**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 MenaJavier SánchezJackelyn Rojas |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 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 remota.
* **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 remotamente el robot para que sea capaz de comunicarse vía remota 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”
	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 mindstorms: $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: $420.000

Personas involucradas en el proyecto: 5

Total: $2.100.000

 - Costo total del proyecto: $2.600.000 + IVA

* Planificación de Recursos Humanos

Analistas: 1, 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 | NIVELDE 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)
	2. 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 remota: 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
1. 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
	2. Descripción de la Arquitectura (vista desde los módulos en seudocódigo)
	3. Modelo de Implementación

 Modelo de app con conexión remota

* 1. Módulos Implementados

 Movimientos del brazo:





 Giros de la base:







|  |  |  |
| --- | --- | --- |
|  **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 remota | Simón Muñoz – Felipe Valenzuela |

* 1. Reporte de Revisión
1. Aspectos Generales
	1. Avance de acuerdo a la Carta Gantt

 Como se puede observar ya está realizado el 60% del trabajo.

* 1. Problemas Encontrados

 En el grupo se encontraron 3 problemas los cuales fueron solucionados, no en una sola clase si no que en varias los problemas fueron los siguientes:

* Constante error en el giro del cubo
* Constante obstrucción en el brazo de movimiento superficial del cubo
* Cubo se sale de la plataforma
	1. Soluciones Propuestas
* Constante error en el giro del cubo ( des sincronización de la posición del cuadrado para el ensamble posterior del giro en el extremo del cubo), este problema se solucionó con un estudio del movimiento de la pieza en el armado, para solucionar esto se modificó los grados de giro después de cada movimiento de la pieza 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.
* Cubo se sale de la plataforma: este problema costó mucho solucionarlo debido a que si se modificaba la plataforma esta impedia que el cubo se pudiese mover de manera correcta por lo que se optó por modificar el brazo de movimiento superficial.
	1. Conclusiones

 El proceso de armado y creación del código para armado del cubo comenzó a tomar otro sentido del que se tenía al iniciar el proyecto ya que la visión era la de armar un robot y modificar el código, en el camino se tuvo que modificar el robot en casi un 70% y el código en un 100%.

* 1. Trabajo Futuro

 Lo que resta del proyecto es generar la conexión remota al robot, la cual ha ido bastante bien en su avance ya que por tema de que se había trabajado anteriormente con la programación de plataformas de aplicaciones, ya se tiene un mínimo suficiente de conocimiento para la interpretación y creación de aplicaciones para uso en dispositivos.