ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE INGENIERÍA
DE SISTEMAS Y AUTOMÁTICA

Universidad
Carlos III de Madrid

PROGRAMACIÓN E INTEGRACIÓN DE INTERFAZ DE PANTALLA TÁCTIL ACCESIBLE PARA PRESSMATIC

TRABAJO DE FIN DE GRADOGRADO
INGENIERÍA EN TECNOLOGÍAS
INDUSTRIALES

Autor: Rodrigo Martín López

Tutor: Alberto Jardón Huete

Titulación: Grado en Tecnologías Industriales

Fecha: Septiembre 2014

Agradecimientos

A mi familia, por su apoyo.

A mis amigos, por estar presentes cuando necesitados.

A la Universidad, profesores, y compañeros, por estos cuatro años.

Resumen

El trabajo realizado es Programación e Integración de Interfaz de Pantalla Táctil Accesible para PRESSMATIC. El origen de este trabajo de fin de grado se encuentra en la tesis realizada por Gabriel Barroso que tenía como objetivo la creación de un dispositivo, llamado PRESSMATIC, diseñado para asistir a personas con diversidad funcional motora a realizar tareas que conlleven el movimiento de pinzado manual. Este proyecto tiene como objetivo proveer una solución a la necesidad de una interfaz táctil accesible integrada en dicho dispositivo.

Para contribuir a la creación de un dispositivo que promueve la autonomía de personas que ven en riesgo su capacidad de realizar tareas que conlleven este movimiento, se busca ofrecer un producto que pueda ser empleado con facilidad y altamente intuitivo. Por ello la generación de una interfaz táctil fue la elección para el control del dispositivo.

El Proyecto se realiza en coordinación con otros dos trabajos realizados por Alejandro Vega y Alonso Rosado para ofrecer un mayor abanico de interfaces compatibles para el control de PRESSMATIC para mejorar la experiencia del usuario y aumentar la accesibilidad al producto. Estos son: Programación en Android Accesible para el Control de PRESSMATIC y Protocolo de Comunicación Bluetooth Y Control Por Voz para PRESSMATIC.

Al inicio del trabajo se realiza un estudio del estado del arte de módulos de pantalla integrables compatibles con Arduino. El módulo empleado para la programación de la interfaz es el uLCD-28PTU-AR de 4D Systems. Los módulos de 4D Systems incluyen un software de programación, Workshop4, un IDE que incluye un entorno de diseño gráfico con el que se desarrolla la interfaz, ViSi-Genie.

La programación del módulo se realiza mediante diseño gráfico y la personalización de objetos disponibles en el entorno. En el diseño se siguen las pautas de accesibilidad descritas por distintos organismos, adaptándolas a nuestros casos de uso, para la creación de una aplicación accesible a usuarios con distintas diversidades funcionales. Para la integración de la interfaz con el control del dispositivo el módulo y el Arduino emplean comunicación serial y se adapta el código para interactuar con la información recibida a través de mensajes de eventos y, a su vez, enviar información de vuelta al módulo.

Por último tanto el diseño como la implementación son similares a los realizados en las interfaces correspondientes a dispositivos Android y Easy VR, para obtener como resultado un conjunto de interfaces compatibles. Así, la manipulación de cualquiera de éstas repercute en el estado de la pantalla táctil.

Como resultado, el trabajo realizado ofrece una solución integrable y compatible con otros dispositivos de control a la interfaz implementada en una pantalla táctil para PRESSMATIC mediante la programación del módulo de 4D Systems empleando un IDE y del Arduino.

Abstract

The proyect is Programación e Integración de Interfaz de Pantalla Táctil Accesible para PRESSMATIC. This end of degree proyect has its origin in the thesis by Gabriel Barroso which aimed to create a device called PRESSMATIC, designed to assist people with motor functional diversity perform tasks that involve the movement of manual clamping. This project aims to provide a solution to the need for an integrated accessible touch interface in the device.

To contribute to the creation of a device that offers aid to people who may lack the ability to perform tasks that may require clamping movements, the offer of a product that can be used easily and highly intuitive is seeked. Therefore, the generation of a touch interface was the choice to control the device.

The project is coordinated with other two by Alejandro Vega and Alonso Rosado to offer a wider range of compatible interfaces for controlling PRESSMATIC, to improve user experience and increase accessibility to the product. These are: Programación en Android Accesible para el Control de PRESSMATIC and Protocolo de Comunicación Bluetooth Y Control Por Voz para PRESSMATIC.

At the beggining, a study of the state of the art of integrated display modules compatible with Arduino is performed. The module chosen for the programming of the interface is the uLCD 28PTU-AR from 4D Systems. 4D Systems modules include developing software, Workshop4, an IDE that includes a graphical design environment with which the interface will be developed, ViSi-Genie.

The module programming is performed using graphic design and customization of objects available in the environment. When designing, accessibility guidelines described by various agenciesare followed, adapting them to our use case, to create an accessible application for users with different functional diversities. For the integration of the interface with the device, the display module and Arduino use serial communication and the code is adapted to interact with the information received through event messages and, in reverse, send information back to the module.

Finally, both design and implementation are similar to those made in the interfaces corresponding to Android devices and Easy VR, to result in a set of compatible interfaces. Thus, the manipulation of any of these affects the state of the touch screen.

As a result, the realized proyect offers as a solution an integrable interface, compatible with other control devices, implemented on a touch screen for PRESSMATIC, achieved by programming the module using a 4D Systems IDE and Arduino.

Tabla de Contenido

Índice de Ilustraciones

Índice de Tablas

1 Motivación

1.1 Introducción

La razón de ser de este proyecto se encuentra en la tesis realizada previamente por Gabriel Barroso de María. En su tesis, "Dispositivo automático de apoyo para uso de herramientas requeridas de movimiento manual de pinzado" [1], Gabriel estudia la viabilidad de la creación de un dispositivo que asista a personas con ciertas limitaciones motoras a realizar tareas manuales que conllevan un movimiento de pinzado.

Las personas a las que va dirigido el dispositivo que Gabriel ideó son, de entre las diversidades funcionales, aquellas cuya habilidad manual se haya visto disminuida. Para facilitar la realización de ciertas actividades domésticas, de aseo, o aficiones a dichas personas, se propone la creación de dicho dispositivo, PRESSMATIC, un aparato electromecánico que sirva de asistencia.

Entre las funcionalidades que se pretenden abarcar y cubrir son las de unas tijeras, cortaúñas, pinzas de agarre, etc. Sustituyendo a un número de herramientas y acciones empleadas diariamente que precisan de cierta gracialidad manual. El dispositivo tiene como objetivo aportar eficacia, seguridad y comodidad al usuario a la hora realizar estas tareas.

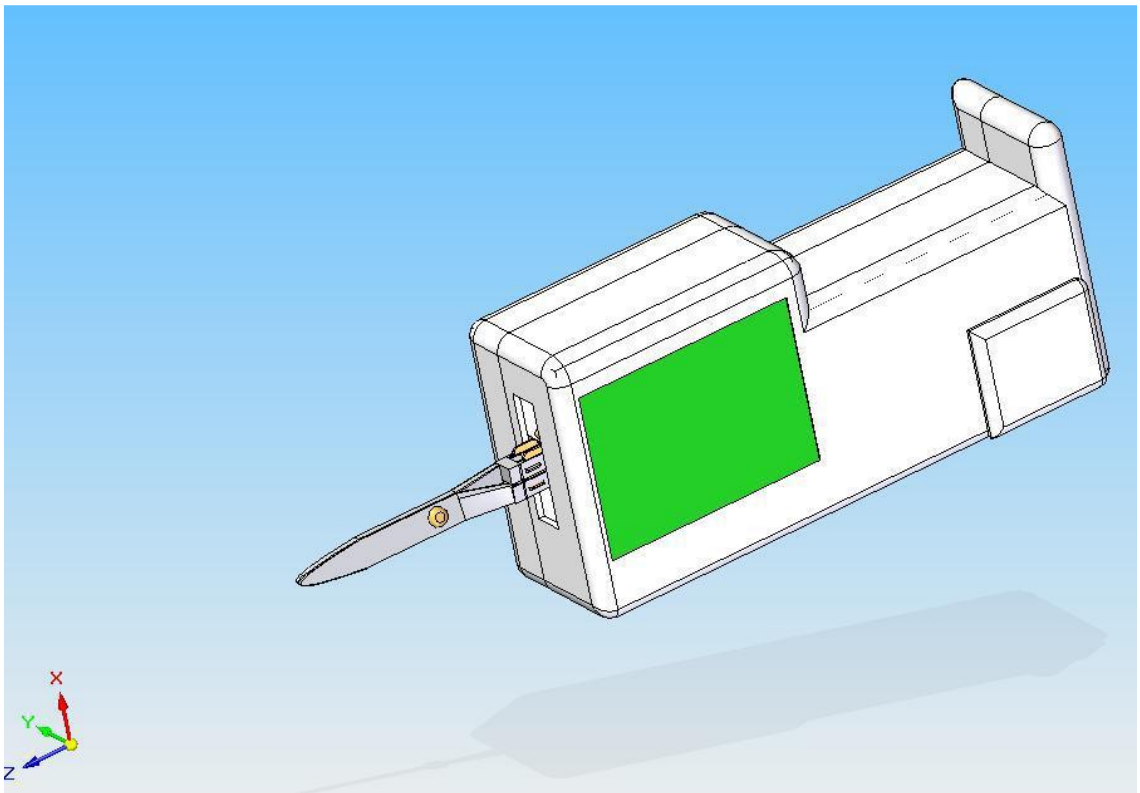

Ilustración 1.1 Diseño del prototipo de PRESSMATIC [1].

El diseño, mostrado en la figura anterior, que se propone en la tesis tiene ciertas características a partir de las que se intenta implementar un prototipo funcional. PRESSMATIC se compone de un cuerpo principal y cabezales intercambiables, que

son las herramientas. En el cuerpo del aparato se alojan el sistema eléctrico, la electrónica de control, una pantalla táctil, un motor, un sistema mecánico de transmisión y los anclajes de los cabezales, mientras que los cabezales intercambiables se componen de clavijas de anclaje y una herramienta. A partir de la tesis surgen diversos proyectos de aspecto mecánico, eléctrico, y de control.

1.2 Introducción al control de PRESSMATIC

Atendiendo al aspecto del control de PRESSMATIC, surgen tres proyectos con el fin de diseñar e implementar interfaces que sean accesibles. En su tesis, Gabriel, propone que para el control de dicho dispositivo se emplee como interfaz una pantalla táctil. El principio por el que se interactúa con ellas facilita el manejo a las personas a las que va destinado. Al ser, además de táctil, una interfaz visual la información puede ser presentada al usuario de manera intuitiva. Entre las ventajas que ofrece una pantalla táctil como interfaz encontramos la accesibilidad, la integración con un microcontrolador, el diseño de una interfaz que aumente las posibilidades de control mediante la navegación entre pantallas, y flexibilidad en el planteamiento del control, permitiendo actualizarlo.

Como variante, los smartphones, se han propuesto como interfaz alternativa en consonancia con la pantalla táctil integrada en el dispositivo. Por lo tanto el desarrollo de una aplicación para dispositivos Android surgió como idea para la creación de una interfaz de control a distancia.

Por último, Gabriel añade que una interfaz de control por voz "facilitaría el manejo de PRESSMATIC a las personas a las que está dirigido" [1]. Así, se contempla la viabilidad de la integración de un dispositivo con micrófono a través de Bluetooth.

1.3 Objetivos

El objetivo de este Trabajo Fin de Grado, Programación e Integración de Interfaz de Pantalla Táctil Accesible para PRESSMATIC, es Diseñar e implementar la interfaz de PRESSMATIC en una pantalla táctil. La interfaz debe:

- Para controlar el dispositivo, se ha optado por utilizar una placa de desarrollo arduino que será detallada más adelante. Por tanto, uno de los requisitos para el TFG es utilizar una pantalla compatible con el entorno arduino y que se comunique por puerto serie. Para que la aplicación sea eficaz y cumpla sus objetivos, se hace totalmente imprescindible que la **conexión** entre PRESSMATIC y la interfaz sea **robusta** y **estable**.
- Mostrar accesibilidad en la interfaz diseñada. Se tendrá en cuenta principalmente la población a la que va dirigida y las necesidades que puedan tener. En este caso se tendrá que diseñar una aplicación que sea **accesible** y cuya utilización sea **simple e intuitiva**.
- Mostrar compatibilidad y una similaridad consistente con la interfaz diseñada para dispositivos Android y el control por voz.

Este proyecto se realiza en paralelo con otros dos que facilitarán medios de control alternativos. El proyecto de Alejandro Vega Gómez es Programación en Android accesible para el control de PRESSMATIC y tiene como objetivos la generación de una aplicación accesible para dispositivos Android, compatible con la interfaz de la pantalla táctil, y que mantenga una conexión estable y robusta [2]. Por su parte, Alonso Rosado García realiza el trabajo Protocolo de comunicación Bluetooth y control por voz para PRESSMATIC que tiene como objetivos crear un protocolo de control para la comunicación entre dispositivos Android y el dispositivo, e implementar un módulo de control por voz compatible con la interfaz de la pantalla táctil [3].

1.3.1 Control a través de Arduino

Para implementar el control del sistema, se ha utilizado la placa de desarrollo Arduino Mega ADK, que está basada en el micro-controlador AtMega 2560. Dada su naturaleza open-source, una gran comunidad ha impulsado a Arduino a la hegemonía en cuanto a microcontroladores de propósito general para la creación y desarrollo de proyectos de éste calibre.

Para controlar el aparato las interfaces se comunicarán con el Arduino, envían y reciben información ya sea por serial o a través de Bluetooth. Las interfaces aportan el valor de las variables programadas en el microcontrolador para que se realice la operación deseada. El Arduino controla el motor con ayuda de una librería de modulación de ancho de pulso (PWM).

1.3.2 Accesibilidad

La accesibilidad se define como el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio. Así, con el objeto de que el máximo de usuarios potenciales de todas las diversidades funcionales puedan

controlar el dispositivo, se adaptarán, en lo posible, las interfaces a las pautas de accesibilidad. Incluyendo diseño y herramientas alternativas de control, como el control por voz.

1.3.3 Compatibilidad

La interfaz en la pantalla táctil se desarrolla en paralelo con una aplicación para móviles Android y un dispositivo de control por voz. Se diseñan las interfaces de manera que puedan interactuar, ofreciendo así multitud de posibilidades para el control de PRESSMATIC, aumentando la funcionalidad del dispositivo.

2 Elección de la pantalla táctil

La realización de este trabajo requiere de la selección de un módulo sobre el que implementar la interfaz. Este es un módulo de entrada táctil y salida de imagen.

Por su parte, Gabriel Barroso concluyó que una pantalla táctil adecuada sería el modelo iTouch. Las características de este módulo incluyen un tamaño de 2,8 pulgadas, resolución QVGA, pantalla LCD-TFT con iluminación LED de fondo, una interfaz de 8 pines, puertos UART, programabilidad [1].

Al comienzo de este trabajo se lleva a cabo un estudio del estado del arte sobre pantallas táctiles programables y se reevalua la elección tomada por Gabriel Barroso en su tesis.

2.1 Estado del arte

Las pantallas táctiles son periféricos de entrada y salida de datos. Constan de un monitor para la visualización de información y de una o varias capas de material que permita al usuario interactuar a través de la entrada de datos mediante el contacto digital.

2.1.1 Historia

La tecnología que permite la interacción táctil con interfaces electrónicas tiene sus inicios en 1965, cuando E.A. Johnson describe una pantalla táctil capacitiva en un artículo [4]. En 1973 unos investigadores de la Universidad de Illinois adquieren la patente sobre la tecnología basada en una matriz de rayos infrarrojos para la interacción con pantallas [5]. Dos años más tarde G. Samuel Hurst recibe la patente sobre su pantalla táctil resistiva [6]. De estas tecnologías, la capacitiva y la resistiva serán las que impulsen la popularidad de las pantallas táctiles cuando se implementan en puntos de información públicos, cajeros automáticos, en monitores personales, ordenadores, PDAs, y por último en teléfonos móviles y tabletas.

Las pantallas táctiles se dividen según la tecnología que permita la interacción con la interfaz, principalmente: resistivas y capacitivas.

La tecnología táctil se puede implantar sobre cualquier tipo de pantalla, por lo tanto se encuentran, a su vez, distintos tipos de tecnologías. Para el desarrollo de la interfaz se empleara una pantalla plana de tamaño reducido.

Las tecnologías que dominan el campo de los monitores de vídeo digitales son las pantallas de plasma, las pantallas LED y las pantallas de cristal líquido. La idea de una pantalla plana fue patentada por primera vez en 1939 y se basaba en descargas eléctricas puntuales sobre electrodos para generar píxeles [7]. Así, la primera pantalla plana inventada en ver la luz fue una pantalla de plasma monocromática, inventada en la Universidad de Illinois en 1964 [8]. Fue en 1977 cuando se desarrolla la primera plana que emplea únicamente diodos LED para la emisión de imágenes [9]. Desde el descubrimiento de los cristales líquidos en 1888 hasta la producción de la primera pantalla LCD moderna en 1972 se investigaron, a partir de 1962 [10], las propiedades luminosas de este material [11].

Hoy en día las pantallas de plasma han sido remitidas a un segundo plano, siendo algo más comunes en monitores gigantes, mientras que las tecnologías LED y LCD lideran el mercado.

2.1.2 Tecnologías de pantallas planas

Para determinar las ventajas y desventajas de cada tipo de pantalla se estudian brevemente las tecnologías empleadas:

Las pantallas LCD emplean las propiedades de los cristales líquidos para modular la luz incidente en éstos, que no emiten luz propia y por ello emplean un foco de luz trasero, normalmente un LED. Para los distintos requerimientos de visualización que requieran las aplicaciones de un monitor surgen dos estructuras: matriz pasiva y matriz activa.

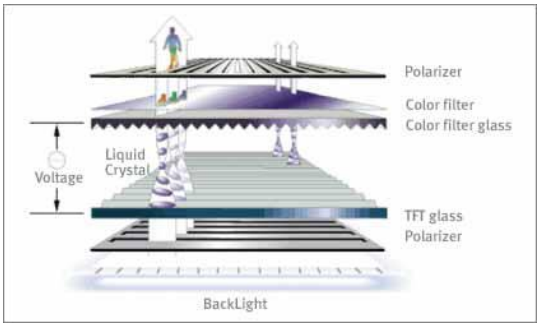

Ilustración 2.1 Pantalla LCD de matriz activa [12].

Las pantallas LCD con una matriz activa emplean una matriz de transistores de película fina o TFT que aporta un transistor a cada píxel. Esto resulta en una imagen de calidad superior. Las que emplean una matriz pasiva, como las que emplean la tecnología HPA, se aplican a dispositivos que necesiten menor calidad específica. Las matrices pasivas se componen pueden ser compuestas por STN o DSTN, para un resultado monocromo, o CSTN, para un resultado en color [13].

Las pantallas que emplean diodos emisores de luz se basan en la creación de una matriz de píxeles, donde cada píxel está compuesto por tres diodos, rojo, verde, y azul, para generar color. La limitación en cuanto a tamaño de estas pantallas depende del tamaño y la cantidad de estos píxeles. Recientemente los diodos orgánicos emisores de luz, una fina pantalla compuesta por material orgánico es la encargada de emitir luz. Estas pantallas también pueden ser de matriz pasiva o matriz activa, también conocidas como AMOLED. Las pantallas AMOLED igualmente emplean una matriz de TFT para encender o apagar cada píxel individualmente [14].

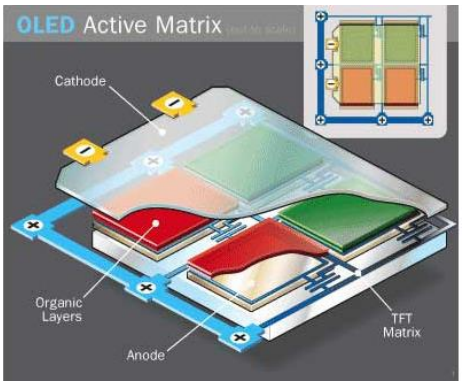

Ilustración 2.2 Estructura pantalla OLED de matriz activa [14].

Ilustración 2.3 Diodos LED [15].

Las pantallas de plasma emplean las propiedades de gases nobles y distintos tipos de fósforos para la emisión de luz de distintas longitudes de onda. La emisión de luz se da al cuando la celda que contiene los gases recibe una alta descarga eléctrica, generando el plasma.

2.1.3 Tecnologías táctiles

Las tecnologías que permiten la entrada de datos táctilmente que se estudian para su uso en la interfaz de PRESSMATIC son la resistiva y la capacitiva.

Las resistivas se basan en la superposición de dos láminas conductoras separadas que al entar en contacto por presión detectan la posición donde se ha producido por un cambio en la corriente eléctrica.

Las capacitivas emplean distintos métodos de medición de cambios en capacitancia para detectar la posición en la que el dedo o algún otro elemento conductor entra en contacto con la superficie conductora que se haya superpuesto sobre la pantalla.

Las pantallas resistivas son más lentas, reducen el brillo de la pantalla, pero son más precisas. Las pantallas capacitivas son más intuitivas y rápidas pero presentan una pequeña disminución en precisión. Además hay que distinguir que las pantallas que emplean tecnología resistiva son una solución mucho más económicas que su alternativa.

2.2 Pantalla elegida

A la hora de realizar la elección se evalúa el tipo de pantalla, la tecnología táctil que integra, el tamaño del módulo, la compatibilidad con Arduino, y su coste, entre otras cualidades.

La pantalla táctil que se escoge para crear la interfaz es la uLCD-28PTU-AR, un producto de 4D Systems. 4D Systems es una empresa que se dedica a la investigación, el desarrollo y la producción de módulos gráficos inteligentes. La pantalla tiene la opción de venir en un pack, SK-28PTU-AR, que incluye cables adaptadores para la programación, y una tarjeta microSD de 2GB. El módulo y el pack incluyen un Arduino Adaptor Shield y un cable de 5 bits [16].

Ilustración 2.4 4D Systems pack uLCD-28PTU-AR [16].

2.2.1 Hardware

Las características del Hardware son similares a las encontradas en los modelos de otros fabricantes comparados. La pantalla, de 2.8 pulgadas, emplea la tecnología LCD-TFT de matriz activa y resistiva de tacto, la resolución es QVGA, de 320x240 píxeles. El procesador que emplea el módulo es el PICASO, hecho a medida por 4D-Labs y la interfaz es serial y de cinco pines [17].

Las especificaciones son detalladas por 4D Systems como sigue [18]:

- 14KB of flash memory for user code storage and 14KB of SRAM for user variables, or 14KB shared user code and program variables.
- 2 x Asynchronous hardware serial ports (COM0, COM1), TTL interface, with 300 to 600K baud.
- 1 x I2C interface (Master).
- Easy 5 pin interface to any host device:

  VCC, TX, RX, GND, RESET

- 8 x 16 bit timers with 1 millisecond resolution.

- 13 x General Purpose I/O pins. Supports fast 8-bit parallel data transfer through Upper 8 bits.
- On-board micro-SD memory card socket for multimedia and data logging with SDHC memory card support. A SPI compatible micro-SD card is required.
- DOS compatible file access (FAT16 format) as well as low level access to card memory.
- Lithium Polymer battery support, with built in battery charger and automatic change-over.
- Dedicated PWM Audio pin driven by WAV files from micro-SD card.
- On-board audio amplifier with a tiny 8W speaker for sound generation and WAV file playback.
- Built in extensive 4DGL graphics and system library functions.
- Display full colour images, animations, icons and video clips.
- Supports all available Windows fonts.
- A 30 pin header for I/O expansion and future plug-in daughter boards.
- 4.0V to 5.5V range operation (single supply).
- Module dimensions: 52.0 x 83.69 x 15.9mm (including corner plates).
- Weighing ~ 43g.
- Display Viewing Area: 43.20 x 57.60mm
- 4 x corner plates with 2.7mm holes for mechanical mounting.
- RoHS Compliant.
- CE Compliant (on modules with CE Mark).

2.2.2 Software

El software que provee 4D Systems para la programación de sus módulos es un programa gratuito. El Workshop4 es un entorno de desarrollo integrado o IDE compatible con Microsoft Windows que incluye editor, compilador, enlazador y descargador para el desarrollo de aplicaciones.

Workshop4 incluye cuatro entornos de desarrollo: Designer, ViSi, ViSi-Genie, y Serial. Que ofrecen distintas posibilidades a la hora de generar una aplicación a usuarios con distintos niveles de conocimiento y para distintas aplicaciones de uso.

Además para complementar su compatibilidad con Arduino, 4D Systems pone a disposición distintas librerías para Arduino para distintas aplicaciones, como serial, y ViSi-Genie.

Por último, la disposición de notas sobre las aplicaciones, con información en forma de guías, ejemplos, y tutoriales, es una gran ventaja.

2.2.3 Factores de elección

La pantalla de 4D Systems, modelo uLCD-28PTU-AR, presenta las cualidades deseadas, está diseñada para ser integrada con un Arduino, se complementa de un entorno de diseño muy útil para la creación de una interfaz prototipo, y tiene el tamaño adecuado para poder ser integrada en el aparato.

Las características del software hacen ideal el módulo gracias a su fácil programación y la información aportada en notas, ejemplos y tutoriales.

El hardware cumple con las expectativas y ofrecerá una solución más que capaz para soportar la interfaz. Por último, el precio del módulo se encuentra dentro del rango de lo posible.

3 Accesibilidad

3.1 Introducción a la accesibilidad

La accesibilidad es una cualidad propia de un objeto, lugar o servicio que determina el grado en el que la población de todas las diversidades funcionales puede hacer uso de ellos. El objetivo de crear objetos, lugares o servicios accesibles es, por lo tanto, permitir al mayor número de personas posible su uso.

Del deseo de proporcionar accesibilidad a las tecnologías, más específicamente a los contenidos en Internet, surge la accesibilidad web. La accesibilidad web se refiere a un diseño que permita a personas de todas las diversidades funcionales puedan "puedan percibir, entender, navegar e interactuar con la Web, aportando a su vez contenidos" [17]. Existen unas pautas emitidas por el máximo organismo que promueve la accesibilidad web, World Wide Web Consortium (W3C). Estas pautas se dividen en tres bloques: Pautas de Accesibilidad al Contenido en la Web (WCAG), Pautas de Accesibilidad para Herramientas de Autor (ATAG), y Pautas de Accesibilidad para Agentes de Usuario (UAAG).

Al igual surge una normativa para asegurar la accesibilidad software, que se recoge en la Norma ISO 9241-171:2008 de Requisitos de accesibilidad del software [18].

Se quiere desarrollar una interfaz accesible para el control de PRESSMATIC empleando una pantalla táctil. El término empleado para interfaces de este tipo en dispositivos táctiles actuales es aplicación o "app". En nuestro entorno la forma en la que se encuentran aplicaciones más comúnmente es aplicada a dispositivos móviles. Aun así una estimación realizada a mediados del 2013 del Grupo Fundosa apunta que las pautas de accesibilidad no son cumplidas en su totalidad en un 99% de los casos [19].

3.2 Aplicaciones accesibles

Entre la escasa selección de aplicaciones accesibles para dispositivos móviles encontramos varias que asisten al usuario en tareas cotidianas o aportan apoyo según sus distintas diversidades funcionales. Algunas de estas aplicaciones accesibles son:

- **AudescMobile** es una aplicación para la asistencia visual que describe mediante audio las imágenes que aparezcan en un vídeo, en la televisión o el cine, la descripción se reproduce en tiempo real, a medida que se sucedan las imágenes, mediante sincronización sonora. Es una aplicación gratuita fomentada por la ONCE [20].
- **Voice Dream Reader** es un lector de archivos de texto que incluye una herramienta de conversión de texto a voz. Facilita el acceso a documentos a personas de distintas diversidades funcionales visuales. Es una aplicación soportada en iOS y de pago [20].
- **TUR4all** proporciona información sobre el estado y calidad de la accesibilidad de distintos tipos de establecimientos localizados en España. Los establecimientos fueron evaluados por técnicos del PREDIF y siguiendo un protocolo [21].
- **SIGNAME** facilita videos donde un profesional reproduce los gestos empleados en centros educativos como el Centro María Corredentora para comunicarse con los niños. Los gestos están basados en la Lengua de Signos Española [21].

3.3 Pautas de Accesibilidad

Las pautas y normas dispuestas por diferentes organizaciones para el desarrollo de software y contenido web accesible se extrapolan al diseño de una aplicación accesible para la pantalla táctil. Esto resulta en una interfaz que puede ser comprendida y empleada por el máximo número de personas.

3.3.1 Accesibilidad visual

Según la normativa impuesta por W3C debe asegurarse que el texto y los gráficos puedan ser entendidos vistos sin color.

Las personas que no puedan diferenciar entre ciertos colores podrían no recibir la información contenida. Una solución para evitar la pérdida de información es emplear una combinación de colores que provea un contraste lo suficientemente alto.

Según la Unidad de Investigación ACCESO [22], las fuentes de los textos se dividen en dos familias principales: 'serif' y 'sans-serif'. Las fuentes que pertenecen a la familia 'serif' se caracterizan por ostentar remates embellecidos, a lo que se refiere en este caso como 'serifa', estas fuentes facilitan la lectura de textos de extensiones considerables. Un ejemplo es la fuente "Times New Roman". Por otro lado las fuentes 'sans-serif' son de una caligrafía más sencilla. Esto da a la familia 'sans-serif' la cualidad de ser más agradables cuando visualizadas en una pantalla incluso en tamaños reducidos. Un ejemplo es la fuente "Verdana".

3.3.2 Navegación accesible

La normativa WCAG describe la navegación accesible del siguiente modo: la navegación accesible se basa en proporcionar un mecanismo de navegación, disposición, y diseño consistentes para aumentar la probabilidad de que el usuario encuentre lo que busque. Esto se puede conseguir proporcionando información acerca del entorno y de la orientación de la interfaz, describiendo las funcionalidades de los enlaces claramente, usando mecanismos de navegación y control constantes, empleando un lenguaje sencillo y adecuado al entorno, y facilitando mensajes de texto que notifiquen al usuario de eventos [17].

Este apartado está enfocado para facilitar los aspectos cognitivos de un entorno.

3.3.3 Control accesible

El control accesible tiene como objetivo aumentar la facilidad con la que el usuario puede llevar acabo cualquier acción en una interfaz. Para ello las pautas definen una serie de requisitos. La optimización para que el usuario acceda a cualquier parte de ella en el menor número de pasos es clave a la hora de diseñar una interfaz. Las notificaciones deben ser consistentes y deben poder permanecer activas mientras su causa también lo esté. Los botones deben ser lo suficientemente grandes y estar lo suficientemente separados entre sí para reducir la posibilidad de que se pulse un botón erróneamente [23]. Esto influye en el

número de botones que se podrá disponer en una pantalla. Se debe incluir una funcionalidad que permita retroceder entre pantallas por la posibilidad de una selección anterior indeseada y, simplemente, facilitar la navegación. Las acciones que se pueda realizar en la interfaz deben ser fácilmente descubiertas de manera clara.

Proporcionar alternativas de entrada al control, como puede ser un servicio de reconocimiento de voz, aumenta la accesibilidad del producto. Esto permite al usuario un gran abanico de posibilidades, al ofrecer un control manos libres.

4 El Control de PRESSMATIC

4.1 Implementación en Arduino

El control de PRESSMATIC se realiza de tal modo que el dispositivo sustituya el movimiento de pinzado que realizan los dedos pulgar e índice y, equipado con una herramienta, pueda llevar a cabo acciones tales como cortar y agarrar objetos.

El control del motor incorporado para la propulsión de los cabezales se implementa en un Arduino. Edwin Daniel Oña diseña el control de manera que permita un control versátil de las herramientas, basándose en el diseño original de Gabriel Barroso que sigue un funcional como el mostrado en la figura extraída de su tesis.

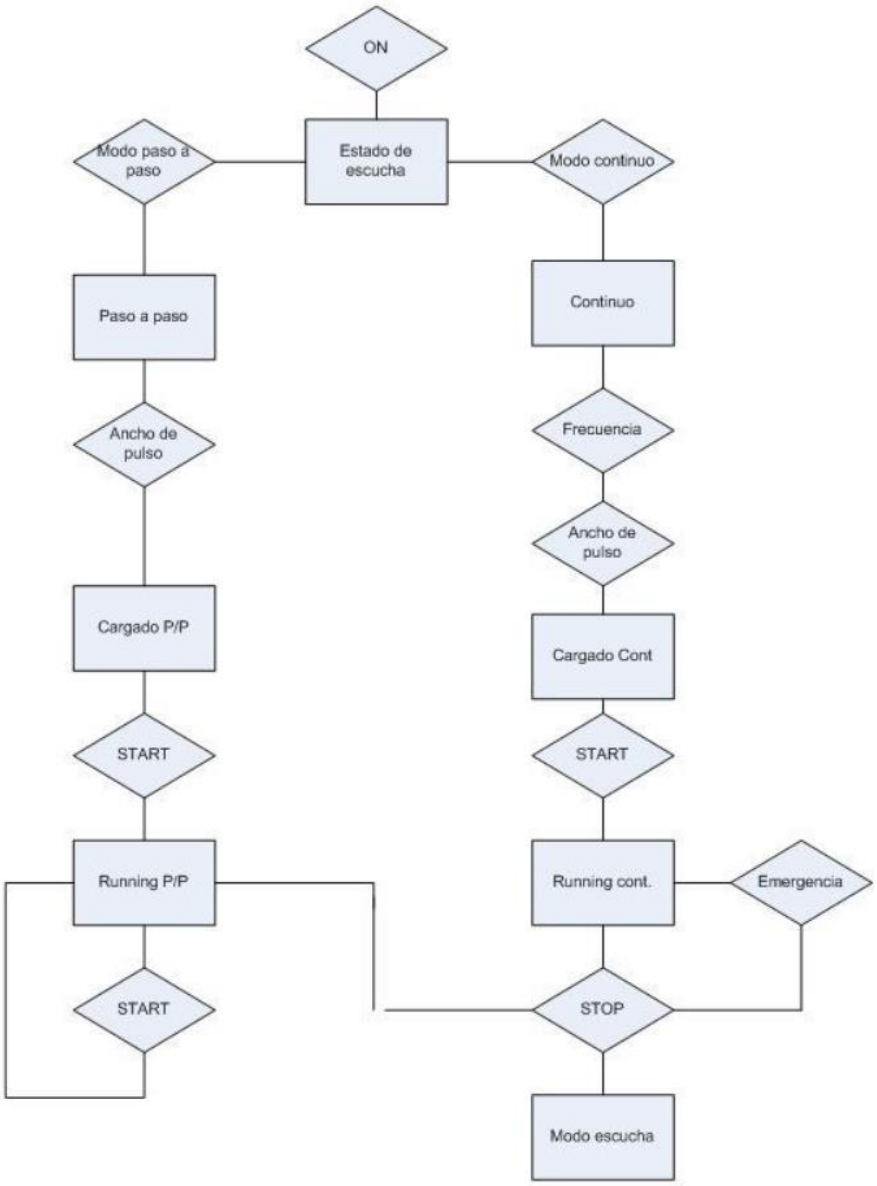

Ilustración 4.1 Diagrama de estados del control de PRESSMATIC [1].

4.2 Modos de funcionamiento de PRESSMATIC

Los modos de funcionamiento ideados para el control de PRESSMATIC se basan en las distintas acciones tales como cortar y agarrar. Los modos ideados permiten un control basado en la regulación de la velocidad, los incrementos y los ciclos de apertura y cierre, según aquél que se seleccione, que realiza el dispositivo. Para el control de PRESSMATIC se definen tres modos de funcionamiento y un estado de escucha.

Mediante el modo continuo el dispositivo abrirá y cerrará los cabezales de manera constante a la velocidad que se seleccione entre los momentos que se reciba la orden de inicio y la de parada.

El modo grip o paso a paso permite que los actuadores del dispositivo se abran o cierren por incrementos al recibir la orden correspondiente. Este modo permite al usuario emplear PRESSMATIC como unas pinzas con las que puede agarrar y maniobrar objetos de distintas envergaduras.

El modo de funcionamiento de ciclos permite controlar el número de veces que se realiza el ciclo de apertura y cierre antes de que se detenga automáticamente, o se pare manualmente. Como una variación del modo ciclos que sería útil para herramientas como el cortaúñas, cuando solo se desea realizar un ciclo de corte, se define el modo cortar una en el que el número de ciclos a realizar es constante e igual a uno.

Además de los modos de funcionamiento empleados para accionar el dispositivo se han dispuesto un modo de escucha o espera y un modo de activación y desactivación del Bluetooth.

En el estado de escucha es en el que el dispositivo está inactivo a la espera de que el usuario introduzca a través de una de las interfaces para acceder a cualquier otro modo.

Tras una nueva valoración, se opta por cambiar el enfoque de la interacción usuario final - PRESSMATIC. Es decir, para conseguir una interacción más intuitiva, los modos de operación se orientan hacia herramientas. De esta manera, el nuevo diagrama de funcionamiento sería el siguiente:

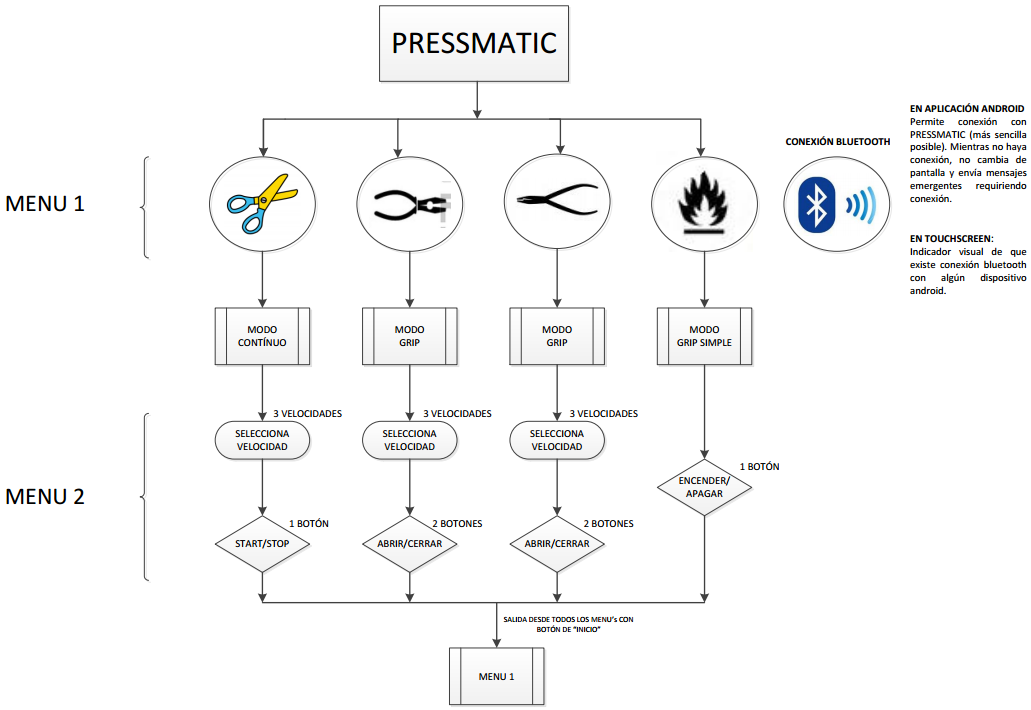

Ilustración 4.2 Diagrama de operación intuitivo.

La interfaz gráfica desarrollada en el presente trabajo, permitirá el usuario final desplazarse entre los diferentes modos de operación de una forma sencilla e intuitiva.

5 Creación de la aplicación

5.1 Entornos de programación Workshop4

El hardware de los productos de 4D Systems se complementa con un software específico con el que configurar y programar los módulos de manera independiente pero compatible con el entorno de Arduino. El software Workshop4 es un entorno de desarrollo integrado (o IDE) que consta de diversos entornos con los que trabajar [24]. Estos incluyen editor, compilador, enlazador y descargador para el desarrollo del código 4DGL para la aplicación. El código 4DGL es específico al software y el hardware (EVE graphics core) de los módulos de 4D Systems.

Los entornos disponibles en Workshop4 son Designer, ViSi, ViSi-Genie, y Serial:

- El entorno Designer permite programar la pantalla táctil escribiendo código 4DGL en su forma nativa.
- ViSi hace de la pantalla un módulo esclavo por serial y el entorno permite emplear el diseño gráfico de la pantalla para generar el código correspondiente a los objetos empleados, asistiendo el desarrollo de código 4DGL.
- ViSi-Genie, al igual, hace de la pantalla un módulo esclavo por serial y el entorno genera automáticamente todo el código 4DGL correspondiente al diseño visual de los objetos y sus propiedades y eventos.
- El entorno Serial convierte al módulo en un esclavo para ser controlado por cualquier maestro, microcontrolador o dispositivo, por un puerto serial.

En la página web de 4D systems encontramos documentación relacionada con todos los entornos de programación, su funcionamiento, tutoriales, y ejemplos con ilustraciones y extractos de código.

A lo largo de la realización del trabajo se busca versionar y modificar constantemente el diseño para su mejora. 4D Systems proporciona una herramienta intuitiva y simple que facilita el desarrollo de una aplicación. El usuario puede centrarse en el aspecto y contenido visual.

Junto a los entornos, 4D Systems proporciona librerías para Arduino, lo que facilita la posibilidad de enviar y recibir información entre el microcontrolador y el módulo de la pantalla a través de un puerto serie. La librería permite leer y modificar el estado y los otros valores correspondientes a los objetos dispuestos en la pantalla.

5.2 Creación de proyectos utilizando Workshop4 (Guia - Tutorial)

1. Para crear un nuevo proyecto en Workshop4 se selecciona la pestaña *New* en el menú de la pantalla de inicio.

Ilustración 5.1 Pantalla inicial Workshop4 [26].

2. se escoge el producto de 4D Systems que será objeto del diseño
3. se determina la posición en la que se quiere generar el display (*Portrait, Landscape, Portrait Rotated, Landscape Rotated*). En nuestro caso teniendo en cuenta la posición que tendrá la pantalla en el dispositivo PRESSMATIC seleccionamos Landscape haciendo click en la representación de la pantalla hasta que se muestre la disposición deseada

Ilustración 5.2 Cuadro de selección de módulo [26].

4. seleccionamos *Next*
5. se escoge el entorno de programación que se desee emplear, en nuestro caso emplearemos ViSi-Genie. Además de los 4 entornos tenemos dos editores a elegir: Create system file y Create text file.

Ilustración 5.3 Cuadro de selección de entorno [26].

Así, se abre la ventana correspondiente al entorno seleccionado.

5.3 Entorno ViSi-Genie

Una vez abierto el entorno de se observa una pantalla con seis secciones distinguibles:

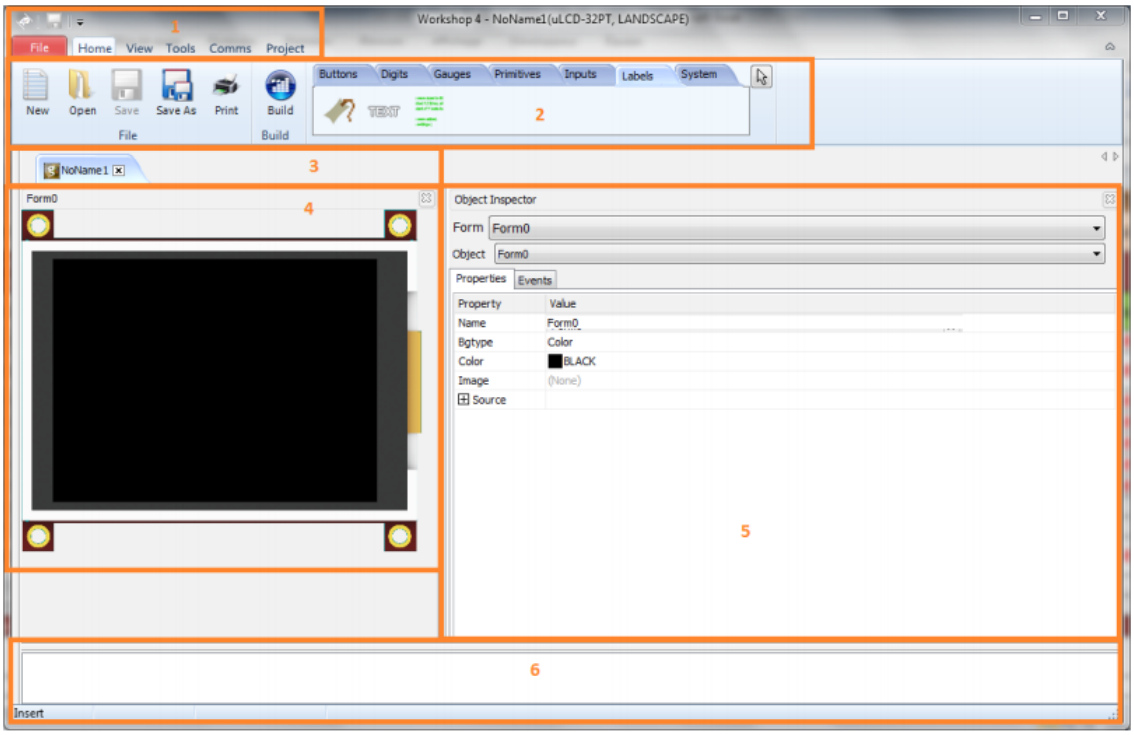

Ilustración 5.4 Entorno de ViSi-Genie y sus secciones [26].

Menús: esta sección incluye las opciones estándar de Windows.

1. Cinta de iconos: una barra con accesos rápidos (New project, Open project, Save project, Save as project, Print project, and Build project) y un selector de objetos.
2. Lista de proyectos abiertos.
3. Forma y pantalla WYSIWYG (what you see is what you get) para colocar objetos: la forma y sus objetos se muestran en esta ventana tal y como aparecerán en la pantalla táctil, en ella se diseña gráficamente el aspecto, el tamaño, y la posición que tienen los objetos.
4. Inspector de Objetos: se muestra una lista de propiedades y eventos que pueden ser modificados.
5. Ventana de mensajes de error, avisos, y notificaciones.

Para la programación de la pantalla en ViSi-Genie se emplea el diseño gráfico a través de la ventana que muestran la forma tipo WYSIWYG (what you see is what you get), y con el Inspector de Objetos se definen las propiedades y eventos de los objetos insertados en la forma.

5.4 Objetos

Los objetos son los elementos predefinidos que se emplean en ViSi-Genie para la programación y diseño de una aplicación. Encontramos tres tipos de objetos según la comunicación serial o propia a la aplicación que puedan mantener [26]:

- Objetos de entrada: objetos que el usuario puede emplear para introducir información. Producen eventos para otros objetos de entrada o para la salida del puerto serial (report event). Por ejemplo un botón.
- Objetos de salida: reaccionan ante la entrada de datos, ya sea a través del puerto serie o de un objeto de entrada. El estado de estos objetos cambia según los valores de entrada y sirven para representar información al usuario. Estos objetos no producen eventos, por ejemplo un "meter".
- Objetos de entrada y salida: la mayoría de los objetos disponibles son de entrada y salida, permitiéndoles enviar y recibir información. Muchos objetos necesitan ser controlados a través de la entrada de información por puerto serial.

Los objetos disponibles se encuentran entre las siguientes categorías:

- Button
- Digits
- Gauges
- Primitives
- Inputs
- Labels
- System / Media
- I/O

A continuación se describen los objetos que más relevancia han tenido a lo largo del desarrollo de la aplicación para PRESSMATIC.

5.4.1 Forms

Las formas son descritas como *páginas en la pantalla*.

El objeto "form" es de salida y representa una página sobre la que se dispondrá el resto de objetos. Las formas representan el espacio en el que se mostrarán los objetos tal y como aparecen (WYSIWYG). Tienen un único evento que es onActive, así, el mensaje de aviso de evento se transmite al maestro tras activarse. El maestro puede cambiar y/o activar cualquier forma a través del comando Write Object Value. A su vez cualquier objeto de entrada puede activar cualquier forma desde a través de un evento. Al activarse cualquier forma se muestra junto a los objetos que se hayan dispuesto en ella.

5.4.2 Button objects

Entre los botones disponibles encontramos cuatro tipos: Win Button, User Button, Animated Button, y 4D Button. Los botones tienen un solo evento, onChanged. Los botones que se emplean en el proceso de diseño son Win Buttons y User Buttons principalmente.

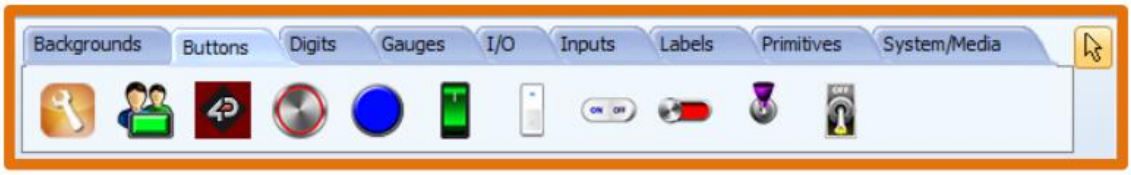

Ilustración 5.5 Barra de botones.

Los Win Button son unos objetos que emulan el estilo de los botones de Windows. Cambia de estado al ser presionado y soltado. Esto activa el evento, que permite enviar un mensaje por serial o activar otro objeto.

El User Button es un botón genérico que emplea una imagen cualquiera seleccionada por el usuario para crear un objeto personalizado. Puede tener hasta cuatro estados diferentes para los cuales el usuario debe proveer una imagen [28].

Los botones usados en el diseño final son User Buttons, debido a que muestran una imagen personalizada la versatilidad que ofrecen hará más fácil la creación de un diseño accesible y original.

Entre las propiedades de los botones de usuario encontramos:

- Name
- Height: tamaño y.
- Images: imagenes provistas por el usuario.
- Left: posición x.
- Matrix: matriz de botones donde solo uno es seleccionado.
- Momentary: momentáneo o toggle.
- Stretch
- Top: posición y.
- Width: tamaño x.

Además disponemos de los botones Animated Button y los 4D Buttons. Los primeros reproducen una serie de imágenes al ser pulsados, y entre los segundo encontramos cuatro objetos distintos: Button, Rocker, Slider, y Toggle.

5.4.3 LEDs and Digits Objects

La pestaña Dígitos contiene cuatro objetos: LED Digits, Custom Digits, LED, y User LED. Estos son objetos de entrada y salida, tenien eventos, onCanged, y reaccionan ante entradas.

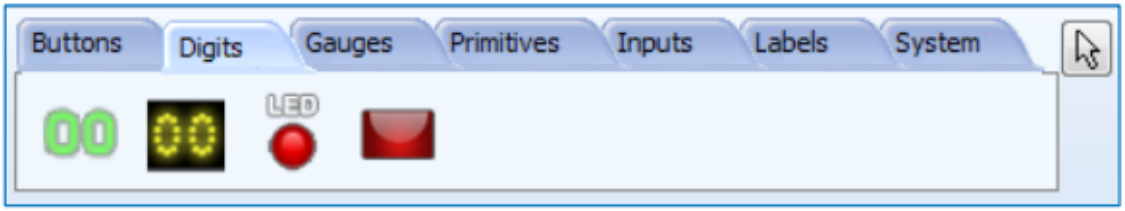

Ilustración 5.6 Barra de dígitos.

Los LED Digits y Custom Digits son displays de 7 segmentos. Se puede definir el color, la forma, y el tamaño de los segmentos de los custom digits según un 'Bitmap'. Un objeto de entrada puede cambiar el estado de los dígitos y a su vez éstos pueden generar un evento, onChanged. Se emplean para mostrar en la interfaz los valores que toman ciertas variables en los primeros diseños.

Los LEDs son objetos de entrada y salida, personalizables. Se emplean para representar el estado de ciertas variables, como el Bluetooth (activo/inactivo).

5.4.4 Inputs Objects

La pestaña de Inputs contiene selectores rotativos, selectores lineales, teclados, interruptores, y paleta de colores.

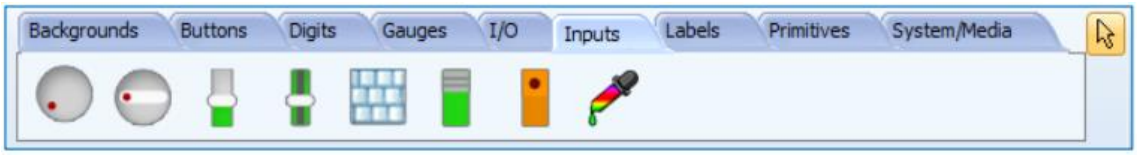

Ilustración 5.7 Barra de objetos de entrada.

El Slider es un objeto de entrada que consta de una barra y un deslizador que se mueve entre unos valores máximo y mínimo customizables. Tiene dos eventos: onChanged y onChanging. Puede enviar información a tiempo real mientras está pulsado el objeto y una vez se ha soltado.

En los diseños iniciales se emplea para determinar el valor de ciertas variables, ya sea la velocidad o el número de ciclos. Es un objeto muy intuitivo.

El Dip-Switch es un interruptor que puede tener dos o más posiciones.Tiene dos eventos: onChanged y onChanging. Puede enviar información a tiempo real mientras está pulsado el objeto y una vez se ha soltado

5.4.5 Labels Objects

La pestaña de Labels ofrece tres métodos de entrada de texto en la pantalla: Label, Static Text, y Strings.

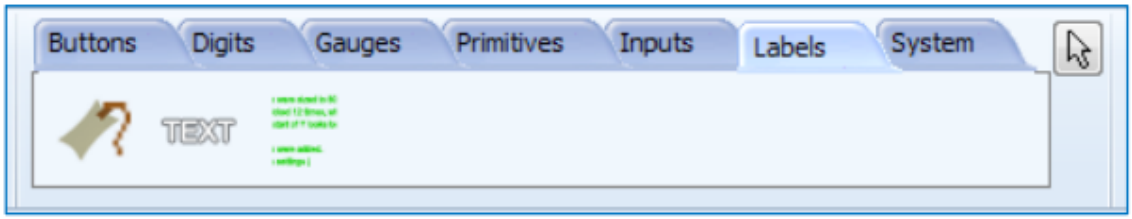

Ilustración 5.8 Barra de etiquetas.

Static Text es un texto que aparece como parte de la forma.

Las strings son objetos que pueden mostrar distintos textos a través de comandos por serial desde el maestro. Emplear valores predeterminados incrementa la efficiencia y reduce el tamaño del código necesario. El tamaño máximo predeterminado es de 75 caracteres pero puede ser cambiado a través de Workshop4. Un objeto de entrada puede tener como evento el objeto string, haciendo cambiar el mensaje que se lee. A su vez el objeto string puede tener otro objeto de salida como evento.

5.5 Diseño de botones

El software de 4D Systems pone a disposición una serie de botones en forma de objetos que pueden ser personalizados. Estos son los User Buttons, que permiten decidir al usuario qué imagen mostrarán en los distintos estados en los que se puede encontrar el objeto.

Así, para lograr una interfaz más accesible se diseñan gráficamente unos iconos para su empleo en la representación de los botones de los que dispondrá. Además esto hará que resulte más fácil crear distintas interfaces cohesionadas en cuanto al su diseño.

5.5.1 Herramientas de diseño gráfico vectorial

Para el diseño de unos iconos que se adaptasen lo mejor posible a las pautas de accesibilidad, expuestas en la sección 3.3, se decide emplear un software de diseño gráfico vectorial, ajeno a 4D Systems.

Entre las diferentes herramientas que se consideraron para la tarea en cuestión se encuentran: Adobe Photoshop, Adobe Illustrator e Inkscape [3]. Al comparar los distintos softwares de Adobe, se extrae que el empleo de una herramienta de diseño gráfico vectorial, como Illustrator, facilita la creación de iconos y logos frente a un editor de gráficos rasterizados, como Photoshop. Inkscape es un programa libre y de código abierto que ofrece un gran abanico de herramientas para la maquetación y el diseño de los logos e iconos. Además, Inkscape ofrece gran cantidad de material de aprendizaje y el soporte de una comunidad de usuarios.

Debido a las ventajas que ofrece un software libre y de código abierto frente a un producto como Adobe Illustrator, finalmente se decide emplear Inkscape como herramienta de diseño para la creación de los iconos.

5.5.2 Diseño de los iconos para las interfaces

La creación de los iconos necesarios a través de la herramienta Inkscape fue llevada a cabo por Alonso Rosado. El diseño de éstos está sujeto a las normas y pautas de accesibilidad, en la medida de lo posible, adaptándolas a las interfaces empleadas.

El primer asunto que se tratado, es el de la visualización de los objetos. Los botones debían ser fácilmente percibidos, por lo que atendiendo a las pautas establecidas por W3C se debe generar un alto contraste entre los iconos y el fondo en el que disponen. Esto se consigue de dos maneras:

El empleo de colores complementarios en el fondo y los iconos. La naturaleza de estos colores les otorga vistosidad cuando se presentan superpuestos o próximos.

Ilustración 5.9 Círculo cromático.

La aplicación de sombreados y brillos en la periferia de los botones y los logos, que también genera contraste debido a la diferencia de brillo.

En la figura siguiente se observa la aplicación de las pautas y las alternativas que ofrece el uso de distinto colores complementarios a la hora de diseñar la interfaz.

Ilustración 5.10 Opciones de diseño [3].

Para permitir una navegación accesible de la aplicación, siguiendo las pautas expuestas en punto 3.3.2, se crean iconos que empleen texto o bien una representación pictográfica, en caso de que el texto que lo describa no se breve, que describa de manera clara y concisa la funcionalidad de cada botón.

A su vez, los iconos deben ser grandes y tener una definición alta para una visualización clara. Para ello se diseñan tres tamaños y proporciones para la presentación de distintos botones. Las distintas dimensiones en píxeles y alto por ancho son: 120x120, 120x360, y 60x120.

Ilustración 5.11 Botones de distintos tamaños de izquierda a derecha 120x120, 120x360, 60x120.

Por último, para facilitar la visualización de información, se distinguen los estados de los botones empleando distintas configuraciones gráficas.

- No Pulsado: tono más claro, bordeado de sombra negro y brillo de blanco puro en la esquina izquierda inferior.
- Pulsado: tono más oscuro, sin brillo y con un borde amarillo

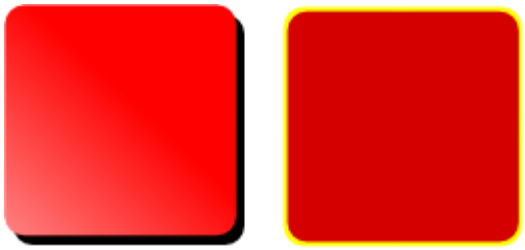

Ilustración 5.12 Diseño de botones no pulsado y pulsado.

6 La Interfaz

6.1 Evolución de la Aplicación

6.1.1 Funcionalidad de la interfaz

Al comienzo del proceso de la creación de las aplicaciones para la pantalla táctil y Android se determinan las funcionalidades, las opciones, y variables que se desea incluir en la interfaz a generar.

La idea inicial se compone de una serie de pantallas y controles que permiten determinar el número de aperturas y cierres o ciclos que se realizan al cortar, la velocidad del proceso, o el incremento que se quiera abrir o cerrar los actuadores, según la función seleccionada y que desempeñe PRESSMATIC. El diseño se basa en la elección de un modo de funcionamiento entre los modos "continuo", "grip" o "paso a paso", y "ciclos".

Esta idea se consideró funcional. El nivel al que se accedería mediante la interfaz sería equivalente al de control implementado en el Arduino. Pero a su vez se concluyó que podría resultar poco intuitivo para usuarios no especializados.

Así, se decide definir un modelo de más alto nivel y más intuitivo donde se reduzcan las variables. Se presenta como alternativa un diseño en el que el usuario determina la herramienta que desea emplear. Con ello ciertas variables de control pueden ser determinadas de antemano y el usuario se encuentra frente a una aplicación simplificada. El diseño se basa en la elección de la herramienta que se desea emplear.

6.1.2 Primera versión

Inicialmente el menú principal se compone de cuatro winbuttons: tres botones de herramientas y uno de ajustes. El winbutton correspondiente a los ajustes se reemplaza por un icono empleando un user button y a la vez se añade otro de ayuda. Se decide poner un color de fondo y de objetos suave, azul y gris o blanco respectivamente. Los botones se enmarcan para hacerlos más visibles, y se juega con su posicionamiento.

Ilustración 6.1 Pantalla de inicio, primera versión.

Se genera un submenú para las tijeras como modelo prestando atención a la selección de parámetros para llevar a cabo la acción de corte. La selección de las tijeras en el menú principal lleva al usuario a una pantalla donde debe seleccionar el modo de corte que desea, ya sea un corte continuo o un determinado número de ciclos de apertura y cierre de la herramienta. Cada modo de funcionamiento tiene una opción de selección de parámetros: en el modo continuo es la velocidad y en el modo paso a paso es el número de ciclos de corte deseados.

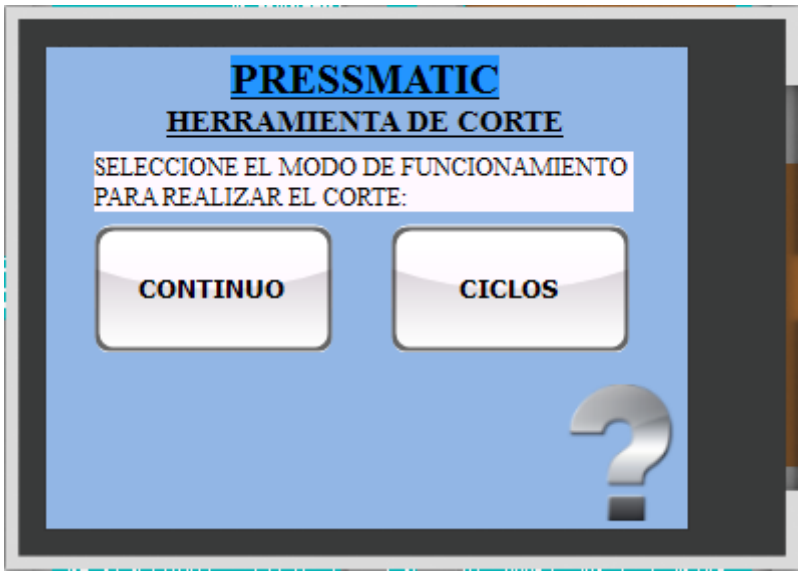

Ilustración 6.2 Pantalla de herramienta de corte, primera versión.

El diseño del fondo y de los botones sigue el mismo patrón que en el menú principal, exceptuando los botones de acción: "START" y "STOP". Éstos se enmarcan y describen en colores vivos, verde lima y rojo. Con esto se quiere aumentar la accesibilidad, dada la conexión que hay entre los colores empleados y las funciones de los botones.

Ilustración 6.3 Pantalla de modo continuo, primera versión.

Ilustración 6.4 Pantalla de modo ciclos, primera versión.

Para la selección de parámetros se emplea un objeto llamado slider. Arrastrando el slider entre unos valores máximo y mínimo se determina el valor que toma la variable correspondiente. Para asistir al usuario se emplea un objeto llamado LED digits para representar numéricamente el valor de las variables de velocidad, en porcentaje, y el número de ciclos. La representación numérica se actualiza a tiempo real mediante la selección del evento OnChanging [27].

6.1.3 Segunda versión

En la siguiente iteración se centra en cambiar los colores empleados en el diseño de la interfaz con el objetivo de aumentar su visibilidad y con ello la accesibilidad. El aspecto que aumenta la visibilidad de los objetos es el contraste. Para aumentar el

contraste entre el fondo y los botones se diseña una pantalla que emplea colores complementarios. El diseño resulta en el uso de un azul de fondo que contrasta con unos botones naranjas.

Ilustración 6.5 Pantalla de inicio, segunda versión.

6.1.4 Tercera Versión

Con objeto de simplificar la utilización de la herramienta de tijeras se elimina la elección de los modos de corte, dejando como único el continuo. Se elimina por lo tanto el modo de paso a paso, donde se variaba el número de ciclos deseado.

La selección de la velocidad de corte se hará mediante la selección de un botón de una matriz de tres botones: velocidad lenta, media, y alta.

La herramienta de pinzado tendrá como interfaz una pantalla con dos botones de acción para abrir o cerrar los actuadores por incrementos. Se estudia la posibilidad de ofrecer una selección de velocidad de acción.

Se observa la posibilidad de añadir mechero pero mecánicamente no resulta viable

Se añade la herramienta de cortaúñas, el aparato realiza un ciclo de corte. La pantalla correspondiente a esta herramienta consta de un botón

Así se simplifica la aplicación considerablemente y es más accesible.

Se diseñan botones personalizados con iconos para las herramientas y se emplean en los user buttons, disponibles en workshop4.

Se emplean strings para generar mensajes emergentes en las pantallas de Bluetooth y de la herramienta de corte.

En cuanto a los colores del diseño, manteniendo el principio de emplear colores complementarios, se escoge como colores para el fondo y los botones, turquesa y

rojo respectivamente. Alternativamente se generará una alternativa que emplee los colores amarillo-anaranjado para el fondo y azul para los botones.

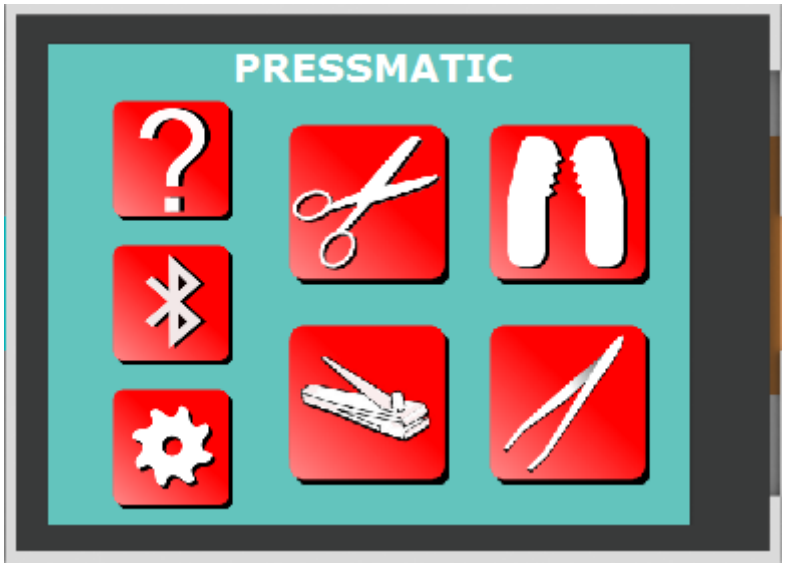

Ilustración 6.6 Pantalla de inicio turquesa/rojo

Ilustración 6.7 Pantalla de inicio amarillo-anaranjado/azul

6.2 Descripción comprensiva de la interfaz

6.2.1 Los menús y sus funcionalidades

La funcionalidad del diseño se rige por la selección de una de las herramientas disponibles en la pantalla inicial: tijeras, pinzas grandes o pequeñas, y cortaúñas.

6.2.1.1 Inicio

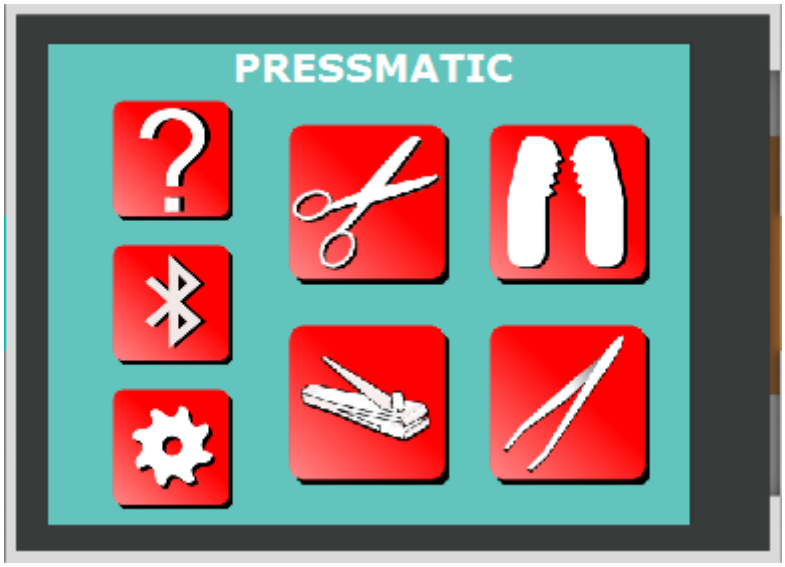

Ilustración 6.8 Pantalla de inicio, tercera versión.

Esta es la pantalla que se muestra al iniciarse la aplicación. Permite seleccionar cualquiera de los modos de funcionamiento del dispositivo, los ajustes, la conexión Bluetooth, o la información. Como cabecera tenemos un texto.

Hay dos grupos de botones: modos y otras opciones

Tabla 1 Configuración de la pantalla de inicio

Objeto	NameIndex	Propiedades	Eventos
Pantalla	Form0	color:0xB9C15D	Report Message
Título	statictext1	Caption: PRESSMATIC	
		Font: white, Verdana, 14, bold	
		Transparent: yes	
Tijeras	Userbutton1	Momentary: yes	Form1 Active
		Width: 80	
		Height: 80	
Pinzas grandes	Userbutton2	Momentary: yes	Form2 Active
		Width: 80	
		Height: 80	
Cortaúñas	Userbutton3	Momentary: yes	Form4 Active

		Width: 80	
		Height: 80	
Pinzas pequeñas	Userbutton4	Momentary: yes	Form2 Active
		Width: 80	
		Height: 80	
Ayuda	Userbutton8	Momentary: yes	
		Width: 60	
		Height: 60	
Bluetooth	Userbutton9	Momentary: yes	Form3 Active
		Width: 60	
		Height: 60	
Ajustes	Userbutton10	Momentary: yes	
		Width: 60	
		Height: 60	

6.2.1.2 Tijeras

Ilustración 6.9 Pantalla de tijeras, tercera versión.

La selección de las tijeras en el menú redirige al usuario a una pantalla, donde podrá seleccionar la velocidad de operación deseada entre tres velocidades disponibles y mediante el botón de acción iniciar o detener la acción de corte. A través de una matriz de botones se selecciona la velocidad, ya sea lenta, media, o alta. Gracias a la función las matrices de botones, solamente uno de los dispuestos en ella será seleccionado, no permite seleccionar varios simultáneamente. La acción de corte se inicia al pulsar y soltar el botón de "START/STOP" y se detiene una vez se vuelve a pulsar y soltar el mismo botón.

Tabla 2 Configuración de la pantalla de tijeras

Objeto	NameIndex	Propiedades	Eventos
Pantalla	Form1	color:0xB9C15D	Report Message
Título	statictext2	Caption: PRESSMATIC	
		Font: White, Verdana, 14, Bold	
		Transparent: yes	
Velocidad lenta	Userbutton0	Momentary: no	Report Message
		Matrix: 1	
		Width: 80	
		Height: 80	
Velocidad media	Userbutton5	Momentary: no	Report Message
		Matrix: 1	
		Width: 80	
		Height: 80	
Velocidad rapida	Userbutton6	Momentary: no	Report Message
		Matrix: 1	
		Width: 80	
		Height: 80	
Start/Stop	Userbutton7	Momentary: no	Report Message
		Width: 210	
		Height: 72	
Inicio	Userbutton18	Momentary: yes	Form0 Active
		Width: 80	
		Height: 40	
Notificación	Strings0	BGcolor: White	
		FGcolor: Black	
		StringStyle: Message	
		Width: 160	
		Height: 24	

6.2.1.3 Pinzas

La selección de las pinzas grandes o pequeñas en el menú de inicio redirige al usuario a una pantalla, donde dispone de dos botones de acción de apertura y cierre de los actuadores. La velocidad con la que se realiza la operación viene determinada por la última velocidad seleccionada por el usuario. El proceso de apertura o cierre por incrementos fijos se inicia al pulsar y soltar el botón correspondiente a la acción, ya sea “ABRIR” o “CERRAR”.

***La velocidad de la operación podría ser determinada constante en el código de Arduino, o bien habilitar una sección en los ajustes para seleccionar una velocidad base de acción.

Ilustración 6.10 Pantalla de pinzas grandes y pequeñas, tercera versión.

Tabla 3 Configuración de la pantalla de pinzas

Objeto	NameIndex	Propiedades	Eventos
Pantalla	Form2	color:0xB9C15D	Report Message
Título	statictext0	Caption: PRESSMATIC	
		Font: White, Verdana, 14, Bold	
		Transparent: yes	
Abrir	Userbutton11	Momentary: no	Report Message
		Width: 240	
		Height: 80	
Cerrar	Userbutton12	Momentary: no	Report Message
		Width: 240	
		Height: 80	
Inicio	Userbutton17	Momentary: yes	Form0 Active
		Width: 80	
		Height: 40	

6.2.1.4 Cortaúñas

La selección del cortaúñas en el menú de inicio redirige al usuario a una pantalla, donde dispone de un botón de acción: cortar. Esta herramienta lleva a cabo un ciclo

de corte completo, posicionando, cerrando, y abriendo los actuadores. La operación se inicia al pulsar y soltar el boton de "CORTAR". La velocidad con la que se realiza la operación viene determinada por la última velocidad seleccionada por el usuario.

Ilustración 6.11 Pantalla de cortaúñas, tercera versión.

Tabla 4 Configuración de la pantalla de cortaúñas

Objeto	NameIndex	Propiedades	Eventos
Pantalla	Form4	color:0xB9C15D	Report Message
Título	statictext5	Caption: PRESSMATIC	
		Font: White, Verdana, 14, Bold	
		Transparent: yes	
Cortar	Userbutton14	Momentary: yes	Report Message
		Width: 270	
		Height: 90	
Inicio	Userbutton15	Momentary: no	Form0 Active
		Width: 240	
		Height: 80	

6.2.1.5 Conexión Bluetooth

Para que sea posible la conexión Bluetooth entre PRESSMATIC y un dispositivo Android o el módulo de control por voz es necesario que sea activado accediendo al menú a través del botón correspondiente de la pantalla de inicio.

Una vez se habilita, tras pulsar y soltar el botón, cualquier dispositivo Android y el módulo de control por voz podrá establecer conexión con PRESSMATIC.

El botón dispuesto en el menú Bluetooth se emplea para habilitar y deshabilitar la conexión de PRESSMATIC mediante Bluetooth con otros dispositivos. Una Luz de

indicación "LED" indica el estado de las conexiones y unos mensajes emergentes avisan de los cambios.

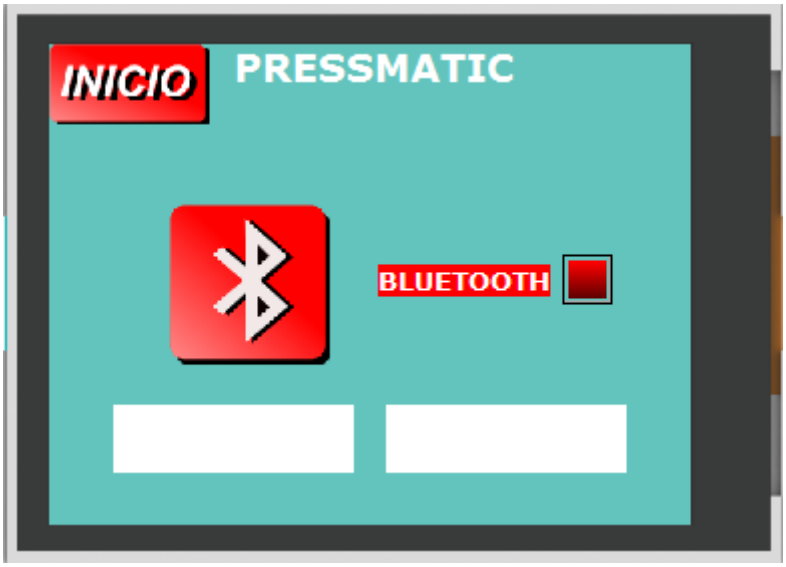

Ilustración 6.12 Pantalla de Bluetooth, tercera versión.

Tabla 5 Configuración de la pantalla de Bluetooth

Objeto	NameIndex	Propiedades	Eventos
Pantalla	Form3	color:0xB9C15D	Report Message
Título	statictext4	Caption: PRESSMATIC	
		Font: white, Verdana, 14, bold	
		Transparent: yes	
Etiqueta	statictext3	Caption: BLUETOOTH	
		Color: Red	
		Font: white, Verdana, 10, bold	
		Transparent: No	
Acción Bluetooth	Userbutton13	Momentary: No	Report Message
		Width: 80	
		Height: 80	
Inicio	Userbutton16	Momentary: yes	Form0 Active
		Width: 80	
		Height: 80	
LED	Userled0	Active: no	
		Width: 25	
		Height: 25	
		BorderColor, InnerColor, InnerHighlight, OuterColor, OuterHighlight: 0xB9C15D	

		InnerStyle: none	
Notificación 1	Userbutton8	BGcolor: White	
		FGcolor: Black	
		StringStyle: Message	
		Width: 120	
		Height: 34	
Notificación 2	Userbutton9	BGcolor: White	
		FGcolor: Black	
		StringStyle: Message	
		Width: 120	
		Height: 34	

6.2.1.6 Sección de ayuda

El menú de ayuda se compone de una serie de pantallas donde se explica y aclara el funcionamiento de la aplicación.

- ¿Qué es PRESSMATIC? y ¿Cuál es su propósito?: PRESSMATIC es una herramienta ideada para la asistencia en la realización de tareas y acciones cotidianas que conllevan un movimiento manual de pinzado, tales como el uso de tijeras (cortar) y agarrar.
- La interfaz y el control de PRESSMATIC: La interfaz integrada en PRESSMATIC es una pantalla táctil en la que se presenta una serie de posibles acciones y configuraciones mediante botones que se activan tras presionarlos y soltarlos. PRESSMATIC ofrece la posibilidad de ser controlado a distancia por Bluetooth mediante un dispositivo android, a través de una aplicación específica, o mediante un dispositivo de control por voz. Los métodos de control alternativos son compatibles con la pantalla táctil, por lo tanto, cualquier acción sobre ellos se verá reflejada también en la interfaz integrada.
- Los menús y sus botones:
    - Inicio: Para volver a la pantalla de inicio se ha dispuesto un botón titulado "inicio" en las demás pantallas. Los botones disponibles, que abren los submenús correspondientes, son:
        - Tijeras
        - Pinzas Grandes
        - Pinzas Pequeñas
        - Cortaúñas
        - Bluetooth
        - Ayuda
    - Tijeras: tres botones para la selección de la velocidad y un botón de acción que inicia y para la ejecución. Además hay una ventana donde se muestra un mensaje emergente.

- Pinzas (GyP): llevan a la misma pantalla con dos botones de acción para abrir y cerrar los actuadores.
- Cortaúñas: Lleva a una pantalla con un botón de acción.
- Bluetooth: Un botón de acción, un LED que indica el estado, y dos ventanas para mensajes emergentes.
- Ayuda: botones de navegación: imágenes de los objetos dispuestos en la aplicación y una breve descripción comprensiva de su funcionamiento.

6.3 Adaptación de las pautas a la aplicación

6.3.1 Accesibilidad visual

Los botones emplean un color complementario al fondo. Botones de color rojo o azul y el fondo de color turquesa o amarillo/naranja. Los botones y sus logos emplean sombreados y marcos para mayor contraste.

El texto se escribe de manera que contraste con el fondo, negro sobre blanco, con contorno en caso del texto en los botones. Además se configura con un tamaño y una fuente, Verdana, que lo haga fácilmente visible.

6.3.2 Navegación

Para orientar al usuario se proporciona información sobre el plano de la aplicación e información sobre la disposición de las pantallas en la sección de ayuda.

La navegación y el estilo son consistente dentro de distintos menús, siguiendo una configuración de botones y diseño consistente.

Se facilita la percepción cognitiva de las acciones y propósitos de los botones mediante el diseño de iconos y logos que los representen de manera clara y concisa, ya sea gráficamente o mediante texto.

Se proporcionan mensajes de texto cuando se activan ciertas acciones dentro de un menú. (Se podría añadir un mensaje indicando la velocidad seleccionada en el menú tijeras)

6.3.3 Control

Desde el diseño inicial el número de pasos y/o transiciones requerido para realizar una acción o acceder a una pantalla se reduce significativamente. Inicialmente el proceso consta en escoger un modo de funcionamiento para la herramienta y tras ello habría que determinar dos variables en menús separados para llegar a una pantalla de acción. El modelo final consta de un diseño en el cual, tras la elección de la herramienta, se muestra el submenú donde se pueden realizar las acciones correspondientes y el usuario debe determinar a lo sumo un parámetro.

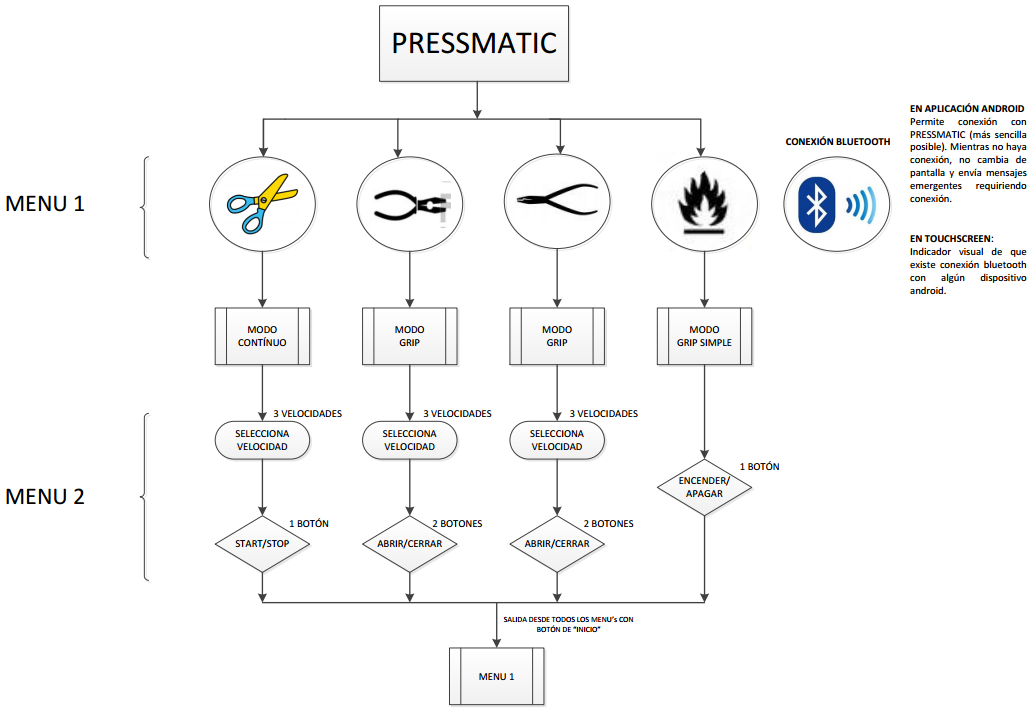

Ilustración 6.13 Control accesible de PRESSMATIC

Existen notificaciones de texto disponibles en ciertos menús para determinadas acciones.

Los botones son grandes y se encuentran espaciados para hacer más fácil la selección de la acción deseada. Se reduce la probabilidad de seleccionar un botón erróneamente. Esto además limita los botones que se pueden disponer en una pantalla.

Se proporciona un botón de inicio en todas las pantallas para hacer fácil el retroceso a la pantalla principal en caso necesario.

Se proporcionan alternativas de control:

- Control por voz: módulo EasyVR compatible con la pantalla táctil. Conexión con Arduino mediante Bluetooth.

- Control desde un dispositivo Android: hecho posible a través de una "app" específica para PRESSMATIC compatible con la pantalla táctil. Conexión con Arduino mediante Bluetooth.

Esto proporciona la posibilidad de manejar el aparato a distancia y facilita el control a un mayor número de personas de distintas diversidades funcionales.

6.4 Compatibilidad con otras interfaces

Para el control de PRESSMATIC desde Android se requiere la aplicación desarrollada por Alejandro Vega y Alonso Rosado. Esta aplicación tiene un diseño que emula al de la pantalla. Los cambios realizados empleando la App se reflejan en la pantalla táctil a través del programa de Arduino que recibe la información y manda órdenes a la pantalla.

Entre la aplicación móvil y la pantalla del dispositivo hay una gran similitud en:

- Colores
- Botones de diseño
- Menús
- Transiciones

Todo ello tiene como objetivo facilitar un uso comprensivo del dispositivo, aumentando la accesibilidad, y ofrecer nuevas posibilidades para el control y manejo a distancia de PRESSMATIC.

Ilustración 6.15 Pantalla de inicio Android [2].

Ilustración 6.14 Pantalla de tijeras Android [2].

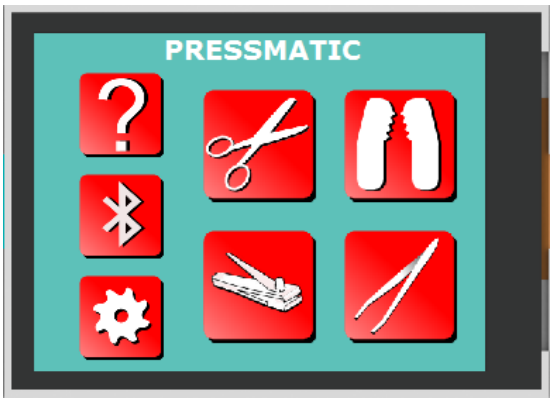

Ilustración 6.16 Pantalla de inicio pantalla táctil

Ilustración 6.17 Pantalla de tijeras pantalla táctil

Los comandos disponibles para el control a través del dispositivo de Easy VR deben ser los equivalentes a los que se ofrecen en la interfaz integrada a través de botones.

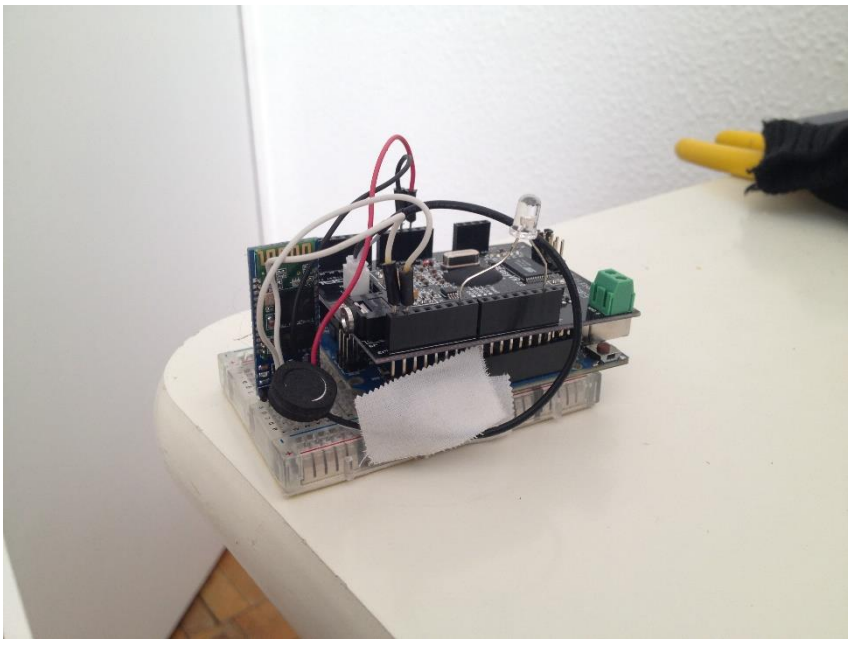

Ilustración 6.18 Módulo Easy VR y Arduino para el control por voz [3].

A través del código implementado en Aruino, incluido en el anexo A.1, se se incluyen elementos que permiten la interacción de estos dispositivos con la pantalla táctil empleando funciones de escritura, que cambian el estado de los objetos de la pantalla. Así si el estado de un objeto se cambia mediante un dispositivo Android o el Easy VR, se podrá mostrar el mismo cambio en la pantalla.

7 Integración de la interfaz en Arduino

Arduino es una plataforma hardware y software de desarrollo, ideada para proyectos interactivos, basada en un modelo de diseño y código abiertos. El entorno está implementado en lenguaje Java y se basa en entornos de código libre como Processing y AVR.

Se han empleado los Arduino UNO y Mega ADK. Para la programación del Arduino se emplea un lenguaje híbrido, simplificado, de los lenguajes C y C++ [29]. El código va enfocado al control del funcionamiento de PRESSMATIC, las conexiones seriales correspondientes a los dispositivos Bluetooth y la pantalla táctil. Se emplea la información que llega desde cualquiera de los dispositivos integrados en el control para ejecutar la acción correspondiente.

Para la asistencia del control del motor y de la comunicación con la pantalla táctil se emplean distintas librerías. La librería PWM facilita la conversión de frecuencias de modulación por ancho de pulso y la librería genieArduino incluye funciones que facilitan la comunicación entre el Arduino y los módulos de 4D Systems que se hayan configurado empleando ViSi-Genie.

7.1 Conexión

El módulo de 4D Systems emplea una interfaz de conexión serial de 5 pines para la programación y para comunicarse con cualquier maestro.

7.1.1 Introducción a las conexiones seriales

La comunicación serial es un método de transmisión de información, en forma de bytes, entre dispositivos electrónicos digitales, un microprocesador o microcontrolador y un periférico. Habitualmente, los datos que se envían suelen tener el formato ASCII.

Los puertos que emplean este método de comunicación emplean una o dos líneas para el envío y la recepción de datos, por lo tanto cada byte se transmite un bit a la vez. Comparándolo con la comunicación en paralelo, que envía un byte a la vez, resulta lento y a su vez sencillo.

La comunicación serial se puede establecer empleando tres métodos según los canales que se dispongan entre el transmisor y el receptor: Simplex, Half-duplex, y Full-duplex [30].

Simplex: Se dispone de un canal de comunicación a través del que se envía información de manera unidireccional. Por lo tanto cada dispositivo será transmisor, o bien receptor. Éste método es poco empleado dado que el receptor nunca podrá informar al transmisor acerca de su estado o de errores en la información recibida.

Half-duplex: empleando una línea de comunicación, tanto el microprocesador como el periférico, envían y reciben información a través de ella de manera no simultánea. Así, el problema que plantea la comunicación Simplex se ve resuelto empleando este protocolo, al ser ámbos dispositivos capaces de informar al otro acerca de su estado y la existencia o falta de errores en los bits recibidos.

Full duplex: En este caso se emplea una estructura de dos líneas de transmisión a través de las cuales se envía o recibe información. Así, tanto el microcontrolador como el periférico pueden enviar y recibir datos simultáneamente.

Por las ventajas que suponen los métodos Half-duplex y Full-duplex, éstos son los más comunes.

Además de los canales de comunicación que se establezcan entre distintos dispositivos, otra característica que diferencia distintos tipos de conexiones seriales es el método que empleen para la propia transmisión de información. Se diferencian dos modos de transmisión: modo asíncrono y modo síncrono.

La transmisión asíncrona complementa cada byte de información con bits de inicio y de parada con objeto de que el emisor y el receptor se sincronicen con el envío de cada palabra. Los bits complementarios son necesarios para esto porque no existe relación temporal entre los dispositivos conectados.

La transmisión síncrona se basa en el envío de información en bloques de entre 128 y 1024 bytes, denominados trama, consecutivamente. Al inicio de cada bloque se envía una secuencia de datos, los bits de sincronismo, que sincronizan los relojes de ambos dispositivos. Existe así una relación temporal entre el emisor y el receptor.

Para evitar la corrupción de datos enviados existen protocolos de detección de errores y en caso de detectarse uno, el receptor podrá solicitar el reenvío de estos. Distintos métodos de detección de errores son el bit de paridad y la verificación por redundancia cíclica (CRC).

7.1.2 Protocolo empleado

La comunicación serial se lleva a cabo a través de una interfaz de 5 pines: +5V, TX, RX, GND, y RES. Así, la estructura incluye dos canales de comunicación y, por lo tanto, el método empleado es Full-duplex.

El modo de transmisión de información es asíncrono. El formato en el que se envían los datos son palabras de 8 bits, sin bit de paridad y con bits de inicio y parada. La información se puede transmitir a frecuencias de entre 300 y 600.000 Baudios.

La información se envía con la siguiente estructura:

- Bit de inicio igual a 0.
- Palabra de 8 bits conteniendo el mensaje. El bit menos significativo o LSB es el primero de la secuencia.
- Bit de parada igual a 1.

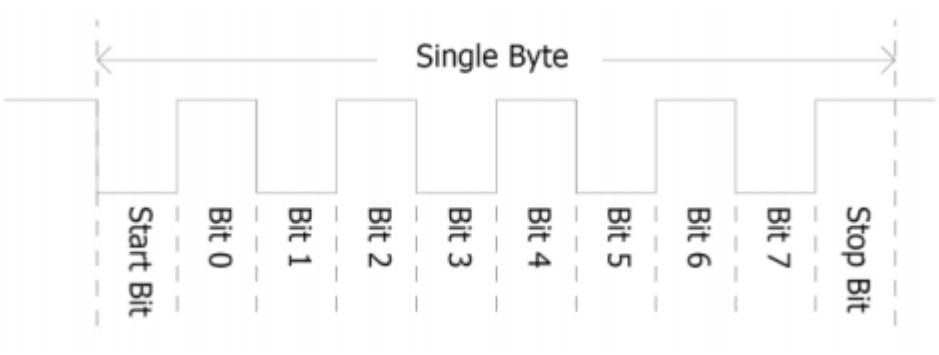

Ilustración 7.1 Estructura de la información enviada por serial [18].

Incluye una aplicación para almacenar información transmitida por el puerto serial que se ejecuta en segundo plano para liberar la aplicación de carga y que pueda atender otras tareas.

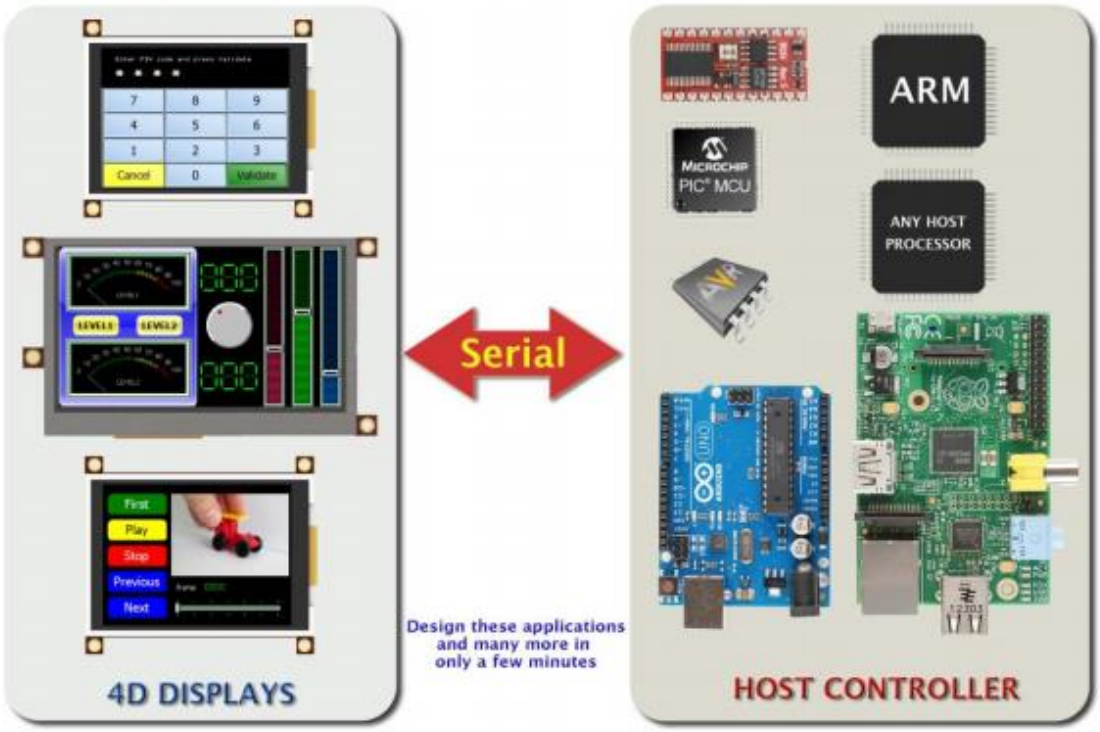

Ilustración 7.2 Diagrama de comunicación entre módulos 4D y maestros [16].

7.2 Librería ViSi-Genie

Una librería es un conjunto de recursos no volátiles usados en la programación y el desarrollo de software. Los recursos que aportan las librerías suelen ser funciones y variables.

4D Systems proporciona librerías para asistir al desarrollo de aplicaciones en sus módulos, entre estas encontramos una dedicada a la integración de los módulos programados empleando ViSi-Genie. Así, en GitHub, una página web dedicada a compartir código, encontramos las librerías que pone a disposición 4D Systems bajo la URL https://github.com/4dsystems.

La librería incluye funciones para establecer conexión por puerto serial, enviar información a objetos, interceptar flags, obtener información sobre los objetos, entre otras utilidades.

Funciones extraídas de la librería genieArduino del archivo genieArduino.h:

```
class Genie {

public:
                        Genie();
        void            Begin                   (Stream &serial);
        bool            ReadObject              (uint16_t object,
uint16_t index);

        uint16_t        WriteObject             (uint16_t object,
uint16_t index, uint16_t data);

        void            WriteContrast           (uint16_t value);
        uint16_t        WriteStr                (uint16_t index, char
*string);

        uint16_t        WriteStrU               (uint16_t index, uint16_t
*string);

        bool            EventIs                 (genieFrame * e, uint8_t
cmd, uint8_t object, uint8_t index);

        uint16_t        GetEventData            (genieFrame * e);
        bool            DequeueEvent            (genieFrame * buff);
        uint16_t        DoEvents                (void);
        void            AttachEventHandler      (UserEventHandlerPtr
userHandler);

        void            pulse                   (int pin);
        void            assignDebugPort         (Stream &port);

private:
        void            FlushEventQueue         (void);
        void            handleError             (void);
        void            SetLinkState            (uint16_t newstate);
        uint16_t        GetLinkState            (void);
        bool            EnqueueEvent            (uint8_t * data);
        uint8_t         Getchar                 (void);
        uint16_t        GetcharSerial           (void);
        void            WaitForIdle             (void);
        void            PushLinkState           (uint8_t newstate);
```

```
        void            PopLinkState            (void);
        void            FatalError              (void);
        void            FlushSerialInput        (void);
        void            Resync                  (void);
```

Los objetos se definen en los mensajes de comunicación serial con las variables contenidas en la estructura:

```
struct FrameReportObj {
        uint8_t         cmd;
        uint8_t         object;
        uint8_t         index;
        uint8_t         data_msb;
        uint8_t         data_lsb;

};
```

7.3 Implementación de la aplicación en Arduino

En la tabla se muestran los eventos que causan la activación de los controles.

Tabla 6 Controles y su interfaz

Modo	Controles	Trigger
Modo Escucha	Modo Continuo	Form 1
	Modo Paso a Paso	Form 2
	Modo Cortaúñas	Form 4
	Bluetooth	Form 3
Modo Continuo	Lento	User Button 0
	Medio	User Button 5
	Rápido	User Button 7
	Start/Stop	User Button 6
Modo Cortaúñas	Cortar Una	User Button 14
Modo Paso a Paso	Abrir	User Button 11
	Cerrar	User Button 12
Bluetooth	Habilitar/Deshabilitar	User Button13
Común a todos los modos	Modo Escucha (inicio)	Form 0

El código escrito en el Arduino se puede visualizar en el apartado A.1, del anexo, y se compone de tres funciones principales:

- SetUp: en ella se definen los parámetros que determinan la velocidad y el inicio de la comunicación serial, el valor del contraste de la pantalla para su visualización, y se incluye la función AttachEventHandler.
- Loop: mediante esta función se determinara el modo de operación según la pantalla activa y las acciones a realizar, se implementa la función doEvents, se escriben objetos y strings en función de la app, y se leen los valores recibidos desde android y el control por voz.
- MyGenieEventHandler: En esta función se obtiene la información sobre los eventos de la pantalla, y se escriben objetos y strings.

Para enviar información a objetos y escribir “strings” se emplean las funciones de escritura:

- genie.WriteObject();
- genie.WriteStr();

Funciones de set up y comandos de comunicación:

- genieFrame Event;
- genie.DequeueEvent(&Event);
- genie.Begin(Serial);
- genie.AttachEventHandler(myGenieEventHandler);
- genie.WriteContrast(1);
- genie.DoEvents();

La función SetUP incluye los comandos necesarios para establecer la comunicación serial, enlazar la función que procesa los eventos, y asignar un pin de reset y realizar la secuencia de inicio de la pantalla. Se ha elegido trabajar con la velocidad marcada por defecto para el serial 0 de la pantalla, siendo 115200 baudios.

```
// opciones de comunicación serial
  Serial.begin(115200);
  genie.Begin(Serial);  //Serial0

  genie.AttachEventHandler(myGenieEventHandler); //enlaza la función que procesa
los eventos

  //Inicio de la pantalla
  pinMode(22, OUTPUT);  // PIN 22 del Arduino como Output para el RESET
  digitalWrite(22, 0);  // Reseteo de la pantalla
  delay(100);
  digitalWrite(22, 1);  // unReset
  delay (3500); //let the display start up
  //Encender la pantalla
  genie.WriteContrast(1); // 1 = Pantalla ON, 0 = Pantalla OFF
```

En MyGenieEventHandler recibe los mensajes de eventos que originan en la pantalla táctil y para reconocer la naturaleza de estos, las tres funciones empleadas que aportan la información sobre cualquier evento que se reciba de la pantalla táctil son:

- *Event.reportObject.cmd*: devuelve el valor de la variable cmd (comando)
- *Event.reportObject.object*: devuelve el tipo de objeto que ha causado el evento
- *Event.reportObject.index*: devuelve el valor correspondiente al index del objeto

Estas funciones se emplean para saber en qué pantalla se encuentra la interfaz y qué objetos, en los que el evento report message esté activo, han cambiado de estado. El empleo de estas funciones se muestra a continuación.

```
void myGenieEventHandler(void)
{
  //Habilitar la entrada de eventos
  genieFrame Event;
  genie.DequeueEvent(&Event);
//******************************************************************************
//REPORT AL CAMBIAR DE PANTALLA
```

```
//**************************************************************************************
  if(Event.reportObject.cmd == GENIE REPORT EVENT)
  {
    if (Event.reportObject.object == GENIE_OBJ_FORM)
    {
      if (Event.reportObject.index == 0)     // If pantalla INICIO
      {
        form=6;
        start=0;
        inicio=0;
      }
```

En la función Loop, se accede a los distintos modos de operación según la pantalla activa en la interfaz y se llevan a cabo las acciones correspondientes. Entre las acciones a realizar se encuentra devolver información a la pantalla para actualizar el estado de determinados objetos. Por ejemplo, para actualizar el estado de las strings en la pantalla de Bluetooth se emplea la función genie.WriteStr como se muestra a continuación:

```
              case 5:  //PANTALLA CONEXIÓN BLUETOOTH

                  if(cuenta == 37)
                  {
                    if(unos >=1 & unos <= 6){
                      genie.WriteStr(2,FRASE6);//escribir string DISPOSITIVO
    CONECTADO
                    }else{
                      genie.WriteStr(2,FRASE5); //escribir string ESPERANDO
    CONEXION
                    }
                    cuenta=0;
                    unos=0;
                  }
                  break;
```

8 Presupuesto

A continuación se desarrolla el presupuesto correspondiente al proyecto realizado, para ello se describen los recursos empleados en forma de costes directos e indirectos. Los costes se desglosan según la naturaleza de los recursos y se analiza el importe a realizar por cada uno de estos.

Los costes directos del proyecto provienen de las implicaciones inmediatas a la producción del mismo e incluyen recursos humanos, recursos materiales, recursos software, y materiales fungibles.

- De **recursos humanos**: En este concepto se incluyen los costes generados por las personas y las horas de trabajo dedicadas en la realización del proyecto. El coste de un graduado en Tecnologías Industriales que ha dedicado 300 horas de trabajo resulta en **5.750€**. El importe se calcula mediante el coste de los hombre mes dedicados, siendo el hombre mes 131.25 horas[1] y el coste de un hombre mes por un graduado en Tecnologías Industriales 2.500€.
  A lo largo de los 6 meses dedicados al proyecto, las horas dedicadas se estiman en:
  - Elección de material 20 horas.
  - Programación y diseño 150 horas.
  - Documentación 100 horas.
  - Reuniones de proyecto 30 horas.

Tabla 7 Costes de recursos humanos

Costes de Recursos Humanos			
Concepto	**Dedicación (Hombre mes)**	**Costes (€/hombre mes)**	**Coste (Euros)**
Graduado en Ingeniería Industrial	2,3 (300 horas)	2.500	5.750
		Total	5.750€

- De **recursos materiales**: gastos derivados del material empleado durante la realización del proyecto. Se incluye el coste proporcional amortizado[2] del hardware de los equipos empleados. El importe total equivale a **165€**.
  - Ordenador: 890€ (con IVA), empleado durante 6 meses de trabajo a depreciación de 60 meses. El coste de amortización resultante es de 89€. Con un 50% de uso dedicado al proyecto el Importe son 44,50€.

[1] 1 Hombre mes equivale a 131.25 horas, siendo el máximo anual de dedicación unos 12 hombres mes, equivalentes a 1575 horas. Para un PDI de la Universidad Carlos III de Madrid, este máximo difiere y se sitúa en 8,8 hombres mes, equivalentes a 1155 horas.

[2] $Valor\ \ de\ Amortización = \frac{A}{B}\ x\ C\ x\ D$

Donde: A = nº de meses desde la fecha de facturación en que el equipo es utilizado, B = periodo de depreciación (generalmente 60 meses), C = coste del equipo, D = porcentaje de uso que se dedica al proyecto.

- Pantalla táctil 79,50€.
- Arduino Mega ADK 39€ (sin IVA).
- Arduino Uno 20€ (sin IVA).

Tabla 8 Costes de recursos materiales

Costes de Recursos Materiales					
Concepto	**Coste (€)**	**Uso dedicado proyecto (%)**	**Dedicación (meses)**	**Periodo de depreciación (meses)**	**Coste Imputable (€)**
Ordenador	890	50	6	60	44,50
uLCD-28PTU-AR	79,50	100	indefinida	60	79,50
Arduino UNO	20	100	6	60	2
Arduino Mega ADK	39	100	indefinida	60	39
				Total	165€

- De **recursos software**: En este apartado se incluye el coste amortizado de los softwares empleados en la realización del proyecto. Estimando, al igual que para el hardware, una obsolescencia en 60 meses se calcula la parte a amortizar en los softwares correspondientes. El resultado del importe son **66€**.

Tabla 9 Costes de software

Costes de Software				
Concepto	**Coste licencia (€)**	**Duración de la licencia**	**Tiempo de uso en el proyecto (meses)**	**Coste imputable (€)**
S.O. Windows 8	119	Indefinida	6	12
Office Professional 2013	539	Indefinida	6	54
Workshop4	0	Indefinida	6	0
Inkscape	0	Indefinida	4	0
			Total	66€

- De **materiales fungibles**: se estiman los costes derivados de materiales de oficina, tales como papelería y cartuchos de tinta, en unos 100€.

Tabla 10 Costes de materiales fungibles

Materiales fungibles

Concepto	Coste (€)
Material de oficina	50
Total	50€

Por otra parte los costes indirectos incluyen gastos derivados indirectamente de la realización del proyecto. Estos se valoran por estimación en un 20% del valor de los costes directos. Esto resulta en un coste indirecto de **1205€**.

Así, el coste resultante de la agrupación de los listados anteriormente, como se muestra en la siguiente tabla, resulta en un presupuesto final de **7230€**.

Tabla 11 Costes totales

Costes Totales	
Concepto	**Coste(€)**
Costes de Personal	5750
Costes de Material	165
Costes de Software	66
Costes de Materiales Fungibles	50
Costes indirectos	1205
Total	**7230€**

9 Conclusión y líneas futuras

9.1 Conclusión

En vista a la conclusión de este proyecto, el objetivo principal era la programación e integración de una interfaz accesible en una pantalla táctil para PRESSMATIC. Este objetivo se ha alcanzado mediante la elección de un módulo integrable en el Arduino empleado que consistiese de una pantalla táctil programable.

El resultado es un prototipo que incluye una interfaz adaptada a las necesidades de personas de distintas diversidades funcionales. Esto se logra mediante una programación en un entorno gráfico, como lo es ViSi-Genie, empleando los elementos provistos por el software de 4D Systems para adaptar la aplicación a las normativas de accesibilidad, tales como las establecidas para la web en por W3C, entre otras.

Se seleccionó una pantalla táctil que se pudiera integrar en el prototipo de PRESSMATIC, gobernado por un microprocesador de Arduino en la placa MEGA. La conexión entre estos dispositivos se realiza a través de una conexión serial full-duplex, asíncrona, mediante una interfaz de 5 pines. Esto resulta en una conexión fiable y robusta que permite el control de PRESSMATIC.

La creación de una interfaz táctil se hizo mediante la programación del módulo a través del entorno ViSi-Genie que permite generar código mediante el diseño gráfico y la personalización de los parámetros de los objetos disponibles para el desarrollo de una aplicación.

Con el objetivo de crear una aplicación accesible, se tienen en cuenta las pautas y normativas a las que se ha tenido acceso a la hora de realizar el proyecto, se han adaptado el entorno y los objetos a fin de que el máximo número de personas de distintas diversidades funcionales puedan hacer uso del dispositivo.

La interfaz generada se compatibiliza con las otras dos opciones de entrada desarrolladas por Alejandro Vega y Alonso Rosado. Así, el control por voz y el control mediante un dispositivo Android se hacen posibles. Al ser compatibles las interfaces interactúan mediante Arduino y los cambios en cualquiera de las otras dos, será visible en la pantalla táctil.

Mediante el empleo de un módulo totalmente compatible con Arduino y programable a través de un entorno gráfico se logra llegar a la resolución de las cuestiones planteadas acerca de la creación de una interfaz táctil accesible para PRESSMATIC.

9.2 Líneas Futuras

Con objeto de llevar más allá el nivel de accesibilidad que ofrece la aplicación, se propone una serie de metas a conseguir en un futuro.

En cuanto al hardware empleado para crear la interfaz, una de las principales mejoras sería encontrar una **pantalla táctil capacitiva** y compatible con el entorno arduino, para sustituir a la actual pantalla resistiva. El uso de una pantalla capacitiva reduciría en gran medida el esfuerzo por parte del usuario.

La adición de **notificaciones auditivas** al accionar un botón incrementaría la accesibilidad cognitiva y auditiva de la interfaz. Habría que grabar las pistas necesarias e integrarlas en la interfaz mediante un objeto de audio.

Para conseguir una aplicación más personalizable se propone poner a disposición del usuario una **sección de ajustes** para variar el tamaño y colores, por ejemplo, de distintos objetos: botones, textos, LEDs, etc. Esto implicaría una mayor accesibilidad visual y cada usuario podría ajustar los parámetros a sus necesidades.

Para hacer esto posible es necesario alejarse en cierta medida de una programación en un entorno tan gráfico para acercarse a un entorno que permita mayor personalización y libertad a la hora de generar una interfaz, tales como los entornos **ViSi** y **Serial**. Mediante el empleo de ViSi o Serial, una vez realizado el primer prototipo, se puede desarrollar el potencial de la aplicación al máximo.

A. Anexo

A.1 Código implementado en Arduino

```
#include <genieArduino.h>
#include <PWM.h>

// en esta version se ha incluido pines de control para ajustar el
microstepping del POLOLOU y además se utiliza la libreria PWM
// para modificar la velocidad del pin 11 que saca la señal step para el motor
// SE AÑADE LA COMUNICACIÓN VIA BLUETOOTH CON LA APLICACIÓN ANDROID (SE UTILIZA
EL PUERTO SERIE 3). SE CAMBIAN LOS COMANDOS UTILIZADOS EN LA COMUNICACIÓN.

#define       FRASE1   "MODO CONTINUO ON"
#define       FRASE2   "MODO CONTINUO OFF"
#define       FRASE3   "BLUETOOTH ACTIVADO"
#define       FRASE4   "BLUETOOTH APAGADO"
#define       FRASE5   "ESPERANDO CONEXION"
#define       FRASE6   "DISPOSITIVO CONECTADO"

char dato_bluetooth = 0;    // Recibe los datos del puerto serie
int c_hechos= 0;         //cuenta el numero de veces que se cierra la pinza, es
decir, numero de veces que se actica la interrupción 1
//int N_ciclos =0;         //numero de ciclos que se quiere realizar estando en
el MODO CICLOS, valor indicado por el usuario
int start = 0;           //permite pausar el sistema en MODO CICLOS
//int estado, estado_ant = 0;        //variable que controla los estados o modos
de operacion en el bucle principal. Varía su valor desde el puerto serie
const int cerrado = 2;      // pin del final de carrera 2 PINZA COMPLETAMENTE
CERRADA
const int abierto = 3;      // pin del final de carrera 1 PINZA COMPLETAMENTE
ABIERTA
const int DIRPin =  4;       // pin que controla el sentido de avance del motor
const int ledverdePin = 5; // LED verde que indica dirección de avance-
retroceso motor
const int en_motor = 6; // Pin conectado al enable del pololou para detener el
motor
const int step_motor = 7; // Salida step para el motor
const int Pinled13 =  13;       // LED de la placa arduino
const int pin21 = 21;       //interrupcion 2 conectada al boton de la shield
para apagar el sistema
const int pin18 = 18;       //interrupcion 5 conectada al boton externo de
CERRAR
const int pin19 = 19;       //interrupcion 4 conectada al boton externo de ABRIR
const int Vblue = 35;       //Pin digital para encender/apagar el bluetooth
const int state_blue = 29;

/*const int MS1 = 23;
const int MS2 = 24;
const int MS3 = 25;*/
const int MS1 = A8;
const int MS2 = A9;
const int MS3 = A10;


// variables will change:
int boton1='0';
int boton2='0';
//volatile int en_motor_on= LOW;
//volatile int motor_estado= HIGH;
volatile int DIRestado = LOW;          // variable for setting the led status
volatile int parpadeo = LOW;          // variable for setting the direction
shaft
volatile int encendido = LOW;          // variable para detener el motor.
volatile int cuenta = 0;
volatile int unos = 0;

volatile int estado = LOW;
```

```
//VALORES PARA LA LIBRERIA PWM PARA CAMBIAR LA FRECUENCIA DE LOS PINES DEL
ARDUINO UTILIZANDO LA LIBRERIA PWM.h
int ciclo = 127;         // ciclo de trabajo
int32_t Vlenta =  1200; //frequency (in Hz)(CON HALF STEP)
int32_t Vnormal = 1000; //frequency (in Hz)
int32_t Vrapida = 1300; //frequency (in Hz)

//*************************************************
//VARIABLES PARA COMUNICACIÓN CON TOUCHSCREEN
//*************************************************
int N ciclos = 0;   //numero de ciclos que se quiere realizar estando en el
MODO CICLOS, valor indicado por el usuario
int led_est = LOW;
int activar = LOW;
int form=0;
boolean inicio=0;
boolean blue_conectado = '0';
boolean flag cerrar = '0';
boolean cortar_una = '0';

void setup()
{
  pinMode(cerrado, INPUT);
  pinMode(abierto, INPUT);
  pinMode(DIRPin, OUTPUT);
  pinMode(ledverdePin, OUTPUT);
  pinMode(en_motor, OUTPUT);
  pinMode(step_motor, OUTPUT);
  pinMode(Pinled13, OUTPUT);
  pinMode(pin21, INPUT);
  pinMode(pin18, INPUT);
  pinMode(pin19, INPUT);
  pinMode(Vblue, OUTPUT);
  pinMode(state_blue, INPUT);

  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(MS3, OUTPUT);
  digitalWrite(MS1,LOW);
  digitalWrite(MS2,LOW);
  digitalWrite(MS3,LOW);


//*************************************************************************
*
  //Se establece la frecuencia del pin que controla la velocidad del motor
  //initialize all timers except for 0, to save time keeping functions
  InitTimersSafe();
  //sets the frequency for the specified pin
  bool success = SetPinFrequencySafe(step_motor, Vnormal);
  //if the pin frequency was set successfully, turn pin 13 on
  if(success) {
    digitalWrite(Pinled13, HIGH);
    }


//*************************************************************************
**
  //HABILITA LAS INTERRUPCIONES EXTERNAS PARA FINALES DE CARRERA y BOTON
  attachInterrupt(0, abrir, RISING);
  attachInterrupt(1, cerrar, RISING);
  //attachInterrupt(2, on_off, RISING);
  attachInterrupt(4, boton_exterior_1, RISING);
  attachInterrupt(5, boton_exterior_2, RISING);


//*************************************************************************
**
  // opciones de comunicación serial
  Serial.begin(115200);
  genie.Begin(Serial);  //Serial0
```

```
  genie.AttachEventHandler(myGenieEventHandler); //enlaza la función que
procesa los eventos

  //Inicio de la pantalla
  pinMode(22, OUTPUT);  // PIN 22 del Arduino como Output para el RESET
  digitalWrite(22, 0);  // Reseteo de la pantalla
  delay(100);
  digitalWrite(22, 1);  // unReset
  delay (3500); //let the display start up
  //Encender la pantalla
  genie.WriteContrast(1); // 1 = Pantalla ON, 0 = Pantalla OFF

  //*********************************************************************
  //INICIALIZACIÓN DEL TIMER1_16BITS PARA PANTALLA BLUETOOTH
  TCCR1A = 0; //No genera formas de onda de salida
  TCCR1B |= (1 << WGM12);
  TCCR1B |= (0 << CS12); // set prescaler = 8
  TCCR1B |= (1 << CS11);
  TCCR1B |= (0 << CS10);
  TCNT1H = 0;//initialize counter value to 0;
  TCNT1L = 0;
  // set compare match register to desired timer count:
  OCR1AH = 26;
  OCR1AL = 60;

  TIMSK1 |= (1 << OCIE1A); //HABILITO INTERRUPCIÓN DE MATCH A EN MODO CTC
  //*********************************************************************
  Serial3.begin(9600);
  delay(10);

}

void loop()
{

   genie.DoEvents();
   //Comunicación serie puerto 3
   if (Serial3.available() > 0){
     dato_bluetooth = Serial3.read();
     Serial3.println(dato bluetooth);
     switch (dato_bluetooth){ //Casos de uso
        case 'a': form=6;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x00, 0x00);
                  start=0;
                  inicio=0;
                  break;
        case 'b': form=1;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x01, 0x01);
                  inicio=0;
                  break;
        case 'c': form=2;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x02, 0x02);
                  start=0;
                  inicio=0;
                  break;
        case 'd': form=3;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x03, 0x03);
                  break;
        case 'e': form=4;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x04, 0x04);
                  break;
        case 'f': form=5;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x05, 0x05);
                  break;
        case 'g': form=0;
                  genie.WriteObject(GENIE_OBJ_FORM, 0x00, 0x00);
                  break;
  //*********************************
        case '0': led_est = !led_est;
                  if(led_est==HIGH){
                    start=1;
```

```
                    genie.WriteObject(GENIE_OBJ_LED, 0x00, 0x01);     // Write
to LED Digits 1
                    genie.WriteStr(0,FRASE1);
                    }
                  if(led_est==LOW){
                    start=0;
                    genie.WriteObject(GENIE_OBJ_LED, 0x00, 0x00);     // Write
to LED Digits 0
                    genie.WriteStr(0,FRASE2);
                    }
                  break;
        case '1': boton1='1';
                  break;
        case '2': boton2='1';
                  break;
        case '3': N_ciclos = N_ciclos + 1;  // Receive the event data from the
Slider0
                genie.WriteObject(GENIE OBJ LED DIGITS, 0x00, N ciclos);     //
escribe el numero de ciclos en LED Digits 0
                  break;
        case '4': inicio=!inicio;
                  break;
        case '5': SetPinFrequencySafe(step_motor, Vlenta);
                  digitalWrite(MS1,HIGH);
                  break;
        case '6': SetPinFrequencySafe(step_motor, Vnormal);
                  digitalWrite(MS1,LOW);
                  break;
        case '7': SetPinFrequencySafe(step_motor, Vrapida);
                  digitalWrite(MS1,LOW);
                  break;
        case '8': cortar_una = '1';
                  break;
        default: form=0;
                 break;
     }
   }
//*************************************************************************
************************
      switch (form) {
        //código para las distintas pantallas de control
        case 0: //MODO ESPERA
                digitalWrite(en_motor, HIGH);
                //DIRestado=LOW;
                N_ciclos = 0; cuenta=0; unos=0; flag_cerrar='0'; cortar_una =
'0';
                //genie.WriteObject(GENIE_OBJ_LED, 0x00, 0x00);  // Write to
LED 0
                //genie.WriteObject(GENIE_OBJ_LED_DIGITS, 0x00, N_ciclos);  //
Write 0 value to LED Digits 0
                detachInterrupt(4);
                detachInterrupt(5);
                break;

        case 1: //MODO CONTINUO
                if(start==1){
                  digitalWrite(DIRPin,DIRestado);
                  pwmWrite(step_motor, ciclo);
                  digitalWrite(en_motor, LOW);
                }else{
                  digitalWrite(DIRPin,DIRestado);
                  digitalWrite(en_motor, HIGH);
                }
                break;

         case 2: //MODO PASO A PASO O GRIP
                attachInterrupt(4, boton_exterior_1, RISING);
                attachInterrupt(5, boton_exterior_2, RISING);
                digitalWrite(en_motor, HIGH);

//*************************************************************************
```

```
                //ESTANDO YA CON EL NUEVO DISPLAY DE LA TOUCHSCREEN SE
MOSTRARAN 2 BOTONES
                //UNO PARA ABRIR Y OTRO PARA CERRAR
                if(boton1=='1'){
                    if(digitalRead(abierto)==1){//Serial.println("MODO GRIP
ABRIENDO...");
                      digitalWrite(en_motor, HIGH);
                      boton1='0';
                    }else{
                      DIRestado=LOW;
                      grip(DIRestado);
                      digitalWrite(en_motor, HIGH);
                      boton1='0';
                    }
                    //Serial.println(en_motor);
                    //Serial.println(incomingByte);
                  }
                if(boton2=='1'){
                    //Serial.println("MODO GRIP CERRANDO...");
                    if(digitalRead(cerrado)==1){
                      digitalWrite(en_motor, HIGH);
                      boton2='0';
                    }else{
                      DIRestado=HIGH;
                      grip(DIRestado);
                      digitalWrite(en_motor, HIGH);
                      boton2='0';
                    }
                    //Serial.println(en_motor);
                     //Serial.println(incomingByte);
                  }
                 break;

          case 3://MODO CICLOS//AQUI SE CAMBIARÍA EL DISPLAY EN LA TOUCHSCREEN
               if((inicio == 1) && (N_ciclos > 0)){
                  if (c_hechos >= N_ciclos){
                    digitalWrite(en_motor, HIGH);
                    //digitalWrite(en_motor, motor_estado);
                    c_hechos=0;
                    N_ciclos=0;
                    inicio=0;
                    genie.WriteObject(GENIE_OBJ_LED_DIGITS, 0x00, N_ciclos);
                  }else{
                    pwmWrite(step_motor, ciclo);
                    digitalWrite(DIRPin,DIRestado);
                    digitalWrite(en_motor, LOW);
                  }
               }else{
                 digitalWrite(DIRPin,DIRestado);
                 digitalWrite(en_motor, HIGH);
               }
                break;

          case 4:  //PANTALLA HERRAMIENTA CORTAUÑAS
               if(cortar_una == '1'){
                 digitalWrite(DIRPin,HIGH);
                 digitalWrite(en_motor, LOW);//CORTA
                 if(flag_cerrar == '1'){
                   digitalWrite(DIRPin,LOW);
                   delay(800);
                   digitalWrite(en_motor, HIGH);
                   cortar_una='0';
                   flag_cerrar='0';
                 }
               }else{
                  digitalWrite(en_motor, HIGH);
               }
               break;

          case 5:  //PANTALLA CONEXIÓN BLUETOOTH

              if(cuenta == 37)
```

```
              {
                if(unos >=1 & unos <= 6){
                  genie.WriteStr(2,FRASE6);//escribir string DISPOSITIVO
CONECTADO
                }else{
                  genie.WriteStr(2,FRASE5); //escribir string ESPERANDO
CONEXION
                }
                cuenta=0;
                unos=0;
              }
              break;

          case 6: //COMPRUEBA SI PINZA ESTA COMPLETAMENTE CERRADA O ABIERTA
              digitalWrite(en_motor, HIGH);
              if(digitalRead(abierto)==1){
                DIRestado=HIGH;
              }
              if(digitalRead(cerrado)==1){
                DIRestado=LOW;
              }
              form = 0;
              break;
          default:
              digitalWrite(en_motor, HIGH);
              form = 0;
              break;
      }//FIN DE SENTENCIA SWITCH-CASE

      Serial.flush();
      Serial3.flush();

//*************************************************************************
           //PARA ENTRAR EN MODO ESCUCHA


//*************************************************************************
*******************
        /*while(parpadeo==1){
        digitalWrite(ledverdePin, HIGH);   // turn the LED on (HIGH is the
voltage level)
        delay(100);               // wait for a second
        digitalWrite(ledverdePin, LOW);    // turn the LED off by making the
voltage LOW
        delay(100);
        }*/
}//FIN BUCLE PRINCIPAL

/////////////////////////////////////////////////////////////////////
//
// This is the user's event handler. It is called by genieDoEvents()
// when the following conditions are true
//
//           The link is in an IDLE state, and
//           There is an event to handle
//
// The event can be either a REPORT_EVENT frame sent asynchrounously
// from the display or a REPORT_OBJ frame sent by the display in
// response to a READ_OBJ request.
//

//LONG HAND VERSION, MAYBE MORE VISIBLE AND MORE LIKE VERSION 1 OF THE LIBRARY
void myGenieEventHandler(void)
{
  genieFrame Event;
  genie.DequeueEvent(&Event);

  //char FRASE[18];
//*************************************************************************
*
//REPORT AL CAMBIAR DE PANTALLA
```

```
//************************************************************************
*
  if(Event.reportObject.cmd == GENIE_REPORT_EVENT)
  {
    if (Event.reportObject.object == GENIE_OBJ_FORM)
    {
      if (Event.reportObject.index == 0)     // If pantalla INICIO
      {
        form=6;
        start=0;
        inicio=0;
      }
      if (Event.reportObject.index == 1)    // If pantalla MODO CONTINUO
      {
        form=1;
        inicio=0;
      }
      if (Event.reportObject.index == 2)    // If pantalla MODO GRIP
      {
        form=2;
        start=0;
        inicio=0;
      }
      if (Event.reportObject.index == 5)    // If pantalla MODO CICLOS
      {
        form=3;
      }
      if (Event.reportObject.index == 4)    // If pantalla CORTAUÑAS
      {
        form=4;
      }
      if (Event.reportObject.index == 3)    // If pantalla BLUETOOTH
      {
        form=5;
      }
    }
  }
//************************************************************************
*
//REPORT AL TOCAR ALGUN ELEMENTO (BOTONES, SLIDERS, ETC)
//************************************************************************
*

  //If the cmd received is from a Reported Event
  if(Event.reportObject.cmd == GENIE_REPORT_EVENT)
  {
    if (Event.reportObject.object == GENIE_OBJ_USERBUTTON)                // If
the Reported Message was from a Slider
    {
      if (Event.reportObject.index == 0x0D)                              // If
Slider0
      {
        activar = !activar;
        if(activar==HIGH){
          digitalWrite(Vblue, HIGH);
          genie.WriteObject(GENIE_OBJ_USER_LED, 0, 0x01);     // Write to LED
Digits 1
          genie.WriteStr(1,FRASE3);
          }
        if(activar==LOW){
          digitalWrite(Vblue, LOW);
          genie.WriteObject(GENIE_OBJ_USER_LED, 0, 0x00);     // Write to LED
Digits 0
          genie.WriteStr(1,FRASE4);
        }
       }
    }
  }

  if(Event.reportObject.cmd == GENIE_REPORT_EVENT)
  {
```

```
    if (Event.reportObject.object == GENIE_OBJ_USERBUTTON)                 // If
the Reported Message was from BOTON
    {
      if (Event.reportObject.index == 0x07)                               // If
boton 0
      {
        led_est = !led_est;
        if(led_est==HIGH){
          start=1;
          genie.WriteStr(0,FRASE1);
          }
        if(led_est==LOW){
          start=0;
          genie.WriteStr(0,FRASE2);
          }
      }
      if (Event.reportObject.index == 0x0B)                               // If
boton 1
      {
         boton1='1';
      }
      if (Event.reportObject.index == 0x0C)                               // If
boton 2
      {
         boton2='1';
      }
      if (Event.reportObject.index == 0x00)                               // If
boton 5
      {  //VELOCIDAD LENTA
         SetPinFrequencySafe(step_motor, Vlenta);
         digitalWrite(MS1,HIGH);
      }
      if (Event.reportObject.index == 0x05)                               // If
boton 6
      {  //VELOCIDAD NORMAL
         SetPinFrequencySafe(step_motor, Vnormal);
         digitalWrite(MS1,LOW);
      }
      if (Event.reportObject.index == 0x06)                               // If
boton 7
      {  //VELOCIDAD RÁPIDA
         SetPinFrequencySafe(step_motor, Vrapida);
         digitalWrite(MS1,LOW);
      }
      if (Event.reportObject.index == 0x0E)                               // If
boton 8
      {  //CORTAR UÑA
         //SetPinFrequencySafe(step_motor, Vrapida);
         //digitalWrite(MS1,LOW);
         cortar_una = '1';
      }
    }
  }

//*******************************************************************************
*
//REPORT If the cmd received is from a Reported Object, which occurs if a Read
Object is requested in the main code, reply processed here.
//*******************************************************************************
*

  //This can be expanded as more objects are added that need to be captured

  //Event.reportObject.cmd is used to determine the command of that event, such
as an reported event
  //Event.reportObject.object is used to determine the object type, such as a
Slider
  //Event.reportObject.index is used to determine the index of the object, such
as Slider0
  //genieGetEventData(&Event) us used to save the data from the Event, into a
variable.
}
```

```
//*****************************************************************************
*
//
//FUNCUIONES DE ACCION PRESSMATIC
void cerrar()
{
  DIRestado=HIGH;
  c_hechos = c_hechos++;
  parpadeo=0;
  flag_cerrar='0';
  //Serial3.println("cerrar");
//}
}//FIN CERRAR

void abrir()
{
  //n_abrir=n_abrir++;
  flag cerrar = '1';
  DIRestado = LOW;
}//FIN ABRIR

void posicion_inicial()
{
  if(digitalRead(cerrado)==1){
    //Serial.println("PINZA CERRADA");
    DIRestado=LOW;
    digitalWrite(Pinled13, LOW);
  }
    else{
        DIRestado=LOW;
        digitalWrite(Pinled13, LOW);
   }
 parpadeo=1;
}//FIN POSICION_INICIAL

void on_off()
{
  encendido= !encendido;
  if(encendido == 1){
     digitalWrite(ledverdePin, HIGH);
     //digitalWrite(22, 1);  // unReset the Display via D4

     //genieWriteContrast(1); // 1 = Display ON, 0 = Display OFF
     }
  if(encendido == 0){
    digitalWrite(ledverdePin, LOW);
    //digitalWrite(22, 0);  // Reset the Display via D4
    //genieWriteContrast(0); // 1 = Display ON, 0 = Display OFF
  }
  /*AQUI HAY QUE COMPLEMENTAR CON FUNCIONES DE AHORRO DE ENERGÍA PARA OPTIMIZAR
  LA BATERÍA*/
}
//ACCIONAMIENTO DE LOS TOPES
void boton_exterior_1()
{
  boton1='1';
}

void boton_exterior_2()
{
  boton2='1';
}
//ACION MODO GRIP
void grip(int DIRestado)
{
 digitalWrite(DIRPin, DIRestado);
 digitalWrite(en_motor, LOW);
 pwmWrite(step_motor, ciclo);
 //analogWrite(step_motor, ciclo);
 delay(150);
}
```

```
ISR(TIMER1_COMPA_vect){//Subrutina de atención a la interrupción del Timer 1 en
modo CTC
  //TIFR1 |= (1 << OCF1A);  //Clear del flag de la interrupción
  //TCCR2A |= (0 << COM2A1);
  //TCCR2A |= (1 << COM2A0);  //Modo TOGGLE
  //DDRB |= (1 << DDB4);
  //estado = !estado;
  cuenta = cuenta + 1;
  if(digitalRead(state_blue)==HIGH){
    unos=unos + 1;
  }
  //Serial3.println(unos);
}
```

Glosario

IDE	Integrated development environment
VR	Voice recognition
PWM	Pulse-width modulation
QVGA	Quarter Video Graphics Array
LCD	Liquid-crystal display
TFT	Thin-film transistor
LED	Light-emitting diode
UART	Universal asynchronous receiver/transmitter
PDA	Personal digital assistant
HPA	High-Performance Addressing
STN	Super-twisted nematic
DSTN	Dual-scan super-twist nematic
CSTN	Color super-twist nematic
AMOLED	Active-matrix organic light-emitting diode
GB	Gigabyte
KB	Kilobyte
SRAM	Static random-access memory
COM	Communication port
K	Kilo
I2C	Inter-Integrated Circuit
VCC	Power-supply pins
TX	Transmit (pin)
RX	Recieve (pin)
GND	Ground (pin)
I/O	Input/Output
SD	Secure Digital
SDHC	Secure Digital High Capacity
SPI	Serial Peripheral Interface
DOS	Disk Operating System
FAT	File Allocation Table
WAV	Waveform Audio File Format
V	Volts
RoHS	Restriction of Hazardous Substances
CE(MARK)	Conformité Européenne
W3C	World Wide Web Consortium
WCAG	Web Content Accessibility Guidelines
ATAG	Authoring Tool Accessibility Guidelines
UAAG	User Agent Accessibility Guidelines
ISO	International Organization for Standardization
ONCE	Organización Nacional de Ciegos Españoles
iOS	iPhone Operating System
EVE	Extensible Virtual Engine
WYSIWYG	What you see is what you get
ASCII	American Standard Code for Information Interchange
CRC	Cyclic redundancy check
RES	Reset
LSB	Least Significant Bit
cmd	Comand

IVA	Impuesto sobre el valor añadido

Bibliografía

[1] G. B. d. María, *DISPOSITIVO AUTOMÁTICO DE APOYO PARA USO DE HERRAMIENTAS,* 2012.

[2] A. V. Gómez, *Programación en Android accesible para el control de PRESSMATIC (Trabajo de Fin de Grado),* Leganés: Universidad Carlos III de Madrid, 2014.

[3] A. R. García, *Protocolo de comunicación Bluetooth y control por voz para PRESSMATIC (Trabajo de Fin de Grado),* Leganés: Universidad Carlos III de Madrid, 2014.

[4] J. E. Arthur, «Touch displays». Estados Unidos Patente US3482241 A, 2 Diciembre 1969.

[5] F. Ebeling, R. Johnson y R. Goldhor, «Infrared light beam x-y position encoder for display devices». Estados Unidos Patente US3775560 A, 27 Noviembre 1973.

[6] W. C. Colwell Jr y G. S. Hurst, «Discriminating contact sensor». Estados Unidos Patente US3911215 A, 7 Octubre 1975.

[7] K. Tihanyl, «Television apparatus». Estados Unidos Patente US2158259 A, 16 Mayo 1939.

[8] D. D. Bushnell, The automation of school information systems, , 1964.

[9] *29th International Science and Engineering Exposition "book of abstracts",* Washington D.C.: Science Service, 1978.

[10] R. Williams, «Electro-optical elements utilizing an organic nematic compound». Estados Unidos Patente US3322485 A, 30 Mayo 1967.

[11] J. W. Steed y J. L. Atwood, Supramolecular Chemistry, John Wiley & Sons, 2009.

[12] Samsung Electronics, «Learn About LCD TV and TFT LCD Displays,» [En línea]. Available: http://www.plasma.com/classroom/what_is_tft_lcd.htm. [Último acceso: Septiembre 2014].

[13] E. Lueder, Liquid Crystal Displays, John Wiley & Sons, 2010.

[14] Kunshan Govisionox Optoelectronics, «GVO AMOLED,» [En línea]. Available: http://www.ksgvo.com/EN/info/9.html. [Último acceso: Septiembre 2014].

[15] P. Namek, «Wikipedia Light-Emiting Diode,» 13 Agosto 2005. [En línea]. Available: http://en.wikipedia.org/wiki/Light-emitting_diode#mediaviewer/File:RBG-LED.jpg. [Último acceso: Septiembre 2014].

[16] 4D Systems, «4D Systems,» [En línea]. Available: http://www.4dsystems.com.au/. [Último acceso: Septiembre 2014].

[17] 4D Systems, «Arduino Display Module Pack,» 24 Enero 2014. [En línea]. Available: http://www.4dsystems.com.au/productpages/uLCD-28PTU-AR/downloads/uLCD-28PTU-AR_datasheet_R_1_4.pdf. [Último acceso: septiembre 2014].

[18] 4D Systems, «2.8" microLCD PICASO Display,» 15 Enero 2014. [En línea]. Available: http://www.4dsystems.com.au/productpages/uLCD-28PTU/downloads/uLCD-28PTU_datasheet_R_1_7.pdf. [Último acceso: septiembre 2014].

[19] W3C, «Web Content Accessibility Guidelines 1.0,» 5 mayo 1999. [En línea]. Available: http://www.w3.org/TR/WAI-WEBCONTENT/. [Último acceso: Septiembre 2014].

[20] AENOR, *EN ISO 9241-171:2008 de Requisitos de accesibilidad del software,* Madrid: AENOR, 2009.

[21] J. Burgos, «¿Por qué las aplicaciones móviles deben ser accesibles?,» 27 Mayo 2013. [En línea]. Available: http://www.grupofundosa.es/es/%C2%BFpor-qu%C3%A9-las-aplicaciones-m%C3%B3viles-deben-ser-accesibles. [Último acceso: Septiembre 2014].

[22] E. Archanco, «Las 5 apps accesibles e imprescindibles de iOs para utilizar día a día,» 20 Mayo 2014. [En línea]. Available: http://www.applesfera.com/aplicaciones-ios-1/las-5-apps-accesibles-e-imprescindibles-de-ios-para-utilizar-dia-a-dia. [Último acceso: Septiembre 2014].

[23] Google, «Tienda web oficial de Google Play,» [En línea]. Available: https://play.google.com/store. [Último acceso: Septiembre 2014].

[24] R. R. Zúnica, «Cómo elegir el tipo de letra para la página web,» Unidad de Investigación ACCESO, 23 Marzo 2001. [En línea]. Available: http://acceso.uv.es/accesibilidad/artics/01-tipo-letra.htm. [Último acceso: Septiembre 2014].

[25] The OneVoice for Accessible ICT Coalition, «Create a user interface that is easy to understand and operate,» [En línea]. Available: http://www.onevoiceict.org/first-seven-steps-accessible-mobile-apps/create-user-interface-easy-understand-and-operate. [Último acceso: Septiembre 2014].

[26] 4D Systems, «Workshop 4 User Guide,» 11 Marzo 2014. [En línea]. Available: http://www.4dsystems.com.au/productpages/4D-Workshop-4-IDE/downloads/Workshop-4_userguide_R_1_4.pdf. [Último acceso: Septiembre 2014].

[27] 4D Systems, «Workshop 4 - ViSi-Genie User Guide,» 23 Septiembre 2013. [En línea]. Available: http://www.4dsystems.com.au/downloads/Software/4D-Workshop4-

IDE/Docs/ViSi-Genie/Visi-Genie-User-Guide_R_1_02.pdf. [Último acceso: Septiembre 2014].

[28] 4D Systems, «Application Note: 4D-AN-D4006, ViSi Genie – User Button,» 1 Octubre 2013. [En línea]. Available: http://www.4dsystems.com.au/downloads/Application-Notes/4D-AN-D4006_R_1_0.pdf. [Último acceso: Septiembre 2014].

[29] Arduino, «Arduino Build Process,» [En línea]. Available: http://arduino.cc/en/Hacking/BuildProcess. [Último acceso: Septiembre 2014].

[30] E. Toboso, «La comuniciación serie,» 4 Enero 2013. [En línea]. Available: http://perso.wanadoo.es/pictob/comserie.htm#la_comunicacion_serie. [Último acceso: Septiembre 2014].