experiment) que se dedicará a estudiar las propiedades de los mesones y bariones b, analizando la violación de CP para esas partículas, un esquema del detector LHCb se presentan en la figura 28.

• ALICE (A Large Ion Collider Experiment) que estudiará colisiones de iones pesados de plomo a energías de s =1150 TeV en el centro de masas y con una luminosidades del orden de $10^{27} cm^{-2} s^{-1}$. Como resultado de esas colisiones altamente energéticas, la cromodinámica cuántica postula que se forma un plasma de quarks y gluones, un esquema del detector ALICE se presentan en la figura 29.

• ATLAS (A Toroidal LHC ApparatuS) y CMS (Compact Muon Solenoid) son detectores de propósito más general que los anteriores que detectarán la física (tanto las nuevas partículas como las interacciones implicadas) que se genera por colisiones de haces protón-protón con energías en sistema centro de masas de s =14 TeV (de 7 TeV por haz) y a luminosidades del orden de $10^{34} cm^{-2} s^{-1}$ con un tiempo de vida de esta luminosidad de 10 horas. El objetivo central es comprobar si existe el bosón de Higgs según lo predice el modelo estándar, y/o la variedad de bosones de Higgs que predice el Modelo Super Simétrico Mínimo (MSSM).

Figura 28. Esquema del detector LHCb.

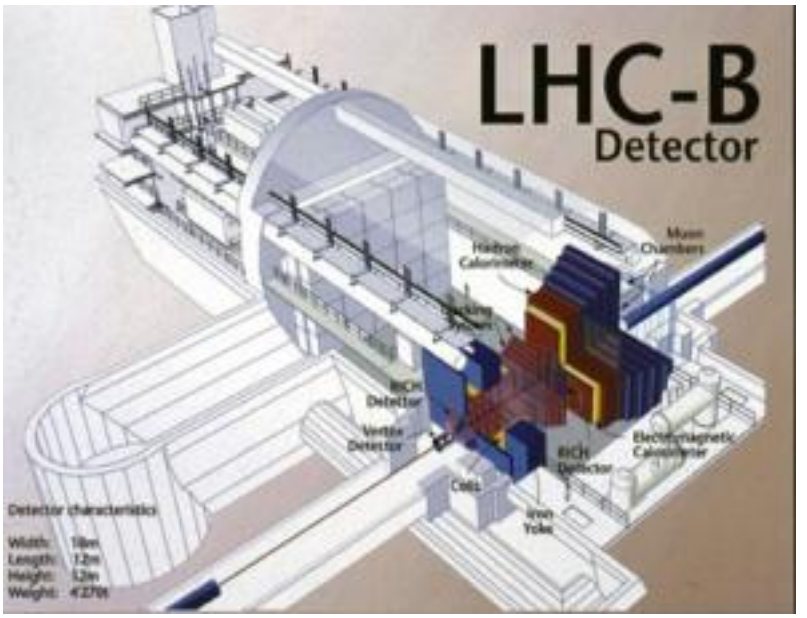


Figura 29. Esquema del detector ALICE

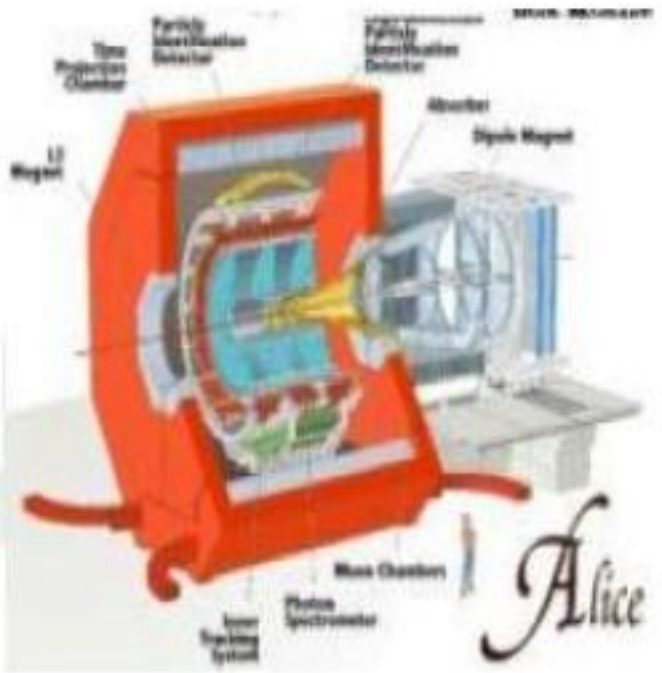


Figura 30. Esquema del detector ATLAS

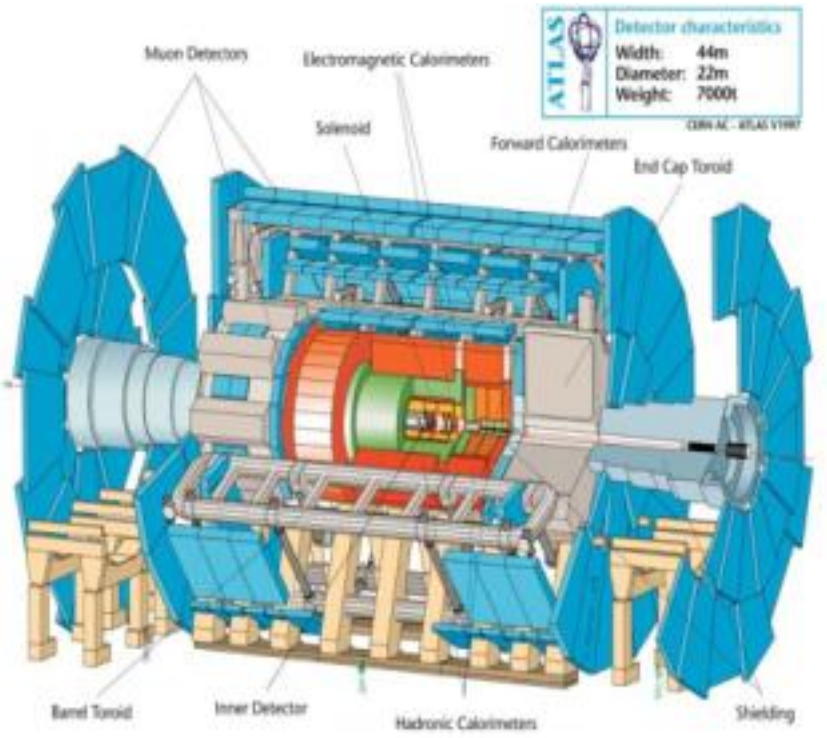
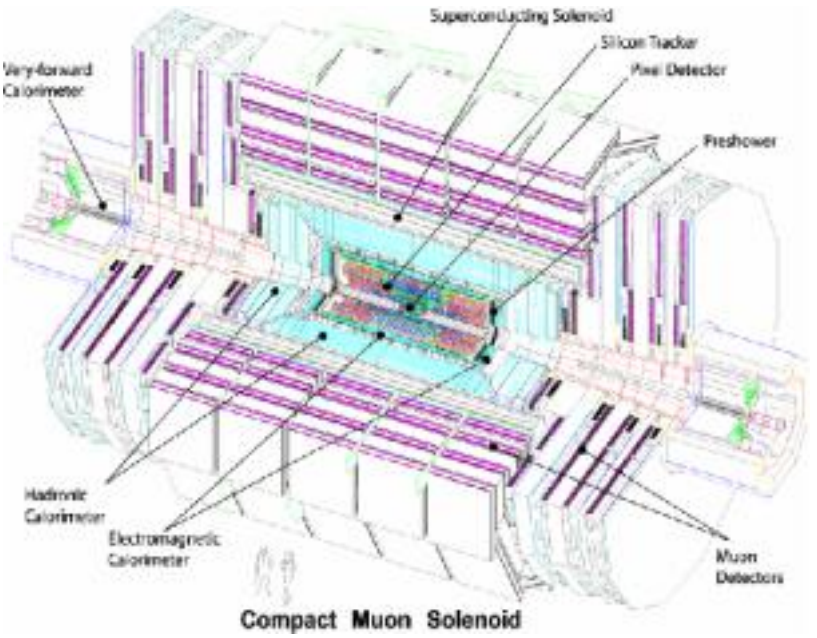


Figura 31. Esquema del detector CMS



Además, se pretende determinar si se cumplen algunas de las predicciones que proponen los modelos supersimétricos (SUSY), así como la posible existencia de dimensiones extras, todo ello dentro del intervalo de energías que cubre el experimento, de hasta unos pocos TeV por quark, un esquema del detector ATLAS y CMS se presentan en las figuras 30 y 31.

ANEXO B. Codigo en Geant4 para la implementacion del modelo realista de una RPC del experimento CMS.

En Geant4 una aplicacion contiene dos directorios principales (include y src) y dos archivos, el ejecutable (GNUmakefile) y el archivo principal.

B.1. ARCHIVO PRINCIPAL: rpc.cc

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "G4UItcsh.hh"
#include "Randomize.hh"
#include "DetectorConstruction.hh"
#include "PhysicsList.hh"
#include "PrimaryGeneratorAction.hh"
#include "RunAction.hh"
#include "EventAction.hh"
#include "SteppingAction.hh"
#ifdef G4VIS_USE
#include "G4VisExecutive.hh"
#endif
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
int main(int argc,char** argv) {
 // Construct the default run manager
 G4RunManager * runManager = new G4RunManager;
  // set mandatory initialization classes
  DetectorConstruction* det;
  PrimaryGeneratorAction* prim;
  runManager->SetUserInitialization(det = new DetectorConstruction);
  runManager->SetUserInitialization(new PhysicsList);
  runManager->SetUserAction(prim = new PrimaryGeneratorAction);

  G4VisManager* visManager = 0;
  // set user action classes
  RunAction* run;
  runManager->SetUserAction(run = new RunAction(det,prim));
  runManager->SetUserAction(new EventAction(run));
  runManager->SetUserAction(new SteppingAction(det,prim,run));
/*//////
  G4UImanager* UII = G4UImanager::GetUIpointer();
  UII->ApplyCommand("/run/verbose 1");
  UII->ApplyCommand("/event/verbose 1");
  UII->ApplyCommand("/tracking/verbose 1");
///////*/
    G4UImanager* UI = G4UImanager::GetUIpointer();

  if (argc==1)   // Define UI terminal for interactive mode
    {
#ifdef G4VIS_USE
      // visualization manager
      visManager = new G4VisExecutive;
      visManager->Initialize();
#endif

      G4UIsession * session = 0;
#ifdef G4UI_USE_TCSH
      session = new G4UIterminal(new G4UItcsh);
```

```
#else
      session = new G4UIterminal();
#endif
      session->SessionStart();
      delete session;
    }
  else           // Batch mode
    {
      G4String command = "/control/execute ";
      G4String fileName = argv[1];
      UI->ApplyCommand(command+fileName);
    }

  // job termination
  if(visManager) delete visManager;

  delete runManager;

  return 0;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.2. EJECUTABLE: GNUmakefile

```
----------------------------------------------------------------

name := rpc
G4TARGET := $(name)
G4EXLIB := true

ifndef G4INSTALL
  G4INSTALL = ../../../..
endif

.PHONY: all
all: lib bin

include $(G4INSTALL)/config/binmake.gmk
```

B.3. ARCHIVOS DE LA CARPETA SRC

B.3.1. DetectorConstruction.cc

```
#include "DetectorConstruction.hh"
#include "G4Material.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "G4PVPlacement.hh"
#include "globals.hh"
#include "G4VisAttributes.hh"
#include "G4Colour.hh"
#include "G4NistManager.hh"
#include "G4ios.hh"
```

```
DetectorConstruction::DetectorConstruction()
  :  VolumenGeneral_log(0), VolumenGeneral_phys(0), CuboGas_log(0), CuboGas_phys(0),
 CuboGas1_log(0), CuboGas1_phys(0), CuboBakelita_log(0), CuboBakelita_phys(0),
CuboBakelita1_log(0), CuboBakelita1_phys(0), CuboGrafito_log(0), CuboGrafito_phys(0),
CuboMylar_log(0), CuboMylar_phys(0), CuboAluminio_log(0), CuboAluminio_phys(0)
{;}

DetectorConstruction::~DetectorConstruction()
{
}

G4VPhysicalVolume* DetectorConstruction::Construct()
{

  //------------------------------------------ DEFINICION DE MATERIALES

  G4double a;  // atomic mass
  G4double z;  // atomic number
  G4double density;
  G4int nel;
  // C2H2F4
  G4int natoms;
  G4Element* C = new G4Element("Carbono", "C", z=6., a= 12.01*g/mole);
  G4Element* H = new G4Element("Hidrogeno", "H", z=1., a= 1.01*g/mole);
  G4Element* F= new G4Element("Fluor", "F", z=9., a= 18.9*g/mole);
  G4Material* Tetrafluoretano = new G4Material("Tetrafluoretano", density= 4.25*mg/cm3, nel=3);
  Tetrafluoretano->AddElement(C, natoms=2 );
  Tetrafluoretano->AddElement(H, natoms=2);
  Tetrafluoretano->AddElement(F, natoms=4);
//
G4NistManager* man = G4NistManager::Instance();
man->SetVerbose(1);
G4Material* BAKELITE = man->FindOrBuildMaterial("G4_BAKELITE");
G4Material* AIR = man->FindOrBuildMaterial("G4_AIR");
G4Material* MYLAR = man->FindOrBuildMaterial("G4_MYLAR");
G4Material* GRAPHITE = man->FindOrBuildMaterial("G4_GRAPHITE");
G4Material* ALUMINIO = man->FindOrBuildMaterial("G4_Al");

//-------------------------------------------------VOLUMENES primera rpc sencilla

//------------------------------ volumen general

  G4double VolGen_x = 30*cm;
  G4double VolGen_y = 30*cm;
  G4double VolGen_z = 6.0*cm;
  G4Box* VolumenGeneral_box
    = new G4Box("VolGen_box",VolGen_x,VolGen_y,VolGen_z);
  VolumenGeneral_log = new G4LogicalVolume(VolumenGeneral_box,
                                           AIR,"VolGen_log",0,0,0);
   VolumenGeneral_phys = new G4PVPlacement(0,G4ThreeVector(),
                                    VolumenGeneral_log,"VolGen",0,false,0);

//---------------------------------------VOLUMEN DE GAS
  ///capas delgadas de gas
  G4double CuboGas_x = 30.0*cm;
  G4double CuboGas_y = 30.0*cm;
  G4double CuboGas_z = 0.25*mm;
  G4Box* CuboGas_box
    = new G4Box("CuboGas_box",CuboGas_x,CuboGas_y,CuboGas_z);
  CuboGas_log = new G4LogicalVolume(CuboGas_box,
                                           Tetrafluoretano,"CuboGas_log",0,0,0);
  G4double PosGas_x = 0*cm;
  G4double PosGas_y =0*cm;
  G4double PosGas_z = 4.35*mm;
  CuboGas_phys = new G4PVPlacement(0,G4ThreeVector(PosGas_x,PosGas_y,PosGas_z),
```

```
                                    CuboGas_log,"CuboGas",VolumenGeneral_log,false,0);
 ///////////volumen de gas
  G4double CuboGas1_x = 30.0*cm;
  G4double CuboGas1_y = 30.0*cm;
  G4double CuboGas1_z = 0.75*mm;
  G4Box* CuboGas1_box
    = new G4Box("CuboGas1_box",CuboGas1_x,CuboGas1_y,CuboGas1_z);
  CuboGas1_log = new G4LogicalVolume(CuboGas1_box,
                                          Tetrafluoretano,"CuboGas1_log",0,0,0);

  G4double PosGas1_x = 0*cm;
  G4double PosGas1_y =0*cm;
  G4double PosGas1_z = 3.35*mm;
  CuboGas1_phys = new G4PVPlacement(0,G4ThreeVector(PosGas1_x,PosGas1_y,PosGas1_z),
                                    CuboGas1_log,"CuboGas1",VolumenGeneral_log,false,0);

//-------------volumen de BAKELITA
  G4double CuboBakelita_x = 30.0*cm;
  G4double CuboBakelita_y = 30.0*cm;
  G4double CuboBakelita_z = 0.9*mm;
  G4Box* CuboBakelita_box
    = new G4Box("CuboBakelita_box",CuboBakelita_x,CuboBakelita_y,CuboBakelita_z);
  CuboBakelita_log = new G4LogicalVolume(CuboBakelita_box,
                                     BAKELITE,"CuboBakelita_log",0,0,0);
//volumenes
for(G4int i=0;i<2;i++)
  {
  G4double PosBak_x = 0*cm;
  G4double PosBak_y = 0*cm;
  G4double PosBak_z = (1.5+4.2*i)*mm;

  CuboBakelita_phys = new G4PVPlacement(0,G4ThreeVector(PosBak_x,PosBak_y,PosBak_z),
                                      CuboBakelita_log,"CuboBakelita",VolumenGeneral_log,false,i);
  }
///////capas delgadas de bakelita
  G4double CuboBakelita1_x = 30.0*cm;
  G4double CuboBakelita1_y = 30.0*cm;
  G4double CuboBakelita1_z = 0.1*mm;
  G4Box* CuboBakelita1_box
    = new G4Box("CuboBakelita1_box",CuboBakelita1_x,CuboBakelita1_y,CuboBakelita1_z);
  CuboBakelita1_log = new G4LogicalVolume(CuboBakelita1_box,
                                     BAKELITE,"CuboBakelita1_log",0,0,0);
//volumenes

for(G4int i=0;i<2;i++)
  {
  G4double PosBak1_x = 0*cm;
  G4double PosBak1_y = 0*cm;
  G4double PosBak1_z = (2.5+2.2*i)*mm;

  CuboBakelita1_phys = new G4PVPlacement(0,G4ThreeVector(PosBak1_x,PosBak1_y,PosBak1_z),
                                      CuboBakelita1_log,"CuboBakelita1",VolumenGeneral_log,false,i);
}

////////////////CUBO GRAFITO
  G4double CuboGra_x = 30.0*cm;
  G4double CuboGra_y = 30.0*cm;
  G4double CuboGra_z = 0.1*mm;
  G4Box* CuboGra_box
    = new G4Box("CuboGra_box",CuboGra_x,CuboGra_y,CuboGra_z);
  CuboGrafito_log = new G4LogicalVolume(CuboGra_box,
                                          GRAPHITE,"CuboGrafito_log",0,0,0);
for(G4int i=0;i<2;i++)
  {
 G4double PosGra_x = 0*cm;
```

```
  G4double PosGra_y =0*cm;
  G4double PosGra_z =(0.5+6.2*i)*mm;
  CuboGrafito_phys = new G4PVPlacement(0,G4ThreeVector(PosGra_x,PosGra_y,PosGra_z),
                                    CuboGrafito_log,"CuboGrafito",VolumenGeneral_log,false,i);
  }
///////////CUBO MYLAR
  G4double CuboMylar_x = 30.0*cm;
  G4double CuboMylar_y = 30.0*cm;
  G4double CuboMylar_z = 0.15*mm;
  G4Box* CuboMylar_box
    = new G4Box("CuboMylar_box",CuboMylar_x,CuboMylar_y,CuboMylar_z);
  CuboMylar_log = new G4LogicalVolume(CuboMylar_box,
                                      MYLAR,"CuboMylar_log",0,0,0);
for(G4int i=0;i<2;i++)
  {
  G4double PosMylar_x = 0*cm;
  G4double PosMylar_y =0*cm;
  G4double PosMylar_z =(0.25+6.7*i)*mm;
  CuboMylar_phys = new G4PVPlacement(0,G4ThreeVector(PosMylar_x,PosMylar_y,PosMylar_z),
                                    CuboMylar_log,"CuboMylar",VolumenGeneral_log,false,i);

  }
///// cubos de aluminios
  G4double CubAl_x = 30.0*cm;
  G4double CubAl_y = 30.0*cm;
  G4double CubAl_z = 0.1*mm;
  G4Box* CuboAluminio_box
    = new G4Box("CubAl_box",CubAl_x,CubAl_y,CubAl_z);
  CuboAluminio_log = new G4LogicalVolume(CuboAluminio_box,
                                     ALUMINIO,"CubAl_log",0,0,0);


  CuboAluminio_phys = new G4PVPlacement(0,G4ThreeVector(),
                                      CuboAluminio_log,"CubAl",VolumenGeneral_log,false,0);


  /////////CUBOS SEGUNDA RPC
//---------------------------------------VOLUMEN DE GAS

  G4double PosGas2_x = 0*cm;
  G4double PosGas2_y =0*cm;
  G4double PosGas2_z =-4.35*mm;
  CuboGas_phys = new G4PVPlacement(0,G4ThreeVector(PosGas2_x,PosGas2_y,PosGas2_z),
                                    CuboGas_log,"CuboGas",VolumenGeneral_log,false,0);

  G4double PosGas3_x = 0*cm;
  G4double PosGas3_y =0*cm;
  G4double PosGas3_z =-3.35*mm;
  CuboGas1_phys = new G4PVPlacement(0,G4ThreeVector(PosGas3_x,PosGas3_y,PosGas3_z),
                                    CuboGas1_log,"CuboGas1",VolumenGeneral_log,false,0);


//-------------volumen de BAKELITA

//volumenes

for(G4int i=0;i<2;i++)
  {
  G4double PosBak2_x = 0*cm;
  G4double PosBak2_y = 0*cm;
  G4double PosBak2_z = (-1.5-4.2*i)*mm;

  CuboBakelita_phys = new G4PVPlacement(0,G4ThreeVector(PosBak2_x,PosBak2_y,PosBak2_z),
                                      CuboBakelita_log,"CuboBakelita",VolumenGeneral_log,false,i);
  }
```

```
//////////capas de bakelita
for(G4int i=0;i<2;i++)
  {
  G4double PosBak3_x = 0*cm;
  G4double PosBak3_y = 0*cm;
  G4double PosBak3_z = (-2.5-2.2*i)*mm;
  CuboBakelita1_phys = new G4PVPlacement(0,G4ThreeVector(PosBak3_x,PosBak3_y,PosBak3_z),
                                         CuboBakelita1_log,"CuboBakelita1",VolumenGeneral_log,false,i);
}

////////////////CUBO GRAFITO

for(G4int i=0;i<2;i++)
  {
 G4double PosGra1_x = 0*cm;
  G4double PosGra1_y =0*cm;
  G4double PosGra1_z =(-0.5-6.2*i)*mm;
  CuboGrafito_phys = new G4PVPlacement(0,G4ThreeVector(PosGra1_x,PosGra1_y,PosGra1_z),
                                       CuboGrafito_log,"CuboGrafito",VolumenGeneral_log,false,i);
  }
//////////CUBO MYLAR

for(G4int i=0;i<2;i++)
  {
  G4double PosMylar1_x = 0*cm;
  G4double PosMylar1_y =0*cm;
  G4double PosMylar1_z =(-0.25-6.7*i)*mm;
  CuboMylar_phys = new G4PVPlacement(0,G4ThreeVector(PosMylar1_x,PosMylar1_y,PosMylar1_z),
                                       CuboMylar_log,"CuboMylar",VolumenGeneral_log,false,i);

  }

//////////barra de aluminio

  G4double CubAl1_x = 30.0*cm;
  G4double CubAl1_y = 30.0*cm;
  G4double CubAl1_z = 5*mm;
  G4Box* CuboAluminio1_box
    = new G4Box("CubAl1_box",CubAl1_x,CubAl1_y,CubAl1_z);
  CuboAluminio_log = new G4LogicalVolume(CuboAluminio1_box,
                                         ALUMINIO,"CubAl_log",0,0,0);
  G4double PosAl1_x = 0*cm;
  G4double PosAl1_y =0*cm;
  G4double PosAl1_z =-11.95*mm;
  CuboAluminio_phys = new G4PVPlacement(0,G4ThreeVector(PosAl1_x,PosAl1_y,PosAl1_z),
                                         CuboAluminio_log,"CubAl1",VolumenGeneral_log,false,0);

 //////////atributos de colores

  G4VisAttributes* Cubo1Att = new G4VisAttributes(G4Colour(0.0,0.0,1.0));
  Cubo1Att ->SetForceSolid(true);
  CuboGas_log->SetVisAttributes(Cubo1Att);

  G4VisAttributes * Cubo2Att
    = new G4VisAttributes(G4Colour(1.0,0.0,0.2));
 Cubo2Att ->SetForceSolid(true);
 CuboBakelita_log->SetVisAttributes(Cubo2Att);

   G4VisAttributes * Cubo3Att
    = new G4VisAttributes(G4Colour(1.0,2.0,0.8));
  Cubo3Att ->SetForceSolid(true);
  CuboGrafito_log->SetVisAttributes(Cubo3Att);

 G4VisAttributes* Cubo4Att = new G4VisAttributes(G4Colour(0.2,0.0,0.17));
  Cubo4Att ->SetForceSolid(true);
```

```
  CuboMylar_log->SetVisAttributes(Cubo4Att);

 G4VisAttributes * Cubo5Att
   = new G4VisAttributes(G4Colour(0.0,1.0,0.0));
  Cubo5Att ->SetForceSolid(true);
  CuboAluminio_log->SetVisAttributes(Cubo5Att);



  return VolumenGeneral_phys;
}
```

B.3.2. DetectorMessenger.cc

```
#include "DetectorMessenger.hh"
#include "DetectorConstruction.hh"
#include "G4UIdirectory.hh"
#include "G4UIcmdWithAString.hh"
#include "G4UIcmdWithADoubleAndUnit.hh"
#include "G4UIcmdWithoutParameter.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DetectorMessenger::DetectorMessenger(DetectorConstruction * Det)
:Detector(Det)
{
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DetectorMessenger::~DetectorMessenger()
{
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DetectorMessenger::SetNewValue(G4UIcommand*, G4String)
{
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.3. PhysicsList.cc

```
#include "globals.hh"
#include "PhysicsList.hh"
#include "G4ProcessManager.hh"
#include "G4ParticleTypes.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
PhysicsList::PhysicsList()
: G4VUserPhysicsList() {
  defaultCutValue = 0.1*mm;
   SetVerboseLevel(1);
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo
PhysicsList::~PhysicsList() {}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo
void PhysicsList::ConstructParticle() {
  // In this method, static member functions should be called
  // for all particles which you want to use.
  // This ensures that objects of these particle types will be
  // created in the program.
  ConstructBosons();
  ConstructLeptons();
  ConstructMesons();
  ConstructBaryons();
```

```
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo
void PhysicsList::ConstructBosons() {
  // pseudo-particles
  //G4Geantino::GeantinoDefinition();
  //G4ChargedGeantino::ChargedGeantinoDefinition();
  // gamma
  G4Gamma::GammaDefinition();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo
void PhysicsList::ConstructLeptons() {
  // leptons
  // e+/-
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
  // mu+/-
  G4MuonPlus::MuonPlusDefinition();
  G4MuonMinus::MuonMinusDefinition();
  // nu_e
  G4NeutrinoE::NeutrinoEDefinition();
  G4AntiNeutrinoE::AntiNeutrinoEDefinition();
  // nu_mu
  G4NeutrinoMu::NeutrinoMuDefinition();
  G4AntiNeutrinoMu::AntiNeutrinoMuDefinition();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOoo
void PhysicsList::ConstructMesons() {
  // mesons
  //    light mesons
  G4PionPlus::PionPlusDefinition();
  G4PionMinus::PionMinusDefinition();
  G4PionZero::PionZeroDefinition();
  G4Eta::EtaDefinition();
  G4EtaPrime::EtaPrimeDefinition();
  G4KaonPlus::KaonPlusDefinition();
  G4KaonMinus::KaonMinusDefinition();
  G4KaonZero::KaonZeroDefinition();
  G4AntiKaonZero::AntiKaonZeroDefinition();
  G4KaonZeroLong::KaonZeroLongDefinition();
  G4KaonZeroShort::KaonZeroShortDefinition();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOoo
void PhysicsList::ConstructBaryons() {
  // barions
  G4Proton::ProtonDefinition();
  G4AntiProton::AntiProtonDefinition();
  G4Neutron::NeutronDefinition();
  G4AntiNeutron::AntiNeutronDefinition();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOoo
void PhysicsList::ConstructProcess() {
  AddTransportation();
  ConstructEM();
  ConstructGeneral();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4MultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
#include "G4MuIonisation.hh"
#include "G4MuBremsstrahlung.hh"
```

```
#include "G4MuPairProduction.hh"
#include "G4hIonisation.hh"
#include "G4StepLimiter.hh"
#include "G4UserSpecialCuts.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
void PhysicsList::ConstructEM() {
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    G4String particleName = particle->GetParticleName();
    if (particleName == "gamma") {
      // gamma
      pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
      pmanager->AddDiscreteProcess(new G4ComptonScattering);
      pmanager->AddDiscreteProcess(new G4GammaConversion);
    } else if (particleName ==  "e-") {
      //electron
      pmanager->AddProcess(new  G4MultipleScattering,-1, 1,1);
      pmanager->AddProcess(new  G4eIonisation,       -1, 2,2);
      pmanager->AddProcess(new  G4eBremsstrahlung,   -1, 3,3);
    } else if (particleName == "e+") {
      //positron
      pmanager->AddProcess(new G4MultipleScattering,-1, 1,1);
      pmanager->AddProcess(new G4eIonisation,        -1, 2,2);
      pmanager->AddProcess(new G4eBremsstrahlung,    -1, 3,3);
      pmanager->AddProcess(new G4eplusAnnihilation, 0,-1,4);
    } else if( particleName ==  "mu+" ||
               particleName ==  "mu-"    ) {
      //muon
      pmanager->AddProcess(new  G4MultipleScattering,-1,  1,1);
      pmanager->AddProcess(new  G4MuIonisation,      -1,  2,2);
      pmanager->AddProcess(new  G4MuBremsstrahlung, -1,   3,3);
      pmanager->AddProcess(new  G4MuPairProduction, -1,   4,4);
    } else if ((!particle->IsShortLived()) &&
          (particle->GetPDGCharge() != 0.0) &&
          (particle->GetParticleName() != "chargedgeantino")) {
      //all others charged particles except geantino
      pmanager->AddProcess(new G4MultipleScattering,-1, 1,1);
      pmanager->AddProcess(new G4hIonisation,        -1, 2,2);
      //step limit
      ////pmanager->AddProcess(new G4StepLimiter,         -1,-1,3);
      ///pmanager->AddProcess(new G4UserSpecialCuts,     -1,-1,4);
    }
  }
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo..
#include "G4Decay.hh"
void PhysicsList::ConstructGeneral() {
  // Add Decay Process
  G4Decay* theDecayProcess = new G4Decay();
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    if (theDecayProcess->IsApplicable(*particle)) {
      pmanager ->AddProcess(theDecayProcess);
      // set ordering for PostStepDoIt and AtRestDoIt
      pmanager ->SetProcessOrdering(theDecayProcess, idxPostStep);
      pmanager ->SetProcessOrdering(theDecayProcess, idxAtRest);
    }
  }
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo..
void PhysicsList::SetCuts() {
```

```
SetCutsWithDefault();
 if (verboseLevel>0) DumpCutValuesTable();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo...
```

B.3.4. PhysListEmStandard.cc

```
#include "PhysListEmStandard.hh"
#include "G4ParticleDefinition.hh"
#include "G4ProcessManager.hh"
#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4MultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
#include "G4MuIonisation.hh"
#include "G4MuBremsstrahlung.hh"
#include "G4MuPairProduction.hh"
#include "G4hIonisation.hh"
#include "G4ionIonisation.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
PhysListEmStandard::PhysListEmStandard(const G4String& name)
   :  G4VPhysicsConstructor(name)
{}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
PhysListEmStandard::~PhysListEmStandard()
{}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void PhysListEmStandard::ConstructProcess()
{
  // Add standard EM Processes
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    G4String particleName = particle->GetParticleName();

    if (particleName == "gamma") {
      // gamma
      pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
      pmanager->AddDiscreteProcess(new G4ComptonScattering);
      pmanager->AddDiscreteProcess(new G4GammaConversion);

    } else if (particleName == "e-") {
      //electron
      pmanager->AddProcess(new G4MultipleScattering,   -1, 1, 1);
      pmanager->AddProcess(new G4eIonisation,          -1, 2, 2);
      pmanager->AddProcess(new G4eBremsstrahlung(),    -1, 3, 3);

    } else if (particleName == "e+") {
      //positron
      pmanager->AddProcess(new G4MultipleScattering,   -1, 1, 1);
      pmanager->AddProcess(new G4eIonisation,          -1, 2, 2);
      pmanager->AddProcess(new G4eBremsstrahlung(),    -1, 3, 3);
      pmanager->AddProcess(new G4eplusAnnihilation,     0,-1, 4);

    } else if( particleName == "mu+" ||
               particleName == "mu-"    ) {
      //muon
```

```
      pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
      pmanager->AddProcess(new G4MuIonisation,       -1, 2,2);
      pmanager->AddProcess(new G4MuBremsstrahlung,   -1, 3,3);
      pmanager->AddProcess(new G4MuPairProduction,   -1, 4,4);

    } else if( particleName == "alpha" ||
               particleName == "He3" ||
               particleName == "GenericIon" ) {
      pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
      pmanager->AddProcess(new G4ionIonisation,      -1, 2,2);

    } else if ((!particle->IsShortLived()) &&
               (particle->GetPDGCharge() != 0.0) &&
               (particle->GetParticleName() != "chargedgeantino")) {
      //all others charged particles except geantino
      pmanager->AddProcess(new G4MultipleScattering, -1,1,1);
      pmanager->AddProcess(new G4hIonisation,        -1,2,2);
    }
  }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.5. PhysListEmPolarized.cc

```
#include "PhysListEmPolarized.hh"
#include "G4ParticleDefinition.hh"
#include "G4ProcessManager.hh"
#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4MultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
#include "G4PolarizedCompton.hh"
#include "G4PolarizedGammaConversion.hh"
#include "G4ePolarizedIonisation.hh"
#include "G4ePolarizedBremsstrahlung.hh"
#include "G4eplusPolarizedAnnihilation.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

PhysListEmPolarized::PhysListEmPolarized(const G4String& name)
   :  G4VPhysicsConstructor(name)
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

PhysListEmPolarized::~PhysListEmPolarized()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void PhysListEmPolarized::ConstructProcess()
{
  // Add standard EM Processes

  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
```

```
    G4String particleName = particle->GetParticleName();

    if (namePhysics=="polarized") {
      if (particleName == "gamma") {
        pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
        pmanager->AddDiscreteProcess(new G4PolarizedCompton);
        pmanager->AddDiscreteProcess(new G4PolarizedGammaConversion);
      }
      else if (particleName == "e-") {
        //electron
        pmanager->AddProcess(new G4MultipleScattering,   -1,1,1);
        pmanager->AddProcess(new G4ePolarizedIonisation, -1,2,2);
        pmanager->AddProcess(new G4ePolarizedBremsstrahlung,      -1,3,3);
      }
      else if (particleName == "e+") {
        //positron
        pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
        pmanager->AddProcess(new G4ePolarizedIonisation, -1, 2,2);
        pmanager->AddProcess(new G4ePolarizedBremsstrahlung,    -1, 3,3);
        pmanager->AddProcess(new G4eplusPolarizedAnnihilation,   0,-1,4);
      }
    }
    else {

      if (particleName == "gamma") {
        // gamma
        pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
        pmanager->AddDiscreteProcess(new G4ComptonScattering);

        pmanager->AddDiscreteProcess(new G4GammaConversion);
      } else if (particleName == "e-") {
        //electron
        pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
        pmanager->AddProcess(new G4eIonisation,        -1, 2,2);
        pmanager->AddProcess(new G4eBremsstrahlung,    -1, 3,3);

      } else if (particleName == "e+") {
        //positron
        pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
        pmanager->AddProcess(new G4eIonisation,        -1, 2,2);
        pmanager->AddProcess(new G4eBremsstrahlung,    -1, 3,3);
        pmanager->AddProcess(new G4eplusAnnihilation,   0,-1,4);
      }
    }
  }

}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.6. PrimaryGeneratorAction.cc

```
#include "PrimaryGeneratorAction.hh"
#include "DetectorConstruction.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "globals.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
PrimaryGeneratorAction::PrimaryGeneratorAction()
```

```
{
  particleGun  = new G4ParticleGun(1);
  G4ParticleDefinition* particle
           = G4ParticleTable::GetParticleTable()->FindParticle("mu-");
  particleGun->SetParticleDefinition(particle);
  particleGun->SetParticleEnergy(500.0*GeV);

  }
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

PrimaryGeneratorAction::~PrimaryGeneratorAction()
{
  delete particleGun;
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  particleGun->SetParticlePosition(G4ThreeVector(0., 0., -40.0*mm));
  particleGun->GeneratePrimaryVertex(anEvent);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.7. EventAction.cc

```
#include "EventAction.hh"
#include "EventActionMessenger.hh"
#include "G4Event.hh"
#include "G4TrajectoryContainer.hh"
#include "G4Trajectory.hh"
#include "G4VVisManager.hh"
#include "G4UnitsTable.hh"
#include "RunAction.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
EventAction::EventAction(RunAction * ra)
  :printModulo(10000),eventMessenger(0),runAction(ra)
{
  eventMessenger = new EventActionMessenger(this);
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
EventAction::~EventAction()
{
  delete eventMessenger;
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void EventAction::BeginOfEventAction(const G4Event* evt)
{
 G4int evtNb = evt->GetEventID();

 //printing survey
 if (evtNb%printModulo == 0)
    G4cout << "\n---> Begin of Event: " << evtNb << G4endl;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void EventAction::EndOfEventAction(const G4Event* evt)
{
  if (runAction) runAction->EventFinished();
  if (G4VVisManager::GetConcreteInstance())
  {
```

```
    G4TrajectoryContainer* trajectoryContainer = evt->GetTrajectoryContainer();
    G4int n_trajectories = 0;
    if (trajectoryContainer) n_trajectories = trajectoryContainer->entries();
    for (G4int i=0; i<n_trajectories; i++) {
       G4Trajectory* trj = (G4Trajectory*)
                                    ((*(evt->GetTrajectoryContainer()))[i]);
       trj->DrawTrajectory(1000);
    }
  }
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.8. EventActionMessenger.cc

```
#include "EventActionMessenger.hh"
#include "EventAction.hh"
#include "G4UIdirectory.hh"
#include "G4UIcmdWithAnInteger.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
EventActionMessenger::EventActionMessenger(EventAction* EvAct)
:eventAction(EvAct)
{
  eventDir = new G4UIdirectory("/testem/event/");
  eventDir ->SetGuidance("event control");

  PrintCmd = new G4UIcmdWithAnInteger("/testem/event/printModulo",this);
  PrintCmd->SetGuidance("Print events modulo n");
  PrintCmd->SetParameterName("EventNb",false);
  PrintCmd->SetRange("EventNb>0");
  PrintCmd->AvailableForStates(G4State_PreInit,G4State_Idle);
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
EventActionMessenger::~EventActionMessenger()
{
  delete PrintCmd;
  delete eventDir;
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void EventActionMessenger::SetNewValue(G4UIcommand* command, G4String newValue)
{
  if (command == PrintCmd)
    {eventAction->SetPrintModulo(PrintCmd->GetNewIntValue(newValue));}
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.9. RunAction.cc

```
#include "RunAction.hh"
#include "DetectorConstruction.hh"
#include "PrimaryGeneratorAction.hh"
#include "G4Run.hh"
#include "G4RunManager.hh"
#include "G4UnitsTable.hh"
#include "G4EmCalculator.hh"
#include "Randomize.hh"
#include <iomanip>
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
RunAction::RunAction(DetectorConstruction* det, PrimaryGeneratorAction*prim)
  : detector(det), primary(prim), ProcCounter(0)
```

```
{
  totalEventCount=0;
 }
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
RunAction::~RunAction()
{ }
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void RunAction::BeginOfRunAction(const G4Run* aRun)
{
  G4cout << "### Run " << aRun->GetRunID() << " start." << G4endl;

  if (ProcCounter) delete ProcCounter;
  ProcCounter = new ProcessesCount;
  totalEventCount = 0;
  photonStats.Clear();
  electronStats.Clear();
  positronStats.Clear();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void RunAction::FillData(const G4String & particleName,
                         G4double kinEnergy, G4double costheta,
                         G4double /* phi*/,
                         G4double longitudinalPolarization)
{
  if (particleName=="gamma")
    photonStats.FillData(kinEnergy, costheta);
  else if (particleName=="e-")
    electronStats.FillData(kinEnergy, costheta);
  else if (particleName=="e+")
    positronStats.FillData(kinEnergy, costheta);
}
void RunAction::CountProcesses(G4String procName)
{
  size_t nbProc = ProcCounter->size();
  size_t i = 0;
  while ((i<nbProc)&&((*ProcCounter)[i]->GetName()!=procName)) i++;
  if (i == nbProc) ProcCounter->push_back( new OneProcessCount(procName));

  (*ProcCounter)[i]->Count();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void RunAction::EndOfRunAction(const G4Run* aRun)
{
  G4int NbOfEvents = aRun->GetNumberOfEvent();
  if (NbOfEvents == 0) return;
  G4int  prec = G4cout.precision(5);
  G4ParticleDefinition* particle =
                          primary->GetParticleGun()->GetParticleDefinition();
  G4String Particle = particle->GetParticleName();
  G4double energy = primary->GetParticleGun()->GetParticleEnergy();
  G4cout << "\n The run consists of " << NbOfEvents << " "<< Particle << " of "
         << G4BestUnit(energy,"Energy") << " through "
        ")" << G4endl;

  //frequency of processes
  G4cout << "\n Process calls frequency --->\n";
  for (size_t i=0; i< ProcCounter->size();i++) {
     G4String procName = (*ProcCounter)[i]->GetName();
     G4int    count    = (*ProcCounter)[i]->GetCounter();
     G4cout << "\t" << procName << " = " << count<<"\n";
  }
```

```
  if (totalEventCount == 0) return;

  G4cout<<" Gamma: \n";
  photonStats.PrintResults(totalEventCount);
  G4cout<<" Electron: \n";
  electronStats.PrintResults(totalEventCount);
  G4cout<<" Positron: \n";
  positronStats.PrintResults(totalEventCount);

  G4cout.precision(prec);

  CLHEP::HepRandom::showEngineStatus();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void RunAction::EventFinished()
{
  ++totalEventCount;
  photonStats.EventFinished();
  electronStats.EventFinished();
  positronStats.EventFinished();
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

RunAction::ParticleStatistics::ParticleStatistics()
  : currentNumber(0),
    totalNumber(0), totalNumber2(0),
    sumEnergy(0), sumEnergy2(0),
    sumPolarization(0), sumPolarization2(0),
    sumCosTheta(0), sumCosTheta2(0)
{}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
RunAction::ParticleStatistics::~ParticleStatistics()
{}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void RunAction::ParticleStatistics::EventFinished()
{
  totalNumber+=currentNumber;
  totalNumber2+=currentNumber*currentNumber;
  currentNumber=0;
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void RunAction::ParticleStatistics:: FillData(G4double kinEnergy,
                                              G4double costheta,
                                              G4double longitudinalPolarization)
{
  ++currentNumber;
  sumEnergy+=kinEnergy;
  sumEnergy2+=kinEnergy*kinEnergy;
  sumPolarization+=longitudinalPolarization;
  sumPolarization2+=longitudinalPolarization*longitudinalPolarization;
  sumCosTheta+=costheta;
  sumCosTheta2+=costheta*costheta;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void RunAction::ParticleStatistics::PrintResults(G4int totalNumberOfEvents)
{
  G4cout<<"Mean Number per Event :"
        <<G4double(totalNumber)/G4double(totalNumberOfEvents)<<"\n";
  if (totalNumber==0) totalNumber=1;
  G4double energyMean=sumEnergy/totalNumber;
```

```
  G4double energyRms=std::sqrt(sumEnergy2/totalNumber-energyMean*energyMean);
  G4cout<<"Mean Energy :"<< G4BestUnit(energyMean,"Energy")
        <<" +- "<<G4BestUnit(energyRms,"Energy")<<"\n";
  G4double polarizationMean=sumPolarization/totalNumber;
  G4double polarizationRms=
    std::sqrt(sumPolarization2/totalNumber-polarizationMean*polarizationMean);
  G4cout<<"Mean Polarization :"<< polarizationMean
        <<" +- "<<polarizationRms<<"\n";
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void RunAction::ParticleStatistics::Clear()
{
  currentNumber=0;
  totalNumber=totalNumber2=0;
  sumEnergy=sumEnergy2=0;
  sumPolarization=sumPolarization2=0;
  sumCosTheta=sumCosTheta2=0;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.10. StepMax.cc

```
#include "StepMax.hh"
#include "StepMaxMessenger.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
StepMax::StepMax(const G4String& processName)
 : G4VDiscreteProcess(processName),MaxChargedStep(DBL_MAX)
{
  pMess = new StepMaxMessenger(this);
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

StepMax::~StepMax() { delete pMess; }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4bool StepMax::IsApplicable(const G4ParticleDefinition& particle)
{
  return (particle.GetPDGCharge() != 0.);
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void StepMax::SetMaxStep(G4double step) {MaxChargedStep = step;}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4double StepMax::PostStepGetPhysicalInteractionLength(const G4Track& aTrack,
                                                       G4double,
                                                       G4ForceCondition* condition )
{
  // condition is set to "Not Forced"
  *condition = NotForced;
  G4double ProposedStep = DBL_MAX;
  if((MaxChargedStep > 0.) &&
     (aTrack.GetVolume() != 0) &&
     (aTrack.GetVolume()->GetName() != "World"))
     ProposedStep = MaxChargedStep;
  return ProposedStep;
}
```

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4VParticleChange* StepMax::PostStepDoIt(const G4Track& aTrack, const G4Step&)
{
   // do nothing
   aParticleChange.Initialize(aTrack);
   return &aParticleChange;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.11. StepMaxMessenger.cc

```
#include "StepMaxMessenger.hh"
#include "StepMax.hh"
#include "G4UIcmdWithADoubleAndUnit.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
StepMaxMessenger::StepMaxMessenger(StepMax* stepM)
:pStepMax(stepM)
{
  StepMaxCmd = new G4UIcmdWithADoubleAndUnit("/testem/stepMax",this);
  StepMaxCmd->SetGuidance("Set max allowed step length");
  StepMaxCmd->SetParameterName("mxStep",false);
  StepMaxCmd->SetRange("mxStep>0.");
  StepMaxCmd->SetUnitCategory("Length");
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
StepMaxMessenger::~StepMaxMessenger()
{
  delete StepMaxCmd;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void StepMaxMessenger::SetNewValue(G4UIcommand* command, G4String newValue)
{
  if (command == StepMaxCmd)
    { pStepMax->SetMaxStep(StepMaxCmd->GetNewDoubleValue(newValue));}
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.3.12. SteppingAction.cc

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
#include "SteppingAction.hh"
#include "DetectorConstruction.hh"
#include "PrimaryGeneratorAction.hh"
#include "RunAction.hh"
#include "G4RunManager.hh"
#include "G4PolarizationHelper.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
SteppingAction::SteppingAction(DetectorConstruction* det,
                               PrimaryGeneratorAction* prim, RunAction* RuAct)
:detector(det), primary(prim), runAction(RuAct)
{ }
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
SteppingAction::~SteppingAction()
{ }
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

```
void SteppingAction::UserSteppingAction(const G4Step* aStep)
{

  G4StepPoint* beginPoint = aStep->GetPostStepPoint();

  G4String procName = beginPoint->GetProcessDefinedStep()->GetProcessName();
  runAction->CountProcesses(procName);

   if (   beginPoint->GetTouchableHandle()->GetVolume()==detector->GetWorld()) {
 G4Track* aTrack = aStep->GetTrack();
  G4String particleName = aTrack->GetDynamicParticle()->GetDefinition()->GetParticleName();

 //G4cout << "particula " << particleName << G4endl;
 G4ThreeVector position  = beginPoint->GetPosition();
 G4double positionX = beginPoint->GetPosition().x();
 G4double positionY = beginPoint->GetPosition().y();
 G4double positionZ = beginPoint->GetPosition().z();
 G4ThreeVector direction = beginPoint->GetMomentumDirection();
 G4double kinEnergy = beginPoint->GetKineticEnergy();
 G4ThreeVector beamDirection = primary->GetParticleGun()->GetParticleMomentumDirection();
 G4double costheta=direction*beamDirection;

 /*{
  if (particleName=="e-")
  G4cout << "" << positionX << G4endl;
 }

 {
  if (particleName=="e-")
  G4cout << "" << positionY << G4endl;
 }
 {
  if (particleName=="e-")
  G4cout << "" << positionZ << G4endl;
 }
 {if (particleName=="e-")
  G4cout << "" << kinEnergy  << G4endl;
 }*/
 {if (particleName=="e-")
  G4cout << "" << costheta << G4endl;
 }

  }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B.4. ARCHIVOS DE LA CARPETA INCLUDE

B.4.1. DetectorConstruction.hh

```
#ifndef DetectorConstruction_H
#define DetectorConstruction_H 1
class G4LogicalVolume;
class G4VPhysicalVolume;
#include "G4VUserDetectorConstruction.hh"
class DetectorConstruction : public G4VUserDetectorConstruction
{
  public:

    DetectorConstruction();
    ~DetectorConstruction();
```

```
    G4VPhysicalVolume* Construct();
   const G4VPhysicalVolume* GetWorld() {return CuboGas_phys;};
   //const G4VPhysicalVolume* GetWorld() {return CuboBakelita1_phys;};
 private:
    // Logical volumes
    G4LogicalVolume* VolumenGeneral_log;
    G4VPhysicalVolume* VolumenGeneral_phys;
    G4LogicalVolume* CuboGas_log;
    G4VPhysicalVolume* CuboGas_phys;
    G4LogicalVolume* CuboGas1_log;
    G4VPhysicalVolume* CuboGas1_phys;
    G4LogicalVolume* CuboBakelita_log;
    G4VPhysicalVolume* CuboBakelita_phys;
    G4LogicalVolume* CuboBakelita1_log;
    G4VPhysicalVolume* CuboBakelita1_phys;
    G4LogicalVolume* CuboGrafito_log;
    G4VPhysicalVolume* CuboGrafito_phys;
    G4LogicalVolume* CuboMylar_log;
    G4VPhysicalVolume* CuboMylar_phys;
    G4LogicalVolume* CuboAluminio_log;
    G4VPhysicalVolume* CuboAluminio_phys;
};
#endif
```

B.4.2. DetectorMessenger.hh

```
#ifndef DetectorMessenger_h
#define DetectorMessenger_h 1
#include "G4UImessenger.hh"
#include "globals.hh"
class DetectorConstruction;
class G4UIdirectory;
class G4UIcmdWithAString;
class G4UIcmdWithADoubleAndUnit;
class G4UIcmdWithoutParameter;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class DetectorMessenger: public G4UImessenger
{
  public:

    DetectorMessenger(DetectorConstruction* );
   ~DetectorMessenger();

    void SetNewValue(G4UIcommand*, G4String);

  private:

    DetectorConstruction*      Detector;

};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.3. PhysicsList.hh

```
#ifndef PhysicsList_h
```

```
#define PhysicsList_h 1
#include "G4VModularPhysicsList.hh"
#include "globals.hh"
//class PhysicsListMessenger;
//class G4VPhysicsConstructor;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class PhysicsList: public G4VModularPhysicsList
{
  public:
    PhysicsList();
   ~PhysicsList();
protected:
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();
   protected:
     // these methods Construct particles
     void ConstructBosons();
     void ConstructLeptons();
     void ConstructMesons();
     void ConstructBaryons();
   protected:
   // these methods Construct physics processes and register them
     void ConstructGeneral();
     void ConstructEM();
};
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOO
#endif
```

B.4.4. PhysicsListMessenger.hh

```
#ifndef PhysicsListMessenger_h
#define PhysicsListMessenger_h 1
#include "G4UImessenger.hh"
#include "globals.hh"
class PhysicsList;
class G4UIdirectory;
class G4UIcmdWithADoubleAndUnit;
class G4UIcmdWithAString;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class PhysicsListMessenger: public G4UImessenger
{
  public:

    PhysicsListMessenger(PhysicsList* );
   ~PhysicsListMessenger();

    void SetNewValue(G4UIcommand*, G4String);

  private:

    PhysicsList*              pPhysicsList;

    G4UIdirectory*            physDir;
    G4UIcmdWithADoubleAndUnit* gammaCutCmd;
    G4UIcmdWithADoubleAndUnit* electCutCmd;
    G4UIcmdWithADoubleAndUnit* protoCutCmd;
    G4UIcmdWithADoubleAndUnit* allCutCmd;
    G4UIcmdWithAString*       pListCmd;

};
```

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.5. PhysListEmStandard.hh

```
#ifndef PhysListEmStandard_h
#define PhysListEmStandard_h 1
#include "G4VPhysicsConstructor.hh"
#include "globals.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class PhysListEmStandard : public G4VPhysicsConstructor
{
  public:
    PhysListEmStandard(const G4String& name = "standard");
    ~PhysListEmStandard();

  public:
    // This method is dummy for physics
    void ConstructParticle() {};

    // This method will be invoked in the Construct() method.
    // each physics process will be instantiated and
    // registered to the process manager of each particle type
    void ConstructProcess();
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.6. PhysListEmPolarized.hh

```
#ifndef PhysListEmPolarized_h
#define PhysListEmPolarized_h 1
#include "G4VPhysicsConstructor.hh"
#include "globals.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class PhysListEmPolarized : public G4VPhysicsConstructor
{
  public:
    PhysListEmPolarized(const G4String& name = "polarized");
   ~PhysListEmPolarized();

  public:
    // This method is dummy for physics
    void ConstructParticle() {};

    // This method will be invoked in the Construct() method.
    // each physics process will be instantiated and
    // registered to the process manager of each particle type
    void ConstructProcess();
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.7. PrimaryGeneratorAction.hh

```
#ifndef PrimaryGeneratorAction_h
#define PrimaryGeneratorAction_h 1
#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4ParticleGun.hh"
#include "globals.hh"
class G4Event;
//class DetectorConstruction;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

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

  public:
    void GeneratePrimaries(G4Event* anEvent);
    G4ParticleGun* GetParticleGun() {return particleGun;};

  private:
    G4ParticleGun*        particleGun;
   // DetectorConstruction* detector;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.8. EventAction.hh

```
#ifndef EventAction_h
#define EventAction_h 1
#include "G4UserEventAction.hh"
#include "globals.hh"
class EventActionMessenger;
class RunAction;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class EventAction : public G4UserEventAction
{
  public:
    EventAction(RunAction *);
   ~EventAction();

  public:
    void BeginOfEventAction(const G4Event*);
    void   EndOfEventAction(const G4Event*);

    void SetPrintModulo(G4int val) {printModulo = val;};

  private:
    G4int                printModulo;
    EventActionMessenger* eventMessenger;
    RunAction*           runAction;
};
```

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.9. EventActionMessenger.hh

```
#ifndef EventActionMessenger_h
#define EventActionMessenger_h 1
#include "G4UImessenger.hh"
#include "globals.hh"
class EventAction;
class G4UIdirectory;
class G4UIcmdWithAnInteger;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class EventActionMessenger: public G4UImessenger
{
  public:
    EventActionMessenger(EventAction*);
   ~EventActionMessenger();

    void SetNewValue(G4UIcommand*, G4String);

  private:
    EventAction* eventAction;

    G4UIdirectory*        eventDir;
    G4UIcmdWithAnInteger* PrintCmd;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.10. RunAction.hh

```
#ifndef RunAction_h
#define RunAction_h 1
#include "G4UserRunAction.hh"
#include "ProcessesCount.hh"
#include "globals.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class DetectorConstruction;
class PrimaryGeneratorAction;
class G4Run;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class RunAction : public G4UserRunAction
{
  class ParticleStatistics {
  public:
    ParticleStatistics();
    ~ParticleStatistics();
    void EventFinished();
    void FillData(G4double kinEnergy, G4double costheta,
                  G4double longitudinalPolarization);
    void PrintResults(G4int totalNumberOfEvents);
    void Clear();
  private:
    G4int currentNumber;
    G4int totalNumber, totalNumber2;
```

```
    G4double sumEnergy, sumEnergy2;
    G4double sumPolarization, sumPolarization2;
    G4double sumCosTheta, sumCosTheta2;
  };

public:
  RunAction(DetectorConstruction*, PrimaryGeneratorAction*);
  virtual ~RunAction();

public:
  void BeginOfRunAction(const G4Run*);
  void   EndOfRunAction(const G4Run*);

  void CountProcesses(G4String);

  void FillData(const G4String & particleName,
                G4double kinEnergy, G4double costheta, G4double phi,
                G4double longitudinalPolarization);
  void EventFinished();

private:
  DetectorConstruction*   detector;
  PrimaryGeneratorAction* primary;
  ProcessesCount*         ProcCounter;


  G4int totalEventCount;

  ParticleStatistics photonStats;
  ParticleStatistics electronStats;
  ParticleStatistics positronStats;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.11. StepMax.hh

```
#ifndef StepMax_h
#define StepMax_h 1
#include "globals.hh"
#include "G4VDiscreteProcess.hh"
#include "G4ParticleDefinition.hh"
#include "G4Step.hh"
class StepMaxMessenger;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class StepMax : public G4VDiscreteProcess
{
  public:

     StepMax(const G4String& processName ="stepMax");
    ~StepMax();

     G4bool   IsApplicable(const G4ParticleDefinition&);
     void     SetMaxStep(G4double);
     G4double GetMaxStep() {return MaxChargedStep;};

     G4double PostStepGetPhysicalInteractionLength( const G4Track& track,
                                             G4double   previousStepSize,
                                             G4ForceCondition* condition);
```

```
    G4VParticleChange* PostStepDoIt(const G4Track&, const G4Step&);

    G4double GetMeanFreePath(const G4Track&, G4double, G4ForceCondition*)
      {return 0.;};     // it is not needed here !

  private:

    G4double    MaxChargedStep;
    StepMaxMessenger* pMess;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.12. StepMaxMessenger.hh

```
#ifndef StepMaxMessenger_h
#define StepMaxMessenger_h 1
#include "G4UImessenger.hh"
#include "globals.hh"
class StepMax;
class G4UIcmdWithADoubleAndUnit;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class StepMaxMessenger: public G4UImessenger
{
  public:
    StepMaxMessenger(StepMax*);
   ~StepMaxMessenger();

    void SetNewValue(G4UIcommand*, G4String);

  private:
    StepMax* pStepMax;
    G4UIcmdWithADoubleAndUnit* StepMaxCmd;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.13. SteppingAction.hh

```
#ifndef SteppingAction_h
#define SteppingAction_h 1
#include "G4UserSteppingAction.hh"
#include "globals.hh"
class DetectorConstruction;
class PrimaryGeneratorAction;
class RunAction;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

class SteppingAction : public G4UserSteppingAction
{
  public:
    SteppingAction(DetectorConstruction*, PrimaryGeneratorAction*, RunAction*);
   ~SteppingAction();

    void UserSteppingAction(const G4Step*);
```

```
  private:
    DetectorConstruction*   detector;
    PrimaryGeneratorAction* primary;
    RunAction*              runAction;

};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B.4.14. ProcessesCount.hh

```
#ifndef ProcessesCount_HH
#define ProcessesCount_HH
#include "globals.hh"
#include <vector>
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
class OneProcessCount
{
public:
    OneProcessCount(G4String name) {Name=name; Counter=0;};
   ~OneProcessCount() {};

public:
    G4String  GetName()       {return Name;};
    G4int     GetCounter()    {return Counter;};
    void      Count()         {Counter++;};

private:
    G4String Name;            // process name
    G4int    Counter;         // process counter
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

typedef std::vector<OneProcessCount*> ProcessesCount;

#endif
```

B.5 MACRO PARA CORRER EL ARCHIVO rpc.cc

```
/control/verbose 2
/run/verbose 2
/run/initialize
/gun/particle mu-
/gun/energy 500.0 GeV
/run/beamOn 1000000
```