TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TUXTLA GUTIERREZ

“DISEÑO Y DESARROLLO DE UN PROTOTIPO PARA LA COMUNICACIO DE PERSONAS CON PROBLEMA DE SORDERA Y PROBLEMAS MOTRICES ATRAVEZ DE UNA APLICACIÓN MOVIL Y UN SISTEMA DE TABLERO ELECTRONICO.”

TESIS

PARA OBTENER EL TÍTULO PROFESIONAL DE:
INGENIERO EN ELECTRÓNICA

PRESENTA:

ORTEGA REYES ALMA GUADALUPE

ASESORES:

ING. ALVARO HERNANDEZ SOL

TUXTLA GUTIERREZ, CHIAPAS. AGOSTO 19 DEL 2017

INDICE DE CONTENIDO

Pág.

INDICE DE FIGURAS

Resumen.

El término comunicación procede del latín communicare que significa “hacer a otro partícipe de lo que uno tiene”. La comunicación es la acción de comunicar o comunicarse, se entiende como el proceso por el que se trasmite y recibe una información. Todo ser humano y animal tiene la capacidad de comunicarse con los demás. Para que un proceso de comunicación se lleve a cabo, es indispensable la presencia de seis elementos: que exista **un emisor**; es decir, alguien que trasmita la información; **un receptor**, alguien a quien vaya dirigida la información y que la reciba; **un contacto** por medio de un canal de comunicación, que puede ser muy variado: el aire por el que circulan las ondas sonoras, el papel que sirve de soporte a la comunicación escrita, la voz, etc. Asimismo, que exista una información o mensaje a transmitir; un código o sistema de signos común al receptor y al emisor, donde el mensaje va cifrado, los signos pueden ser no lingüísticos (símbolos, señales e iconos) y lingüísticos (escrituras, sonidos, concepto asociado, sentido, etc.); y por último, que el mensaje tenga un referente o realidad, al cual alude mediante el código, para que exista una comunicación han de darse, cuando menos, otras dos condiciones, tales como que el canal funcione adecuadamente y no exista ruido. Este último se entiende como toda perturbación que afecte la transmisión del mensaje, sea de carácter auditivo o de cualquier otro tipo. Las interferencias en el medio, la distracción del receptor, los errores lingüísticos son algunos factores que constituyen al ruido. Se tiene también que el receptor conozca el código en el que se cifra el mensaje, si desconoce el determinado código, pues ya no se tendría el significado del mensaje, y hace imposible la comunicación. La comunicación como valor social, es la base de la autoafirmación personal y grupal, ya que a través de ella intercambiamos opiniones y sentimientos con otras personas. Aprender a comunicarse es fundamental para el desarrollo de nuestra personalidad.

Por eso, ante todo, una conversación debe estar rodeada de sinceridad y honestidad. A través de la palabra comunicamos nuestros pensamientos y sentimientos y establecemos relaciones personales con nuestros familiares, amigos, en la escuela, en el trabajo, y en la comunidad. Por lo tanto, cada día debemos esmerarnos más por

lograr perfección en las habilidades de comunicación: hablar, escuchar, escribir y leer. Por otro lado, se tiene como comunicación al escrito breve en que se informa o notifica alguna cosa importante; por ejemplo, el presidente ha transmitido un comunicado oficial. La comunicación aumentativa y alternativa incluye todas las modalidades de comunicación (aparte del habla) utilizadas para expresar pensamientos, necesidades, deseos e ideas. Todos utilizamos este tipo de comunicación cuando usamos gestos, expresiones faciales, símbolos, ilustraciones o escritura. Las personas con graves disfunciones de habla o de lenguaje dependen de la comunicación aumentativa y alternativa para complementar el habla residual o como una alternativa al habla no funcional. Los instrumentos especiales de comunicación aumentativa, como los aparatos electrónicos y los tableros de comunicación con dibujos y símbolos, ayudan a las personas a expresarse y comunicarse. Esto puede mejorar la interacción social, el aprovechamiento escolar y los sentimientos de autoestima. Las personas que utilizan los sistemas aumentativos y alternativos de comunicación no deben de dejar de hablar si son capaces de hacerlo. Estos instrumentos están encaminados a ayudarlos a comunicarse con mayor eficacia.

Introducción.

Existen muchos tipos de sistemas aumentativos y alternativos de comunicación, y por lo general están clasificados en una de dos categorías: con ayuda o sin ayuda.

El mejor sistema de comunicación para una persona dada puede incluir una combinación de modalidades con ayuda y sin ayuda para adaptarse a diversas situaciones. Los sistemas de comunicación sin ayuda no proporcionan salida de voz ni equipo electrónico. El interlocutor tiene que estar presente para que estos sistemas puedan funcionar (no pueden ser usados por teléfono ni para comunicarse con alguien que esté en otra habitación). Algunos ejemplos de este tipo de comunicación incluyen:

*Gestos
*Lenguaje por señales
*Lenguaje corporal
*Tableros de comunicación

Los tableros de comunicación pueden mostrar palabras, letras, números, ilustraciones o símbolos especiales. Los sistemas de comunicación con ayuda son aparatos electrónicos que pueden contar o no con algún tipo de salida de voz. Los instrumentos que brindan salida de voz se denominan comunicadores con salida de voz. Estos aparatos pueden mostrar letras, palabras y frases, o una variedad de símbolos que permiten al usuario construir mensajes. Los mensajes pueden ser comunicados mediante voz electrónica o pueden aparecer impresos en una pantalla o en una cinta de papel. Muchos de estos sistemas pueden también conectarse a una computadora para obtener comunicación por escrito. Algunos de ellos pueden ser programados para producir distintos idiomas.A continuación se explica el diseño, el desarrollo y la etapa de pruebas de un sistema de tableros hospitalario para personas con sordera y problemas motrices, programando una LCD TFT 5” con su shield para facilitar la conexión al arduino mega empleado para la pantalla conectando un arduino nano para la comunicación mediante un módulo ***NFR24L01,***realizando una interfaz con el software arduino, conectando un módulo de blutooth para la comunicación con la aplicación android previamente programada en un móvil.

CAPITULO Ⅰ

Estado del arte.

Ⅰ.1 Historia.

El portal ARASAAC ofrece recursos gráficos y materiales para facilitar la comunicación de aquellas personas con algún tipo de dificultad en esta área. Este proyecto es financiado por el Departamento de Educación Cultura y Deporte del Gobierno de Aragón y coordinado por la Dirección General de Innovación, Equidad y Participación de dicho departamento (ASHA, 2000). En la siguiente figura se muestra los últimos pictogramas realizados. **Fig.I.1**

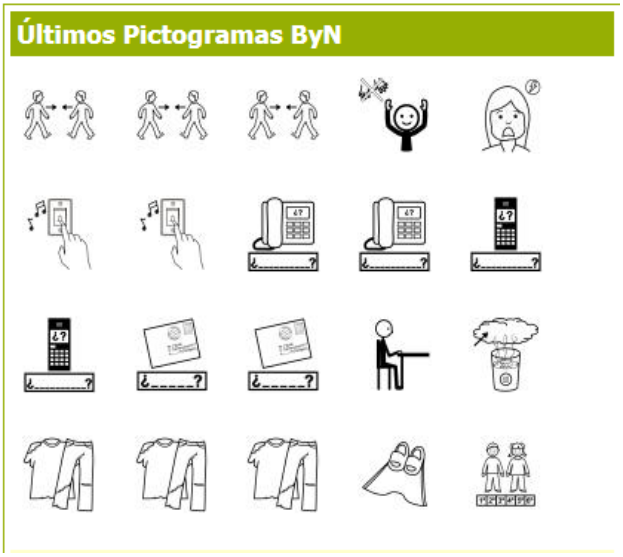

Fig. I.1

¿Qué son los sistemas aumentativos y alternativos de comunicación (saac)?

Los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) son formas de expresión distintas al lenguaje hablado, que tienen como objetivo aumentar (aumentativos) y/o compensar (alternativos) las dificultades de comunicación y lenguaje de muchas personas con discapacidad. **Fig. I.2**

Fig. I.2

La comunicación y el lenguaje son esenciales para todo ser humano, para relacionarse con los demás, para aprender, para disfrutar y para participar en la sociedad y hoy en día, gracias a estos sistemas, no deben verse frenados a causa de las dificultades en el lenguaje oral. Por esta razón, todas las personas, ya sean niños, jóvenes, adultos o ancianos, que por cualquier causa no han adquirido o han perdido un nivel de habla suficiente para comunicarse de forma satisfactoria, necesitan usar un SAAC. Entre las causas que pueden hacer necesario el uso de un SAAC encontramos la parálisis cerebral (PC), la discapacidad intelectual, los trastornos del espectro autista (TEA), las enfermedades neurológicas tales como la esclerosis lateral amiotrofia (ELA), la esclerosis múltiple (EM) o el párkinson, las distrofias musculares, los traumatismos cráneo-encefálicos, las afasias o las pluridis capacidades de tipologías diversas, entre muchas otras.

La **Comunicación Aumentativa y Alternativa (CAA)** no es incompatible sino complementaria a la rehabilitación del habla natural, y además puede ayudar al éxito de la misma cuando éste es posible. No debe pues dudarse en introducirla a edades tempranas, tan pronto como se observan dificultades en el desarrollo del lenguaje oral, o poco después de que cualquier accidente o enfermedad haya provocado su deterioro. No existe ninguna evidencia de que el uso de CAA inhiba o interfiera en el desarrollo o la recuperación del habla.

¿Qué recursos se utilizan?

La Comunicación Aumentativa y Alternativa incluye diversos **sistemas de símbolos**, tanto gráficos (fotografías, dibujos, pictogramas, palabras o letras) como gestuales (mímica, gestos o signos manuales) y, en el caso de los primeros, requiere también el uso de **productos de apoyo**. Los diversos sistemas de símbolos se adaptan a las necesidades de personas con edades y habilidades motrices, cognitivas y lingüísticas muy dispares.

Los **productos de apoyo para la comunicación** incluyen recursos tecnológicos, como los comunicadores de habla artificial o los ordenadores personales y tablets con programas especiales, que permiten diferentes formas de acceso adaptadas algunas para personas con movilidad muy reducida, y facilitan también la incorporación de los diferentes sistemas de signos pictográficos y ortográficos, así como diferentes formas de salida incluyendo la salida de voz. También pueden consistir en recursos no tecnológicos, como los tableros y los libros de comunicación. Para acceder a los ordenadores, comunicadores, tableros o libros de comunicación existen diversas estrategias e instrumentos denominados genéricamente **estrategias y productos de apoyo para el acceso**, tales como los punteros, los teclados y ratones adaptados o virtuales o conmutadores. **Fig. I.3**

Fig. I.3

Los sistemas de símbolos

En líneas anteriores hemos dividido los sistemas de símbolos para la CAA en gestuales y gráficos. En ambos casos encontramos una gradación desde sistemas muy sencillos, que se adaptan a personas con déficits cognitivos y lingüísticos de diversa consideración, hasta sistemas complejos que permiten niveles avanzados de lenguaje signado (basado en signos manuales) o asistido (basado en signos gráficos). **Fig. I.4**

Fig.I.4

En el caso de los **símbolos gestuales**, esta gradación abarca desde el uso de mímica y gestos de uso común hasta el uso de signos manuales, generalmente en el orden correspondiente al lenguaje hablado; es lo que se denomina lenguaje signado o bimodal. **Fig. I.5**

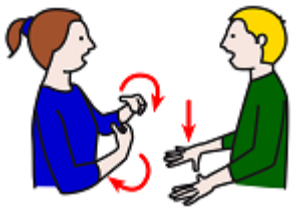

Fig. I.5

Las lenguas de signos utilizadas por las personas no oyentes no se consideran SAAC, ya que constituyen idiomas que se han desarrollado y se adquieren de forma natural, al igual que ocurre con el lenguaje hablado. El uso de signos manuales

requiere disponer de habilidades motrices suficientes, como puede ser el caso de personas con discapacidad intelectual o TEA.

Los **símbolos gráficos** abarcan desde sistemas muy sencillos basados en dibujos o fotografías hasta sistemas progresivamente más complejos como los sistemas pictográficos o la ortografía tradicional (letras, palabras y frases). Gracias a los productos de apoyo para la comunicación y los diversos recursos para el acceso, los sistemas gráficos pueden ser usados por personas con movilidad reducida, incluso en casos de extrema gravedad. Por ello, además de ser usados, como en el caso anterior, por personas con discapacidad intelectual o TEA, los usan también personas con discapacidades motoras (PC, ELA, EM, etc.).

Los **sistemas pictográficos** se aplican a personas que no están alfabetizadas a causa de la edad o la discapacidad. Tienen la ventaja de permitir desde un nivel de comunicación muy básico, que se adapta a personas con niveles cognitivos bajos o en etapas muy iniciales, hasta un nivel de comunicación muy rico y avanzado, aunque nunca tan completo y flexible como el que se puede alcanzar con el uso de la lengua escrita.

Los productos de apoyo para la comunicación

Podemos dividir los productos de apoyo para la comunicación en básicos y tecnológicos. Los **tableros de comunicación** son productos de apoyo básicos que consisten en superficies de materiales diversos en las que se disponen los símbolos gráficos para la comunicación (fotografías, pictogramas, letras, palabras y/o frases) que la persona indicará para comunicarse. **(Fig. I.6)**

Fig. I.6

Cuando los símbolos se distribuyen en varias páginas hablamos de **libros de comunicación**. Entre los productos tecnológicos encontramos los **comunicadores electrónicos** especialmente diseñados para tal fin y los **ordenadores portátiles** o las **tablets** con programas especiales que los convierten en comunicadores. Los comunicadores electrónicos dedicados o emulados en ordenadores se personalizan con los símbolos gráficos que requiere cada persona y se caracterizan por ser portátiles y adaptarse a las formas de acceso apropiadas para cada persona (teclados, ratones, conmutadores, etc.). **Fig. I.7**

Fig. I.7

Disponen de una salida para los mensajes en forma de habla digitalizada o sintetizada, así como también, a menudo, de otras salidas como pantalla, papel impreso o incluso funciones de control del entorno. Por ejemplo, en el sitio web www.utac.cat se pueden encontrar diversas versiones de un vocabulario pictográfico organizado (CACE-UTAC), elaborado con distintos programas de comunicación, y preparado para ser personalizado y usado directamente en ordenadores personales (que quedan así convertidos en comunicadores), así como para ser impreso y construir libros de comunicación. Las versiones para Plaphoons con pictogramas ARASAAC en castellano y en catalán son de acceso libre y gratuito. El resto de versiones son también de libre acceso pero requieren disponer de los programas y/o sistemas de símbolos comerciales correspondientes.

Para indicar los símbolos gráficos en los comunicadores, tableros y libros de comunicación existen cinco estrategias fundamentales, a saber:

- La selección directa: consiste en señalar o pulsar las teclas directamente, con el dedo, con la mirada o con otras partes del cuerpo, para indicar los pictogramas, palabras o letras que se quieren comunicar. Los punteros de distinto tipo son ejemplos de productos de apoyo que puede facilitar la selección o acceso directo.
- La selección con ratón: solamente para productos electrónicos, consiste en acceder con un ratón a teclados o cuadrículas con símbolos para la comunicación en pantalla. Se puede usar una gran variedad de ratones adaptados, en forma de joystick, trackball, así como el ratón facial (controlado con movimientos de la cabeza), el ratón controlado con la mirada o el multimouse, consistente en cinco teclas o conmutadores.
- La exploración o barrido dependiente: solamente en tableros o libros, consiste en que el interlocutor vaya señalando, uno en uno o por grupos, filas y columnas, los símbolos o letras a comunicar, hasta que el hablante asistido indique con un gesto que se ha dado con el que quería comunicar.
- La exploración o barrido independiente: solamente para productos electrónicos, en este caso es el comunicador u ordenador el que presenta las diferentes opciones a comunicar hasta que el hablante asistido selecciona la que le conviene pulsando un conmutador. Existen muchos tipos de conmutadores que se pueden activar con diferentes partes del cuerpo.
- La selección codificada: en este caso cada símbolo o letra tiene un código, por ejemplo un número de dos o tres cifras o un color y un número, de manera que el hablante asistido indica de forma directa o por barrido este código para transmitir el símbolo o letra. De esta forma con pocas teclas o casillas puede acceder a un gran número de símbolos.

Gracias a las diferentes estrategias y productos de apoyo para el acceso, por muy restringida que se encuentre la movilidad de una persona, casi siempre es posible encontrar una solución para que pueda acceder a la comunicación, así como a otras actividades, tales como la movilidad asistida, el control del entorno o el acceso al ordenador para la escritura, el dibujo, el juego o la comunicación a través de la red.

¿Cómo podemos fomentar el éxito de la intervención con saac?

Los sistemas y productos de apoyo para la CAA son solamente un medio o una condición necesaria para que la persona con discapacidad del habla pueda comunicarse, desarrollar sus capacidades y participar en el mundo que la rodea, pero no resultan nunca suficientes. Lo verdaderamente importante es el proceso de educación, habilitación y asesoramiento que debe acompañarlos.

El proceso de intervención debe empezar por una **evaluación** de las capacidades, habilidades, necesidades y deseos de la persona, así como de las características, apoyos, demandas y restricciones de su entorno, con el fin de definir los componentes que va a tener el sistema o sistemas que vayan a resultar más adecuados. Habrá que seleccionar con mucho esmero los productos de apoyo así como las estrategias de acceso y, para usuarios de SAAC no lectores, habrá que realizar una buena selección del vocabulario signado o pictográfico que se va a ir enseñando. Este proceso de evaluación no ha de ser puntual sino continuado a lo largo de la vida.

La **habilitación** y la **enseñanza** deben dirigirse tanto a la persona como a su entorno, incluyendo todos los contextos en los que participa o desea participar, así como todas las personas significativas de estos contextos, incluyendo profesionales y, sobre todo, familiares, compañeros y amigos. Esta enseñanza debe llevarse a cabo en entornos educativos y terapéuticos, pero también en entornos naturales, en un enfoque de 24 horas que garantice que la persona se verá inmersa en un buen ambiente de lenguaje, rodeada de interlocutores sensibles y competentes, e implicada en actividades interesantes y enriquecedoras. Para fomentar el éxito de la intervención con SAAC lo más importante es conseguir que la persona con discapacidad de habla tenga cosas interesantes para comunicar a los demás, sepa

cómo hacerlo y cuente con interlocutores que quieran escucharle y sepan entenderle. Este objetivo no debe dejarse en manos del azar sino que se debe conseguir a través del esfuerzo y el acierto de profesionales competentes, apoyados por una sociedad cada vez más concienciada y libre de prejuicios.

Proyecto TICO

Descripción:

El Proyecto TICO (Tableros Interactivos de Comunicación) es una aplicación informática para generar y utilizar tableros de comunicación de forma interactiva.

El programa se compone de dos aplicaciones independientes y diferenciadas pero complementarias entre sí: Editor e Intérprete.

Con el Editor se pueden crear los tableros que contendrán todos los elementos visuales, auditivos o de control de entorno. El Intérprete permite usar los tableros de comunicación previamente creados con el Editor para superar las limitaciones comunicativas. Esta aplicación está dotada de una función de barrido que hace un recorrido secuencial por los elementos del tablero, con lo que se facilita el acceso a las personas que tienen trastornos graves en la motricidad. Además, los elementos del tablero se pueden agrupar para construir frases. **Fig. I.8**

Fig. I.8

Información adicional:

Esta aplicación está enmarcada dentro del proyecto de colaboración existente entre el Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza y el Colegio Público de Educación Especial Alborada, que tiene como principal objetivo desarrollar herramientas *software* y *hardware* de apoyo a personas con limitaciones físicas o psíquicas**,** y ha estado bajo la dirección de Joaquín Ezpeleta Mateo, Catedrático de esta Universidad, junto con la colaboración de César Canalís Casasús y José Manuel Marcos Rodrigo, profesores del CPEE Alborada de Zaragoza.

Desarrolladores.

- Noviembre 2005: "Desarrollo de herramientas para la creación y utilización de tableros de comunicación en el ámbito de la educación especial" por Fernando Negré Ramos y David Ramos Siguín
- Noviembre 2006: "Tico 2.0: Modificación y extensión de una herramienta para la edición y ejecución de tableros interactivos de comunicación" por Pablo Muñoz Orbañanos
- Agosto 2007: "Intérprete Tico 2.0: Herramienta para la ejecución de tableros interactivos de comunicación y control del entorno" por Antonio Rodríguez Villanueva
- Junio 2008: "Validación automática, basada en reglas, de la accesibilidad de los proyectos Tico" por Beatriz Mateo Mendaza, bajo la co-dirección de Sandra Baldassarri
- Septiembre 2008: "Integración de *plugins* en la aplicación Tico. Desarrollo e integración de un *plugin* para la gestión de una galería de imágenes" por Patricia Martínez Jaray
- Junio 2010: "Tico e1.0: mejora y extensión de la aplicación Tico para su primera distribución estable" por Carolina Palacio Julián
- Febrero 2013: "TICO4Android: Implementación de TICO para dispositivos móviles basados en Android®" por Eduardo Ferrer Domingo

¿Qué otras organizaciones tienen información sobre los sistemas aumentativos y alternativos de comunicación?

La lista no incluye todas las posibles organizaciones que publican información sobre el tema, e inclusión en la misma no constituye aprobación por parte de la Asociación Americana del Habla, Lenguaje y Audición (ASHA, por sus siglas en inglés) de la organización ni del contenido del sitio.

- AAC Institute
- International Society for Augmentative and Alternative Communication
- Rehabilitation Engineering and Assistive Technology Society of North America
- Rehabilitation Engineering Research Center on Communication Enhancement
- State Assistive Technology Programs

Consultar también:

- AAC: Information for AAC Users
- Augmentative Communication: A Glossary
- Augmentative and Alternative Communication Decisions

I.2 Tableros hospitalarios actuales.

I.2.1 Tableros tipos.

Para comunicarse con un usuario del sistema es necesario ir leyendo los símbolos que éste señala. El número de símbolos contenidos en las carpetas o tableros viene determinado por el nivel de vocabulario propio del emisor, así como por la restricciones de espacio que impone el soporte físico sobre el que opera el sistema de representación, de manera que éste pueda ser funcional.El tablero de comunicación tradicional posee tres características que lo hacen un instrumento muy potente:

*La portabilidad y flexibilidad

*Favorece el acercamiento interpersonal

*Su bajo coste

Existen 2 tipos de tableros para ayudar a comunicarse al paciente con la persona responsable de su cargo.

*-Ayudas técnicas para la comunicación de baja tecnología; a*quellos soportes tales como tableros, plafones, carpetas, cuadernos, etc. que utilizan papel impreso y que no incluyen elementos electrónicos.

-Ayudas técnicas para la comunicación de alta tecnología; Incluyen elementos electrónicos, como por ejemplo ordenadores dotados de software y periféricos específicos.

Recursos tecnológicos

*A*yudas técnicas para la comunicación

- Sistemas de baja Sistemas de alta tecnología
    * Cuadernos de comunicación
    * Tableros y plafones
    * Etran
    * Relojes de comunicación

- Sistemas de alta tecnología
    * Comunicadores electrónicos
    * El ordenador

Una vez elegido el sistema de comunicación, deberemos prever y/o en su caso diseñar el modelo concreto de soporte que va a utilizar, cuál será el modo de señalización, los ítems que conformarán el vocabulario, la disposición de los mismos. Elegir el tipo de soporte Probablemente no existe un modelo o soporte perfecto para ningún sujeto, pudiéndose intercambiar en función de las diversas situaciones. Tableros de comunicación: se incluyen aquí varios modelos, incluyendo un tablero de pared, una bandeja sujeta a la silla de ruedas o un tablero portátil. Libros de comunicación: se presentan en diversidad de tamaños, formas, colores, etc., pudiendo estar sujetos a una parte de un tablero de comunicación. Corrientemente se utilizan en formato de carpetas o álbumes de fotos. Tarjetas de comunicación: son similares a un libro a excepción de las páginas, que no están sujetas unas a otras. **(Fig. I.2.1)**

Fig. I.2.1

Ⅰ.2.2 Tableros analógicos.

Numeroso software ha sido implementado para emular sistemas de comunicación no-vocal de símbolos gráficos en un ordenador personal.

Sistemas de acceso alternativo al ordenador

1. Soluciones basadas en el hardware
2. Soluciones surgidas desde el software

Otros dispositivos no electrónicos utilizados como Ayudas Técnicas

- Tecnologías mecánicas
- Tecnologías ópticas
- Otras tecnologías

Soluciones basadas en el hardware

Pulsadores o conmutadores (Dispositivos interruptores de entrada)

Un conmutador o pulsador es, básicamente, un dispositivo que permite cerrar un contacto normalmente abierto, a partir de un movimiento limitado producido por la capacidad residual del sujeto. **(Fig. I.2.2)**

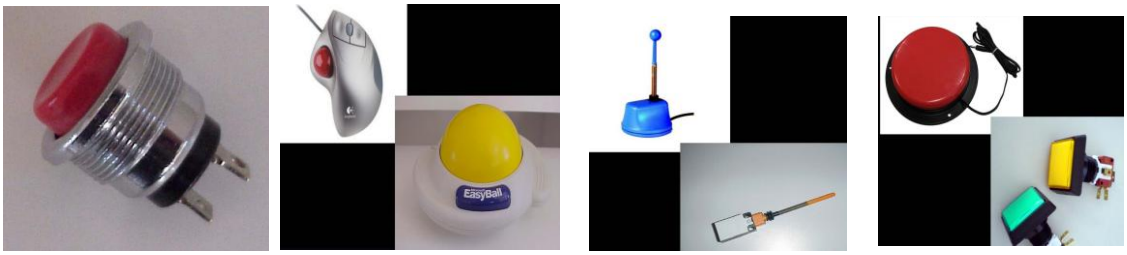

Fig. I.2.2

Ⅰ.2.3 Tableros Digitales.

Cuando nos referimos a un tablero digital, hablamos de una invención que parte de un tablero analógico, haciéndolo más cómodo, más compacto y sobre todo facilita la atención hacia el paciente.

Basados en:

*Objetos reales

*Símbolos gráficos

*Caracteres alfanuméricos (lectoescritura).

Ocupando una LCD para su mayor visualicion como lo son; gestos de uso común, gestos idiosincrásicos ocupando; Códigos gestuales,sistema de signos manuales de los no-oyentes,sistemas de signos manuales pedagógicos, Lenguajes codificados gestuales, comunicación con estructura lingüística compleja, signos tangibles, imágenes Sistemas Pictográficos,sistemas logo gráficos.

Una escritura ortográfica,lenguajes codificados con ayuda (Braille, Morse...),comunicación bimodal y vocabulario Makaton.

Ilustrándolo con imágenes y un botón de reinicio del programa.

1.3.- Planteamiento del problema.

la comunicación con los pacientes de un hospital o personas con problemas de sordera y problemas motrices en común sigue siendo una problemática ya que esto afecta su pronta y mejora atención, pero sobre todo la confianza del paciente o familiares en cuanto al hospital o persona encargadas de su cuidado, el paciente no puede interpretar lo que necesita, lo que afecta el cuidado del mismo, para lo cual se han empleado maneras de poder interpretar lo que necesitan sienten o piensan, desarrollando un tablero digital que reúna necesidades básicas de un paciente adecuado, fácil y cómodo para la utilización del mismo mejora la comunicación entre el paciente y el enfermero o persona designada a su cuidado.

1.4.- Objetivos

1.4.1 Objetivo general.

Desarrollar un software que pueda comunicarse a través de un sistema de tablero hacia una aplicación móvil, para la interpretación de necesidades básicas humanas entre personas con sordera y personas con problemas motrices a personas encargadas del cuidado de estas personas.

1.4.2 Objetivo Específico.

*Diseño del software que comunicara de un sistema de tableros a un dispositivo electrónico.

*Diseño del proyecto para la comunicación desde un sistema de tableros a una aplicación móvil.

*Construcción del proyecto de un sistema de tablero hacia la aplicación móvil.

*Implementación del software en el proyecto sistema de tableros.

1.5.- Justificación

Al implementar un tablero electrónico de necesidades básicas en un hospital o casa habitación, mejorara la comunicación entre el paciente y su cuidador, ya que el paciente podrá interpretar por medio de imágenes vibraciones táctiles lo que le está sucediendo pudiendo o no hablar el paciente será atendido, sin necesidad de estar o no presente, el tablero se comunicara a un servidor que este a su vez enviara un mensaje por medio de una aplicación móvil respondiendo de inmediato al paciente que será atendido enseguida.

CAPITULO Ⅱ

Marco Teórico

Ⅱ.1 Arduino

Ⅱ.1.1 ¿Qué es arduino?

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el Arduino Programming Language (basado en Wiring) y el Arduino Development Environment (basado en Processing). Los proyectos de Arduino pueden ser autonomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo con Flash, Processing, MaxMSP, etc.).

Las placas se pueden ensamblar a mano o encargarlas preensambladas; el software se puede descargar gratuitamente. Los diseños de referencia del hardware (archivos CAD) están disponibles bajo licencia open-source, por lo que eres libre de adaptarlas a tus necesidades.

Arduino recibió una mención honoríca en la sección Digital Communities del Ars Electronica Prix en 2006. **Fig. II.1.1**

ARDUINO™

Fig. II.1.1

II.1.2 ¿Por qué Arduino?

Hay muchos otros microcontroladores y plataformas microcontroladoras disponibles para computación física. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, y muchas otras ofertas de funcionalidad similar. Todas estas herramientas toman los desordenados detalles de la programación de microcontrolador y la encierran en un paquete fácil de usar. Arduino también simplifica el proceso de trabajo con microcontroladores, pero ofrece algunas ventajas para profesores, estudiantes y a aficionados interesados sobre otros sistemas:

***Barato:** Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino preensamblados cuestan menos de $50.00

***Multiplataforma:** El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.

***Entorno de programación simple y claro:** El entorno de programación de Arduino es fácil de usar para principiantes, pero sucientemente flexible para que usuarios avanzados puedan aprovecharlo también. Para profesores, está convenientemente basado en el entorno de programación Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen de Arduino.

***Código abierto y software extensible:** El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerias C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si quieres.

***Código abierto y hardware extensible:**

El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por

lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero. **Fig. II.1.2**

Fig. II.1.2

Ⅱ.2. ¿Qué es arduino uno R3?

Este es el nuevo Arduino Uno R3 utiliza el microcontrolador ATmega328. En adición a todas las características de las tarjetas anteriores, el Arduino Uno utiliza el ATmega16U2 para el manejo de USB en lugar del 8U2 (o del FTDI encontrado en generaciones previas). Esto permite ratios de transferencia más rápidos y más memoria. No se necesitan drivers para Linux o Mac (el archivo inf para Windows es necesario y está incluido en el IDE de Arduino). La tarjeta Arduino Uno R3 incluso añade pins SDA y SCL cercanos al AREF. Es más, hay dos nuevos pines cerca del pin RESET. Uno es el IOREF, que permite a los shields adaptarse al voltaje brindado por la tarjeta. El otro pin no se encuentra conectado y está reservado para propósitos futuros. La tarjeta trabaja con todos los shields existentes y podrá adaptarse con los nuevos shields utilizando esos pines adicionales. El Arduino es una plataforma computacional física open-source basada en una simple tarjeta de I/O y un entorno de desarrollo que implementa el lenguaje Processing/Wiring.

El Arduino Uno R3 puede ser utilizado para desarrollar objetos interactivos o puede ser conectado a software de tu computadora (por ejemplo, Flash, Processing,

MaxMSP). El IDE open-source puede ser descargado gratuitamente (actualmente para Mac OS X, Windows y Linux). **Fig. II.2**

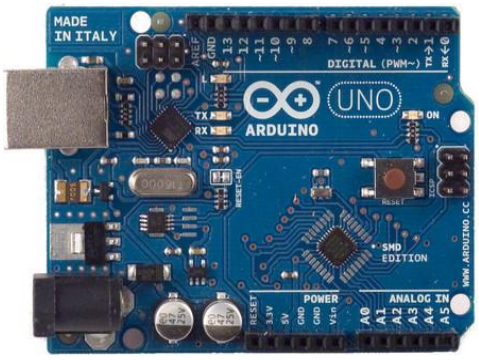

Fig. II.2

II.2.1 Características de alimentación

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

II.3. ¿Qué es arduino mega?

El Arduino Mega es probablemente el microcontrolador más capaz de la familia Arduino. Posee 54 pines digitales que funcionan como entrada/salida; 16 entradas análogas, un cristal oscilador de 16 MHz, una conexión USB, un boton de reset y una entrada para la alimentación de la placa.

La comunicación entre la computadora y Arduino se produce a través del Puerto Serie. Posee un convertidor USB-serie, por lo que sólo se necesita conectar el dispositivo a la computadora utilizando un cable USB como el que utilizan las impresoras. **Fig.II.3.1**

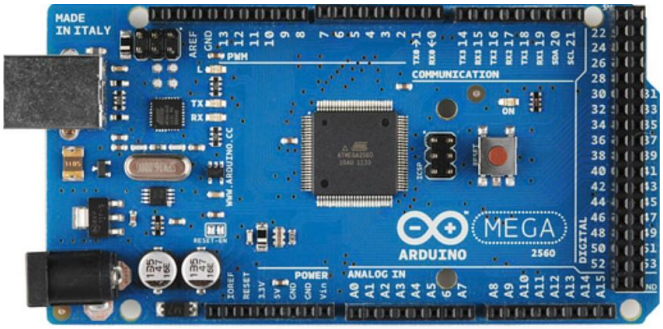

Fig. II.3.1

Ⅱ 3.1 Características de alimentacion

Arduino Mega posee las siguientes especificaciones:

- Microcontrolador: ATmega2560
- Voltaje Operativo: 5V
- Voltaje de Entrada: 7-12V
- Voltaje de Entrada(límites): 6-20V
- Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM)
- Pines análogos de entrada: 16
- Corriente DC por cada Pin Entrada/Salida: 40 mA
- Corriente DC entregada en el Pin 3.3V: 50 mA
- Memoria Flash: 256 KB (8KB usados por el bootloader)
- SRAM: 8KB
- EEPROM: 4KB
- Clock Speed: 16 MHz

Arduino Mega puede ser alimentado mediante el puerto USB o con una fuente externa de poder. La alimentación es seleccionada de manera automática. Cuando se trabaja con una fuente externa de poder se debe utilizar un convertidor AC/DC y regular dicho voltaje en el rango operativo de la placa.

De igual manera se puede alimentar el micro mediante el uso de baterías. Preferiblemente el voltaje debe estar en el rango de los 7V hasta los 12V. Arduino Mega posee algunos pines para la alimentación del circuito aparte del adaptador para la alimentación:

- **VIN:** A través de este pin es posible proporcionar alimentación a la placa.
- **5V:** Podemos obtener un voltaje de 5V y una corriente de 40mA desde este pin.
- **3.3V:** Podemos ubtener un voltaje de 3.3V y una corriente de 50mA desde este pin.
- **GND:** El ground (0V) de la placa.

Ⅱ.4. ¿Qué es arduino nano?

Ⅱ.4.1 Descripción.

Descripción del producto

El Arduino Nano es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o el ATmega168 en sus versiones anteriores (Arduino Nano 2.x) que se usa conectándola a una protoboard. Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B.

Ⅱ.4.2 Características.

- Microcontrolador: Atmel ATmega328 (ATmega168 versiones anteriores)
- Tensión de Operación (nivel lógico): 5 V
- Tensión de Entrada (recomendado): 7-12 V
- Tensión de Entrada (límites): 6-20 V
- Pines E/S Digitales: 14 (de los cuales 6 proveen de salida PWM
- Entradas Analógicas: 8 Corriente máx por cada PIN de E/S: 40 mA
- Memoria Flash: 32 KB (ATmega328) de los cuales 2KB son usados por el bootloader (16 KB – ATmega168)
- SRAM: 2 KB (ATmega328) (1 KB ATmega168)
- EEPROM: 1 KB (ATmega328) (512 bytes – ATmega168)
- Frecuencia de reloj: 16 MHz
- Dimensiones: 18,5mm x 43,2mm

Ⅱ.4.3 Características de alimentación.

El Arduino Nano posee selección automática de la fuente de alimentación y puede ser alimentado a través de:

- Una conexión Mini-B USB.
- Una fuente de alimentación no regulada de 6-20V (pin 30).
- Una fuente de alimentación regulada de 5V (pin 27)

Al alimentar el arduino a través del Mini USB, el FTDI FT232RL proporciona una salida de 3.3V en el pin 16 de la placa. Por ende, cuando se conecta a una fuente externa (no USB), los 3.3V no se encuentran disponibles.

Ⅱ.4.4. Diagrama de pines.

Diagrama de Pines Fig.II.4.4.1

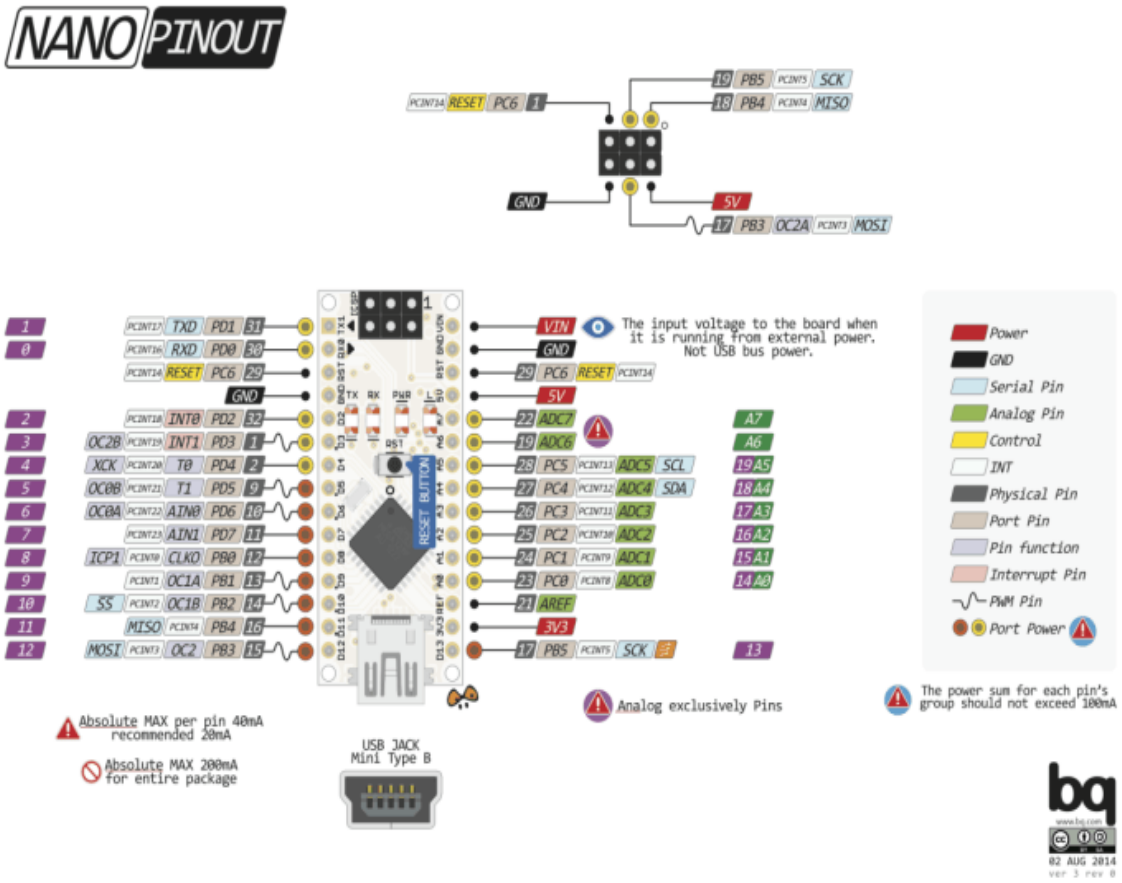

Fig.II.4.4.1

Ⅱ.5 LCD SSD1963 TFT.

Arduino: Pantalla Lcd Tft De 5 800x480 Touch Ssd1963

Agrega gráficos, imagenes, teclado virtual, graficos, etc. con este super LCD de 5 pulgadas en diagonal. Incluye touch screen SPI, lector de memoria SD.

Es de 16 bits. Contyrolador LCD: SSD1963

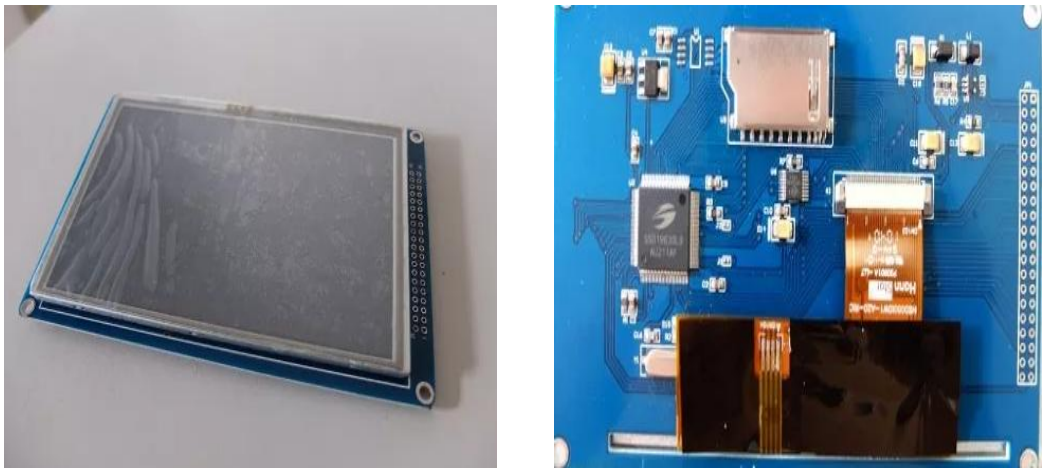

Fig.II.5.1

Specification:

- SSD1963 touch panel controller
- **voltage:2.8-3.3V**
- pcb adapter for lcd
- one sd card socket
- 2*20 Pin 2.54mm double row pin header interface for connecting MCU.
- can be driverd by 8051 / AVR / PIC and other low power controllers.
- Resolution:800x 480 Dots

Ⅱ.5.1 Shield.

Arduino: Shield Para Lcd Tft 2.4/3.2/4.3/5/7 Para Arduino

Ese Shield te sirve para LCDs de 2.4" 3.2" 4.3" 5" 7" con touch screen y SD card que tengan este acomodo de pines**. Fig. 5.1.1**

1	GND	**11**	LCD_DB12	**21**	LCD_DB0	**31**	TouchP _DIN
2	VCC	**12**	LCD_DB13	**22**	LCD_DB1	**32**	TouchP _BUSY
3	N.C.	**13**	LCD_DB14	**23**	LCD_DB2	**33**	TouchP _OUT
4		**14**	LCD_DB15	**24**		**34**	TouchP _Penirq
5	LCD_RS	**15**	LCD_CS	**25**	LCD_DB3	**35**	SD_OUT
6	LCD_WR	**16**	N.C.	**26**	LCD_DB4	**36**	SD_SCK
7	LCD_RD	**17**	LCD_RESET	**27**	LCD_DB5	**37**	SD_DIN
8	LCD_DB8	**18**	N.C.	**28**	LCD_DB6	**38**	SD_CS
9	LCD_DB9	**19**	LED_BL	**29**	LCD_DB7	**39**	N.C.
10	LCD_DB10	**20**	N.C.	**30**	TouchP_CLK	**40**	N.C.

Fig.5.1.1

Descripción:

Soporte: TFT 3.2 " 4.3 " 5.0 " 7.0 "

Este Arduino TFT01 en dos dispositivos adicionales: el panel de control Arduino, recomiendan nuestro Arduino2560, otro adaptador TFT01 al arduino este módulo TFT01 Arduino Shield.

TFT01 LCD es el trabajo en el voltaje 3.3V, por lo que no se puede utilizar directamente en la parte superior de una placa base Arduino estándar, por lo que con el fin de hacer que el TFT01 LCD compatible utilizar estándar Arduino bordo, diseñado esta sección TFT Shield, Tarjeta Arduino Uso del módulo LCD TFT01.

El TFT01 LCD es ahora el modo de apoyo de 16 bits, no existirá encuentro como en 328S, sólo con un conjunto de interfaz de tarjeta SD o interfaz de pantalla táctil Desde Arduino Mega2560 **Fig. 5.1.2**

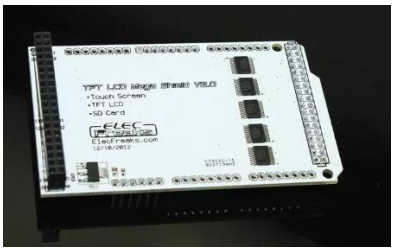

Fig.II.5.1.2

Ⅱ.6. Modulo bluetooth.

El modulo BlueTooth HC-06 utiliza el protocolo UART RS 232 serial. Es ideal para aplicaciones inalámbricas, fácil de implementar con PC, microcontrolador o módulos Arduinos. La tarjeta incluye un adaptador con 4 pines de fácil acceso para uso en protoboard.Los pines de la board correspondientes son:

- VCC
- GND
- RX
- TX

Además, posee un regulador interno que permite su alimentación de 3.6 a 6V.

Ⅱ.6.1 Características.

Características

- Compatible con el protocolo Bluetooth V2.0.
- Voltaje de alimentación: 3.3VDC – 6VDC.
- Voltaje de operación: 3.3VDC.
- Baud rate ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- Tamaño: 1.73 in x 0.63 in x 0.28 in (4.4 cm x 1.6 cm x 0.7 cm)
- Corriente de operación: < 40 mA
- Corriente modo sleep: < 1mA

Transceiver Bluetooth montado en tarjeta base de 4 pines para fácil utilización, interface serial, modo esclavo, Bluetooth v2.0 + EDR, 2.4 GHz, alcance 5 m a 10 m

- Especificación bluetooth v2.0 + EDR (Enhanced Data Rate)
- Modo esclavo (Solo puede operar en este modo)
- Puede configurarse mediante comandos AT (Deben escribirse en mayúscula)
- Chip de radio: CSR BC417143
- Frecuencia: 2.4 GHz, banda ISM
- Modulación: GFSK (Gaussian Frequency Shift Keying)
- Antena de PCB incorporada
  - Potencia de emisión: ≤ 6 dBm, Clase 2
  - Alcance 5 m a 10 m

- Sensibilidad: ≤ -80 dBm a 0.1% BER
- Velocidad: Asincrónica: 2 Mbps (max.)/160 kbps, sincrónica: 1 Mbps/1 Mbps
- Seguridad: Autenticación y encriptación (Password por defecto: 1234)
- Perfiles: Puerto serial Bluetooth
- Módulo montado en tarjeta con regulador de voltaje y 4 pines suministrando acceso a VCC, GND, TXD, y RXD
- Consumo de corriente: 30 mA a 40 mA
- Voltaje de operación: 3.6 V a 6 V
- Dimensiones totales: 1.7 cm x 4 cm aprox.
- Temperatura de operación: -25 ºC a +75 ºC

Ⅱ.6.2 Aplicaciones.

Aplicaciones:

- Comunicación inalámbrica entre microcontroladores
- Comunicación inalámbrica entre computadoras y microcontroladores
- Comunicación inalámbrica entre teléfonos móviles o tabletas y microcontroladores

Ⅱ.6.3 Advertencias.

Advertencias: La comunicación Bluetooth entre dos módulos debe realizarse entre un módulo configurado como maestro y otro como esclavo. Para la comunicación Bluetooth con computador, teléfono, PDA, tableta, etc., el módulo debe ser esclavo. No conecte este dispositivo directamente a un puerto serial de computador, para hacer esto requerirá un conversor de serial TTL a RS232. También puede conectarse por USB utilizando un conversor USB a serial TTL.

Para conexión via inalámbrica con una computadora, la interface o adaptador Bluetooth de esta debe soportar el perfil de puerto serie sobre bluetooth. También se puede conectar uno de estos módulos via cable a la computadora y que se comunique con un segundo módulo, siempre que uno de ellos sea esclavo y el otro maestro.

Ⅱ.6.4 Configuración.

El módulo suele venir configurado con velocidad de transmisión serial de 9600 bps, 1 bit de parada, y sin bit de paridad, nombre: linvor, password: 1234

Para su configuración se puede conectar a el viejo puerto serial RS232 de la computadora a través de un convertidor TTL a RS232, o mejor empleando un conversor USB a serial TTL y utilizando el Hyperterminal de Windows u otro programa con funciones de terminal serial para enviar los comandos AT (Por ej. el SSCOM32, PuTTY, etc.). (A partir de Win Vista el hyperterminal ya no está incluido en el SO). Con Arduino también se puede hacer fácilmente y sin ningún convertidor con un pequeño sketch que utiliza el monitor serial del IDE de Arduino para escribir los comandos AT y observar la respuesta del módulo.

Como este monitor emplea la comunicación serial que el Arduino utiliza para comunicarse con la computadora en los pines 0 y 1 digitales, se crea un puerto serial por software para pasar los datos al módulo Bluetooth empleando los pines digitales 10 y 11.
También se podría hacer con un sketch mas sencillo, directamente conectando el módulo al puerto serie de la placa Arduino (Pines digitales 0 y 1), pero se tendría que remover el microcontrolador del Arduino primero.
Por supuesto también se pueden enviar los comandos AT desde cualquier microcontrolador sin ayuda de computadoras.

Ⅱ.7. Módulo NFR24L01.

El NRF24L01 es un chip de comunicación inalámbrica fabricado por Nordic Semiconductor que podemos conectar a un procesador como Arduino. El NRF24L01 integra un transceptor RF (transmisor + receptor) a una frecuencia entre 2.4GHz a 2.5GHz, una banda libre para uso gratuito. La velocidad de transmisión es configurable entre 250 Kbps, 1Mbps, y 2 Mbps y permite la conexión simultánea con hasta 6 dispositivos. El NRF24L01 también incorpora la lógica necesaria para que la comunicación sea robusta, como corrección de errores y reenvío de datos si es necesario, liberando de esta tarea al procesador. El control del módulo se realiza a través de bus SPI, por lo que es sencillo controlarlo desde un procesador como Arduino. La banda de frecuencia es de 2400 a 2525 MHz, pudiendo elegir entre 125 canales espaciados a razón de 1MHz. Se recomienda usar las frecuencias de 2501 a 2525 MHz para evitar interferencias con las redes Wifi. La tensión de alimentación del NRF24L01 es de 1.9 a 3.6V, aunque los pines de datos son tolerantes a 5V. El consumo eléctrico en Stand By es bajo, y de unos 15mA durante el envío y recepción. Existen dos versiones de módulos que montan el NRF24L01, uno con antena integrada en forma de zig-zag y un alcance máximo de 20-30 metros, y la versión de alta potencia que incorpora amplificador y antena externa, con un alcance máximo de 700-1000 metros. Sin embargo, el alcance real se ve limitado por muchos factores, incluso en condiciones visibilidad directa sin obstáculos. Con el módulo de antena integrada y alimentación desde Arduino y velocidad de transmisión de 2 Mbps el alcance será apenas de 2-3 metros. Un factor de gran impacto en el alcance es la alimentación del módulo. Para conseguir el máximo alcance conviene alimentar el módulo con una fuente externa de 3.3V, estable y con potencia suficiente. Los módulos NRF24L01 son ampliamente empleados por su bajo precio y buenas características. Podemos emplearlos, por ejemplo, para recepción remota de sensores como temperatura presión, aplicaciones de domótica y edificios inteligentes, activación remota de dispositivos como iluminación, alarmas, y control o monitorización de robots en el rango de hasta 700 metros.

Ⅱ.7.1 Esquema de montaje.

La conexión de un módulo RF 2.4 GHz NRF24L01 es sencillo, simplemente alimentamos el integrado desde Arduino mediante GND y 5V y conectamos los pines de datos como vimos en la entrada sobre el bus SPI.**Fig. II.7.1.1**

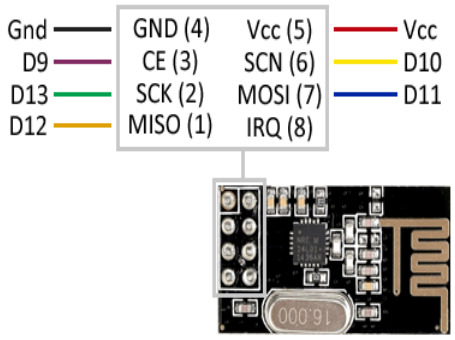

Fig. II.7.1.1

Mientras que la conexión vista desde el lado de Arduino quedaría así.**(Fig.II.7.1.2)**

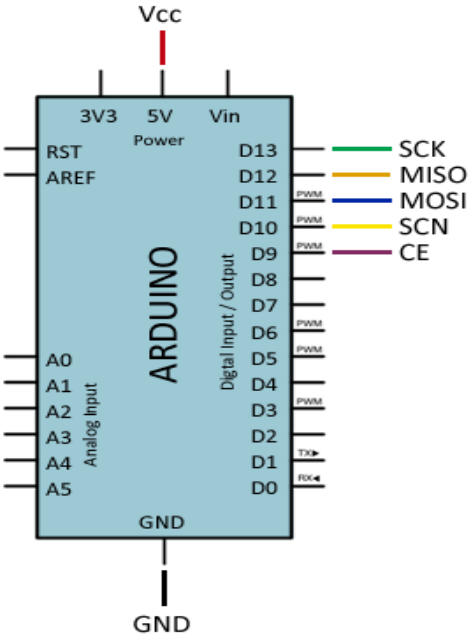

Fig. II.7.1.2

Como hemos dicho, el funcionamiento del módulo funciona si empleamos una fuente externa de 3.3V. En este caso, recordar poner Gnd en común. Para un óptimo funcionamiento también resulta conveniente conectar un condensador de 100pF – 1uF entre Gnd y 3.3V del NRF24L01.

El módulo funciona a 3.3V. No lo alimentéis a 5V o dañaréis el módulo. Los pines de SPI que figuran son válidos para los modelos En Arduino Uno, Nano y Mini Pro.

Ⅱ.8. Android.

Ⅱ.8.1 Que es android.

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, **un núcleo de sistema operativo libre, gratuito y multiplataforma**. El sistema permite programar aplicaciones en una variación de Java llamada Dalvik.

El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Esta sencillez, junto a la existencia de herramientas de programación gratuitas, hacen que una de las cosas más importantes de este sistema operativo sea **la cantidad de aplicaciones disponibles**, que extienden casi sin límites la experiencia del usuario. **Fig.II.8.1.1**

App inventor

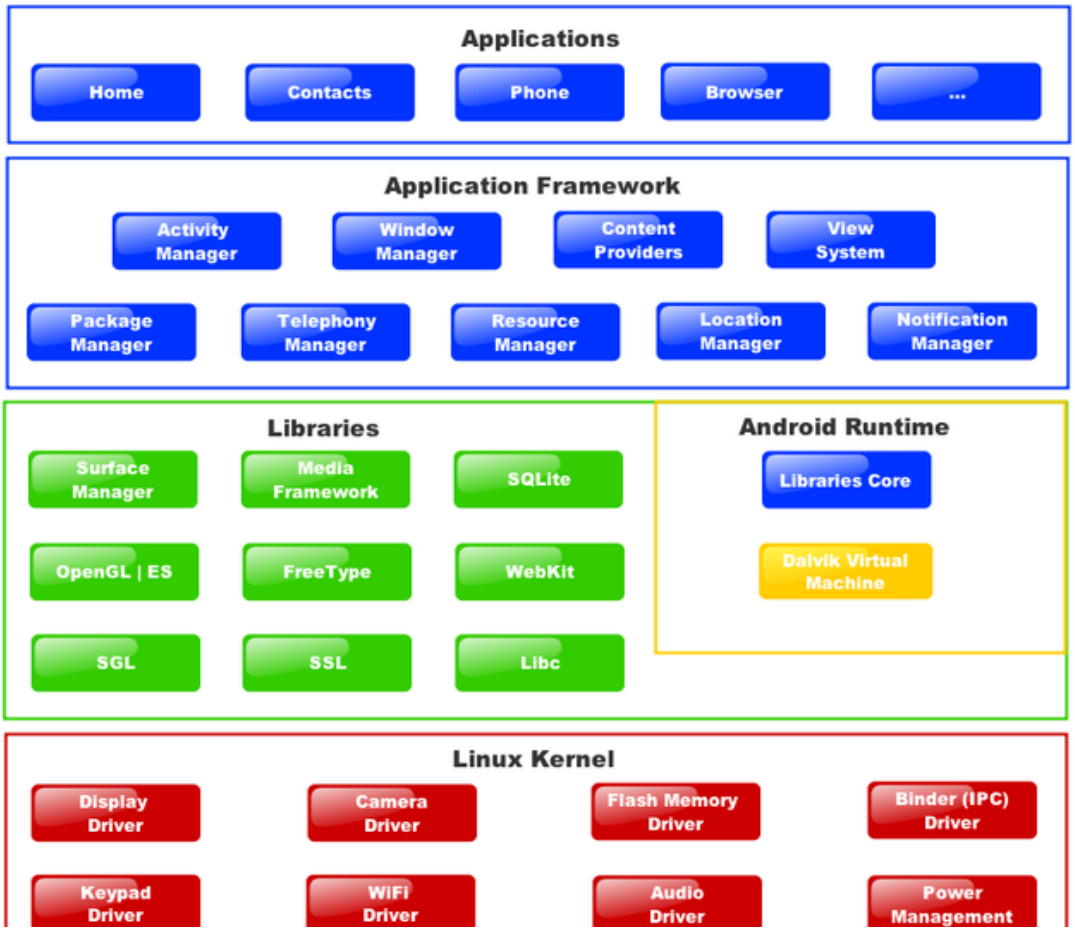

Fig.II.8.1.1

Caracteristicas

El editor de bloques utiliza la libreria Open Blocks de Java para crear un lenguaje visual a partir de bloque. estas librerias estan distribuidad por MIT bajo su licencia libre, el compilador que traduce el lenguaje visual de los bloques para la aplicaciòn en android utiliza Kawa como lenguaje de programacion, distribuido como parte del sistema operativo GNU de la Free Softeare Foundation. **Fig.II. 8.1.2**

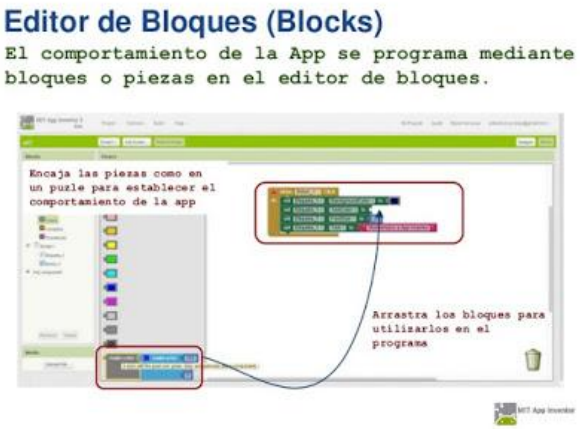

Fig.II. 8.1.2

Con App Inventor se puede tener una aplicacion en funcionamiento en una hora o menos, también se pueden programar aplicaciones mas complejas en menos tiempo que con los lenguajes mas utilizados que son basados en texto. App Inventor se ejecuta como servicio Web que es administrado por el personal del Centro del MIT para el aprendizaje y la creación de aplicaciones móviles.

Configurar El Teléfono

Los Teléfonos compatibles

hay muchos diferentes modelos de teléfonos Android, estos son los Móviles con los que se están utilizando con éxito App Inventor:

- Google: Nexus, One, Nexus S
- Motorola: Droid, Droid X, Droid Incredible
- T-Mobile: G1
- HTC: Increíble, Hero, Desire
- LG: Optimus Me

Sin embargo, hay que tener en cuenta que algunos móviles necesitan las instalaciones del controlador adecuado para trabajar con Windows, para que la App Inventor funcione todo Móvil debe de tener incluida una tarjeta MiCro SD o de

lo contrario no funcionara. Incluso si el teléfono Android no esta en la lista hay posibilidad de que se pueda utilizar App Inventor.

Comprueba la configuración de tu teléfono

Para tener el teléfono listo para utilizar App Inventor , sigue estos pasos:

1.Pulsar el botón de inicio, para entrar en la pantalla principal del teléfono.

2.Pulsar el botón de menú, seleccionar ajustes y luego nos desplazamos hasta aplicaciones. **Fig.II.8.1.3**

Fig. II.8.1.3

3.Activar la casilla de orígenes desconocidos que le permite instalar aplicaciones sin tener que utilizar Google Play.

4.Pulsar en desarrollo. **Fig.8.1.4**

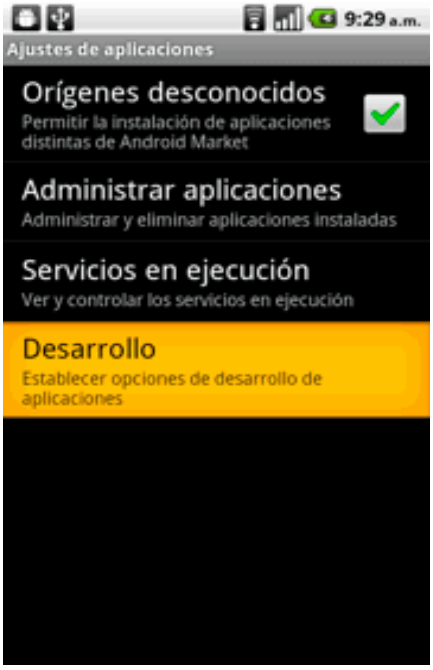

Fig.II.8.1.4

5.Asegurarse que las casillas de verificación tanto de depuración USB como de pantalla activa queden marcadas. **Fig.II.8.1.5**

Fig.II.8.1.5

CAPITULO Ⅲ

Diseño e integración del sistema

Ⅲ.1 Sistema de tableros electrónicos.

Ⅲ.1.1 Introducción.

El paciente con problemas auditivos y motrices tiene problemas para comunicarse con otras personas para lo cual sean desarrollado diferentes formas de comunicación que va desde tarjetas hasta dispositivos electrónicos. El tablero de comunicación electrónico es capaz de comunicar de paciente a enfermero/ cuidador, haciendo más sencillo y practico la atención de paciente; con casilleros de las necesidades básicas y selección táctil.

Ⅲ.1.2 Desarrollo el hardware.

El desarrollo del hardware se divide en 4 etapas explicadas representadas en el siguiente diagrama de bloques. **Fig. II.1.2.1**

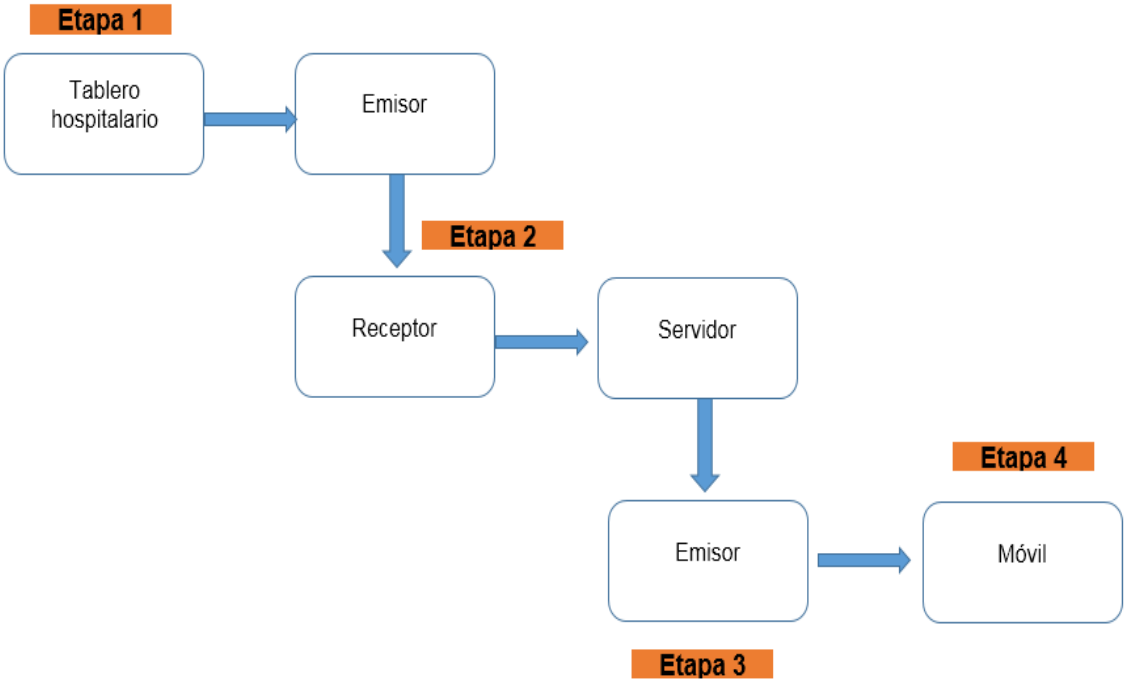

Fig.III.1.2 1

Etapa 1

Tablero hospitalario

La etapa1; está conformada por la LCD ***SSD1963 TFT de 5",*** que va conectada a una shield.

Para programar la LCD se conecta a una placa arduino Mega .**Fig.III.1.2.2**

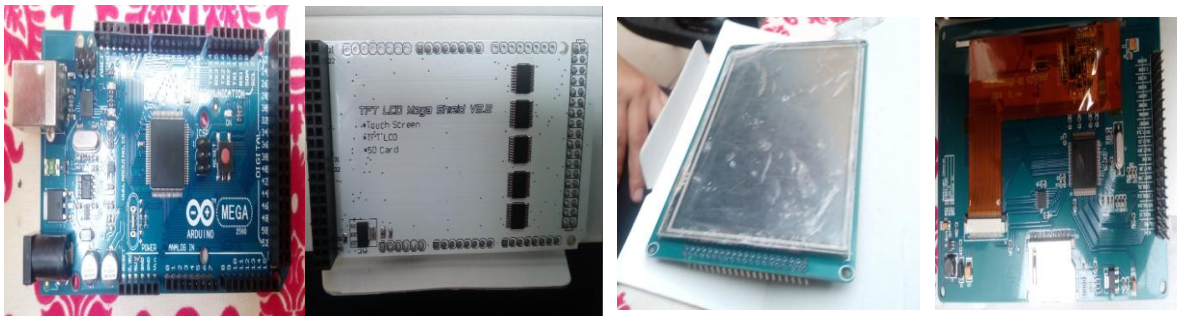

Fig.III.1.2 2

Para el funcionamiento de la primera etapa se descargan las librerías correspondientes a utilizar para poder declararlas. **Fig. III.1.2.3**

residencia_pantalla

```
#if defined(__AVR__)
        #define imagedatatype  unsigned int
#elif defined(__PIC32MX__)
        #define imagedatatype  unsigned short
#elif defined(__arm__)
        #define imagedatatype  unsigned short
#endif

// End of multi-architecture block
#include <tinyFAT.h>  // memoria sd
#include <UTFT_tinyFAT.h>  // libreria para leer la sd de la pantalla
#include <UTFT.h>  // libreria para la pantalla
#include <UTouch.h>// libreria para el touch
#include <UTFT_Buttons.h>  // libreria para los botones
#define ORIENTACION  LANDSCAPE //portrait  // orientacion de la pantalla y del touch
// Declare which fonts we will be using
extern uint8_t BigFont[];  // tamaño de la letra en la pantalla
```

Fig. III.1.2.3

Es importante definir la orientación de nuestra LCD, así como el tamaño, nuestra pantalla nos permite cambiar nuestro tipo de letra, así como su tamaño y color. **Fig. III.1.2.4**

```
// Set up UTFT...
// Set the pins to the correct ones for your development board
// -----------------------------------------------------------
// Standard Arduino 2009/Uno/Leonardo shield   : NOT SUPPORTED DUE TO LACK OF MEMORY
// Standard Arduino Mega/Due shield            : <display model>,38,39,40,41
// CTE TFT LCD/SD Shield for Arduino Due       : <display model>,25,26,27,28
// Standard chipKit Uno32/uC32                 : <display model>,34,35,36,37
// Standard chipKit Max32                      : <display model>,82,83,84,85
// AquaLEDSource All in One Super Screw Shield : <display model>,82,83,84,85
// CC3200 LaunchPad (pins used in the examples): <display model>,15,18,11,32
//
// Remember to change the model parameter to suit your display module!
UTFT          myGLCD(ITDB50,38,39,40,41);  // declaracion de pines en la pantalla y controlador a utilizar

// Set up URTouch...
// Set the pins to the correct ones for your development board
// -----------------------------------------------------------
// Standard Arduino 2009/Uno/Leonardo shield   : NOT SUPPORTED DUE TO LACK OF MEMORY
// Standard Arduino Mega/Due shield            : 6,5,4,3,2
// CTE TFT LCD/SD Shield for Arduino Due       : 6,5,4,3,2
// Standard chipKit Uno32/uC32                 : 20,21,22,23,24
// Standard chipKit Max32                      : 62,63,64,65,66
// AquaLEDSource All in One Super Screw Shield : 62,63,64,65,66
// CC3200 LaunchPad (pins used in the examples): 31,13,19,28,17
//
UTouch        myTouch(6,5,4,3,2);// declaracion de los pines del touch

// Finally we set up UTFT_Buttons :)
```

Fig. III.1.2.4

Se declara los pines a utilizar para nuestra LCD. así como los pines del touch

Empieza con la programación

```
#if defined(__AVR__)
        #define imagedatatype  unsigned int
#elif defined(__PIC32MX__)
        #define imagedatatype  unsigned short
#elif defined(__arm__)
        #define imagedatatype  unsigned short
#endif
```

Se declaran las librerías con include

```
// End of multi-architecture block
#include <tinyFAT.h>  // memoria sd
#include <UTFT_tinyFAT.h>  // libreria para leer la sd de la pantalla
#include <UTFT.h>  // libreria para la pantalla
#include <UTouch.h>// libreria para el touch
#include <UTFT_Buttons.h>  // libreria para los botones
```

Se define la orientación de la pantalla y touch así como el tamaño de letra

```
#define ORIENTACION  LANDSCAPE //portrait  // orientacion de la pantalla y del touch
// Declare which fonts we will be using
extern uint8_t BigFont[];  // tamaño de la letra en la pantalla
```

como comentario se explica las conexiones de la shield así como los pines a utilizar

```
// Set up UTFT...
// Set the pins to the correct ones for your development board
// -----------------------------------------------------------
// Standard Arduino 2009/Uno/Leonardo shield    : NOT SUPPORTED DUE TO LACK OF MEMORY
// Standard Arduino Mega/Due shield          : <display model>,38,39,40,41
// CTE TFT LCD/SD Shield for Arduino Due      : <display model>,25,26,27,28
// Standard chipKit Uno32/uC32               : <display model>,34,35,36,37
// Standard chipKit Max32                    : <display model>,82,83,84,85
// AquaLEDSource All in One Super Screw Shield : <display model>,82,83,84,85
// CC3200 LaunchPad (pins used in the examples): <display model>,15,18,11,32//
// Remember to change the model parameter to suit your display module!
UTFT      myGLCD(ITDB50,38,39,40,41);  // declaracion de pines en la pantalla y controlador a utilizar
```

```
// Set up URTouch...
// Set the pins to the correct ones for your development board
// ----------------------------------------------------------
// Standard Arduino 2009/Uno/Leonardo shield  : NOT SUPPORTED DUE TO LACK OF MEMORY
// Standard Arduino Mega/Due shield           : 6,5,4,3,2
// CTE TFT LCD/SD Shield for Arduino Due      : 6,5,4,3,2
// Standard chipKit Uno32/uC32                : 20,21,22,23,24
// Standard chipKit Max32                     : 62,63,64,65,66
// AquaLEDSource All in One Super Screw Shield : 62,63,64,65,66
// CC3200 LaunchPad (pins used in the examples): 31,13,19,28,17
```

Declarando los pines del touch y quienes intervienen en los botones, quien utilizara la SD, ya que ahí se almacenarán las imágenes de los botones

```
//
UTouch        myTouch(6,5,4,3,2);// declaracion de los pines del touch
// Finally we set up UTFT_Buttons :)
UTFT_Buttons  myButtons(&myGLCD, &myTouch); // se declara quienes intervienen en los botones (touch y pantalla)
UTFT_tinyFAT myFiles(&myGLCD); // se declara quien utilizara la sd
Int but1, but2, but3, but4,
but5,but6,but7,but8,but9,but10,but11,but12,but13,but14,but15,but16,but17,but18,
pressed_button;
// se declara las variables para darle nombre a los botones
```

Etapa 2

Emisor-Receptor

La etapa 2: en la etapa 2 se lleva acabo las conexione para la comunicación a través de una interfaz, para la conexión del tablero hospitalario hacia el servidor (etapa 3), se consideran módulos NFRl01 de comunicación por su bajo costo y su eficaz conectado un módulo a la LCD haciendo la función de un emisor anexando un arduino nano a la conexión del tablero por la variación del voltaje, ya que al utilizar la shield no dejo espacio para poder conectar el modulo y por ende escogiendo un arduino nano para su mayor compactación del equipo. **Fig.III.1.2.5**

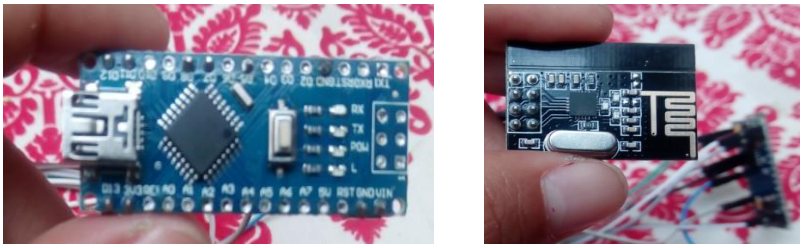

Fig. III.1.2.5

En esta etapa se programan los módulos uno como emisor y el segundo como receptor. Se empiezan a incluir las librerías para el emisor. **Fig.III.1.2.6**

Incluyendo librería

receptor_pwm_nano

```
// ARDUINO NANO
#include <SPI.h>  // libreria para lacomunicacion spi para el modulo
#include "nRF24L01.h" // libreria para el modulo
#include "RF24.h"
RF24 radio(7,8);  // ce, scn

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };
```

Fig. III.1.2.6

Se comienza a programar el emisor

```
// ARDUINO NANO
#include <SPI.h>  // libreria para la comunicacion spi para el modulo
#include "nRF24L01.h" // libreria para el modulo
#include "RF24.h"
RF24 radio(7,8);  // ce, scn
const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL }
```

Etapa 3

Servidor-Emisor

Etapa 3: en esta etapa se desarrolla el servidor quien hace la función de emisor y receptor, al recibir los datos enviados de la LCD conformado por un arduino uno, un módulo bluetooth para la comunicación entre módulos y un módulo NFR24L01 como receptor.

Haciendo la comunicación a través de un módulo NFR24L01 leyendo, interpretando y codificándola para hacer la función de un emisor y enviarla a un dispositivo móvil en este caso un móvil Android-aplicación, a través. **Fig. III.1.2.7**

Fig. III.1.2.7

El servidor almacenara los datos recibidos a travezdel modulo y los enviara a la aplicación móvil, empezando con la programación incluiremos las librerías. **Fig.III.1.2.8**

Incluyendo librerías

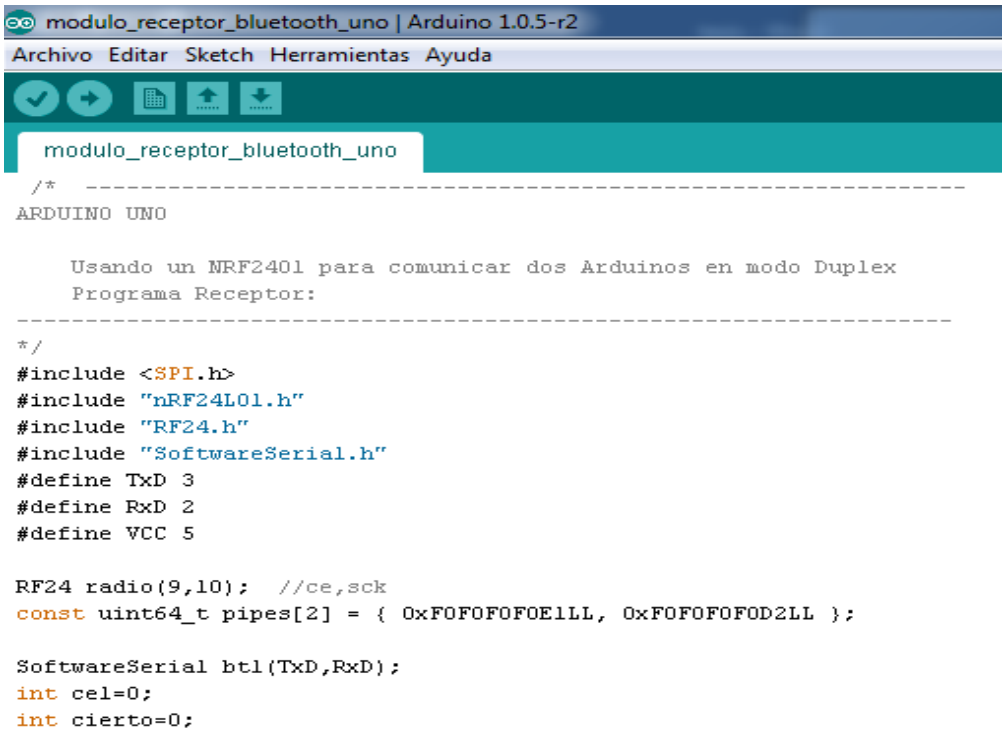

Fig. III.1.2.8

RF24 radio(9,10); //ce,sck

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

SoftwareSerial bt1(TxD,RxD);

int cel=0;

int cierto=0;

Etapa 4

Móvil.

Etapa 4: en esta etapa el servidor envía los datos ya codificados por medio de un mensaje a un teléfono inteligente, capaz de leerla a través de una aplicación Android en App inventor. **Fig. III.1.2.9**

Fig. III.1.2.9

Programando en App inventor. Fig.III.1.2.10

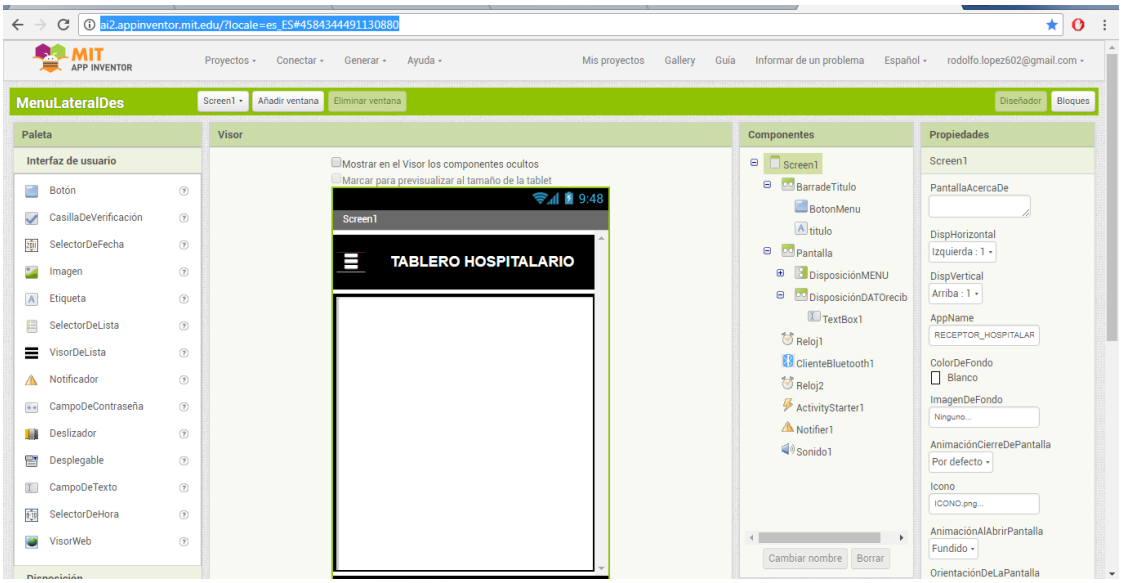

Fig. III.1.2.10

Programando a bloques la aplicación en App inventor. Fig.III.1.2.11 (pantalla 1)- Fig.III.1.2.12 (pantalla 2)

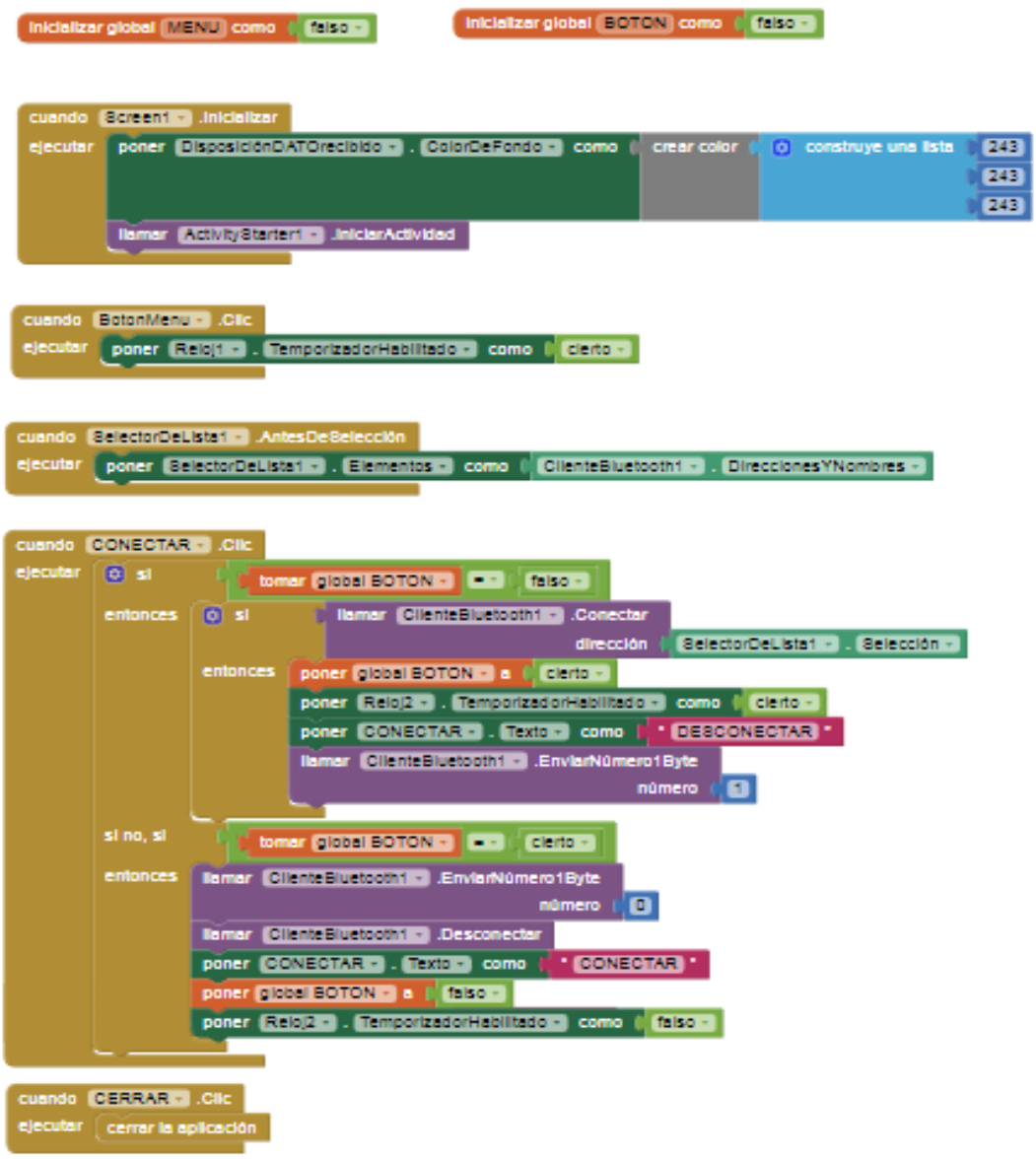

Fig. III.1.2.11

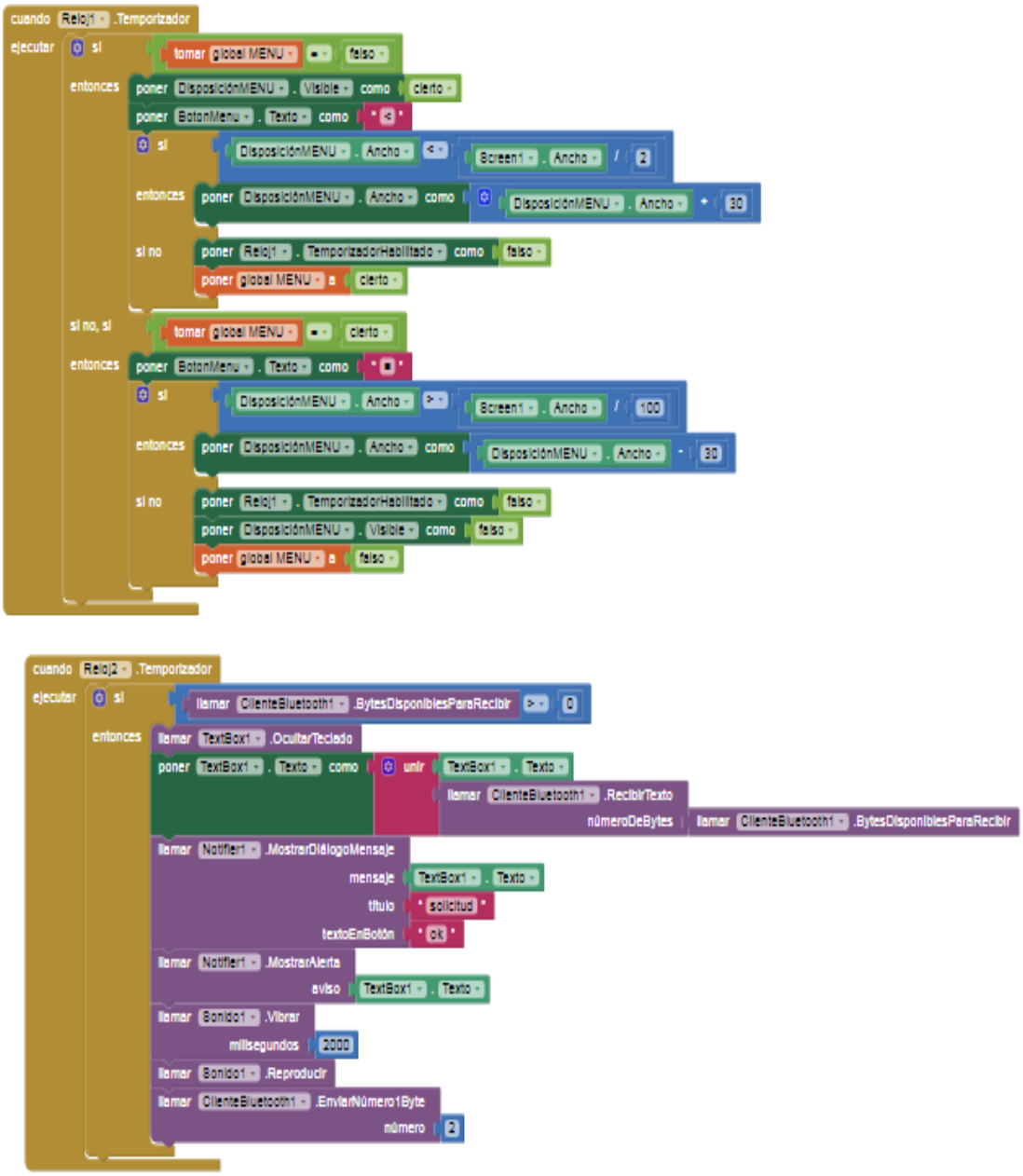
cuando Reloj1 .Temporizador
ejecutar si
tomar global MENU = falso
entonces poner DisposiciónMENU . Visible como cierto
poner BotonMenu . Texto como "<"
si DisposiciónMENU . Ancho < Screen1 . Ancho / 2
entonces poner DisposiciónMENU . Ancho como DisposiciónMENU . Ancho + 30
si no poner Reloj1 . TemporizadorHabilitado como falso
poner global MENU a cierto
si no, si tomar global MENU = cierto
entonces poner BotonMenu . Texto como "■"
si DisposiciónMENU . Ancho > Screen1 . Ancho / 100
entonces poner DisposiciónMENU . Ancho como DisposiciónMENU . Ancho - 30
si no poner Reloj1 . TemporizadorHabilitado como falso
poner DisposiciónMENU . Visible como falso
poner global MENU a falso
cuando Reloj2 .Temporizador
ejecutar si
llamar ClienteBluetooth1 .BytesDisponiblesParaRecibir > 0
entonces llamar TextBox1 .OcultarTeclado
poner TextBox1 . Texto como unir TextBox1 . Texto
llamar ClienteBluetooth1 .RecibirTexto
númeroDeBytes llamar ClienteBluetooth1 .BytesDisponiblesParaRecibir
llamar Notifier1 .MostrarDiálogoMensaje
mensaje TextBox1 . Texto
título "solicitud"
textoEnBotón "ok"
llamar Notifier1 .MostrarAlerta
aviso TextBox1 . Texto
llamar Sonido1 .Vibrar
milisegundos 2000
llamar Sonido1 .Reproducir
llamar ClienteBluetooth1 .EnviarNúmero1Byte
número 2

Fig. III.1.2.12

Diagrama esquemático Fig.III.1.2.13

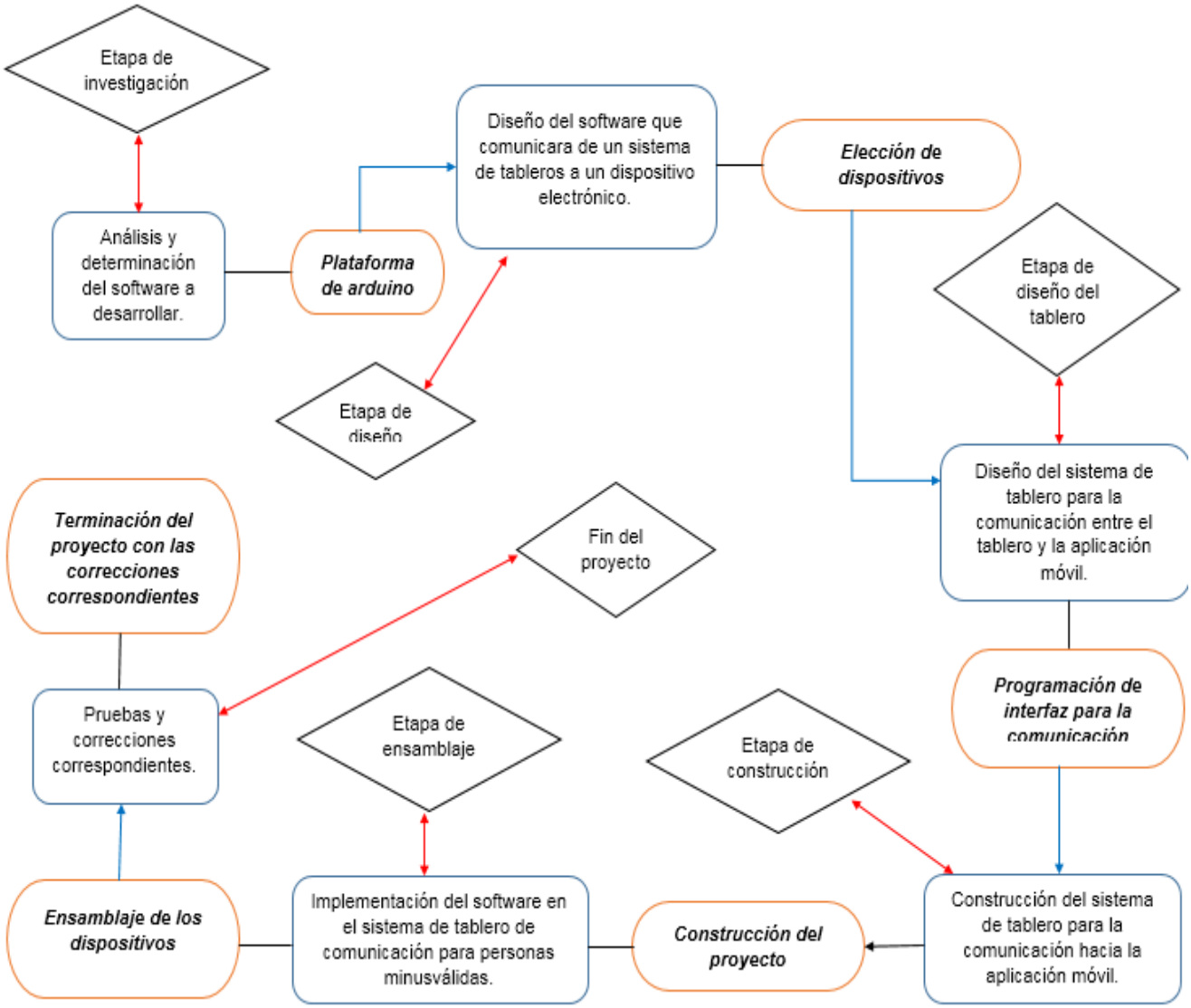

Fig. III.1.2.13

Etapa de investigación

Análisis y determinación del software a desarrollar, la primer etapa dl proyecto y de vital importancia, se determina que software se ocupara para el desarrollo del hardware. Se determina plataforma de arduino para el desarrollo del proyecto.

Etapa de diseño

Diseño del software que comunicara de un sistema de tableros a un dispositivo electrónico, se diseña el prototipo a utilizar para el proyecto por lo que en este tapase hace la selección de dispositivos

Etapa de diseño del tablero

Diseño del sistema de tablero para la comunicación entre el tablero y la aplicación móvil. determinación de la interfaz y el diseño del servidor para la programación de interfaz para la comunicación del sistema de tableros.

Etapa de construcción

Construcción del sistema de tablero para la comunicación hacia la aplicación móvil. En esta etapa se determinan que dispositivos se ocuparan para el ensamblaje del proyecto.

Etapa de ensamblaje

Implementación del software en el sistema de tablero de comunicación para personas minusválidas. Se hace la simulación del proyecto, se lleva a practica y se hace el ensamblaje de los dispositivos.

Etapa de diseño

Pruebas y correcciones correspondientes, se hacen las pruebas y se observan los resultados de ser necesario se cambian los dispositivos.

Terminación del proyecto con las correcciones correspondientes

Ⅲ.2 Electrónica.

Ⅲ.2.1 selección de materiales.

En la selección de materiales se trata de reducir en cuanto al costo y la facilidad de obtener los materiales así como el manejo de estos.

Para LCD TFT.

*Arduino mega 2560

*Arduino nano mega 328

*LCD TFT “5 ***SSD1963***

*Memoria SD

*shield para LCD TFT “5 SSD1963

*Módulo NFR24L01

*Batería de 9v recargable

*software Arduino 1.0.5

*Memoria SD

Para el servidor.

*Arduino uno R3

*Módulo NFR24L01

*Módulo Buetooth

*Batería o fuente de poder de 5v

*software Arduino 1.0.5

Para la aplicación.

*móvil con Android

*Aplicación app inventor

CAPITULO Ⅳ

Pruebas del instrumento

Ⅳ.1 análisis del desempeño del sistema.

El desempeño del sistema es muy bueno, la comunicación tiene un retardo de 30 milisegundos, la sensibilidad del touch es el esperado. Como se observa en **Fig.IV.1**.

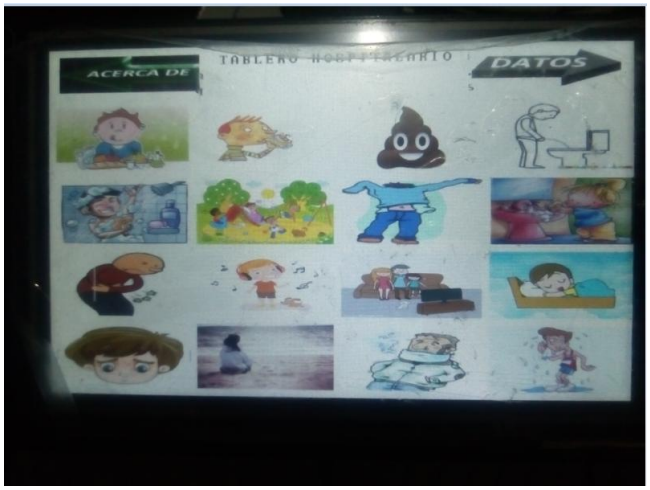

Fig.IV 1

se configuraron 12 botones para las necesidades básica, un botón para los datos del programador y uno para los datos dela escuela. El sistema cuenta con un sistema de apagado por medio de un interruptor, un botón para reiniciar el equipo de ser necesario y una pila recargable Li-Po de 850. A 9v. para lo cual se le adapto un regulador de voltaje a la entrada al arduino.

Fig.IV 2.1

CAPITULO Ⅴ

Resultados

Ⅴ.1 Resultados

El tablero electrónico, funciona en su totalidad, se lograron los siguientes objetivos.

*Diseño del software que comunica de un sistema de tableros a un dispositivo electrónico.

*Diseño del proyecto para la comunicación desde un sistema de tableros a una aplicación móvil.

*Construcción del proyecto de un sistema de tablero hacia la aplicación móvil.

*Implementación del software en el proyecto sistema de tableros.

En base a los resultados realizados y pruebas de precisión todo los objetivos planteados fueron realizados.

Conclusiones

El trabajo se cumplió en su totalidad dejando pendiente la carcasa del equipo así como una notificación de recibido por parte del móvil, la comunicación entre personas con capacidades diferentes en mayor parte es difícil de realizar ya que no se tienen el equipo ni las personas que puedan interpretar por completo lo que el paciente necesita, el sistema de tableros cubre en su totalidad estos problemas, realizando la pruebas necesarias se llego al buen funcionamiento de este tablero electrónico.

Trabajo futuro

Los objetivos planteados se llevaron a cabo en su totalidad, para trabajos futuros se podrían emplear más detalles en la comunicación, así como el diseño siendo un más compacto y menos pesado y con ella una carcasa en algún material resistente.

Referencias

MCI Electronic .(2017). Recuperado de http://arduino.cl/que-es-arduino/

MCI Electronic .(2017). Recuperado de http://arduino.cl/arduino-uno/

García G. (2013). Panama Hiteke. Recuperado dehttp://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

ELECTRONILB.(2017).Recuperado de.https://electronilab.co/tienda/modulo-bluetooth-hc-06-serial-rs232ttl/

ELECTRONILB.(2017).Recuperado de.http://www.electronicoscaldas.com/modulos-rf/482-modulo-bluetooth-hc-06.html

Luis L. (2016). Ingenieria, Informática y diseño. https://www.luisllamas.es/comunicacion-inalambrica-a-2-4ghz-con-arduino-y-nrf24l01/

Xataka A. (2017).https://www.xatakandroid.com/sistema-operativo/que-es-android

Arduino p. (2017)http://arduino.cc/en/Main/ArduinoBoardMega2560

Dolores A. , Abadín C. , Delgado S. (2010) Recuperado de. http://www.asha.org/public/speech/disorders/Los-Sistemas-Aumentativos-y-Alternativos-de-Comunicacion/

ARASAAC.(2017). Recuperado de.http://www.arasaac.org/software.php

General. C. (2014). Recuperado de. http://conceptodefinicion.de/comunicacion/

David C. (2011). Recuperado de. http://www.crmfalbacete.org/recursosbajocoste/archivos/pdf/SAACs.pdf

Anexos

Se anexa las fotografías del resultado del tablro hospitalario. Fig.1-3.

Fig. 1

Fig. 2

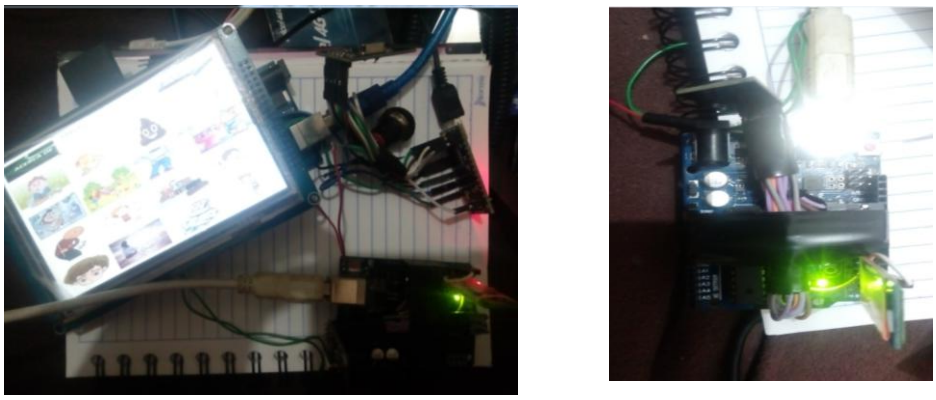

Fig. 3

Se anexa la programación de todo el sistema.

Código LCD TFT SSD1963

```
#if defined(__AVR__)
#define imagedatatype  unsigned int
#elif defined(__PIC32MX__)
#define imagedatatype  unsigned short
#elif defined(__arm__)
#define imagedatatype  unsigned short
#endif
// End of multi-architecture block
#include <tinyFAT.h>  // memoria sd
#include <UTFT_tinyFAT.h>  // libreria para leer la sd de la pantalla
#include <UTFT.h>  // libreria para la pantalla
#include <UTouch.h> // libreria para el touch
#include <UTFT_Buttons.h>  // libreria para los botones
#define ORIENTACION  LANDSCAPE //portrait  // orientacion de la pantalla y del touch
// Declare which fonts we will be using
extern uint8_t BigFont[];  // tamaño de la letra en la pantalla

// Set up UTFT...
// Set the pins to the correct ones for your development board
// ----------------------------------------------------------
--
// Standard Arduino 2009/Uno/Leonardo shield  : NOT SUPPORTED DUE TO LACK OF MEMORY
```

```
// Standard Arduino Mega/Due shield            : <display model>,38,39,40,41
// CTE TFT LCD/SD Shield for Arduino Due       : <display model>,25,26,27,28
// Standard chipKit Uno32/uC32                 : <display model>,34,35,36,37
// Standard chipKit Max32                      : <display model>,82,83,84,85
// AquaLEDSource All in One Super Screw Shield : <display model>,82,83,84,85
// CC3200 LaunchPad (pins used in the examples): <display model>,15,18,11,32
//
// Remember to change the model parameter to suit your display module!
UTFT          myGLCD(ITDB50,38,39,40,41);  // declaracion de pines en la pantalla y controlador a utilizar
// Set up URTouch...
// Set the pins to the correct ones for your development board
// -----------------------------------------------------------
--
// Standard Arduino 2009/Uno/Leonardo shield  : NOT SUPPORTED DUE TO LACK OF MEMORY
// Standard Arduino Mega/Due shield            : 6,5,4,3,2
// CTE TFT LCD/SD Shield for Arduino Due       : 6,5,4,3,2
// Standard chipKit Uno32/uC32                 : 20,21,22,23,24
// Standard chipKit Max32                      : 62,63,64,65,66
// AquaLEDSource All in One Super Screw Shield : 62,63,64,65,66
// CC3200 LaunchPad (pins used in the examples): 31,13,19,28,17
//
UTouch        myTouch(6,5,4,3,2);// declaracion de los pines del touch
// Finally we set up UTFT_Buttons :)
```

```
UTFT_Buttons  myButtons(&myGLCD, &myTouch); // se declara quienes intervinen en los botones (touch y pantalla)
UTFT_tinyFAT myFiles(&myGLCD); // se declara quien utilizara la sd
int but1, but2, but3, but4, but5,but6,but7,but8,but9,but10,but11,but12,but13,but14,but15,but16,but17,but18,pressed_button; // se declara las variables para darle nombre a los botones
```

En el void setup

```
void setup()
{
  pinMode(10,OUTPUT);
    myGLCD.InitLCD(ORIENTACION); // se inicializa la lcd con la orientacion deseada
  myGLCD.clrScr(); // se limpia la lcd
  myGLCD.setFont(BigFont); // se declara el tamaño de letra que se va a usar
 file.initFAT(); // se inicializa el almacenamiento de la sd
  myTouch.InitTouch(ORIENTACION);  // se inicializa el touch y se define la orientacion landscape es horizontal y portrait es vertical
myTouch.setPrecision(PREC_MEDIUM);
//      myGLCD.drawBitmap (0, 0, 187, 88, popo);

myButtons.setTextFont(BigFont); // se define el tamaño de la letras de los botones
//  myFiles.loadBitmap(10,80,190,90,"comer.RAW");  // se manda a leer la sd para imprimir en la lcd una imagen
```

Nombrando botones

```
but1 = myButtons.addButton( 15,  10, 200,  50, "ACERCA DE"); // x , y ,
10pixeles      x 15 pixeles , nombre de la imagen
but2 = myButtons.addButton( 585,  10, 200,  50, "DATOS");
but3 = myButtons.addButton( 10,  80, 190,  90,  "COMER"); //X,Y
but4 = myButtons.addButton( 210,  80, 190, 90, "BEBER");
but5 = myButtons.addButton( 410, 80, 190, 90, "POPO");
but6 = myButtons.addButton( 610,  80, 190,  90, "PIPI");
but7 = myButtons.addButton( 10, 180, 190, 90, "BANARSE");
but8 = myButtons.addButton( 210, 180, 190, 90, "JUGAR");
but9 = myButtons.addButton( 410,  180, 190,  90, "CAMBIARSE");
but10 = myButtons.addButton( 610, 180, 190, 90, "LAVARSE");
but11 = myButtons.addButton( 10,  280, 190,  90, "DOLOR");
but12 = myButtons.addButton( 210, 280, 190, 90, "OIR MUSICA");
but13 = myButtons.addButton( 410,  280, 190,  90, "VER TV");
but14 = myButtons.addButton( 610, 280, 190, 90, "DORMIR");
but15 = myButtons.addButton( 10, 380, 190, 90, "ESTOY TRISTE");
but16 = myButtons.addButton( 210,  380, 190,  90, "ESTOY SOLO");
but17 = myButtons.addButton( 410, 380, 190, 90, "FRIO");
but18 = myButtons.addButton( 610, 380, 190, 90, "CALOR");
acerca(); // funcion para imprimir la presentacion
}
```

Se cierra la función de void setup

Programando en void loop

```
void loop()
{ myGLCD.clrScr(); // se limpia la pantalla
myGLCD.fillScr(VGA_WHITE);
   // se configuran los botones, posicion y lo que dira cada boton
myButtons.drawButtons();  // se imprimen en la pantalla los botones configurados arriba
```

para mandar a llamar las imágenes para cada botón

```
myFiles.loadBitmap(10,80,190,90,"comer.RAW");
myFiles.loadBitmap(210,80,190,90,"beber.RAW");
myFiles.loadBitmap(410,80,190,90,"popo.RAW");
myFiles.loadBitmap(610,80,190,90,"pipi.RAW");
myFiles.loadBitmap(10,180,190,90,"banarse.RAW");
myFiles.loadBitmap(210,180,190,90,"jugar.RAW");
myFiles.loadBitmap(410,180,190,90,"c ambio.RAW");
myFiles.loadBitmap(610,180,190,90,"lavarse.RAW");
myFiles.loadBitmap(10,280,190,90,"duele.RAW");
myFiles.loadBitmap(210,280,190,90,"musica.RAW");
myFiles.loadBitmap(410,280,190,90,"tele.RAW");
myFiles.loadBitmap(610,280,190,90,"dormir.RAW");
myFiles.loadBitmap(10,380,190,90,"triste.RAW");
myFiles.loadBitmap(210,380,190,90,"soledad.RAW");
myFiles.loadBitmap(410,380,190,90,"frio.RAW");
myFiles.loadBitmap(610,380,190,90,"calor.RAW");
myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
// se lee la memoria sd para imprimir las imagenes en la pantalla
```

```
  myGLCD.setColor(VGA_BLACK);
  myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("TABLERO HOSPITALARIO",CENTER,10); // left izquierda
```

Se empieza con las condiciones

```
 while(1)
 {
   if (myTouch.dataAvailable() == true)  // si se detecta que se preciono un boton
   {
     pressed_button = myButtons.checkButtons();
// se condicionan los botones que se precionaron
if (pressed_button==but1)
     {
      acerca();
       myGLCD.clrScr();
         myGLCD.fillScr(VGA_WHITE);
// se configuran los botones, posicion y lo que dira cada boton
  myButtons.drawButtons();  // se imprimen en la pantalla los botones configurados arriba
  myFiles.loadBitmap(10,80,190,90,"comer.RAW");
  myFiles.loadBitmap(210,80,190,90,"beber.RAW");
  myFiles.loadBitmap(410,80,190,90,"popo.RAW");
  myFiles.loadBitmap(610,80,190,90,"pipi.RAW");
  myFiles.loadBitmap(10,180,190,90,"banarse.RAW");
  myFiles.loadBitmap(210,180,190,90,"jugar.RAW");
  myFiles.loadBitmap(410,180,190,90,"cambio.RAW");
  myFiles.loadBitmap(610,180,190,90,"lavarse.RAW");
  myFiles.loadBitmap(10,280,190,90,"duele.RAW");
```

```
  myFiles.loadBitmap(210,280,190,90,"musica.RAW");
  myFiles.loadBitmap(410,280,190,90,"tele.RAW");
  myFiles.loadBitmap(610,280,190,90,"dormir.RAW");
  myFiles.loadBitmap(10,380,190,90,"triste.RAW");
  myFiles.loadBitmap(210,380,190,90,"soledad.RAW");
  myFiles.loadBitmap(410,380,190,90,"frio.RAW");
  myFiles.loadBitmap(610,380,190,90,"calor.RAW");
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
  myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  myGLCD.setColor(VGA_BLACK);
   myGLCD.setBackColor(VGA_WHITE);
  myGLCD.print("TABLERO HOSPITALARIO",CENTER,10); // left izquierda
   // se lee la memoria sd para imprimir las imagenes en la pantalla


}
      if (pressed_button==but2)
        {
            datos();
myGLCD.clrScr();
myGLCD.fillScr(VGA_WHITE);
myGLCD.setColor(VGA_BLACK);
myGLCD.setBackColor(VGA_WHITE);
myGLCD.print("TABLERO HOSPITALARIO",CENTER,10); // left izquierda
   // se configuran los botones, posicion y lo que dira cada boton
myButtons.drawButtons();  // se imprimen en la pantalla los botones configurados
arriba
  myFiles.loadBitmap(10,80,190,90,"comer.RAW");
  myFiles.loadBitmap(210,80,190,90,"beber.RAW");
```

```
  myFiles.loadBitmap(410,80,190,90,"popo.RAW");
  myFiles.loadBitmap(610,80,190,90,"pipi.RAW");
  myFiles.loadBitmap(10,180,190,90,"banarse.RAW");
  myFiles.loadBitmap(210,180,190,90,"jugar.RAW");
  myFiles.loadBitmap(410,180,190,90,"cambio.RAW");
  myFiles.loadBitmap(610,180,190,90,"lavarse.RAW");
  myFiles.loadBitmap(10,280,190,90,"duele.RAW");
  myFiles.loadBitmap(210,280,190,90,"musica.RAW");
  myFiles.loadBitmap(410,280,190,90,"tele.RAW");
  myFiles.loadBitmap(610,280,190,90,"dormir.RAW");
  myFiles.loadBitmap(10,380,190,90,"triste.RAW");
  myFiles.loadBitmap(210,380,190,90,"soledad.RAW");
  myFiles.loadBitmap(410,380,190,90,"frio.RAW");
  myFiles.loadBitmap(610,380,190,90,"calor.RAW");
    myFiles.loadBitmap(585,10,200,50,"datos.RAW");
  myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  // se lee la memoria sd para imprimir las imagenes en la pantalla
}
      if (pressed_button==but3)
      {   myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
myGLCD.print("quiero comer",CENTER,50);
digitalWrite(10,HIGH);
delay(500);
digitalWrite(10,LOW);
delay(2500);
myGLCD.setColor(VGA_WHITE);
myGLCD.fillRect(220,30,580,80); // x,y,xy
```

```
//myButtons.drawButton(but1);
//myButtons.drawButton(but2);
myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  }
    if (pressed_button==but4)
{
myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
myGLCD.print("quiero tomar agua",CENTER,50);
for(int n=1;n<=3;n++)
      {
        digitalWrite(10,HIGH);
        delay(250);
        digitalWrite(10,LOW);
        delay(250);
      }
      delay(1500);
      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
 */
myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
      }
      if (pressed_button==but5)
```

```
      {
myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("quiero hacer popo",CENTER,50);
for(int n=1;n<=5;n++)
      {
        digitalWrite(10,HIGH);
        delay(100);
        digitalWrite(10,LOW);
        delay(100);
      }         delay(2000);
 myGLCD.setColor(VGA_WHITE);
 myGLCD.fillRect(220,30,580,80);// x,y,xy
 /*
 myButtons.drawButton(but1);
 myButtons.drawButton(but2);
  */
 myFiles.loadBitmap(585,10,200,50,"datos.RAW");
 myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
   }
       if (pressed_button==but6)
      {
           myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("quiero hacer pipi",CENTER,50);
for(int n=1;n<=7;n++)
      {
        digitalWrite(10,HIGH);
```

```
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }         delay(2300);
                  myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
 */
 myFiles.loadBitmap(585,10,200,50,"datos.RAW");
 myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
      }
        if (pressed_button==but7)
      {
                  myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("quiero bañarme",CENTER,50);
for(int n=1;n<=9;n++)
      {
        digitalWrite(10,HIGH);
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }      delay(2100);
                  myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
```

```
 myButtons.drawButton(but1);
 myButtons.drawButton(but2);
   */
myFiles.loadBitmap(585,10,200,50,"datos.RAW");

myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
   }
    if (pressed_button==but8)
   {
                myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
myGLCD.print("quiero jugar",CENTER,50);
 for(int n=1;n<=11;n++)
       {
         digitalWrite(10,HIGH);
         delay(50);
         digitalWrite(10,LOW);
         delay(50);
       }              delay(1900);
                    myGLCD.setColor(VGA_WHITE);
         myGLCD.fillRect(220,30,580,80);// x,y,xy
 /*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
   */
myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
}
```

```
      if (pressed_button==but9)
     {
                        myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere...",CENTER,30);
       myGLCD.print("quiero cambiarme",CENTER,50);
for(int n=1;n<=13;n++)
     {
       digitalWrite(10,HIGH);
       delay(50);
       digitalWrite(10,LOW);
       delay(50);
     }          delay(1700);
                        myGLCD.setColor(VGA_WHITE);
       myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
 */
myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
   }
    if (pressed_button==but10)
   {

myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
       myGLCD.print("quiero lavarme las manos",CENTER,50);
for(int n=1;n<=15;n++)
```

```
      {
        digitalWrite(10,HIGH);
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }       delay(1500);
                        myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
 */
 myFiles.loadBitmap(585,10,200,50,"datos.RAW");
 myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
 }
            if (pressed_button==but11)
      {
        myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
myGLCD.print("tengo dolor",CENTER,50);
        for(int n=1;n<=17;n++)
      {
        digitalWrite(10,HIGH);
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }             delay(1300);
       myGLCD.setColor(VGA_WHITE);
```

```
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
*/
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
      }
            if (pressed_button==but12)
      {
               myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("quiero escuchar musica",CENTER,50);
for(int n=1;n<=19;n++)

      {
        digitalWrite(10,HIGH);
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }
       delay(1100);
     myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
  */
```

```
myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  }
          if (pressed_button==but13)
      {
              myGLCD.setColor(VGA_BLACK);
 myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("quiero ver television",CENTER,50);
for(int n=1;n<=21;n++)
      {
        digitalWrite(10,HIGH);
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }
          delay(900);
      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
  */
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
  myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  }
           if (pressed_button==but14)
      {
              myGLCD.setColor(VGA_BLACK);
```

```
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("quiero dormir",CENTER,50);
for(int n=1;n<=23;n++)
      {
        digitalWrite(10,HIGH);

        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }
            delay(700);
      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
  */
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
  myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  }
             if (pressed_button==but15)
      {
                myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("estoy triste",CENTER,50);

for(int n=1;n<=25;n++)
      {
```

```
        digitalWrite(10,HIGH);
        delay(50);
        digitalWrite(10,LOW);
        delay(50);
      }
          delay(500);
      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
 */
 myFiles.loadBitmap(585,10,200,50,"datos.RAW");
 myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
}
            if (pressed_button==but16)
      {
              myGLCD.setColor(VGA_BLACK);
        myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("me siento solo",CENTER,50);
for(int n=1;n<=27;n++)
      {
        digitalWrite(10,HIGH);
        delay(40);

digitalWrite(10,LOW);
        delay(40);
```

```
      }
           delay(300);

      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
  */
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
  myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  }
            if (pressed_button==but17)
      {
              myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("tengo frio",CENTER,50);
for(int n=1;n<=29;n++)
      {
        digitalWrite(10,HIGH);
        delay(40);
        digitalWrite(10,LOW);
        delay(40);
      }
           delay(100);
      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
```

```
myButtons.drawButton(but1);
myButtons.drawButton(but2);
  */
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
  myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
  }
       if (pressed_button==but18)
      {
myGLCD.setColor(VGA_BLACK);
myGLCD.print("enviando mensaje espere",CENTER,30);
        myGLCD.print("tengo calor",CENTER,50);
for(int n=1;n<=31;n++)
      {
        digitalWrite(10,HIGH);
        delay(40);
        digitalWrite(10,LOW);

        delay(40);
      }
      myGLCD.setColor(VGA_WHITE);
        myGLCD.fillRect(220,30,580,80);// x,y,xy
/*
myButtons.drawButton(but1);
myButtons.drawButton(but2);
  */
  myFiles.loadBitmap(585,10,200,50,"datos.RAW");
myFiles.loadBitmap(15,10,200,50,"acerca.RAW");
}
```

```
    }// finaliza el touch avaliable
  }// finaliza el while
} // finaliza el void loop
oid acerca()
{  myGLCD.clrScr();
//  myGLCD.fillScr(VGA_WHITE);  //(255,255,255)
  myFiles.loadBitmap(0,0,800,480,"logo.RAW");    //    coordenadas    (x,y),
300pixeles x 300 pixeles, ittg.RAW
myGLCD.setColor(VGA_BLACK);
myGLCD.setBackColor(VGA_WHITE);
myGLCD.print("TABLERO HOSPITALARIO ELECTRONICO", CENTER,50);
myGLCD.print("***RESIDENTE***",CENTER,150); // left izquierda
myGLCD.print("ALMA GUADALUPE ORTEGA REYES",CENTER,250); // left
izquierda
myGLCD.print("***INGENIERIA   ELECTRONICA***",CENTER,350);   //   left
izquierda
 delay(10000);
}
void datos()
{
  myGLCD.clrScr();
  myGLCD.fillScr(VGA_WHITE);  //(255,255,255)
  myFiles.loadBitmap(10,10,300,300,"ittg.RAW");    //    coordenadas    (x,y),
300pixeles          x          300          pixeles,          ittg.RAW          //
myFiles.loadBitmap(10,10,300,300,"ittg.RAW");    //    coordenadas    (x,y),
300pixeles x 300 pixeles, ittg.RAW
myGLCD.setColor(VGA_BLACK);
    myGLCD.setBackColor(VGA_WHITE);
```

```
myGLCD.print("***INSTITUTO TECNOLOGICO***  ",RIGHT,50);       // right derecha
myGLCD.print("***DE TUXTLA GUTIERREZ***  ",RIGHT,150);
myGLCD.print("***RESIDENCIA PROFESIONAL***  ",RIGHT,250);
myGLCD.print("***ASESOR  INTERNO/EXTERNO***",CENTER,350);  // left izquierda
myGLCD.print("***ALVARO  HERNANDEZ  SOL***",CENTER,450);  // left izquierda
delay(10000);}
```

Programación del emisor-modulo nfr24l01

```
//----------------------configuración del emisor----------------
const int boton = 5;
int valor=0;
int n=0;
int contador=0;
int estadoanteriorboton=0;
int periodo=3000;
unsigned long t0;
unsigned long tiempo;
char conectar[32]="";//unsigned long got_time;
//-----------------configuracion del contador de pulsos-------------
--------------------------------
void setup() {
Serial.begin(9600);
pinMode(boton,INPUT);
digitalWrite(boton,HIGH);
```

```
//--------------configuracion e inicializacion del emisor ---------
pinMode(2, OUTPUT);
radio.begin();
radio.setRetries(15,15);          // Maximos reintentos
 //radio.setPayloadSize(8);   // Reduce el payload de 32 si tienes problemas
// Open pipes to other nodes for communication
 radio.openWritingPipe(pipes[0]);
 radio.openReadingPipe(1,pipes[1]);
//--------------configuracion e inicializacion del emisor ---------
}
void setup()
{
Serial.begin(9600);
pinMode(boton,INPUT);
digitalWrite(boton,HIGH);
//--------------configuracion e inicializacion del emisor ---------
pinMode(2, OUTPUT);
   radio.begin();
   radio.setRetries(15,15);       // Maximos reintentos
 //radio.setPayloadSize(8);   // Reduce el payload de 32 si tienes problemas
// Open pipes to other nodes for communication
    radio.openWritingPipe(pipes[0]);
    radio.openReadingPipe(1,pipes[1]);
//--------------configuracion e inicializacion del emisor ---------
}
```

Comenzando el void loop

```
void loop()
{valor=digitalRead(boton);
```

```
if(valor==1)
{
   t0=millis();
while(millis()<=periodo+ t0)
{
   valor=digitalRead(boton);
 if(valor!=estadoanteriorboton)
 {
 if(valor==1)
 {
 contador+ + ;
 }
 }
  estadoanteriorboton=valor;
}//termina while
}//termina if principal
//----------------------termina programa para contar pulsos------------
//---------------------------------------inicia programa para enviar datos----------
  if(contador==1)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero comer ";
  //  Serial.print("Enviando  ");
    //Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
    /*if (ok)
      //  Serial.println("ok...");
```

```
    else
      //  Serial.println("failed");*/
    radio.startListening();          //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
if ( timeout )
        {}
    else
      { // Leemos el mensaje recibido
char recibido [32]="";   //  unsigned long got_time;
        radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned
long) );
//Serial.print("Respuesta = ");
       // Serial.println(recibido);//(got_time);
      }
  }
//--------------------termina envio 1--------------------------
//---------------------inicia envio 2----------------------------
if(contador==3)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 tomar agua ";
//    Serial.print("Enviando  ");
 //   Serial.println (p);   //(time);
```

```
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
  //      Serial.println("ok...");
    else
     //   Serial.println("failed");*/
    radio.startListening();          //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
       {}//  Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
// Serial.print("Respuesta = ");
       // Serial.println(recibido);//(got_time);
}
  }
 //-------------------------termina envio 2----------------------------
 //----------------------inicia envio3---------------------------------
  if(contador==5)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
```

```
    const char p [] = " p1 quiero hacer popo ";
  //  Serial.print("Enviando  ");
  //  Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
    /*if (ok)
     //   Serial.println("ok...");
    else
    //    Serial.println("failed");
*/
    radio.startListening();         //Volvemos a la escucha

    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
{}       //  Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
// Serial.print("Respuesta = ");
       // Serial.println(recibido);//(got_time);
      }
  }// ----------------------termina envio 3-----------------------------
   //----------------------inicia envio4---------------------------------
```

```
  if(contador==7)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero hacer pipi ";

//    Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
  //      Serial.println("ok...");
    else
   //      Serial.println("failed");*/
    radio.startListening();          //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
    {}//      Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//  Serial.print("Respuesta = ");
      //  Serial.println(recibido);//(got_time);
```

```
      }
  }// ----------------------termina envio 4-----------------------------
   //----------------------inicia envio5---------------------------------
  if(contador==9)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero bañarme ";
//    Serial.print("Enviando  ");
  //  Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
    //    Serial.println("ok...");
    else
      //  Serial.println("failed");
*/
    radio.startListening();         //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
  {}        // Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
```

```
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
}
  }// ----------------------termina envio 5-----------------------------
   //----------------------inicia envio3--------------------------------
  if(contador==11)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero jugar ";
//Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
   /* if (ok)
  //      Serial.println("ok...");
    else
    //    Serial.println("failed");
*/
    radio.startListening();         //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;

    if ( timeout )
```

```
      {}//   Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);

//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
      }

  }// ----------------------termina envio 6-------------------
-----------
   //----------------------inicia envio 7----------------------
----------

  if(contador==13)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero cambiarme ";
    //Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
   /* if (ok)
  //      Serial.println("ok...");
    else
    //    Serial.println("failed");
```

```
*/
    radio.startListening();         //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
      {}//   Serial.println("Failed, response timed out");
Else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
      }
 }// ----------------------termina envio 7-----------------------------
  //----------------------inicia envio 8---------------------------------
  if(contador==15)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero lavarme ";
    //Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
```

```
  //      Serial.println("ok...");
    else
    //    Serial.println("failed");
*/
    radio.startListening();          //Volvemos a la escucha
unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
      {}//   Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
      }
  }// ----------------------termina envio 8-------------------------------
   //----------------------inicia envio 9--------------------------------
  if(contador==17)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 tengo dolor ";
//    Serial.print("Enviando  ");
```

```
  //  Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
    //    Serial.println("ok...");
    else
      //  Serial.println("failed");
*/
    radio.startListening();          //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
        {}// Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");


  //      Serial.println(recibido);//(got_time);
}
  }// ----------------------termina envio 9------------------------------
   //----------------------inicia envio 10--------------------------------
  if(contador==19)
  {
```

```
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero escuchar musica ";
    //Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
  //      Serial.println("ok...");
    else
    //    Serial.println("failed");
*/
    radio.startListening();         //Volvemos a la escucha

unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
      {}//   Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
      }
```

```
 }// ----------------------termina envio 10-------------------------------
  //----------------------inicia envio 11---------------------------------
 if(contador==21)
 {
 radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
   const char p [] = " p1 quiero ver tele ";
//    Serial.print("Enviando  ");
 //  Serial.println (p);   //(time);
   bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
   //    Serial.println("ok...");
   else
     //  Serial.println("failed");
*/
   radio.startListening();         //Volvemos a la escucha
   unsigned long started_waiting_at = millis();
   bool timeout = false;
   while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
       if (millis() - started_waiting_at > 200 )timeout = true;
   if ( timeout )
       {}// Serial.println("Failed, response timed out");
else
     { // Leemos el mensaje recibido
       char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
```

```
//        Serial.print("Respuesta = ");
  //       Serial.println(recibido);//(got_time);
}
 }// ----------------------termina envio 11-----------------------------
   //----------------------inicia envio 12--------------------------------
  if(contador==23)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 quiero dormir ";
    //Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
  //       Serial.println("ok...");
    else
    //    Serial.println("failed");
*/
    radio.startListening();          //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
      {}//   Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
```

```
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);


//Serial.print("Respuesta = ");
        //Serial.println(recibido);//(got_time);
      }
  }// ----------------------termina envio 12-------------------
------------
    //----------------------inicia envio13----------------------
----------
  if(contador==25)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 estoy triste ";
//    Serial.print("Enviando  ");
  //  Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
    //    Serial.println("ok...");
    else
      //  Serial.println("failed");
*/
    radio.startListening();         //Volvemos a la escucha
unsigned long started_waiting_at = millis();
    bool timeout = false;
```

```
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;


    if ( timeout )
        {}// Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
      }
  }// ----------------------termina envio 13-----------------------------
    //----------------------inicia envio14---------------------------------
  if(contador==27)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 estoy solo ";
    //Serial.print("Enviando  ");
//    Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
  //      Serial.println("ok...");
    else
    //    Serial.println("failed");
```

```
*/
    radio.startListening();          //Volvemos a la escucha
unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
      {}//   Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //       Serial.println(recibido);//(got_time);
}
 }// ----------------------termina envio 14-----------------------------
     //----------------------inicia envio15---------------------------------
  if(contador==29)
  {
  radio.stopListening();     // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 tengo frio ";
//    Serial.print("Enviando  ");
  //  Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
```

```
    //    Serial.println("ok...");
    else
      //  Serial.println("failed");
*/
    radio.startListening();          //Volvemos a la escucha
unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
        {}// Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
}
 }// ----------------------termina envio 15------------------------------
     //----------------------inicia envio16--------------------------------
  if(contador==31)
  {
  radio.stopListening();    // Paramos la escucha para poder hablar
unsigned long time = millis();
    const char p [] = " p1 tengo calor ";
//    Serial.print("Enviando  ");
```

```
  //  Serial.println (p);   //(time);
    bool ok = radio.write (&p,sizeof(p));  //( &time, sizeof(unsigned long) );
/*    if (ok)
    //    Serial.println("ok...");
     else
      //  Serial.println("failed");
*/
    radio.startListening();          //Volvemos a la escucha
    unsigned long started_waiting_at = millis();
    bool timeout = false;
    while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
        if (millis() - started_waiting_at > 200 )timeout = true;
    if ( timeout )
        {}// Serial.println("Failed, response timed out");
else
      { // Leemos el mensaje recibido
        char recibido [32]="";   //  unsigned long got_time;
radio.read (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned long)
);
//        Serial.print("Respuesta = ");
  //      Serial.println(recibido);//(got_time);
      }

  }// ----------------------termina envio 16------------------------------
//-------------------------termina envio de datos-------------------------
  if(contador!=n)
  {
```

```
//  Serial.println(contador);
  contador=n;
  }// termina if contador!=n
/*
else
{
   Serial.print("desconectar     ");
  Serial.println(conectar);
  delay(500);
}*/
}//---------- termina loop--------------------------
```

Programacion para el servidor

```
/*  ---------------------------------------------------------------
ARDUINO UNO

   Usando un NRF2401 para comunicar dos Arduinos en modo Duplex
   Programa Receptor:
---------------------------------------------------------------
*/
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "SoftwareSerial.h"
#define TxD 3
#define RxD 2
#define VCC 5
```

```
RF24 radio(9,10);  //ce,sck
const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };
SoftwareSerial bt1(TxD,RxD);
int cel=0;
int cierto=0;
void setup(void)
{
    bt1.begin(9600);
  pinMode(VCC,OUTPUT);
  digitalWrite(VCC,HIGH);
    pinMode(9, OUTPUT);
    Serial.begin(9600);
      radio.begin();
    radio.setRetries(15,15);
    //radio.setPayloadSize(8);
  radio.startListening();
    radio.openWritingPipe(pipes[1]);
    radio.openReadingPipe(1,pipes[0]);
    }
void loop(void)
{
    if ( radio.available() )  // Si hay datos disponibles
    {
        char recibido[32]="";//unsigned long got_time;
        bool done = false;
        while (!done)        // Espera aqui hasta recibir algo
          {
```

```
            done = radio.read (&recibido,sizeof(recibido));//( &got_time,
sizeof(unsigned long) );
            Serial.print("Dato Recibido =");
            Serial.println (recibido); //(got_time);
            delay(20);       // Para dar tiempo al emisor
          }
     unsigned long time=millis();
     bt1.println(recibido);
        unsigned long started_waiting_at = millis();
   bool timeout = false;
   while ( ! radio.available() && ! timeout )  // Esperasmos repsuesta hasta
200ms
       if (millis() - started_waiting_at > 200 )timeout = true;  // tiempo de
respuesta del bluetooth entre mas distancia mas tardara
       if ( timeout )
     {  Serial.println("Failed, response timed out");}
   else
     { // Leemos el mensaje recibido
         radio.stopListening();  // Dejamos d escuchar para poder hablar

     radio.write (&recibido,sizeof(recibido));   //( &got_time, sizeof(unsigned
long) );
     Serial.println("Enviando Respuesta");
     radio.startListening();       // Volvemos a la escucha para recibir mas
paquetes}
   }// termina if available radio
// termina comparacion de respuesta a conectar y recibido
}// termina void loop
```