**UNIVERSIDAD DE TARAPACÁ**



**ESCUELA UNIVERSITARIA DE INGENIERÍA INDUSTRIAL, INFORMÁTICA Y DE SISTEMAS**



Área de Ingeniería en Computación e Informática



**Proyecto MindCuber**

**Autor(es): Claudio Mena, Simón Muñoz,**

**Jackelyn Rojas, Javier Sánchez,**

**Felipe Valenzuela**

**Asignatura: Proyectos 1**

**Profesor(es): Ricardo Valdivia, Diego Aracena**

ARICA, 05-09-2017

# Historial de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor(es)** |
| 05/09/2017 | 1.0 | Versión preliminar del formato | Jackelyn Rojas,  Felipe Valenzuela,  Claudio Mena |
| 10/09/2017 | 1.1 | Avance preliminar (1er informe) | Jackelyn Rojas,  Felipe Valenzuela,  Claudio Mena |
| 14/09/2017 | 1.2 | Avance preliminar y correcciones | Jackelyn Rojas,  Javier Sánchez |
| 22/10/2017 | 1.3 | Avance preliminar (2do informe) | Claudio Mena  Javier Sánchez  Jackelyn Rojas |
| 16/11/2017 | 1.4 | Avance preliminar(3er informe) | Claudio Mena  Simón Muñoz  Juan Rojas  Jackelyn Rojas  Javier Sánchez  Felipe Valenzuela |

# Tabla de contenidos

1. Panorama General
   1. Resumen del Proyecto

* **Propósito:** Modificar el código fuente y la estructura del robot MindCuber para que interactúe con un cubo rubik y realice patrones predefinidos, ya desde un estado inicial armado, a otros 3 con distintos patrones predefinidos.
* **Alcance**: El robot contara con la característica de llevar el cubo rubik de un estado a otro, además de poder realizar las mismas tareas de forma inalámbrica.
* **Objetivos**:

Objetivo General

* + - Desarrollar un robot capaz de realizar una actividad que consta de armar 3 patrones específicos para un cubo rubik.

Específicos

* + - Construir el robot que sea capaz de manipular un cubo rubik de 3x3x3.
    - Desarrollar la codificación que a partir de un menú, sea capaz de armas 3 patrones específicos desde un cubo armado.
    - Manipular inalámbricamente el robot para que sea capaz de comunicarse vía Bluetooth con el computador.
* **Suposiciones y restricciones:** 
  + El robot, no podrá después de llevar al cubo a un diseño previamente elegido, regresar la acción y dejarlo en su estado inicial.
  + Si el cubo llega desarmado, no habrá forma de saberlo.
* **Entregables del Proyecto:**
  + “PLAN DE PROYECTO 1”
  + **“**2do Informe **“**
  + “Entregable Final **“**
  1. Historial de versiones

1. Referencias
   1. Módulo 3: Programación NXC- Robots lego NXT Mindstorms.
   2. Módulo 3: Programación NXC- Sensores.
2. Organización del Proyecto

3.1. Personal y Entidades internas

* Analista, Programador, Diseñador gráfico, Jefe de proyecto, Constructores.

3.2. Roles y responsabilidades

* **Jefe de Proyecto:** Verifica que todas las etapas del proyecto se estén cumpliendo.
* **Programador:** Codifica las especificaciones detalladas según las complejidades dadas.
* **Analista:** Proyecta las etapas del proyecto, además de ingeniárselas para resolver problemas en el camino.
* **Constructores:** llevan a cabo el armado del robot con las especificaciones dadas y las modificaciones necesarias para llevar a cabo el proyecto.
* **Diseñados Grafico:** lleva las publicaciones en la wiki, además de documentar semana a semana el avance del trabajo.

3.3. Mecanismos de Comunicación

* Cuentas en redes sociales: Grupo WhatsApp del proyecto.

1. Planificación de los procesos de gestión

4.1. Planificación inicial del proyecto

* Planificación de estimaciones

- Costo del software de desarrollo:

Software libre NXC: $0

- Costo material:

Kit robot Lego mind storms: $380.000

Kit piezas adicionales: $90.000

Cubo rubik: $15.000 x2

Total: $500.000  
- Costo total horas de trabajo:

1 Hora de trabajo: $7000

Total de horas: 60 horas

Costo por persona: $520.000

Personas involucradas en el proyecto: 6

Total: $3.120.000

- Costo total del proyecto: $3.120.000 + IVA

* Planificación de Recursos Humanos

Analistas: 2, Diseñador: 2, Programador: 1, Jefe de Proyecto: 1.

4.2. Lista de actividades

* Actividades de trabajo

Planificación del proyecto: Establecer los Patrones que utilizaremos para el cubo rubik.

Modificación del código para lograr lo establecido del proyecto, etc.

Ejecución del Proyecto: Análisis, diseño, codificación, etc.

Cierre del Proyecto: Producto final y funcionalidad.

* Asignación de tiempo

Planificación del proyecto: 2-3 semanas.  
Ejecución del proyecto: 4 a 5 semanas.  
Cierre de proyecto: 15 semanas.

4.3. Planificación de la gestión de riesgos

|  |  |  |  |
| --- | --- | --- | --- |
| RIESGOS | PROBABILIDAD  DE  OCURRENCIA | NIVEL  DE IMPACTO | ACCION REMEDIAL |
| La estimación de la duración del proyecto esta errónea. | 40% | 2 | Acelerar tareas básicas dentro del proyecto y reacomodar actividades futuras a un periodo de tiempo más corto. |
| Falta de preparación del equipo de trabajo | 25% | 3 | Preparar de mejor manera la siguiente sesión de trabajo. |
| Algún miembro del equipo ausente | 15% | 3 | Se reacomodara el trabajo del miembro ausente de esa sesión a los miembros del equipo presentes. |
| Incapacidad para editar el código del MindCuber | 15% | 1 | Se trabajaran horas extraordinarias para investigación y análisis más extensivos sobre los patrones y lógica en la cual se basa el código. |
| Piezas rotas | 15% | 2 | Se enviara al encargado de construcción a buscar la o las piezas de repuesto. |
| Escases de piezas | 10% | 2 | Rediseñar el robot para que funcione sin las piezas faltantes. |
| Destrucción accidental del MindCuber | 5% | 1 | Trabajar horas extraordinarias para rehacer el MindCuber desde cero, con todos los miembros del equipo involucrados. |

1. Planificación de los procesos técnicos
   1. Modelo de proceso



* 1. Herramientas y técnicas
* Software NXC.
* Adobe Illustrator
* Microsoft Word
* Redmine

1. Análisis
   1. Modelo de Diseño (Seudocódigo General)

*Definición de funciones:*

**Función** girar\_derecha ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(290));

RotarMotor (Motor\_A, Velocidad(100), grados (-12));

Apagar(Motor\_A);

**Fin-Función**

**Función** girar\_derecha\_tres ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(290));

RotarMotor (Motor\_A, Velocidad(100), grados (-12));

Apagar(Motor\_A);

**Fin-Función**

**Función** girar\_izquierda ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(-290));

RotarMotor (Motor\_A, Velocidad(100), grados (12));

Apagar(Motor\_A);

**Fin-Función**

**Función** girar\_izquierda\_tres ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(-290));

RotarMotor (Motor\_A, Velocidad(100), grados (12));

Apagar(Motor\_A);

**Fin-Función**

**Función** brazo ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_C, Velocidad(70), grados(-85));

RotarMotor (Motor\_C, Velocidad(70), grados (85));

**Fin-Función**

**Función** brazo\_arriba ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_C, Velocidad(40), grados(50));

Apagar(Motor\_C);

**Fin-Función**

**Función** brazo\_abajo ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_C, Velocidad(40), grados(-50));

Apagar(Motor\_C);

**Fin-Función**

**Función** giro\_completo ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(720));

Apagar(Motor\_A);

**Fin-Función**

**Función** girar\_izquierda\_doble ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(-290));

RotarMotor(Motor\_A, Velocidad(100), grados(12));

RotarMotor(Motor\_A, Velocidad(100), grados(-290));

RotarMotor(Motor\_A, Velocidad(100), grados(12));

Apagar(Motor\_A);

**Fin-Función**

**Función** girar\_derecha\_doble ():Void

**Inicio**

    Instrucciones:

RotarMotor(Motor\_A, Velocidad(100), grados(290));

RotarMotor(Motor\_A, Velocidad(100), grados(-12));

RotarMotor(Motor\_A, Velocidad(100), grados(290));

RotarMotor(Motor\_A, Velocidad(100), grados(-12));

Apagar(Motor\_A);

**Fin-Función**

*Definición de patrones: son funciones que llaman a funciones ya declaradas arriba.*

**patron\_uno(){**

repetir (2){

girar\_izquierda();

brazo();

brazo();

girar\_derecha();

brazo();

brazo\_arriba();

girar\_izquierda();

brazo\_abajo();

}

girar\_derecha();

brazo();

brazo();

girar\_izquierda();

brazo();

brazo\_arriba();

girar\_izquierda();

brazo\_abajo();

girar\_izquierda();

brazo();

brazo();

girar\_derecha();

brazo\_arriba();

giro\_completo();

}

**patron\_dos()**

{

repetir(2){

girar\_izquierda\_doble();

brazo();

brazo();

girar\_derecha\_doble();

brazo();

brazo\_arriba();

girar\_izquierda();

brazo\_abajo();

}

girar\_derecha\_doble();

brazo();

brazo();

girar\_izquierda\_doble();

brazo();

brazo\_arriba();

giro\_completo();

}

**patron\_tres()**

{

girar\_derecha\_tres(); // F

brazo(); // U

girar\_derecha\_tres();

brazo();

brazo();

brazo();

girar\_derecha\_tres(); // F

brazo();

brazo\_arriba();

girar\_izquierda\_tres();

brazo\_abajo();

brazo();

girar\_derecha\_tres(); // R

brazo();

brazo();

girar\_izquierda\_doble(); //2L

brazo();

brazo\_arriba();

girar\_derecha\_tres();

brazo\_abajo();

brazo();

girar\_derecha\_tres(); // B

brazo();

girar\_izquierda\_tres(); // D´

brazo();

brazo\_arriba();

girar\_izquierda\_tres();

brazo\_abajo();

brazo();

girar\_derecha\_tres(); // R

brazo();

brazo\_arriba();

girar\_izquierda\_tres();

brazo\_abajo();

brazo();

brazo();

girar\_izquierda\_doble(); // 2D

brazo();

brazo\_arriba();

girar\_derecha\_tres();

brazo\_abajo();

brazo();

girar\_derecha\_tres(); // L

brazo();

brazo\_arriba();

girar\_derecha\_tres();

brazo\_abajo();

brazo();

girar\_izquierda\_tres(); // D´

brazo();

brazo();

brazo();

girar\_derecha\_tres(); // B

brazo();

brazo\_arriba();

girar\_izquierda\_tres();

brazo\_abajo();

brazo();

girar\_derecha\_doble(); // 2R

brazo();

brazo();

girar\_derecha\_tres(); // L

brazo();

brazo\_arriba();

girar\_derecha\_tres();

brazo\_abajo();

brazo();

girar\_derecha\_tres(); // F

brazo();

girar\_derecha\_tres(); // U

brazo();

brazo();

brazo();

girar\_derecha\_tres(); // F

brazo\_arriba();

giro\_completo();

}

*Definición del main:*

**Función** main ():Void

**Inicio**

    Instrucciones:

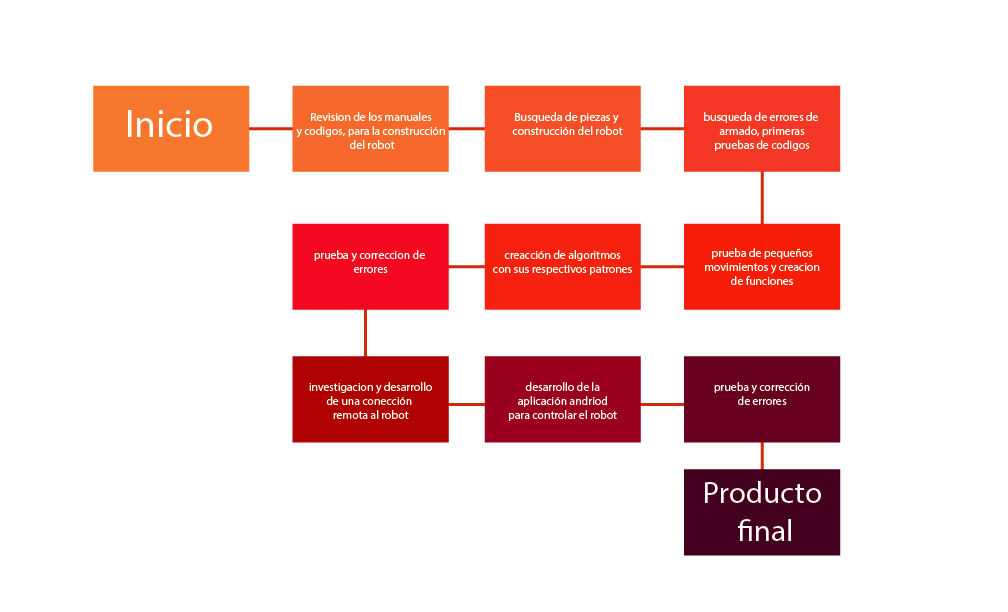
Patrón\_uno;

Patrón\_dos;

Patrón\_tres;

**Fin-Función**

* 1. Descripción de la Arquitectura (vista del modelo diseño)



* 1. Documento de Diseño de Interface Usuario

Se hizo evidente la necesidad de crear dos interfaces de usuario para la comunicación de tal con el robot y dicho algoritmo.

1.- Comunicación directa: el usuario deberá a través del computador conectado al dispositivo, enviarle las instrucciones al robot, con el algoritmo seleccionado.

2.- Comunicación Inalámbrica: el usuario podrá ejecutar los patrones desde una aplicación Android, con el fin de poder comunicarse con el dispositivo de forma inalámbrica.

* 1. Especificación de Requerimientos

Principalmente: conocimientos en app inventor y funcionamiento del NTX de LEGO Mindstorms.

Para la implementación principal de armado y configuración del robot, nuestros contribuyentes necesitaron:

1. Robot NXT LEGO Mindstorms (facilitado por la institución).
2. Piezas adicionales de armado como rieles de más de 9 puntos de largo (facilitado por la institución).
3. Computador e internet necesarios para el armado del robot y codificación de este mismo.

Luego de haber terminado el armado del robot de haber implementado mejoras en el sistema serán necesarios los siguientes materiales para la etapa final del proyecto:

1. Robot NXT ya armado y mejorado
2. Dispositivo Android (Para ejecutar la APP desarrollada en AppInventor 2)
3. Computador con Flash y Java previamente cargados
4. Diseño
   1. Modelo de Diseño (Seudocódigo detalles)

**Función** girar\_derecha: Gira una cara del cubo rubik en 90 grados.

**Función** girar\_derecha\_tres: Gira una cara del cubo rubik en 90 grados (corrección para el patrón 3).

**Función** girar\_izquierda: Gira una cara del cubo rubik -90 grados.

**Función** girar\_izquierda\_tres: Gira una cara del cubo rubik en -90 grados (corrección para el patrón 3).

**Función** brazo: Acciona el brazo para hacer girar el cubo rubik en 90 grados.

**Función** brazo\_arriba: Acciona el brazo para soltar el cubo rubik.

**Función** brazo\_abajo: Acciona el brazo para atrapar el cubo rubik.

**Función** giro\_completo: Gira la base en 360 grados.

**Función** girar\_izquierda\_doble: Gira la base -180 grados.

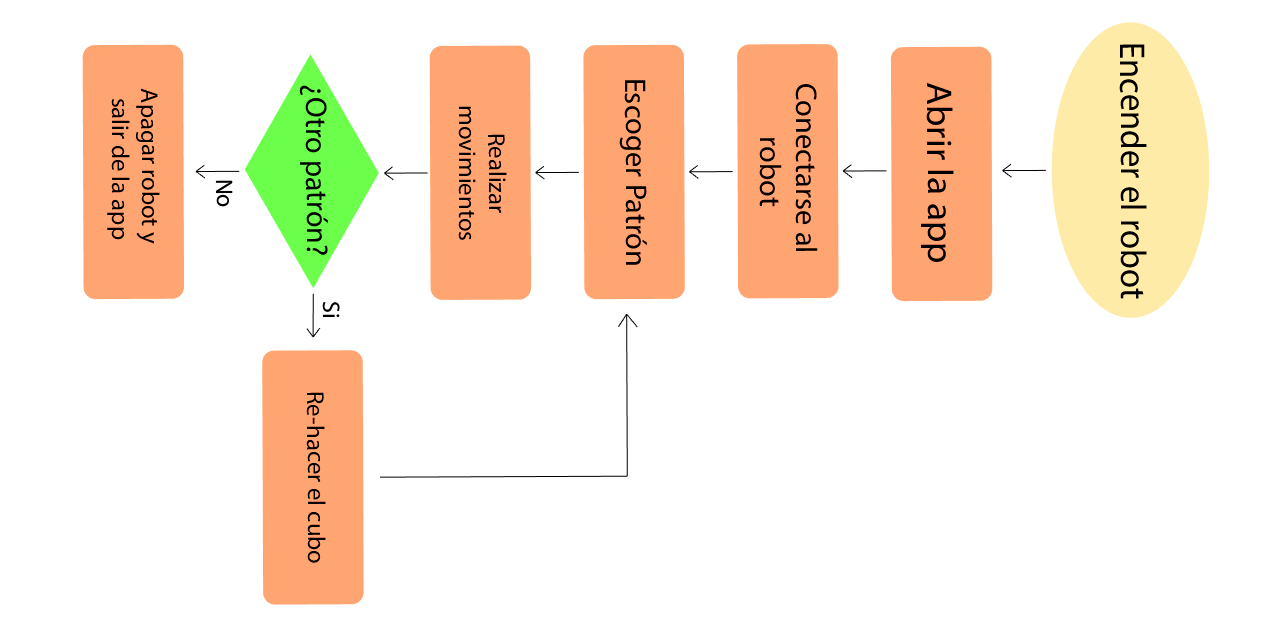
**Función** girar\_derecha\_doble: Gira la base 180 grados.

**Función** patron\_uno: Ejecuta el patrón intercambios de centros (Dot).

**Función** patron\_dos: Ejecuta el patrón tablero de ajedrez (Checkerboard).

**Función** patron\_tres: Ejecuta el patrón líneas verticales.

* 1. Modelo de Interacción (Diagrama de flujo, o flujo de las acciones)



* 1. Descripción de la Arquitectura con respecto a los modelos.

1° Abrir la app: En un dispositivo Android, se abrirá la aplicación para la Selección de los patrones.

2° Conectarse al robot: Mediante la aplicación y la interfaz Bluetooth, se conectara el robot al dispositivo Android, para la selección de patrones.

3° Escoger Patrón: Se seleccionara uno de los 3 patrones para que posteriormente el robot ejecute los movimientos.

4° Realizar movimientos: El robot hará el patrón dependiendo de los movimientos programados por los algoritmos.

5° Re-hacer el cubo: El cubo rubik volverá a su estado inicial para posteriormente escoger un nuevo patrón a formar.

6° Apagar robot y salir de la app: El robot se apagara y él se terminara el proceso de la app en el dispositivo Android.

1. Implementación
   1. Plan de Integración

**Contenido:**

-Para todo aquel que desee diseñar e implementar un proyecto en base al nuestro, aquí se detallan suministros, módulos, plazos y competencias necesarias para la correcta implementación del mismo.

**Suministros necesarios:**

* Kit LEGO Mindstorms
* Cubos de rubik
* Guía de armado de MindCuber

**Módulos y Plazos aproximados:**

* Construir el Kit de acuerdo a la guía de MindCuber: 1-2 semanas
* Buscar patones para el cubo de rubik: 1 semana
* Analizar los patrones y transformar cada patrón a seudocódigo: 2-3 semanas
* Fase de programación de patrones: 2-3 semanas
* Pruebas en el robot: 1-2 semanas
* Documentación: 1-2 semanas

**Competencias necesarias:**

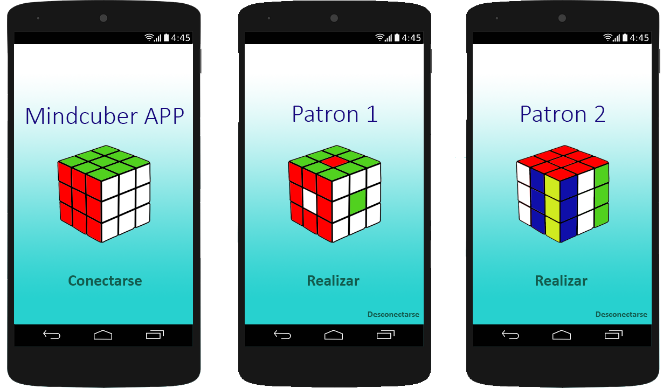
* Iniciativa
* Conocimientos de programación en C
* Percepción visual

**-**Si el equipo de trabajo (o integrante nuevo) están de acuerdo al plan, puede llevar a cabo el proyecto en los plazos establecidos e incluso en menos tiempo, reduciendo costo de horas de trabajo y dejando más tiempo para las pruebas en el robot, mejoras de funcionamiento y una documentación mejor y más detallada.

* 1. Descripción de la Arquitectura (vista desde los módulos en seudocódigo)

|  |  |
| --- | --- |
| Función | Descripción |
| **void** brazo()  {RotateMotor(OUT\_C, 70, -85);  RotateMotor(OUT\_C, 70, 85);  } | -El brazo que sostiene al cubo hace un  movimiento hacia atrás y luego  Hacia adelante para voltear el cubo. |
| **void** brazo\_arriba()  {RotateMotor(OUT\_C, 40, 50);  Off(OUT\_C);  } | -El brazo suelta el cubo sin voltearlo para asi poder girar el cubo completo. |
| **void** brazo\_abajo()  {RotateMotor(OUT\_C, 40,-50);  Off(OUT\_C);  } | -El brazo agarra al cubo para sostenerlo y asi mover solamente la cara de abajo. |
| **void** girar\_derecha()  {RotateMotor(OUT\_A, 100, 290);  RotateMotor(OUT\_A,100, -12);  Off(OUT\_A);  } | -Gira la base 90 grados (aprox) hacia la derecha. |
| **void** girar\_izquierda()  {RotateMotor(OUT\_A, 100, -290);  RotateMotor(OUT\_A, 100, 12);  Off(OUT\_A);  } | -Gira la base 90 grados (aprox) hacia la izquierda. |

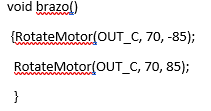
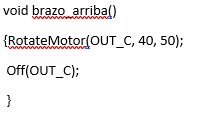
* 1. Modelo de Implementación

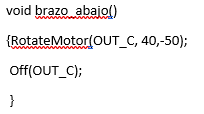


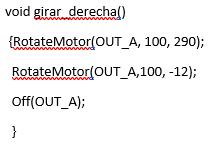
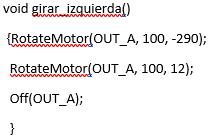
Modelo de App con conexión Inalámbrica

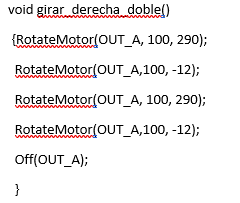
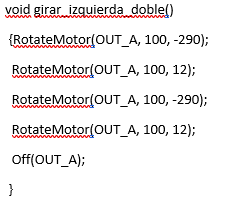
* 1. Módulos Implementados

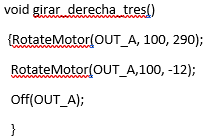
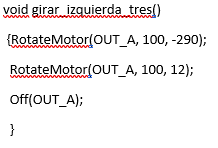
Movimientos del brazo:

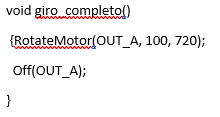




 Giros de la base:







Plan de Integración Y Modulo de Implementación:

El usuario elige cualquiera uno de los 3 patrones del cubo de rubik que se encuentran dentro de la aplicación, luego se le enviara el patrón elegido al NXT mediante conexión inalámbrica (Bluetooth).



El” Robotzuelo “recibirá la orden del patrón elegido y lo ejecutara, mediante la aplicación Android y el NXT conectados vía bluetooth entre sí.

8.5 Reporte de Revisión

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Descripción** | **Autor(es)** |
| 26/09/2017 | Robot armado | Javier Sánchez – Simón Muñoz |
| 28/09/2017 | Robot modificado | Javier Sánchez – Simón Muñoz |
| 3/10/2017 | Movimientos del robot implementados | Claudio Mena – Javier Sánchez |
| 5/10/2017 | Patrón 1 realizado correctamente | Claudio Mena – Javier Sánchez |
| 10/10/2017 | Patrón 2 implementado. Corrección de los movimientos del robot. Patrón 2 realizado correctamente | Claudio Mena – Javier Sánchez |
| 10/10/2017 | Robot modificado según nuestros requerimientos | Simón Muñoz |
| 10/10/2017 | Patrón 3 implementado y realizado correctamente | Claudio Mena – Javier Sánchez |
| 12/10/2017 | Problemas con la base del robot y pronta modificación | Simón Muñoz |
| 12/10/2017 | Todos los patrones se realizan correctamente | Claudio Mena – Javier Sánchez |
| 12/10/2017 | Primeros avances conexión inalámbrica | Simón Muñoz – Felipe Valenzuela |

|  |  |  |
| --- | --- | --- |
| 02/11/2017 | Corrección de detalles del funcionamiento del robot | Javier Sánchez – Claudio Mena |
| 02/11/2017 | Creación de la Aplicación en Appinventor 2 | Felipe Valenzuela |

|  |  |  |
| --- | --- | --- |
| 07/11/2017 | Primera prueba de Conexión inalámbrica entre la aplicación y el NXT | Felipe Valenzuela |

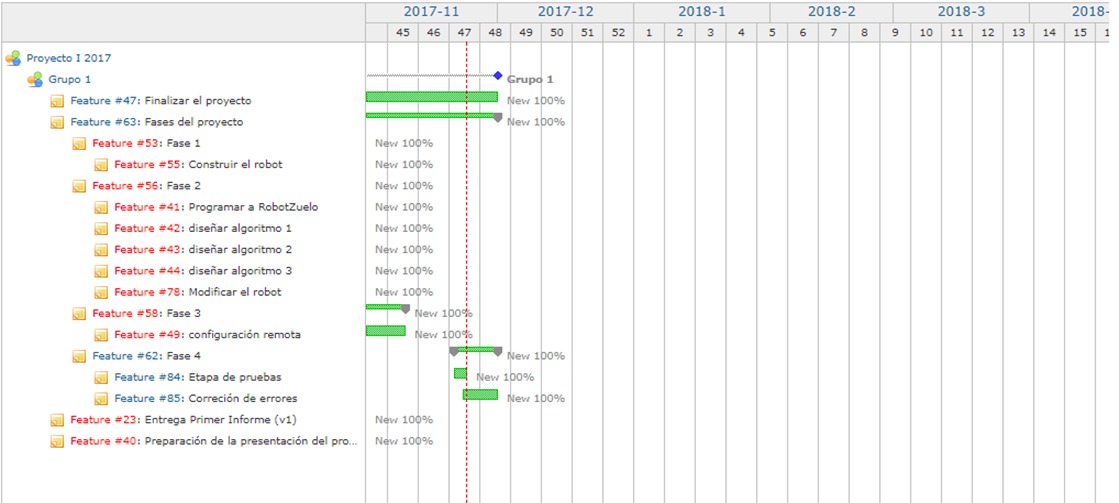
|  |  |  |
| --- | --- | --- |
| 09/11/2017 | Termino de la aplicación en App inventor2 | Felipe Valenzuela |

|  |  |  |
| --- | --- | --- |
| 14/11/2017 | Avance del entregable final del proyecto. | Jackelyn Rojas |

|  |  |  |
| --- | --- | --- |
| 14/11/2017 | Termino de la presentación y entregable final. | Claudio Mena  Simón Muñoz  Jackelyn Rojas  Juan Rojas  Javier Sánchez  Felipe Valenzuela |

1. Aspectos Generales

9.1 Avance de acuerdo a la carta Gantt



Como se puede observar ya está realizado el 100% del trabajo.

9.2Problemas encontrados

9.3Soluciones Propuestas

|  |  |
| --- | --- |
| **PROBLEMAS**  **ENCONTRADOS** | **SOLUCIONES PROPUESTAS** |
| -Constante error en el giro del cubo. | -Constante error en el giro del cubo (cara mal encajada respecto al cubo): Este problema se solucionó con un estudio del movimiento de la cara en el armado, para solucionar esto se modificó los grados de giro después de cada movimiento de la cara del cubo. |
| -Constante obstrucción en el brazo de movimiento superior del cubo. | -Constante obstrucción en el brazo de movimiento superficial del cubo: Este problema se solucionó modificando la estructura del brazo para movimiento superficial del cubo. |
| -Base del robot demasiado débil. | -Base del robot demasiado débil: Este problema se solucionó modificando la estructura inferior de la base para así tener una mejor estabilidad. |

9.4Conclusiones

Luego de haber terminado el proyecto dado, al revisar hacia atrás todo lo trabajado, nos dimos cuenta que el proyecto jamás fue como lo pensábamos, ya que, nos imaginábamos un trabajo más de edición que de creación, pero este ha sido un trabajo más que de creación y edición, una escuela del trabajo en equipo, ya que, al principio nos costó mucho sincronizarnos. Aprendimos al igual la metodología de trabajo por la Gantt y también el registro de avances.

A futuro gracias a este curso podremos tener una idea de cómo hacer funcionar un proyecto estructurado y con fecha límite de entrega.

10 Trabajo Futuro

Con la aplicacion terminada y el proyecto total, ahora el trabajo será el perfeccionar, elimimar errores y bugs de la aplicacion, luego al terminar de detallar la aplicación se procederá al detalle del manual mejorando su metodologia para el rapido entendimiento y facil acceso al uso del sistema completo.