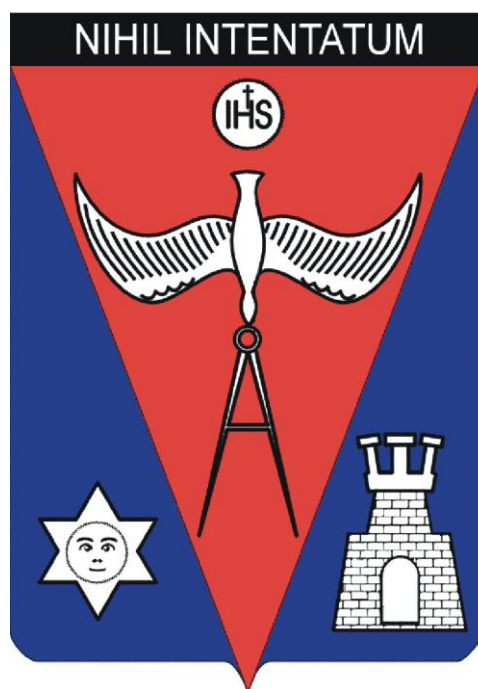


Robot de Telepresencia

Jeremías Frère

Ingeniería en Informática

Facultad de Ingeniería



UCASAL

2016

Agradecimientos y Dedicatoria

El presente trabajo está dedicado muy especialmente a mi familia: a mi madre, que mientras estuvo conmigo me ayudó incasablemente a lo largo de mi vida a volver a levantarme las muchas veces que tropecé, y a entender que a veces aquello que nos inquieta solo es necesario mirarlo desde una perspectiva más amplia para disolver la preocupación. Agradecer también a mi papá y hermana quienes me recordaron a cada paso mi fuerza, para recorrer este último tramo de la carrera.

Quiero agradecer particularmente a mi profesora guía, la doctora Alicia Pérez, quien con su enorme apoyo, confianza, e impecable dirección me ayudó en la realización de este trabajo; ayuda clave sin la cual no hubiera sido posible realizarlo.

Estoy muy agradecido con la Universidad, la cual fue realmente generosa en muchos momentos de mi carrera donde necesité sostén, y también me siento agradecido con todo el entorno que allí encontré: por el cariño que recibí de mis compañeros, las enseñanzas que me dejaron los profesores, el ejemplo brindado por los directivos, e inclusive el aliento sutil pero infaltable de nuestra querida secretaria Patricia, a quien le debo innumerables risas. También mencionar mis especiales gracias a la Ing. Guillermina Nievas.

Para terminar quiero agradecer eternamente a esos seres cercanos y queridos, que forman parte de mi realidad más íntima, gracias a los cuales he vivido las experiencias hasta ahora más lindas y significativas de esta vida, y que forjaron los ideales y convicciones que hoy forman parte de lo que soy. Agradecerles para siempre por permitirme ser como soy en mi grado más puro y transparente, a través de lo cual he aprendido a lo largo de los años a conocerme más y más a mí mismo, a todos/as ellos/as: ¡¡¡ Muchísimas Gracias !!!

Índice del Trabajo

Resumen del Trabajo	1
1. INTRODUCCIÓN	2
1.1 Descripción del problema y su importancia.....	2
1.2 Carácter del problema	2
1.3 Abordaje.....	3
1.4 Pasos realizados	3
1.5 Criterios de éxito.....	4
2. ESTADO DE LA CUESTIÓN.....	5
2.1 Antecedentes relacionados con el problema	5
2.2 Fundamentos teóricos y prácticos que cimientan la solución escogida.....	6
2.2.1 Telepresencia.....	6
2.2.2 iRobot Create.....	18
2.3 Trabajos relacionados	27
2.4 Desarrollo actual de las técnicas, herramientas, y otros recursos requeridos	36
2.5 Documentación y citas de soluciones al problema	40
3. DEFINICIÓN DEL PROBLEMA	44
3.1 Definición del problema	44
3.2 Objetivos	44
3.3 Alcances o procesos involucrados	44
3.4 Restricciones o límites del trabajo	45
3.5 Recursos disponibles.....	45
3.6 Alternativas tecnológicas de soluciones al problema	46
3.6.1 Comunicación por video y audio.....	46
3.6.2 Comunicación entre robot y ordenador	48
3.6.3 Conexión entre ordenadores	51
3.6.4 Movimiento del robot	54
3.6.5 Solución escogida	66
4. SOLUCIÓN PROPUESTA.....	68
4.1 Establecer la solución aportada.....	68

4.1.1 Ordenadores	68
4.1.2 Robot Create.....	71
4.1.3 Programa “iRobot Create Control”	71
4.2 Análisis económico y financiero.....	75
4.2.1 Costos de Hardware	76
4.2.2 Costos de Software.....	76
4.2.3 Costos de Recursos Humanos	77
4.2.4 Costo total	77
4.3 Análisis de los riesgos.....	78
5. VERIFICACIÓN EXPERIMENTAL.....	80
5.1 Primeros pasos	80
5.2 Pruebas realizadas	81
5.2.1 Navegación del robot desde Ordenador B.....	81
5.2.2 Conexión entre ordenadores a través de una red local	82
5.2.3 Conexión entre ordenadores a través de una red externa	84
5.2.4 Navegación del robot desde Ordenador R.....	88
5.2.5 Navegación del robot con videollamada en simultáneo	89
6. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN.....	96
7. BIBLIOGRAFIA.....	97
ANEXOS	101
Anexo 1. Código del software “iRobot Create Control”	101
Anexo 1.1. Archivos del formulario.....	101
Anexo 1.2. Clases del programa iCC	137
Anexo 2. Manual de uso: Programa “iRobot Create Control”	157
Anexo 3. Diagrama de GANTT.....	159
8. GLOSARIO.....	161

Índice de Tablas

Tabla 1 – Costos de Hardware.....	76
Tabla 2- Costos de Software.....	76
Tabla 3- Costos de Recursos Humanos.	77
Tabla 4- Tabla Final de Costos.....	77
Tabla 5 - Matriz de análisis de riesgos.	79
Tabla 6 - Tabla de equivalencias de probabilidad.	79

Índice de Ilustraciones

Ilustración 1 – Diagrama de Telepresencia	3
Ilustración 2 – Esquema de un sistema de teleoperación (Galiana, 2015)	8
Ilustración 3 – Operador dirigiendo al robot a través de una interfaz (Murphy, 2000).....	9
Ilustración 4 – Robot de telepresencia “Ava 500” de la empresa iRobot. (iRobot, 2015)	11
Ilustración 5 – Robot de telepresencia “TeleMe” de MantaroBot (MantaroBot, 2015).....	13
Ilustración 6 – Robot de telepresencia “VGo” de Vgo Communications. (Vgo, 2015).....	14
Ilustración 7 – Robot “QB” de Anybots (Anybots, 2014).....	15
Ilustración 8 – “BeamPro” de Suitable Technologies (Suitable Technologies, 2015).	16
Ilustración 9 – Robot Create, vista superior (iROBOT, 2015).....	19
Ilustración 10 - Botones y luces LED del robot Create (iROBOT, 2015).....	19
Ilustración 11 – Vista inferior del robot Create (iROBOT, 2015).	20
Ilustración 12 – Tabla con descripción de los pines del <i>Cargo Bay Connector</i> (iROBOT, 2015).....	21
Ilustración 13 – Dispositivo BAM conectado al robot Create de la Universidad	22
Ilustración 14 – Dispositivo Bluetooth BAM.....	23
Ilustración 15 – Combinación de los elementos necesarios para comunicación inalámbrica con el Create desde una computadora. (Element Direct, 2015)	23
Ilustración 16 – Indicación para conectar el elemento BAM al robot Create. (Columbia University, 2015).....	24
Ilustración 17 – Ventana de diálogo que se muestra al seleccionar la opción “Agregar un dispositivo”. (Columbia University, 2015)	25
Ilustración 18 – Código de seguridad para el Elemento Serial Bluetooth. (Columbia University, 2015).....	25
Ilustración 19 – Propiedades del Elemento Serial Bluetooth. (Columbia University, 2015).....	26
Ilustración 20 – Topología del "Ornitorrinco in Eden", 1994. (Kac, 1994)	28
Ilustración 21 – Participantes remotos, compartiendo el cuerpo del robot simultáneamente y visualizando la instalación a través de su visor. (Kac, 1994)	28
Ilustración 22 – Telerobot Ornitorrinco. (Kac, 1994)	29
Ilustración 23 – Rara Avis. (Kac, 1996).....	30
Ilustración 24 – Cabeza del telerobot Macowl (Kac, 1996).....	30
Ilustración 25 – Vista de la instalación desde afuera (Kac, 1996).	31
Ilustración 26 – Instalación vista desde el telerobot (Kac, 1996).	31
Ilustración 27 – Participantes remotos conectados al visor del Macowl a través de Internet. (Kac, 1996).....	32
Ilustración 28 – Parte superior del VGo (Vgo, 2015).....	33
Ilustración 29 – Parte inferior del VGo. (Vgo, 2015).....	33
Ilustración 30 – Aplicación VGo App corriendo en pantalla. (VGo, 2015).....	34
Ilustración 31 – Johnny Lee junto a su herramienta (Nathan, 2011).....	35
Ilustración 32 – Pantalla de inicio de Skype (Neetguias, 2013).....	37

Ilustración 33 – Imagen de una Arquitectura de un Sistema de Telepresencia (RoboSavvy, 2015).	41
Ilustración 34 - Devon Carrow en la pantalla del VGo (Comercio, 2013).....	42
Ilustración 35 – Interfaz de Google Hangouts para videollamada (Strategic Content, 2014).....	47
Ilustración 36 – Pantalla para iniciarse en Video Calling de Facebook (Kainth, 2011).....	48
Ilustración 37 - Conector Mini-DIN del robot Create (iRobot, 2015).....	49
Ilustración 38 - Cable Mini-DIN a Puerto Serie de iRobot. (Vic7767, 2015).....	49
Ilustración 39 - Dispositivo RooWifi. (RooWiFi, 2013).....	50
Ilustración 40 - Robot Create 2 de la empresa iRobot (iRobot, 2015).	51
Ilustración 41 – Programa TeamViewer en su versión gratuita de uso no comercial (TeamViewer, 2015).....	52
Ilustración 42 – Imagen del programa AnyDesk (AnyDesk, 2015)	53
Ilustración 43 – Imagen del programa Ammyy Admin (Ammyy, 2015).....	53
Ilustración 44 – Programa iRobot Create Control.	55
Ilustración 45 - Terminal de mensajes del programa iCC.	56
Ilustración 46 - Programa iCC, pestaña "Remote Robot".	57
Ilustración 47 - Terminal del Ordenador R controlando al robot a través de red local.	58
Ilustración 48 - Interfaz en Ordenador R mostrando velocidad de las ruedas del robot.	59
Ilustración 49 - Datos de Conexión Remota.....	60
Ilustración 50 - Terminal de Windows, comando "ipconfig" ejecutado	61
Ilustración 51 - Menú de configuración del modem Cisco DPC2320.....	62
Ilustración 52 - Sección de Port Forwarding del modem Cisco DPC2320.	63
Ilustración 53 – Pestaña del Joystick del programa iCC.	65
Ilustración 54 - Diagrama del Robot de Telepresencia	68
Ilustración 55 – Ordenador notebook local (Ordenador B).....	69
Ilustración 56 – Características del ordenador local.....	69
Ilustración 57 – Ordenador notebook remoto (Ordenador R).	70
Ilustración 58 – Captura del código donde se ocasionaba la primera falla	72
Ilustración 59 - Captura del código modificado para solucionar la primera falla.	73
Ilustración 60 - Captura del código donde se producía la segunda falla.	74
Ilustración 61 - Captura del código una vez implementada la solución de la segunda falla.	75
Ilustración 62 - Robot Create durante la prueba de navegación.....	82
Ilustración 63 - Ordenador B recibe petición y establece conexión con Ordenador R.	83
Ilustración 64 - Ordenador R conectado al ordenador B.	84
Ilustración 65 - Ordenador R activando <i>port forwarding</i> en el ordenador B a través de TeamViewer, para aceptar conexiones entrantes desde el Programa iCC.	85
Ilustración 66 - Conexión establecida entre ordenadores R y B.....	86
Ilustración 67 - Ordenador B siendo controlado.	86
Ilustración 68 - Red de Internet utilizada para la prueba de conexión entre ordenadores.....	87
Ilustración 69 - Router-Modem WiFi utilizado para estas pruebas.....	88
Ilustración 70 – Robot Create y ordenadores involucrados en la prueba.	90

Ilustración 71 - Ordenador R previo a la conexión.	91
Ilustración 72 - Ordenador B previo a la conexión.	92
Ilustración 73 - Ordenador B esperando conexión entrante.	93
Ilustración 74 - Ordenador B siendo controlado	94
Ilustración 75 - Ordenador R controlando robot Create	94

Resumen del Trabajo

El siguiente proyecto de grado trata sobre la obtención de una herramienta de telepresencia mediante el uso de un robot móvil programable, gracias al cual se brindará al usuario de la herramienta la posibilidad de hacerse presente en un lugar físico remoto.

En el capítulo 1 de este proyecto de grado se describe la problemática detectada, así como el enfoque desde donde se lo aborda, lo cual es importante tener en cuenta a la hora de decidir los resultados que se busca obtener, considerando además el perfil investigativo del mismo.

En segunda instancia y ya entrando de lleno en esta investigación, se ofrece un marco teórico sobre la telepresencia junto a los diferentes campos que la conforman, y en particular: los robots de telepresencia. Se detallan diferentes robots desarrollados en la actualidad, y a la par se da un pantallazo del mercado que existe alrededor de los mismos, y las soluciones que brindan a sus usuarios en el día a día. Este panorama se acompaña con un análisis que detalla las diferentes tecnologías de las que se puede disponer para la obtención de un robot de telepresencia, y también un estudio financiero de estas, que desemboca en distintas alternativas para la obtención de una herramienta de este tipo.

A partir de los recursos que se encuentran disponibles de entre estas alternativas, se detalla el diseño y la realización de pruebas que gradualmente permitieron al alumno llegar a una satisfactoria obtención de un robot de telepresencia. Además, dichas pruebas se diseñaron de manera tal que el lector se sienta igualmente capaz de realizarlas en su propio entorno, así como también permitirle a través del análisis desarrollado, elegir por sí mismo las tecnologías a utilizar.

Se concluye con un pensamiento final sobre los aspectos más destacables de la telepresencia en la actualidad, que a la vez son aquellos a los cuales este trabajo trata de dar visibilidad, abriendo paso a nuevas ideas y posibilidades concretas, que aporten soluciones a la vida cotidiana de las personas.

1. INTRODUCCIÓN

1.1 Descripción del problema y su importancia

La necesidad de la gente hoy en día, para comunicarse con personas en otros lugares, a veces a grandes distancias, cada vez son mayores y a la par de este crecimiento es necesaria la tecnología que pueda cumplir con las expectativas de comunicación de las personas. Así como sucedió en el pasado, las nuevas tecnologías de la información redefinen la experiencia humana y hoy en día, la tecnología televirtual en general es la que está reordenando nuestro campo de acción e interacción, en todas las esferas públicas.

“La creciente tendencia hacia el teletrabajo y la externalización, ha impulsado a las empresas a buscar formas para que tanto los equipos de personas como los trabajadores remotos, se comuniquen y colaboren con mayor eficiencia y efectividad. Esta necesidad ha inspirado a un puñado de empresas innovadoras de alta tecnología a crear un nuevo mercado: los robots de telepresencia a distancia.” (Lehrbaum, 2013)

Un robot de telepresencia permite que una persona se mueva virtualmente en una ubicación distante mediante el control de un robot móvil, equipado con un medio audio-visual que lo muestra en vivo, mediante video y sonido.

1.2 Carácter del problema

El mundo se hace cada vez más complejo, y tanto los negocios como las relaciones interpersonales se dan y crecen cada vez a mayor velocidad, convirtiendo en una prioridad estratégica y fundamental lograr la colaboración entre las personas. Esto hace que sea importante ampliar los recursos para coordinar y mantener la relación cara a cara, con un número cada vez mayor de colegas, clientes, socios, empleados, etc.

Considerando el gran caudal de información que fluye gracias a Internet y la rapidez con la que las comunicaciones se llevan a cabo hoy en día, es clara la necesidad de la gente para hacerse presente, real o virtualmente, en diferentes lugares para sus actividades profesionales y de negocios. Debido a esto, es cada vez más ansiada, la nueva tecnología que permita a las personas minimizar las distancias que tienen que recorrer, para concretar sus respectivas actividades.

“Hay una necesidad creciente en cuanto a la tecnología que permite la participación a distancia en las actividades profesionales y personales, debido en parte a las congestionadas carreteras y las vías aéreas, y un imperativo ambiental para reducir el consumo de combustibles fósiles. Por otra parte, para algunas personas, como los individuos con movilidad reducida, la participación a distancia es esencial para poder trabajar y permanecer profesionalmente activos.” (Herring, 2013)

Claramente los usos de la telepresencia no se limitan solo a ámbitos empresariales o laborales, sino que puede haber muchas formas más de implementarla según las necesidades y requisitos que se presenten, pero una cosa cierta que no podemos dejar de lado, es que disponer de

una herramienta interactiva como la aquí propuesta, implicaría grandes ventajas para la vida de las personas que la utilicen.

1.3 Abordaje

Un robot de telepresencia es un excelente ejemplo para mostrar una aplicación práctica de esta tecnología, y mostrar también cómo la telepresencia puede facilitar las tareas cotidianas de una persona que desea mantenerse activa en sus negocios y relaciones sociales, no solo en su localidad particularmente, sino también a nivel mundial.

La Ilustración 1 muestra un diagrama simplificado de lo que implica un robot de telepresencia.

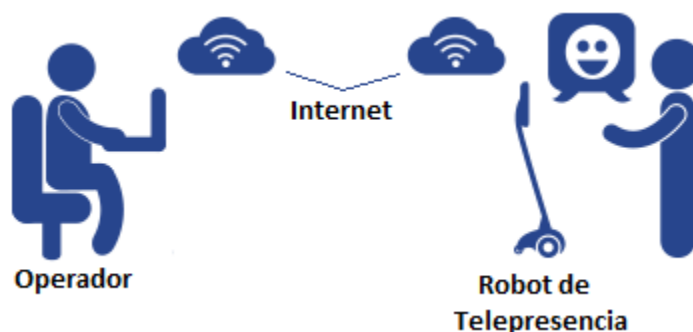


Ilustración 1 – Diagrama de Telepresencia

Aquí vemos entonces, cómo un operador desde su ubicación se conecta al robot de telepresencia, haciéndose presente y pudiendo interactuar con personas que se encuentran en donde está el robot.

En este trabajo se propone realizar un robot de telepresencia, haciendo uso de la herramienta que la Universidad dispone, el robot “Create” de la empresa iRobot, que funcionará como la base móvil que permita al usuario tanto moverse virtualmente por un espacio determinado enviando él mismo las órdenes de movimiento al robot, cómo hacerse presente a otras personas, mediante voz e imagen.

1.4 Pasos realizados

A continuación se detallan brevemente cuáles son las principales tareas que se describirán a lo largo de este trabajo:

- Averiguar sobre los antecedentes documentados del problema, así como las tecnologías y soluciones existentes usadas hoy en día para esta clase de trabajos.
- Investigar sobre herramientas que permitan el control de robots, a fin de realizar todas las

pruebas necesarias.

- Familiarizarme con la herramienta física, el robot Create.
- Conocer las maneras en que ya se ha tratado el problema para tener en cuenta entre otras cosas, que problemas quedan sin resolver, o los puntos que varían entre una solución y otra.

1.5 Criterios de éxito

Fue vital para el avance de este proyecto haber realizado las tareas y pruebas que se relacionan con la comunicación de la estación local con el robot, lo cual permitirá el movimiento de la herramienta que aquí se propone mostrar.

Dado este punto clave; y también dada la facilidad que permiten hoy en día las herramientas de videoconferencia, para lograr una comunicación audio-visual entre la estación local y la estación remota, para que la persona que se conecta remotamente sea vista y escuchada, se considerará exitoso este proyecto si se logra mostrar que un ordenador diferente al montado sobre el robot Create, puede manipular el movimiento del mismo mediante comandos sencillos que lo harán avanzar, retroceder y rotar sobre su propio eje, en ambos sentidos, dentro de la habitación donde se encuentre la herramienta.

2. ESTADO DE LA CUESTIÓN

2.1 Antecedentes relacionados con el problema

Gracias a los avances en la tecnología, hoy en día las personas pueden trabajar de forma remota, e inclusive tener reuniones con gente, que está del otro lado del mundo.

Este es el caso que se da con la tecnología de videoconferencia (por ejemplo, Skype, Google Talk, Hangouts) que hemos visto crecer año a año desde hace ya un tiempo. Con estas herramientas las personas no sólo logran comunicarse, sino que pueden verse el uno al otro mientras lo hacen. En algunos casos, un conjunto de cámaras son utilizadas para crear una imagen panorámica de una habitación, lo que permite al usuario del otro lado, ver todo el salón al mismo tiempo, e interactuar con varias personas a la vez. (Arthur, 2013)

La telepresencia es una tecnología que lleva la videoconferencia a otro nivel y la hace aún más realista y avanzada, además de tener una mayor oferta de detalles. Cabe señalar que el uso de dispositivos tecnológicos actualizados, como ser monitores y cámaras de alta definición, y sistemas de audio con supresión de ruidos colocados especialmente para escuchar a las personas del lado que realmente están ubicadas, aportarán a que la herramienta sea sumamente útil, eficiente y de mayor calidad.

Si bien las herramientas de videoconferencia cumplen ampliamente sus cometidos en cuanto a la comunicación más que nada personal, hay otros aspectos que simplemente no puede contemplar, y es ahí donde toma riendas la telepresencia. En la Universidad Católica de Salta, actualmente no existe implementación alguna de este tipo, más allá de que las personas puedan usar en sus ordenadores algún software como por ejemplo Skype, u otros.

El principal objetivo de este proyecto es abrir paso a la idea de que disponiendo de tan rica y explotable herramienta como lo es el “Create”, y como lo sería un robot de telepresencia, se pueden expandir los límites que a las personas se les presentan en su vida cotidiana a la hora de trasladarse de un lugar a otro, incluso casi, borrando estos límites que les impiden a veces, atender a todas las actividades en las que se involucra cada una de ellas.

Un claro ejemplo de esta expansión de los límites, sería la posible aplicación de este tipo de herramientas en la Universidad. Lo cual permitiría pensar en nuevas alternativas a la hora de brindar servicios y educación a sus alumnos; de la manera en que lo sería un robot de telepresencia de esta clase para utilizar con variadas finalidades, ya sea desde prácticas y enseñanza con la herramienta, hasta brindar acceso remoto a alumnos que por algún motivo se encuentran imposibilitados para asistir a clases presencialmente, y que de esta manera puedan asistir a las mismas a través de dicha herramienta. Incluso a la hora de realizarse conferencias o actos universitarios, la herramienta permitiría que algún directivo o invitado importante, pueda hacerse presente en la universidad, y hasta dar un discurso para dicho evento, desde la ubicación lejana en la que se encuentra, la cual le hubiera impedido llegar al establecimiento si no es a través del robot de telepresencia.

Básicamente, la telepresencia es un sistema de videoconferencia más avanzado, que se diferencia especialmente en brindarle al usuario la libertad para moverse en el lugar remoto que se hace presente, además de que evita desplazamientos innecesarios, ahorra tiempos improductivos, acelera los procesos de decisión, mejora la comunicación y reduce las emisiones de dióxido de carbono (CO₂), como las que producen los desplazamientos por carretera o avión, entre algunas de sus ventajas a nivel general. (Trends, 2010)

A grandes rasgos, algunas ventajas claras de la telepresencia que podemos apreciar también son:

- Reducción de gastos
- Ahorro de tiempo
- Aceleración de la toma de decisiones (eficacia y competitividad)
- Mayor y mejor comunicación
- Mejor imagen de empresa
- Mayor fidelidad y capacidad de servicio con terceros

Así podemos ver cómo en un robot de telepresencia se integra la tecnología multimedia de interacción visual y auditiva, junto con un dispositivo controlado remotamente, que permite la movilidad de estos elementos, logrando que el usuario se haga presente en un lugar, y elija él mismo a quién o qué le prestará atención.

2.2 Fundamentos teóricos y prácticos que cimientan la solución escogida

2.2.1 Telepresencia

Hacia la década de los 80, ya surgía en los ámbitos científicos el término de *telepresencia* y se la dejaba de considerar como ciencia ficción, para empezar a ser tomada como una posibilidad real en el mundo. Una posibilidad para poder lograr esa sensación de “estar allí presente” (Minsky, 1980). Allí presentes, en un lugar al que debemos ir y que, no siempre es tan fácil llegar; ya sea por temas de distancias, tiempos, obligaciones que se interponen, lugares de difícil acceso, o incluso lugares en los que sólo es posible acceder a través de la tecnología.

Wikipedia define Telepresencia de la siguiente manera (Wikipedia, 2011):

“La telepresencia hace referencia a un conjunto de tecnologías que permiten que una persona se sienta como si estuviera presente en un lugar que no es su ubicación real, además dan la apariencia de que esa persona está allí presente, y permiten inclusive que pueda producir un efecto en ese lugar, a través de telerrobótica.

Esto requiere que los sentidos de los usuarios sean estimulados de tal manera que tengan la sensación de estar en ese otro lugar. Además, los usuarios pueden tener la capacidad de producir efectos en la ubicación remota. Para lograr este efecto, su posición, las acciones, la voz del usuario, los movimientos, etc. son detectados, transmitidos y duplicados en la ubicación remota.

Por lo tanto, la información estaría viajando en ambas direcciones entre el usuario y la ubicación remota.”

La investigación científica sobre telepresencia se ha centrado principalmente en la telerrobótica y la teleoperación de robots, para permitir un nuevo nivel de interacción entre seres humanos y ordenadores, buscando un desarrollo como el que han alcanzado las tecnologías comerciales de realidad virtual.

En cuanto a las diferencias entre telepresencia y realidad virtual, podemos verlas respecto a sus procesos: por un lado la realidad virtual hace perceptivamente real lo que de hecho sólo tiene una existencia virtual (es decir, digital). Por el contrario, la telepresencia transporta a un individuo de un espacio físico a otro, a menudo a través de una conexión de telecomunicaciones. Las telecomunicaciones y la robótica pueden unir la transmisión y la recepción de señales de control de movimiento, con el feedback audiovisual, táctil y de fuerza. La telepresencia hace virtual algo que en realidad tiene una presencia física y tangible. Presencia que es generalmente remota al lugar en donde se encuentra la herramienta de telepresencia (Kac, 1994).

Así como es bueno ver las diferencias entre telepresencia y realidad virtual, en este punto también es útil establecer de qué tratan algunos términos que, si bien se encuentran estrechamente relacionados con la telepresencia, no la definen sino que más bien la conforman; algunos de ellos son:

2.2.1.1 Telerrobótica

Es el área de la robótica que se dedica a la operación a distancia de robots (FreeDictionary, 2015). Se trata de un conjunto de técnicas y estudios que permiten la manipulación de robots ubicados en lugares físicamente distantes a donde se encuentran los operadores, lo cual también les brinda mayor seguridad a estos. Por ejemplo, este tipo de tecnología ha tenido un gran impacto en el área de la medicina ya que posibilita brindar un mejor servicio, en comparación al uso de los dispositivos tradicionales; aun así no es el único ámbito donde estas herramientas robóticas demuestran sus beneficios, ya que también han fomentado grandes avances en otras áreas que van desde la geología, con robots que permiten la exploración de sitios fuera del alcance de las personas, hasta la astronomía y la arqueología subacuática donde brindan plena seguridad a los investigadores, a la hora de estudiar las atmósferas hostiles de planetas lejanos, o las profundidades de los océanos; lugares a los cuales sería imposible acceder de no ser por la existencia de esos robots (Monreal, 2013) (Gamero, 2008).

La telerrobótica, en internet, es definida por Wikipedia de la siguiente manera (Wikipedia, 2015):

“La telerrobótica es el área de la robótica concerniente al control de robots desde la distancia, principalmente usando conexiones inalámbricas como Wi-Fi, Bluetooth y similares, o a través de Internet. Es una combinación de dos campos importantes que son, teleoperación y telepresencia”.

Justamente estos dos campos, aplicados al uso de los robots, son los que también dan forma, y conforman, al término de telepresencia, con lo que éste abarca y representa en la actualidad.

2.2.1.2 Teleoperación de robots

La teleoperación de robots, al ser uno de los campos que conforman la telerrobótica, se define muy similarmente a ésta, ya que concretamente hace referencia, a la realización del trabajo a la distancia. Es sabido que a casi cualquier tarea que nos propongamos hacer se le puede llamar “trabajo”, y que el término “distancia” también es vago, ya que puede aludir tanto a una distancia física que separa a la herramienta robótica de su operador, como a los cambios de escalas que ocurren por ejemplo en intervenciones quirúrgicas, donde los cirujanos usan herramientas robóticas de micro-manipulación para dirigir cirugías a nivel microscópico. (Wikipedia, 2015)

La connotación principal de la teleoperación, es que la distancia que separa al robot del operador, es demasiado grande para que éste pueda ver lo que el robot está haciendo, por lo que un auto de juguete a control remoto por ejemplo, no se considera como un sistema tele-operado. El operador y el robot mantienen un tipo de relación llamado “maestro-esclavo” como el que se puede ver en la Ilustración 2. En la mayoría de los casos, el operador humano se ubica en una estación de trabajo, desde la cual dirige al robot a través de algún tipo de interfaz, como se detalla en la Ilustración 3.

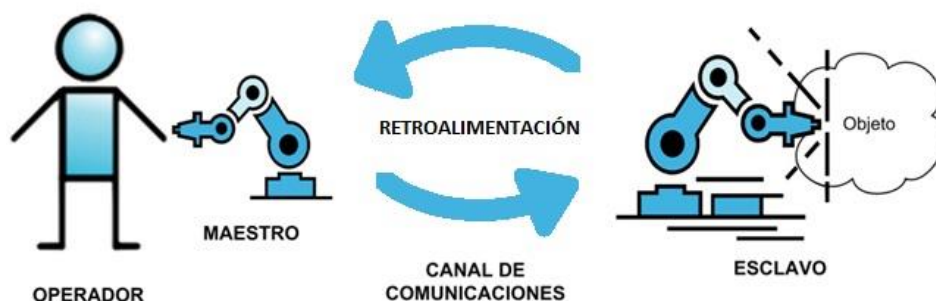


Ilustración 2 – Esquema de un sistema de teleoperación (Galiana, 2015)

La interfaz de control siempre varía de acuerdo al trabajo que se tiene que realizar y a la implementación con la cual se lo va a resolver, pero sin embargo, si se presta atención existe un punto en común para la mayoría de estas herramientas, y es que casi siempre, el robot maestro que el operario maneja es muy similar al robot esclavo que está recibiendo órdenes y ejecutándolas en simultáneo. Esto brinda al operario cierta sensación de estar manejando directamente la herramienta real, aunque solo se trate de una réplica a través de la cual enviar los correspondientes comandos.



Ilustración 3 – Operador dirigiendo al robot a través de una interfaz (Murphy, 2000)

De acuerdo a lo que postula Charles Wampler en su trabajo “Teleoperadores, Control de Supervisión”, la teleoperación se adecúa mucho mejor a situaciones en las que (Murphy, 2000):

1. Las tareas no son ni estructuradas, ni repetitivas.
2. El espacio de trabajo no puede diseñarse para permitir el uso de manipuladores industriales.
3. Porciones clave del trabajo requieren intermitente destreza de manipulación, especialmente la coordinación de ojos y manos.
4. Parte importante del trabajo requiere reconocimiento de objetos, conocimiento de la situación, o de otro tipo de percepción avanzada.
5. Las necesidades de la tecnología de interfaz no excede las limitaciones del canal de comunicación (ancho de banda, retardos de tiempo).
6. La disponibilidad de personal capacitado no es un problema.

2.2.1.3 Robots de telepresencia

En cuanto a la telepresencia, este campo ya se ha convertido en un hecho en la actualidad, y son las empresas, quienes principalmente se han visto obligadas a considerarla para beneficiarse de su uso, por ejemplo para las reuniones de sus equipos de trabajo. Suele pasar a menudo, que dichas reuniones no se den como es esperado, siendo cada vez más frecuente que alguno de los integrantes sufra contratiempos al momento de trasladarse al punto de reunión, y esto se traduce en una pérdida de tiempo sumamente valioso, que también compromete a los demás involucrados.

En un artículo de 2013, la página El Comercio publicó una recopilación de casos y situaciones, en las que las tecnologías de telepresencia han solventado distintas problemáticas, como ser la de Devon Carrow, un niño estadounidense que debido a sus alergias no puede asistir a la escuela, y lo hace a través de la herramienta “VGo” (problemática que se detallará en la sección 2.5), o como ser el caso de chequeos clínicos realizados por médicos, que a través de estos robots se hacen presentes virtualmente en lugares remotos y atienden a sus pacientes (Comercio, 2013).

Además de estas aplicaciones, empresas como ISERTEC o CISCO, hoy en día se caracterizan por ofrecer la implantación de servicios y salas de telepresencia, disponiendo de tecnología de punta para el ofrecimiento de diferentes paquetes amoldables a cada cliente. Entre los distintos tipos de ofertas, encontramos propuestas como la telepresencia inmersiva, donde los participantes de la reunión se muestran con dimensiones reales y se podrá hablar y observar el lenguaje corporal de los otros como si estuviera en la misma sala (Isertec, 2013) (Cisco, 2015).

Por otro lado tenemos la telepresencia de sala, donde la solución apunta hacia una implementación más amplia y donde las decisiones se puedan tomar de manera más informada sin dejar de lado a nadie. O en todo caso se dispone de soluciones para telepresencia personal, llevando el video de alta definición a ordenadores portátiles, robots, móviles, hogares, etc.

Las tecnologías de Polycom en telepresencia, o herramientas como el Servidor de TelePresencia Cisco, el Conductor de TelePresencia Cisco o el Servidor de Contenido de TelePresencia, también de la empresa Cisco, son algunos de los dispositivos que se pueden tener en mente a la hora de implementar tecnología que apunte a brindar soluciones empresariales. (Cisco, 2015)

Sin embargo, hoy en día es posible el acceso a tecnologías menos costosas y rentables, como las que ofrece iRobot, con las cuales cualquiera puede utilizar elementos del hogar y valerse de la asistencia de Internet para diseñar su propia estación de telepresencia.

Entre los productos que actualmente se encuentran en el mercado nos encontramos con el “Ava 500”, de la empresa iRobot, que anunció dicho lanzamiento en un artículo con la siguiente información (Ortiz & Orellana, 2013):

“iRobot anuncia un robot de colaboración que incluye la tecnología TelePresencia de Cisco:

La compañía líder en soluciones de tecnologías robóticas, iRobot, anunció en la feria InfoComm de Orlando (Florida, EE. UU.), que está trabajando con Cisco para lanzar al mercado un robot de colaboración mediante vídeo de clase empresarial, el “iRobot Ava 500”.

Se trata de un robot que combina las tecnologías de navegación autónoma de iRobot y las tecnologías de TelePresencia de Cisco para ayudar a los profesionales remotos a participar en reuniones y presentaciones en lugares donde la movilidad resulta importante. El nuevo modelo también está diseñado para proporcionar un acceso visual móvil en plantas de fabricación, laboratorios, centros de pruebas y otras ubicaciones remotas de la cadena de suministro.”



Ilustración 4 – Robot de telepresencia “Ava 500” de la empresa iRobot. (iRobot, 2015)

En la Ilustración 4 se puede ver una ilustración del Ava 500 en acción, lo que permite obtener ciertos beneficios que iRobot se ha encargado de analizar, estudiar y definir, a continuación se detallan esos beneficios (iRobot, 2015):

- Permite la programación flexible e inmediata de reuniones "cara a cara".
- Habilita la colaboración entre personas y equipos de todo el mundo.
- Permite reuniones en áreas abiertas.
- Proporciona la capacidad de tener conversaciones fuera de las habitaciones.
- Elimina la necesidad de viajes costosos en tiempo y dinero.
- Aumenta las comunicaciones y la transferencia de conocimiento.
- Apoya la toma de decisiones oportuna.
- Permite una mayor productividad para los gerentes y empleados.

Estas son algunas de las principales ventajas que se pueden lograr con la tecnología de telepresencia, especialmente cuando se la enfoca en brindar soluciones a nivel empresarial y laboral. Aun así, ciertamente sus aplicaciones y beneficios no terminan aquí, sino que dependerán del problema que se trate de solventar, y el ámbito en el que se las implemente.

La ficha de datos del recién mencionado Ava 500, brindada específicamente por iRobot, se puede ver y descargar en su página de internet en formato PDF¹; allí se detallan las especificaciones y características de esta herramienta.

Este sistema se encuentra disponible en el mercado hoy en día con un precio aproximado de U\$S 69.000 (TelepresenceRobots, 2015), por lo que dentro de este tipo de herramientas es una de las que ostenta el precio más elevado, principalmente debido a la tecnología de alta gama que utiliza, y a que admite varias formas distintas de conectividad.

Claramente iRobot no es la única empresa que ha detectado la necesidad de herramientas de este tipo en el mundo, y en los últimos años, entraron al mercado una gran variedad de opciones de robots de telepresencia, de otras empresas también especializadas en el tema como ser el robot “TeleMe” de MantaroBot, “VGo” de Vgo Communications, “QB” de Anybots, el robot “BeamPro” de Sutable Technologies, entre otros.

A continuación se puede apreciar información e imágenes de estas herramientas recién mencionadas:

1

<http://www.irobot.com/~media/Files/Robots/Commercial%20Applications/Ava%20500/iRobot-Ava-500-Data-Sheet.pdf>

“TeleMe” de MantaroBot

Ilustración 5 – Robot de telepresencia “TeleMe” de MantaroBot (MantaroBot, 2015).

El “Teleme TelePresence Robot” de MantaroBot es una plataforma móvil donde se puede conectar un iPad, iPhone o Tablet Android personal. Esté conectado a Skype, o cualquier otra aplicación de conferencia de audio/vídeo, como FaceTime, Google Hangouts, Cisco, Polycom, Vido, etc. el TeleMe permite al usuario interactuar remotamente con compañeros de trabajo, asociados y su entorno. Este sistema de telepresencia que se muestra en la Ilustración 5 proporciona un nivel de interactividad que no es posible con la tradicional videoconferencia "call-in". El usuario puede moverse por la habitación y observar a los participantes a través de la cámara de la tableta, lo que permite una comunicación más efectiva. El usuario podrá ver las actividades en vivo, reuniones o mantener reuniones “cara a cara” (MantaroBot, 2015).

Esta herramienta se puede conseguir en el mercado por un monto de U\$S 1.650 (TelepresenceRobots, 2015), lo que la convierte en la más accesible de las que se detallarán aquí.

“VGo” de Vgo Communications

Ilustración 6 – Robot de telepresencia “VGo” de Vgo Communications. (Vgo, 2015)

VGo es una nueva solución de mejora de la productividad que permite a una persona replicarse a sí misma en un lugar distante y tener la libertad de moverse como si estuviera físicamente allí. Con VGo, se puede ver, oír, hablar, interactuar e ir a cualquier parte. Como vemos en la Ilustración 6, VGo no es una solución tradicional de videoconferencia, donde dos o más personas se encuentran utilizando habitaciones, o las computadoras habilitadas en sus oficinas. Con VGo, el usuario es completamente independiente de las demás personas en el lugar donde se hace presente - es 100% controlado a distancia. La persona no hace una llamada, directamente va allí. Su presencia se establece de forma remota a través de la presencia física de VGo (Vgo, 2015). Su precio ronda los U\$S 6.000 (TelepresenceRobots, 2015).

En la Sección 2.3, donde se habla de algunas herramientas del tipo propuesto que han sido realizadas, se describirá al VGo con mayor detalle.

“QB” de AnyBots

Ilustración 7 – Robot “QB” de Anybots (Anybots, 2014).

QB le lleva al instante donde tiene que estar. Permite al usuario moverse a través de espacios reales e interactuar con colegas y familiares a distancia. O simplemente explorar un nuevo espacio. QB posee una plataforma robusta, patentada, y equilibrada dinámicamente brindando una movilidad de amplio alcance. Sus principales características son (Anybots, 2014):

- **Altura** - El rango de altura ajustable del QB permite interactuar a la altura de la cabeza de la gran mayoría de las personas, como se puede ver en la Ilustración 7.
- **Tamaño** - QB tiene un tamaño reducido. Sólo ocupa tanto espacio como una persona.
- **Peso** - QB es ligero, sólo pesa 32 libras. QB es seguro en una multitud.
- **Movilidad** - Las grandes ruedas permiten al QB conducir sobre baches de hasta 2 centímetros.
- **Rampas** - QB asciende y desciende rampas para discapacitados.
- **Flexibilidad** – Posee asistencia de dirección a través de puertas y alrededor de obstáculos.

En cuanto al costo de esta herramienta, se puede adquirir por U\$S 9.700 (TelepresenceRobots, 2015).

“Beam” de Suitable Technologies

Ilustración 8 – “BeamPro” de Suitable Technologies (Suitable Technologies, 2015).

El BeamPro de la Ilustración 8, es el sistema premier de *Smart Presence*. Yendo más allá de los productos tradicionales de videoconferencia, BeamPro elimina las limitaciones de una pantalla mediante el acoplamiento de vídeo de alta calidad y audio, con la libertad de la movilidad. Esta combinación proporciona una experiencia enriquecedora que conecta a las personas de maneras que ninguna otra herramienta puede ofrecer (Suitable Technologies, 2015).

BeamPro se encuentra disponible en el mercado a un precio cercano a los U\$S 16.000 (TelepresenceRobots, 2015), lo que lo convierte en el segundo sistema más costoso de los aquí mencionados.

Las personas hoy en día, suelen trabajar en lugares distantes de donde se encuentran, y si bien se han manejado durante muchos años con herramientas dedicadas a la videoconferencia como Skype, Google Talk, Microsoft Lync y muchas otras aplicaciones al estilo; lo que no se ofrecía del todo, era una sensación de “estar presente”, o mejor aún, “estar allí moviéndose en el lugar”.

Si consideramos las ofertas de ciertas empresas que rentan sus salas de videoconferencia, podemos tomar como ejemplo la propuesta de una empresa mexicana llamada VITECH, que por una cuota de U\$S 100 (cien dólares estadounidenses) por hora más un 16% de IVA (VITECH, 2015) ofrece sus salas de videoconferencia, para el uso de otras personas; lo cual al principio podría parecer un precio y un servicio accesible, aunque si las necesidades aumentasen yuviésemos que realizar diariamente o semanalmente reuniones de este tipo con gente a distancia (que además por lo general suelen durar más que sólo una hora), comenzaríamos a notar que este tipo de soluciones no tiene tanto rédito a largo plazo como lo podría tener un robot de telepresencia. Aún si se quisiera montar una sala de videoconferencia propia para no tener esta clase de gastos por alquiler, cuando empezamos a notar la cantidad de tecnología que se debe invertir para lograr una sala de videoconferencia, sin contar además la necesidad de una habitación especialmente dedicada para colocar toda esa tecnología, también se llegaría a la conclusión de que es más rentable buscar en otros ámbitos la forma de solventar las necesidades que estas salas tratan de cubrir.

El Centro de Desarrollo Tecnológico de la Universidad de Cantabria (CDTUC) describe detalladamente toda la tecnología que ha invertido en sus salas de videoconferencia, con lo que nos podremos dar una idea del equipamiento que hace falta para esta clase de implementaciones; a continuación se listan dichos elementos (CDTUC, 2015):

- Pantalla de plasma de 50 pulgadas.
- Pantalla CRT plana de 29 pulgadas 100Hz.
- Dos cámaras robotizadas.
- Estación base de videoconferencia.
- Dos micrófonos omnidireccionales y uno direccional.
- Dos altavoces con equipo de amplificación y ecualización.
- Impresora-escáner.
- Dos ordenadores portátiles.
- Lector de DVD-vídeo VHS.
- Pizarra electrónica con impresora a color.
- Línea telefónica.

- Toma de red.
- Tres líneas RDSI.

Teniendo en cuenta esos elementos y haciendo cálculos de su costo, fácilmente veremos que se puede invertir tales cantidades de dinero, o aún menos, en una solución que provenga de un área de la tecnología diferente a la de videoconferencias.

Y aunque si bien no se podría comparar un robot de telepresencia con una sala de videoconferencias bien equipada y con tecnología de punta, sabiendo además que existe una gran diferencia entre las necesidades que cada uno de estos sistemas cubre, es justamente en los casos en que incrementa el flujo de información entre personas a distancia que la telepresencia adquiere su ventaja, principalmente por el bajo costo de sus herramientas junto a la flexibilidad de uso que brindan al piloto; dándole la libertad de conversar con quien quiera en la ubicación que se hace presente, antes que estar limitado a una habitación y un horario específicos.

2.2.2 iRobot Create

Create será el robot que se utilice para este trabajo, la Universidad adquirió uno de estos robots para el uso del actual equipo de robótica en la Facultad de Ingeniería, dirigido por la Dra. Alicia Pérez Abelleira. El mismo sirve actualmente como herramienta de trabajo para las investigaciones y experimentos que el equipo realiza, y tendrán la amabilidad de prestarlo para la muestra de la solución que en este trabajo se propone.

El robot aspiradora Roomba, de la empresa iRobot, es el predecesor del Create, que reemplaza el sistema de aspiradora del Roomba por una bodega de carga en su lugar; que tiene la particularidad de albergar un puerto DB-25 que proporciona comunicación serial, además posee entrada y salida tanto digital como analógica, y una fuente de alimentación eléctrica. (Wikipedia, 2015)

En la Ilustración 9 vemos una imagen de la vista superior del robot, con el detalle de los elementos que lo conforman. Estos son el receptor omnidireccional de señales infrarrojas (*Omnidirectional IR Receiver*), una manija para levantar el robot (*Handle*), un puerto (*Serial Port*) para conectar el robot a un ordenador mediante un cable serial, la ficha de carga (*Charging Socket*), y la bodega de carga (*Cargo Bay*), junto con su compuerta (*Tailgate*), el puerto DB-25 también conocido como *Cargo Bay Connector*, y 4 cavidades para ensamblar con tornillos alguna estructura (*Mounting Cavities*).

Top View

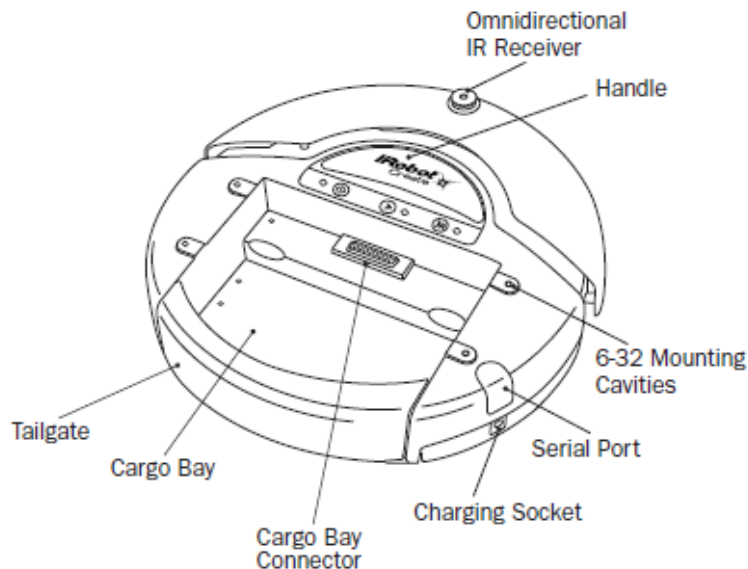


Ilustración 9 – Robot Create, vista superior (iROBOT, 2015)

Buttons and Lights

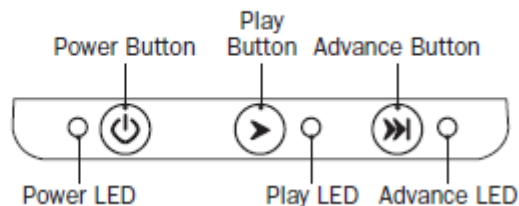


Ilustración 10 - Botones y luces LED del robot Create (iROBOT, 2015)

En la Ilustración 10 se visualizan los botones de encendido (*Power Button*), de inicio (*Play Button*) y de avance (*Advance Button*), con sus respectivos LED indicadores. Con el botón de avance se puede seleccionar algunas demostraciones que el robot trae ya instaladas, y con el botón de inicio, le damos la orden de que ejecute dicha demostración.

Más abajo en la Ilustración 11, la vista inferior del Create nos muestra el resto de los elementos que lo conforman. Las aperturas con los sensores de riscos (*Cliff Sensor Openings*), los puntos de contacto para su estación base (*Contact Points for Home Base*), la batería (*Battery*), dos ruedas cada una con su respectivo motor para controlarlas (*Wheel Clips*), y dos ruedas de menor tamaño que las anteriores, las cuales se usan para el equilibrio del robot; una frontal que está fija entre los puntos de contacto con su estación base, y una trasera (*Fourth Wheel*) puramente opcional que puede ser colocada a elección.

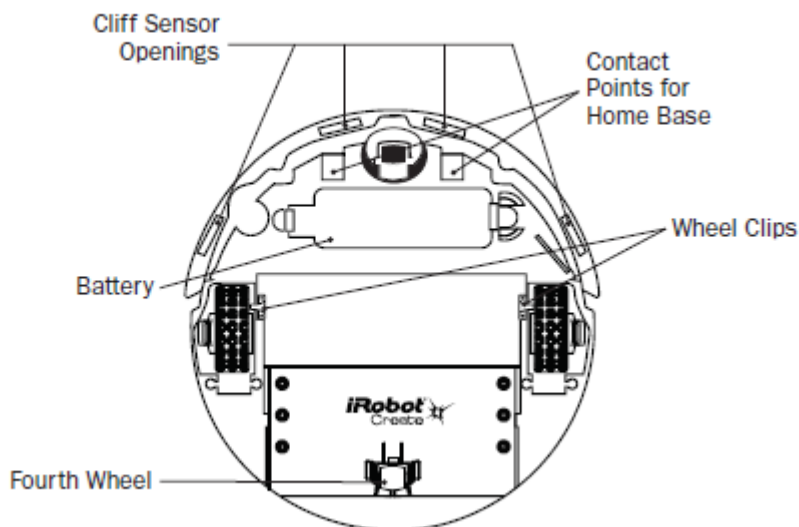
Bottom View

Ilustración 11 – Vista inferior del robot Create (iROBOT, 2015).

La plataforma iRobot Create es un robot móvil preensamblado y accesible, que proporciona a estudiantes, desarrolladores y educadores una oportunidad fuera de lo común para programar comportamientos, sonidos, movimientos y hasta agregar elementos electrónicos adicionales a la plataforma. El Create está marcando una diferencia tanto en el avance de la investigación robótica y el desarrollo de productos, como en la enseñanza de la Ciencia, la Tecnología, la Ingeniería y las Matemáticas (*STEM*). (iRobot Company, 2015)

El manual de usuario del Create, una guía en formato PDF que la empresa iRobot brinda desde su página oficial² contiene información detallada de interés para quienes adquieren esta plataforma.

En dicho documento encontramos por ejemplo, la descripción del *Cargo Bay Connector* de la plataforma, con la siguiente información al respecto.

Cargo Bay Connector

Este elemento, ubicado en la mitad frontal de la bodega de carga del robot, contiene 25 pines que se pueden utilizar para agregar al robot elementos electrónicos u otros dispositivos periféricos como sensores, luces o motores adicionales. (iROBOT, 2015)

² http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf

Pin	Name	Description
1	RXD	0 – 5V Serial input to iRobot Create
2	TXD	0 – 5V Serial output from iRobot Create
3	Power control toggle	Turns iRobot Create on or off on a low-to-high transition
4	Analog input	0 - 5V analog input to iRobot Create
5	Digital input 1	0 - 5V digital input to iRobot Create
6	Digital input 3	0 - 5V digital input to iRobot Create
7	Digital output 1	0 - 5V, 20 mA digital output from iRobot Create
8	Switched 5V	Provides a regulated 5V 100 mA supply and analog reference voltage when iRobot Create is switched on
9	Vpwr	iRobot Create battery voltage (unregulated), 0.5A
10	Switched Vpwr	Provides battery power @ 1.5 A when iRobot Create is powered on.
11	Switched Vpwr	Provides battery power @ 1.5 A when iRobot Create is powered on.
12	Switched Vpwr	Provides battery power @ 1.5 A when iRobot Create is powered on.
13	Robot charging	When iRobot Create is charging, this pin is high (5V)
14	GND	iRobot Create battery ground
15	BRC	0-5V digital input to iRobot Create which can also be used to change the baud rate to 19200 (see below)
16	GND	iRobot Create battery ground
17	Digital input 0	0 - 5V digital input to iRobot Create
18	Digital input 2	0 - 5V digital input to iRobot Create
19	Digital output 0	0 - 5V, 20 mA digital output from iRobot Create
20	Digital output 2	0 - 5V, 20 mA digital output from iRobot Create
21	GND	iRobot Create battery ground
22	Low side driver 0	0.5A low side driver from Create
23	Low side driver 1	0.5A low side driver from Create
24	Low side driver 2	1.5A low side driver from Create
25	GND	iRobot Create battery ground

Ilustración 12 – Tabla con descripción de los pines del *Cargo Bay Connector* (iROBOT, 2015)

En la Ilustración 12 se muestra una tabla que contiene etiquetados el nombre y una descripción para cada uno de los 25 pines que conforman el conector.

En este trabajo, se lo utilizará para insertar un dispositivo llamado Element Direct BAM (*Bluetooth Adapter Module*) que permitirá el control inalámbrico del robot, por parte de un ordenador. El mismo será descrito a continuación.

Element Direct BAM

Este dispositivo se conecta al *Cargo Bay Connector* de la bodega de carga del robot Create, el cual se describió antes, y habilita una comunicación inalámbrica vía bluetooth, que permite controlar al robot desde un ordenador Windows, Macintosh o Linx, sin necesidad de cables adicionales.

El elemento BAM provee una conexión de puerto serie virtual, entre un *host* (anfitrión) Bluetooth y el robot Create. De esta manera, un ordenador que también tenga habilitado un servicio de Bluetooth, se puede comunicar con el robot de la misma manera que lo haría si estuviera conectado por el cable serial.

Se trata de un dispositivo Bluetooth Clase 1 (alta potencia) mediante el cual se hace posible la comunicación del robot con un host de bluetooth, como ser una laptop o una PC de escritorio. Existen dos tipos básicos de hardware Bluetooth: los dispositivos Bluetooth, y los hosts Bluetooth. Un ordenador portátil es un ejemplo de un host Bluetooth. Un mouse inalámbrico, una impresora inalámbrica, y el BAM son ejemplos de dispositivos Bluetooth. Con el fin de conectarse al BAM, se debe disponer de un equipo que actúe como host Bluetooth. (Element Direct, 2015)

A continuación en la Ilustración 13 se muestra el robot Create perteneciente a la Universidad, junto con el dispositivo BAM insertado en su *Cargo Bay Connector*. Debajo de ésta la Ilustración 14 es un acercamiento a dicho dispositivo, y la Ilustración 15 es una representación de los elementos necesarios que se deben combinar, para que se establezca una comunicación inalámbrica entre el robot y un ordenador.



Ilustración 13 – Dispositivo BAM conectado al robot Create de la Universidad



Ilustración 14 – Dispositivo Bluetooth BAM

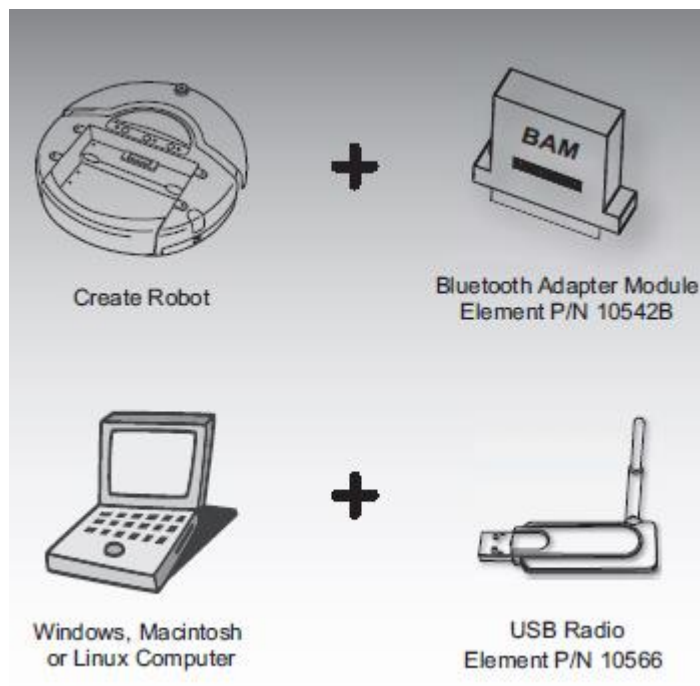


Ilustración 15 – Combinación de los elementos necesarios para comunicación inalámbrica con el Create desde una computadora. (Element Direct, 2015)

Una vez que se dispone de los elementos que se ubicaron en la Ilustración 15, se deben seguir estos pasos para que ambos sistemas se reconozcan entre sí (cabe aclarar que en este caso se explicará el proceso, asumiendo que se trata de un ordenador que utiliza Windows como sistema operativo) (Columbia University, 2015):

1. Conectar el dispositivo BAM en el *Cargo Bay Connector* como lo indica la Ilustración 16 (debe quedar insertado como se ve en la Ilustración 13), y luego encender el robot.

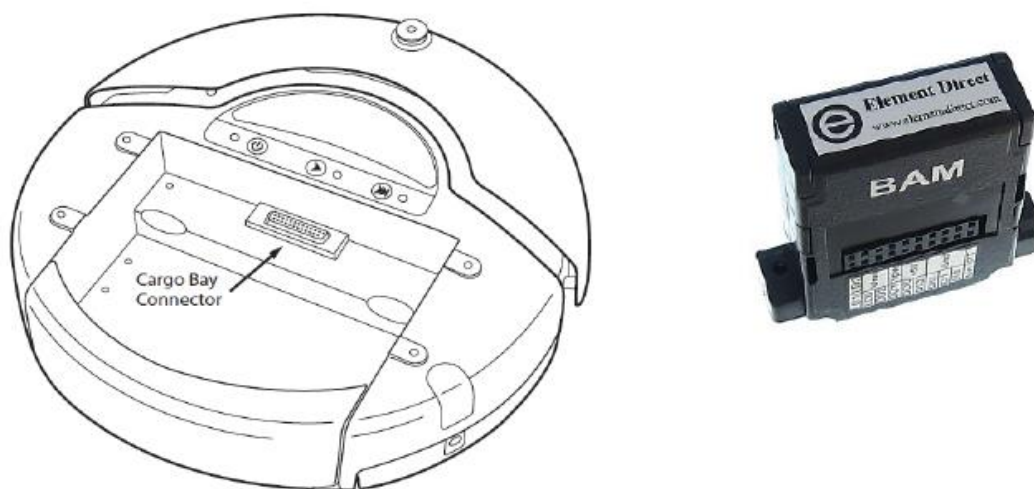


Ilustración 16 – Indicación para conectar el elemento BAM al robot Create. (Columbia University, 2015)

2. Dirigirnos a la sección de “*Hardware y sonido*” en el **Panel de Control** del ordenador, y dar clic en la opción “*Agregar un dispositivo*”.
3. El sistema comenzará a buscar dispositivos que se encuentren disponibles, debemos seleccionar nuestro Elemento Serial Bluetooth, que tendrá el nombre de “*Element Serial*” tal como lo muestra la Ilustración 17. Luego damos clic en la opción Next.

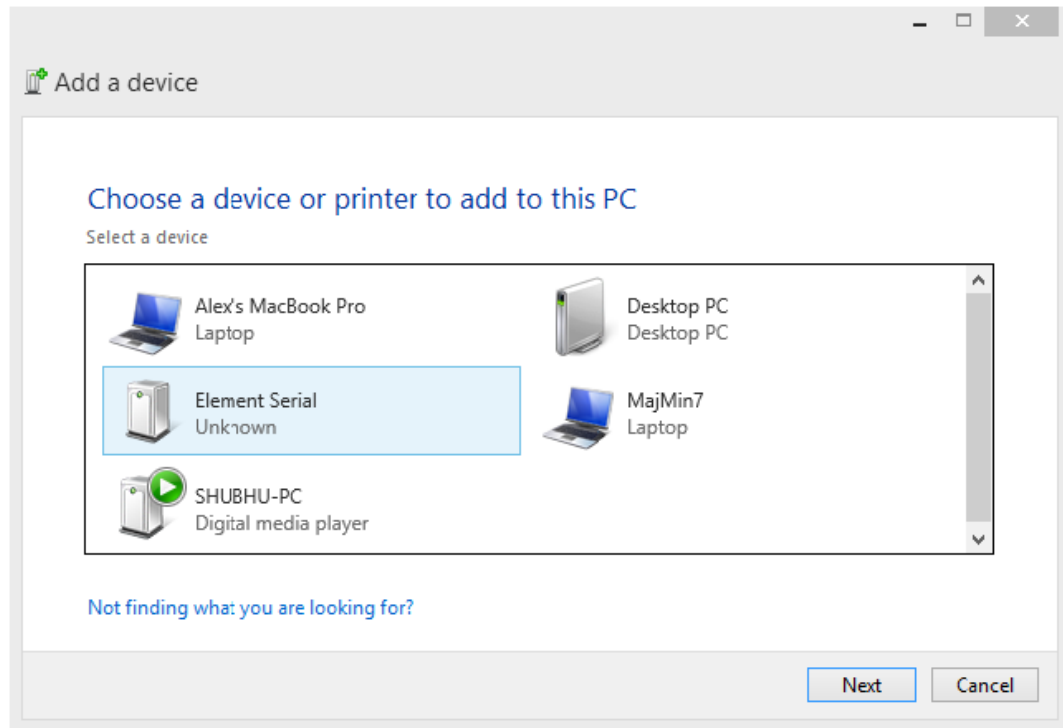


Ilustración 17 – Ventana de diálogo que se muestra al seleccionar la opción “Agregar un dispositivo”. (Columbia University, 2015)

4. El Elemento Serial Bluetooth pedirá que se ingrese el código de seguridad. Ingresar el código “0000” tal como indica la Ilustración 18, y dar clic en el botón “Next”.

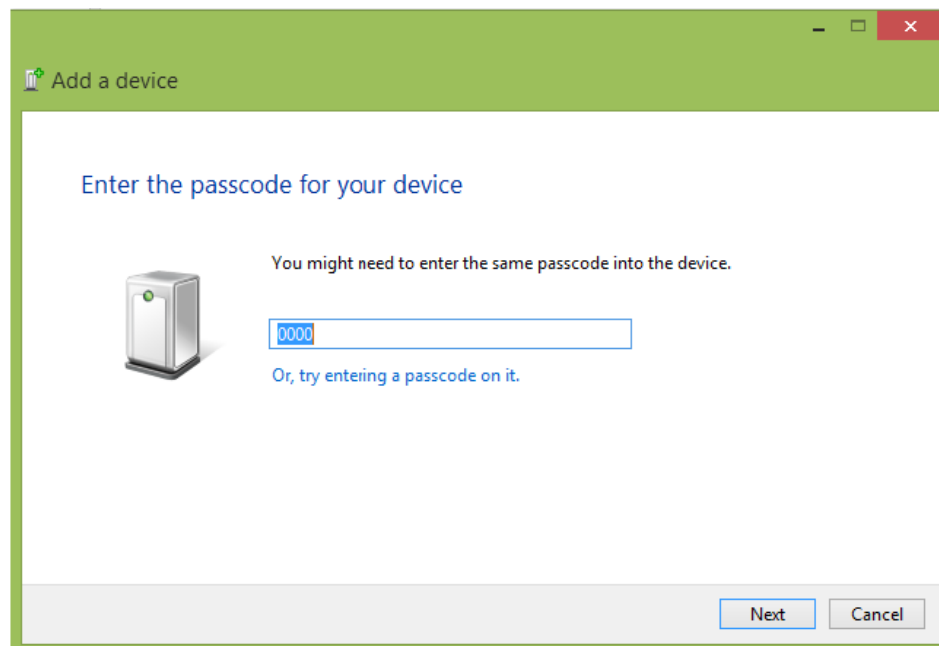


Ilustración 18 – Código de seguridad para el Elemento Serial Bluetooth. (Columbia University, 2015)

5. Una vez hecho esto, Windows comenzará a instalar los controladores para el adaptador del Elemento Serial Bluetooth. Al finalizar este proceso, daremos clic en la opción “OK”.
6. Para terminar nos dirigimos nuevamente al **Panel de control** de Windows, a la sección **Dispositivos e impresoras** y damos clic en **Ver dispositivos e impresoras**. Navegaremos hasta el final de la ventana de diálogo que se nos presenta, y allí podremos ver al dispositivo Elemento Serial Bluetooth debajo de la categoría de “Sin especificar”. Le damos clic derecho al dispositivo, y abrimos la opción de “*Propiedades*”. En la pestaña de “*Servicios*”, debajo del título “*Servicios de Bluetooth*” podremos ver el Puerto Serial que el ordenador ha asignado a nuestro dispositivo Bluetooth, tal como se ve en la Ilustración 19. Por ejemplo, “**COM 3**”. Debemos recordar este puerto para futuras referencias. Este es nuestro puerto serial del robot Create.

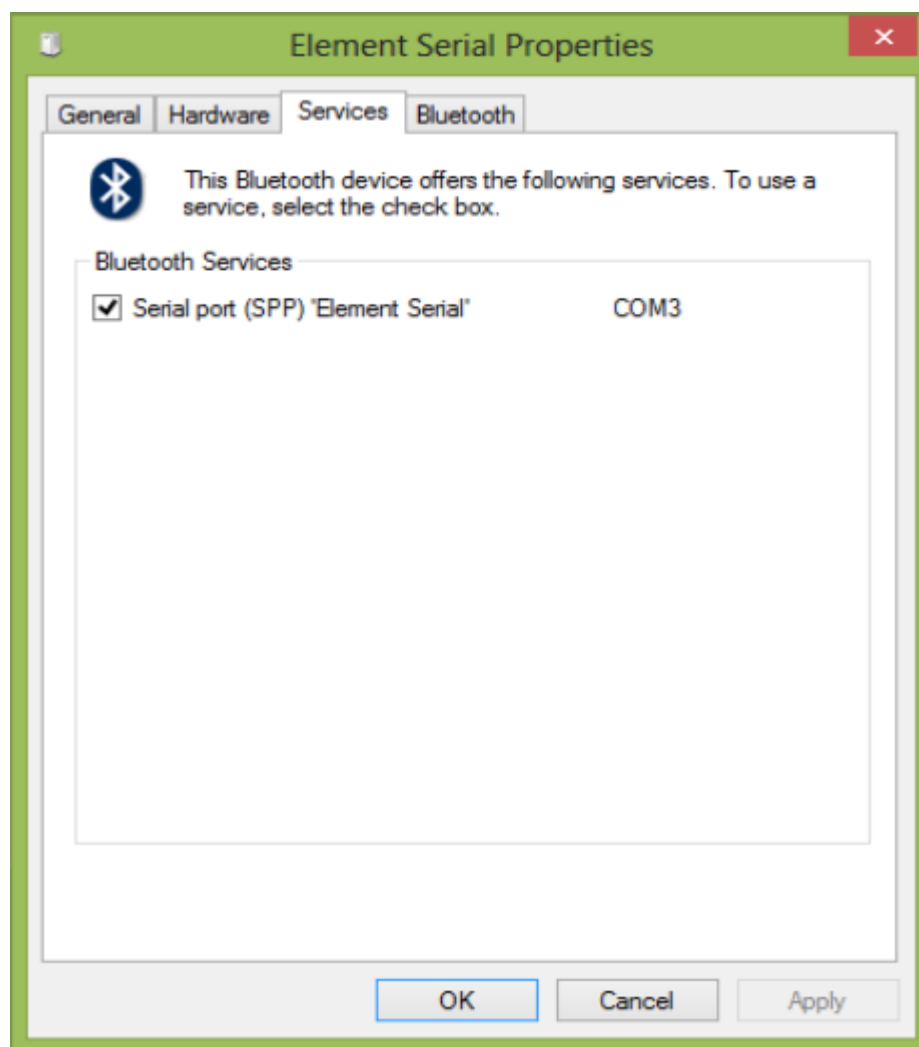


Ilustración 19 – Propiedades del Elemento Serial Bluetooth. (Columbia University, 2015)

2.3 Trabajos relacionados

Durante mediados de los años 90, el artista Eduardo Kac y el diseñador de hardware Ed Bennett realizaron trabajos de telepresencia con dos herramientas que se llamaron “Ornitorrinco in Eden” y “Rara Avis”. Kac publicó en su página de internet un artículo, donde describe ambas herramientas y cuenta en que consistió la implementación de cada una. A continuación se describe lo que Kac y Bennett nos cuentan respecto a cada una de ellas. (Kac, 1994)

Ornitorrinco in Eden

Ornitorrinco es el nombre de un conjunto de instalaciones de telepresencia que desarrolló varias veces durante la década de 1990, y también es el nombre que se le dio al telerobot utilizado para realizarlas.

El primer evento de telepresencia Ornitorrinco tuvo lugar en 1990, entre Río de Janeiro y Chicago, se lo llamó "Ornitorrinco: Experiencia 1".

Como ese, hubo cuatro eventos más y fueron:

- "Ornitorrinco en Copacabana", 1992, un enlace entre dos sitios remotos en Chicago;
- "Ornitorrinco en la Luna", 1993, un vínculo entre Graz, Austria, y Chicago;
- "Ornitorrinco in Eden", 1994, un vínculo entre Lexington, KY, Seattle, WA, Chicago, y el Internet;
- y "Ornitorrinco en el Sahara", 1996, un enlace entre dos sitios remotos en Chicago y San Petersburgo, Rusia.

La instalación “Ornitorrinco in Eden” se llevó a cabo públicamente a través de Internet en 1994. Este experimento duró aproximadamente cinco horas (después de más de un año de experimentos privados). La obra consistía en tres nexos de participación activa, y múltiples nexos de observación en todo el mundo, que podemos ver representados en la Ilustración 20.

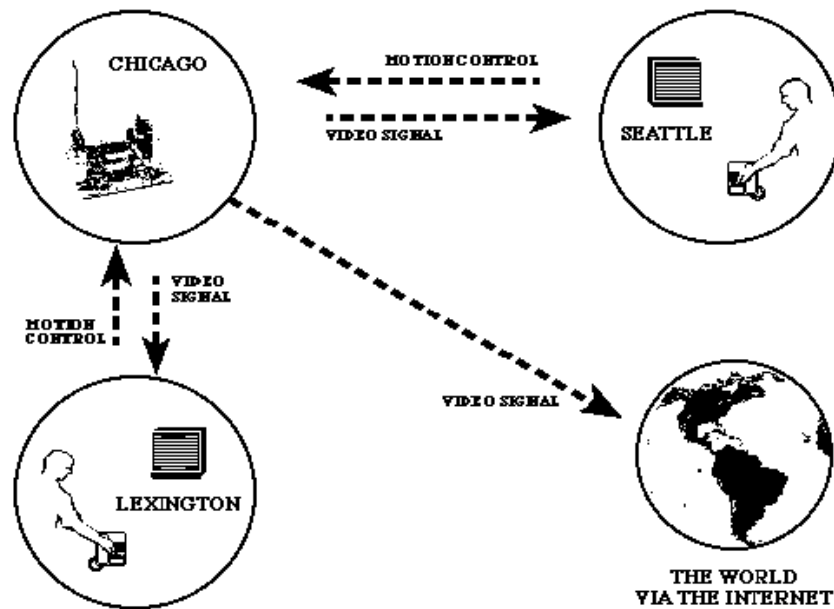


Ilustración 20 – Topología del "Ornitorrinco in Eden", 1994. (Kac, 1994)

Espectadores anónimos de varias ciudades estadounidenses y de otros países (incluyendo Finlandia, Canadá, Alemania e Irlanda), que recibieron información sobre el evento que se llevaba a cabo, pudieron conectarse y navegar por la instalación remota en Chicago, desde el punto de vista del Ornitorrinco, como lo vemos en la Ilustración 21.



Ilustración 21 – Participantes remotos, compartiendo el cuerpo del robot simultáneamente y visualizando la instalación a través de su visor. (Kac, 1994)

En su trabajo, Kac detalla de manera breve los resultados que se obtuvieron durante la implementación de “Ornitorrinco in Eden”, citados a continuación:

“El telerrobot Ornitorrinco, móvil e inalámbrico, que se encontraba en Chicago, fue controlado en tiempo real por participantes que estaban en Lexington y Seattle. Los participantes a distancia compartieron el cuerpo del Ornitorrinco simultáneamente. Mediante Internet, vieron la instalación remota a través del ojo del Ornitorrinco. Los participantes controlaron al telerrobot simultáneamente, a través de una conexión telefónica en tiempo real.” (Kac, 1994).

En la Ilustración 22 se muestra el telerrobot utilizado para la implementación del “Ornitorrinco in Eden”.

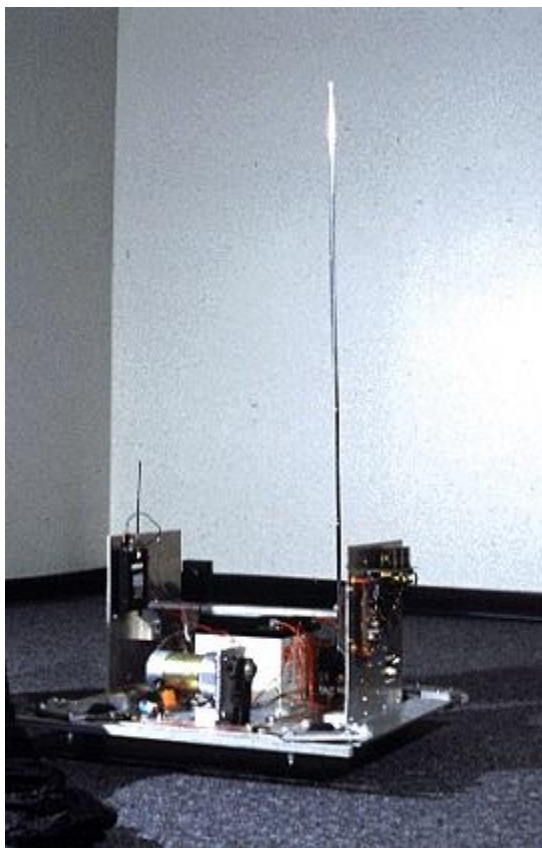


Ilustración 22 – Telerrobot Ornitorrinco. (Kac, 1994)

Rara Avis

Esta fue una instalación interactiva de telepresencia en red, creada por Ed Bennett como director técnico junto a Kac, para una exposición que tuvo lugar entre junio y agosto de 1996, en Atlanta (Georgia, EE.UU.), como parte del “Olympic Art Festival”.

Al entrar en la sala el participante veía desplegada una gran pajarera detrás de un vidrio, con muchas aves revoloteando dentro. Delante de esta pajarera, el participante veía un casco de

realidad virtual y dentro de ella podía ver que había un gran ave, que se distinguía notablemente de las demás y estaba posada inmóvil en una rama.



Ilustración 23 – Rara Avis. (Kac, 1996)

Esta clase de aves se denomina arara, aunque es también conocida como guacamayo, y como vemos en la Ilustración 23 – Rara Avis. , fue adornada con rasgos típicos como ser su larga cola, su pico y un brillante plumaje. Esta ave era en realidad, un telerrobot y dado que la arara tenía los ojos en la parte frontal de la cabeza, como una lechuza, al telerrobot se le llamó Macowl.



Ilustración 24 – Cabeza del telerrobot Macowl (Kac, 1996).

El espectador era invitado a ponerse el casco, y al colocárselo era transportado al interior de la pajarera y podía percibir entonces, la pajarera desde la perspectiva del telerrobot. De esta manera podía observarse a sí mismo desde el punto de vista de la arara. En la Ilustración 24 vemos que para los ojos se utilizaron dos cámaras CCD (Charge-Coupled Device) y cuando el participante movía su cabeza hacia los lados, la cabeza del Macowl se movía también, permitiendo al participante ver todo el interior de la pajarera.



Ilustración 25 – Vista de la instalación desde afuera (Kac, 1996).



Ilustración 26 – Instalación vista desde el telerrobot (Kac, 1996).

En su artículo, Kac cuenta además, que la instalación estaba conectada en red permitiendo a participantes remotos conectarse a través de Internet, y observar el espacio de la galería, desde el visor del robot.

“Los participantes a distancia también podían usar sus micrófonos, para poner en marcha el aparato vocal de la arara telerrobótica, que se oía en la galería. El cuerpo del Macowl era compartido en tiempo real por participantes locales y a la vez, por participantes de todo el mundo, a través de Internet”, explica Kac.

En la Ilustración 25 y la Ilustración 26 se muestra desde dentro y desde afuera, cómo era la instalación en dicha galería. Por otra parte, la Ilustración 27 más abajo nos muestra a varios participantes de diferentes lugares que se conectaron para compartir la visión desde el punto de vista del telerrobot.



Ilustración 27 – Participantes remotos conectados al visor del Macowl a través de Internet. (Kac, 1996).

Con el correr de los años, este tipo de tecnologías se ha ido desarrollando más y más, llegando a tener herramientas con tecnología avanzada y mucho más actualizada, a continuación se detallan dos robots de telepresencia que actualmente se utilizan a nivel mundial y pueden encontrarse en el mercado, estos son:

“VGo” de Vgo Communications

Esta novedosa herramienta diseñada por la compañía Vgo Communications se encuentra ya en el mercado de hoy en día, enfocado principalmente en brindar beneficios a las empresas. Esto es lo que Vgo Communications nos cuenta sobre su producto:

Vgo es una solución todo-en-uno que incluye todo lo necesario para establecer la presencia física del usuario, en una ubicación distante.

Usando la aplicación software dedicada, en una PC o Mac, una persona conectada a internet en cualquier lugar, al instante se conecta a un VGo en otro lugar distante, dándole la capacidad de interactuar no sólo con las personas que están allí, sino también para moverse allí, mediante su propio control del VGo al igual que lo haría si estuviera físicamente en persona. Algunos clientes describen al VGo como su "avatar" personal. (Vgo, 2015)

La parte superior del VGo es lo que las demás personas enfocan, y donde todo el video y audio son capturados, procesados y presentados. Esta parte superior se puede ver claramente en la Ilustración 28 a continuación



Ilustración 28 – Parte superior del VGo (Vgo, 2015).

En esta parte se encuentran integradas la cámara que muestra el lugar donde se encuentra el VGo, una pantalla LCD de 6 pulgadas donde se ve a la persona conectada, luces que muestran el estado de la herramienta, WiFi embebido, botones táctiles para la configuración y el control local, 4 micrófonos para captar el sonido de la ubicación remota y los altavoces. (Vgo, 2015)

En la parte inferior del VGo, mostrado en la Ilustración 29, es donde se genera el movimiento, aquí se encuentran dos motores de rueda controlados independientemente, junto al compartimento donde se conecta la batería, sensores que brindan seguridad a la herramienta, otro altavoz para complementar el sonido y unas atractivas luces led que permiten mayor visibilidad en habitaciones oscuras. (Vgo, 2015)



Ilustración 29 – Parte inferior del VGo. (Vgo, 2015)

Se trata de una tecnología sofisticada desarrollada por Vgo Communications, pero que es de suma simpleza para el manejo del usuario. El mismo corre remotamente la aplicación llamada VGo App, en su iPad o computadora (PC o Mac), cliquea la ubicación en la que se quiere hacer presente, e instantáneamente aparece allí. (VGo, 2015)

El VGo se conecta a internet usando WiFi (o un servicio 4G LTE brindado específicamente por la empresa Verizon, servicio que se debe adquirir por aparte). El VGo es monitoreado continuamente por una red (VGoNet) desarrollada con éste propósito, que mantiene un rastro de la herramienta e inicia la telepresencia a partir de la solicitud de un usuario remoto. (VGo, 2015)



Ilustración 30 – Aplicación VGo App corriendo en pantalla. (VGo, 2015)

En la Ilustración 30 se puede ver lo que la aplicación VGo App muestra en pantalla al usuario que se conecta remotamente, brindándole desde una nítida imagen del lugar en el que se encuentra el VGo, así como también información de importancia como ser el nivel de batería, la calidad de señal de la conexión y opciones útiles para el usuario remoto como la posibilidad de

silenciar sus micrófonos, encender las luces del VGo o realizar una captura de lo que se ve en pantalla.

“Low-Cost Video Chat Robot”

Esta herramienta, fue llamada así por su creador Johnny Chung Lee; un muchacho estadounidense de Mountain View (California, EE.UU.) quien debido a su relocalización en dicha ciudad, quiso buscar la forma de mantenerse en contacto con su prometida, que permanecería en la ciudad de Seattle, Washington. Para esto creó un robot remotamente controlado con el que podría realizar video-llamadas, y así mantenerse en contacto con su pareja. En la Ilustración 31 podemos ver dicha herramienta.



Ilustración 31 – Johnny Lee junto a su herramienta (Nathan, 2011).

J. Lee, en un blog personal destinado a documentar sus creaciones, explica que su proyecto se conforma de dos importantes componentes: el robot Create de la empresa iRobot (descrito en la Sección 2.2.2 de este trabajo) que le costó U\$S 250, y el ordenador netbook que vemos montado sobre el Create en la Ilustración 31, que consiguió también por un precio de U\$S 250 aproximadamente. Así, por un total de U\$S 500 implementó una herramienta que en el mercado, fácilmente cuesta miles de dólares (ver Sección 2.2.1.3 de este trabajo). Además diseñó él mismo un software para su herramienta, basado en el lenguaje C# utilizando la herramienta *Visual Studio Express 2010*, al cual llamó “*iRobot Create Control*”. En su blog comparte abiertamente dicho software para ser descargado, y al respecto hace dos importantes advertencias: la primera es que fue únicamente probado en su ordenador utilizando el sistema operativo Windows 7; y en la segunda advertencia, si bien aclara que el software puede tener varios problemas al ser utilizado, alienta a quienes lo descarguen a sentirse libres de mejorarlo. (Lee, 2011)

Para la solución de este trabajo se utilizó el software que Lee diseñó, justamente con mejoras implementadas por el alumno, las cuales se mostrarán en detalle en la Sección 4.1.3.

Esta herramienta es un claro ejemplo de cómo la telepresencia muestra su utilidad y rentabilidad, además de ayudar enormemente a la dinámica con la que se dan las relaciones personales en la actualidad, sobrepasando las limitaciones de tiempo y espacio que imponen otras tecnologías como ser las de videoconferencia.

Más adelante, en la Sección 3.6.4, se hablará del funcionamiento del software aquí mencionado.

2.4 Desarrollo actual de las técnicas, herramientas, y otros recursos requeridos

Para la realización de este trabajo se utilizará el robot Create de iRobot, en conjunto con el elemento serial BAM (ambos descritos en la Sección 2.2.2 de este trabajo), los cuales la Universidad dispone, y tendrá la cortesía de prestarlos al alumno para su proyecto. Estos elementos, en conjunto, serán implementados en un robot de telepresencia, donde tendrán como función, brindar al usuario la movilidad que caracteriza a esta clase de herramientas.

Además de estos elementos que la Universidad proporcionará, el alumno dispone de dos ordenadores portátiles (ver Sección 4.1.1), que también serán utilizados para este proyecto. Los mismos servirán para la parte que involucra las comunicaciones en un robot de telepresencia, mediando entre el lugar remoto en el cual el usuario de la herramienta se hace presente, y su ubicación real.

Las comunicaciones involucradas en esta solución, se pueden diferenciar claramente en dos partes respecto a los elementos que se deben conectar entre sí; donde una parte corresponde a la comunicación audio-visual que se establecerá entre los ordenadores portátiles, para lo cual se utilizará el mundialmente conocido software *Skype*. Los mismos además, ejecutarán simultáneamente el software “*iRobot Create Control*” diseñado por J.C. Lee, el cual se mencionó al final de la sección anterior, y será gracias a las mejoras que el alumno agregó a este programa (descritos en la Sección 4.1.3), que el usuario remotamente conectado podrá controlar el movimiento de la herramienta, desde su propia ubicación.

La otra parte, corresponde a la vinculación entre el robot Create y el ordenador local, el cual estará en la misma ubicación que el robot, y se conectarán a través del dispositivo BAM descrito en la Sección 2.2.2, mediante la utilización de los cuatro elementos que se señalan en la Ilustración 15.

Skype

Skype es un software que habilita la comunicación entre todo aquél que lo utiliza. Hoy en día ya se cuentan por millones las personas, y empresas, que lo usan para hacer llamadas y videoconferencias gratis, además de poder enviar mensajes instantáneos y compartir archivos a través de esta plataforma. Skype ya tiene sus distintas versiones para cada entorno, sea un teléfono móvil, un equipo informático o una TV con Skype, permitiendo adaptarlo a las necesidades individuales de cada usuario. Su manejo es muy intuitivo y se puede descargarlo gratuitamente, desde su página oficial de internet. En la Ilustración 32 se muestra su característica pantalla de inicio de sesión. (Skype, 2015)



Ilustración 32 – Pantalla de inicio de Skype (Neetguias, 2013).

Entonces, estableciendo una videollamada como lo permite esta aplicación, el usuario del robot de telepresencia podrá interactuar tanto visual como auditivamente en la ubicación de la herramienta con personas que allí se encuentren, ya que recibirá a la vez el feedback de imagen y sonido del lugar donde se hace presente.

Programa “iRobot Create Control”

“iRobot Create Control” es el nombre que J.C. Lee decidió darle al software que creó para el control de su herramienta de telepresencia. En su blog personal aparte de compartir dicho

software, con su correspondiente código y manuales en formato PDF con instrucciones para su manejo, describe algunas de las funciones que el mismo permite realizar como ser: configurar automáticamente la redirección de puertos en nuestro ordenador, y el protocolo UPnP (*Universal Plug and Play*) de nuestro router, para poder realizar una conexión a través de la parte pública de Internet (es decir, de nuestra red privada hogareña por ejemplo, hacia afuera), para lo cual también se encarga de consultar la dirección IP externa que identifica a nuestra computadora. Además, mantiene una pulsación a través de la red que detiene el movimiento del robot si se pierde la conexión con el mismo, permite definir una contraseña personal para dicha conexión, nos informa del nivel de carga de la batería del robot mediante un indicador ubicado en la interfaz del programa, e inclusive nos brinda la posibilidad de establecer el nivel máximo de aceleración de los motores de las ruedas del Create, para que la herramienta no se sacuda bruscamente, ni se quede atrás o se caiga. (Lee, 2011)

Luego de esta descripción de las funciones que el software trae consigo, realiza otra importante advertencia en cuanto al establecimiento de la redirección de puertos, aclarando que esta característica se encuentra lejos de tener un perfecto funcionamiento, y tampoco ha pasado por un proceso de testeo, por lo que podría suceder que no funcione a la hora de utilizarlo. Aquí es donde toman relevancia las modificaciones que el alumno realizó a dicho programa, logrando que dichas características funcionen correctamente para los fines específicos de este trabajo.

A continuación se describen los conceptos de *UPnP* y *Redirección de Puertos*, lo cual ayudará a entender mejor el funcionamiento de este software, cuando se lo describa en el capítulo siguiente.

Universal Plug and Play

Universal Plug and Play (UPnP) es un conjunto de protocolos de comunicación que permite que los periféricos de una red (como ser ordenadores personales, impresoras, puntos de acceso Wi-Fi o dispositivos móviles) puedan detectar la presencia de otros dispositivos en esa misma red, y establecer servicios de comunicación entre ellos, pudiendo compartir datos a través de la misma. UPnP está diseñado principalmente para redes de hogar sin dispositivos del ámbito empresarial. Existe un foro dedicado especialmente a UPnP que es el encargado de promocionar el uso de ésta tecnología; se trata de una iniciativa de la industria informática para permitir una conectividad simple y robusta entre los dispositivos autónomos y ordenadores personales de diferentes fabricantes. El *Foro UPnP* está compuesto por fabricantes de diferentes ámbitos que van desde la electrónica de consumo, hasta las redes de ordenadores. (Wikipedia, 2015)

En esa misma fuente se detallan algunas características del protocolo, entre las cuales encontramos (Wikipedia, 2015):

- **Direccionamiento:** el direccionamiento IP es la base de UPnP, donde cada dispositivo debe implementar un cliente DHCP (*Dynamic Host Configuration Protocol*) y buscará a su vez un servidor DHCP la primera vez que se conecte a una red. En caso de no encontrar un servidor disponible, el dispositivo se asigna a sí mismo una dirección, mediante el proceso llamado “AutoIP”. El dispositivo utilizará su propia dirección IP

para las operaciones en la red, a menos que durante la detección del servidor DHCP obtenga un nombre de dominio, por ejemplo, mediante un servidor DNS, con lo cual utilizará este nombre para posteriores operaciones.

- **Descubrimiento:** Una vez establecida una dirección IP el siguiente paso en UPnP es el *descubrimiento*, lo que permitirá al dispositivo anunciarse a sí mismo en la red mediante el Protocolo Simple Service Discovery (SSDP). Esto habilita a los dispositivos que acaban de añadirse a una red, anunciar sus servicios, a los puntos de control presentes en la misma.
- **Descripción:** La *descripción* de un dispositivo permite a un punto de control tener mayor información acerca de él, luego de que ha sido descubierto en la red. A través de esto podrá conocer mejor sus capacidades y podrá interactuar con él. Esta descripción incluye una lista de dispositivos o servicios embebidos, así como URLs de control, manejo de eventos y presentación. Para cada servicio se incluyen comandos, o *acciones*, a los que responderá el servicio, y parámetros de cada comando, lo que permitirá a los puntos de control poder consumirlos de ser necesario.
- **Control:** Al obtener la recién nombrada descripción del dispositivo, el punto de control puede enviar acciones a los servicios de un dispositivo, y el servicio responderá con un mensaje de control, con los resultados de la acción, de forma similar a una llamada a una función. Dicha respuesta consistirá en información respectiva a las variables relacionadas con el servicio iniciado, las cuales moldean e informan sobre su estado en tiempo de ejecución.
- **Notificación de Eventos:** Esta capacidad adicional de UPnP consiste en un protocolo (General Event Notification Architecture) mediante el cual se envían actualizaciones a los puntos de control. Las mismas permiten mantenerlos informados ante modificaciones en las variables de algún servicio al cual se han suscripto en un dispositivo de la red.
- **Presentación:** Por último en UPnP está la *presentación*; si un dispositivo tiene una URL de presentación, el punto de control puede obtener una página de la misma y mostrarla en un navegador, lo que permitirá al usuario controlar el dispositivo y/o consultar su estado, lo cual dependerá en realidad, de las características y el nivel de interactividad presente en la interfaz de dicha página.

Redirección de Puertos

En cuanto a esta característica mencionada entre las funciones del programa; en las redes de computadoras, la redirección de puertos o también llamada asignación de puertos, es una aplicación de traducción de direcciones de red (NAT), mediante la cual se redirige una solicitud de comunicación proveniente de una combinación de dirección IP y número de puerto, a otra combinación así, mientras que los paquetes atraviesan una puerta de enlace a la red, como ser un router o un firewall. Esta técnica es frecuentemente utilizada para que los servicios de un host que reside en una red protegida (interna), estén disponibles para otros hosts en el lado opuesto de la puerta de enlace (red externa), lo cual se logra reasignando la dirección IP de destino y el número de puerto de la comunicación, hacia el host interno. Esto posibilita que computadoras remotas (por ejemplo, computadoras en la Internet) se conecten a una computadora o servicio específicos, que se encuentran dentro de una red de área local (LAN) privada. (Wikipedia, 2015)

2.5 Documentación y citas de soluciones al problema

En secciones anteriores cuando se brinda un marco teórico acerca de la telepresencia, se detalla cómo la misma forma parte de un conjunto mayor de conceptos, investigaciones, desarrollo de ideas y avances tecnológicos. En particular se describe el término *teleoperación* y la cercana relación que guarda con la telepresencia.

Si bien la teleoperación de robots fue una de las primeras aplicaciones concretas de la telerrobótica y trajo muchas nuevas posibilidades, a medida que se la aplicaba se pudo ver con mayor claridad que no es la solución ideal para todas las situaciones. Existen muchas tareas que a la larga se vuelven repetitivas y extremadamente aburridas.

Por ejemplo, si consideramos una situación en la que un operario debe manejar un auto controlado por radio desde un joystick, luego de unas horas se le hará cada vez más difícil prestar atención a la tarea. Ahora agreguemos a esta situación, la idea de que el mismo operario tenga que controlar el auto mientras mira a través de una pantalla colocada ante a él, donde visualiza lo que el auto tiene en frente. La tarea se volverá aún más difícil debido al campo de visión limitado, y a que no dispone de una visual periférica. Imaginemos además, que a causa de un ancho de banda limitado en el enlace de comunicación, dicha cámara no retroalimenta de nuevas imágenes acorde al movimiento del auto, lo que hace aún más tosca la visión del entorno. Esta serie de factores desemboca en que a las pocas horas de comenzada su labor, el operario experimente algo que se denominó *fatiga cognitiva*; su atención deambula y hasta puede que sufra dolores de cabeza u otro síntoma físico provocado por el estrés. Aun si el despliegue visual fuese excelente, el teleoperador puede que sienta la así llamada *enfermedad del simulador*, que surge a partir de una descoordinación entre el oído interno de la persona, el cual indica que el operador se mantiene estático, mientras el sistema visual le muestra que se está moviendo. (Murphy, 2000)

Si comparamos la siguiente imagen de un sistema de telepresencia (Ilustración 33), con la arquitectura característica de la teleoperación mostrada en la Ilustración 3, a primera vista es posible notar cómo éstas se diferencian entre sí, en que mientras en los sistemas de telepresencia el

operador mantiene una interacción directa con la herramienta en cuestión, en la teleoperación esto varía ya que el operario interactúa con una réplica del robot real encargado de llevar a cabo las tareas que el humano no puede por razones específicas. Otro claro ejemplo de esto se evidencia en la Ilustración 3.

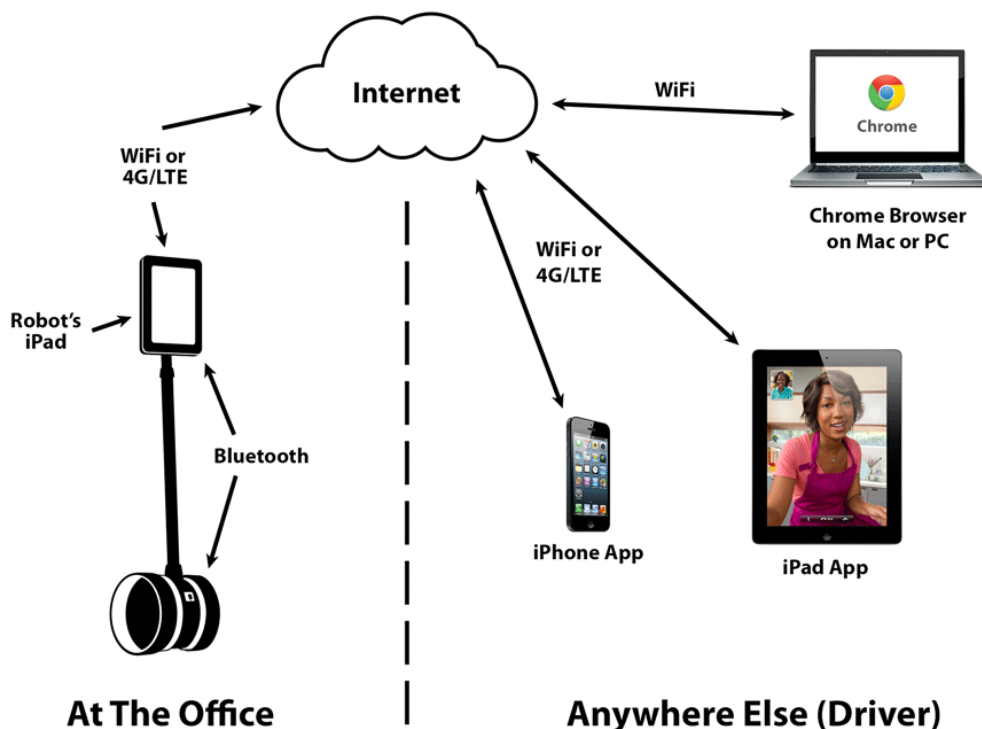


Ilustración 33 – Imagen de una Arquitectura de un Sistema de Telepresencia (RoboSavvy, 2015).

Durante la década de los años 90, a la par de las primeras incursiones en la teleoperación, se realizaron intentos por reducir la fatiga cognitiva de los operarios mediante la adición de una mayor cantidad de cámaras, con tasas de actualización más rápida, de manera que se ampliase el campo de visión para que fuese más consistente con cómo los humanos vemos el mundo. Esto claramente no fue práctico debido a los limitados anchos de bandas existentes en esa época, lo cual se tradujo en actualizaciones visuales descoordinadas a través de Internet, sumamente molestas. Entonces, si un ancho de banda limitado nos da como resultado un movimiento trabado, y por lo tanto, un mayor incremento de fatiga cognitiva, agregar más cámaras solamente exacerbó el problema añadiendo más información que debía ser transmitida a través de estas limitadas velocidades de comunicación. (Murphy, 2000)

La telepresencia surge a partir de la teleoperación como un área de nuevas investigaciones y un enfoque diferente en la búsqueda por reducir los malestares de la simulación y la fatiga cognitiva, tratando de que la interfaz entre el humano y el robot sea más natural, a través de tecnologías que permitan al usuario tener la mayor sensación posible de estar allí presente donde se encuentra la herramienta que utiliza.

El caso de Devon Carrow en Estados Unidos

En el año 2013, la página El Comercio publicó un artículo sobre Devon Carrow, un niño de 9 años cuyas mortíferas alergias le impiden asistir presencialmente a la escuela. Padece de *esofagitis eosinofílica*, un cuadro causado por glóbulos blancos que crecen en el estómago y el esófago; y también tiene el síndrome del shock anafiláctico, que causa reacciones alérgicas potencialmente mortales a algunos desencadenantes, como la leche y los maníes.

A pesar de los desalentadores diagnósticos de Devon, esto no le imposibilita del todo concurrir a clases, ya que lo hace a través de la herramienta VGo, de Vgo Communications, la cual se ha descrito anteriormente en la Sección 2.4 de este trabajo. En la Ilustración 34 podemos ver una imagen de Devon asistiendo a clases a través del VGo.



Ilustración 34 - Devon Carrow en la pantalla del VGo (Comercio, 2013).

El robot de 1,20 metros con enlace inalámbrico de video le permite asistir a sus clases desde la distancia, participar en ellas, caminar por los pasillos y hasta dialogar en el recreo con sus compañeros. Lo que más llama la atención es la naturalidad con que esta maravilla tecnológica es aceptada por sus compañeros de clases. Estos niños de nueve años criados con videojuegos, avatares y juguetes a control remoto no ven un robot, sino sencillamente a Devon. El hecho de que él no esté físicamente allí no significa que los demás no sientan su presencia; inclusive cuando debió ser hospitalizado en el año 2012, sus compañeros de colegio le escribieron tarjetas de solidaridad y todos lo dibujaron como un niño, y no como un robot. En su artículo, la página de noticias además cuenta lo siguiente (Comercio, 2013):

“Desde que fue presentado en el 2011 por VGo Communications, con sede en Nashua, Nueva Hampshire, un puñado de estudiantes en todo el país lo han usado, incluso en Nueva Jersey, Wisconsin, Texas y Iowa.

También ha llamado la atención de los círculos médicos y empresariales, permitiendo que los pacientes consulten a los médicos y que los trabajadores se presenten virtualmente en la oficina, aun mientras están de viaje.

La tecnología amplía la experiencia escolar de Devon más allá de lo que sería posible mediante un diálogo de video, y las únicas restricciones que se presentan son físicas: El robot tantea las escaleras y se detiene. Pero tampoco es un obstáculo insalvable porque, con un peso inferior a los 9 kilogramos (18 libras), es suficientemente liviano como para que una maestra lo alce. Antes de avanzar, Devon enfoca la cámara hacia abajo para asegurarse de no embestir a algún compañerito que esté agachado atándose los cordones, y para esto incluso el mismo VGo advierte sobre la presencia de objetos grandes situados en frente.”

Este es un claro ejemplo de cómo la tecnología de telepresencia ayuda enormemente a solventar las problemáticas que se pueden presentar en el mundo de hoy, mostrando que ya no todo se trata de presencia física, sino que debemos dar paso a nuevas ideas, incorporando la tecnología que se desarrolla día a día, más aun con los ritmos tan acelerados y dinámicos que la sociedad actual nos impone en todos los ámbitos.

3. DEFINICIÓN DEL PROBLEMA

3.1 Definición del problema

Las comunicaciones hoy en día precisan ir de la mano de la tecnología de punta para que los usuarios puedan satisfacer sus necesidades de negocio y relaciones personales, por lo que es indispensable la integración de un nuevo tipo de herramientas que quiten de la ecuación, las grandes distancias que las personas hoy consideran a la hora de desenvolverse el mundo, no solo en su ámbito laboral, sino también escolar, familiar, social, etc.

Sin ir más lejos, aquellas personas que por alguna razón no pudiesen asistir a sus actividades personales diarias, por ejemplo por enfermedad o dolencias, son éstos los casos que se verían especialmente beneficiados por una herramienta que los pudiese hacer presentes en el lugar donde necesiten estar, permaneciendo a la vez en donde deben hacer reposo. Inclusive casos de otra índole, por ejemplo, profesionales que desean dar conferencias, cursos o seminarios en distintas universidades del mundo, podrían acceder a muchos más lugares gracias a este tipo de herramientas de telepresencia, sin verse limitados por la imposibilidad del traslado físico a esos lugares.

3.2 Objetivos

La meta de este trabajo es mostrar una posibilidad de aplicación de la telepresencia, la cual se puede implementar utilizando tecnología y herramientas sencillas al alcance de la mano.

Utilizando el robot Create que posee la Universidad se diseñará una herramienta de telepresencia, mediante la cual un usuario en una ubicación remota pueda controlar el robot, y así interactuar con personas que estén en la cercanía del mismo, tal como sucede en la Ilustración 1.

3.3 Alcances o procesos involucrados

Los marcos del trabajo estarán delimitados a abarcar la comunicación entre el robot Create y un ordenador remoto. Desde el mismo, un usuario manejará el robot con comandos sencillos de movimiento.

Además el usuario podrá hacerse “presente” donde se encuentra el robot, gracias a los elementos de audio y video ubicados en el ordenador remoto que enviarán su imagen y sonido. De igual manera, recibirá imagen y sonido del lugar donde se encuentra el robot, gracias al ordenador local montado sobre él. El robot se vincula con este ordenador local a través de un dispositivo de conexión bluetooth.

La conexión entre los ordenadores se da vía Internet, y para ello se utiliza un software que permite la comunicación de los mismos, a través de un puerto que se especifica dentro de dicho programa.

De esta manera, el ordenador remoto ejecutará los comandos de movimiento (avanzar, retroceder, girar a la derecha o a la izquierda) que el ordenador local montado sobre el robot recibirá a través de Internet, y los transmitirá al robot a través de su vinculación bluetooth con el mismo.

Este programa además, posee en su interfaz un indicador que muestra el estado de la batería del robot, lo cual es muy útil para la retroalimentación del usuario que utiliza la herramienta, además de esto puede definir la velocidad con la que se moverá el robot al dar las órdenes de movimiento, y existe también la posibilidad, de modificar este software para que el robot tenga otras funciones, por ejemplo, que si se llegase a chocar emita un sonido para dar aviso. Esto último en particular no existe actualmente, por lo que se lo propone aquí como una posible función a implementar.

3.4 Restricciones o límites del trabajo

Debido al alcance de este trabajo y su naturaleza académica, además de la dificultad que existe para conseguir una herramienta como lo es el robot, dada la situación de las importaciones en nuestro país, al momento de implementar una solución de este tipo la primera prioridad es tener en cuenta la seguridad del robot Create, del cual depende enteramente la movilidad de la solución.

Es por esto que se intentará diseñar un elemento de soporte para montar encima del robot, al ordenador local que se vincula por *bluetooth* con el mismo, de manera tal que no haya perjuicios ni para el robot, ni para el ordenador en cuestión.

3.5 Recursos disponibles

Para este trabajo se utilizarán las siguientes herramientas que la Universidad dispone: el robot “Create” (ver Ilustración 13) será el principal elemento y eje central de este trabajo, brindando la movilidad que tendrá la solución. La Universidad además dispone de un dispositivo de conexión Bluetooth (ver Ilustración 14) utilizado específicamente para que el robot pueda vincularse a un ordenador, y a través de ese enlace, recibir comandos de movimiento.

Aparte de estos elementos que la Universidad pone a disponibilidad, el alumno utilizará para el prototipo de la solución, dos ordenadores portátiles personales:

- Una notebook desde la cual un usuario conectado remotamente controlará el movimiento del robot (ver Ilustración 57) y;
- Un segundo ordenador notebook (ver Ilustración 55) que se vinculará localmente mediante bluetooth al Create, y funcionará de intermediario entre el robot y el ordenador anterior. Además se utilizará un programa de videoconferencia para mostrar en pantalla con audio incluido, al usuario que se conecta remotamente.

La comunicación entre estos ordenadores se dará a través de un programa que permite enviar comandos por Internet (ver Ilustración 44). De esta manera, el ordenador local vinculado al robot, será el receptor de los comandos de movimiento enviados por el ordenador remoto, para que el robot los ejecute.

Se intentará diseñar un soporte mediante el cual el segundo ordenador notebook se sostenga sobre el robot, dando el toque final a la solución ensamblada ideal. Sin embargo, como se aclara en el punto anterior, lograr esto está limitado por la seguridad del robot, que es la primera prioridad a la hora de diseñar un soporte como éste.

3.6 Alternativas tecnológicas de soluciones al problema

A continuación se describen las herramientas que se encuentran disponibles para implementar una solución de este tipo, de acuerdo al uso que se les da. Y luego se detallan las alternativas que surgieron como posibles prototipos de la solución.

3.6.1 Comunicación por video y audio

En esta sección se encuentran descriptas las herramientas y tecnologías que existen en la actualidad para conectar a las personas a través de elementos audiovisuales (como una cámara web y un micrófono), de manera que puedan entablar una conversación a través del diálogo verbal, sea donde sea que se encuentren; siempre y cuando dispongan de una computadora con una conexión a Internet, así como de los elementos necesarios para transmitir y recibir tanto audio como video.

Skype

Este software, descripto en la Sección 2.4 de este trabajo (ver Ilustración 32), es la herramienta que se ha elegido para utilizar en el prototipo de la solución que se propone. La misma fue elegida por sobre las demás que se detallan a continuación, debido a su eficiencia en el uso de los recursos, a su fácil acceso, instalación y manejo; y también a que ya cuenta con una reputación a nivel mundial, en cifras de usuarios que van por el rango de los millones, de manera que aunque estén aquellos que no lo utilizan, al menos conocen de su existencia y podrían acceder a ella fácilmente si así lo desearan.

Además de Skype existen otras herramientas afines que buscan brindar estos servicios de comunicación, algunas de ellas son Hangouts de la red social *Google+* o Facebook Calls de la red social *Facebook*, y serán descritas a continuación.

Google Hangouts

Hangouts es una funcionalidad que la red social *Google+* implementó junto con su lanzamiento en el año 2011, y tiene como objetivo permitir a los usuarios de esta red social conectarse mediante llamadas de voz o inclusive video.

Es una aplicación multiplataforma de mensajería instantánea, desarrollada por la compañía de Google Inc. Y se creó para sustituir los servicios *Google Talk*, *Google+ Messenger* y *Google+ Hangouts*, unificándolos todos en una única aplicación. (Wikipedia, 2015)

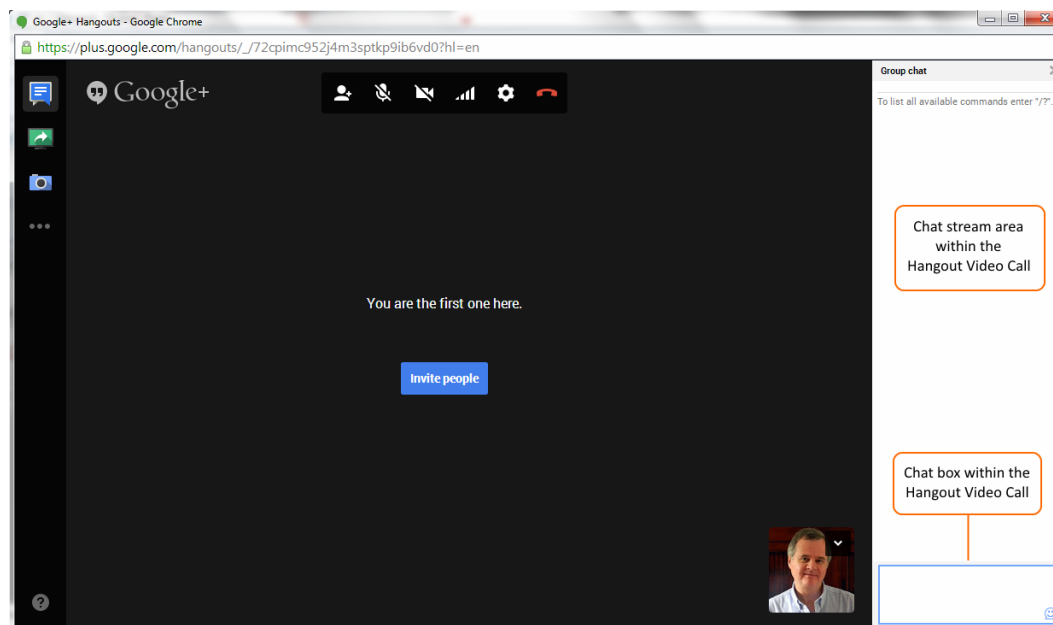


Ilustración 35 – Interfaz de Google Hangouts para videollamada (Strategic Content, 2014).

La principal diferencia entre Hangouts y Skype, es la posibilidad que permite el primero, de realizar una videoconferencia simultánea con un número de personas mayor que Skype, que permite esta funcionalidad solamente entre dos usuarios al mismo tiempo. Como se puede ver en la Ilustración 35, mediante el botón “Invite people” en color azul, podemos invitar a una o varias personas a unirse a la videollamada.

En un artículo de “El Confidencial Digital”, del año 2013, Francisco Rubira dice lo siguiente (Rubira, 2013):

“Skype parece estar más centrado en las telecomunicaciones, mientras que Google parece querer abarcarlo todo, desde la publicidad, hasta los contenidos, pasando por ser la alternativa a los servicios de llamadas telefónicas y de videoconferencia.”

Facebook Video Calling

Este servicio es similar a la implementación que realizó Google+ del servicio Hangout. Está pensado para entablar videollamadas con amigos dentro de la red social Facebook. En la Ilustración 36 se muestra la pantalla de inicio para activar esta característica en nuestra propia cuenta personal de Facebook.

En la página de ayuda de Facebook, se detalla lo siguiente (Facebook, 2015):

“Las videollamadas de Facebook, que funcionan con los navegadores Google Chrome, Firefox y Opera, te permiten realizar llamadas directamente desde Facebook sin necesidad de descargar otros programas o plug-ins. Tan solo tienes que hacer clic en el botón con forma de videocámara (📹), ubicado en la esquina superior derecha de una conversación para iniciar una videollamada.”



Ilustración 36 – Pantalla para iniciarse en Video Calling de Facebook (Kainth, 2011).

Existen otros softwares de videoconferencia en Internet, como Meeting Burner, WebEx, Huddle o Fuze Meeting; pero la desventaja de estos software, en comparación con las tres descritas anteriormente, es que se trata de servicios que no son gratuitos, por lo que para la finalidad de esta solución, han sido descartados.

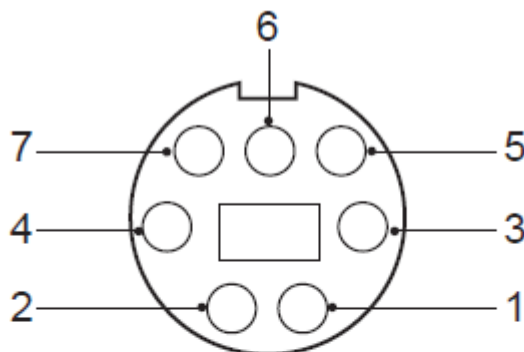
3.6.2 Comunicación entre robot y ordenador

En el capítulo anterior cuando se detalló la herramienta robótica que será utilizada en este trabajo (ver Sección 2.2.2), fue descrito junto a ella, el dispositivo bluetooth (ver Ilustración 13) que se usará para conectarla y comunicarla con el ordenador que se intentará montar encima. Este método de conexión fue elegido para este trabajo debido a que la Universidad ya disponía del mismo, pero existen además otras maneras de lograr la conexión entre el robot y el ordenador, que se detallarán a continuación; las cuales son vía WiFi, o a través de un Cable serial USB.

Cable Serial “RS232 (DB9) a Mini-DIN”

Una de las alternativas a la conexión bluetooth que se utiliza con el Create, es por vía de un cable que se conecta a un puerto serie de la computadora y al puerto serial del robot.

Así como la bodega de carga del robot trae un conector DB-25, su puerto serial, también conocido como conector *Mini-DIN*, trae una entrada para un conector de 7 pines, y en la Ilustración 37 vemos la ilustración incluida en el manual del *Open Interface* del Create.



Pin	Name	Description
1	Vpwr	Create battery + (unregulated)
2	Vpwr	Create battery + (unregulated)
3	RXD	0 – 5V Serial input to Create
4	TXD	0 – 5V Serial output from Create
5	BRC	Baud Rate Change
6	GND	Create battery ground
7	GND	Create battery ground

Ilustración 37 - Conector Mini-DIN del robot Create (iRobot, 2015).

En dicho manual se recomienda usar el cable serial del Create de iRobot, y no un cable serial cualquiera, debido a la diferencia de voltajes existente entre los pin 3 y 4 señalados en la imagen (RXD, TXD) y los voltajes que manejan las PC. A continuación en la Ilustración 38 vemos el cable que iRobot recomienda utilizar (iRobot, 2015).



Ilustración 38 - Cable Mini-DIN a Puerto Serie de iRobot. (Vic7767, 2015)

Al igual que el dispositivo BAM, este cable serial también permite establecer una conexión con un ordenador, la cual asignará un número de puerto serial a esa comunicación, de la misma manera que sucede con el BAM (ver Ilustración 19).

Conexión WiFi

La segunda alternativa para la comunicación con el robot Create, es lograrla a través de un dispositivo WiFi, que desde el año 2013 se comercializa con el nombre de *RooWifi*.

Se trata de un módulo embebido para los robots Roomba de la empresa iRobot, que permite realizar varias operaciones con esta gama de robots, desde programar la limpieza y controlar cada motor del robot, hasta obtener la información del dispositivo en tiempo real y hacerlo funcionar desde cualquier localización donde se posea acceso a Internet. De esta manera, el manejo se puede realizar tanto desde iPhones, iPads, dispositivos Android y Blackberry; como desde un servidor o PC con sistemas operativos de uso doméstico. (RooWifi, 2013)



Ilustración 39 - Dispositivo RooWifi. (RooWifi, 2013)

En la Ilustración 39 podemos apreciar el dispositivo WiFi en cuestión. Las características del mismo, y su compatibilidad respecto a una conectividad Wi-Fi, son las que se describen a continuación (RooWifi, 2013):

- **Redes compatibles:** WiFi 802.11b/802.11g/802.11n con claves de encriptación estándares.
- **Radio de cobertura:** 400 metros.
- **Velocidad de transmisión:** 1 y 2 Mbps.
- **Encriptación:** AES128.
- **Certificación Radio:**
 - FCC para Estados Unidos
 - IC para Canadá
 - ETSI para Europa y ARIB para Japón

iRobot Create 2

El éxito de los robots Roomba fue evidente para iRobot, y aún más evidente es el creciente número de personas que adquieren este dispositivo con fines de “hackearlo” y usarlo para otros propósitos, diferentes al de su utilidad como aspiradora. Debido a esto la empresa iRobot decidió en diciembre de 2014, lanzar al mercado un nuevo robot al que llamó *Create 2*, ya que vendría a ser una evolución sobre el robot Create que se ha detallado a lo largo de este trabajo, y que será el que se utilice para la solución propuesta.

La idea de este nuevo producto fue diseñarlo de manera que modificarlo y adaptarlo para la función que se quiera sea aún más fácil que con su versión anterior, acercándolo de esta manera tanto a programadores novatos, como a personas con avanzados conocimientos de robótica (iRobot, 2015).



Ilustración 40 - Robot Create 2 de la empresa iRobot (iRobot, 2015).

3.6.3 Conexión entre ordenadores

Investigando en Internet de qué forma se puede controlar el movimiento del robot desde el ordenador remoto, a través del ordenador que se tratará de montar sobre el robot, existe una rápida y eficiente solución brindada por parte de la tecnología de escritorios remotos que existe hoy en día.

Existen varias opciones para esto, que van desde el uso de la aplicación de Escritorio Remoto que Windows trae por defecto, o la aplicación que posee Chrome para escritorios remotos a través del navegador, hasta herramientas más sofisticadas y dedicadas especialmente a esto como TeamViewer (Ilustración 41), AnyDesk (Ilustración 42), Ammyy Admin (Ilustración 43), entre otros.

De todos ellos, el que mejor se ajustó en un principio a las necesidades de este trabajo, fue la aplicación TeamViewer, la cual posee una licencia gratuita para uso no-comercial como lo es en este caso. En los comienzos, la misma fue elegida para la solución de este trabajo, aunque más adelante sería reemplazada por el uso del programa “**iRobot Create Control**”.

TeamViewer, es una de las soluciones líderes en todo el mundo para intercambio de escritorio y colaboración en línea a través de Internet. La tecnología básica desarrollada por TeamViewer GmbH permite el funcionamiento eficiente, económico y global de una extensa red de servidores, a través de la cual se enrutan las conexiones que se basan en la tecnología geolocalización (TeamViewer, 2015).

Mediante este programa corriendo en ambos ordenadores que se quieran vincular, es posible acceder de uno al otro, utilizando el ID y Contraseña proporcionados por el programa para cada uno, tal como se puede ver en la Ilustración 41.

En cuanto al software mencionado (*iRobot Create Control - iCC*), el mismo fue elegido ya que mediante su uso, fue posible descartar la utilización de otras herramientas más sofisticadas y a la vez pesadas. Por ejemplo, al principio se consideró buscar un algoritmo de movimiento para el robot a través de herramientas de Microsoft como *Robotics Developer Studio* o *Visual Studio*, las cuales tienen un consumo considerable de recursos.

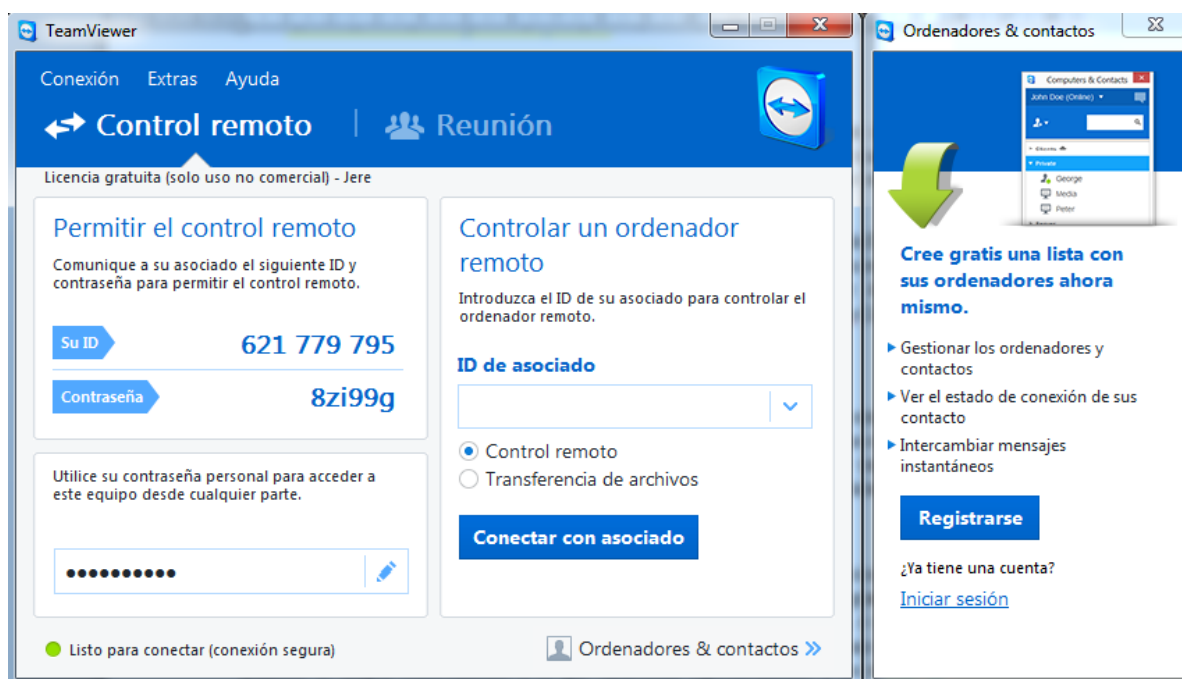


Ilustración 41 – Programa TeamViewer en su versión gratuita de uso no comercial (TeamViewer, 2015)

Mediante el uso del software *iCC*, se pudo prescindir de esta clase de herramientas tan pesadas, y también a su vez de las tecnologías de escritorios remotos, las cuales ya no fueron necesarias gracias a las utilidades de este programa. A continuación en la siguiente sección, se describe en detalle dicho software.

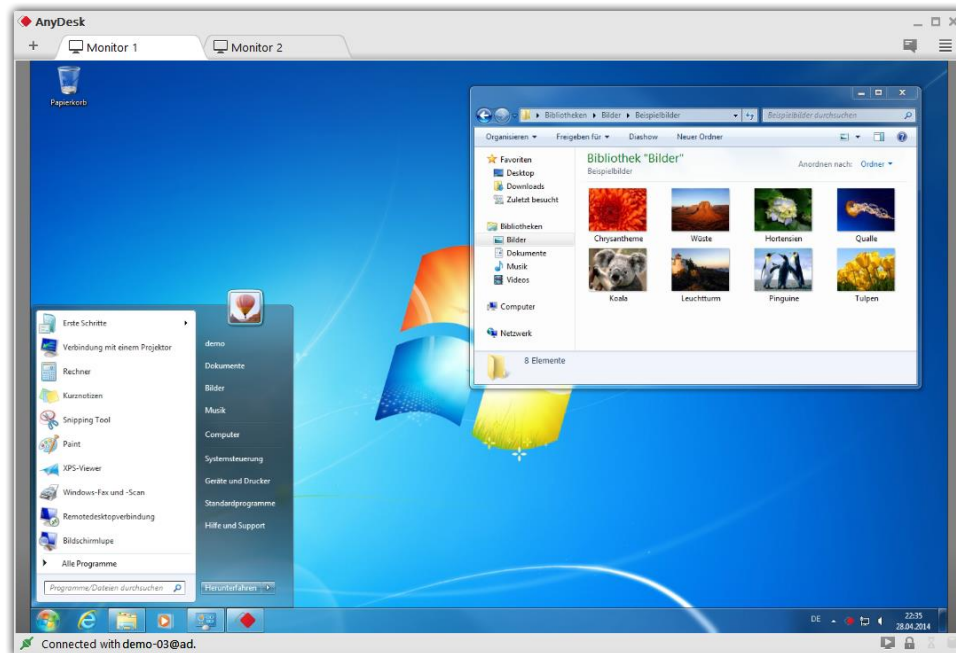


Ilustración 42 – Imagen del programa AnyDesk (AnyDesk, 2015)

Una vez conectados los mismos, es posible configurar dentro de dicha conexión variadas opciones que permitirán una utilización totalmente personalizada, y adaptada a nuestras necesidades particulares.

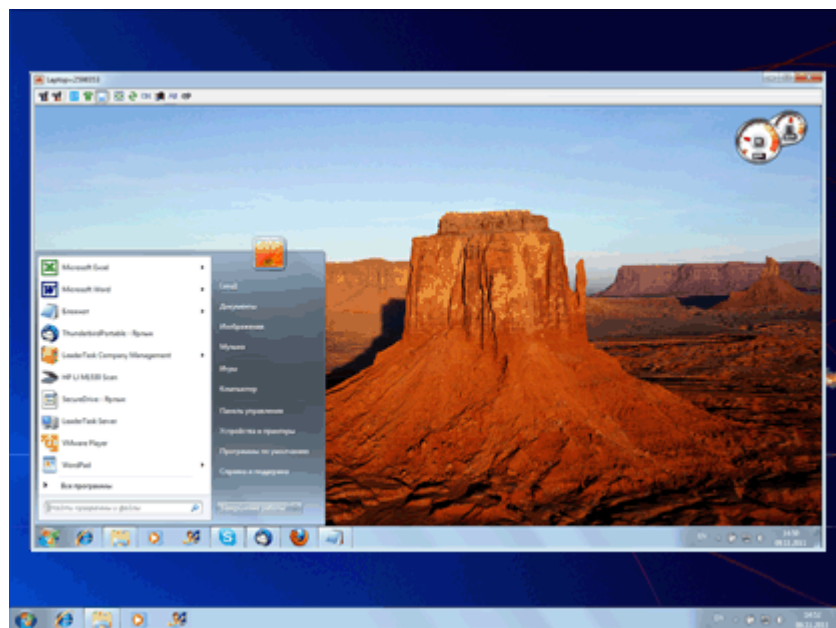


Ilustración 43 – Imagen del programa Ammy Admin (Ammyy, 2015)

3.6.4 Movimiento del robot

Para establecer la movilidad del robot y lograr la navegación de los espacios, existen diversos algoritmos o enfoques de movimiento, como ser recorridos programados y preestablecidos, detección de caminos posibles en una ubicación determinada, o inclusive distintas opciones para el barrido completo de una habitación; aun así ninguno de estos planteos se adapta concretamente a lo que caracteriza específicamente a los robots de telepresencia, que es su capacidad para moverse libremente en un determinado espacio, permitiendo a su operador elegir el camino a recorrer, y brindando así esa mencionada sensación de “estar allí presente”.

Fue así que indagando entre ejemplos ya trabajados y robots existentes de este tipo, se llega a la conclusión de que lo mejor, es que la persona que se conecta remotamente sea quien manipule el robot. De esta manera podrá moverse libremente por el espacio haciendo uso del recurso visual que se utilice, con lo cual sería incluso mayor la sensación de presencia o “telepresencia” que se busca mediante este trabajo.

Existen plataformas que se pueden utilizar para lograr este manejo y control del robot, como ser la herramienta de Microsoft, el *Microsoft Robotics Developer Studio*. Se trata de un entorno de programación basado en el *framework* .NET que se encuentra disponible de manera gratuita, para ser utilizado tanto por desarrolladores profesionales y no profesionales, como por los aficionados de estos temas (Microsoft, 2015).

Por otra parte existe también ROS (*Robot Operating System*), que al igual que la herramienta Microsoft Robotics Developer Studio, provee de un *framework* para la escritura de software de un robot. La diferencia radica en que ROS está basado en Ubuntu, por lo tanto se trata de una herramienta de *open source* e indefectiblemente, requerirá de un mayor conocimiento de ese sistema operativo. Se trata de una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento complejo y robusto en un robot, con una amplia variedad de plataformas robóticas (ROS, 2015).

Como se mencionó en el capítulo anterior, entre los trabajos existentes este tipo, está el caso de la solución implementada por J.C. Lee, quien comparte abiertamente el programa que diseñó para vincularse y controlar un robot Create, tal como el que posee la Universidad.

Se utilizará para este trabajo dicho programa, el cual luego de algunas modificaciones en su código, logró ajustarse a las necesidades que se presentaban, permitiendo controlar libremente el robot con sencillos comandos de movimiento (avanzar, retroceder, girar a la izquierda o derecha). A continuación se describe dicho software.

3.6.4.1 Programa “iRobot Create Control”

En la Ilustración 44 vemos una captura de pantalla con lo que el programa muestra cuando se lo inicia por primera vez. Mediante este, el ordenador que está vinculado al robot vía bluetooth (y que se intentará montar sobre el mismo), permite que otro ordenador utilice el puerto serie que se le asigna a esa conexión bluetooth, para manipular el movimiento del robot de manera local o remota; en el caso de esta imagen, dicho puerto sería COM3.

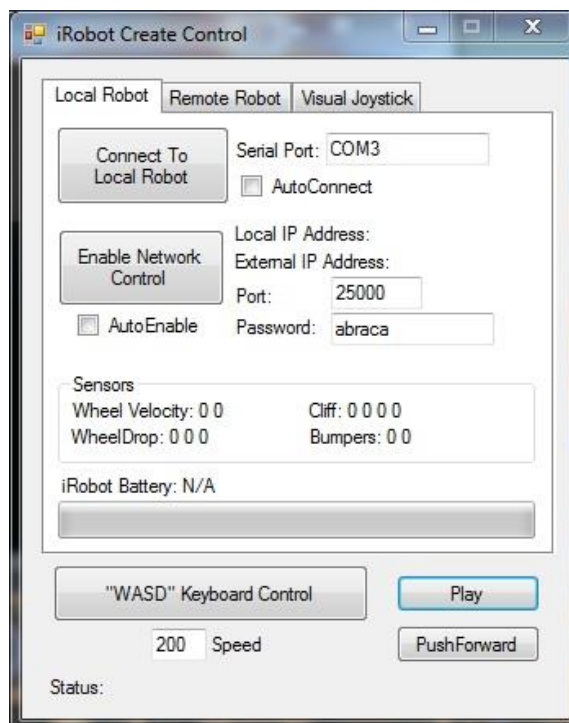
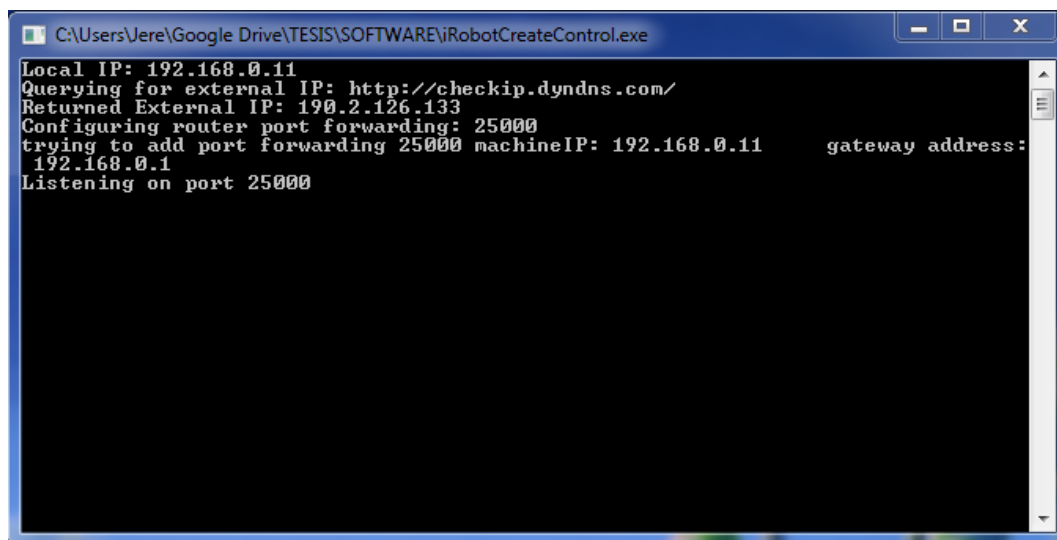


Ilustración 44 – Programa iRobot Create Control.

La utilización de este software es sencilla e intuitiva, pero es necesario diferenciar claramente cómo debe ser usado en cada ordenador, dependiendo si se trata del que se intentará montar sobre el robot, o de aquél que se conecta remotamente a la herramienta de telepresencia. Por cuestiones de facilidad, el ordenador que se conecta por bluetooth al robot, será llamado **ordenador B**, y aquél que se conecta a la herramienta desde una ubicación remota, se le llamará **ordenador R**. El programa, además de la interfaz mostrada en la Ilustración 44, cuenta con una ventana terminal (ver Ilustración 45) donde se brinda información al usuario, respecto a los eventos que van ocurriendo.

El ordenador B se encarga de funcionar como un canal entre el ordenador R que envía comandos de movimiento, y el robot que los ejecuta. Para esto debe iniciarse el programa iRobot Create Control (iCC) en el ordenador B, y con el botón “*Enable Network Control*” habilitar la función de redirección de puertos, o también conocida como *Port Forwarding*. Función que, si se inicia correctamente, lo informará a través de la terminal de la manera que lo muestra la Ilustración 45.



```
C:\Users\Jere\Google Drive\TESIS\SOFTWARE\iRobotCreateControl.exe
Local IP: 192.168.0.11
Querying for external IP: http://checkip.dyndns.com/
Returned External IP: 190.2.126.133
Configuring router port forwarding: 25000
trying to add port forwarding 25000 machineIP: 192.168.0.11 gateway address:
192.168.0.1
Listening on port 25000
```

Ilustración 45 - Terminal de mensajes del programa iCC.

Habilitar esta funcionalidad configurará el ordenador para recibir una conexión a través del puerto que indiquemos en la interfaz del programa. En este caso, el puerto 25000. Si esto se habilita correctamente la terminal nos lo indicará, como vemos en la anterior imagen, con el último mensaje que se ve: *Listening on port 25000*. Que en inglés significa “escuchando en el puerto 25000”.

Una vez que esto se activa, resta que el ordenador R se conecte al ordenador B, a través del mismo programa iCC.

A continuación se explica cómo el programa se utiliza de diferente manera, si se trata de una conexión local o remota con el robot.

3.6.4.1.1 Conexión Local

El término “local” refiere a que la vinculación entre los ordenadores B y R se da a través de una misma red de internet a la cual ambos están conectados, justamente lo que se conoce una red local.

Para establecer esa vinculación, el ordenador R debe iniciar también el programa iCC indicando en la pestaña “*local robot*” el número de puerto que el ordenador B utiliza (COM3 siguiendo el ejemplo), tal como se ve en la Ilustración 44. Luego deberá ubicarse en la pestaña *Remote Robot*, como se ve en la Ilustración 46.

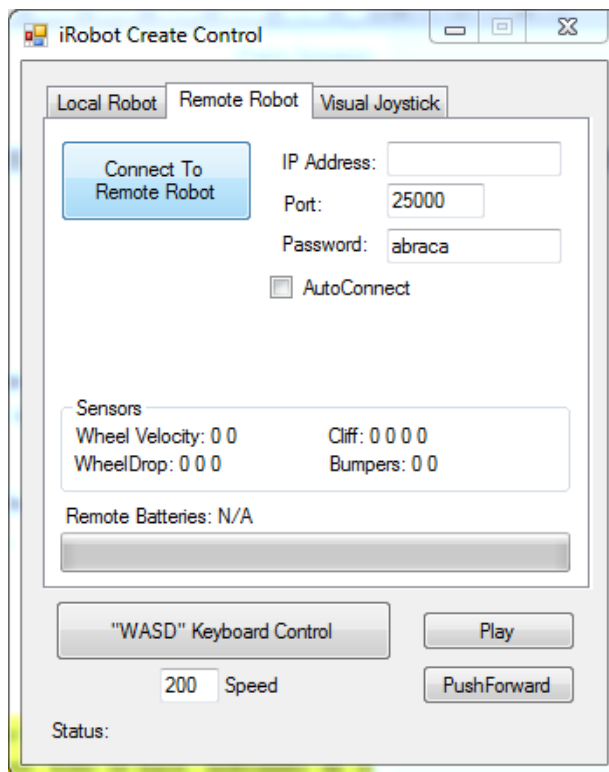


Ilustración 46 - Programa iCC, pestaña "Remote Robot".

En esta pestaña, para establecer la conexión, el usuario deberá ingresar los mismos datos mediante los cuales se habilitó el control del robot a través de Internet en el ordenador B. Continuando con el ejemplo de las anteriores ilustraciones, se debe indicar la dirección IP del ordenador B, y el puerto por el cual se comunicarán los ordenadores (puerto 25000); y por último, ingresar la contraseña establecida que vimos en la Ilustración 44, es decir, “*abraca*”.

La dirección IP es mostrada por la terminal del programa en el ordenador B, como se puede ver con el mensaje: “*Local IP: 192.168.0.11*” en la Ilustración 45. Al estar ambos ordenadores conectados a la misma red, se podrán vincular utilizando simplemente la dirección local asignada por ésta. Una vez ingresados estos datos, el usuario del ordenador R debe dar clic en el botón “*Connecto To Remote Robot*” y al establecerse exitosamente dicha vinculación se devolverá el mensaje correspondiente en la ventana terminal, como lo muestra la Ilustración 47.

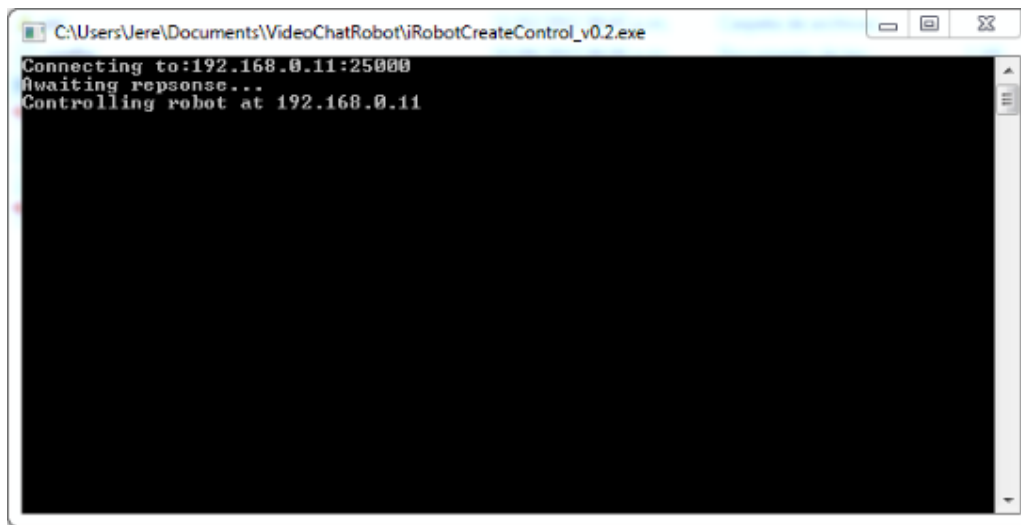


Ilustración 47 - Terminal del Ordenador R controlando al robot a través de red local.

Una vez establecida la comunicación entre ambos ordenadores, el ordenador R puede comenzar a mover el robot habilitando la función que trae el programa iCC para el envío de los comandos de movimiento, la cual se habilita haciendo clic en el botón ubicado en la parte inferior de la interfaz con la leyenda “*“WASD” Keyboard Control*”, que se ve en la Ilustración 46.

A través de esta función, el usuario del ordenador R podrá mover el robot utilizando las teclas de las letras W, A, S y D de su propio teclado, con la siguiente correspondencia:

- W: Avanzar.
- S: Retroceder.
- A: Girar sobre sí mismo en sentido anti horario.
- D: Girar sobre sí mismo en sentido horario.

Además el usuario podrá visualizar en su pantalla, la velocidad a la que el robot se está moviendo cuando se presionan las teclas de movimiento. Por ejemplo, presionar la tecla W (avanzar), mostrará la información sobre la velocidad de las ruedas, como se ve en la Ilustración 48, señalado con un círculo rojo.

El programa incluso brindará información respectiva a otros sensores del robot como la detección de riscos (*Cliff*), informará también si el robot es levantado del piso (*WheelDrop*) o si se choca contra algún objeto frente a él (*Bumpers*).

También se mostrará el nivel de la batería del robot, en la barra debajo de la leyenda “*Remote Batteries*”, y el estado de la conexión indicando la dirección IP del Ordenador B, como también se ve en la Ilustración 48, con el mensaje “*Status: Controlling robot at 192.168.0.11*”. Cabe señalar que el indicador de la batería no muestra su nivel de carga en la imagen, debido a que la misma fue capturada durante pruebas en las que no se disponía de un robot desde el cual se pudiera medir esta información.

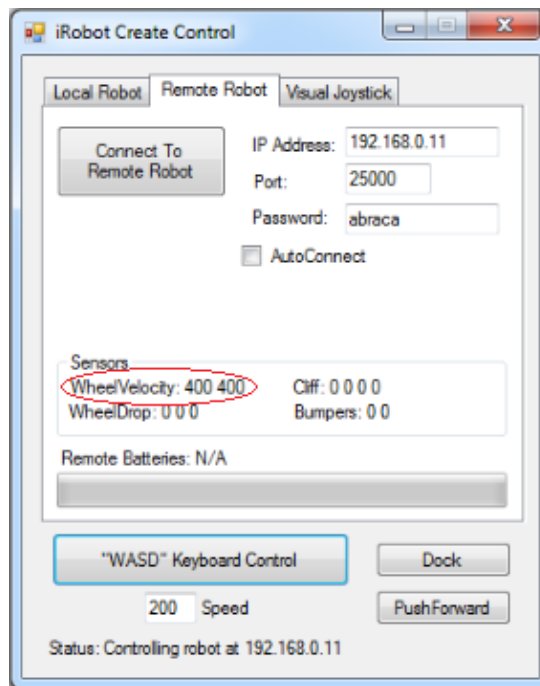


Ilustración 48 - Interfaz en Ordenador R mostrando velocidad de las ruedas del robot.

En el caso de que el ordenador B y el ordenador R, no estén conectados a Internet a través de la misma red, el procedimiento para establecer la comunicación entre ellos es diferente, y será descrito a continuación.

3.6.4.1.2 Conexión Remota

Con el término “remota” se hace referencia a que la conexión a establecer entre el ordenador B y el ordenador R, se dará a través de diferentes redes mediante las cuales los ordenadores están conectados a Internet.

En el ordenador B debemos realizar lo mismo que se describió antes del inciso anterior, debiéndose correr el programa iCC, y clickeando el botón “*Enable Network Control*” para habilitar conexiones entrantes.

Con el ordenador R el procedimiento cambia ligeramente, ya que en lugar de usar la dirección IP local que la terminal de mensajes nos brinda, se deberá usar la dirección externa que se indica con el mensaje “*Returned External IP: 190.2.126.133*”. Esta dirección es consultada a Internet por el programa y devuelta en forma de mensaje dentro de la terminal. Se la conoce como “*External IP*” o “IP externa” debido a que es la dirección que se asigna a los ordenadores en el nivel global de la Internet.

Así los pasos a seguir, serán los mismos que los descritos en el inciso anterior, donde el usuario del ordenador R deberá indicar en la pestaña “*Remote Robot*” los datos de la conexión: la dirección IP externa, el número de puerto y la contraseña establecida. Continuando con el ejemplo que se viene mostrando, la Ilustración 49 muestra los respectivos datos para la conexión.

Luego solo resta que se conecte al robot mediante el botón “Connect To Remote Robot” y mediante las teclas W, A, S, D de su teclado se mueva por el espacio utilizando la interfaz visual que le permitirá ver donde se encuentra, y hacia dónde se dirige.

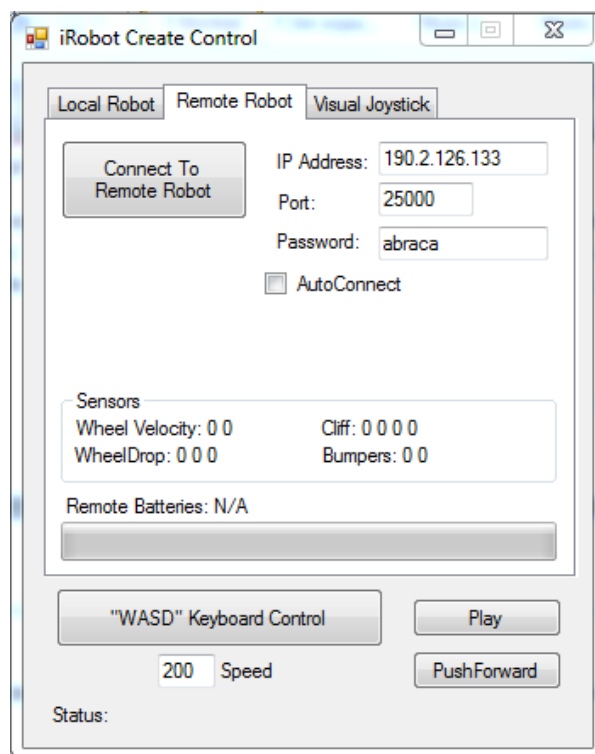


Ilustración 49 - Datos de Conexión Remota

Como vemos, también se trata de un procedimiento sencillo para poder conectar remotamente los ordenadores. Aun así en este caso, existe un paso más que se debe realizar antes de que los ordenadores B y R puedan estar conectados a través de la Internet pública. A continuación se describe ese paso.

Habilitación de Puertos

Si bien el programa iCC es bastante sencillo de usar para establecer las conexiones entre los ordenadores, para que los mismos puedan comunicarse entre sí remotamente, es necesario configurar los dispositivos enrutadores que cada uno de los ordenadores utiliza, conocidos generalmente con el término *router*.

Esta configuración consiste en habilitar en cada uno de ellos el puerto a utilizarse, tanto en el **router B** correspondiente al ordenador B, como en el **router R** del ordenador R. Esto permitirá que la petición de control del robot llegue del ordenador R al ordenador B, y debe hacerse ya que, aunque el programa iCC habilita por sí mismo el puerto que le indiquemos (25000 por ej.), este puerto corresponde a los contrafuegos de los ordenadores en cuestión, y no al de los router involucrados.

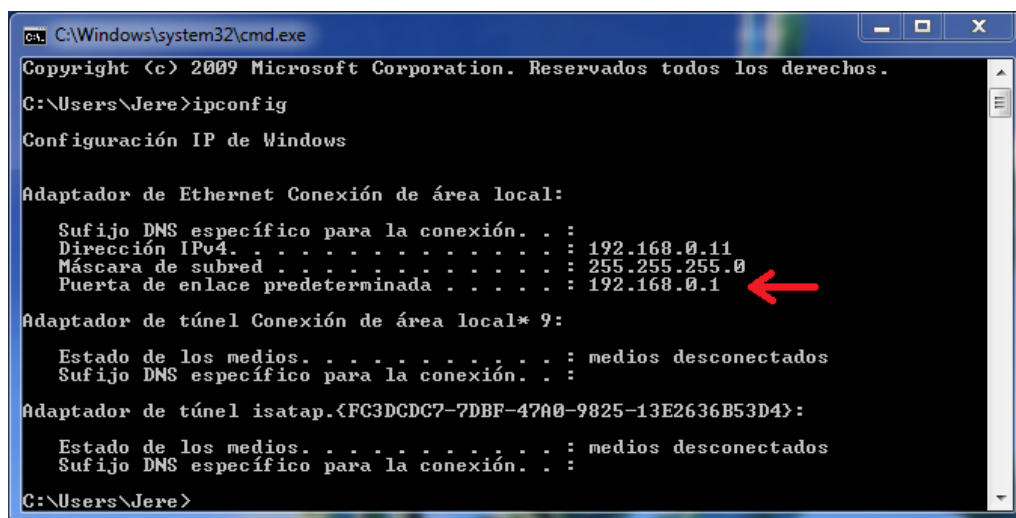
Existen puertos de uso general que estos dispositivos traen ya configurados para utilizar. Por ejemplo, los más comunes del protocolo TCP (protocolo que nos asegura que los paquetes serán recibidos sin errores y en el orden que fueron enviados) son los puertos 80 y 443 que corresponden a HTTP (Protocolo de Transferencia de HiperTexto) y HTTPS (Protocolo utilizado para la transferencia segura en páginas web) respectivamente.

Por defecto, los router no traen esta configuración de puertos habilitada, por lo que debemos realizarla manualmente en ambas redes involucradas, tanto para la del ordenador B (**red B**), como la del ordenador R (**red R**), de manera que los mismos puedan vincularse entre sí para el manejo del robot.

A continuación, a través de una serie de imágenes se muestra la manera en que se habilitan estos puertos como el del ejemplo que se viene ubicando (p. 25000), en el router de la red de su hogar, que el alumno utilizó para la realización de las pruebas:

Dicha habilitación consiste en cambiar la configuración del router, para que acepte una conexión entrante que se dirija a una IP dentro de su red, a través de un puerto específico. En este caso, la IP en cuestión es la que correspondería al ordenador B, junto con el 25000 puerto que vamos a utilizar.

Para realizar esto, debemos ingresar a la configuración de nuestro router, lo cual se puede hacer ingresando a través del navegador, usando la dirección que el dispositivo usa como puerta de enlace predeterminada de la red. Podemos obtener esta dirección ejecutando el comando "ipconfig" en una terminal de comandos de Windows, como lo muestra la Ilustración 50.



```
C:\Windows\system32\cmd.exe
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\Jere>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Conexión de área local:

    Sufijo DNS específico para la conexión. . . : 
    Dirección IPv4. . . . . : 192.168.0.11
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . : 192.168.0.1
    Sufijo DNS específico para la conexión. . . : 

Adaptador de túnel Conexión de área local* 9:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . : 

Adaptador de túnel isatap.{FC3DCDC7-7DBF-47A0-9825-13E2636B53D4}:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . : 

C:\Users\Jere>
```

Ilustración 50 - Terminal de Windows, comando "ipconfig" ejecutado

Ingresando a esta dirección en el navegador, nos llevará al menú de configuración de nuestro router, como puede verse en la Ilustración 51. Cabe señalar que no todos los router permiten ingresar libremente a este tipo de configuraciones; en este caso se trata de un modem de Cisco, modelo DPC2320, con el cual no hay problemas para ingresar a este menú. Pero existen otros en los que el usuario no tiene permitido ingresar a estas opciones, como sucedió por ejemplo con la empresa Fibertel cuando se intentó aplicar esta misma configuración para realizar pruebas en otra red, y se vio que no era posible acceder al correspondiente panel de configuración.

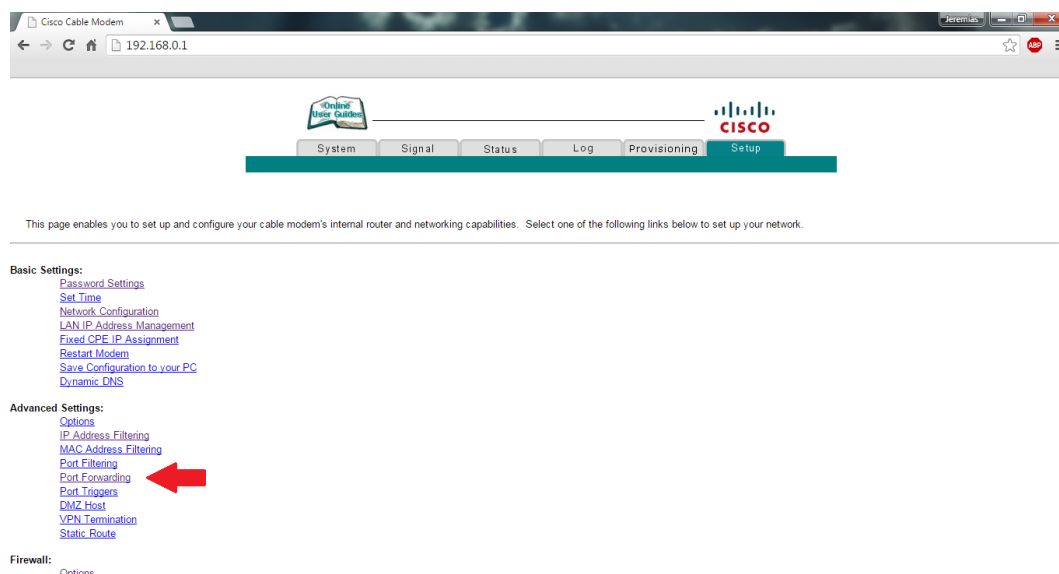


Ilustración 51 - Menú de configuración del modem Cisco DPC2320

Una vez ingresados a este panel, nos dirigimos a la opción “Port Forwarding” del menú de configuraciones avanzadas, señalado con una flecha en la Ilustración 51.

Dentro de esta sección se encuentra una lista de campos donde podremos ingresar los datos correspondientes al puerto que deseamos habilitar. En la Ilustración 52 se puede ver agregada la regla, que corresponde a la red del ordenador B, y que permitirá la aceptación de las conexiones que provengan desde el ordenador R.

Los campos en cuestión que se deben completar son cinco: la dirección IP local de nuestra maquina (ordenador B), el valor de inicio y el valor fin del rango de puertos que deseamos permitir, el protocolo y por último un casillero booleano que indicará si la regla que estamos ingresando estará habilitada o deshabilitada. Siguiendo con el ejemplo mostrado, los datos a ingresar serán la dirección 192.168.0.11. El puerto seguirá siendo el número 25000, y dejaremos la opción “Both” que significa *ambos*, para la opción del protocolo. Finalmente tildamos el casillero de habilitación, y damos clic al botón “Apply” para que las configuraciones tomen efecto.

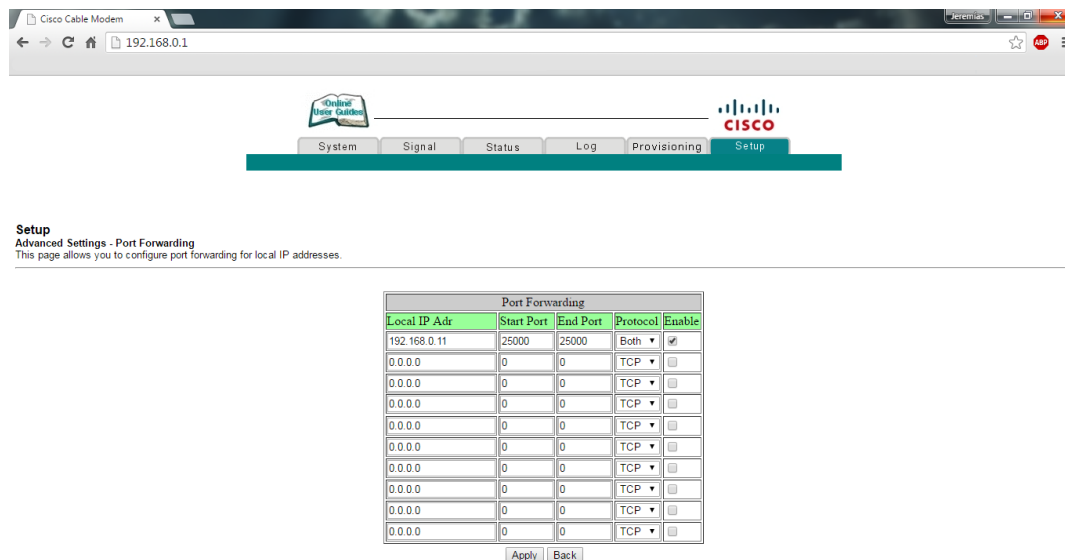


Ilustración 52 - Sección de Port Forwarding del modem Cisco DPC2320.

Este mismo procedimiento debe llevarse a cabo también con el router R, es decir, aquel correspondiente a la red del ordenador R, de manera que al momento de vincular ambos ordenadores los router no interfieran con la comunicación entre ellos, permitiendo la conexión de los mismos.

3.6.4.1.3 Correspondencia en el código

En este inciso se procederá a explicar las líneas de código clave, que permiten controlar el movimiento del robot desde el programa iCC.

Lo primero a señalar es que en el diseño del formulario que funciona como interfaz, se declaran las siguientes líneas (entre muchas otras configuraciones iniciales):

```
this.btnKeyboardControl.KeyDown += new
    System.Windows.Forms.KeyEventHandler(this.button1_KeyDown);

this.btnKeyboardControl.KeyUp += new
    System.Windows.Forms.KeyEventHandler(this.button1_KeyUp);
```

Estas dos declaraciones del botón “WASD Keyboard Control” (*btnKeyboardControl*) lo que hacen es asignar a los eventos *KeyDown* (Se presiona una tecla) y *KeyUp* (Se deja de presionar una tecla), los procedimientos *button1_KeyDown* y *button1_KeyUp* respectivamente.

Esto significa que al activar el botón “WASD Keyboard Control” y luego presionar las teclas de movimiento anteriormente indicadas, se ejecutarán los procedimientos *button1_KeyDown* y *button1_KeyUp* según corresponda.

Cabe destacar además, que cuando el botón “WASD Keyboard Control” es clickeado, ejecuta la siguiente línea de código:

```
robot.StartInFullMode();
```

Aquí, *robot* es una instancia de la clase *iRobotCreate*, la cual representa al robot físico y en ella se encuentran relacionadas las operaciones que puede ejecutar el Create. Dichas operaciones se corresponden numéricamente, y según el número enviado al robot, éste ejecutará una acción en particular: en este caso *StartInFullMode* es un procedimiento de la clase *iRobotCreate*, que ejecuta los comandos *Start* (código de operación 128) y *Full* (código de operación 132) los cuales hacen que el Create se encienda, iniciando todos sus sensores, motores, actuadores, etc.

Aclarado esto entonces, al presionarse alguna de las teclas de movimiento y ejecutarse los procedimientos *button1_KeyDown* y *button1_KeyUp*, los mismos comienzan inicializando una variable por defecto que servirá como valor para indicar la velocidad de los motores:

```
int offset = 100;
try
{
    offset = int.Parse(tbKeyBoardSpeed.Text);
}
catch (Exception ex)
{
    Console.Out.WriteLine(ex);
}
```

El bloque Try que le continúa a dicha asignación es para cambiar el valor de la velocidad en caso de que el usuario haya definido una por sí mismo.

Luego de esto, procede a reconocer cuál tecla fue la que se presionó y lanzó el evento:

```
if (KeyState[(int)Keys.W])
    y -= offset;
if (KeyState[(int)Keys.S])
    y += offset;
if (KeyState[(int)Keys.A])
    x -= offset;
if (KeyState[(int)Keys.D])
    x += offset;
```

Aquí las variables X e Y representan el plano horizontal y vertical respectivamente. A continuación de esta detección de la tecla, las variables son utilizadas para ejecutar el procedimiento *UpdateJoystickAndDriveCommand*:

```
MouseEventArgs me = new MouseEventArgs(MouseButtons.Left, 0, x,y, 0);
UpdateJoystickAndDriveCommand(me);
```

Este procedimiento (como lo indica el nombre) comienza actualizando datos que son correspondientes al Joystick (que se encuentra en la tercer pestaña de la interfaz del programa, pero no es utilizado en este trabajo para mover el robot). La Ilustración 53 muestra dicha pestaña.

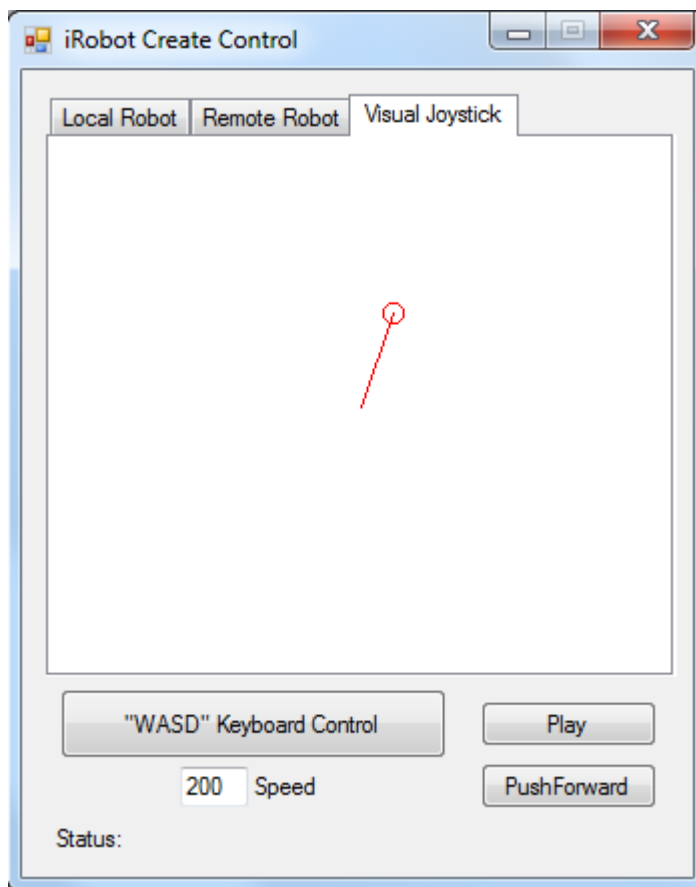


Ilustración 53 – Pestaña del Joystick del programa iCC.

Continuando el procedimiento *UpdateJoystickAndDriveCommand*, luego de realizar las conversiones correspondientes al Joystick, invoca al método *DriveDirect* de la instancia *robot* indicada más arriba:

```
robot.DriveDirect(left, right);
```

A dicho método se le envían los valores correspondientes a las velocidades de la rueda izquierda (*left*) y derecha (*right*) del robot. Este método chequeará primero si el robot está conectado, y en caso afirmativo procederá a asignar las velocidades correspondientes a cada rueda para luego enviar dicha información al método *DriveThreadCallBack*.

Finalmente este último, con toda la información recibida, enviará al robot la instrucción de ejecutar **DriveDirect** (cuyo código de operación es 145) junto con las velocidades correspondientes a cada rueda. Esta operación se encargará entonces de poner en movimiento las ruedas del robot según la velocidad establecida, y según la tecla que haya presionado el usuario.

3.6.5 Solución escogida

A partir de la investigación de todas las herramientas que se detallaron en esta sección de alternativas tecnológicas, y principalmente gracias a la modificación y uso del programa “*iRobot Create Control*” mencionado anteriormente, surgió de manera bien definida la solución elegida para este trabajo, la cual se describirá a continuación.

Como se explicó, para el movimiento del robot será utilizado el programa iCC, lo que permitirá controlarlo desde la ubicación remota en donde se encuentre el usuario con el ordenador R; tanto el ordenador R, como el B, serán notebooks personales del alumno (ver Ilustración 55 e Ilustración 57).

Para la conexión entre el ordenador B y el robot, se utilizara el dispositivo bluetooth BAM descrito anteriormente (ver Ilustración 14), del cual dispone la Universidad. Como el ordenador irá montado sobre el robot, no habrá inconvenientes respecto a limitaciones de distancia entre ellos.

Por último para la comunicación audiovisual entre los ordenadores, se utilizará el software Skype (ver Ilustración 32) detallado entre los primeros párrafos de estas secciones dedicadas a las tecnologías existentes. El mismo permitirá que el usuario del *Ordenador R*, pueda ver y escuchar el ambiente en el cual se hace presente, también pudiendo ser visto y escuchado en dicha ubicación.

En este punto también es necesario aclarar que el ordenador portátil que se utilizará para el prototipo (ver Ilustración 55), el cual funcionará como *Ordenador B*, actualmente tiene fuera de funcionamiento su cámara, por lo que se utilizará una cámara web USB externa (también visible en la Ilustración 55) como medio que permitirá al usuario del *Ordenador R* visualizar la ubicación en la que se hace presente.

En caso de que se logre construir un soporte adecuado para el ordenador, se buscará que el mismo se eleve al menos 1 metro del suelo de modo que las personas que se acercan a interactuar con el robot puedan enfocar su vista a una altura relativamente cómoda, como lo hacen cuando se relacionan con otra persona. Partiendo desde este principio, la cámara web se colocará sobre la pantalla del ordenador, como la Ilustración 55 lo muestra.

Se busca que la interacción con el usuario remoto, sea lo más fluida y visualmente cómoda posible para las personas que se acercan a la herramienta. De esta manera, enfocarán su vista en la pantalla del ordenador donde se muestra al usuario remoto, y él a su vez podrá visualizar la cara de las personas con quien habla ya que ésta ubicación de la cámara permitirá enfocarlos correctamente. Principalmente se eligió colocar la cámara de esta manera, ya que si se la coloca sobre el robot y no sobre el ordenador, la visión del usuario en la ubicación remota estará al nivel del suelo, o enfocar los pies de la persona que se aproxime eventualmente a interactuar con la herramienta, y no su rostro.

En cuanto a las pruebas que se realizaron con los ordenadores y las conexiones entre ellos, para llegar a la solución de este trabajo, fueron necesarias ciertas modificaciones en el código del programa, de manera que pudieran hacerse experimentos a través de la función que posee, para

conectar dos ordenadores a través de internet, sin necesidad de que hubiese conectado un robot a alguno de los ordenadores.

En el capítulo siguiente se explica cómo el alumno, aplicando conocimientos adquiridos durante el cursado de su carrera, realizó dichas modificaciones en el código del programa, obteniendo el funcionamiento necesario para llegar a la solución alcanzada. Esto le permitió realizar varias pruebas, que se explican detalladamente junto con sus resultados, en la Unidad 5 de este trabajo.

4. SOLUCIÓN PROPUESTA

4.1 Establecer la solución aportada

En la siguiente imagen podemos ver un diagrama de los elementos que se ven involucrados en la solución que se eligió para este trabajo.



Ilustración 54 - Diagrama del Robot de Telepresencia

La solución propuesta consistirá en el uso del robot que la Universidad dispone, junto con dos ordenadores notebook que se comunicarán a través de internet mediante un software.

Uno de los ordenadores (*Ordenador B*) estará conectado al robot vía bluetooth, y montado sobre él usando algún tipo de soporte de apoyo; el otro ordenador (*Ord. R*) será utilizado como herramienta mediante la cual el operador controle remotamente el movimiento del robot y a la vez, se haga presente visual y auditivamente en la ubicación donde se encuentra el robot de telepresencia.

Esta herramienta estará conformada por tres aparatos físicos bien diferenciados que son el robot Create por un lado, y dos ordenadores notebook del alumno por el otro. En la Ilustración 54 podemos ver cómo estarán vinculados entre sí estos elementos.

Mediante el uso de un programa (*iRobot Create Control*) que correrá en ambos portátiles al mismo tiempo, las órdenes de movimiento se enviarán al robot a través de Internet desde el ordenador remoto.

A continuación, se describen los dispositivos que forman parte del prototipo que se desarrolló para mostrar cómo funcionaría la solución propuesta para este problema.

4.1.1 Ordenadores

Un ordenador notebook funcionará tanto como dispositivo audiovisual de videoconferencia, como medio al que se accederá remotamente para controlar el robot Create en el espacio donde éste se encuentre, a través de una conexión bluetooth con dicho ordenador.



Ilustración 55 – Ordenador notebook local (Ordenador B).

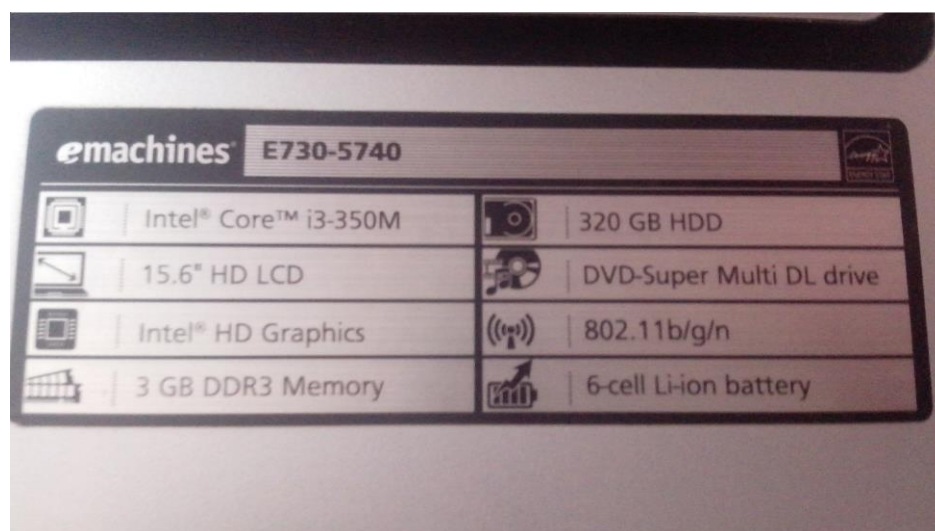


Ilustración 56 – Características del ordenador local.

En la Ilustración 55 podemos ver una imagen del ordenador, y en la Ilustración 56 sus características principales. El mismo corre un Windows 7 de 64 bits con sus actualizaciones al día, y en él se ejecutará el software que permitirá que el movimiento del robot sea controlado a través de otro ordenador.

Vale aclarar además, que la cámara web que se puede ver sobre la pantalla del ordenador en la Ilustración 55, se utilizará como reemplazo debido a que no funciona la cámara que trae propiamente la notebook.

Esto en conjunto es el primer dispositivo clave de la solución, que irá montado sobre el robot mediante algún tipo de soporte seguro, y permitirá el acceso remoto al control del robot de telepresencia.

Con un prototipo, se mostrará la forma en que el “Create” puede ser controlado por un ordenador remotamente ubicado (*Ordenador R*), a través de la comunicación que entablará con el ordenador local (*Ordenador B*). En la Ilustración 57 se puede ver dicho ordenador remoto.

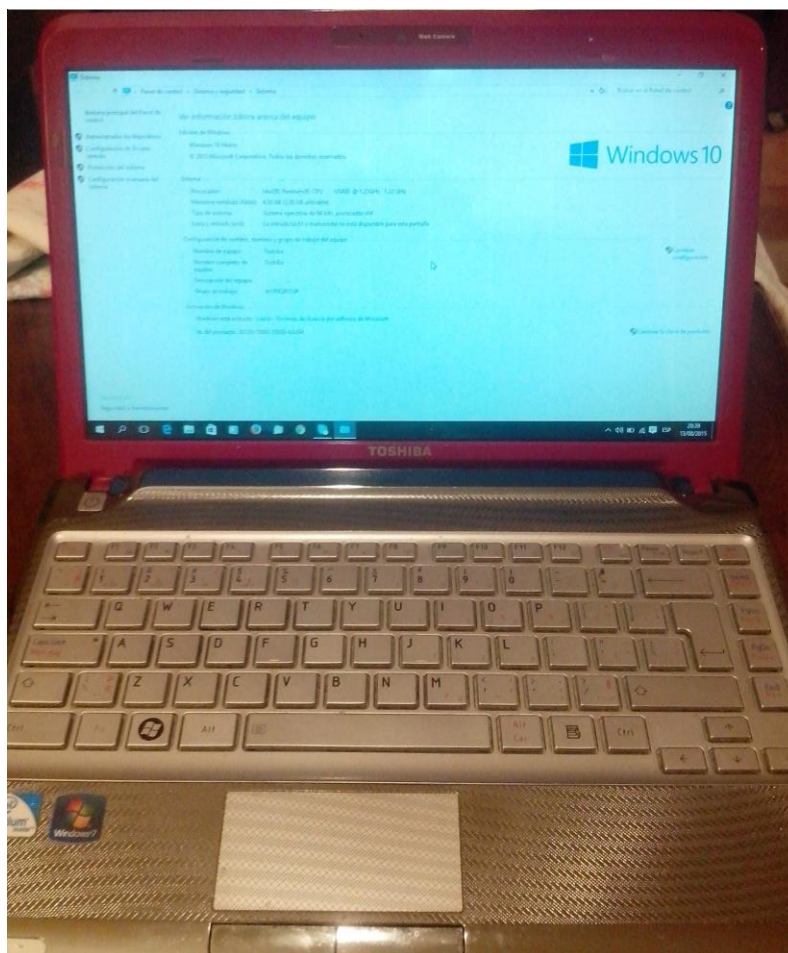


Ilustración 57 – Ordenador notebook remoto (Ordenador R).

4.1.2 Robot Create

En la Ilustración 13 podemos ver una imagen del robot Create que la Universidad adquirió, y el cual se utiliza como elemento que provee la movilidad de la solución.

Aun así debido a su liviandad y tamaño, resulta excelente para mostrar cómo se puede implementar la telepresencia, con una plataforma de muy bajo costo en relación a los gastos que implican las soluciones de este tipo que se desarrollan a mayor escala en el mundo.

La Ilustración 14 muestra un acercamiento al dispositivo bluetooth que se usa para la conexión con el *ordenador B*; el mismo se inserta en el “*Cargo Bay Connector*” del robot, y se vincula con el ordenador mediante un conector USB que funciona como emisor y receptor de señales bluetooth.

Este otro conjunto es el segundo dispositivo clave para la solución, sobre el cual se montará el ordenador de la Ilustración 55. De esta manera se brinda movilidad al usuario que se conecta remotamente al lugar donde el robot de telepresencia se encuentra.

4.1.3 Programa “iRobot Create Control”

Tanto el segundo ordenador como el primero, correrán ambos, este mismo programa mencionado en el capítulo anterior. Este permitirá la vinculación entre el ordenador remoto (*ordenador R*) y el robot, a través del ordenador local (*ordenador B*) que será parte del robot de telepresencia.

Para la utilización de este programa en el proyecto, fueron necesarias ciertas modificaciones en el código fuente del mismo, ya que al momento de probarlo luego de su descarga, se presentaron algunas fallas en su funcionamiento que hacían que el programa se detuviera mientras era ejecutado.

El creador de este programa, Johnny C. Lee, expone el trabajo que hizo en su blog personal, y en el mismo ya nos advierte que diseñó el software a su propia medida. Es decir, se aseguró de que funcionara correctamente, pero sólo en sus ordenadores correspondientes, por lo que era de esperarse que el alumno se encontrara con fallas en el programa cuando lo utilizase en ordenadores diferentes.

Mediante la utilización de la herramienta *Microsoft Visual Studio 2010* (VS2010) y la aplicación de los conocimientos sobre la misma, adquiridos durante el cursado de la carrera, ante estas fallas presentadas se prosiguió a depurarlas en VS2010 de manera que se descubrieran las porciones de código exactas que hacían que el programa iRobot Create Control (iCC) se detuviera.

A partir de estos descubrimientos, se pudieron realizar entonces modificaciones en esas porciones de código defectuosas, de manera que el programa funcionase sin detenerse, y que tampoco se vieran comprometidas ni modificadas sus funcionalidades, las cuales lo hacen crucial para este proyecto. A continuación, se describen las dos fallas claves que se presentaron y solucionaron durante la realización de este proyecto.

4.1.3.1 Primera Modificación: Obtención de dirección IP Externa

La primera falla en el programa, se hizo visible durante los primeros experimentos de uso, cuando se intentaba habilitar la función de redirección de puertos (*Port Forwarding*) con el botón “*Enable Network Control*” (ver Ilustración 44 e Ilustración 45). Al querer habilitar dicha función, se vio que el programa realiza una consulta a una dirección de internet (<http://checkip.dyndns.com>), la cual devuelve la *dirección IP Externa* del ordenador en el cual está corriendo el programa.

Esta consulta en algunos casos fallaba; a veces por conflictos internos del programa a la hora de transformar la información devuelta por la consulta, y a veces también porque dicha dirección no estaba funcionando bien al momento de realizarse la consulta, de manera que no se podía accederla correctamente, y entonces el programa iCC se detenía.

Al tenerse ya definido un PC principal que se utilizaría como *Ordenador B* para las distintas pruebas a realizar, no era necesario que cada vez que se corriera el programa iCC, se realizara la consulta de la dirección IP externa; ya que el ordenador sería siempre el mismo, por lo tanto también dicha IP. En la Ilustración 58 se muestra resaltada la porción de código donde esta primera falla ocurría, al intentarse realizar la consulta a Internet.

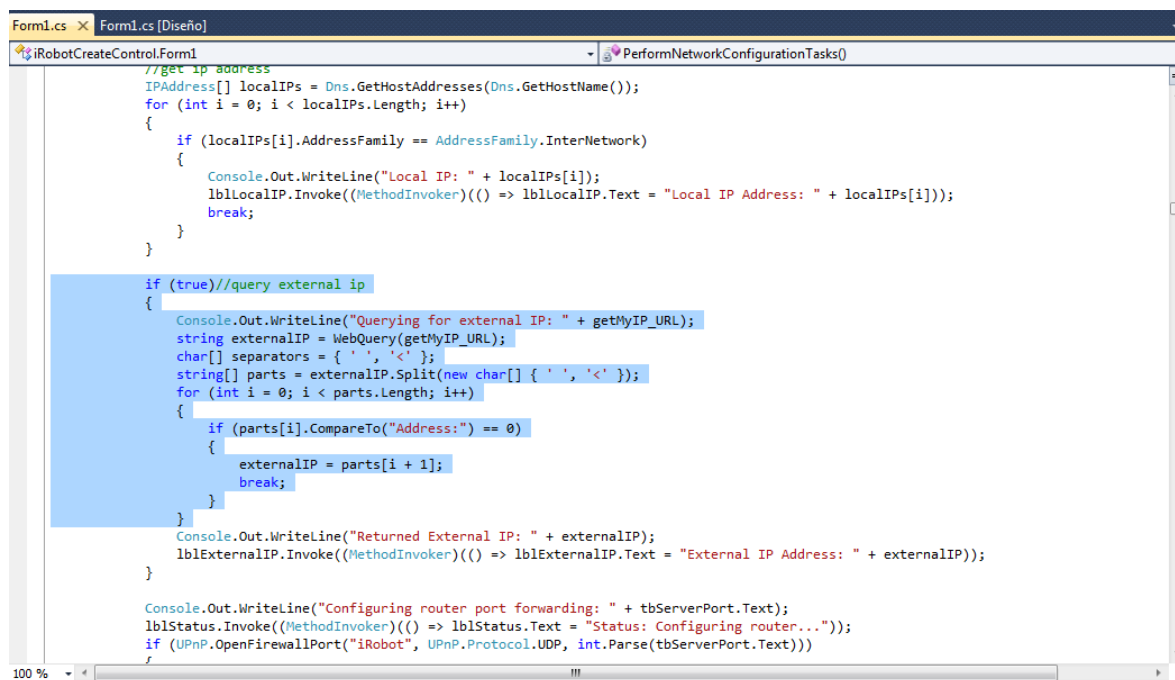


Ilustración 58 – Captura del código donde se ocasionaba la primera falla

La solución a esta falla se logró agregando un bloque *Try/Catch* a la consulta, y estableciendo dentro del bloque *Catch*, una variable fija correspondiente a la dirección IP externa:

```
externalIP = "190.2.126.133";
```

De esta forma, al habilitar la opción “*Enable Network Control*” el programa iCC en primera instancia realiza la consulta a través de Internet, y en caso de que se genere un problema durante la misma, evita que el software se detenga ingresando al bloque *Catch*, donde asignará automáticamente la dirección IP a la variable en cuestión.

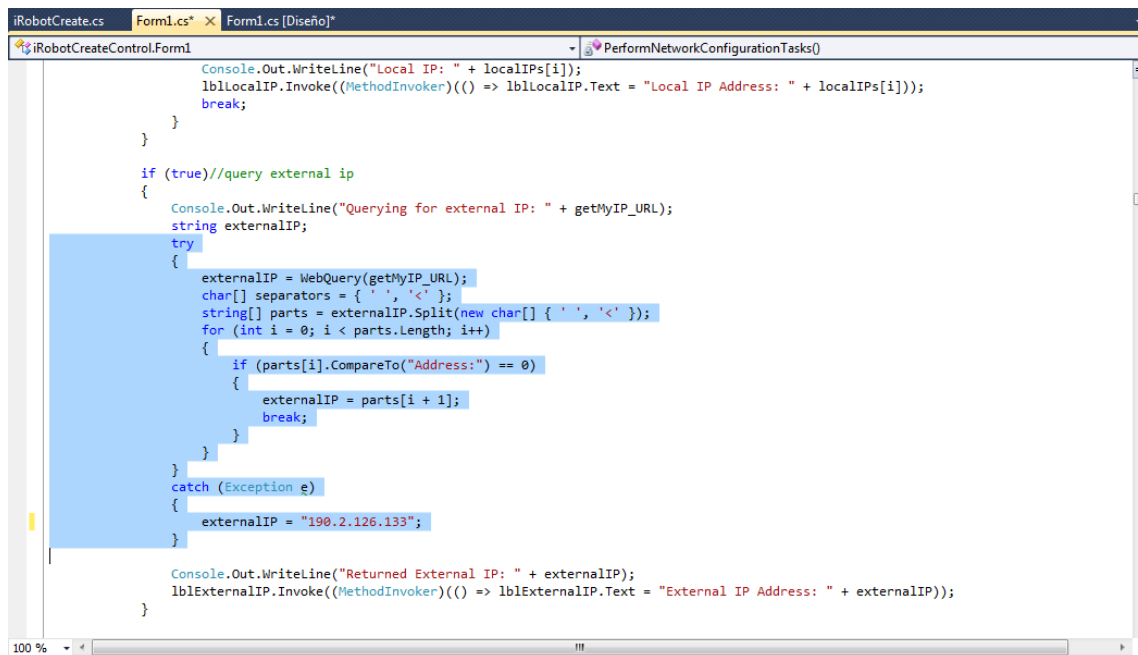


Ilustración 59 - Captura del código modificado para solucionar la primera falla.

En la Ilustración 59 vemos una captura de la reforma realizada al código para llegar a la solución de esta primera falla. Como podemos ver en las líneas resaltadas, se realiza una declaración fija de la variable **externalIP** dentro del bloque *Catch*, a donde el programa ingresará en caso de que la consulta dentro del *Try* no tenga éxito.

Así fue como se solucionó esta falla, asegurando la correcta activación de la función correspondiente al botón “*Enable Network Control*”, en el ordenador B. Una vez implementada esta solución, al proseguir con la realización de las pruebas, se descubrió entonces una segunda falla en el programa, la cual también causaba que el mismo se detuviera mientras era ejecutado. A continuación se describe la solución a dicha falla.

4.1.3.2 Segunda Modificación: Porcentaje de carga de la batería del robot

Tal como se explicó en capítulos anteriores sobre las funcionalidades del programa iCC, posee en la interfaz un indicador del nivel de carga que tiene la batería del robot (ver Sección 2.4 o también Ilustración 48) de manera de retroalimentar al usuario que utiliza la herramienta. La segunda falla encontrada en el programa se correspondía con esta funcionalidad justamente.

La mayoría de las pruebas con el programa iCC se realizaron en el hogar del alumno, por lo que no se disponía de un robot para conectar con al PC que actuaría como *Ordenador B*. Una vez

arreglada la primera falla, al probar conectar un ordenador portátil (*ordenador R*) a la PC a través de la red del hogar, la petición de conexión del ordenador R al B se llevó a cabo correctamente, pero en ese momento el programa se detuvo nuevamente.

Al depurarlo a través de la herramienta VS2010, se detectó que la falla ocurría en el momento de la conexión entre los ordenadores, ya que luego de ese evento, el programa iCC procedía a recopilar información del robot; al no haber un robot conectado, el error se producía cuando el programa intentaba mostrar en la interfaz el nivel de carga de la batería.

En la Ilustración 60 se muestra en resaltado, la línea de código específica que provocaba la detención del programa:

```
int robotCharge = (f.robot.sensorState.BatteryCharge * 100 / f.robot.sensorState.BatteryCapacity);
```

Al no estar conectado el robot al ordenador B, cuando el programa intentaba determinar el nivel de carga de la batería, se producía un error en la división, ya que la variable *BatteryCapacity* de los sensores del robot era igual a 0 (cero). Esta división por cero probablemente no fue contemplada por el creador del programa, ya que al tener el robot conectado este error nunca se produce.

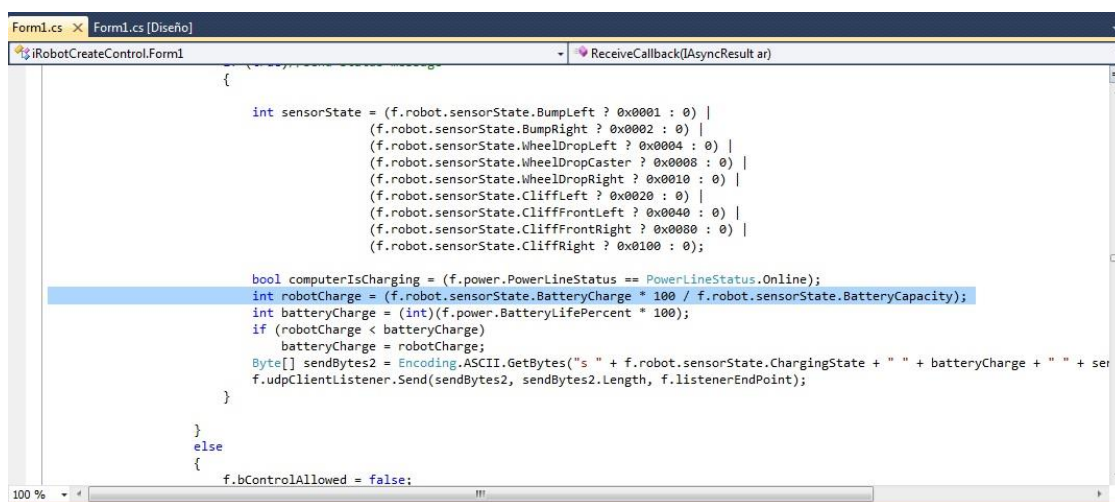


Ilustración 60 - Captura del código donde se producía la segunda falla.

De esta manera, la corrección implementada consistió en añadir un bloque condicional (*IF*) que encerrara la porción de código que provocaba el error. En la Ilustración 61 se muestra una captura de pantalla de cómo quedó finalmente el código una vez añadido dicho bloque IF.

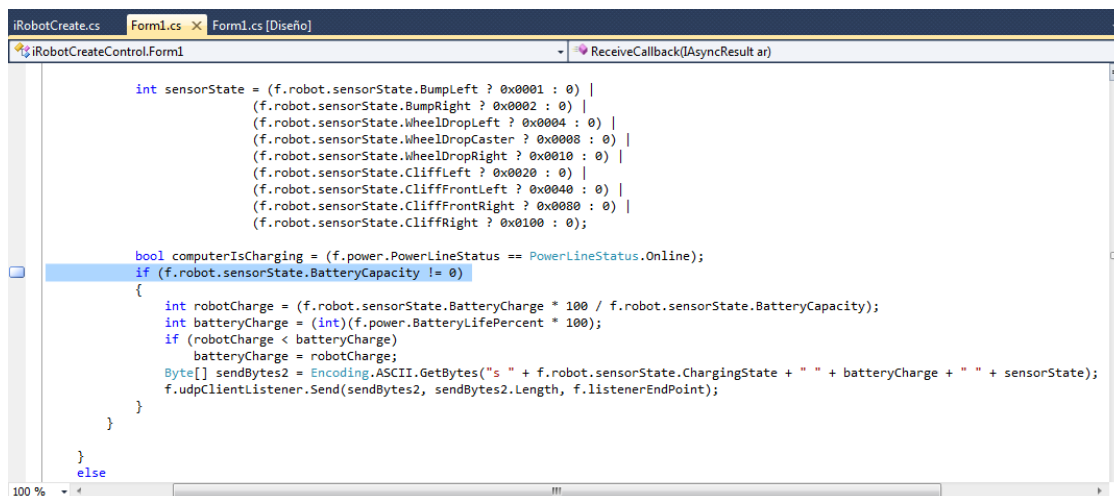


Ilustración 61 - Captura del código una vez implementada la solución de la segunda falla.

Como se puede ver, la solución consistió en que la condición del bloque IF fuese que el valor de la variable “*BatteryCapacity*” sea distinto de cero; de esta manera si no había un robot conectado esta porción del código no se ejecutaba, evitándose así la división por cero que provocaba la detención del programa.

```
if (f.robot.sensorState.BatteryCapacity != 0)
```

Aplicada esta modificación en el código fuente, el programa iCC permaneció funcionando correctamente luego de conectar los ordenadores, aún sin que hubiese un robot vinculado al Ordenador B.

Gracias a que se solucionaron estas fallas en el programa, se pudieron realizar variadas pruebas con el mismo, las cuales permitieron llegar a la solución que se presenta en este trabajo. Dichas pruebas serán descritas en el capítulo siguiente.

4.2 Análisis económico y financiero

En esta sección se detallan los costos que se calcularon durante el análisis de este trabajo. Dichos cálculos se detallarán en cuanto a costos correspondientes a *hardware*, *software* y recursos humanos.

4.2.1 Costos de Hardware

Hardware	Cantidad	Precio
Robot Create	1	\$ 1.950,00
Element Direct BAM	1	\$ 675,00
Notebook eMachines Intel i3 c/ HD Graphics + Pantalla HD de 15.6’’ + 320 GB + 3GB RAM	1	\$ 6.700,00
Webcam Logitech C170	1	\$ 260,00
Adaptador Bluetooth	1	\$ 70,00
Total		\$ 9.655,00

Tabla 1 – Costos de Hardware.

Costos calculados en base al valor del dólar tomado el día 30/11/15 (\$ 9.65) (Cotización Dolar, 2015).

En este análisis fue incluido solo uno de los ordenadores (correspondiente a *ordenador B*), que es el cual formará parte del robot de telepresencia. El *ordenador R* no fue incluido debido a que el mismo variará según las distintas personas que utilicen la herramienta, también desde diferentes ubicaciones, por lo que utilizarán sus propios ordenadores. Tampoco fue incluido el costo potencial de construir un soporte para montar el ordenador sobre el robot.

4.2.2 Costos de Software

Software	Cantidad	Precio
Software “iRobot Create Control”	1	\$ 0,00
Skype	1	\$ 0,00
Microsoft Visual Studio Professional 2010 ¹	1	\$ 0,00
Team Viewer	1	\$ 0,00
Total		\$ 0,00

Tabla 2- Costos de Software.

¹ Con licencia de uso, otorgada por la Universidad al alumno durante el cursado.

En este punto cabe aclarar que si bien las herramientas TeamViewer y Visual Studio no son utilizadas por el robot de telepresencia, el uso de las mismas fue clave en este proyecto tanto para las modificaciones que se debieron implementar al Software “iRobot Create Control”, como para las pruebas que con él se hicieron. Es por esto que se los incluye en el análisis de costos de *software*.

4.2.3 Costos de Recursos Humanos

Recurso Humano	Horas	Precio por Hora	Subtotal
Analista/Programador/Diseñador y Testeo	250	\$ 85,00	\$ 21.250,00
Total			\$ 21.250,00

Tabla 3- Costos de Recursos Humanos.

4.2.4 Costo total

Ámbito	Costo
Hardware	\$ 9.655,00
Software	\$ 0,00
Recursos Humanos	\$ 21.250,00
Costo total final	\$ 30.905,00

Tabla 4- Tabla Final de Costos.

Costos calculados al día 30/11/2015.

Según los cálculos realizados en base al *hardware*, *software*, y recursos humanos involucrados en este proyecto, la cotización final del mismo supera apenas los treinta mil pesos (\$ 30.000), un precio sumamente económico respecto a otras herramientas de telepresencia existentes en el mercado.

De todas formas, gracias a que el laboratorio de la Universidad permitirá usar el robot y el módulo BAM, se dispone de todo el hardware necesario ya que el alumno posee la Webcam y su Notebook, las cuales utilizará para el proyecto. El *software* tampoco requiere gastos debido a que las licencias de uso de ambos son gratuitas.

Además será el alumno quien realice las tareas de análisis, diseño, programación y testeo, por lo que no será necesaria la contratación de recursos humanos.

De esta manera, particularmente para este proyecto, no hará falta realizar gastos económicos.

4.3 Análisis de los riesgos

A continuación, se muestra a través de una tabla el análisis de riesgos realizado para el proyecto en cuestión, donde se podrá ver tanto la probabilidad de ocurrencia y el impacto de los riesgos posibles, como las acciones preventivas que se pueden implementar para minimizar los efectos de esos eventos.

Riesgo	Probabilidad de Ocurrencia (1-5)	Impacto	Contingencias
El robot falla o se rompe.	1	Alto. Este y otros proyectos dependen de su buen funcionamiento.	Mantener el robot dentro de las instalaciones de la Facultad. Dejar que el personal del laboratorio manipule la herramienta como corresponde.
Pérdida de la información.	2	Alto. Generaría tener que implementar acciones de recuperación de información.	Almacenar los datos en la nube. Almacenar respaldo de los datos en otros dispositivos de almacenamiento.
Falla en los dispositivos, o incompatibilidad de los mismos.	3	Medio. Generaría necesidad de buscar opciones diferentes para la solución final.	Realizar pruebas exhaustivas para evaluar el funcionamiento, y así decidir correctamente la tecnología a utilizar.
Desvío de los tiempos establecidos para el proyecto	4	Medio. Necesidad de replantear los plazos de entrega.	Trabajar dentro de los límites establecidos. Considerar factores externos que puedan provocar desviaciones.

La herramienta robótica no está disponible para ser utilizada	4	Bajo. Puede ser utilizada en otro momento, aunque puede generar desvíos en los tiempos establecidos.	Coordinar con el personal del laboratorio de la Facultad, y con demás compañeros que utilizan la herramienta para sus proyectos.
Variaciones de los costos calculados	5	Bajo. Debido a la naturaleza académica de este trabajo, no presenta mayor riesgo a nivel económico.	Considerar las variaciones en el mercado. Realizar cálculos y análisis financieros lo más actualizados posible.

Tabla 5 - Matriz de análisis de riesgos.

Tabla de equivalencias en relación a la probabilidad de ocurrencia:

Probabilidad	Equivalencia
5	Casi seguro
4	Probable
3	Medio
2	Poco probable
1	Casi nunca

Tabla 6 - Tabla de equivalencias de probabilidad.

5. VERIFICACIÓN EXPERIMENTAL

5.1 Primeros pasos

En esta sección se detallan las pruebas y experimentos realizados en este trabajo, donde se explicará la manera en que dan por resultado al prototipo elegido como solución propuesta.

La solución se puede dividir en dos puntos clave que son: la parte móvil, donde se usa el robot Create como elemento de transporte, y sobre el cual se monta la segunda parte; que involucra el ordenador notebook de la Ilustración 55 y representa la parte de comunicación audio-visual de esta solución; dos partes que en conjunto dan sentido al término de robot de telepresencia.

Inicialmente se realizó un experimento preliminar que involucró las tecnologías de TeamViewer, Microsoft Robotics Developers Studio y Skype. En el mismo, se utilizaba los dos primeros programas (TeamV. y MRDS) para manipular el movimiento del robot remotamente: el ordenador B corría un algoritmo programado en MRDS, mientras que desde el ordenador R se accedía a través de TeamViewer al ordenador B, para ejecutar los comandos de movimiento. A la par de esto, se entablaba una videollamada vía Skype entre ambos ordenadores, con lo cual se completaba el sentido de telepresencia de la herramienta. Sin bien este prototipo funcionó, el rendimiento del mismo fue muy lento, debido principalmente al peso y los requerimientos que estos tres programas precisaban para ejecutarse, así como tampoco era óptima la calidad de la llamada, debido a la cantidad de ancho de banda que los mismos consumían.

Con el avance del trabajo y las investigaciones involucradas, se encontró la solución aquí propuesta, en una herramienta que condensaba tanto las funciones de controlar el movimiento del robot, cómo permitir la vinculación entre ordenadores remotamente ubicados, en un solo programa. En este capítulo se detallarán las pruebas y resultados obtenidos con el uso de este software.

El orden entonces aquí elegido, para mostrar estos ensayos y resultados no es cronológico sino que, más bien contiene una lógica gradual, es decir que, a medida que se avance en la lectura de estas pruebas se podrá entender de manera progresiva, cómo se llegó al prototipo final que se propone como solución.

Para acompañar este avance progresivo, se puede ver en la sección de los anexos (Anexo 3) el diagrama de GANTT que acompaña al proyecto, donde se plasman en función del tiempo, los esfuerzos que implicaron las tareas involucradas en este trabajo.

Además cabe aclarar al lector, que para la cantidad establecida a cada tarea se consideró, y por lo tanto está incluido, el tiempo de redacción que implicó en este informe cada una de esas diferentes tareas.

5.2 Pruebas realizadas

A continuación se detallan los distintos testeos que se realizaron, y los resultados que se obtuvieron gracias a los mismos. Se podrá ver también hacia el final de estos, cómo se fue llegando gradualmente al prototipo final de este trabajo.

Antes de comenzar, cabe señalar que para todas las pruebas que aquí se detallarán, el método elegido y utilizado para mover el robot es aquel que involucra las teclas W, A, S y D del teclado, que se relacionan con el botón “WASD Keyboard Control” de la interfaz del programa iCC.

En la Sección 3.6.4.1.3 del capítulo 3 se explica en detalle la correspondencia entre las órdenes enviadas desde el teclado, y las órdenes de movimiento que el robot Create recibe.

5.2.1 Navegación del robot desde Ordenador B

Entre las principales funciones que debía cubrir esta herramienta de telepresencia, se encontraba brindar al usuario, la posibilidad de moverse libremente dentro de la habitación donde se hace presente. Las respectivas pruebas realizadas, consistieron entonces en comprobar que el robot Create recibiera correctamente las órdenes de movimiento, desde el Programa *iRobot Create Control* (iCC).

Para esto se coordinó un encuentro en el laboratorio de robótica de la Facultad, con el objetivo de realizar una prueba concreta con el robot real. Mediante la asistencia de quienes conforman el equipo del robótica, se conectó por vía Bluetooth el robot a la notebook del alumno (*ordenador B* – Ver Ilustración 55).

Una vez que el ordenador detectó la vinculación con el robot, le asignó un número de puerto (ej. “COM3”) y se procedió a iniciar el programa iCC. En la casilla para indicar el puerto serial (ver Ilustración 44), se agregó el puerto que el ordenador había asignado, y al dar clic en el botón “Connect To Local Robot” el robot emitió un pitido de confirmación, y la terminal del programa a la vez, confirmó la conexión establecida con el robot en el puerto indicado.

Luego de esto, activando los controles de movimiento con la opción “WASD Keyboard Control” (ver Ilustración 44), al presionar la tecla W para mover el robot hacia adelante, el mismo respondió inmediatamente avanzando una corta distancia, la cual fue acorde a la duración que se mantuvo presionada dicha tecla.

En la Ilustración 62 vemos una imagen del robot Create al momento de las pruebas, también se observa sobre el mismo una Webcam que dispone la facultad, la cual se montó sobre el robot en ese momento y se conectó al ordenador, para tener imagen en pantalla de hacia dónde se dirigía el robot mientras se lo movía por la habitación.

El movimiento del mismo fue probado durante algunos minutos en los cuales se envió los cuatro comandos de movimientos posibles que el robot podrá realizar (avanzar, retroceder, girar hacia la izquierda o hacia la derecha).

La respuesta a cada uno de estos movimientos fue totalmente satisfactoria: la prueba se realizó en una habitación del laboratorio de robótica de la Facultad, elegida por encontrarse libre

de obstáculos, con lo cual se logró un ambiente lo más espacioso posible para el robot, a la vez que esto brindó la correspondiente seguridad al mismo, en caso de ocurrir algún evento no deseado.



Ilustración 62 - Robot Create durante la prueba de navegación.

Esta prueba dio como resultado la confirmación de que el movimiento del robot puede ser controlado a través del programa iCC, utilizando Bluetooth como vía de conexión entre el ordenador y el robot.

A partir de obtener esta confirmación, lo siguiente a comprobar fue la posibilidad de vincular un ordenador remoto (*ordenador R*), al notebook (ver Ilustración 55) que aquí el alumno utilizó en carácter de *ordenador B*.

5.2.2 Conexión entre ordenadores a través de una red local

Como se menciona al final del paso anterior, una vez confirmada la capacidad del programa iCC para controlar el movimiento del robot, lo siguiente a comprobar fue la conexión entre los ordenadores.

Para estas pruebas, se utilizó en primera instancia una red de área local (LAN del hogar del alumno) como escenario en el cual probar la conexión entre los dos ordenadores portátiles del alumno; solo luego de esto se intentó a través de una red externa de Internet, prueba que se detallará en el siguiente ítem.

Para empezar se inició en ambos ordenadores el programa iCC, activando en uno el control a través de la red, con el botón “*Enable Network Control*” (ver Ilustración 44). Una vez activada esta funcionalidad, en el segundo ordenador se completó los campos de la conexión y al presionar el botón “*Connect To Remote Robot*” ambos ordenadores indicaron que la conexión se había establecido correctamente.

En la Ilustración 63 se muestra una captura de pantalla del ordenador B una vez ya establecida la conexión con el ordenador remoto que envía la petición. En la terminal de mensajes de la imagen, se indica respectivamente con flechas rojas, el momento en que se habilita la función de *Port Forwarding*, donde el ordenador B queda a la espera de una conexión entrante; y luego el momento en que la conexión con el ordenador R se establece correctamente, y es indicado en la terminal con la línea señalada.

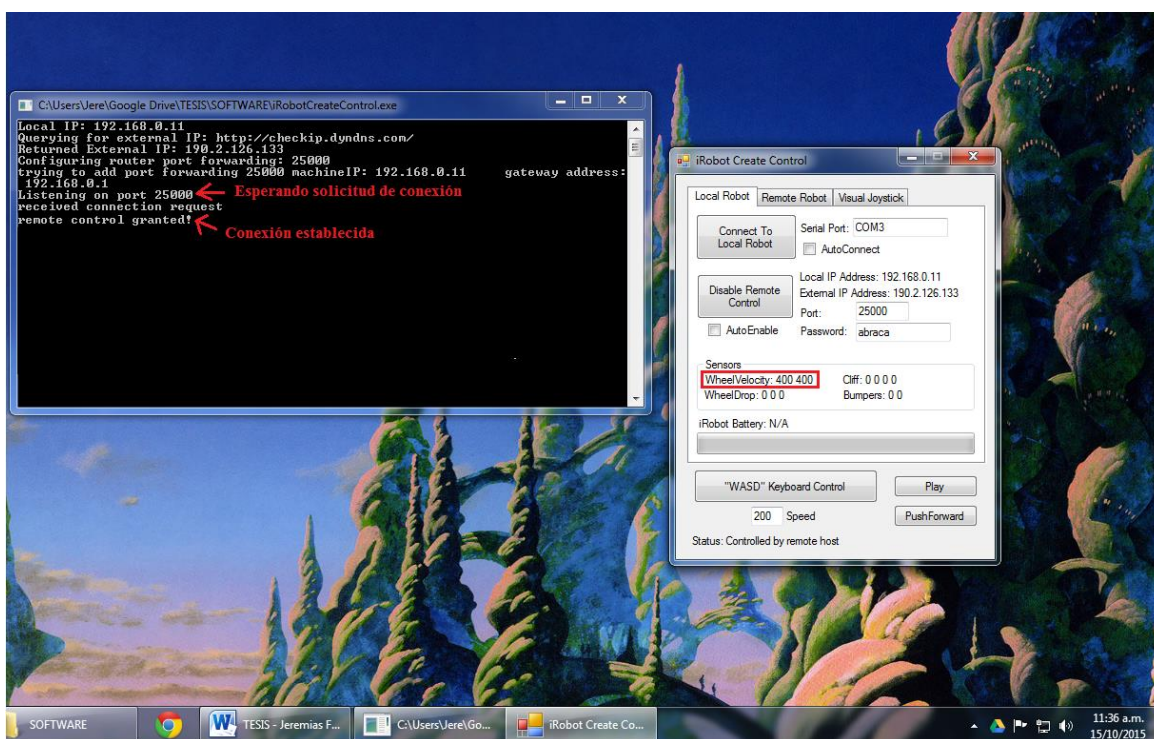


Ilustración 63 - Ordenador B recibe petición y establece conexión con Ordenador R.

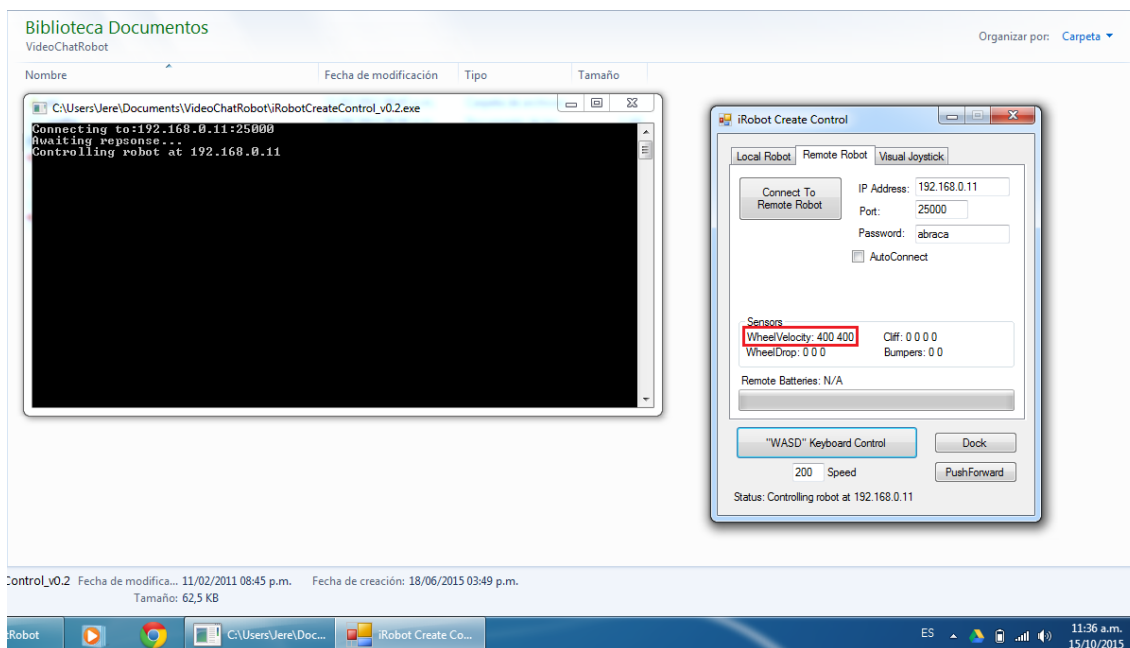


Ilustración 64 - Ordenador R conectado al ordenador B.

Por otro lado, vemos en la Ilustración 64 una captura de pantalla del ordenador R, en el mismo momento que la imagen anterior, cuando la conexión se establece.

Mediante un recuadro de color rojo, se señala cómo en ambas ilustraciones podemos ver que la comunicación entre los ordenadores ocurría perfectamente, enviando (Ord. R) y recibiendo (Ord. B) las señales de movimiento ejecutadas desde el ordenador R, con la opción “WASD Keyboard Control” activada, la cual en el panel de sensores muestra la velocidad a la que están haciéndose funcionar los motores del robot.

Esta prueba y la siguiente no fueron realizadas en la Facultad, por lo que no se disponía de un robot real para comprobar que se enviara la señal a los motores; sin embargo esta correcta comunicación entre los ordenadores, es el resultado que se buscaba obtener mediante estas pruebas de conexión. Al tenerse éxito utilizando una red local, lo siguiente entonces, fue comprobar si se podía lograr lo mismo a través de dos redes de Internet diferentes, prueba que se detalla a continuación.

5.2.3 Conexión entre ordenadores a través de una red externa

En esta segunda instancia se hizo uso de dos redes de Internet diferentes, una fue la misma LAN anterior (hogar del alumno) y la segunda, la red de Internet del hogar de un compañero del alumno, quien comprobó tener acceso a las configuraciones de su router, para habilitar los puertos a utilizar del modem en su correspondiente menú de *Port Forwarding*, tal como se describió en la Sección 3.6.4.1.

Uno de los portátiles (ordenador R) fue llevado al hogar del compañero, y el otro (ordenador B) se dejó en el hogar del alumno conectado a su Internet local, corriendo el programa iCC y el programa TeamViewer, mediante el cual se lo controlaría desde el ordenador R.

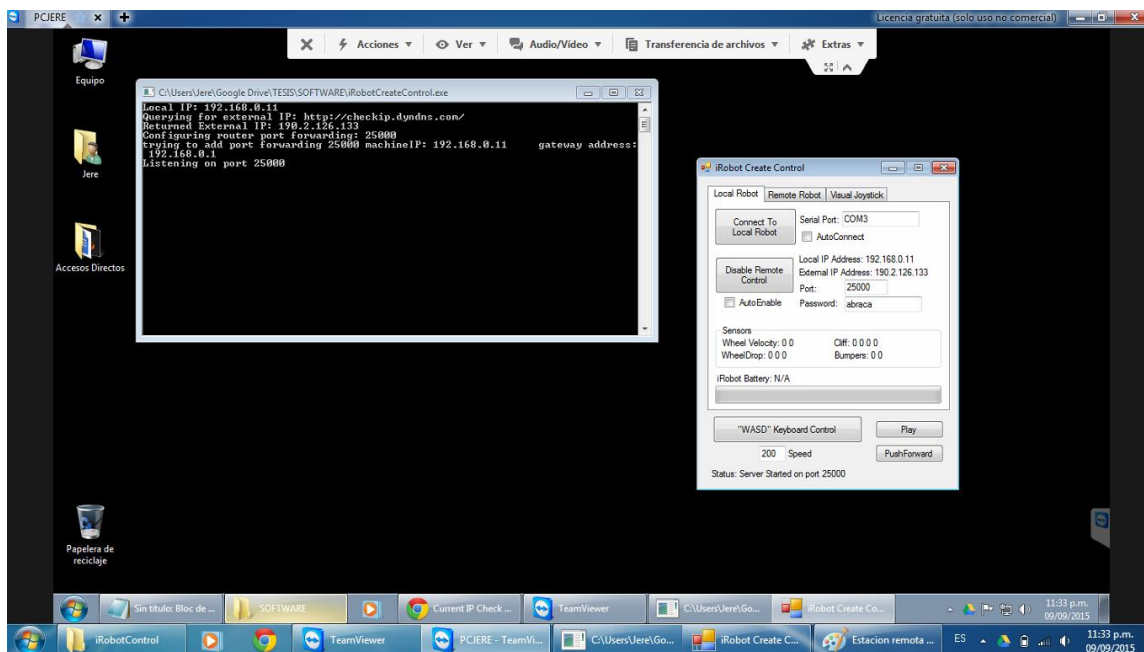


Ilustración 65 - Ordenador R activando *port forwarding* en el ordenador B a través de TeamViewer, para aceptar conexiones entrantes desde el Programa iCC.

La Ilustración 65 muestra una captura de pantalla del ordenador R, con el que se tomó control remoto del ordenador B usando TeamViewer, y se activó la función de *Port Forwarding* del programa iCC.

Exitosamente activada esta función, lo siguiente fue ingresar la dirección IP externa que el mismo iCC nos indica en la imagen (190.2.126.133), en el panel de conexión del ordenador R para así vincularse al ordenador B.

Mediante otra captura de pantalla, la Ilustración 66 muestra al ordenador R luego de haber ingresado los datos de conexión y presionado el botón “*Connect to Remote Robot*”. Esto produjo una petición que el ordenador B recibió, y ante esta entregó correctamente el control al ordenador R.

Con un recuadro rojo, se muestra nuevamente cómo el programa envía perfectamente las señales de movimiento al ordenador B, las cuales son recibidas también de su lado, como lo muestra la Ilustración 67.

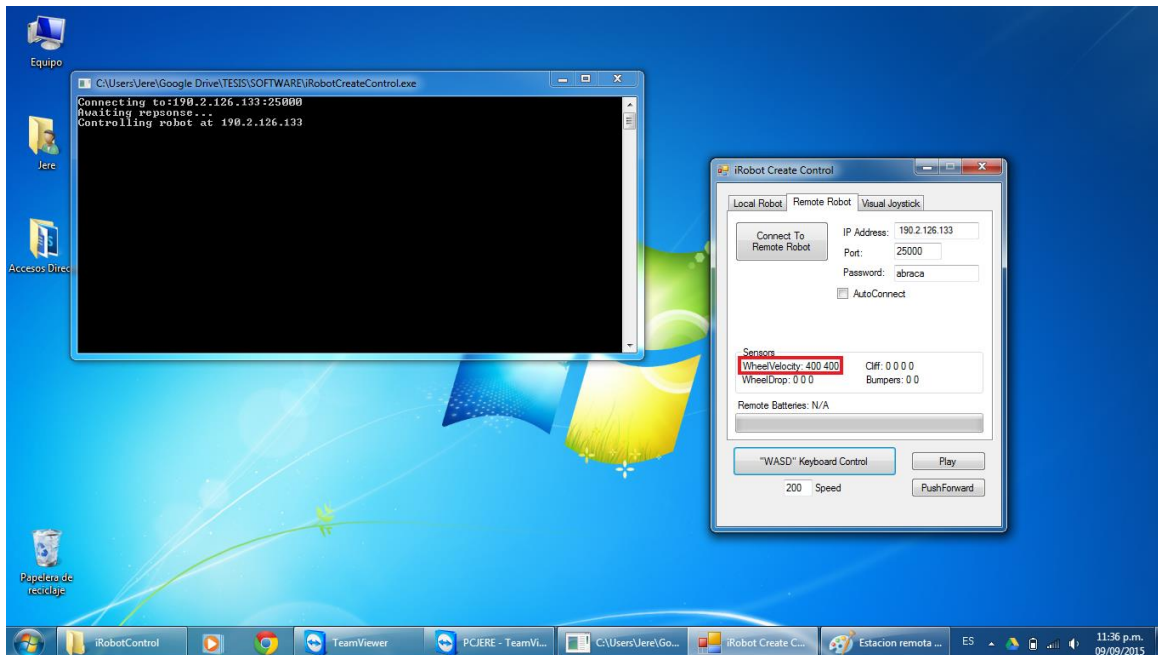


Ilustración 66 - Conexión establecida entre ordenadores R y B

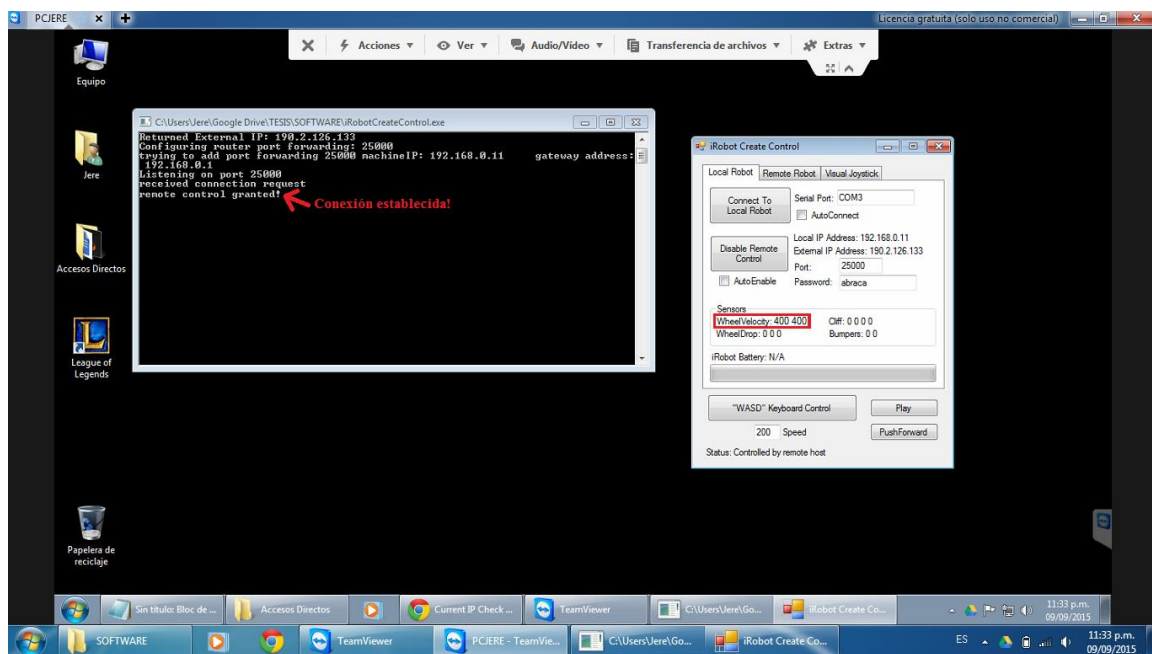


Ilustración 67 - Ordenador B siendo controlado.

De esta manera, se comprobó que el programa iCC podía utilizarse para vincular dos ordenadores remotamente ubicados uno del otro, y a través del Internet abierto, sin ningún tipo de impedimento, más que el de configurar correctamente los routers de las redes.

Este resultado fue muy satisfactorio y beneficioso para el avance de este trabajo, ya que permitió confirmar que la distancia entre los ordenadores B y R, no es un impedimento para la herramienta de telepresencia si se configuran correctamente todos los dispositivos.

La Ilustración 68 es una captura de pantalla que muestra la red de Internet externa que se utilizó al momento de realizar las pruebas que aquí se detallaron.

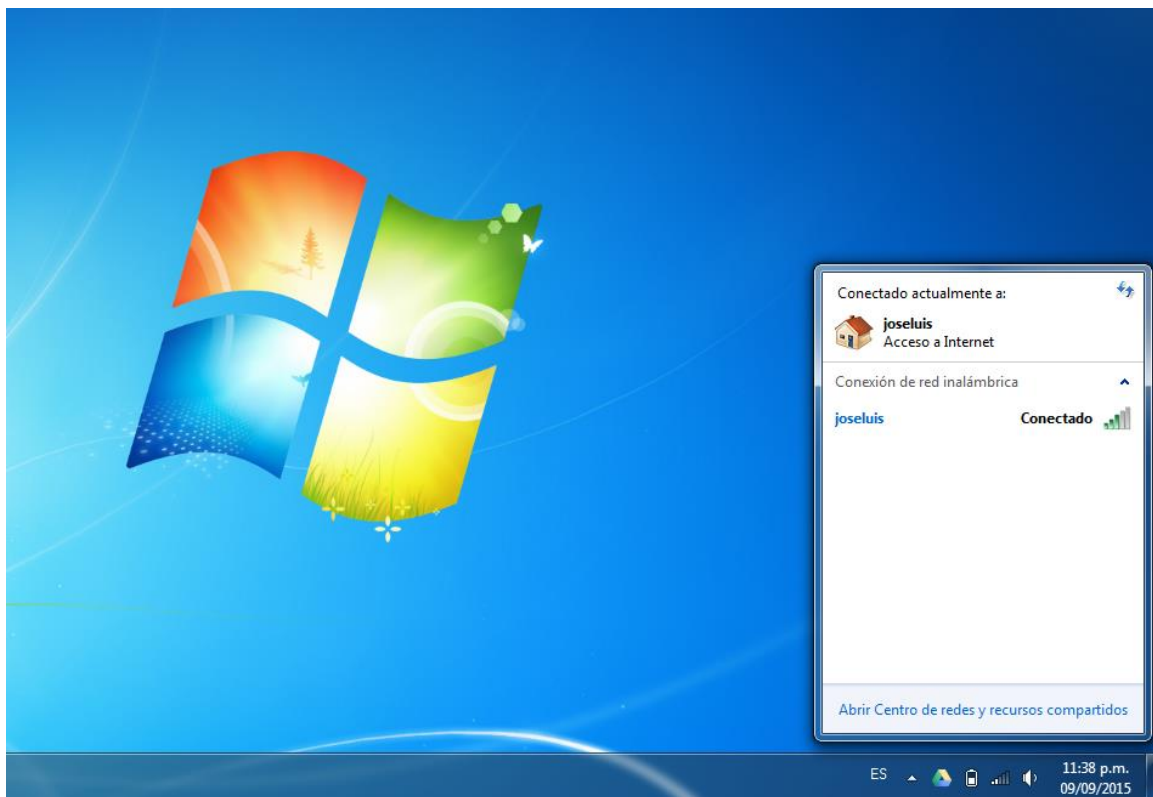


Ilustración 68 - Red de Internet utilizada para la prueba de conexión entre ordenadores.

Tanto en esta, como en la anterior prueba detallada, si bien no se contó con un robot real a la hora de realizarlas, esto no fue un inconveniente ya que hacia donde apuntaban las mismas, iba en otra dirección que la de testear los movimientos del robot. La naturaleza de estas pruebas consistió principalmente en la comprobación del funcionamiento de internet mientras estas conexiones se establecían y permanecían establecidas durante varios minutos.

De esta manera con los exitosos resultados obtenidos, se comprobó tanto la correcta comunicación entre los ordenadores (sea local o externamente), como también se comprobó la estabilidad de las conexiones establecidas, que además demostraron poder permanecer activas mientras fuese necesario.

Al obtenerse este esperado funcionamiento del programa iCC, el siguiente paso entonces consistió en agregar a esta combinación anterior un robot real, cuyo movimiento sea controlado desde el Ordenador R. A continuación, se describe la prueba que se llevó a cabo al respecto.

5.2.4 Navegación del robot desde Ordenador R

Para esta prueba, se coordinó un nuevo encuentro en el laboratorio de robótica donde se intentaría entonces, controlar el movimiento del robot desde un segundo ordenador (*Ord. R*).

Para esto se utilizaron dos computadores portátiles, y un router ajeno a la Facultad, el cual ya tenía habilitado el puerto necesario para la comunicación entre los ordenadores, de manera que mediante el mismo los ordenadores se vincularían de manera local, a través del Wi-Fi del router.

Esto debió realizarse ya que no es posible ingresar sin los debidos permisos a las configuraciones de los router de la Facultad, más específicamente los del Laboratorio de Robótica. Este impedimento hace que no se pueda habilitar el puerto a través del cual el *ordenador B* envía la petición de conexión del *ordenador R*; por esto, se utilizó un router ajeno a la Facultad.



Ilustración 69 - Router-Modem WiFi utilizado para estas pruebas.

Una vez encendidos los portátiles, conectados ambos vía WiFi a la red local que este router genera, y con el robot Create ya vinculado por *bluetooth* al ordenador B; se procedió así a establecer la conexión con el ordenador R, iniciando en ambos el programa iCC y realizando luego los mismos pasos que se describieron en la Sección 5.2.2.

Vinculados ya los ordenadores a través del programa iCC, llegó el momento de comprobar que el robot Create recibiera las señales de movimiento desde el ordenador R.

Se activó desde éste, la función de control de movimiento con el botón “*WASD Keyboard Control*”, y al presionar las teclas de movimiento, el robot Create comenzó a navegar correctamente por la habitación según las órdenes que se enviaban desde el teclado del ordenador R, confirmando exitosamente así la posibilidad de controlar su navegación desde un ordenador remoto.

De la misma manera que para la prueba descrita en la Sección 5.2.1, se eligió una habitación libre de obstáculos, para así brindar seguridad y amplia libertad de movimiento al robot. Durante más de 15 minutos se mantuvo la conexión entre los ordenadores y el robot, mientras se iban realizando distintos movimientos en el lugar, los cuales el Create ejecutaba a la perfección. El mismo respondió satisfactoriamente a los comandos de avance, retroceso y de vire en ambas direcciones; el recorrido que realizó fue principalmente lineal, se lo hacía avanzar y retroceder en línea recta, luego girando en una dirección u otra, para entonces volver al punto de retorno y repitiendo así varias veces esta secuencia.

En toda la duración de esta prueba las conexiones establecidas funcionaron de manera correcta, sin presentar inconvenientes ni cuando se ejecutaban comandos de movimiento del robot, ni cuando se dejaba al mismo reposando estático en un lugar de su recorrido.

Este resultado obtenido se consideró enormemente valioso en el transcurso de todo este trabajo y de las pruebas que se realizaron. Siendo esta la más relevante de todas, cuyo éxito clave desembocó en la decisión final del prototipo que este trabajo ofrece como solución propuesta.

Comprobando que las señales enviadas desde el ordenador R al B son ejecutadas correctamente por el robot, se concluyó entonces que el movimiento del mismo puede ser dirigido a través de Internet, desde un ordenador remotamente ubicado. A la par de esto, simplemente restaría iniciar una videollamada por medio de Skype entre ambos ordenadores, con lo cual se obtendría imagen y sonido del lugar donde el robot se encuentra, permitiendo además al usuario remotamente ubicado, ser visto y escuchado.

Al tratarse de un router ajeno a la Universidad no proveía del servicio de Internet a los ordenadores, por lo que en esta ocasión, no se pudo realizar una videollamada simultánea a la vinculación de los ordenadores a través del programa iCC. Sin embargo, para esto se coordinó una nueva visita al laboratorio, durante la cual se realizó la prueba que se describe a continuación.

5.2.5 Navegación del robot con videollamada en simultáneo

Habiendo confirmado la posibilidad de manipular el movimiento del robot desde un ordenador remoto como lo demostró la prueba anterior, solamente restaba entablar una videollamada entre los ordenadores, de manera que se obtuviese imagen y sonido del lugar donde el robot se ubica, así como del usuario de la herramienta.

Para esto se realizó una nueva prueba en el laboratorio de robótica, en la cual se utilizaron los siguientes elementos que se muestran en la Ilustración 70.



Ilustración 70 – Robot Create y ordenadores involucrados en la prueba.

En la misma podemos ver al robot Create y los dos ordenadores que se usaron para el experimento. El monitor de la derecha se usó en condición de *Ordenador B*, el cual se vinculó al robot por vía de *bluetooth*. Dado el carácter experimental de esta prueba, la cámara web que se ve sobre el robot también se vinculó a este ordenador, para que se pudiera ver hacia donde apunta el mismo cuando se lo mueve.

La computadora portátil a la izquierda fue entonces el *Ordenador R*. Como se aclaró en capítulos anteriores, la cámara web que ésta trae incorporada no funciona correctamente, es por esto que en el *ordenador B* no hay imagen definida. En esta ocasión no se encontraba disponible la webcam que se utilizará en reemplazo (ver Ilustración 55), sin embargo este factor no interfirió para realizar la prueba en cuestión.

Se procedió entonces a entablar las comunicaciones correspondientes entre los ordenadores; a continuación a través de capturas de pantallas de los ordenadores se mostrará paso a paso el resultado final de este experimento:

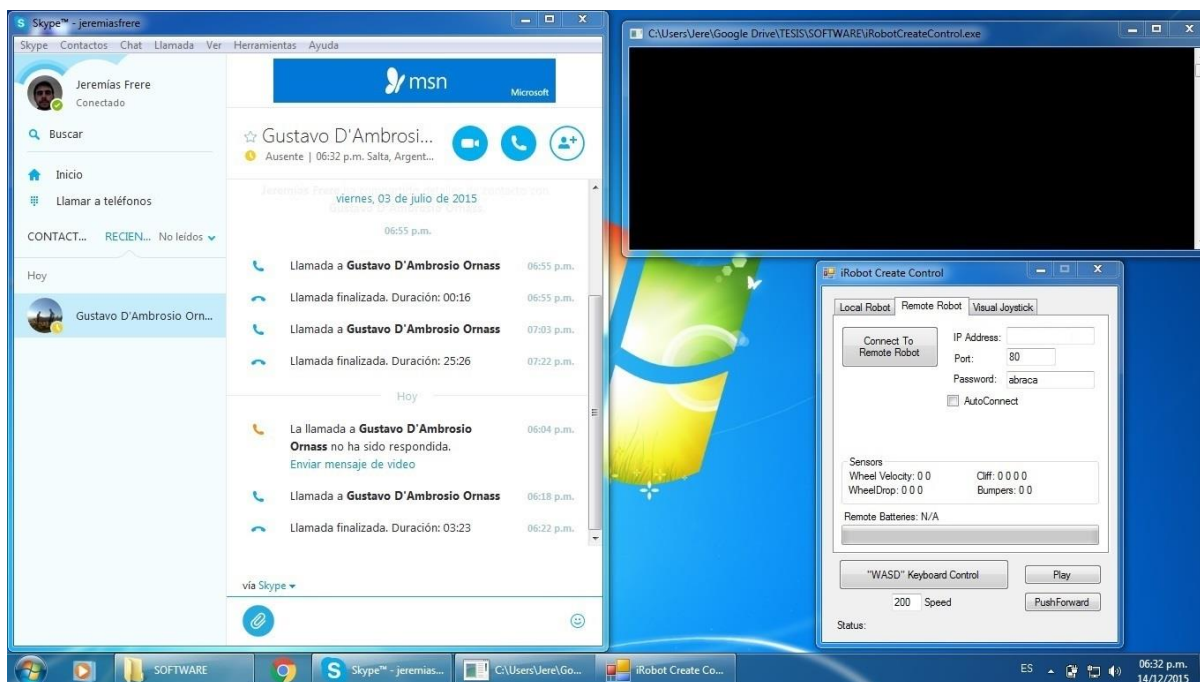


Ilustración 71 - Ordenador **R** previo a la conexión.

En la Ilustración 71 se muestra el *ordenador R*, en los momentos previos a la conexión, con los programas correspondientes ya iniciados, tanto Skype como el programa iCC.

En la Ilustración 72 por otro lado vemos una captura del *ordenador B*, también en los momentos antes de establecer las conexiones. Se iniciaron los programas iCC y Skype, y se activó la conexión con el robot por vía *bluetooth*, para lo cual se utilizó el software *BlueSoleil* disponible en el ordenador del laboratorio.

En la interfaz del mismo se puede ver que el puerto asignado a dicha vinculación fue el puerto **COM6**.

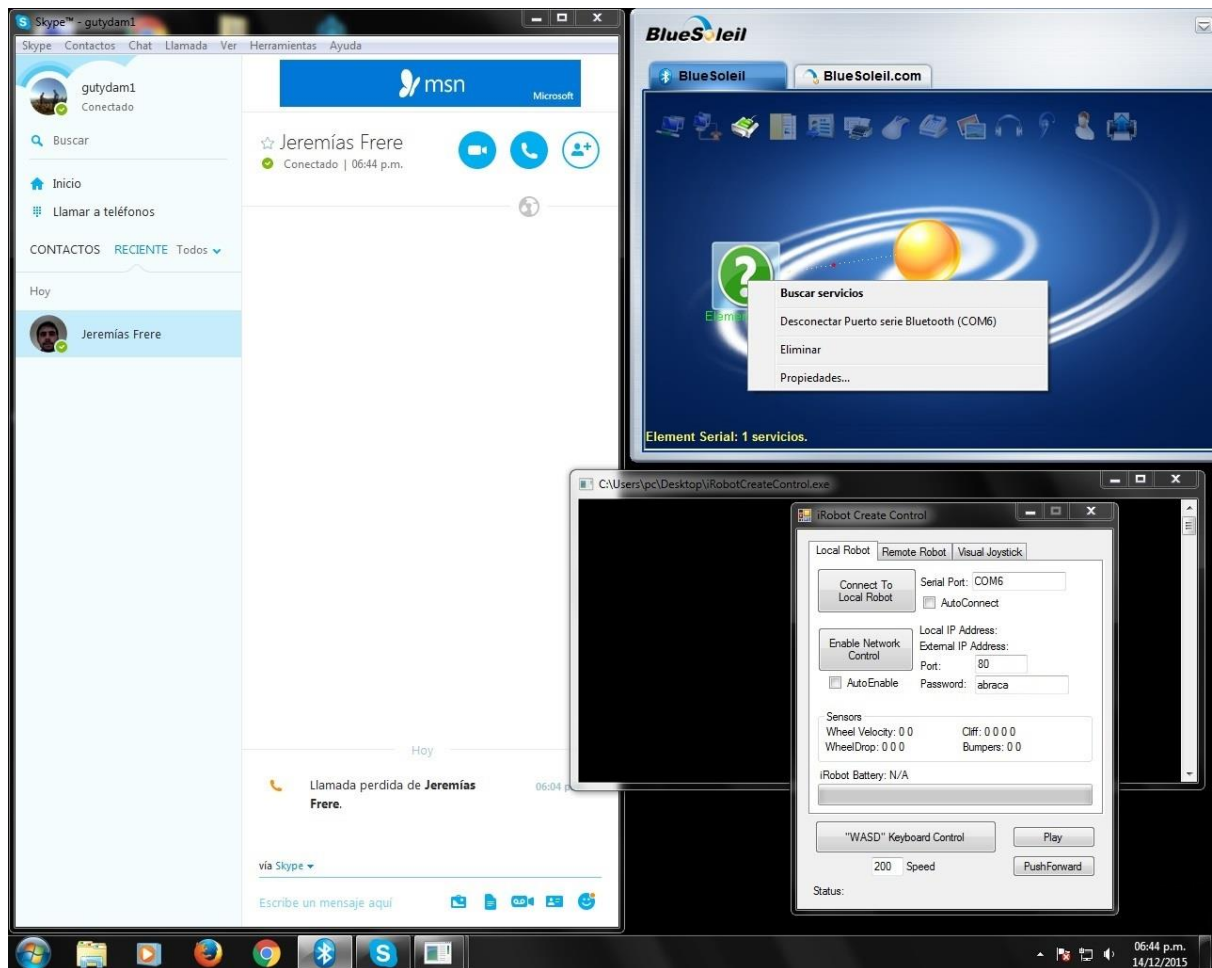


Ilustración 72 - Ordenador B previo a la conexión.

Teniendo ya preparado el escenario lo siguiente fue activar la función de *Port Forwarding* (PF) en el ordenador B; para que esto fuera posible dentro de la red de la Facultad, se utilizó un puerto diferente que suele no tener restricciones en ninguna red en general, el puerto número 80.

Cómo lo muestra la Ilustración 73, efectivamente dicho puerto no se encontraba restringido, por lo que dicha funcionalidad se activó de manera correcta en el *ordenador B*. En dicha imagen también vemos ya entablada la videollamada con el *ordenador R* a través de Skype.

Una vez activada correctamente la función de PF y la videollamada entre los ordenadores, lo siguiente fue enviar la petición de conexión desde el *ordenador R*, ingresando los datos correspondientes, que para esta prueba fueron:

- Dirección IP: 192.168.1.136
- Puerto: 80
- Contraseña: abraca

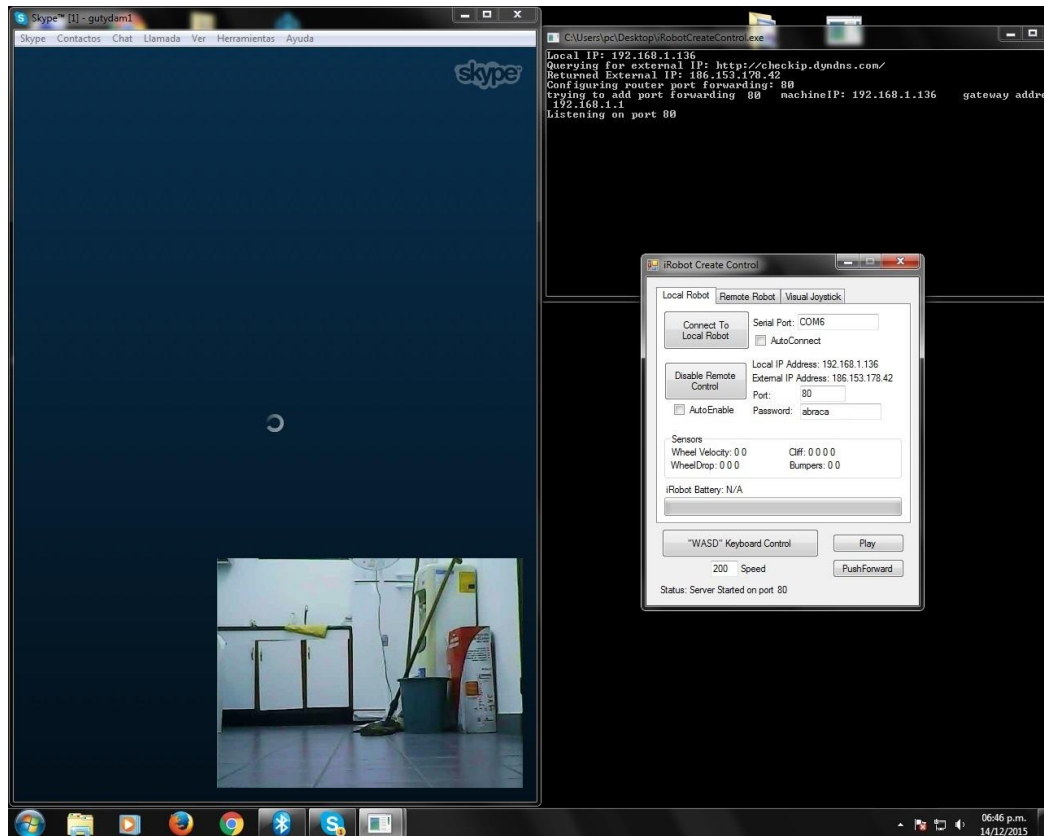


Ilustración 73 - Ordenador B esperando conexión entrante.

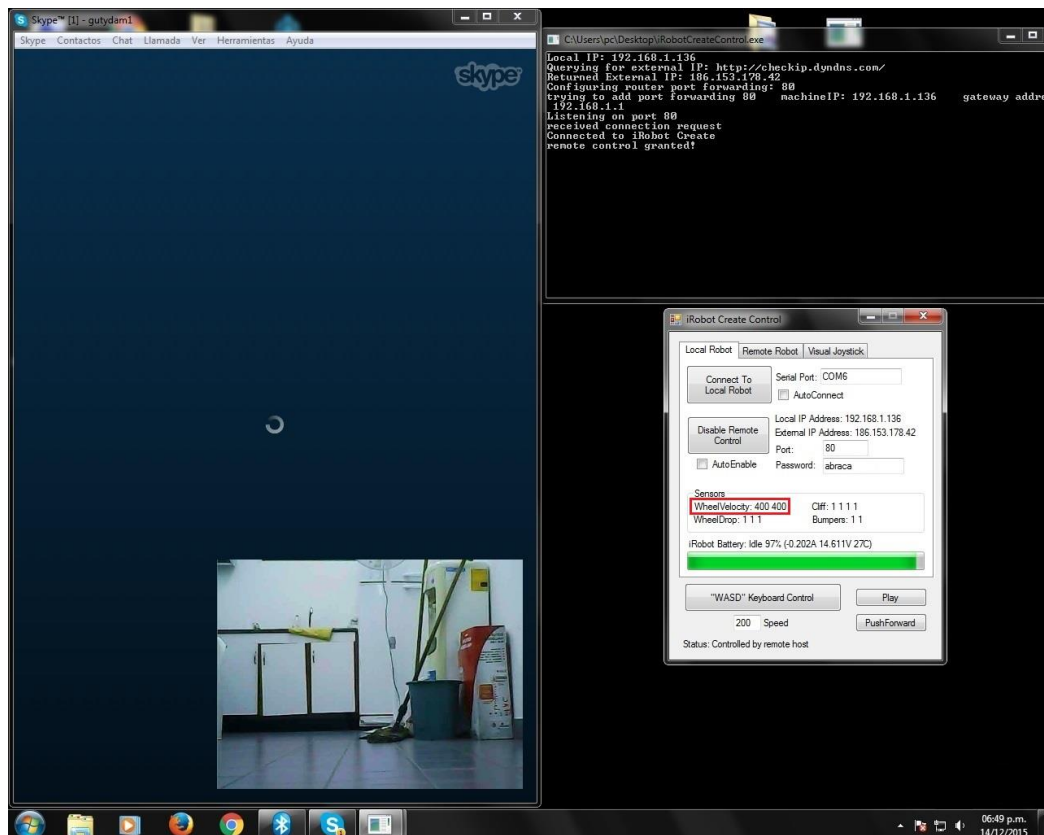
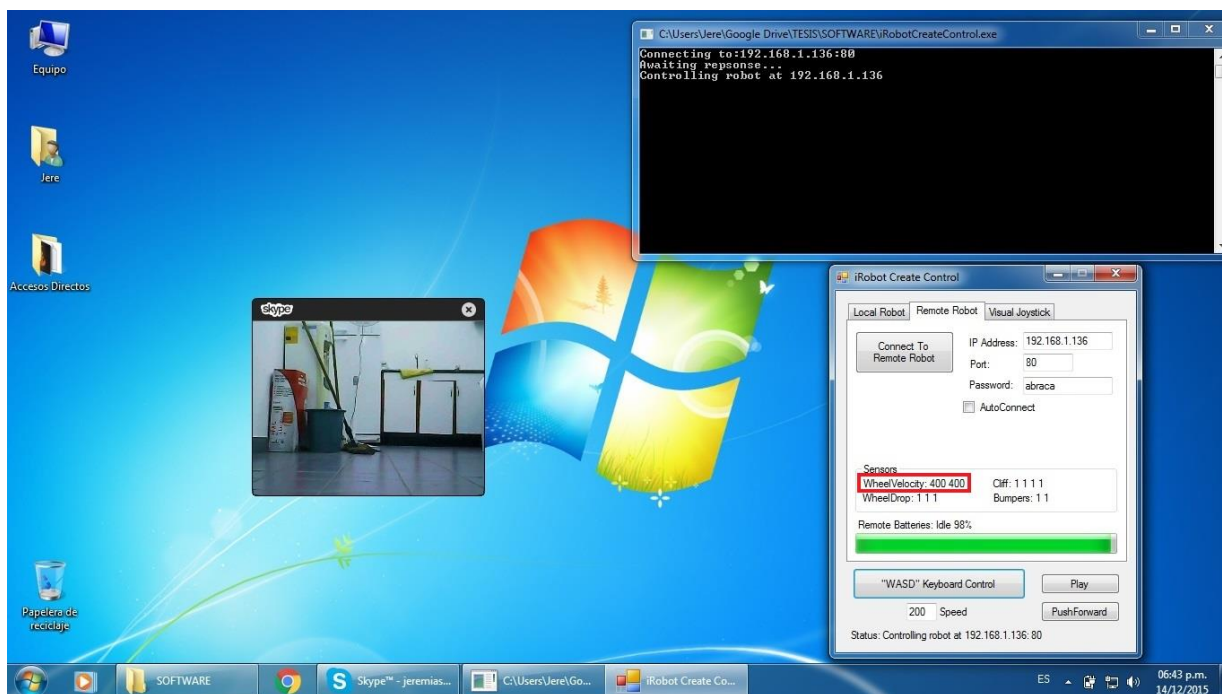


Ilustración 74 - Ordenador **B** siendo controlado

En la Ilustración 74 vemos cómo habiendo ingresado los datos de conexión, al enviar la petición desde el *ordenador R* la misma es recibida por el *ordenador B*, el cual entrega inmediatamente el control sobre el robot Create. Con un recuadro de color rojo, se señala que las órdenes de movimiento son recibidas correctamente, así como también vemos en dicha imagen, que el porcentaje de carga de la batería del robot se muestra en la interfaz con la barra de color verde.

Ilustración 75 - Ordenador **R** controlando robot Create

Por ultimo en la Ilustración 75, se muestra una captura del *ordenador R* donde se ve que el pedido de conexión fue recibido exitosamente por el *ordenador B*, el cual entrega el control sobre el robot permitiendo dirigir su movimiento, lo que se indica con un recuadro de color rojo, de la misma manera que en la Ilustración 74. Vemos también cómo el *ordenador R* recibe correctamente la información respectiva tanto al nivel de carga de la batería del robot, como la imagen de la dirección en la que está apuntando el mismo dentro de la habitación.

Esta última prueba también tuvo lugar en la Facultad, en el laboratorio de robótica. Dentro de ésta se despejó el espacio de obstáculos para que el robot pudiera moverse libremente: los comandos de avance y retroceso fueron recibidos correctamente, de la misma manera que los comandos de rotación, sin presentarse inconveniente alguno. La única diferencia entre los primeros y los segundos, fue la respuesta visual brindaba por la cámara colocada sobre el robot: al tratarse de una cámara web que transmitía vía WiFi, la velocidad de respuesta de la misma poseía un leve retraso, lo cual hizo que los comandos de rotación debieran ser ejecutados con más cuidado, ya que si utilizábamos el visor de la cámara para guiarnos, debía esperarse a que se refresque la imagen en cada giro, antes de realizar el siguiente movimiento.

De esta manera, este experimento final permitió comprobar que sí es posible mantener simultáneamente una videollamada por Skype, mientras se navega el Create dentro de la habitación donde se encuentra, desde un ordenador (R) diferente de aquél que se vincula por *bluetooth* (B) al robot.

Este resultado es lo que se esperaba obtener por medio de este último experimento, cuyo éxito desemboca satisfactoriamente en la obtención del prototipo esperado.

Llegado este punto, lo único que restaría para dar el toque final a esta solución de telepresencia, es encontrar la forma de que el *ordenador B* esté unido al robot, mediante algún tipo de soporte que lo mantenga sobre el mismo.

Es así, que a través de estas pruebas y testeos graduales de los aspectos de una herramienta de telepresencia, se llegó progresivamente a definir el prototipo que este trabajo tenía por objetivo encontrar, y proponer como solución final.

Al final de este trabajo, en la sección de anexos, se adjunta un instructivo de uso del programa *iRobot Create Control*, junto con el código fuente del mismo.

6. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

A partir de todo lo visto e indagado, uno de los pensamientos más nítidos que surge, es la enorme cantidad de posibilidades que hay de aplicar la robótica hoy en día, la cual ya no sólo se trata de robots que hacen determinadas acciones, sino que se trata también de cómo los mismos pueden ser fusionados con otras tecnologías emergentes de la actualidad, y así brindar nuevas opciones, soluciones, e ideas totalmente transformadoras en variadas áreas de la vida.

Es así como entre tantas de ellas, se trabajó aquí una en especial: **la telepresencia**; la cual ya no involucra simples conferencias entre personas a grandes distancias una de otra, sino que se trata de realmente lograr una sensación física de presencia, de la persona que está conectada desde un lugar remoto, aun así sea a través de métodos virtuales.

Este trabajo tuvo como objetivo no solamente mostrar una herramienta que se puede obtener aplicando ciertos conocimientos, y los recursos que la Universidad dispone; sino también abrir paso a considerar la telepresencia como algo que (entre video-llamadas de celulares, tabletas y computadores personales) ya es casi cotidiano en nuestro día a día, y así abrir camino a una nueva cartera de posibilidades en nuestras vidas.

La aplicación concreta de estas posibilidades que se abren en la actualidad, no dependerá únicamente de la tecnología que haya disponible a la mano, sino también de las nuevas necesidades que a cada momento surgen hoy en día para cada uno de nosotros; necesidades que llegado el caso, pueden impulsarnos aún más a la concretización de estas posibilidades, obteniendo así alternativas y soluciones innovadoras que faciliten el desempeño y desenvolvimiento de las distintas áreas de la vida; ya sea en ámbitos estudiantiles, profesionales o de negocios, como también los ámbitos sociales o espirituales que tanto varían de persona a persona, y que redefinen las metas y experiencias personales de cada uno de nosotros.

7. BIBLIOGRAFIA

- Ammyy, A. (2015). *Ammyy, Admin*. Recuperado el Agosto de 2015, de Ammyy, Admin:
http://www.ammyy.com/es/admin_screenshots.html
- Anybots. (2014). *Anybots Products*. Recuperado el Agosto de 2015, de <https://www.anybots.com/products/>
- AnyDesk, S. G. (2015). *AnyDesk*. Recuperado el Agosto de 2015, de AntDesk:
<http://anydesk.es/escritorio-remoto>
- Arthur, L. (2013). *Telepresencia Vs. Videoconferencia*. Recuperado el Julio de 2015, de eHow en Español:
http://www.ehowenespanol.com/telepresencia-versus-videoconferencia-info_81188/
- CDTUC. (2015). *Sala de videoconferencia CDTUC*. Recuperado el Septiembre de 2015, de
http://www.cdtuc.com/sala_conferencia.php
- Cisco. (2015). *Cisco*. Recuperado el Agosto de 2015, de Cisco:
<http://www.cisco.com/web/ES/telepresence/telepresence-basics.html>
- Columbia University. (2015). *Connection Guide for the iRobot Create*. Recuperado el Septiembre de 2015, de <http://www1.cs.columbia.edu/~allen/F15/GUIDE/iRobotCreateConnection%20Guide.pdf>
- Comercio, E. (Marzo de 2013). *Estados Unidos: niño asiste a clases a través de un robot*. Recuperado el Octubre de 2015, de
<http://elcomercio.pe/tecnologia/actualidad/estados-unidos-nino-asiste-clases-traves-robot-noticia-1547274>
- Comercio, E. (9 de Marzo de 2013). *Los robots de telepresencia dejaron la ficción y se transformaron en realidad*. Recuperado el Agosto de 2015, de El Comercio:
<http://elcomercio.pe/tecnologia/actualidad/fotos-robots-telepresencia-dejaron-ficcion-se-transformaron-realidad-noticia-1547620>
- Cotización Dolar. (2015). *Cotización-Dólar.com.ar © 2007 - 2015*. Recuperado el 30 de Noviembre de 2015, de <http://www.cotizacion-dolar.com.ar/>
- Element Direct. (2015). *BAM for the iRobot® Create™*. Recuperado el Septiembre de 2015, de
<http://www.elementdirect.com/files/10542B.pdf>
- Facebook. (2015). *Servicio de Ayuda*. Recuperado el Octubre de 2015, de
<https://www.facebook.com/help/439078162792430/>
- FreeDictionary. (2015). *The Free Dictionary*. Recuperado el Julio de 2015, de www.thefreedictionary.com:
<http://www.thefreedictionary.com/telerobotics>
- Galiana, I. C.-C. (2015). *Arquitectura de Control Bilateral para Manipulación Remota en Instalaciones de Fusión Nuclear*. Recuperado el Septiembre de 2015, de National Instruments:
<http://sine.ni.com/cs/app/doc/p/id/cs-14427>

- Gamero, C. R. (Septiembre de 2008). *Interfaces para Aplicaciones de Telerrobóticas y de Teleoperación*. Recuperado el Septiembre de 2015, de <https://cosmemarcano.files.wordpress.com/2008/10/iatt.pdf>
- Herring, S. C. (2013). "Telepresence Robots for Academics". "*Telepresence Robots for Academics*". Bloomington, Indiana, EE.UU.: Department of Information & Library Science, Indiana University.
- iRobot. (2015). "*How to Program Roomba - Introduction*". Recuperado el Octubre de 2015, de Robot App Store:
<http://www.robotappstore.com/Knowledge-Base/1-Introduction-to-Roomba-Programming/15.html>
- iRobot. (2015). *iRobot Create Open Interface Specification*. Recuperado el Octubre de 2015, de http://www.irobot.com/filelibrary/create/Create%20Open%20Interface_v2.pdf
- iRobot. (2015). *iRobot Create® 2 Programmable Robot*. Recuperado el Noviembre de 2015, de <http://www.irobot.com/About-iRobot/STEM/Create-2.aspx>
- iROBOT. (2015). *Manual de Usuario del iRobot® Create*. Recuperado el Septiembre de 2015, de http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf
- iRobot. (2015). *Product: Ava 500*. Recuperado el Agosto de 2015, de iRobot:
<http://www.irobot.com/For-Business/Ava-500.aspx>
- iRobot Company. (2015). *iRobot Create®*. Recuperado el Septiembre de 2015, de <http://store.irobot.com/irobot-create/product.jsp?productId=2663416>
- Isertec. (2013). *Isertec*. Recuperado el Agosto de 2015, de Isertec:
<http://www.isertec.com/videoconferencia.html>
- Kac. (1994). *Ekac*. Recuperado el 14 de Agosto de 2015, de Ekac:
<https://en.wikipedia.org/wiki/Telepresence>
- Kac. (1994). *KAC*. Recuperado el Julio de 2015, de sitio web de E. Kac: <http://www.ekac.org/>
- Kac. (1996). *Rara Avis*. Recuperado el Julio de 2015, de sitio web de Eduardo Kac:
<http://www.ekac.org/raraavis.html>
- Kainth, D. S. (2011). *How to setup Facebook video calling feature*. Recuperado el Octubre de 2015, de To The PC: <http://www.tothepc.com/archives/install-facebook-video-calling-setup/>
- Lee, J. C. (Febrero de 2011). *Low-Cost Video Chat Robot*. Recuperado el Septiembre de 2015, de Procrastineering: <http://procrastineering.blogspot.com.ar/2011/02/low-cost-video-chat-robot.html>
- Lehrbaum, R. (1 de Noviembre de 2013). *Information Week*. Recuperado el 5 de 8 de 2014, de Information Week:
http://www.informationweek.com/applications/attack-of-the-telepresence-robots!/d/d-id/1108137?page_number=2

- MantaroBot. (2015). *Product: TeleMe*. Recuperado el Agosto de 2015, de Mantarobot TelePresence Robots: <http://www.mantarobot.com/products/teleme/index.htm>
- Microsoft. (2015). *Microsoft Downlad Center*. Recuperado el Agosto de 2015, de Microsoft Downlad Center: <https://www.microsoft.com/en-us/download/details.aspx?id=29081>
- Minsky, M. (Junio de 1980). *Telepresence*. Recuperado el Julio de 2015, de MIT Media Lab: <http://web.media.mit.edu/~minsky/papers/Telepresence.html>
- Monreal, M. L. (2013). *Telerrobótica en el Área de la Medicina*. Recuperado el Septiembre de 2015, de Prezi: <https://prezi.com/cp4fg92iorvp/telerrobotica-en-el-area-de-la-medicina/>
- Murphy, R. R. (2000). *Introduction to IA Robotics*. Cambridge, Massachusetts: The MIT Press.
- Nathan, M. (Febrero de 2011). *Low-Cost Video Chat Robot*. Recuperado el Septiembre de 2015, de HACKDAY: <http://hackaday.com/2011/02/09/low-cost-video-chat-robot/>
- Neetguias. (Abril de 2013). *Comenzar a utilizar Skype*. Recuperado el Septiembre de 2015, de [www.neetguias.org](http://www.neetguias.org/comenzar-a-utilizar-skype/): <http://www.neetguias.org/comenzar-a-utilizar-skype/>
- Ortiz, J., & Orellana, M. (11 de Junio de 2013). *Cisco.com*. Recuperado el 24 de Julio de 2015, de Cisco.com: <http://www.cisco.com/web/ES/about/press/2013/2013-06-11-irobot-colaboracion-telepresencia-d-e-cisco.html>
- RoboSavvy. (2015). *Double Telepresence Robot*. Recuperado el Octubre de 2015, de <http://robosavvy.com/store/double-robotics-double-telepresence-robot.html>
- RooWifi. (2013). *RooWifi: Wi-Fi Remote for Roomba*. Recuperado el Octubre de 2015, de <http://www.roowifi.com/>
- RooWifi. (2013). *Características Técnicas de Roomba Wi-Fi Remote*. Recuperado el Octubre de 2015, de <http://es.roowifi.com/caracteristicas-tecnicas/>
- RooWiFi. (2013). *Pictures of RooWifi*. Recuperado el Octubre de 2015, de <http://www.roowifi.com/pictures-roomba-wifi-remote/roowifi-1/>
- ROS. (2015). *ros.org*. Recuperado el Agosto de 2015, de ros.org: <http://www.ros.org/about-ros/>
- Rubira, F. (15 de Julio de 2013). *¿Qué es Google Hangouts y para qué sirve?* Recuperado el Octubre de 2015, de El Confidencial Digital: http://www.elconfidencialdigital.com/opinion/tribuna_libre/Google-Hangouts-sirve_0_2089591050.html
- Skype. (2015). *¿Qué es Skype?* Recuperado el Septiembre de 2015, de support.skype.com: <https://support.skype.com/es/faq/FA6/que-es-skype>
- Strategic Content. (Junio de 2014). *How to Setup a Google+ Hangout Video Call*. Recuperado el Octubre de 2015, de <http://strategiccontent.co/setup-hangout-video-call/>

Suitable Technologies. (2015). *Product: BeamPro*. Recuperado el Agosto de 2015, de suitabletech:
<https://d1qfx8fl7tbbtn.cloudfront.net/static/site/pdf/BeamProductBrochure.pdf>

TeamViewer. (2015). *TeamViewer*. Recuperado el Agosto de 2015, de TeamViewer:
<https://www.teamviewer.com/es/>

TelepresenceRobots. (2015). *Comparison Telepresence Robots*. Recuperado el Septiembre de 2015, de
<http://telepresencerobots.com/comparison>

Trends, T. (2010). *¿Qué es la telepresencia?* Recuperado el Agosto de 2015, de Salas de Telepresencia:
<http://www.salasdetelepresencia.com/que-es-la-telepresencia.html>

VGo, C. (2015). *How Does VGo Work?* Recuperado el Agosto de 2015, de VgoCom:
<http://www.vgocom.com/how-does-vgo-work>

Vgo, C. (2015). *VGo Overview*. Recuperado el Agosto de 2015, de VgoCom.com:
<http://www.vgocom.com/solutions>

Vgo, C. (2015). *What is VGo?* Recuperado el Agosto de 2015, de VgoCom:
<http://www.vgocom.com/what-vgo>

Vic7767. (2015). *Hacking Roomba Hardware/Software*. Recuperado el Noviembre de 2015, de
<http://vic7767.com/hacking-roomba-create-hardware-software/>

VITECH. (2015). *Renta de Salas de Videoconferencia*. Recuperado el Septiembre de 2015, de
<http://www.vitech.com.mx/index.php/servicios/renta-de-salas-de-videoconferencia.html>

Wikipedia. (2011). *Telepresence*. Recuperado el 5 de Agosto de 2015, de Wikipedia:
<https://en.wikipedia.org/wiki/Telepresence>

Wikipedia. (2015). *Hangouts*. Recuperado el Octubre de 2015, de <https://es.wikipedia.org/wiki/Hangouts>

Wikipedia. (2015). *iRobot Create*. Recuperado el Septiembre de 2015, de
https://en.wikipedia.org/wiki/IRobot_Create

Wikipedia. (2015). *Port Forwarding*. Recuperado el Septiembre de 2015, de
https://en.wikipedia.org/wiki/Port_forwarding

Wikipedia. (2015). *Telerrobótica*. Recuperado el Septiembre de 2015, de
<https://es.wikipedia.org/wiki/Telerrob%C3%B3tica>

Wikipedia. (2015). *Telerrobótica*. Recuperado el Septiembre de 2015, de Wikipedia:
<https://es.wikipedia.org/wiki/Telerrob%C3%B3tica>

Wikipedia. (2015). *Universal Plug and Play*. Recuperado el Septiembre de 2015, de
https://es.wikipedia.org/wiki/Universal_Plug_and_Play

ANEXOS

Anexo 1. Código del software “iRobot Create Control”

Anexo 1.1. Archivos del formulario

Anexo 1.1.1. Diseño del formulario

Código del archivo de diseño, de la interfaz del programa iCC. (Form1.Designer.cs)

```
namespace iRobotCreateControl
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
        otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.lblBumpers = new System.Windows.Forms.Label();
            this.lblWheelDrop = new System.Windows.Forms.Label();
            this.lblCliff = new System.Windows.Forms.Label();
            this.lblWheelVelocity = new System.Windows.Forms.Label();
            this.btnCoverAndDock = new System.Windows.Forms.Button();
            this.btnControl = new System.Windows.Forms.Button();
            this.lblBattery = new System.Windows.Forms.Label();
            this.tbCOMPort = new System.Windows.Forms.TextBox();
            this.tbRemoteIP = new System.Windows.Forms.TextBox();
            this.btnConnectIP = new System.Windows.Forms.Button();
            this.lblStatus = new System.Windows.Forms.Label();
            this.lblLocalIP = new System.Windows.Forms.Label();
        }
    }
}
```

```

this.lblExternalIP = new System.Windows.Forms.Label();
this.cbAutoConnectCOM = new System.Windows.Forms.CheckBox();
this.cbAutoConnectNetwork = new System.Windows.Forms.CheckBox();
this.pbBattery = new System.Windows.Forms.ProgressBar();
this.btnStartServer = new System.Windows.Forms.Button();
this.cbAutoStartServer = new System.Windows.Forms.CheckBox();
this.btnKeyboardControl = new System.Windows.Forms.Button();
this.tbKeyboardSpeed = new System.Windows.Forms.TextBox();
this.label11 = new System.Windows.Forms.Label();
this.btnStop = new System.Windows.Forms.Button();
this.label12 = new System.Windows.Forms.Label();
this.label13 = new System.Windows.Forms.Label();
this.tabControl = new System.Windows.Forms.TabControl();
this.tabPage1 = new System.Windows.Forms.TabPage();
this.tbServerPort = new System.Windows.Forms.TextBox();
this.label16 = new System.Windows.Forms.Label();
this.groupBox1 = new System.Windows.Forms.GroupBox();
this.label14 = new System.Windows.Forms.Label();
this.tbServerPassword = new System.Windows.Forms.TextBox();
this.tabPage2 = new System.Windows.Forms.TabPage();
this.groupBox2 = new System.Windows.Forms.GroupBox();
this.lblWheelVelocityR = new System.Windows.Forms.Label();
this.lblBumpersR = new System.Windows.Forms.Label();
this.lblWheelDropR = new System.Windows.Forms.Label();
this.lblCliffR = new System.Windows.Forms.Label();
this.lblBatteryR = new System.Windows.Forms.Label();
this.pbBatteryR = new System.Windows.Forms.ProgressBar();
this.tbRemotePort = new System.Windows.Forms.TextBox();
this.label17 = new System.Windows.Forms.Label();
this.label15 = new System.Windows.Forms.Label();
this.tbRemotePassword = new System.Windows.Forms.TextBox();
this.tabJoystick = new System.Windows.Forms.TabPage();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
this.tabControl.SuspendLayout();
this.tabPage1.SuspendLayout();
this.groupBox1.SuspendLayout();
this.tabPage2.SuspendLayout();
this.groupBox2.SuspendLayout();
this.tabJoystick.SuspendLayout();
this.SuspendLayout();
//
// pictureBox1
//
this.pictureBox1.Location = new System.Drawing.Point(28, 9);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(250, 250);
this.pictureBox1.TabIndex = 3;
this.pictureBox1.TabStop = false;
this.pictureBox1.Click += new System.EventHandler(this.pictureBox1_Click);
this.pictureBox1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseDown);
this.pictureBox1.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseMove);
this.pictureBox1.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseUp);
//
// lblBumpers

```

```

//
this.lblBumpers.AutoSize = true;
this.lblBumpers.Location = new System.Drawing.Point(151, 32);
this.lblBumpers.Name = "lblBumpers";
this.lblBumpers.Size = new System.Drawing.Size(69, 13);
this.lblBumpers.TabIndex = 5;
this.lblBumpers.Text = "Bumpers: 0 0";
//
// lblWheelDrop
//
this.lblWheelDrop.AutoSize = true;
this.lblWheelDrop.Location = new System.Drawing.Point(5, 32);
this.lblWheelDrop.Name = "lblWheelDrop";
this.lblWheelDrop.Size = new System.Drawing.Size(91, 13);
this.lblWheelDrop.TabIndex = 6;
this.lblWheelDrop.Text = "WheelDrop: 0 0 0";
//
// lblCliff
//
this.lblCliff.AutoSize = true;
this.lblCliff.Location = new System.Drawing.Point(151, 16);
this.lblCliff.Name = "lblCliff";
this.lblCliff.Size = new System.Drawing.Size(63, 13);
this.lblCliff.TabIndex = 7;
this.lblCliff.Text = "Cliff: 0 0 0 0";
//
// lblWheelVelocity
//
this.lblWheelVelocity.AutoSize = true;
this.lblWheelVelocity.Location = new System.Drawing.Point(6, 16);
this.lblWheelVelocity.Name = "lblWheelVelocity";
this.lblWheelVelocity.Size = new System.Drawing.Size(99, 13);
this.lblWheelVelocity.TabIndex = 8;
this.lblWheelVelocity.Text = "Wheel Velocity: 0 0";
//
// btnCoverAndDock
//
this.btnCoverAndDock.Location = new System.Drawing.Point(229, 315);
this.btnCoverAndDock.Name = "btnCoverAndDock";
this.btnCoverAndDock.Size = new System.Drawing.Size(88, 23);
this.btnCoverAndDock.TabIndex = 9;
this.btnCoverAndDock.Text = "Play";
this.btnCoverAndDock.UseVisualStyleBackColor = true;
this.btnCoverAndDock.Click += new
System.EventHandler(this.btnCoverAndDock_Click);
//
// btnControl
//
this.btnControl.Location = new System.Drawing.Point(5, 7);
this.btnControl.Name = "btnControl";
this.btnControl.Size = new System.Drawing.Size(106, 48);
this.btnControl.TabIndex = 10;
this.btnControl.Text = "    Connect To    Local Robot";
this.btnControl.UseVisualStyleBackColor = true;
this.btnControl.Click += new System.EventHandler(this.btnControl_Click);
//
// lblBattery
//

```

```

        this.lblBattery.AutoSize = true;
        this.lblBattery.Location = new System.Drawing.Point(6, 220);
        this.lblBattery.Name = "lblBattery";
        this.lblBattery.Size = new System.Drawing.Size(100, 13);
        this.lblBattery.TabIndex = 11;
        this.lblBattery.Text = "iRobot Battery: N/A";
        //
        // tbCOMPort
        //
        this.tbCOMPort.Location = new System.Drawing.Point(170, 11);
        this.tbCOMPort.Name = "tbCOMPort";
        this.tbCOMPort.Size = new System.Drawing.Size(100, 20);
        this.tbCOMPort.TabIndex = 12;
        this.tbCOMPort.Text = "COM3";
        //
        // tbRemoteIP
        //
        this.tbRemoteIP.Location = new System.Drawing.Point(193, 12);
        this.tbRemoteIP.Name = "tbRemoteIP";
        this.tbRemoteIP.Size = new System.Drawing.Size(100, 20);
        this.tbRemoteIP.TabIndex = 13;
        this.tbRemoteIP.Text = "127.0.0.1";
        this.tbRemoteIP.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.tbRemoteIP_KeyPress);
        //
        // btnConnectIP
        //
        this.btnConnectIP.Location = new System.Drawing.Point(6, 11);
        this.btnConnectIP.Name = "btnConnectIP";
        this.btnConnectIP.Size = new System.Drawing.Size(110, 47);
        this.btnConnectIP.TabIndex = 14;
        this.btnConnectIP.Text = "Connect To Remote Robot";
        this.btnConnectIP.UseVisualStyleBackColor = true;
        this.btnConnectIP.Click += new
System.EventHandler(this.btnConnectIP_Click);
        //
        // lblStatus
        //
        this.lblStatus.AutoSize = true;
        this.lblStatus.Location = new System.Drawing.Point(14, 377);
        this.lblStatus.Name = "lblStatus";
        this.lblStatus.Size = new System.Drawing.Size(43, 13);
        this.lblStatus.TabIndex = 15;
        this.lblStatus.Text = "Status: ";
        //
        // lblLocalIP
        //
        this.lblLocalIP.AutoSize = true;
        this.lblLocalIP.Location = new System.Drawing.Point(111, 66);
        this.lblLocalIP.Name = "lblLocalIP";
        this.lblLocalIP.Size = new System.Drawing.Size(93, 13);
        this.lblLocalIP.TabIndex = 16;
        this.lblLocalIP.Text = "Local IP Address: ";
        //
        // lblExternalIP
        //
        this.lblExternalIP.AutoSize = true;
        this.lblExternalIP.Location = new System.Drawing.Point(111, 83);

```

```

this.lblExternalIP.Name = "lblExternalIP";
this.lblExternalIP.Size = new System.Drawing.Size(105, 13);
this.lblExternalIP.TabIndex = 17;
this.lblExternalIP.Text = "External IP Address: ";
//
// cbAutoConnectCOM
//
this.cbAutoConnectCOM.AutoSize = true;
this.cbAutoConnectCOM.Location = new System.Drawing.Point(118, 36);
this.cbAutoConnectCOM.Name = "cbAutoConnectCOM";
this.cbAutoConnectCOM.Size = new System.Drawing.Size(88, 17);
this.cbAutoConnectCOM.TabIndex = 18;
this.cbAutoConnectCOM.Text = "AutoConnect";
this.cbAutoConnectCOM.UseVisualStyleBackColor = true;
this.cbAutoConnectCOM.CheckedChanged += new
System.EventHandler(this.cbAutoConnect_CheckedChanged);
//
// cbAutoConnectNetwork
//
this.cbAutoConnectNetwork.AutoSize = true;
this.cbAutoConnectNetwork.Location = new System.Drawing.Point(126, 88);
this.cbAutoConnectNetwork.Name = "cbAutoConnectNetwork";
this.cbAutoConnectNetwork.Size = new System.Drawing.Size(88, 17);
this.cbAutoConnectNetwork.TabIndex = 19;
this.cbAutoConnectNetwork.Text = "AutoConnect";
this.cbAutoConnectNetwork.UseVisualStyleBackColor = true;
//
// pbBattery
//
this.pbBattery.Location = new System.Drawing.Point(6, 236);
this.pbBattery.Name = "pbBattery";
this.pbBattery.Size = new System.Drawing.Size(292, 23);
this.pbBattery.TabIndex = 20;
//
// btnStartServer
//
this.btnStartServer.Location = new System.Drawing.Point(6, 70);
this.btnStartServer.Name = "btnStartServer";
this.btnStartServer.Size = new System.Drawing.Size(102, 47);
this.btnStartServer.TabIndex = 21;
this.btnStartServer.Text = "Enable Network Control";
this.btnStartServer.UseVisualStyleBackColor = true;
this.btnStartServer.Click += new
System.EventHandler(this.btnStartServer_Click);
//
// cbAutoStartServer
//
this.cbAutoStartServer.AutoSize = true;
this.cbAutoStartServer.Location = new System.Drawing.Point(18, 121);
this.cbAutoStartServer.Name = "cbAutoStartServer";
this.cbAutoStartServer.Size = new System.Drawing.Size(81, 17);
this.cbAutoStartServer.TabIndex = 22;
this.cbAutoStartServer.Text = "AutoEnable";
this.cbAutoStartServer.UseVisualStyleBackColor = true;
//
// btnKeyboardControl
//
this.btnKeyboardControl.Location = new System.Drawing.Point(19, 309);

```

```

        this.btnKeyboardControl.Name = "btnKeyboardControl";
        this.btnKeyboardControl.Size = new System.Drawing.Size(193, 35);
        this.btnKeyboardControl.TabIndex = 23;
        this.btnKeyboardControl.Text = "\\\"WASD\\\" Keyboard Control";
        this.btnKeyboardControl.UseVisualStyleBackColor = true;
        this.btnKeyboardControl.Click += new
System.EventHandler(this.btnKeyboardControl_Click);
        this.btnKeyboardControl.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.button1_KeyDown);
        this.btnKeyboardControl.KeyUp += new
System.Windows.Forms.KeyEventHandler(this.button1_KeyUp);
        //
        // tbKeyBoardSpeed
        //
        this.tbKeyBoardSpeed.Location = new System.Drawing.Point(79, 348);
        this.tbKeyBoardSpeed.Name = "tbKeyBoardSpeed";
        this.tbKeyBoardSpeed.Size = new System.Drawing.Size(34, 20);
        this.tbKeyBoardSpeed.TabIndex = 24;
        this.tbKeyBoardSpeed.Text = "200";
        this.tbKeyBoardSpeed.TextChanged += new
System.EventHandler(this.tbKeyBoardSpeed_TextChanged);
        this.tbKeyBoardSpeed.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.tbKeyBoardSpeed_KeyPress);
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(113, 351);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(38, 13);
        this.label1.TabIndex = 25;
        this.label1.Text = "Speed";
        //
        // btnStop
        //
        this.btnStop.Location = new System.Drawing.Point(229, 346);
        this.btnStop.Name = "btnStop";
        this.btnStop.Size = new System.Drawing.Size(88, 23);
        this.btnStop.TabIndex = 26;
        this.btnStop.Text = "PushForward";
        this.btnStop.UseVisualStyleBackColor = true;
        this.btnStop.Click += new System.EventHandler(this.btnStop_Click);
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Location = new System.Drawing.Point(112, 15);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(58, 13);
        this.label2.TabIndex = 27;
        this.label2.Text = "Serial Port:";
        //
        // label3
        //
        this.label3.AutoSize = true;
        this.label3.Location = new System.Drawing.Point(130, 17);
        this.label3.Name = "label3";
        this.label3.Size = new System.Drawing.Size(61, 13);

```

```

this.label3.TabIndex = 28;
this.label3.Text = "IP Address:";
//
// tabControl
//
this.tabControl.Controls.Add(this.tabPage1);
this.tabControl.Controls.Add(this.tabPage2);
this.tabControl.Controls.Add(this.tabJoystick);
this.tabControl.Location = new System.Drawing.Point(12, 12);
this.tabControl.Name = "tabControl";
this.tabControl.SelectedIndex = 0;
this.tabControl.Size = new System.Drawing.Size(315, 291);
this.tabControl.TabIndex = 29;
//
// tabPage1
//
this.tabPage1.Controls.Add(this.tbServerPort);
this.tabPage1.Controls.Add(this.label6);
this.tabPage1.Controls.Add(this.groupBox1);
this.tabPage1.Controls.Add(this.label4);
this.tabPage1.Controls.Add(this.tbServerPassword);
this.tabPage1.Controls.Add(this.btnControl);
this.tabPage1.Controls.Add(this.tbCOMPort);
this.tabPage1.Controls.Add(this.label2);
this.tabPage1.Controls.Add(this.lblLocalIP);
this.tabPage1.Controls.Add(this.pbBattery);
this.tabPage1.Controls.Add(this.lblExternalIP);
this.tabPage1.Controls.Add(this.lblBattery);
this.tabPage1.Controls.Add(this.cbAutoConnectCOM);
this.tabPage1.Controls.Add(this.btnStartServer);
this.tabPage1.Controls.Add(this.cbAutoStartServer);
this.tabPage1.Location = new System.Drawing.Point(4, 22);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(307, 265);
this.tabPage1.TabIndex = 0;
this.tabPage1.Text = "Local Robot";
this.tabPage1.UseVisualStyleBackColor = true;
//
// tbServerPort
//
this.tbServerPort.Location = new System.Drawing.Point(173, 99);
this.tbServerPort.Name = "tbServerPort";
this.tbServerPort.Size = new System.Drawing.Size(56, 20);
this.tbServerPort.TabIndex = 32;
this.tbServerPort.Text = "25000";
this.tbServerPort.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.tbServerPort_KeyPress);
//
// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(112, 104);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(29, 13);
this.label6.TabIndex = 31;
this.label6.Text = "Port:";
//

```

```

// groupBox1
//
this.groupBox1.Controls.Add(this.lblWheelVelocity);
this.groupBox1.Controls.Add(this.lblBumpers);
this.groupBox1.Controls.Add(this.lblWheelDrop);
this.groupBox1.Controls.Add(this.lblCliff);
this.groupBox1.Location = new System.Drawing.Point(6, 158);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(292, 56);
this.groupBox1.TabIndex = 30;
this.groupBox1.TabStop = false;
this.groupBox1.Text = "Sensors";
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(112, 123);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(56, 13);
this.label4.TabIndex = 29;
this.label4.Text = "Password:";
//
// tbServerPassword
//
this.tbServerPassword.Location = new System.Drawing.Point(173, 121);
this.tbServerPassword.Name = "tbServerPassword";
this.tbServerPassword.Size = new System.Drawing.Size(100, 20);
this.tbServerPassword.TabIndex = 28;
this.tbServerPassword.Text = "abracadabra";
this.tbServerPassword.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.tbServerPassword_KeyPress);
//
// tabPage2
//
this.tabPage2.Controls.Add(this.groupBox2);
this.tabPage2.Controls.Add(this.lblBatteryR);
this.tabPage2.Controls.Add(this.pbBatteryR);
this.tabPage2.Controls.Add(this.tbRemotePort);
this.tabPage2.Controls.Add(this.label7);
this.tabPage2.Controls.Add(this.label5);
this.tabPage2.Controls.Add(this.tbRemotePassword);
this.tabPage2.Controls.Add(this.btnConnectIP);
this.tabPage2.Controls.Add(this.label3);
this.tabPage2.Controls.Add(this.tbRemoteIP);
this.tabPage2.Controls.Add(this.cbAutoConnectNetwork);
this.tabPage2.Location = new System.Drawing.Point(4, 22);
this.tabPage2.Name = "tabPage2";
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(307, 265);
this.tabPage2.TabIndex = 1;
this.tabPage2.Text = "Remote Robot";
this.tabPage2.UseVisualStyleBackColor = true;
//
// groupBox2
//
this.groupBox2.Controls.Add(this.lblWheelVelocityR);
this.groupBox2.Controls.Add(this.lblBumpersR);
this.groupBox2.Controls.Add(this.lblWheelDropR);

```

```

this.groupBox2.Controls.Add(this.lblCliffR);
this.groupBox2.Location = new System.Drawing.Point(6, 158);
this.groupBox2.Name = "groupBox2";
this.groupBox2.Size = new System.Drawing.Size(292, 56);
this.groupBox2.TabIndex = 37;
this.groupBox2.TabStop = false;
this.groupBox2.Text = "Sensors";
//
// lblWheelVelocityR
//
this.lblWheelVelocityR.AutoSize = true;
this.lblWheelVelocityR.Location = new System.Drawing.Point(6, 16);
this.lblWheelVelocityR.Name = "lblWheelVelocityR";
this.lblWheelVelocityR.Size = new System.Drawing.Size(99, 13);
this.lblWheelVelocityR.TabIndex = 8;
this.lblWheelVelocityR.Text = "Wheel Velocity: 0 0";
//
// lblBumpersR
//
this.lblBumpersR.AutoSize = true;
this.lblBumpersR.Location = new System.Drawing.Point(151, 32);
this.lblBumpersR.Name = "lblBumpersR";
this.lblBumpersR.Size = new System.Drawing.Size(69, 13);
this.lblBumpersR.TabIndex = 5;
this.lblBumpersR.Text = "Bumpers: 0 0";
//
// lblWheelDropR
//
this.lblWheelDropR.AutoSize = true;
this.lblWheelDropR.Location = new System.Drawing.Point(5, 32);
this.lblWheelDropR.Name = "lblWheelDropR";
this.lblWheelDropR.Size = new System.Drawing.Size(91, 13);
this.lblWheelDropR.TabIndex = 6;
this.lblWheelDropR.Text = "WheelDrop: 0 0 0";
//
// lblCliffR
//
this.lblCliffR.AutoSize = true;
this.lblCliffR.Location = new System.Drawing.Point(151, 16);
this.lblCliffR.Name = "lblCliffR";
this.lblCliffR.Size = new System.Drawing.Size(63, 13);
this.lblCliffR.TabIndex = 7;
this.lblCliffR.Text = "Cliff: 0 0 0 0";
//
// lblBatteryR
//
this.lblBatteryR.AutoSize = true;
this.lblBatteryR.Location = new System.Drawing.Point(6, 220);
this.lblBatteryR.Name = "lblBatteryR";
this.lblBatteryR.Size = new System.Drawing.Size(114, 13);
this.lblBatteryR.TabIndex = 35;
this.lblBatteryR.Text = "Remote Batteries: N/A";
//
// pbBatteryR
//
this.pbBatteryR.Location = new System.Drawing.Point(6, 236);
this.pbBatteryR.Name = "pbBatteryR";
this.pbBatteryR.Size = new System.Drawing.Size(292, 23);

```

```

        this.pbBatteryR.TabIndex = 36;
        //
        // tbRemotePort
        //
        this.tbRemotePort.Location = new System.Drawing.Point(193, 36);
        this.tbRemotePort.Name = "tbRemotePort";
        this.tbRemotePort.Size = new System.Drawing.Size(56, 20);
        this.tbRemotePort.TabIndex = 34;
        this.tbRemotePort.Text = "25000";
        this.tbRemotePort.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.tbRemotePort_KeyPress);
        //
        // label7
        //
        this.label7.AutoSize = true;
        this.label7.Location = new System.Drawing.Point(131, 41);
        this.label7.Name = "label7";
        this.label7.Size = new System.Drawing.Size(29, 13);
        this.label7.TabIndex = 33;
        this.label7.Text = "Port:";
        //
        // label5
        //
        this.label5.AutoSize = true;
        this.label5.Location = new System.Drawing.Point(130, 64);
        this.label5.Name = "label5";
        this.label5.Size = new System.Drawing.Size(56, 13);
        this.label5.TabIndex = 31;
        this.label5.Text = "Password:";
        //
        // tbRemotePassword
        //
        this.tbRemotePassword.Location = new System.Drawing.Point(193, 62);
        this.tbRemotePassword.Name = "tbRemotePassword";
        this.tbRemotePassword.Size = new System.Drawing.Size(100, 20);
        this.tbRemotePassword.TabIndex = 30;
        this.tbRemotePassword.Text = "abracadabra";
        this.tbRemotePassword.TextChanged += new
System.EventHandler(this.tbRemotePassword_TextChanged);
        this.tbRemotePassword.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.tbRemotePassword_KeyPress);
        //
        // tabJoystick
        //
        this.tabJoystick.Controls.Add(this.pictureBox1);
        this.tabJoystick.Location = new System.Drawing.Point(4, 22);
        this.tabJoystick.Name = "tabJoystick";
        this.tabJoystick.Padding = new System.Windows.Forms.Padding(3);
        this.tabJoystick.Size = new System.Drawing.Size(307, 265);
        this.tabJoystick.TabIndex = 2;
        this.tabJoystick.Text = "Visual Joystick";
        this.tabJoystick.UseVisualStyleBackColor = true;
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(335, 403);

```

```

        this.Controls.Add(this.tabControl);
        this.Controls.Add(this.btnStop);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.tbKeyBoardSpeed);
        this.Controls.Add(this.btnKeyboardControl);
        this.Controls.Add(this.lblStatus);
        this.Controls.Add(this.btnCoverAndDock);
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
        this.MaximizeBox = false;
        this.Name = "Form1";
        this.Text = "iRobot Create Control";
        this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.Form1_FormClosing);
        this.Load += new System.EventHandler(this.Form1_Load);
        this.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.Form1_KeyPress);

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.tabControl.ResumeLayout(false);
        this.tabPage1.ResumeLayout(false);
        this.tabPage1.PerformLayout();
        this.groupBox1.ResumeLayout(false);
        this.groupBox1.PerformLayout();
        this.tabPage2.ResumeLayout(false);
        this.tabPage2.PerformLayout();
        this.groupBox2.ResumeLayout(false);
        this.groupBox2.PerformLayout();
        this.tabJoystick.ResumeLayout(false);
        this.ResumeLayout(false);
        this.PerformLayout();

    }

#endregion

private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.Label lblBumpers;
private System.Windows.Forms.Label lblWheelDrop;
private System.Windows.Forms.Label lblCliff;
private System.Windows.Forms.Label lblWheelVelocity;
private System.Windows.Forms.Button btnCoverAndDock;
private System.Windows.Forms.Button btnControl;
private System.Windows.Forms.Label lblBattery;
private System.Windows.Forms.TextBox tbCOMPort;
private System.Windows.Forms.TextBox tbRemoteIP;
private System.Windows.Forms.Button btnConnectIP;
private System.Windows.Forms.Label lblStatus;
private System.Windows.Forms.Label lblLocalIP;
private System.Windows.Forms.Label lblExternalIP;
private System.Windows.Forms.CheckBox cbAutoConnectCOM;
private System.Windows.Forms.CheckBox cbAutoConnectNetwork;
private System.Windows.Forms.ProgressBar pbBattery;
private System.Windows.Forms.Button btnStartServer;
private System.Windows.Forms.CheckBox cbAutoStartServer;
private System.Windows.Forms.Button btnKeyboardControl;
private System.Windows.Forms.TextBox tbKeyBoardSpeed;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Button btnStop;

```

```
private System.Windows.Forms.Label label2;  
private System.Windows.Forms.Label label3;  
private System.Windows.Forms.TabControl tabControl;  
private System.Windows.Forms.TabPage tabPage1;  
private System.Windows.Forms.TabPage tabPage2;  
private System.Windows.Forms.GroupBox groupBox1;  
private System.Windows.Forms.Label label4;  
private System.Windows.Forms.TextBox tbServerPassword;  
private System.Windows.Forms.Label label5;  
private System.Windows.Forms.TextBox tbRemotePassword;  
private System.Windows.Forms.TabPage tabPageJoystick;  
private System.Windows.Forms.TextBox tbServerPort;  
private System.Windows.Forms.Label label6;  
private System.Windows.Forms.TextBox tbRemotePort;  
private System.Windows.Forms.Label label7;  
private System.Windows.Forms.GroupBox groupBox2;  
private System.Windows.Forms.Label lblWheelVelocityR;  
private System.Windows.Forms.Label lblBumpersR;  
private System.Windows.Forms.Label lblWheelDropR;  
private System.Windows.Forms.Label lblCliffR;  
private System.Windows.Forms.Label lblBatteryR;  
private System.Windows.Forms.ProgressBar pbBatteryR;  
    }  
}
```

Anexo 1.1.2. Código del formulario

Código del funcionamiento de la interfaz del programa iCC (Form1.cs)

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.Net.Sockets;
using System.Net;
using iRobot;
using System.IO;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace iRobotCreateControl
{
    public partial class Form1 : Form
    {
        string getMyIP_URL = "http://checkip.dyndns.com/";

        public PowerStatus power = SystemInformation.PowerStatus;

        iRobotCreate robot;
        Bitmap joyBitmap;
        Graphics joyGraphics;
        bool mouseDown = false;
        long lastNetworkMessageTimestamp = 0;
        UdpClient udpClientListener;
        IPEndPoint listenerEndPoint = null;
        UdpClient udpClientSender = null;
        IPEndPoint senderEndPoint = null;
        bool updating = false;

        bool networkConfigurationTasksComplete = false;

        bool bControlAllowed = false;

        Point mousePoint = new Point();
        Thread senderThread;
        Thread recieveHeartbeatThread = null;
        Thread delayedConnectThread;
        Thread networkConfigurationThread;

        long lastSentNetworkMessageTimestamp = 0;
        long lastRecievedNetworkMessageTimestamp = 0;
    }
}

```

```

bool[] KeyState = new bool[500];

[DllImport("user32.dll", SetLastError = true)]
static extern uint SendInput(uint nInputs, ref INPUT pInputs, int cbSize);

[StructLayout(LayoutKind.Sequential)]
public struct KEYBDINPUT
{
    public ushort wVk;
    public ushort wScan;
    public uint dwFlags;
    public uint time;
    public IntPtr dwExtraInfo;
}

[StructLayout(LayoutKind.Explicit, Size = 28)]
public struct INPUT
{
    [FieldOffset(0)]
    public uint type;
    [FieldOffset(4)]
    public KEYBDINPUT ki;
};

public enum Win32Consts
{
    // For use with the INPUT struct, see SendInput for an example
    INPUT_MOUSE = 0,
    INPUT_KEYBOARD = 1,
    INPUT_HARDWARE = 2,
}

[DllImport("user32.dll")]
private static extern IntPtr GetForegroundWindow();
[DllImport("user32.dll")]
static extern int GetWindowTextLength(IntPtr hWnd);
// int GetWindowText(
//     __in HWND hWnd,
//     __out LPTSTR lpString,
//     __in int nMaxCount
// );
[DllImport("user32.dll")]
private static extern int GetWindowText(IntPtr hWnd, StringBuilder lpString, int
nMaxCount);
// DWORD GetWindowThreadProcessId(
//     __in HWND hWnd,
//     __out LPDWORD lpdwProcessId
// );
[DllImport("user32.dll")]
private static extern uint GetWindowThreadProcessId(IntPtr hWnd, out uint
lpdwProcessId);
//HANDLE WINAPI OpenProcess(
//     __in DWORD dwDesiredAccess,
//     __in BOOL bInheritHandle,
//     __in DWORD dwProcessId    //);
[DllImport("kernel32.dll")]

```

```

        private static extern IntPtr OpenProcess(uint dwDesiredAccess, bool
bInheritHandle, uint dwProcessId);

        [DllImport("kernel32.dll")]
        private static extern bool CloseHandle(IntPtr handle);
        //  DWORD WINAPI GetModuleBaseName(
        //      __in      HANDLE hProcess,
        //      __in_opt   HMODULE hModule,
        //      __out      LPTSTR lpBaseName,
        //      __in      DWORD nSize
        //  );
        [DllImport("psapi.dll")]
        private static extern uint GetModuleBaseName(IntPtr hWnd, IntPtr hModule,
StringBuilder lpFileName, int nSize);
        //  DWORD WINAPI GetModuleFileNameEx(
        //      __in      HANDLE hProcess,
        //      __in_opt   HMODULE hModule,
        //      __out      LPTSTR lpFilename,
        //      __in      DWORD nSize
        //  );
        [DllImport("psapi.dll")]
        private static extern uint GetModuleFileNameEx(IntPtr hWnd, IntPtr hModule,
StringBuilder lpFileName, int nSize);
        public static string GetTopWindowText()
        {
            IntPtr hWnd = GetForegroundWindow();
            int length = GetWindowTextLength(hWnd);
            StringBuilder text = new StringBuilder(length + 1);
            GetWindowText(hWnd, text, text.Capacity);
            return text.ToString();
        }

        [DllImport("user32.dll", CharSet=CharSet.Auto, ExactSpelling=true)]
        public static extern IntPtr SetFocus(HandleRef hWnd);

        [DllImport("User32.dll")]
        public static extern Int32 SetForegroundWindow(IntPtr hWnd);

        [DllImport("user32.dll")]
        public static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

        private void BringToFront(string windowName)
        {
            foreach (Process p in Process.GetProcesses("."))
            {
                try
                {
                    if (p.MainWindowTitle.Length > 0)
                    {
                        string windowTitle = p.MainWindowTitle.ToString();
                        Console.WriteLine("Window Title:" + windowTitle);
                        Console.WriteLine("Process Name:" + p.ProcessName.ToString());
                        // Console.WriteLine("Window Handle:" +
p.MainWindowHandle.ToString());

                        if (windowTitle.CompareTo(windowName) == 0)

```

```

        {
            Console.WriteLine("\tmatch-----");
            IntPtr hWnd = p.MainWindowHandle;
            SetForegroundWindow(hWnd);

            Thread.Sleep(500);
            Console.Out.WriteLine("pressing key");

            INPUT structInput;
            structInput = new INPUT();
            structInput.type = (uint)Win32Consts.INPUT_KEYBOARD;

            // Key down shift, ctrl, and/or alt
            structInput.ki.wScan = 0;
            structInput.ki.time = 0;
            structInput.ki.dwFlags = 0x0;
            structInput.ki.wVk = (ushort)0x0D;
            SendInput(1, ref structInput, Marshal.SizeOf(new
// INPUT()));
        }
    }
    catch { }
}
//SetForegroundWindow(FindWindow(className, CaptionName));
}

public static string GetTopWindowName()
{
    IntPtr hWnd = GetForegroundWindow();
    uint lpdwProcessId;
    GetWindowThreadProcessId(hWnd, out lpdwProcessId);
    IntPtr hProcess = OpenProcess(0x0410, false, lpdwProcessId);
    StringBuilder text = new StringBuilder(1000);

    //GetModuleBaseName(hProcess, IntPtr.Zero, text, text.Capacity);
    GetModuleFileNameEx(hProcess, IntPtr.Zero, text, text.Capacity);
    CloseHandle(hProcess);
    return text.ToString();
}

public Form1()
{
    InitializeComponent();

    joyBitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
    joyGraphics = Graphics.FromImage(joyBitmap);

    Point center = new Point(joyBitmap.Width / 2, joyBitmap.Height / 2);
    joyGraphics.Clear(Color.White);
    int size = 10;
    joyGraphics.DrawEllipse(new Pen(Color.Red), center.X - size / 2, center.Y
- size / 2, size, size);
}

```

```

        pictureBox1.Image = joyBitmap;
        mousePoint = new Point(joyBitmap.Width / 2, joyBitmap.Height / 2);
    }

    void PerformNetworkConfigurationTasks()
    {
        lblStatus.Invoke((MethodInvoker)(() => lblStatus.Text = "Status: Getting IP
addresses"));

        //get ip address
        IPAddress[] localIPs = Dns.GetHostAddresses(Dns.GetHostName());
        for (int i = 0; i < localIPs.Length; i++)
        {
            if (localIPs[i].AddressFamily == AddressFamily.InterNetwork)
            {
                Console.WriteLine("Local IP: " + localIPs[i]);
                lblLocalIP.Invoke((MethodInvoker)(() => lblLocalIP.Text = "Local IP
Address: " + localIPs[i]));
                break;
            }
        }

        if (true)//query external ip
        {
            Console.WriteLine("Querying for external IP: " + getMyIP_URL);
            string externalIP;
            try
            {
                externalIP = WebQuery(getMyIP_URL);
                char[] separators = { ' ', '<' };
                string[] parts = externalIP.Split(new char[] { ' ', '<' });
                for (int i = 0; i < parts.Length; i++)
                {
                    if (parts[i].CompareTo("Address:") == 0)
                    {
                        externalIP = parts[i + 1];
                        break;
                    }
                }
            }
            catch (Exception e)
            {
                externalIP = "190.2.126.179";
            }

            Console.WriteLine("Returned External IP: " + externalIP);
            lblExternalIP.Invoke((MethodInvoker)(() => lblExternalIP.Text =
"External IP Address: " + externalIP));
        }

        Console.WriteLine("Configuring router port forwarding: " +
tbServerPort.Text);
        lblStatus.Invoke((MethodInvoker)(() => lblStatus.Text = "Status:
Configuring router..."));
        if (UPnP.OpenFirewallPort("iRobot", UPnP.Protocol.UDP,
int.Parse(tbServerPort.Text)))
        {

```

```

        lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status: Port
forwarding set: " + tbServerPort.Text)));
    }
    else
    {
        lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status:
Deleting old port forwarding..."));
        if (!UPnP.CloseFirewallPort(UPnP.Protocol.UDP,
int.Parse(tbServerPort.Text)))
        {
            lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status:
Port forwarding change FAILED"));
            MessageBox.Show("Error occurred during router configuration.");
            return;
        }
        Thread.Sleep(500);
        lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status: Adding
new port forwarding..."));
        if (UPnP.OpenFirewallPort("iRobot", UPnP.Protocol.UDP,
int.Parse(tbServerPort.Text)))
        {
            lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status:
Port forwarded to this computer"));
        }
        else
        {
            lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status:
Port forwarding change FAILED"));
            MessageBox.Show("Error occurred during router configuration.");
        }
    }

    if (true)
    {
        lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status:
Starting server on port " + tbServerPort.Text)));
        listenerEndPoint = new IPEndPoint(IPAddress.Any,
int.Parse(tbServerPort.Text));
        udpClientListener = new UdpClient(listenerEndPoint);
        Console.Out.WriteLine("Listening on port " + tbServerPort.Text);
        udpClientListener.BeginReceive(new AsyncCallback(ReceiveCallback),
this);
    }

    lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text = "Status: Server
Started on port " + tbServerPort.Text)));
    btnStartServer.Invoke((MethodInvoker)((() => btnStartServer.Text = "Disable
Remote Control"));
    btnStartServer.Invoke((MethodInvoker)((() => btnStartServer.Enabled =
true)));

    networkConfigurationTasksComplete = true;
}

void QueryTopWindow(object o)
{
    Thread.Sleep(4000);
    while(this.Visible)

```

```

        {
//            BringToFront("Sponsored session");
            BringToFront("TeamViewer Panel");

            Thread.Sleep(2000);
        }
    }

    void SaveSettings()
    {
        if (File.Exists("config.txt"))
        {
            File.Delete("config.txt");
            Thread.Sleep(100);
        }

        FileStream fs = new FileStream("config.txt", FileMode.CreateNew);
        StreamWriter filewrite = new StreamWriter(fs);
        filewrite.WriteLine("AutoConnectLocalRobot: " + cbAutoConnectCOM.Checked);
        filewrite.WriteLine("COMPort: " + tbCOMPort.Text);
        filewrite.WriteLine("AutoStartServer: " + cbAutoStartServer.Checked);
        filewrite.WriteLine("ServerPort: " + tbServerPort.Text);
        filewrite.WriteLine("ServerPassword: " + tbServerPassword.Text);
        filewrite.WriteLine("AutoConnectRemoteRobot: " +
cbAutoConnectNetwork.Checked);
        filewrite.WriteLine("RemoteIP: " + tbRemoteIP.Text);
        filewrite.WriteLine("RemotePort: " + tbRemotePort.Text);
        filewrite.WriteLine("RemotePassword: " + tbRemotePassword.Text);
        filewrite.WriteLine("KeyboardControlMotorSpeed: " + tbKeyBoardSpeed.Text);
        filewrite.Flush();
        filewrite.Close();
        fs.Close();
    }

    private string WebQuery(string url)
    {
        // used to build entire input
        StringBuilder sb = new StringBuilder();

        // used on each read operation
        byte[] buf = new byte[8192];
        string tempString = null;

        try
        {
            // prepare the web page we will be asking for
            HttpRequest request = (HttpRequest)WebRequest.Create(url);
            request.Timeout = 2000;

            // execute the request
            HttpResponse response = (HttpResponse)request.GetResponse();
            // we will read data via the response stream
            Stream resStream = response.GetResponseStream();

            int count = 0;

            do
            {

```

```

        // fill the buffer with data
        count = resStream.Read(buf, 0, buf.Length);

        // make sure we read some data
        if (count != 0)
        {
            // translate from bytes to ASCII text
            tempString = Encoding.ASCII.GetString(buf, 0, count);

            // continue building the string
            sb.Append(tempString);
        }
        while (count > 0); // any more data to read?
    }
    catch (Exception x)
    {
        Console.Out.WriteLine("could no contact IP address server: " +
x.ToString());
    }

    return tempString;
}

void DelayedConnectCallback()
{
    Thread.Sleep(1000);
    if (cbAutoConnectCOM.Checked)
    {
        this.Invoke((MethodInvoker)delegate
        {
            btnControl_Click(this, null);
        });
    }
    if (cbAutoConnectNetwork.Checked)
    {
        this.Invoke((MethodInvoker)delegate
        {
            btnConnectIP_Click(this, null);
        });
    }
}

public static void ResponseCallback(IAsyncResult ar)
{
    try
    {
        Form1 f = (Form1)ar.AsyncState;
        if (f.udpClientSender == null)
            return;
        Byte[] receiveBytes = f.udpClientSender.EndReceive(ar, ref
f.senderEndPoint);
        string receiveString = Encoding.ASCII.GetString(receiveBytes);

        string[] parts = receiveString.Split(' ');
    }
}

```

```

        if (parts[0].CompareTo("control") == 0)
        {
            if (parts[1].CompareTo("ok") == 0)
            {
                f.Invoke((MethodInvoker)delegate
                {
                    Console.Out.WriteLine("Controlling robot at " +
f.tbRemoteIP.Text);
                    f.lblStatus.Text = "Status: Controlling robot at " +
f.tbRemoteIP.Text;
                });
            }

            if (parts[1].CompareTo("no") == 0)
            {
                f.Invoke((MethodInvoker)delegate
                {
                    f.lblStatus.Text = "Status: " + f.tbRemoteIP.Text + "
refused, password is incorrect!";
                });
            }
        }
        if (parts[0].CompareTo("s") == 0)
        {
            f.Invoke((MethodInvoker)delegate
            {
                int batteryStateEnum = int.Parse(parts[1]);
                int batteryCharge = int.Parse(parts[2]);
                int sensorStatus = int.Parse(parts[3]);

                f.robot.sensorState.BumpLeft = ((sensorStatus & 0x0001)>0);
                f.robot.sensorState.BumpRight = ((sensorStatus & 0x0002)>0);
                f.robot.sensorState.WheelDropLeft = ((sensorStatus &
0x0004)>0);
                f.robot.sensorState.WheelDropCaster = ((sensorStatus &
0x0008)>0);
                f.robot.sensorState.WheelDropRight = ((sensorStatus &
0x0010)>0);
                f.robot.sensorState.CliffLeft = ((sensorStatus & 0x0020)>0);
                f.robot.sensorState.CliffFrontLeft = ((sensorStatus &
0x0040)>0);
                f.robot.sensorState.CliffFrontRight = ((sensorStatus &
0x0080)>0);
                f.robot.sensorState.CliffRight = ((sensorStatus & 0x0100) > 0);

                string batteryState = "Idle";
                if (batteryStateEnum == 1)
                    batteryState = "Reconditioning";
                if (batteryStateEnum == 2)
                    batteryState = "Charging";
                if (batteryStateEnum == 3)
                    batteryState = "Trickle";
                if (batteryStateEnum == 4)
                    batteryState = "Waiting";
                if (batteryStateEnum == 5)
                    batteryState = "Fault";
            }
        }
    }
}

```

```

        f.lblBatteryR.Text = "Remote Batteries: " + batteryState + " "
+ batteryCharge + "%";
        if (batteryCharge > 100) batteryCharge = 100;
        if (batteryCharge < 0) batteryCharge = 0;
        f.pbBatteryR.Value = batteryCharge;

        f.lblBumpersR.Text = "Bumpers: " +
(f.robot.sensorState.BumpLeft ? 0 : 1) + " " + (f.robot.sensorState.BumpRight ? 0 : 1);
        f.lblWheelDropR.Text = "WheelDrop: " +
(f.robot.sensorState.WheelDropLeft ? 0 : 1) + " " + (f.robot.sensorState.WheelDropCaster
? 0 : 1) + " " + (f.robot.sensorState.WheelDropRight ? 0 : 1);
        f.lblCliffR.Text = "Cliff: " + (f.robot.sensorState.CliffLeft
? 0 : 1) + " " + (f.robot.sensorState.CliffFrontLeft ? 0 : 1) + " " +
(f.robot.sensorState.CliffFrontRight ? 0 : 1) + " " + (f.robot.sensorState.CliffRight
? 0 : 1);

        f.lblStatus.Text = "Status: Controlling robot at " +
f.tbRemoteIP.Text + ":" + f.tbRemotePort.Text;

    });
}

if (f.udpClientSender == null)
    return;
f.udpClientSender.BeginReceive(new AsyncCallback(ResponseCallback),
f);
}
catch (Exception x)
{
    Console.Out.WriteLine(x);
}

}

public static void ReceiveCallback(IAsyncResult ar)
{
    Form1 f = (Form1)ar.AsyncState;

    f.lastRecievedNetworkMessageTimestamp = System.Environment.TickCount;
    if (f.recieveHeartbeatThread == null)
    {
        //we received a network message, now require that we regularly receive
it
        f.recieveHeartbeatThread = new Thread(new
ThreadStart(f.RecieveHeartbeatCallback));
        f.recieveHeartbeatThread.Start();
    }
    if (f.udpClientListener.Client == null)
        return;

    Byte[] receiveBytes = f.udpClientListener.EndReceive(ar, ref
f.listenerEndPoint);
    string receiveString = Encoding.ASCII.GetString(receiveBytes);
    string[] parts = receiveString.Split(' ');
    if (parts.Length == 2)
    {
        if (parts[0].CompareTo("control") == 0)
        {

```

```

Console.Out.WriteLine("received connection request");
if (parts[1].CompareTo(f.tbServerPassword.Text) == 0)
{
    f.btnControl_Click(f, null);
    f.bControlAllowed = true;

    Console.Out.WriteLine("remote control granted!");
    f.robot.StartInFullMode();

    f.lblStatus.Invoke((MethodInvoker)((() => f.lblStatus.Text =
"Status: Controlled by remote host")));

    Byte[] sendBytes = Encoding.ASCII.GetBytes("control ok");
    f.udpClientListener.Send(sendBytes,
sendBytes.Length, f.listenerEndPoint);

    if (true)//send status message
    {
        int sensorState = (f.robot.sensorState.BumpLeft ? 0x0001 :
0) |
                                (f.robot.sensorState.BumpRight ? 0x0002 :
0) |
                                (f.robot.sensorState.WheelDropLeft ?
0x0004 : 0) |
                                (f.robot.sensorState.WheelDropCaster ?
0x0008 : 0) |
                                (f.robot.sensorState.WheelDropRight ?
0x0010 : 0) |
                                (f.robot.sensorState.CliffLeft ? 0x0020 :
0) |
                                (f.robot.sensorState.CliffFrontLeft ?
0x0040 : 0) |
                                (f.robot.sensorState.CliffFrontRight ?
0x0080 : 0) |
                                (f.robot.sensorState.CliffRight ? 0x0100 :
0);

        bool computerIsCharging = (f.power.PowerLineStatus ==
PowerLineStatus.Online);
        if (f.robot.sensorState.BatteryCapacity != 0)
        {
            int robotCharge = (f.robot.sensorState.BatteryCharge *
100 / f.robot.sensorState.BatteryCapacity);
            int batteryCharge = (int)(f.power.BatteryLifePercent *
100);

            if (robotCharge < batteryCharge)
                batteryCharge = robotCharge;
            Byte[] sendBytes2 = Encoding.ASCII.GetBytes("s " +
f.robot.sensorState.ChargingState + " " + batteryCharge + " " + sensorState);
            f.udpClientListener.Send(sendBytes2,
sendBytes2.Length, f.listenerEndPoint);
        }
    }
}
else
{

```

```

        f.bControlAllowed = false;
        Console.Out.WriteLine("password mismatch, control denied");

        Byte[] sendBytes = Encoding.ASCII.GetBytes("control no");
        f.udpClientListener.Send(sendBytes, sendBytes.Length,
f.listenerEndPoint);
    }
    }
    else
    {
        int x = int.Parse(parts[0]);
        int y = int.Parse(parts[1]);
        if ((!f.updating)&&f.bControlAllowed)
        {
            f.Invoke((MethodInvoker)delegate
            {
                f.UpdateJoystickAndDriveCommand(new
MouseEventArgs(MouseButtons.Left, 0, x, y, 0));
            });

            if (true)//send status message
            {
                int sensorState = (f.robot.sensorState.BumpLeft ? 0x0001 :
0) |
                                (f.robot.sensorState.BumpRight ? 0x0002 :
0) |
                                (f.robot.sensorState.WheelDropLeft ?
0x0004 : 0) |
                                (f.robot.sensorState.WheelDropCaster ?
0x0008 : 0) |
                                (f.robot.sensorState.WheelDropRight ?
0x0010 : 0) |
                                (f.robot.sensorState.CliffLeft ? 0x0020 :
0) |
                                (f.robot.sensorState.CliffFrontLeft ?
0x0040 : 0) |
                                (f.robot.sensorState.CliffFrontRight ?
0x0080 : 0) |
                                (f.robot.sensorState.CliffRight ? 0x0100 :
0);

                if (f.robot.sensorState.BatteryCapacity != 0)
                {
                    int robotCharge = (f.robot.sensorState.BatteryCharge *
100 / f.robot.sensorState.BatteryCapacity);
                    int batteryCharge = (int)(f.power.BatteryLifePercent *
100);

                    if (robotCharge < batteryCharge)
                        batteryCharge = robotCharge;

                    Byte[] sendBytes2 = Encoding.ASCII.GetBytes("s " +
f.robot.sensorState.ChargingState + " " + batteryCharge + " " + sensorState);
                    f.udpClientListener.Send(sendBytes2, sendBytes2.Length,
f.listenerEndPoint);
                }
            }
        }
    }
}

```

```

    }
    if (parts.Length == 1)
    {
        if (f.bControlAllowed)
        {
            if (parts[0].CompareTo("cover_and_dock") == 0)
                f.btnCoverAndDock_Click(f, null);
            if (parts[0].CompareTo("home") == 0)
                f.btnStop_Click(f, null);
        }
    }

    if (f.Visible)
        f.udpClientListener.BeginReceive(new AsyncCallback(ReceiveCallback),
f);
}

private void OnWheelDropChanged(object sender, EventArgs e)
{
    if (!robot.prevSensorState.WheelDropLeft &&
robot.sensorState.WheelDropLeft)
        Console.WriteLine("wheel left dropped!");
    if (!robot.prevSensorState.WheelDropRight &&
robot.sensorState.WheelDropRight)
        Console.WriteLine("wheel right dropped!");
    if (!robot.prevSensorState.WheelDropCaster &&
robot.sensorState.WheelDropCaster)
        Console.WriteLine("wheel caster dropped!");
}

private void OnCliffDetectChanged(object sender, EventArgs e)
{
    if (!robot.prevSensorState.CliffLeft && robot.sensorState.CliffLeft)
        Console.WriteLine("cliff left!");
    if (!robot.prevSensorState.CliffFrontLeft &&
robot.sensorState.CliffFrontLeft)
        Console.WriteLine("cliff front left!");
    if (!robot.prevSensorState.CliffFrontRight &&
robot.sensorState.CliffFrontRight)
        Console.WriteLine("cliff front right!");
    if (!robot.prevSensorState.CliffRight && robot.sensorState.CliffRight)
        Console.WriteLine("cliff right!");
}

private void OnBumperChanged(object sender, EventArgs e)
{
    if(!robot.prevSensorState.BumpLeft && robot.sensorState.BumpLeft)
        Console.WriteLine("bump left!");
    if (!robot.prevSensorState.BumpRight && robot.sensorState.BumpRight)
        Console.WriteLine("bump right!");
}

private void OnSensorUpdate(object sender, EventArgs e)
{
    try
    {

```

```

        if (this.IsDisposed)
            return;
        this.Invoke((MethodInvoker)delegate
        {
            UpdateSensorIU();
        });
    }
    catch (Exception x)
    {
        Console.Out.WriteLine(x);
    }
}

private void UpdateSensorIU()
{
    if (!this.Visible)
        return;
    if (this.IsDisposed)
        return;
    lblBumpers.Text = "Bumpers: " + (robot.sensorState.BumpLeft ? 0 : 1) + " "
+ (robot.sensorState.BumpRight ? 0 : 1);
    lblWheelDrop.Text = "WheelDrop: " + (robot.sensorState.WheelDropLeft ? 0 :
1) + " " + (robot.sensorState.WheelDropCaster ? 0 : 1) + " " +
(robot.sensorState.WheelDropRight ? 0 : 1);
    lblCliff.Text = "Cliff: " + (robot.sensorState.CliffLeft ? 0 : 1) + " " +
(robot.sensorState.CliffFrontLeft ? 0 : 1) + " " + (robot.sensorState.CliffFrontRight
? 0 : 1) + " " + (robot.sensorState.CliffRight ? 0 : 1);
    string state = "Idle";
    if (robot.sensorState.ChargingState == 1)
        state = "Reconditioning";
    if (robot.sensorState.ChargingState == 2)
        state = "Charging";
    if (robot.sensorState.ChargingState == 3)
        state = "Trickle";
    if (robot.sensorState.ChargingState == 4)
        state = "Waiting";
    if (robot.sensorState.ChargingState == 5)
        state = "Fault";
    if (robot.sensorState.BatteryCapacity != 0)
    {
        int batteryCharge = (robot.sensorState.BatteryCharge * 100 /
robot.sensorState.BatteryCapacity);

        lblBattery.Text = "iRobot Battery: " + state + " " + batteryCharge + "%
(" + ((robot.sensorState.Current > 0) ? "+" : "") + robot.sensorState.Current / 1000.0
+ "A " + robot.sensorState.Voltage / 1000.0 + "V " + robot.sensorState.BatteryTemperature
+ "C)";

        if (batteryCharge > 100) batteryCharge = 100;
        if (batteryCharge < 0) batteryCharge = 0;
        pbBattery.Value = batteryCharge;
    }
}

private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    mouseDown = true;
}

```

```

        UpdateJoystickAndDriveCommand(e);
    }

    void UpdateJoystickAndDriveCommand(MouseEventArgs e)
    {
        updating = true;
        try
        {
            mousePoint.X = e.X;
            mousePoint.Y = e.Y;
            Point center = new Point(joyBitmap.Width / 2, joyBitmap.Height / 2);
            joyGraphics.Clear(Color.White);
            int size = 10;
            int dx = (e.X - center.X)/2;
            int dy = -1*(center.Y - e.Y);
            float mag = (float)Math.Sqrt(dx * dx + dy * dy);
            float maxRad = 200;
            float scale = maxRad / mag;
            if (scale < 1)
            {
                dx = (int)(dx * scale);
                dy = (int)(dy * scale);
            }

            Point newE = new Point(center.X + 2*dx, center.Y + dy);

            joyGraphics.DrawEllipse(new Pen(Color.Red), newE.X - size / 2, newE.Y
- size / 2, size, size);
            joyGraphics.DrawLine(new Pen(Color.Red), center, newE);
            pictureBox1.Image = joyBitmap;

            float maxVel = 400;
            dx = (int)(dx*maxVel/ maxRad);
            dy = (int)(dy*maxVel / maxRad);

            int left = (dy - dx);
            int right = (dy + dx);

            robot.DriveDirect(left, right);
            lblWheelVelocity.Invoke((MethodInvoker) (() => lblWheelVelocity.Text =
"WheelVelocity: " + left + " " + right));

            if ((Environment.TickCount - lastNetworkMessageTimestamp) > 50)
            {
                lastNetworkMessageTimestamp = Environment.TickCount;
                if (udpClientSender != null)
                {
                    if (udpClientSender.Client.Connected)
                    {
                        lblWheelVelocityR.Invoke((MethodInvoker) (() =>
lblWheelVelocityR.Text = "WheelVelocity: " + left + " " + right));

                        Byte[] sendBytes = Encoding.ASCII.GetBytes(mousePoint.X +
" " + mousePoint.Y);
                        udpClientSender.Send(sendBytes, sendBytes.Length);
                        lastSentNetworkMessageTimestamp =
System.Environment.TickCount;

```

```

        }
    }
}
catch (Exception x)
{
    Console.Out.WriteLine(x);
}
updating = false;
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (mouseDown)
    {
        UpdateJoystickAndDriveCommand(e);
    }
    else
    {
        robot.DriveDirect(0, 0);
    }
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    mouseDown = false;
    mousePoint = new Point(joyBitmap.Width / 2, joyBitmap.Height / 2);
    UpdateJoystickAndDriveCommand(new MouseEventArgs(MouseButtons.Left, 0,
mousePoint.X, mousePoint.Y, 0));
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (robot != null)
    {
        lblStatus.Invoke((MethodInvoker) (() => lblStatus.Text = "Status:
Powering Off Robot"));
        Console.Out.WriteLine("Shutting down local robot...");
        Thread.Sleep(100);
        robot.Disconnect();
    }

    if (udpClientSender != null)
    {
        udpClientSender.Close();
    }
    if (udpClientListener != null)
    {
        udpClientListener.Close();
    }
    SaveSettings();
}

private void btnCoverAndDock_Click(object sender, EventArgs e)
{
    if (robot.connected)
    {
        robot.StartInPassiveMode();
    }
}

```

```

        Thread.Sleep(500);
        robot.CoverAndDock();
    }
    //try remote send
    if (udpClientSender == null)
        return;
    if (udpClientSender.Client == null)
        return;
    if (udpClientSender.Client.Connected)
    {
        Byte[] sendBytes = Encoding.ASCII.GetBytes("cover_and_dock");
        udpClientSender.Send(sendBytes, sendBytes.Length);
        lastSentNetworkMessageTimestamp = System.Environment.TickCount;
    }
}

private void btnControl_Click(object sender, EventArgs e)
{
    if (!robot.connected)
    {
        if (robot.Connect(tbCOMPort.Text))
        {
            Console.Out.WriteLine("Connected to iRobot Create");
            //lblStatus.Text = "Status: " + "connected to Create on port " +
tbCOMPort.Text;
        }
        else
        {
            MessageBox.Show("Could not Connect to iRobot Create on port:" +
tbCOMPort.Text);

            this.Invoke((MethodInvoker)delegate
            {
                lblStatus.Text = "Status: Failed to connect to iRobot Create";
            });
        }
    }
    robot.StartInFullMode();
    robot.StartSensorStreaming();
}

void RecieveHeartbeatCallback()
{
    try
    {
        bool beenReset = false;
        while (this.Visible && !this.IsDisposed)
        {
            Thread.Sleep(250);
            long timeSinceLastRecievedMessage = System.Environment.TickCount -
lastRecievedNetworkMessageTimestamp;
            if (timeSinceLastRecievedMessage > 3000)
            {
                //don't bother resetting if we are already charging
                if (robot.sensorState.ChargingState == 2)
                    continue;
            }
        }
    }
}

```

```

        if (!beenReset)
        {
            //this will make it go into "off" state
            //which is lower power and can will accept a charge if already
            docked.
            Console.Out.WriteLine("Remote host disconnected, powering
            down");

            robot.SoftReset();
            beenReset = true;
            lblBattery.Invoke((MethodInvoker)((() => lblBattery.Text =
            "iRobot Battery: N/A")));
            pbBattery.Invoke((MethodInvoker)((() => pbBattery.Value =
            0)));
            lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text =
            "Status: Powered Off - Remote Control Disconnected")));
        }
        continue;
    }

    if (timeSinceLastRecievedMessage > 1000)
    {
        Console.Out.WriteLine("remote control heartbeat lost, stopping
        (poor network connection?)");
        mousePoint = new Point(joyBitmap.Width / 2, joyBitmap.Height /
        2);

        this.Invoke((MethodInvoker)delegate
        {
            UpdateJoystickAndDriveCommand(new
            MouseEventArgs(MouseButtons.Left, 0, mousePoint.X, mousePoint.Y, 0));
        });

        lblStatus.Invoke((MethodInvoker)((() => lblStatus.Text =
        "Status: Stopping - Remote Control Communication Lost")));

        continue;
    }

    //we are recieving and running, so no reset
    beenReset = false;
}
}
catch (Exception x)
{
    Console.Out.WriteLine(x);
}
}

void SendCallback()
{
    while (this.Visible)
    {
        if (udpClientSender.Client == null)
            return;
        if (udpClientSender.Client.Connected)
        {
            if ((System.Environment.TickCount -
            lastSentNetworkMessageTimestamp) > 100)
            {

```

```

        Byte[] sendBytes = Encoding.ASCII.GetBytes(mousePoint.X + " "
+ mousePoint.Y);
        udpClientSender.Send(sendBytes, sendBytes.Length);
        lastSentNetworkMessageTimestamp =
System.Environment.TickCount;
    }
}
else
{
    return;
}
Thread.Sleep(100);
}
}

private void btnConnectIP_Click(object sender, EventArgs e)
{
    if (udpClientSender == null)
    {
        senderEndPoint = new IPEndPoint(IPAddress.Any,
int.Parse(tbRemotePort.Text));
        udpClientSender = new UdpClient(senderEndPoint);
    }

    if (udpClientSender.Client.Connected)
    {
        Byte[] sendBytes = Encoding.ASCII.GetBytes("control
"+tbRemotePassword.Text);
        udpClientSender.Send(sendBytes, sendBytes.Length);
        lastSentNetworkMessageTimestamp = System.Environment.TickCount;
    }
    else
    {
        lblStatus.Text = "Status: Connecting to " + tbRemoteIP.Text + ":" +
tbRemotePort.Text + " ...";

        Console.Out.WriteLine("Connecting to:" + tbRemoteIP.Text + ":" +
tbRemotePort.Text);
        udpClientSender.Connect(tbRemoteIP.Text,
int.Parse(tbRemotePort.Text));
        if (udpClientSender.Client.Connected)
        {
            Console.Out.WriteLine("Awaiting repsonse...");
            lblStatus.Text = "Status: Awaiting repsonse from " + tbRemoteIP.Text
+ ":" + tbRemotePort.Text + " ...";

            senderThread = new Thread(new ThreadStart(SendCallback));
            senderThread.Start();

            udpClientSender.BeginReceive(new AsyncCallback(ResponseCallback),
this);

            Thread.Sleep(500);
            Byte[] sendBytes = Encoding.ASCII.GetBytes("control " +
tbRemotePassword.Text);
            udpClientSender.Send(sendBytes, sendBytes.Length);

```

```

        lastSentNetworkMessageTimestamp = System.Environment.TickCount;
    }
}

private void cbAutoConnect_CheckedChanged(object sender, EventArgs e)
{
}

private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
}

private void Form1_Load(object sender, EventArgs e)
{
    this.KeyPreview = true;

    robot = new iRobotCreate();
    robot.OnSensorUpdateRecieved += new
iRobotCreate.SensorUpdateHandler(OnSensorUpdate);
    robot.OnBumperChanged += new
iRobotCreate.BumperChangedHandler(OnBumperChanged);
    robot.OnCliffDetectChanged += new
iRobotCreate.CliffDetectChangedHandler(OnCliffDetectChanged);
    robot.OnWheelDropChanged += new
iRobotCreate.WheelDropChangedHandler(OnWheelDropChanged);

    if (File.Exists("config.txt"))
    {
        FileStream fs = new FileStream("config.txt", FileMode.OpenOrCreate);
        StreamReader filereader = new StreamReader(fs);
        try
        {
            string[] parts;
            parts = (filereader.ReadLine()).Split(' ');
            cbAutoConnectCOM.Checked = Boolean.Parse(parts[1]);

            parts = (filereader.ReadLine()).Split(' ');
            tbCOMPort.Text = parts[1];

            parts = (filereader.ReadLine()).Split(' ');
            cbAutoStartServer.Checked = Boolean.Parse(parts[1]);

            parts = (filereader.ReadLine()).Split(' ');
            tbServerPort.Text = int.Parse(parts[1]).ToString();

            parts = (filereader.ReadLine()).Split(' ');
            tbServerPassword.Text = parts[1];

            parts = (filereader.ReadLine()).Split(' ');
            cbAutoConnectNetwork.Checked = Boolean.Parse(parts[1]);

            if (cbAutoConnectNetwork.Checked)
            {
                tabControl.SelectedTab = tabControl.TabPages[1];
            }
        }
        catch { }
    }
}

```

```

    }

    parts = (filereader.ReadLine()).Split(' ');
    tbRemoteIP.Text = parts[1];

    parts = (filereader.ReadLine()).Split(' ');
    tbRemotePort.Text = int.Parse(parts[1]).ToString();

    parts = (filereader.ReadLine()).Split(' ');
    tbRemotePassword.Text = parts[1];

    parts = (filereader.ReadLine()).Split(' ');
    tbKeyBoardSpeed.Text = parts[1];

    Thread.Sleep(1000);

    //connect the local robot
    if (cbAutoConnectCOM.Checked || cbAutoConnectNetwork.Checked)
    {
        delayedConnectThread = new Thread(new
ThreadStart(DelayedConnectCallback));
        delayedConnectThread.Start();
    }
    if (cbAutoStartServer.Checked)
    {
        btnStartServer_Click(this, null);
    }
}
catch (Exception x)
{
    Console.Out.WriteLine("File read error: " + x.ToString());
}
filereader.Close();
fs.Close();
}
else
{
    if(udpClientListener != null)
        udpClientListener.Close();
    SaveSettings();
}
}

private void btnStartServer_Click(object sender, EventArgs e)
{
    if (networkConfigurationTasksComplete)
    {
        udpClientListener.Close();
        lblStatus.Text = "Status: Server Stopped";

        btnStartServer.Text = "Enable Remote Control";
        btnConnectIP.Enabled = true;
        networkConfigurationTasksComplete = false;
    }
    else
    {

```

```

        btnStartServer.Enabled = false;
        btnConnectIP.Enabled = false;
        btnStartServer.Text = "Starting server...";
        networkConfigurationThread = new Thread(new
ThreadStart(PerformNetworkConfigurationTasks));
        networkConfigurationThread.Start();
    }
}

private void button1_KeyDown(object sender, KeyEventArgs e)
{
    int offset = 100;
    try
    {
        offset = int.Parse(tbKeyBoardSpeed.Text);
    }
    catch (Exception ex)
    {
        Console.Out.WriteLine(ex);
    }

    KeyState[(int)e.KeyCode] = true;
    int x = joyBitmap.Width / 2;
    int y = joyBitmap.Height / 2;

    if (KeyState[(int)Keys.W])
        y -= offset;
    if (KeyState[(int)Keys.S])
        y += offset;
    if (KeyState[(int)Keys.A])
        x -= offset;
    if (KeyState[(int)Keys.D])
        x += offset;
    {
        MouseEventArgs me = new MouseEventArgs(MouseButtons.Left, 0, x, y, 0);
        UpdateJoystickAndDriveCommand(me);
    }
}

private void button1_KeyUp(object sender, KeyEventArgs e)
{
    KeyState[(int)e.KeyCode] = false;

    int offset = 100;
    try
    {
        offset = int.Parse(tbKeyBoardSpeed.Text);
    }
    catch (Exception ex)
    {
        Console.Out.WriteLine(ex);
    }

    int x = joyBitmap.Width / 2;
    int y = joyBitmap.Height / 2;

```

```

        if (KeyState[(int)Keys.W])
            y -= offset;
        if (KeyState[(int)Keys.S])
            y += offset;
        if (KeyState[(int)Keys.A])
            x -= offset;
        if (KeyState[(int)Keys.D])
            x += offset;
    {
        MouseEventArgs me = new MouseEventArgs(MouseButtons.Left, 0, x, y, 0);
        UpdateJoystickAndDriveCommand(me);
    }
}

private void btnStop_Click(object sender, EventArgs e)
{
    if (robot.connected)
    {
        robot.RunDemo(6);
    }

    //try remote send
    if (udpClientSender == null)
        return;
    if (udpClientSender.Client == null)
        return;
    if (udpClientSender.Client.Connected)
    {
        Byte[] sendBytes = Encoding.ASCII.GetBytes("home");
        udpClientSender.Send(sendBytes, sendBytes.Length);
        lastSentNetworkMessageTimestamp = System.Environment.TickCount;
    }
}

private void tbKeyBoardSpeed_TextChanged(object sender, EventArgs e)
{
    int v = 0;
    if (!Int32.TryParse(tbKeyBoardSpeed.Text, out v))
        return;
    if (v >= iRobotCreate.MaxMotorSpeed)
    {
        tbKeyBoardSpeed.Text = iRobotCreate.MaxMotorSpeed.ToString();
    }
    if (v < 0)
    {
        tbKeyBoardSpeed.Text = "0";
    }
}

private void tbRemotePassword_TextChanged(object sender, EventArgs e)
{
}

private void tbKeyBoardSpeed_KeyPress(object sender, KeyPressEventArgs e)
{
    if ((int)e.KeyChar == 8)
        return;
    if (!System.Text.RegularExpressions.Regex.IsMatch(e.KeyChar.ToString(),
"\\d+"))

```

```

        e.Handled = true;
    }

    private void btnKeyboardControl_Click(object sender, EventArgs e)
    {
        robot.StartInFullMode();
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
    }
    private void tbServerPassword_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((int)e.KeyChar == 8)
            return;
        if (!System.Text.RegularExpressions.Regex.IsMatch(e.KeyChar.ToString(),
"[\w!@#$$%^&*~+=()]+"))
            e.Handled = true;
    }

    private void tbRemotePassword_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((int)e.KeyChar == 8)
            return;
        if (!System.Text.RegularExpressions.Regex.IsMatch(e.KeyChar.ToString(),
"[\w!@#$$%^&*~+=()]+"))
            e.Handled = true;
    }

    private void tbRemotePort_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((int)e.KeyChar == 8)
            return;
        if (!System.Text.RegularExpressions.Regex.IsMatch(e.KeyChar.ToString(),
"\\d+"))
            e.Handled = true;
    }

    private void tbRemoteIP_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((int)e.KeyChar == 8)
            return;
        if (!System.Text.RegularExpressions.Regex.IsMatch(e.KeyChar.ToString(),
"\\\\d\\\\.\\\\.\\\\.+"))
            e.Handled = true;
    }

    private void tbServerPort_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((int)e.KeyChar == 8)
            return;
        if (!System.Text.RegularExpressions.Regex.IsMatch(e.KeyChar.ToString(),
"\\d+"))
            e.Handled = true;
    }
    }
}

```

Anexo 1.2. Clases del programa iCC

Anexo 1.2.1. iRobotCreate.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO.Ports;
using System.Threading;

namespace iRobot
{
    public class iRobotCreate
    {
        public enum OpCode
        {
            SoftReset = 7,
            Start = 128,
            Baud = 129,
            Control = 130,
            Safe = 131,
            Full = 132,
            Spot = 134,
            Cover = 135,
            Demo = 136,
            Drive = 137,
            LowSideDrivers = 138,
            LEDs = 139,
            Song = 140,
            Play = 141,
            Sensors = 142,
            CoverAndDock = 143,
            PWMLowSideDrivers = 144,
            DriveDirect = 145,
            DigitalOutputs = 147,
            Stream = 148,
            QueryList = 149,
            PauseResumeStream = 150,
            SendIR = 151,
            Script = 152,
            PlayScript = 153,
            ShowScript = 154,
            WaitTime = 155,
            WaitDistance = 156,
            WaitAngle = 157,
            WaitEvent = 158,
        }

        public enum SensorPacket
        {
            Group0 = 0,    //7-26
            Group1 = 1,    //7-16
            Group2 = 2,    //17-20
            Group3 = 3,    //21-26
            Group4 = 4,    //27-34
            Group5 = 5,    //35-42
        }
    }
}
```

```

Group6 = 6, //7-42
BumpsAndWheelDrops = 7,
Wall = 8,
CliffLeft = 9,
CliffFrontLeft = 10,
CliffFrontRight = 11,
CliffRight = 12,
VirtualWall = 13,
LowSideDriverAndWheelOverCurrents = 14,
Infrared = 17,
Buttons = 18,
Distance = 19,
Angle = 20,
ChargingState = 21,
Voltage = 22,
Current = 23,
BatteryTemperature = 24,
BatteryCharge = 25,
BatteryCapacity = 26,
WallSignal = 27,
CliffLeftSignal = 28,
CliffFrontLeftSignal = 29,
CliffFrontRightSignal = 30,
CliffRightSignal = 31,
CargoBayDigitalInputs = 32,
CargoBayAnalogSignal = 33,
ChargingSourcesAvailable = 34,
OIMode = 35,
SongNumber = 36,
SongPlaying = 37,
NumberOfStreamPackets = 38,
RequestedVelocity = 39,
RequestedRadius = 40,
RequestedRightVelocity = 41,
RequestedLeftVelocity = 42,
}

```

```

public struct SensorState
{
    //only 7-31 implemented
    public bool WheelDropCaster;
    public bool WheelDropLeft;
    public bool WheelDropRight;
    public bool BumpLeft;
    public bool BumpRight;
    public bool Wall;
    public bool CliffLeft;
    public bool CliffFrontLeft;
    public bool CliffFrontRight;
    public bool CliffRight;
    public bool VirtualWall;
    public bool OverCurrentLeftWheel;
    public bool OverCurrentRightWheel;
    public bool OverCurrentLD2;
    public bool OverCurrentLD0;
    public bool OverCurrentLD1;

    public byte IR;
}

```

```

        public bool ButtonPlay;
        public bool ButtonAdvance;
        public int Distance;
        public int Angle;
        public int ChargingState;
        public int Voltage;
        public int Current;
        public int BatteryTempurature;
        public int BatteryCharge;
        public int BatteryCapacity;
        public int WallSignal;
        public int CliffLeftSignal;
        public int CliffLeftFrontSignal;
        public int CliffRightFrontSignal;
        public int CliffRightSignal;
    }

    public delegate void SensorUpdateHandler(object sender, EventArgs e);
    public delegate void BumperChangedHandler(object sender, EventArgs e);
    public delegate void CliffDetectChangedHandler(object sender, EventArgs e);
    public delegate void WheelDropChangedHandler(object sender, EventArgs e);

    byte[] messageBuffer = new byte[64];
    byte[] recieveBuffer = new byte[1024];
    byte[] buffer = new byte[64];
    int recieveBufferReadIndex = 0;
    int recieveBufferWriteIndex = 0;
    int messageIndex = 0;

    int targetLeftWheelSpeed = 0;
    int targetRightWheelSpeed = 0;
    int lastSentLeftWheelSpeed = 0;
    int lastSentRightWheelSpeed = 0;

    Thread drivingThread;

    public int maxAcclerationPerSec = 2500;
    public const int MaxMotorSpeed = 500;
    public const int MaxAngularVelocity = 2000; //with weird forward exception of
32678
    public int updateIntervalMS = 50;

    public bool connected = false;
    SerialPort port = null;
    public SensorState sensorState = new SensorState();
    public SensorState prevSensorState = new SensorState();

    public event SensorUpdateHandler OnSensorUpdateRecieved;

    public event BumperChangedHandler OnBumperChanged;
    public event CliffDetectChangedHandler OnCliffDetectChanged;
    public event WheelDropChangedHandler OnWheelDropChanged;

    public iRobotCreate()
    {
    }
    public bool Connect(string portName)

```

```

{
    try
    {
        if (!connected)
        {
            port = new SerialPort();
            port.PortName = portName;
            port.BaudRate = 57600;
            port.DataBits = 8;
            port.DtrEnable = false;
            port.StopBits = StopBits.One;
            port.Handshake = Handshake.None;
            port.Parity = Parity.None;
            port.RtsEnable = false;
            port.Close();
            port.Open();
            port.DataReceived += new
SerialDataReceivedEventHandler(port_DataReceived);
            connected = true;
            Thread.Sleep(50);
        }
    }
    catch (System.Exception ex)
    {
        Console.Out.WriteLine(ex);
        return false;
    }
    return true;
}

public void Disconnect()
{
    if(!connected)
        return;

    SoftReset();
    if (port != null)
    {
        if (port.IsOpen)
        {
            //          port.DataReceived -= new
SerialDataReceivedEventHandler(port_DataReceived);
            //causes hang?
            //          port.Close();
        }
    }
    connected = false;
}

public void QueueCommand(OpCode op)
{
    if (!connected)
        return;
    messageBuffer[messageIndex] = (byte)op;
    messageIndex++;
}

public void QueueCommand(OpCode op, byte parameter)

```

```

    {
        if (!connected)
            return;
        messageBuffer[messageIndex] = (byte)op;
        messageIndex++;
        messageBuffer[messageIndex] = parameter;
        messageIndex++;
    }
    public void QueueCommand(OpCode op, byte parameter1, byte parameter2)
    {
        if (!connected)
            return;
        messageBuffer[messageIndex] = (byte)op;
        messageIndex++;
        messageBuffer[messageIndex] = parameter1;
        messageIndex++;
        messageBuffer[messageIndex] = parameter2;
        messageIndex++;
    }
    public void QueueCommand(OpCode op, byte parameter1, byte parameter2, byte
parameter3)
    {
        if (!connected)
            return;
        messageBuffer[messageIndex] = (byte)op;
        messageIndex++;
        messageBuffer[messageIndex] = parameter1;
        messageIndex++;
        messageBuffer[messageIndex] = parameter2;
        messageIndex++;
        messageBuffer[messageIndex] = parameter3;
        messageIndex++;
    }

    public void QueueCommandParameter(OpCode op, int parameter)
    {
        if (!connected)
            return;
        messageBuffer[messageIndex] = (byte)op;
        messageIndex++;
        messageBuffer[messageIndex] = (byte)(parameter >> 8);
        messageIndex++;
        messageBuffer[messageIndex] = (byte)(parameter & 255);
        messageIndex++;
    }

    public void QueueCommandTwoParameters(OpCode op, int p1, int p2)
    {
        if (!connected)
            return;

        messageBuffer[messageIndex] = (byte)op;
        messageIndex++;
        messageBuffer[messageIndex] = (byte)(p1 >> 8);
        messageIndex++;
        messageBuffer[messageIndex] = (byte)(p1 & 255);
        messageIndex++;
        messageBuffer[messageIndex] = (byte)(p2 >> 8);

```

```

        messageIndex++;
        messageBuffer[messageIndex] = (byte)(p2 & 255);
        messageIndex++;
    }

    public void CoverAndDock()
    {
        QueueCommand(OpCodes.CoverAndDock);
        SendCommand();
    }

    public void RunDemo(byte demo)
    {
        QueueCommand(OpCodes.Demo, demo);
        SendCommand();
    }

    public void QuerySensor(SensorPacket sensorID)
    {
        QueueCommandParameter(OpCodes.Sensors, (int)SensorPacket.BatteryCharge);
        SendCommand();
    }

    public void StopStreaming()
    {
        QueueCommandParameter(OpCodes.PauseResumeStream, 0);
        SendCommand();
    }

    public void StartSensorStreaming()
    {
        QueueCommand(OpCodes.Stream, (byte)1, (byte)SensorPacket.Group0);
        SendCommand();
    }

    void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        //read as much as possible to clear the communication buffer
        int bytes = port.Read(buffer, 0,buffer.Length);
        //    for(int i =0; i < bytes; i++)
        //        Console.Write((char)buffer[i]);
        //    Console.WriteLine("");

        //copy to main buffer
        for (int i = 0; i < bytes; i++)
        {
            recieveBuffer[(recieveBufferWriteIndex++) % recieveBuffer.Length] =
buffer[i];
        }

        //if we have enough data parse
        int group0messageSize = 30;
        while (recieveBufferReadIndex < recieveBufferWriteIndex-group0messageSize)
        {
            if(ParseStreamingGroup0(group0messageSize, recieveBuffer,
recieveBufferReadIndex))
            {

```

```

//          for(int i=recieveBufferReadIndex; i <
recieveBufferReadIndex+group0messageSize; i++)
//          Console.Out.Write((int)recieveBuffer[i %
recieveBuffer.Length] + " ");
//          Console.Out.WriteLine(" message " + sensorState.ChargingState +
" " + sensorState.BatteryCharge + " " + sensorState.BatteryCapacity + " " +
Environment.TickCount);

        recieveBufferReadIndex += group0messageSize;

    }
    else
    {
        recieveBufferReadIndex++;
    }
}

//shift out the parsed data
if (recieveBufferReadIndex > 0)
{
    int count = recieveBufferWriteIndex - recieveBufferReadIndex;
    for (int i = 0; i < count; i++)
        recieveBuffer[i] = recieveBuffer[recieveBufferReadIndex++];
    recieveBufferWriteIndex = count;
    recieveBufferReadIndex = 0;
}
}

bool PassesCheckSum(int bytes, byte[] buffer, int offset)
{
    int sum = 0;
    for(int i = offset; i < offset+bytes; i++)
        sum += buffer[i];
    return ((sum & 0xff)==0);
}

bool ParseStreamingGroup1(int bytes, byte[] buffer, int offset)
{
    if (!PassesCheckSum(bytes, buffer, offset))
    {
        Console.WriteLine("Recieved data: failed checksum");
        return false;
    }

    if(buffer[offset] != 19) //streaming header
    {
        Console.WriteLine("Recieved data: not streaming packet " +
buffer[offset]);
        return false;
    }

    if (buffer[offset+1] != 11) //streaming header
    {
        Console.WriteLine("Recieved data: wrong message length");
        return false;
    }
}

```

```

if (buffer[offset+ 2] != (byte)SensorPacket.Group1)
{
    Console.WriteLine("Recieved data: not yet supported packet type");
    return false;
}

int sensorDataIndex = offset + 3;

prevSensorState = sensorState;

sensorState.BumpRight = (buffer[sensorDataIndex] & 1) > 0;
sensorState.BumpLeft = (buffer[sensorDataIndex] & 2) > 0;
sensorState.WheelDropLeft = (buffer[sensorDataIndex] & 4) > 0;
sensorState.WheelDropRight = (buffer[sensorDataIndex] & 8) > 0;
sensorState.WheelDropCaster = (buffer[sensorDataIndex] & 16) > 0;

if ((prevSensorState.BumpLeft != sensorState.BumpLeft) ||
    (prevSensorState.BumpRight != sensorState.BumpRight))
{
    if(OnBumperChanged!= null)
        OnBumperChanged(this, new EventArgs());
}

if ((prevSensorState.WheelDropLeft != sensorState.WheelDropLeft) ||
    (prevSensorState.WheelDropRight != sensorState.WheelDropRight) ||
    (prevSensorState.WheelDropCaster != sensorState.WheelDropCaster))
{
    if (OnWheelDropChanged != null)
        OnWheelDropChanged(this, new EventArgs());
}

sensorState.Wall = (buffer[sensorDataIndex+1] > 0);
sensorState.CliffLeft = (buffer[sensorDataIndex + 2] > 0);
sensorState.CliffFrontLeft = (buffer[sensorDataIndex + 3] > 0);
sensorState.CliffFrontRight = (buffer[sensorDataIndex + 4] > 0);
sensorState.CliffRight = (buffer[sensorDataIndex + 5] > 0);

if ((prevSensorState.CliffLeft != sensorState.CliffLeft) ||
    (prevSensorState.CliffFrontLeft != sensorState.CliffFrontLeft) ||
    (prevSensorState.CliffFrontRight != sensorState.CliffFrontRight) ||
    (prevSensorState.CliffRight != sensorState.CliffRight))
{
    if (OnCliffDetectChanged != null)
        OnCliffDetectChanged(this, new EventArgs());
}

sensorState.VirtualWall = (buffer[sensorDataIndex + 6] > 0);

sensorState.OverCurrentLD1 = (buffer[sensorDataIndex + 7] & 1) > 0;
sensorState.OverCurrentLD0 = (buffer[sensorDataIndex + 7] & 2) > 0;
sensorState.OverCurrentLD2 = (buffer[sensorDataIndex + 7] & 4) > 0;
sensorState.OverCurrentLeftWheel = (buffer[sensorDataIndex + 7] & 8) > 0;
sensorState.OverCurrentRightWheel = (buffer[sensorDataIndex + 7] & 16) > 0;

```

```

        if (connected)
        {
            if (OnSensorUpdateRecieved != null)
            {
                OnSensorUpdateRecieved(this, new EventArgs());
            }
        }

        return true;
    }

    bool ParseStreamingGroup0(int bytes, byte[] buffer, int offset)
    {
        if (buffer[offset] != 19) //streaming header
        {
            // Console.WriteLine("Recieved data: not streaming
packet " + buffer[offset]);
            return false;
        }

        if (buffer[offset + 1] != 27) //streaming header
        {
            // Console.WriteLine("Recieved data: wrong message length");
            return false;
        }

        if (!PassesChecksum(bytes, buffer, offset))
        {
            // Console.WriteLine("Recieved data: failed checksum");
            return false;
        }

        if (buffer[offset + 2] != (byte)SensorPacket.Group0)
        {
            // Console.WriteLine("Recieved data: not yet supported packet
type");
            return false;
        }

        // 19 27 0 1 0 0 0 0 0 0 0 0 255 0 0 0 0 0 64 120 255 158 23 10 57 10 142
139 ;

        int sensorDataIndex = offset + 3;

        prevSensorState = sensorState;

        sensorState.BumpRight = (buffer[sensorDataIndex] & 1) > 0;
        sensorState.BumpLeft = (buffer[sensorDataIndex] & 2) > 0;
        sensorState.WheelDropLeft = (buffer[sensorDataIndex] & 4) > 0;
        sensorState.WheelDropRight = (buffer[sensorDataIndex] & 8) > 0;
        sensorState.WheelDropCaster = (buffer[sensorDataIndex] & 16) > 0;

        if ((prevSensorState.BumpLeft != sensorState.BumpLeft) ||
            (prevSensorState.BumpRight != sensorState.BumpRight))
        {
            if (OnBumperChanged != null)
            {
                OnBumperChanged(this, new EventArgs());
            }
        }
    }

```

```

if ((prevSensorState.WheelDropLeft != sensorState.WheelDropLeft) ||
    (prevSensorState.WheelDropRight != sensorState.WheelDropRight) ||
    (prevSensorState.WheelDropCaster != sensorState.WheelDropCaster))
{
    if (OnWheelDropChanged != null)
        OnWheelDropChanged(this, new EventArgs());
}

sensorState.Wall = (buffer[sensorDataIndex + 1] > 0);
sensorState.CliffLeft = (buffer[sensorDataIndex + 2] > 0);
sensorState.CliffFrontLeft = (buffer[sensorDataIndex + 3] > 0);
sensorState.CliffFrontRight = (buffer[sensorDataIndex + 4] > 0);
sensorState.CliffRight = (buffer[sensorDataIndex + 5] > 0);

if ((prevSensorState.CliffLeft != sensorState.CliffLeft) ||
    (prevSensorState.CliffFrontLeft != sensorState.CliffFrontLeft) ||
    (prevSensorState.CliffFrontRight != sensorState.CliffFrontRight) ||
    (prevSensorState.CliffRight != sensorState.CliffRight))
{
    if (OnCliffDetectChanged != null)
        OnCliffDetectChanged(this, new EventArgs());
}

sensorState.VirtualWall = (buffer[sensorDataIndex + 6] > 0);

sensorState.OverCurrentLD1 = (buffer[sensorDataIndex + 7] & 1) > 0;
sensorState.OverCurrentLD0 = (buffer[sensorDataIndex + 7] & 2) > 0;
sensorState.OverCurrentLD2 = (buffer[sensorDataIndex + 7] & 4) > 0;
sensorState.OverCurrentLeftWheel = (buffer[sensorDataIndex + 7] & 8) > 0;
sensorState.OverCurrentRightWheel = (buffer[sensorDataIndex + 7] & 16) > 0;

sensorState.IR = buffer[sensorDataIndex + 10];
sensorState.ButtonPlay = (buffer[sensorDataIndex + 11] & 1) > 0;
sensorState.ButtonAdvance = (buffer[sensorDataIndex + 11] & 4) > 0;
sensorState.Distance = (short)((buffer[sensorDataIndex + 12] << 8) +
buffer[sensorDataIndex + 13]);
sensorState.Angle = (short)((buffer[sensorDataIndex + 14] << 8) +
buffer[sensorDataIndex + 15]);
sensorState.ChargingState = buffer[sensorDataIndex + 16];
sensorState.Voltage = (buffer[sensorDataIndex + 17] << 8) +
buffer[sensorDataIndex + 18];
sensorState.Current = (short)((buffer[sensorDataIndex + 19] << 8) +
buffer[sensorDataIndex + 20]);
sensorState.BatteryTemperature = buffer[sensorDataIndex + 21];
if (buffer[sensorDataIndex + 22] == 255)
    sensorState.BatteryCharge = buffer[sensorDataIndex + 23];
else
    sensorState.BatteryCharge = (buffer[sensorDataIndex + 22] << 8) +
buffer[sensorDataIndex + 23];
sensorState.BatteryCapacity = (buffer[sensorDataIndex + 24] << 8) +
buffer[sensorDataIndex + 25];
if (connected)
{
    if (OnSensorUpdateRecieved != null)
    {

```

```

        OnSensorUpdateRecieved(this, new EventArgs());
    }

}

return true;
}

public void StartInPassiveMode()
{
    QueueCommand(OpCodes.Start);
    SendCommand();
}

public void StartInSafeMode()
{
    QueueCommand(OpCodes.Start);
    QueueCommand(OpCodes.Safe);
    SendCommand();
}

public void StartInFullMode()
{
    QueueCommand(OpCodes.Start);
    QueueCommand(OpCodes.Full);
    SendCommand();
}

public void SoftReset()
{
    if (drivingThread != null)
    {
        drivingThread.Abort();
        Thread.Sleep(100);
        drivingThread = null;
    }

    QueueCommand(OpCodes.SoftReset);
    SendCommand();
    Thread.Sleep(2500);
}

public void Drive(int velocity, int angle)
{
    if (velocity > MaxMotorSpeed) velocity = MaxMotorSpeed;
    if (velocity < -MaxMotorSpeed) velocity = -MaxMotorSpeed;
    QueueCommandTwoParameters(OpCodes.Drive, velocity, angle);
    SendCommand();
}

public void DriveDirect(int left, int right)
{
    if (!connected)
        return;
    if (left > MaxMotorSpeed) left = MaxMotorSpeed;
    if (left < -MaxMotorSpeed) left = -MaxMotorSpeed;
    if (right > MaxMotorSpeed) right = MaxMotorSpeed;
    if (right < -MaxMotorSpeed) right = -MaxMotorSpeed;
    targetLeftWheelSpeed = left;
    targetRightWheelSpeed = right;
}

```

```

        if (drivingThread == null)
        {
            //too fast, so create a thread to moderate acceleration
            drivingThread = new Thread(new ThreadStart(DriveThreadCallback));
            drivingThread.Start();
        }

        return;
    }

    void DriveThreadCallback()
    {
        int maxAccelPerInterval = maxAccelerationPerSec * updateIntervalMS / 1000;

        while (connected)
        {
            if (!port.IsOpen)
                return;

            //Console.Out.WriteLine(lastSentLeftWheelSpeed + "," +
            lastSentRightWheelSpeed + " " + targetLeftWheelSpeed + "," + targetRightWheelSpeed);

            float targetForwardMag = targetLeftWheelSpeed + targetRightWheelSpeed;
            float lastForwardMag = lastSentLeftWheelSpeed +
            lastSentRightWheelSpeed;
            float deltaForwardMag = Math.Abs(targetForwardMag - lastForwardMag);
            int currLeftWheelSpeed = 0;
            int currRightWheelSpeed = 0;

            if (deltaForwardMag > maxAccelPerInterval)
            {
                float overAccel = deltaForwardMag / maxAccelPerInterval;
                int leftWheelAccel = targetLeftWheelSpeed - lastSentLeftWheelSpeed;
                int rightWheelAccel = targetRightWheelSpeed -
            lastSentRightWheelSpeed;

                currLeftWheelSpeed = lastSentLeftWheelSpeed + (int)(leftWheelAccel
            / overAccel);
                currRightWheelSpeed = lastSentRightWheelSpeed +
            (int)(rightWheelAccel / overAccel);
            }
            else
            {
                currLeftWheelSpeed = targetLeftWheelSpeed;
                currRightWheelSpeed = targetRightWheelSpeed;
            }

            if ((lastSentLeftWheelSpeed != currLeftWheelSpeed) ||
            (lastSentRightWheelSpeed != currRightWheelSpeed))
            {
                QueueCommandTwoParameters(OpCodes.DriveDirect, currLeftWheelSpeed,
            currRightWheelSpeed);
                SendCommand();
            }

            lastSentLeftWheelSpeed = currLeftWheelSpeed;
            lastSentRightWheelSpeed = currRightWheelSpeed;
        }
    }

```

```

        Thread.Sleep(updateIntervalMS);
    }
    //changing slow enough, just relay
    lastSentLeftWheelSpeed = targetLeftWheelSpeed;
    lastSentRightWheelSpeed = targetRightWheelSpeed;
    QueueCommandTwoParameters(OpCodes.DriveDirect, lastSentLeftWheelSpeed,
lastSentRightWheelSpeed);
    SendCommand();
}

public void WaitForDistance(int distance)
{
    QueueCommandParameter(OpCodes.WaitDistance, distance);
    SendCommand();
}
public void WaitForAngle(int angle)
{
    QueueCommandParameter(OpCodes.WaitAngle, angle);
    SendCommand();
}

public void SendCommand()
{
    if (port == null)
        return;
    if (!port.IsOpen)
        return;
    try
    {
        port.Write(messageBuffer, 0, messageIndex);
        //        for (int i = 0; i < messageIndex; i++)
        //            Console.Out.Write((int)messageBuffer[i] + " ");
        //            Console.Out.WriteLine(" sent");
    }
    catch (Exception x)
    {
        Console.Out.WriteLine(x);
    }
    messageIndex = 0;
    Thread.Sleep(5);
}
}
}

```

Anexo 1.2.2. Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace iRobotCreateControl
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Anexo 1.2.3. UPnP.cs

```

using System;
using System.Text;
using System.Diagnostics;
using System.Net.Sockets;
using System.Net;
using System.Net.NetworkInformation;

namespace iRobotCreateControl
{
    public class UPnP
    {
        public enum Protocol
        {
            UDP,
            TCP,
        };

        public UPnP()
        {
        }

        public static bool OpenFirewallPort(string serviceName, Protocol p, int port)
        {
            NetworkInterface[] nics =
                NetworkInterface.GetAllNetworkInterfaces();
            try
            {
                //for each nic in computer...
                foreach (NetworkInterface nic in nics)
                {
                    if (nic.OperationalStatus != OperationalStatus.Up)
                        continue;

                    foreach (UnicastIPAddressInformation addrInfo in
                        nic.GetIPProperties().UnicastAddresses)
                    {
                        if (addrInfo.Address.AddressFamily !=
                            AddressFamily.InterNetwork)
                            continue;
                        string machineIP = addrInfo.Address.ToString();
                        //send msg to each gateway configured on this nic
                        foreach (GatewayIPAddressInformation gwInfo in
                            nic.GetIPProperties().GatewayAddresses)
                        {
                            string gateWayAddr = gwInfo.Address.ToString();
                            Console.Out.WriteLine("trying to add port forwarding {0}
machineIP: {1}\tgateway address: {2}",port, machineIP, gateWayAddr);
                            OpenFirewallPort(serviceName,p ,machineIP,
                                gwInfo.Address.ToString(), port);
                        } //each gateway
                    } //each ip
                } //each nic
            } //try
            catch (Exception x)
            {
                Console.Out.WriteLine(x);
            }
        }
    }
}

```

```

        return false;
    }
    return true;
} //open firewall port

public static bool CloseFirewallPort(Protocol p, int port)
{
    NetworkInterface[] nics =
        NetworkInterface.GetAllNetworkInterfaces();
    try
    {
        //for each nic in computer...
        foreach (NetworkInterface nic in nics)
        {
            if (nic.OperationalStatus != OperationalStatus.Up)
                continue;

            //send msg to each gateway configured on this nic
            foreach (GatewayIPAddressInformation gwInfo in
nic.GetIPProperties().GatewayAddresses)
            {
                string gateWayAddr = gwInfo.Address.ToString();
                Console.Out.WriteLine("trying to close port forwarding
{0}\tgateway address: {1}", port, gateWayAddr);
                CloseFirewallPort(p, gwInfo.Address.ToString(), port);
            } //each gateay
        } //each nic
    } //try
    catch (Exception x)
    {
        Console.Out.WriteLine(x);
        return false;
    }
    return true;
} //open firewall port

public static void OpenFirewallPort(string serviceName, Protocol p, string
machineIP, string firewallIP, int openPort)
{
    int gateWayPort = 0;
    string svc = getServicesFromDevice(firewallIP, ref gateWayPort);

    AddPortMapping(svc, "urn:schemas-upnp-org:service:WANIPConnection:1",
serviceName, p, machineIP, firewallIP, gateWayPort, openPort);
    // AddPortMapping(svc, "urn:schemas-upnp-org:service:WANPPPConnection:1",
serviceName, Protocol.UDP, machineIP, firewallIP, gateWayPort, openPort);
}

public static void CloseFirewallPort(Protocol p, string firewallIP, int
openPort)
{
    int gateWayPort = 0;
    string svc = getServicesFromDevice(firewallIP, ref gateWayPort);

```

```

        DeletePortMapping(svc,
"urn:schemas-upnp-org:service:WANIPConnection:1",p, firewallIP, gateWayPort,
openPort);
//        DeletePortMapping(svc,
"urn:schemas-upnp-org:service:WANPPPConnection:1", serviceName, Protocol.UDP,
machineIP, firewallIP, gateWayPort, openPort);
    }

    private static string getServicesFromDevice(string firewallIP, ref int
gatewayPort)
    {
        //To send a broadcast and get responses from all, send to 239.255.255.250
        string queryResponse = "";

        try
        {
            string query = "M-SEARCH * HTTP/1.1\r\n" +
"Host:" + firewallIP + ":1900\r\n" +
"ST:upnp:rootdevice\r\n" +
"Man:\"ssdp:discover\"\r\n" +
"MX:3\r\n" +
"\r\n" +
"\r\n";

            //use sockets instead of UdpClient so we can set a timeout easier
            Socket client = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
            IPEndPoint endPoint = new
IPEndPoint(IPAddress.Parse(firewallIP), 1900);

            //1.5 second timeout because firewall should be on same segment (fast)
            client.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReceiveTimeout, 1500);

            byte[] q = Encoding.ASCII.GetBytes(query);
            client.SendTo(q, q.Length, SocketFlags.None, endPoint);
            IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
            EndPoint senderEP = (EndPoint)sender;

            byte[] data = new byte[1024];
            int recv = client.ReceiveFrom(data, ref senderEP);
            queryResponse = Encoding.ASCII.GetString(data);
        }
        catch { }

        if(queryResponse.Length == 0)
            return "";

        /* QueryResult is something like this:
        *
        HTTP/1.1 200 OK
        Cache-Control:max-age=60
        Location:http://10.10.10.1:80/upnp/service/des_ppp.xml
        Server:NT/5.0 UPnP/1.0
        ST:upnp:rootdevice
        EXT:

```

```

USN:uuid:upnp-InternetGatewayDevice-1_0-00095bd945a2::upnp:rootdevice
*/

string location = "";
string[] parts = queryResponse.Split(new string[] {
System.Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);
foreach (string part in parts)
{
    if (part.ToLower().StartsWith("location"))
    {
        location = part.Substring(part.IndexOf(':') + 1);
        break;
    }
}

if (location.Length == 0)
    return "";

//then using the location url, we get more information:
string gatewayPortStr = location.Substring(location.IndexOf(firewallIP) +
firewallIP.Length+1);
gatewayPortStr = gatewayPortStr.Substring(0, gatewayPortStr.IndexOf('/'));
gatewayPort = int.Parse(gatewayPortStr);

System.Net.WebClient webClient = new WebClient();
try
{
    string ret = webClient.DownloadString(location);
    return ret; //return services
}
catch (System.Exception ex)
{
    Debug.WriteLine(ex.Message);
}
finally
{
    webClient.Dispose();
}
return "";
}

private static void DeletePortMapping(string services, string serviceType,
Protocol p, string firewallIP, int gatewayPort, int portToForward)
{
    if (services.Length == 0)
        return;

    int svcIndex = services.IndexOf(serviceType);
    if (svcIndex == -1)
        return;
    string controlUrl = services.Substring(svcIndex);
    string tag1 = "<controlURL>";
    string tag2 = "</controlURL>";
    controlUrl = controlUrl.Substring(controlUrl.IndexOf(tag1) + tag1.Length);
    controlUrl = controlUrl.Substring(0, controlUrl.IndexOf(tag2));
}

```

```

string protocolStr = (p == Protocol.UDP)? "UDP" : "TCP";
string soapBody = "<s:Envelope " +
    "xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/ \" " +
    "s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/ \">" +
    "<s:Body>" +
    "<u:DeletePortMapping xmlns:u=\"\" + serviceType + "\">" +
    "<NewRemoteHost></NewRemoteHost>" +
    "<NewExternalPort>" + portToForward.ToString() +
    "</NewExternalPort>" +
    "<NewProtocol>" + protocolStr + "</NewProtocol>" +
    "</u:DeletePortMapping>" +
    "</s:Body>" +
    "</s:Envelope>";

byte[] body = System.Text.UTF8Encoding.ASCII.GetBytes(soapBody);

string url = "http://" + firewallIP + ":" +
    gatewayPort.ToString() + controlUrl;

Console.Out.WriteLine(url);

System.Net.WebRequest wr =
    System.Net.WebRequest.Create(url); //+ controlUrl);
wr.Method = "POST";
wr.Headers.Add("SOAPAction", "\" + serviceType + "#AddPortMapping\"");
wr.ContentType = "text/xml; charset=\"utf-8\"";
wr.ContentLength = body.Length;

System.IO.Stream stream = wr.GetRequestStream();
stream.Write(body, 0, body.Length);
stream.Flush();
stream.Close();

WebResponse wres = wr.GetResponse();
System.IO.StreamReader sr = new
    System.IO.StreamReader(wres.GetResponseStream());
string ret = sr.ReadToEnd();
sr.Close();

Debug.WriteLine("Port Mapping Deleted:" + portToForward.ToString());
}

private static bool AddPortMapping(string services, string serviceType, string
serviceName, Protocol p, string machineIP, string firewallIP, int gatewayPort, int
portToForward)
{
    if (services.Length == 0)
        return false;

    int svcIndex = services.IndexOf(serviceType);
    if (svcIndex == -1)
        return false;
    string controlUrl = services.Substring(svcIndex);
    string tag1 = "<controlURL>";
    string tag2 = "</controlURL>";
    controlUrl = controlUrl.Substring(controlUrl.IndexOf(tag1) + tag1.Length);
    controlUrl = controlUrl.Substring(0, controlUrl.IndexOf(tag2));

```

```

string protocolStr = (p == Protocol.UDP) ? "UDP" : "TCP";
string soapBody = "<s:Envelope " +
    "xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/ \" " +
    "s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/ \">" +
    "<s:Body>" +
    "<u:AddPortMapping xmlns:u=\"\" + serviceType + "\">" +
    "<NewRemoteHost></NewRemoteHost>" +
    "<NewExternalPort>" + portToForward.ToString() +
    "</NewExternalPort>" +
    "<NewProtocol>" + protocolStr + "</NewProtocol>" +
    "<NewInternalPort>" + portToForward.ToString() +
    "</NewInternalPort>" +
    "<NewInternalClient>" + machineIP +
    "</NewInternalClient>" +
    "<NewEnabled>1</NewEnabled>" +
    "<NewPortMappingDescription>" + serviceName +
    "</NewPortMappingDescription>" +
    "<NewLeaseDuration>0</NewLeaseDuration>" +
    "</u:AddPortMapping>" +
    "</s:Body>" +
    "</s:Envelope>";

byte[] body = System.Text.UTF8Encoding.ASCII.GetBytes(soapBody);

string url = "http://" + firewallIP + ":" +
    gatewayPort.ToString() + controlUrl;

Console.Out.WriteLine(url);

System.Net.WebRequest wr =
    System.Net.WebRequest.Create(url); //+ controlUrl);
wr.Method = "POST";
wr.Headers.Add("SOAPAction", "\"" + serviceType + "#AddPortMapping\"");
wr.ContentType = "text/xml; charset=\"utf-8\"";
wr.ContentLength = body.Length;

System.IO.Stream stream = wr.GetRequestStream();
stream.Write(body, 0, body.Length);
stream.Flush();
stream.Close();

WebResponse wres = wr.GetResponse();
System.IO.StreamReader sr = new
    System.IO.StreamReader(wres.GetResponseStream());
string ret = sr.ReadToEnd();
sr.Close();

Debug.WriteLine("Port Mapping Added:" + portToForward.ToString());
return true;
    }
} //upnp class
}

```

Anexo 2. Manual de uso: Programa “iRobot Create Control”

Programa “iRobot Create Control”

Manual de uso

1) INSTALACIÓN DEL PROGRAMA

Este programa no requiere instalación alguna, para abrirlo se debe descomprimir el archivo “SoftwareiCC.zip”. Una vez terminado, ejecutar la aplicación iRobotCreateControl.exe incluida dentro del Zip y el programa se iniciará.

2) NAVEGACIÓN DEL ROBOT DESDE ORDENADOR LOCAL

Para usar el programa para navegar el robot, se debe vincular el mismo al ordenador desde el cual será comandado. Dependiendo del medio de conexión que se disponga, se podrán seguir las instrucciones brindadas en este trabajo en caso de disponer de la tecnología *bluetooth*. En caso de disponer de un cable serial para la conexión, se debe enchufar en el puerto serial del robot de un lado, y en un puerto serial del ordenador del otro lado.

Navegación: Seguir estos pasos.

- Una vez vinculado el robot al ordenador, este último habrá asignado un número de puerto serie a dicha vinculación. Conociendo ya el puerto correspondiente, iniciar el Software iCC como se indica en el **punto 1**.
- Colocar el número de puerto serie en la casilla llamada “*Serial Port*”, dentro de la pestaña **Local Robot** del programa.
- Presionar el botón “*Connect to Local Robot*”. Si la conexión se da exitosamente se obtendrá control del robot.
- Presionar el botón “*WASD Keyboard Control*” y luego de esto utilizar las teclas W, A, S y D del teclado para mover el robot:
 - S: Avanzar
 - W: Retroceder
 - A: Girar hacia la izquierda
 - D: Girar hacia la derecha
- Para desconectar la navegación simplemente cierre el programa.

3) CONEXIÓN ENTRE ORDENADORES

3.1) Conexión local: Para esto necesitará dos ordenadores que estén conectados a una misma red local de Internet. En ambos se debe iniciar el programa iCC y seguir estos pasos para cada ordenador.

Ordenador B: Es el ordenador al cual se conecta el robot según la tecnología que se disponga.

- Iniciar el programa iCC y colocar en la casilla *Serial Port*, el número de puerto serie asignado a la conexión con el robot. Definir también una contraseña (*Password*), y el puerto a utilizar (Se recomienda dejar el que viene por defecto: *Port: 25000*).
- Presionar el botón “*Enable Network Control*” y esperar a que en la terminal aparezca el mensaje: “*Listening on port 25000*” (o el número de puerto que hayan utilizado).

Ordenador R: Aquél que se vinculará al otro ordenador para manejar el robot.

- Iniciar el programa iCC y agregar en la casilla *Serial Port* el mismo número que ingresó en el ordenador B.
- Dirigirse a la pestaña **Remote Robot** del programa e ingresar los datos de conexión definidos en el ordenador B: la misma contraseña, el número de puerto y en la casilla *IP Address* ingresar la dirección IP que figura en el ordenador B, en la pestaña **Local Robot**, donde dice “*Local IP Address*”.
- Presionar el botón “*Connect to Remote Robot*”. Si la conexión es exitosa la terminal de mensajes lo indicará en ambos ordenadores con un mensaje de confirmación.
- Presionar el botón “*WASD Keyboard Control*” y comenzar a navegar el robot con las mismas instrucciones dadas en el **punto 2**.
- Para terminar la conexión puede presionar el botón “*Disable Network Control*” en el ordenador B, o simplemente cierre los programas en ambos ordenadores.

3.2) Conexión remota: Para esta conexión necesitará también dos ordenadores, solo que en éste caso deben estar conectados a Internet desde diferentes redes. Además se necesitará habilitar en el router de cada red, el puerto que utilizará para la comunicación entre los ordenadores. Esto puede hacerse desde la sección de *Port Forwarding* de cada router. Si desconoce cómo hacer esto puede buscar en Internet, o pedirle a algún conocido que configure esto por ustedes.

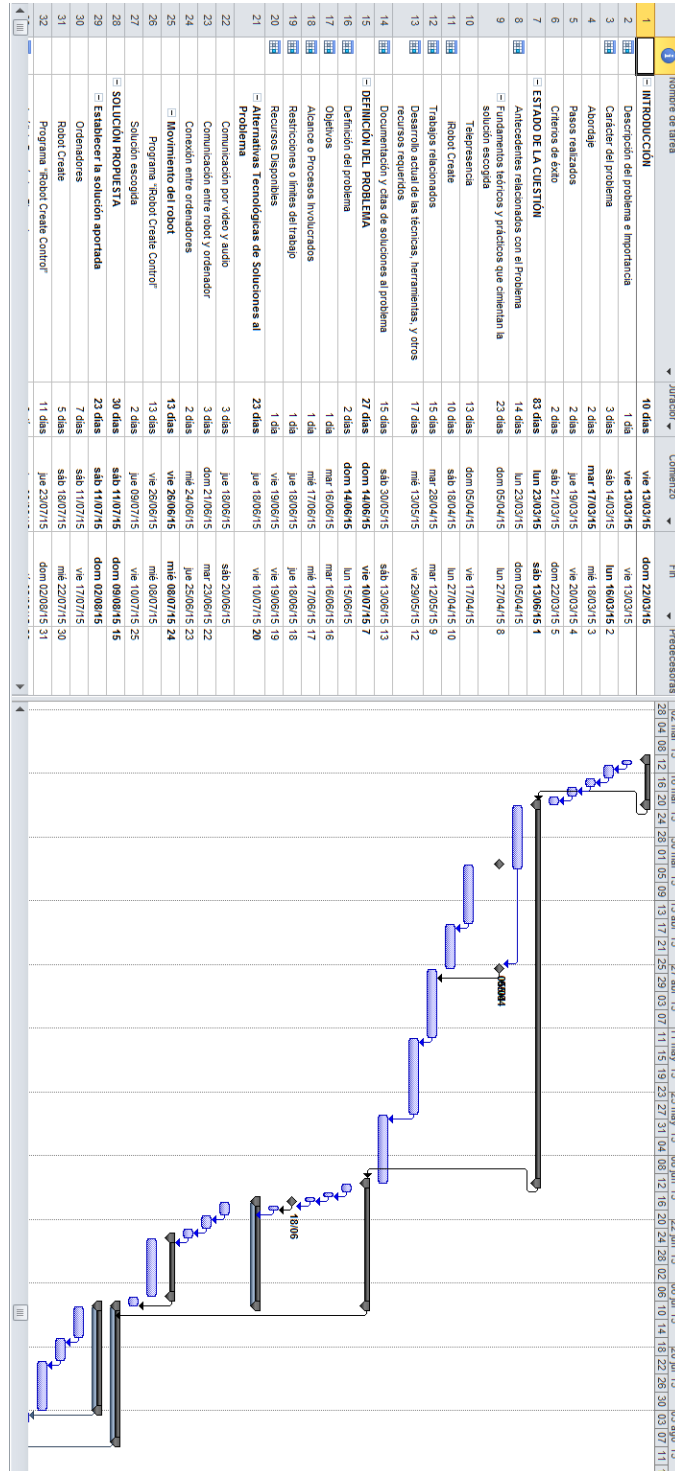
Ordenador B: Realizar los mismos pasos que se detallan anteriormente para la conexión local. Una vez que en la terminal figura el mensaje “*Listening on port ###*” seguir con el ordenador R.

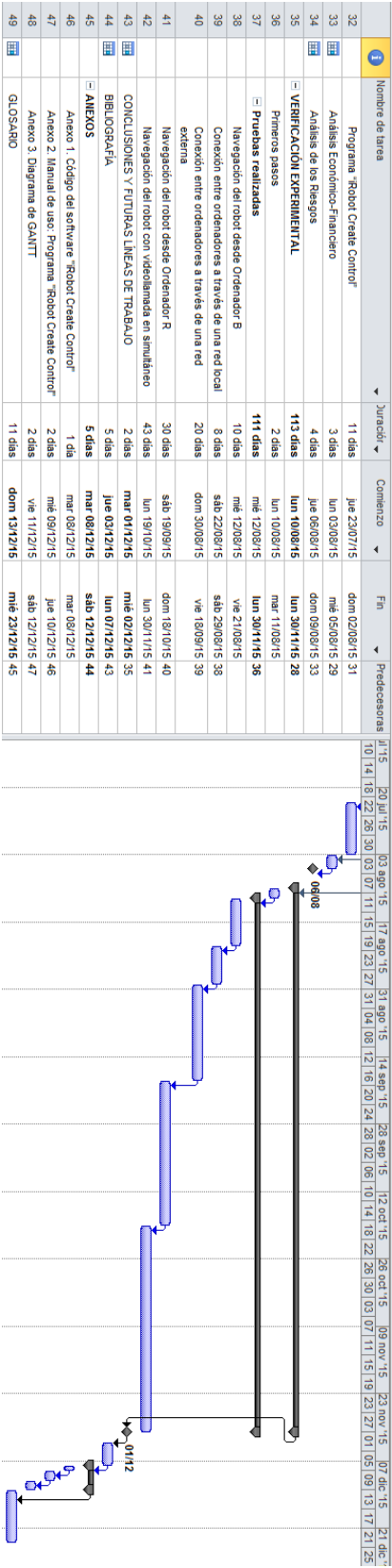
Ordenador R: También se deben seguir los mismos pasos que se describieron anteriormente, sólo cambia el hecho de que la dirección IP a utilizar, será aquella que en la pestaña **Local Robot** del ordenador B se indica con el mensaje: “*External IP Address*”. Una vez conectado, seguir las instrucciones del **paso 2** para navegar el robot.

Se recomienda utilizar Skype, para entablar una videoconferencia entre ambos ordenadores, y así obtener imagen y sonido de ambos lugares.

Anexo 3. Diagrama de GANTT

Las siguientes dos imágenes muestran el diagrama de GANTT con la distribución del tiempo en las tareas de este proyecto.





8. GLOSARIO

B

BAM

Siglas en inglés para *Adaptador del Módulo Bluetooth* 21, 23

L

LAN

Local Area Network 40

M

MRDS

Siglas de la herramienta *Microsoft Robotis Developer Studio* 77

N

NAT

Siglas del concepto anglosajón para *Network Address Translation*. 40

P

PF

Abreviación de *Port Forwarding* 88

R

RDSI

Red Digital de Servicios Integrados 18

S

STEM

Abreviación de las siglas en inglés para *Science, Technology, Engineering and Mathematics education* 20

U

UPnP

Abreviación para *Universal Plug and Play* 38

V

VS2010

Herramienta *Microsoft Visual Studio 2010* 68