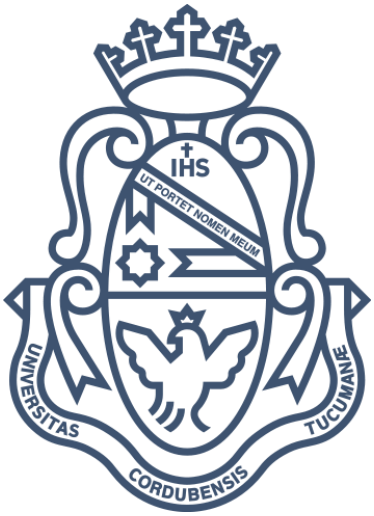
Universidad Nacional de Córdoba – Facultad de Ciencias Exactas Físicas y Naturales

Ingeniería Biomédica

Proyecto Integrador

Domótica aplicada a un paciente con discapacidad motriz.



Alumno:

Libson, Lucas Martín

Matrícula:

35.476.230

Director:

Beltramone, Diego Antonio

FCM
Facultad de
Ciencias Médicas

Ingeniería
Biomédica

Córdoba, diciembre de 2017

Resumen

Este proyecto tiene el propósito de idear, diseñar, programar e implementar un sistema domótico que podría permitirle a una persona con una discapacidad motriz (cuadriparesia) adquirir cierta autonomía en su vida cotidiana. Para llevarlo a cabo, se realizó un análisis con el objetivo de determinar cuáles habilidades habían sido reducidas y cuales se mantenían, para utilizarlas como método de entrada al sistema. Al ser un proyecto piloto, era necesario limitar el número de objetos a intervenir: Una cama ortopédica eléctrica y un elevador de personas con arnés. Ambos se utilizan a través de un control remoto y ya están instalados en el hogar a domotizar. El objetivo principal es realizar una intervención en los objetos nombrados para ser usados de otra forma, alternativa a la utlización del control remoto, y así lograr que sean más cómodos para el usuario.

Es importante destacar, que la implementación y la instalación del sistema ya ha sido parcialmente realizada, por lo que el usuario se encuentra actualmente utilizando una parte de la herramienta tecnológica asistiva. Este documento trata de sintetizar el "know how" de un sistema domótico aplicado a un paciente con discapacidad motriz.

Index Terms— Arduino, Discapacidad, Domótica, EasyVR, RF, Solución, Cuadriparesia

Abstract

This project had the purpose of devising, designing, programming and implementing a domotic system that could allow a person with a motor disability (quadriparesis) acquire autonomy in daily life. An analysis has been made in order to find out about his reduced capabilities and remaining skills, to take them as an input method for the system. Being a pilot project, it was necessary to limit the number of objects to control to two of them: An electric orthopedic bed and a person's elevator with harness. Both of them are activated with a remoted control and are already installed in the user's house. The goal is to make an intervention in the named objects for being used in order to be more comfortable for the user.

It is important to remark, that implementation and installation of the system has been partially made, so the user is already using one part of the described assistive technology tool. This document tries to synthesize the "know how" of a domotic system applied to a patient with motor disability.

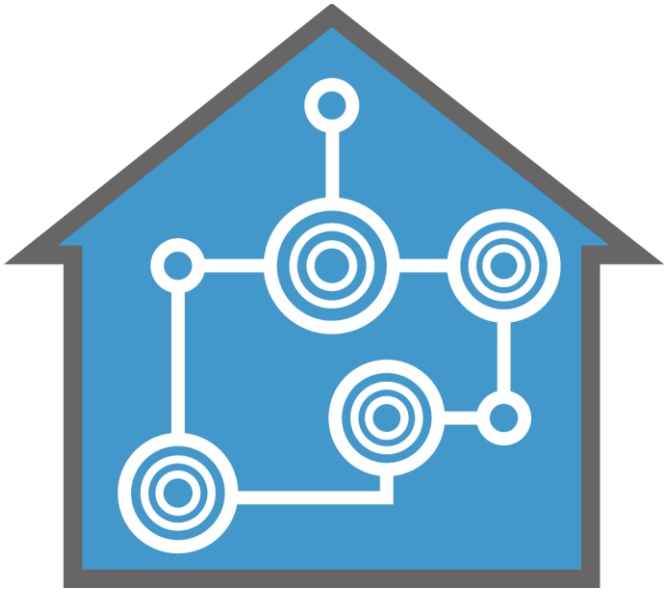
Index Terms— Arduino, Disability, Domotics, EasyVR, RF, Solution, Quadriparesis

Índice de contenido

Índice de ilustraciones

Índice de tablas

Marco teorico



Introducción

¿Qué es la domótica?

"La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema. Un sistema domótico es capaz de recoger información proveniente de unos sensores o entradas, procesarla y emitir órdenes a unos actuadores o salidas. El sistema puede acceder a redes exteriores de comunicación o información.

El sector de la domotización del hogar ha evolucionado considerablemente en los últimos años, y en la actualidad ofrece una oferta más consolidada. Hoy en día, la domótica aporta soluciones dirigidas a todo tipo de viviendas. En definitiva, la oferta es mejor y de mayor calidad, y su utilización es ahora más intuitiva y perfectamente manejable por cualquier usuario. Contribuye a aumentar la calidad de vida, hace más versátil la distribución de la casa, cambia las condiciones ambientales creando diferentes escenas predefinidas, y consigue que la vivienda sea más funcional al permitir desarrollar facetas domésticas, profesionales, y de ocio bajo un mismo techo. "(Qué es la domótica. Asociación Española de Domótica e Inmótica – CEDOM).

Los principales aportes que la domótica provee son:

- Ahorro energético:

Gestiona inteligentemente los recursos, utilizando las tarifas horarias de menor coste, y reduciendo así, la factura energética. Además, mediante la monitorización de consumos, se obtiene la información necesaria para modificar los hábitos y aumentar el ahorro y la eficiencia.

- Seguridad:

Puede realizar la vigilancia automática de personas, animales y bienes, así como de incidencias y averías.

- Aumento en el confort:

Permite regular y automatizar electrodomésticos, la climatización, ventilación, iluminación natural y artificial, persianas, toldos, puertas, cortinas, etc.

- Comunicación:

El control y supervisión remota de la vivienda permite la recepción de avisos de anomalías e información del funcionamiento de equipos e instalaciones. La instalación domótica permite la transmisión datos con redes locales (LAN) e Internet.

- Accesibilidad:

Facilita el manejo de los elementos del hogar a las personas con discapacidades de la forma que más se ajuste a sus necesidades.

Este proyecto estaría enfocado principalmente en este último punto. La accesibilidad se define como: “nivel en el que cualquier ser humano, más allá de su condición física o de sus facultades cognitivas, puede usar una cosa, disfrutar de un servicio o hacer uso de una infraestructura.

Existen diversas ayudas técnicas para impulsar la accesibilidad y equiparar las posibilidades de todas las personas. Esto supone que un espacio que presenta buenas condiciones de accesibilidad puede recibir a toda clase de gente sin que exista un perjuicio o una complicación para nadie. Las rampas para discapacitados, las sillas de ruedas, el alfabeto Braille y las señales auditivas son algunas de estas ayudas técnicas.

La accesibilidad supone un derecho que otorga a un individuo la posibilidad concreta y real de entrar, permanecer y recorrer un lugar con seguridad, comodidad y la mayor autonomía posible. “(Definición de accesibilidad. (2012). Autores: Julián Pérez Porto y Ana Gardey).

¿Qué es la discapacidad?

“La discapacidad es un término general que abarca las deficiencias, las limitaciones de la actividad y las restricciones de la participación. Las deficiencias son problemas que afectan a una estructura o función corporal; las limitaciones de la actividad son dificultades para ejecutar acciones o tareas, y las restricciones de la participación son problemas para participar en situaciones vitales.

Por consiguiente, la discapacidad es un fenómeno complejo que refleja una interacción entre las características del organismo humano y las características de la sociedad en la que vive. ”(OMS – Discapacidades. (2017))

La relación entre la domótica y discapacidad resulta explícita: la accesibilidad. La utilización de diversos conceptos alojados en el mundo de la domótica podría resultar en una herramienta indispensable a la hora de hablar de autonomía en la vida diaria de una persona con discapacidad. En general al utilizar la domótica en hogares y edificios lo que se está haciendo es que esos lugares sean accesibles para todos, convierte a la edilicia misma en la herramienta de rehabilitación.

Domótica

El término domótica viene de la unión de las palabras domus (que significa “casa” en latín) y tica (de automática, palabra en griego, “que funciona por sí sola”).

La bibliografía la define como: “Conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado. “(CIEC - Colegio de Ingenieros Especialistas de Córdoba CD - Comisión de Domótica).

Un sistema domótico es capaz de recoger información proveniente de unos sensores o entradas, procesarla y emitir órdenes a unos actuadores o salidas. El sistema puede acceder a redes exteriores de comunicación o información.

Dicho en otras palabras, se puede definir a la domótica como a una rama transdisciplinaria que busca obtener ciertas utilidades deseables sobre una casa. Estas utilidades pueden ser, por ejemplo:

- Ahorro energético:
  Climatización, control de persianas eléctricas, gestión eléctrica, etc.
- Confort:
  Iluminación, automatización de sistemas, integración del portero al teléfono, control vía Internet, generación de macros a pedido del usuario.
- Seguridad:
  Alarmas, cierre automático de puertas y persianas, detectores/alarmas de incendios/fugas de gas/escape de agua, alertas médicas, etc.
- Comunicaciones:
  Control remoto desde la PC o el celular, tele asistencia, informes de consumos y costes, transmisión de alarmas, etc.
- Accesibilidad:
  Se incluyen todas las aplicaciones o instalaciones destinadas a favorecer la autonomía de personas con algún tipo de discapacidad.

Y es justamente en este último punto a donde estaría enfocado este proyecto.

Arquitecturas

Una de las características que acompaña a cada sistema domótico puede definirse como la arquitectura, la cual puede ser:

- Arquitectura centralizada:

Un controlador centralizado recibe información de múltiples sensores y, una vez procesada, genera las órdenes oportunas para los actuadores.

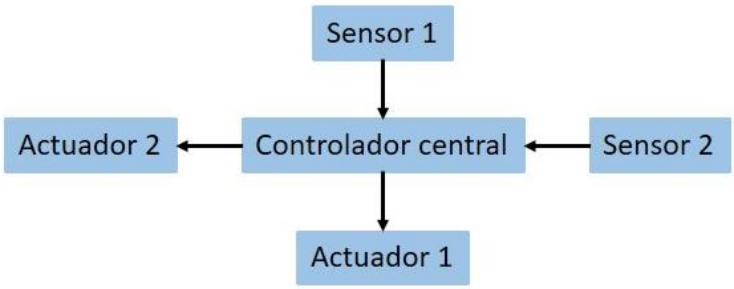


Ilustración 1: Arquitectura centralizada

- Arquitectura distribuida:

Toda la inteligencia del sistema está distribuida por todos los módulos, sean sensores o actuadores.

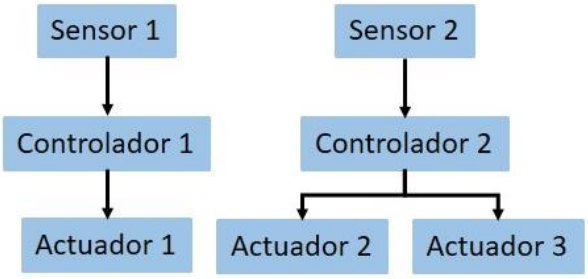


Ilustración 2: Arquitectura distribuida

- Arquitectura mixta:

Sistemas que combinan tanto arquitectura centralizada como distribuida. Estos sistemas disponen de varios dispositivos pequeños distribuidos capaces de adquirir y procesar la información de múltiples sensores y transmitirlos a módulos centrales de procesamiento.

A continuación, se comparan las distintas ventajas y desventajas que representa cada arquitectura:

Tipo de arquitectura	Ventajas	Desventajas
Centralizada	• Los elementos sensores y actuadores son del tipo universal • Coste reducido • Fácil uso y formación • Instalación sencilla	• Cableado significativo • Sistema muy dependiente de la central • Ampliabilidad reducida • Necesidad de una interfaz de usuario
Distribuida	• Mayor seguridad de funcionamiento • Posibilidad de rediseño • Se reduce el cableado • Fácil ampliabilidad	• Los elementos sensores y actuadores no son del tipo universal (limitados a la oferta) • Costes más elevados • Necesidad de un interfaz de usuario • Complejidad de programación
Mixta	• Mayor seguridad de funcionamiento • Posibilidad de rediseño • Fácil ampliabilidad • Sensores y actuadores del tipo universal • Coste moderado • Cableado moderado	• Resulta ser el que tiene la programación más compleja

Tabla 1: Ventajas y desventajas de las arquitecturas

Topologías

Otra cualidad destacable a tener en cuenta al hablar de sistemas domotizados es sin dudas la topología, que se refiere a la forma en que está diseñada la red, bien físicamente (rigiéndose de algunas características en su hardware) o bien lógicamente (basándose en las características internas de su software).

La topología de red es la representación geométrica de la relación entre todos los enlaces y los dispositivos que los conectan entre sí (habitualmente denominados nodos), y pueden ser:

- Topología estrella:

Cada dispositivo tiene un enlace punto a punto dedicado con el controlador central, habitualmente llamado concentrador. Los dispositivos no están directamente enlazados entre sí. Conexión utilizada típicamente por los sistemas centralizados donde existe un único controlador sobre el que pasa toda la información.

Ilustración 3: Tipología estrella

- Topología en anillo:

Cada dispositivo tiene una línea de conexión dedicada y punto a punto solamente con los dos dispositivos que están a sus lados. La señal pasa a lo largo del anillo en una dirección, o de dispositivo a dispositivo, hasta que alcanza su destino. Cada dispositivo del anillo incorpora un repetidor.

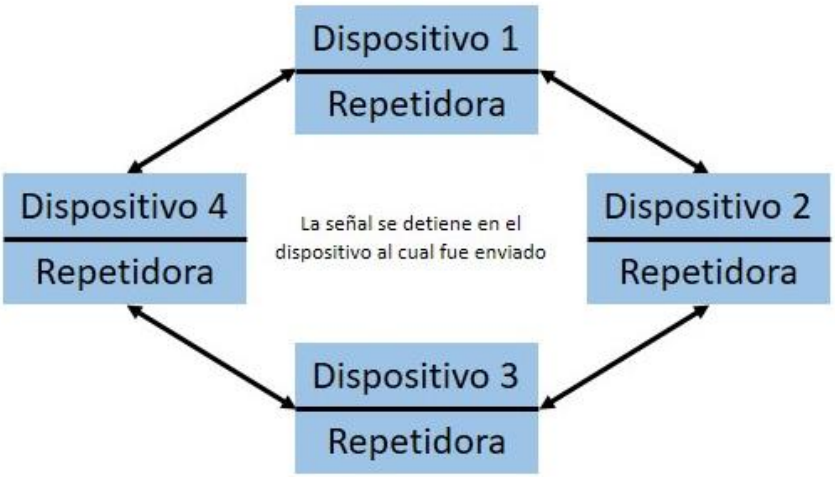


Ilustración 4: Tipología en anillo

- Topología en bus:

Un cable largo actúa como una red troncal que conecta todos los dispositivos en la red, dicha topología es multipunto. Los nodos se conectan al bus mediante cables de conexión y derivadores. Un cable de conexión es una conexión que va desde el dispositivo al cable principal. Un derivador es un conector que conecta al cable principal con el cable de conexión.

Ilustración 5: Tipología en bus

- Topología malla:

Cada dispositivo tiene un enlace punto a punto y dedicado con cualquier otro dispositivo. El término dedicado significa que el enlace conduce el tráfico únicamente entre los dos dispositivos que conecta. Conexión utilizada típicamente por los sistemas distribuidos en donde todos los dispositivos están intercomunicados entre sí.

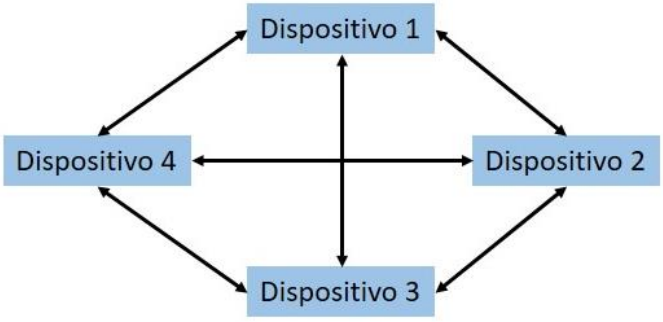


Ilustración 6: Tipología en malla

- Topología árbol:

Es una variante de la topología estrella. Como en la estrella, los nodos del árbol están conectados a un concentrador central que controla el tráfico de la red. Sin embargo, no todos los dispositivos se conectan directamente al concentrador central. La mayoría de los dispositivos se conectan a un concentrador secundario que, a su vez, se conecta al concentrador central. El controlador central del árbol es un concentrador activo. Un concentrador activo contiene un repetidor, es decir, un dispositivo hardware que regenera los patrones de bits recibidos antes de retransmitirlos.

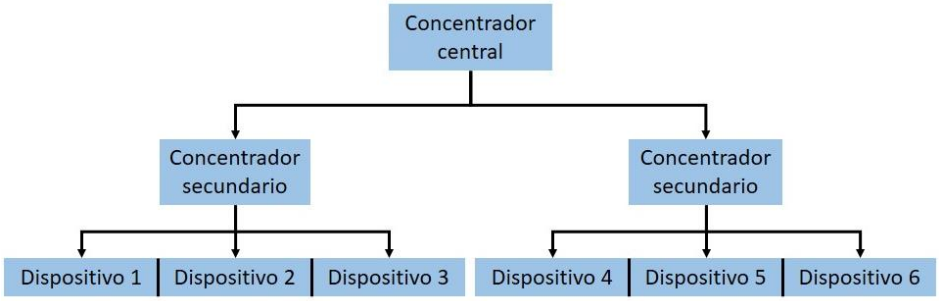


Ilustración 7: Tipología árbol

- Topología mixta:

Es la combinación de dos o más topología en una misma red.

Conexionado de la red

Otro aspecto importante a tener en cuenta es el cableado con el que cuenta la red para la transmisión/recepción de datos. El mismo puede ser:

- Cableado:
  - Línea Eléctrica: A través de una modulación sobre los cables de potencia se logran conectar todos los dispositivos.
  - Par trenzado: Se realiza un cableado independiente de la línea eléctrica, con este tipo de conexión se logran mejores velocidades de comunicación.
- Inalámbrico:
  - RF: Se denomina así a todo canal de transmisión inalámbrico. En este grupo entrarían todos los enlaces cuya tecnología del mismo fue desarrollada por el fabricante y no se encuentra bajo ningún estándar (no se recomienda utilizar comunicaciones RF no estandarizadas).
  - Bluetooth: Es una especificación industrial para redes inalámbricas de área personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz.
  - WiFi: Es una marca de la Wi-Fi Alliance, que es la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local.

- Óptico:
  - Infrarrojo: Esta tecnología está basada en rayos luminosos que se mueven en el espectro infrarrojo. Los estándares soportan una amplia gama de dispositivos eléctricos, informáticos y de comunicaciones, permite la comunicación bidireccional entre dos extremos a velocidades que oscilan entre los 9.600 bps y los 4 Mbps.
  - Fibra óptica: La fibra óptica es un medio de transmisión empleado habitualmente en redes de datos; consta de un hilo muy fino de material transparente: vidrio o materiales plásticos, por el que se envían pulsos de luz que representan los datos a transmitir. El haz de luz queda completamente confinado y se propaga por el interior de la fibra con un ángulo de reflexión por encima del ángulo límite de reflexión total, en función de la ley de Snell. La fuente de luz puede ser láser o un LED. Las fibras se utilizan ampliamente en telecomunicaciones, ya que permiten enviar gran cantidad de datos a una gran distancia, con velocidades similares a las de radio o cable. Son el medio de transmisión por excelencia al ser inmune a las interferencias electromagnéticas, también se utilizan para redes locales, en donde se necesite aprovechar las ventajas de la fibra óptica sobre otros medios de transmisión.
- Mixto:
  - Se denomina así cuando los enlaces están formados por distintos tipos de tecnología.

Nodos

El nodo es el componente del sistema domótico que interconecta los dispositivos entre sí. Como se vio anteriormente, una vivienda domotizada puede tener uno solo, como es en el caso de la arquitectura centralizada, o puede tener varios nodos interconectados o no entre ellos mismos, como es en el caso de la arquitectura distribuida. Existen diferentes tipos de nodos, según la función que se solicita de ellos:

- Nodos de control estándar: son los encargados de controlar los parámetros de cada estancia. Son los que reciben la información proveniente de los sensores del sistema y efectúan la respuesta solicitada por el usuario sobre los actuadores.
- Nodos de supervisión: dedicados a realizar la interfaz con el usuario. Cada función que el usuario necesita para supervisar y controlar el sistema está implementada en el correspondiente nodo. De esta manera, el usuario puede elegir para su vivienda las funciones que considere necesarias.
- Nodos de comunicación: Dedicados específicamente a soportar la red de comunicaciones de la vivienda. Son los routers o gateways.

Discapacidad

"Discapacidad es un término general que abarca las deficiencias, las limitaciones de la actividad y las restricciones de la participación. Las deficiencias son problemas que afectan a una estructura o función corporal; las limitaciones de la actividad son dificultades para ejecutar acciones o tareas, y las restricciones de la participación son problemas para participar en situaciones vitales.

Por consiguiente, la discapacidad es un fenómeno complejo que refleja una interacción entre las características del organismo humano y las características de la sociedad en la que vive. "(OMS – Discapacidades. (2017)).

Según ; dicho de otra forma, se estima que más de 1000 millones de personas en el mundo poseen algún tipo de discapacidad, y que, dentro de esa parte de la población, hay casi 200 millones de personas que experimentan alguna dificultad considerable en su funcionamiento diario. También se asegura que este número iría en aumento debido al incremento de la edad promedio en la tierra; esto quiere decir que gracias a los avances científicos relacionados a la medicina la población promedio fallece a una edad cada vez mayor, incrementando notablemente la población con alguna discapacidad devenida por enfermedades discapacitantes propias de la edad.

La "Clasificación Internacional del Funcionamiento, de la Discapacidad y de la Salud" (CIF), órgano perteneciente a la OMS, distingue entre las funciones del cuerpo y las estructuras del cuerpo que aportan a esta función. La CIF enumera 9 dominios del funcionamiento que pueden verse afectados por alguna discapacidad:

- Aprendiendo y aplicando conocimiento
- Tareas y demandas generales
- Comunicación
- Movilidad
- Cuidado en sí mismo
- Vida doméstica
- Interacciones y relaciones interpersonales
- Áreas importantes de la vida
- Vida de la comunidad, social y cívica

Clasificación Internacional del Funcionamiento, de la Discapacidad y de la Salud

La Organización Mundial de la Salud cuenta entre sus grupos de trabajo con uno dedicado a la clasificación, evaluación, encuestas y terminología aplicables al campo de la salud. La "*Clasificación Internacional del Funcionamiento, de la Discapacidad y de la Salud*" (CIF) tiene como objetivo principal de brindar un lenguaje unificado y estandarizado, y un marco conceptual para la descripción de la salud. Define los componentes de la salud y algunos componentes relacionados con la salud en dos listados básicos:

1. Funciones y Estrucuras Corporales
2. Actividades – Participación

La CIF enumera además Factores Ambientales que interactúan con estos "constructos". Por lo tanto, la clasificación permite a sus usuarios elaborar un perfil de gran utilidad sobre el funcionamiento, la discapacidad y la salud del individuo.

Objetivos de la CIF

Los objetivos de la CIF son:

- Proporcionar una base científica para la comprensión y el estudio de la salud y los estados relacionados con ella, los resultados y los determinantes.
- Establecer un lenguaje común para describir la salud y los estados relacionados con ella, para mejorar la comunicación entre distintos usuarios, tales como profesionales de la salud, investigadores, diseñadores de políticas sanitarias y la población general, incluyendo a las personas con discapacidades.
- Permitir la comparación de datos entre países, entre disciplinas sanitarias, entre los servicios, y en diferentes momentos a lo largo del tiempo.
- Proporcionar un esquema de codificación sistematizado para ser aplicado en los sistemas de información sanitaria.

Estos objetivos están relacionados entre si, ya que la necesidad y el uso de la CIF requiere la construcción de un sistema relevante y útil que pueda aplicarse en distintos ambitos en política sanitaria, en evaluación de la calidad asistencial y para la evaluación de consecuencias en diferentes culturas.

Asimismo, la CIF plantea un esquema, incorporando los elementos sociales, que sintetiza a grandes rasgos los factores involucrados en cualquier actividad que conlleva a la utilización de determinadas habilidades, que podrían estar disminuidas en una persona con alguna discapacidad:

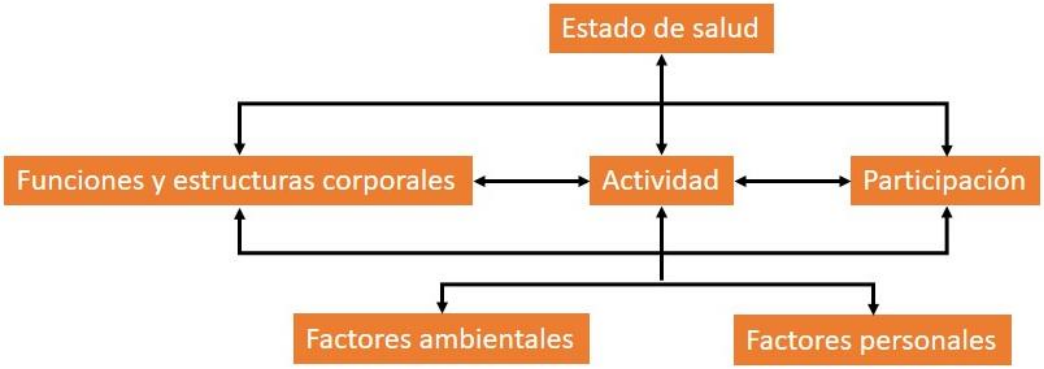


Ilustración 8: Factores que afectarían en la realización de una actividad por determinada persona

Bajo esta óptica, resulta clave para este proyecto que la Organización Mundial de la Salud definiese la discapacidad involucrando a las características de la sociedad en las que está inmerso la persona en cuestión. Esto quiere decir que una adaptabilidad por parte de la sociedad reduciría significativamente el impacto que una discapacidad tiene sobre una persona en sus quehaceres diarios. Resulta interesante, entonces, el planteo d e empezar adaptando algo con lo que una persona con discapacidad convive cotidianamente: su hogar.

Encuesta Nacional de Personas con Discapacidad (ENDI)

"La Encuesta Nacional de Personas con Discapacidad (ENDI) se realizó en la Provincia de Córdoba durante el primer semestre del año 2003. Se trató de la Primera Encuesta Nacional de Personas con Discapacidad -Complementaria del Censo Nacional de Población, Hogares y Viviendas 2001-, cuyo operativo estuvo a cargo de la Gerencia de Estadísticas y Censos del Gobierno de Córdoba conjuntamente con el Instituto Nacional de Estadística y Censos (INDEC).

Se efectuó en dos etapas. En la primera, en la cédula censal del Censo 2001 se incluyó una pregunta destinada a detectar hogares con al menos una persona con discapacidad. Los datos obtenidos proporcionaron el marco para seleccionar la muestra de hogares en la que se aplicó la encuesta. La muestra estuvo conformada por una mayoría de hogares con al menos una persona con discapacidad y una proporción menor de hogares sin ninguna persona con discapacidad." (Dirección General de Estadísticas y Censos. Secretaría de la Gobernación. 2003).

Con el objetivo cuantificar y caracterizar a las personas con discapacidad en lo referente al desenvolvimiento de la vida cotidiana dentro de su entorno físico y social, la Encuesta relevó información sobre los siguientes temas:

- Tipo y causa de la discapacidad.
- Edad de origen de la misma.

- Tipo de ayuda que reciben las personas con discapacidad por parte de las obras sociales, organismos estatales, organismos no gubernamentales, etc.
- Autonomía.
- Uso de beneficios legales y sociales.
- Características sociodemográficas de todos los miembros del hogar.
- Características y adaptaciones de la vivienda.

A continuación, se muestra la tabla que resume los datos obtenidos mediante la Encuesta:

Grupos de edad	Población con discapacidad		Población con una discapacidad		Tipo de discapacidad											
					Sólo visual		Sólo auditiva		Sólo del habla		Sólo motora		Sólo mental		Sólo otra discapacidad	
-	Valor absoluto	%	Valor absoluto	%	Valor absoluto	%	Valor absoluto	%	Valor absoluto	%	Valor absoluto	%	Valor absoluto	%	Valor absoluto	%
Total	**204.829**	**100,0**	**145.017**	**70,8**	**41.502**	**28,6**	**24.733**	**17,1**	(a)	(a)	**51.872**	**35,8**	**21.680**	**14,9**	(a)	(a)
0-14	19.657	100,0	15.265	77,7	(a)	(a)	(a)	(a)	(a)	(a)	(a)	(a)	(a)	(a)	(a)	(a)
15-64	102.378	100,0	83.569	81,6	27.888	33,4	12.536	15,0	(a)	(a)	26.894	32,2	14.296	17,1	(a)	(a)
65 y más	82.794	100,0	46.183	55,8	(a)	(a)	10.933	23,7	(a)	(a)	23.366	50,6	(a)	(a)	(a)	(a)

Tabla 2: Datos estadìsticos obtenidos de la Encuesta Nacional de Personas con Discapacidad (ENDI).
**(a) coeficiente de variación mayor al 25%*

Es importante remarcar el impacto que provoca una discapacidad del tipo motriz (remarcada en rojo) en la población que encuestada. Esta condición es la segunda causa de afección discapacitante (luego de la discapacidad visual).

Discapacidad motriz

"La Discapacidad Motriz (DM) es una condición de vida que afecta el control y movimiento del cuerpo, generando alteraciones en el desplazamiento, equilibrio, manipulación, habla y respiración de las personas que la padecen, limitando su desarrollo personal y social.

Esta discapacidad se presenta cuando existen alteraciones en los músculos, huesos, articulaciones o medula espinal, así como por alguna afectación del cerebro en el área motriz impactando en la movilidad de la persona.

Es importante mencionar que la DM no implica afectación en el funcionamiento cerebral de la persona, no es una consideración que afecte el rendimiento intelectual de la misma." (Dirección General de Educación Especial. Discapacidad Motriz. 1999)

Las dificultades que presenta una persona con Discapacidad Motriz pueden ser muy variadas dependiendo del momento de aparición, los grupos musculares afectados (topografía), el origen y el grado de afectación (ligera, moderada o grave):

- Según el momento de aparición:
  - **Antes del nacimiento o prenatal**: Tal es el caso de malformaciones congénitas, mielomeningocele, luxación congénita de cadera, etc.

- **Perinatales:** Cuando existe afectación (alteración o perdida) del control motriz por Enfermedad Motriz Cerebral (EMOC).
- **Después del nacimiento**: Miopatías, como la distrofia muscular progresiva de Duchenne o la distrofia escapular, afecciones cráneo-cefálicas, traumatismos cráneo-encefálicos-vertebrales, tumores, etc.

- Grupos musculares afectados:

Parálisis	Paresias: Parálisis leve o incompleta
Monoplejía: Afecta un solo miembro ya sea brazo o pierna. **Hemiplejía**: Afecta a un lado del cuerpo, izquierdo o derecho. **Paraplejía**: Parálisis de los dos miembros inferiores. **Cuadriplejía**: Parálisis de los cuatro miembros.	**Monoparesia**: De un solo miembro. **Hemiparesia**: De un lado del cuerpo (derecho o izquierdo). **Paraparesia**: De los dos miembros inferiores. **Cuadriparesia**: Parálisis leve de los cuatro miembros.

Tabla 3: DM según los músculos afectados

- Según la etiología:
  - Por transmisión genética
  - Por infecciones microbianas
  - Por traumatismos
  - Otras de origen desconocido

- En función de su origen

Cerebral	• Parálisis cerebral • Traumatismo craneoencefálico • Tumores
Espinal	• Poliomelitis • Espina bífida • Lesiones medulares degenerativas • Traumatismo medular
Muscular	• Miopatías
Óseo-articular	• Malformaciones congénitas (amputaciones, luxaciones, artrogriposis) • Distróficas (condrodistrofia, osteogénesis imperfecta) • Microbianas (osteomelitis aguda, tuberculosis, óseo-articular). • Reumatismos infantiles (Reumatismo articular agudo, reumatismo crónico) • Lesiones ósteo-articulares por desviación del caquis (cifosis, escoliosis, lordosis)

Tabla 4: DM según su origen

Accesibilidad

La accesibilidad es el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o físicas. Es indispensable e imprescindible, ya que se trata de una condición necesaria para la participación de todas las personas independientemente de las posibles limitaciones funcionales que puedan tener.

Para promover la accesibilidad se hace uso de ciertas facilidades que ayudan a salvar los obstáculos o barreras de accesibilidad del entorno, consiguiendo que estas personas realicen la misma acción que pudiera llevar a cabo una persona sin ningún tipo de discapacidad. Estas facilidades son llamadas ayudas técnicas. Entre éstas se encuentran el alfabeto Braille, la lengua de señas, las sillas de ruedas, las señales auditivas de los semáforos, etc.

La accesibilidad en Argentina

En Argentina, en el año 1981 se sancionó la Ley Nacional Nº 22.431, que crea el *"sistema de protección integral de las personas con discapacidad"* regulando la accesibilidad al medio físico. Detalla claramente las condiciones que deben reunir los espacios para ser accesibles: espacios arquitectónicos, urbanos y del transporte.

Posteriormente, en el año 1994, se sanciona la Ley Nacional Nº 24.314 que amplía la 22.431 y establece que:

"A los fines de la presente ley, entiéndese por accesibilidad a la posibilidad de las personas con movilidad reducida de gozar de las adecuadas condiciones de seguridad y autonomía como elemento primordial para el desarrollo de las actividades de la vida diaria sin restricciones derivadas del ámbito fisico urbano, arquitectónico o del transporte para su integración y equiparación de oportunidades." (Ley 22.314. Accesibilidad de personas con movilidad reducida. Argentina, 8 de Abril de 1994.)

Tecnologías de apoyo

El uso de las tecnologías como medio para incrementar, mantener o mejorar las capacidades funcionales de los individuos es una práctica común en el ámbito de la intervención con personas con discapacidad (Alcantud y Soto, 2003). Con tecnología de apoyo se hace referencia a la ayuda técnica tanto en accesibilidad como en movilidad.

Se considera una tecnología de apoyo a todo tipo de equipo, objeto, sistema, producto, máquina, instrumento, programa o servicio que puede ser usado para suplir, aumentar, mantener, compensar o mejorar las capacidades funcionales de las personas con impedimento o discapacidad (motriz, sensorial o cognitiva). También es llamada tecnología de "adaptación" o de "ayuda" para la vida independiente, ya que les facilita a los individuos que las utilizan, llevar a cabo tareas que antes eran incapaces de cumplir o tenían grandes dificultades para realizarlas.

Por lo tanto, cabe destacar la importancia que tienen tres conceptos fundamentales a la hora de hablar de accesibilidad y las tecnologías de apoyo que se podrían incorporar:

- Utilidad: Capacidad que tiene una cosa de servir o de ser aprovechada para un fin determinado.
- Usabilidad: Facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. También se la describe como la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico, incorporando así el entorno y las habilidades que podría tener el usuario para hacer uso de la herramienta en cuestión.
- Flexibilidad: Capacidad para adaptarse con facilidad a las diversas circunstancias o para acomodar las normas a las distintas situaciones o necesidades.

Cuando estos conceptos logran incorporarse a la tecnología utilizada para asistir a la ejecución de una acción por determinada persona, es cuando puede considerarse que se lograron alcanzar en gran medida los objetivos propuestos por este proyecto. Resulta, tal vez, de gran interés desarrollar una herramienta lo suficientemente pulida como para considerarla sofisticada desde el punto de vista tecnológico, pero esto de nada serviría si al usuario la incorporación de esta tecnología le resulte aún más dificultosa que la realización de la actividad sin ella. Dicho de otra forma, se trata de enfocar los mayores esfuerzos en hacer que el producto sea lo más adaptable posible a las habilidades que el usuario pueda tener, reduciendo así el impacto que las habilidades disminuidas implican en una persona que tiene alguna discapacidad.

Los productos de apoyo se clasifican, según la norma ISO 9999, de acuerdo a su función. La clasificación consta de tres niveles jerárquicos y cada código consta de tres pares de dígitos. A continuación, se definirán términos utilizados en la norma aplicada:

- Actividad: Realización de una tarea o acción por parte de un individuo.
- Limitación en la actividad: Dificultades que un individuo puede tener en el desempeño/realización de actividades.
- Productos de apoyo: Cualquier producto (incluyendo dispositivos, equipo, instrumentos, tecnología y software) fabricado especialmente o disponible en el mercado, para prevenir, compensar, controlar, mitigar o neutralizar deficiencias, limitaciones en la actividad y restricciones en la participación.

Clasificación del proyecto según la norma ISO 9999

El proyecto, según la norma tratada, se clasifica de la siguiente forma. La selección se encuentra marcada en rojo:

Clasificación a un nivel: Clases

- 04 Productos de apoyo para tratamiento médico personalizado
- 05 Productos de apoyo para el entrenamiento/aprendizaje de capacidades
- 06 Ortesis y prótesis
- 09 Productos de apoyo para el cuidado y la protección personal
- 12 Productos de apoyo para la movilidad personal
- 15 Productos de apoyo para actividades domésticas
- 18 Mobiliario y adaptaciones para viviendas y otros inmuebles
- 22 Productos de apoyo para la comunicación y la información
- 24 Productos de apoyo para la manipulación de objetos y dispositivos
- 27 Productos de apoyo para mejorar el ambiente, herramientas y maquinas
- 30 Productos de apoyo para el esparcimiento

Clasificación a dos niveles: Clases y subclases

- 24 04 Materiales y herramientas para marcar
- 24 06 Productos de apoyo para manipular recipientes
- 24 09 Productos de apoyo para accionar y/o controlar dispositivos
- 24 13 Productos de apoyo para controlar a distancia
- 24 18 Productos de apoyo para compensar las funciones del brazo
- 24 21 Productos de apoyo para alcanzar a distancia
- 24 24 Productos de apoyo para colocación
- 24 27 Productos de apoyo para fijación
- 24 30 Productos de apoyo para reposicionar y levantar
- 24 36 Productos de apoyo para cargar y transportar
- 24 39 Vehículos de transporte industrial
- 24 42 Transportadores
- 24 45 Grúas

Clasificación a tres niveles:

- 24 13 03 Sistemas de control remoto
  Sistemas para operar dispositivos a distancia
  Sistemas de control de entorno incluidos
- 24 13 06 Software para control de entorno

Por lo tanto, la codificación que encasilla este trabajo según la norma es 24 13 03 y corresponde a los *"sistemas por control remoto para operar dipositivos a distancia, incluyendo el control de entorno"*.

Cobertura por parte de Obras Sociales

Cualquier persona que tenga un certificado de discapacidad expedido en los términos del art. 3 de la ley 22.431 y art. 10° de la ley 24.901 tiene una cobertura económica total por parte de su obra social para aquellos bienes o servicios que se consideren necesario para la rehabilitación, autonomía, reeducación, etc. A continuación, se citarán brevemente las leyes nombradas:

Ley N° 22.431

Asegura un "sistema de protección integral de las personas discapacitadas, tendiente a asegurar a éstas su atención médica, su educación y su seguridad social, así como a concederles las franquicias y estímulos que permitan en lo posible neutralizar la desventaja que la discapacidad les provoca y les den oportunidad, mediante su esfuerzo, de desempeñar en la comunidad un rol equivalente al que ejercen las personas normales." (Ley N° 22.431. Sistema de protección integral de los discapacitados. Buenos Aires, 16 de marzo de 1981)

Ley 24.901

Reglamenta un "sistema de prestaciones básicas de atención integral a favor de las personas con discapacidad, contemplando acciones de prevención, asistencia, promoción y protección, con el objeto de brindarles una cobertura integral a sus necesidades y requerimientos.

Cobertura económica: Se otorgará cobertura económica con el fin de ayudar económicamente a una persona con discapacidad y/o su grupo familiar afectados por una situación económica deficitaria, facilitando la permanencia de la persona con discapacidad en el ámbito social donde reside o elija vivir." (Ley 24.901. Sistema de prestaciones básicas en habilitación y rehabilitación integral a favor de las personas con discapacidad. Buenos Aires, 5 de noviembre de 1997)

En la implementación de este proyecto, se plantea un conteo de los gastos que el mismo conlleva para que que, una vez materializada la solución, se utilice el derecho que las leyes 22.431 y 24.901 otorgan. En otras palabras, la intención es la de presentar un documento que justifique la incorporación de este sistema al hogar de Santiago, detallando gastos con el fin de recuperar el dinero invertido mediante el reintegro contra factura, aportado por su obra social. (Ver "Tabla 11: Costos del proyecto")

Relación existente entre la domótica y la discapacidad

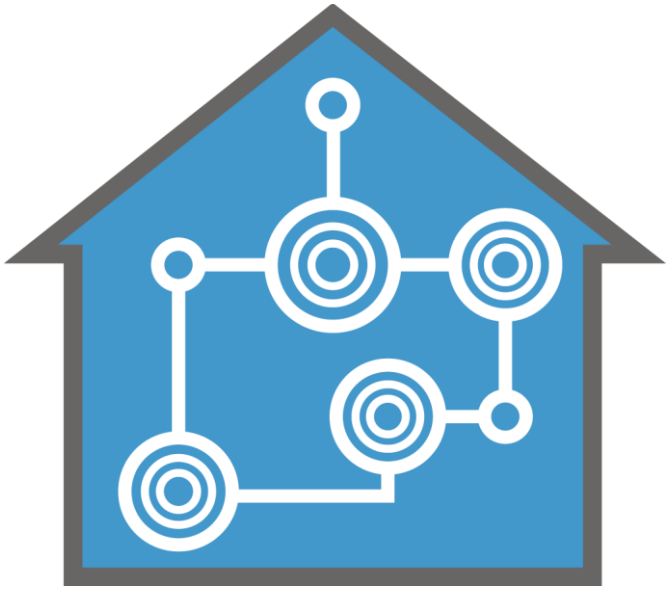
En consonancia con el capítulo anterior, la relación resulta explícita. La utilización de diversos conceptos alojados en el mundo de la domótica podría resultar en una herramienta indispensable a la hora de hablar de autonomía en la vida diaria de una persona con discapacidad. En general al utilizar la domótica en hogares y edificios lo que se está haciendo es que esos lugares sean accesibles para todos, convierte a la edilicia misma en la herramienta de rehabilitación.

La domótica asistiva se centra en hacer posible que la tercera edad y las personas con discapacidad puedan gozar de las instalaciones de su hogar de forma segura y autónoma. Se está convirtiendo en una opción viable para los ancianos y las personas con capacidades reducidas que prefieren quedarse en la comodidad de sus hogares en lugar de trasladarse a un centro de atención médica. Este campo utiliza mucha de la misma tecnología y equipo de automatización que la domótica general para la seguridad, el entretenimiento y el ahorro de la energía, pero teniendo como objetivo la situación específica en la que se encuentran las personas ancianas o discapacitadas; en especial, su sencillez de manejo, los botones grandes y adaptación a la situación del usuario.

A continuación, un listado de las soluciones domóticas más ventajosas para actualizar el hogar de personas de edad avanzada, que les permitirá realizar sus tareas con mayor facilidad, estar más seguros y estar en contacto permanente con las personas que los cuidan:

- Cerraduras: facilitan el acceso a la vivienda, mejoran la confirmación de seguridad
- Sistema de seguridad: monitoriza la casa de forma remota, permite vigilar lugares apartados o inaccesibles, envía alertas por intrusos.
- Iluminación: automatiza los ambientes del hogar, garantiza la visibilidad en cualquier condición
- Sensores inteligentes: monitorizan consumo de servicios, y alertan sobre intrusiones.
- Dispositivos de activación por voz: permiten activar o desactivar funciones solo con el habla, manos libres en todo momento.
- Termostatos inteligentes: permiten regular y automatizar la climatización (temperatura, humedad) del hogar.

Desarrollo



Caso presentado: Santiago

Fue necesario encontrar una necesidad no resuelta en una persona con alguna capacidad reducida, para así poder ser abordada por una solución domótica. Dicho de otra forma, la finalización de este trabajo culmina con la materialización de la solución propuesta, y es justamente al momento de explicar "por qué" o el "cómo" de cada detalle que involucran la implementación, en donde es necesario hacer una breve descripción de la situación que acompaña al usuario del dispositivo.

Santiago es abogado, tiene 42 años y es casado. Nació el 6 de Julio de 1974. Cursó la carrera de Letras Modernas desde el año 1992 hasta el año 1994, para luego en el año 1996 comenzar y terminar en el siguiente año la carrera de Filosofía, ambas en la Facultad de Humanidades en la Universidad Nacional de Córdoba. Fue en el año 1995 donde comenzó su carrera de Abogacía en la Facultad de Derecho de la Universidad Nacional de Córdoba.

En 1993 Santiago tuvo una lesión traumática en las vértebras cervicales 4, 5 y 6, generando cuadriplejía hasta la actualidad, lo cual genera una disminución en sus capacidades motrices.

Debido a su formación, tiene amplia experiencia en Derecho relacionado a las personas con movilidad reducida (PMR). Además de haber prestado colaboración y asesoría al ex Director de Discapacidad Municipal (Lic. Javier Torresi), también ha trabajado activamente con distintas instituciones como Acceso ya (ONG nacional), Derecho Sobre Ruedas, Fundación Neuro Cinesis y otras. En 2012 funda un grupo de trabajo a partir de la red social Facebook (Discapacidad en Córdoba) que nuclea a más de 2.000 miembros, conformado en su gran mayoría por instituciones, empresas y profesionales vinculados directamente con las PMR. Cada miembro de este grupo realiza acciones concretas y diversas (asesoramiento y ayuda mutua, difusión en medios masivos, presentaciones legales antes organismos oficiales para el cumplimiento de leyes y ordenanzas, servicios gratuitos a PMR carenciados), lo que convierte al (meta) grupo Discapacidad en Córdoba en un dinámico potenciador de las capacidades colectivas o de las partes con intereses convergentes.

En agosto de 2013 es reconocido con el otorgamiento de un subsidio del Banco Mundial, canalizado por el Fondo Nacional Sectorial (FONARSEC), para la creación de una empresa de base tecnológica con el propósito de fabricar soluciones tecnológicas para personas con movilidad reducida. Se trata de productos de alta gama, similares a los existentes en países altamente desarrollados, que tienen un alto impacto social en la calidad de vida de las personas con movilidad reducida y sus familiares y allegados.

No obstante, la circunstancia clave a destacar, a la hora de hacer un análisis sobre el usuario final, es que Santiago posee una base importante en lo que refiere al manejo de múltiples paquetes de software y a la utilización de herramientas mediante feedbacks audiovisuales. Sobre todo, este último punto resulta interesante a la hora de evaluar las posibilidades para diseñar el dispositivo domótico, ya que se puede contar

con la facilidad de aprendizaje por parte del usuario para implementar este tipo de retroalimentación.

Luego, Santiago planteó las dos necesidades reales que requería solucionar con el sistema domótico:

- Modificar el respaldo y las piernas de la camilla ortopédica eléctrica.
- Que el elevador de personas con arnés se traslade de un sitio a otro al realizarse transferencias desde la cama a la silla y de la silla a la cama.

Al realizarse un breve análisis de las capacidades que Santiago conservaba, primó la modulación. Sumado a lo descipto anteriormente, la familiarización en el uso de tecnologías comandadas por la voz, la solución más permeable en este caso sería la utilización de comandos verbales.

Se utilizó un consentimiento informado, donde se le explicó a Santiago todos los detalles que conllevan el proyecto, brindándole la documentación que se encuentra en el Anexo: “Declaración de Consentimiento Informado”.

Entonces, los objetos abordados para ser domotizados consistían en dos elementos que Santiago ya tenía en su domicilio:

- Una cama ortopédica eléctrica.
- Un elevador de personas con arnés.

A continuación, se detallarán algunos aspectos relevantes:

Cama ortopédica eléctrica

La cama ortopédica se encuentra instalada en el domicilio de Santiago. La marca es "Linak" y se utiliza actualmente a través de un control remoto que activa los servomotores, situación que resulta incómoda para el usuario.

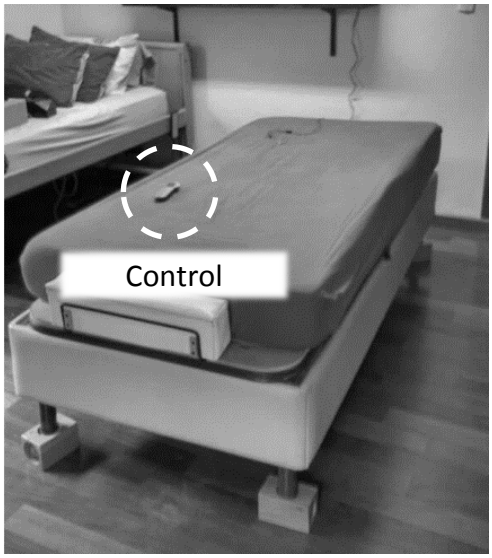


Ilustración 9: Cama ortopédica eléctrica "Linak". Señalado el control remoto que activa la misma.

Cuenta con un sistema de dos motores, de los cuales uno sube y baja el respaldo y el otro sube y baja las piernas. Estos motores tienen su activación mediante un relé, ya integrado en el circuito propio de la camilla.

Una medida de seguridad que ya tiene incorporada es un sensor de fin de carrera, que frena cualquier motor una vez que el respaldo o las piernas hayan alcanzado una posición extrema. Esto quiere decir, a modo de ejemplo, que cuando el respaldo se encuentra en su máxima posición de elevación, es imposible seguir activando el motor que eleva el respaldo. A la hora de diseñar el proyecto, se considera una cuestión de seguridad menos a considerar, puesto que ya viene incorporada en el mismo producto que se va a intervenir.

Elevador de personas con arnés

El diseño e implementación del elevador de personas con arnés se encuentra instalado en la edilicia de Santiago fue diseñada por el mismo. La misma cuenta con un movimiento horizontal a través de un riel que atraviesa el techo de la habitación de extremo a extremo. Además, tiene incorporado un movimiento vertical mediante un malacate, activado a través de un pulsador para subir y otro para bajar. Cabe destacar que el movimiento horizontal es completamente manual, mientras que el movimiento

vertical se efectua a través del motor del malacate. El motor se activa, ya sea para subir o bajar el arnés, mediante un control externo con pulsadores manuales.

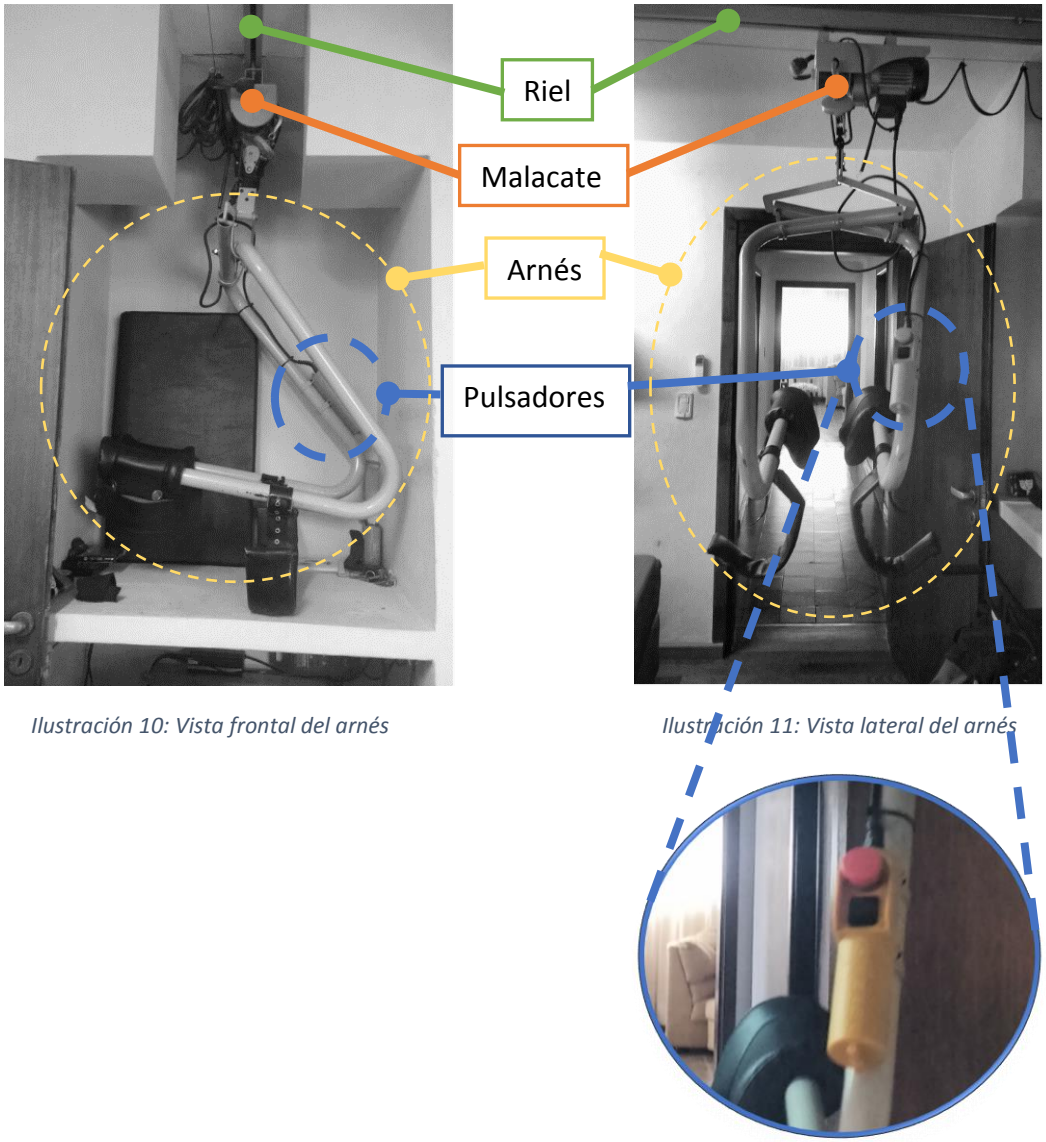


Ilustración 10: Vista frontal del arnés

Ilustración 11: Vista lateral del arnés

Ilustración 12: Zoom pulsadores

La intención es que Santiago pueda utilizar estos dos elementos, que normalmente son de utilización manual, mediante el uso de comandos expresados de forma oral, brindándole la autonomía e independencia buscada con la solución.

Solución propuesta

Este proyecto tiene la finalidad de brindar autonomía, mediante una solución domótica, a una persona con una discapacidad motriz. Dicha herramienta debería posibilitarle el uso de una cama ortopédica eléctrica y un elevador de personas con arnés, ambos ya instalados en la residencia del usuario y de accionamiento manual. Asimismo, se idea la explotación de la modulación y habla de Santiago, habilidades que se conservan intactas. Dicho de otras palabras, se plantea un dispositivo que actúe sobre los artefactos a domotizar activado por órdenes verbales. A continuación, un esquema del funcionamiento de este sistema:

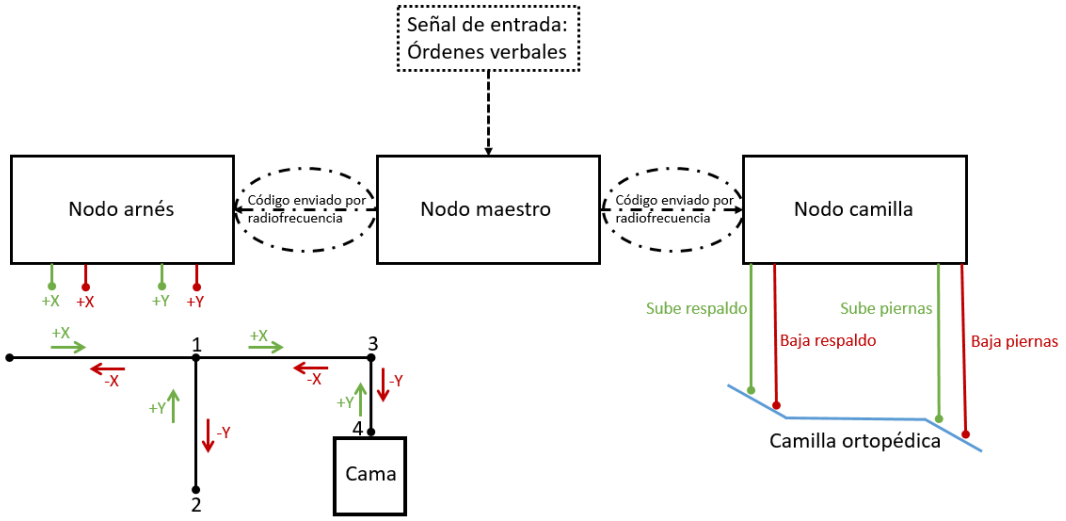


Ilustración 13: Esquema del funcionamiento inalámbrico sistema domótico con actuadores

Para la solución se propone un sistema mixto de 3 nodos:

- Un nodo maestro.
- Un nodo que controla la camilla.
- Un nodo que controla el arnés.

La tipología sería estrella, según el siguiente esquema:

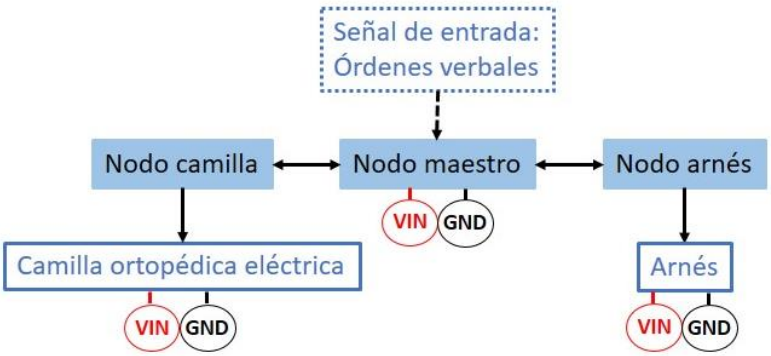


Ilustración 14: Tipología de la solución propuesta

Al momento de elegir el modo de conexionado, en primera instancia se optó por un sistema domótico cableado mediante un par trenzado, independiente de la línea eléctrica; la misma se utilizaría sólo para energizar los nodos. No obstante, al hacer un análisis de la disposición física del cableado en la habitación del usuario, se optó por un sistema inalámbrico. Esto incurriría no solo en un ahorro económico debido a la facilidad en la instalación (cada nodo se conectaría directamente a la tensión de línea para energizarse), sino también a una mejor presentación estética y acabado final del producto. Tras una búsqueda por diferentes tecnologías para incorporar este tipo de comunicación entre nodos, se optó por la radiofrecuencia por su bajo costo y fácil implementación. A continuación, se ilustra la actualización en el esquema según la metodología de cableado elegida:

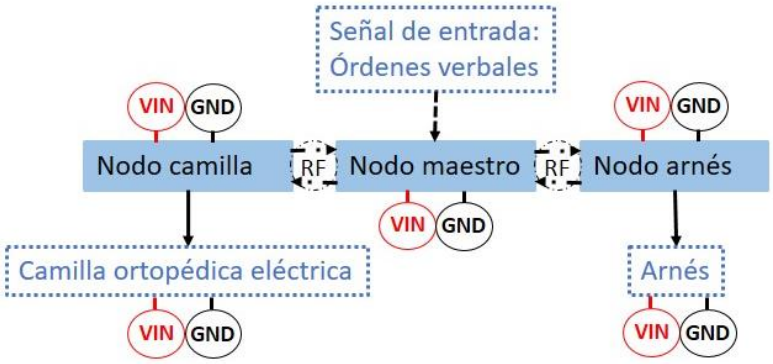


Ilustración 15: Tipología inalámbrica de la solución propuesta

La intención sería que los nodos “camilla” y “arnés” sean una suerte de intervención en el mecanismo que los acciona; esto sería, que reciban las órdenes desde el nodo maestro y generen un cambio en la posición de los mismos. El siguiente paso sería esquematizar las diferentes posiciones deberían adquirir la cama ortopédica o el arnés según las necesidades del usuario.

Cama ortopédica eléctrica

Cuenta con dos motores, uno ejecuta el levantamiento o el rebaje del respaldo y el otro ejecuta el levantamiento o el rebaje de las piernas:

Ilustración 16: Esquema cama ortopédica “Linak”

Ambos motores tienen una alimentación en común a la línea eléctrica de la residencia, y se controlan a través de 4 relés: uno para energizar el levantamiento del respaldo, otro para bajar el mismo, un tercero para el levantamiento de las piernas y el último para bajarlas.

Luego se definieron las posibles posiciones que se podrían adoptar tanto con el respaldo como para las piernas:

- Posición 0 → Corresponde a un 0% de inclinación al piso.
- Posición 1 → Corresponde a un 25% de inclinación al piso.
- Posición 2 → Corresponde a un 50% de inclinación al piso.
- Posición 3 → Corresponde a un 75% de inclinación al piso.
- Posición 4 → Corresponde a un 100% de inclinación al piso.

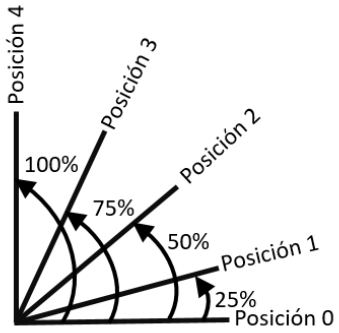


Ilustración 17: Posiciones que pueden adoptar tanto el respaldo como las piernas

Las mismas estarían basadas en la fracción de tiempo que uno de los cuatro relés es estimulado por el dispositivo domótico, generado la traslación hacia la posición deseada. Se procedió a cronometrar el tiempo que les llevaba, tanto al respaldo como a las piernas, ir desde una posición de horizontalidad total hasta la máxima verticalidad; esto es, ir desde el 0% hasta el 100% de inclinación.

Los resultados fueron:

- 16 segundos para el respaldo.
- 13 segundos para las piernas.

Entonces, a modo de ejemplo, para ir con el respaldo desde la posición 0 a la posición 2, se debería activar el relé de subir el respaldo durante 8 segundos (el 50% de 16 segundos). Luego para ir desde la posición 2 a la posición 1, el relé activado debería ser el de bajar el respaldo durante 4 segundos (el 25% de 16 segundos).

De manera adicional, dado que entre cada una de las posiciones definidas hay un paso de un 25%, resulta interesante brindarle la posibilidad al usuario de alcanzar posiciones intermedias, por lo tanto, se agregarían las siguientes órdenes:

- Subir → Corresponde a subir un 5%.
- Bajar → Corresponde a bajar un 5%.

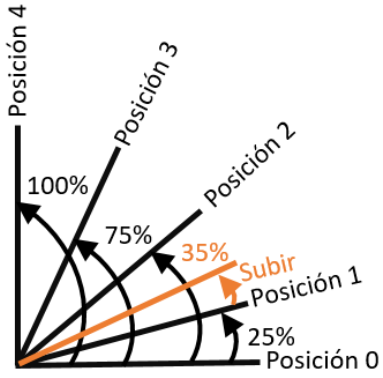


Ilustración 18: Posiciones intermedias

Por lo tanto, la manera de llegar a una cifra intermedia, como por ejemplo a un 35%, sería ir a la posición 2 y luego subir dos veces.

Elevador de personas con arnés

El elevador de personas con arnés cuenta con dos tipos de movimientos: uno horizontal o en el eje "x" y un movimiento vertical o en el eje "y". El desplazamiento en el eje "x" va a ser a través de un riel montado sobre el techo, cuyo cable (que transporte el arnés) va a ser tirado o empujado a través de una polea accionada por un motor. El recorrido del arnés en el eje "y" consiste en un segundo motor con otro sistema de polea que sube y baja el arnés.

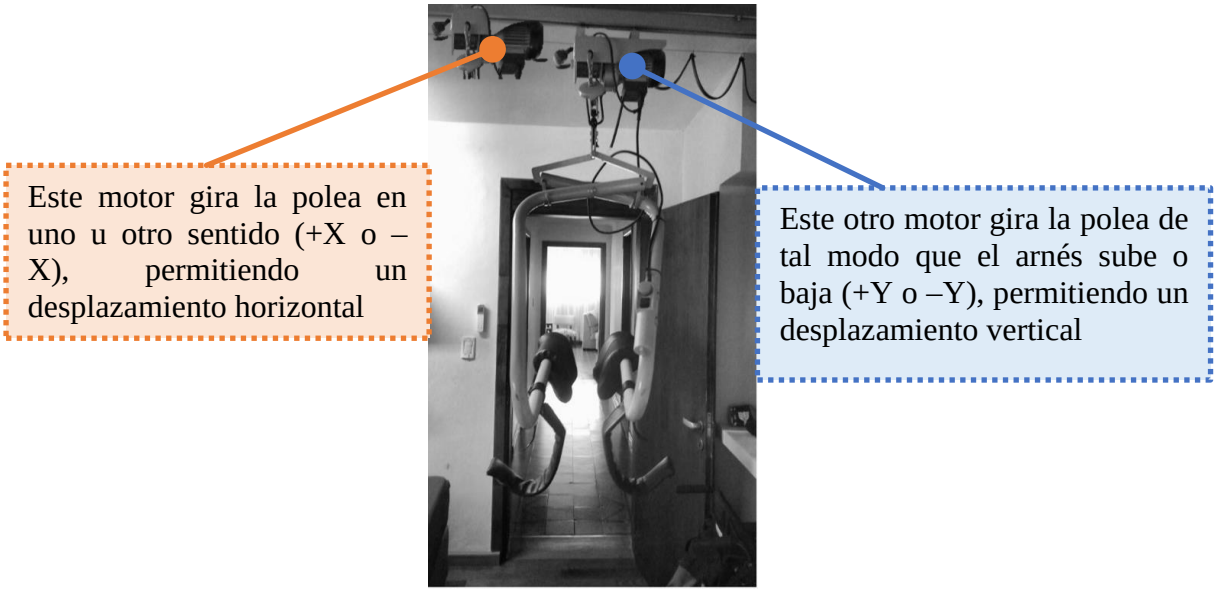


Ilustración 19: Desplazamientos del arnés

Como se explicó anteriormente, el uso que se le da al arnés es para la transferencia desde la silla de ruedas a la camilla ortopédica y viceversa. Por lo tanto, se determinó que había 3 ubicaciones útiles para el usuario:

- Una posición de guardado o de repliegue – **Posición 0**.
- Una posición al lado de la cama y a la altura de la silla de ruedas – **Posición 2**: para realizar la transferencia desde la silla de ruedas hacia la cama.
- Una posición sobre la cama y a la altura de la misma – **Posición 4**: para realizar la transferencia desde la cama hacia la silla de ruedas.

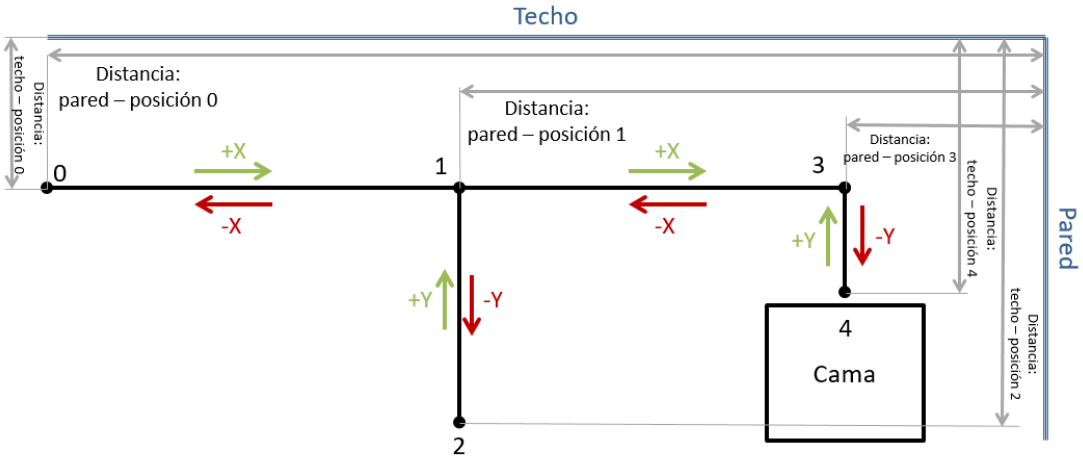


Ilustración 20: Posiciones arnés

Las posiciones 1 y 3 son intermedias, ya que los desplazamientos son unilaterales; esto es que el arnés puede moverse en el eje horizontal o en el eje vertical, pero nunca en los dos ejes a la vez. Por lo tanto, existe la necesidad de crear estas dos ubicaciones, no obstante, se supone el arnés nunca se detendría en ellas, solo las alcanzaría de manera temporaria para llegar a la ubicación de interés. Por ejemplo, si el sistema se encontrara en la posición 0 y necesitaría llegar hasta la posición 2, el desplazamiento sería un avance en el sentido horizontal hasta llegar a la posición 1, para luego comenzar a bajar hasta llegar a la posición 2.

Materiales y métodos

En la implementación del proyecto se utilizaron elementos del tipo hardware y software, con la intención de que el sistema incorpore señales verbales y las traduzca a la activación de un motor determinado, el cual ejecutaría el movimiento requerido en los objetos a domotizar.

Hardware

De la solución propuesta se desprenden ciertas características que definen el hardware utilizado. Se busca que el sistema:

1. Incorpore la señal de entrada para activar el nodo maestro de forma verbal, que se transmita en forma oral.

Supone el planteo de un sistema que "escuche" todo el tiempo; se pretende que el mismo esté en un estado tipo monitoreo, stand by o consumo mínimo, y se active ante determinada orden en particular: una contraseña. Una vez pasada esta instancia, la idea es que empiece la verdadera interacción con el sistema, que el mismo asimile las órdenes que le imparte el usuario, generando la respuesta esperada.

Luego de una exploración por todas las posibilidades, se decidió por el sistema EasyVR. Este módulo multipropósito es capaz de reconocer el habla ("VR" → "Voice Recognition"), y está diseñado para incorporar versatilidad, robustez y bajo costo a cualquier aplicación que requiera este tipo de entrada de señal:

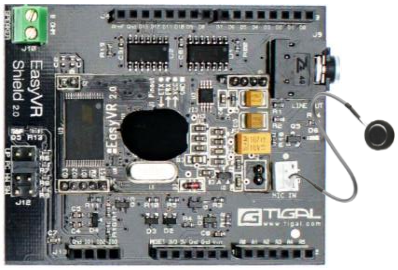


Ilustración 21: Shield EasyVR 2.0

Este módulo se utiliza como Shield del microcontrolador Arduino UNO:

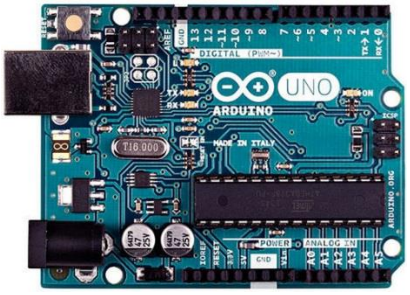


Ilustración 22: Microcontrolador Arduino UNO

La metodología de programación del módulo se podría enumerar de la siguiente forma:

I. Fase de estudio: Se definen las posibles órdenes que puede recibir el dispositivo. Las mismas deben adaptarse a las necesidades detectadas por el usuario del sistema domótico. Cabe recordar, que es necesario que el sistema se mantenga en un estado de "stand by" hasta recibir un estímulo en particular o contraseña ("X0").

   A continuación, se ilustran las órdenes utilizadas en este proyecto, donde se muestra el flujo secuencial que seguiría el usuario para llegar a la posición deseada:

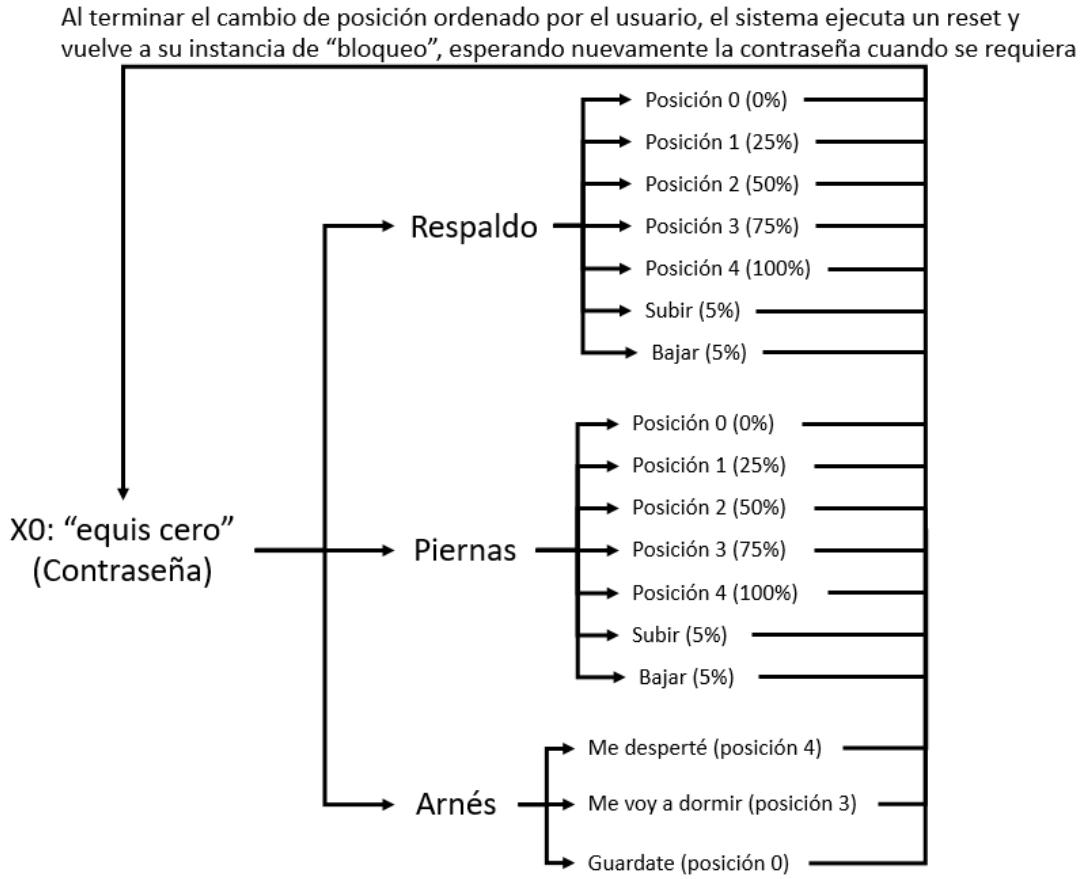


Ilustración 23: Definición de órdenes

II. <u>Fase de entrenamiento:</u> Se procede a grabar las órdenes, determinadas en la fase de estudio, en la memoria que viene incorporada en el módulo EasyVR. A esta instancia se le dará el tratado correspondiente en el apartado “Método de introducción de órdenes”, en el capítulo “Software”. Una vez completada esta etapa, ya se debería de tener grabado en la memoria del Shield los clips de audio que utilizaría el sistema como patrón comparativo. Esta situación se puede esquematizar de la siguiente manera:

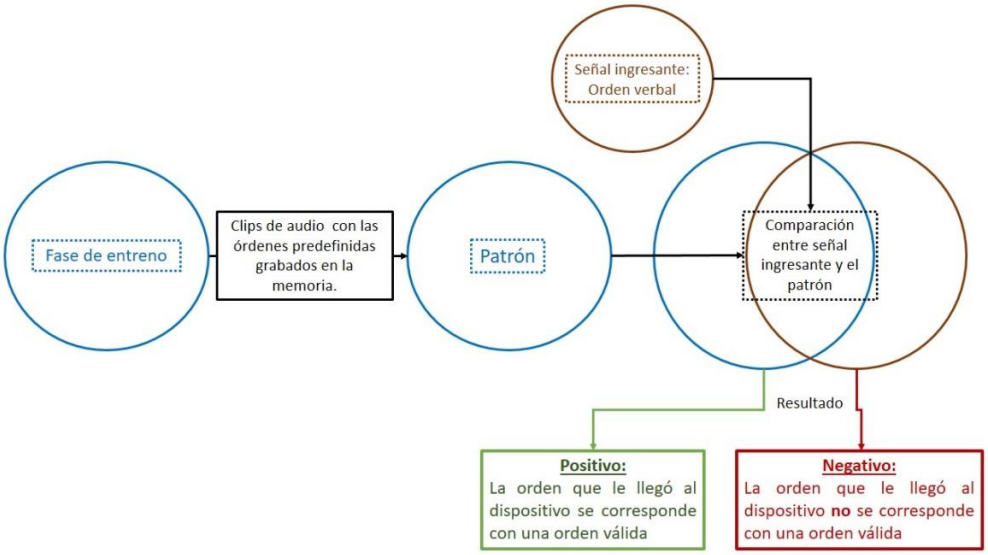


Ilustración 24: Esquema de programación del módulo EasyVR 2.0

2. <u>Esté distribuido en 3 nodos, donde la comunicación entre ellos debería ser inalámbrica, utilizando la tecnología de la radiofrecuencia.</u>

Incorporar una comunicación inalámbrica incurre no solo en un mejor aspecto estético y acabado final del proyecto, sino también en el ahorro de cables necesarios tanto para la energización como para la transmisión de datos entre los distintos dispositivos. La solución que se plantea tendría 3 módulos o nodos, cada con una función bien definida:

I. <u>Nodo maestro:</u> Interpreta la orden y genera el código correspondiente. Los otros dos nodos que reciban este mensaje deberían de ejecutar alguna acción según corresponda.
II. <u>Nodo camilla:</u> Denominado “Nodo 1” de forma arbitraria, recibe un mensaje codificado desde el nodo maestro que determina un cambio en la posición del respaldo o de las piernas de la cama ortopédica eléctrica.
III. <u>Nodo arnés:</u> Denominado “Nodo 2”, también de forma arbitraria, recibe un mensaje desde el nodo maestro que determina un cambio en la posición del arnés.

Para el envío y la recepción de los mensajes por radiofrecuencia se utilizó un módulo NRF24L01, compatible con el microcontrolador Arduino:

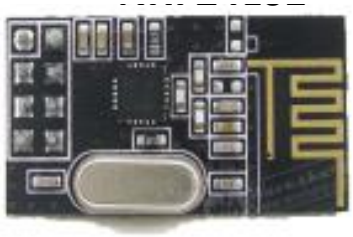


Ilustración 25: Módulo de comunicación por RF: NRF24L01

Su uso incluye una librería que facilita la implementación del mismo.

Resumiendo, la idea es que la persona que utilice el dispositivo lo haga de manera oral refiriéndose al nodo maestro, acto seguido éste generaría una orden codificada de forma digital y se la comunicaría por radiofrecuencia al nodo correspondiente. Una vez la orden sea recogida y decodificada, el nodo activaría uno de sus 4 relés dependiendo del movimiento que se solicita.

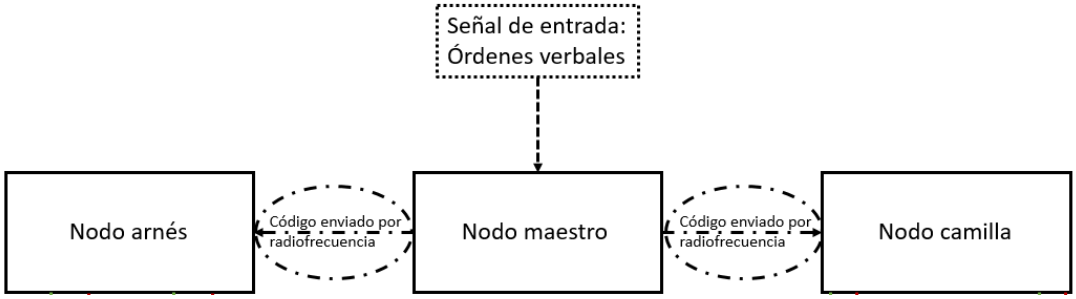


Ilustración 26: Esquema del funcionamiento inalámbrico sistema domótico

3. Incorpore salidas de ambos nodos que actúen sobre los objetos a domotizar (camilla y arnés). Las mismas serían 4 relés por nodo, cada uno activaría determinado movimiento.

Tal y como fue planteado, la intención de este proyecto es que el usuario maneje a su antojo las posiciones de los objetos a domotizar, por lo tanto, es necesario conectar los nodos que controlan el arnés y la cama ortopédica eléctrica a estos mismos elementos con el fin de interactuar con los motores que los desplazarían. Para esto fue necesario que los mismos tengan relés que activen los actuadores según la necesidad.

A continuación, se muestra una actualización de la Ilustración 26: Esquema del funcionamiento inalámbrico sistema domótico, incorporando los actuadores:

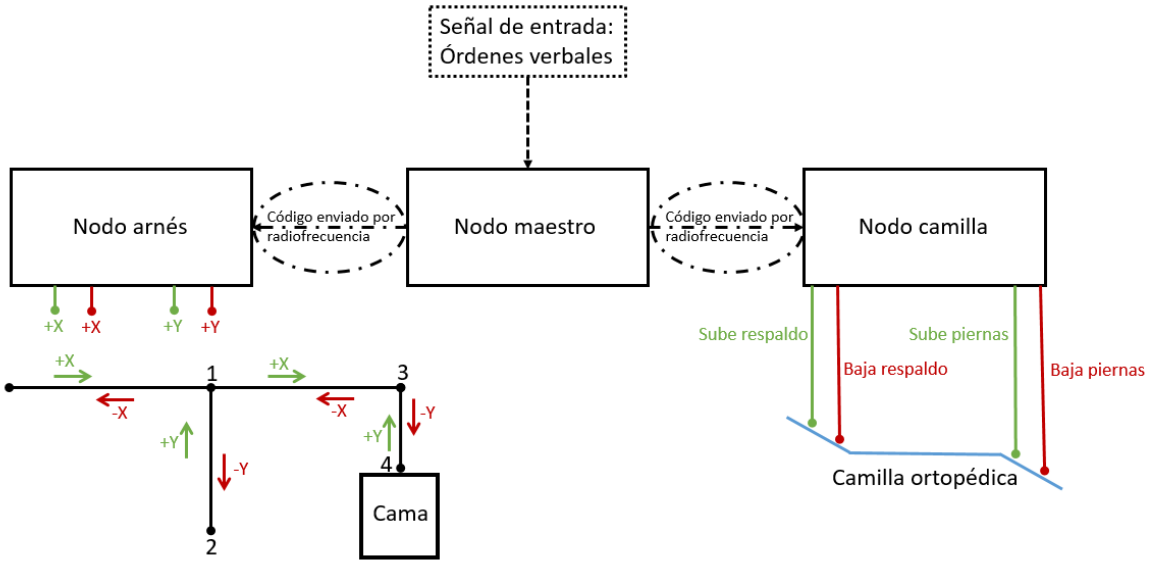


Ilustración 27: Esquema del funcionamiento inalámbrico sistema domótico con actuadores

Luego la descripción de cada relé o actuador se resume en la siguiente tabla:

Nodo	Relé		Descripción	
Arnés	Desplazamiento horizontal	+X	Activa el desplazamiento del arnés en el sentido horizontal de avance	→
		-X	Activa el desplazamiento del arnés en el sentido horizontal de retroceso	←
	Desplazamiento vertical	+Y	Activa el desplazamiento del arnés en el sentido vertical de subida	↑
		-Y	Activa el desplazamiento del arnés en el sentido vertical de bajada	↓
Camilla	Respaldo	Sube respaldo	Activa el desplazamiento de ascenso del respaldo de la camilla	↑
		Baja respaldo	Activa el desplazamiento de descenso del respaldo de la camilla	↓
	Piernas	Sube piernas	Activa el desplazamiento de ascenso de las piernas de la camilla	↑
		Baja piernas	Activa el desplazamiento de descenso de las piernas de la camilla	↓

Tabla 5: Descripción de los actuadores o relés del sistema domótico

Nodo maestro

A continuación, se muestra un esquema del conexionado del nodo maestro:

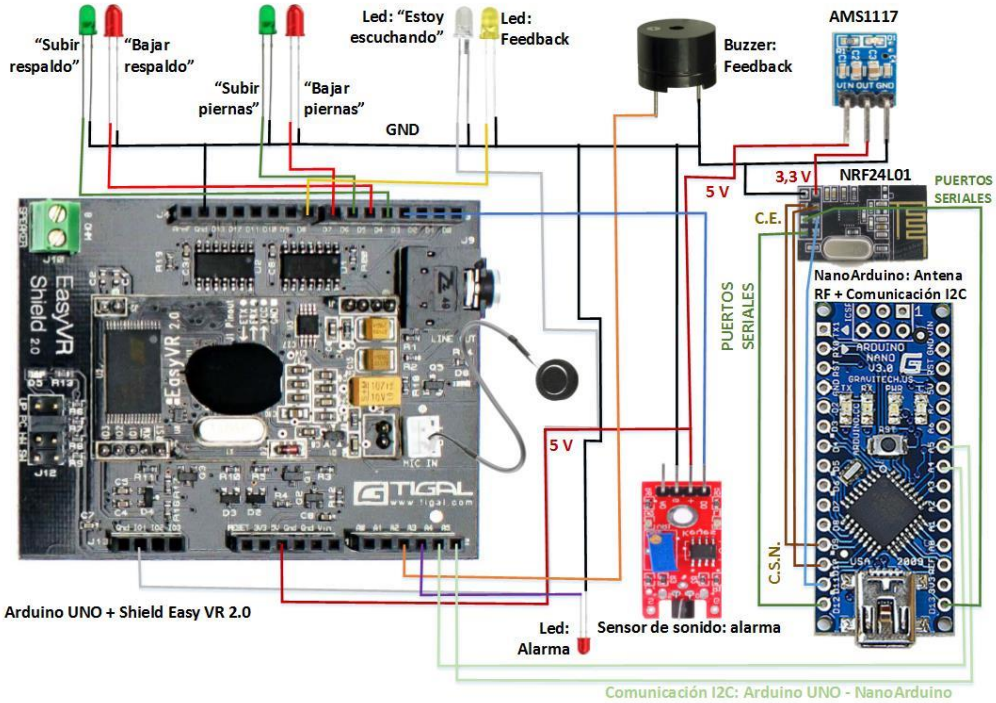


Ilustración 28: Conexionado nodo maestro

Luego, la siguiente tabla detalla la función de cada componente:

Componente	Cantidad	Función
EasyVR 2.0	1	Incorpora la señal. Las órdenes pregrabadas en su memoria EEPROM ("fase de entrenamiento") son comparadas con las señales que ingresan, determinando si las mismas se corresponden o no.
Arduino UNO	1	Genera el código (en forma de array de datos) en función de los positivos adquiridos de las señales procesadas por el Shield EasyVR2.0. Una vez codificada la orden, la transmite al Arduino NANO.
Arduino NANO	1	Funciona de router o "Gateway". Recibe el mensaje ya desde el Arduino Uno y lo transmite por pulsos a través del módulo NRF24L01.
NRF24L01	1	Realiza la comunicación inalámbrica entre los nodos. El mensaje enviado es un paquete de datos (Array) generado en el Arduino UNO.
AMS1117	1	Alimenta el módulo de radio NRF24L01, ya que la salida de 3,3 voltios de Arduino no posee suficiente corriente para energizarlo correctamente.
KY-038	1	Es la alarma del sistema. Ante un sonido de determinado umbral (un grito), revierte el desplazamiento que se estaba produciendo. *Por ejemplo: si se estaba subiendo el respaldo, ante una detección por parte del módulo de un sonido por encima del umbral programado, el desplazamiento se detiene y se comienza a bajar el respaldo durante 2 segundos.*
Leds	7	Feedback visual.
Buzzer	1	Feedback auditivo.

Tabla 6: Nodo maestro: componentes y funciones

Nodo cama ortopédica eléctrica

A continuación, se expone un esquema del conexionado del nodo camilla:

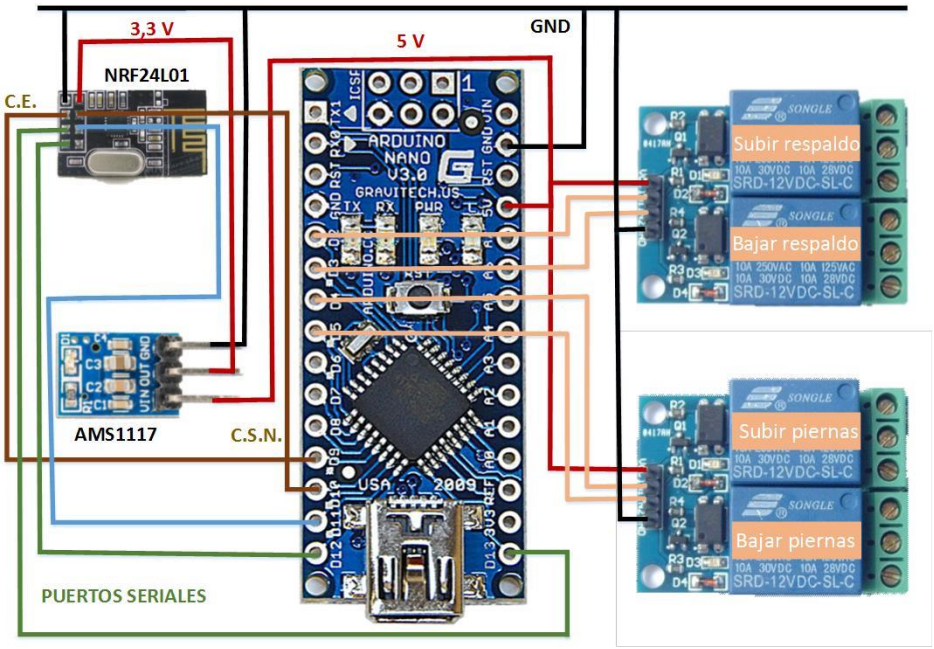


Ilustración 29: Conexionado nodo camilla

Luego, la siguiente tabla detalla la función de cada componente:

Componente	Cantidad	Función
Arduino NANO	1	Está constantemente esperando que llegue un mensaje a través del módulo NRF24L01. Cuando llega, lo decodifica y energiza uno de los 4 relés durante el tiempo calculado en el nodo maestro (“delay”).
NRF24L01	1	Realiza la comunicación inalámbrica entre los nodos. Se encuentra continuamente “leyendo” la red.
AMS1117	1	Alimenta el módulo de radio NRF24L01, ya que la salida de 3,3 voltios de Arduino no posee suficiente corriente para energizarlo correctamente.
Relés	2	Ejecutan el desplazamiento necesario. Para ello, cada uno de ellos está conectado a la camilla de manera tal que energiza los motores según se quiera: ○ Subir el respaldo. ○ Bajar el respaldo. ○ Subir las piernas. ○ Bajar las piernas.

Tabla 7: Nodo camilla: componentes y funciones

Nodo arnés

A continuación, un esquema del conexionado del nodo arnés:

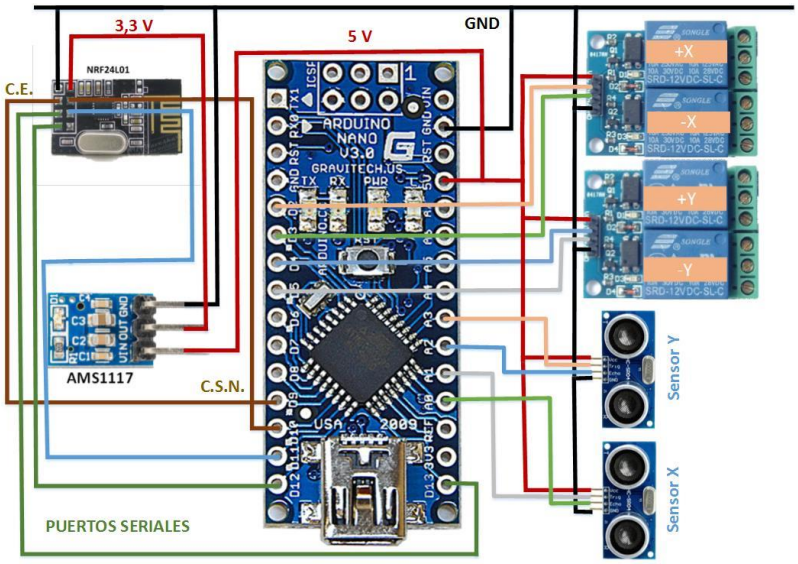


Ilustración 30: Conexionado nodo arnés

Luego, la siguiente tabla detalla la función de cada componente:

Componente	Cantidad	Función
Arduino NANO	1	Está constantemente esperando que llegue un mensaje a través del módulo NRF24L01. Cuando llega, lo decodifica y energiza uno de los 4 relés hasta que el sensor correspondiente detecte que el arnés llegó a la posición requerida por el usuario.
NRF24L01	1	Realiza la comunicación inalámbrica entre los nodos. Se encuentra continuamente "leyendo" la red.
AMS1117	1	Alimenta el módulo de radio NRF24L01, ya que la salida de 3,3 voltios de Arduino no posee suficiente corriente para energizarlo correctamente.
Relés	2	Ejecutan el desplazamiento necesario. Para ello, cada uno de ellos está conectado a los motores que desplazan el arnés de manera tal que pueda generarse un: ○ Desplazamiento horizontal de avance o "+X" ○ Desplazamiento horizontal de retroceso o "-X" ○ Desplazamiento vertical de subida o "+Y" ○ Desplazamiento vertical de bajada o "+-Y"
HC-SR04	2	Sensan la posición del arnés según corresponda dependiendo del desplazamiento realizado: ○ Sensor X: Mide la distancia que existe entre el arnés y la pared, determinando cuando el arnés llegó a la posición deseada en el desplazamiento horizontal. ○ Sensor Y: Mide la distancia que existe entre el arnés y el techo, determinando cuando el arnés llegó a la posición deseada en el desplazamiento vertical. Ambos sensores cortan la alimentación del relé activado del nodo, deteniendo así el desplazamiento del arnés y logrando que éste se ubique en el sitio requerido.

Tabla 8: Nodo arnés: componentes y funciones.

Software

Método de introducción de órdenes

Como se mostró en la Ilustración 24: Esquema de programación del módulo EasyVR 2.0, este sistema domótico supone una fase de entrenamiento, en la cual el usuario debería por única vez verbalizar todas las órdenes de manera tal que se graben los patrones en la memoria. La grabación de los mismos se logra a través de una interfaz creada por los fabricantes del shield EasyVR 2.0, llamada EasyVR Commander.

Nodo maestro

Las siguientes librerías fueron utilizadas en la implementación del sistema domótico descripto:

```
#include "Arduino.h"
#include "Platform.h"
#include "SoftwareSerial.h"
#include "EEPROM.h"
#include "Wire.h"
#include "I2C_Anything.h"
```

Declaraciones de los pines del Arduino UNO, el mismo se condice con el esquema de conexionado Ilustración 28: Conexionado nodo maestro:

```
//comienza declaración de pines
int led_respaldo_sube=4;
int led_respaldo_baja=5;
int led_piernas_sube=6;
int led_piernas_baja=7;
int led_feedback=8;
int buzzer=A2;
int sensor_alarma=2;
int led_alarma=A3;
//termina declaración de pines
```

Luego se declaran las variables utilizadas a lo largo del programa:

```
//comienza declaracion de variables
int i=0; // contador de la alarma
int j=0; // para los for de Array del mensaje de envio
int contador_errores=0; // contador de errores. a los 5 errores de reconocimiento, se
llama al timeout
const int tiempo_respaldo=16000; //tiempo que toma desde el respaldo en ir de la
posicion 0 a la posicion 100
const int tiempo_piernas=13000; //tiempo que toma desde las piernas en ir de la
posicion 0 a la posicion 100
const int add_pos_respaldo=0; //lugar de la EEPROM en donde se guarda la posicion del
respaldo --> robustez a cortes de luz
const int add_pos_piernas=1; //lugar de la EEPROM en donde se guarda la posicion del
pie --> robustez a cortes de luz
const int add_pos_grua=2; //lugar de la EEPROM en donde se guarda la posicion de la
grua --> robustez a cortes de luz
const float escalon=5;  //aqui se define el valor del escalon para hacer el paso
(0<paso<25, ya que 0,1,2,3 y 4 representan del 0 al 100 con un paso de 25 por nivel)
volatile float posicion_respaldo=0;
volatile float posicion_piernas=0;
volatile float posicion_final_camilla=0;
volatile float posicion_final_grua=1;
```

```
  float distancia_camilla=0;
  float tiempo_delay=0;
  float distancia_grua=0;
  volatile int manejo_respaldo=0; //si esta en 0 manejo piernas. si esta en 1 manejo
respaldo.
  byte bit_enviado=0; //variable auxiliar que uso para enviar bit por bit por I2C del
Master al NanoAntena
  const byte nodo1_address = 1;
  //termina declaracion de variables
```

Las funciones utilizadas fueron declaradas de la siguiente forma:

```
  //comienza declaracion de funciones
  void ejecucion_respaldo();
  void ejecucion_piernas();
  void feedback_positivo(); //funcion de feedback que da a entender que Arduino entendio
la orden.
  void feedback_findeorden(); //funcion de feedback que da a entender que Arduino
termino de ejecutar la orden
  void action(); //funcion de accion
  //termina la declaracion de funciones
```

Debido a una limitación propia de la tecnología, fue necesario incorporar un microcontrolador Arduino NANO, el cual funcionaba de “Gateway”. La función era simplemente recibir un mensaje codificado desde el Arduino UNO (cerebro del nodo maestro) y retransmitirlo por radiofrecuencia hacia el nodo correspondiente. El paquete de datos codificados enviados desde el Arduino UNO hacia el Arduino NANO fue declarado de la siguiente forma:

```
  //comienza declaracion del paquete enviado por I2C, que luego el NanoArduino enviara a
los nodos por RF
  float msj[]={ 0, 0, 0 };
  //finaliza declaracion del paquete enviado por I2C, que luego el NanoArduino enviara a
los nodos por RF
```

Luego comienza el setup, función que se ejecuta una sola vez al energizarse el dispositivo, y de la cual se transcriben cinco fragmentos que merecen explicación ya que mejoran la comprensión del uso del sistema:

El primer fragmento corresponde a la declaración de grupos por los que el usuario “navegará” con el fin de concatenar una orden.

```
  enum Groups
  {
    GROUP_0  = 0,
    GROUP_1  = 1,
    GROUP_2  = 2,
    GROUP_3  = 3,
  };
  enum Group0
  {
    X0 = 0,
  };
  enum Group1
  {
    G1_RESPALDO = 0,
    G1_PIERNAS = 1,
    G1_GRUA = 2,
    G1_CALIBRATE = 3,
  };
  enum Group2
  {
```

```
    G2_POSICION_0 = 0,
    G2_POSICION_1 = 1,
    G2_POSICION_2 = 2,
    G2_POSICION_3 = 3,
    G2_POSICION_4 = 4,
    G2_SUBIR_5PORC = 5,
    G2_BAJAR_5PORC = 6,
  };
  enum Group3
  {
    G3_ME_VOY_A_DORMIR = 0,
    G3_ME_LEVANTE = 1,
    G3_GUARDATE = 2,
  };
```

Luego, la función “timeout estaría destinada a resetear el dispositivo una vez transcurrido cierto tiempo sin recibir una orden:

```
   if (easyvr.isTimeout())
      {
        if(group!=GROUP_0)  //Ya que si proviene de “GROUP_0” significa que el “timeout” se generó en la misma instancia de bloqueo, por lo tanto se descarta y se opta por no realizar ninguna acción
        {
            if(group==GROUP_1)
            {
              group=GROUP_0;
              digitalWrite(led_feedback,HIGH);    //inicio del feedback timeout. (2 parpadeos de led amarillo)
              delay(1000);
              digitalWrite(led_feedback,LOW);
              delay(250);
              digitalWrite(led_feedback,HIGH);
              delay(1000);
              digitalWrite(led_feedback,LOW);   //finalizacion del feedback timeout. (2 parpadeos de led amarillo)
            }
            else
            {
              group=GROUP_0;
              digitalWrite(buzzer,HIGH);   //inicio del feedback timeout. (2 BEEPS + 2 parpadeos de led amarillo)
              digitalWrite(led_feedback,HIGH);
              delay(50);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);
              delay(200);
              digitalWrite(buzzer,HIGH);
              digitalWrite(led_feedback,HIGH);
              delay(50);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);   //finalizacion del feedback timeout. (2 BEEPS + 2 parpadeos de led amarillo)
            }
        }
      Serial.println("Tiempo fuera, el dispositivo se va a resetear...");
      }
```

El tercer fragmento destacado es la función “error”. Esta instancia se utiliza cuando el dispositivo que realiza el reconocimiento de comandos por voz (Shield EasyVR, explicado en el apartado **“Hardware”**) no reconoce una orden. La intención es purgar dichas señales, no obstante resulta necesario brindarle al usuario más de una oportunidad a la hora de enviarle órdenes al dispositivo:

```
      int16_t err = easyvr.getError();
      if (err >= 0)
      {
        if(group!=GROUP_0)  // Si la señal que genera el error proviene desde el grupo
0, significa que el dispositivo está en la instancia de bloqueo, toda señal debe ser
desechada.
        {
          if(contador_errores<4)  // El contador de errores propone 5 intentos para el
usuario.
          {
            digitalWrite(led_feedback,HIGH);  //Inicio del feedback error. (3 parpadeos
de led amarillo)
            delay(200);
            digitalWrite(led_feedback,LOW);
            delay(50);
            digitalWrite(led_feedback,HIGH);
            delay(200);
            digitalWrite(led_feedback,LOW);
            delay(50);
            digitalWrite(led_feedback,HIGH);
            delay(200);
            digitalWrite(led_feedback,LOW);    //Finalizacion  del  feedback  error.  (3
parpadeos de led amarillo)
            contador_errores++;
          }
          else //reseteo cuando detecto 5 errores seguidos
          {
            if(group!=GROUP_1)
            {
              group=GROUP_0;
              digitalWrite(buzzer,HIGH);  //inicio del feedback de reseteo (es similar
al timeout porque es basicamente lo mismo). (2 BEEPS + 2 parpadeos de led amarillo)
              digitalWrite(led_feedback,HIGH);
              delay(100);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);
              delay(200);
              digitalWrite(buzzer,HIGH);
              digitalWrite(led_feedback,HIGH);
              delay(100);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);   //inicio  del  feedback  de  reseteo  (es
similar  al  timeout  porque  es  basicamente  lo  mismo).  (2  BEEPS  +  2  parpadeos  de  led
amarillo)
            }
           else
           {
              group=GROUP_0;
              digitalWrite(led_feedback,HIGH);   //inicio  del  feedback  de  reseteo  (es
similar  al  timeout  porque  es  basicamente  lo  mismo).  (2  BEEPS  +  2  parpadeos  de  led
amarillo)
              delay(500);
              digitalWrite(led_feedback,LOW);
              delay(200);
              digitalWrite(led_feedback,HIGH);
              delay(500);
              digitalWrite(led_feedback,LOW);   //inicio  del  feedback  de  reseteo  (es
similar  al  timeout  porque  es  basicamente  lo  mismo).  (2  BEEPS  +  2  parpadeos  de  led
amarillo)
           }
          }
        }
      Serial.print("Error ");
      Serial.println(err, HEX);
      }
```

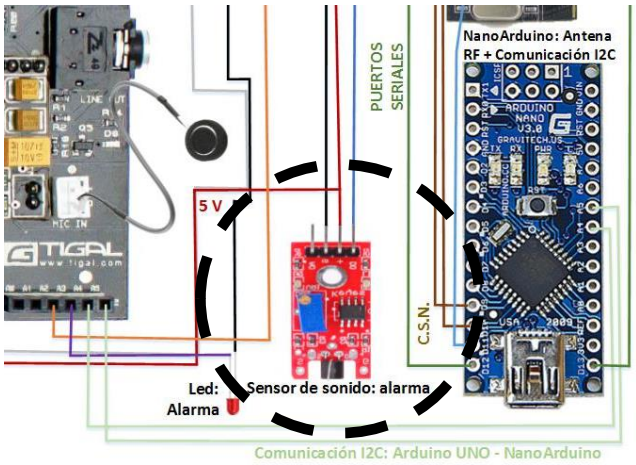
También resulta necesario que el proyecto tenga cierta robustez hacia los cortes eléctricos, ya que, de caso contrario, ante cada bajada de tensión que sea lo suficiente

como para resetear el micro, la referencia relativa que el mismo guardaba sobre el posicionamiento de los elementos que desplaza (cama ortopédica eléctrica y elevador de personas con arnés) se perdería. Para ello, se recurre a la memoria EEPROM integrada en el microcontrolador Arduino UNO:

```
        EEPROM.write(add_pos_respaldo,0); //escribo en la memoria EEPROM que todas las
variables de posicion estan en 0
        EEPROM.write(add_pos_piernas,0); //escribo en la memoria EEPROM que todas las
variables de posicion estan en 0
```

Por lo que en caso de que se corte el suministro eléctrico, la posición referencial no se perdería.

Finalmente, el dispositivo en funcionamiento supone el movimiento de ciertos elementos como, por ejemplo, un arnés elevador de personas. Por ende, por cuestiones de seguridad hacia el propio usuario del sistema, se necesita un sistema integrado de freno de emergencia. Por ser un sistema de emergencia, la metodología de entrada debería ser una distinta a la de ingreso general al sistema. Retomando la ilustración “Ilustración 28: Conexionado nodo maestro”:



El chip resaltado posee un micrófono (o método alternativo de ingreso al sistema), el cual capta todo el sonido externo. Esta señal de sonido pasa a través de un potenciómetro (perilla de umbral) y pone en alto su salida ante determinado valor de dB. Lo que se busca con este sistema, es que, ante un sonido por encima de determinado umbral, por ejemplo un grito, la placa Arduino UNO reciba un alto en uno de sus puertos. La lectura de este puerto supone una interrupción, que ejecuta la siguiente función:

```
   void alarma_respaldo()
   {
   digitalWrite(led_alarma,HIGH);

      if(distancia_camilla>0) //la alarma se activo cuando la camilla estaba subiendo
      {
         {
         digitalWrite(led_respaldo_sube,LOW); //tengo que dejar de subir
         digitalWrite(led_respaldo_baja,HIGH); //y tengo que comenzar a bajar
         for(i=0;i<120;i++)
         {                              // este for es para contar 2 segundos usando el
valor maximo que soporta la funcion
```

```
        delayMicroseconds(16383);     // delayMicroseconds. Para recalcular dividir la
cant de segundos que se quiere tener esta funciona activa
        }                            // y dividirla por 0,016383

        digitalWrite(led_respaldo_baja,LOW); //dejo de bajar
        }
     }

      if(distancia_camilla<0) // la alarma se activo cuando la camilla estaba bajando
      {
        {
        digitalWrite(led_respaldo_baja,LOW); //tengo que dejar de bajar
        digitalWrite(led_respaldo_sube,HIGH); //y tengo que comenzar a subir
        for(i=0;i<120;i++)
        {                              // este for es para contar 2 segundos usando el
valor maximo que soporta la funcion
        delayMicroseconds(16383);     // delayMicroseconds. Para recalcular dividir la
cant de segundos que se quiere tener esta funciona activa
        }                            // y dividirla por 0,016383

        digitalWrite(led_respaldo_sube,LOW); //dejo de subir
        }
     }
  digitalWrite(led_alarma,LOW);
  }
```

La cual detiene el movimiento que se estaba realizando al momento de la interrupción, y comienza a ejecutar el movimiento contrario durante 2 segundos. Por ejemplo, si el usuario estaba utilizando el dispositivo para bajar el respaldo, ante un sonido por encima del umbral programado, el respaldo se detendría y empezaría a subirse durante 2 segundos.

Finalmente, restaría detallar brevemente el código que compone el Arduino NANO utilizado como “Gateway” en el nodo maestro.

Se utilizaron las siguientes librerías:

```
#include "Wire.h"
#include "I2C_Anything.h"
#include "SPI.h"
#include "nRF24L01.h"
#include "RF24.h"
```

Este Gateway solo necesitaba de una variable, simplemente para contar el array que llegaba desde el Arduino UNO, para luego transmitirlo por radiofrecuencia hacia el nodo correspondiente. A continuación se muestra la forma de utilizada para declararla:

```
//comienza declaracion de variables
int i=0; //variable que se usa para el for que recorre el array
//finaliza declaracion de variables
```

Este microcontrolador tiene la función de conectarse a la “red” creada por radiofrecuencia para interactuar con los otros nodos, por lo tanto es necesario declarar una dirección para “escuchar la red” y otra dirección para “escribir en la red”:

```
//comienza declaracion de la instancia de radio
const byte slaveAddress[5] = {'R','x','A','A','A'}; //address del slave
const byte masterAddress[5] = {'T','X','a','a','a'}; //address del master
RF24 radio(9, 10);
```

```
//termina la declaracion de la instancia de radio
```

Luego es necesario definir el espacio de memoria dedicado al array (con el mensaje codificado) que llegaría desde el Arduino UNO y necesitaría ser enviado por radio a los demás nodos:

```
//comienza declaracion de los paquetes enviados y recibidos por RF
float msj[]={ 0, 0, 0 }; //mensaje que se recibe por I2C desde el master y este NanoArduino envia a los nodos por RF
char nodo2master[50]; //es el char de ACK que manda el nodo cuando lo recibio correctamente (por RF)
bool dato_nuevo = false; //se crea una instancia que hace de flag cuando se tiene un dato nuevo
//finaliza la declaracion de los paquetes enviados y recibidos por RF
```

Se declara una función para enviar y otra función para recibir paquetes de datos:

```
//comienza declaracion de funciones
void envio(); //funcion de envio de datos por RF
void recibo(); //funcion de recibo de datos por RF
//termina la declaracion de funciones
```

Del setup se destacan dos partes:

- La comunicación I2C mediante cableado con el Arduino UNO a través de los pines A4 y A5. (Ver Ilustración 28: Conexionado nodo maestro):

```
//Inicializacion del seteo de la comunicación I2C
Wire.begin(nodo1_adress);    // se une a la red I2C con el address asignado
Wire.onReceive(receiveEvent);    // interrupcion por recepcion de datos por I2C
Serial.begin(9600);
//finalizacion del seteo de la comunicación I2C
```

- El encendido de la radio para la comunicación inalámbrica con los otros nodos:

```
//inicializacion del seteo de la radio
radio.begin(); //se enciende la radio
Serial.println("Se encendio la radio. Rol del nodo: Master");
radio.setDataRate( RF24_250KBPS ); //velocidad de transferencia
radio.openWritingPipe(slaveAddress); //estructura que se usa para escribir mensajes
radio.openReadingPipe(1, masterAddress); //estructura que se usa para leer mensajes
radio.setRetries(3,5); // 3 = delay ; 5 = cuenta.
radio.startListening(); //enciendo la radio y empiezo a escuchar
//finalizacion del seteo de la radio
```

Luego el loop simplemente se encarga de constantemente “leer” tanto la red I2C como la red inalámbrica.

- Si llega un mensaje desde la red I2C, simplemente lo recibe y lo reenvía por radiofrecuencia.

```
if(DatoNuevo)  //se recibe un mensaje por I2C
{
  Serial.println(" ");
  Serial.println("Dato recibido por I2C desde el Master. Mensaje para el nodo: ");
  Serial.println(msj[0]);
  Serial.println("2da variable que se envia: ");
  Serial.println(msj[1]);
  Serial.println("3era variable que se envia: ");
```

```
    Serial.println(msj[2]);
    Serial.println(" ");

    envio();  //se llama a la función de envío por radiofrecuencia
    DatoNuevo = false;  //se baja el flag
  }
```

- Si en cambio llegase un mensaje desde la red de radio, la decodifica y la reenvía al Arduino UNO, quién se encarga de procesar la respuesta:

```
  if(radio.available())  //se recibe un mensaje por radiofrecuencia
  {
    radio.read( &nodo2master, sizeof(nodo2master));
    Serial.println(" ");
    Serial.println(nodo2master[50]);
    Serial.println(" ");
  }
```

Por último, para finalizar con la explicación de las partes más relevantes del código que compone el funcionamiento del nodo maestro, se expone a continuación la función de envío de mensajes por radiofrecuencia:

```
void envio()
{
  radio.stopListening(); //dejo de escuchar para enviar
  radio.write( &msj, sizeof(msj) ); //envio el array
  radio.startListening(); //vuelvo a escuchar
  Serial.println("Paquete RF Enviado: ");
  for(i=0; i<3; i++)
  {
    Serial.println(msj[i]);
  }
}
```

Nodo camilla

Este nodo está diseñado para recibir órdenes desde el nodo maestro a través de la red de radiofrecuencia y energizar uno de sus cuatro relés por determinada cantidad de tiempo. Las librerías empleadas son:

```
#include "SPI.h"
#include "nRF24L01.h"
#include "RF24.h"
```

Este microcontrolador tiene la función de conectarse a la "red" creada por radiofrecuencia para interactuar con los otros nodos, por lo tanto es necesario declarar una dirección para "escuchar la red" y otra dirección para "escribir en la red":

```
//comienza declaracion de la instancia de radio
const byte slaveAddress[5] = {'R','x','A','A','A'}; //address del slave
const byte masterAddress[5] = {'T','X','a','a','a'}; //address del master
RF24 radio(9, 10);
//termina la declaracion de la instancia de radio
```

Luego es necesario definir el espacio de memoria dedicado al array (con el mensaje codificado) que llegaría desde el nodo maestro por radiofrecuencia. El mensaje codificado se compone de tres valores, donde:

1. El primer valor corresponde a la dirección del nodo que se quiere controlar. Puede tomar los valores 1 o 2:

$$1 = Nodo\ camilla$$
$$2 = Nodo\ arnés$$

   Por lo tanto, para utilizar el nodo que maneja la camilla, este valor debería ser 1.

2. El segundo valor puede ser 1,2, 3 o 4 y corresponde a una codificación arbitraria de acciones, ya que resulta más liviano a nivel informático enviar un valor que enviar una serie de caracteres. La codificación se puede resumir con la siguiente tabla:

Acción a realizar	Codificación
Subir respaldo	1
Bajar respaldo	2
Subir piernas	3
Bajar piernas	4

Tabla 9: Codificación de acciones para el nodo camilla

   Por lo tanto, si el usuario deseara, por ejemplo, subir las piernas, este espacio tomaría valor 3.

3. El último valor del mensaje corresponde a la cantidad de tiempo que el relé debería estar activado, ya calculado previamente en el nodo maestro. Ese valor se introduce dentro de una función de delay.

Entonces, el espacio de memoria para recibir este array con 3 valores se definió de la siguiente manera:

```
//comienza declaracion de los paquetes enviados y recibidos por RF
float msj[]={ 0, 0, 0 }; //mensaje que se recibe por I2C desde el nodo maestro
char nodo2master[50]; //es el char de ACK que manda el nodo
bool dato_nuevo = false; //se crea una instancia que hace de flag cuando se tiene un dato nuevo
//finaliza la declaracion de los paquetes enviados y recibidos por RF
```

Declaraciones de los pines del Arduino NANO, el mismo se condice con el esquema de conexionado Ilustración 29: Conexionado nodo camilla:

```
//comienza declaracion de pines
int subir_respaldo=2;
int bajar_respaldo=3;
int subir_piernas=4;
int bajar_piernas=5;
int j=0; //contador para leer el array recibido del master
//termina declaracion de pines
```

El funcionamiento de este nodo resulta sencillo y se podría resumir según el siguiente gráfico:

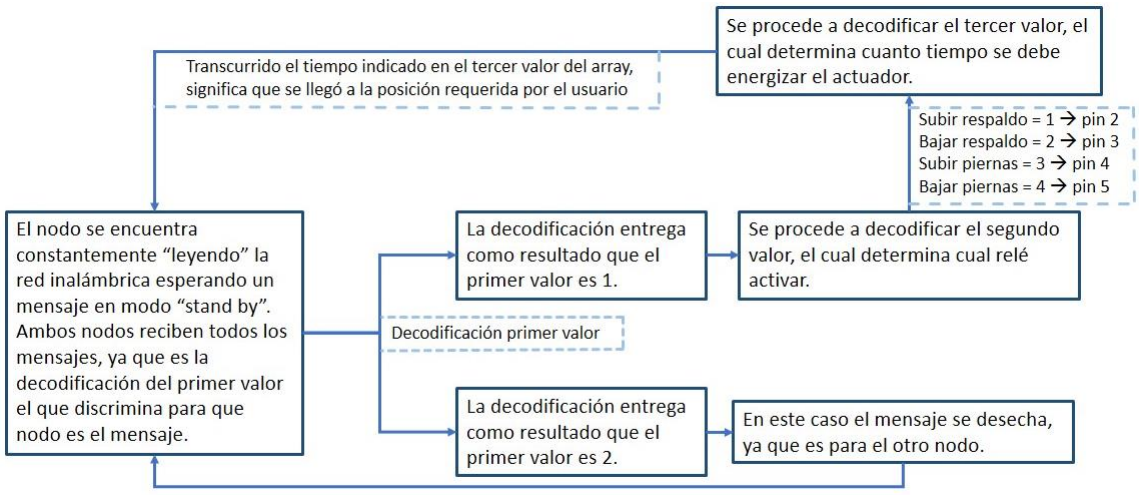


Ilustración 31: Funcionamiento del nodo camilla

A continuación se muestra la forma que se usó para la lectura de la red inalámbrica:

```
  if(radio.available()) //leo la radio para ver si hay un dato disponible
  {
    radio.read( &msj, sizeof(msj) ); //si lo hubiera, lo lee y lo guarda de la misma forma en la que llega (array) en el lugar de la memoria "&msj"
    if(msj[0]==1) //pregunta si el mensaje es para este nodo
    {dato_nuevo = true; //como le hablan a este nodo, activo el booleano que indica la entrada de un dato nuevo
    else if (msj[0]==2)
```

```
    {Serial.println("El mensaje es para el nodo arnés")//no hacer nada ya que no le
hablan a este nodo}  }
```

Nodo arnés

Este nodo está diseñado para recibir órdenes desde el nodo maestro a través de la red de radiofrecuencia y energizar uno de sus cuatro relés hasta que el sensor de posición (sensor ultrasónico x o sensor ultrasónico y, Ilustración 30: Conexionado nodo arnés) detecte la distancia programada. Las librerías empleadas son:

```
#include "SPI.h"
#include "nRF24L01.h"
#include "RF24.h"
#include "Ultrasonido.h"
```

Este microcontrolador tiene la función de conectarse a la "red" creada por radiofrecuencia para interactuar con los otros nodos, por lo tanto es necesario declarar una dirección para "escuchar la red" y otra dirección para "escribir en la red":

```
//comienza declaracion de la instancia de radio
const byte slaveAddress[5] = {'R','x','A','A','A'}; //address del slave
const byte masterAddress[5] = {'T','X','a','a','a'}; //address del master
RF24 radio(9, 10);
//termina la declaracion de la instancia de radio
```

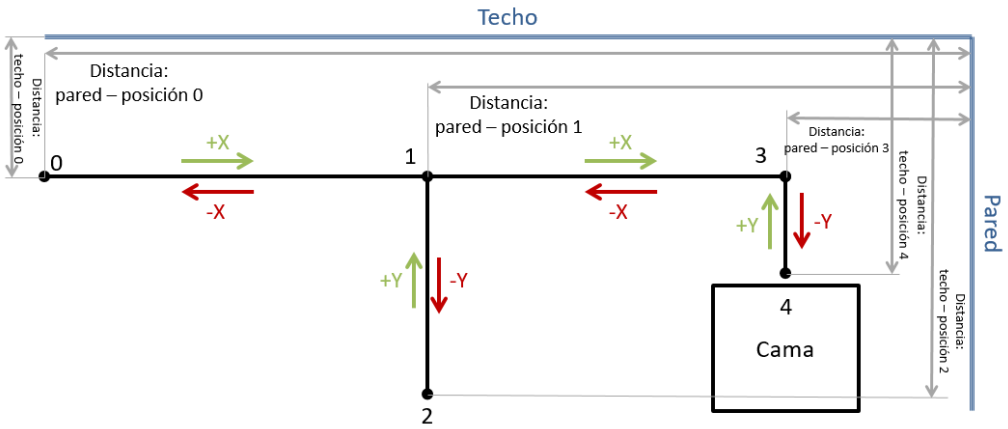
Luego es necesario definir el espacio de memoria dedicado al array (con el mensaje codificado) que llegaría desde el nodo maestro por radiofrecuencia. El mensaje codificado se compone de tres valores, donde:

1. El primer valor corresponde a la dirección del nodo que se quiere controlar. Puede tomar los valores 1 o 2:

$$1 = Nodo\ camilla$$
$$2 = Nodo\ arnés$$

   Por lo tanto, para utilizar el nodo que maneja la camilla, este valor debería ser 2.

2. El segundo valor corresponde a la ubicación actual del arnés. Vale la pena recordar el gráfico "Ilustración 20: Posiciones arnés":



Por lo tanto, este espacio del mensaje codificado debería de ser uno de los siguientes valores numéricos: 0, 2 o 4, que representaría la ubicación actual del arnés como se planteó en un principio.

3. De manera similar que el segundo carácter del mensaje descripto arriba, el tercero corresponde a la posición final que debería de adoptar el arnés luego de realizar la acción requerida por el usuario. Dicho de otra forma, es la posición a la que el usuario desea "enviar" el arnés.

A modo de ejemplo, supongamos que el arnés se encuentra en la posición de guardado 0 y se requiera el arnés se desplace hacia la posición 2. Luego de decodificar la orden, el arnés realizaría un desplazamiento horizontal en sentido de avance (+X) hasta la posición 1:

La posición sería determinada por los sensores ultrasónicos (ver Ilustración 30: Conexionado nodo arnés), los cuales cortarían la alimentación de los relés según corresponda; o sea, hasta que se haya alcanzado la posición requerida. Siguiendo con el ejemplo, el siguiente desplazamiento del arnés sería en sentido vertical en bajada (-Y) hasta la posición 2:

Entonces, el espacio de memoria para recibir este array con 3 valores se definió de la siguiente manera:

```
  //comienza declaracion de los paquetes enviados y recibidos por RF
  float msj[]={ 0, 0, 0 }; //mensaje que se recibe por I2C desde el nodo maestro
  char nodo2master[50]; //es el char de ACK que manda el nodo
  bool dato_nuevo = false; //se crea una instancia que hace de flag cuando se tiene un dato nuevo
  //finaliza la declaracion de los paquetes enviados y recibidos por RF
```

La declaración de variables utilizadas:

```
  //comienza declaracion de variables
  int j=0; //contador para leer el array recibido del master
  int sensor=0; //0 signfica el sensor X; 1 significa el sensor Y
  float distancia; //variable en donde se va almacenando las distancias sensadas por ultrasonido
  int dist_min=0; //variable que representa el valor minimo del intervalo del sensado
  int dist_max=0;; //variable que representa el valor maximo del intervalo del sensado
  float pos_x=0; //posicion x actual de la grua
  float pos_y=0; //posicion y actual de la grua
  //termina declaracion de variables
```

Las funciones utilizadas en este nodo se declararon de la siguiente forma:

```
//comienza declaracion de funciones
void accion();
//termina declaracion de funciones
```

El funcionamiento de este nodo se podría sintetizar de la siguiente manera:

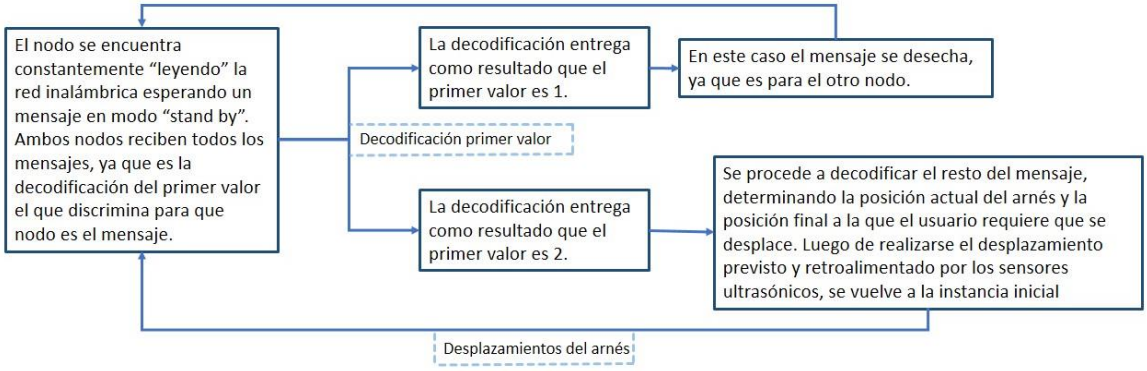


Ilustración 32: Funcionamiento del nodo arnés

Luego la siguiente tabla resume los posibles desplazamientos según las posiciones iniciales y finales recibidas en el mensaje codificado:

Posición inicial	Posición final	Desplazamiento		
0	2	0 (+X) 1	1 (-Y) 2	
0	4	0 (+X) 3	3 (-Y) 4	
2	0	2 (+Y) 1	1 (-X) 0	
2	4	2 (+Y) 1	1 (+X) 3	3 (-Y) 4
4	0	4 (+Y) 3	3 (-Y) 0	
4	2	4 (+Y) 3	3 (-X) 1	1 (-Y) 2

Tabla 10: Desplazamientos del arnés según las posiciones recibidas

Para tener mayor control y comodidad a la hora de programar los desplazamientos, se confeccionó una tabla donde se indexarían las distintas distancias hacia la pared según el sensor X o hacia el techo según el sensor Y.

```
//comienza declaracion de intervalos que representan cada posicion
//posicion 0: X: distancia a la pared 100 cm; Y: distancia al techo 10 cm
int x_min_1=90;
int x_max_1=110;
int y_min_1=5;
int y_max_1=15;
//posicion 1: X: distancia a la pared 50 cm; Y: distancia al techo 10 cm
int x_min_2=40;
int x_max_2=60;
int y_min_2=5;
int y_max_2=15;
//posicion 2: X: distancia a la pared 50 cm; Y: distancia al techo 100 cm
```

```
int x_min_3=40;
int x_max_3=60;
int y_min_3=90;
int y_max_3=110;
//posicion 3: X: distancia a la pared 20 cm; Y: distancia al techo 10 cm
int x_min_4=15;
int x_max_4=25;
int y_min_4=5;
int y_max_4=15;
//posicion 4: X: distancia a la pared 20 cm; Y: distancia al techo 50 cm
int x_min_5=15;
int x_max_5=25;
int y_min_5=40;
int y_max_5=60;
//termina declaracion de intervalos que representan cada posicion
```

También resultaba necesario declarar los pines utilizados por los sensores ultrasónicos:

```
//comienza seteo sensores ultrasonicos (pin Trigger, pin Echo)
Ultrasonido SensorY=Ultrasonido(A1,A0);
Ultrasonido SensorX=Ultrasonido(A3,A2);
//termina seteo sensores ultrasonicos (pin Trigger, pin Echo)
```

A continuación se muestra la forma que se usó para la lectura de la red inalámbrica:

```
  if(radio.available()) //leo la radio para ver si hay un dato disponible
  {radio.read( &msj, sizeof(msj) ); //si lo hubiera, lo lee y lo guarda de la misma
forma en la que llega (array) en el lugar de la memoria "&msj"
    if(msj[0]==2) //pregunta si el mensaje es para este nodo
    {dato_nuevo = true; //como le hablan a este nodo, activo el booleano que indica la
entrada de un dato nuevo
    else if (msj[0]==1)
    {Serial.println("El mensaje es para el nodo camilla")//no hacer nada ya que no le
hablan a este nodo}
```

La forma de decodificación resulta similar a la del nodo camilla. No obstante, resulta interesante explayar la función de acción, la cual activa y desactiva el sistema actuador del sistema domótico según se requiera por el usuario. A la misma ingresan 4 variables:

1. Relé que se activa: +X, -X, +Y o –Y.
2. Sensor que se activa: Sensor X para desplazamientos horizontales y Sensor Y para desplazamientos verticales.
3. Posición inicial del arnés (enviado desde el maestro).
4. Posición final del arnés (enviado desde el maestro).

```
void accion(int rele, int sensor,int dist_min, int dist_max)
{
  digitalWrite(rele,HIGH);  //comienza el desplazamiento
  Serial.println(" ");
  Serial.println("Comienza el desplazamiento, el rele activado es: ");
  Serial.println(rele);
  Serial.println("Y el desplazamiento que se va a realizar va a ser el siguiente: ");
  Serial.println("Parte desde la posicion: ");
  Serial.println(msj[1]);
  Serial.println("Y se dirige hacia la posicion: ");
  Serial.println(msj[2]);
  Serial.println(" ");
  for(int i=0 ; i<5000 ; i++) //se cuenta por determinada cantidad de tiempo (esta
variable debe sobre estimarse) --> i=(valor/4)/60 [minutos]
  {if(sensor==0) //debo leer el sensorX
    {
      distancia=SensorX.Distancia();
```

```
        pos_x=distancia; //si el desplazamiento fue en x, guardo la posicion alcanzada
como la posicion acutal de la grua respecto a la pared
        Serial.println(" ");
        Serial.println("Distancia medida: ");
        Serial.println(pos_x);
      }
      if(sensor==1) //debo leer el sensorY
      {
        distancia=SensorY.Distancia();
        pos_y=distancia; //si el desplazamiento fue en y, guardo la posicion alcanzada
como la posicion acutal de la grua respecto al techo
        Serial.println("Distancia medida: ");
        Serial.println(pos_y);
        Serial.println(" ");
      }
      if((dist_min<distancia) && (distancia<dist_max))
      {
        digitalWrite(rele,LOW);   //termina desplazamiento
        i=9999;
        Serial.println(" ");
        Serial.println("Posicion actual de la grua: ");
        Serial.print(pos_x);
        Serial.println(" cm de la pared");
        Serial.print(pos_y);
        Serial.println(" cm del techo");
        Serial.println(" ");
      }
      delay(250);
    }
```

Costos: Hardware

A continuación, se enumeran las compras realizadas para materializar este proyecto. Las mismas se agrupan en la siguiente tabla, detallando el producto, el costo y el proveedor:

Proveedor	Gasto total	Detalle
Nubbeo: tienda virtual	$151,00	• Pack 50 Leds 5mm Rojos Amarillos Verdes Azul Blanco • Modulo Alimentación 5v A 3v3 800ma Ams1117
MercadoLibre: particular	$79,99	• Sensor De Sonido Arduino Pic Córdoba
Mercadolibre: particular	$1200	• Procesador De Voz Arduino Easyvr
SynerGeek SRL	$210	• Arduino Nano Scable • Cable MiniUSB – USB x 2
SynerGeek SRL	$705	• Módulo NRF24L01 x 3 • Arduino Nano Scable x 2 • Modulo Relé (2 relés por módulo) x 4
SynerGeek SRL	$700	• Arduino Uno • ProtoBoard de 800 Puntos • Kit Cable Hembra/Hembra • Kit Cable Hembra/Macho • Kit Cable Macho/Macho • Leds 3 mm verdes • Leds 3 mm rojos • Leds 3 mm amarillos. • Leds 3 mm blancos.
SynerGeek SRL	$150	• Sensores Ultrasónicos x 2
Electrocomponentes	$580	• Fuente 9 V 1 A x 2 • Conector Jack hembra 3,5 stereo x 15 • Conector Jack macho 3,5 stereo x 15 • Conector DC x 2 • Cable x 3 metros
Electrocomponentes	$140	• Fuente 9 V 1 A
Impresión placas PCB	$900	• Impresión PCB + montaje + soldadura circuito x 3
Asia Cortes Laser	$350	• Acrílico + cortes láser x 3
Bulonera Centro	$35	• Tuercas x 100 • Tornillos x 12 • Varillas x 2
Ferretería Suiza	$52	• Poxipol transparente.
Total		**$5.252,99**

Tabla 11: Costos del proyecto

Costos: Software

Para estimar los costos en lo que refiere al software del dispositivo domótico, se acudió a la tabla de honorarios emitida por el Consejo Profesional de Ciencias Informáticas de la Provincia de Córdoba. A continuación, se muestra la versión más actualizada de la misma, resaltando el tipo de puesto utilizado para la estimación:

CARGO	POR MES		POR HORA	
	DESDE	HASTA	DESDE	HASTA
Administrador de ISP	$ 25,774	$ 48,239	$ 657	$ 1,119
Administrador de Redes	$ 25,774	$ 48,239	$ 778	$ 1,167
Analista Senior	$ 25,774	$ 48,239	$ 622	$ 1,167
Analista Junior	$ 19,330	$ 25,774	$ 462	$ 803
Analista Programador	$ 25,774	$ 45,273	$ 657	$ 1,119
Auditor Interno Informático	$ 24,607	$ 45,273	$ 657	$ 1,119
Consultor Informático	$ 0	$ 0	$ 778	$ 1,167
Data Base Administrator	$ 25,774	$ 48,239	$ 657	$ 1,119
Diseñador Gráfico Senior	$ 19,330	$ 29,080	$ 414	$ 657
Diseñador Gráfico Junior	$ 12,886	$ 19,330	$ 268	$ 487
Diseñador Industrial	$ 19,330	$ 32,216	$ 341	$ 657
Diseño de Páginas Web	$ 12,886	$ 25,774	$ 341	$ 657
Dibujante de Articulación PC	$ 9,750	$ 16,170	$ 268	$ 341
Forense	$ 41,820	$ 64,432	$ 826	$ 1,215
Especialista en Comercio Electrónico	$ 41,820	$ 64,432	$ 826	$ 1,215
Gerente de Sistemas PYMEs	$ 45,102	$ 70,706	$ 0	$ 0
Gerente de Sistemas de Grandes Empresas	$ 77,125	$ 121,765	$ 0	$ 0
Graboverificador	$ 0	$ 0	$ 243	$ 414
Jefe de Servicio Técnico	$ 16,170	$ 25,774	$ 0	$ 0
Jefe de Proyectos	$ 22,636	$ 41,820	$ 0	$ 0
Jefe de Gabinete Informático	$ 16,170	$ 25,774	$ 0	$ 0
Operador – Data Entry	$ 9,750	$ 16,169	$ 0	$ 0
Perito Informático	$ 0	$ 0	$ 778	$ 1,215
Perito Operador PC	$ 6,444	$ 11,744	$ 0	$ 0
Programador de Páginas Web	$ 19,330	$ 35,353	$ 0	$ 0
Programador Ambientes Windows	$ 22,490	$ 41,820	$ 657	$ 998
Programador Ambientes Unix/Linux	$ 25,774	$ 45,102	$ 778	$ 1,215
Profesor Informático	$ 9,750	$ 19,330	$ 268	$ 487
Instructor Informático	$ 0	$ 0	$ 414	$ 608
Técnico de Harware	$ 11,774	$ 19,330	$ 414	$ 608

Ilustración 33: Tabla de honorarios

Por lo tanto:

$$Honorarios = \$22.490 \times 6\ meses = \ \$134.940$$

Diseño

A la hora de materializar el sistema domótico propuesto se vio sometido a diferentes transformaciones que acompañaron la evolución del mismo, intentando que el producto final sea, dentro de las posibilidades que abarcan este proyecto, lo más eficiente, seguro y amigable con el usuario. Este proceso de transformaciones involucra básicamente dos etapas:

1. PCBs (Printed Circuit Board): Se confeccionaron de 3 placas de circuito impreso, donde cada una contenga toda la electrónica integrada que corresponde a cada nodo. Recordar que el dispositivo tiene 3 nodos que se comunican entre sí por radiofrecuencia, ver Ilustración 27: Esquema del funcionamiento inalámbrico sistema domótico con actuadores.
2. Carcasas: Se diseñaron 3 carcasas, una por nodo, teniendo en cuenta diferentes aspectos tales como la seguridad (tratándose de un dispositivo eléctrico), la estética y la comodidad en el uso. Este último aspecto resultó en un profundo análisis a la hora de la decisión de que información debería de transmitirle el sistema domótico al usuario, como respuesta a sus órdenes (feedback). Las mismas fueron diseñadas en SolidWorks para luego materializarlas en acrílico negro de 2,4 mm cortado a láser.

A continuación, se va a explayar sobre ciertos detalles relevantes que conciernen en los procesos brevemente recién descriptos:

Nodo maestro

Contiene la electrónica descripta en la Ilustración 28: Conexionado nodo maestro:

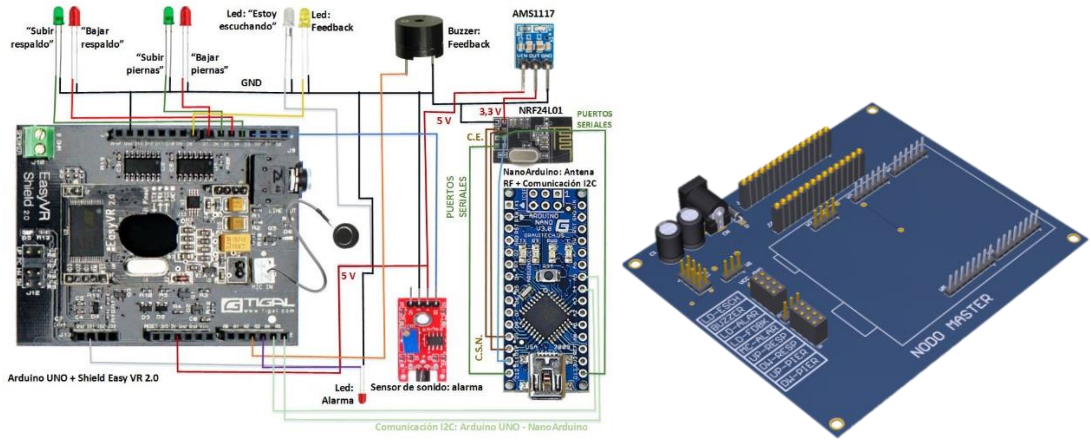


Ilustración 34: PCB nodo maestro: Vista general

Planos

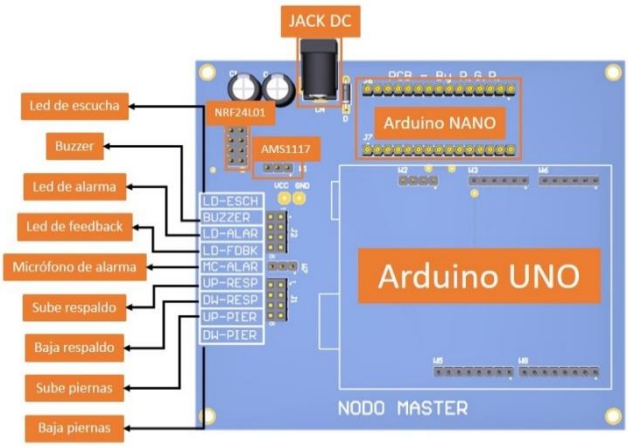


Ilustración 35: PCB nodo maestro: Vista inferior

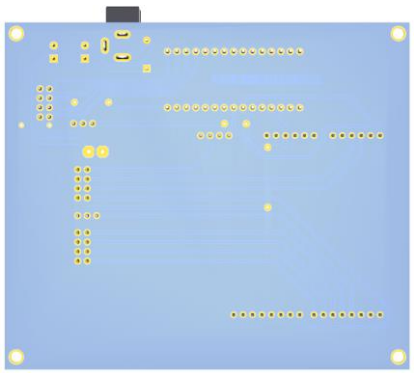
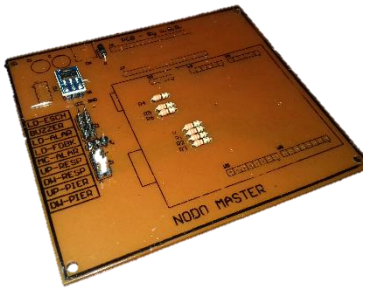
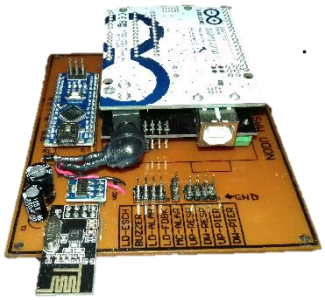
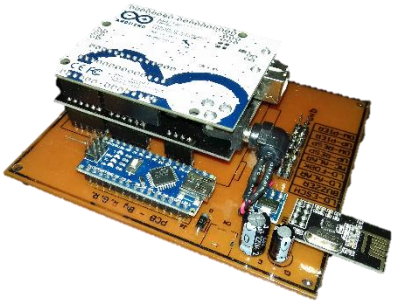


Ilustración 36: PCB nodo maestro: Vista superior

Resultado final:







Carcasa

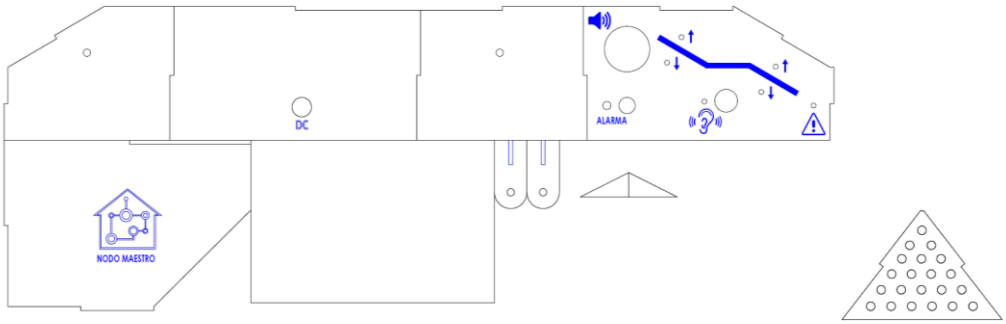


Ilustración 37: Carcasa nodo maestro: Cortes del acrílico

Nodo camilla

Contiene la electrónica descripta en la Ilustración 29: Conexionado nodo camilla:

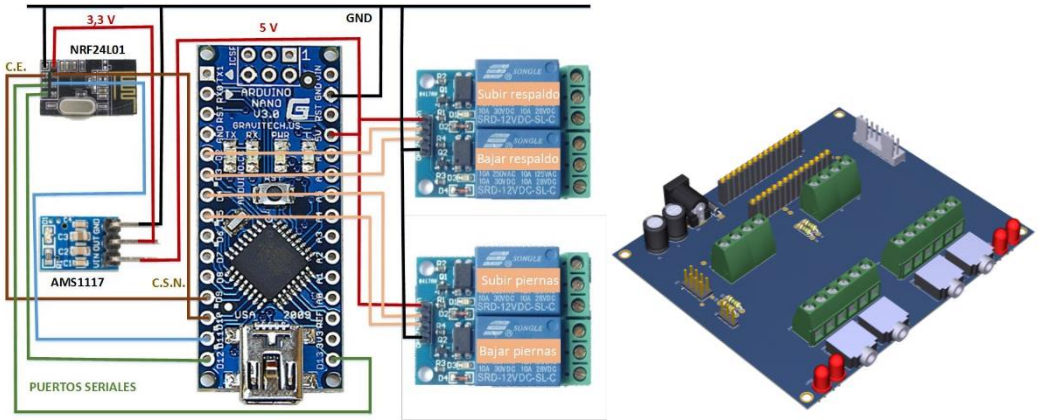


Ilustración 38: PCB nodo camilla: Vista general

Planos

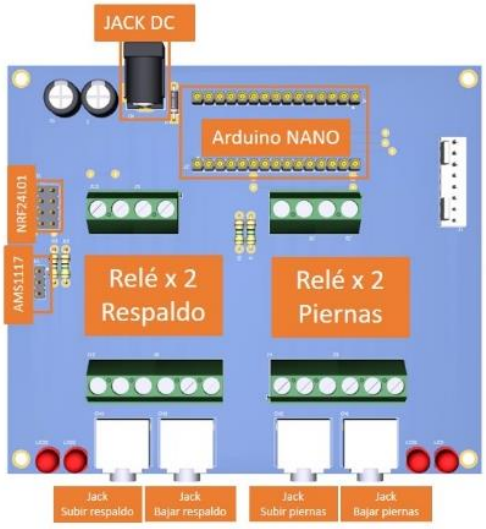


Ilustración 39: PCB nodo camilla: Vista superior

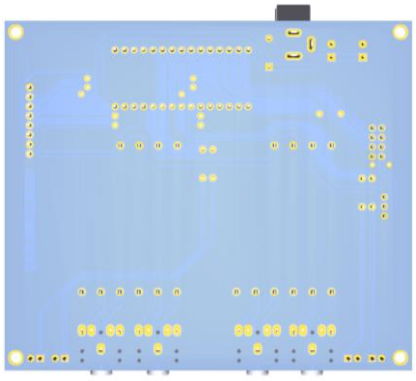


Ilustración 40: PCB nodo camilla: Vista inferior

Carcasa

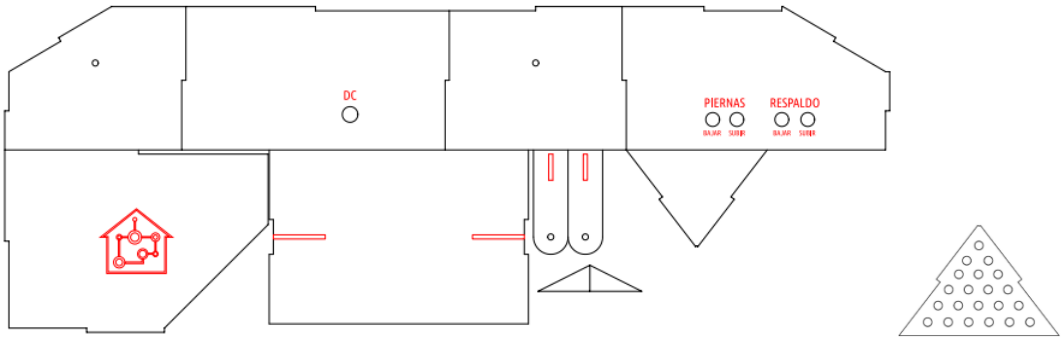
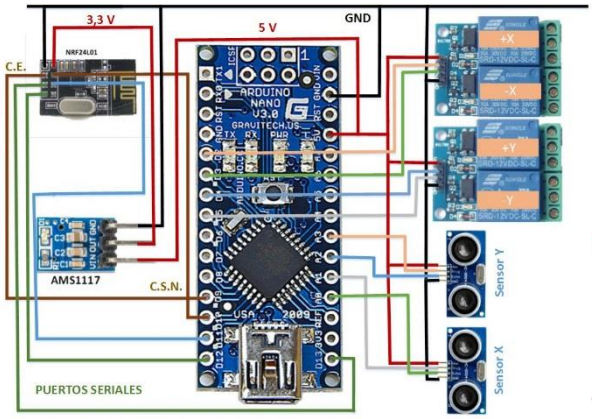


Ilustración 41: Carcasa nodo camilla: Cortes del acrílico

Nodo arnés

Contiene la electrónica descripta en la Ilustración 30: Conexionado nodo arnés:



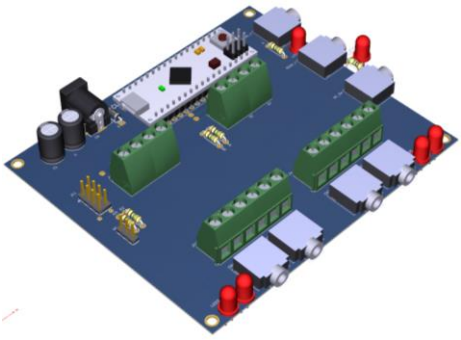
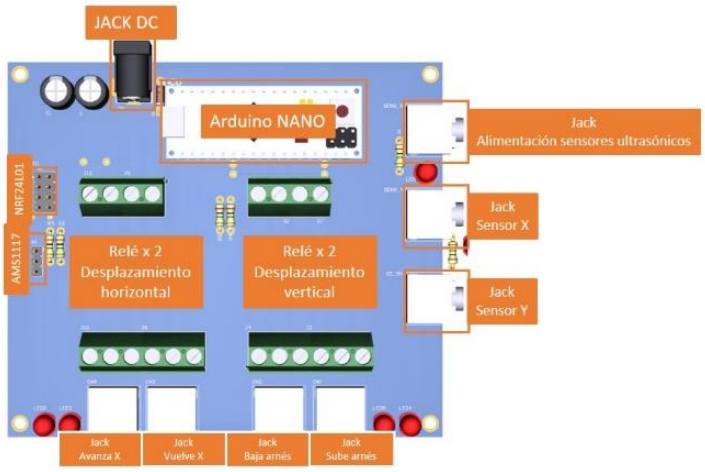


Ilustración 42: PCB nodo arnés: Vista general

Planos



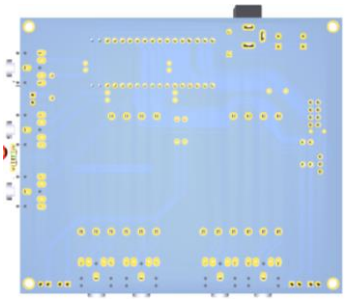


Ilustración 43: PCB nodo arnés: Vista inferior

Ilustración 44: PCB nodo arnés: Vista superior

Carcasa

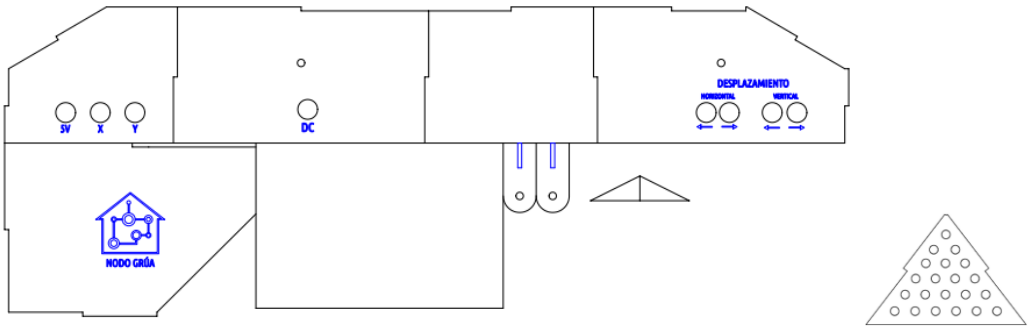


Ilustración 45: Carcasa nodo arnés: Cortes del acrílico

Aspecto final

Si bien a lo largo de la realización del proyecto se primó en la funcionalidad por sobre la estética, una vez resueltos los aspectos técnicos del producto, se decidió darle un tratamiento a la artística del mismo. Luego de realizar el ensamblaje de todos los cortes de acrílico, y de incorporar los circuitos impresos a cada uno de los nodos, el resultado final es satisfactorio desde el punto de vista estético, tal y como se muestra en Ilustración 46, Ilustración 47, Ilustración 48 y Ilustración 49:

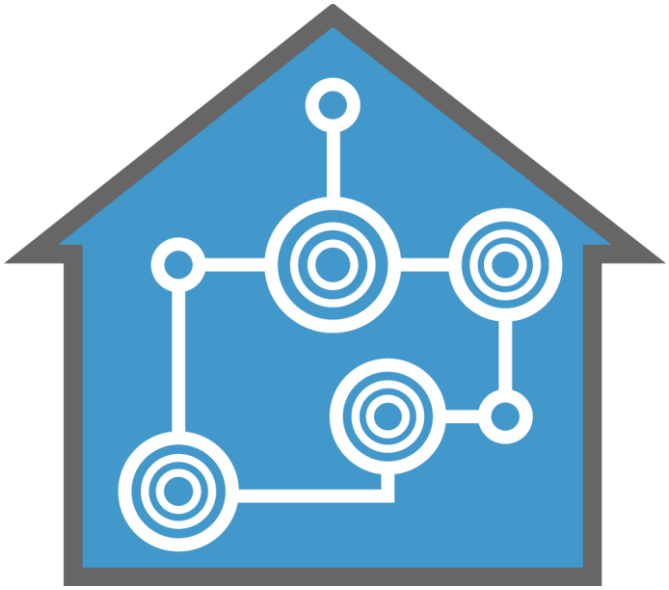
Ilustración 46: Nodo ensamblado

Ilustración 47: Nodo ensamblado

Ilustración 48: Nodos ensamblados

Ilustración 49: Componentes físicos involucrados: A la izquierda el nodo que controla la Camilla. En el centro el nodo que controla el Arnés. A la derecha el nodo Maestro

Pruebas y resultados



Implementación de la solución

Una vez resueltos los aspectos digitales y de diseño del proyecto, se pasó a la etapa de instalación. Para ello, en primera instancia era necesario entrenar al dispositivo con la voz de Santiago (ver Ilustración 23: Definición de órdenes).

Incorporación de órdenes

Tiempo aproximado que requirió el entrenamiento del dispositivo:

20 minutos.

Se comprobó el funcionamiento general del sistema, arrojando resultados positivos en cuanto al reconocimiento de órdenes. Por otro lado, cabe recordar que Santiago previamente tenía experiencia en el uso de dispositivos de estas características, por lo que el entrenamiento del usuario se limitó a una explicación general del dispositivo y la manera de concatenar órdenes para llegar a un resultado, lográndose en poco tiempo. A continuación, se muestra el dispositivo instalado en su lugar de funcionamiento:

Ilustración 50: Nodo maestro instalado

Instalación: nodo camilla

Posteriormiente, se procedió a la instalación del nodo que controla la camilla. Se contrataron los servicios de Sergio Rojas, Electricista matriculado para llevar a cabo esta tarea. Primero se conectó del lado del nodo según lo planificado:

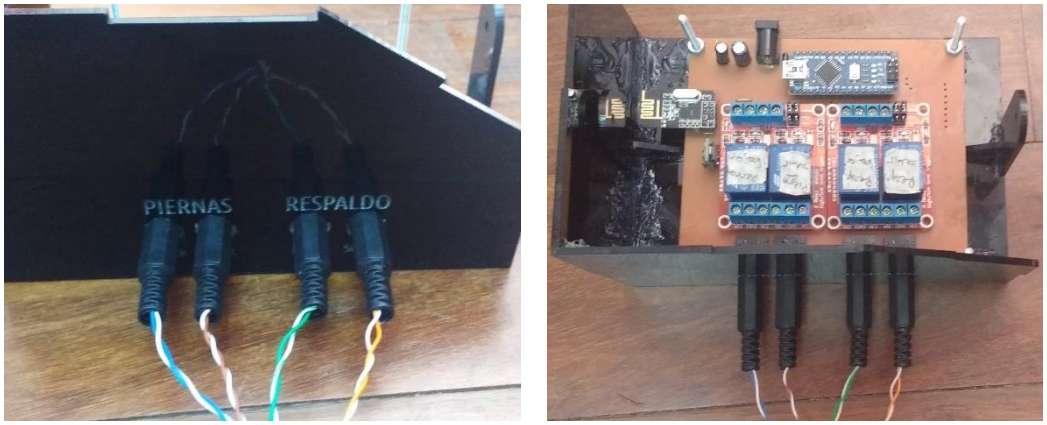


Ilustración 51: Instalación nodo camilla

Luego, se ubicaron los 4 relés que activan los motores en la camilla y se alimentaron con las 4 salidas del nodo:

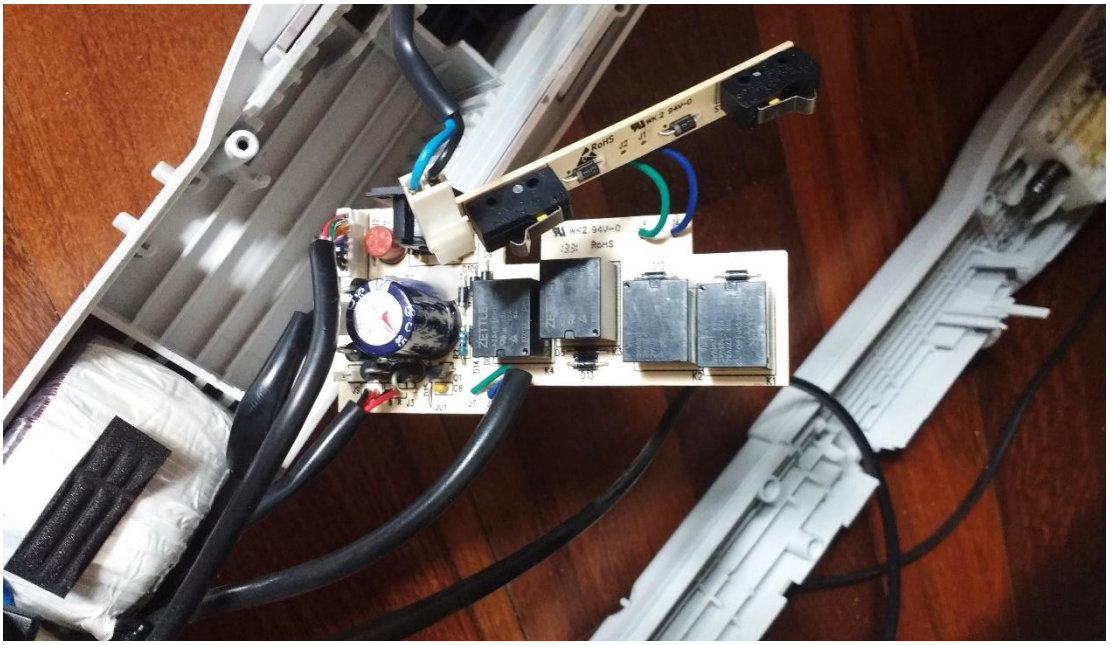


Ilustración 52: Instalación nodo camilla

Habiendo hecho esto, los 4 relés que abren o cierran el circuito de los motores que mueven el respaldo y las piernas de la camilla serían accionados según los relés que activa el Arduino NANO según corresponda. El resultado final de la instalación se puede ver en la siguiente imagen:

Ilustración 53: Instalación nodo camilla

Instalación: nodo elevador de personas con arnés

Debido a que requiere una instalación que demanda una cantidad de tiempo significativa, este proyecto no la contempla. El mismo se instalará en su momento, completando así este documento. Para su instalación se requiere:

- Un motor que ejecute el movimiento horizontal o en "x". (Ver Ilustración 32: Funcionamiento del nodo arnés).
- El acoplamiento del nodo al arnés, manteniendo la mayor estaticidad posible en los sensores ultrasónicos.
- Medir las distancias exactas entre el lugar del guardado del elevador de personas, la posición al lado de la cama y sobre la misma. Con estos datos, calibrar y reprogramar el Sotware que controla el nodo.
- Realizar las pruebas de funcionamiento pertinentes.

Pruebas de funcionamiento

Se realizaron distintas pruebas de funcionamiento, arrojando en su mayoría resultados positivos, no obstante, también se evidenciaron algunos falsos positivos en el dispositivo. Esto se interpreta que es por causa de la tecnología utilizada, que, a pesar de ser barata y fácil de programar, pareciera presentar este tipo de contrapartidas.

A continuación, se enumeran las fallas detectadas y la resolución provista:

Falla	Acción correctiva
Nivel de volumen del buzzer	Incorporación de un potenciómetro a la alimentación del buzzer para seleccionar el volumen requerido.
Falsos positivos	Cambios diversos, desde modificaciones en el feedback hasta en la reestructuración de las órdenes.
Reconocer una órden por otra	Aquellas órdenes que sonaban parecidas en su fonética, se cambiaban alguna del par.

Tabla 12: Fallas detectadas y acciones correctivas

Cabe recalcar que luego de numerosas calibraciones y mediante las distintas soluciones adoptadas, se llegó a lo requerido.

Conclusión

Como conclusión, se puede afirmar que el proyecto fue llevado a cabo con éxito de forma parcial. No obstante, en el momento de implementarlo surgieron distintas perspectivas de la forma que iba tomando el producto, por lo que existen numerosos aspectos que serían interesantes de mejorar:

Aspectos a mejorar

- Diseño de carcasas: Resultado de la inexperiencia en el proceso que conlleva a la materialización de un producto, se pueden contabilizar numerosos errores en lo que concierne al "cage"; se podría haber incluido un botón de encendido/apagado, un enchufe USB hembra que posibilitaría reprogramar el dispositivo sin necesidad de desarmarlo, un potenciómetro que regularía la alimentación del Buzzer (en otras palabras, para subir y bajarle el volumen).

- Distribución de los componentes en las PCBs: Otro proceso en el que se carecía de conocimientos técnicos era a la hora de diseñar las PCBs. Un análisis más exhaustivo previo a la fabricación de las mismas hubiera resultado en un mejor aprovechamiento de los espacios, facilitando la integración de la misma con la carcasa e inclusive reduciendo el tamaño del producto.

- Nodo arnés: Una falla en el diseño particularmente de este nodo, fue la de no incorporar los sensores ultrasónicos en carcasa propia del nodo (los mismos son "externos"). Un montaje sobre la propia carcasa hubiera brindado mayor robustez, obteniendo un resultado similar.

- Incorporaración más símbolos en las carcasas: El proceso de grabado a láser de los acrílicos de las carcasas tiene un bajo costo, por lo que podrían haberse incluido más simbología que acompañe un entendimiento más rápido del producto por parte del usuario, sin recaer en un incremento excesivo del costo de la solución. Otra posibilidad de mejora en el feedback visual del nodo maestro sería la de incorporar elementos adicionales compatibles con Arduino UNO, tales como una pantalla LCD, que podría hasta incluso graficar los desplazamientos y posiciones de los objetos domotizados.

- Comunicación Bluetooth entre nodos: La comunicación por radiofrecuencia utilizada en el proyecto cumple el objetivo sin ningún problema; sin embargo, el módulo Bluetooth compatible con Arduino resulta tener un mayor alcance y más seguridad en comunicación que el chip NRF24L01.

- Utilizar como nodo maestro un procesador con mayor capacidad informática: A la hora de elegir el microcontrolador, se tuvo principalmente en cuenta el lenguaje de programación utilizado y la comodidad que contemplaba su uso. Pero en lo que a costos se refiere, si en vez de utilizar un sistema Arduino UNO con un shield montado EasyVR 2.0 se hubiese empleado simplemente un microcontrolador Raspberry Pi, se hubiera visto reflejado en el costo final del proyecto. Cabe destacar que el microprocesador que trae incorporado el chip Raspberry Pi cuenta con la posibilidad de un entorno del sistema operativo Linux, facilitando el proceso de programación además de generar el ahorro mencionado.

Sin embargo, también merece la pena destacar las bondades del producto logrado:

Aspectos positivos

- Posibilidad de expansión: Al haber sido pensado en un sistema del tipo modular, resulta fácilmente expansible en la cantidad de nodos, incrementando la cantidad de dispositivos que se comandan por voz. La incorporación de un nodo requeriría (además del propio desarrollo del nodo, en su electrónica y programación) de una pequeña reprogramación en el nodo maestro y la grabación de los "comandos patrones" (ver Ilustración 24: Esquema de programación del módulo EasyVR 2.0).

- Robustez a los cortes eléctricos: Se aprovecha la memoria EEPROM que tiene el microcontrolador Arduino para guardar las posiciones de los objetos que el dispositivo estaría destinado a comandar. Esto permite que la misma no se pierda como referencia.

- Estética: Los nodos tienen una estética cuidada, la distribución de los mismos en la habitación no resulta "aparatoso" a la vista, sobre todo por el hecho de haber incorporado la comunicación internodal de forma inalámbrica.

- Seguridad para el usuario: Al tratarse de ser un sistema comandado por voz, no existe la necesidad de tocar ningún componente energizado, lo que le confiere mucha seguridad desde el punto de vista eléctrico.

- Seguridad para el propio equipo: La solución propuesta supone la interacción eléctrica de un sistema que funciona a un nivel de voltaje de electrónica lógica (Arduino funciona a 5v) y de un sistema que maneja un nivel de tensión más elevado (los motores que comandan la cama ortopédica eléctrica y el arnés funcionan a 220v). La manera en que se abordó esta diferencia fue utilizando un elemento intermediario: un relé. Al incorporar este separador de circuitos, se le concedió al sistema seguridad eléctrica.

Bibliografía

- CIEC - Colegio de Ingenieros Especialistas de Córdoba CD - Comisión de Domótica. (2016). Córdoba, Argentina.
- Qué es la domótica. Asociación Española de Domótica e Inmótica – CEDOM. Recuperado de: http://www.cedom.es/sobre-domotica/que-es-domotica
- Definición de accesibilidad. (2012). Autores: Julián Pérez Porto y Ana Gardey. Recuperado de: https://definicion.de/accesibilidad/
- OMS – Discapacidades. (2017). Recuperado de: http://www.who.int/topics/disabilities/es/
- CIF - Versión completa. (2012). Ministerio de Sanidad, Política Social e Igualdad. Gobierno de España.
- Accesibilidad. 5 de noviembre de 2011. En *Wikipedia.* Recuperado en: https://es.wikipedia.org/wiki/Accesibilidad
- Ley 22.314. Accesibilidad de personas con movilidad reducida. Argentina, abril 8 de 1994.
- Norma ISO 9999. Productos de apoyo para personas con discapacidad Clasificación y terminología. Madrid, España. Septiembre, 2007.
- Ley N° 22.431. Sistema de protección integral de los discapacitados. Buenos Aires, 16 de marzo de 1981.
- Ley 24.901. Sistema de prestaciones básicas en habilitación y rehabilitación integral a favor de las personas con discapacidad. 5 de Noviembre de 1997.
- Domótica: Accesibilidad universal. Harold Rozo. 27 de octubre de 2016. Recuperado de: https://www.atontechnologies.com/
- EasyVR 2.0 – Veear. (2017). Recuperado en: http://www.veear.eu/products/old-products/easyvr/
- Arduino UNO Rev3. (2017). Recuperado en: https://store.arduino.cc/usa/arduino-uno-rev3
- Arduino NANO. (2017). Recuperado en: https://store.arduino.cc/usa/arduino-nano
- Arduino FORUM. Recuperado en: https://forum.arduino.cc/
- Índice de tutoriales – Tutoriales Arduino. Prometec. Recuperado en: https://www.prometec.net/indice-tutoriales/
- Norma ISO 9999. Productos de apoyo para personas con discapacidad Clasificación y terminología. Madrid, España. Septiembre, 2007.
- Dirección General de Estadísticas y Censos. Secretaría de la Gobernación. 2003. Recuperado en: http://estadistica.cba.gov.ar/Encuestas/PersonasconDiscapacidad/tabid/440/language/es-AR/Default.aspx
- Dirección General de Educación Especial. Discapacidad Motriz. Recuperado en: http://eespecial.sev.gob.mx/difusion/motriz.php

Anexos

Código – Nodo maestro – Arduino UNO

```
/*Este archivo tiene el codigo necesario para utilizar el shield EasyVR 2.0.
Es el nodo Master del sistema, el cual hace la interepretacion de la voz, y lleva la cuenta de la posicion de los objetos que controla de forma interna.
Posee feedbacks auditivos y acusticos.
Esta conectado por los pones A4 y A5 a un NanoArduino cuya unica funcion es la de recibir el array msj que contiene la informacion necesaria para mapear y activar de manera correcta los nodos de la camilla y de la grua.
*/

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#include "Platform.h"
#include "SoftwareSerial.h"
#include "EEPROM.h"
#include "Wire.h"
#include "I2C_Anything.h"
#ifndef CDC_ENABLED
  // Shield Jumper on SW
  SoftwareSerial port(12,13);
#else
  // Shield Jumper on HW (for Leonardo)
  #define port Serial1
#endif
#else // Arduino 0022 - use modified NewSoftSerial
  #include "WProgram.h"
  #include "NewSoftSerial.h"
  NewSoftSerial port(12,13);
#endif
#include "EasyVR.h"

EasyVR easyvr(port);
enum Groups  //Grupos y comandos
{
  GROUP_0  = 0,
  GROUP_1  = 1,
  GROUP_2  = 2,
  GROUP_3  = 3,
};
enum Group0
{
  G0_XX123 = 0,
};
enum Group1
{
  G1_RESPALDO = 0,
  G1_PIERNAS = 1,
  G1_GRUA = 2,
  G1_CALIBRATE = 3,
};
enum Group2
{
  G2_POSICION_0 = 0,
  G2_POSICION_1 = 1,
  G2_POSICION_2 = 2,
  G2_POSICION_3 = 3,
  G2_POSICION_4 = 4,
  G2_SUBIR_5PORC = 5,
  G2_BAJAR_5PORC = 6,
};
enum Group3
{
  G3_ME_VOY_A_DORMIR = 0,
```

```
    G3_ME_LEVANTE = 1,
    G3_GUARDATE = 2,
  };

  EasyVRBridge bridge;

  int8_t group, idx;

  //comienza declaracion de pines
  int led_respaldo_sube=4;
  int led_respaldo_baja=5;
  int led_piernas_sube=6;
  int led_piernas_baja=7;
  int led_feedback=8;
  int buzzer=A2;
  int sensor_alarma=2;
  int led_alarma=A3;
  //termina declaracion de pines

  //comienza declaracion de variables
  int i=0; // contador de la alarma
  int j=0; // para los for de Array del mensaje de envio
  int contador_errores=0; // contador de errores. a los 5 errores de reconocimiento, se
llama al timeout
  const int tiempo_respaldo=16000; //tiempo que toma desde el respaldo en ir de la
posicion 0 a la posicion 100
  const int tiempo_piernas=13000; //tiempo que toma desde las piernas en ir de la
posicion 0 a la posicion 100
  const int add_pos_respaldo=0; //lugar de la EEPROM en donde se guarda la posicion del
respaldo --> robustez a cortes de luz
  const int add_pos_piernas=1; //lugar de la EEPROM en donde se guarda la posicion del
pie --> robustez a cortes de luz
  const int add_pos_grua=2; //lugar de la EEPROM en donde se guarda la posicion de la
grua --> robustez a cortes de luz
  const float escalon=5;  //aqui se define el valor del escalon para hacer el paso
  volatile float posicion_respaldo=0;
  volatile float posicion_piernas=0;
  volatile float posicion_final_camilla=0;
  volatile float posicion_final_grua=1;
  float distancia_camilla=0;
  float tiempo_delay=0;
  float distancia_grua=0;
  volatile int manejo_respaldo=0; //si esta en 0 manejo piernas. si esta en 1 manejo
respaldo.
  byte bit_enviado=0; //variable auxiliar que uso para enviar bit por bit por I2C del
Master al NanoAntena
  const byte nodo1_address = 1;
  //termina declaracion de variables

  //comienza declaracion de funciones
  void ejecucion_respaldo();
  void ejecucion_piernas();
  void feedback_positivo(); //funcion de feedback que da a entender que Arduino entendio
la orden.
  void feedback_findeorden(); //funcion de feedback que da a entender que Arduino
termino de ejecutar la orden
  void action(); //funcion de accion
  //termina la declaracion de funciones

  //comienza declaracion del paquete enviado por I2C, que luego el NanoArduino enviara a
los nodos por RF
  float msj[]={ 0, 0, 0 };
  char nodo2master[50]; //mensaje que recibiria el Master desde el Nodo Camilla
  //finaliza declaracion del paquete enviado por I2C, que luego el NanoArduino enviara a
los nodos por RF

  void setup()
  {
  posicion_respaldo=EEPROM.read(add_pos_respaldo); //recupero la posicion que se tenia
de respaldo guardado en la memoria EEPROM (robustez a cortes electricos)
```

```
  posicion_piernas=EEPROM.read(add_pos_piernas); //recupero la posicion que se tenia de
piernas guardado en la memoria EEPROM (robustez a cortes electricos)
  pinMode(led_piernas_sube, OUTPUT);
  pinMode(led_piernas_baja, OUTPUT);
  pinMode(led_respaldo_sube, OUTPUT);
  pinMode(led_respaldo_baja, OUTPUT);
  pinMode(led_feedback, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(sensor_alarma, INPUT);
  pinMode(led_alarma, OUTPUT);

  Wire.begin(); //empezamos el protocolo de comunicacion I2C para comunicarnos con la
NanoArduino-Antena

  #ifndef CDC_ENABLED
  if (bridge.check())
  {
      cli();
      bridge.loop(0, 1, 12, 13);
    }
    Serial.begin(9600);
    Serial.println("Bridge not started!");
  #else
    if (bridge.check())
    {
      port.begin(9600);
      bridge.loop(port);
    }
    Serial.println("Bridge connection aborted!");

  #endif

    port.begin(9600);
    while (!easyvr.detect())
    {
      Serial.println("El modulo EasyVR no ha sido detectado");
      delay(1000);
    }
  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("El modulo EasyVR ha sido detectado!");
  easyvr.setTimeout(20);
  easyvr.setLanguage(4);

  group = EasyVR::TRIGGER; //<-- start group (customize)
  }

  uint32_t displayTimer = 0;

  void loop()

  {
  easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

  Serial.println("Nodo Master");
  Serial.println(" ");
  Serial.print("Diga un comando que se encuentre dentro del grupo:  ");
  Serial.println(group);
  easyvr.recognizeCommand(group);
  do
    {
      //puede realizar algun procesamiento mientras se espera por un comando
    }

  while (!easyvr.hasFinished());
  easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off



  idx = easyvr.getWord();
```

```
  if (idx >= 0)
    {
      // built-in trigger (ROBOT)
      // group = GROUP_X; <-- jump to another group X
      return;
    }

    idx = easyvr.getCommand();

    if (idx >= 0)
    {
      // print debug message
      uint8_t train = 0;
      char name[32];
      Serial.print("Command: ");
      Serial.print(idx);
      if (easyvr.dumpCommand(group, idx, name, train))
      {
        Serial.print(" = ");
        Serial.println(name);
      }
      else
        Serial.println();
      easyvr.playSound(0, EasyVR::VOL_FULL);

      // perform some action

      action();
    }

    else // errors or timeout
    {
      if (easyvr.isTimeout())
      {
        if(group!=GROUP_0)
        {
            if(group==GROUP_1)
            {
              group=GROUP_0;
              digitalWrite(led_feedback,HIGH);    //inicio  del  feedback  timeout.  (2
parpadeos de led amarillo)
              delay(1000);
              digitalWrite(led_feedback,LOW);
              delay(250);
              digitalWrite(led_feedback,HIGH);
              delay(1000);
              digitalWrite(led_feedback,LOW);   //finalizacion del feedback timeout. (2
parpadeos de led amarillo)
            }
            else
            {
              group=GROUP_0;
              digitalWrite(buzzer,HIGH);  //inicio del feedback timeout. (2 BEEPS + 2
parpadeos de led amarillo)
              digitalWrite(led_feedback,HIGH);
              delay(50);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);
              delay(200);
              digitalWrite(buzzer,HIGH);
              digitalWrite(led_feedback,HIGH);
              delay(50);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);   //finalizacion del feedback timeout. (2
BEEPS + 2 parpadeos de led amarillo)
            }
        }
      Serial.println("Timed out, try again...");
      }
```

```
      int16_t err = easyvr.getError();
      if (err >= 0)
      {
        if(group!=GROUP_0)
        {
          if(contador_errores<4)
          {
            digitalWrite(led_feedback,HIGH);  //inicio del feedback error. (3 parpadeos
de led amarillo)
            delay(200);
            digitalWrite(led_feedback,LOW);
            delay(50);
            digitalWrite(led_feedback,HIGH);
            delay(200);
            digitalWrite(led_feedback,LOW);
            delay(50);
            digitalWrite(led_feedback,HIGH);
            delay(200);
            digitalWrite(led_feedback,LOW);    //finalizacion  del  feedback  error.  (3
parpadeos de led amarillo)
            contador_errores++;
          }
          else //reseteo cuando detecto 5 errores seguidos
          {
            if(group!=GROUP_1)
            {
              group=GROUP_0;
              digitalWrite(buzzer,HIGH);  //inicio del feedback de reseteo (es similar
al timeout porque es basicamente lo mismo). (2 BEEPS + 2 parpadeos de led amarillo)
              digitalWrite(led_feedback,HIGH);
              delay(100);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);
              delay(200);
              digitalWrite(buzzer,HIGH);
              digitalWrite(led_feedback,HIGH);
              delay(100);
              digitalWrite(buzzer,LOW);
              digitalWrite(led_feedback,LOW);   //inicio  del  feedback  de  reseteo  (es
similar  al  timeout  porque  es  basicamente  lo  mismo).  (2  BEEPS  +  2  parpadeos  de  led
amarillo)
            }
           else
           {
              group=GROUP_0;
              digitalWrite(led_feedback,HIGH);   //inicio del feedback de reseteo (es
similar  al  timeout  porque  es  basicamente  lo  mismo).  (2  BEEPS  +  2  parpadeos  de  led
amarillo)
              delay(500);
              digitalWrite(led_feedback,LOW);
              delay(200);
              digitalWrite(led_feedback,HIGH);
              delay(500);
              digitalWrite(led_feedback,LOW);   //inicio  del  feedback  de  reseteo  (es
similar  al  timeout  porque  es  basicamente  lo  mismo).  (2  BEEPS  +  2  parpadeos  de  led
amarillo)
            }
          }
        }
      Serial.print("Error ");
      Serial.println(err, HEX);
      }
    }
  }
```

```
  void action()
  {
  switch (group)
  {

  //SEPARACION ENTRE GRUPOS//

    case GROUP_0:
    switch (idx)
    {
      case G0_XX123:

      group=GROUP_1;  //entendida la contraseña, va al grupo 1
      digitalWrite(led_feedback,HIGH);   //inicio del feedback de que se entendió la
contraseña (1/2 segundo led feedback prendido, 1/5 de segundo apagado, 1/2 segundo
prendido)
      delay(750);
      digitalWrite(led_feedback,LOW);
      delay(250);
      digitalWrite(led_feedback,HIGH);
      delay(750);
      digitalWrite(led_feedback,LOW);  //finalizacion del feedback de que se entendió la
contraseña
      break;
    }
    break;

  //SEPARACION ENTRE GRUPOS//

    case GROUP_1:
    switch (idx)
    {
      case G1_RESPALDO:
      {
        //escribo en el mensaje que se va a enviar por RF que el nodo a activar es el
nodo 1 (camilla)
        msj[0]=1; // 1 = nodo camilla
        //termino de escribir en el mensaje
        manejo_respaldo=1; //incorporo si el usuario desea manejar respaldo o piernas
        feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.
        Serial.println("Manejas el respaldo");
        group=GROUP_2;
      break;
      }

  //SEPARACION ENTRE CASES//

      case G1_PIERNAS:
      {
        //escribo en el mensaje que se va a enviar por RF que el nodo a activar es el
nodo 1 (camilla)
        msj[0]=1; // 1 = nodo camilla
        //termino de escribir en el mensaje
        manejo_respaldo=0;
        feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.
        Serial.println("Manejas las piernas");
        group=GROUP_2;
      break;
      }

  //SEPARACION ENTRE CASES//

      case G1_GRUA:
      {
        //escribo en el mensaje que se va a enviar por RF que el nodo a activar es el
nodo 1 (camilla)
        msj[0]=2; // 2 = nodo grua
```

```
        //termino de escribir en el mensaje
        group=GROUP_3;
        feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.
      break;
      }

  //SEPARACION ENTRE CASES//

      case G1_CALIBRATE:
      {
        group=GROUP_0;

        feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.

        //comienzo a escribir el mensaje por RF
        msj[0]=1; //mensaje para el nodo camilla
        msj[1]=5; //codificacion que corresponde a la calibracion
        msj[2]=tiempo_respaldo; //enviamos como delay el tiempo que le toma al respaldo
en volverse a la posicion 0, que son 16 segundos --> leer el delay usado abajo para
prender el led
        //finaliza la escritura del mensaje enviado por RF

        //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
        Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
        for(i=0;i<3;i++)
        {
          I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
        }
        Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
        for(i=0;i<3;i++)
        {
          Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar
que es el que se recibio)
        }
        Wire.endTransmission (); //terminamos la transmision
        delay(20);     //delay necesario para que se termine de enviar el mensaje de
calibracion al nodo camilla, para que a continuacion se empiece a codificar y enviar el
mensaje de calibracion al nodo grua
        //terminamos de enviar el array

        //comienzo a escribir el mensaje por RF (ESTE MENSAJE SOLO CALIBRA EL NODO GRUA)
        msj[0]=2; //mensaje para el nodo grua
        msj[1]=10; //codificacion que corresponde a la calibracion
        msj[2]=0; //este valor no se usa para la calibracion de la grua
        //finaliza la escritura del mensaje enviado por RF

        //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
        Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
        for(i=0;i<3;i++)
        {
          I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
        }
        Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
        for(i=0;i<3;i++)
        {
          Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar
que es el que se recibio)
        }
        Wire.endTransmission (); //terminamos la transmision
        delay (20);     //delay necesario para que se termine de enviar el mensaje de
calibracion al nodo grua
        //terminamos de enviar el array
```

```
        digitalWrite(led_respaldo_baja,HIGH); //baja el respaldo a 0 (en un futuro
deberia bajar todas las variables de posicion)
        digitalWrite(led_piernas_baja,HIGH); //baja las piernas a 0 (en un futuro
deberia bajar todas las variables de posicion)
        delay(tiempo_respaldo); //lo baja lo que demora en ir desde 100 a 0 (la camilla
tiene fin de carrera, asi que no hay problema de "pasarse") y se hace por el tiempo mas
largo. (respaldo = 16 seg)
        digitalWrite(led_respaldo_baja,LOW); //dejo de bajar el respaldo --> ya estaria
en 0
        digitalWrite(led_piernas_baja,LOW); //dejo de bajar las piernas --> ya estaria
en 0

        EEPROM.write(add_pos_respaldo,0); //escribo en la memoria EEPROM que todas las
variables de posicion estan en 0
        EEPROM.write(add_pos_piernas,0); //escribo en la memoria EEPROM que todas las
variables de posicion estan en 0
        EEPROM.write(add_pos_grua,1); //escribo en la EEPROM la ubicacion que tendra la
grua luego de la calibracion, osea que vuelve a la posicion 1
        Serial.println("Todas las variables estan puestas a 0");
        feedback_findeorden();
        break;
      }
    }
    break;

  //SEPARACION ENTRE GRUPOS//

    case GROUP_2:
    switch (idx)
    {
      case G2_POSICION_0:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.

      posicion_final_camilla=0; //tomamos el valor sensado

      if(manejo_respaldo==1)
      {
        ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
        ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      }
      break;

  //SEPARACION ENTRE CASES//

      case G2_POSICION_1:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.

      posicion_final_camilla=25; //tomamos el valor sensado y lo dividimos por 100 para
que sea porcentual

      if(manejo_respaldo==1)
      {
        ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
        ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
```

```
      }
      break;

  //SEPARACION ENTRE CASES//

      case G2_POSICION_2:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.

      posicion_final_camilla=50; //tomamos el valor sensado y lo dividimos por 100 para
que sea porcentual

      if(manejo_respaldo==1)
      {
        ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
        ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      }
      break;

  //SEPARACION ENTRE CASES//

      case G2_POSICION_3:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden. //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.

      posicion_final_camilla=75; //tomamos el valor sensado y lo dividimos por 100 para
que sea porcentual

      if(manejo_respaldo==1)
      {
        ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
        ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      }
    break;

  //SEPARACION ENTRE CASES//

      case G2_POSICION_4:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.

      posicion_final_camilla=100; //tomamos el valor sensado y lo dividimos por 100 para
que sea porcentual

      if(manejo_respaldo==1)
      {
        ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
        ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama
a la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
```

```
      }
      break;

  //SEPARACION ENTRE CASES//

      case G2_SUBIR_5PORC:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden
      if(manejo_respaldo==1)
      {
      posicion_final_camilla=EEPROM.read(add_pos_respaldo)+escalon;    //a la posicion
final anterior (que la leo del adress de la EEPROM correspondiente) le sumo el escalon
      ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama a
la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
      posicion_final_camilla=EEPROM.read(add_pos_piernas)+escalon;    //a la posicion
final anterior (que la leo del adress de la EEPROM correspondiente) le sumo el escalon
      ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama a
la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      }
      break;

  //SEPARACION ENTRE CASES//

      case G2_BAJAR_5PORC:
      {
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.
      if(manejo_respaldo==1)
      {
      posicion_final_camilla=EEPROM.read(add_pos_respaldo)-escalon;    //a la posicion
final anterior (que la leo del adress de la EEPROM correspondiente) le sumo el escalon
      ejecucion_respaldo(posicion_final_camilla); //una vez sensado el valor, se llama a
la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      else
      {
      posicion_final_camilla=EEPROM.read(add_pos_piernas)-escalon;    //a la posicion
final anterior (que la leo del adress de la EEPROM correspondiente) le sumo el escalon
      ejecucion_piernas(posicion_final_camilla); //una vez sensado el valor, se llama a
la funcion ejecucion, y se manda la variable posicion_final_camilla
      }
      break;
      }
    break;
    }

  //SEPARACION ENTRE GRUPOS//

    case GROUP_3:
    switch (idx)
    {
      case G3_ME_VOY_A_DORMIR:
      {
      posicion_final_grua=3;
      msj[1]=EEPROM.read(add_pos_grua); //escribo en el array a enviar la posicion
anterior de la grua
      msj[2]=posicion_final_grua; //escribo en el array a enviar la posicion final que
tendra la grua luego del movimiento

      //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
      Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
      for(i=0;i<3;i++)
      {
```

```
      I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
      }
      Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
      for(i=0;i<3;i++)
      {
      Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar que
es el que se recibio)
      }
      Wire.endTransmission (); //terminamos la transmision

      delay (20);

      //terminamos de enviar el array

      group=GROUP_0;
      EEPROM.write(add_pos_grua,posicion_final_grua);  //escribo  en  la  EEPROM  la
ubicacion que tendra la grua luego del movimiento realizado.
      feedback_positivo(); //llamo  a  la  funcion  de  feedback  que  da  a  entender  que
Arduino entendio la orden.
      }
      break;

  //SEPARACION ENTRE CASES//

      case G3_ME_LEVANTE:
      {
      posicion_final_grua=5;
      msj[1]=EEPROM.read(add_pos_grua); //escribo  en  el  array  a  enviar  la  posicion
anterior de la grua
      msj[2]=posicion_final_grua; //escribo en el array a enviar la posicion final que
tendra la grua luego del movimiento

      //enviamos  el  array  ya  completo  al  NanoArduino  con  el  modulo  NRF24L01
("NanoArduino Antena") por el protocolo I2C
      Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
      for(i=0;i<3;i++)
      {
      I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
      }
      Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
      for(i=0;i<3;i++)
      {
      Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar que
es el que se recibio)
      }
      Wire.endTransmission (); //terminamos la transmision

      delay (20);

      //terminamos de enviar el array

      group=GROUP_0;
      EEPROM.write(add_pos_grua,posicion_final_grua);  //escribo  en  la  EEPROM  la
ubicacion que tendra la grua luego del movimiento realizado.
      feedback_positivo(); //llamo  a  la  funcion  de  feedback  que  da  a  entender  que
Arduino entendio la orden.
      }
      break;

  //SEPARACION ENTRE CASES//

      case G3_GUARDATE:
      {
      posicion_final_grua=1;
      msj[1]=EEPROM.read(add_pos_grua); //escribo  en  el  array  a  enviar  la  posicion
anterior de la grua
      msj[2]=posicion_final_grua; //escribo en el array a enviar la posicion final que
tendra la grua luego del movimiento
```

```
      //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
      Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
      for(i=0;i<3;i++)
      {
      I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
      }
      Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
      for(i=0;i<3;i++)
      {
      Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar que
es el que se recibio)
      }
      Wire.endTransmission (); //terminamos la transmision

      delay (20);

      //terminamos de enviar el array

      group=GROUP_0;
      EEPROM.write(add_pos_grua,posicion_final_grua);   //escribo   en   la   EEPROM   la
ubicacion que tendra la grua luego del movimiento realizado.
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.
      feedback_positivo(); //llamo a la funcion de feedback que da a entender que
Arduino entendio la orden.
      }
      break;
    }
    break;
  }
  }


  //FUNCION ALARMA RESPALDO//

  void alarma_respaldo()
  {
  digitalWrite(led_alarma,HIGH);

     if(distancia_camilla>0) //la alarma se activo cuando la camilla estaba subiendo
     {
        {
        digitalWrite(led_respaldo_sube,LOW); //tengo que dejar de subir
        digitalWrite(led_respaldo_baja,HIGH); //y tengo que comenzar a bajar
        for(i=0;i<120;i++)
        {                              // este for es para contar 2 segundos usando el
valor maximo que soporta la funcion
        delayMicroseconds(16383);     // delayMicroseconds. Para recalcular dividir la
cant de segundos que se quiere tener esta funciona activa
        }                             // y dividirla por 0,016383

        digitalWrite(led_respaldo_baja,LOW); //dejo de bajar
        }
     }

      if(distancia_camilla<0) // la alarma se activo cuando la camilla estaba bajando
      {
        {
        digitalWrite(led_respaldo_baja,LOW); //tengo que dejar de bajar
        digitalWrite(led_respaldo_sube,HIGH); //y tengo que comenzar a subir
        for(i=0;i<120;i++)
        {                              // este for es para contar 2 segundos usando el
valor maximo que soporta la funcion
        delayMicroseconds(16383);     // delayMicroseconds. Para recalcular dividir la
cant de segundos que se quiere tener esta funciona activa
        }                             // y dividirla por 0,016383

        digitalWrite(led_respaldo_sube,LOW); //dejo de subir
```

```
        }
      }
  digitalWrite(led_alarma,LOW);
  }



  //FUNCION ALARMA PIERNAS//

  void alarma_piernas()
  {
  digitalWrite(led_alarma,HIGH);

     if(distancia_camilla>0) //la alarma se activo cuando la camilla estaba subiendo
     {
       //comienzo a escribir el Array para mandar por RF
       msj[0]=1; //le estoy escribiendo al nodo 1
       msj[1]=2; //si venia subiendo, tengo que mandar la orden de bajar por RF
       msj[2]=2000; //se manda el delay, que serian 2 segundos, que es la cantidad de
tiempo que a ejecutar la accion opuesta a la que se venia ejecutando
       //termino de escribir el Array para mandar por RF


       {
        digitalWrite(led_piernas_sube,LOW); //tengo que dejar de subir
        digitalWrite(led_piernas_baja,HIGH); //y tengo que comenzar a bajar
        for(i=0;i<120;i++)
        {                                  // este for es para contar 2 segundos usando el
valor maximo que soporta la funcion
        delayMicroseconds(16383);     // delayMicroseconds. Para recalcular dividir la
cant de segundos que se quiere tener esta funciona activa
        }                             // y dividirla por 0,016383

        digitalWrite(led_piernas_baja,LOW); //dejo de bajar
        }

     }

      if(distancia_camilla<0) // la alarma se activo cuando la camilla estaba bajando
      {

        {
        digitalWrite(led_piernas_baja,LOW); //tengo que dejar de bajar
        digitalWrite(led_piernas_sube,HIGH); //y tengo que comenzar a subir
        for(i=0;i<120;i++)
        {                                  // este for es para contar 2 segundos usando el
valor maximo que soporta la funcion
        delayMicroseconds(16383);     // delayMicroseconds. Para recalcular dividir la
cant de segundos que se quiere tener esta funciona activa
        }                             // y dividirla por 0,016383

        digitalWrite(led_piernas_sube,LOW); //dejo de subir
        }
     }

  digitalWrite(led_alarma,LOW);
  }

  //FUNCION EJECUCION RESPALDO//

  void ejecucion_respaldo(float posicion_final_camilla) //comienzo de la funcion de
ejecucion del movimiento del respaldo de la camilla
  {
        if(posicion_final_camilla>100)
        {
        posicion_final_camilla=100; //con esto hago que el valor maximo que toma la
distancia_camilla puede ser 100.
        }
        if(posicion_final_camilla<0)
        {
```

```
        posicion_final_camilla=0;  //y con esto que el valor minimo sea 0
        }

        distancia_camilla=(posicion_final_camilla-EEPROM.read(add_pos_respaldo))/100;
//lo dividimos por 100 para que sea porcentual

        attachInterrupt(1, alarma_respaldo, RISING); //activacion de la alarma
        if(distancia_camilla==0)
        {
          //no hacer nada porque se pidio ir a la posicion en la que ya se encuentra
        }
        if(distancia_camilla>0)                  //circunstancia                    donde
posicion_final_camilla>posicion_respaldo --> implica subir
        {

        //comienza escritura del mensaje enviado por RF
        msj[1]=1; //la accion=1 significa subir el respaldo
        //termina la escritura del mensaje enviado por RF

        digitalWrite(led_respaldo_sube,HIGH); //pongo en alto el led de subir (verde)
        tiempo_delay=(tiempo_respaldo*distancia_camilla);
        Serial.println("El delay calculado es: ");
        Serial.println(tiempo_delay);

        //comienza escritura del mensaje enviado por RF
        msj[2]=tiempo_delay; //pongo el valor calculado del delay en el array para saber
cuanto activar los reles del nodo de la camilla
        //termina la escritura del mensaje enviado por RF

        //enviamos  el  array  ya  completo  al  NanoArduino  con  el  modulo  NRF24L01
("NanoArduino Antena") por el protocolo I2C
        Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
        for(i=0;i<3;i++)
        {
          I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
        }
        Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
        for(i=0;i<3;i++)
        {
          Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar
que es el que se recibio)
        }
        Wire.endTransmission (); //terminamos la transmision

        delay (20);

        //terminamos de enviar el array

        delay(tiempo_delay);  // 16.000 segundos por % que subo o bajo
        digitalWrite(led_respaldo_sube,LOW); //pongo en bajo el led de subir (verde)
        }
        else // caso contrario posicion_final_camilla<posicion_respaldo -->implica bajar
        {

        //comienza escritura del mensaje enviado por RF
        msj[1]=2; //la accion=2 significa bajar el respaldo
        //termina la escritura del mensaje enviado por RF

        digitalWrite(led_respaldo_baja,HIGH); //pongo en alto el led de bajar (rojo)
        tiempo_delay=(tiempo_respaldo*(distancia_camilla*(-1)));     //pongo  el  valor
distancia_camilla en positivo, ya que el signo me servia solo para saber si subir o
bajar.
        Serial.println("El delay calculado es: ");
        Serial.println(tiempo_delay);

        //comienza escritura del mensaje enviado por RF
        msj[2]=tiempo_delay; //pongo el valor calculado del delay en el array para saber
cuanto activar los reles del nodo de la camilla
        //termina la escritura del mensaje enviado por RF
```

```
        //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
        Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
        for(i=0;i<3;i++)
        {
          I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
        }
        Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
        for(i=0;i<3;i++)
        {
          Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar
que es el que se recibio)
        }
        Wire.endTransmission ();

        delay (20);
        //terminamos la transmision

        delay(tiempo_delay); // 16.000 segundos por % que subo o bajo (multiplico por (-
1) por una cuestion matematica)
        digitalWrite(led_respaldo_baja,LOW); //pongo en bajo el led de subir (rojo)
        }
        detachInterrupt(1); // desactivacion de la alarma

        Serial.println("La posicion del respaldo antes del movimiento es: ");
        Serial.println(EEPROM.read(add_pos_respaldo));
        Serial.println("La distancia_camilla a recorrer es: ");
        Serial.print(distancia_camilla*100);
        Serial.println("%");
        Serial.println("La posicion del respaldo luego del movimiento sera:");
        Serial.println(posicion_final_camilla);
        EEPROM.write(add_pos_respaldo,posicion_final_camilla); //escribo en la EEPROM la
posicion que tendra el respaldo al finalizar el movimiento
        feedback_findeorden();
        group=GROUP_2;
  }

  //FUNCION EJECUCION PIERNAS//

  void ejecucion_piernas(float posicion_final_camilla) //comienzo de la funcion de
ejecucion del movimiento de las piernas de la camilla
  {
        if(posicion_final_camilla>100)
        {
        posicion_final_camilla=100; //con esto hago que el valor maximo que toma la
distancia_camilla puede ser 100.
        }
        if(posicion_final_camilla<0)
        {
        posicion_final_camilla=0;  //y con esto que el valor minimo sea 0
        }

        distancia_camilla=(posicion_final_camilla-EEPROM.read(add_pos_piernas))/100;
//lo dividimos por 100 para que sea porcentual

        attachInterrupt(1, alarma_piernas, RISING); //activacion de la alarma
        if(distancia_camilla==0)
        {
          //no hacer nada porque se pidio ir a la posicion en la que ya se encuentra
        }
        if(distancia_camilla>0)                  //circunstancia                  donde
posicion_final_camilla>posicion_piernas --> implica subir
        {

        //comienza escritura del mensaje enviado por RF
        msj[1]=3; //la accion=3 significa subir las piernas
        //termina la escritura del mensaje enviado por RF
```

```
        digitalWrite(led_piernas_sube,HIGH); //pongo en alto el led de subir (verde)
        tiempo_delay=(tiempo_piernas*distancia_camilla);
        Serial.println("El delay calculado es: ");
        Serial.println(tiempo_delay);

        //comienza escritura del mensaje enviado por RF
        msj[2]=tiempo_delay; //pongo el valor calculado del delay en el array para saber
cuanto activar los reles del nodo de la camilla
        //termina la escritura del mensaje enviado por RF

        //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
        Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
        for(i=0;i<3;i++)
        {
          I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
        }
        Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
        for(i=0;i<3;i++)
        {
          Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar
que es el que se recibio)
        }
        Wire.endTransmission (); //terminamos la transmision

        delay (20);
        //terminamos de enviar el array

        delay(tiempo_delay);  // 13.000 milisegundos por % que subo o bajo
        digitalWrite(led_piernas_sube,LOW); //pongo en bajo el led de subir (verde)
        }
        else // caso contrario posicion_final_camilla<posicion_piernas -->implica bajar
        {

        //comienza escritura del mensaje enviado por RF
        msj[1]=4; //la accion=4 significa bajar las piernas
        //termina la escritura del mensaje enviado por RF

        digitalWrite(led_piernas_baja,HIGH); //pongo en alto el led de bajar (rojo)
        tiempo_delay=(tiempo_piernas*(distancia_camilla*(-1)));     //pongo   el   valor
distancia_camilla en positivo, ya que el signo me servia solo para saber si subir o
bajar.
        Serial.println("El delay calculado es: ");
        Serial.println(tiempo_delay);

        //comienza escritura del mensaje enviado por RF
        msj[2]=tiempo_delay; //pongo el valor calculado del delay en el array para saber
cuanto activar los reles del nodo de la camilla
        //termina la escritura del mensaje enviado por RF

        //enviamos el array ya completo al NanoArduino con el modulo NRF24L01
("NanoArduino Antena") por el protocolo I2C
        Wire.beginTransmission (nodo1_address); //empezamos la transmision hacia el nodo
NanoArduinoRF
        for(i=0;i<3;i++)
        {
          I2C_writeAnything (msj[i]);  //enviamos el array bit por bit
        }
        Serial.println("Se ha enviado el siguiente mensaje por I2C al NanoArduinoRF: ");
        for(i=0;i<3;i++)
        {
          Serial.println(msj[i]); //imprimimos el mensaje que se envio (para corroborar
que es el que se recibio)
        }
        Wire.endTransmission (); //terminamos la transmision
        delay (20);
        //terminamos de enviar el array
```

```
        delay(tiempo_delay); // 13.000 segundos por % que subo o bajo (multiplico por (-
1) por una cuestion matematica)
        digitalWrite(led_piernas_baja,LOW); //pongo en bajo el led de subir (rojo)
        }
        detachInterrupt(1); // desactivacion de la alarma

        Serial.println("La posicion de las piernas antes del movimiento es: ");
        Serial.println(EEPROM.read(add_pos_piernas));
        Serial.println("La distancia_camilla a recorrer es: ");
        Serial.print(distancia_camilla*100);
        Serial.println("%");
        Serial.println("La posicion de las piernas luego del movimiento sera:");
        Serial.println(posicion_final_camilla);
        EEPROM.write(add_pos_piernas,posicion_final_camilla); //escribo en la EEPROM la
posicion que tendran las piernas al finalizar el movimiento
        feedback_findeorden();
        group=GROUP_2;
  }


  //FUNCION FEEDBACK POSITIVO//

  void feedback_positivo()
  {
    digitalWrite(buzzer,HIGH);              //inicio de feedback de que se entendio la
señal (BEEP+led amarillo)
    digitalWrite(led_feedback,HIGH);
    delay(200);
    digitalWrite(buzzer,LOW);
    digitalWrite(led_feedback,LOW);      //finalizacion del feedback de que se entendio
la señal (BEEP+led amarillo)
    contador_errores=0; //reseteo el contador de errores.
  }

  //FUNCION FEEDBACK FIN DE ORDEN//

  void feedback_findeorden()
  {
    digitalWrite(buzzer,HIGH);       //inicio de feedback de que se completo la orden
requerida (3 BEEPS+ 3 led amarillo)
    digitalWrite(led_feedback,HIGH);
    delay(50);
    digitalWrite(buzzer,LOW);
    digitalWrite(led_feedback,LOW);
    delay(150);
    digitalWrite(buzzer,HIGH);
    digitalWrite(led_feedback,HIGH);
    delay(50);
    digitalWrite(buzzer,LOW);
    digitalWrite(led_feedback,LOW);
    delay(150);
    digitalWrite(buzzer,HIGH);
    digitalWrite(led_feedback,HIGH);
    delay(50);
    digitalWrite(buzzer,LOW);
    digitalWrite(led_feedback,LOW);  //finaliza el feedbackde que se completo la orden
requerida (3 BEEPS+ 3 led amarillo)
    contador_errores=0; //reseteo el contador de errores
```

Código – Nodo maestro – Arduino NANO

```
/* NanoArduino que esta conectado al master via "Wire".
Sirve simplemente de "antena", envia y recibe datos y los transfiere al Master, quien decide que hacer.
*/

#include "Wire.h"
#include "I2C_Anything.h"
#include "SPI.h"
#include "nRF24L01.h"
#include "RF24.h"

#define CE_PIN   9
#define CSN_PIN 10

//comienza declaracion de variables
int i=0; //variable que se usa para el for que recorre el array
//finaliza declaracion de variables

//comienza declaracion de la instancia de la comunicacion I2C
const byte nodo1_adress = 1;
volatile boolean haveData = false;
//termina declaracion de la instancia de la comunicacion I2C

//comienza declaracion de la instancia de radio
const byte slaveAddress[5] = {'R','x','A','A','A'}; //address del slave
const byte masterAddress[5] = {'T','X','a','a','a'}; //address del master
RF24 radio(9, 10);
//termina la declaracion de la instancia de radio

//comienza declaracion de los paquetes enviados y recibidos por RF
float msj[]={ 0, 0, 0 }; //mensaje que se recibe por I2C desde el master y este NanoArduino envia a los nodos por RF
char nodo2master[50]; //es el char de ACK que manda el nodo cuando lo recibio correctamente (por RF)
bool dato_nuevo = false; //se crea una instancia que hace de flag cuando se tiene un dato nuevo
//finaliza la declaracion de los paquetes enviados y recibidos por RF

//comienza declaracion de funciones
void envio(); //funcion de envio de datos por RF
void recibo(); //funcion de recibo de datos por RF
//termina la declaracion de funciones

void setup()
{
Wire.begin(nodo1_adress);    // se une a la red I2C con el address asignado
Wire.onReceive(receiveEvent);    // interrupcion por recepcion de datos por I2C
Serial.begin(9600);

//inicializacion de la radio
radio.begin(); //se enciende la radio
Serial.println("Se encendio la radio. Rol del nodo: Master");
radio.setDataRate( RF24_250KBPS ); //velocidad de transferencia
radio.openWritingPipe(slaveAddress); //estructura que se usa para escribir mensajes
radio.openReadingPipe(1, masterAddress); //estructura que se usa para leer mensajes
radio.setRetries(3,5); // 3 = delay ; 5 = cuenta.
radio.startListening(); //enciendo la radio y empiezo a escuchar
//finalizacion del seteo de la radio
}

void loop()
{
  if(haveData)
  {
    Serial.println(" ");
    Serial.println("Dato recibido por I2C desde el Master. Mensaje para el nodo: ");
```

```
      Serial.println(msj[0]);
      Serial.println("2da variable que se envia: ");
      Serial.println(msj[1]);
      Serial.println("3era variable que se envia: ");
      Serial.println(msj[2]);
      Serial.println(" ");

      envio();
      haveData = false;
    }

    if(radio.available())
    {
      radio.read( &nodo2master, sizeof(nodo2master));
      Serial.println(" ");
      Serial.println(nodo2master[50]);
      Serial.println(" ");
    }
  }

  // La siguiente funcion se ejecuta cada vez que se recibe un dato por I2C desde el
Master
  void receiveEvent(int bytes_msj) //    bytes_msj = cantidad de datos ingresantes por
I2C
  {
  if(bytes_msj >= (sizeof msj))
  {
    for(i=0;i<3;i++)
    {
      I2C_readAnything (msj[i]);
    }
  haveData = true;
  }
  }

  void envio()
  {
    radio.stopListening(); //dejo de escuchar para enviar
    radio.write( &msj, sizeof(msj) ); //envio el Array
    radio.startListening(); //vuelvo a escuchar
    Serial.println("Paquete RF Enviado: ");
    for(i=0; i<3; i++)
    {
      Serial.println(msj[i]);
    }
  }
```

Código – Nodo cama ortopédica – Arduino NANO

```
/* Nodo camilla: este nodo solo recibe desde el Master.
Recibe un Array de 3 valores:
  1er valor) nodo al que se le habla: 1(camilla)/2(grua) --> tiene que ser =1 para que se este hablando a ESTE nodo,
  2do valor) accion a realizar: 1(subir respaldo) ,2(bajar respaldo) ,3(subir piernas) ,4(bajar piernas)
  3er valor) delay calculado en el master --> es el tiempo en el que el rele tiene que estar en 1
*/

#include "SPI.h"
#include "nRF24L01.h"
#include "RF24.h"

#define CE_PIN   9
#define CSN_PIN 10

const byte slaveAddress[5] = {'R','x','A','A','A'};
const byte masterAddress[5] = {'T','X','a','a','a'};

RF24 radio(CE_PIN, CSN_PIN); // Se crea la instancia de radio

//comienza declaracion de los paquetes enviados y recibidos por RF
float msj[]={ 0, 0, 0 };
char nodo2master[50]={'Mensaje recibido en el nodo 1.'}; //es el char de nodo2master que manda el nodo cuando lo recibio correctamente
bool dato_nuevo = false; //se crea una instancia que hace de flag cuando se tiene un dato nuevo
//finaliza la declaracion de los paquetes enviados y recibidos por RF

//comienza declaracion de pines
int subir_respaldo=2;
int bajar_respaldo=3;
int subir_piernas=4;
int bajar_piernas=5;
int j=0; //contador para leer el array recibido del master
//termina declaracion de pines

void setup()
{
  pinMode(subir_respaldo,OUTPUT);
  pinMode(bajar_respaldo,OUTPUT);
  pinMode(subir_piernas,OUTPUT);
  pinMode(bajar_piernas,OUTPUT);

  Serial.begin(9600);

  Serial.println("Nodo 1: Camilla");

  radio.begin();
  radio.setDataRate( RF24_250KBPS );
  radio.openWritingPipe(masterAddress);  //pipes o tuberias de comunicacion. Hay una para hablar y otra para escribir.
  radio.openReadingPipe(1, slaveAddress);
  radio.setRetries(3,5);  //delay, cuenta (tiene 5 intentos de 3 segundos por intento)
  radio.startListening(); //enciendo la radio y empiezo a escuchar
}

void loop()
{
  getData();
  showData();
  send();
}
```

```
  void send()
  {
    if (dato_nuevo == true)
      {
        radio.stopListening(); //para enviar el nodo2master debo dejar de escuchar.
        radio.write( &nodo2master, sizeof(nodo2master) );  //"escribo" en la radio el
nodo2master
        radio.startListening(); //vuelvo a escuchar

        Serial.println(" ");
        Serial.println("Se envio el nodo2master al Master."); //aviso en el nodo que se
envio el mensaje
        Serial.println(" ");

        Serial.println();

      if(msj[1]==1) //subir respaldo
      {
        digitalWrite(subir_respaldo,HIGH);
        delay(msj[2]); //espero por el delay calculado y enviado desde el Master
        digitalWrite(subir_respaldo,LOW);
      }

      if(msj[1]==2)  //bajar respaldo
      {
        digitalWrite(bajar_respaldo,HIGH);
        delay(msj[2]); //espero por el delay calculado y enviado desde el Master
        digitalWrite(bajar_respaldo,LOW);
      }

      if(msj[1]==3)  //subir piernas
      {
        digitalWrite(subir_piernas,HIGH);
        delay(msj[2]); //espero por el delay calculado y enviado desde el Master
        digitalWrite(subir_piernas,LOW);
      }

      if(msj[1]==4)  //bajar piernas
      {
        digitalWrite(bajar_piernas,HIGH);
        delay(msj[2]); //espero por el delay calculado y enviado desde el Master
        digitalWrite(bajar_piernas,LOW);
      }

      if(msj[1]==5) //calibrate
      {
        digitalWrite(bajar_respaldo,HIGH);
        digitalWrite(bajar_piernas,HIGH);
        delay(msj[2]);
        digitalWrite(bajar_respaldo,LOW);
        digitalWrite(bajar_piernas,LOW);
      }

      dato_nuevo = false;
      }
  }

  void getData()
  {
    if(radio.available()) //leo la radio para ver si hay un dato disponible para este
nodo
    {
      radio.read( &msj, sizeof(msj) ); //si lo hubiera, lo lee y lo guarda de la misma
forma en la que llega (array) en el lugar de la memoria "&msj"
      if(msj[0]==1)
      {
        dato_nuevo = true; //activo el booleano que indica la entrada de un dato nuevo
      }
      else
      {
```

```
      //no hacer nada ya que no le hablan a este nodo
    }
  }
}

void showData()
{
  if (dato_nuevo == true)
  {
    Serial.println(" ");
    Serial.println("Paquete RF recibido desde el Master: ");
    for(j=0; j<3; j++)
      {
        Serial.println(msj[j]);
      }
    Serial.println(" ");
  }
}
```

Código – Nodo arnés – Arduino NANO

```
/* Nodo grua: este nodo solo recibe desde el Master.
Recibe un Array de 3 valores:
  1er valor) Nodo al que se le habla: 1(camilla)/2(grua) --> Tiene que ser =2 para que se este hablando a ESTE nodo.
  2do valor) Posicion inicial de la grua (1-3-5) -- 2do valor = 10 para calibracion.
  3er valor) Posicion final de la grua luego del movimiento (1-3-5).
Este nodo cuenta con 2 sensores ultrasonicos. Cada sensor apunta a una de las 2 direcciones posibles que puede tomar la grua: x o y.
*/

#include "SPI.h"
#include "nRF24L01.h"
#include "RF24.h"
#include "Ultrasonido.h"

#define CE_PIN   9
#define CSN_PIN 10

const byte slaveAddress[5] = {'R','x','A','A','A'};
const byte masterAddress[5] = {'T','X','a','a','a'};

RF24 radio(CE_PIN, CSN_PIN); // Se crea la instancia de radio

//comienza declaracion de los paquetes enviados y recibidos por RF
float msj[]={ 0, 0, 0 };
char nodo2master[50]={'Mensaje recibido en el nodo 2.'}; //es el char de nodo2master que manda el nodo cuando lo recibio correctamente
bool dato_nuevo = false; //se crea una instancia que hace de flag cuando se tiene un dato nuevo
//finaliza la declaracion de los paquetes enviados y recibidos por RF

//comienza declaracion de pines
int avanza_x=4; //se prende el rele que activa el motor que hace avanzar la grua en el sentido +X
int vuelve_x=5; //se prende el rele que activa el motor que hace avanzar la grua en el sentido -X
int sube=2; //se prende el rele que activa el motor que hace avanzar la grua en el sentido +Y
int baja=3; //se prende el rele que activa el motor que hace avanzar la grua en el sentido -Y
//termina declaracion de pines

//comienza declaracion de variables
int j=0; //contador para leer el array recibido del master
int sensor=0; //0 signfica el sensor X; 1 significa el sensor Y
float distancia; //variable en donde se va almacenando las distancias sensadas por ultrasonido
int dist_min=0; //variable que representa el valor minimo del intervalo del sensado
int dist_max=0;; //variable que representa el valor maximo del intervalo del sensado
float pos_x=0; //posicion x actual de la grua
float pos_y=0; //posicion y actual de la grua
//termina declaracion de variables

//comienza declaracion de intervalos que representan cada posicion
//posicion 1: X: 100 cm ; Y: 10 cm
int x_min_1=90;
int x_max_1=110;
int y_min_1=5;
int y_max_1=15;
//posicion 2: X: 50 cm ; Y: 10 cm
int x_min_2=40;
int x_max_2=60;
int y_min_2=5;
int y_max_2=15;
//posicion 3: X: 50 cm ; Y: 100 cm
int x_min_3=40;
```

```
  int x_max_3=60;
  int y_min_3=90;
  int y_max_3=110;
  //posicion 4: X: 20 cm ; Y: 10 cm
  int x_min_4=15;
  int x_max_4=25;
  int y_min_4=5;
  int y_max_4=15;
  //posicion 5: X: 20 cm ; Y: 50 cm
  int x_min_5=15;
  int x_max_5=25;
  int y_min_5=40;
  int y_max_5=60;
  //termina declaracion de intervalos que representan cada posicion

  //comienza seteo sensores ultrasonicos (pin Trigger, pin Echo) -->al utilizar la
libreria, no hace falta configurar los pines de eco como entrada y de trigger como
salida ya que lo hace intermanete la libreria
  Ultrasonido SensorY=Ultrasonido(A1,A0);
  Ultrasonido SensorX=Ultrasonido(A3,A2);
  //termina seteo sensores ultrasonicos (pin Trigger, pin Echo)  -->al utilizar la
libreria, no hace falta configurar los pines de eco como entrada y de trigger como
salida ya que lo hace intermanete la libreria

  //comienza declaracion de funciones
  void accion();
  //termina declaracion de funciones

  void setup()
  {
    pinMode(avanza_x,OUTPUT);
    pinMode(vuelve_x,OUTPUT);
    pinMode(sube,OUTPUT);
    pinMode(baja,OUTPUT);

    Serial.begin(9600);

    Serial.println("Nodo 2: Grua");

    radio.begin();
    radio.setDataRate( RF24_250KBPS );
    radio.openWritingPipe(masterAddress);  //pipes o tuberias de comunicacion. Hay una
para hablar y otra para escribir.
    radio.openReadingPipe(1, slaveAddress);
    radio.setRetries(3,5);  //delay, cuenta (tiene 5 intentos de 3 segundos por intento)
    radio.startListening(); //enciendo la radio y empiezo a escuchar
  }

  void loop()
  {
    getData();
    showData();
    send();
  }

  void getData()
  {
    if(radio.available()) //leo la radio para ver si hay un dato disponible para este
nodo
    {
      radio.read( &msj, sizeof(msj) ); //si lo hubiera, lo lee y lo guarda de la misma
forma en la que llega (array) en el lugar de la memoria "&msj"
      if(msj[0]==2)
      {
        dato_nuevo = true; //como le hablan a este nodo, activo el booleano que indica
la entrada de un dato nuevo
      }
      else
      {
        //no hacer nada ya que no le hablan a este nodo
```

```
      }
    }
  }

  void showData()
  {
    if (dato_nuevo == true)
    {
      Serial.println(" ");
      Serial.println("Paquete RF recibido desde el Master: ");
      for(j=0; j<3; j++)
        {
          Serial.println(msj[j]);
        }
      Serial.println(" ");
    }
  }

  void send()
  {
    if (dato_nuevo == true)
    {
      radio.stopListening(); //para enviar el nodo2master debo dejar de escuchar.
      radio.write( &nodo2master, sizeof(nodo2master) );   //"escribo" en la radio el
nodo2master
      radio.startListening(); //vuelvo a escuchar

      Serial.println(" ");
      Serial.println("Se envio el ACK al Master."); //aviso en el nodo que se envio el
mensaje
      Serial.println(" ");

      Serial.println();

  //SALE DESDE LA POSICION 1//

      if(msj[1]==1) //sale desde la posicion 1
      {
        if(msj[2]==1)
        {
          //no hace nada ya que se le pidio ir a la posicion 1 desde la posicion 1
        }

        if(msj[2]==3)  //desplazamiento: posicion 1 -(+x)-> posicion 2 -(-y)-> posicion
3
        {
          accion(avanza_x, 0, x_min_2, x_max_2); //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(baja, 1, y_min_3, y_max_3); //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
        }

        if(msj[2]==5) //desplazamiento: posicion 1 -(+x)-> posicion 4 -(-y)-> posicion 5
        {
          accion(avanza_x, 0, x_min_4, x_max_4); //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(baja, 1, y_min_5, y_max_5); //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
        }
      }

  //SALE DESDE LA POSICION 3//

      if(msj[1]==3)  //sale desde la posicion 3
```

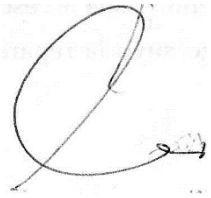
```
      {
        if(msj[2]==1) //desplazamiento: posicion 3 -(+y)-> posicion 2 -(-x)-> posicion 1
        {
          accion(sube, 1, y_min_2, y_max_2); //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(vuelve_x, 0, x_min_1, x_max_1);  //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
        }

        if(msj[2]==3)
        {
          //no hace nada ya que se le pidio ir a la posicion 3 desde la posicion 3
        }

        if(msj[2]==5) //desplazamiento: posicion 3 -(+y)-> posicion 2 -(+x)-> posicion 4
-(-y)->posicion 5
        {
          accion(sube, 1, y_min_2, y_max_2);   //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(avanza_x, 0, x_min_4, x_max_4);  //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(baja, 1, y_min_5, x_max_5);   //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
        }
      }

  //SALE DESDE LA POSICION 5//

      if(msj[1]==5)  //sale desde la posicion 5
      {
        if(msj[2]==1)  //desplazamiento: posicion 5 -(+y)-> posicion 4 -(-x)-> posicion
1
        {
          accion(sube, 1, y_min_4, y_max_4);   //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(vuelve_x, 0, x_min_1, x_max_1);  //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
        }

        if(msj[2]==3)       //desplazamiento: posicion 5 -(+y)-> posicion 4 -(-x)->
posicion 2 -(-y)-> posicion 3
        {
          accion(sube, 1, y_min_4, y_max_4);   //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(vuelve_x, 0, x_min_2, x_max_2);  //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
          delay(10);
          accion(baja, 1, y_min_3, y_max_3);   //mando a la funcion de accion los
parametros necesarios para llevar a cabo el desplazamiento solicitado: rele, sensor
actuante: 0=x y=1; minimo del intervalo de sensado; maximo del intervalo de sensado
        }

        if(msj[2]==5)
        {
          //no hace nada ya que se le pidio ir a la posicion 5 desde la posicion 5
        }
```

```
      }

      if(msj[1]==10) //calibrate
      {
        accion(sube, 1, y_min_2, y_max_2);  //primero tengo que subir desde donde este
hasta el techo; cabe aclarar que y_min_2=y_min_4 ; y_max_2=y_max_4,  por lo que da igual
usar los de valores de la posicion 2 o la posicion 4, el objetivo es subir hasta el
techo
        delay(10);
        accion(vuelve_x, 2, x_min_1, x_max_1);  //luego debo volver hasta la posicion de
guardado
      }


      dato_nuevo = false;
      }
  }

  void accion(int rele, int sensor,int dist_min, int dist_max) //ingresan el 4
variables: *Variable 1) relé que se activa: +x, -x, +y o -y. *Variable 2)sensor=0 -->
SensorX ; sensor=1 --> SensorY. *Variable 3 y Variable 4) son los valores que
corresponden al intervalo de distancia.
  {
    digitalWrite(rele,HIGH);  //comienza el desplazamiento
    Serial.println(" ");
    Serial.println("Comienza el desplazamiento, el rele activado es: ");
    Serial.println(rele);
    Serial.println("Y el desplazamiento que se va a realizar va a ser el siguiente: ");
    Serial.println("Parte desde la posicion: ");
    Serial.println(msj[1]);
    Serial.println("Y se dirige hacia la posicion: ");
    Serial.println(msj[2]);
    Serial.println(" ");
    for(int i=0 ; i<5000 ; i++) //se cuenta por determinada cantidad de tiempo (esta
variable debe sobre estimarse) --> i=(valor/4)/60 [minutos]
    {
      if(sensor==0) //debo leer el sensorX
      {
        distancia=SensorX.Distancia();
        pos_x=distancia; //si el desplazamiento fue en x, guardo la posicion alcanzada
como la posicion acutal de la grua respecto a la pared
        Serial.println(" ");
        Serial.println("Distancia medida: ");
        Serial.println(pos_x);
      }
      if(sensor==1) //debo leer el sensorY
      {
        distancia=SensorY.Distancia();
        pos_y=distancia; //si el desplazamiento fue en y, guardo la posicion alcanzada
como la posicion acutal de la grua respecto al techo
        Serial.println("Distancia medida: ");
        Serial.println(pos_y);
        Serial.println(" ");
      }
      if((dist_min<distancia) && (distancia<dist_max))
      {
        digitalWrite(rele,LOW);   //termina desplazamiento
        i=9999;
        Serial.println(" ");
        Serial.println("Posicion actual de la grua: ");
        Serial.print(pos_x);
        Serial.println(" cm de la pared");
        Serial.print(pos_y);
        Serial.println(" cm del techo");
        Serial.println(" ");
      }
      delay(250);
    }
  }
```

Autorización de publicación de videos e imágenes

Córdoba, 15 de septiembre de 2017

……..., DNI…… en calidad de PARTICIPANTE, en presencia de……, DNI…….. en calidad de TESTIGO, por el presente documento autorizo y expreso mi libre conformidad para publicar imágenes y videos de las pruebas realizadas en el PROYECTO: "Domótica aplicada a un paciente con discapacidad motriz" como parte de un proyecto integrador de la carrera de Ingeniería Biomédica, en eventos y publicaciones de divulgación científica.



---------- ----------

Firma del Voluntario Firma del Testigo

Arévalo, Santiago

---------- ----------

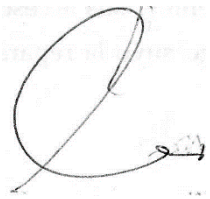
Aclaración Aclaración

Declaración de Consentimiento Informado

Córdoba, 15 de septiembre de 2017

……..., DNI ……. en calidad de PARTICIPANTE, en presencia de……, DNI …… en calidad de TESTIGO, por el presente documento declaro haber leído y comprendido la hoja de información adjunta, haber podido hacer preguntas, estar satisfecho con la información brindada por el INVESTIGADOR Libson, Lucas Martìn, DNI 35476230 y conocer que mi participación es voluntaria y conocer que puedo retirarme de la investigación en cualquier momento, sin que ello implique un perjuicio para mi persona.

Expreso mi libre conformidad para participar en las pruebas en el PROYECTO: "Domótica aplicada a un paciente con discapacidad motriz" como parte de un proyecto integrador de la carrera de Ingeniería Biomédica.



---------- ----------

Firma del Voluntario Firma del Testigo

Arévalo, Santiago

---------- ----------

Aclaración Aclaración

Hoja de Información de Investigación

Título de la Investigación:

‘‘Domótica aplicada a un paciente con discapacidad motriz’’ como parte de un proyecto integrador de la carrera de Ingeniería Biomédica

Investigadores:

Nombre y Apellido	DNI	Organización
Beltramone Diego Antonio	**22.371.253**	**Director**
Libson Lucas Martín	**35.476.230**	**Investigador**

Datos de contacto
Diego Beltramone – Director del proyecto
Teléfono celular: (0351) 155-731114
Correo Electrónico: diego.beltramone@unc.edu.ar

Libson Lucas Martín – Investigador
Teléfono celular: (0385) 154061974
Correo Electrónico: lucaslibson@gmail.com

Objetivo que se busca alcanzar el proyecto
La meta del presente proyecto consiste en facilitar las actividades de la vida diaria a personas que conviven con algún tipo de discapacidad motriz. Así también, permitirle mejorar su relación con su entorno social y contribuir en la mejora de la calidad de vida del usuario.
El objetivo entonces, es la de generar un dispositivo que logre incorporar conceptos del mundo de la domótica y que, a través de estos, le permita al usuario recuperar autonomía diaria. El artefacto se instalará en la edilicia donde el paciente reside y mediante la adquisición de órdenes, se activarán distintos actuadores que le permitan utilizar electrodomésticos de su entorno.

Etapas de la investigación:

1. Información al Voluntario
   1.1. Explicación verbal de los pasos a seguir durante la Investigación
   1.2. Firma del Consentimiento Informado
2. Evaluación de las habilidades remanentes que se podrían aprovechar para interactuar con el dispositivo domótico.
3. Análisis sobre los objetos a domotizar. Órdenes de utilidad para el Voluntario.
4. Implementación de lo concluido en el punto 2 y 3. Pruebas en Hardware y Software.
5. Evaluación de la performance del dispositivo en funcionamiento.
6. Consultas sobre cómo sintió el proceso y qué resultados se obtuvieron en el control.

Consideraciones

- Se lo invita a participar de la investigación de forma voluntaria, para lo que debe conocer que ningún procedimiento de los que se llevarán a cabo durante la investigación será invasivo.
- Se prevé que el Voluntario se comprometa con las reuniones pactadas en su edilicia para diagramar el proyecto.
- Todos los archivos en los que se identifique al Voluntario serán manejados con total confidencialidad, teniendo acceso a los mismos sólo el Voluntario, los investigadores y, en el caso que lo amerite y con el permiso del paciente, algún asesor externo.
- Solo con el consentimiento por escrito del Voluntario podrán presentarse los avances de la investigación en congresos, seminarios, etc.
- Al tratarse de una participación voluntaria, el Voluntario es libre de negarse a participar o retirarse de la investigación cuando lo desee, sin que ello implique un perjuicio para su persona.