UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Optimización y estructuración de proyecto de código abierto *TJBot* con fines educativos

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Lizarraga Velarde Jair Axel

ASESOR DE INFORME

Ing. Carlos Saucedo Maciel

Ciudad Universitaria, Cd. Mx., 2020

Agradecimientos

A Lourdes, por hacer de mí un hombre sensato, inteligente y mostrarme el amor más puro en esta vida: el de una madre.

A Luis, por su siempre presente cariño fraternal. Porque cada consejo y guía me llevó a ser la mejor versión de mí. Siempre me dijo que algún día entendería sus lecciones. Estaba en lo correcto.

A Karely, la mejor hermana que tengo la fortuna de tener en mi vida. Tu sentido del humor me acompañará todos los días.

A Sara, por apoyarme y motivarme desde que me acompañabas a clase en la preparatoria, hoy como profesionista y lo que sea que aguarde el futuro.

A Pedro y Carolina, a quienes guardo un profundo agradecimiento por recibirme con los brazos abiertos. Siempre los consideraré mi segunda familia.

A Carlos, mi maestro, tutor y amigo.

A mis amigos. Por escucharme, dejarme escucharlos, por sanar mis angustias y enseñarme tanto acerca de la vida.

Gracias.

Índice

Introducción 1
Capítulo 1: Responsabilidad Social en la empresa 2
1.1 Origen y evolución tecnológica de la empresa 3
1.2 Responsabilidad Social Corporativa. 7
1.3 El equipo. 9
1.4 Problemática. 10
1.5 Objetivo 12
Capítulo 2: Antecedentes y tecnologías del proyecto 13
2.1 La nube de la empresa 14
2.2 TJBot 17
2.3 Node-RED. 19
2.4 Nodos TJBot. 22
Capítulo 3: Resolución de la problemática 24
3.1 Componentes físicos del kit TJBot. 25
3.2 Problemáticas y soluciones: Componentes físicos. 30
3.2.1 Carcasa. 30
3.3 Componentes de Software. 34
3.4 Compatibilidad de los componentes instalados. 39
3.4.1 Creación de una cuenta y servicios en la nube de la empresa. 40
3.4.2 Configuración de un Bot en Node-RED. 42
3.5 Problemáticas y soluciones: Componentes de Software. 45
3.5.1 Optimizar el proceso de instalación. 46
3.5.2 Lograr la compatibilidad de componentes de software. 50
3.6 Estructuración de un taller y generación de recursos auxiliares. 62
3.7 Problemáticas y soluciones: Estructuración de un taller y generación de recursos auxiliares. 63
3.7.1 Facilitando la conexión entre TJBot y el equipo de cómputo. 63
3.7.2 Recursos auxiliares. 67

3.7.2.1 Manual de uso. 68
3.7.2.2 Página web. 71
3.7.2.3 Videotutoriales. 73
Capítulo 4: Resultados 75
4.1 La integración de soluciones logran la adopción educativa deseada. 77
4.2 Certificaciones. 78
4.3 Resultados primer semestre del 2019. 80
4.4 Resultados segundo semestre del 2019. 81
4.4.1 ¡Hello, TJBot! 82
4.4.2 Service Corps de la empresa 82
4.4.3 Trabajo a futuro 83
4.5 Conclusiones 86
Glosario 87
Bibliografía 92

Introducción

«El arte supremo del maestro es despertar el placer de la expresión creativa y el conocimiento». Albert Einstein.

La educación y la creatividad son los principales motivantes del trabajo realizado estos últimos meses, un proyecto de mejora que transforma un proyecto de código libre a uno que sirve como una introducción al uso de tecnologías cognitivas y el uso de la nube, como una fuente de soluciones tanto para expertos como para aquel que nunca ha programado.

El presente trabajo aborda la estructuración de un proyecto de código libre dentro del área de Responsabilidad Social Corporativa (RSC) en una empresa transnacional de tecnologías de la información (la empresa), donde se le otorga un enfoque educativo para permear las tecnologías más avanzadas en instituciones educativas y crear en los alumnos la ambición y curiosidad por involucrarse en temas STEM: Ciencia, tecnologías, ingeniería y matemáticas por sus siglas del inglés.

A lo largo de 4 capítulos se describe el camino recorrido para llevar a cabo esta transformación; a continuación, una breve reseña de lo que el lector puede esperar en cada uno de ellos.

En el Capítulo 1, Responsabilidad Social en la empresa, se resume la historia de la empresa desde un enfoque tecnológico, el antes y el después de su tecnología disruptiva en el mundo. Hablo acerca del rol del área de RSC dentro de la empresa y cómo mi rol dentro del área conlleva una estrecha relación entre esta tecnología y los objetivos del área. Por último, se detalla el origen del proyecto de código libre, llamado TJBot, como una herramienta para desarrolladores que toma la tecnología de la empresa para demostrar las funcionalidades de la nube.

El Capítulo 2, Antecedentes y tecnologías del proyecto, se explora a detalle el funcionamiento de cada una de las tecnologías involucradas en el proyecto TJBot y su interrelación: Rasberry PI, la nube de la empresa, JavaScript, Debian, Node-RED.

A lo largo del capítulo 3 se explorarán a detalle las problemáticas que se encontraban presentes en el proyecto, que se clasificaron en tres: Problemáticas de componentes físicos, problemáticas de estructuración de un taller y generación de recursos auxiliares.

Por último, el capítulo 4 muestra los resultados satisfactorios de la resolución de las problemáticas abordadas en el capítulo, donde se refleja el impacto que ha tenido en estudiantes, profesores, empleados de la empresa y en la responsabilidad social en el área misma.

Capítulo 1:

Responsabilidad Social en la empresa

Es conveniente exponer los antecedentes históricos de la empresa para comprender su filosofía y por qué durante más de 100 años de historia continúa transformando digitalmente ecosistemas completos a través de sus tecnologías y el cómo esto ha impactado el estilo de vida de la humanidad hasta hoy en día.

1.1 Origen y evolución tecnológica de la empresa

La empresa transnacional en cuestión ha sido vista a lo largo de la historia como una pionera tecnológica, que gracias a una constante innovación busca mejorar los procesos de sus clientes con infraestructura de última generación.

La empresa fue constituida formalmente en 1911, aunque sus orígenes datan de 1880. En esos tiempos a finales del siglo 19 la computación aún no permeaba en la industria, motivo por el cual la mano de obra manual resultaba costosa, lenta, no era fácilmente replicable y conceptos como la automatización de procesos no tomaban forma como lo entendemos hoy.

Procesos como inventarios, formularios de control o listas de asistencia se llevaban a cabo de forma manual por medio de hojas de apuntes en libretas, generando información difícil de manejar y consultar. Era necesario automatizar algunos procesos, tarea a la cual una compañía se dio a la tarea de resolver. Fue el director de esa empresa quien trajo al mundo un sistema de automatización de almacenamiento de información por medio del procesamiento de tarjetas perforadas de datos para apoyar el proceso de censo en Estados Unidos en el año de 1890.

Dicha empresa surge de la fusión en 1911 de algunas compañías antecesoras, una dedicada a la producción de relojes mecánicos registradores de tiempo y otra dedicada a las máquinas de pesar. Tras esta fusión se consolida una marca con su primer presidente. Dicho presidente genera una disciplina de ventas que llevan a la empresa a triplicar su valor en menos de 6 años llegando a más de 30 mil empleados. Este crecimiento lleva a tomar la decisión de cambiar el nombre de la compañía por su marca actual (que ha permanecido hasta la fecha con un reconocimiento mundial) para sugerir el alcance de su influencia y la variedad de negocios que estaban atendiendo.

A lo largo de la presidencia del fundador y la de su hijo (asumida en 1952) se produjeron innovaciones tecnológicas que superaron a sus competidores en su momento: La primera computadora de válvulas de vacío, el primer disco duro, el lenguaje de programación FORTRAN, el computador de transistores y el primer computador comercial de la historia (véase la figura 1) son solo algunos ejemplos.

Figura 1: El primer computador comercial de la historia

Fue en los años 80s cuando la empresa orienta sus esfuerzos en la producción de hardware y principalmente en el negocio de los computadores personales. La época fue próspera para la empresa, pero las computadoras personales comenzaron a popularizarse, donde competidores encontraron nichos de mercado donde la empresa tuvo que ceder: La empresa estaría cediendo su sistema operativo y su microprocesador respectivamente. Tras esto, la empresa estaría perdiendo poco a poco la fuerza en el mercado de los ordenadores personales.

Es hasta el año 2004 cuando la empresa sale del mercado de las computadoras personales con un nuevo negocio en la mira: Venta de software y servicios orientados al usuario. Los consultores de la empresa se ven orientados por las necesidades que el consumidor necesita resolver, usando para esto la tecnología y software adquiridos por la empresa a lo largo de la historia.

Actualmente la empresa ofrece soluciones a sus clientes a través de tres modelos de servicios en la nube: IaaS, SaaS y PaaS.

a) IaaS: *Infraestructure as a Service*, o infraestructura como servicio.

Es una propuesta donde el proveedor ofrece toda infraestructura de cómputo necesaria para que el usuario acceda a recursos como servidores, almacenamiento y red, de manera que el usuario utiliza sus propias plataformas y aplicaciones soportadas en la infraestructura ofrecida. Gracias a esto el usuario no deberá preocuparse por mantener ni dar soporte al equipo de cómputo.

b) PaaS: *Platform as a Service,* o plataforma como servicio.

Este modelo, además de infraestructura (IaaS), ofrece una nube ambientada donde los usuarios pueden desarrollar, gestionar y poner en funcionamiento sus aplicaciones. Esto se logra a través de una plataforma que cuenta con herramientas de desarrollo y pruebas.

c) SaaS: *Software as a Service,* o software como servicio.
Esta propuesta de cómputo en la nube provee a los usuarios acceso a software alojado en la nube. Las aplicaciones residen de forma remota en la nube y son accesibles a través de una página web o de una API. SaaS incluye además todos los beneficios de los modelos IaaS y SaaS.

El conjunto de las tres infraestructuras mencionadas ofrecen un conjunto de tecnologías para la resolución de problemas de industria, tales como *blockchain*, contenedores, gestión de redes, análisis de datos, internet de las cosas, bases de datos y una categoría de principal importancia para este trabajo: Inteligencia Artificial.

En esta última categoría la empresa ha desarrollado un servicio en la nube para acceder a la tecnología de una super computadora con capacidades de Inteligencia Artificial (la cual llamaré WIA, para no hacer mención de la empresa y respetar la confidencialidad de sus productos) que hace uso de diversas técnicas de aprendizaje máquina y redes neuronales para adquirir capacidades cognitivas, entre ellas entender el lenguaje natural, extraer la información más relevante y dar respuestas apropiadas a la información obtenida del entorno aprendiendo de cada interacción que tiene el sistema.

La principal materia prima para el aprendizaje de WIA son los datos. Estos datos pueden ser información estructurada y no estructurada.

La información estructurada se encuentra normalmente en archivos con formatos definidos, tienen longitud medida y se conoce el tamaño de los datos. Suelen estar almacenados en tablas, bases de datos u hojas de cálculo que son fácilmente analizados por una computadora.

Por otro lado, la información no estructurada carece de un formato de fácil acceso para computadoras tradicionales ya que no cuentan con un formato específico o medible. Un ejemplo de información no estructurada son artículos académicos o la conversación humana.

El sistema de Inteligencia Artificial de WIA ha utilizado sus capacidades cognitivas para obtener datos relevantes de información no estructurada (la más difícil de analizar) de forma que sus interacciones con la comunicación humana a través de internet, documentos científicos, o incluso conversaciones le ha dotado de inteligencia que permite a los humanos una fuente de información como apoyo en la toma de decisiones.

WIA ha permitido soluciones en los sectores financieros, climáticos y de salud, mejorando las decisiones que se toman en compañías, instituciones y gobiernos.

Los sistemas de Inteligencia Artificial se deben entrenar de acuerdo con la aplicación que se les desee dar, por tal motivo es que WIA cuenta con una plataforma abierta para que desarrolladores creen soluciones basados en esta tecnología combinando los servicios con los que la nube cuenta, todo por medio de su plataforma de nube, la cual es ofrecida al público de forma gratuita con ciertas limitantes y de paga para usos comerciales y empresariales.

La Inteligencia Artificial es un tema que gana popularidad estos años, pero aún es desconocida para grandes sectores de la población y escuelas a nivel nacional, por lo que surge la necesidad de acercar estas nuevas tecnologías a los centros educativos del país y así preparar a los futuros profesionistas que buscan aprovechar el potencial que este tipo de tecnología puede proveer a la humanidad.

1.2 Responsabilidad Social Corporativa.

Podemos entender una empresa responsable socialmente como aquella que hace uso de sus procesos de negocio y los orienta para generar un bien a la sociedad. Hoy en día son variadas y de todo tipo las empresas que ejercen responsabilidad social en la localidad donde llevan a cabo sus operaciones ya que éstas traen bienes a la empresa de diferentes maneras: Bienestar para los empleados, confianza y aceptación por parte de la región y las diversas entidades que la conforman o formar relaciones con sus grupos de interés son sólo algunas de ellas.

Figura 2: Actividad de voluntario por voluntarios de la empresa

A lo largo de su historia la empresa se ha consolidado como una empresa socialmente responsable, promoviendo temas como la inclusión, resiliencia, salud y educación a nivel global. Es el área de RSC quien juega el importante papel de ejecutar los planes de RSC en cada uno de los países donde la empresa tiene presencia.

Los programas por medio de los cuales la empresa ejerce su responsabilidad social evolucionan año con año y son distintos en cada región en la que la empresa se desempeña acorde a las prioridades locales, pero como marco general sus actividades pueden describirse en las siguientes categorías:

a) Desarrollo económico.

Programas enfocados en mejorar la calidad de vida de una población específica, que consiste en un equipo de voluntarios multidisciplinario y global que donan horas de consultoría a lo largo de 4 semanas para solventar un problema muy puntual.

b) Tecnología, educación, formación y empleo.

Se busca incentivar entre los jóvenes alumnos al estudio de carreras STEM (Ciencia, tecnología, ingeniería y matemáticas por sus siglas del inglés), donde por medio de capacitaciones y tutorías se busca despertar el interés por estas disciplinas. Además, se busca permear las tecnologías de la actualidad en centros educativos, entidades sociales y usuarios finales.

c) Desastres.

A través de actividades y capacitaciones se busca la mejora en resiliencia en caso de desastres naturales, mejorando la capacidad de reacción, lo que le permite a la sociedad volver a sus actividades normales en el menor tiempo posible.

Recordemos que la responsabilidad social de una empresa hace uso de sus procesos de negocio para ejercerla, esto se cumple de forma satisfactoria al hacer uso de los conocimientos generados dentro de la empresa en su día a día (Metodologías de trabajo, procesos innovativos, desarrollos tecnológicos) y llevarlos a la sociedad a través de soluciones, talleres y pláticas.

Precisamente uno de los enlaces para llevar este conocimiento a la sociedad es el área de RSC, la clave de este acercamiento son los voluntarios de la empresa: Aquellas personas quienes organizan su tiempo para dedicarse a su labor profesional y a su vez compartir su experticia, mostrando y compartiendo sus conocimientos a otros. Los voluntarios de la empresa facilitan trasladar los conocimientos requeridos por el sector empresarial a los centros educativos, esto incluye las tecnologías que la empresa maneja en su día a día y que alumnos de escuelas de cualquier nivel educativo aprovechan para conocer lo que en el mundo laboral aguarda.

Los apoyos que RSC de la empresa otorga con fines educativos van desde talleres de capacitación hasta donativos en efectivo o hardware de acuerdo con la naturaleza de lo que la fundación o el centro educativo busque o necesite.

Hablando específicamente de proyectos que permean tecnologías actuales en el sector educativo, a finales del año 2017 surgió un proyecto llamado TJBot, una propuesta de código abierto que permite crear a un robot programable, el cual hace uso de la Inteligencia Artificial de WIA para acercar servicios cognitivos de la nube de la empresa de forma sencilla y accesible para el público en general de desarrolladores.

Su atractivo diseño, las tecnologías que usa y la facilidad de uso fueron características relevantes para que el área de RSC en México se planteara la posibilidad de que TJBot potencialmente se convirtiera en una herramienta disruptiva para integrarse al currículo tecnológico de las escuelas, apoyando a los docentes en la impartición de sus materias y facilitando la incursión de temas como programación e Inteligencia Artificial.

La Inteligencia Artificial resulta ser una de las apuestas más importantes para la empresa debido a la fuerte incursión de estas tecnologías en el mercado y en los negocios de la actualidad; y que en estos momentos está viviendo su gran auge multidisciplinario en diferentes industrias, esto no necesariamente va de la mano con el sector educativo, el cual no se transforma con la misma velocidad como lo ha hecho el mundo laboral hoy. Somos testigos de una brecha tecnológica entre lo que en las escuelas se imparte y lo que el campo laboral demanda.

El área de RSC encontró el proyecto TJBot como un enlace entre la tecnología de la empresa y las escuelas del país, al mismo tiempo que podía ejercer su responsabilidad social. Gracias a esto el área designa a una persona dedicada a adaptar y mejorar el proyecto a las condiciones académicas de las aulas para que este pudiera ser escalado a nivel nacional. La mejora del proyecto permitiría generar curiosidad y compromiso en los alumnos para el estudio de temas relacionados con las habilidades STEM.

Sin embargo, este proyecto debía ser mejorado: Se presentaban fallas de componentes, su software se encontraba desactualizado e incompatible a la plataforma de la nube, su uso aún requería de conocimientos especializados para poder ser manejado y utilizado de forma apropiada.

1.3 El equipo.

Los equipos de trabajo de RSC en la empresa son diferentes de acuerdo con las necesidades de cada región, en el caso de México actualmente formo parte de un equipo de 3 integrantes quienes implementamos proyectos de RSC en la región integrada por México y Costa Rica.

Si bien los integrantes del equipo comparten las responsabilidades e interaccionan de algún modo u otro con la realización de éstos, sus responsabilidades están bien definidas (véase la figura 3):

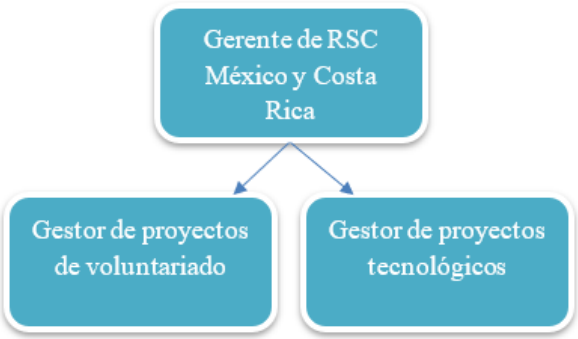


Figura 3: Organigrama: RSC México y Costa Rica.

El primer integrante, el gerente del área, es el encargado de ser el enlace principal entre las organizaciones sin fines de lucro en la empresa, así como de transmitir las prioridades, lineamientos y proyectos mandatorios del área de RSC mundial.

Luego se encuentra la persona encargada de gestionar los proyectos de voluntariado para actividades de asistencia, integración, donativos en efectivo y de apoyo a las organizaciones sin fines de lucro. Coordina convocatorias internas y moviliza voluntarios a los proyectos que lo requieran.

El tercer perfil es el cargo que cumplo yo, el puesto lleva por nombre Gestor de Proyectos de RSC y particularmente en la vacante que ocupo las responsabilidades de dividen en dos:

a) Por un lado, son responsabilidades orientadas a utilizar mis conocimientos de ingeniería para hacer uso de las tecnologías que la empresa ofrece, como son Internet de las Cosas, Asistentes virtuales, Inteligencia Artificial, entre otros y acercarlos a las organizaciones sin fines de lucro o instituciones educativas, de forma que éstas puedan entenderlas y aplicarlas a la solución de sus necesidades.
   Considerando que los participantes se encuentran en un nivel educativo desde escuela primaria hasta Universidades, llevar a cabo este proceso de enseñanza requiere de la realización de talleres, pláticas, capacitaciones, materiales y guías.

b) Por otro lado, la realización de actividades con estas organizaciones requiere de habilidades de gestión de proyectos que van desde el primer contacto con la organización, llegar a acuerdos acerca de sedes y fechas, firma de convenios, compra de recursos, documentación de apertura y cierre y controles financieros.

1.4 Problemática**.**

El proyecto de código libre TJBot fue una creación de una empleada de la empresa, una inventora en Estados Unidos. TJBot nace con el propósito de acercar a desarrolladores a las tecnologías de la empresa demostrando lo fácil que es integrar tecnologías de Inteligencia Artificial en la vida cotidiana.

Gracias a que TJBot es de código libre el proyecto fue retomado por otro ingeniero de la empresa para integrarlo a una herramienta de programación abierta al público y de código libre llamada Node-Red (programa basado en programación por bloques), éste facilita la programación a personas sin experiencia previa en lenguajes de programación.

Los resultados de estas dos iniciativas se vieron reflejadas en innovadoras soluciones que desarrolladores creaban a partir de TJBot alrededor del mundo, haciendo uso de tecnologías cognitivas en la nube. Mientras esto ocurría el proyecto dejó de ser actualizado por sus creadores y pasó a ser actualizado únicamente por desarrolladores alrededor del mundo.

Al ver las capacidades de TJBot y el alcance que este podría tener como un recurso educativo, el área de RSC de la empresa en México tomó la iniciativa planteando la posibilidad de integrar el proyecto a sus planes de Responsabilidad Social, de tal manera que pudiera ser un apoyo al docente en el salón de clases y una forma innovadora de formar a los alumnos en actividades relacionadas con las habilidades STEM (Ciencia, Tecnología, Ingeniería y Matemáticas, por sus siglas del inglés).

a) Para lograr esto existían tres grandes impedimentos a resolver: El proyecto tenía un fuerte enfoque a desarrolladores experimentados.

   Su comprensión requería de estar familiarizado con temas como sistemas operativos, ventanas de comandos, comandos y lenguajes de programación.
b) Proyecto desactualizado.
   Los creadores no mantenían el proyecto con actualizaciones oficiales, por lo que su funcionamiento dependía de la comunidad de desarrolladores.
c) Documentación descentralizada y escasa.
   No existía una guía clara para comenzar a trabajar con TJBot.

Con la poca documentación existente, el equipo de RSC realizó pruebas iniciales con TJBot en algunas demostraciones. En general, los resultados obtenidos arrojaron los siguientes problemas:

a) El software y hardware fallaba de forma aleatoria, algunos componentes no funcionaban en ninguna circunstancia y la carcasa resultaba frágil para el armado.
b) Instalación susceptible a errores.
c) Poca documentación para resolución de problemas.
d) La conectividad con TJBot requería de componentes externos al proyecto (mouse, teclado, pantalla o proyector, cables de conexión), lo que lo volvía un proceso más lento y costoso.
e) El proyecto no estaba apto para implementarse de forma masiva en instituciones académicas del país.

Las problemáticas anteriores requerían de un perfil de ingeniero en computación que por medio de sus habilidades y conocimientos pudiera llevar el proyecto a un estado óptimo de funcionamiento, tanto el software como hardware; dicho perfil no existía en el área, esto llevó a mi integración al equipo. Mis conocimientos en lenguajes de programación y antecedentes en Inteligencia Artificial permitirían concluir la integración del proyecto a los planes del área de RSC.

Debido a las responsabilidades de los demás miembros del equipo, todo lo relacionado con el proyecto TJBot correría a cargo de mi responsabilidad: Plan de trabajo, solución y optimización del proyecto y generación de documentación. Mi gerente directo sería el responsable de supervisar los resultados.

1.5 Objetivo

El objetivo del presente trabajo es la optimización y estructuración de un proyecto de código abierto (en particular el proyecto *"TJBot"*) como un apoyo tecnológico en la adquisición de habilidades STEM (Por sus siglas del inglés Ciencia, Tecnología, Ingeniería y Matemáticas). Para llevar a cabo esto se plantearon los siguientes objetivos específicos:

a) Identificación e investigación de las fuentes de información relacionadas al proyecto.
b) Construcción, pruebas y correcciones del proyecto.
c) Estructuración de un taller para su incorporación en procesos educativos.
d) Documentación de un manual de usuario y recursos adicionales.

Capítulo 2:

Antecedentes y tecnologías del proyecto

TJBot trabaja con una base sólida de tecnologías, las cuales en conjunto se traducen en un poderoso proyecto multidisciplinario, flexible y de fácil entendimiento. Para lograr tales ventajas, es importante comprender el poder que traen consigo las tecnologías detrás de TJBot, entre los que se encuentra una nube de Inteligencia Artificial, software de código libre, JavaScript, JSON y una comunidad de desarrolladores que han aportado forma y estructura al proyecto actual.

2.1 La nube de la empresa

La empresa cuenta con una amplia gama de soluciones digitales que contribuyen al desarrollo tecnológico de la sociedad y que son de fácil acceso para desarrolladores a través de la nube. La nube de la empresa integra más de 190 servicios (hasta la fecha) accesibles a través de internet que apoyan en la solución de problemáticas de diversa índole: Cálculo, gestión de redes, almacenamiento, bases de datos, internet de las cosas, Inteligencia Artificial entre otros; ésta última categoría es la que dará vida al proyecto TJBot.

Los servicios de Inteligencia Artificial que ofrece la empresa son provistos a través de WIA, la cual utiliza cómputo cognitivo potenciado por Inteligencia Artificial, un tipo de cómputo que busca imitar las capacidades cognitivas del cerebro humano tales como la comprensión del lenguaje natural, aprendizaje basado en grandes cantidades de datos tanto estructurados como no estructurados, razonamiento, escuchar, observar o hablar.

Para facilitar el acceso a la tecnología de WIA se creó una página web con cierto nombre (la cual llamaré BX, para no mencionar a la empresa y respetar la confidencialidad de sus productos), una plataforma tipo PaaS (Plataforma como Servicio, por sus siglas del inglés).

Una PaaS es un tipo de entorno de desarrollo con base en la nube, contiene dentro de sí la infraestructura necesaria para soportar la tecnología que ofrece, así mismo cuenta con middleware que permite la interconexión entre la infraestructura, el sistema operativo, los servicios ofrecidos y las aplicaciones que sean creadas a partir de los servicios.

El que la nube de la empresa sea del tipo PaaS permite que los usuarios de cualquier parte del mundo puedan conectar sus aplicaciones con los servicios de la empresa de forma sencilla a través de credenciales de acceso; así mismo cada uno de los servicios permite como consultar la documentación correspondiente y las instrucciones para ser implementados en aplicaciones de cualquier tipo y en distintas plataformas tecnológicas. Otra característica para destacar es que el usuario solamente pagará lo que consuma de acuerdo con los planes de consumo definidos para cada servicio.

En general los servicios ofrecidos a través de la nube cuentan con 3 planes de precios:

a) Lite.

Los servicios con un plan lite son de uso gratuito limitado a cierta cantidad de eventos o tiempo por mes; cada mes los valores se reinician y se puede continuar usando hasta el límite establecido. Este tipo de plan es recomendado para pruebas y proyectos pequeños que no serán comercializados. Para el proyecto TJBot, este es el plan que utilizamos y es suficiente para hacer funcionar al proyecto.

b) Estándar.

Conlleva una tarifa por monto de uso. Este plan es recomendado para productos comerciales a pequeña y mediana escala.

c) Premium.

Está pensado para un consumo empresarial, cuyo costo varía de acuerdo con las necesidades del negocio.

A finales del año 2017 se anunció la fusión de la marca BX con una nueva marca, evolucionando entonces en un proceso que finalizaría en los últimos meses del año 2018. Hoy en día los procesos de migración han concluido y la nueva liga de acceso a los servicios de cambió con un nombre bajo una nueva marca, logo y diferentes actualizaciones sobre el acceso a sus servicios.

Si bien son más de 190 los servicios ofrecidos por esta plataforma, TJBot fue concebido originalmente para utilizar 6 servicios:

a) Text to speech (Texto a habla).
b) Speech to text (Habla a texto).
c) Visual recognition (Reconocimiento visual).
d) Conversation. (Conversación)
e) Tone analyzer (Análisis de tonos).
f) Language translation. (Traducción de lenguaje).

Para hacer uso de cualquiera de estos servicios en TJBot o cualquier otra aplicación, los pasos a seguir son sencillos y se enlistan a continuación:

1- Crear una cuenta en la página de la nube de la empresa.
2- Dirigirse al catálogo de servicios ofrecidos, alojado en la parte superior de la pantalla (véase la figura 4).

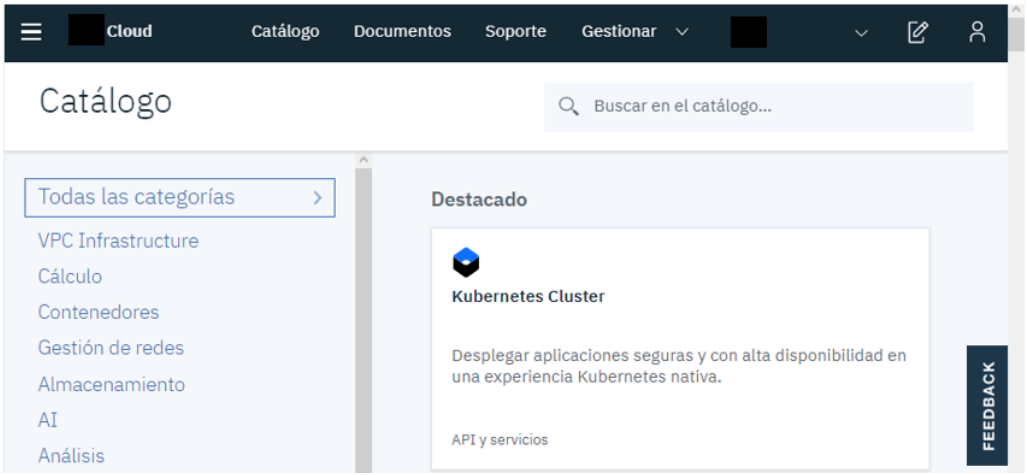


Figura 4: Vista del catálogo de la empresa

3- En el buscador colocar el nombre del servicio a utilizar y seleccionarlo.
4- Dar clic al botón “Crear”.
5- Dirigirse a la pestaña “Credenciales de servicio” (véase la figura 5) y copiar la API Key generada.

Figura 5: Pagina para visualizar credenciales del servicio Speech to Text

La API Key es una credencial de acceso que permitirá a cualquier aplicación, con el código de autenticación correcto, invocar los servicios de la nube y hacer uso de ellos.

Este sistema de autenticación es tan sencillo que permite la integración de tecnologías de diferentes tipos en cualquier dispositivo que cuente con capacidades de procesamiento y conectividad a internet, por lo que se puede extrapolar esta tecnología flexible a teléfonos inteligentes, automóviles, computadores o tarjetas de desarrollo.

Gracias a esta compatibilidad es que TJBot puede invocar servicios de Inteligencia Artificial ya que sólo requiere de la clave de API para hacer uso de esta tecnología.

2.2 TJBot

Con miras a acercar la tecnología de WIA al público en general, un grupo de investigadores de Inteligencia Artificial de la empresa trabajaron en su centro de investigaciones con sede en Nueva York (véase la figura 6), quienes presentaron a finales del año 2016 un proyecto que funcionaría como interfaz humano-máquinas entre los servicios de la WIA y desarrolladores.

Figura 6: Centro de investigaciones de la empresa

El equipo de investigadores se centró en crear un proyecto de código abierto, esto significa que la comunidad de desarrolladores tiene la posibilidad de explorar el código fuente y realizar modificaciones de software y hardware para extender las capacidades para las cuales fue originalmente desarrollado.

El resultado fue un robot llamado TJBot, en honor el primer CEO de la empresa. TJBot se define como un proyecto de código abierto que permite al usuario construir y programar a un robot que cobra vida gracias a la Inteligencia Artificial, integra algunos componentes electrónicos que son controlados por una tarjeta de desarrollo RPI (RPI).

La conectividad de la tarjeta Raspberry PI (RPI) a internet permite realizar conexiones con WIA a través de la nube, lo que le otorga habilidades como sostener conversaciones con las personas, realizar reconocimiento visual o entender sentimientos.

Estas actividades son sólo algunos ejemplos ya que la realidad es que el proyecto está enfocado a que el usuario programe sus propias actividades. El hecho de que este proyecto sea de código abierto permite modificar su funcionamiento de manera que es posible instalar cualquier cantidad de recursos (software y hardware). El robot (véase la figura 7) está hecho para ser usado como propósito general para desarrolladores, pero si el usuario final lo dispone puede programarlo para resolver necesidades específicas.

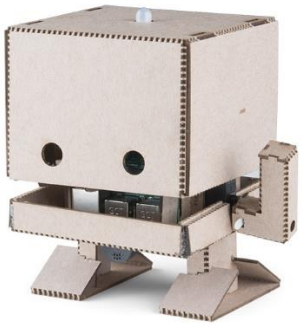


Figura 7: Proyecto TJBot

Los componentes de hardware de TJBot son los siguientes:

a) 1 estructura de TJBot (Impresión 3D o cartón).
b) 1 tarjeta de desarrollo RPI 3 ó modelos similares.
c) 1 memoria microSD.
d) 6 cables jumper (3 hembra-hembra y 3 hembra-macho).
e) 1 foco Led.
f) 1 servomotor.
g) 1 bocina USB.
h) 1 micrófono USB.
i) 1 cámara RPI v2 ó v3.
j) 1 fuente de alimentación de 5 volts.

El control de estos componentes está definido en el código fuente de TJBot escrito en lenguaje de programación JavaScript, apoyándose en el uso del *framework* NodeJS.

JavaScript es un lenguaje de programación mayormente utilizado para desarrollar la parte visual de una aplicación web (frontend), pero que por medio de NodeJS (framework para el lado del servidor) se construye código backend para la lógica. Así los inventores construyeron la librería de funciones que abstrae la comunicación entre el usuario y el robot. Finalmente la librería fue alojada en Github en noviembre de 2016 para un acceso libre y gratuito.

Esto consolida el primer acercamiento de TJBot la comunidad, pero el nivel de complejidad es elevado pues aún requiere de experiencia en el manejo de JavaScript y en la ventana de comandos del sistema operativo.

2.3 Node-RED.

Node-RED (véase la figura 8) es una interfaz de programación basada en bloques (llamados nodos) interconectados que permiten crear soluciones robustas integrando toda clase de servicios y tecnologías, todo al alcance de usuarios sin experiencia técnica previa.

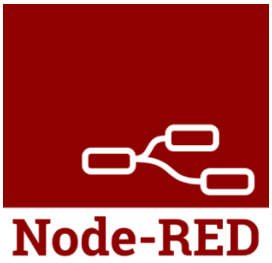


Figura 8: Logo de Node-RED

Este sistema nace como una manera de facilitar la interconexión entre hardware y software. El sistema es presentado como un proyecto de código libre que además de la sencillez de su uso facilita la integración de APIs de terceros. Gracias a esto, Node-RED funciona como interfaz donde el usuario final no necesitará conocimientos previos de programación para poder realizar programas que interactúen directamente con los componentes del robot.

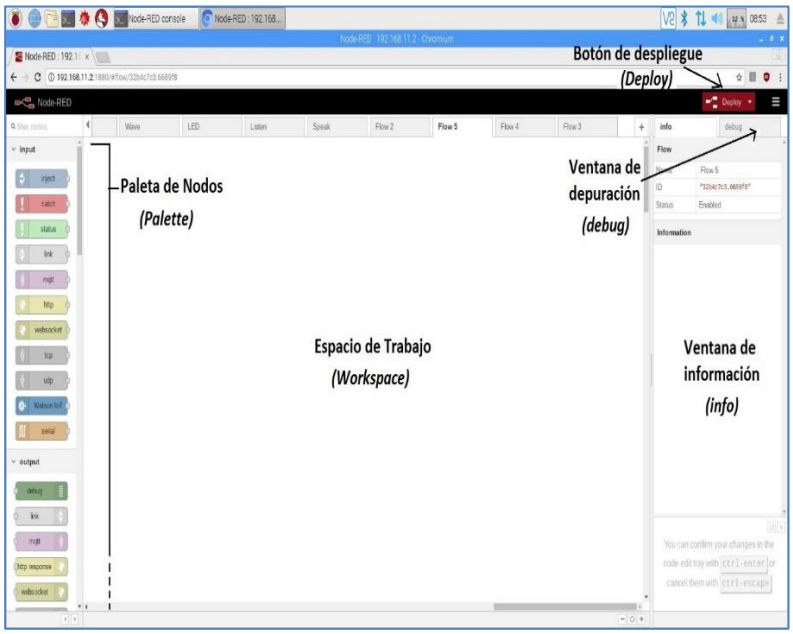


Figura 9: Interfaz de programación de Node-RED

La interfaz (véase la figura 9) se integra principalmente de los siguientes componentes:

a) Una paleta de nodos que se encuentran del lado izquierdo.
b) Un espacio de trabajo donde los flujos son construidos.
c) Una ventana de información que describe el funcionamiento del nodo seleccionado.
d) Una ventana de depuración que muestra información relevante durante el funcionamiento de los flujos.
e) Un botón de despliegue que guardará los cambios realizados en el espacio de trabajo.

El eje central del programa son los nodos (véase la figura 10), donde cada nodo existente en el programa es un pequeño módulo preprogramado que requiere de configuraciones mínimas para su funcionamiento; y en su conjunto definirán el alcance de los programas creados. Además, los nodos de Node-RED pueden ser contribuidos por la comunidad de desarrolladores, lo que ha permitido que la librería de nodos crezca con nuevos servicios de terceros.

Figura 10: Nodo inject que transmite la hora y fecha actual

La interconexión de los nodos cumple un rol importante en Node-RED ya que cada nodo recibe un mensaje del nodo anterior. La información contenida en este mensaje le puede ser de utilidad al nodo siguiente para llevar a cabo su labor. Cuando este último nodo culmina su proceso puede modificar la información del mensaje y nuevamente transmitirlo al siguiente nodo. A esta interconexión de nodos se le llama flujo (véase la figura 11).

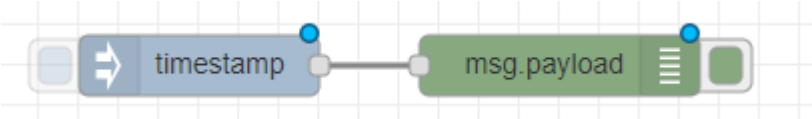


Figura 11: Flujo ejemplo que realiza el debug de un mensaje

Los mensajes que se envían a través del flujo se encuentran en un formato llamado JSON (*JavaScript Object Notation*), un formato de texto simplificado que consta de una estructura del tipo "Llave – Valor" que permite a los programas acceder fácilmente a la información que contienen. JSON puede ser utilizado como medio de transmisión de información por una gran variedad de lenguajes de programación y en general la manera de acceder a estos datos suele ser similar.

El operador punto "." generalmente permite a los lenguajes de programación acceder a los datos. El siguiente ejemplo de un mensaje JSON permitirá ejemplificar el acceso a los datos contenidos por medio de JavaScript.

Supongamos que tenemos un mensaje JSON (véase la figura 12) contenido en una variable de datos llamada "*msg"* con la siguiente estructura:

```
msg = {
        "Nombre": "Sara",
        "Edad": 25,
        " msg_id": 1234
}
```

Figura 12: Ejemplo de un mensaje JSON

Podemos acceder al dato contenido en la llave "Edad" escribiendo la instrucción *"msg.Edad"* en un programa. De la misma manera, si escribimos "*msg.Nombre"* obtendremos el valor de la llave "Nombre". Los nodos de Node-RED acceden a la información contenida en estos mensajes en formato JSON con esta operación punto, lo que les permite hacerse de datos que otros nodos han generado para realizar sus propias operaciones y alojar en el mensaje nueva información.

Gracias a esta poderosa característica es que los nodos pueden trabajar con tecnologías tan distintas, haciendo uso de un medio de comunicación común que todos comprenden.

2.4 Nodos TJBot.

En junio de 2017, a tan solo unos meses después de la liberación del código fuente de TJBot, un técnico especialista en innovaciones de la empresa diseñó una interconexión entre TJBot y Node-RED, creando los nodos de TJBot.

Node-RED integra la posibilidad de añadir nuevos nodos por medio de un buscador integrado que utiliza como fuente la página https://flows.nodered.org, un repositorio colaborativo que ofrece nodos de todos tipos. Como alternativa al buscador, los nodos pueden ser agregados además por la ventana de comandos; esto resulta útil para aquellos nodos que no se encuentran en el repositorio.

El inventor codificó nodos de TJBot que funcionan como una interfaz para el usuario entre el código fuente creado por el equipo de investigación de la empresa y Node-RED, permitiendo a usuarios sin experiencia en programación manipular al robot. Estos nodos (excepto los dedicados a activar el servomotor y el led) realizan una conexión con la nube de la empresa, envían la información al servicio que están utilizando, reciben la información de vuelta, la ejecutan y la transmiten al siguiente nodo.

Para que los nodos creados por el empleado de la empresa anteriormente mencionado estén disponibles en la interfaz de Node-RED, es necesario instalarlos por medio de la ventana de comandos, descargándolos desde un repositorio en Github. Los nodos de TJBot se encuentran alojados en el repositorio https://github.com/JeanCarl/Node-RED-contrib-tjbot y su documentación describe que los nodos hacen uso de los siguientes servicios de WIA:

1- Text to speech (Texto a habla).
2- Speech to text (Habla a texto).
3- Visual Recognition (Reconocimiento visual).
4- Tone Analyzer (Analizador de tonos).
5- Language translator (Traducción de lenguaje).
6- Conversation (Conversación).

Los nodos en el repositorio son los siguientes (véase la figura 13):

Figura 13: Nodos TJBot del repositorio Github

La función que cumple cada uno de ellos se describe a continuación:

1- *Converse*: Utiliza el servicio *Conversation* de WIA, para entablar conversaciones con el robot.
2- *Listen*: Utiliza el servicio *Speech to Text* para escuchar transformar la voz a texto. Hace uso del micrófono.
3- *See*: Utiliza el servicio *Visual Recognition* para analizar imágenes obtenidas por medio de la cámara de la RPI. Tiene 3 modos de funcionamiento: Obtención de texto basado en imagen, análisis de imágenes y toma de fotografías.
4- *Shine*: Interactúa con el led del robot.
5- *Speak*: Utiliza el servicio *Text to Speech* para tomar texto y sintetizar voz humana que lo hable. Hace uso de la bocina para mostrar sonido.
6- *Analyze tone*: Gracias al servicio que lleva el mismo nombre, toma el texto y analiza los tonos en él tales como alegría, tristeza o ira.
7- *Translate*: El servicio *Language Translator* permitirá habilitar el funcionamiento mediante 3 modos de uso: Identificar si un lenguaje puede ser traducido mediante el servicio, identificar alguno de los 62 lenguajes que WIA conoce y traducir un lenguaje entre más de 20 lenguajes distintos.
8- *Wave*: Activa el servomotor para hacer distintos movimientos.

Capítulo 3:
Resolución de la problemática

Para abordar el problema fue requisito indispensable, además de la investigación, construir mi primer TJBot. Al experimentar en mis propias manos el armado y la preparación del software pude analizar su funcionamiento y obtener los puntos clave de mejora.

Cuando me integré al equipo de RSC el equipo contaba con un kit TJBot completo, desarmado en sus componentes internos, pero no se contaba con información concisa acerca de su armado o sus bases de funcionamiento, así que la información para comenzar se encontraba alojada en internet y en los conocimientos propios de la ingeniería en computación.

Tracé un plan de trabajo, que desde un punto de vista general, resulta en tres etapas que debían ser exploradas y resueltas:

1- Ensamble de los componentes físicos.
   Todo lo relacionado al armado del robot y el correcto funcionamiento de las interconexiones entre los componentes del hardware.
2- Componentes de software.
   Esto se refiere a todo lo relacionado a la inicialización del sistema, instalación del Sistema Operativo, de las librerías de TJBot, habilitación de Node-RED, compatibilidad con hardware y correcto funcionamiento de los componentes integrados.
3- Estructuración de un taller y generación de recursos auxiliares.
   Son aquellos aspectos que permiten la implementación de un taller por parte de un voluntario o un profesor. Considera la creación de recursos de apoyo tales como manuales, presentaciones o recursos web para el correcto flujo de un taller.

Antes de indagar en la búsqueda de puntos de mejora me tomé un tiempo de investigación para explorar las bases del funcionamiento de TJBot, los cuales son los temas desarrollados en los antecedentes de este trabajo.

3.1 Componentes físicos del kit TJBot.

TJBot cuenta con 10 componentes (véase la figura 14) de hardware fácilmente localizables en el mercado y una carcasa que conforma la estructura del robot. Éste último elemento debe fabricarse o mandarse a hacer por corte laser o una impresora 3D basado en unos archivos de diseño.

Figura 14: Componentes del kit TJBot

Los componentes de TJBot son (véase la Tabla 1):

Tabla 1: Componentes de TJBot

Nombre	Cantidad
Mini micrófono USB.	1
Bocina USB	1
Tarjeta de desarrollo RPI 3.	1
Módulo de cámara para RPI.	1
Cables jumper macho-hembra.	3
Cables jumper hembra-hembra.	3
Servomotor.	1
Led multicolor Adafruit.	1
Memoria SD de 16 GB con NOOBS preinstalado.	1
Estructura de cartón cortado en laser de TJBot.	1
Fuente de alimentación 5V 3000mA.	1

Las instrucciones iniciar con el armado se encuentran en YouTube en un video llamado *TJBot assembly* (*https://youtu.be/bLt3Cf2Ui3o*); en él se describe la manera en que el cartón se dobla y cómo los componentes se introducen en el cartón doblado (véase la figura 15).

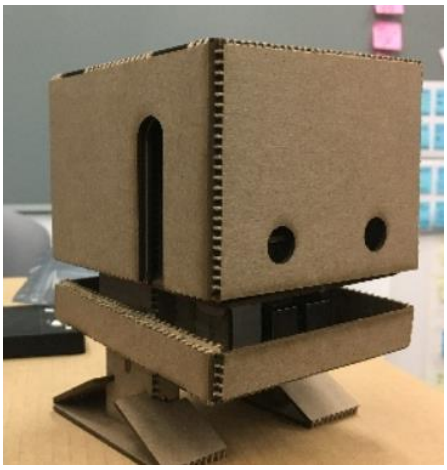


Figura 15: Estructura armada de TJBot

Siguiendo estas instrucciones el proceso de armado me llevaba aproximadamente 45 minutos. El video me permitió construir al robot en su mayoría, pero no es claro con algunas instrucciones sobre cómo llevar a cabo las conexiones entre el led, el servomotor, la cámara y la memoria SD; si bien algunas pueden ser intuitivas, muchos usuarios podrían requerir instrucciones más puntuales para una construcción más fluida.

El área de RSC de la empresa contaba además con una carcasa de impresión 3D de TJBot (véase la figura 16) cuyo armado es similar al cartón, con la diferencia de que la impresión 3D ahorra pasos pues algunas piezas ya se encuentran listas para el ensamble, debido a que no se requieren dobleces por ser impresión 3D. Para el armado es sólo cuestión de ensamblar las piezas y los componentes electrónicos entre sí. El armado me tomó alrededor de 30 minutos.

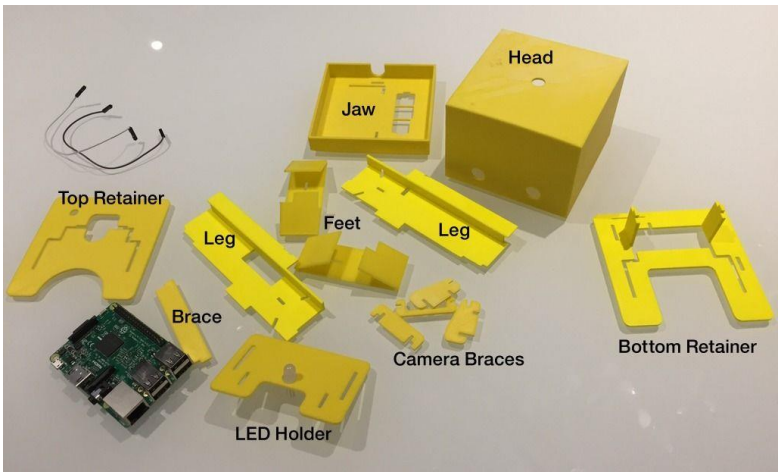


Figura 16: Estructura de impresión 3D de TJBot

Tras explorar en internet encontré un diagrama (véase la figura 17) que muestra las conexiones físicas entre el led, servomotor y la tarjeta, que a través de los colores mostrados en el diagrama se convierten en conexiones sencillas de realizar. Se utilizan 6 cables *jumper*: 3 del tipo hembra-hembra y otros 3 del tipo hembra-macho.

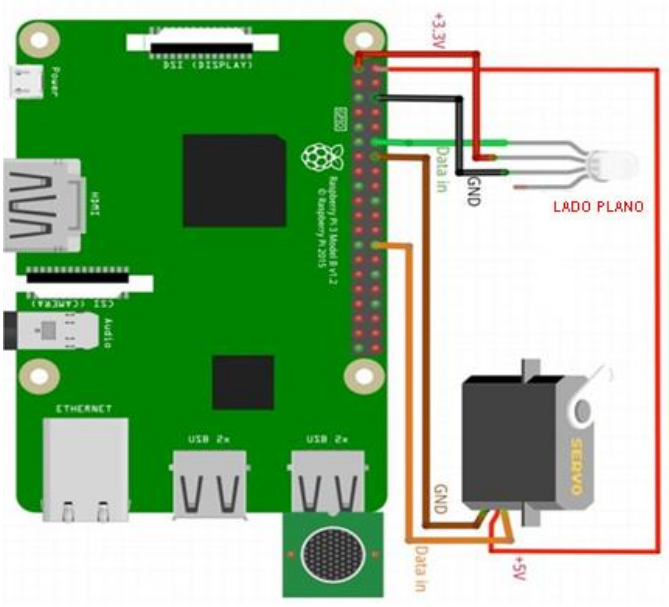


Figura 17: Diagrama de conexiones entre componentes

En el diagrama anterior se muestra un ejemplo de la tarjeta de desarrollo RPI, la cual cuenta del lado derecho con pines disponibles para realizar conexiones físicas con cualquier componente. En el caso de TJBot se utilizan 6 de estos pines para conectar el led y el servomotor, sin embargo, vale la pena mencionar que la tarjeta puede controlar éste y muchos otros dispositivos a través de los pines y que tal funcionamiento dependerá de la programación que el usuario genere a través del sistema.

Lo siguiente es colocar la memoria SD (véase la figura 18) dentro de la bandeja metálica que se encuentra en la parte posterior de la RPI. Ésta es una memoria de 16 GB clase 10 y vienen pregrabadas con un sistema de inicio llamado NOOBS (*New Out Of Box* software), sistema que permite llevar a cabo una instalación limpia de algún sistema operativo en una RPI.

Figura 18: Tarjeta de memoria SD

En este punto descubrí que para comenzar a trabajar con cualquier RPI se aconseja comenzar con una memoria pregrabada con NOOBS. Si un usuario tiene una memoria que no está pregrabada puede descargar este software a través de la página de internet oficial de Raspberry PI y grabarlo en la memoria.

Finalmente, los últimos componentes que fueron conectados fueron el micrófono y la bocina (véase la figura 19). Estos componentes se conectan en cualquiera de los puertos USB de la tarjeta.

Figura 19: Micrófono y bocina USB

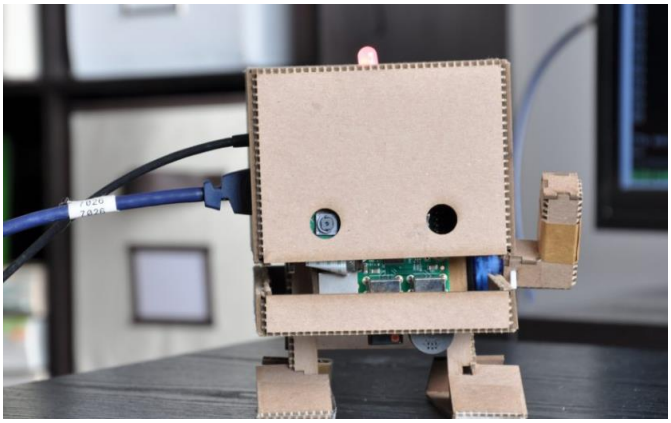


Figura 20: TJBot completamente construido

El armado fue realizado además por cuatro personas más, sin experiencia en temas de electrónica o robótica, haciendo uso de las instrucciones que recabé por internet; esto permitió tener más de un punto de vista de las imperfecciones y complicaciones de que alguien sin experiencia podría tener al momento de construir completamente el robot.

En promedio los participantes tardaron una hora y media construyendo a TJBot con la carcasa de cartón, mientras que para construir el kit con la carcasa 3D tardaron poco menos de una hora. Sus comentarios respecto a los puntos de mejora coincidieron en que fue más sencillo armar el kit con impresión 3D y que tener una guía que concentrara la información referente al armado del kit les permitiría ensamblarlo más rápidamente.

3.2 Problemáticas y soluciones: Componentes físicos.

En cuanto a las problemáticas encontradas en los componentes físicos de TJBot encontré la necesidad de contar con una carcasa que pudiera facilitar el armado sin aumentar los costos del material, así como que el material fuera fácil de manipular y construir por cualquier usuario. Con esto en mente concluí que la solución implicaría la creación de un material nuevo para la construcción del robot.

3.2.1 Carcasa.

Los archivos de diseño para la carcasa de cartón (véase la figura 21) cuentan con un total de 15 piezas. Su armado consiste en doblar las piezas para dar la forma poco a poco de la estructura del robot.

El doblar las piezas del diseño trae como consecuencia la debilidad más importante ya que particularmente las de las piernas, los pies y el brazo del robot se dañaban accidentalmente, requiriendo cinta adhesiva o pegamento para conservar su forma requerida.

Aunado a esto, si el material se utilizaba con un número más grande de participantes durante un taller, las consecuencias eran además:

a) Alta fragilidad del material.
b) Poca durabilidad de la carcasa a mediano y largo plazo.
c) La necesidad de materiales extra como cinta adhesiva o pegamento.
d) Aumenta la cantidad de tiempo empleado en el armado.

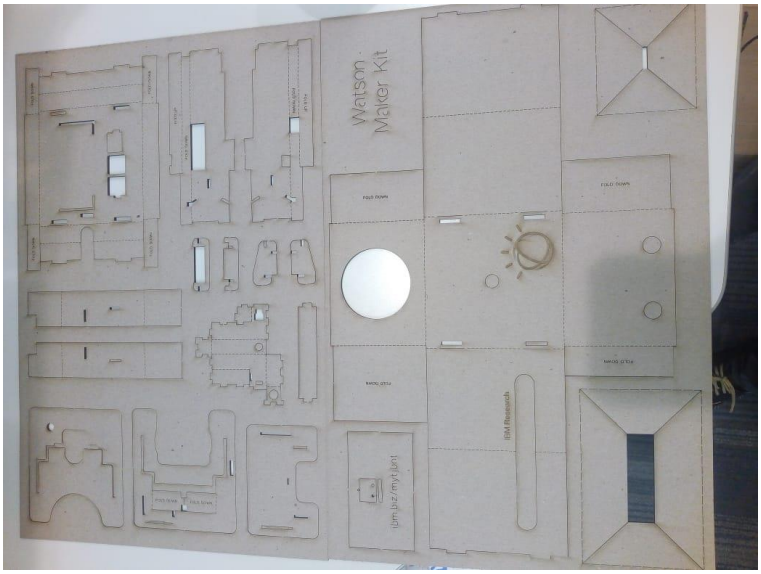


Figura 21: Estructura de TJBot de cartón

Como alternativa al material de cartón se encuentra también la estructura en impresión 3D, cuyos planos se encuentran alojados en internet.

La siguiente tabla (véase la Tabla 2) describe las características más importantes de los dos materiales.

Tabla 2: Comparativa de materiales de carcasa

Material	Cartón	Impresión 3D
Armado	Doblado y ensamblado de piezas	Ensamblado de piezas
¿Requiere material extra?	Si (Pegamento, cinta)	No
Tiempo de producción por carcasa en maquiladora o impresión.	10 - 20 minutos	24 horas.

A su vez, el costo del material en impresión 3D es poco más de tres veces más costoso respecto al de cartón.

Comparando las ventajas entre ambos materiales era necesario encontrar un material que contara con un precio accesible, consistente en un simple ensamblado de piezas, que no requiriera de materiales adicionales en su armado y cuya producción tomara el menor tiempo posible.

En conversaciones con nuestro proveedor de recursos y materiales, encontré dos alternativas de materiales que cumplían con lo requerido: Acrílico y MDF (Tablero de fibra de mediana densidad, por sus siglas en inglés). Para cualquiera de los dos materiales sería necesario el rediseño de las piezas que requerían de dobleces, de tal manera que fueran piezas bidimensionales ensamblables para dar la forma tridimensional.

El proveedor accedió a realizar un rediseño del material con mi supervisión y guía, con lo que comenzaron las pruebas y ajustes necesarios para el correcto ensamble de piezas, consiguiendo un primer acercamiento al nuevo diseño realizado sobre MDF (véase la figura 22).

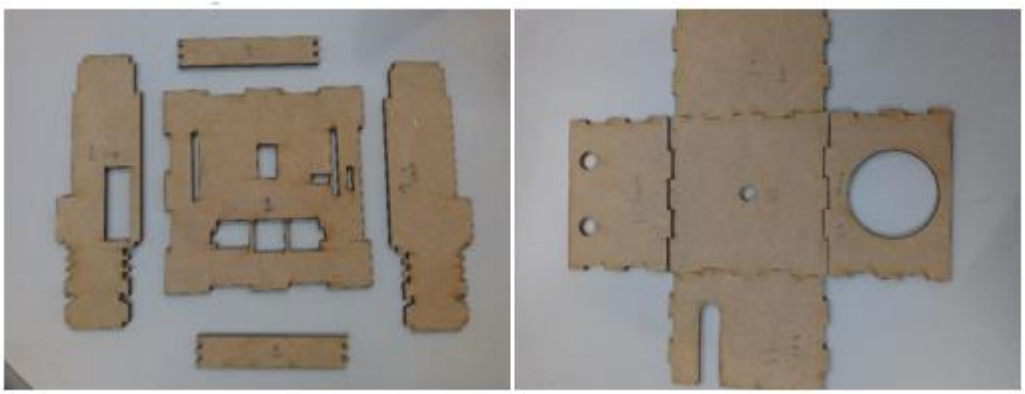


Figura 22: Prototipo 2D con anotaciones

Tras analizar el primer prototipo fueron sugeridos algunos ajustes: Numeración de cada una de las piezas, añadir flechas en algunas de ellas para orientar un sentido de armado y el ajuste de algunas medidas de ensamble.

Concluido el proceso obtuvimos propuestas de dos materiales diferentes cuyas medidas cumplían exactamente con lo esperado para un armado ideal. Las siguientes son las características obtenidas de dichos materiales (véase la Tabla 3):

Tabla 3: Comparativa de materiales alternos propuestos

Material	MDF	Acrílico
Armado	Ensamblado de piezas	Ensamblado de piezas
¿Requiere material extra?	No	No
Tiempo de producción por pieza.	20 minutos	20 minutos

El precio del material también se vio optimizado. El material en MDF terminó con una disminución del 31.6% de su precio respecto del precio de la impresión 3D, mientras que el acrílico con un 46% menos en costo respecto al material en 3D.

La nueva carcasa (véase la figura 23) es la siguiente.

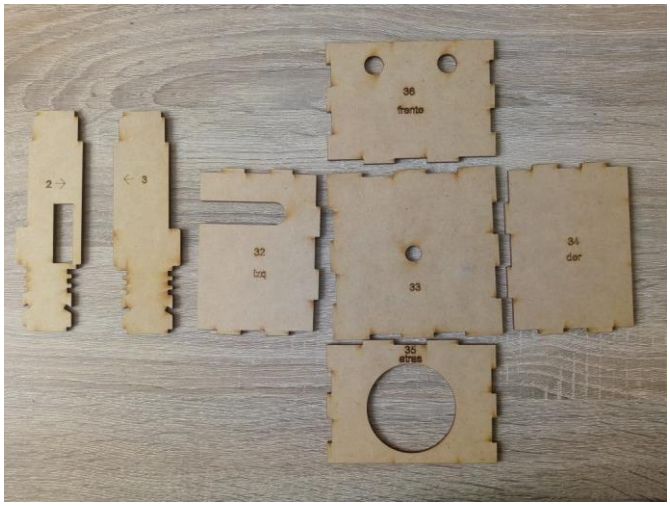


Figura 23: Piezas en MDF de piernas y cabeza del robot

Se logró la optimización de tiempo de ensamble, de producción y de costo de producción para la carcasa, el principal problema a resolver para la parte de componentes físicos.

Este nuevo material es duradero, resiste golpes sin doblarse y puede ser armado varias veces sin sufrir deformaciones; el material cuenta con una numeración de piezas que guiarán a su armado y cuenta además con orificios auxiliares en algunas de sus piezas para que los cables del circuito puedan ser introducidos con facilidad.

El nuevo diseño requiere instrucciones distintas a las de los otros dos materiales ya que las piezas se deben ensamblar, por lo que creé un video (https://youtu.be/Yee2ylksP4U) disponible en línea que narra paso a paso las piezas a ensamblar y las conexiones de los componentes electrónicos para finalizar con el armado completo del robot (véase la figura 24).

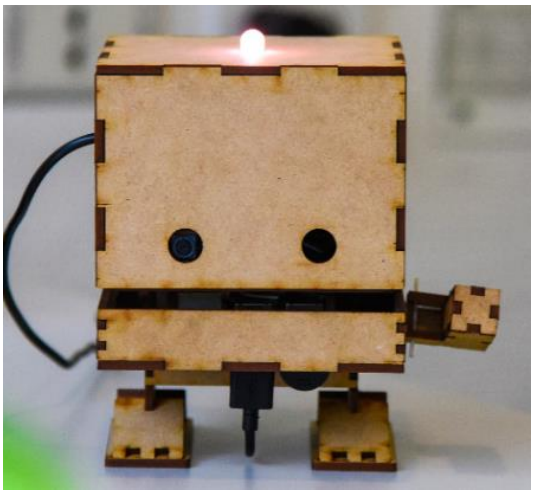


Figura 24: TJBot con material MDF construido

La prueba final de que el armado fue adecuado fue conectarlo a la corriente eléctrica: al momento de hacerlo, la tarjeta encendió dos luces indicativas de corriente eléctrica y lectura de memoria SD, indicando el correcto funcionamiento de la tarjeta. Así mismo, la luz led enciende y el motor emite un ligero ruido indicando que la conexión fue exitosa.

3.3 Componentes de Software.

El sistema operativo con el que el proyecto trabaja es un derivado de Linux y Debian llamado Raspbian, un sistema operativo que ha sido adaptado a la arquitectura de la RPI; este sistema operativo cuenta con interfaz gráfica desde la cual se pueden acceder a todos los ajustes necesarios para realizar una configuración adecuada, además de contar con un acceso rápido a la ventana de comandos, donde tuve que realizar gran parte de las configuraciones necesarias.

Sin embargo, Raspbian no se encuentra por defecto en la memoria SD del kit, por lo que hay que proceder con una instalación manual del sistema operativo. Para poder comenzar con la instalación se deben conectar los siguientes componentes a la tarjeta: Teclado, mouse, cable ethernet, cable HDMI al televisor, cable de corriente y tarjeta microSD con NOOBS preinstalado.

De acuerdo con la página de TJBot creada por el equipo de investigación de la empresa, la instalación del proyecto se lleva a cabo en 3 pasos generales:

1- Instalación del sistema operativo.
2- Actualización de Node-RED.
3- Instalación del proyecto TJBot en Node-RED.

En primer lugar, se conectan los siguientes componentes a la RPI (véase la figura 25):

1- Tarjeta microSD.
2- Cable HDMI conectado a un televisor.
3- Cable LAN Ethernet con conectividad a internet.
4- Teclado USB.
5- Mouse USB.
6- Cable de corriente eléctrica.

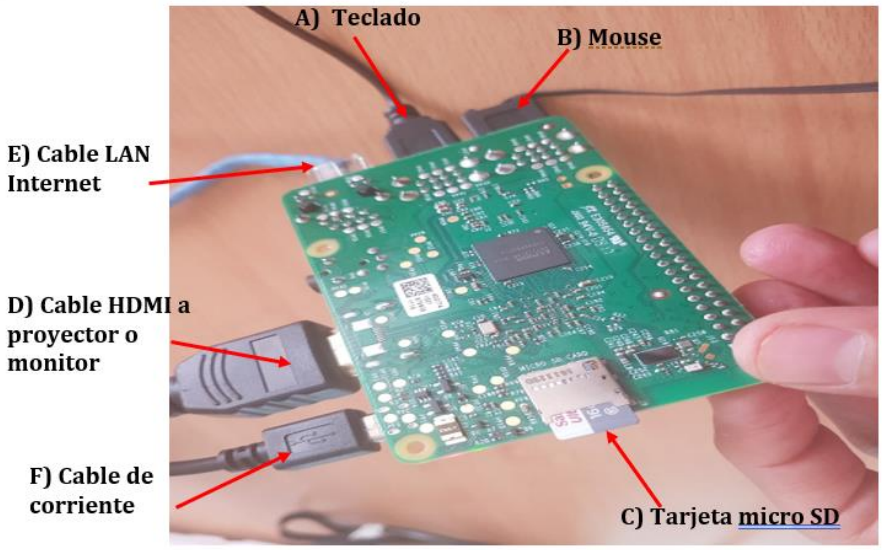


Figura 25: Conexiones iniciales en la RPI

Lo último en conectarse es el cable de corriente, pues tras alimentar de corriente eléctrica a la tarjeta se muestra en el televisor un menú de instalación, por medio del cual es posible instalar algunas opciones de sistemas operativos, entre ellos el sistema operativo Raspbian. Este proceso puede demorar hasta 30 minutos y el internet es opcional ya que NOOBS integra Raspbian cuando es descargado en la memoria.

La siguiente imagen (véase la figura 26) muestra las opciones de sistemas operativos de NOOBS al proveer de corriente eléctrica a la Raspberry.

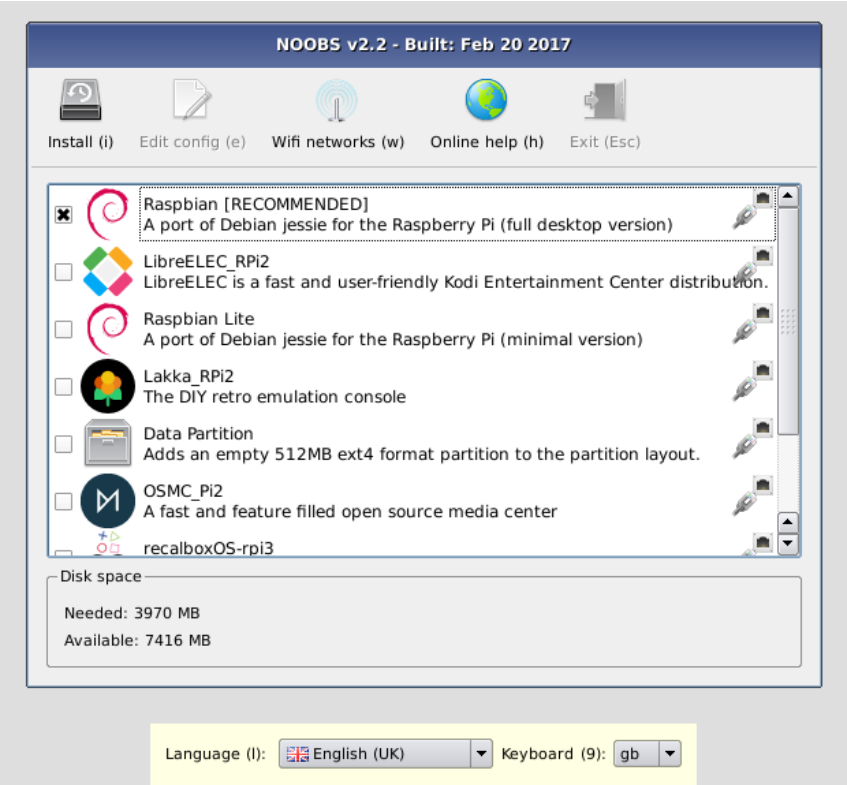


Figura 26: Listado de Sistemas Operativos disponibles en la configuración inicial

Cuando el proceso de instalación del sistema operativo concluye la tarjeta se reinicia y muestra la interfaz gráfica de Raspbian (véase la figura 27). Como todo sistema operativo debe haber una autenticación inicial, en este caso la autenticación se lleva a cabo de forma automática por medio del usuario nombrado "*pi*", cuya contraseña por default es "*raspberry*".

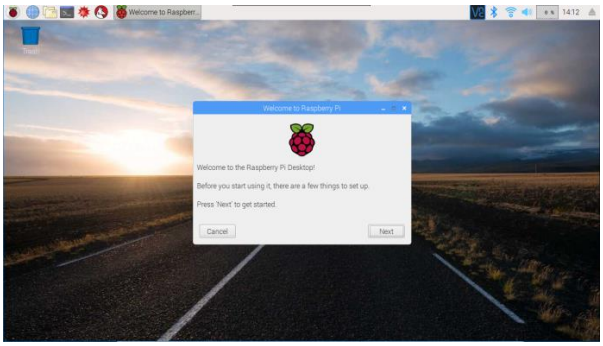


Figura 27: Interfaz gráfica del Sistema Operativo

Con ayuda del mouse y el teclado se realiza el cambio de contraseña del usuario por medio de la interfaz, usando el menú correspondiente en la esquina superior izquierda, en las opciones de configuración de seguridad. A continuación, por medio de un navegador web se accede a la dirección web https://nodered.org y en la página se busca un comando específico que actualiza el programa Node-RED.

El comando se deberá introducir a la ventana de comandos de la RPI (véase la figura 27), dicha ventana de comandos puede ser abierta dando click al ícono de terminal que se muestra en la parte superior de la interfaz.

Figura 28: Ícono de terminal

Una vez abierta la ventana de comandos se debía introducir el siguiente comando que nos sirve para actualizar Node-RED en el sistema:

```
bash < (curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

El comando anterior obtiene de la URL a la que hace referencia un código en lenguaje consola, por medio del símbolo *"menor que"* (el cual funciona como un redireccionamiento del resultado) y que será ejecutado realizando los siguientes cambios al sistema de forma automática.

a) Elimina versiones antiguas de NodeJS, Node-RED.
b) Instala globalmente ambos sistemas en su versión actualizada.
c) Limpia el caché.
d) Instala localmente algunos nodos que por default Node-RED contiene.

Durante la ejecución se muestra en pantalla el procedimiento de actualización que demora alrededor de 20 minutos, una vez concluida la actualización se recomienda reiniciar la RPI. En este punto concluye la actualización de Node-RED.

Finalmente, para llevar a cabo la instalación del proyecto TJBot, basta con mencionar de forma general que se utilizan 13 comandos en la ventana de comandos, dos ediciones de archivos por medio del editor de archivos *Nano* y se accede un repositorio Github para la descarga de archivos auxiliares para el proceso.

Los pasos más importantes del proceso anterior se describen a continuación:

a) Instalación de nodos locales de TJBot para Node-RED usando el repositorio de Github del empleado:

```
git clone https://github.com/JeanCarl/Node-RED-contrib-tjbot
```

En este paso estamos obteniendo de un repositorio en Github los nodos específicos de TJBot para que estén disponibles en nuestra interfaz de programación, estamos alojando estos archivos en una carpeta creada por el usuario.

b) Configuración del servicio Node-RED para ejecutarse con permisos de super usuario *root*.

Recordemos que la autenticación en la interfaz se hace de forma automática con el usuario *pi* y esto es algo que no hemos cambiado, por lo que los comandos que estamos ejecutando son realizados a nombre de dicho usuario.

Por lo anterior el sistema trabaja por default con permisos del usuario pi, los cuales son limitados a comparación de un usuario administrador como lo es *root***.** En este paso se asigna a *root* como usuario dueño del servicio, otorgando privilegios que le permitirán al sistema instalar nuevos nodos y administrar recursos del sistema, entre otros.

c) Modificación de parámetros de Node-RED para nuevos nodos agregados por medio de la ventana de comandos.

Cuando se instalan nodos ajenos a Node-RED por medio de la ventana de comandos es conveniente almacenarlos en una ruta conocida por el usuario, por lo que modificamos algunos parámetros de Node-RED indicándole dónde se encuentran los nodos de TJBot que acabamos de instalar.

Tras realizar estos pasos, reiniciamos la tarjeta RPI y la instalación concluye, permitiendo abrir Node-RED haciendo uso de la ventana de comandos para iniciar el servicio. A continuación se muestra el procedimiento correcto.

1- Abrir la ventana de comandos.
2- Introducir el comando ***node-red-start.***
3- Observar la ventana de información para obtener la dirección IP de TJBot.

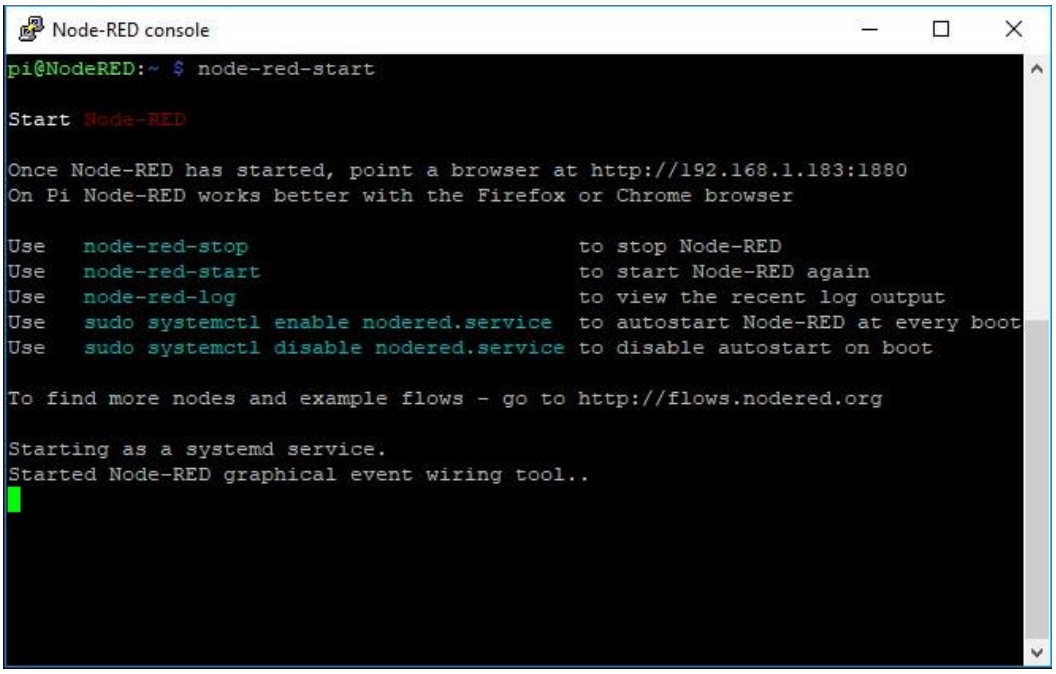


Figura 29: Resultado de iniciar el servicio de Node-RED

La imagen anterior (véase la figura 29) muestra información relativa a el servicio de Node-RED iniciándose, la información contenida puede mostrar errores de instalación y ejecución sobre la marcha de lo programado en la interfaz, e inicialmente sirve para obtener la dirección que hay que colocar en el navegador; ésta es la dirección IP del robot junto con el número de puerto, que en el caso del proyecto es el puerto 1880 (véase la figura 30).

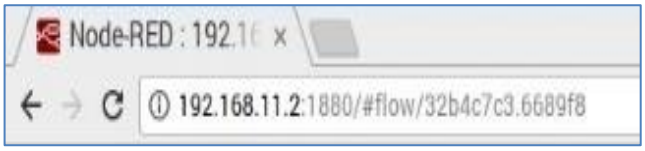


Figura 30: Ejemplo de la dirección IP con número de puerto

Con esta dirección podremos abrir el navegador de internet de TJBot, introducir la dirección proporcionada y acceder a la interfaz gráfica de programación, donde se puede comprobar que la instalación de los nodos de TJBot ha sido exitosa y se encuentran disponibles

3.4 Compatibilidad de los componentes instalados.

La instalación del sistema concluyó sin impedimentos y aunque existían muchos puntos de mejora que identifiqué durante el proceso que debían ser atendidos, decidí continuar comprobando el correcto funcionado de lo recién instalado.

Para poder realizar estas comprobaciones es necesario configurar inicialmente a un *Bot*, un elemento representativo de TJBot que guarda dentro de sí configuraciones como el idioma y las contraseñas de los servicios de la empresa.

Para lograr esto necesitaremos tener creada una cuenta en la nube de la empresa, obtener las credenciales de los servicios de Inteligencia Artificial de WIA, configurar a un Bot y programar las actividades básicas para las cuales TJBot fue diseñado. A continuación se muestra el procedimiento.

3.4.1 Creación de una cuenta y servicios en la nube de la empresa.

La información del proyecto en Github hace referencia a cierta página de la nube de la empresa para obtener una cuenta, sin embargo la dirección no estaba disponible. A finales del año 2018 ocurrió una actualización en la nube de la empresa y una migración de sus servicios a una nueva dirección, por lo que cualquier asunto referente a los servicios de la nube de la empresa deben ser atendidos en la nueva liga.

En esta dirección es visible un botón para crear una nueva cuenta o iniciar sesión, creé una nueva cuenta para llegar al panel de control de la página. Esta página (véase la figura 31) da un panorama del estatus actual de los servicios que se hayan contratado con información relevante, como su uso y el gasto que se ha hecho de ellos, el estatus de la nube por regiones y botones de fácil acceso a otras secciones de la nube.

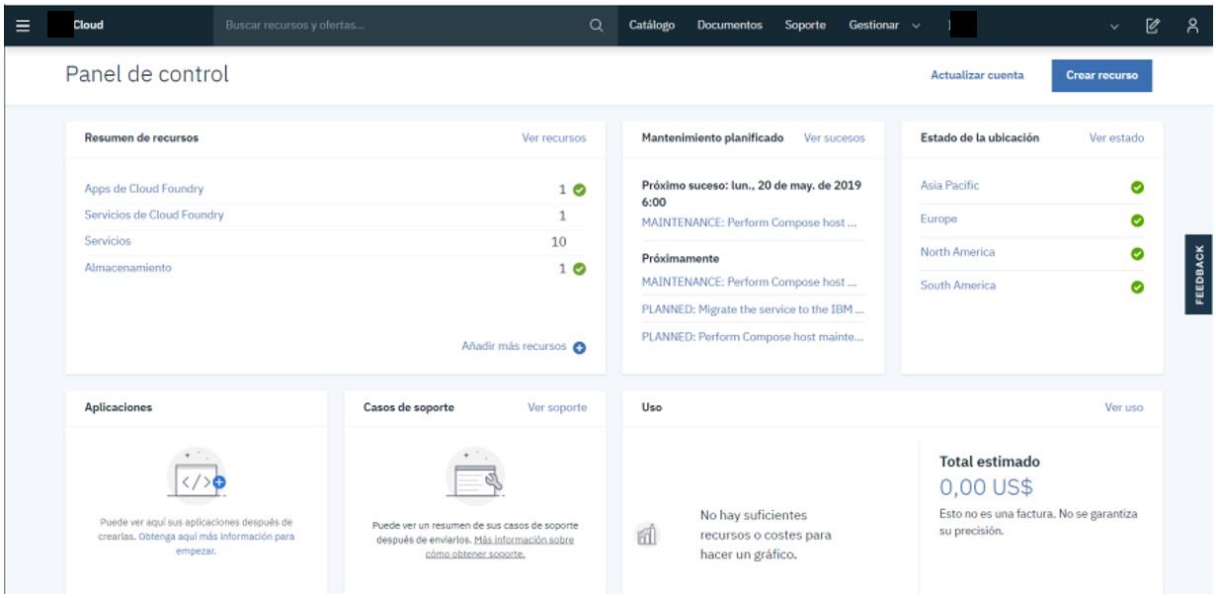


Figura 31: Panel de control de la nube de la empresa

En la parte superior de la pantalla hay un botón llamado catálogo, el cual lista los servicios disponibles que se ofrecen en la plataforma desde la cual es posible contratar los servicios que TJBot requiere.

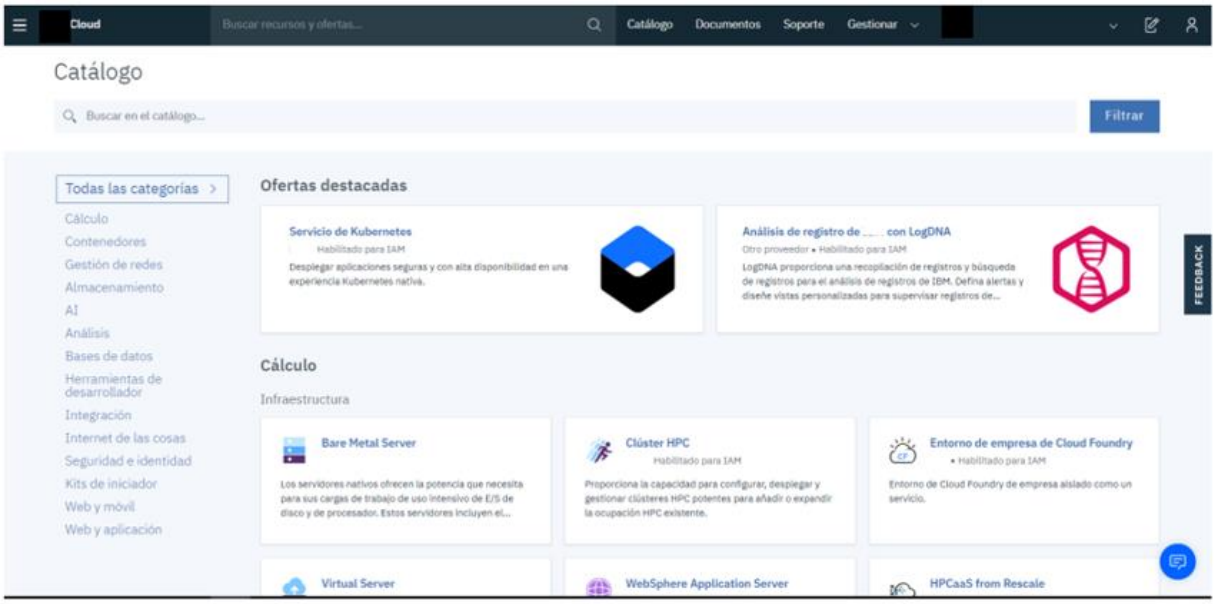


Figura 32: Catálogo de la nube de la empresa

En la vista del catálogo (véase la figura 32), al hacer clic a cualquier servicio que se desee contratar se despliega una descripción general del servicio, documentación y planes de contratación, así como un botón que permite la creación del servicio en sí.

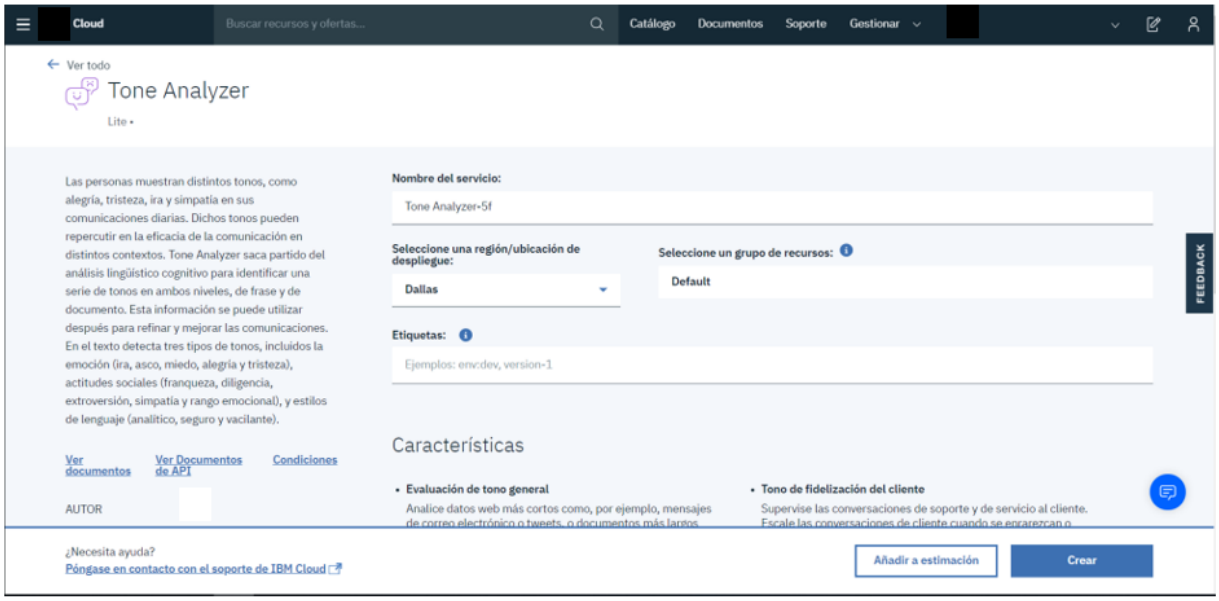


Figura 33: Ventana de creación del servicio Tone Analyzer

Basta con dar click al botón de crear (véase la figura 33)para que el servicio se habilite en nuestra cuenta. Finalmente en la vista siguiente habrá una sección llamada "*Manage*", que al dar click en ella mostrará la clave de API (véase la figura 34).

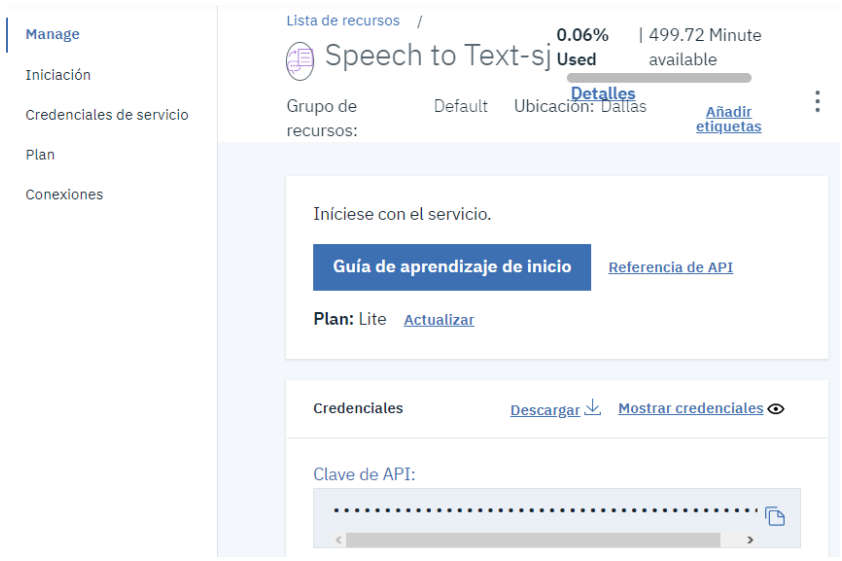


Figura 34: Vista "Manage" para obtener una clave de API

Como se mencionó en los antecedentes, es necesario obtener la clave de API de seis servicios de la nube para que TJBot haga uso de la Inteligencia Artificial de WIA.

a) *Text to speech* (Texto a habla).
b) *Speech to text* (Habla a texto).
c) *Visual Recognition* (Reconocimiento visual).
d) *Tone Analyzer* (Analizador de tonos).
e) *Language translator* (Traducción de lenguaje).
f) *Conversation (Conversación).*

3.4.2 Configuración de un Bot en Node-RED.

La configuración básica de TJBot en la interfaz se realiza configurando un primer Bot. Un Bot es una abstracción del código fuente de TJBot creado por el equipo de investigación de la empresa, en el cual se habilitan los parámetros obligatorios para el correcto funcionamiento del sistema:

a) Género del robot.
b) Lenguaje de habla y escucha.
c) Componentes conectados a la RPI.
d) Nombre del Bot.
e) Credenciales de los servicios de la nube.

Para configurar un *Bot* es importante tener a la mano el nombre de los servicios creados y las credenciales de API generadas y con éstas conectarse a la interfaz de programación de Node-RED en TJBot. Dentro de la interfaz se arrastra un nodo de la categoría TJBot al espacio de trabajo y se da doble click para configurarlo.

La primera opción de configuración indica la selección de un *Bot* (véase la figura 35), en caso de no existir puede crearse uno (o modificar uno ya existente) dando click al ícono de lápiz, como se muestra en la siguiente imagen.

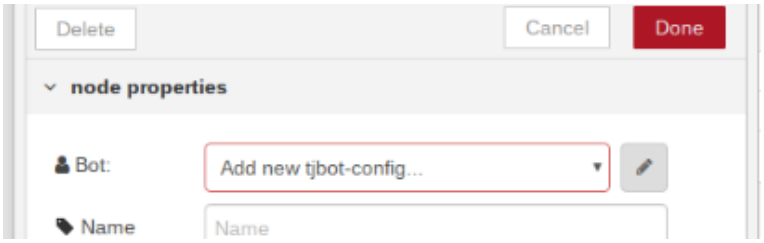


Figura 35: Opción Bot en el menú de configuración

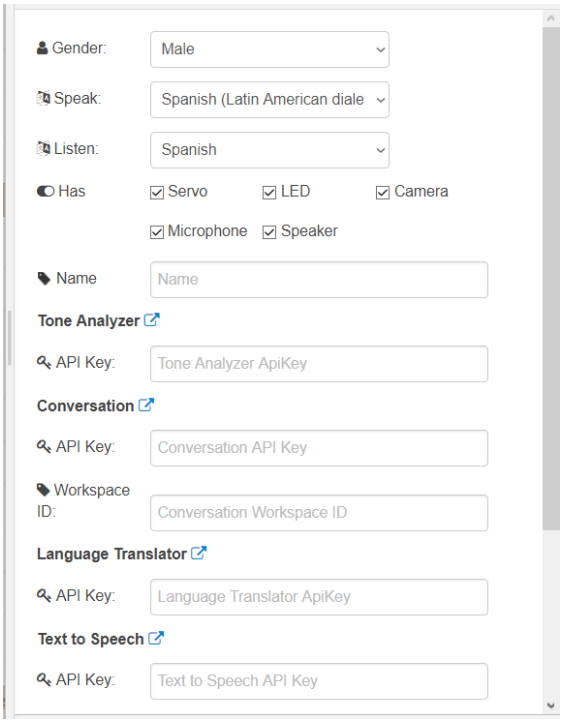


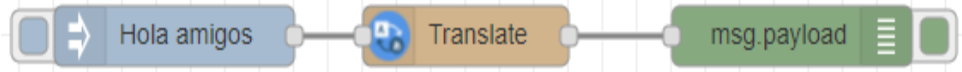
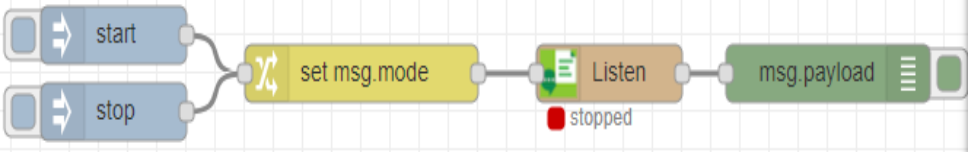
Figura 36: Menú de configuración de un Bot

La configuración de parámetros y servicios del Bot (véase la figura 36) permiten al robot realizar las siguientes actividades:

a) Manipular el LED
b) Manipular el servomotor.
c) Hablar.
d) Traducir.
e) Observar.
f) Escuchar.

Realicé la programación de cada una de las actividades, generando las interconexiones entre los nodos necesarios e implementando la configuración necesaria en cada uno de ellos. El resultado de los flujos generados (véase la Tabla 4) se muestra a continuación.

Tabla 4: Flujos básicos implementados en Node-RED

Nombre de la actividad	Flujo programado
Manipular LED	
Mover el brazo	
Hablar	
Traducir	
Observar	
Escuchar	

La ejecución de cada uno de los flujos trajo diversos resultados, desde satisfactorios hasta fallidos como se muestra a continuación (véase la Tabla 5).

Tabla 5: Resultado de pruebas realizadas a los flujos básicos

Actividad	Resultado	Éxito
Manipular LED	El led cambia de color.	Si
Manipular servomotor	El brazo se mueve.	Si
Hablar	El robot habla.	No (Falla aleatoriamente)
Traducir	Nada ocurre.	No
Observar	Nada ocurre.	No
Escuchar	Nada ocurre.	No

Los resultados obtenidos de las primeras tres pruebas de la tabla 5 sugerían el correcto funcionamiento del led, servomotor y de la bocina. En el caso de la actividad donde TJBot habla se comprobó que la autenticación por medio de la llave de API y la conectividad con la nube fue correcta, pero existía un fallo aleatorio donde el sonido en ocasiones funcionaba y en otras ocasiones no.

Por otro lado, las actividades de traducir, observar y escuchar no mostraron resultados en la ventana debug, ni algún cuadro informativo que revelara algún tipo de error, por lo que era importante indagar dentro del código fuente de TJBot haciendo uso de la ventana de comandos para identificar la raíz del problema.

3.5 Problemáticas y soluciones: Componentes de Software.

Para llevar al proyecto a un nivel óptimo de funcionalidad y éste pudiera ser replicado con fiabilidad, era necesario que sus componentes funcionaran individualmente y en conjunto, por lo que era indispensable que los nodos TJBot funcionaran adecuadamente.

Lograr esto desde un enfoque de Software involucró abordar dos problemáticas:

a) Optimizar el proceso de instalación de TJBot.

Esto debería reducir el tiempo de instalación, reducir el riesgo que involucra que el usuario inexperto introduzca comandos de forma errónea, reducir el tiempo de instalación y llevar a cabo una instalación lo más limpia posible.

b) Lograr la compatibilidad de los componentes de software.

Que los flujos programados fallen de forma esporádica es inadmisible, era necesario revisar a profundidad el funcionamiento de cada componente y conseguir que funcionaran adecuadamente bajo cualquier circunstancia.

En los siguientes párrafos se abordará la solución a los puntos anteriores.

3.5.1 Optimizar el proceso de instalación.

La interacción directa con la ventana de comandos de Raspbian no es una actividad familiar para el usuario inexperto en este tipo de tecnologías. Esto supone un claro riesgo de una instalación incompleta, donde el usuario podría ser incapaz de identificar el paso en el que ha fallado y carecer de conocimientos para resolver el conflicto; el único camino que este usuario podría tomar en caso de no contar con asesoría sería formatear la memoria y comenzar nuevamente el proceso de instalación desde un inicio.

Se plantearon dos soluciones para este problema. La primera solución implicó un proceso de instalación más sencillo (pero de forma manual) para el usuario con experiencia que le permitiría realizar modificaciones a los componentes instalados y comprender el funcionamiento interno. La segunda solución implicó una copia idéntica del sistema operativo de TJBot ya configurado alojada en internet, útil para el usuario inexperto que no requiere profundizar en la instalación.

Por lo anterior se eligió desarrollar ambas soluciones. Para el usuario inexperto se realizó en una memoria el procedimiento completo de instalación de TJBot, se realizó un respaldo de la memoria por medio de un programa de grabado de memorias llamado "*Win32 disk imager*" y de esto se obtuvo un archivo con formado "*.img*", el cual fue comprimido y alojado en la nube para fácil acceso.

La segunda solución era más compleja de resolver, se requería de un proceso de simplificación y automatización de la instalación, que a su vez fuera transparente para cualquier desarrollador que deseara personalizar el procedimiento.

Basado en esta premisa la solución resultó en un script de Shell que ejecuta de manera automática los comandos de configuración necesarios para el ambiente de TJBot. Un script de Shell es un programa de computadora para ser ejecutado en un sistema operativo como Linux. Para lograr esto fue necesario encontrar el equivalente de los pasos ejecutados paso a paso en la ventana de comandos en su respectivo código en Shell.

Para llevar a cabo esta equivalencia, algunos de los comandos introducidos por el usuario manualmente deben ser sustituidos por código equivalente que tenga el mismo efecto. Este cambio radica en que algunos comandos requieren intervención directa del usuario de forma interactiva, pero si se desea un script más automatizado se requiere de modificar parámetros de configuración del sistema sin intervención.

Hice una recopilación de los comandos que utilicé en un inicio para configurar inicialmente a TJBot y los clasifiqué en dos: Aquellos que requieren de intervención directa del usuario y aquellos que no. Prestando particular atención a los comandos que requerían intervención del usuario me encontré con estas operaciones principales:

1- Actualización de NodeJS y Node-RED:

El usuario introducía el siguiente comando:

```
bash <(curl -sL https://raw.githubusercontent.com/Node-RED/raspbian-deb-package/master/resources/update-nodejs-and-Node-RED)
```

Este comando instalaba ambas paqueterías mostrando el progreso en pantalla, pidiendo al usuario algunas confirmaciones y pidiendo el reinicio de la RPI.

2- Descarga e instalación de proyecto TJBot.

Requería la navegación entre distintos directorios, la creación de algunos de ellos y la edición de archivos.

3- Configuración de servicios.

Requería de la modificación del estado del servicio Node-RED y la configuración de algunos parámetros internos de NodeJS y Node-RED.

Evitaré colocar aquí el código anterior pero cabe destacar que algunos comandos en este código requerían de ser ejecutados por el usuario *pi*, debido a que es este el usuario con el que el sistema se autentica al encender y es el usuario que iniciará el servidor Node-RED. A su vez, algunos comandos de instalación requerían de ser ejecutados por el usuario *root* pues modifica agrega y elimina archivos en el sistema. En algunos de los siguientes comandos podremos apreciar el cambio automático entre usuarios para realizar instalaciones con diferentes niveles de permisos.

Las operaciones anteriormente listadas fueron optimizadas a través de los siguientes códigos.

1- Nuevo código para la actualización de NodeJS y Node-RED.

```
git clone https://github.com/Node-RED/raspbian-deb-package.git
cd raspbian-deb-package/resources/
su pi -c " ./update-nodejs-and-Node-RED"
```

El código anterior obtiene el código de actualización directamente del proyecto Node-RED alojado en Github, lleva el estado de ejecución del script a la carpeta donde se encuentra el ejecutable de la actualización y por último, por medio del comando "*su pi -c*", cambia temporalmente de permisos de ejecución para instalar como el usuario *pi* los archivos.

2- Nuevo código para la descarga e instalación del proyecto TJBot.

```
node-RED-stop
cd /home/pi/.Node-RED
mkdir /home/pi/.Node-RED/nodes
cd /home/pi/.Node-RED/nodes
git clone https://github.com/JeanCarl/Node-RED-contrib-tjbot
cd /home/pi/.Node-RED/nodes/ Node-RED-contrib-tjbot
npm install –unsafe-perm
```

Esta serie de comandos navegan entre carpetas, creando la estructura de directorios requeridos e integra a TJBot al sistema de Node-RED. El último comando es el encargado de la instalación de los nodos TJBot.

La ejecución del último comando con la bandera *unsafe-perm* activa se debe a que ciertos paquetes del proyecto requieren ser instalados con permisos propios del usuario privilegiado *root* y con esta bandera habilitamos la instalación de cada paquete con dichos permisos. En caso de que esta bandera no sea activada la instalación no podrá culminar, pues deriva en una serie de incompatibilidades que deshabilitan los canales de comunicaciones PWM de los pines de TJBot y comienza por inhabilitar el sonido de la bocina, el led y el servomotor.

3- Nuevo código para la configuración de servicios.

```
sed -i -e 's/User=pi/User=root/g' /lib/systemd/system/Node-RED.service
systemctl daemon-reload
systemctl enable Node-RED.service
node-red-start &
echo "Iniciando servidor"
sleep 10
node-red-stop
echo "Deteniendo servidor"
sleep 5
sed -i -e "s/\/\/userDir: '\/home\/nol\/.node-red\/'/userDir: '\/home\/pi\/.node-red\/'/g" /root/.node-red/settings.js
sed -i -e "s/\/\/nodesDir: '\/home\/nol\/.node-red\/nodes',/nodesDir: '\/home\/pi\/.node-red\/nodes',/g" /root/.node-red/settings.js
```

Para la configuración de servicios hice uso del comando *sed*, el cual permite editar el contenido de ficheros de forma no interactiva basado en las banderas activadas dentro de los parámetros del comando. Primero se cambia el usuario que es dueño del servicio de Node-RED por el usuario *root*, lo cual permitirá ejecutar este servicio con privilegios dentro del sistema. Seguido de esto se añade el servicio de Node-RED a la lista de servicios que iniciarán cuando la RPI cargue su sistema operativo, se reinicia el servidor, se indica a Node-RED la ruta de los nodos recién instalados y se añaden a la paleta de nodos.

Por último, se realizan cambios en el funcionamiento del hardware, primero habilita la cámara y luego define la salida de sonido default por medio del id 3.

Existen 3 identificadores posibles para utilizar: El número 1 es sonido por medio del Jack, el número 2 emite sonido por medio de HDMI y el número 3 toma la salida de audio USB por default.

```
echo "start_x=1" | tee -a /boot/config.txt
amixer cset numid=3 0
reboot now
```

El script resultante, con nombre *Bootstrap.sh*, fue guardado con permisos de ejecución en entornos Raspbian y luego alojado en Github para ser instado a través de pocos comandos por cualquier TJBot.

Para comprobar la correcta funcionalidad de este script basta con la ejecución de los siguientes 3 comandos en una ventana de comandos de Raspbian:

```
git clone https://github.com/JairLizarraga-/tjbot_setup
cd tjbot_setup
sudo ./bootstrap.sh
```

Esto es posible gracias a que se ha eliminado toda interacción del usuario con los parámetros repetitivos de instalación que anteriormente se requerían, éstos se concentran en una paquetería que se descarga desde Github y simplemente se ejecuta con permisos de super usuario *root.*

Al ejecutar el tercer comando *sudo* el sistema automáticamente comienza con la descarga de paqueterías, configuración de parámetros y finalmente se reinicia.

Podemos comprobar que la instalación de TJBot fue exitosa porque antes de la instalación la interfaz Node-RED sólo reflejaba los nodos que por default se encontraban en la paquetería; ahora, tras ejecutar los comandos propuestos para la instalación automática, TJBot se reinicia automáticamente, al acceder a la interfaz Node-RED se puede apreciar en la paleta de nodos la categoría de nodos de TJBot para hacer uso de ellos.

3.5.2 Lograr la compatibilidad de componentes de software.

Recordemos que las incompatibilidades se reflejaron en la falla intermitente del sonido y en la ejecución de algunos flujos que hacen uso de los servicios de WIA. Al no haber un error presente en la interfaz de trabajo es necesario profundizar poco a poco en las capas que componen el proyecto para dar con el problema, el cual abarca componentes tanto de software como de hardware.

a) Sonido intermitente.

Inicialmente comprobé que las configuraciones de TJBot fueran correctas y que la bocina funcionara con normalidad. Una vez comprobado debí verificar una posible incompatibilidad entre el proyecto TJBot en alguno de sus componentes y la bocina. Para esto existe un comando que nos permite visualizar los dispositivos de sonido conectados al sistema: *aplay -l.* Cuando realicé el escaneo de dispositivos conectados a la RPI se detectaba de forma intermitente la bocina USB.

Experimenté habilitando y deshabilitando los componentes del proyecto TJBot poco a poco hasta encontré el patrón que definía dicho comportamiento: La bocina funcionaba correctamente cuando el servicio de Node-RED no había sido iniciado, pero cuando éste inicia, la bocina comienza a tener intermitencias. Esto significaba que alguna de las librerías que Node-RED era incompatible con la bocina de alguna manera.

Al indagar el componente del proyecto que podría ser incompatible con el funcionamiento de la bocina descubrí que existe una relación entre las paqueterías que habilitan el LED de TJBot y el funcionamiento de la bocina, la cual es la capacidad de la RPI para generar modulación por ancho de pulsos, también llamada PWM por sus siglas del inglés.

La tarjeta RPI utiliza PWM para diversos propósitos, en el caso del proyecto TJBot utiliza PWM para generar pulsos de diferentes duraciones, mandar estos pulsos al led y configurar los diferentes niveles de brillo y colores. Por otro lado, cuando el sistema operativo comienza su carga inicializa todas las salidas de audio y video que no han sido explícitamente bloqueadas. En el caso de las salidas de audio se cuentan con tres: Audio por HDMI, audio por Jack 3.5mm y audio por USB. Todas estas salidas están habilitadas e inicialmente solo se configuraba el sistema operativo para emitir sonido por default por USB, pero esto no impedía que se inicialicen las librerías necesarias para emitir sonido por cualquiera de las otras dos salidas. Aquí es donde se originaba el problema ya que la salida de audio Jack 3.5mm es una salida analógica, la cual hace uso de PWM para generar sonido.

Es entonces cuando se encontró el origen del problema. Cuando se iniciaba el sistema operativo, en ocasiones se inicializaba la salida de sonido por el puerto Jack aun cuando ésta no fuera a ser utilizada y cuando el servicio de Node-RED iniciaba el led, ambos buscaban hacerse del control del generador de pulsos PWM y es cuando la intermitencia aparecía.

Para resolver esta problemática bloquee la activación de la tarjeta de sonido del Jack. Esta configuración se logró por medio de la creación de un archivo que lleva por nombre *sound.blacklist.conf*, su contenido es solamente la siguiente línea de texto:

blacklist snd_bcm2835

Esta línea indicará al sistema operativo que debe incluir en su lista negra a la tarjeta de sonido *bcm2835*. Esta se encarga de inicializar el componente de audio análogo por medio del Jack. Para que esto haga efecto el archivo debía ser alojado en la ruta */etc/modprobe.d/sound.blacklist.conf/*.

Copié el archivo en la ruta indicada y reinicié la RPI. Una vez iniciada, por medio del comando "*aplay -l*" listé las tarjetas de sonido que el sistema operativo detectaba y había cargado y el resultado fue satisfactorio: El sistema operativo cargó únicamente la tarjeta de sonido relacionada a la bocina USB conectada a la RPI.

A partir de estos cambios realizados el sistema operativo evita inicializar la tarjeta de sonido del Jack mientras inicializa todos los componentes y busca de forma automática la tarjeta de sonido de la bocina USB, logrando así resolver la incompatibilidad entre la tarjeta de sonido con el proyecto y activando en todos los casos la bocina USB por encima de las otras tarjetas de sonido. Con esto no hay incompatibilidad entre la bocina contra cualquier componente del sistema.

b) Funcionamiento de actividades básicas.

De acuerdo con los resultados de las pruebas realizadas sobre los componentes, el hardware conectado a los pines GPIO de la tarjeta funcionaron en su totalidad: La luz led y el servomotor. Tras la resolución de la incompatibilidad de la bocina el robot reproduce sonidos cuando se le solicita. Ahora es momento de resolver el problema con las otras actividades básicas tales como traducción, escucha y visión, las cuales fallaban sin explicaciones visibles sobre la interfaz.

Para determinar la raíz del problema realicé una conexión al servidor Node-RED por medio de la ventana de comandos para analizar la información arrojada cada que un flujo se ejecutaba.

La información obtenida de la ejecución del servidor fue mucho más detallada que la que se obtuvo de la interfaz gráfica, pues cada proceso que los flujos ejecutaban mostraban un rastro informativo que podían apreciarse a través de la ventana de comandos. En este punto pudieron apreciarse tanto los procesos exitosos como los procesos fallidos.

El servidor Node-RED es un conjunto complejo de módulos y paqueterías JavaScript que fueron unidos para dar vida al sistema, de hecho generalmente así se construyen los sistemas basados en NodeJS. Uno de estos módulos de los que Node-RED se integra es un módulo *logger*, lo cual se puede traducir al español como un módulo de registro. Este módulo administra los mensajes importantes del programa en ejecución y los guarda en un registro, además de que los muestra en la ventana de comandos de acuerdo con la configuración que tenga dicho *logger*.

De acuerdo a la documentación oficial de Node-RED (https://nodered.org/docs/user-guide/runtime/logging) una de estas configuraciones permite "niveles" de logs generados en el sistema, con base en la siguiente jerarquía:

a) *Fatal*: Sólo muestra errores que impiden el funcionamiento del sistema.
b) *Error*: Sólo muestra errores que son considerados fatales para ciertas peticiones.
c) *Warn*: Muestra problemas que no son fatales.
d) *Info*: Muestra información sobre la ejecución general de la aplicación.
e) *Debug*: Muestra información más detallada que el modo *Info*.
f) *Trace*: Muestra un log realmente detallado.
g) *Off*: No muestra logs.

Para asegurarme de observar cada detalle que el sistema pudiese arrojar configuré el logger en su nivel máximo de detalle, *trace*, para poder observar la información correspondiente a los procesos que fallaban en las programaciones básicas.

Con este log preparado, fue momento de reiniciar manualmente el servicio Node-RED; para esto se utilizaron los comandos node-red-start y node-red-stop respectivamente. Al arranque, la información (véase la figura 37) fue la siguiente:

```
pi@raspberrypi:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.100.32:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use   node-red-stop                         to stop Node-RED
Use   node-red-start                        to start Node-RED again
Use   node-red-log                          to view the recent log output
Use   sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use   sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
Started Node-RED graphical event wiring tool.
12 Nov 02:57:28 - [info]
Welcome to Node-RED
===================
12 Nov 02:57:28 - [info] Node-RED version: v0.19.5
12 Nov 02:57:28 - [info] Node.js  version: v10.15.0
12 Nov 02:57:28 - [info] Linux 4.14.50-v7+ arm LE
12 Nov 02:57:29 - [info] Loading palette nodes
12 Nov 02:57:34 - [info] Dashboard version 2.15.5 started at /ui
12 Nov 02:57:34 - [info] Settings file  : /root/.node-red/settings.js
12 Nov 02:57:34 - [info] HTTP Static    : /tmp
12 Nov 02:57:34 - [info] Context store  : 'default' [module=memory]
12 Nov 02:57:34 - [info] User directory : /home/pi/.node-red/
12 Nov 02:57:34 - [warn] Projects disabled : editorTheme.projects.enabled=false
12 Nov 02:57:34 - [info] Flows file     : /home/pi/.node-red/flows_raspberrypi.json
12 Nov 02:57:34 - [info] Server now running at http://127.0.0.1:1880/
12 Nov 02:57:34 - [warn]
---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------
12 Nov 02:57:35 - [info] Starting flows
info: Hello from TJBot! My name is TJBot.
info: Hello from TJBot! My name is Jair.
12 Nov 02:57:35 - [info] Started flows
```

Figura 37: Inicio del servidor Node-RED

De los datos generados se puede destacar el puerto en el que se encuentra iniciado el servidor, la dirección web que se requiere contactar, el nombre del robot y una confirmación del estatus de iniciado.

Cuando el servidor concluye su carga éste permanece en modo espera hasta que cualquier flujo sea activado. Inicié el flujo de cambio de color del led que funcionó sin problemas y la ventana mostró la siguiente información (véase la figura 38):

```
info: Hello from TJBot! My name is TJBot.
info: Hello from TJBot! My name is Jair.
info: TJBot shining my LED to RGB color #000000
12 Nov 03:12:02 - [info] Started flows
info: TJBot shining my LED to RGB color #FFFFF0
```

Figura 38: Resultado de la ejecución del flujo de cambio de color.

Los datos generados describen el estatus al cual el led cambió, mostrando que los logs se muestran como se esperaba.

Ahora se ejecuta el flujo (véase la figura 39) que permitiría a TJBot escuchar utilizando su micrófono y que anteriormente había fallado:

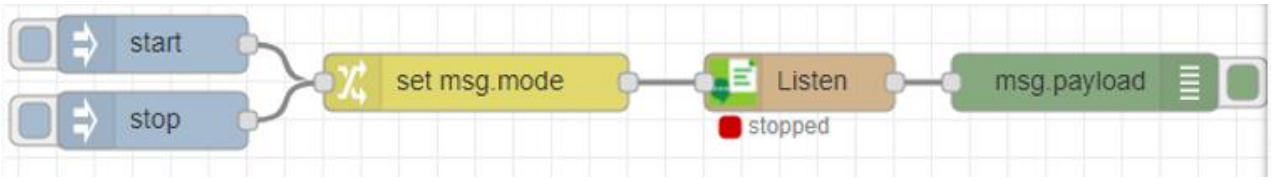


Figura 39: Flujo para iniciar la función de escucha

Al iniciar la actividad se apreciaban líneas de información en la ventana de comandos que hacían referencia al error que no se pudo apreciar por otros medios. La ventana arrojó muchos detalles que no necesariamente aportaban información útil, pero pude destacar una línea importante donde se podía leer "*Provided API Key could not be found*". Para comprobar si este era el origen del error verifiqué la API Key que se ingresaba a la aplicación contra la API Key obtenida de la nube de la empresa y pude constatar que era la correcta.

Esto sólo podía significar que el error se originaba internamente en alguna parte del código fuente del proyecto, por lo que era necesario realizar un debug paso por paso del algoritmo del proyecto hasta dar con el problema y explorar específicamente la parte del código donde la API Key es utilizada.

Para explorar esta parte es conveniente explicar las capas que componen al proyecto TJBot a nivel de código, distinguiendo entre los archivos más importantes para el funcionamiento del proyecto: El código relacionado a los nodos de TJBot y el código fuente de TJBot.

Los archivos del código correspondiente a los nodos se encuentran en el directorio: *"/home/pi/.Node-RED/nodes/nodes-tjbot-latam/tjbot",* por otro lado, el código relacionado al código fuente está en: *"/home/pi/.Node-RED/nodes/nodes-tjbot-latam/node_modules/tjbot/lib/tjbot.js".* En conjunto ambos permiten controlar a TJBot desde la interfaz, sin embargo cada uno representa una capa de abstracción para reducir la complejidad de la codificación a los desarrolladores y usuarios finales.

La estructura interna del proyecto TJBot puede representarse como una serie de capas (véase la figura 40), las cuales no pueden funcionar sin su antecesora y gráficamente podemos entenderla de la siguiente manera.

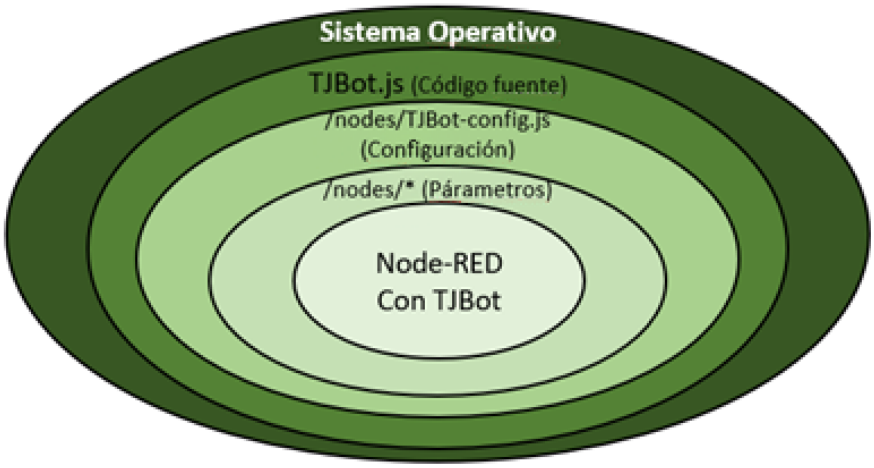


Figura 40: Capas de funcionamiento del proyecto

Hasta este punto se exploró Node-RED en su entorno gráfico sin encontrar el origen del error, seguido de esto se exploró el servidor Node-RED por medio de la ventana de comandos y si bien conseguí encontrar un mensaje de error, no había manera de resolverlo desde esta capa de procesamiento. Fue hora de explorar la capa antecesora a Node-RED: El código de construcción individual de los nodos de TJBot.

Para esta exploración continué utilizando como referencia el flujo de escucha que hace uso del servicio *text to speech* que el nodo *listen* utiliza y de esta forma explorar los dos archivos que construyen internamente al nodo: *listen.js* y *listen.html*. El archivo en formato HTML codifica la manera en que visualmente se mostrará el nodo en la interfaz gráfica (color, forma, imagen y campos de configuración) por lo que su contenido era irrelevante para el propósito actual.

Para profundizar en el origen del error los archivos de código JavaScript se volvieron el punto focal ya que es el código que hace uso del código fuente TJBot para funcionar; a su vez, debido a que me encontraba en la capa de abstracción de los nodos, este código tenía una estrecha relación con la interfaz gráfica, por lo que este archivo contenía declaraciones propias de Node-RED que se encargaban principalmente de las siguientes funciones:

a) Instanciar el nodo en la interfaz.
b) Definir el manejo y estructura de la información de entrada y de salida del nodo.
c) Haciendo uso de la configuración implementada por el usuario, realizar operaciones sobre los datos y hacer llamados a las funciones del código fuente.

De estas tres características, la atención se dirigió a las líneas de código que involucraban a los llamados al código fuente, dónde una línea de código en particular destacaba en importancia en la línea 41 del código:

```
tj.bots[botId].listen(function(text) {broadcastText(botId, text);});
```

La sección *tj.bots[botId]* identificaba el *Bot* que estaba llevando a cabo la configuración y que a continuación utilizaba el método listen que hacía uso de los parámetros posteriores para funcionar.

Ésta era la línea de código que iniciaba el escucha en el robot. Cuando implementé una función *debug* para mostrar la información que resultaba de iniciar el escucha se mostró que la actividad no arrojaba resultados tras la activación, una frase en inglés que explicaba que "la función listen no pudo ser activada exitosamente". Por lo tanto, el fallo se encontraba en el método *listen*, método que se encuentra en la variable *tj*. El valor de esta variable se definía en las primeras líneas de código:

```
var tj = require("./tjbot.js");
```

Esta línea de código importaba una librería llamada *tjbot.js*, que hacía referencia a la capa de abstracción del código fuente de TJBot (véase la figura 41).

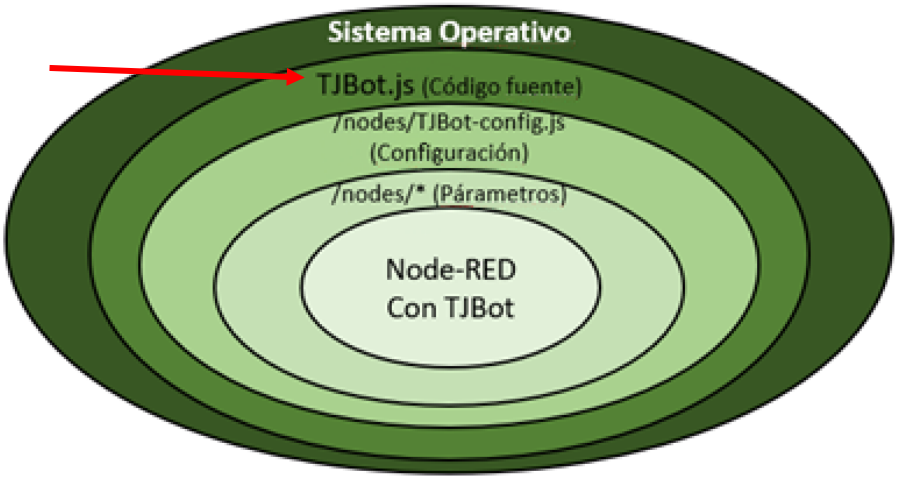


Figura 41: Capa de funcionamiento que arroja resultados erróneos

El archivo TJBot.js (véase figura 41) se encontraba en la ruta *"/home/pi/.Node-RED/nodes/Node-RED-contrib-tjbot/node_modules/tjbot/lib/tjbot.js*" y su codificación cumple lo siguiente:

a) Realizar el levantamiento de componentes de hardware (cámara, led, micrófono, servomotor y bocina).
b) Realizar la autenticación con los servicios de la WIA en la nube (*Conversation, Language Translator, Text to Speech, Speech to Text, Visual Recognition, Tone Analyzer*).
c) Configurar el idioma y componentes del Bot creado.
d) Definir los métodos que harán uso de los servicios de la WIA.

De estas 4 funciones principales, el origen del fallo se encontraba en la definición de métodos, particularmente en la definición del método *listen* que hacía uso del servicio *speech to text* (véase la figura 42).

```
747  TJBot.prototype.listen = function(callback) {
748      // make sure we can listen
749      this._assertCapability('listen');
750
751      // capture 'this' context
752      var self = this;
753
754      // (re)initialize the microphone because if stopListening() was called, we don't s
755      // be able to re-use the microphone twice
756      this._setupMicrophone();
757
758      // create the microphone -> STT recognizer stream
759      // see this page for additional documentation on the STT configuration parameters:
760      // https://www.ibm.com/watson/developercloud/speech-to-text/api/v1/#recognize_audi
761      var params = {
762          content_type: 'audio/l16; rate=44100; channels=2',
763          interim_results: true, // need 'true' for watson-developer-cloud 3.x, otherwis
764          model: this.configuration.listen.language + '_BroadbandModel'
765      };
766
767      if (this.configuration.listen.inactivityTimeout != undefined) {
768          params.inactivity_timeout = this.configuration.listen.inactivityTimeout;
769      }
770
771      winston.silly('recognizeUsingWebSocket() params:');
772      winston.silly(params);
773
774      this._recognizeStream = this._stt.recognizeUsingWebSocket(params);
775
776      // create the mic -> STT recognizer -> text stream
777      this._sttTextStream = this._micInputStream.pipe(this._recognizeStream);
778      this._sttTextStream.setEncoding('utf8');
779
780      // handle errors in the text stream
781      this._sttTextStream.on('error', function(err) {
782          if (err) {
783              winston.error('the speech_to_text service returned an error.', err);
784
785              // stop microphone
786              self.stopListening();
787
788              // attempt to reconnect
789              self.listen(callback);
790          }
791      });
792
793      // deliver STT data to the callback
794      this._sttTextStream.on('data', function(transcript) {
795          winston.info('TJBot heard: ' + transcript);
796
797          if (callback != undefined) {
798              callback(transcript);
799          }
800      });
801
802      // start the microphone
803      this._mic.start();
804  };
```

Figura 42: Definición (en código) del método de escucha

Sobre este código realicé una comprobación paso por paso para verificar en que parte del código las cosas dejaban de funcionar adecuadamente. Para llevar a cabo esto utilicé la

función *console.log()* de JavaScript, para así mostrar en consola el valor de algunas variables para comprobar que su contenido fuese el esperado.

A lo largo de las pruebas pude comprobar que el micrófono se detectaba y se activaba correctamente. Comprobé que la capacidad de escucha del robot estuviera habilitada con el método *_assertCapability*. El error fue descubierto al comprobar la línea número 744, donde se ejecutaba la siguiente instrucción.

```
this._recognizeStream = this._stt.recognizeUsingWebSocket(params);
```

El resultado de la operación anterior estaba trayendo como resultado un valor erróneo, pues la variable *_stt* fallaba en su operación cada que se le ejecutaba. La definición de esta variable se encontraba en la autenticación del servicio *speech to text*, un código breve que consistía en lo siguiente (véase la figura 43).

```
case 'speech_to_text':
    assert(credentials.hasOwnProperty('username'), "credentials for the " + service + " service missing 'username'");
    assert(credentials.hasOwnProperty('password'), "credentials for the " + service + " service missing 'password'");

    var SpeechToTextV1 = require('watson-developer-cloud/speech-to-text/v1');
    this._stt = new SpeechToTextV1({
        username: credentials['username'],
        password: credentials['password'],
        url: 'https://stream.watsonplatform.net/speech-to-text/api/',
        version: 'v1'
    });
    break;
```

Figura 43: Método de autenticación del servicio Speech to Text

Las anteriores líneas (ver figura 43) instanciaban a un objeto que contenía funciones y conexiones directa con el servicio *Speech to Text* de la nube de la empresa. Este objeto se crea automáticamente cuando nosotros agregamos nuestra llave de API de *Speech to Text* en la configuración de TJBot.

De acuerdo con esta codificación la autenticación definida para este servicio hacía uso de dos valores introducidos por el usuario: usuario y contraseña, sin embargo se había introducido una llave de API para llevar a cabo la autenticación.

Esta discordancia se debía claramente a que el código no había sido actualizado para la última versión de la nube de la empresa. Esta actualización fue uno de los cambios más grandes que ha tenido la plataforma, e impactó los métodos de autenticación requeridos para sus servicios, los cuales pasaron de utilizar el método de usuario y contraseña al método de llaves de API.

Tras ahondar en la documentación referida por la nube acerca del servicio de Speech to Text, en el apartado de autenticación, pude encontrar la siguiente leyenda: *"Cloud is migrating*

to token-based Identity and Access Management (IAM) authentication.”, a su vez encontré el siguiente ejemplo de autenticación (véase la figura 44):

```
var SpeechToTextV1 = require('watson-developer-cloud/speech-to-text/v1');
this._stt = new SpeechToTextV1({
    username: credentials['username'],
    password: credentials['password'],
    url: 'https://stream.watsonplatform.net/speech-to-text/api/',
    version: 'v1'
    url: url,
    version: version
```

```
const SpeechToTextV1 = require('ibm-watson/speech-to-text/v1');

const speechToText = new SpeechToTextV1({
  iam_apikey: '{apikey}',
  url: '{url}'
});
```

Figura 44: Autenticación actual / Autenticación descrita en la documentación

Los códigos en lenguaje JavaScript anteriores (véase la figura 44) representan la manera de generar la autenticación de cualquier programa con el servicio Speech to text. La primera imagen muestra la autenticación que el proyecto tenía haciendo uso del usuario y contraseña generados por la nube y en la segunda imagen se muestra la autenticación descrita en la documentación de la nube de la empresa, la cual hace uso de la llave de API.

La nube migró de método de autenticación por lo que el proyecto se encontraba desactualizado. Procedí a actualizar el código correspondiente a la autenticación en dos niveles (véase la figura 45): El código fuente y el código referente a la configuración de parámetros.

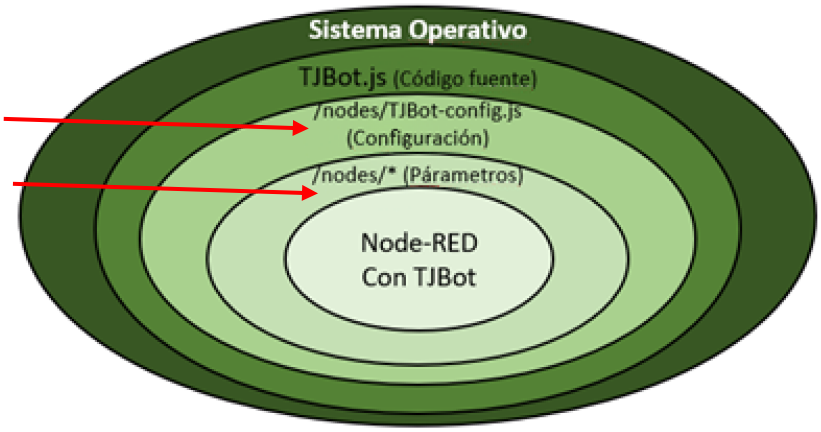


Figura 45: Capas de funcionamiento que requirieron actualización

Primero efectué la corrección del código fuente. Actualicé las líneas correspondientes a la autenticación del servicio (véase la figura 46) reemplazando la autenticación de usuario y contraseña por la que hace uso de las API Keys.

```
if (iam_apikey != undefined ) {
    this._stt = new SpeechToTextV1({
        iam_apikey: iam_apikey,
        url: url
    });
```

Figura 46: Nueva autenticación para servicio Speech to Text

Después, en la capa de "Configuración de parámetros" se describía el traspaso de información entre la capa de Node-RED y el código fuente, particularmente la información que se colocaba en el momento de configurar un nuevo *Bot*. Descubrí que igual que con el código fuente, todas las referencias aquí descritas respecto a la autenticación del servicio *Text to Speech* estaban configuradas para la autenticación con usuario y contraseña. Realicé la actualización de dicho código para implementar la autenticación más actual por medio de llaves de API.

Una vez realizado este proceso reinicié el servidor y probé nuevamente la actividad. Los resultados se vieron reflejados una vez que inicié la actividad de escucha, pues TJBot comenzó a utilizar su micrófono para obtener audio del entorno, enviarlo a la nube y obtener el texto de lo que escuchaba.

Mientras actualizaba este código pude notar que los servicios de *Language translation, Visual Recognition, Conversation y Tone Analyzer* estaban de igual forma desactualizados. Realicé el mismo procedimiento para actualizar los servicios faltantes, gracias a lo cual las habilidades de traducción y el observar se habilitaron nuevamente, logrando así finalmente la compatibilidad buscada.

En este punto vale la pena recordar que el código modificado forma parte del proyecto de código libre alojado en Github, donde pude corroborar que los archivos del repositorio se encontraban desactualizados. Para que este código alojado en Github estuviera disponible para el resto de las personas que descargaran el proyecto en futuras ocasiones, realicé un *push request* que es una solicitud por parte de algún usuario de la comunidad para realizar una actualización del código origen alojado en el repositorio a una nueva versión.

Para realizar esta solicitud me puse en contacto con el creador y dueño del repositorio de los nodos TJBot, con quien colaboré brevemente para adaptar los cambios que realicé en los nodos para así dar paso a una nueva versión consolidada libre para todos.

Una vez que el empleado realizó los cambios en su repositorio realicé una instalación desde cero del proyecto haciendo uso de los códigos de instalación que había generado anteriormente. Al concluir y programar las actividades básicas todas funcionaron con normalidad, solucionando con esto los problemas de compatibilidad del proyecto.

3.6 Estructuración de un taller y generación de recursos auxiliares.

El reto de llevar el proyecto a talleristas que se desempeñen con grupos de alumnos en distintas localidades implica la creación de recursos auxiliares que simplifiquen el proceso de aprendizaje tanto de talleristas como de alumnos. La creación de manuales será indispensable, pues es el primer recurso que centralizará la información para iniciar con la programación de TJBot sin requerir obtener información de internet.

Para entender el punto de vista de los alumnos tomé como referencia un caso de taller que se presentó antes de mi integración al equipo de trabajo. Unas semanas antes se había realizado un taller piloto con TJBot donde los alumnos de escuelas primarias de 11 a 14 años de diferentes escuelas compitieron en el armado y puesta a prueba de algunos TJBots; los ganadores se llevaron algunos kits TJBot de regalo a sus escuelas. El reto consistió en construir la carcasa de TJBot utilizando el cartón cortado, luego colocar los componentes internos, conectar a TJBot a la corriente, conectarse a la interfaz de programación y realizar 3 actividades básicas: Encender el led, mover el brazo y que TJBot hablara.

La actividad tuvo una duración aproximada de 5 horas y al finalizar los participantes pudieron proveer su retroalimentación de lo que les resultó más complicado durante la construcción y programación.

Algunos de los puntos más importantes que se destacaron de esta actividad fueron:

a) La necesidad de contar con un material didáctico de apoyo para el usuario final.
   Se sugiere contar con un manual integrado que guie al usuario de inicio a fin en la construcción e implementación, para que no exista la necesidad de buscar información en internet.

b) Facilitar la obtención de la dirección IP.
   El tema de la obtención de la dirección IP requería optimizar la obtención de ésta. Las pruebas locales que había realizado habían sido únicamente con dos o tres robots conectados al mismo tiempo, para cada uno era necesario conectarle un cable HDMI hacia una pantalla, mouse y teclado; lo que tomaba algunos minutos en la conexión de cada uno de ellos para obtener su dirección IP. Además, durante la programación ocurría que la dirección IP del robot cambiaba sin advertencia alguna, debido a la inestabilidad de la red y a que el servidor DHCP asigna direcciones dinámicas, lo que obligaba al usuario a realizar las conexiones para obtener la dirección IP nuevamente. Este método resulta ineficiente si consideramos que los salones de clases tendrían que habilitar al menos 10 robots de forma simultánea.

Resolver las problemáticas anteriores fue vital para lograr un proyecto fácilmente replicable para un profesor o tutor en condiciones variadas, con una guía que pudiera consultar aun cuando un experto de la empresa no estuviera presente, de fácil uso para un estudiante y muy útil para cualquier usuario que desee construir y desarrollar desde cero a TJBot.

3.7 Problemáticas y soluciones: Estructuración de un taller y generación de recursos auxiliares.

La estructuración de un taller y la generación de recursos auxiliares van totalmente de la mano en la solución de esta problemática, pues el fin común de ambos es facilitar la adopción tanto de los profesores como de los alumnos con el proyecto. Dividí la solución de las problemáticas detectadas en dos partes.

a) Facilitar la conexión entre TJBot y la computadora.
b) Generación de recursos y material de apoyo que serviría de guía a los usuarios finales.

3.7.1 Facilitando la conexión entre TJBot y el equipo de cómputo.

Recordemos que conocer la dirección IP de TJBot dentro de la red a la que se encuentra conectado permite al equipo de cómputo acceder a Node-RED (la interfaz de programación). El hecho de requerir de una pantalla, mouse y teclado para obtener la dirección IP resulta contraproducente por el tiempo invertido y el aumento del costo de llevar a TJBot a talleres en diferentes sedes con muchos participantes.

Para optimizar la obtención de la dirección creé un script que atendiera apropiadamente los asuntos relacionados con la red de TJBot, tales como guardar nuevas redes a través de la memoria, anunciar la dirección IP asignada al robot o identificar cuando la red se pierde o cambia durante el uso del robot; todo esto sin hacer uso de algún componente de hardware externo al proyecto.

La solución que construí está codificada en dos archivos:

a) bootstrap.sh:

Este es el nombre del archivo que generado para la automatización de la instalación, lo actualicé para incluir comandos que instalan y configuran los recursos necesarios para obtener y anunciar la dirección. Dichos recursos serán utilizados en el archivo *getipaddress.sh*.

La decisión de agregarlo al archivo *bootstrap* y no en uno separado ocurrió debido a que el usuario no tendrá que introducir más comandos al instalar la totalidad del proyecto, pues bastará con ejecutar el mismo archivo de instalación inicial, simplificando el proceso.

b) getipaddress.sh:

Creé un nuevo archivo llamado *getipaddress.sh*, el cual obtiene e informa los datos relacionados con la red del robot y los anuncia a través de un sintetizador de voz.

El objetivo fue anunciar la información relevante referente a la red por medio de audio. Hice uso de la bocina de TJBot para anunciar la información de la red, pero para habilitar esto fue necesario un sistema de síntesis de texto a voz de código abierto, compatible con Raspbian y que no requiriese de conectividad a internet para funcionar adecuadamente.

Existen algunos proyectos de código libre de síntesis de voz que cumplen estos requisitos, pero el que encontré más adecuado fue el proyecto *Espeak.* De acuerdo con la página oficial del proyecto (http://espeak.sourceforge.net) esta paquetería tiene la capacidad de sintetizar el texto a voz entre una variedad de idiomas incluido el español, con la ventaja de permitir modular la voz de forma muy práctica.

Una vez que *Espeak* ha sido instalado, podemos hacer uso de esta librería a través la ventana de comandos, donde se introduce el comando *"espeak"* seguido del texto a anunciar y a continuación los parámetros requeridos, entre los que se encuentran el idioma, el volumen, la tonalidad y la velocidad entre otros.

Pude comprobar el funcionamiento del sintetizador por medio del comando:

espeak "Hola" -v es+m3

Este comando invoca al sintetizador y por medio del parámetro *-v* configuro el lenguaje; en este caso configuré el sintetizador al lenguaje español, voz masculina y una tonalidad del número 3. Al introducir el comando se escucha una voz que dice correctamente "Hola" en español.

Una vez preparado el sintetizador codifiqué un algoritmo que cumple con las siguientes tareas.

1- Verificar si el usuario está tratando de guardar una nueva red WiFi en la RPI.

Este código verifica cambios en un archivo guardado en la memoria SD. El contenido de este archivo es modificable a través de una computadora, la información contenida permite agregar el nombre de la red que deseamos que TJBot recuerde y la contraseña de esta. Cuando se introduce nuevamente la memoria a TJBot y se enciende, el algoritmo reconoce el cambio y guarda esta nueva red en el sistema para futuras conexiones.

El procedimiento anterior genera un archivo, por lo que modifiqué el archivo de instalación inicial *bootstrap.sh* (véase la figura 47) para que lo agregue en la ruta donde se necesita.

```
72  #Enable Wi-Fi bootable configuration
73  cp /home/pi/tjbot_setup/mi_red_wifi.default.txt /boot/
74  cp /home/pi/tjbot_setup/mi_red_wifi.txt /boot/
75  cp /home/pi/tjbot_setup/wpa_supplicant.conf /etc/wpa_supplicant/wpa_supplicant.default.conf
```

Figura 47: Comandos para permitir al usuario guardar redes WiFi

El nombre del archivo es *mi_red_wifi.txt*, dentro del cual el usuario guarda el nombre de la red (*ssid*) y la contraseña de la red deseada.

2- Verificar si el sistema cuenta con conectividad alámbrica o inalámbrica.
   Se identifica si hemos conectado un cable ethernet a TJBot y si la conexión ha sido exitosa.
3- Verificar si el sistema tiene una dirección IP asignada.
   Esto indica si el modem de internet ha asignado una dirección IP a TJBot.
4- Verificar si el sistema tiene conectividad a internet.
   El algoritmo realiza un *ping* al sitio *www.google.com*, si el ping es exitoso indica conectividad a internet.
5- Verificar si el servidor Node-RED se encuentra correctamente funcionando. Verifica que el servidor Node-RED inició correctamente, indicando entonces que su configuración no se encuentra dañada y es posible acceder a la interfaz desde otros dispositivos.

Al finalizar la verificación de estos cinco pasos anteriores el algoritmo anuncia a través de la bocina del robot los resultados; para cada paso indica si fue o no exitoso el procedimiento, permitiendo al usuario identificar el origen de cualquier problema que pueda surgir durante la conexión con TJBot.

Para evitar perder la dirección IP en caso de que TJBot se conecte a otra red o que pierda conectividad con internet, el algoritmo se configuró para verificar la conectividad cada cinco segundos. Esto es, si TJBot cae en alguno de los casos anteriores, en cinco segundos anunciará su nuevo estatus y en dado caso de que su dirección IP haya cambiado anunciará la nueva información a través de su sintetizador de voz.

Esta verificación cada cinco segundos es muy útil en caso de que el algoritmo haya detectado que alguno de los cinco pasos de verificación falló. Si alguna verificación falló no podremos conectarnos a Node-RED o trabajar adecuadamente con el kit, por lo que TJBot anunciará cada cinco segundos que hay un error y no se detendrá hasta que este error sea corregido.

```
#!/bin/bash
MI_RED_WIFI="/boot/mi_red_wifi.txt"
WPA_SUPPLICANT="/etc/wpa_supplicant/wpa_supplicant.conf"

# Parametros para voz espeak
VOLUME=200 # -a
SPEED=160 # -s
PITCH=60 # -p
PAUSA=10 # -g
ENFASIS=20 # -k
LENGUAJE=es+m3 # -v

TIEMPO_DE_COMPROBACION=5
# Estatus de la conexion actual: Se pone en cero para definir que no hay conexion inicial
CONEXION_ACTIVA=0

# Busca si existe una nueva red agregada manualmente por el usuario
check_is_new_network_added(){ ...
}

# Comprobacion de tipo de conexion: Alambrica o Inalambrica
check_connection_type(){ ...
}

# Ping a google.com para comprobar conectividad a internet.
check_connectivity(){ ...
}

# Comprobacion si el puerto de Node-RED esta funcionando
check_server(){ ...

}

# Loop que comprueba cada
network_status_loop(){ ...
}


check_is_new_network_added
network_status_loop
```

Figura 48: Algoritmo del archivo getipaddress.sh

Por último hice uso de la funcionalidad *Cron* de Raspbian. *Cron* es un programa que se ejecuta en segundo plano y comienza a funcionar apenas el sistema operativo carga, el propósito de este programa es el de ejecutar tareas programadas por el sistema en ciertos momentos definidos. Uno de los recursos con los que cuenta *Cron* es un archivo de texto que puede editarse por medio del comando *Crontab*, donde el usuario puede definir manualmente tareas en el sistema.

En este caso deseaba que el archivo *getipaddress.sh* iniciara apenas el sistema operativo cargara, así que modifiqué una vez más el archivo *bootstrap.sh* para que configurara automáticamente el *Crontab* del sistema (véase la figura 49), programando la ejecución del algoritmo cada que la RPI encendiera.

```
69  #Enable speakable ip address on startup with a crontab task, we use the user pi for this job
70  su pi -c "(crontab -l 2>/dev/null; echo '@reboot /home/pi/tjbot_setup/getipaddress.sh') | crontab -"
```

Figura 49: Comando que configura el Crontab del sistema

Tras esto puse a prueba la efectividad del algoritmo creado en conjunto con el algoritmo de instalación desde cero. Formateé la memoria y con una instalación limpia del sistema operativo ejecuté los siguientes comandos.

```
git clone https://github.com/JairLizarraga/tjbot_setup
cd tjbot_setup
sudo ./bootstrap.sh
```

Los comandos iniciaron la instalación del proyecto TJBot junto con la mejora del algoritmo de red, al terminar el proceso se reinició automáticamente la RPI y, tras cargar el sistema operativo, la bocina anunció el estatus de la red actual: el nombre, la dirección IP, la conectividad a internet y el estatus del servidor Node-RED. Por último agregué una nueva red WiFi (hice uso de una red WiFi creada desde mi teléfono), TJBot la detecta, se reinicia y se conecta automáticamente a esta nueva red. Con esto se comprobó que el algoritmo funcionaba de acuerdo con lo esperado.

3.7.2 Recursos auxiliares.

La replicabilidad del proyecto en instituciones educativas depende de varios factores tales como la facilidad de adopción por parte de los docentes, utilidad, funcionalidad, innovación y recursos didácticos auxiliares entre otros.

Algunos de los factores anteriores han sido cubiertos por la naturaleza del proyecto mismo ya que la tecnología con la que trabaja TJBot es disruptiva para muchas instituciones educativas en la actualidad.

Si combinamos la innovación con el potencial de Node-RED para desarrollar algoritmos de forma sencilla, se propicia que profesores integren actividades basadas en el robot en un salón de clases, involucrando temas como programación, robótica y electrónica. Inclusive no se limita a estos temas, pues el impacto llega al grado de que sirve de apoyo en cualquier materia, dónde puede apoyar en la impartición de exámenes, encuestas o incentivar al alumno a su participación en clase de una forma innovadora involucrando tecnología.

Los recursos didácticos auxiliares aún faltaban por integrarse, conjuntando todos los resultados y aprendizajes encontrados durante mi trabajo. A continuación se muestran los recursos didácticos auxiliares generados que facilitan a los tutores y alumnos la adopción del proyecto de forma más sencilla e informada.

3.7.2.1 Manual de uso.

Inicialmente elaboré un conjunto de tres manuales de uso que concentraban los aprendizajes obtenidos en el transcurso de mejora del proyecto, la información recabada de internet y el uso de los códigos generados durante la optimización del proyecto TJBot. En general este material cubría de forma general los siguientes temas:

1- Antecedentes teóricos de la nube de la empresa, cómputo cognitivo, RPI, Node-RED y TJBot.
2- Preparación del software.
3- Construcción de TJBot.
4- Programación de TJBot.
5- Programación de actividades avanzadas.
6- Anexos y resolución de problemas comunes.

El material generado se vio gratamente nutrido por experiencias en talleres con alumnos, profesores y trabajadores de la empresa, quienes dieron su retroalimentación con puntos de mejora en la capacitación que recibían.

Como un primer acercamiento, los resultados de estas experiencias dieron hincapié a la creación de 3 manuales diferentes, todos llevaban por nombre inicial *Explorando a TJBot: Un vistazo al mundo de los robots. Manual para el voluntario de la empresa,* cada uno de diferente tipo: Manual de instalación, guía básica y guía avanzada.

1) Manual de instalación

Figura 50: Manual de instalación

Este breve manual (véase la figura 50) de 16 páginas describía paso por paso las instrucciones para instalar el proyecto TJBot desde cero y del cómo tratar el tema de la conectividad a internet.

El propósito es que el usuario prepare el software de TJBot desde cero y esté listo para programar sus primeras actividades.

2) Guía básica

Figura 51: Guía básica

Tras la preparación del software, la RPI dejó de ser una simple tarjeta de desarrollo y se transformó en el cerebro de TJBot. En este segundo manual (véase la figura 51) el usuario tenía una introducción pedagógica en el mundo de la robótica. Se aprendía a construir físicamente a TJBot, a realizar la conexión remota al entorno de programación Node-RED y a programar 6 actividades básicas (las más sencillas) que posee TJBot.

3) Guía avanzada

Figura 52: Guía avanzada

Éste último manual (véase la figura 52) estaba orientado para aquellos usuarios que desearan profundizar en el funcionamiento interno de TJBot, para poder diseñar y

programar soluciones más complejas que integraran tecnologías variadas en conjunto con las de la WIA.

En el manual se explicaban algunos principios técnicos del funcionamiento de Node-RED y la estructura de los datos que pasan entre los nodos; se mostraba a detalle otros nodos más avanzados que ayudan a desarrollar las 5 actividades avanzadas sugeridas en el resto del manual, las cuales integran tecnologías como JavaScript, Twitter y la obtención del clima por medio de APIs.

El objetivo de haber dividido en 3 el manual surge por la idea de llevar al usuario paso por paso y que él mismo decida con cuanta profundidad desea involucrarse en el tema.

Se liberaron estos tres manuales disponibles en formato digital y se entregaron impresos en algunos proyectos de donación de TJBots en instituciones educativas y en talleres de capacitación a empleados voluntarios.

Posterior a la creación de estos manuales la forma en que los talleres se impartían se transformó; pasó de ser impartido directamente de la empresa a alumnos a impartirse a un modo que llamamos en RSC *"Teach the teachers"* o "capacitación de capacitadores". En esta modalidad se buscaba que tanto profesores como voluntarios de la empresa dominaran el proyecto, de manera que pudieran habilitar a otros profesores para impartir talleres y poder atender las dudas que los alumnos pudieran tener.

Este gran salto generó conocimientos colectivos ya que de forma exponencial aumentaron los tutores y capacitadores que se enfrentaron a dudas y cuestionamientos por parte de los alumnos, nuevas ideas y formas de hacer las cosas. Esto permitió encontrar puntos de mejora en los manuales inicialmente generados. Al mismo tiempo, la nube de la empresa recibió algunas actualizaciones menores que podrían causar confusión a quienes usaran la información contenida en los manuales con lo que se observara en la nube de la empresa.

Lo anterior me llevó a desarrollar un nuevo manual, una versión más simple, integrada y actualizada de los tres primeros manuales en uno solo llamado *¡Hola, TJBot! Manual de usuario*, lo que hizo más sencillo simplificar y transmitir los conocimientos a los participantes.

Figura 53: Manual integrado TJBot

Este manual (véase la figura 53) sirve como guía integrada de primera mano para los profesores, tutores y capacitadores que cuenten con TJBots, se encuentra disponible en formato impreso y digital.

3.7.2.2 Página web.

Encontrar un canal único donde alojar el manual de usuario se volvió un tema complicado. Parte vital de los donativos de TJBots realizados a escuelas era acompañarlos de un manual de usuario para poder compartirlo durante las sesiones. Probamos inicialmente imprimir los manuales, pero esto conllevaba costos innecesarios y problemas de logística.

Por otro lado la empresa sugiere alojar archivos en la nube de *Box* (www.box.com), pero suponía complicaciones debido a que la URL no era amigable, se desconfiguraba constantemente y en ocasiones aleatorias no permitía la correcta descarga de los recursos.

La solución que creé para facilitar el acceso a este manual (y posteriores recursos adicionales) fue la creación una página web disponible para el público, la cual se encuentra alojada en los servidores de la nube de la empresa. Esta página la programé haciendo uso de código HTML, CSS, JavaScript y el *framework Express* del lado del servidor. Por la naturaleza de esta página su uso es completamente informativo, por lo que su complejidad no requirió de bases de datos que añadieran dificultad en el *Backend.*

Para alojar la aplicación en la nube de la empresa hice uso de la CLI (Interfaz de la línea de comandos, por sus siglas del inglés) de la empresa, ésta permite interactuar con los servicios de la nube de la empresa desde una computadora a nivel local. Entre sus funcionalidades permite la creación de servicios, instancias, intercambio de información y levantamiento de proyectos. Para preparar la CLI de la empresa la instalé localmente en mi computadora, me autentiqué con mis credenciales de la nube y preparé un espacio de trabajo.

Posterior a esto comencé en mi computadora local la construcción del proyecto, el cual fue creado utilizando *Express* como *framework* para agilizar la programación del servidor.

La página cuenta con las siguientes categorías.

a) Badges: Lo relacionado con las certificaciones que los usuarios pueden obtener.
b) Calendario: Calendario de talleres y capacitaciones futuras.
c) Tutoriales
    a. TJBot: Página que aloja todo lo relacionado al proyecto TJBot, como manuales, archivos, videos y actividades.
    b. Chatbots: Página que aloja todo lo relacionado a la construcción de chatbots con WIA.
d) Cupón para académicos: Una sección que alberga instrucciones para obtener un cupón de la nube de la empresa.
e) Información y contacto: Datos de contacto para resolución de problemas y dudas.

El resultado fue el siguiente (véase la figura 54 y 55):

Figura 54: Página principal del sitio web desarrollado

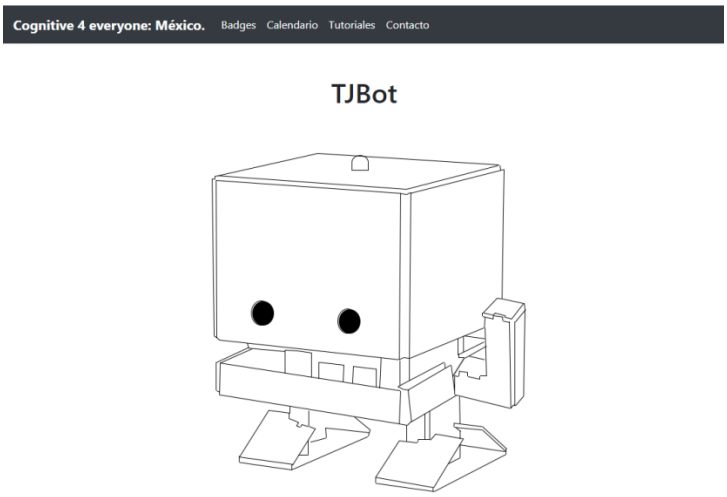


Figura 55: Apartado TJBot del sitio desarrollado

Cuando la programación de la página concluyó utilicé la CLI de la empresa para subir la página web a la nube y habilitar el servidor de la página para que en la nube se le asignara un dominio habilitado accesible públicamente. Modifiqué el nombre del dominio para elegir uno amigable (https://cognitiveparatodos.mybluemix.net). Luego de esto la URL y la página web quedó habilitada.

La página aloja el manual, archivos y videos necesarios para cualquier persona que desee aprender sobre TJBot. Esta página no es un recurso oficial de la empresa, sin embargo la propuse como un recurso de apoyo más eficiente para conjuntar recursos didácticos para el proyecto durante los talleres.

3.7.2.3 Videotutoriales.

La página resultó una forma funcional de centralizar los archivos y las instrucciones para entender, construir y comenzar a programar a TJBot, así como algunas ideas de aplicaciones de este proyecto en un manual paso por paso.

Con la experiencia recabada durante las sesiones de capacitación, los participantes sugirieron constantemente la creación de videos de apoyo con temas específicos de TJBot, por lo que propuse videotutoriales alojados en YouTube a través de mi perfil personal (véase la figura 56). Como en el caso de la página web, este canal no es un recurso oficial de la empresa. Esto ayuda mucho a los usuarios al ser más demostrativo un video que leer hojas de un manual. Gracias a estos videos los usuarios pueden escoger el formato en el que desean aprender, donde por lo general son los profesores de escuelas quienes prefieren un formato impreso o digital del manual, mientras que los alumnos prefieren videos que puedan seguir a su propio ritmo.

Por ejemplo, uno de los videos que se pueden encontrar en este perfil es de la construcción de la carcasa de TJBot en MDF, video que fue generado tras haber conseguido las primeras carcasas MDF del proceso de mejora del robot. Por otro lado, también existe un video para entrenar modelos de reconocimiento visual personalizado con TJBot para enseñarle a jugar piedra papel o tijera.

Estos videos buscan ser breves y concisos para transmitir la información más importante sin entrar en mucho detalle, esto le permite al usuario lograr la actividad propósito del video y al mismo tiempo cultivar la curiosidad de ahondar en los temas que se involucran.

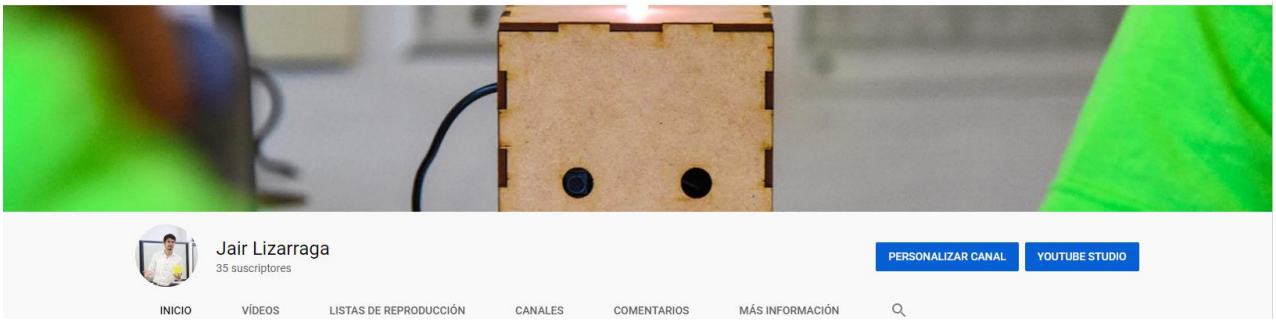


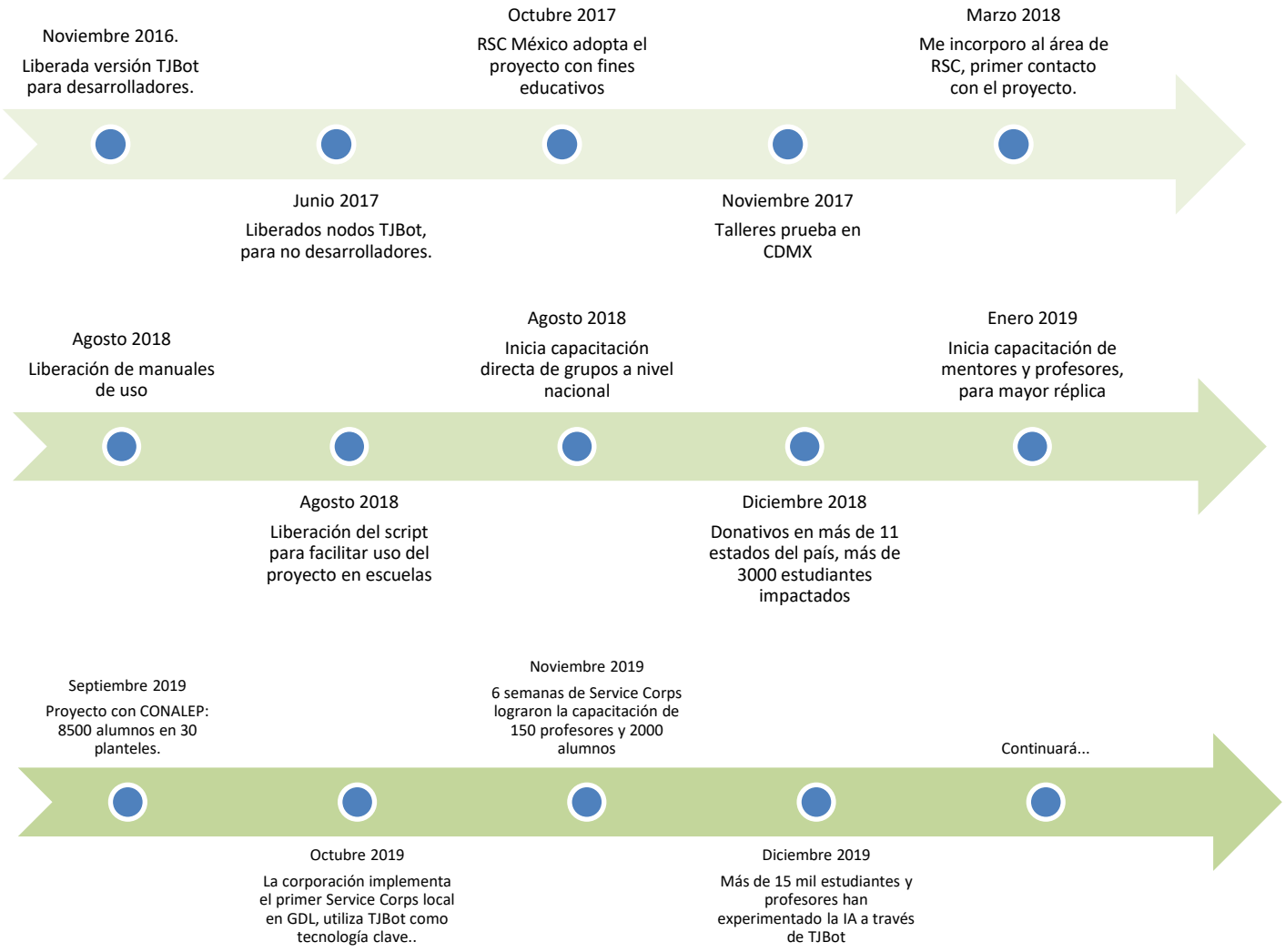
Figura 56: Perfil de YouTube donde se alojan videotutoriales de TJBot

Capítulo 4:
Resultados

Capítulo 4:
Resultados

En la empresa existe la filosofía de que ningún proyecto es perfecto, toda actividad es susceptible de ser mejorada a lo largo de su existencia y un ejemplo ha sido el trayecto de este proyecto. Ha pasado poco más de año y medio que ha transcurrido desde que comencé a trabajar en él y sus mejoras se han percibido gradualmente; pasó de ser un proyecto desconocido a ser una herramienta educativa disruptiva en instituciones educativas en el país.

La siguiente es una línea de tiempo que resume de forma breve los sucesos clave del proyecto hasta ahora.



4.1 La integración de soluciones logran la adopción educativa deseada.

La optimización de la estructura física del robot permitió una construcción más rápida del robot, logrando centrar al alumno en lo más importante: Cómo los componentes electrónicos trabajan entre sí para dar vida a un robot. Profesores de robótica y electrónica aprovechan esta parte para ejemplificar el armado de robots más complejos que involucran sensores de todos tipos.

La mejora del software y de la conectividad lograron abstraer la complejidad interna del proyecto para que el usuario no debiera preocuparse por el código fuente de TJBot.

Adicionalmente, el algoritmo desarrollado para proporcionar la información de la red permitieron prescindir de un mouse, teclado y pantalla en todo momento, de forma que el kit se ha convertido en un dispositivo “Plug and Play”. Gracias a esto los usuarios pueden programar a TJBot en casa o en la escuela, prácticamente en cualquier lugar con red disponible.

Los recursos auxiliares como el manual de uso, la página web y los videos son uno de los apoyos primordiales para los profesores, éstos les son entregados y referidos a los profesores cuando reciben la capacitación (el manual se les da impreso) con lo que cuentan con un material de apoyo a futuro.

Aún mejor, el material está hecho de tal modo que no se requiera de una capacitación presencial para poder replicar el proyecto y entenderlo a la perfección. Aun así, la capacitación presencial acelera el proceso de aprendizaje y resolución de dudas.

Actualmente los profesores están descubriendo en conjunto con sus alumnos las soluciones que estas tecnologías pueden traer y mientras algunos profesores están fusionando a TJBot con su materia asignada, algunos alumnos crean proyectos para sus trabajos finales, colaboran con sus profesores o incluso con voluntarios de la empresa para desarrollar las soluciones a problemáticas actuales.

Algunos ejemplos de soluciones de alumnos que han recibido capacitación por parte de sus profesores y de voluntarios son:

a) Páginas web con TJBot.
b) Reconocimiento visual para jugar piedra, papel o tijera.
c) Identificación de sentimientos por medio de reconocimiento facial.
d) Separación de basura.
e) TJBot para domótica (automatización del hogar).
f) TJBot conectado a Arduino para Internet de las Cosas.
g) Realización de cuestionarios con chatbots.
h) Encuestas de satisfacción al cliente.
i) Juegos didácticos para la introducción de alumnos de primarias y secundarias a la programación.
j) TJBot cuenta chistes.

4.2 Certificaciones.

Las certificaciones que la empresa maneja son llamados "*Badges*", certificaciones digitales con valor curricular que demuestran el conocimiento o el dominio de quien lo posee en algún campo o área. Estos badges son reconocidos internacionalmente y hoy en día son muchas las empresas quienes han adoptado este tipo de certificaciones, pues la ventaja es que ofrecen a cualquier reclutador una forma sencilla de verificar de forma segura que los badges que la persona posee son reales, cuándo fue obtenido, qué conocimientos certifica y qué empresa es la que ha certificado.

En este contexto se estudió la posibilidad de generar badges que certificaran y distinguieran a las personas que se hubiesen capacitado en TJBot, pues los conocimientos adquiridos por aprender esta tecnología son distintivos en el campo laboral. Tras realizar algunos ajustes en el temario de las capacitaciones se tramitó con el área correspondiente de certificaciones digitales la liberación de tres badges que demostraran los conocimientos adquiridos durante estas capacitaciones. Por políticas de la empresa los badges debieron ser divididos en dos tipos: Para internos y para externos.

TJBot Workshop Advocate.

El poseedor de este badge (véase la figura 57) sabe cómo construir, configurar y programar TJBots, desarrollando soluciones en Node-RED utilizando servicios de WIA y APIs de terceros. Adicionalmente, el poseedor de este badge ha demostrado habilidades liderando workshops. Este badge está disponible sólo para trabajadores de la empresa, para obtenerlo son requisitos:

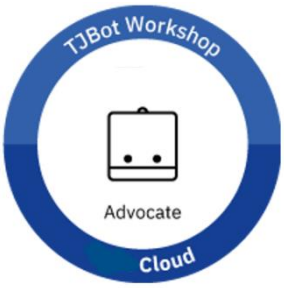


Figura 57: Badge TJBot nivel 1

a) Completar el entrenamiento presencial de 8 horas "*TJBot Advocacy Workshop*".Construir, configurar y habilitar las capacidades cognitivas de un TJBot.
b) Liderar o colaborar en la implementación de un taller de TJBot con 15 participantes.

TJBot Teacher Advocate.

El poseedor de este badge (véase la figura 58) sabe cómo construir, configurar y programar TJBots, desarrollando soluciones en Node-RED utilizando servicios de WIA y APIs de terceros. Adicionalmente, el poseedor de este badge ha demostrado habilidades liderando workshops. A diferencia del badge "*TJBot workshop*", el poseedor de este badge es un profesor que cuenta con las bases necesarias para implementar los principios de Inteligencia Artificial y computación cognitiva en el salón de clases.

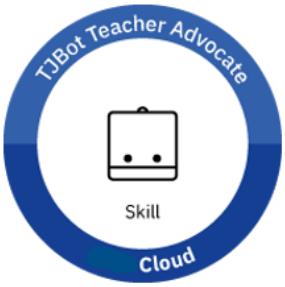


Figura 58: Badge TJBot nivel 2

Los requisitos para obtener este badge son:

a) Ser profesor activo de una Institución Educativa en América Latina.
b) Completar el entrenamiento presencial de 8 horas "*TJBot Teacher Advocacy*".
c) Implementar los conocimientos adquiridos en una sesión presencial con un grupo de al menos 15 alumnos.
d) Construir, configurar y habilitar las capacidades cognitivas de un TJBot.

TJBot Teach the Teachers.

Este badge (véase la figura 59) es el siguiente paso del badge *TJBot Teacher Advocate Badge*. El poseedor de este badge sabe construir, configurar y programar actividades avanzadas de TJBot. El poseedor tiene experiencia compartiendo estos conocimientos a otros profesores, empoderándolos para liderar e impartir talleres de TJBot en sus salones de clases.

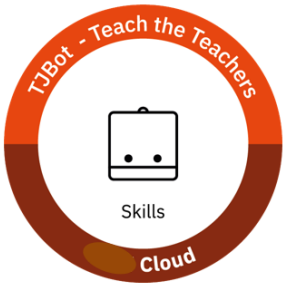


Figura 59: Badge TJBot nivel 3

Los requisitos para obtener este badge son:

a) Haber obtenido el badge "*TJBot Teacher Advocate*".
b) Liderar e implementar un taller "*Train the trainers*" para compartir conocimientos y habilidades con otros profesores.
c) Guiar al menos a 3 profesores para que obtengan su badge "*TJBot Teacher Advocate*"

La existencia de estos badges cumplen un doble propósito. Por un lado la persona certificada puede demostrar por medio de una certificación oficial emitida por la empresa, que cuenta con los conocimientos necesarios para encabezar un taller de TJBot, conoce los materiales necesarios e inclusive puede motivar la creación de un laboratorio de robótica basado en TJBot desde cero. Por el otro lado, generamos un ecosistema donde cada vez más empleados de la empresa querrán acercarse como voluntarios de RSC y ganar además su certificación; los profesores que se hayan certificado con el primer badge tendrán la posibilidad de aspirar a un siguiente nivel de certificación con solo enseñar a sus compañeros a hacer uso de TJBot, donde estos nuevos profesores a su vez podrán obtener su primer nivel.

4.3 Resultados primer semestre del 2019.

Uno de los resultados más importantes que debo destacar es el antes y después del propósito del proyecto en 3 etapas que nosotros mismos no habíamos vislumbrado que existirían.

Inicialmente, en la primera etapa, TJBot fue creado para una comunidad de desarrolladores que desearan involucrarse con la tecnología de Inteligencia Artificial de la empresa y que conocieran el lenguaje JavaScript; además que se sintieran cómodos con la ventana de comandos y pudieran desarrollar algoritmos que hicieran uso de la librería TJBot para acceder a servicios de computación cognitiva.

Después TJBot llega al área de RSC con el propósito acercar sus ventajas al área educativa, la visión inmediata fue conseguir su funcionalidad, dominar su tecnología y programación; el foco principal era el estudiante en lugar de los desarrolladores experimentados, con el propósito de que los alumnos experimentaran con programación. Esto se logró una vez que el software y hardware fue mejorado, la versatilidad del proyecto favoreció la adopción por parte de los alumnos y las escuelas comenzaron a mostrar cada vez más interés en conocer a TJBot.

Figura 60: Estudiante perteneciente a escuela primaria que se capacitó en TJBot

Los talleres comenzaron a impartirse a personas de todas las edades, jóvenes, adultos y alumnos desde primarias hasta universitarios, profesores y trabajadores de la empresa. Esta etapa abarcó principalmente desde agosto del 2018 hasta inicios de 2019, periodo durante el cual obtuvimos los siguientes resultados:

a) Acercamiento en 11 estados de la república.
b) Más de 40 talleres realizados.
c) Más de 3000 participantes.

Además, durante este periodo se logró donar cierto número de kits TJBot que hoy en día siguen siendo utilizados por las escuelas que recibieron el equipo.

El éxito y la demanda fue tal que el proyecto evolucionó hacia una tercera etapa centrada en el docente. No sólo necesitábamos alumnos expertos en la tecnología, necesitábamos de un modelo replicable donde el profesor fuera un experto en todos los aspectos de TJBot, que sin importar cuantas generaciones de alumnos pasaran por el aula, el docente tuviera los conocimientos suficientes para involucrar a cada estudiante en el estudio de la programación e Inteligencia Artificial con TJBot.

4.4 Resultados segundo semestre del 2019.

Esta última etapa se define por sí misma como "*Teach to Teachers*", donde las capacitaciones poco a poco dejaron de darse a alumnos directamente y los esfuerzos se centraron en preparar a capacitadores o profesores de escuelas en dominar a profundidad los temas necesarios para replicar esto con el paso del tiempo. Dos grandes proyectos fueron realizados.

4.4.1 ¡Hello, TJBot!

En alianza con una fundación dedicada a la enseñanza de robótica en distintos niveles educativos, se desarrolló un temario de 20 horas para capacitar a 72 profesores de 30 instituciones educativas a nivel medio superior (CONALEP y Vocacional), quienes a lo largo de 4 meses se capacitaron mientras aplicaban sus conocimientos con sus alumnos. Un donativo de TJBots fue realizado, el cual logró un impacto que llegó a 8500 alumnos a diciembre del 2019.

Figura 61: Profesor capacitado en TJBot presenta el proyecto a sus alumnos de CONALEP

4.4.2 Service Corps de la empresa

Desde hace más de 10 años el programa Service Corps de la empresa (anteriormente llamado Corporate Service Corps) es un programa que selecciona a un número reducido de trabajadores de la empresa de todo el mundo para formar equipos multidisciplinarios que a lo largo de un mes invierten su tiempo y esfuerzos en la resolución de un problema social en alguna región del mundo; es una iniciativa de la corporación de la empresa a nivel mundial.

Por primera vez en la empresa la corporación decidió llevar a cabo un programa piloto bajo el esquema de Service Corps a nivel local; éste sería basado en el proyecto TJBot con sede en la ciudad de Guadalajara, Jalisco, para desplegar un equipo de 20 empleados de la empresa en Guadalajara en 5 instituciones educativas.

Haciendo uso de la última versión de TJBot que desarrollé y los manuales de uso, realizamos una capacitación de dos días cuyo propósito era otorgarles todas las herramientas para que comprendieran la programación y las bases del funcionamiento de TJBot.

Cada una de las cinco escuelas participantes contaron con un equipo de 4 voluntarios y un número de kits TJBot. Tanto la escuela como la empresa colaboraron estrechamente para acondicionar y preparar las aulas de cómputo, la conectividad a internet y enseñar al mayor número de profesores posibles para que éstos lo pudieran integrar en sus planes educativos.

Los números en este proyecto se vuelven algo exponencial. Se planteó como propósito alrededor de 150 profesores capacitados durante los 45 días de duración del proyecto y un impacto directo de al menos 2000 alumnos a fines de 2019, se espera que a partir de 2020 los profesores puedan continuar utilizando el proyecto y llevar el programa a una mayor cantidad de alumnos.

Derivado de estos dos grandes proyectos se están logrando grandes avances en cuanto a usos y utilidades del proyecto, integraciones con nuevas tecnologías tanto de software como de hardware, esto debido a que muchos de los profesores que están tomando las capacitaciones actualmente imparten materias de robótica, programación, electrónica y materias relacionadas que empatan con las potencialidades de TJBot.

4.4.3 Trabajo a futuro

Los avances obtenidos en la mejora del material y del proyecto han permitido cumplir con el objetivo planteado de TJBot en el área de RSC.

a) Mejorar el proyecto TJBot para lograr un funcionamiento estable y controlado.
b) La simplificación del proceso de instalación y conexión remota.
c) Creación de material auxiliar como manuales de uso ofrecen la replicabilidad buscada entre los profesores de la comunidad educativa.

El trabajo a futuro está pensado en la masificación del taller TJBot a nivel internacional. Actualmente algunos países de Latinoamérica desean replicar el modelo implementado en México para llevar estos talleres a las escuelas de sus países. Estaremos compartiendo el material generado en nuestro país, el formato de los talleres y los temarios desarrollados, los cuales podrán ser adaptados y actualizados.

Las certificaciones pasarán de ser brindadas nacionalmente a ser expedidas en otros países aumentando el alcance del reconocimiento de estas habilidades adquiridas por los asistentes de los talleres. Establecer los requisitos de la forma más clara y definida posible, además de buscar una forma automatizada de confirmar el cumplimiento de estos requisitos para certificar a los usuarios será una de las tareas a realizar, pues el número de certificaciones expedidas mes por mes podrá pasar de ser cientos a ser miles alrededor del mundo.

La conectividad a TJBot tiene puntos de mejora. TJBot puede conectarse a internet de forma alámbrica e inalámbrica. Éste último caso es uno de los más problemáticos cuando muchos TJBots están conectados a la misma red WiFi de una escuela, sobre todo cuando la red está compartida con otros dispositivos tanto alámbricos como inalámbricos; los robots tienden a perder conectividad a internet y por lo tanto a las computadoras que los están programando.

A futuro es posible que se desarrolle un método de conexión a TJBot que no requiera de conectividad a WiFi o ethernet para mantener la conexión con la computadora, donde puedan programarse actividades offline y en el momento de iniciar aquellas que requieran conexión a internet muestre una advertencia, pero permita continuar con las actividades. Hay algunas posibilidades que requieren de cambios importantes en el sistema para que esto funcione, algunas de estas propuestas son conectividad por bluetooth para conectarse a Node-RED y otra es activar en la RPI un punto de acceso local como si se tratara de un router que habilite una red propia donde la computadora se conecte y permita acceder a Node-RED.

Otro punto de mejora es la reducción de costo por TJBot. En 2020 el costo de un TJBot en el mercado es de aproximadamente alrededor de $3,700 pesos mexicanos, donde los componentes más caros son la tarjeta RPI y la cámara. Recientemente he encontrado información de la empresa en Japón donde han creado una versión mini de TJBot donde se reemplaza la tarjeta RPI 3B+ por una tarjeta RPI Zero, cuyo costo es aproximadamente la mitad.

Figura 62: Comparativa de RPI Zero vs RPI 3B+

RPI Zero (véase la figura 62) es una versión más compacta y barata respecto a la versión 3b+. Las diferencias radican en que cuenta con un menor número de componentes, pues se prescinde de los puertos USB tradicionales, puerto ethernet, cuenta con una memoria RAM menor y el puerto HDMI es más pequeño. Esto reduce su costo en más del 50% pero así mismo sus capacidades.

En japón lograron adaptar parcialmente el proyecto TJBot para trabajar con esta nueva tarjeta. Debido a la perdida de los puertos USB tradicionales debieron conectar el micrófono con un adaptador de micro USB a USB y la bocina tuvo que sustituirse por una analógica conectada a través de un circuito electrónico construido en una protoboard hacia los pines de la tarjeta (véase la figura 463.

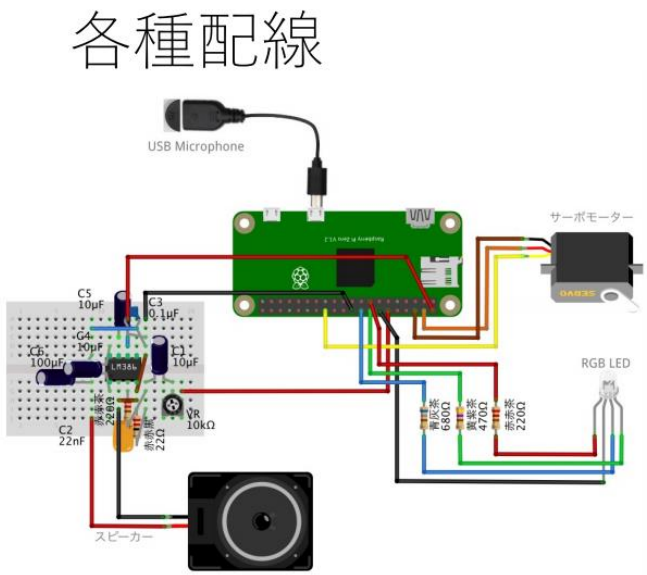


Figura 63: Diagrama de conexiones TJBot con RPI Zero

Éste es el primer problema, la construcción se vuelve más compleja y aunque podría adicionar conocimientos al involucrar más componentes electrónicos el proceso se vuelve más elaborado.

Y en cuanto al software de este proyecto tiene modificaciones considerables, el control de los componentes se hacen por medio de nodos más complejos de configurar y con muchos pasos intermedios. Cuando se trata de utilizar los nodos de TJBot me encontré con incompatibilidades de la tarjeta con los componentes conectados a los pines, tales como la bocina y el led.

La optimización de costos requeriría de adaptar el proyecto a esta nueva tarjeta, analizar el software que fue configurado para este modelo (cuya documentación se encuentra en japonés y desactualizada) y hacerlo compatible con los nodos de TJBot, o crear nodos más sencillos para la manipulación de los componentes.

Por último sólo queda la actualización constante del contenido de los manuales, la página web, los videos y de la integración de las actividades que la comunidad comience a crear, de forma que la página y los videos se conviertan en una referencia general para comenzar a programar.

4.5 Conclusiones

El impacto del proyecto en los participantes (docentes, alumnos, voluntarios de la empresa) han dado testimonio de la efectividad de TJBot como una herramienta disruptiva en el aprendizaje de tecnologías de este tipo. Inclusive lleva a los alumnos a considerar estudiar una carrera relacionada con las tecnologías de la información, integrar estas tecnologías en sus carreras o profesiones actuales, o bien pasar tiempo los fines de semana desarrollando soluciones que pueden mejorar su vida.

El panorama es contrastante una vez concluido este trabajo respecto al inicio del proyecto. Los recursos tecnológicos tradicionales de un aula están lejanos de las tecnologías de Inteligencia Artificial y pueden verse como algo muy remoto de implementar. Ahora podemos ver profesores que de forma independiente están desarrollando actividades haciendo uso de estas tecnologías por medio de TJBot con sus alumnos. Se están creando laboratorios, temarios y planes de estudio donde TJBot forma parte.

El open source permitirá a TJBot crecer cada vez más, conforme la comunidad vaya creciendo y las necesidades vayan surgiendo. La capacidad de modificar y crear nuevos módulos hará que existan funcionalidades que quizá hoy no imaginamos todavía, pero que cambiarán la vida de las personas.

A partir de la conclusión de este trabajo y de las mejoras hechas en TJBot, el futuro del proyecto es claro, la cantidad de alumnos impactados crecerá y con esto cada vez más jóvenes adquirirán conocimientos de Inteligencia Artificial, los cuales serán habilidades importantes y claves en la carrera que elijan ya que en un futuro no muy lejano todas las profesiones se verán impactadas por este nuevo paradigma tecnológico.

Glosario

Algoritmo	Serie de pasos requeridos para la solución de un problema
API	Del inglés Application Programming Interface, significa interfaz de programación de aplicaciones, simplifica la comunicación entre dos sistemas distintos.
API Key	Es una llave de acceso necesaria para integrar una interfaz de programación de aplicaciones a otro sistema.
Backend	Se refiere al desarrollo de sistemas del lado del servidor y todo lo relacionado.
Badge	Insignia digital que una empresa o compañía emite para certificar conocimientos acerca de un tema.
Blockchain.	En español significa Cadena de Bloques, es una estructura de datos cuyo propósito es evitar la falsificación de datos.
Bot	En el contexto de TJBot, un Bot es un elemento que agrupa las configuraciones básicas de comportamiento del robot, tales como género, lenguajes que maneja y servicios a los que se encuentra conectado.
BX	BX hace referencia al nombre de la página que no es revelado por cuestiones de confidencialidad de la empresa y sus productos
Caché	Capa de almacenamiento de datos intermedia que almacena datos temporales con el propósito de aumentar la velocidad del traspaso de información entre componentes de hardware.
Canal de comunicación PWM	Pulse Width Modulation, o modulación por ancho de pulso, es una técnica que transforma señales de sonido análogo a un formato digital que pueda ser procesado fácilmente por una computadora y viceversa.
CLI	Command Line Interface, o interfaz de línea de comandos, es un entorno que permite a los usuarios interactuar con un sistema, ya sea una aplicación específica o un Sistema Operativo completo.
Cloud	Su traducción al español es nube, en sistemas computacionales hace referencia a la una red de servidores alojados en internet y accesibles de forma remota.
Comando	Es una instrucción que el usuario da desde una ventana de comandos a un Sistema Operativo para ejecutar una tarea.
Computación cognitiva	Es la tecnología computacional que imita las capacidades cognitivas del ser humano, principalmente la capacidad de un sistema de cómputo de razonar y aprender.

Corporate Service Corps	Es un programa donde la empresa ejerce su responsabilidad social a través de la implementación de un proyecto de voluntariado con un mes de duración.
Cron	Es un programa del sistema operativo LINUX/UNIX que permite programar la ejecución de comandos en un horario y fecha específicos.
Crontab	Nombre del archivo que contiene las instrucciones de qué comandos ejecutar y en qué momento, de forma automática.
CSS	Cascading Style Sheets, significa hojas de estilo en cascada, es lenguaje que permite describir las características visuales del contenido de una página web.
ExpressJS	Framework que facilita el desarrollo de páginas web, permite la construcción de estas en lenguaje JavaScript tanto frontend como backend.
Flujo	En Node-RED puede significar dos cosas. En primera puede ser la interconexión de dos o más flujos. En segunda es una ventana que agrupa nodos para organizar el trabajo.
Framework	También llamado entorno de trabajo es un conjunto de herramientas enfocadas en facilitar el desarrollo de una tecnología específica, otorga simplicidad y velocidad al desarrollador.
Frontend	En desarrollo web, hace referencia al desarrollo de componentes que están en contacto directamente con el usuario, de parte del navegador web, como pueden ser componentes visuales y animaciones.
Github	Es una plataforma que aloja proyectos (generalmente de código libre) haciendo uso de un sistema de control de versiones.
GPIO	General Purpose Input Output, o pines de entrada y salida de propósito general, son aquellos pines de la Raspberry PI que sirven para conectar componentes electrónicos externos a la tarjeta.
Hardware	Son aquellos componentes físicos, eléctricos y electrónicos que constituyen una computadora o un sistema informático.
HTML	HyperText Markup Language, o lenguaje de marcado de hipertexto, es un lenguaje para la estructuración de páginas web.
Internet de las Cosas	Se refiere a la interconexión de dispositivos que se encuentran conectados a internet, lo que permite enviar u obtener datos a través de ellos.
Dirección IP	Una dirección IP es un identificador numérico que identifica de manera única a un dispositivo en una red ya sea local o en internet.

Puerto Jack 3.5	Se trata de un conector analógico de audio, comúnmente utilizado para la conexión de audífonos para la reproducción de sonidos o música.
JavaScript	Lenguaje de programación interpretado utilizado para la creación de páginas web dinámicas frontend y backend.
JSON	JavaScript Object Notation, o notación de objetos JavaScript, es un formato de texto para el intercambio de datos. Es ligero, fácil de entender para una persona y es independiente de un lenguaje de programación.
Led	Light-emmiting diode, o diodo emisor de luz, es un componente electrónico que emite luz gracias a la corriente eléctrica.
logger	Logger, o registrador, es un programa encargado de informar de errores o mensajes de alerta al usuario en un formato estructurado y bien definido.
MDF	Medium density fibreboard, o tablero de fibra de densidad media, es una tabla de material similar a la madera comprimida útil para la construcción de soportes y carcasas.
Middleware	Componente de software que se ubica entre el Sistema Operativo y una aplicación, su propósito es facilitar la lógica de intercambio de información entre estos componentes.
nano	Herramienta de edición de textos para la ventana de comando de sistemas basados en Linux/Debian.
NodeJS	Framework de desarrollo web basado en JavaScript.
Nodo	En Node-RED, es el componente de programación más simple dentro del entorno.
NOOBS	New Out Of the Box Software, o software nuevo salido de caja, es una herramienta de Raspberry PI para la instalación de un sistema operativo dentro de una memoria para Raspberry PI.
ONG	Organización no gubernamental.
Open Source	En español significa código libre, hace referencia a el código fuente de software o proyectos que es accesible y libre de ser modificado o distribuido públicamente.
Ping	Es un comando utilizado en la ventana de comandos que mide la latencia de una llamada a una dirección específica.
Plug and Play	Hace referencia a aquel software o dispositivo que está listo para ser utilizado sin necesidad de configuraciones complejas.
Raspbian	Sistema Operativo diseñado para las tarjetas de desarrollo Raspberry PI.

Repositorio	En Github, es un "almacén" de conserva a un código fuente o un proyecto completo de Software.
root	Nombre del usuario con los más altos privilegios en un sistema operativo basado en Linux/Debian.
RPI	Forma corta de referirse a la tarjeta de desarrollo Raspberry PI.
RSC	Abreviación del nombre del área Responsabilidad Social Corporativa.
Script	En programación de computadoras, es una secuencia de comandos que ejecuta funciones dentro del Sistema Operativo
Servidor DHCP	Protocolo de red que asigna automáticamente una dirección IP y configura otros parámetros a los dispositivos (hosts) conectados a éste.
Servomotor	También llamado servo, es un motor diseñado para controlar la posición de un eje en diferentes ángulos.
Shell	También llamado ventana de comandos, es un intérprete de comandos u órdenes que interactúa directamente con el Sistema Operativo.
Software	Son los componentes lógicos que permiten el correcto funcionamiento de un sistema informático.
Teach to Teachers	Significa enseñanza a maestros, representa un nivel de enseñanza que permite a un tutor replicar sus conocimientos a un grupo de estudiantes.
Terminal	También llamado ventana de comandos, es un intérprete de comandos u órdenes que interactúa directamente con el Sistema Operativo.
URL	Uniform Resource Locator, o localizador uniforme de recursos, es un nombre o dirección web asignado a un recurso específico en internet.
WIA	WIA hace referencia al nombre de la página que no es revelado por cuestiones de confidencialidad de la empresa y sus productos.
Win32 disk imager	Programa del sistema operativo Windows que permite la lectura y escritura de tarjetas de memoria.
Workshop	Taller, puede ser teórico o práctico.

Bibliografía

Bisson Jean Carl. (2017). *TJBot Node-RED nodes*. Septiembre 19, 2020, de IBM. Sitio web: https://github.com/jeancarl/node-red-contrib-tjbot

Conway-Jones Dave. (2019). *Linux Installers for Node-RED*. Septiembre 19, 2020, de asociación no especificada. Sitio web: https://github.com/node-red/linux-installers

Duddington Jonathan. (2006). *eSpeak text to speech*. Septiembre 19, 2020, de asociación no especificada. Sitio web: http://espeak.sourceforge.net

IBM Research. (2018). *TJBot*. Septiembre 19, 2020, de IBM Research. Sitio web: https://www.research.ibm.com/tjbot/

Lizarraga Jair. (2018). *TJBot setup*. Septiembre 19, 2020, de asociación no especificada. Sitio web: https://github.com/JairLizarraga/tjbot_setup

Lizarraga Jair. (2018). *Cognitive para todos*. Septiembre 19, 2020, de asociación no especificada. Sitio web: https://cognitiveparatodos.mybluemix.net

OpenJS Foundation. (2016). *Node-RED library*. Septiembre 19, 2020, de OpenJS Foundation. Sitio web: https://flows.nodered.org

OpenJS.Foundation (2016). *Logging*. Septiembre 19, 2020, de OpenJS Foundation. Sitio web: https://nodered.org/docs/user-guide/runtime/logging

O'Leary Nick & Conway-Jones Dave. (2016). *Node-RED*. Septiembre 19, 2020, de OpenJS Foundation. Sitio web: https://nodered.org