

SOFTWARE PARA PROCESADORES
INFORME PROYECTO N° 1

PRESENTADO POR:

CAMILO ANDRESTRUJILLO MUÑOZ
JAVIER ENRIQUE HUÉRFANO DIAZ
SERGIO ANDRES ROJAS MORENO
JUAN DAVID ROJAS RODRIGUEZ

PRESENTADO PARA:

MIGUEL EDUARDO SARMIENTO JURADO

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
BOGOTA DC.

2016

CONTENIDO

Introducción

1. Objetivos
2. Diagramas de flujo
 - 2.1. Control del sistema de desplazamiento de la plataforma
 - 2.2. Control del sistema de manipulación de materiales
3. Codificación en c/c++.
 - 3.1. Control del sistema de desplazamiento de la plataforma:
 - 3.1.1. Void Desplazamiento_Avanzar.
 - 3.1.2. Void Desplazamiento_Detener.
 - 3.1.3. Void Desplazamiento_Rotar_AgujasReloj.
 - 3.1.4. Void Desplazamiento_Rotar_Opuesto_AgujasReloj
 - 3.1.5. Char S1_Lectura.
 - 3.1.6. Char S2_Lectura.
 - 3.1.7. Char S3_Lectura.
 - 3.2. Control del sistema de manipulación de materiales:
 - 3.2.1. Void Materiales_ArrojarCubo.
 - 3.2.2. Void Materiales_No_Arrojar_Cubo.
 - 3.2.1. Char S1_Lectura.
 - 3.2.2. Char S4_Lectura.
 - 3.2.3. Char S5_Lectura.
 - 3.2.4. Char S6_Lectura.
 - 3.2.5. Char S7_Lectura.
4. Loop Fusion.
5. Roles finales de cada uno de los integrantes.
6. Gráficos de las diferentes situaciones a las que se enfrenta el robot.

Conclusiones.

Referencias.

INTRODUCCION

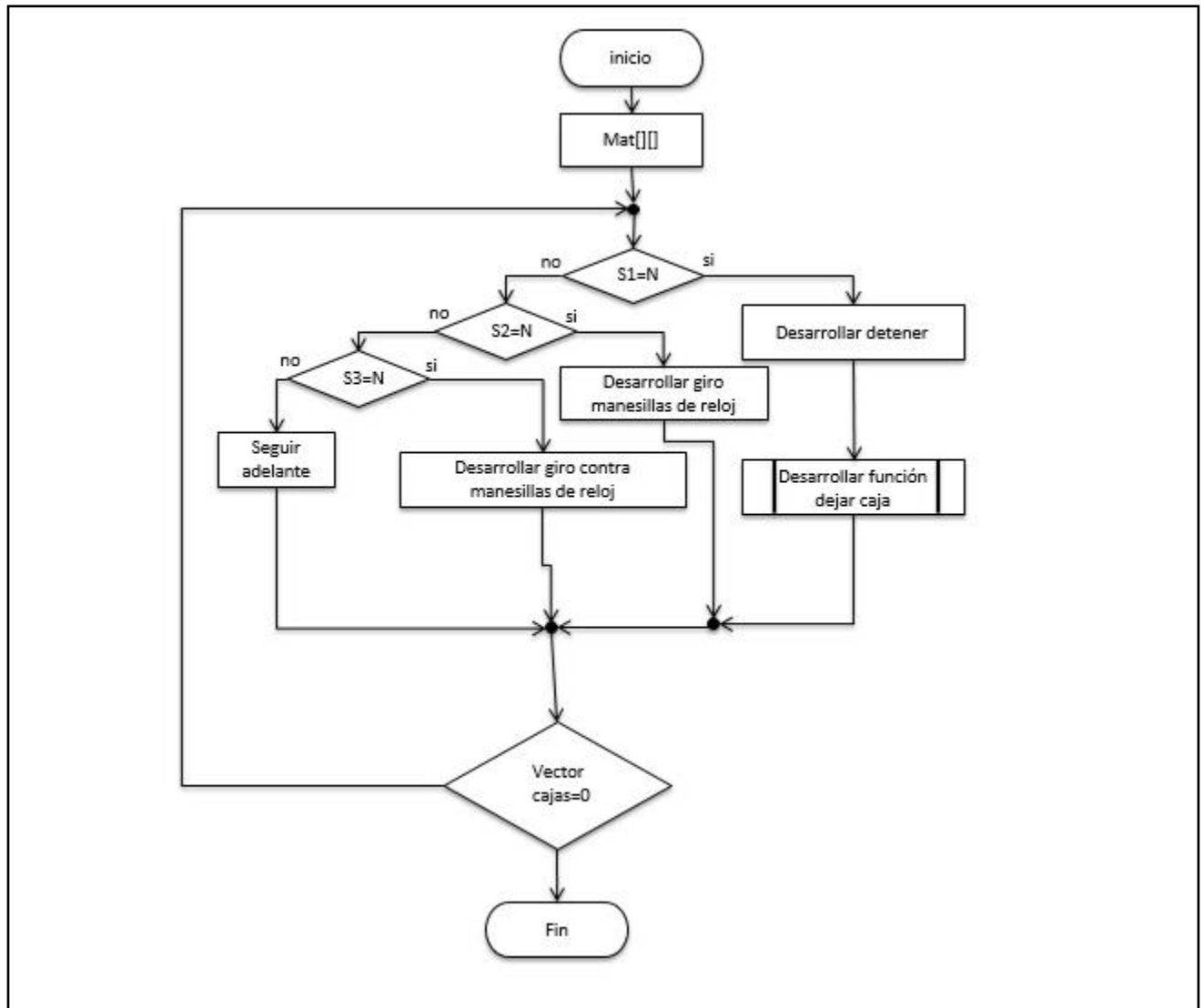
El mundo de la tecnología electrónica está en expansión constante desde hace ya un buen tiempo. Como ya sabemos, la tecnología ayuda a que el humano pueda desarrollar sus actividades diarias de una manera más sencilla, facilitando sus deberes y disminuyendo el esfuerzo que éste tiene que desempeñar para cumplir dichas labores. Como ejemplo está el arco y la flecha que como sabemos mejoró increíblemente la efectividad en la caza del hombre primitivo. Ocurre lo mismo con la tecnología electrónica, cada día está en expansión, cada día se conoce una nueva tecnología que supera a la anterior y reordena la manera en la cual vemos el mundo. De las calculadoras robustas que solo podían hacer operaciones elementales, a las actuales calculadoras capaces de hacer integrales tanto teórica como numéricamente en un tiempo muy corto. De los grandes y un poco pesados celulares a los actuales *"Smartphone"* capaces de una y otra tarea que nos deja perplejos. El avance de ésta tecnología se debe a la capacidad intelectual del ser humano, que de una manera u otra logra llevar éstas hazañas de la idea al hecho. Éste proyecto está dirigido a una de las máquinas que se ven mucho en la actualidad, los *"robots"*, que cada vez son más y más avanzados y sofisticados para hacer varias labores según el uso que se les vaya a dar. Aunque no vamos a entrar en detalle con cosas físicas como construir un *"robot"* que cumpla una labor, vamos a hacer énfasis a una de las cosas más importantes (Sino la más importante) que se tiene que tener en cuenta a la hora de utilizar un *"robot"*, y esto es su programación. Lógicamente no haremos énfasis a una programación parecida a lo que fuera una que en realidad fuese de un *"robot"* de la vida real, ya que no estaremos interactuando con ningún objeto que se adecúe a la construcción de tal, sino que estudiaremos lo más parecido a un escenario de lo que sería su programación, como estudiar diagramas de flujo y sus respectivas simulaciones en código en lenguaje C/C++.

1. OBJETIVOS

- Hacer un análisis y soluciones detalladas de problemas planteados acerca de las funcionalidades básicas de un Robot.
- Hacer diagramas de flujo y su respectiva representación en código del mecanismo de movimiento de un robot y aparte una posible implementación de éste en el campo de trabajo.
- Dar cumplimiento a las peticiones de Construcciones Robotizadas SAS, entregándole un software que implementará la solución final que requiere dicha compañía para controlar su robot.
- Desarrollarse en el ámbito profesional al enfrentarse a proyectos de calidad de ingeniería que requiere de una dedicación y sacrificio de cada uno de los integrantes del grupo de trabajo.

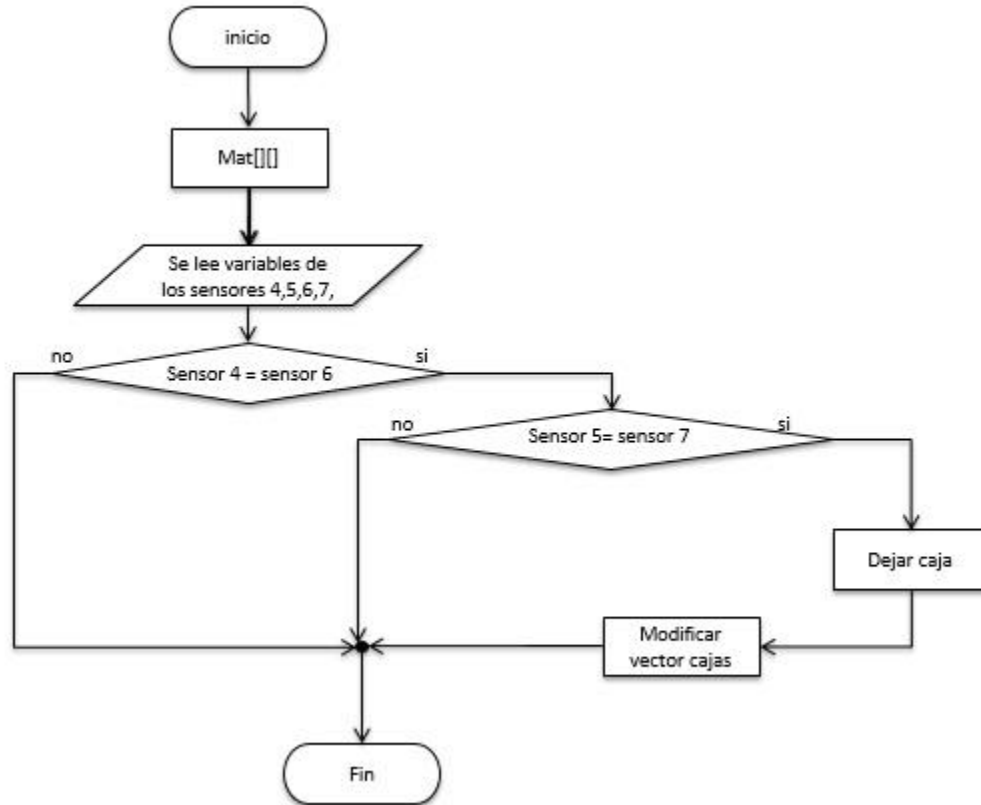
2.DIAGRAMAS DE FLUJO

2.1. Control del sistema de desplazamiento de la plataforma



Software para Procesadores @ "Electrónica Javeriana" – Bogotá – Colombia		
D. Flujo: Desplazamiento_Robot.		Supervisor: Ing. Miguel Sarmiento M.Sc.
Dedicación: 20 min		Diseñado por: Proyecto_1_3
Plancha: 1 de 2	Versión:	Fecha: 17/08/2016

2.2. Control del sistema de manipulación de materiales.



Software para Procesadores @ "Electrónica Javeriana" – Bogotá – Colombia		
Diagrama Flujo: función dejar caja		Supervisor: Ing. Miguel Sarmiento M.Sc.
Dedicación: 10 min		Diseñado por: Proyecto_1_3.
Plancha: 2 de 2	Versión:	Fecha: 31/07/2016

3.CODIFICACION EN C/C++.

3.1. Control del sistema de desplazamiento de la plataforma:

3.1.1. Void Desplazamiento_Avanzar:

Inicialmente utilizamos una matriz de 10x10 para representar la trayectoria a seguir del robot. La zona por la cual el robot va a transitar está marcada con un guion al piso ' _ ', (Los guiones representan también a los puntos en negro por los cuales el robot puede circular), y el resto de por donde el robot no circularía está marcado con un punto '.', (representa los puntos blancos por los cuales el robot no puede circular), en la figura número 1 se puede observar nuestro prototipo de la trayectoria.



Figura 1. Representación de la trayectoria del robot.

A partir de ésta trayectoria, comenzamos con lo que sería la función a la que llevaría el avance del robot de la cual su nombre, "Desplazamiento_Avanzar". Un contador llamado "dirección", es nuestro guía para poder hacer el movimiento del robot. Resulta que con un par de funciones se va a determinar hacia a donde el robot tiene que moverse, supongamos estamos en el punto (0,0) de la matriz, a la derecha hay un punto, eso le muestra al código que la siguiente dirección no puede ser hacia la derecha ya que no está en la trayectoria, y que, si debe hacerlo hacia abajo ya que hay un guion al piso o un punto en negro, que respectivamente el sensor s1 detectaría.

En el ejemplo anterior, en el que el robot se encuentra en la posición número (0,0), el robot no puede moverse ni a la derecha ni hacia la izquierda ni hacia arriba, sólo hacia abajo. Así que nuestra solución es asignarle un valor a la variable “dirección”, para que cuando se determine hacia a donde hay que mover el robot, a ésta variable se le agregue un valor entre 1 y 4 para designar el movimiento. A la variable dirección se le va a asignar el valor de 1, si el siguiente movimiento es hacia la derecha, 2 si es hacia la izquierda, 3 si es hacia abajo y 4 si es hacia arriba. En las figuras número 2,3,4 y 5 respectivamente se pueden apreciar en código como se vería representar el movimiento del robot.

```
45      if (dirreccion == 1){  
46          derecha = true;  
47          izquierda = false;  
48          abajo = false;  
49          arriba = false;  
50          b++;  
51      }  
52
```

*Figura 2. Representación en código de
un movimiento hacia la derecha.*

Para un respectivo movimiento hacia la izquierda se haría lo que ilustra la figura 3.

```
54      if (dirreccion == 2){  
55          derecha = false;  
56          izquierda = true;  
57          abajo = false;  
58          arriba = false;  
59          b--;  
60
```

*Figura 3. Representación en código de
un movimiento hacia la derecha.*

Para un respectivo movimiento abajo se haría lo que ilustra la figura 4.

```
62 if (dirreccion == 3){  
63     derecha = false;  
64     izquierda = false;  
65     abajo = true;  
66     arriba = false;  
67     a++;  
68 }
```

*Figura 4. Representación en código de
un movimiento hacia abajo.*

Para un respectivo movimiento abajo se haría lo que ilustra la figura 4.

```
69 if (dirreccion == 4){  
70     derecha = false;  
71     izquierda = false;  
72     abajo = false;  
73     arriba = true;  
74     a--;  
75 }
```

*Figura 5. Representación en código de
un movimiento hacia abajo.*

Nótese que cada vez que el robot va a hacer un movimiento, la variable booleana que queda en verdadero o “true” es la que designa hacia qué dirección se va a mover el robot. Para el ejemplo de si el robot se encuentra en el punto (0,0), en el que el movimiento es hacia abajo, pues a la variable “dirección” se le asignaría un valor de 1, y la variable booleana derecha sería la única que estaría en “true” (ver figura 2).

3.1.2. Void Desplazamiento_Detener:

Un punto crucial cuando hay que tratar con robots que van a hacer recorridos es poder llevar un control de cuando el robot tiene que detenerse. La función “Desplazamiento_Detener”, es la encargada de hacer ésta tarea. Para hacer ésta función lo que utilizamos fueron los mismos booleanos de dirección, (derecha, izquierda...etc.), ya cambiados actualizados y cambiados de valor por la función anterior lo que ocurre es que cuando se ingresa a la función “Detener”, va a empezar a comparar las variables booleanas de dirección para saber hacia a donde hay que mover el carro. Una vez sepa hacia a donde lo que va a hacer es verificar en la matriz plataforma si en el siguiente movimiento que haga el robot habrá o no una estación. Para saber esto por medio de un par de funciones a la matriz plataforma se le asignaron datos en representación de una letra ‘n’, en la cual están las estaciones, si en un punto en específico hay un carácter ‘n’ en vez de un ‘b’, quiere decir que en ése punto hay una estación y que por lo tanto el robot tiene que detenerse. En las figuras 6,7,8,9 se muestra lo que realiza la función detenerse.

```
198
199     bool Desplazamiento_Detener(terreno plataforma[][10],
200     int a,int b,bool derecha,bool izquierda,bool abajo,bool arriba){
201         bool hay_estacion = false;
202         if(derecha == true){
203             if(plataforma[a][b+1].estacion.detenerse == 'n'){
204                 cout << "detenerse"<<endl;
205                 hay_estacion = true;
206             }else{
207                 cout << "avanzar"<<endl;
208             }
209         }
```

Figura 6. Representación de como el robot debe trasladarse o detenerse.

Cuando el siguiente movimiento es a la derecha.

En la figura 6, anteriormente mostrada, se puede observar que cuando el movimiento es hacia la derecha, en el código se hacen otros dos condicionales, en los que se verifica si en la siguiente posición el robot debe seguir con su trayectoria y girar hacia la derecha o detenerse en la estación.

```

210         if(izquierda == true && b > 0){
211             if(plataforma[a][b-1].estacion.detenerse == 'n'){
212                 cout << "detenerse" << endl;
213                 hay_estacion = true;
214             }else{
215                 cout << "avanzar" << endl;
216             }
217         }

```

Figura 7. Representación de si el robot debe trasladarse o detenerse.

Cuando el siguiente movimiento es hacia la izquierda.

En la figura 7 anteriormente mostrada, se puede notar que cuando la siguiente dirección es hacia la izquierda, se vuelve a verificar si en el siguiente movimiento hay o no una estación, por lo cual el robot debe parar y en caso de que no seguir su trayecto.

```

218         if(arriba == true && a > 0){
219             if(plataforma[a-1][b].estacion.detenerse == 'n'){
220                 cout << "detenerse" << endl;
221                 hay_estacion = true;
222             }else{
223                 cout << "avanzar" << endl;
224             }
225         }

```

Figura 8. Representación de si el robot debe trasladarse o detenerse.

Cuando el siguiente movimiento es hacia arriba.

En la figura 8 anteriormente mostrada, se puede notar que cuando la siguiente dirección es hacia la arriba, se vuelve a verificar si en el siguiente movimiento hay o no una estación, por lo cual el robot debe parar y en caso de que no seguir su trayecto.

```

226         if(abajo == true){
227             if(plataforma[a+1][b].estacion.detenerse == 'n'){
228                 cout << "detenerse" << endl;
229                 hay_estacion = true;
230             }else{
231                 cout << "avanzar" << endl;
232             }
233         }

```

Figura 9. Representación de si el robot debe trasladarse o detenerse.

Cuando el siguiente movimiento es hacia abajo.

Finalmente, en la figura 9 anteriormente mostrada, se puede notar que cuando la siguiente dirección es hacia la abajo, se vuelve a verificar si en el siguiente movimiento hay o no una estación, por lo cual el robot debe parar y en caso de que no seguir su trayecto.

3.1.3. Void Desplazamiento_Rotar_Agujas_Del_Reloj:

Otro de las cosas importantes a la hora de utilizar un robot es la implementación a la hora de girar, para ésta función, por su nombre podríamos intuir que es cuando se requiera que gire hacia la derecha o en el sentido de las agujas del reloj. Para representar ésta función lo que hacemos es que cuando los booleanos anteriormente actualizados de valor lleguen a ésta función, va a empezar a compararlos y cada vez que se encuentre con un valor positivo imprimirá, "Giró hacia la derecha" y va a cambiar el valor de dirección para hacer saber que se ha desplazado. En la figura 10, se puede observar de una manera más clara el enfoque de ésta función.

```
237 int Desplazamiento_Rotar_AgujasReloj(int a,int b, bool derecha,
238 bool izquierda,bool abajo,bool arriba,int direccion){
239     if(derecha == true){
240
241         cout << "giro a la derecha"<<endl;
242         direccion = 3;
243     }
244     if(izquierda == true){
245
246         cout << "giro a la derecha"<<endl;
247         direccion = 4;
248     }
249     if(arriba == true){
250
251         cout << "giro a la derecha"<<endl;
252         direccion = 1;
253     }
254     if(abajo == true){
255
256         cout << "giro a la derecha"<<endl;
257         direccion = 2;
258     }
259     return direccion;
260 }
```

Figura 10. Representación cómo se mostraría el movimiento en sentido de las agujas del reloj, del movimiento del robot.

Nótese que en cada uno de los booleanos que sea el que esté en “true”, va a imprimir giró a la derecha, esto se debe a la funcionalidad de la función.

3.1.4. Void Desplazamiento_Rotar_Opuesto_Agujas_Del_Reloj:

Igual que en el caso anterior otro mecanismo muy importante a la hora de practicar con un robot es a la hora de girar hacia la izquierda o al contrario de las manecillas del reloj. Para esto se hizo lo mismo que en la función anterior solamente que cambiando en vez de “giró a la derecha” por “giró a la izquierda”.

```
261 int Desplazamiento_Rotar_Opuesto_AgujasReloj(int a,int b,bool derecha
262 |,bool izquierda,bool abajo,bool arriba,int direccion){
263     if(derecha == true){
264
265         cout << "giro a la izquierda"<<endl;
266         direccion = 4;
267     }
268     if(izquierda == true){
269
270         cout << "giro a la izquierda"<<endl;
271         direccion= 3;
272     }
273     if(arriba == true){
274
275         cout << "giro a la izquierda"<<endl;
276         direccion = 2;
277     }
278     if(abajo == true){
279
280         cout << "giro a la izquierda"<<endl;
281         direccion = 1;
282     }
283     return direccion;
284 }
```

Figura 11. Representación cómo se mostraría el movimiento en sentido de las agujas del reloj, del movimiento del robot.

En la figura 11, anteriormente mostrada se puede analizar cómo cambia en un poco la estructura de ésta figura a la de la anterior. Aunque en un poco mínimo el cambio, en la práctica sería algo considerable.

3.1.5. Char S1_Lectura:

Otro factor a tener en cuenta es la manera en la cual el robot se movería en un entorno un poco más real. En el programa las direcciones son tomados como booleanos. La función “Char S1_Lectura”, nos ayuda para saber el color que el robot está leyendo en el sensor 1. Cuando el color a retornar sea negro el robot seguirá normal su trayecto, pero cuando sea blanco quiere decir que la línea cambio de rumbo y verificar con los sensores 2 y 3 (encargados de un movimiento hacia la izquierda y derecha respectivamente. En la figura 12 y 13 se puede observar de una manera más clara como hace el robot esta operación.

```
92 char s1_lectura(terreno plataforma[][10],int a,int b,bool derecha,  
93                bool izquierda,bool abajo,bool arriba){  
94     char sensor = 'b' ;  
95     if(derecha == true){  
96         if(plataforma[a][b+1].trayectoria== 'n'){  
97             sensor = 'n';  
98         }else{  
99             sensor = 'b';  
100        }  
101    }  
102    if(izquierda == true&& b>0){  
103        if(plataforma[a][b-1].trayectoria== 'n'){  
104            sensor = 'n';  
105        }else{  
106            sensor = 'b';  
107        }  
108    }
```

*Figura 12. Representación función Char1_lectura
Cuando el movimiento es hacia la derecha e izquierda.*

En la figura 12 se puede observar como el robot hace lo nombrado anteriormente, para los casos de derecha e izquierda revisa si en la siguiente posición hay un carácter ‘n’, que le indique girar, por lo tanto, lo retornará de lo contrario, retornará ‘b’, o sea que por ahí no puede circular.


```

109         if(arriba == true&& a > 0){
110             if(plataforma[a-1][b].trayectoria== 'n'){
111                 sensor = 'n';
112             }else{
113                 sensor = 'b';
114             }
115         }
116         if(abajo == true){
117             if(plataforma[a+1][b].trayectoria== 'n'){
118                 sensor = 'n';
119             }else{
120                 sensor = 'b';
121             }
122         }
123
124         return sensor;

```

Figura 13. Representación función Char1_lectura

Cuando el movimiento es hacia arriba o abajo

En la figura anterior se puede observar lo que sucede cuando el robot se va a dirijir hacia la abajo o hacia arriba. En éste caso vuelve a hacer lo mismo que ocurrió anteriormente, revisa en que trayectoria está el robot y dependiendo de ésta verificará si debe o no avanzar.

3.1.6. Char S2_Lectura:

La función charS2_Lectura.Ésta función es la que nos retorna lo que está leyendo el sensor

2. En las imágenes 14 y 15 se puede apreciar cómo funciona.

```

126 char s2_lectura(terreno plataforma[][10],int a, int b
127 ,bool derecha,bool izquierda,bool abajo,bool arriba){
128     char sensor = 'b' ;
129     if(derecha == true){
130         if(plataforma[a+1][b].trayectoria== 'n'){
131             sensor = 'n';
132         }else{
133             sensor = 'b';
134         }
135     }
136     if(izquierda == true&& a > 0){
137         if(plataforma[a-1][b].trayectoria== 'n'){
138             sensor = 'n';
139         }else{
140             sensor = 'b';
141         }
142     }

```

Figura 14. Representación función Char1_lectura

Cuando el movimiento es hacia derecha o izquierda

En la figura 14 se puede observar como el robot hace lo nombrado anteriormente, para los casos de derecha e izquierda revisa si en la siguiente posición hay un carácter 'n', que le indique girar, por lo tanto, lo retornará de lo contrario, retornará 'b', o sea que por ahí no puede circular.

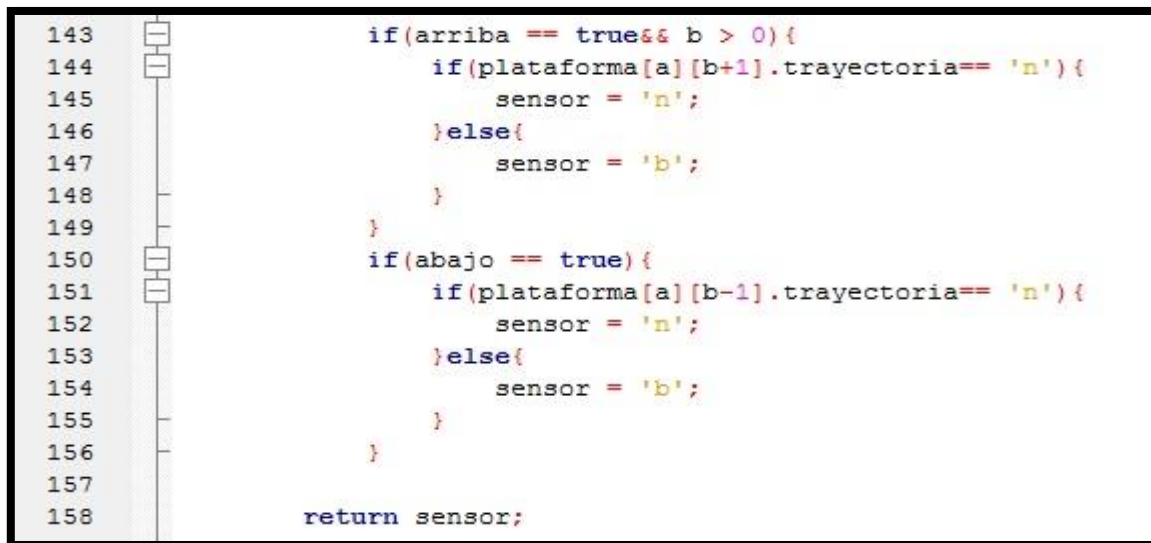


Figura 15. Representación función `Char1_lectura`

Cuando el movimiento es hacia arriba o abajo

En la figura anterior se puede observar lo que sucede cuando el robot se va a dirigir hacia la abajo o hacia arriba. En éste caso vuelve a hacer lo mismo que ocurrió anteriormente, revisa en que trayectoria está el robot y dependiendo de ésta verificará si debe o no avanzar.

3.1.7. Char S3_Lectura:

La función `charS3_Lectura`. Ésta función es la que nos retorna lo que está leyendo el sensor 3. En las imágenes 16 y 17 se puede apreciar como funciona.

```

161 char s3_lectura(terreno plataforma[][10],int a, int b,
162               bool derecha,bool izquierda,bool abajo,bool arriba){
163
164     char sensor = 'b' ;
165     if(derecha == true&& a > 0){
166         if(plataforma[a-1][b].trayectoria== 'n'){
167             sensor = 'n';
168         }else{
169             sensor = 'b';
170         }
171     }
172     if(izquierda == true){
173         if(plataforma[a+1][b].trayectoria== 'n'){
174             sensor = 'n';
175         }else{
176             sensor = 'b';
177         }

```

Figura 16. Representación función Char1_ lectura

Cuando el movimiento es hacia arriba o abajo

En la figura 17 se puede observar como el robot hace lo nombrado anteriormente, para los casos de derecha e izquierda revisa si en la siguiente posición hay un carácter 'n', que le indique girar, por lo tanto, lo retornará de lo contrario, retornará 'b', o sea que por ahí no puede circular.

```

143     if(arriba == true&& b > 0){
144         if(plataforma[a][b+1].trayectoria== 'n'){
145             sensor = 'n';
146         }else{
147             sensor = 'b';
148         }
149     }
150     if(abajo == true){
151         if(plataforma[a][b-1].trayectoria== 'n'){
152             sensor = 'n';
153         }else{
154             sensor = 'b';
155         }
156     }
157
158     return sensor;

```

Figura 17. Representación función Char1_ lectura

Cuando el movimiento es hacia arriba o abajo

En la figura anterior se puede observar lo que sucede cuando el robot se va a dirigir hacia la abajo o hacia arriba. En éste caso vuelve a hacer lo mismo que ocurrió anteriormente, revisa en que trayectoria está el robot y dependiendo de ésta verificará si debe o no avanzar.

3.2. Control del sistema de manipulación de materiales:

3.2.1. Void Materiales_ArrojarCubo.

Unos de los factores importantes a la hora de la implementación del robot es a la hora en la cual tiene que tener que saber cuándo tiene que arrojar un cubo. La función “Materiales_ArrojarCubo”, es la encargada de ésta acción. En la figura 18 se puede observar a la función “Materiales_ArrojarCubo”.

```
104 bool Materiales_ArrojarCubo(char sensor_4,char sensor_5,char sensor_6
105     , char sensor_7,int entregar,int n,int banda_transportadora[]){
106     bool entregar_caja = false;
107     int temporal = 0 ;
108     if(sensor_4 == sensor_6&&sensor_5==sensor_7){
109         entregar_caja = true;
110         for(int j = 0;j<4;j++){
111             temporal = banda_transportadora[j+1];
112             banda_transportadora[j+1] = banda_transportadora[j];
113             banda_transportadora[j]= temporal;
114         }
115         banda_transportadora[3] = 0;
116         cout << "entregar caja"<<endl;
117     }
118     return entregar_caja;
119 }
```

Figura 18. Representación función Materiales_ArrojarCubo

Lo que hace ésta función, es que cuando entra al primer condicional (Ver figura 18), lo que va a hacer es que una variable booleana “entregar_caja”, se convierta en “true”, de ésta manera se sabe que hay que entregar una caja en éste momento, y lo que va a hacer después es mover la posición del vector una posición por iteración y a la última posición le va a asignar un valor de 0, para que así sepa que ya se entregó la caja actual y faltan algunas.

Por ejemplo:

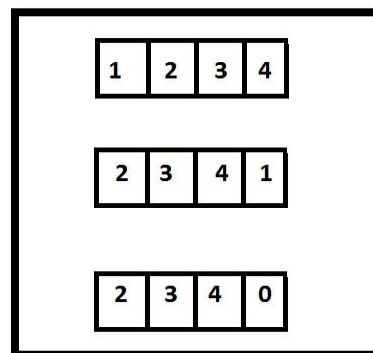
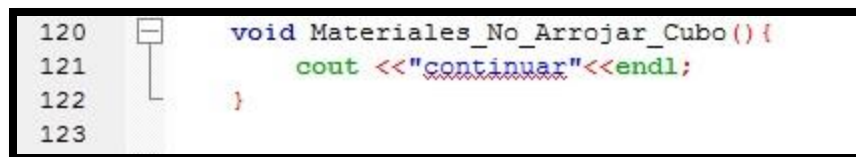


Figura 19. Representación de lo que sucede en la función.

En la figura 19, se puede observar lo que sucede con el vector “banda_transportadra”. Éste vector es el encargado de mostrarnos en cual estación debe ir la caja. Si observamos el ejemplo anterior en la posición 0 del vector, hay un numero 1, por éste motivo sabemos que para la primera caja, el robot tiene que dejarla o arrojarla en la estación 1, al moverse de posiciones el arreglo, la siguiente caja no sería la 1 sino la 2, y precisamente iría hacia la estación 2. Finalmente, lo que haría la función es a la última posición agregarle un 0, ya que es una caja que ya no está dentro del arreglo o es una caja que el robot ya habría arrojado.

3.2.2. Void Materiales_No_Arrojar_Cubo.

Ésta función es un poco simple, cuando el robot no necesite detenerse para dejar una caja simplemente seguirá con su trayectoria y no habrá cambios. Sería lo mismo que simplemente llamar a la función avanzar.



```
120 void Materiales_No_Arrojar_Cubo(){
121     cout << "continuar" << endl;
122 }
123
```

Figura 20. Función Materiales_No_ArrojarCubo.

3.2.3. Char S1_Lectura:

La función “Char S1_Lectura”, nos ayuda para saber el color que el robot está leyendo en el sensor 1. Cuando el color a retornar sea negro el robot seguirá normal su trayecto, pero cuando sea blanco quiere decir que la línea cambio de rumbo y verificar con los sensores 2 y 3 (encargados de un movimiento hacia la izquierda y derecha respectivamente. En la figura 22 y 23 se puede observar de una manera más clara como hace el robot esta operación.

```

92      char s1_lectura(terreno plataforma[][10],int a,int b,bool derecha,
93                      bool izquierda,bool abajo,bool arriba){
94          char sensor = 'b' ;
95          if(derecha == true){
96              if(plataforma[a][b+1].trayectoria== 'n'){
97                  sensor = 'n';
98              }else{
99                  sensor = 'b';
100             }
101         }
102         if(izquierda == true&& b>0){
103             if(plataforma[a][b-1].trayectoria== 'n'){
104                 sensor = 'n';
105             }else{
106                 sensor = 'b';
107             }
108         }

```

Figura 22. Representación función Char1_ lectura

Cuando el movimiento es hacia la derecha e izquierda.

En la figura 22 se puede observar como el robot hace lo nombrado anteriormente, para los casos de derecha e izquierda revisa si en la siguiente posición hay un carácter 'n', que le indique girar, por lo tanto, lo retornará de lo contrario, retornará 'b', o sea que por ahí no puede circular.

```

109          if(arriba == true&& a > 0){
110              if(plataforma[a-1][b].trayectoria== 'n'){
111                  sensor = 'n';
112              }else{
113                  sensor = 'b';
114              }
115          }
116          if(abajo == true){
117              if(plataforma[a+1][b].trayectoria== 'n'){
118                  sensor = 'n';
119              }else{
120                  sensor = 'b';
121              }
122          }
123
124      return sensor;

```

Figura 23. Representación función Char1_ lectura

Cuando el movimiento es hacia arriba o abajo

En la figura 23 se puede observar lo que sucede cuando el robot se va a dirigir hacia la abajo o hacia arriba. En éste caso vuelve a hacer lo mismo que ocurrió anteriormente, revisa en que trayectoria está el robot y dependiendo de ésta verificará si debe o no avanzar.

3.2.4. Char S4_Lectura:

Este carácter es de suma importancia, en conjunto con el carácter s5 hacen la búsqueda de un carácter 'n', en plataforma. Identificación para así saber en qué estación pertenece dicha caja. Si hay un carácter 'n' en el módulo buscador quiere decir que hay una estación y por tanto va a la función del sensor s5 para buscar la otra parte del código. De modo que para encontrar en que estación cumple un algoritmo, cuando el sensor s4 envíe negro y el sensor s5 encuentra un blanco quiere decir que están en la estación uno, si fuera al revés, el 4 en blanco y el 5 en negro sería la estación 2 y así sucesivamente. En la figura 24 se puede observar la funcionalidad de éste sensor en código.

```
5      char S4_Lectura(int a,int b,bool derecha,bool izquierda
6      ,bool abajo,bool arriba,terreno plataforma[][10]){
7      char sensor;
8      if(derecha == true){
9          if(plataforma[a][b+1].estacion.identificacion[0] == 'b'){
10             sensor = 'b';
11         }else{
12             sensor = 'n';
13         }
14     }
15     if(izquierda == true){
16         if(plataforma[a][b-1].estacion.identificacion[0] == 'b'){
17             sensor = 'b';
18         }else{
19             sensor = 'n';
20         }
21     }
```

Figura 24. Representación función Char S4_lectura
Cuando el movimiento es hacia derecha o izquierda.

Como se observa en la figura anterior, el sensor 4 o S4_Lectura, hace una búsqueda en la posición siguiente de la plataforma. Identificación para de ésta manera saber si donde está parado el robot es una posible estación o no. En la siguiente figura se puede observar lo que haría la función si el movimiento es hacia arriba o hacia abajo.

```
15     if(izquierda == true){
16         if(plataforma[a][b-1].estacion.identificacion[0] == 'b'){
17             sensor = 'b';
18         }else{
19             sensor = 'n';
20         }
21     }
22     if(abajo == true){
23         if(plataforma[a+1][b].estacion.identificacion[0] == 'b'){
24             sensor = 'b';
25         }else{
26             sensor = 'n';
27         }
28     }
29     if(arriba == true){
30         if(plataforma[a-1][b].estacion.identificacion[0] == 'b'){
31             sensor = 'b';
32         }else{
33             sensor = 'n';
34         }
35     }
36     return sensor;
```

Figura 25. Representación función Char S4_lectura

Cuando el movimiento es hacia arriba o abajo.

3.2.5. Char S5_Lectura:

Éste sensor en conjunto con el sensor anterior, como ya dijimos hace la búsqueda de si en la siguiente posición hay o no una letra 'n' o 'b', dependiendo del carácter que el sensor s4 haya encontrado. En la figura 26 se puede observar cómo hace esto la función.

```
38     char S5_Lectura(int a,int b,bool derecha,bool izquierda
39         ,bool abajo,bool arriba,terreno plataforma[][10]){
40         char sensor;
41
42         if(derecha == true){
43             if(plataforma[a][b+1].estacion.identificacion[1] == 'b'){
44                 sensor = 'b';
45             }else{
46                 sensor = 'n';
47             }
48         }
```

Figura 26. Representación función Char S5_lectura

De la misma manera en la que trabaja el sensor s4 el sensor s5 hace la búsqueda y dependiendo del resultado se sabe en qué estación hay que detenerse.


```

49     if(izquierda == true){
50         if(plataforma[a][b-1].estacion.identificacion[1] == 'b'){
51             sensor = 'b';
52         }else{
53             sensor = 'n';
54         }
55     }
56     if(abajo == true){
57         if(plataforma[a+1][b].estacion.identificacion[1] == 'b'){
58             sensor = 'b';
59         }else{
60             sensor = 'n';
61         }
62     }
63     if(arriba == true){
64         if(plataforma[a-1][b].estacion.identificacion[1] == 'b'){
65             sensor = 'b';
66         }else{
67             sensor = 'n';
68         }
69     }
70     return sensor;
71 }

```

Figura 27. Representación función Char S5_ lectura.

3.2.6. Char S6_Lectura:

Este sensor de una manera parecida a los sensores anteriores hace una búsqueda de color para determinar si se tiene o no que arrojar la caja. Este sensor en conjunto con los sensores S4, S5 y S6 hacen una búsqueda de colores, pero de una manera distinta, se busca en la banda transportadora hasta encontrar números, para este sensor si en la banda transportadora se encuentra un 1 o un 3, la función retorna 'n', de lo contrario retornará 'b'. En la figura 28 se puede observar cómo el robot hace esto.

```

72 char S6_Lectura(int banda_transportadora[], int n){
73     char sensor;
74     if(banda_transportadora[n] == 1){
75         sensor='n';
76     }
77     if(banda_transportadora[n] == 2){
78         sensor = 'b';
79     }
80     if(banda_transportadora[n] == 3 ){
81         sensor = 'n';
82     }
83     if(banda_transportadora[n] == 4){
84         sensor = 'b';
85     }
86     return sensor;

```

Figura 28. Representación función Char S6_ lectura.

En la imagen anterior se puede analizar cómo realiza esto el sensor, efectivamente como se puede notar si el sensor encuentra un 1 o un 3 asigna un 'n' de lo contrario un 'b'.

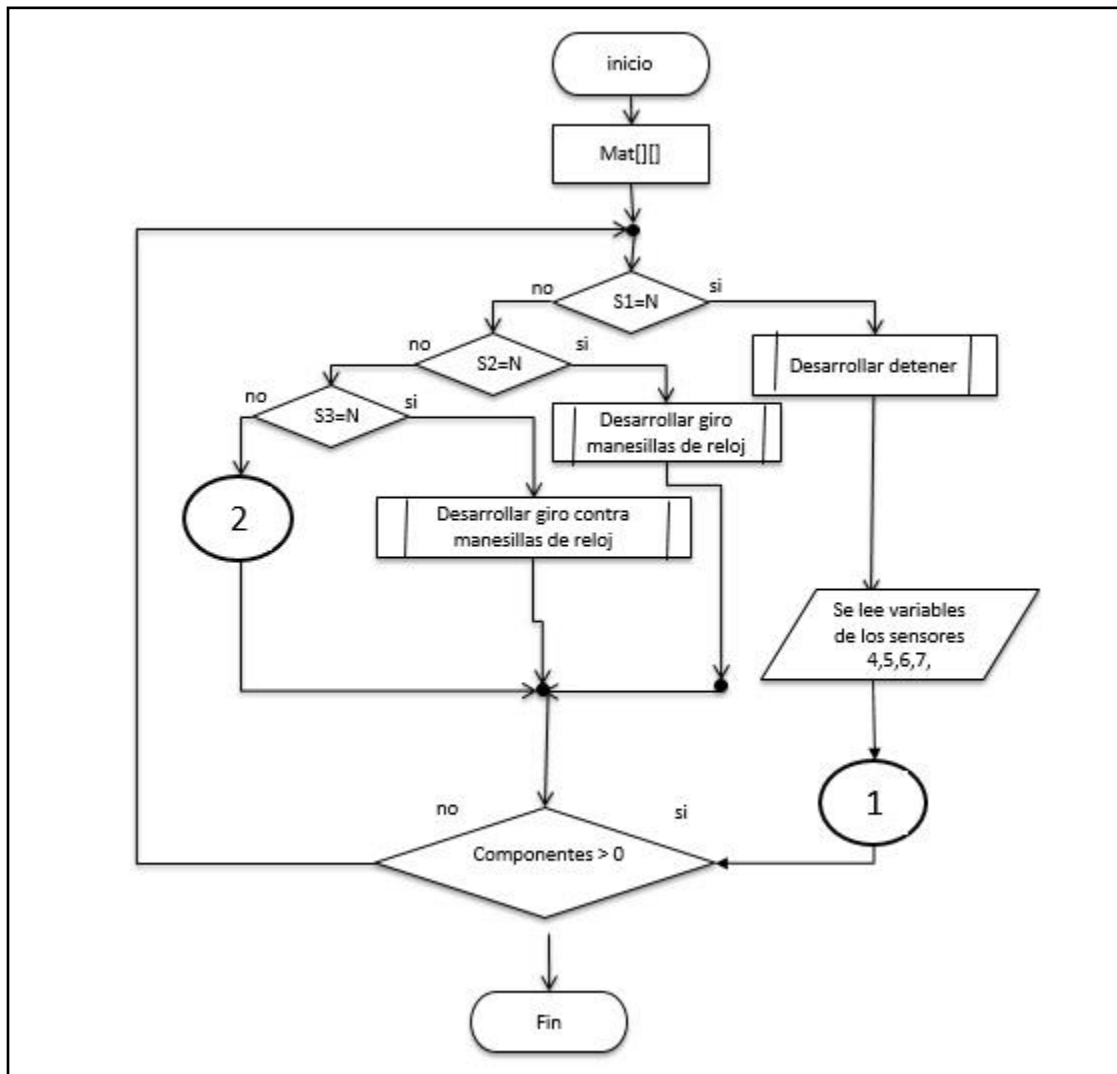
3.2.7. Char S7_Lectura:

Este sensor hace prácticamente la misma comparación que el sensor s6, sólo que en el momento de buscar el valor es distinto. Si encuentra un 1 o un 4 retornará 'n', de lo contrario retornará 's'. En la figura 29 se puede notar como en código de hace esto.

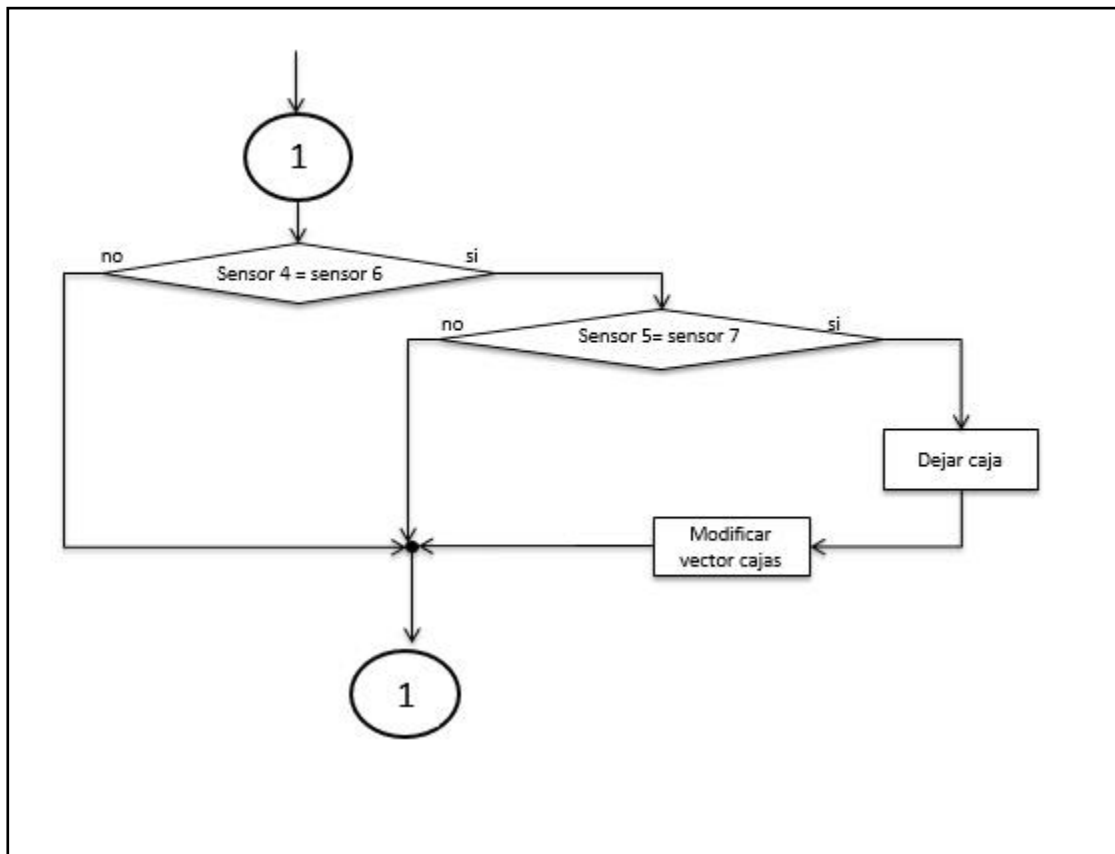
```
88  char S7_Lectura(int banda_transportadora[], int n){
89      char sensor;
90
91      if(banda_transportadora[n] == 1){
92          sensor = 'b';
93      }
94      if(banda_transportadora[n] == 2){
95          sensor = 'n';
96      }
97      if(banda_transportadora[n] == 3){
98          sensor = 'n';
99      }
100     if(banda_transportadora[n] == 4){
101         sensor = 'b';
102     }
103     return sensor;
104 }
```

Figura 29. Representación función Char S7_lectura.

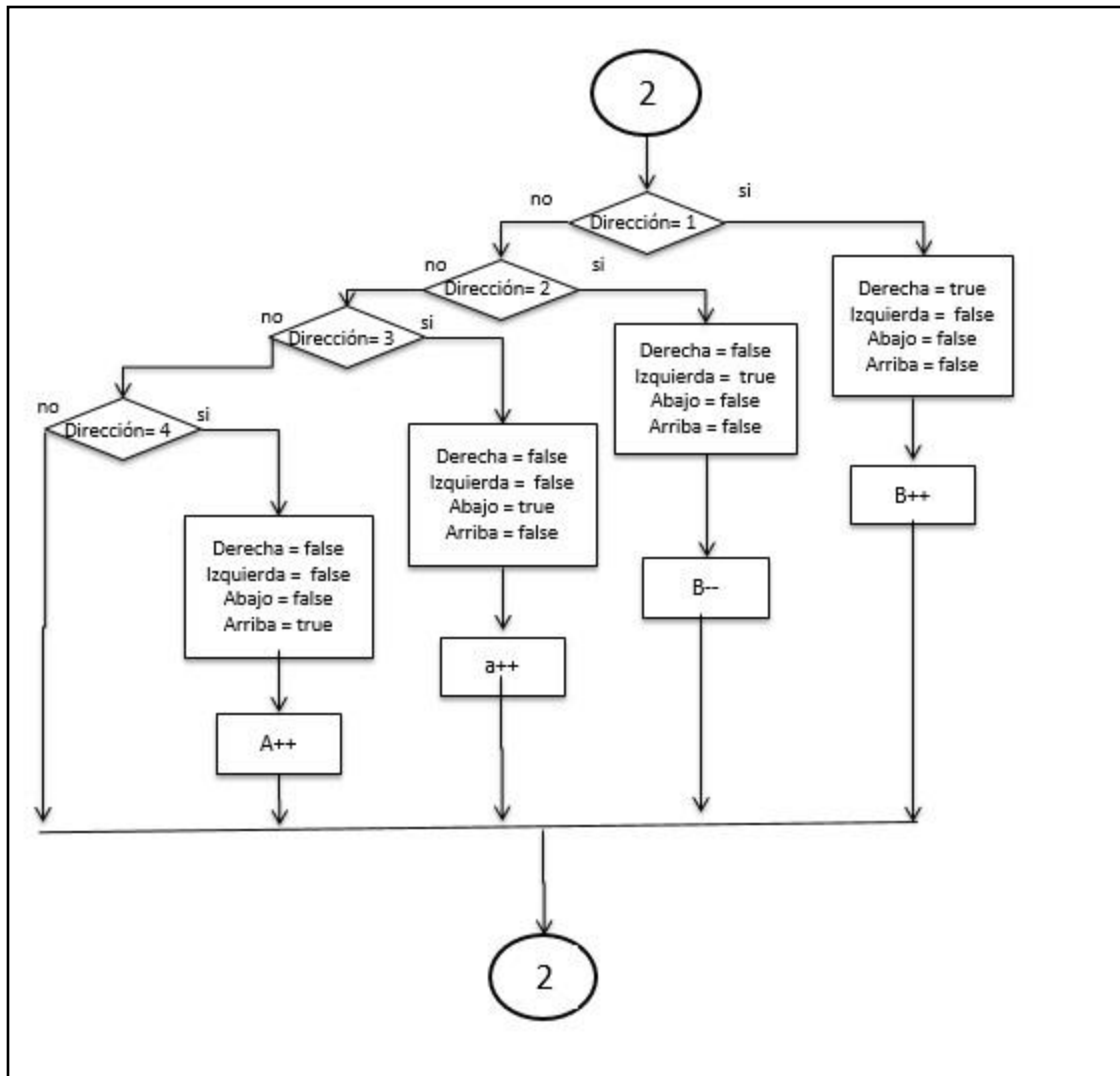
4. Loop Fusion.



Software para Procesadores @ "Electrónica Javeriana" – Bogotá – Colombia		
D. Flujo: diagrama general proyecto.		Supervisor: Ing. Miguel Sarmiento M.Sc.
Dedicación: 10 min		Diseñado por: Proyecto_1_3.
Plancha: 1 de 3	Versión:	Fecha: 31/07/2016



Software para Procesadores @ "Electrónica Javeriana" – Bogotá – Colombia		
Diagrama Flujo: función dejar caja		Supervisor: Ing. Miguel Sarmiento M.Sc.
Dedicación: 20 min		Diseñado por: Proyecto_1_3-
Plancha: 2 de 3	Versión:	Fecha: 31/07/2016

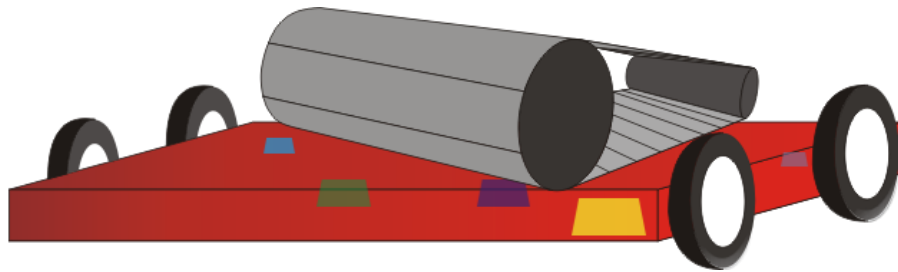


Software para Procesadores @ "Electrónica Javeriana" – Bogotá – Colombia		
Diagrama Flujo: función dejar caja		Supervisor: Ing. Miguel Sarmiento M.Sc.
Dedicación: 20 min		Diseñado por: Proyecto_1_3-
Plancha: 3 de 3	Versión:	Fecha: 31/07/2016

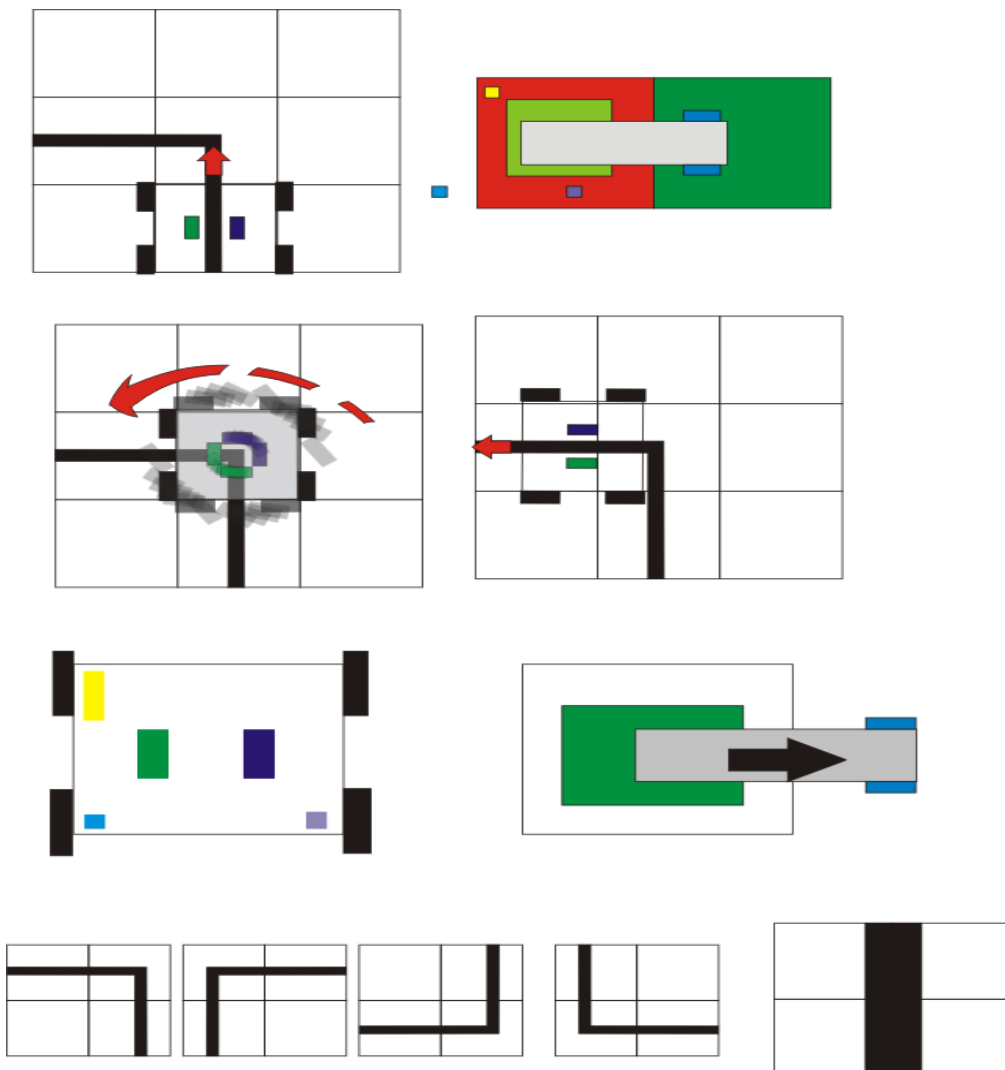
5. Roles finales de cada uno de los integrantes.

- Sergio Andres Rojas Moreno: Desempeñó el rol de *liderazgo*.
- Javier Enrique Huérfano Díaz: Desempeñó el rol de *Comunicación Oral*.
- Camilo Andrés Trujillo Muñoz: Desempeñó el rol de *Comunicación Escrita*.
- Juan David Rodríguez Álvarez: Desempeñó el rol de *Comunicación Gráfica*.

6. Gráficos de las diferentes situaciones a las que se enfrenta el robot.



Gráfica 1. Modelo en 3D del robot con sus respectivos sensores.



Gráfica 2. Vistas superiores de cada uno de los movimientos del robot, sus sensores y banda transportadora, y lozas del escenario de funcionamiento.

CONCLUSIONES

Como resultado del desarrollo de este proyecto, se logró solucionar un problema con la correcta implementación de un software que permitiría controlar a un robot transportador. Se plantearon muchas ideas sobre una posible solución de este problema con el cual encontramos una solución más eficiente que todas las ideas propuestas con anterioridad. Esta solución incluía la implementación del método de "*Loop Fusion*", con el cual se puede lograr un resultado más limpio y óptimo para un sistema de c/c++.

Por otro lado, nos encontramos con varios factores que nos impidieron concluir el proyecto al pie de la letra, y esto, en cuestiones de pruebas que nos constatarán la eficacia del programa, por lo tanto, en este trabajo se aprecia la falta de pruebas de escritorio de cada diagrama de flujo. Se sabe del papel de las pruebas de escritorio, pero con cuestiones alternas como el tiempo, se imposibilita cumplir con todos los numerales del enunciado.

Se ampliaron las ideas acerca de la aplicación de un programa en c/c++ para la solución y el control de sistemas físicos como los que un robot necesita. Finalmente, hubo una retroalimentación de los temas vistos en el área de *Pensamiento Algorítmico* para poder realizar este proyecto y pudimos mejorar nuestra capacidad de análisis algorítmico.

Referencias

Rodríguez, J, (2016). *Graficas del informe técnico de Proyecto_1 grupo 3.*

Trujillo, C, (2016). *Trabajo escrito del informe técnico de Proyecto_1 grupo 3.*

Huérfino, J, (2016). *Código en c/c++ del informe técnico de Proyecto_1 grupo 3.*

Rojas, S, (2016). *Objetivos y conclusiones del informe técnico de Proyecto_1 grupo 3.*